Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors for IoT Applications

Datasheet, Volume 1

May 2024

Revision 1.9

Document Number: 636112-1.9

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at Intel.com, or from the OEM or retailer.

No computer system can be absolutely secure. Intel does not assume any liability for lost or stolen data or systems or any damages resulting from such losses.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at intel.com, or from the OEM or retailer.

All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or visit www.intel.com/design/literature.htm.

No computer system can provide absolute security under all conditions. Intel® Trusted Execution Technology (Intel® TXT) requires a computer system with Intel® Virtualization Technology, an Intel TXT-enabled processor, chipset, BIOS, Authenticated Code Modules and an Intel TXT-compatible measured launched environment (MLE). The MLE could consist of a virtual machine monitor, an OS or an application. In addition, Intel TXT requires the system to contain a TPM v1.2, as defined by the Trusted Computing Group and specific software for some uses. For more information, see http://www.intel.com/technology/security/

Intel® Virtualization Technology requires a computer system with an enabled Intel® processor, BIOS, virtual machine monitor (VMM) and, for some uses, certain computer system software enabled for it. Functionality, performance or other benefits will vary depending on hardware and software configurations and may require a BIOS update. Software applications may not be compatible with all operating systems. Please check with your application vendor.

Intel® High Definition Audio (Intel® HD Audio): Requires an Intel® HD Audio enabled system. Consult your PC manufacturer for more information. Sound quality will depend on equipment and actual implementation. For more information about Intel® HD Audio, refer to http://www.intel.com/design/chipsets/hdaudio.htm

Hyper-Threading Technology requires a computer system with a processor supporting HT Technology and an HT Technology-enabled chipset, BIOS and operating system. Performance will vary depending on the specific hardware and software you use. For more information including details on which processors support HT Technology, see http://www.intel.com/info/hyperthreading.

Enhanced Intel SpeedStep® Technology See the Processor Spec Finder or contact your Intel representative for more information.

64-bit computing on Intel architecture requires a computer system with a processor, chipset, BIOS, operating system, device drivers and applications enabled for Intel® 64 architecture. Performance will vary depending on your hardware and software configurations. Consult with your system vendor for more information.

Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families. See www.intel.com/products/processor_number for details.

The Bluetooth® word mark and logos are registered trademarks owned by Bluetooth SIG, Inc. and any use of such marks by Intel is under license.

Contents

1 Introduction 17
1.1 About this Manual 17
1.2 References 18
1.3 Processor Overview 18
1.4 Overview 18
1.5 Intel Atom® x6000E Series processors, Intel® Pentium® and Celeron® N and J Series processors Block Diagram 23
1.6 Processor SKUs 26
1.7 Processor Volatility Statement 28
2 Technologies 29
2.1 Tremont ISA Extensions 30
2.2 Security Technologies 31
2.3 Power and Performance Technologies 36
3 Power Management 40
3.1 Power Management States Supported 40
3.2 Processor IA Core Power Management 43
3.3 PM Interface Signals 49
3.4 Processor Voltage Rails 51
3.5 Voltage Rail Electrical Specifications 54
3.6 Intel® Programmable Services Engine (Intel® PSE) Power Management 60
3.7 SMI#/SCI Generation 64
3.8 Sleep States 66
3.9 Event Input Signals and Their Usage 69
3.10 Reset Behavior 73
4 Thermal Management 76
4.1 Thermal and Power Specifications 76
4.2 Processor Thermal Management 77
4.3 PCH Thermal Management 86
5 Memory 89
5.1 System Memory Interface 89
5.2 Power Management 99
5.3 IBECC 102
6 Mapping Address Spaces 107
6.1 System Address Mapping 107
6.2 DOS Legacy Address Range 109
6.3 Lower Main Memory Address Range (1 MB – TOLUD) 112
6.4 PCI Memory Address Range (TOLUD – 4 GB) 114
6.5 Upper Main Memory Address Space (4 GB to TOUUD) 116
6.6 Graphics Memory Address Ranges 118
6.7 System Management Mode (SMM) 119
6.8 SMM and VGA Access Through GTT TLB 119
6.9 Legacy VGA and I/O Range Decode Rules 120
6.10 I/O Mapped Registers 122
6.11 PCH Address Mapping 123
6.12 Variable I/O Decode Ranges 126
6.13 Memory Map 128
7 Graphics 132
7.1 Processor Graphics 132

7.2 Registers 135
8 Display 136
8.1 Display Technologies 136
8.2 General Capabilities 136
8.3 Display Features 137
8.4 Port Configuration 137
8.5 Display Interfaces 139
8.6 Multi-Stream Transport (MST) Configuration 141
8.7 Multiple Display Configurations 141
8.8 High-bandwidth Digital Content Protection (HDCP) 141
8.9 Display Technologies 142
8.10 PCH Display 146
8.11 Panel Control Signals 147
8.12 Embedded DisplayPort (eDP) Signals 147
8.13 MIPI DSI Signals 148
8.14 Digital Display Interface (DDI) Signals 149
9 Flexible I/O 150
9.1 Acronyms 150
9.2 HSIO Controller (PCH) 150
9.3 Overview/Functional Description 152
9.4 Registers 153
10 Audio, Voice, and Speech 154
10.1 Feature Overview 154
10.2 Legacy Audio Interface - Signal Description 155
10.3 Intel® High Definition Audio (Intel® HD Audio) Controller Capabilities 156
10.4 Direct Attached Digital Microphone (PDM) Interface 158
10.5 I2S/PCM Interface 158
10.6 References 159
11 Universal Serial Bus (USB) 160
11.1 Overview 160
11.2 Integrated Pull-Ups and Pull-Down11.3 Registers 165
11.3 Registers 165
12 PCI Express 166
12.1 Acronyms 166
12.2 Signal Description 166
12.3 I/O Signal Planes and States 166
12.4 PCI Express* Port Support Feature Details 167
12.5 Overview/Functional Description 167
12.6 Registers 174
13 Serial ATA (SATA) 175
13.1 Acronyms 175
13.2 References 175
13.3 Overview 175
13.4 I/O Signal Planes and States 175
13.5 Functional Description 176
13.6 Registers 180
14 Host System Management Bus (SMBus) Controller 181
14.1 Functional Description 181
14.2 Signal Description 181
14.3 Host Controller 181
14.4 Registers 192

15 Gigabit Ethernet Controller and Time-Sensitive Networking 193
15.1 Overview 193
15.2 Features Description 194
15.3 GbE Time-Stamping Logic 198
15.4 GbE Cross-Timestamp Logic 198
15.5 External Interfaces 199
15.6 Signal Description 200
15.7 GbE-TSN Interrupts and Message Signaled Interrupt 202
15.8 GbE TSN Register/Programming Differences Between GbE PSE MAC and GbE HOST MAC 203
15.9 Supported System Configurations 205
15.10 Registers 206
15.11 References 206
16 Enhanced Serial Peripheral Interface (eSPI) 207
16.1 Functional Overview 207
16.2 Registers 213
17 Serial Peripheral Interface (SPI) Flash Memory and TPM Only 214
17.1 Acronyms 214
17.2 Feature Overview 214
17.3 Signal Description 215
17.4 Integrated Pull-Ups and Pull-Downs 215
17.5 I/O Signal Planes and States 216
17.6 Functional Description 216
17.7 VCCSPI Voltage (3.3V or 1.8V) Selection 223
17.8 Registers 224
18 SIO (LPSS) 225
18.1 Intel® Serial I/O Universal Asynchronous Receiver/Transmitter (UART) Controllers 225
18.2 Intel® Serial I/O Inter-Integrated Circuit (I2C) Controllers 233
18.3 Serial Peripheral Interface (SIO SPI) 238
19 Storage 244
19.1 embedded Multi Media Card (eMMC*) 244
19.2 Secure Digital eXtended Capacity (SDXC) 245
20 Clocking 247
20.1 Integrated Clock Controller (ICC) 247
20.2 PCH ICC Clocking 247
20.3 PCH ICC XTAL Input Configuration 252
20.4 Summary of Clock Signal 252
20.5 Registers 253
21 General Purpose Input and Output (GPIO) 254
21.1 Overview 254
21.2 Pad Grouping, Muxing, and Capabilities 254
21.3 Functional Description 254
21.4 GPIO Multiplexing Table 263
21.5 Registers 305
22 Intel® Programmable Services Engine (Intel® PSE) 306
22.1 Overview 306
22.2 Functional Description 306
22.3 Block Diagram 307
22.4 Intel® PSE Resources Required 308
22.5 Arm* Cortex*-M7 Subsystem 309
22.6 L2 SRAM 310

22.7 Clock Control Unit (CCU) and PLL 310
22.8 Power Management Unit (PMU) 311
22.9 Address Translation Table (ATT) 311
22.10 AON Controller 311
22.11 Timer 312
22.12 I/O Ownership and Interrupts 312
22.13 Controller Area Network (CAN) Bus Controller 317
22.14 I2C Controller 334
22.15 UART Controller 352
22.16 SPI Controller 378
22.17 GPIO Controller 391
22.18 Time-Aware GPIO 393
22.19 I2S Controller 400
22.20 Pulse Width Modulation (PWM) 407
22.21 Quadrature Encoder Peripheral (QEP) 410
22.22 Time Synchronous Support 424
22.23 DMA 424

23 Intel® Safety Island (Intel® SI) 437
23.1 Feature Overview 437
23.2 Error Reporting 440
23.3 Integrated Pull-Ups and Pull - Downs 442
23.4 I/O Signal Planes and States 442
23.5 Registers 443

24 Functional Safety (FuSa) 444
24.1 Overview 444
24.2 Processor and FuSa Safety Package 444

25 Primary to Sideband Bridge (P2SB) 445
25.1 Overview 445
25.2 Integrated Error Handler 446
25.3 Registers 447

26 Legacy Interfaces 448
26.1 8254 Timers 448
26.2 I/O APIC 451
26.3 8259 Programmable Interrupt Controller (PIC) 453
26.4 Real-Time Real Time Clock (RTC) 459
26.5 System Management 464
26.6 High Precision Event Timer (HPET) 468
26.7 Processor Interface 473

27 Pin Strap 475

28 Test and Debug 479
28.1 Debug Capability and Technologies 479
28.2 Signal Description 484
28.3 Intel® Atom Debug and Tool 485
28.4 Arm* Debug and Tool 487
28.5 Debug Interface Availability 490
28.6 References 490

29 Intel® Time Coordinated Computing 491
29.1 Intel® Time Coordinated Computing Overview 491
29.2 Intel® Time Coordinated Computing Features 492
29.3 Intel® TCC Tools 499

30 Global Device IDs 500
30.1 Overview 500
30.2 PCH Global Device IDs 500
30.3 PCH ACPI IDs 504
30.4 Compute Die Global Device ID 504
31 Processor Ball Map and Pin Location 507
32 Package Information 546
32.1 Package Mechanical Drawing - Non IHS 546
32.2 Package Mechanical Drawing - IHS 548
33 Processor Transaction Router (PTR) 549
33.1 Overview 549
33.2 I/O Port (IOP) 549
34 Machine Check Architecture (MCA) 551
34.1 Overview 551
34.2 Machine Check Architecture (MCA) MSR Addresses 551
34.3 Registers 553
35 Intel® Converged Security Engine (Intel® CSE) 554
35.1 Overview 554
36 Electrical Specifications 557
36.1 Crystal Specifications 557
36.2 Storage Conditions 558
36.3 DC Specifications 559
37 Terminology 583

List of Figures

Figure 1-1Compute Die Block Diagram 23
Figure 1-2PCH Block Diagram 24
Figure 1-3PSE Disabled - PCH Block Diagram 25
Figure 3-1System Power States 40
Figure 3-2Idle Power Management Breakdown of the Processor IA Cores 44
Figure 3-3Package C-State Entry and Exit 47
Figure 3-4System Power States 61
Figure 3-5Device States Latency 63
Figure 6-1System Address Range Example 109
Figure 6-2DOS Legacy Address Range 110
Figure 6-3PAM Region Space 111
Figure 6-4Main Memory Address Range 112
Figure 6-5PCI Memory Address Range 115
Figure 7-1Block Diagram 133
Figure 8-1Display Subsystem Block Diagram 138
Figure 8-2DisplayPort Overview 142
Figure 8-3HDMI Overview 144
Figure 8-4MIPI DSI Overview 145
Figure 8-5Panel Self Refresh Diagram 146
Figure 9-1HSIO Controller Port Configuration 150
Figure 9-2HSIO Controller Lanes to x12 ModPHY Lane Multiplexing 151
Figure 9-3Configuration SATA + SGMII GbE 152

Figure 11-1USB 3.1/PCIe*/SATA Port Mapping 161
Figure 12-1PCIE Controller Port Configuration 166
Figure 12-2Single Virtual Channel PCIe* Controller 168
Figure 12-1Generation of SERR# to Platform 172
Figure 12-3PCI Express* Controller Lane Reversal 174
Figure 13-1Flow for Port Enable/Device Present Bits 179
Figure 15-1GbE-TSN MAC Placement 193
Figure 16-1Basic eSPI Protocol 208
Figure 17-1Flash Descriptor Regions 219
Figure 18-1UART Serial Protocol 227
Figure 18-2UART Receiver Serial Data Sample Points 228
Figure 18-3Data Transfer on the I2C Bus 235
Figure 20-1Internal Clock Diagram - “iSCLK” 248
Figure 20-2Internal Clock Diagram - “modPHY” 248
Figure 20-3PSE_Clocking 249
Figure 20-4PSE_GBe Clocking 250
Figure 20-5PCH ICC XTAL Input Configuration 252
Figure 21-1Input Capture Rising Edge 256
Figure 21-2Input Capture Falling Edge 256
Figure 21-3Input Capture Both (Toggle) Edge(s) 257
Figure 21-4Output Generation Rising Pulse 258
Figure 21-5Output Generation Falling Pulse 258
Figure 21-6Output Generation Toggle Edge 259
Figure 22-1Intel® PSE Block Diagram 308
Figure 22-2Bus monitoring mode 322
Figure 22-3Internal Loop Back Mode 324
Figure 22-4Standard Message ID Filter Path 326
Figure 22-5Extended Message ID Filtering 327
Figure 22-6Rx FIFO Status 328
Figure 22-7Rx FIFO Overflow Handling 329
Figure 22-87-bit Address Format 338
Figure 22-910-bit address format 338
Figure 22-10 Initiator-Transmitter Protocol 340
Figure 22-11Initiator-Receiver Protocol 341
Figure 22-12IC_DATA_CMD register if IC_EMPTYFIFO_HOLD_MASTER_EN= 1 342
Figure 22-13Breakdown of DMA Transfer into Burst Transactions 347
Figure 22-14Breakdown of DMA Transfer into Single and Burst Transactions 348
Figure 22-15Case 1 Watermark Levels 349
Figure 22-16Case 2 Watermark Levels 349
Figure 22-17I2C Receive FIFO 351
Figure 22-18Serial Data Format 354
Figure 22-19Auto Address Transmit Flow Chart 356
Figure 22-20Hardware Address Match Receive Mode 357
Figure 22-21Flowchart of Interrupt Generation for Programmable THRE Interrupt Mode 368
Figure 22-22Flowchart of Interrupt generation when not in Programmable THRE Interrupt Mode ... 369
Figure 22-23Breakdown of DMA Transfer into Burst Transaction 371
Figure 22-24Breakdown of DMA Transfer into Single and Burst Transactions 372
Figure 22-25Case 1 Watermark Levels 373
Figure 22-26Case 2 Watermark Levels 374
Figure 22-27UART Receive FIFO 376
Figure 22-28Serial Format Continuous Transfers (SCPH = 0) when SSI_SCPH0_SSTOGGLE = 1 379
Figure 22-29SPI Serial Format (SCPH=1) 380
Figure 22-30SPI controller Configured as Initiator Device 382

Figure 22-31Breakdown of DMA Transfer into Burst Transactions 385
Figure 22-32Breakdown of DMA Transfer into Single and Burst Transactions 386
Figure 22-33Case 1 Watermark Levels 387
Figure 22-34Case 2 Watermark Levels 387
Figure 22-35SPI Controller Receive FIFO 389
Figure 22-36Sync out configuration 397
Figure 22-37Sync In Configuration 398
Figure 22-38Time Slice Generator controller 399
Figure 22-39Example of basic transmission for I2S bus 401
Figure 22-40Timers Usage Flow Diagram 408
Figure 22-41Block Diagram of QEP 411
Figure 22-42Controller Block Diagram 413
Figure 22-43Edge Selection and Phase Swapping Block Diagram 414
Figure 22-44Quadrature Decoder Block Diagram 415
Figure 22-45Phase relationship example between PhA and PhB signals 416
Figure 22-46State diagram for direction decoding 416
Figure 22-47Software Flow Diagram for QEP Functionality 421
Figure 22-48Software flow diagram for Capture Compare Functionality 423
Figure 22-49DMA Multi-Block and Update Flowchart 433
Figure 22-50Multi-Block Transfer Setup using Linked Lists 434
Figure 23-1Intel® SI Block Diagram 438
Figure 26-1TCO Compatible Mode SMBus Configuration 465
Figure 28-1Overall Debug Capability 480
Figure 28-2Switching Flows Between JTAG and SWD 488
Figure 29-1Intel® TCC Features within System and TSN between Systems 491
Figure 29-2Platform Time Synchronization 493
Figure 29-3IOTLB Usage 496
Figure 29-4LLC without Cache QoS 498
Figure 29-5LLC with Cache QoS 498
Figure 32-1Package Mechanical drawing - Part 1 of 2 546
Figure 32-2Package Mechanical drawing - Part 2 of 2 547
Figure 32-3Package Mechanical Drawing 548
Figure 34-1Processor Core, Module, and Compute Die Machine Check Registers 553

List of Tables

Table 1-1Processor Features 19
Table 1-2Processor SKU MAP (High Level) 26
Table 3-1System States 41
Table 3-2Integrated Memory Controller (IMC) States 41
Table 3-3G, S, and C Interface State Combinations 41
Table 3-4State Transition Rules for the PCH 42
Table 3-5System Power Plane 42
Table 3-6Core C-States 45
Table 3-7Module C-States 46
Table 3-8Package C-States 47
Table 3-9Intel Atom® x6000E Series processors, Intel® Pentium® and Celeron® N and J Series processors S0ix Power Sub-States 49
Table 3-10Signal Descriptions 49
Table 3-11Platform Voltage Rails 52
Table 3-12Additional Voltage Rail Signals 53
Table 3-13Processor VccIN Active and Idle Mode DC Voltage and Current Specifications 55
Table 3-14Processor VCC_1P8A Supply DC Voltage and Current Specifications 57

Table 3-15VccIN_AUX Supply DC Voltage and Current Specifications 57
Table 3-16Memory Controller (VDDQ) Supply DC Voltage and Current Specifications..... 59
Table 3-17VCCIO Supply DC Voltage and Current Specifications 59
Table 3-18Additional Rails Estimated Icc3 59
Table 3-19PSE D0ix states 62
Table 3-20Causes of SMI and SCI 64
Table 3-21Sleep Types 67
Table 3-22Causes of Wake Events..... 67
Table 3-23 Transitions Due to Power Failure..... 69
Table 3-24Transitions Due to Power Button 70
Table 3-25Causes of Host and Global Resets..... 73
Table 4-1Processor Specifications..... 76
Table 4-2Thermal Trip Points and Response (Typical) 88
Table 5-1DDR Support Matrix Table 89
Table 5-2LPDDR4/4x Sub-Channels Population Rules 91
Table 5-3DDR4 Channel Population Rules 91
Table 5-4Supported SA Speed Enhanced Speed steps (SA-GV) and Gear Mode Frequencies 91
Table 5-5Supported DDR4 SODIMM Module Configurations 92
Table 5-6Supported DDR4 DRAMs (Memory Down) Configurations..... 93
Table 5-7Supported LPDDR4/4x x32 DRAMs Configurations 93
Table 5-8System Memory Interface Signals Terminology 93
Table 5-9Firmware/Software Initiated Memory Access 104
Table 6-1SMM Regions 119
Table 6-2Processor Graphics Frame Buffer Accesses 120
Table 6-3Processor Graphics VGA I/O Mapping 121
Table 6-4MDA IO Transaction Mapping..... 121
Table 6-5MDA Resources 122
Table 6-6Fixed I/O Ranges Decoded by PCH 124
Table 6-7Variable I/O Decode Ranges..... 126
Table 6-8PCH Memory Decode Ranges (Compute Die Perspective)..... 129
Table 6-9Addressing Swapping 131
Table 7-1Hardware Accelerated Video Decode and Encode..... 134
Table 8-1Display Features..... 137
Table 8-2Ports Availability..... 137
Table 8-3Digital Display Signals 139
Table 8-4Pin Mapping for PCH Die 140
Table 8-5Panel Control Signals 147
Table 8-6Embedded DisplayPort Signals..... 147
Table 8-7MIPI DSI Signals 148
Table 8-8Display Interface Signals 149
Table 10-1Legacy Audio Signals..... 155
Table 11-1USB Bandwidth Information 160
Table 11-2Processor USB Specification 161
Table 11-3Signal Description 162
Table 14-1I2C* Multi-Byte Read 184
Table 14-2Enable for SMB_ALERT_N..... 186
Table 14-3Enables for SMBus Target Write and SMBus Host Events..... 186
Table 14-4Enables for the Host Notify Command..... 187
Table 14-5Target Write Registers 188
Table 14-6Command Types 188
Table 14-7Target Read Cycle Format 189
Table 14-8Data Values for Target Read Registers..... 190
Table 14-9Host Notify Format..... 192
Table 15-1TSN IEEE Standards 197
Table 15-2SGMII GbE LAN Signals 200

Table 15-3RGMII Signals 200
Table 15-4MDIO Signals 201
Table 15-5Miscellaneous Signals 201
Table 15-6GbE-TSN interrupts and Message Signaled Interrupt (MSI) Vector Number 202
Table 15-7GbE TSN Register List Differences Between GbE PSE MAC and GbE HOST MAC 203
Table 15-8Supported System Configurations 205
Table 16-1eSPI Signals 207
Table 16-2eSPI Channels and Supported Transactions 209
Table 16-3eSPI Virtual Wires (VW) 210
Table 16-4eSPI Target Request to PCH for PCH Temperature 211
Table 16-5PCH Response to eSPI Target with PCH Temperature 211
Table 16-6eSPI Target Request to PCH for PCH RTC Time 212
Table 16-7PCH Response to eSPI Target with RTC Time 212
Table 17-1SPI Flash Regions 217
Table 17-2Region Size Versus Erase Granularity of Flash Components 218
Table 17-3Region Access Control Table 220
Table 18-1UART Signals 225
Table 18-3Signal Description 234
Table 19-1eMMC Signal Descriptions 244
Table 19-2eMMC* Working Modes 245
Table 19-3SDXC Signals 246
Table 19-4SD Working Modes 246
Table 20-1Intel ® PSE Clock Distribution 250
Table 21-1GPIO Multiplexing Table 264
Table 22-1List of Arm* Cortex*-M7 resources required 309
Table 22-2Timers 312
Table 22-3Intel® PSE Interrupt Routing 314
Table 22-4Intel® PSE ARM Interrupt And MSI Vector Mapping 314
Table 22-5Size and breakdown of the MSG_RAM allocation for CAN[0/1] message RAM 318
Table 22-6Coding of DLS in CAN FD 320
Table 22-7Rx buffer/FIFO Element Size 328
Table 22-8Example Filter Configuration for Rx Buffers 330
Table 22-9Tx Buffer/FIFO Element Size 331
Table 22-10CANBUS Signal 334
Table 22-11I2C Definition of Bits in First Byte 338
Table 22-12I2C Signal Description 352
Table 22-13Divisor Latch Fractional Values 363
Table 22-14UART Signal Description 376
Table 22-15Transmit FIFO Threshold (TFT) Decode Values 383
Table 22-16Receive FIFO Threshold (TFT) Decode Values 383
Table 22-17SPI Signal Description 391
Table 22-18GPIO an TGPIO Muxed 400
Table 22-19TGPIO/GPIO Signal Description 400
Table 22-20Audio interface models 402
Table 22-21Example of audio settings and sample rate 404
Table 22-22I2S Status Register 405
Table 22-23I2S Signal Description 407
Table 22-24PWM Signal Description 410
Table 22-25Capture compare options 418
Table 22-26FIFO_THRE Decode 419
Table 22-27QEP Signal Description 424
Table 22-28DMA Capabilities/Restrictions 425
Table 22-29DMA Hardware Handshake Peripheral Assignments 427
Table 22-30Parameters Used for DMA Setup 429
Table 22-31Basic Block Transfer Example Settings 430

Table 22-32Programming of Transfer Types and Channel Register Update Method 432
Table 23-1Signal Description 439
Table 23-2Legacy Error Reporting Logic 442
Table 23-3Integrated Pull-Ups and Pull-Downs 442
Table 23-4I/O Signal Planes and States 443
Table 25-1Private Configuration Space Register Target Port IDs 445
Table 25-2Error Sources 446
Table 26-1Counter Operating Modes 449
Table 26-2Interrupt Status Registers 454
Table 26-3RTC Crystal Requirements 463
Table 26-4External Crystal Oscillator Requirement 463
Table 26-5Event Transitions that Cause Messages 465
Table 26-6Legacy Replacement Routing 470
Table 26-7Cause of INIT# 473
Table 26-8Cause of NMI 474
Table 27-1Pin Straps 475
Table 28-1JTAG, DBG_PMODE, CFG and BPM_N Testability Signal 484
Table 28-2SWD & ETM Signal Description 484
Table 28-3Debug Interface Availability 490
Table 30-1PCH Global Device IDs 500
Table 30-2ACPI IDs 504
Table 30-3Compute Die Global Device ID 504
Table 31-1Processor Ball Names 507
Table 34-1Processor Machine Check MSR Address 551
Table 36-1Integrated Clock Crystal Specification 557
Table 36-2RTC Crystal Specification 557
Table 36-3Storage Conditions (PC Client Only) 558
Table 36-4Storage Conditions (Embedded and Indu Only) 558
Table 36-5Single-Ended Signal DC Characteristics as Inputs or Outputs 560
Table 36-6CMOS Signal Group DC Specifications 571
Table 36-7GTL Signal Group and Open Drain (OD) Signal Group DC Specifications 572
Table 36-8Display Port* Transmitter DC Specification 572
Table 36-9HDMI* DC Specification 573
Table 36-10Embedded Display Port* DC Specification 573
Table 36-11MIPI*-DSI DC Specification 573
Table 36-12DDR4 Signal Group DC Specifications (Sheet 1 of 2) 575
Table 36-13LPDDR4/x DC Specifications 577
Table 36-14USB 2.0 Host DC Specification 579
Table 36-15USB 3.1 Interface DC Specification 580
Table 36-16SATA DC Specification 582
Table 37-1Terminology 583

Revision History

Revision Date	Revision Number	Description
May 2024	1.9	Chapter 4, "Thermal Management," Updated Section 4.1 Updated Section 4.2.1
April 2024	1.8	Chapter 22, "Intel® Programmable Services Engine (Intel® PSE)," Updated Section 17.2 Updated Section 22.17.1 Updated Table 25-1 Updated Section 35.1.1
March 2023	1.7	Chapter 3, "Power Management," Updated the following for THRMTRIP_N Signal to 1 ms. PMC_SLP_S4_N, and PMC_SLP_S5_N low within 1 ms after sampling THRMTRIP_N active. Chapter 4, "Thermal Management," Added Section 4.2.2.9 Chapter 5, "Memory," Added instances of SDP - 2, SDP - 8, and SDP - 16 in Table 5-1. Added two rows for PKG Type (Die bits per Ch x PKG bits) "SDP 16x32" in Table 5-7. Updated header cell "Die Density" in Table 5-7. Chapter 15, "Gigabit Ethernet Controller and Time-Sensitive Networking," Added Note on jumbo packet support in Section 15.2.1. Added Note under Table 15-1.

Revision Date	Revision Number	Description
November 2022	1.6	Chapter 1, "Introduction" Added SKU 8PU Intel Atom® x6214RE Processor column with details to Table 1-2 Added SKU 9PU Intel Atom® x6416RE Processor column with details to Table 1-2 Removed MIPI-DSI from Figure 1-1 Chapter 8, "Display" Updated PNL x2 to PNL in Figure 8-1 Updated PNL[1:0]_xxx to PNL[0]_xxx in Figure 8-1 Updated DDI1 (Port B) Internal Port in Table 8-2 Updated MIPIB Data 0, MIPIB Data 1, MIPIB Data 2, MIPIB Clock and MIPIB Data 3 to NC IN Table 8-3 Updated PNL[1:0]_VDDEN to PNL0_VDDEN in Table 8-5 Updated PNL[1:0]_BKLTEN to PNL0_BKLTEN in Table 8-5 Updated PNL[1:0]_ BKLTCTL to PNL0_BKLTCTL in Table 8-5 Updated DDI0/1_TXP[3:1] to DDI0_TXP[3:1] in Table 8-7 Updated DDI0/1_TXN[3:1] to DDI0_TXN[3:1] in Table 8-7 Updated DDI0/1_AUXP to DDI0_AUXP in Table 8-7 Updated DDI0/1_AUXN to DDI0_AUXN in Table 8-7 Updated MDSI_DE_TE_[2:1] to MDSI_DE_TE_1 in Table 8-7 Removed MIPIB with DSI1 from Table 8-2 Ports Availability Removed MDSI_DE_TE_2 and DSI1 PP1 from Figure 8-1 Removed Usage Model DDI1 MIPIB in Table 8-4 Removed MDSI_DE_TE_2 from Table 8-4 Removed Display Signal PNL1_VDDEN and the details from Table 8-4 Removed Display Signals PNL1_BKLTEN and the details from Table 8-4 Removed Display Signal PNL1_BKLTCTL and the details from Table 8-4 Removed DDI1_TXP0 and DDI1_TXN0 from Table 8-7 Chapter 20, "Clocking" Updated DDI[1:0]_TXN2 to DDI0_TXN2 in Section 20.4 Updated DDI[1:0]_TXP2 to DDI0_TXP2 in Section 20.4 Updated (MIP[IA/B] Clock) to MIPIA Clock in Section 20.4 Chapter 22, "Intel® Programmable Services Engine (Intel® PSE)" Updated Section 22.17.1 Updated Section 22.23.2.2 Updated Section 22.23.2.4 Removed sub-chapter "Memory to Peripheral Transfers - Multi Block Transfers" in Chapter 22 Removed sub-chapter "Peripheral to Peripheral Transfers - Direct Programming" in Chapter 22 Chapter 30, "Global Device IDs" Updated Processor Transaction Router (Processor SKU 8) to Processor Transaction Router (Processor SKU 8/8PU) in Table 30-3 Updated Processor Transaction Router (Processor SKU 9) to Processor Transaction Router (Processor SKU 9/9PU) in Table 30-3 Chapter 31, "Processor Ball Map and Pin Location" Added superscript note 1 to GP_E23/PNL1_BKLTEN in Table 31-1 Added superscript note 1 to GP_E17/PNL1_VDDEN in Table 31-1 Added Note 1 to Table 31-1 Chapter 36, "Electrical Specifications" Updated Associated Signals in Table 36-6 Removed MDSI_DE_TE_2 from Table 36-6

Revision Date	Revision Number	Description
April 2022	1.5	Chapter 2, "Technologies" Updated Section 2.1 Tremont ISA Extensions Chapter 3, "Power Management" Added new note for PMC_PWRBTN_N in Table 3-10 Added new note to Section 3.9 Event Input Signals and Their Usage Note in Table 3-18 ICCMAX specification for the VCC_RTC_3P3 rail when the processor is in a G3 state has been updated. Added VCC_OUT_1P24A to Table 3-18 Chapter 15, "Gigabit Ethernet Controller and Time-Sensitive Networking" Added new note to Section 15.2.1 Ethernet Features Description Chapter 20, "Clocking" Updated Figure 20-4 Summary of Clock Signal Figure 20-4 PSE_GBE Clocking updated to remove (gbe_ptp_clock(256M/100M) and pse_hh_xtal_clk (19.2M) Chapter 21, "General Purpose Input and Output (GPIO)" Removed "BOOT_PWR_EN" from Table 21-1 GPIO Multiplexing Table Added new Section 21.3.9 I/O Standby Added new Section 21.3.9.1 I/O Standby State (IOSSTATE) Added new Section 21.3.9.2 I/O Standby Termination (IOSTERM) Chapter 22, "Intel® Programmable Services Engine (Intel® PSE)" Updated Section 22.13.3.1 Bit Rate Configuration on the supported Max bit rate Updated Section 22.15.2 Features on the UART capabilities Removed Section 22.13.4.7 Restricted Operation Mode Section 22.13.5.4 Debug on CAN Support Section 22.13.4.10 Power Down (Sleep Mode) Section 22.13.6.4 Tx Queue Section 22.13.6.5 Mixed Dedicated Tx Buffers / Tx FIFO Section 22.13.6.6 Mixed Dedicated Tx Buffers / Tx Queue Table 22-20 Audio Interface models, I2S DSP Mode Chapter 28, "Test and Debug" Updated BPM signal IO Direction in Table 28-1 Added new note to Section 28.4.3 JTAG (TAP) and Serial Wire Debug selection Chapter 31, "Processor Ball Map and Pin Location" Removed "BOOT_PWR_EN" from Table 31-1 Processor Ball Names Chapter 36, "Electrical Specifications" Removed "BOOT_PWR_EN" from Table 36-3 Single-Ended Signal DC Characteristics as Inputs or Outputs Added new section 36.2 Storage Conditions
October 2021	1.3	Chapter 1, "Introduction" Table 1-2 Updated Sku 6 & 7 typo on TCC support Added new SKU 14 to the table Added new note under table Chapter 4, "Thermal Management" Updated Section 4.2.2.3.1 and added a new note. Updated Section 4.2.4 with a new table and notes on the Dynamic Temperature Range(DTR).

Revision Date	Revision Number	Description
October 2021	1.3	Chapter 5, "Memory" Added new table with DDR4 Channels Population Rules in Section 5.1.1 Updated Table 5-1 Maximum RPC for 3733MT/s Removed Note under Section 5.1.5 Added Max Frequency for SKu 14 Updated Table 5-6 LPDDR4/4x DRAMs Configurations Chapter 9, "Flexible I/O" Added new note under Figure 9-1. Updated note under section 9.3.1. Chapter 15, "Gigabit Ethernet Controller and Time-Sensitive Networking" Updated table 15-8 Supported System Configurations Chapter 20, "Clocking" Updated Figure 20-4 PSE_Gbe Clocking Chapter 22, "Intel® Programmable Services Engine (Intel® PSE)" Added new bullet under Section 22.12 for G3 power cycle Updated Section 22.13.3.1 Bit Rate Configuration Removed Section 22.13.4.7 Restricted Operation Mode Removed Section 22.13.4.10 Power Down (Sleep Mode) Removed Section 22.13.5.4 Debug on CAN Support Removed Section 22.13.5.4.1 Filtering for Debug Messages and Table 22-9 Removed Section 22.13.5.4.2 Debug Message Handling and Figure 22-9 Removed Section 22.13.6.4 Tx Queue Removed Section 22.13.6.5 Mixed Dedicated Tx Buffers / Tx FIFO and Figure 22-10 Removed Section 22.13.6.6 Mixed Dedicated Tx Bufferes / Tx Queue and Figure 22-11 Removed I2S DSP Mode from Table 22-22 Chapter 26, "Legacy Interfaces" Section 26.4.2 Signal description table is updated on the Crystal Input 1 maximum Voltage Chapter 28, "Test and Debug" Updated Section 28.4.1.2 Arm* Debug via JTAG (TAP) Chapter 30, "Global Device IDs"Table 30-3 Updated Device ID for GPU (16 Execution Unit (EU) SKU). Updated Description for Device ID 4536 Chapter 36, "Electrical Specifications" Updated all instances of VoH Min to Vcc - 0.45V in Table 36-3 Added PROC_PWR_GD to Table 36-3 (Sheet 11 of 11) and Table 36-4 Updated Table 36-6 GTL Signal Group and Open Drain (OD) Signal Group DC Specifications for VOL, IOL, RON PD and added a new note.
April 2021	1.2	**Updated:** • Table 3-11, Platform Voltage Rails • Table 3-12, Additional Voltage Rail Signals • Table 36-1, Integrated Clock Specification • Table 36-6, GTL and OD DC Specification (Compute Die)
March 2021	1.0	Initial release.

§ §

1 Introduction

This is the core reference document for external design specifications. Information provided here takes precedence, if there are any discrepancies found in related documents.

Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors platform is targeted towards various Internet of Things (IoT) segments, such as industrial, transportation, retail, and embedded. It features real time compute with technologies such as Time-Sensitive Networking (TSN) and Intel® Time Coordinated Computing (Intel® TCC), which are expected to drive the future of IoT.

The Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors are Intel® Architecture (IA) Multi-Chip Processor (MCP) 2-Chip Package, built on a 10-nanometer Compute Die and a 14-nanometer Platform Controller Hub (PCH) into a single package. Both dies are connected through the On Package Interface (OPI).

1.1 About this Manual

This document is intended for Original Equipment Manufacturers (OEMs), Original Design Manufacturers (ODM) and BIOS vendors creating products based on the Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors.

Throughout this document, the name “Processor” is used as a general term and refers to all Processor SKUs, unless specifically noted otherwise. The compute die may be referred to simply as “Compute Die” and the Mule Creek Canyon Platform Controller Hub may be referred to simply as “PCH”.

This manual assumes a working knowledge of the vocabulary and principles of interfaces and architectures such as PCI express* (PCIe*), Universal Serial Bus (USB), Advanced Host Controller Interface (AHCI), eXtensible Host Controller Interface (xHCI), and so forth.

This manual abbreviates PCI buses as B*n*, devices as D*n* and functions as F*n*. For example, Device 31 Function 0 is abbreviated as D31:F0, and Bus 1 Device 8 Function 0 is abbreviated as B1:D8:F0. Generally, the bus number will not be used, and can be considered to be Bus 0. These numbers are shown as decimal unless otherwise indicated.

1.1.1 Terminology Usage

This document uses the terms ‘initiator’ and ‘target’ (formerly known as ‘master’ and ‘slave’).

1.2 References

Specification	Document #/Location
Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors for Internet of Things (IoT) Applications, Datasheet, Volume 2 (Book 1 of 3), Compute Die Registers Only	635255
Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors for Internet of Things (IoT) Applications, Datasheet, Volume 2 (Book 2 of 3), Mule Creek Canyon	636722
Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors for Internet of Things (IoT) Applications, Datasheet, Volume 2 (Book 3 of 3), Intel® Programmable Services Engine (Intel® PSE)	636723

1.3 Processor Overview

Category	Feature Description
CPU Cores	Quad/Dual IA Processor Core • 3-way Superscalar, Out of Order Execution (OOE) • 10 nm processor technology
Modules/Caches	• 1 module of 2 cores (for Dual Core) or 4 cores (for Quad Core) • On-die, parity protected 32KiB 8-way (64 sets) L1 instruction cache and 32KiB 8-way (64 sets) L1 data cache per core • On-die, ECC protected 1.5MiB, 12-way (2048 sets) L2 unified cache per module
Architecture	Intel® 64-bit
Virtualization Architecture	Intel® Virtualization Technology • VTx-2 with Extended Page Table • VT-d
Burst Technology	1/2/3/4 Core Burst Technology • All cores in C0 state will run at the same frequency
Thermal Management	Supported by means of Intel® Thermal Monitor (TM1 and TM2)
Power Management	• Enhanced Intel SpeedStep® Technology & Intel® Speed Shift Technology • Core C-States: C0, C1, C1E, C6, C6S, C7:C10 • Module C-States: MC0, MC6
IFWI Boot Feature	Support Integrated Firmware Image (IFWI) boot from SPI
Other features	Security Technologies: • Branch Monitoring Counters, Intel® AES-NI, PCLMULQD, Intel® Secure Key, Execute Disable Bit, Boot Guard, Intel® SMEP, Intel® SMAP, Intel® MPX, Intel® SHA Extensions, User Mode Instruction Prevention, Read Processor ID Power and Performance Technologies • x2APIC, Cache Line Write Back Debug Technologies • Intel® Processor Trace

Note: Further information on some of these features can be found in chapters 2, 3 and 4.

1.4 Overview

Processor features and capabilities are listed below.

Table 1-1. Processor Features (Sheet 1 of 3)

Component	Category	Specification
CPU	Number of Cores	Up to 4 Tremont Cores
	Burst Speed	≤ 3GHz
	LFM/HFM	400MHz / ≤ 2.0GHz
	L1 Cache	32 KB Instruction, 32KB Data per core
	L2 Cache	1.5MB
	L3 Cache	4MB
	Type	Type 3 BGA 35mm x 24mm
	I/O Count	795
	Pin Count	1493
Package	Minimum Ball Pitch	0.593mm
	Z-Height	1.464 to 2.502mm
	TDP	4.5W-12W
	Temperature Range T_J	PC Client SKUs: 0ºC to 105ºC
		Embedded SKUs: -40ºC to 105ºC
		Industrial SKUs: -40ºC to 110ºC
	Temperature Range T_A	PC Client SKUs: 0ºC to 70ºC
		Embedded SKUs: -40ºC to 85ºC
		Industrial SKUs:-40ºC to 85ºC
Graphics	Gen	Gen11 LP
	LFM/HFM/Burst	200MHz /≤ 500MHz / ≤ 900MHz
	Execution Units	Up to 32
Display	Gen	Gen11 3x Pipe
	DDI 0	eDP / MIPI DSI 4L / DP / HDMI
	DDI 1	DP / HDMI
	DDI 2	DP / HDMI
	Display Serial Interface (DSI)	MIPI-DSI 1.2 @ 2.5Gbps
	Embedded Display Port (eDP*)	eDP 1.3 @ 5.4 Gbps
	Display Port (DP)	DP 1.4* @ 5.4 Gbps
	High Definition Multimedia Interface (HDMI)	HDMI 2.0b @ 5.94 Gbps (With Platform Level Shifters above HD resolution)
	Max Resolution	4K60x3 HDR
Media	Decode/Encode Codec	HEVC/VP9 4K2K60FPS 420 8bit Codec HEVC/VP9 4K2K30FPS 444 8/10bit Codec

Table 1-1. Processor Features (Sheet 2 of 3)

Component	Category	Specification
Memory	Max Size Supported	4x32 LPDDR4/4x 4267MT/s Max 8GB / 4x32 LPDDR4/4x 3200MT/s Max 16GB / 2x64 DDR4 3200MT/s Max 32GB
	Interface	4x32 LPDDR4/x (Memory Down) 2x64 DDR4 (SODIMM & Memory Down)
	Supported Transfer Data Rates (MT/s)	LPDDR4/x = 4267MT/s; 68GB/s DDR4 = 3200 MT/s; 51GB/s
	IBECC	1 bit correction, 2 bits detection
Audio	External Codec Links	HDA, I2S & DMIC
	SDO/SDI	Up to 48Mbps / 24Mbps
	Codec Support	44.1 kHz sampling rate up to 24MHz BCLK
	Audio Engine	Quad Core Tensilica Diamond Extensa LX 6 core with HIFI 3 Audio Engine @ 400 MHz 768KB L2 Cache / L2 SRAM
	Speech Accelerator	GNA 1.0
USB	USB3.x Ports	4 (2x dedicated port[1x Dual Role], 2x multiplexed with PCIe* 3.0)
	Maximum USB 3.1 speed	10Gb/s
	Maximum USB 3.0 Speed	5Gb/s
	USB 2.0 Ports	10 ports
	Maximum USB 2.0 Speed	480Mbps
PCIe* Gen3	Ports	Up to 6 Ports 8 Lanes (multiplexed with HSIO)
	Maximum Speed	8GT/s
SATA Gen3	Maximum Configurable Ports	2
	Maximum Speed	6Gb/s
Storage	eMMC*	5.1
	Maximum eMMC* Speed	400 MBps
	Secure Digital	SD 3.01 SDIO 3.0
	SD Speed	Default Mode: Up to 12.5 MBps High Speed Mode: Up to 25 MBps UHS-I Mode: Up to 100 MBps
SMBus	Ports	1 x SM Bus 3 Wire 2.0
	Maximum Speed	100 kHz

Table 1-1. Processor Features (Sheet 3 of 3)

Component	Category	Specification
Intel® PSE	Real Time micro controller	Arm Cortex-M7 with 384KB CCM + 1MB L2 SRAM
	Pulse Width Modulation (PWM)	16
	I2S	2
	GPIO	60
	Time-aware GPIO	40
	UART	6
	SPI	4
	I^2C	8
	GbE with TSN	2
	Quadrature Encoder Pulse (QEP)	4
	CAN-FD	2
Gigabit Ethernet	Controllers	3 GbE with TSN (2 Controllers are accessible by Intel® PSE)
	Maximum Speed	Full Duplex @2500Mbps(SGMII), 1000Mbps(RGMII) Half and Full Duplex @ 10/100 Mbps
SIO	SPI	3
	SPI Speed	25Mbps (Initiator Mode Only)
	UART	3
	UART Speed	3.8Mbps
	I^2C	8
	I^2C Speed	3.4Mbps (Initiator Mode Only)
Fast SPI	Controller	Controller: 1 Devices supported: 3 (2 for Flash, 1 for TPM) (FST_SPI supports up to 3 loads)
	Maximum Fast SPI Frequency	50 MHz
eSPI	Maximum eSPI Frequency	50 MHz
Clocks	38.4 Xtal inputs 100 MHz Spread Spectrum Clock (SSC)	
Interrupt	2 X Interrupt Controller (8259 & I/O APIC)	21 Interrupts, Message Signaled Interrupts (MSI) Support
Timer		8x HPET Intel® 8254 timer
RTC	256 byte Battery backed RAM	
ACPI	Advanced Configuration Power Interface (ACPI) 6.1 Compliant Power Management	
Security	Technology	Platform Trust Technology (PTT) Gen3 and Trusted Platform Module (TPM2.0)
Real Time (RT)	Hardware	Always Running Timer (ART)
	TSN	Gigabit Ethernet
Intel® Safety Island (Intel® SI)	CPU Core	Latent Fault for MLC and LLC
	Safety Integrity Level	SIL2

Note: Not all functions and capabilities may be available on all SKUs. The table above provides an overview of the processor's capabilities.

1.5 Intel Atom® x6000E Series processors, Intel® Pentium® and Celeron® N and J Series processors Block Diagram

Figure 1-1. Compute Die Block Diagram

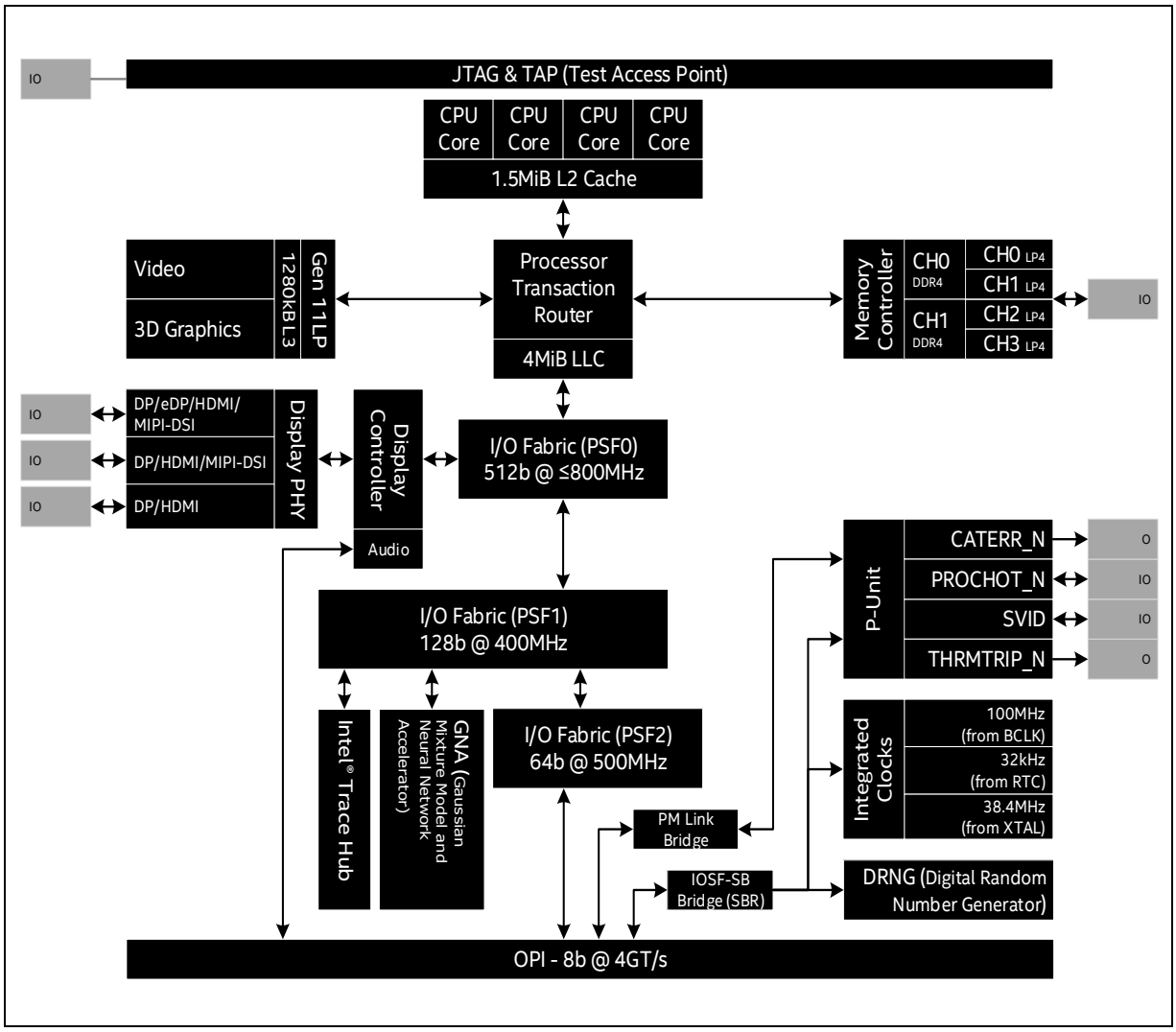


Figure 1-2. PCH Block Diagram

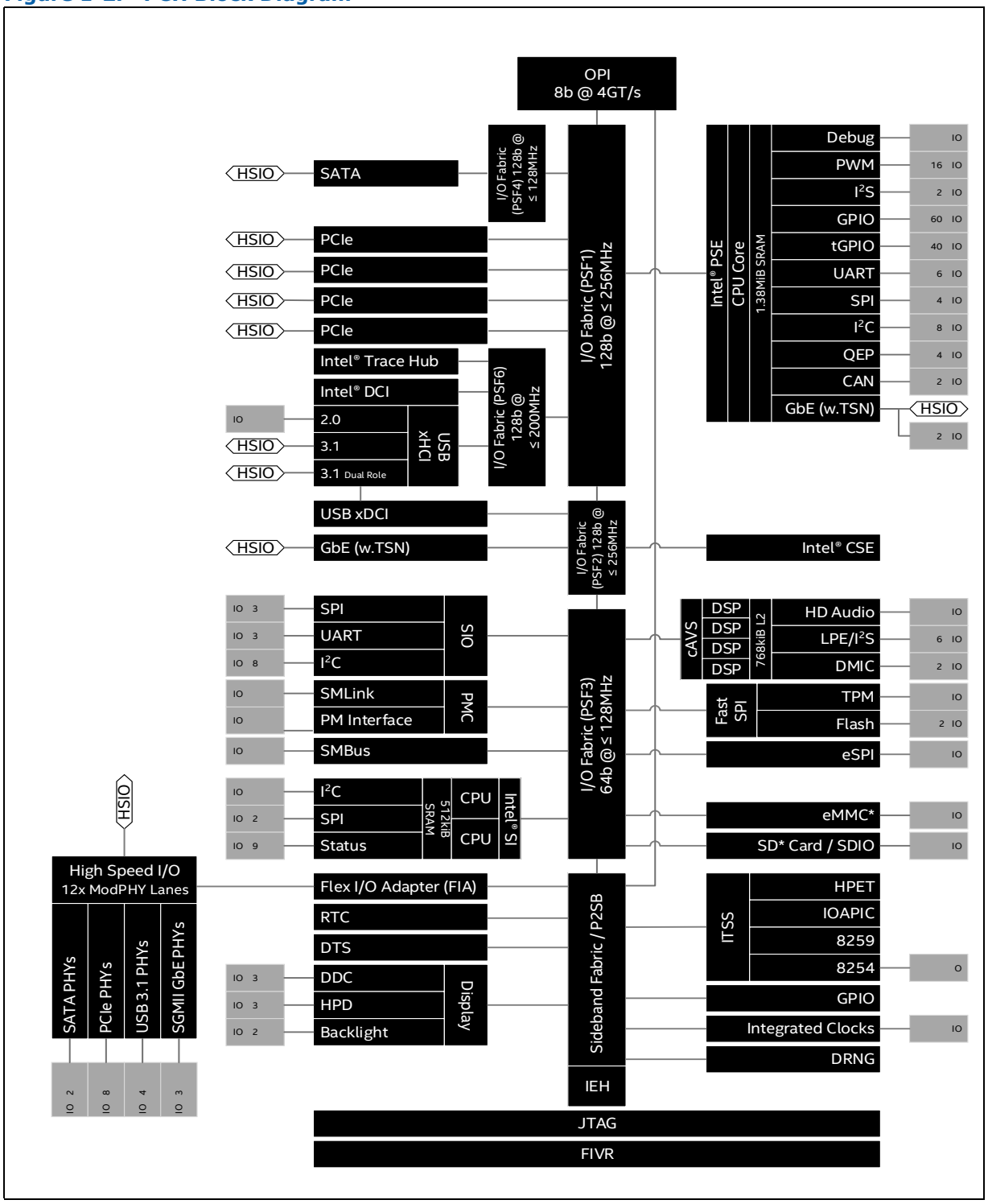
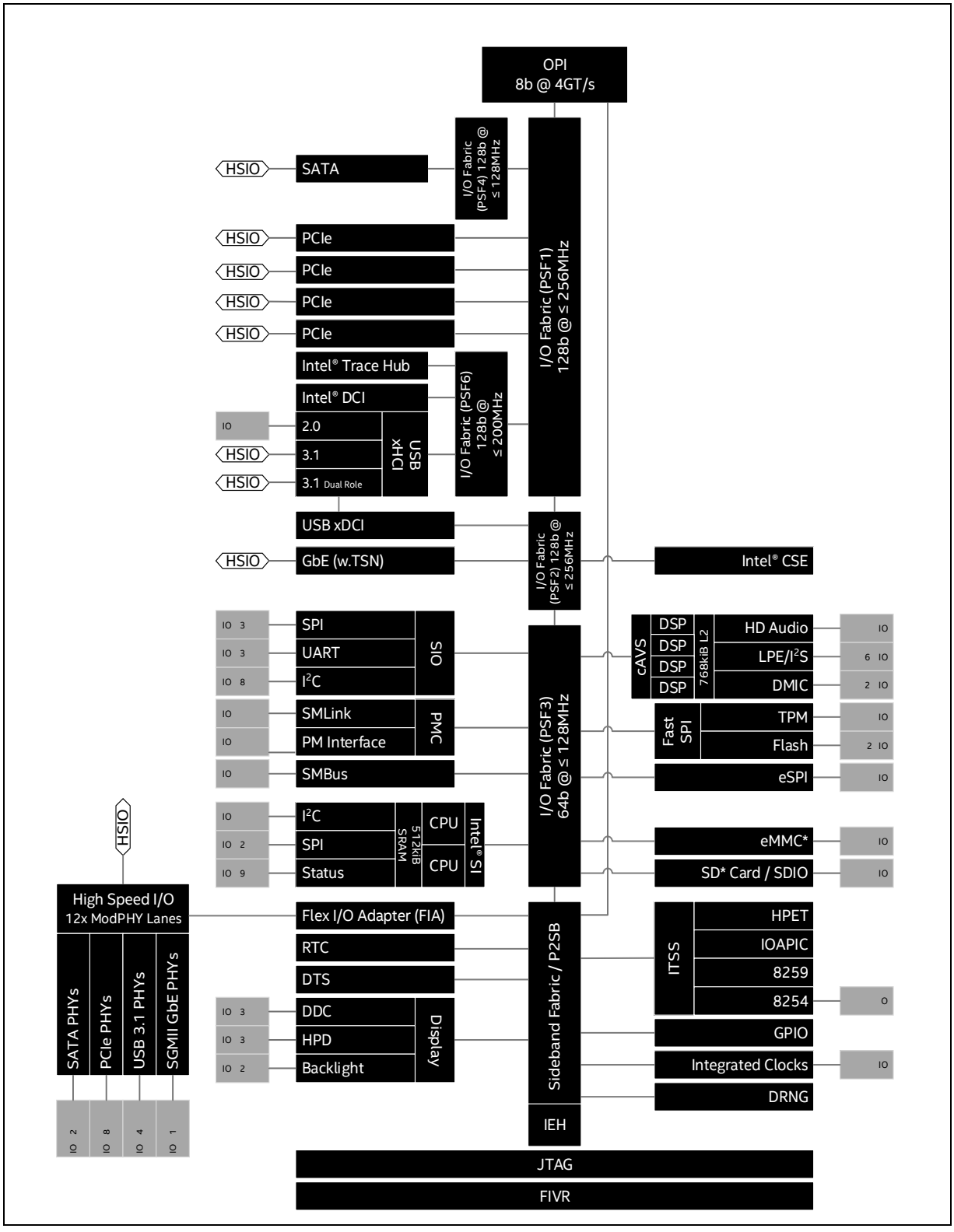


Figure 1-3. PSE Disabled - PCH Block Diagram



1.6 Processor SKUs

Table 1-2. Processor SKU MAP (High Level)

Processor	SKU 1 Intel® Celeron® Processor J6413	SKU 1A Intel® Celeron® Processor J6412	SKU 2 Intel® Pentium® Processor J6426	SKU 3 Intel® Celeron® Processor N6211	SKU 3A Intel® Celeron® Processor N6210	SKU 4 Intel® Pentium® Processor N6415	SKU 5 Intel Atom® x6211E Processor
	PC Client						Embedded
Use Condition	PC Client	PC Client	PC Client	PC Client	PC Client	PC Client	Embedded
Cores	4	4	4	2	2	4	2
Last Level Cache (LLC)	4MB	4MB	4MB	4MB	4MB	4MB	4MB
TDP	10W	10W	10W	6.5W	6.5W	6.5W	6W
HFM Frequency	1.8GHz	2.0GHz	2.0GHz	1.2GHz	1.2GHz	1.2GHz	1.3GHz
Burst (Turbo) Mode Single Core Frequency [1]	3.0GHz	2.6GHz	3.0GHz	3.0GHz	2.6GHz	3.0GHz	3.0GHz
Burst (Turbo) Mode Dual Core Frequency [1]	3.0GHz	2.6GHz	3.0GHz	3.0GHz	2.6GHz	3.0GHz	3.0GHz
Burst (Turbo) Mode Triple Core Frequency	2.7GHz	2.6GHz	2.7GHz	N/A	N/A	2.7GHz	N/A
Burst (Turbo) Mode Quad Core Frequency	2.7GHz	2.6GHz	2.7GHz	N/A	N/A	2.7GHz	N/A
Gen 11LP	16EUs	16EUs	32EUs	16EUs	16EUs	16EUs	16EUs
GFX HFM Frequency	400MHz	400MHz	400MHz	250MHz	250MHz	350MHz	350MHz
GFX Burst (Turbo) Mode Frequency	800MHz	800MHz	850MHz	750MHz	750MHz	800MHz	750MHz
T_J	0 to 105°C	0 to 105°C	0 to 105°C	0 to 105°C	0 to 105°C	0 to 105°C	-40 to 105°C
T_A	0 to 70°C	0 to 70°C	0 to 70°C	0 to 70°C	0 to 70°C	0 to 70°C	-40 to +85°C
Integrated Heat Spreader (I.H.S) (i.e.Lid)	No	No	No	No	No	No	Yes
Intel® Programmable Services Engine	Enabled	Disabled	Enabled	Enabled	Disabled	Enabled	Enabled
IBECC (in Band)	No	No	No	No	No	No	Yes
Intel® TCC	No	No	No	No	No	No	No
FuSa	No	No	No	No	No	No	No

Processor	SKU 6 Intel Atom® x6413E Processor	SKU 7 Intel Atom® x6425E Processor	SKU 8 Intel Atom® x6212RE Processor	SKU 8PU Intel Atom® x6214RE Processor	SKU 9 Intel Atom® x6414RE Processor	SKU 9PU Intel Atom® x6416RE Processor	SKU10 Intel Atom® x6425RE Processor	SKU 11 Intel Atom® x6427FE Processor	SKU 12 Intel Atom® x6200FE Processor
	Embedded		Indu					FuSa	
Use Condition	Embedded	Embedded	Industrial	Industrial	Industrial	Industrial	Industrial	Industrial	Industrial
Cores	4	4	2	2	4	4	4	4	2
Last Level Cache (LLC)	4MB	4MB	4MB	4MB	4MB	4MB	4MB	4MB	2MB
TDP	9W	12W	6W	6W	9W	9W	12W	12W	4.5W
HFM Frequency	1.5GHz	2.0GHz	1.2GHz	1.4GHz	1.5GHz	1.7GHz	1.9GHz	1.9GHz	1.0GHz
Burst (Turbo) Mode Single Core Frequency [1]	3.0GHz	3.0GHz	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Burst (Turbo) Mode Dual Core Frequency [1]	3.0GHz	3.0GHz	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Burst (Turbo) Mode Triple Core Frequency	2.7GHz	2.7GHz	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Burst (Turbo) Mode Quad Core Frequency	2.7GHz	2.7GHz	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Gen 11LP	16EUs	32EUs	16EUs	16EUs	16EUs	16EUs	32EUs	32EUs	N/A
GFX HFM Frequency	500MHz	500Mhz	350Mhz	400MHz	400MHz	450MHz	400MHz	400MHz	N/A
GFX Burst (Turbo) Mode Frequency	750MHz	750MHz	N/A	N/A	N/A	N/A	N/A	N/A	N/A
T_J	-40 to 105°C	-40 to 105°C	-40 to 110°C	-40 to +110C	-40 to 110°C	-40 to +110C	-40 to 110°C	-40 to 110°C	-40 to 110°C
T_A	-40 to +85°C	-40 to +85°C	-40 to +85°C	-40 to +85C	-40 to +85°C	-40 to +85C	-40 to +85°C	-40 to +85°C	-40 to +85°C

Processor	SKU 6 Intel Atom® x6413E Processor	SKU 7 Intel Atom® x6425E Processor	SKU 8 Intel Atom® x6212RE Processor	SKU 8PU Intel Atom® x6214RE Processor	SKU 9 Intel Atom® x6414RE Processor	SKU 9PU Intel Atom® x6416RE Processor	SKU10 Intel Atom® x6425RE Processor	SKU 11 Intel Atom® x6427FE Processor	SKU 12 Intel Atom® x6200FE Processor
Integrated Heat Spreader (I.H.S) (i.e.Lid)	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Intel® Programmable Services Engine	Enabled	Enabled	Enabled	Enabled	Enabled	Enabled	Enabled	Enabled	Enabled
IBECC (in Band)	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Intel® TCC [2]	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
FuSa	No	No	No	No	No	No	No	Yes	Yes

Note: 1. All cores in C0 state run at the same frequency - frequency is only available for SKUs 1-2, 4, and 6-7 when at least two cores are disabled or are in a C1 state or as above.

2. Processors that support Intel® Time Coordinated Computing (TCC) are expected to have improved high bandwidth workload performance on PCH features such as PCI Express and Serial ATA.

1.7 Processor Volatility Statement

This processor does not retain any end user data when powered down and/or when the processor is physically removed.

Note: Powered down refers to state which all processor power rails are off.

§ §

2 Technologies

This chapter provides a high-level description of Intel technologies implemented in the processor.

The implementation of the features may vary between the processor SKUs.

Details on the different technologies of Intel processors and other relevant external notes are located at the Intel technology web site: http://www.intel.com/technology/

2.1 Tremont ISA Extensions

Tremont is the codename for the next generation 64-bit Intel Atom® CPU cores in the processor.

- x87 FPU and SIMD State Management
- Multimedia Extensions (MMX) Technology
- Streaming SIMD Extensions (SSE), SSE2, SSE3, SSSE3, SSE4.1 & SSE4.2 Extensions
- IA-32e mode: 64-bit mode instructions
- Virtual Machine Extensions (VMX) Instructions
- Safer Mode Extensions (SMX) Instructions
- CRC32 - Accumulate CRC32 Value
- POPCNT - Return the Count of Number of Bits Set to 1
- Intel® AES-NI
- PCLMULQDQ - Carry-Less Multiplication Quadword
- RDRAND - Read Random Number
- PREFETCHW - Prefetch Data into Caches in Anticipation of a Write
- FS/GS base access
  - — RDFSBASE/RDGSBASE - Read FS/GS Segment Base
  - — WRFSBASE/WRGSBASE - Read FS/GS Segment Base
- Intel® SHA Extensions
- RDSEED - Read Random SEED
- CLAC - Clear AC Flag in EFLAGS Register
- STAC - Set AC Flag in EFLAGS Register
- CLFLUSHOPT - Flush Cache Line Optimized
- XSAVEC - Save Processor Extended States with Compaction
- XSAVES - Save Processor Extended States Supervisor
- Intel® Memory Protection Extensions (Intel® MPX)
- UMIP - User-Mode Instruction Prevention
- PTWRITE - Write Data to a Processor Trace Packet
- RDPID - Read Processor ID
- CLWB - Cache Line Write Back
- GFNI (SSE) - Galois Field New Instructions (SSE)
- Split Lock Detection
- CLDEMOTE - Cache Line Demote
- Direct stores
  - MOVDIRI - Move Doubleword as Direct Store
  - MOVDIR64B - Move 64 Bytes as Direct Store

- User wait
  - TPAUSE - Timed PAUSE
  - UMONITOR - User Level Set Up Monitor Address
  - UMWAIT - User Level Monitor Wait
- MOVBE - Move Data After Swapping Bytes
- RDTSCP - Read Time-Stamp Counter and Process ID
- WBINVD - Write Back and Invalidate Cache
- XRSTOR - Restore Processor Extended States
- XRSTORS - Restore Processor Extended States Supervisor
- XSAVE - Save Processor Extended States
- XSAVEOPT - Save Processor Extended States Optimized

Note: No variant of AVX instructions are supported.

2.2 Security Technologies

2.2.1 Branch Monitoring Counters

Branch monitoring technology allows monitor and detection a set of heuristics within an execution window in a program. This heuristics can be used for detecting abnormal behavior in code execution and signal the anti-malware software of its occurrence.

These technology allows software such Anti-Virus software to receive a signal (interrupt) when a counter threshold has been reached. Branch Monitoring allows software to perform non-intrusive runtime analysis of ROP (Return Oriented Programming) attacks on applications.

The heuristics are based of certain performance monitoring statistics, measured dynamically over a short configurable window period. Anti-malware software has the responsibility to configure the Hardware statistics of interest and the Window size via MSR registers. Anti Malware SW is also for responsible for post-processing any signaled event due to a detection condition. Such signaling is not considered 100% reliable and thus the anti-malware software is the ultimate decision maker to avoid false positives, while at the same time maintaining sufficient sensitivity for detecting malware.

2.2.2 Intel® Advanced Encryption Standard New Instructions (Intel® AES-NI)

The processor supports Intel® Advanced Encryption Standard New Instructions (Intel® AES-NI) that are a set of Single Instruction Multiple Data (SIMD) instructions that enable fast and secure data encryption and decryption based on the Advanced Encryption Standard (AES). Intel® AES-NI are valuable for a wide range of cryptographic applications, such as applications that perform bulk encryption/ decryption, authentication, random number generation, and authenticated encryption. AES is broadly accepted as the standard for both government and industry applications, and is widely deployed in various protocols.

Intel® AES-NI consists of six Intel® SSE instructions. Four instructions, AESENC, AESENCLAST, AESDEC, and AESDELAST facilitate high performance AES encryption and decryption. The other two, AESIMC and AESKEYGENASSIST, support the AES key expansion procedure. Together, these instructions provide full hardware for supporting AES; offering security, high performance, and a great deal of flexibility.

This generation of the processor has increased the performance of the Intel® AES-NI significantly compared to previous products.

The Intel® AES-NI specifications and functional descriptions are included in the *Intel® 64 Architectures Software Developer's Manual, Volume 2*. Available at:

http://www.intel.com/products/processor/manuals

2.2.3 PCLMULQDQ (Perform Carry-Less Multiplication Quad word) Instruction

The processor supports the carry-less multiplication instruction, PCLMULQDQ. PCLMULQDQ is a Single Instruction Multiple Data (SIMD) instruction that computes the 128-bit carry-less multiplication of two 64-bit operands without generating and propagating carries. Carry-less multiplication is an essential processing component of several cryptographic systems and standards. Hence, accelerating carry-less multiplication can significantly contribute to achieving high speed secure computing and communication.

PCLMULQDQ specifications and functional descriptions are included in the *Intel® 64 Architectures Software Developer's Manual, Volume 2*. Available at:

http://www.intel.com/products/processor/manuals

2.2.4 Intel® Secure Key

The processor supports Intel® Secure Key (formerly known as Digital Random Number Generator (DRNG), a software visible random number generation mechanism supported by a high quality entropy source. This capability is available to programmers through the RDRAND instruction. The resultant random number generation capability is designed to comply with existing industry standards in this regard (ANSI X9.82 and NIST SP 800-90).

Some possible usages of the RDRAND instruction include cryptographic key generation as used in a variety of applications, including communication, digital signatures, secure storage, etc.

RDRAND specifications and functional descriptions are included in the *Intel® 64 Architectures Software Developer's Manual, Volume 2*. It is available at:

http://www.intel.com/products/processor/manuals

Usage recommendations for the PCH DRNG (Digital Random Number Generator) - PCH DRNG activity time must be restricted to a maximum of 3,153,600 seconds across the entire duration of the product's operating lifetime. This activity time can be quantified by the number of accesses triggered by firmware running on the Intel® Programmable Services Engine ARM microcontroller core, with each single access resulting in a maximum DRNG activity time of 2.26µs.

Note: In cases where the Zephyr reference firmware stack provided by Intel is used, the number of accesses can be measured by the number of calls of the mbedtls_ctr_drbg_random() function.

2.2.5 Execute Disable Bit

The Execute Disable Bit allows memory to be marked as non-executable when combined with a supporting operating system. If code attempts to run in non-executable memory, the processor raises an error to the operating system. This feature can prevent some classes of viruses or worms that exploit buffer overrun vulnerabilities and can, thus, help improve the overall security of the system.

2.2.6 Boot Guard Technology

Boot Guard technology is a part of boot integrity protection technology. Boot Guard can help protect the platform boot integrity by preventing execution of unauthorized boot blocks. With Boot Guard, platform manufacturers can create boot policies such that invocation of an unauthorized (or untrusted) boot block will trigger the platform protection per the manufacturer's defined policy.

With verification based in the hardware, Boot Guard extends the trust boundary of the platform boot process down to the hardware level.

Boot Guard accomplishes this by:

- Providing of hardware-based Static Root of Trust for Measurement (S-RTM) and the Root of Trust for Verification (RTV) using Intel architectural components.
- Providing of architectural definition for platform manufacturer Boot Policy.
- Enforcing of manufacture provided Boot Policy using Intel architectural components.

Benefits of this protection is that Boot Guard can help maintain platform integrity by preventing re-purposing of the manufacturer’s hardware to run an unauthorized software stack.

2.2.7 Intel® Supervisor Mode Execution Protection (SMEP)

Intel® Supervisor Mode Execution Protection (SMEP) is a mechanism that provides the next level of system protection by blocking malicious software attacks from user mode code when the system is running in the highest privilege level. This technology helps to protect from virus attacks and unwanted code from harming the system.

2.2.8 Intel® Supervisor Mode Access Protection (SMAP)

Intel® Supervisor Mode Access Protection (SMAP) is a mechanism that provides next level of system protection by blocking a malicious user from tricking the operating system into branching off user data. This technology shuts down very popular attack vectors against operating systems.

2.2.9 Intel® Memory Protection Extensions (Intel® MPX)

Intel® MPX provides hardware accelerated mechanism for memory testing (heap and stack) buffer boundaries in order to identify buffer overflow attacks.

An Intel® MPX enabled compiler inserts new instructions that tests memory boundaries prior to a buffer access. Other Intel® MPX commands are used to modify a database of memory regions used by the boundary checker instructions.

The Intel® MPX ISA is designed for backward compatibility and will be treated as no-operation instructions (NOPs) on older processors.

Intel® MPX can be used for:

- Efficient runtime memory boundary checks for security-sensitive portions of the application.
- As part of a memory checker tool for finding difficult memory access errors. Intel® MPX is significantly of magnitude faster than software implementations.

Intel® MPX emulation (without hardware acceleration) is available with the Intel® C++ Compiler 13.0 or newer.

2.2.10 Intel® Secure Hash Algorithm Extensions (Intel® SHA Extensions)

The Secure Hash Algorithm (SHA) is one of the most commonly employed cryptographic algorithms. Primary usages of SHA include data integrity, message authentication, digital signatures, and data de-duplication. As the pervasive use of security solutions continues to grow, SHA can be seen in more applications now than ever. The Intel® SHA Extensions are designed to improve the performance of these compute-intensive algorithms on Intel® architecture-based processors.

The Intel® SHA Extensions are a family of seven instructions based on the Intel® Streaming SIMD Extensions (Intel® SSE) that are used together to accelerate the performance of processing SHA-1 and SHA-256 on Intel architecture-based processors. Given the growing importance of SHA in our everyday computing devices, the new instructions are designed to provide a needed boost of performance to hashing a single buffer of data. The performance benefits will not only help improve responsiveness and lower power consumption for a given application, they may enable developers to adopt SHA in new applications to protect data while delivering to their user experience goals. The instructions are defined in a way that simplifies their mapping into the algorithm processing flow of most software libraries, thus enabling easier development.

2.2.11 User Mode Instruction Prevention (UMIP)

User Mode Instruction Prevention (UMIP) provides additional hardening capability to the OS kernel by allowing certain instructions to execute only in supervisor mode (Ring 0).

If the OS opt-in to use UMIP, the following instruction are enforced to run in supervisor mode:

- SGDT - Store the GDTR register value
- SIDT - Store the IDTR register value
- SLDT - Store the LDTR register value
- SMSW - Store Machine Status Word
- STR - Store the TR register value

An attempt at such execution in user mode causes a general protection exception (#GP).

UMIP specifications and functional descriptions are included in the *Intel® 64 Architectures Software Developer's Manual, Volume 3*. Available at:

http://www.intel.com/products/processor/manuals

2.2.12 Read Processor ID (RDPID)

A companion instruction that returns the current logical processor's ID and provides a faster alternative to using the RDTSCP instruction.

RDPID specifications and functional descriptions are included in the *Intel® 64 Architectures Software Developer's Manual, Volume 2*. Available at:

http://www.intel.com/products/processor/manuals

2.3 Power and Performance Technologies

2.3.1 Intel® Smart Cache Technology

The Intel® Smart Cache Technology is a shared Last Level Cache (LLC).

The LLC may also be referred to as a 3rd level cache.

The LLC is shared between all IA cores as well as the Processor Graphics. Also, is an additional 1280kB L3 cache dedicated to the Graphics.

The 1st level cache is not shared between physical cores and each physical core has a separate level 1 cache. The 2nd level cache is shared between all physical cores.

For SKUs 1 - 11, the size of the LLC is 4MB and is a 16 way associative cache. For SKU 12, the size of the LLC is 2MB and is a 8 way associative cache. It is ECC protected.

2.3.2 IA Core Level 1 and Level 2 Caches

The 1st level cache is divided into a data cache and an instruction cache. The processor 1st level cache size is 32KB for data and 32KB for instructions. The 1st level cache is an 8 way associative cache and is parity protected.

The 2nd level cache holds both data and instructions. The L2 cache size is 1.5MB and is a 12 way associative cache. It is shared across the 4 cores in the module and is ECC protected (1-bit correct & 2-bits detect).

2.3.3 Enhanced Intel SpeedStep® Technology

Enhanced Intel SpeedStep® Technology enables OS to control and select P-state. The following are the key features of Enhanced Intel SpeedStep® Technology:

- Multiple frequency and voltage points for optimal performance and power efficiency. These operating points are known as P-states.
- Frequency selection is software controlled by writing to processor MSRs. The voltage is optimized based on the selected frequency and the number of active processor IA cores.
  - Once the voltage is established, the PLL locks on to the target (refer to Section 1.1.1 for more information on target) frequency.
  - All active processor IA cores share the same frequency and voltage. In a multi-core processor, the highest frequency P-state requested among all active IA cores is selected.
  - Software-requested transitions are accepted at any time. If a previous transition is in progress, the new transition is deferred until the previous transition is completed.
- The processor controls voltage ramp rates internally to ensure glitch-free transitions.

Notes: Because there is low transition latency between P-states, a significant number of transitions per-second are possible. All of the Compute Die Cores must be in the same P-state at any given time.

Enhanced Intel SpeedStep® Technology should be disabled by BIOS in safety critical systems. Enhanced Intel SpeedStep® Technology may need to be disabled by BIOS in real time systems, since it can cause latency jitter.

2.3.4 Intel® Speed Shift Technology

Intel® Speed Shift Technology is an energy efficient method of frequency control by the hardware rather than relying on OS control. OS is aware of available hardware P-states and request a desired P-state or it can let the hardware determine the P-state. The OS request is based on its workload requirements and awareness of processor capabilities. Processor decision is based on the different system constraints for example: Workload demand, thermal limits while taking into consideration the minimum and maximum levels and activity window of performance requested by the Operating System.

Notes: Intel® Speed Shift Technology may not be available on all SKUs. Also, it is not possible for different cores to have different P-states.

Intel® Speed Shift Technology should be disabled by BIOS in safety critical systems. Intel® Speed Shift Technology may need to be disabled by BIOS in real time systems, since it can cause latency jitter.

2.3.5 Intel® 64 Architecture x2APIC

The x2APIC architecture extends the xAPIC architecture that provides key mechanisms for interrupt delivery. This extension is primarily intended to increase processor addressability.

Specifically, x2APIC:

- Retains all key elements of compatibility to the xAPIC architecture:
  - Delivery modes
  - Interrupt and processor priorities
  - Interrupt sources
  - Interrupt destination types
- Provides extensions to scale processor addressability for both the logical and physical destination modes
- Adds new features to enhance performance of interrupt delivery
- Reduces complexity of logical destination mode interrupt delivery on link based architectures

The key enhancements provided by the x2APIC architecture over xAPIC are the following:

- Support for two modes of operation to provide backward compatibility and extensibility for future platform innovations:
  - In xAPIC compatibility mode, APIC registers are accessed through memory mapped interface to a 4K-Byte page, identical to the xAPIC architecture.
  - In x2APIC mode, APIC registers are accessed through Model Specific Register (MSR) interfaces. In this mode, the x2APIC architecture provides significantly increased processor addressability and some enhancements on interrupt delivery.
- Increased range of processor addressability in x2APIC mode:
  - Physical xAPIC ID field increases from 8 bits to 32 bits, allowing for interrupt processor addressability up to 4G-1 processors in physical destination mode. A

processor implementation of x2APIC architecture can support fewer than 32-bits in a software transparent fashion.

  - Logical xAPIC ID field increases from 8 bits to 32 bits. The 32-bit logical x2APIC ID is partitioned into two sub-fields – a 16-bit cluster ID and a 16-bit logical ID within the cluster. Consequently, ((2^20) - 16) processors can be addressed in logical destination mode. Processor implementations can support fewer than 16 bits in the cluster ID sub-field and logical ID sub-field in a software agnostic fashion.

- More efficient MSR interface to access APIC registers:
  - To enhance inter-processor and self-directed interrupt delivery as well as the ability to virtualize the local APIC, the APIC register set can be accessed only through MSR-based interfaces in x2APIC mode. The Memory Mapped IO (MMIO) interface used by xAPIC is not supported in x2APIC mode.
- The semantics for accessing APIC registers have been revised to simplify the programming of frequently-used APIC registers by system software. Specifically, the software semantics for using the Interrupt Command Register (ICR) and End Of Interrupt (EOI) registers have been modified to allow for more efficient delivery and dispatching of interrupts.
- The x2APIC extensions are made available to system software by enabling the local x2APIC unit in the "x2APIC" mode. To benefit from x2APIC capabilities, a new operating system and a new BIOS are both needed, with special support for x2APIC mode.
- The x2APIC architecture provides backward compatibility to the xAPIC architecture and forward extensible for future Intel platform innovations.

2.3.6 Cache Line Write Back (CLWB)

Writes back to memory the cache line (if dirty) that contains the linear address specified with the memory operand from any level of the cache hierarchy in the cache coherence domain. The line may be retained in the cache hierarchy in non-modified state. Retaining the line in the cache hierarchy is a performance optimization (treated as a hint by hardware) to reduce the possibility of cache miss on a subsequent access. Hardware may choose to retain the line at any of the levels in the cache hierarchy, and in some cases, may invalidate the line from the cache hierarchy. The source operand is a byte memory location.

2.3.7 Intel® Programmable Services Engine

Programmable Services Engine is an IP that serves primarily as the connection point for many of the sensors and real time peripherals across all platform SKUs. This IP provides the ability to "offload services" from the processor reducing the bandwidth consumption. Compute die cores can be turned off as services are offloaded, reducing power consumption.

Tight loop industrial controls, low power Network Proxy mode for printers, sensing use cases that ISH used to support in client PCHs, are examples of services that would be offloaded to Programmable Services Engine.

Note: For more information, refer to Table 1-1 and Chapter 22, "Intel® Programmable Services Engine (Intel® PSE)".

2.3.8 Intel® Safety Island (Intel® SI)

Intel® Safety Island is a dedicated diagnostic IP which collects all errors originating from different elements / sub-parts of the processor and signals to the system using dedicated interfaces.

Note: For more information, refer to Chapter 23, "Intel® Safety Island (Intel® SI)".

2.3.9 Converged Audio Voice Speech (cAVS)

Converged Audio Voice Speech (cAVS) subsystem consists of a collection of controller, DSP, memory, and link interfaces that provides the audio experience to the platform. This subsystem provides streaming of audio from the host SW to external audio codecs, with the host CPU and/or DSP providing the audio enrichment. It may also be used as a host based sensor hub for managing various context info on the platform

Note: For more information, refer to Table 1-1 and Chapter 10, "Audio, Voice, and Speech".

§ §

3 Power Management

This chapter provides information on the following power management topics:

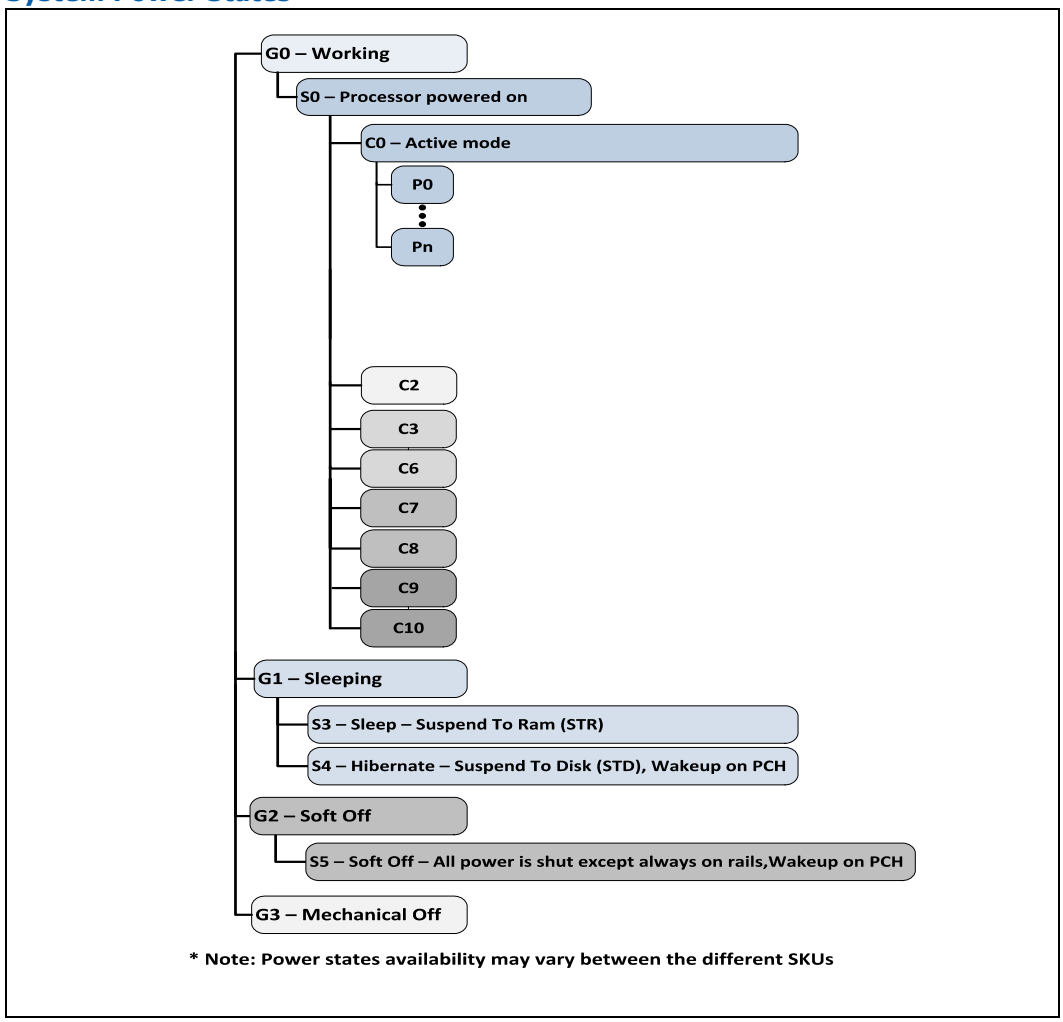
- Advanced Configuration and Power Interface (ACPI) States Supported
- Processor IA Core Power Management
- Power Management Interface Signals
- Fully Integrated Voltage Regulator (FIVR)

Note: The Deep Sx state is not supported by the processor.

3.1 Power Management States Supported

This section describes the ACPI states supported by the processor.

Figure 3-1. System Power States



This figure shows how the platform ACPI states work with the compute die C power states (package C-states) and the compute die P performance states.

Note: All cores within the compute die will share the same P performance states at any given time.

Table 3-1. System States

The following table describes the Gx/Sx ACPI states.

State	Description
G0/S0	Full On
G1/S3	Sleep/Suspend-to-RAM (STR). Context saved to memory
G1/S4	Suspend-to-Disk (STD). All power lost (except wake-up on PCH).
G2/S5	Soft off. All power lost (except wake-up on PCH). Total reboot.
G3	Mechanical off. All power removed from system.

Table 3-2. Integrated Memory Controller (IMC) States

The following table provides information on the IMC states.

State	Description
Power up	CKE asserted. Active mode.
Pre-charge Power down	CKE de-asserted (not self-refresh) with all banks closed.
Active Power down	CKE de-asserted (not self-refresh) with minimum one bank active.
Self-Refresh	CKE de-asserted using device self-refresh.

Table 3-3. G, S, and C Interface State Combinations

The following table provides information on how the Global and Sleep states relate to the Processor states and system clocks.

Global (G) State	Sleep (S) State	Processor Package (C) State	Processor State	System Clocks	Description
G0	S0	C0	Full On	On	Full On
G0	S0	C2	Deep Sleep	On	Deep Sleep
G0	S0	C3	Deep Sleep	On	Deep Sleep
G0	S0	C6/C7	Deep Power Down	On	Deep Power Down
G0	S0	C8/C9	Off	On	Deeper Power Down
G0	S0ix	C10	Off	On	Enters S0ix
G1	S3	Power off	Off	Off, except RTC	Suspend to RAM
G1	S4	Power off	Off	Off, except RTC	Suspend to Disk
G2	S5	Power off	Off	Off, except RTC	Soft Off
G3	N/A	Power off	Off	Power off	Hard off

Table 3-4. State Transition Rules for the PCH

The following table provides information on the state Gx/Sx/Cx state transitions.

Present State	Transition Trigger	Next State
G0/S0/C0	• OPI Msg • SLP_EN bit set • Power Button Override[3] • Mechanical Off/Power Failure	• G0/S0/Cx • G1/Sx or G2/S5 state • G2/S5 • G3
G0/S0/Cx	• OPI Msg • Power Button Override[3] • Mechanical Off/Power Failure	• G0/S0/C0 • S5 • G3
G1/S3	• Any Enabled Wake Event • Power Button Override[3] • Mechanical Off/Power Failure	• G0/S0/C0[2] • G2/S5 • G3
G1/S4	• Any Enabled Wake Event • Power Button Override[3] • Mechanical Off/Power Failure	• G0/S0/C0[2] • G2/S5 • G3
G2/S5	• Any Enabled Wake Event • Mechanical Off/Power Failure	• G0/S0/C0[2] • G3
G3	• Power Returns	• S0/C0 (reboot) or G2/S5[4] (stay off until power button pressed or other wake event)[1,2]

Notes:

1. Some wake events can be preserved through power failure.
2. Transitions from the S3–S5 or G3 states to the S0 state are deferred until PMC_BATLOW_N is inactive in mobile configurations.
3. Includes all other applicable types of events that force the host into and stay in G2/S5.
4. If the system was in G1/S4 before G3 entry, then the system will go to S0/C0 or G1/S4.

Table 3-5. System Power Plane

The System has several independent power planes as described in the table. When a particular power plane is shut off, it should go to a 0V level.

Plane	Controlled By	Description
CPU	PMC_SLP_S3_N signal	The PMC_SLP_S3_N signal can be used to cut the power to the compute die completely.
Main (Applicable to Platform, PCH does not have a Main well)	PMC_SLP_S3_N signal	When PMC_SLP_S3_N goes active, power can be shut off to any circuit not required to wake the system from the S3 state. Since the S3 state requires that the memory context be preserved, power must be retained to the main memory. The processor, PCI Express* will typically be power-gated when the Main power plane is shut, although there may be small subsections powered. ***Note:*** The PCH power is not controlled by the PMC_SLP_S3_N signal, but instead by the PMC_SLP_SUS_N signal.
Device & Memory	PMC_SLP_S4_N signal PMC_SLP_S5_N signal	When PMC_SLP_S4_N goes active, power can be shut off to any circuit not required to wake the system from the S4. Since the memory context does not need to be preserved in the S4 state, the power to the memory can also be shut down. When PMC_SLP_S5_N goes active, power can be shut off to any circuit not required to wake the system from the S5 state. Since the memory context does not need to be preserved in the S5 state, the power to the memory can also be shut.

Plane	Controlled By	Description
Primary/ Suspend Well	PMC_SLP_SUS_N	This signal is asserted when the Primary/Suspend rails can be externally shut off for enhanced power saving
VCCIO & VCCSTG	PMC_CPU_C10_GATE_N	This signal is asserted (LOW) when the processor enters C10 and can handle VCCIO, VCC_AGSH, VCCSTG, and VCCSFR_OC being lowered to 0V.
DEVICE[n]	Implementation Specific	Individual subsystems may have their own power plane. For example, GPIO signals may be used to control the power to disk drives, audio amplifiers, or the display screen.

3.2 Processor IA Core Power Management

While executing code, Enhanced Intel SpeedStep® Technology and Intel Speed Shift® technology optimizes the processor's IA core frequency and voltage based on workload. Each frequency and voltage operating point is defined by ACPI as a P-state. When the processor is not executing code, it is idle. A low-power idle state is defined by ACPI as a C-state. In general, deeper power C-states have longer entry and exit latencies but higher power savings.

Note: The performance configuration requires special tuning or adjustment of specific power management features.

3.2.1 OS/HW controlled P-states

3.2.1.1 Enhanced Intel SpeedStep® Technology

Enhanced Intel SpeedStep® Technology enables OS to control and select P-state. For more information please refer to Section 2.3.3, "Enhanced Intel SpeedStep® Technology".

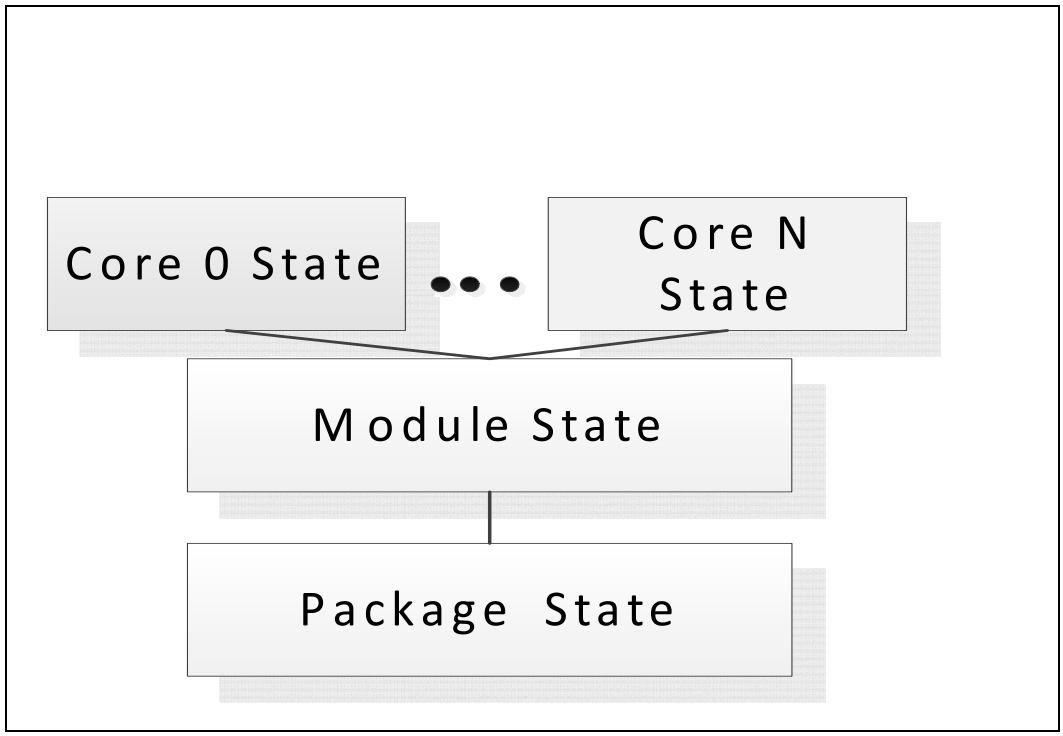
3.2.1.2 Intel® Speed Shift Technology

Intel® Speed Shift Technology is an energy efficient method of frequency control by the hardware rather than relying on OS control. For more details please refer to Section 2.3.4, "Intel® Speed Shift Technology".

3.2.2 Low-Power Idle States

When the processor is idle, low-power idle states (C-states) are used to save power. More power savings actions are taken for numerically higher C-states (deeper C-states). However, deeper C-states have longer exit and entry latencies. Resolution of C-states occur at the thread, processor IA core, and processor package level.

Figure 3-2. Idle Power Management Breakdown of the Processor IA Cores



Processor IA core C-states are automatically resolved by the processor. A transition to and from C0 state is required before entering any other C-state.

3.2.3 Requesting Low-Power Idle States

The primary software interfaces for requesting low-power idle states are through the MWAIT instruction with sub-state hints and the HLT instruction (for C1 and C1E). However, software may make C-state requests using the legacy method of I/O reads from the ACPI-defined processor clock control registers, referred to as P_LVLx. This method of requesting C-states provides legacy support for operating systems that initiate C-state transitions using I/O reads.

For legacy operating systems, P_LVLx I/O reads are converted within the processor to the equivalent MWAIT C-state request. Therefore, P_LVLx reads do not directly result in I/O reads to the system. The feature, known as I/O MWAIT redirection, should be enabled in the BIOS. The BIOS can write to the C-state range field of the PMG_IO_CAPTURE MSR to restrict the range of I/O addresses that are trapped and emulate MWAIT like functionality. Any P_LVLx reads outside of this range do not cause an I/O redirection to MWAIT(Cx) like request. They fall through like a normal I/O instruction.

When P_LVLx I/O instructions are used, MWAIT sub-states cannot be defined. The MWAIT sub-state is always zero if I/O MWAIT redirection is used. By default, P_LVLx I/O redirections enable the MWAIT 'break on EFLAGS.IF' feature that triggers a wake up on an interrupt, even if interrupts are masked by EFLAGS.IF.

3.2.4 Processor IA Core C-State Rules

The following are general rules for all processor IA core C-states, unless specified otherwise:

- A processor IA core transitions to C0 state when:
  - An interrupt occurs
  - There is an access to the monitored address if the state was entered using an MWAIT/Timed MWAIT instruction
  - The deadline corresponding to the Timed MWAIT instruction expires
- Any interrupt coming into the processor package may wake any processor IA core.
- A system reset re-initializes all processor IA cores.

Table 3-6. Core C-States

Core C-State	C-State Request Instruction	Description
C0	N/A	The normal operating state of a processor IA core where code is being executed
C1	MWAIT(C1)	AutoHalt - core execution stopped, autonomous clock gating (package in C0 state)
C1E	MWAIT(C1E)	Core C1 + lowest frequency and voltage operating point (package in C0 state)
C6	MWAIT(C6)	C6: Halt execution, flush core caches, flush core state, stop clock distribution, turn core voltage off
C6S	MWAIT(C6S)	C6S: C6 + allow entry to MC6
C7-C9	MWAIT(C7/C8/C9)	Same as C6S, shrink the LLC
C10	MWAIT(C10)	Same as C6S, LLC flushed. Enable S0ix

Core C-State Auto-Demotion

In general, deeper C-states, such as C6, have long latencies and have higher energy entry/exit costs. The resulting performance and energy penalties become significant when the entry/exit frequency of a deeper C-state is high. Therefore, incorrect or inefficient usage of deeper C-states have a negative impact on battery life and idle power. To increase residency and improve battery life and idle power in deeper C-states, the processor supports C-state auto-demotion.

C-State auto-demotion:

- C6 to C1/C1E

The decision to demote a processor IA core from C6 to C1/C1E is based on each processor IA core's immediate residency history. Upon each processor IA core C6 request, the processor IA core C-state is demoted to C1 until a sufficient amount of residency has been established. At that point, a processor IA core is allowed to go into C6. If the interrupt rate experienced on a processor IA core is high and the processor IA core is rarely in a deep C-state between such interrupts, the processor IA core can be demoted to a C1 state.

This feature is disabled by default. There are also Module C-states related to the core C states.

Table 3-7. Module C-States

Module C-State	Description
MC0	At least one core inC0
MC6	All cores in C6 (powered off) CPLL bypassed (Powered off) L2 flushed, L2 voltage = powered off

3.2.5 Package C-States

The processor supports C0, C2, C3, C6, C7, C8, C9, and C10 package states. The following is a summary of the general rules for package C-state entry. These apply to all package C-states, unless specified otherwise:

- A package C-state request is determined by the lowest numerical processor IA core C-state amongst all processor IA cores and also the module C-state.
- A package C-state is automatically resolved by the processor depending on the processor IA core idle power states and the status of the platform components.
  - Each processor IA core can be at a lower idle power state than the package if the platform does not grant the processor permission to enter a requested package C-state.
  - The platform may allow additional power savings to be realized in the processor.
  - For package C-states, the processor is not required to enter C0 before entering any other C-state.
  - Entry into a package C-state may be subject to auto-demotion – that is, the processor may keep the package in a deeper package C-state then requested by the operating system if the processor determines, using heuristics, that the deeper C-state results in better power/performance.

The processor exits a package C-state when a break event is detected. Depending on the type of break event, the processor does the following:

- If a processor IA core break event is received, the target (refer to Section 1.1.1for more information on target) processor IA core is activated and the break event message is forwarded to the target processor IA core.
  - If the break event is not masked, the target processor IA core enters the processor IA core C0 state and the processor enters package C0.
  - If the break event is masked, the processor attempts to re-enter its previous package state.
- If the break event was due to a memory access or snoop request,
  - But the platform did not request to keep the processor in a higher package C-state, the package returns to its previous C-state.
  - And the platform requests a higher power C-state, the memory access or snoop request is serviced and the package remains in the higher power C-state.

Figure 3-3. Package C-State Entry and Exit

The package level C states C3 through C10 are entered and exited through the C2R state.

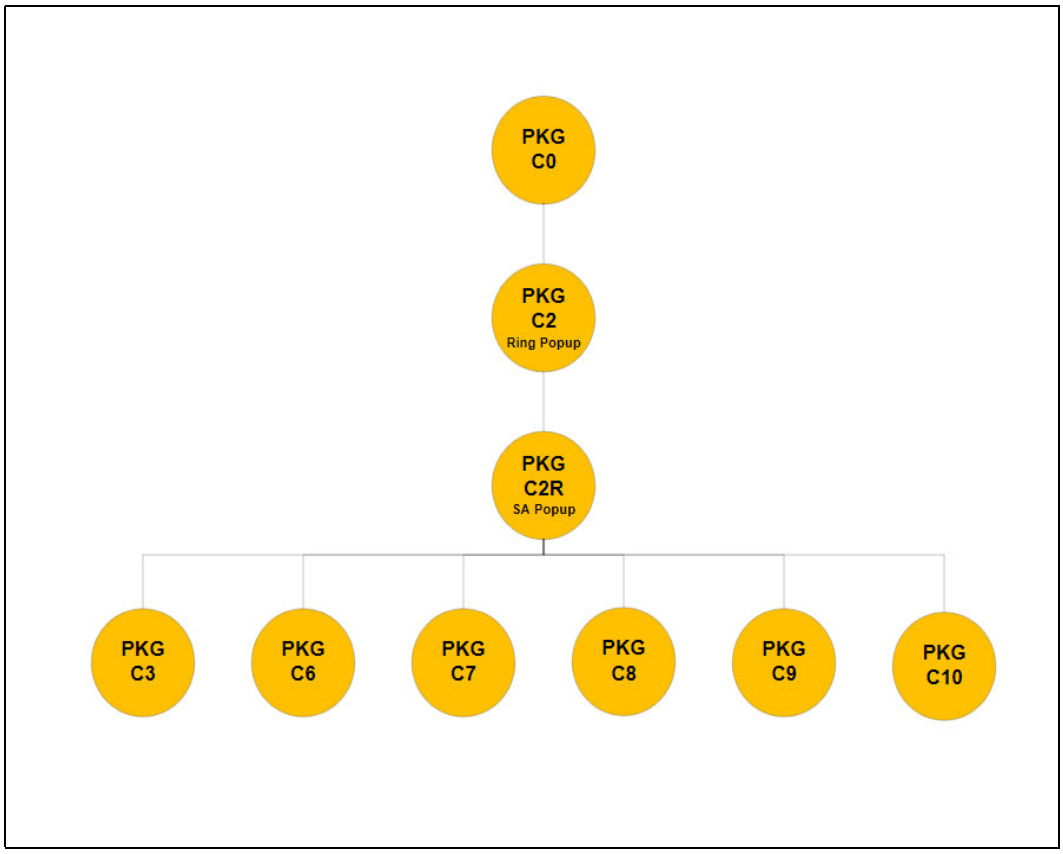


Table 3-8. Package C-States (Sheet 1 of 2)

Package C state	Description
C0	Processor active state
C2	Cannot be requested explicitly by the Software. All processor IA cores in C6 or deeper + Processor Graphic cores in RC6, memory path may be open. The processor will enter Package C2 when: • Transitioning from Package C0 to deep Package C state or from deep Package C state to Package C0. • All IA cores requested C6 or deeper + Processor Graphic cores in RC6 but there are constraints (LTR, programmed timer events in the near future and so forth) prevent entry to any state deeper than C2 state. • All IA cores requested C6 or deeper + Processor Graphic cores in RC6 but a device memory access request is received. Upon completion of all outstanding memory requests, the processor transitions back into a deeper package C-state.
C2R	A transitional package C-State
C3	All cores in C6 or deeper + Processor Graphics in RC6, LLC may be flushed and turned off, memory in self refresh, memory clock stopped. The processor will enter Package C3 when: • All IA cores in C6 or deeper + Processor Graphic cores in RC6. • The platform components/devices allows proper LTR for entering Package C3.

Table 3-8. Package C-States (Sheet 2 of 2)

Package C state	Description
C6	Package C3 + BCLK is off + IMVP VRs voltage reduction/PSx state is possible. The processor will enter Package C6 when: • All IA cores in C6 or deeper + Processor Graphic cores in RC6. • The platform components/devices allow proper LTR for entering Package C6.
C7	Package C6 + If all IA cores requested C7, LLC ways may be flushed until it is cleared. If the entire LLC is flushed, voltage will be removed from the LLC. The processor will enter Package C7 when: • All IA cores in C7 or deeper + Processor Graphic cores in RC6. • The platform components/devices allow proper LTR for entering Package C7.
C7S	Package C6 + If all IA cores requested C7S, LLC is flushed in a single step, voltage will be removed from the LLC. The processor will enter Package C7S when: • All IA cores in C7S or deeper + Processor Graphic cores in RC6. • The platform components/devices allow proper LTR for entering Package C7S.
C8	Package C7 + LLC should be flushed at once. The processor will enter Package C8 when: • All IA cores in C8 or deeper + Processor Graphic cores in RC6. • The platform components/devices allow proper LTR for entering Package C8.
C9	Package C8 + display in PSR or powered off + most Uncore voltages at 0V. IA, GT and SA voltages are reduced to 0V, while Vcc_{IO} stays on. The processor will enter Package C9 when: • All IA cores in C9 or deeper + Processor Graphic cores in RC6. • The platform components/devices allow proper LTR for entering Package C9.
C10	Package C9 + all VRs at PS4 or LPM + 38.4MHz clock off. The processor will enter Package C10 when: • All IA cores in C10 + Processor Graphic cores in RC6. • The platform components/devices allow proper LTR for entering Package C10.

Package C-State Auto-Demotion

The Processor may demote the Package C-state(s) to a shallower C-state(s), for example instead of going into package C10, it will demote to package C8 (and so on as required). The processor decision to demote the package C-state is based on the required C-states latencies, entry/exit energy/power and devices LTR.

Relevant S0ix

Intel Atom® x6000E Series processors, Intel® Pentium® and Celeron® N and J Series processors will support following S0ix variants. As the system goes deeper into S0ix, the overall functionality reduces, thereby, reducing the total power consumption. Longer S0ix residency gives better battery performance for a mobile/hand-held device.

Modern Standby is a relevant platform state in Windows. Other relevant S0ix states exist on other OS. On display time out the OS requests the processor to enter the package C10 state and platform devices at RTD3 (or disabled) in order to attain low power in idle. Relevant S0ix states require proper BIOS and OS configuration.

Table 3-9. Intel Atom® x6000E Series processors, Intel® Pentium® and Celeron® N and J Series processors S0ix Power Sub-States

S0ix Sub-States	Description
S0	All internal FIVR and external rails ON, All platform clocks running
S0i2.0	Only selected IP blocks are active. Vnn is margined down to 0.78V.
S0i3.0	Only wake event detection is active. 38.4MHz crystal clock and derived internal clocks are inactive. Vnn is margined down to 0.78V.

Note: The internal Vnn and V1p05 FIVRs cannot be measured externally.

During S0ix, VNN_BYP and V1P05_BYP, along with the corresponding devices are turned off. The PCH main power controller invokes a Save Restore mechanism to retain the states of these devices.

Dynamic LLC Sizing

When all processor IA cores request C7 or deeper C-state, internal heuristics dynamically flushes the LLC. Once the processor IA cores enter a deep C-state, depending on their MWAIT sub-state request, the LLC is either gradually flushed N-ways at a time or flushed all at once. Upon the processor IA cores exiting to C0 state, the LLC is gradually expanded based on internal heuristics.

3.3 PM Interface Signals

The following table provides the list of power control signals used by the package.

Table 3-10. Signal Descriptions (Sheet 1 of 3)

Name	Type	Description
PMC_ACPRESENT	I	**AC Present:** Used on mobile systems to determine presence of AC power or battery power.
PMC_BATLOW_N	I	**Battery Low:** An input from the battery to indicate that there is insufficient power to boot the system. Assertion will prevent wake from S3–S5 state. This signal can also be enabled to cause an SMI# when asserted. This signal must be tied high to the VCC_3P3A_DSW, which will be tied to VCC_3P3A on this platform. ***Note:*** Require external Pull-up to VCC_3P3A_DSW.
PMC_CORE_VID0	O	**PCH Core VID Bit 0:** May connect to discrete VR on platform and used to control the VCCIN_Aux rail (FIVR input) voltage. **In default mode this pin is driven high ('1')**
PMC_CORE_VID1	O	**PCH Core VID Bit 1:** May connect to discrete VR on platform and used to control the VCCIN_Aux rail (FIVR input) voltage. **In default mode this pin is driven high ('1')**
PMC_CPU_C10_GATE_N	O	Power Gate control for VCCIO, VCC_AGSH, VCCSTG and VCCSFR_OC during C10. When asserted, VCCIO can be 0V, however the power good indicators for these rails must remain asserted. It is recommended to switch off VCCIO when CPU is in C10 state. Platform should use CPU_C10_GATE_N to power off VCCIO. ***Note:*** VCCSTG is gated on-die by PMC_CPU_C10_GATE_N rather than by using an external power gate.
PMC_DRAM_RESET_N	O	**System Memory DRAM Reset:** Active low reset signal, controls reset to the memory subsystems (DDR4/LPDDR4). ***Note:*** An external Pull-up to the DRAM power plane is required.

Table 3-10. Signal Descriptions (Sheet 2 of 3)

Name	Type	Description
PMC_DSW_PWROK	IO	**DSW PWROK:** Power OK Indication for the VCC_3P3A_DSW voltage rail. This signal must be asserted no earlier than 10ms after the DSW power wells are valid.
PMC_PCH_PWROK	IO	**PCH Power OK:** When asserted, PMC_PCH_PWROK is an indication to the PCH that all of its core power rails have been stable for at least 5 ms. PMC_PCH_PWROK can be driven asynchronously. When PMC_PCH_PWROK is negated, the PCH asserts PMC_PLTRST_N. ***Note:*** PMC_PCH_PWROK must not glitch, even if PMC_RSMRST_N is low.
PMC_PLTRST_N	O	**Platform Reset:** The PCH asserts PMC_PLTRST_N to reset devices on the platform (such as SIO, LAN, processor, and so forth.). The PCH asserts PMC_PLTRST_N low in Sx states and when a cold, warm, or global reset occurs. The PCH de-asserts PMC_PLTRST_N upon exit from Sx states and the aforementioned resets. There is no guaranteed minimum assertion time for PMC_PLTRST_N. ***Note:*** PCI/PCIe* specification requires that the power rails associated with PCI/PCIe* (typically the 3.3V, 5V, and 12V core well rails) have been valid for 100 ms prior to PMC_PLTRST_N de-assertion. System designers must ensure the requirement is met on the platform.
PMC_VNN_CTRL	O	Signal used to control the optional VCC_BYP_VNN power rail when the platform is in S0ix or Sx modes.
PMC_V1P05_CTRL	O	Signal used to control the optional VCC_BYP_1P05 power rail when the platform is in S0ix or Sx modes.
PMC_PWRBTN_N	I	**Power Button:** Power button input signal. Used to wake the processor from power button press. The Power Button will cause SMI# or SCI to indicate a system request to go to a sleep state. If the system is already in a sleep state, this signal will cause a wake event. If PMC_PWRBTN_N is pressed for more than 4 seconds (default; timing is configurable), this will cause an unconditional transition (power button override) to the S5 state. Override will occur even if the system is in the S3-S4 states. This signal has an internal Pull-up resistor and has an internal 16 ms de-bounce on the input. ***Note:*** This signal is not able to cause a GPE, which is required to implement a 'Control method' power button as described in the ACPI specification.
PMC_RSMRST_N	I	**Resume Well Reset:** This signal is used for resetting the resume power plane logic. This signal must be asserted for at least 10ms after the suspend power wells are valid. When de-asserted, this signal is an indication that the suspend power wells are stable.
PMC_SLP_S0_N	O	**S0 Sleep Control:** When PCH is idle and processor is in C10 state, this pin will assert to indicate VR controller can go into a light load mode. This signal can also be connected to an external power management controller for other power management related optimizations.
PMC_SLP_S3_N	O	**S3 Sleep Control:** PMC_SLP_S3_N is for power plane control. This signal shuts off power to all non-critical systems when in S3 (Suspend To RAM), S4 (Suspend to Disk), or S5 (Soft Off) states.
PMC_SLP_S4_N	O	**S4 Sleep Control:** PMC_SLP_S4_N is for power plane control. This signal shuts power to all non-critical systems when in the S4 (Suspend to Disk) or S5 (Soft Off) state.
PMC_SLP_S5_N	O	**S5 Sleep Control:** PMC_SLP_S5_N is for power plane control. This signal is used to shut power off to all non-critical systems when in the S5 (Soft Off) states.
PMC_SLP_SUS_N	O	PMC_SLP_SUS_N is used to turn on rest of PRIM rails after 3.3V PRIM is ready, PMC_DSW_PWROK is the powergood indication to get PMC_SLP_SUS_N up.
PMC_SUSCLK	O	**Suspend Clock:** This clock is a digitally buffered version of the RTC clock.

Table 3-10. Signal Descriptions (Sheet 3 of 3)

Name	Type	Description
PMC_SYS_PWROK	I	**System Power OK:** This generic power good input to the PCH is driven and utilized in a platform-specific manner. While PMC_PCH_PWROK always indicates that the core wells of the PCH are stable, PMC_SYS_PWROK is used to inform the PCH that power is stable to other required system component(s) and the system is ready to start the exit from reset. (de-asserts PMC_PLTRST_N to the processor).
PMC_SYS_RESET_N	I	**System Reset:** Reset button input signal to reset the processor. This pin forces an internal reset after being debounced. ***Note:*** This signal should not be allowed to float while PMC_SLP_S3_N is de-asserted.
PMC_VRALERT_N	I	**VR Alert:** ICC Max. throttling indicator from the PCH voltage regulators. PMC_VRALERT_N pin allows the VR to force throttling to prevent an over current shutdown.
PMC_WAKE_N	I/O	**PCI Express* Wake Event in Sx:** Input Pin in Sx. Sideband wake signal on PCI Express* asserted by components requesting wake up. ***Note:*** This is Output pin during S0IX states hence this pin can not be used to wake up the system during S0IX states. ***Note:*** External Pull-up required.
PMC_ALERT_N	I	PD controller's USB-C interrupt request is presented to PMC as processor USB-C Mux Manager, through PMC_ALERT_N pin assertion.
VCCST_OVERRIDE	O	VCCST_OVERRIDE is part of the control signal for VCC_IN_ST. It is used for Sx state entry and during host partition reset with power cycle. The PCH will gate the RTC clock to the CPU when this signal goes low.
THRMTRIP_N	O	**Thermal Trip:** Asserted during a catastrophic thermal event. Platform design should restart or shut down the voltage rails after this event. For platform using discrete VR (voltage regulator) power delivery solution, there is additional platform logic required to initiate VR shut down on the platform when this signal is asserted.
PCHHOT_N	OD	PCHHOT_N indicates that it has exceeded some temperature limit set by BIOS. The temperature limit (programmed into the PHL register) is compared to the present temperature. If the present temperature is greater than the PHL value then the pin is asserted.

3.4 Processor Voltage Rails

3.4.1 Fully Integrated Voltage Regulator (FIVR)

The processor integrates multiple voltage rails in order to reduce BOM costs for the platform, and to enable additional voltage level features the processor can take advantage of.

There are 2 FIVRs integrated on the PCH, Vnn and V1p05 which is sourced from VCCIN_Aux. VCCIN_Aux also sourced the VccSA rail in a compute die. In addition to VCCSA FIVR, compute die integrates 4 additional FIVRs to source VCCCORE, VCCL2, VCCGT and VCCRING, which derives the respective voltages from VCCIN VR on platform. Each FIVR is able to control a specific voltage rail.

3.4.2 Main Platform Voltage Regulators

In the table below are the main platform voltage rails that are regulated and controlled on the platform.

Table 3-11. Platform Voltage Rails

Rail	Voltage	Description
VCCIN	0V - 2.0V	Voltage Rail to power the compute cores, graphics, and cache. This is an SVID controlled voltage rail.
VCCIN_Aux	1.65 or 1.8V - Active Off - Idle States	PCH FIVR Input power supply
VDDQ	1.2V or 1.1V	Powers the memory and the VCC_IN_SFR inside processor
VCC_3P3A	3.3V	Primary 3.3V supply
VCC_1P8A	1.8V	Primary 1.8V supply
VCC_BYP_VNN (Optional)	1.05V or 0.78V	Optional bypass rail for PCH Prime Core Well. Configurable through VID depending on power state.
VCC_BYP_1P05 (Optional)	1.05V	Optional bypass rail for PCH Primary Well (PMC_VIP05_CTRL always configured to 0).
VCCIO	1.05V	I/O power supply
Note: Leakage voltage on the VCC_1P8A rail is expected when VCC_3P3A is powered and VCC_1P8A is un-powered. VCC_3P3A should not be powered while VCC_1P8A is un-powered for more than 518400 seconds, for the entire duration of the product's operating lifetime in order to meet Intel's goals.		

3.4.3 Additional Voltage Rail Signals

There are additional voltage rail pins for routing power between parts of the processor and the platform listed in the table below.

Table 3-12. Additional Voltage Rail Signals

Rail	Voltage	Source Rail	Description
VCC_IN_SFR	1.05V	FIVR	Output from processor for SFR FIVR Rail
VCC_IN_ST	1.05V	FIVR	Output from processor VCC_IN_ST FIVR Rail
VCC_OUT_FET_1P05A	1.05V	FIVR	1.05 FIVR Out for feedback to processor (FUSE)
VCC_OUT_FIVR_1P05A	1.05V	FIVR	FIVR 1.05 out from processor used for decoupling
VCC_IN_STG	1.05V	FIVR	VCC_IN_STG and VCC_OUT_STG route the STG FIVR power in and out to different parts of the processor, for on board decoupling.
VCC_OUT_STG	1.05V	FIVR	VCC_IN_STG and VCC_OUT_STG route the STG FIVR power in and out to different parts of the processor, for on board decoupling.
VCC_OUT_1P05A	1.05V	FIVR	FIVR output to platform to supply VCC_IN_FUSE_V1P05A processor rail through VCC1P05_OUT_PCH power plane.
VCCA_CLKLDO_1P8	1.8V	VCC_1P8V	Clocks rail.
VCC_OUT_1P24A	1.24V	VCC_1P8A	1.24V rail for DPHY
VCC_PGPPR	1.8V / 3.3V	VCC_1P8A/ VCC_3P3A	VCC_PGPPR is sourced from either VCC_3P3A or VCC_1P8A rail and this rail is input supply for the GP_R GPIOs.
VCC_AGSH	1.8V	VCC_1P8A	1.8A rail. This rail is turned off when either PMC_CPU_C10_GATE_N or PMC_SLP_S3_N is asserted (LOW). Thus, this rail is off in package C10 state as well as S3 - S5 states.
VCC_RTC_3P3	3V	VCC_RTC_3P3	RTC Supply (2.0-3.3)
VCC_RTC_EXT	1.5V	VCC_3P3A	1.5V RTC EXT Well. VCC_RTC_EXT pin can be used to probe the internal RTC well voltage.
VCCSFR_OC	(VDDQ voltage)	VDDQ	VDDQ gated rail going back to Intel Atom® x6000E Series processors, Intel® Pentium® and Celeron® N and J Series processors as VCCSFR_OC

3.4.4 VCCIN_Aux

From the platform perspective, the FIVRs require an input rail to generate the internal voltage rails. This rail is referred to as VCCIN_Aux. For the PCH, the input regulator must be able to support at least 1.8V. During the deep S0ix states, the input rail to the FIVRs can be disabled. This will be done by driving the CORE_VID values to `00. VCCIN_Aux powergood during initial reset is tied into the PMC_RSMRST_N signal, requiring that the FIVR input voltage rail is stable in the same window as the other PMC_SLP_SUS_N rails. Internal FIVRs will generate Vnn, V1P05 rails.

Note: Leakage from VCCIN_AUX is expected behavior when CORE_VID[1:0]=00; this leakage voltage may be as high as 1.15 V during Sx and S0ix states.

3.4.5 External Bypass Rails (Vnn and V1p05)

The V1p05 & Vnn rails can also have an input from a separate external voltage rail. These rails are always on and must come up after the V1p8A rail has been brought up. Note that there is no feedback that this rail is valid.

Leakage from the VCC_BYP_VNN power rail may back drive the external bypass voltage regulator (VR) when it is not in use, and VR output may float up as high as 1.125 V. This is an expected behavior. Intel recommends selecting a bypass VR with an Over Voltage Protection (OVP) threshold that is above 1.125 V for all VCC_BYP_VNN voltage settings to avoid false VR shutdown.

3.5 Voltage Rail Electrical Specifications

The processor DC specifications in this section are defined at the processor signal pins, unless noted otherwise. Icc_max specifications are estimates on the currunt consumption by the processor pins only. Other additional devices that consume current on the platform need to be considered separately.

- The *Voltage and Current Specifications* section lists the DC specifications for the processor and are valid only while meeting specifications for junction temperature, clock frequency, and input voltages. Read all notes associated with each parameter.
- AC tolerances for all DC rails include dynamic load currents at switching frequencies up to 1 MHz.

3.5.1 Processor Power Rails DC Specifications

3.5.1.1 Vcc_{IN} DC Specifications

Table 3-13. Processor Vcc_{IN} Active and Idle Mode DC Voltage and Current Specifications (Sheet 1 of 2)

Symbol	Parameter	Remark	Min	Typ	Max	Unit	Note
Operating Voltage	Voltage Range for Processor Operating Mode	All	0	—	2.0	V	1,2,6,10
IccMAX	Maximum VccIN Icc	SKUs 6-7, 9-11	—	—	26	A	3,5,6,9
		SKUs 1-2			28		
		SKUs 3-5, 8, & 12	—	—	18	A	3,5,6,9
Icc_{TDP}	Maximum VccIN Thermal Design Current	SKUs 1-2, 6-7, 9-11	—	—	14.5	A	—
		SKUs 3-5, 8, & 12	—	—	10.3	A	—
TOB_{VCC}	Voltage Tolerance	PS0, PS1	—	—	±20	mV	2, 5, 7
		PS2, PS3	—	—	±35		—
Ripple	Ripple Tolerance	PS0, PS1	—	—	±15	mV	2, 5, 7
		PS2, PS3	—	—	±30		—
DC_LL	Loadline slope within the VR regulation loop capability (0-1kHz)	Loadline for board routing	—	—	3.5	mΩ	8,11,16
		Loadline for LPVRTT test	—	—	8.6	mΩ	12,16
AC_LL3	Loadline slope in response to dynamic load increase events (<1MHz)	Loadline for board routing	—	—	4.0	mΩ	8,11,16
		Loadline for LPVRTT test	—	—	8.8	mΩ	12,16
Slew Rate	SetVID_Slow slew rate. Default is 1/4 of SetVID_Fast	SetVID_Fast	10	48	60	mV/µs	17, 18
		SetVID_Slow	—	12	30	mV/µs	17
T_OVS MAX	Max Overshoot time **TDP/virus mode** (IccMax)	—	—	—	500	µs	—
V_OVS MAX	Max Overshoot Voltage at **TDP/virus mode** (IccMax)	—	—	—	200	mV	—
T_OVS MAX Apps	Max Overshoot time **TDP/virus mode** (IccMax_Apps)	—	—	—	500	µs	—
V_OVS MAX Apps	Max Overshoot Voltage at **TDP/virus mode** (IccMax_Apps)	—	—	—	200	mV	—

Table 3-13. Processor Vcc_{IN} Active and Idle Mode DC Voltage and Current Specifications (Sheet 2 of 2)

Symbol	Parameter	Remark	Min	Typ	Max	Unit	Note
di	Icc_{MAX} transient	—	—	—	28	A	14
dt	Duration for di step	—	—	—	150	ns	15
Load Step	Max Load Step	—	—	—	28	A	—

Notes:

1. Each processor is programmed with a maximum valid voltage identification value (VID) that is set at manufacturing and cannot be altered. Individual maximum VID values are calibrated during manufacturing such that two processors at the same frequency may have different settings within the VID range. Note that this differs from the VID employed by the processor during a power management event (Adaptive Thermal Monitor, Enhanced Intel SpeedStep Technology, or low-power states).
2. The voltage specification requirements are measured across $Vcc_{\_SENSE}$ and $Vss_{\_SENSE}$ as near as possible to the processor with an oscilloscope set to 100MHz bandwidth, 1.5pF maximum probe capacitance, and 1MΩ minimum impedance. The maximum length of ground wire on the probe should be less than 5mm. Ensure external noise from the system is not coupled into the oscilloscope probe.
3. IccMAX is a peak current for a VCC_{IN} VR. Processor Vcc_{IN} VR to be designed to electrically support this current.
4. Processor Vcc_{IN} VR to be designed to thermally support this current indefinitely.
5. Long term reliability cannot be assured if tolerance, ripple, and core noise parameters are violated.
6. Long term reliability cannot be assured in conditions above or below Max/Min functional limits.
7. PSx refers to the voltage regulator power state as set by the SVID protocol.
8. LL measured at sense points inclusive the package.
9. Typ column represents Icc_{MAX} for commercial application it is NOT a specification - it's a characterization of limited samples using limited set of benchmarks that can be exceeded.
10. Operating voltage range in steady state.
11. LL spec values should not be exceeded. If exceeded, power, performance and reliability penalty are expected.
12. Load Line (AC/DC) should be measured by the VRTT tool and programmed accordingly via the BIOS Load Line override setup options. AC/DC Load Line BIOS programming directly affects operating voltages (AC) and power measurements (DC). A superior board design with a shallower AC Load Line can improve on power, performance and thermals compared to boards designed for POR impedance.
13. Overshoot with max voltage of 2.2V is allowed if it sustained for less then 500us.
14. For VR design testing, the recommended initial current is 24A with 28A of di.
15. The time durations given here are for the VR design only. This rise time is not critical to test the Over Current Protection (OCP) feature.
16. Decoupling recommendations and associated VR bandwidth requirements are shown in the Power Integrity Chapter in PDG.
17. For LPVRTT testing, the recommended slew rate setting can be the worst-case scenario which is 60mV/us for SetVID_Fast and 30mV/us for SetVID_Slow.
18. OCP must sustain max inrush current > (total capacitance * derating) *(dV/dt). Please note that dV/dt is a max slew rate which may cause large inrush current that might have invalid OCP triggering or exceed the maximum drain current of the power MOSFET(s). Please check your design VR OCP and MOSFET max current capability.

3.5.1.2 VCC_1P8A DC Specifications

Table 3-14. Processor VCC_1P8A Supply DC Voltage and Current Specifications

Symbol	Parameter	Min	Typ	Max	Unit	Notes[1]
Vcc_{1p8A}	Package voltage (DC + AC specification)	—	1.8	—	V	2
Icc_{MAX_1p8A}	Max Current for Vcc_{1p8A} Rail	—	—	0.649	A	
Icc_{idle}	Sx Icc Idle Current	—	—	6.4	mA	
TOB Vcc_{1p8A}	Vcc_{1p8A} Tolerance	AC+DC:± 5%			%	2
Ripple	Max Ripple	—	—	90	mV	

Notes:

1. Long term reliability cannot be assured in conditions above or below Max/Min functional limits.
2. The voltage specification requirements are measured on package pins as near as possible to the processor with an oscilloscope set to 100-MHz bandwidth, 1.5 pF maximum probe capacitance, and 1 MΩ minimum impedance. The maximum length of ground wire on the probe should be less than 5 mm. Ensure external noise from the system is not coupled into the oscilloscope probe.

3.5.1.3 VccIN_AUX DC Specifications

Table 3-15. VccIN_AUX Supply DC Voltage and Current Specifications (Sheet 1 of 2)

Symbol	Parameter	Remark	Min	Typ	Max	Unit	Note
Vcc_{in_AUX}	Voltage Range of Vcc_{in_AUX} Operating Mode		0	—	1.8	V	2
Icc_{MAX}	Maximum VccIN_AUX Icc	SKUs 1-2, 6-7, 9-11	0	—	15	A	7
		SKUs 3-5, 8, & 12	0	—	13.5	A	7
Icc_{idle}	Sx Icc Idle Current		0	—	201	mA	—
Icc_{TDP}	Maximum VccIN_AUX Thermal Design Current	SKUs 1-2, 6-7, 9-11	—	—	6	A	—
		SKUs 3-5, 8, & 12	—	—	5	A	—
Duty Cycle	Duty cycle of Icc transient from 0% to 60%	Duty cycle_LOW	—	—	50	%	—
	Duty cycle of Icc transient from 40% to 100%	Duty cycle_HIGH	—	—	20	%	—
TOB_{VCC}	Voltage Tolerance Budget		—	—	AC+DC: -10/+5	%	2,3
VOS	Overshoot Voltage		—	—	2.13	V	4
TVOS	Overshoot Time		—	—	500	us	4
DC_LL	DC Loadline (0-1kHz)	Loadline for board routing	—	—	7.0	mΩ	8,10
		Loadline for LPVRTT test	—	—	8.7	mΩ	9,10
AC_LL	AC Loadline 3 (<1MHz)	Loadline for board routing	—	—	7.2	mΩ	8,10
		Loadline for LPVRTT test	—	—	8.8	mΩ	9,10

3.5.1.3 VccIN_AUX DC Specifications

Table 3-15. VccIN_AUX Supply DC Voltage and Current Specifications (Sheet 2 of 2)

Symbol	Parameter	Remark	Min	Typ	Max	Unit	Note

Notes:

1. Long term reliability cannot be assured in conditions above or below Max/Min functional limits.
2. The voltage specification requirements are measured on package pins as near as possible to the processor with an oscilloscope set to 100-MHz bandwidth, 1.5 pF maximum probe capacitance, and 1 MΩ minimum impedance. The maximum length of ground wire on the probe should be less than 5 mm. Ensure external noise from the system is not coupled into the oscilloscope probe.
3. Voltage Tolerance budget values Includes ripples
4. Overshoot with max voltage of 2.13V is allowed if it sustained for less then 500us.
5. This rail can be connect to 1.65V
6. Vcc_{IN_AUX} is having few point of voltage define by VID.
7. The ICCMAX values combine power pins that feed the compute die and the PCH die in the processor.
8. LL measured at sense points inclusive the package.
9. Load Line (AC/DC) should be measured by the VRTT tool and programmed accordingly via the BIOS Load Line override setup options. AC/DC Load Line BIOS programming directly affects operating voltages (AC) and power measurements (DC). A superior board design with a shallower AC Load Line can improve on power, performance and thermals compared to boards designed for POR impedance.
10. Decoupling recommendations and associated VR bandwidth requirements are shown in the Power Integrity Chapter in PDG.

3.5.1.4 V_{DDQ} DC Specifications

Table 3-16. Memory Controller (V_{DDQ}) Supply DC Voltage and Current Specifications

Symbol	Parameter	Min	Typ	Max	Unit	Note
$V_{DDQ\ (LPDDR4/x)}$	Processor I/O supply voltage for LPDDR4/x	1.05	1.1	1.15	V	2,3,4
$V_{DDQ\ (DDR4)}$	Processor I/O supply voltage for DDR4	0.95	1.2	1.25	V	2,3,4
TOB_{VDDQ}	VDDQ Tolerance	AC+DC:± 5%			%	2,3
$Icc_{MAX\_VDDQ\ (LPDDR4/x)}$	Max Current for V_{DDQ} Rail (LPDDR4/x)	—	—	3.5	A	1
$Icc_{MAX\_VDDQ\ (DDR4)}$	Max Current for V_{DDQ} Rail (DDR4)	—	—	3.5	A	

Notes:

1. The current supplied to the DRAM is not included in this specification.
2. Includes AC and DC error, where the AC noise is bandwidth limited to under 100 MHz, measured on package pins.
3. No requirement on the breakdown of AC versus DC noise.
4. The voltage specification requirements are measured on package pins as near as possible to the processor with an oscilloscope set to 100-MHz bandwidth, 1.5 pF maximum probe capacitance, and 1 MO minimum impedance. The maximum length of ground wire on the probe should be less than 5 mm. Ensure external noise from the system is not coupled into the oscilloscope probe.

3.5.1.5 VCCIO DC Specifications

Table 3-17. VCCIO Supply DC Voltage and Current Specifications

Symbol	Parameter	Min	Typ	Max	Unit	Notes
VCCIO	Voltage Range in Operating Mode	1.0	1.05	1.1	V	
Icc_{MAX_VCCIO}	Max Current for VCCIO Rail	—	—	5.5	A	
di	Icc_{MAX} transient	—	—	2.0	A	
dt	Duration for di step			1.0	us	
TOB_{VCCIO}	Voltage Tolerance	AC+DC+Ripple: ±50			mV	

Note: Long term reliability cannot be assured in conditions above or below Max/Min functional limits.

3.5.1.6 Additional Rails DC Characteristics

Table 3-18. Additional Rails Estimated I_{cc}[3] (Sheet 1 of 2)

Voltage Rail	Voltage (V)			Iccmax Current[2] (A)
	Min	Typ	Max	
VCC_BYP_VNN	0.997	1.05	1.102	0.5
VCC_BYP_1P05	0.997	1.05	1.102	0.5
VCC_PGPPR[4]	1.710	1.8	1.890	0.018
VCC_AGSH	1.710	1.8	1.890	0.2
VCCA_CLKLDO_1P8	1.710	1.8	1.890	0.163
VCCPFUSE_1P8[5]	1.710	1.8	1.890	0.089
VCCPFUSE_3P3[6]	3.135	3.3	3.465	0.025
VCC_3P3A[9]	3.135	3.3	3.465	0.321

Table 3-18. Additional Rails Estimated I_{CC}[3] (Sheet 2 of 2)

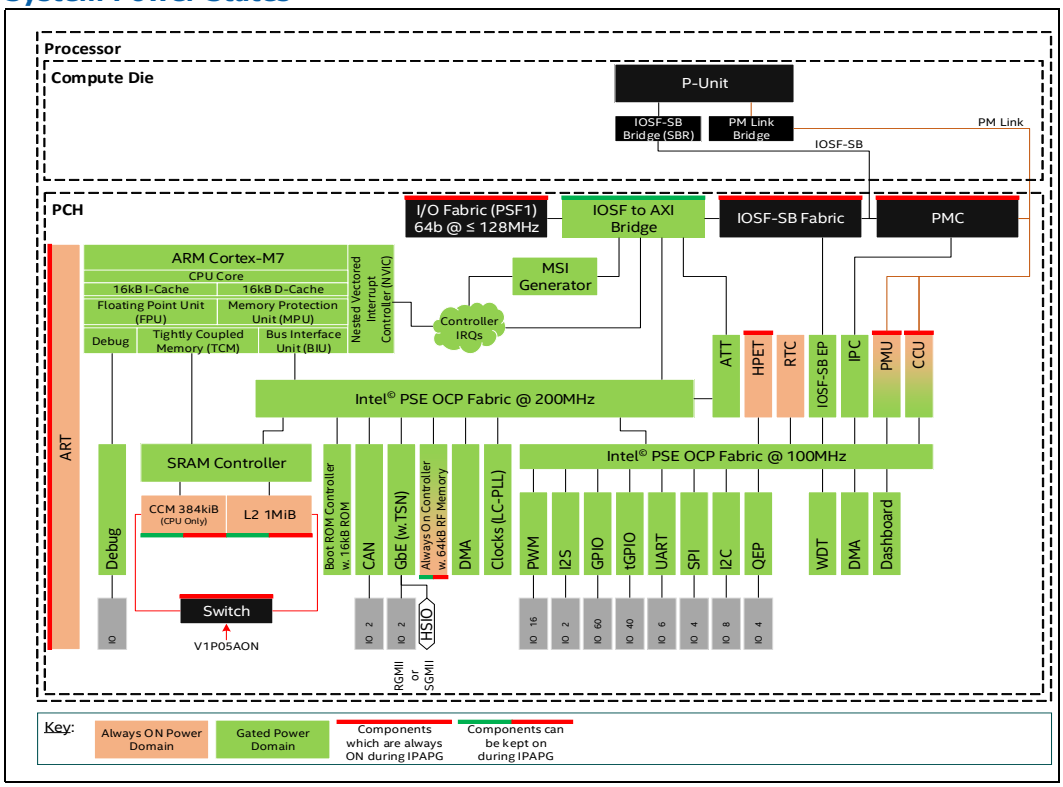
Voltage Rail	Voltage (V)			Iccmax Current[2] (A)
	Min	Typ	Max	
VCC_RTC_3P3[1]	2.000	3.0	3.300	0.0012
VCC_3P3A_DSW	3.135	3.3	3.465	0.008
VCC_IN_ST	0.95	1.05	1.07	0.6
VCCSTG	0.95	1.05	1.07	0.12
VCC_IN_SFR	0.95	1.05	1.07	0.1
VCCSFR_OC[7]	1.05/0.95	1.1/1.2	1.15/1.25	0.1
VCC_OUT_FET_1P05A	0.96	1.05	1.07	0.6
VCC_OUT_1P24A	1.16	1.24	1.31	0.2

Notes:
1. The VCC rail ICCMAX is 10uA while the system is in a mechanical off (G3) state at room temperature. This data is taken at 3.0V.
2. Iccmax estimates assumes 110 °C.
3. The Iccmax value is a steady state current that can happen after respective power ok has asserted (or reset signal has de-asserted).
4. The tolerance voltage for VCC_PGPPR at 3.3V is ±5%.
5. Merged to VCC_1P8A.
6. Merged to VCC_3P3A.
7. Derived from VDDQ.
8. The accuracy of VCC_IN_ST/VCCSTG which is driven from FIVR is +/-20mV.
9. The VCC rail IccMAX is a value after merged with VCCPFUSE_3P3.

3.6 Intel® Programmable Services Engine (Intel® PSE) Power Management

Intel® Programmable Services Engine (PSE) has two power domain, which are the “Gated Wrapper” and “Ungated Wrapper”. Additionally the SRAM banks can be power gated or put into retention (low voltage) mode. Intel® PSE can be fully functional when the processor is in S0ix state or can work at reduced clock when it is in the Sx state including S4 and S5 state.

Figure 3-4. System Power States



3.6.1 Basic PSE Device Power Management Concepts

The following are some of the key terminology that explains the basic understanding of Intel Atom® x6000E Series processors, Intel® Pentium® and Celeron® N and J Series processors power management concept:

3.6.1.1 Dx State

Dx is a host managed device power state, where the host initiates Dx transition by operating PCI PMCSR register. PSE supports D0 and D3 state. As the peripheral owned by host also lays in the PSE power domain, PSE FW will decide the actual power state transition based on other peripheral state as well as ARM core idle state.

Note that IPC PCI function represents the PSE controller as a whole to host (though PSE exposes other PCI functions as well). If there is a D3 entry request from Host IPC, PSE FW puts the PSE into IPAPG state - if ARM and other IPs are in idle state and all other host owned PCI function in PSE are in D3.

3.6.1.2 D0ix State

D0ix is a fine grain power management within D0 state. The host sets the D0i3 bit in D0i3 control register of the corresponding PCI function to intimate PSE to a low power state. PSE determines FW to enter a low power state depending on other IO state - if all the PCI functions have not set D0i3 bit, then PSE can only enter D0i0 with block level CG. D0ix typically has a very small resume latency (< 1ms) compared to D3 latency.

PSE D0ix states are managed by PSE FW (Zephyr).

The following table shows the main platform voltage rails that are regulated and controlled on the platform.

Table 3-19. PSE D0ix states

Power state	PM features enabled
D0	Fully functional state
D0i0.BLOCKCG	Block level clock gating of individual blocks that are idle while the rest of OSE is active. The ARM core also is internally clock gated when the WFI instruction is executed.
D0i1 (TCG)	Trunk level Clock gating of all OSE functional clocks.
D0i2 (SRAM in retention) + IPAPG	Trunk level clock gating, SRAMs in retention, Power gating of Logic, RF (Cache and other IP RF) and ROM
D0i3 (SRAM in PG) + IPAPG	Trunk level clock gating, SRAMs are power gated, Power gating of Logic, RF (Cache and other IP RF) and ROM

3.6.1.3 IPIAPG

IP inaccessible PG where IP is no longer accessible from the IOSF Primary/sideband fabrics during the PG state. PSE will come out of IPIAPG state on cold boot or after a cold reset.

3.6.1.4 IPAPG

IP Accessible PG where the IP remains accessible through the IOSF fabric interfaces. This state can be reached when IP enter D3, D0i2 or D0i3 state.

3.6.2 PMU and Power rails

PMU and CCU: The Power Management Unit (PMU) is a part of PSE which contains the logic for PSE supported power management features, host wakeup, record all the wakeup interrupts received during clock gated state etc. It also interacts with the Clock and reset Control Unit (CCU) which is also a part of PSE for clock, reset, clock gating control etc. PMU sequences the SRAM power gate enable.

- VNN Power Rail: This power rail supplies power to IOSF fabric and any access to IOSF such as DMA to/from DRAM, IPC communication to host etc. requires VNN to be asserted by PSE FW before start any above operation. VNN AON Power Rail: Always ON power rail and PSE is primarily powered from VNN AON. The power rail can be put into reduced voltage for system to enter S0i3 state, but still PSE will be operation at reduced clock frequency.
- V1P05AON power rail: The SRAMs are powered by the V1P05AON rail and each individual SRAM bank can power gated separately or can be kept ON or in retention mode even when the logic is power gated.
- V1P05IS power rail: PLL is powered from V1P05IS rail.

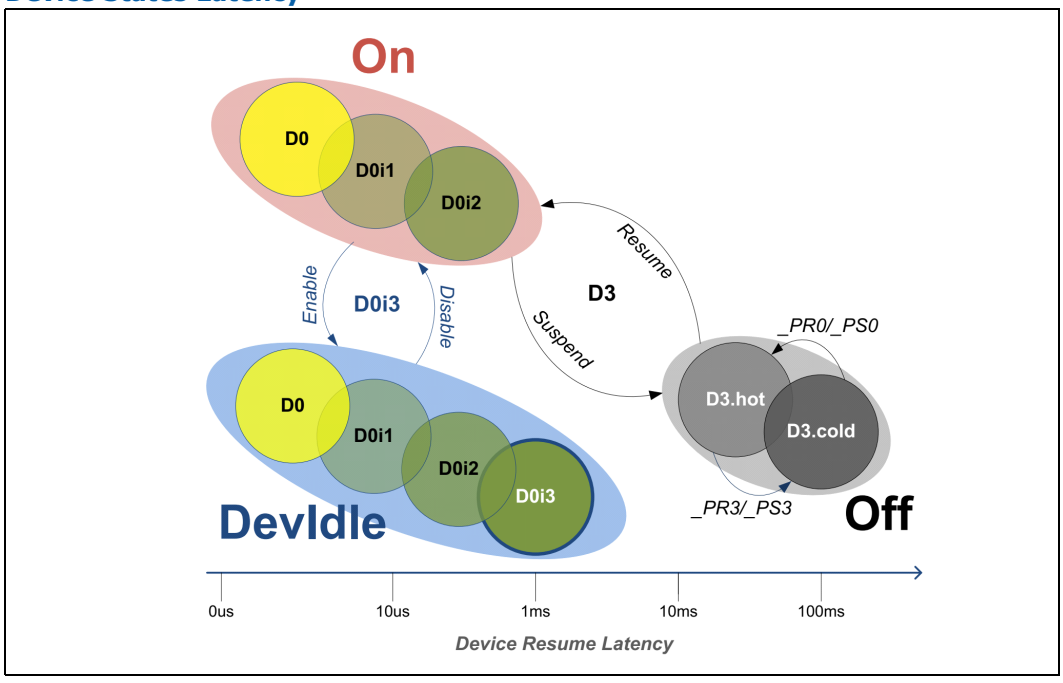
3.6.3 Ungated Wrapper

The ungated wrapper will be remain ON (powered) while PSE subsystem is in PG(IPAPG). The logic which need to wake up PSE from PG such as PMU/CCU, HPET timer etc. are supposed to be in ungated wrapper (see in the Figure 3-4 block diagram marked in orange color). Note that during cold reset, RF (register files), PCI config space, AON RF memory etc. will be turned OFF and then turned ON back (power cycle). CCM and L2 SRAM memory can be keep in retention mode or can be power gated. All SRAM banks in PSE are powered from V1P05AON power rail. All memory including RF memory will power cycle during cold reset.

3.6.4 Gated Wrapper

Gated wrapper will be turned OFF during power gate and will turned ON back by PMC on request from PMU on any PG exit wake up source (HPET interrupt, GPIO pins or downstream IOSF transaction due to host access any of MMIO registers). The gated wrapper of PSE constitute majority of functionality such as all high speed and low speed peripherals (except HPET timer), ARM core, part of PMU/CCU, IOSF interface etc (shown in green box in the Figure 3-4 block diagram). PSE FW must save context of all the peripherals such as peripheral controller register values, CPU registers, CCM and L2 SRAM contents (if not keep in retention mode) to DRAM memory. Please note that the SRAM banks (both CCM and L2SRAM) can be keep in retention mode in IPAPG state.

Figure 3-5. Device States Latency



3.6.4.1 Wake PSE from D0i1, D0i2 and D0i3 state

HPET timer interrupt, GPIO or any downstream transaction such as peripheral register access by Host (for host owned devices), or access to IPC register by Host can bring PSE out of D0i1, D0i2 or D0i3 state. For D0i0, any interrupt can be wakeup source.

3.6.4.2 Wake Host from S0ix State

Host can be waken up from S0ix state by assert VNN request. IPC driver and DMA driver will take care of VNN request internally before send IPC request to host or initiate DMA to host DRAM memory.

3.6.4.3 Wake Host from Sx State

Host can be wake from Sx state by asserting by setting PME_SET bit in the OOB_PME register corresponding to the PCI function from which the wake was received. Alternatively PSE FW can send "Assert_PME" message to PMC as sideband message and can wake up Host from Sx state.

3.7 SMI#/SCI Generation

Upon any enabled SMI event taking place while the End of SMI (EOS) bit is set, the PCH will clear the EOS bit and assert SMI to the processor, which will cause it to enter SMM space. SMI assertion is performed using a Virtual Legacy Wire (VLW) message.

Once the SMI VLW has been delivered, the PCH takes no action on behalf of active SMI events until Host software sets the End of SMI (EOS) bit. At that point, if any SMI events are still active, the PCH will send another SMI VLW message.

The SCI is a level-mode interrupt that is typically handled by an ACPI-aware operating system. In non-APIC systems (which is the default), the SCI IRQ is routed to one of the 8259 interrupts (IRQ 9, 10, or 11). The 8259 interrupt controller must be programmed to level mode for that interrupt.

In systems using the APIC, the SCI can be routed to interrupts 9, 10, 11, 20, 21, 22, or 23. The interrupt polarity changes depending on whether it is on an interrupt shareable with a PIRQ or not. The interrupt remains asserted until all SCI sources are removed.

The table below shows which events can cause an SMI and SCI.

Note: Some events can be programmed to cause either an SMI or SCI. The usage of the event for SCI (instead of SMI) is typically associated with an ACPI-based system. Each SMI or SCI source has a corresponding enable and status bit.

Table 3-20. Causes of SMI and SCI (Sheet 1 of 3)

Cause	SCI	SMI	Additional Enables (Note 1)	Where Reported
PME#	Yes	Yes	PME_EN=1	PME_STS
PME_B0 (Internal, Bus 0, PME Capable Agents)	Yes	Yes	PME_B0_EN=1	PME_B0_STS
PCI Express* PME Messages	Yes	Yes	PCI_EXP_EN=1 (Not enabled for SMI)	PCI_EXP_STS
PCI Express* Hot-Plug Message	Yes	Yes	HOT_PLUG_EN=1 (Not enabled for SMI)	HOT_PLUG_STS
Power Button Press	Yes	Yes	PWRBTN_EN=1	PWRBTN_STS
Power Button Override (Note 6)	Yes	No	None	PWRBTNOR_STS

Table 3-20. Causes of SMI and SCI (Sheet 2 of 3)

Cause	SCI	SMI	Additional Enables (Note 1)	Where Reported
RTC Alarm	Yes	Yes	RTC_EN=1	RTC_STS
ACPI Timer overflow (2.34 seconds)	Yes	Yes	TMROF_EN=1	TMROF_STS
GPIO	Yes	Yes	Refer to note 8	
TCO SCI message from processor	Yes	No	None	CPUSCI_STS
TCO SCI Logic	Yes	No	TCOSCI_EN=1	TCOSCI_STS
TCO SMI Logic	No	Yes	TCO_EN=1	TCO_STS
TCO SMI – Year 2000 Rollover	No	Yes	None	NEWCENTURY_STS
TCO SMI – TCO TIMEROUT	No	Yes	None	TIMEOUT
TCO SMI - INTRUDER_N asserts	No	Yes	INTRD_SEL	INTRD_DET
TCO SMI – OS writes to TCO_DAT_IN register	No	Yes	None	OS_TCO_SMI
TCO SMI – NMI occurred (and NMIs mapped to SMI)	No	Yes	NMI2SMI_EN=1	TCO_STS, NMI2SMI_STS
TCO SMI – Changes of the WPD (Write Protect Disable) bit from 0 to 1	No	Yes	LE (Lock Enable)=1	BIOSWR_STS
TCO SMI – Write attempted to BIOS	No	Yes	WPD=0	BIOSWR_STS
BIOS_RLS written to 1 (Note 7)	Yes	No	GBL_EN=1	GBL_STS
GBL_RLS written to	No	Yes	BIOS_EN=1	BIOS_STS
Write to B2h register	No	Yes	APMC_EN = 1	APM_STS
Periodic timer expires	No	Yes	PERIODIC_EN=1	PERIODIC_STS
64 ms timer expires	No	Yes	SWSMI_TMR_EN=1	SWSMI_TMR_STS
Enhanced USB Legacy Support Event	No	Yes	LEGACY_USB2_EN = 1	LEGACY_USB2_STS
Device monitors match address in its range	No	Yes	Refer DEVTRAP_STS register description	DEVTRAP_STS
SMBus Host Controller	No	Yes	SMB_SMI_EN, Host Controller Enabled	SMBus host status reg
SMBus Target SMI message	No	Yes	None	SMBUS_SMI_STS
SMBus SMB_ALERT_N signal active	No	Yes	None	SMBUS_SMI_STS
SMBus Host Notify message received	No	Yes	HOST_NOTIFY_INTREN	SMBUS_SMI_STS, HOST_NOTIFY_STS
PMC_BATLOW_N assertion	Yes	Yes	BATLOW_EN=1	BATLOW_STS
Access microcontroller 62h/66h	No	Yes	MCSMI_EN	MCSMI_STS
SLP_EN bit written to 1	No	Yes	MI_ON_SLP_EN=1	SMI_ON_SLP_EN_STS
SPI Command Completed	No	Yes	None	SPI_SMI_STS
eSPI SCI/SMI Request	Yes	Yes	eSPI_SCI_EN	eSPI_SCI_STS eSPI_SMI_STS
Software Generated GPE	Yes	Yes	SWGPE_EN=1	SWGPE_STS
Intel® CSE	Yes	Yes	ME_SCI_EN=1 ME_SCI_EN=0; ME_SMI_EN=1;	ME_SCI_STS ME_SMI_STS
GPIO Lockdown Enable bit changes from '1' to '0'	No	Yes	GPIO_UNLOCK_SMI_EN =1	GPIO_UNLOCK_SMI_STS

Table 3-20. Causes of SMI and SCI (Sheet 3 of 3)

Cause	SCI	SMI	Additional Enables (Note 1)	Where Reported
USB 3.1 (xHCI) SMI Event	No	Yes	xHCI_SMI_EN=1	xHCI_SMI_STS
Wake Alarm Device Timer	Yes	Yes	WADT_EN	WADT_STS
PSE	Yes	No	OSE_EN	OSE_STS
RTC update-in-progress	No	Yes	Refer Datasheet Vol2, RDC#636722	RTC_UIP_SMI_STS
SIO SMI events	No	Yes	SIP_SMI_EN	SIO_SMI_STS
eMMC, SD/SDIO	No	Yes	SCC_SMI_EN	SCC_SMI_STS
Legacy (keyboard) logic (Port 64/60)	No	Yes	LEGACY_USB_EN	LEGACY_USB_STS

Notes:
1. SCI_EN must be 1 to enable SCI, except for BIOS_RLS. SCI_EN must be 0 to enable SMI.
2. SCI can be routed to cause interrupt 9:11 or 20:23 (20:23 only available in APIC mode).
3. GBL_SMI_EN must be 1 to enable SMI.
4. EOS must be written to 1 to re-enable SMI for the next 1.
5. The PCH must have SMI fully enabled when the PCH is also enabled to trap cycles. If SMI is not enabled in conjunction with the trap enabling, then hardware behavior is undefined.
6. When a power button override first occurs, the system will transition immediately to S5. The SCI will only occur after the next wake to S0 if the residual status bit (PRBTNOR_STS) is not cleared prior to setting SCI_EN.
7. GBL_STS being set will cause an SCI, even if the SCI_EN bit is not set. Software must take great care not to set the BIOS_RLS bit (which causes GBL_STS to be set) if the SCI handler is not in place.
8. Refer to GPIO chapter for specific GPIOs enabled for SCIs and/or SMIs

3.7.1 PCI Express* SCI

PCI Express* ports and the processor have the ability to cause PME using messages. When a PME message is received, the PCH will set the PCI_EXP_STS bit. If the PCI_EXP_EN bit is also set, the PCH can cause an SCI using the GPE0_STS (replaced GPE1_STS) register.

3.7.2 PCI Express* Hot-Plug

PCI Express* has a hot-plug mechanism and is capable of generating a SCI using the GPE0 (replaced GPE1) register. It is also capable of generating an SMI. However, it is not capable of generating a wake event.

3.8 Sleep States

Sleep States Overview

The PCH supports different sleep states (S3-S5), which are entered by methods such as setting the SLP_EN bit or due to a Power Button press. The entry to the Sleep states is based on several assumptions:

- The G3 state cannot be entered using any software mechanism. The G3 state indicates a complete loss of power.

Initiating Sleep State

Sleep states (S3-S5) are initiated by:

- Masking interrupts, turning off all bus initiator enable bits, setting the desired type in the SLP_TYP field, and then setting the SLP_EN bit. The hardware then attempts to gracefully put the system into the corresponding Sleep state.
- Pressing the PMC_PWRBTN_N Signal for more than 4 seconds to cause a Power Button Override event. In this case the transition to the S5 state is less graceful, since there are no dependencies on OPI messages from the compute die or on clocks other than the RTC clock.
- Assertion of the THRMTRIP_N signal will cause a transition to the S5 state. This can occur when system is in the S0 state.
- Shutdown by integrated manageability functions (PSE OOB Manageability).
- Internal watchdog timer timeout events.

Table 3-21. Sleep Types

Sleep Type	Comment
S3	The PCH asserts PMC_SLP_S3_N. The PMC_SLP_S3_N signal controls the power to non-critical circuits. Power is only retained to devices needed to wake from this sleeping state, as well as to the memory.
S4	The PCH asserts PMC_SLP_S3_N and PMC_SLP_S4_N. The motherboard uses the PMC_SLP_S4_N signal to shut off the power to the memory subsystem and any other unneeded subsystem. Only devices needed to wake from this state should be powered.
S5	The PCH asserts PMC_SLP_S3_N, PMC_SLP_S4_N and PMC_SLP_S5_N.

Existing Sleep State

Sleep states (S3-S5) are exited based on wake events. The wake events forces the system to a full on state (S0), although some non-critical subsystems might still be shut off and have to be brought back manually. For example, the storage subsystem may be shut off during a sleep state and have to be enabled using a GPIO pin before it can be used.

Upon exit from the PCH-controlled Sleep states, the WAK_STS bit is set. The possible causes of wake events (and their restrictions) are shown in the table below.

Note: If the PMC_BATLOW_N signal is asserted, the PCH does not attempt to wake from an S3-S5 state, even if the power button is pressed. This prevents the system from waking when the battery power is insufficient to wake the system. Wake events that occur while PMC_BATLOW_N is asserted are latched by the PCH, and the system wakes after PMC_BATLOW_N is de-asserted.

Table 3-22. Causes of Wake Events (Sheet 1 of 2)

Cause	How Enabled	Wake from Sx	Wake from Sx After Power Loss[2]	Wake from "Reset" Type[3]
RTC Alarm	Set RTC_EN bit in PM1_EN_STS register	Yes	Yes	No
Power Button	Always enabled as Wake event.	Yes	Yes	Yes
Any GPIOs except DSW GPIOs can be enabled for wake	Refer to Note 5	Yes	No	No

Table 3-22. Causes of Wake Events (Sheet 2 of 2)

Cause	How Enabled	Wake from Sx	Wake from Sx After Power Loss[2]	Wake from "Reset" Type[3]
Intel® High Definition Audio	Event sets PME_B0_STS bit; PM_B0_EN must be enabled. Can not wake from S5 state if it was entered due to power failure or power button override.	Yes	Yes	No
Primary PME#	PME_B0_EN bit in GPE0_EN[127:96] register.	Yes	Yes	No
Secondary PME#	Set PME_EN bit in GPE0_EN[127:96] register	Yes	Yes	No
PCI Express* WAKE pin (PMC_WAKE_N)	PCIEXP_WAKE_DIS bit.	Yes	Yes	No
SMB_ALERT_N	(Note 4)	Yes	Yes	Yes
SMBus Target Wake Message (01h)	Wake/SMI# command always enabled as a Wake event. **Note:** SMBus Target Message can wake the system from S4/S5, as well as from S5 due to Power Button Override.	Yes	Yes	Yes
SMBus Host Notify message received	HOST_NOTIFY_WKEN bit SMBus Target Command register. Reported in the SMB_WAK_STS bit in the GPE0_STS register.	Yes	Yes	Yes
Intel®CSE NonMaskable Wake	Always enabled as a wake event.	Yes	Yes	Yes
Integrated WoL Enable Override	WoL Enable Override bit (in Configuration Space).	Yes	Yes	Yes
Wake Alarm Device	WADT_EN in GPE0_EN[127:96]	Yes	No	No
PMC_ACPRESENT	AC_PRESENT_WAKE_EN[6]	No	No	No

Notes:
1. If PMC_BATLOW_N signal is low, PCH will not attempt to wake from S4/S5, even if a valid wake event occurs. This prevents the system from waking when battery power is insufficient to wake the system. However, once PMC_BATLOW_N de-asserts, the system will boot.
2. This column represents what the PCH would honor as wake events but there may be enabling dependencies on the device side which are not enabled after a power loss.
3. Reset Types include: Power Button override, Intel® CSE-initiated power button override, Intel CSE-initiated host partition reset with power down, Intel CSE Watchdog Timer, SMBus unconditional power down, processor thermal trip, PCH catastrophic temperature event.
4. SMB_ALERT_N signal is multiplexed with a GPIO pin that defaults to GPIO mode. Hence, SMB_ALERT_N related wakes are possible only when this GPIO is configured in native mode, which means that BIOS must program this GPIO to operate in native mode before this wake is possible. Because GPIO configuration is in the resume well, wakes remain possible until one of the following occurs: BIOS changes the pin to GPIO mode, a G3 occurs.
5. There are only 72 bits in the GPE registers to be assigned to GPIOs, though any of the GPIOs can trigger a wake, only those status of GPIO mapped to 1-tier scheme are directly accessible through the GPE status registers. For those GPIO mapped under 2-tier scheme, their status would be reflected under single main status, "GPIO_TIER2_SCI_STS" or GPE0_STS and further comparison needed to know which 2-tier GPI(s) has triggered the GPIO Tier 2 SCI.

PCI Express* WAKE# Signal and PME Event Message

PCI Express* ports can wake the platform from S4 or S5 using the WAKE# pin (PMC_WAKE_N). WAKE# is treated as a wake event, but does not cause any bits to go active in the GPE_STS register.

Note: PCI Express* WAKE# pin is an Output in S0ix states hence this pin cannot be used to wake up the system during S0ix states.

PCI Express* ports and the processor have the ability to cause PME using messages.These are logically OR'd to set the single PCI_EXP_STS bit. When a PME message is received, the PCH will set the PCI_EXP_STS bit. If the PCI_EXP_EN bit is also set, the PCH can cause an SCI via GPE0_STS register.

Sx-G3-Sx, Handling Power Failures

Depending on when the power failure occurs and how the system is designed, different transitions could occur due to a power failure.

The AFTERG3_EN bit provides the ability to program whether or not the system should boot once power returns after a power loss event. If the policy is to not boot, the system remains in an S5 state (unless previously in S4). There are only three possible events that will wake the system after a power failure.

1. PMC_PWRBTN_N: PMC_PWRBTN_N is always enabled as a wake event. When PCH_DPWROK is low (G3 state), the PWRBTN_STS bit is reset. When the PCH exits G3 after power returns (PCH_DPWROK goes high), the PMC_PWRBTN_N signal will transition high due internal Pull-up, unless there is an on-board Pull-up/Pull-down) and the PWRBTN_STS bit is 0.
2. RTC Alarm: The RTC_EN bit is in the RTC well and is preserved after a power loss. Like PWRBTN_STS the RTC_STS bit is cleared when PCH_DPWROK goes low.
3. Any enabled wake event that was preserved through the power failure.

PMC_DSW_PWROK going low would place the PCH into a G3 state.

Although PME_EN is in the RTC well, this signal cannot wake the system after a power loss. PME_EN is cleared by RTC_RST_N, and PME_STS is cleared by PMC_RSMRST_N.

Depending on when the power failure occurs and how the system is designed, different transitions could occur due to a power failure.

Table 3-23. Transitions Due to Power Failure

State at Power Failure	AFTERG3_EN bit	Transition When Power Returns and PMC_BATLOW_N is inactive
S0, S3	1 0	S5 S0
S4	1 0	S4 S0
S5	1 0	S5 S0

3.9 Event Input Signals and Their Usage

The PCH has various input signals that trigger specific events. This section describes those signals and how they should be used.

PMC_PWRBTN_N (Power Button)

The PCH PMC_PWRBTN_N signal operates as a "Fixed Power Button" as described in the Advanced Configuration and Power Interface Specification. PMC_PWRBTN_N signal has a 16 ms de-bounce on the input. The state transition descriptions are included in the below table.

Note: This signal is not able to cause a GPE, which is required to implement a 'Control method' power button as described in the ACPI specification.

After any PMC_PWRBTN_N assertion (falling edge), the 16ms de-bounce applies before the state transition starts if PB_DB_MODE='0'. If PB_DB_MODE='1', the state transition starts right after any PMC_PWRBTN_N assertion (before passing through the de-bounce logic) and subsequent falling PMC_PWRBTN_N edges are ignored until after 16ms.

During the time that any PMC_SLP_*_N signal is stretched for an enabled minimum assertion width, the host wake-up is held off. As a result, it is possible that the user will press and continue to hold the Power Button waiting for the system to wake. Unfortunately, a 4 second press of the Power Button is defined as an unconditional power down, resulting in the opposite behavior that the user was intending. Therefore, the Power Button Override Timer will be extended to 9-10 seconds while the PMC_SLP_*_N stretching timers are in progress. Once the stretching timers have expired, the Power Button will awake the system. If the user continues to press Power Button for the remainder of the 9-10 seconds it will result in the override condition to S5.

Extension of the Power Button Override timer is only enforced following graceful sleep entry and during host partition resets with power cycle or power down. The timer is not extended immediately following power restoration after a global reset or G3.

The PCH also supports modifying the length of time the Power Button must remain asserted before the unconditional power down occurs (4-14 seconds). The length of the Power Button override duration has no impact on the "extension" of the power button override timer while PMC_SLP_*_N stretching is in progress. The extended power button override period while stretching is in progress remains 9-10 seconds in all cases.

Table 3-24. Transitions Due to Power Button (Sheet 1 of 2)

Present State	Event	Transition/Action	Comment
S0/Cx	PMC_PWRBTN_N goes low	SMI or SCI generated (depending on SCI_EN, PWRBTN_EN and GLB_SMI_EN)	Software typically initiates a Sleep state **Note:** Processing of transitions starts within 100 us of the PMC_PWRBTN_N input pin to PCH going low.[1]
S5	PMC_PWRBTN_N goes low	Wake Event. Transitions to S0 state	Standard wakeup **Note:** Could be impacted by PMC_SLP_*_N min assertion. The minimum time the PMC_PWRBTN_N pin should be asserted is 150 us. The PCH will start processing this change once the minimum time requirement is satisfied.[1]
G3	PMC_PWRBTN_N pressed	None	No effect since no power Not latched nor detected **Notes:** 1. During G3 exit, PMC_PWRBTN_N pin must be kept de-asserted for a minimum time of 500 us after the PMC_RSMRST_N has deasserted.[2] 2. Beyond this point, the minimum time the PMC_PWRBTN_N pin has to be asserted to be registered by PCH as a valid wake event is 150 us.[1]

Table 3-24. Transitions Due to Power Button (Sheet 2 of 2)

Present State	Event	Transition/Action	Comment
S0-S4	PMC_PWRBTN_N held low for at least four consecutive seconds	Unconditional transition to S5 state.	No dependence on processor or any other subsystem **Note:** Due to internal PCH latency, it could take up to an additional ~1.3s after PMC_PWRBTN_N has been held low for 4s before the system would begin transitioning to S5.

Notes:
1. If PM_CFG.PB_DB_MODE='0', the debounce logic adds 16 ms to the start/minimum time for processing of power button assertions.
2. This minimum time is independent of the PM_CFG.PB_DB_MODE value.
3. The amount of time PMC_PWRBTN_N must be asserted is configurable via PM_CFG2.PBOP. 4 seconds is the default.

Power Button Override Function

If PMC_PWRBTN_N is observed active for at least four consecutive seconds (always sampled after the output from debounce logic), the PCH should unconditionally transition to the G2/S5 state, regardless of present state (S0 – S4), even if the PMC_PCH_PWROK is not active. In this case, the transition to the G2/S5 state does not depend on any particular response from the processor, nor any similar dependency from any other subsystem.

The minimum period is configurable by BIOS and defaults to the legacy value of 4 seconds.

The PMC_PWRBTN_N status is readable to check if the button is currently being pressed or has been released. If PM_CFG.PB_DB_MODE='0', the status is taken after the debounce. If PM_CFG.PB_DB_MODE='1', the status is taken before the de-bounce. In either case, the status is readable using the PWRBTN_LVL bit.

Note: The 4-second PMC_PWRBTN_N assertion should only be used if a system lock-up has occurred.

Sleep Button

The Advanced Configuration and Power Interface Specification defines an optional Sleep button. It differs from the power button in that it only is a request to go from S0 to S4 (not S5). Also, in an S5 state, the Power Button can wake the system, but the Sleep Button cannot.

Although the PCH does not include a specific signal designated as a Sleep Button, one of the GPIO signals can be used to create a "Control Method" Sleep Button. Refer the Advanced Configuration and Power Interface Specification for implementation details.

PME# (PCI Power Management Event)

The PME# signal comes from a PCI Express* device to request that the system be restarted. The PME# signal can generate an SMI#, SCI, or optionally a wake event. The event occurs when the PME# signal goes from high to low. No event is caused when it goes from low to high.

There is also an internal PME_B0_STS bit that will be set by the PCH when any internal device with PCI Power Management capabilities on bus 0 asserts the equivalent of the PME# signal. This is separate from the external PME# signal and can cause the same effect.

PMC_SYS_RESET_N Signal

When the PMC_SYS_RESET_N pin is detected as active (on signal's falling edge if de-bounce logic is disabled, or after 16 ms if 16ms de-bounce logic is enabled), the PCH attempts to perform a "graceful" reset by entering a host partition reset entry sequence.

Once the reset is asserted, it remains asserted for 5 to 6 ms regardless of whether the PMC_SYS_RESET_N input remains asserted or not. It cannot occur again until PMC_SYS_RESET_N has been detected inactive after the de-bounce logic, and the system is back to a full S0 state with PMC_PLTRST_N inactive.

Notes:

1. The normal behavior for a PMC_SYS_RESET_N assertion is host partition reset without power cycle. However, if bit 3 of the CF9h I/O register is set to '1' then PMC_SYS_RESET_N will result in a full power-cycle reset.
2. It is not recommended to use the PMC_PCH_PWROK pin for a reset button as it triggers a global power cycle reset.
3. PMC_SYS_RESET_N is in the primary power well but it only affects the system when PMC_PCH_PWROK is high. It should not be allowed to float while PMC_SLP_S3_N is de-asserted.

THRMTRIP_N Signal

If THRMTRIP_N goes active, the processor is indicating an overheat condition, and the PCH immediately transitions to an S5 state, driving PMC_SLP_S3_N, PMC_SLP_S4_N, PMC_SLP_S5_N low, and setting the GEN_PMCON_2.PTS bit. The transition will generally look like a power button override.

When a THRMTRIP_N event occurs, the PCH will power down immediately without following the normal S0 -> S5 path. The PCH will immediately drive PMC_SLP_S3_N,

PMC_SLP_S4_N, and PMC_SLP_S5_N low within 1 ms after sampling THRMTRIP_N active.

The reason the above is important is as follow: if the processor is running extremely hot and is heating up, it is possible (although very unlikely) that components around it, such as the PCH, are no longer executing cycles properly. Therefore, if THRMTRIP_N goes active, and the PCH is relying on various handshakes to perform the power down, the handshakes may not be working, and the system will not power down. Hence the need for PCH to power down immediately without following the normal S0 -> S5 path.

The PCH provides filtering for short low glitches on the THRMTRIP_N signal in order to prevent erroneous system shut downs from noise. Glitches shorter than 25 nsec are ignored.

PCH must only honor the THRMTRIP_N pin while it is being driven to a valid state by the processor. The THRMTRIP_N Valid Point ='0', implies PCH will start monitoring THRMTRIP_N at PMC_PLTRST_N de-assertion (default). The THRMTRIP_N Valid Point ='1', implies PCH will start monitoring THRMTRIP_N at PROCPWRGD assertion. Regardless of the setting, the PCH must stop monitoring THRMTRIP_N at PROCPWRGD deassertion.

Note: A thermal trip event will clear the PWRBTN_STS bit.

3.10 Reset Behavior

When a reset is triggered, the PCH will send a warning message to the processor to allow the processor to attempt to complete any outstanding memory cycles and put memory into a safe state before the platform is reset. When the processor is ready, it will send an acknowledge message to the PCH. Once the message is received the PCH asserts PMC_PLTRST_N.

The PCH does not require an acknowledge message from the processor to trigger PMC_PLTRST_N. A global reset will occur after four seconds if an acknowledge from the processor is not received.

When the PCH causes a reset by asserting PMC_PLTRST_N, its output signals will go to their reset states.

A reset in which the host platform is reset and PMC_PLTRST_N is asserted is called a Host Reset or Host Partition Reset. Depending on the trigger a host reset may also result in power cycling, refer to the below table for details. If a host reset is triggered and the PCH times out before receiving an acknowledge message from the processor a Global Reset with power-cycle will occur.

A reset in which the host and Intel®CSE partitions of the platform are reset is called a Global Reset. During a Global Reset, all PCH functionality is reset except RTC Power Well backed information and Suspend well status, configuration, and functional logic for controlling and reporting the reset. Intel®CSE and Host power back up after the power-cycle period.

Straight to S5 is another reset type where all power wells that are controlled by the PMC_SLP_S3_N and PMC_SLP_S4_N pins, as well as PMC_SLP_S5_N, are turned off. All PCH functionality is reset except RTC Power Well backed information and Suspend well status, configuration, and functional logic for controlling and reporting the reset. The host stays there until a valid wake event occurs.

The following table shows the various reset triggers.

Table 3-25. Causes of Host and Global Resets (Sheet 1 of 3)

Trigger	Host Reset Without Power Cycle[1]	Host Reset With Power Cycle[2]	Global Reset With Power Cycle[3]	Straight to S5[6] (Host Stays There)
Write of 0Eh to CF9h (RST_CNT Register) when CF9h when Global Reset Bit=0b	No	Yes	No[4]	No
Write of 06h to CF9h (RST_CNT Register) when CF9h when Global Reset Bit=0b	Yes	No	No[4]	No
Write of 06h or 0Eh to CF9h (RST_CNT Register) when CF9h when Global Reset Bit=1b	No	No	Yes	No
PMC_SYS_RESET_N Asserted and CF9h (RST_CNT Register) Bit 3 = 0	Yes	No	No[4]	No

Table 3-25. Causes of Host and Global Resets (Sheet 2 of 3)

Trigger	Host Reset Without Power Cycle[1]	Host Reset With Power Cycle[2]	Global Reset With Power Cycle[3]	Straight to S5[6] (Host Stays There)
Write of 0Eh to CF9h (RST_CNT Register) when CF9h when Global Reset Bit=0b	No	Yes	No[4]	No
PMC_SYS_RESET_N Asserted and CF9h (RST_CNT Register) Bit 3 = 1	No	Yes	No[4]	No
SMBus Target Message received for Reset with Power-Cycle	No	Yes	No[4]	No
SMBus Target Message received for Reset without Power-Cycle	Yes	No	No[4]	No
SMBus Target Message received for unconditional Power Down	No	No	No	Yes
TCO Watchdog Timer reaches zero two times	Yes	No	No[4]	No
Power Failure: PMC_PCH_PWROK signal goes inactive in S0 or PMC_DSW_PWROK drops	No	No	Yes	No
PMC_SYS_PWROK Failure: PMC_SYS_PWROK signal goes inactive in S0	No	No	Yes	No
Processor Thermal Trip (THRMTRIP_N) causes transition to S5 and reset asserts	No	No	No	Yes
PCH internal thermal sensors signals a catastrophic temperature condition	No	No	No	Yes
Power Button 4 second override causes transition to S5 and reset asserts	No	No	No	Yes
Special shutdown cycle from processor causes CF9h-like PMC_PLTRST_N and CF9h Global Reset Bit = 1	No	No	Yes	No
Special shutdown cycle from processor causes CF9h-like PMC_LTRST_N and CF9h Global Reset Bit = 0 and CF9h (RST_CNT Register) Bit 3 = 1	No	Yes	No[4]	No
Special shutdown cycle from processor causes CF9h-like PMC_LTRST_N and CF9h Global Reset Bit = 0 and CF9h (RST_CNT Register) Bit 3 = 0	Yes	No	No[4]	No
Intel® Converged Security Engine Triggered Host Reset without Power-Cycle	Yes	No	No[4]	No

Table 3-25. Causes of Host and Global Resets (Sheet 3 of 3)

Trigger	Host Reset Without Power Cycle[1]	Host Reset With Power Cycle[2]	Global Reset With Power Cycle[3]	Straight to S5[6] (Host Stays There)
Write of 0Eh to CF9h (RST_CNT Register) when CF9h when Global Reset Bit=0b	No	Yes	No[4]	No
Intel® Converged Security Engine Triggered Host Reset with Power-Cycle	No	Yes	No[4]	No
Intel® Converged Security Engine Triggered Power Button Override	No	No	No	Yes
Intel® Converged Security Engine Watchdog Timer Timeout	No	No	No[7]	Yes
Intel® Converged Security Engine Triggered Global Reset	No	No	Yes	No
Intel® Converged Security Engine Triggered Host Reset with power down (host stays there)	No	Yes[5]	No[4]	No
PMC_PLTRST_N Entry Timeout (Note7)	No	No	Yes	No
PROC_PWR_GD Stuck Low	No	No	Yes	No
Power Management Watchdog Timer	No	No	No[7]	Yes
Intel® Converged Security Engine Hardware Uncorrectable Error	No	No	No[7]	Yes

Notes:
1. The PCH drops this type of reset request if received while the system is in S3/S4/S5.
2. PCH does not drop this type of reset request if received while system is in a software-entered S3/S4/S5 state. However, the PCH will perform the reset without executing the RESET_WARN protocol in these states.
3. The PCH does not send warning message to processor, reset occurs without delay.
4. Trigger will result in Global Reset with Power-Cycle if the acknowledge message is not received by the PCH.
5. The PCH waits for enabled wake event to complete reset.
6. PMC_PLTRST_N Entry Timeout is automatically initiated if the hardware detects that the PMC_PLTRST_N sequence has not been completed within 4 seconds of being started.
7. Trigger will result in Global Reset with Power-Cycle if AGR_LS_EN=1 and Global Reset occurred while the current or destination state was S0.

§ §

4 Thermal Management

4.1 Thermal and Power Specifications

Table 4-1. Processor Specifications

SKU Number	SKU Series	Processor IA Cores	Processor IA Core / Burst Frequency [GHz]	Graphics HFM / Burst Frequency [MHz]	Integrated Heat Spreader (IHS)	Thermal Design Power (TDP) [W]	Temperature Specification	
							Tj [°C]	Tc [°C]
1/1A	PC Client	Refer to section 1.6			No	10	0 to 105	N/A
2					No	10	0 to 105	N/A
3/3A					No	6.5	0 to 105	N/A
4					No	6.5	0 to 105	N/A
5	Embedded				Yes	6	-40 to 105	98
6					Yes	9	-40 to 105	95
7					Yes	12	-40 to 105	90
8/8PU	Industrial				Yes	6	-40 to 110	103
9/9PU					Yes	9	-40 to 110	100
10					Yes	12	-40 to 110	97
11	FuSa				Yes	12	-40 to 110	97
12					Yes	4.5	-40 to 110	105

Notes:

The TDP values are the worst-case average power dissipation in junction temperature operating condition limit, for the SKU Segment and Configuration, for which the processor is validated during manufacturing when executing an associated Intel specified TDP workload on an Intel-specified base platform configuration. The actual average power dissipation may vary on a per processor basis below TDP and, in some cases, it may not be possible for a processor to meet TDP power dissipation irrespective of what workload is being executed.

TDP workload may consist of a combination of processor IA core intensive and graphics core intensive applications. Refer to Section 4.2.1 for further workload information.

Under a concurrent workload of IA core intensive and graphics core intensive applications, where both IA & graphics cores are operating at HFM frequency, it may be possible for some processors to exceed the TDP specified in Table 4-1. It is recommended that the processor thermal solution should be designed with a +15% margin to account for the increase in power dissipation to prevent thermal and/or performance throttling.

N/A = Not Applicable

4.2 Processor Thermal Management

The thermal solution provides both component-level and system-level thermal management. To allow optimal operation and long-term reliability of Intel processor-based systems, the system/processor thermal solution should be designed so that the processor:

- Remains below the maximum junction temperature (Tj_{MAX}) specification at the maximum Thermal Design Power (TDP).
- Conforms to system constraints, such as system acoustics, system skin-temperatures, and exhaust-temperature requirements.

Caution: Thermal specifications given in this chapter are on the component and package level and apply specifically to the processor. Operating the processor outside the specified limits may result in permanent damage to the processor and potentially other components in the system.

4.2.1 Thermal Considerations

Processor TDP is the average power dissipation (at Tjmax) that is validated during manufacturing for the base configuration when executing a near worst case commercially available workload as specified by Intel for the SKU segment. The base configuration for this platform is defined in the Thermal Mechanical Design Guide (RDC# 612258). The workload specified for this platform is:

- **PC Client & Embedded Processors:** CPU-intensive with minimal GPU activity. The workload assumes that Compute die & PCH power management features are not turned off.
- **Industrial & FuSa Processors:** CPU-intensive with GPU frequency locked at LFM frequency (200MHz). The workload assumes that Compute die & PCH power management features are turned off.

The average power dissipation of some processors may be caused to exceed the specified TDP value when:

- A concurrently CPU-intensive and GPU-intensive workload is executed.
- The platform design is different from the base configuration. For example, when an HDMI display is used instead of an eDP display.
- The Compute Die and PCH power management features are disabled. Examples of power management features can be found in Chapter 23 of the PCH BIOS Specification (RDC# 610273).

In such cases where the TDP value is exceeded, the processor will opportunistically throttle the CPU and/or GPU frequency to ensure the processor stays within configured power and thermal limits.

The processor integrates multiple processing IA cores, graphics cores and a PCH on a single package.This may result in power distribution differences across the package and should be considered when designing the thermal solution.

Intel® Burst Technology allows processor IA cores to run faster than the base frequency. It is invoked opportunistically and automatically as long as the processor is conforming to its temperature, power delivery and current control limits. When Intel® Burst Technology is enabled:

- Applications are expected to run closer to TDP more often as the processor will attempt to maximize performance by taking advantage of estimated available energy budget in the processor package.
- The processor may exceed the TDP for short durations to utilize any available thermal capacitance within the thermal solution. The duration and time of such operation can be limited by platform runtime configurable registers within the processor.
- Graphics peak frequency operation is based on the assumption of only one of the graphics domains (GT/GTx) being active. This definition is similar to the IA core Burst concept, where peak burst frequency can be achieved when only one IA core is active. Depending on the workload being applied and the distribution across the graphics domains the user may not observe peak graphics frequency for a given workload or benchmark.
- Thermal solutions and platform cooling that are designed to less than thermal design guidance may experience thermal and performance issues.

4.2.1.1 Opportunistic Throttling Management

In platforms where the base configuration and/or workload results in opportunistic throttling, it may be possible to reduce the throttling by increasing the Power Limit 1 (PL1) value above TDP.

Note: PL1 must not be set higher than thermal solution cooling limits.

Note: PL1 must be increased only to the value where CPU and/or GPU stops opportunistically throttling below their High Frequency Mode (HFM) frequencies.

Caution: Increasing PL1 above TDP + 0.9W is not supported and may result in the processor being unable to meet Intel’s reliability goals.

4.2.1.2 Package Power Control

The package power control settings of PL1, PL2, PL3, PL4 and Tau allow the designer to configure Intel® Burst Technology to match the platform power delivery and package thermal solution limitations.

- Power Limit 1 (PL1): A threshold for average power that will not exceed - recommend to set to equal TDP power. PL1 should not be set higher than thermal solution cooling limits.
- Power Limit 2 (PL2): A threshold that if exceeded, the PL2 rapid power limiting algorithms (RAPL) will attempt to limit the spike above PL2 should not be set higher than maximum, non-transient capabilities of the processor's power supplies.
- Power Limit 3 (PL3): A threshold that if exceeded, the PL3 rapid power limiting algorithms will attempt to limit the duty cycle of spikes above PL3 by reactively limiting frequency. This is an optional setting for battery powered systems to reduce stress on the main battery.
- Power Limit 4 (PL4): A limit that will not be exceeded, the PL4 power limiting algorithms will preemptively limit frequency to prevent spikes above PL4.This is an optional setting for battery powered systems to reduce stress on the DC power supply.
- Burst Time Parameter (Tau): An averaging constant used for PL1 Exponential Weighted Moving Average (EWMA) power calculation.

Note: Implementation of Intel® Burst Technology only requires configuring PL1, PL1 Tau and PL2. PL3 and PL4 are disabled by default.

PL2 limit is recommended to be 1.25 * PL1. Higher limit can be set if the thermal solution provides sufficient cooling margins.

4.2.1.3 Burst Time Parameter (Tau)

Burst Time Parameter (Tau) is a mathematical parameter (units of seconds) that controls the burst algorithm. During a maximum power burst event, the processor could sustain PL2 for a duration longer than the Burst Time Parameter. If the power value and/or Burst Time Parameter is changed during runtime, it may take some time based on the new Burst Time Parameter level for the algorithm to settle at the new control limits. The time varies depending on the magnitude of the change, power limits and other factors. There is an individual Burst Time Parameter associated with Package Power Control and Platform Power Control.

4.2.2 Thermal Management Features

Occasionally the processor may operate in conditions that are near to its maximum operating temperature. This can be due to internal overheating or overheating within the platform. In order to protect the processor and the platform from thermal failure, several thermal management features exist to reduce package power consumption and thereby temperature in order to remain within normal operating limits.

4.2.2.1 Adaptive Thermal Monitor

The purpose of the Adaptive Thermal Monitor is to reduce processor IA core power consumption and temperature until it operates below its maximum operating temperature. Processor IA core power reduction is achieved by:

- Adjusting the operating frequency (using the processor IA core ratio multiplier) and voltage.
- Modulating (starting and stopping) the internal processor IA core clocks (duty cycle).

The Adaptive Thermal Monitor can be activated when the package temperature, monitored by any Digital Thermal Sensor (DTS), meets its maximum operating temperature. The maximum operating temperature implies maximum junction temperature Tj_{MAX}.

Reaching the maximum operating temperature activates the Thermal Control Circuit (TCC). When activated the TCC causes both the processor IA core and graphics core to reduce frequency and voltage adaptively. The Adaptive Thermal Monitor will remain active as long as the package temperature remains at its specified limit. Therefore, the Adaptive Thermal Monitor will continue to reduce the package frequency and voltage until the TCC is de-activated.

Clock modulation (Section 4.2.2.1.3) is another means to reduce the processor core clock. The duty cycle of the clock modulation can be programmed through MSR (see Section 4.2.2.11).

Tj_{MAX} is factory calibrated and is not user configurable. The default value is software visible in the TEMPERATURE_TARGET (0x1A2) MSR, bits [23:16].

The Adaptive Thermal Monitor does not require any additional hardware, software drivers, or interrupt handling routines. It is not intended as a mechanism to maintain processor thermal control to PL1 = TDP. The system design should provide a thermal solution that can maintain normal operation when PL1 = TDP within the intended usage range.

Adaptive Thermal Monitor protection is always enabled.

4.2.2.1.1 TCC Activation Offset

TCC Activation Offset can be set as an offset from Tj_{MAX} to lower the onset of TCC and Adaptive Thermal Monitor. In addition, there is an optional time window (Tau) to manage processor performance at the TCC Activation offset value via an EWMA (Exponential Weighted Moving Average) of temperature.

TCC Activation Offset with Tau=0

An offset (degrees Celsius) can be written to the TEMPERATURE_TARGET (0x1A2) MSR, bits [29:24], the offset value will be subtracted from the value found in bits [23:16]. When the time window (Tau) is set to zero, there will be no averaging, the offset, will be subtracted from the Tj_{MAX} value and used as a new max temperature set point for Adaptive Thermal Monitoring. This will have the same behavior as in prior products to have TCC activation and Adaptive Thermal Monitor to occur at this lower target silicon temperature.

If enabled, the offset should be set lower than any other passive protection such as ACPI _PSV trip points.

TCC Activation Offset with Tau

To manage the processor with the EWMA (Exponential Weighted Moving Average) of temperature, an offset (degrees Celsius) is written to the TEMPERATURE_TARGET (0x1A2) MSR, bits [29:24], and the time window (Tau) is written to the TEMPERATURE_TARGET (0x1A2) MSR [6:0]. The Offset value will be subtracted from the value found in bits [23:16] and be the temperature.

The processor will manage to this average temperature by adjusting the frequency of the various domains. The instantaneous Tj can briefly exceed the average temperature. The magnitude and duration of the overshoot is managed by the time window value (Tau).

This averaged temperature thermal management mechanism is in addition, and not instead of Tj_{MAX} thermal management. That is, whether the TCC activation offset is 0 or not, TCC Activation will occur at Tj_{MAX}.

4.2.2.1.2 Frequency / Voltage Control

Upon Adaptive Thermal Monitor activation, the processor attempts to dynamically reduce processor temperature by lowering the frequency and voltage operating point. The operating points are automatically calculated by the processor IA core itself and do not require the BIOS to program them. The processor IA core will scale the operating points such that:

- The voltage will be optimized according to the temperature, the processor IA core bus ratio and number of processor IA cores in deep C-states.
- The processor IA core power and temperature are reduced while minimizing performance degradation.

Once the temperature has dropped below the trigger temperature, the operating frequency and voltage will transition back to the normal system operating point.

Once a target frequency/bus ratio is resolved, the processor IA core will transition to the new target automatically.

- On an upward operating point transition the voltage transition precedes the frequency transition.
- On a downward transition the frequency transition precedes the voltage transition.
- The processor continues to execute instructions. However, the processor will halt instruction execution for frequency transitions.

If a processor load-based Enhanced Intel SpeedStep Technology/P-state transition (through MSR write) is initiated while the Adaptive Thermal Monitor is active, there are two possible outcomes:

- If the P-state target frequency is higher than the processor IA core optimized target frequency, the P-state transition will be deferred until the thermal event has been completed.
- If the P-state target frequency is lower than the processor IA core optimized target frequency, the processor will transition to the P-state operating point.

4.2.2.1.3 Clock Modulation

If the frequency/voltage changes are unable to end an Adaptive Thermal Monitor event, the Adaptive Thermal Monitor will utilize clock modulation. Clock modulation is done by alternately turning the clocks off and on at a duty cycle (ratio between clock “on” time and total time) specific to the processor. The duty cycle is factory configured to 25% on and 75% off and cannot be modified. The period of the duty cycle is configured to 32 microseconds when the Adaptive Thermal Monitor is active. Cycle times are independent of processor frequency. A small amount of hysteresis has been included to prevent excessive clock modulation when the processor temperature is near its maximum operating temperature. Once the temperature has dropped below the maximum operating temperature, and the hysteresis timer has expired, the Adaptive Thermal Monitor goes inactive and clock modulation ceases. Clock modulation is automatically engaged as part of the Adaptive Thermal Monitor activation when the frequency/voltage targets are at their minimum settings. Processor performance will be decreased when clock modulation is active. Snooping and interrupt processing are performed in the normal manner while the Adaptive Thermal Monitor is active.

Clock modulation will not be activated by the Package average temperature control mechanism.

4.2.2.2 Digital Thermal Sensor

Each compute die has up to 12 on-die Digital Thermal Sensors (DTS) that detect the processor IA (with 1 sensor in the core), Graphics Engine(9 sensors), Clock Control Unit CCU (1 sensor), and display (1 sensor).

Temperature values from the DTS can be retrieved through:

- A software interface using processor Model Specific Register (MSR).

When temperature is retrieved by the processor MSR, it is the instantaneous temperature of the given DTS. The average DTS temperature may not be a good indicator of package Adaptive Thermal Monitor activation or rapid increases in temperature that triggers the Out of Specification status bit within the PACKAGE_THERM_STATUS (0x1B1) MSR and IA32_THERM_STATUS (0x19C) MSR.

Code execution is halted in C1 or deeper C-states.

Unlike traditional thermal devices, the DTS outputs a temperature relative to the maximum supported operating temperature of the processor (Tj_{MAX}), regardless of TCC activation offset. It is the responsibility of software to convert the relative temperature to an absolute temperature. The absolute reference temperature is readable in the TEMPERATURE_TARGET (0x1A2) MSR . The temperature returned by the DTS is an implied negative integer indicating the relative offset from Tj_{MAX}. The DTS does not report temperatures greater than Tj_{MAX}. The DTS-relative temperature readout directly impacts the Adaptive Thermal Monitor trigger point. When a package DTS indicates that it has reached the TCC activation (a reading of 0x0, except when the TCC activation offset is changed), the TCC will activate and indicate an Adaptive Thermal Monitor event. A TCC activation will lower both processor IA core and graphics core frequency, voltage, or both. Changes to the temperature can be detected using two programmable thresholds, one set above and another below the current temperature, located in the processor thermal MSRs. These thresholds have the capability of generating interrupts using the processor IA core's local APIC.

The thermal thresholds defined for the processor are:

Core Threshold #1 Temperature in IA32_THERM_INTERRUPT (MSR 0x19B) Bits 14:8. This value indicates the offset in degrees below Tj_{MAX} Temperature that will trigger a Thermal Threshold 1 trip.

Package Threshold #1 Temperature in IA32_THERM_INTERRUPT (MSR 0x1B2) Bits 14:8. This value indicates the offset in degrees below Tj_{MAX} Temperature that will trigger a Package Thermal Threshold 1 trip.

Core Threshold #2 Temperature in IA32_THERM_INTERRUPT (MSR 0x19B) Bits 22:16. This value indicates the offset in degrees below Tj_{MAX} Temperature that will trigger a Thermal Threshold 2 trip. Similar to Threshold Value 1.

Package Threshold #2 Temperature in IA32_THERM_INTERRUPT (MSR 0x1B2) Bits 22:16. This value indicates the offset in degrees below Tj_{MAX} Temperature that will trigger a Thermal Threshold 2 trip to all cores in the package. Similar to Core Threshold Value 2.

4.2.2.2.1 Digital Thermal Sensor Accuracy (Taccuracy)

- ±5°C over the temperature range from 50°C to 110°C.
- ±7°C over the temperature range from 30°C to 50°C.
- ±10°C over the temperature range from -10°C to 30°C.
- The sensor itself is functional, from -40°C to 130°C, no accuracy is specified for temperature range beyond 110°C or below -10°C.

Note: The lowest temperature reported by the DTS is Tjmax - 127°C. Tjmax varies with SKU according to Table 4-1.

4.2.2.2.2 Fan Speed Control with Digital Thermal Sensor

Digital Thermal Sensor based fan speed control (T_{FAN}) is a recommended feature to achieve optimal thermal performance. T_{FAN} temperature (sometimes called $T_{CONTROL}$) indicates the relative offset from the Thermal Monitor Trip Temperature at which fans should be engaged. For current temperature reporting, it is recommended that the value MSR PACKAGE_THERM_MARGIN (1A1h) [15:0] be used for fan control software. Intel recommends full cooling capability before the DTS reading reaches Tj_{MAX}.

4.2.2.3 PROCHOT_N Signal

PROCHOT_N (processor hot) is asserted by the processor when the TCC is active. Only a single PROCHOT_N pin exists at a package level. When any DTS temperature reaches the TCC activation temperature, the PROCHOT_N signal will be asserted. PROCHOT_N assertion policies are independent of Adaptive Thermal Monitor enabling.

4.2.2.3.1 Bi-Directional PROCHOT_N

By default, the PROCHOT_N is configured as an input-only signal. When configured as an input or bi-directional signal, PROCHOT_N can be used for thermally protecting other platform components should they overheat as well. When PROCHOT_N is driven by an external device:

- The package will immediately transition to the lowest P-State (Pn) supported by the processor IA cores and graphics cores. This is contrary to the internally-generated Adaptive Thermal Monitor response.
- Clock modulation is not activated.

The processor package will remain at the lowest supported P-state until the system de-asserts PROCHOT_N. The processor can be configured to generate an interrupt upon assertion and de-assertion of the PROCHOT_N signal.

When PROCHOT_N is configured as a bi-directional signal and PROCHOT_N is asserted by the processor, it is impossible for the processor to detect a system assertion of PROCHOT_N. The system assertion will have to wait until the processor de-asserts PROCHOT_N before PROCHOT_N action can occur due to the system assertion. While the processor is hot and asserting PROCHOT_N, the power is reduced but the reduction rate is slower than the system PROCHOT_N response of < 100 us. The processor thermal control is staged in smaller increments over many milliseconds. This may cause several milliseconds of delay to a system assertion of PROCHOT_N while the output function is asserted.

Note: Output-only PROCHOT_N must be enabled by BIOS for platforms integrating a FuSa SKU processor by configuring the PPOWER_CTL.ENABLE_BIDIR_PROCHOT MSR field to 0h and the PPOWER_CTL.DIS_PROCHOT_OUT MSR field to 0h.

4.2.2.3.2 Voltage Regulator Protection using PROCHOT_N

PROCHOT_N may be used for thermal protection of voltage regulators (VR). System designers can create a circuit to monitor the VR temperature and assert PROCHOT_N and, if enabled, activate the TCC when the temperature limit of the VR is reached. When PROCHOT_N is configured as a bi-directional or input only signal, if the system assertion of PROCHOT_N is recognized by the processor, it will result in an immediate transition to the lowest P-State (Pn) supported by the processor IA cores and graphics cores. Systems should still provide proper cooling for the VR and rely on bi-directional

PROCHOT_N only as a backup in case of system cooling failure. Overall, the system thermal design should allow the power delivery circuitry to operate within its temperature specification even while the processor is operating at its TDP.

4.2.2.3.3 Thermal Solution Design and PROCHOT_N Behavior

With a properly designed and characterized thermal solution, it is anticipated that PROCHOT_N will only be asserted for very short periods of time when running the most power intensive applications. The processor performance impact due to these brief periods of TCC activation is expected to be so minor that it would be immeasurable. However, an under-designed thermal solution that is not able to prevent excessive assertion of PROCHOT_N in the anticipated ambient environment may:

- Cause a noticeable performance loss.
- Result in prolonged operation at the specified maximum junction temperature and affect the long-term reliability of the processor.
- May be incapable of cooling the processor even when the TCC is active continuously (in extreme situations).

4.2.2.3.4 Low-Power States and PROCHOT_N Behavior

Depending on package power levels during package C-states, outbound PROCHOT_N may de-assert while the processor is idle as power is removed from the signal. Upon wake up, if the processor is still hot, the PROCHOT_N will re-assert, although typically package idle state residency should resolve any thermal issues.

4.2.2.4 THRMTRIP_N Signal

Thermal trip typically occurs 15-20C higher than maximum junction temperature. Regardless of enabling the automatic or on-demand modes, in the event of a catastrophic cooling failure, the package will automatically shut down when the silicon has reached an elevated temperature that risks physical damage to the product. At this point the THRMTRIP_N signal will go active.

4.2.2.5 Critical Temperature Detection

Critical Temperature detection is performed by monitoring the package temperature. This feature is intended for graceful shutdown before the THRMTRIP_N is activated. However, the processor execution is not guaranteed between critical temperature and THRMTRIP_N. If the Adaptive Thermal Monitor is triggered and the temperature remains high, a critical temperature status and sticky bit are latched in the PACKAGE_THERM_STATUS (0x1B1) MSR and the condition also generates a thermal interrupt, if enabled.

4.2.2.6 On-Demand Mode

The processor provides an auxiliary mechanism that allows system software to force the processor to reduce its power consumption using clock modulation. This mechanism is referred to as “On-Demand” mode and is distinct from Adaptive Thermal Monitor and bi-directional PROCHOT_N. The processor platforms should not rely on software usage of this mechanism to limit the processor temperature. On-Demand Mode can be accomplished using processor MSR or chipset I/O emulation. On-Demand Mode may be used in conjunction with the Adaptive Thermal Monitor. However, if the

system software tries to enable On-Demand mode at the same time the TCC is engaged, the factory configured duty cycle of the TCC will override the duty cycle selected by the On-Demand mode. If the I/O based and MSR-based On-Demand modes are in conflict, the duty cycle selected by the I/O emulation-based On-Demand mode will take precedence over the MSR-based On-Demand Mode.

4.2.2.7 MSR Based On-Demand Mode

If Bit 4 of the IA32_CLOCK_MODULATION MSR is set to 1, the processor will immediately reduce its power consumption using modulation of the internal processor IA core clock, independent of the processor temperature. The duty cycle of the clock modulation is programmable using bits [3:0] of the same IA32_CLOCK_MODULATION MSR. In this mode, the duty cycle can be programmed in 6.25% increments. Thermal throttling using this method will modulate each processor IA core's clock independently.

4.2.2.8 I/O Emulation-Based On-Demand Mode

I/O emulation-based clock modulation provides legacy support for operating system software that initiates clock modulation through I/O writes to ACPI defined processor clock control registers on the chipset (PROC_CNT). Thermal throttling using this method will modulate all processor IA cores simultaneously.

4.2.2.9 Thermal Throttling Disabled Disclaimers

With thermal throttling disabled, the processor will not automatically throttle its performance when usage results in temperature exceeding the thermal conditions defined in Table 4-1.

The processor is only recommended by Intel for use within the operating conditions defined in this document published by Intel and available on https://www.intel.com/content/www/us/en/resources-documentation/developer.html (which Intel may update from time-to-time).

The processor usage in a manner that exceeds the thermal conditions defined in this document may result in asserting the THRMTRIP_N signal and reduced processor lifetime.

Burst mode usage will increase the risk of the processor exceeding the thermal conditions defined in this document and should only be used with an appropriate form of thermal control management.

The use or operation of this processor outside any of the defined specifications is at your own risk and automatically voids any applicable Intel warranty.

4.2.3 Intel® Memory Thermal Management

The processor provides thermal protection for system memory by throttling memory traffic when using either DIMM modules or a memory down implementation. Two levels of throttling are supported by the processor, either a warm threshold or hot threshold that is customizable through memory mapped I/O registers. Throttling based on the warm threshold should be an intermediate level of throttling. Throttling based on the hot threshold should be the most severe. The amount of throttling is dynamically controlled by the processor.

4.2.4 Dynamic Temperature Range (DTR)

For a single operational cycle, the processor shall execute at full data sheet performance across the full Dynamic Temperature Range (DTR) without requiring a cold reset (S0 to S5 to S0 state transition). The processor DTR is:

SKU	Symbol	Parameter	Temperature Range (°C)		Notes
			Min	Max	
PC Client	DTR	Dynamic Temperature Range	T_{BOOT}-70	T_{BOOT}+70	1
Embedded			T_{BOOT}-90	T_{BOOT}+90	1
Industrial			T_{BOOT}-90	T_{BOOT}+90	1
			T_{BOOT}-110	T_{BOOT}+110	1, 2

Notes:

1. DTR is the range of Tj (Junction Temperature) starting from boot (T_{BOOT}) and transitioning Cold-to-Hot (T_{BOOT} + DTR) and/or Hot-to-Cold (T_{BOOT} - DTR). A Tj outside of the DTR range requires a cold reset but is not enforced by the hardware.

2. A DTR of ±110C is supported only when certain requirements are met. Please contact your Intel representative to obtain details of these requirements.

4.3 PCH Thermal Management

4.3.1 PCH Thermal Sensor

The PCH incorporates one on-die Digital Thermal Sensors (DTS) for thermal management.

4.3.2 Modes of Operation

The DTS has two usages when enabled, as follows:

1. Provide the PCH temperature in units of 1°C to a external temperature monitoring device.
2. Allow programmed trip points to cause alerts via an interrupt (SCI and SMI) or shut down the system (unconditionally transitions the system to S5) with a programmable catastrophic trip point.

4.3.3 Thermal Reporting to an External Device

To support an external device that is managing the system thermals, the PCH provides the ability for an external device to read the PCH temperature over eSPI interface. The external device will issue an eSPI OOB Channel request and receives a single byte of data, indicating a temperature between 0°C and 127°C, where 255 (0xFF) indicates that the sensor is not enabled yet.

Temperature Trip Point:
The internal thermal sensor reports three trip points: Cool, Hot, and Catastrophic trip points in the order of increasing temperature.

Crossing the cool trip point when going from higher to lower temperature may generate an interrupt. Crossing the hot trip point going from lower to higher temp may generate an interrupt. Each trip point has control register bits to select what type of interrupt is generated.

Crossing the cool trip point while going from low to higher temperature or crossing the hot trip point while going from high to lower temperature will not cause an interrupt.

When triggered, the catastrophic trip point will transition the system to S5 unconditionally. The register below is used to enable catastrophic assertion into S5 state. This bit should always be set in all functional cases.

Address Offset: 150Ch

Bit	Access	Default	Description
31	RWLO	0x0	**Policy Lock-Down Bit (CTENLOCK):** When written to 1, this bit prevents any more writes to this register
30:1	RO	0x0	**Reserved**

The thermal alert provides built in hysteresis, by having both a high and a low mark. An example of how it works is explained below:

- Both high and low marks are programmed to their correct values
  - — Assume, for an example, the high value is 90°C, and the low value is 80°C.
- TS is enabled, and assume temperature is at ambient (50°C)
  - — thus the alert signal is de-asserted
- temperature starts to rise as traffic flows through PCH
- temperature reaches greater than 90°C
  - — alert signal is asserted
  - — based on programming a platform indication like SMI, or SCI can occur if SW had enabled such
- temperature reaches 95°C
  - — alert signal remains asserted
- temperature starts to fall and reaches 85°C
  - — alert signal remains asserted because it has not reached less than 80°C, which is the value to turn off alert
- temperature falls to less than 80°C
  - — alert is turned off now since the temperature has fallen to the low value
  - — based on programming a platform indication like SMI, or SCI can occur if SW had enabled such
- temperature starts rising again and goes up to 85°C
  - — alert remains off until temperature rises to the high mark of greater than 90°C

An example of how SW can use the hysteresis would be to program a value for when the fans should be turned up, or cooling should be increased (90°C in example above), then allow the cooling to be sufficient that the extra cooling can be reduced (80°C). This prevents the PCH from oscillating around one temperature with the fans increasing/decreasing every few seconds. Using the hysteresis allows the fans to be on

or off for much longer periods.

Table 4-2. Thermal Trip Points and Response (Typical)

Zone	Nominal Trip Points	Response
Catastrophic	TCatastrophic (fused catastrophic temp value)	Halt Operation required (e.g. going to S5 State)
Hot	Threshold On = value set by OEM	SW Response recommended; e.g., turn fans up
Cool	Threshold Off = value set by OEM	SW Response recommended; e.g. turn fans down

4.3.4 Thermal Sensor Accuracy ($T_{accuracy}$)

The PCH thermal sensor accuracy is:

- ±5°C over the temperature range from 50°C to 110°C.
- ±7°C over the temperature range from 30°C to 50°C.
- ±10°C over the temperature range from -10°C to 30°C.
- The sensor itself is functional, from -40°C to 130°C, no accuracy is specified for temperature range beyond 110°C or below -10°C.

4.3.5 Thermal Trip Signal (PCHHOT_N)

The PCH provides PCHHOT_N signal to indicate that it has exceeded some temperature limit. The limit is set by BIOS. The temperature limit (programmed into the PHL register) is compared to the present temperature. If the present temperature is greater than the PHL value then the pin is asserted.

PCHHOT_N is an O/D output and requires a Pull-up on the motherboard.

The PCH evaluates the temperature from the thermal sensor against the programmed temperature limit every 1 second.

§ §

5 Memory

5.1 System Memory Interface

Intel Atom® x6000E Series processors, Intel® Pentium® and Celeron® N and J Series processors memory controller can support DDR4 and LPDDR4/4x technologies. The memory system supports memory configuration 1x32 LPDDR4/4x, 2x32 LPDDR4/4x, 4x32 LPDDR4/4x, 1x64 DDR4 and 2x64 DDR4. Intel Atom® x6000E Series processors, Intel® Pentium® and Celeron® N and J Series processors has In-band ECC module which helps improve the safety and reliability by providing ECC protection to specific regions of physical memory space. Out of band ECC is not supported by the memory controller.

The table in this section describe the details of the supported configuration matrix.

Table 5-1. DDR Support Matrix Table

Feature	LPDDR4/4x	DDR4	DDR4
		SODIMM	Memory Down
Max Frequency (MT/s)	SKUs 1-2 & 7: 3733 SKUs 3-6 & 8-9: 3200 SKUs 10-11: 4267 SKU 12: 2400	SKUs 1-11: 3200 SKU 12: 2400	SKUs 1-11: 3200 SKU 12: 2400
Voltage	LPDDR4/4x Processor VDDQ is 1.1V. LPDDR4 DRAM VDDQ voltage is 1.1V, VDD2 is 1.1V LPDDR4x DRAM VDDQ voltage is 0.6V, VDD2 is 1.1V	1.2v	1.2v
Channels	1 x 32bits 2 x 32bits 4 x 32bits	1 x 64 bits 2 x 64bits	1 x 64bits 2 x 64bits
DPC[1]	Not Applicable	1	Not Applicable
Maximum RPC[2]	1 (4267MT/s) 2 (3733MT/s)	2 (3200MT/s)	1 (3200MT/s)
IBECC	Yes	Yes	Yes
DRAM Die Density (Gb)	SDP - 16 DDP - 4,8 QDP - 8	8, 16	SDP - 8,16
Chip Level Density (Gb)	SDP - 16 DDP - 8 DDP - 16 QDP - 32	8 16	8 16
Channel Max Capacity (GB)	SDP - 2 DDP - 2 QDP - 4	32	8
System Max Capacity (GB)	SDP - 8 DDP - 8 (1 RPC) QDP - 16 (2 RPC)	32	16
Ball Count	200	Not Applicable	96
Ballmap Mode	Non-Interleave	Non-Interleave	Non-Interleave

Notes:
1. DPC refer to when only 1DIMM slot per channel is routed.
2. RPC = Rank Per Channel

5.1.1 DRAM Channel Support Matrix and Signals Terminology

Table 5-2. LPDDR4/4x Sub-Channels Population Rules

Parameter	Sub-Channel Population
Population rules	1 DRAM - DRAM 0 is connected to Sub Channel 0 **or** DRAM 0 is connected to Sub Channel 2
	2 DRAMs - DRAM 0 is connected to Sub Channel 0 DRAM 1 is connected to Sub Channel 1 **or** DRAM 0 is connected to Sub Channel 2 DRAM 1 is connected to Sub Channel 3 **or** DRAM 0 is connected to Sub Channel 0 DRAM 1 is connected to Sub Channel 2
	3 DRAMs - **N/A**
	4 DRAMs DRAM 0 is connected to Sub Channel 0 DRAM 1 is connected to Sub Channel 1 DRAM 2 is connected to Sub Channel 2 DRAM 3 is connected to Sub Channel 3

Table 5-3. DDR4 Channel Population Rules

Parameter	Channel Population
Population rules	Single Channel Connect DDR4 memory to Channel 0 or Channel 1
	Dual Channel Connect DDR4 memory to Channel 0 & Channel 1

Table 5-4. Supported SA Speed Enhanced Speed steps (SA-GV) and Gear Mode Frequencies

Data rate of Device MT/s	Memory Technology	Channel x Device width	Gear	SA-GV	MC CLK[6] (MHz)	Max Peak BW (GB/s)
1067	LPDDR4/4x	2x32, 4x32	Gear1	Low	1067	17.07
	DDR4	1x64, 2x64	Gear1	Low	1067	17.07
1600	LPDDR4/4x	2x32, 4x32	Gear1	Low	1600	25.6
	DDR4	1x64, 2x64	Gear1	Low	1600	25.6
1866	LPDDR4/4x	2x32, 4x32	Gear1	Low	1866	29.85
	DDR4	1x64, 2x64	Gear1	Low	1866	29.85
2133	LPDDR4/4x	2x32, 4x32	Gear2	Low	1067	34.12
	DDR4	1x64, 2x64	Gear2	Low	1067	34.12

Data rate of Device MT/s	Memory Technology	Channel x Device width	Gear	SA-GV	MC CLK[6] (MHz)	Max Peak BW (GB/s)
2400	LPDDR4/4x	2x32, 4x32	Gear2	Mid	1200	38.4
	DDR4	1x64, 2x64	Gear2	Mid	1200	38.4
2667	LPDDR4/4x	2x32, 4x32	Gear2	Mid	1333	42.67
	DDR4	1x64, 2x64	Gear2	Mid	1333	42.67
3200	LPDDR4/4x	2x32, 4x32	Gear2	Mid	1600	51.2
	DDR4	1x64, 2x64	Gear2	Mid	1600	51.2
3733	LPDDR4/4x	2x32, 4x32	Gear2	High	1866	59.72
	DDR4	1x64, 2x64	Not Supported			
4267[3]	LPDDR4/4x	2x32, 4x32	Gear2	High	2133	68.2
	DDR4	1x64, 2x64	Not Supported			

1. The processor supports dynamic gearing technology where the Memory Controller can run at 1:1 (Gear-1, Legacy mode) or 1:2 (Gear-2 mode) ratio of DRAM speed. Gear ratio is the ratio of DRAM speed to Memory Controller Clock.
   MC Channel Width equal to DDR Channel width multiply by Gear Ratio
2. SA-GV modes
   a. **Low-** Low frequency point, Min Power point. Characterized by low power, low BW, high latency. System will stay at this point during low to moderate BW consumption.
   b. **Mid -** Max Bandwidths Point, this point is the max possible BW point, the DRAM freq limited by Silicon Configuration/BIOS/SPD. Characterized by moderate power and latency, high BW. This point intended for high GT and moderate-high IA BW
   c. **High -** High Point, the minimum memory latency point, Characterized by high power, low latency, moderate BW. Only during IA performance workloads the system will to switch to this point and only in case this point can provide enough BW.
3. Dual rank is not supported for 4267 data rate of device (MT/s).
4. MC CLK is referring to the Memory Controller clock.

Table 5-5. Supported DDR4 SODIMM Module Configurations

Raw Card Version	Speed (Mt/s)	DIMM Capacity	DRAM Device Technology	DRAM Organization	# of DRAM Devices	#of Ranks	# of Row/Col Address bit	# of Banks inside DRAM	Page Size
A	3200	8GB	8Gb	1024M x 8	8	1	16/10	16	8K
A	3200	16GB	16Gb	2048M x 8	8	1	17/10	16	8K
C	3200	4GB	8Gb	512M x 16	4	1	16/10	8	8K
C	3200	8GB	16Gb	1024M x 16	4	1	17/10	8	8K
E	3200	16GB	8Gb	1024M x 8	16	2	16/10	16	8K
E	3200	32GB	16Gb	2048M x 8	16	2	17/10	16	8K

Table 5-6. Supported DDR4 DRAMs (Memory Down) Configurations

Max System Capacity	PKG Type (Die bits per Ch x PKG bits)	Die Density per Channel	PKG Density	Rank Per PKGs	Max Speed
8GB	SDP 16x16	8Gb	8Gb	1	3200(MT/s)
16GB	SDP 16x16	16Gb	16Gb	1	3200(MT/s)

Table 5-7. Supported LPDDR4/4x x32 DRAMs Configurations

Max System Capacity	PKG Type (Die bits per Ch x PKG bits)	Die Density	PKG Density	Rank Per PKGs	Max Speed
8GB	SDP 16x32	16Gb	16Gb	1	4267(MT/s)[1]
8GB	SDP 16x32	16Gb	16Gb	1	3200(MT/s)[2]
4GB	DDP 16x32	4Gb	8Gb	1	4267(MT/s)[1]
8GB	DDP 16x32	8Gb	16Gb	1	4267(MT/s)[1]
4GB	DDP 16x32	4Gb	8Gb	2	3733(MT/s)[1]
8GB	DDP 16x32	8Gb	16Gb	2	3733(MT/s)[1]
4GB	DDP 16x32	4Gb	8Gb	1	3200(MT/s)[2]
8GB	DDP 16x32	8Gb	16Gb	1	3200(MT/s)[2]
16GB	QDP 16x32	8Gb	32Gb	2	3200(MT/s)[2]

Notes:

1. Supported on Type-4 PCB technology only.
2. Supported on Type-3 and Type-4 PCB Technology

Table 5-8. System Memory Interface Signals Terminology (Sheet 1 of 4)

Memory Type	Description	Dir	DDR4 SODIMM (Per Channel)	LPDDR4/4x Memory Down (All Channels)
Signal details			-	-
Clock (CLK)	**SDRAM Differential Clock:** Differential clocks signal pairs, pair per channel and package. The crossing of the positive edge of CLK_DP and the negative edge of the complement CLP_DN are used to sample the command and control signals on the DRAM.	O	DDR_[1:0]_CLK[1:0]_DN, DDR_[1:0]_CLK[1:0]_DP	LP4_[3:0]_CLK_DN, LP4_[3:0]_CLK_DP

Table 5-8. System Memory Interface Signals Terminology (Sheet 2 of 4)

Control (CTRL)	**Chip Select:** (1 per rank). These signals are used to select particular DRAM components during the active state. There is one Chip Select for each DRAM rank. The Chip select signal is Active Low for DDR4 and Active High for LPDDR4/4.	O	DDR_[1:0]_CS[1:0]_N	LP4_[3:0]_CS[1:0]
	On Die Termination: (1 per rank). Active DRAM Termination Control.	O	DDR_[1:0]_ODT[1:0]	N/A
Clock Enable (CKE)	**Clock Enable:** (1 per rank) These signals are used to: • Initialize the DRAMs during power-up. • Power-down DRAM ranks. • Place all DRAM ranks into and out of self-refresh during STR	O	DDR_[1:0]_CKE[1:0]	LP4_[3:0]_CKE[1:0]

Table 5-8. System Memory Interface Signals Terminology (Sheet 3 of 4)

Command (CMD)	**Address (MA)**: These signals are used to provide the multiplexed row and column address to the DRAM. **Command Address (CA):** These signals are used to provide the multiplexed command and address to the DRAM.	O	DDR_[1:0]_MA[13:0], DDR_[1:0]_MA14_WE_N, DDR_[1:0]_MA15_CAS_N, DDR_[1:0]_MA16_RAS_N,	LP4_[3:0]_CA[5:0]
	Bank Group: BG[0:1] define to which bank group an Active, Read, Write or Precharge command is being applied. BG0 also determines which mode register is to be accessed during a MRS cycle.	O	DDR_[1:0]_BG[1:0]	
	Bank Address: BA[1:0] define to which bank an Active, Read, Write or Precharge command is being applied. Bank address also determines which mode register is to be accessed during a MRS cycle.	O	DDR_[1:0]_BA[1:0]	
	Activation Command: ACT# HIGH along with CS_N determines that the signals addresses have command functionality.	O	DDR_[1:0]_ACT_N	
	Command and Address Parity(PAR): These signals are used for parity check.	O	DDR_[1:0]_PAR	
Alert	**Alert:** This signal is used at command training only. It is getting the Command and Address Parity error flag during training. CRC feature is not supported.	I	DDR_[1:0]_ALERT_N	N/A
Strobe	**Data Strobes:** Differential data strobe pairs. The data is captured at the crossing point of DQS during read and write transactions.	I/O	DDR_[1:0]_DQS[7:0]_DN, DDR_[1:0]_DQS[7:0]_DP	LP4_[3:0]_DQS[3:0]_DN, LP4_[3:0]_DQS[3:0]_DP
Data	**Data Buses:** Data signals interface to the DRAM data buses.	I/O	DDR_[1:0]_DQ[63:00]	LP4_[3:0]_DQ[31:00]

Table 5-8. System Memory Interface Signals Terminology (Sheet 4 of 4)

Reset	**Memory Reset:** Signal is used to provide a reset signal to all connected DRAM.	O	PMC_DRAM_RESET_N	PMC_DRAM_RESET_N
RCOMP	**System Memory Resistance Compensation:** This signal needs to be externally terminated to VSS (refer to platform design guide for resistor value). This external resistor termination is used for resistance compensation of memory interface signal buffers.	Analog	DDR_RCOMP[2:0]	LP4_RCOMP[2:0]
Vref	**Memory Reference Voltage for Command and Address:** Signal is used to provide a CA reference voltage to all connected DRAM.	O	DDR_[1:0]_VREF_CA	-
VTT	**System Memory Power Gate Control:** When signal is high platform memory VTT regulator is enable, output high. When signal is low - Disables the platform memory VTT regulator in C8 and deeper and S3.	O	DDR_VTT_CTL	LP4_VTT_CTL

5.1.2 Memory Frequency

In all modes, the frequency of system memory is the lowest frequency of all memory modules placed in the system, as determined through the SPD registers on the memory modules. The system memory controller supports a single DIMM connector per channel. If DIMMs with different latency are populated across the channels, the BIOS will use the slower of the two latencies for both channels. For Dual-Channel modes both channels should have a DIMM connector populated. For Single-Channel mode, only a single channel can have a DIMM connector populated.

5.1.3 Technology Enhancements of Intel® Fast Memory Access (Intel® FMA)

The following sections describe the Just-in-Time Scheduling, Command Overlap, and Out-of-Order Scheduling Intel FMA technology enhancements.

Just-in-Time Command Scheduling

The memory controller has an advanced command scheduler where all pending requests are examined simultaneously to determine the most efficient request to be issued next. The most efficient request is picked from all pending requests and issued to system memory Just-in-Time to make optimal use of Command Overlapping. Thus,

instead of having all memory access requests go individually through an arbitration mechanism forcing requests to be executed one at a time, they can be started without interfering with the current request allowing for concurrent issuing of requests. This allows for optimized bandwidth and reduced latency while maintaining appropriate command spacing to meet system memory protocol.

Command Overlap

Command Overlap allows the insertion of the DRAM commands between the Activate, Pre-charge, and Read/Write commands normally used, as long as the inserted commands do not affect the currently executing command. Multiple commands can be issued in an overlapping manner, increasing the efficiency of system memory protocol.

Out-of-Order Scheduling

While leveraging the Just-in-Time Scheduling and Command Overlap enhancements, the IMC continuously monitors pending requests to system memory for the best use of bandwidth and reduction of latency. If there are multiple requests to the same open page, these requests would be launched in a back to back manner to make optimum use of the open memory page. This ability to reorder requests on the fly allows the IMC to further reduce latency and increase bandwidth efficiency

5.1.4 Data Scrambling

The system memory controller incorporates a Data Scrambling feature to minimize the impact of excessive di/dt on the platform system memory VRs due to successive 1s and 0s on the data bus. Past experience has demonstrated that traffic on the data bus is not random and can have energy concentrated at specific spectral harmonics creating high di/dt which is generally limited by data patterns that excite resonance between the package inductance and on die capacitances. As a result the system memory controller uses a data scrambling feature to create pseudo-random patterns on the system memory data bus to reduce the impact of any excessive di/dt.

5.1.5 Platform Memory System Configuration

The processor line package is optimized only for Non-Interleaving mode (NIL).

Figure 5-1. Intel Atom® x6000E Series processors, Intel® Pentium® and Celeron® N and J Series processors NIL Memory Down Side By Side Platform Configuration

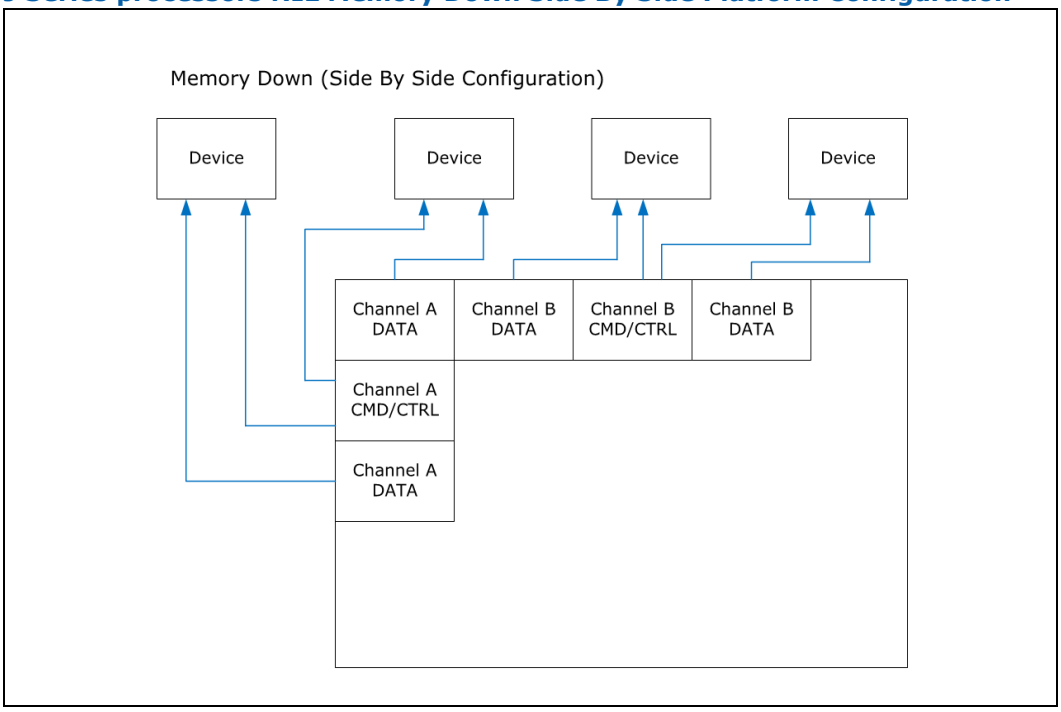
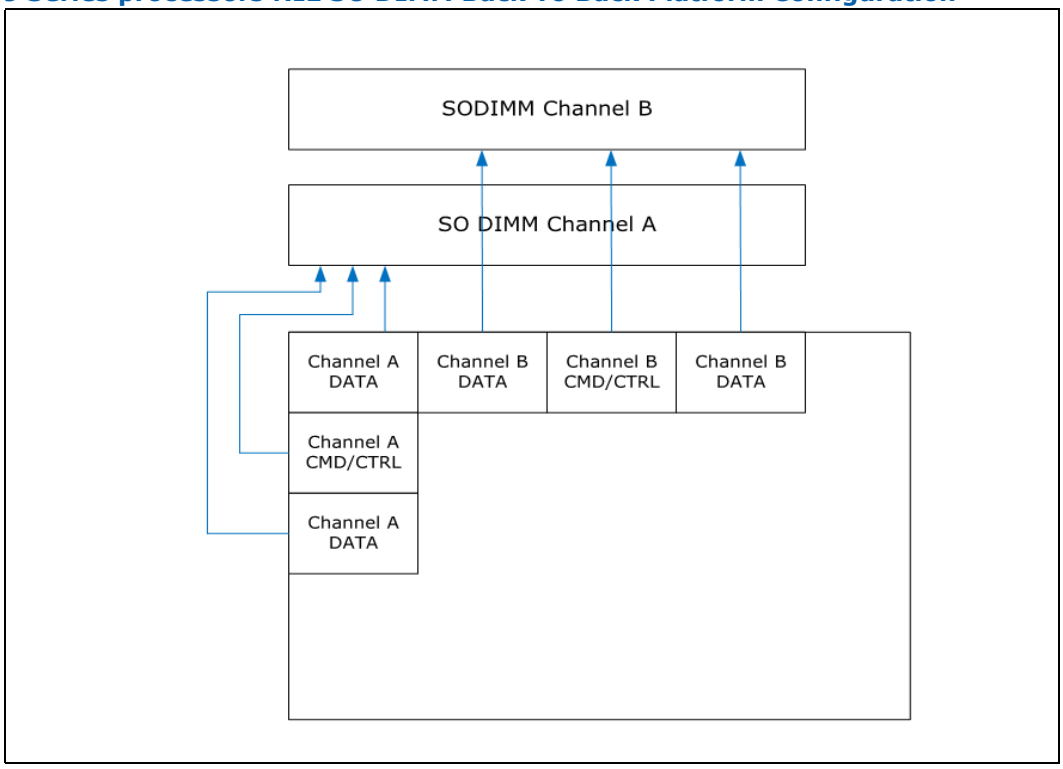


Figure 5-2. Intel Atom® x6000E Series processors, Intel® Pentium® and Celeron® N and J Series processors NIL SO DIMM Back To Back Platform Configuration



5.1.6 Data Swapping

By default, the processor supports on-board data swapping in two manners (for all segments and DRAM technologies):

- byte (DQ+DQS) swapping between bytes in the same channel.
- bit swapping within specific byte.

5.1.7 DRAM Clock Generation

Every supported rank has a differential clock pair. There are a total of four clock pairs driven directly by the processor to DRAM.

5.1.8 DRAM Reference Voltage Generation

The memory controller has the capability of generating the LPDDR4 and DDR4 Reference Voltage (VREF) internally for both read and write operations. The generated VREF can be changed in small steps, and an optimum VREF value is determined for both during a cold boot through advanced training procedures in order to provide the best voltage to achieve the best signal margins.

5.2 Power Management

The main memory is power managed during normal operation and in low-power ACPI C-states.

5.2.1 Disabling Unused System Memory Outputs

Any system memory (SM) interface signal that goes to a memory in which it is not connected to any actual memory devices (such as SODIMM connector is unpopulated, or is single-sided) is tri-stated. The benefits of disabling unused SM signals are:

- Reduced power consumption.
- Reduced possible overshoot/undershoot signal quality issues seen by the processor I/O buffer receivers caused by reflections from potentially un-terminated transmission lines.

When a given rank is not populated, the corresponding control signals (CLK_DP/CLK_DN/CKE/ODT/CS) are not driven.

At reset, all rows should be assumed to be populated, until it can be proven that they are not populated. This is due to the fact that when CKE is tri-stated with a DRAMs present, the DRAMs are not ensured to maintain data integrity. CKE tri-state should be enabled by BIOS where appropriate, since at reset all rows should be assumed to be populated.

5.2.2 DRAM Power Management and Initialization

The processor implements extensive support for power management on the memory interface. Each channel drives up to 2 CKE pins, one per rank.

The CKE is one of the power-saving means. When CKE is off, the internal DDR clock is disabled and the DDR power is reduced. The power-saving differs according to the selected mode and the DDR type used. For more information, refer to the IDD table in the DDR specification.

The processor supports four different types of power-down modes in package C0 state. The different power-down modes can be enabled through configuring PM PDWN config register. The type of CKE power-down can be configured through PDWN_mode (bits 15:12) and the idle timer can be configured through PDWN_idle_counter (bits 11:0). The different power-down modes supported are:

- **No power-down** (CKE disable)
- **Active power-down (APD):** This mode is entered if there are open pages when de-asserting CKE. In this mode the open pages are retained. Power-saving in this mode is the lowest. Power consumption of DDR is defined by IDD3P. Exiting this mode is fined by tXP – small number of cycles. For this mode, DRAM DLL should be on.
- **PPD/DLL-off:** In this mode the data-in DLLs on DDR are off. Power-saving in this mode is the best among all power modes. Power consumption is defined by IDD2P. Exiting this mode is defined by tXP, but also tXPDLL (10–20 according to DDR type) cycles until first data transfer is allowed. For this mode, DRAM DLL should be off.
- **Precharged power-down (PPD):** This mode is entered if all banks in DDR are pre-charged when de-asserting CKE. Power-saving in this mode is intermediate – better than APD, but less than DLL-off. Power consumption is defined by IDD2P. Exiting this mode is defined by tXP. The difference from APD mode is that when waking-up, all page-buffers are empty.) The LPDDR does not have a DLL. As a result, the power savings are as good as PPD/DDL-off but will have lower exit latency and higher performance.

The CKE is determined per rank, whenever it is inactive. Each rank has an idle counter. The idle-counter starts counting as soon as the rank has no accesses, and if it expires, the rank may enter power-down while no new transactions to the rank arrives to queues. The idle-counter begins counting at the last incoming transaction arrival.
It is important to understand that since the power-down decision is per rank, the IMC can find many opportunities to power down ranks, even while running memory intensive applications; the savings are significant (may be few Watts, according to DDR specification). This is significant when each channel is populated with more ranks.
Selection of power modes should be according to power-performance or thermal tradeoff of a given system:
• When trying to achieve maximum performance and power or thermal consideration is not an issue: use no power-down
• In a system which tries to minimize power-consumption, try using the deepest power-down mode possible – PPD/DLL-off with a low idle timer value
• In high-performance systems with dense packaging (that is, tricky thermal design) the power-down mode should be considered in order to reduce the heating and avoid DDR throttling caused by the heating.
The default value that BIOS configures in PM PDWN config register is 6080 – that is, PPD/DLL-off mode with idle timer of 0x80 (128 DCLKs). This is a balanced setting with deep power-down mode and moderate idle timer value.
The idle timer expiration count defines the # of DCLKs that a rank is idle that causes entry to the selected power mode. As this timer is set to a shorter time the IMC will have more opportunities to put the DDR in power-down. There is no BIOS hook to set this register. Customers choosing to change the value of this register can do it by changing it in the BIOS. For experiments, this register can be modified in real time if BIOS does not lock the IMC registers.

5.2.2.1 Initialization Role of CKE

During power-up, CKE is the only input to the SDRAM that has its level recognized (other than the reset pin) once power is applied. It should be driven LOW by the DDR controller to make sure the SDRAM components float DQ and DQS during power-up. CKE signals remain LOW (while any reset is active) until the BIOS writes to a configuration register. Using this method, CKE is ensured to remain inactive for much longer than the specified 200 micro-seconds after power and clocks to SDRAM devices are stable.

5.2.2.2 Conditional Self-Refresh

During S0 idle state, system memory may be conditionally placed into self-refresh state when the processor is in package C3 or deeper power state. When entering the S3 – Suspend-to-RAM (STR) state or S0 conditional self-refresh, the processor IA core flushes pending cycles and then enters SDRAM ranks that are not used by the processor graphics into self-refresh. The CKE signals remain LOW so the SDRAM devices perform self-refresh.
The target behavior is to enter self-refresh for package C3 or deeper power states aslong as there are no memory requests to service.

5.2.2.3 Dynamic Power-Down

Dynamic power-down of memory is employed during normal operation. Based on idle conditions, a given memory rank may be powered down. The IMC implements aggressive CKE control to dynamically put the DRAM devices in a power-down state. The processor IA core controller can be configured to put the devices in active powerdown (CKE de-assertion with open pages) or precharge power-down (CKE de-assertion with all pages closed).
Precharge power-down provides greater power savings but has a bigger performance impact, since all pages will first be closed before putting the devices in power-down mode.
If dynamic power-down is enabled, all ranks are powered up before doing a refresh cycle and all ranks are powered down at the end of refresh.

5.2.3 DDR Electrical Power Gating

The DDR I/O of the processor supports Electrical Power Gating (DDR-EPG) while the processor is at C3 or deeper power state.
In C3 or deeper power state, the processor internally gates VDDQ for the majority of the logic to reduce idle power while keeping all critical DDR pins such as CKE and VREF in the appropriate state.
In C7 or deeper power state, the processor internally gates V_{CCIO} for all non-critical state to reduce idle power.
In S3 or C-state transitions, the DDR does not go through training mode and will restore the previous training information.

5.2.4 Power Training

BIOS MRC performing Power Training steps to reduce DDR I/O power while keeping reasonable operational margins still guaranteeing platform operation. The algorithms attempt to weaken ODT, driver strength and the related buffers parameters both on the MC and the DRAM side and find the best possible trade-off between the total I/O power and the operational margins using advanced mathematical models.

5.3 IBECC

5.3.1 Introduction

The In-Band Error Correction Code (IBECC) module improves accuracy and reliability by providing error check and correct protection to all or specific regions of the physical memory space. The IBECC can be enabled for memory technology that do not support the out-of-band ECC, where the cost of adding an additional device to each channel for ECC data storage is prohibitive.

5.3.2 IBECC Transaction

5.3.2.1 Functionality Overview

The IBECC recognizes whether a region should protected based on the incoming request address. The IBECC will allow up to eight different address regions to be protected. All the regions and the corresponding ECC space will have to be configured at boot.

The IBECC will protect data at a cache line granularity (64 Bytes), with a 16-bit SECDED code. An ECC data cache line will contain the ECC value of 32 data (non-ECC) cache lines. So, the IBECC will add a memory overhead of 1/32, if the entire memory is protected by ECC. However for simplicity, when the IBECC is enabled, 1/32 of the TOUUD size must be reserved for ECC storage regardless of the size and number of the protected regions. BIOS must ensure that a sufficient overflow region is allocated to account for this overhead, and that the space is removed from usable DRAM address space. The IBECC can be enabled for LPDDR4/4x and DDR4 technologies. Nevertheless, all populated memory channels need to be of the same size.

The IBECC converts a read/write transaction (cache line access) to a protected region of memory into two separate memory requests (read/write), one to the actual data cache line and another to the cache line containing the ECC value. The IBECC also needs to ensure that all protected transaction pairs are issued atomically with no intervening transaction in between. Based on the incoming read/write address the IBECC determines the address of the ECC data corresponding to that cache line.

5.3.2.2 Protected Read

For protected reads, the IBECC:

- Ensures that both the data read and the ECC read transaction are issued atomically.
- Tracks when both reads have completed.
- Uses its ECC unit to detect and correct for any errors.
- Forwards the post-correction data to the requesting agent

5.3.2.3 Protected Full Writes

For protected, full writes, the IBECC needs to ensure that it:

- Generates the ECC value for the write, using the ECC generation logic with the appropriate byte enables set.
- Atomically issues both the data write and the ECC data partial write.

5.3.2.4 Protected Partial Writes

For protected, partial writes, the IBECC needs to:

- Issue a protected, read transaction for the underfill.
- Do the merge with the original write data.
- Issue a protected, full write.

Note: In this case, for a protected partial write, the IBECC will have to issue 2 read and 2 write requests in total.

5.3.3 Distinguishing ECC Protected and Unprotected Traffic

The IBECC will support up to eight regions of ECC-protected space. The address for every incoming transaction will be compared against eight separate address ranges to determine whether that request is to an ECC-protected or non-protected region.

Each protected region size has to be power of 2 and must at least be 32 MB in size. The address range that it protects must be at the granularity/aligned to the size. For example if the IBECC needs to protect 256MB of physical memory then the ECC_PROTECT_ADDR_RANGE_[0:7] base will have to be have to be a multiple of 256MB. The base address of a particular region will be stored in ECC_PROTECT_ADDR_RANGE_[0:7].BASE register field. These bits are compared with the result of the ECC_PROTECT_ADDR_RANGE_[0:7].MASK applied to the incoming address to determine if an access falls within that specific protected range.

5.3.4 Recent Syndrome Buffer (RSB)

Since a single ECC cache line contains ECC syndromes for 32 data (non-ECC) cache lines and most high bandwidth workloads involve sequential access, the same ECC cache line may be repeatedly re-fetched from DRAM. In order to reduce the latency & bandwidth penalty associated with re-fetching the cache line from DRAM, recent ECC cache lines are buffered in the RSB. The RSB is able to store up-to 64 ECC cache lines in total, covering all active regions.

The real-life performance benefit of the RSB is completely dependent on the platform workload but the benefit for one firmware/software (FW/SW) initiated memory access can be summarized as follows:

Table 5-9. Firmware/Software Initiated Memory Access

	Firmware/Software Initiated Memory Access		
	Read	**Write**	**Partial Write**
0% RSB Hit	2x DRAM Accesses 1. Data Read 2. ECC Syndrome Read	3x DRAM Accesses 1. Data Write 2. Full ECC Syndrome Read 3. Full ECC Syndrome Write	4x DRAM Accesses 1. Full Data Read 2. Full ECC Syndrome Read 3. Full Data Write 4. Full ECC Syndrome Write
100% RSB Hit	1x DRAM Access 1. Data Read	2x DRAM Accesses 1. Data Write 2. Masked Partial ECC Syndrome Write	3x DRAM Accesses 1. Full Data Read 2. Full Data Write 3. Masked Partial ECC Syndrome Write

5.3.5 ECC Error Reporting

IBECC logs Address, Syndrome and Type of error. There is only one event logged. IBECC always keep the first error till it is cleared by Software. The ECC_ERROR_LOG is valid only when the MERRSTS (Multiple Bit Error Status) or CERRSTS (Correctable Error Status).

The IBECC reports the ECC errors using the same flows as the memory controller.

The IBECC needs to identify ECC errors and report them. For the reported error, the IBECC needs to keep the following fields:

- CMI (Converged Memory Interface) Address
- Syndrome
- Type of error valid bits – correctable and uncorrectable

The IBECC will report all correctable and uncorrectable errors by sending PCH_EVENT message on IOSF-SB to IOP.

Note: IBECC errors are not reported as an MCE (Machine Check Error).

Logging – there is only one event logged. The IBECC shall always keep the worst first error. This means that:

- A correctable error is logged if no error was reported before.
- The first uncorrectable error may override previous correctable errors.
- Later uncorrectable errors do not override the first uncorrectable error.

The ECC_ERROR_LOG content are valid only when either the MERRSTS or CERRSTS bits are set.

An uncorrectable error on under fill read is not logged, but the WDB entry is poisoned so it is rewritten to memory as uncorrectable error.

Note: Since the IBECC module operates on the CMI data bus connecting the Processor Transaction Router and the Memory Controller, rather than on a per memory channel basis, it is not possible to correlate the error syndrome to the external data signal that is in error.

Please refer to Section 33.2.1.2 for a description of how ECC error reporting operates outside IBECC.

5.3.6 Error Injection

It is possible to inject ECC errors in order to check the IBECC mechanism. ECC errors are injected on the write path in order to cause ECC error behavior on the read path.

§ §

6 Mapping Address Spaces

This chapter describes how the memory & IO spaces are mapped to interfaces in the processor.

6.1 System Address Mapping

The compute die supports 512 GB (39 bits) of addressable memory space and 64 KB+3 of addressable I/O space.

This section focuses on how the memory space is partitioned and how the separate memory regions are used. I/O address space has simpler mapping and is explained towards the end of this chapter.

The compute die supports PCIe* port upper prefetchable base/limit registers. This allows the PCIe* bridges to claim Memory Mapped I/O (MMIO) accesses above 32 bit. Addressing of greater than 4 GB is allowed on both the OPI Interface. DRAM capacity is limited by the number of address pins available. There is no hardware lock to prevent more memory from being inserted than is addressable.

In the following sections, it is assumed that all of the compatibility memory ranges reside on the OPI Interface. The exception to this rule is VGA ranges, which may be mapped to OPI, or to the Processor Graphics device (Processor Graphics). The processor does not remap APIC or any other memory spaces above TOLUD (Top of Low Usable DRAM). The TOLUD register is set to the appropriate value by BIOS. The remapbase/remaplimit registers remap logical accesses bound for addresses above 4 GB onto physical addresses that fall within DRAM.

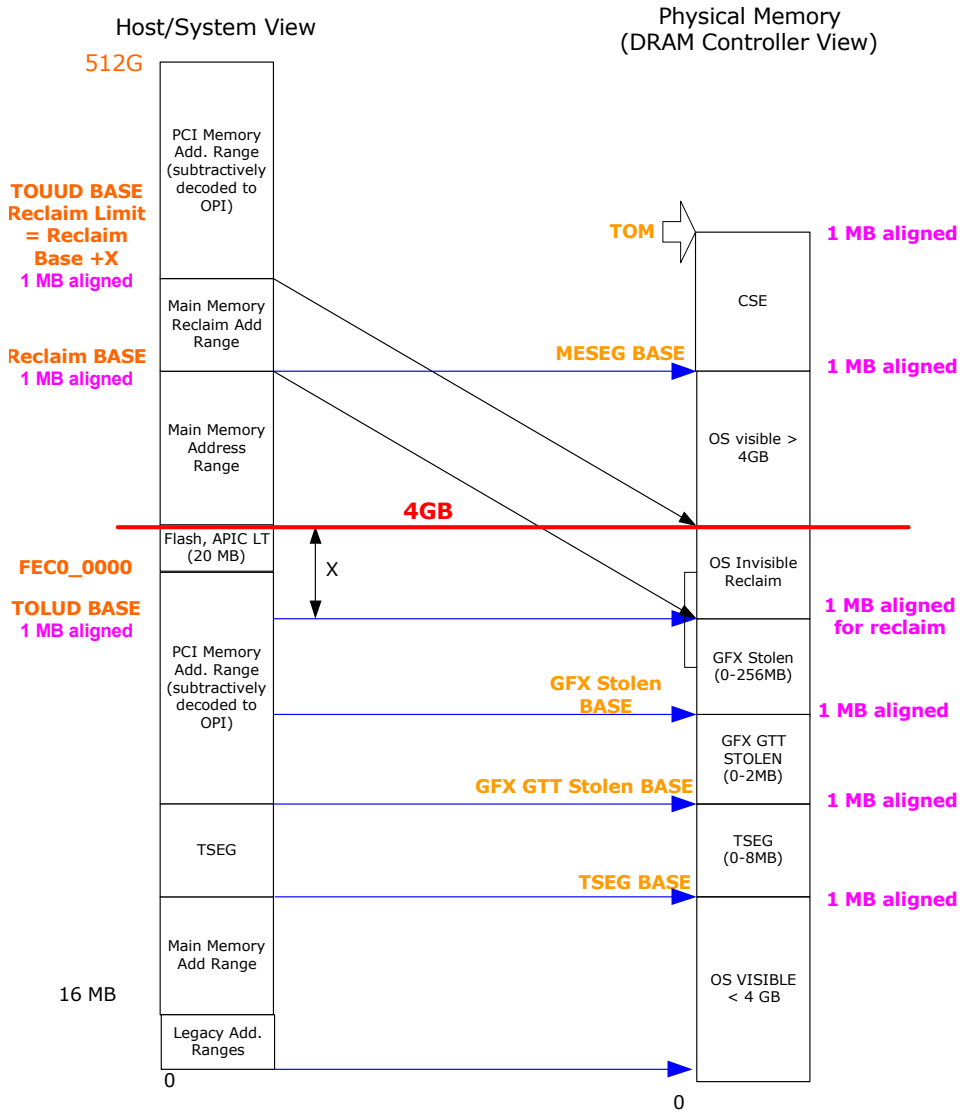
The Address Map includes a number of programmable ranges that are not configured using standard PCI BAR configuration:

- Device 0:
  - MCHBAR – Host Memory Mapped Configuration (memory subsystem and power management registers). (64 KB window)
  - DMIBAR – This window is used to access registers associated with the compute die/PCH Serial Interconnect (OPI) register memory range. (4 KB window).
  - VTDPVC0BAR - Memory mapped range for VT-d configuration
  - GFXVTBAR - Memory mapped range for VT configuration of the processor graphics device (4KB window).
  - REGBAR - Memory mapped range for Processor Transaction Router registers (16MB window).
  - GGC.GMS – Graphics Mode Select. Main memory that is pre-allocated to support the Processor Graphics device in VGA (non-linear) and Native (linear) modes. (0 – 512 MB options).
  - GGC.GGMS – GTT Graphics Memory Size. Main memory that is pre-allocated to support the Processor Graphics Translation Table. (0 – 2 MB options).
- For all other PCI devices within the Compute die that expose PCI configuration space, the behavior is according to PCI specification.

The rules for the above programmable ranges are:

1. For security reasons, the Compute die positively decodes (FFE0_0000h to FFFF_FFFFh) to OPI. This ensures the boot vector and BIOS execute off the PCH.
2. ALL of these ranges should be unique and NON-OVERLAPPING. It is the BIOS or system designer's responsibility to limit memory population so that adequate PCI, PCI Express*, High BIOS, PCI Express* Memory Mapped space, and APIC memory space can be allocated.
3. In the case of overlapping ranges with memory, the memory decode will be given priority.
4. There are NO Hardware Interlocks to prevent problems in the case of overlapping memory ranges.
5. Accesses to overlapped ranges may produce indeterminate results.
6. Peer-to-peer write cycles are allowed below the Top of Low Usable memory (register TOLUD) for OPI Interface to PCI Express* VGA range writes. Peer-to-peer cycles to the Processor Graphics VGA range are not supported.

Figure 6-1. System Address Range Example



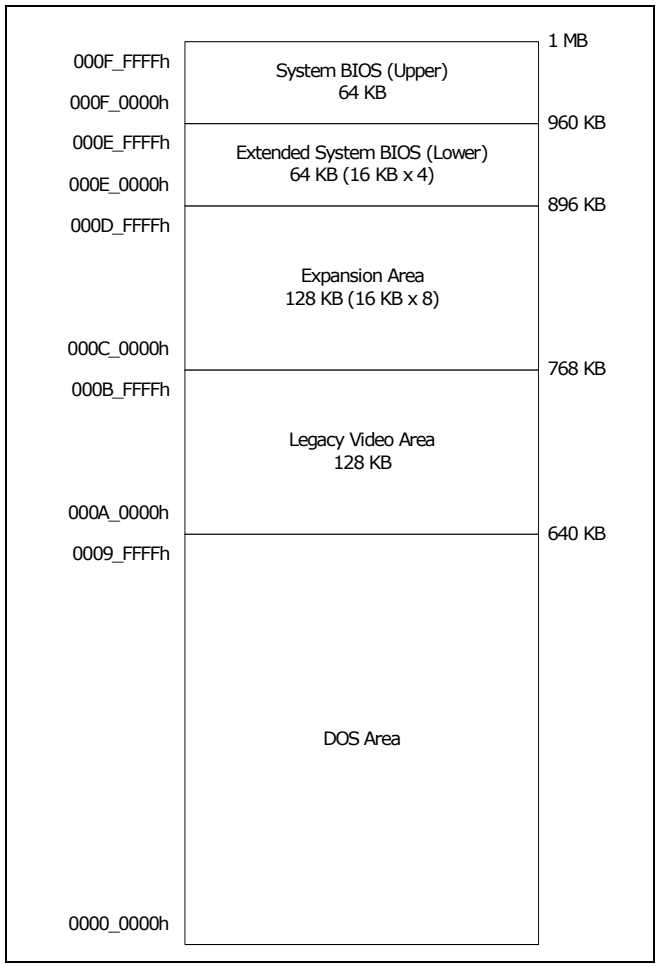
6.2 DOS Legacy Address Range

The memory address range from 0 to 1 MB is known as Legacy Address. This area is divided into the following address regions:

- 0 – 640 KB - DOS Area
- 640 – 768 KB - Legacy Video Buffer Area
- 768 – 896 KB in 16 KB sections (total of 8 sections) – Expansion Area
- 896 – 960 KB in 16 KB sections (total of 4 sections) – Extended System BIOS Area
- 960 KB – 1 MB Memory, System BIOS Area

The area between 768 KB – 1 MB is also collectively referred to as PAM (Programmable Address Memory). All accesses to the DOS and PAM ranges from any device are sent to DRAM. However, access to the legacy video buffer area is treated differently.

Figure 6-2. DOS Legacy Address Range



6.2.1 DOS Range (0h – 9_FFFFh)

The DOS area is 640 KB (0000_0000h – 0009_FFFFh) in size and is always mapped to the main memory.

6.2.2 Legacy Video Area (A_0000h – B_FFFFh)

The same address region is used for both Legacy Video Area.

- Legacy Video Area: The legacy 128 KB VGA memory range, frame buffer, at 000A_0000h – 000B_FFFFh, can be mapped to Processor Graphics (Device 2), and/or to the OPI Interface.
- Monochrome Adapter (MDA) Range: Legacy support for monochrome display adapter

Note: The legacy video area is not available for SMM use.

6.2.2.1 Legacy Video Area

The legacy 128 KB VGA memory range, frame buffer at 000A_0000h – 000B_FFFFh, can be mapped to Processor Graphics (Device 2) and/or to the OPI Interface.

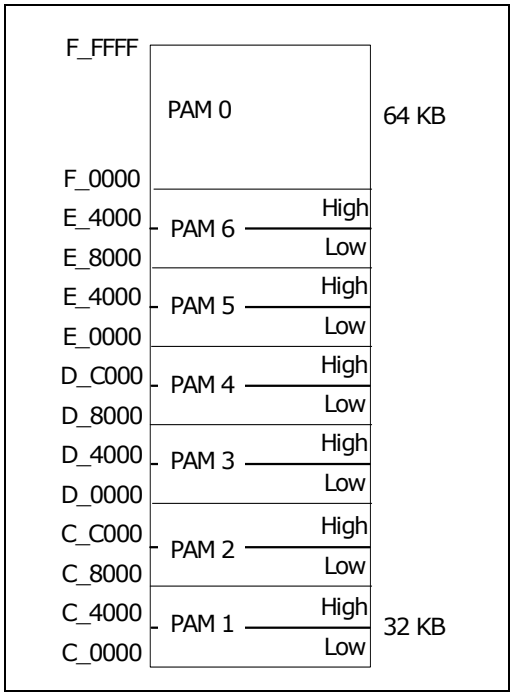
6.2.2.2 Monochrome Adapter (MDA) Range

Legacy support requires the ability to have a second graphics controller (monochrome) in the system. The monochrome adapter may be mapped to Processor Graphics (Device 2) and/or to the OPI Interface.

6.2.3 Programmable Attribute Map (PAM) (C_0000h – F_FFFFh)

PAM is a legacy BIOS ROM area in MMIO. It is overlaid with DRAM and used as a faster ROM storage area. It has a fixed base address (000C_0000h) and fixed size of 256 KB. The 13 sections from 768 KB to 1 MB comprise what is also known as the PAM Memory Area. Each section has Read enable and Write enable attributes.

Figure 6-3. PAM Region Space



The PAM registers are mapped in Device 0 configuration space.

- ISA Expansion Area (C_0000h – D_FFFFh)
- Extended System BIOS Area (E_0000h – E_FFFFh)
- System BIOS Area (F_0000h – F_FFFFh)

The processor decodes the Core request, then routes to the appropriate destination (DRAM or OPI).

Graphics translated requests to this region are not allowed. If such a mapping error occurs, the request will be routed to C_0000h. Writes will have the byte enables de-asserted.

6.3 Lower Main Memory Address Range (1 MB – TOLUD)

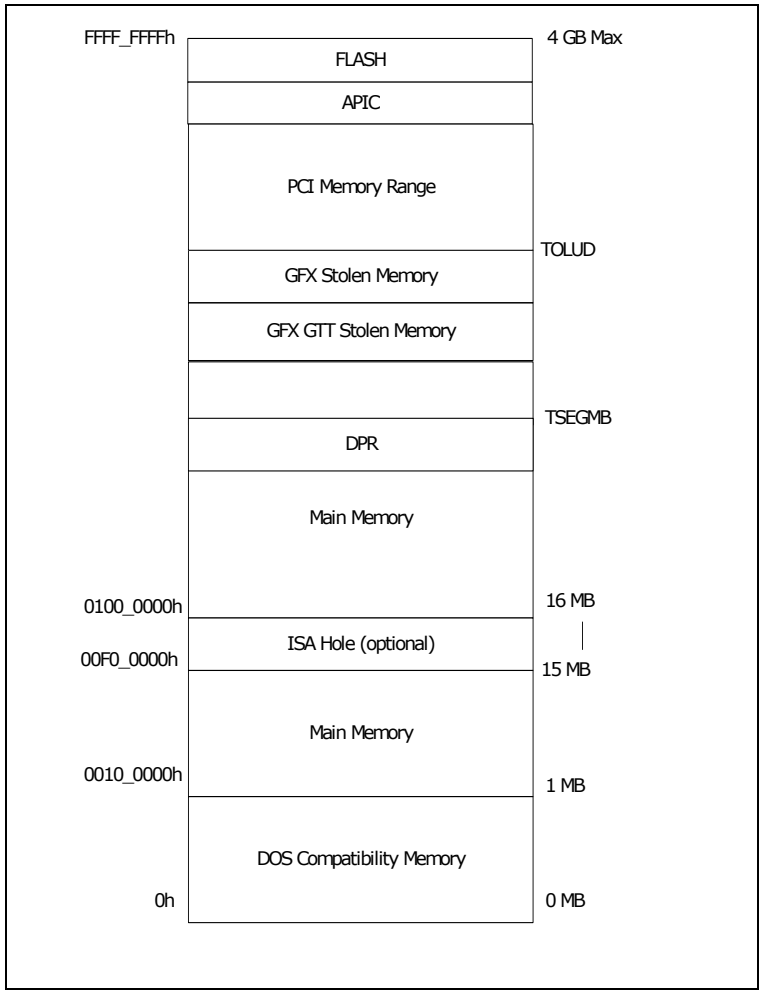
This address range extends from 1 MB to the top of Low Usable physical memory that is permitted to be accessible by the processor (as programmed in the TOLUD register). The processor will route all addresses within this range to the DRAM unless it falls into the optional TSEG, optional ISA Hole or optional Processor Graphics stolen memory.

This address range is divided into two sub-ranges:

- 1 MB to TSEGMB
- TSEGMB to TOULUD

TSEGMB indicates the TSEG Memory Base address.

Figure 6-4. Main Memory Address Range



6.3.1 ISA Hole (15 MB –16 MB)

The ISA Hole (starting at address F0_0000h) is enabled in the Legacy Access Control Register in Device 0 configuration space. If no hole is created, the compute die will route the request to DRAM. If a hole is created, the compute die will route the request to OPI.

Graphics translated requests to the range will always route to DRAM.

6.3.2 1 MB to TSEGMB

Compute die access to this range will be directed to memory with the exception of the ISA Hole (when enabled).

6.3.3 TSEG

For Compute Die initiated transactions, the Compute Die relies on correct programming of SMM Range Registers (SMRR) to enforce TSEG protection.

TSEG is below Processor Graphics stolen memory, which is at the Top of Low Usable physical memory (TOLUD). BIOS will calculate and program the TSEG BASE in Device 0 (TSEGMB), used to protect this region from DMA access. Calculation is:

TSEGMB = TOLUD – DSM SIZE – GSM SIZE – TSEG SIZE

SMM-mode compute die accesses to TSEG always access the physical DRAM.

When the extended SMRAM space is enabled, compute die accesses without SMM attribute or without write-back attribute to the TSEG range are handled as invalid accesses.

Non-compute die originated accesses such as PCI Express*, OPI or processor graphics to enabled SMM space are handled as invalid cycle type with reads and writes to location C_0000h and byte enables turned off for writes.

6.3.4 Protected Memory Range (PMR) - (Programmable)

To optimally support platform configurations supporting varying amounts of main memory, the protected memory region is defined as two non-overlapping regions:

- **Protected Low-memory Region**: This is defined as the protected memory region below 4 GB to hold the VMM code/private data, and the initial DMA-remapping structures that control DMA to host physical addresses below 4 GB.
- **Protected High-memory Region**: This is defined as a variable sized protected memory region above 4 GB, enough to hold the initial DMA-remapping structures for managing DMA accesses to addresses above 4 GB.

Once the protected low/high memory region registers are configured, bus initiator protection to these regions is enabled through the Protected Memory Enable register.

6.3.5 DRAM Protected Range (DPR)

This protection range only applies to DMA accesses and GMADR translations. It serves a purpose of providing a memory range that is only accessible to compute die streams. The range just below TSEGMB is protected from DMA accesses.

The DPR range works independently of any other range, including the PMRC checks in Intel VT-d. It occurs post any Intel VT-d translation. Therefore, incoming cycles are checked against this range after the Intel VT-d translation and faulted if they hit this protected range, even if they passed the Intel VT-d translation.

The system will set up:

- 0 to (TSEG_BASE – DPR size – 1) for DMA traffic
- TSEG_BASE to (TSEG_BASE – DPR size) as no DMA.

After some time, software could request more space for not allowing DMA. It will get some more pages and make sure there are no DMA cycles to the new region. DPR size is changed to the new value. When it does this, there should not be any DMA cycles going to DRAM to the new region.

All upstream cycles from 0 to (TSEG_BASE – 1 – DPR size), and not in the legacy holes (VGA), are decoded to DRAM.

6.3.6 Pre-allocated Memory

Voids of physical addresses that are not accessible as general system memory and reside within the system memory address range (< TOLUD) are created for SMM-mode, legacy VGA graphics compatibility, and GFX GTT stolen memory. **It is the responsibility of BIOS to properly initialize these regions**.

6.4 PCI Memory Address Range (TOLUD – 4 GB)

Top of Low Usable DRAM (TOLUD) – TOLUD is restricted to 4 GB memory (1MB granularity), but the Processor Transaction Router may support up to a much higher capacity, which is limited by DRAM.

This address range from the top of low usable DRAM (TOLUD) to 4 GB is normally mapped to the OPI Interface.

Device 0 exceptions are:

1.Addresses decoded to the memory mapped range for Host Memory Mapped Configuration Space registers (MCHBAR)

2. Addresses decoded to the registers associated with the PCH Serial Interconnect (OPI) register memory range. (DMIBAR)

In Processor Graphics configurations, there are exceptions to this rule:

3. Addresses decode to the Processor Graphics translation window (GMADR)

4. Addresses decode to the Processor Graphics translation table or Processor Graphics registers. (GTTMMADR)

In an Intel VT enabled configuration, there are exceptions to this rule:

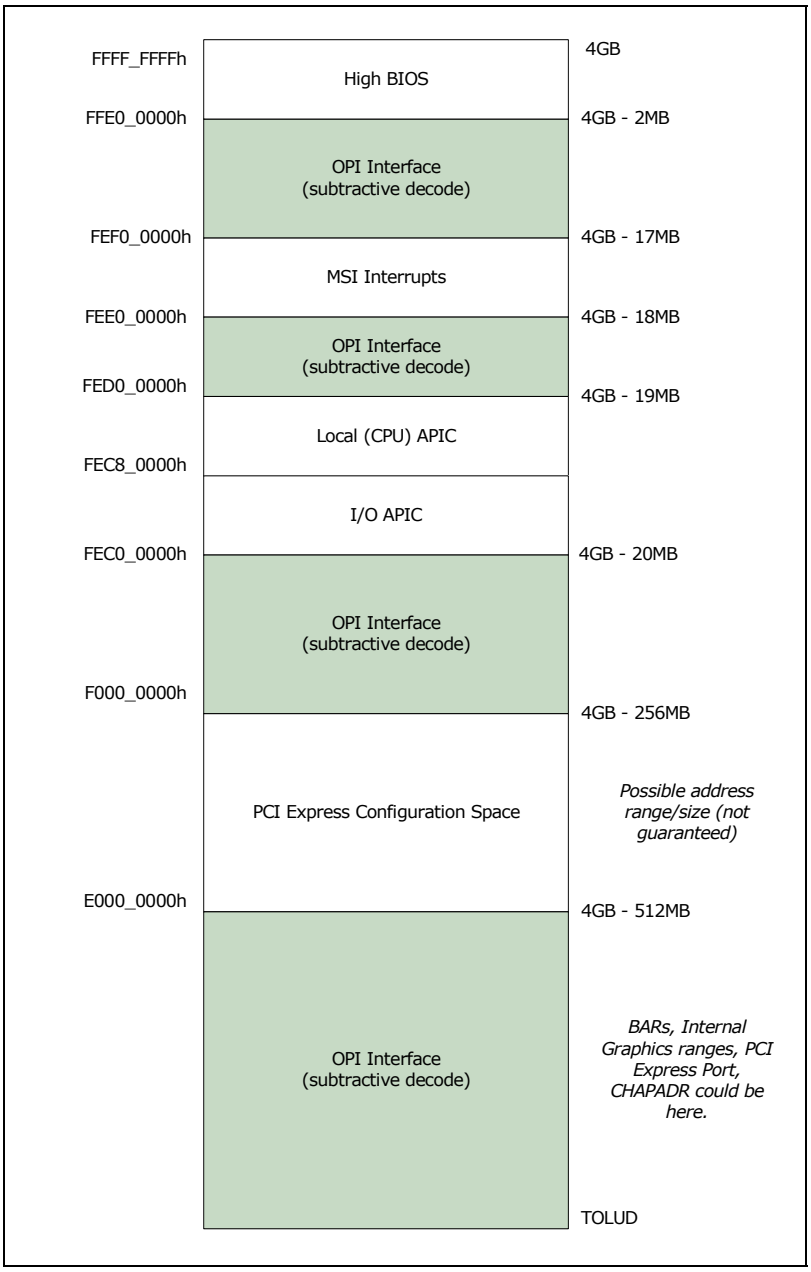
5. Addresses decoded to the memory mapped window to Graphics Intel® VT remap engine registers (GFXVTBAR)

6. Addresses decoded to the memory mapped window to OPI VC0 Intel® VT remap engine registers (VTDPVC0BAR)

Some of the MMIO Bars may be mapped to this range or to the range above TOUUD.

There are sub-ranges within the PCI memory address range defined as APIC Configuration Space, MSI Interrupt Space, and High BIOS address range. The exceptions listed above for Processor Graphics ***should NOT overlap with these ranges***.

Figure 6-5. PCI Memory Address Range



6.4.1 MSI Interrupt Memory Space (FEE0_0000h – FEEF_FFFFh)

Any device may issue a Memory Write to 0FEEx_xxxxh. This Memory Write cycle does not go to DRAM. The processor transaction router will forward this Memory Write along with the data to the processor as an Interrupt Message Transaction.

6.4.2 High BIOS Area

For security reasons, the compute die will positively decode this range to OPI. This positive decode ensures any overlapping ranges will be ignored. This ensures that the boot vector and BIOS execute off the PCH.

The top 2 MB (FFE0_0000h – FFFF_FFFFh) of the PCI Memory Address Range is reserved for System BIOS (High BIOS), extended BIOS for PCI devices, and the A20 alias of the system BIOS.

The processor begins execution from the High BIOS after reset. This region is positively decoded to OPI. The actual address space required for the BIOS is less than 2 MB. However, the minimum processor MTRR range for this region is 2 MB; thus, the full 2 MB should be considered.

6.5 Upper Main Memory Address Space (4 GB to TOUUD)

The maximum main memory size supported is 64 GB total DRAM memory.

A hole between TOLUD and 4 GB occurs when main memory size approaches 4 GB or larger. As a result, TOM and TOUUD registers and REMAPBASE/REMAPLIMIT registers become relevant.

The remap configuration registers exist to remap lost main memory space. The greater than 32-bit remap handling will be handled similar to other processors.

Upstream read and write accesses above 39-bit addressing will be treated as invalid cycles by OPI.

6.5.1 Top of Memory (TOM)

The "Top of Memory" (TOM) register reflects the total amount of populated physical memory. This is NOT necessarily the highest main memory address (holes may exist in main memory address map due to addresses allocated for memory mapped IO above TOM).

6.5.2 Top of Upper Usable DRAM (TOUUD)

The Top of Upper Usable DRAM (TOUUD) register reflects the total amount of addressable DRAM. If remap is disabled, TOUUD will reflect TOM .If remap is enabled, then it will reflect the remap limit.

6.5.3 Top of Low Usable DRAM (TOLUD)

TOLUD register is restricted to 4 GB memory (A[31:20]), but the processor can support up to 64 GB, limited by DRAM pins. For physical memory greater than 4 GB, the TOLUD register helps identify the address range between the 4 GB boundary and the top of

physical memory. This identifies memory that can be directly accessed (including remap address calculation) that is useful for memory access indication and early path indication. TOLUD can be 1 MB aligned.

6.5.4 TSEG_BASE

The "TSEG_BASE" register reflects the total amount of low addressable DRAM, below TOLUD. BIOS will calculate memory size and program this register; thus, the system agent has knowledge of where (TOLUD) – (Gfx stolen) – (Gfx GTT stolen) – (TSEG) is located. I/O blocks use this minus DPR for upstream DRAM decode.

6.5.5 Indirect Accesses to MCHBAR Registers

Similar to prior chipsets, MCHBAR registers can be indirectly accessed using:

- Direct MCHBAR access decode:
  - — Cycle to memory from Compute Die
  - — Hits MCHBAR base, AND
  - — MCHBAR is enabled, AND
  - — Within MMIO space (above and below 4 GB)
- GTTMMADR (10000h – 13FFFh) range -> MCHBAR decode:
  - — Cycle to memory from compute die, AND
  - — Device 2 (Processor Graphics) is enabled, AND
  - — Memory accesses for device 2 is enabled, AND
  - — Targets (refer to Section 1.1.1 for more information on target) GFX MMIO Function 0, AND
  - — MCHBAR is enabled or cycle is a read. If MCHBAR is disabled, only read access is allowed.
- MCHTMBAR -> MCHBAR (Thermal Monitor)
  - — Cycle to memory from compute die, AND
  - — Targets MCHTMBAR base
- IOBAR -> GTTMMADR -> MCHBAR.
  - — Follows IOBAR rules. Refer GTTMMADR information above as well.

6.5.6 Memory Remapping

An incoming address (referred to as a logical address) is checked to view if it falls in the memory re-map window. The bottom of the re-map window is defined by the value in the REMAPBASE register. The top of the re-map window is defined by the value in the REMAPLIMIT register. An address that falls within this window is re-mapped to the physical memory starting at the address defined by the TOLUD register. The TOLUD register should be 1 MB aligned.

6.5.7 Hardware Remap Algorithm

The following pseudo-code defines the algorithm used to calculate the DRAM address to be used for a logical address above the top of physical memory made available using re-claiming.

```
IF (ADDRESS_IN[38:20] >= REMAP_BASE[35:20]) AND
(ADDRESS_IN[38:20] <= REMAP_LIMIT[35:20]) THEN
    ADDRESS_OUT[38:20] = (ADDRESS_IN[38:20] - REMAP_BASE[35:20]) +
0000000b & TOLUD[31:20]
    ADDRESS_OUT[19:0] = ADDRESS_IN[19:0]
```

6.6 Graphics Memory Address Ranges

The integrated memory controller can be programmed to direct memory accesses to the Processor Graphics when addresses are within any of the ranges specified using registers in processor device 2 configuration space.

- The Graphics Memory Aperture Base Register (GMADR) is used to access graphics memory allocated using the graphics translation table.
- The Graphics Translation Table Base Register (GTTADR) is used to access the translation table and graphics control registers. This is part of the GTTMMADR register.

These ranges can reside above the Top-of-Low-DRAM and below High BIOS and APIC address ranges. They should reside above the top of memory (TOLUD) and below 4 GB so they do not take any physical DRAM memory space.

Alternatively, these ranges can reside above 4 GB, similar to other BARs that are larger than 32 bits in size.

GMADR is a Prefetchable range in order to apply USWC attribute (from the processor point of view) to that range. The USWC attribute is used by the processor for write combining.

6.6.1 IOBAR Mapped Access to Device 2 MMIO Space

Device 2, Processor Graphics, contains an IOBAR register. If Device 2 is enabled, Processor Graphics registers or the GTT table can be accessed using this IOBAR. The IOBAR is composed of an index register and a data register.

MMIO_Index: MMIO_INDEX is a 32-bit register located at IOBAR + 0h. A 32-bit (all bytes enabled) I/O write to this port loads the offset of the MMIO register or offset into the GTT that needs to be accessed. An I/O Read returns the current value of this register. I/O read/write accesses less than 32 bits in size (all bytes enabled) will not target this register.

MMIO_Data: MMIO_DATA is a 32-bit register located at IOBAR + 4h. A 32-bit (all bytes enabled) I/O write to this port is re-directed to the MMIO register pointed to by the MMIO-index register. An I/O read to this port is re-directed to the MMIO register pointed to by the MMIO-index register. I/O read/write accesses less than 32 bits in size (all bytes enabled) will not target this register.

The result of accesses through IOBAR can be:

- Accesses directed to the GTT table. (that is, route to DRAM)
- Accesses to Processor Graphics registers with the device.
- Accesses that target the PCH, MCHBAR or SA ranges within GTTMMADR will be aborted

Note: GTT table space writes (GTTADR) are supported through this mapping mechanism.

This mechanism to access Processor Graphics MMIO registers should NOT be used to access VGA I/O registers that are mapped through the MMIO space. VGA registers should be accessed directly through the dedicated VGA I/O ports.

6.7 System Management Mode (SMM)

The CPU Core handles all SMM mode transaction routing. The compute die does not allow I/O devices access to the CSEG/TSEG ranges.

OPI Interface is not allowed to access the SMM space.

Table 6-1. SMM Regions

SMM Space Enabled	Transaction Address Space	DRAM Space (DRAM)
TSEG (T)	(TOLUD – STOLEN – TSEG) to TOLUD – STOLEN	(TOLUD – STOLEN – TSEG) to TOLUD – STOLEN

6.8 SMM and VGA Access Through GTT TLB

Accesses through GTT TLB address translation SMM DRAM space are not allowed. Writes will be routed to memory address 000C_0000h with byte enables de-asserted and reads will be routed to Memory address 000C_0000h. If a GTT TLB translated address hits VGA space, an error is recorded.

OPI Interface originated accesses are **never** allowed to access SMM space directly or through the GTT TLB address translation. If a GTT TLB translated address hits enabled SMM DRAM space, an error is recorded.

OPI Interface write accesses through the GMADR range will not be snooped. Only OPI assesses to GMADR linear range (defined using fence registers) are supported. OPI Interface tileY and tileX writes to GMADR are not supported. If, when translated, the resulting physical address is to enable SMM DRAM space, the request will be remapped to address 000C_0000h with de-asserted byte enables.

OPI Interface read accesses to the GMADR range are not supported. Therefore, there are no address translation concerns. OPI Interface reads to GMADR will be remapped to address 000C_0000h. The read will complete with UR (unsupported request) completion status.

GTT fetches are always decoded (at fetch time) to ensure fetch is not in SMM (actually, anything above base of TSEG or 640 KB - 1 MB). Thus, the fetches will be invalid and go to address 000C_0000h.

6.8.1 I/O Address Space

The processor transaction router generates OPI Interface bus cycles for all compute die I/O accesses that it does not claim. The Configuration Address Register (CONFIG_ADDRESS) and the Configuration Data Register (CONFIG_DATA) are used to generate PCI configuration space access.

The processor allows 64K+3 bytes to be addressed within the I/O space. The upper three locations can be accessed only during I/O address wrap-around.

A set of I/O accesses are consumed by the Processor Graphics device if it is enabled. The mechanisms for Processor Graphics I/O decode and the associated control is explained in following sub-sections.

The I/O accesses are forwarded to the OPI Interface bus. I/O writes are NOT posted. Memory writes to are posted.

The compute die responds to I/O cycles initiated on OPI with an UR status. Upstream I/O cycles and configuration cycles should never occur. If one does occur, the transaction will complete with an UR completion status.

I/O reads that lie within 8-byte boundaries but cross 4-byte boundaries are issued from the processor as one transaction. The reads will be split into two separate transactions. I/O writes that lie within 8-byte boundaries but cross 4-byte boundaries will be split into two transactions by the processor.

6.9 Legacy VGA and I/O Range Decode Rules

The legacy 128 KB VGA memory range 000A_0000h – 000B_FFFFh can be mapped to Processor Graphics (Device 2), and/or to the OPI interface depending on the programming of the VGA steering bits. Priority for VGA mapping is constant in that the Compute die always decodes internally mapped devices first. Internal to the compute die, decode precedence is always given to Processor Graphics. The compute die always positively decodes internally mapped devices, namely the Processor Graphics. Subsequent decoding of regions mapped to OPI Interface depends on the Legacy VGA configurations bits (VGA Enable and MDAP).

VGA range accesses will always be mapped as UC type memory.

Accesses to the VGA memory range are directed to Processor Graphics depend on the configuration. The configuration is specified by:

- Processor Graphics controller in Device 2 is enabled (DEVEN.D2EN bit 4)
- Processor Graphics VGA in Device 0 Function 0 is enabled through register GGC bit 1.
- Processor Graphics's memory accesses (PCICMD2 04h – 05h, MAE bit 1) in Device 2 configuration space are enabled.
- VGA compatibility memory accesses (VGA Miscellaneous Output register – MSR Register, bit 1) are enabled.
- Software sets the proper value for VGA Memory Map Mode register (VGA GR06 Register, bits 3:2). Refer the following table for translations.

Table 6-2. Processor Graphics Frame Buffer Accesses

Memory Access GR06(3:2)	A0000h - AFFFFh	B0000h - B7FFFh MDA	B8000h - BFFFFh
00	Processor Graphics	Processor Graphics	Processor Graphics
01	Processor Graphics	OPI interface	OPI interface
10	OPI interface	Processor Graphics	OPI interface
11	OPI interface	OPI interface	Processor Graphics

Note: Additional qualification within Processor Graphics comprehends internal MDA support. The VGA and MDA enabling bits detailed below control segments not mapped to Processor Graphics.

VGA I/O range is defined as addresses where A[15:0] are in the ranges 03B0h to 03BBh, and 03C0h to 03DFh. VGA I/O accesses are directed to Processor Graphics depends on the following configuration:

- Processor Graphics controller in Device 2 is enabled through register DEVEN.D2EN bit 4.
- Processor Graphics VGA in Device 0 Function 0 is enabled through register GGC bit 1.
- Processor Graphics's I/O accesses (PCICMD2 04 – 05h, IOAE bit 0) in Device 2 are enabled.
- VGA I/O decodes for Processor Graphics uses 16 address bits (15:0) there is no aliasing. This is different when compared to a bridge device (Device 1) that used only 10 address bits (A 9:0) for VGA I/O decode.
- VGA I/O input/output address select (VGA Miscellaneous Output register - MSR Register, bit 0) is used to select mapping of I/O access as defined in the following table.

Table 6-3. Processor Graphics VGA I/O Mapping

I/O Access MSRb0	3CX	3DX	3B0h – 3BBh	3BCh – 3BFh
0	Processor Graphics	OPI interface	Processor Graphics	OPI interface
1	Processor Graphics	Processor Graphics	OPI interface	OPI interface

Note: Additional qualification within Processor Graphics comprehends internal MDA support. The VGA and MDA enabling bits detailed below control ranges not mapped to Processor Graphics.

For regions mapped outside of the Processor Graphics (or if Processor Graphics is disabled), the legacy VGA memory range A0000h – BFFFFh are mapped to the OPI Interface depending on the MDAPxx bits in the Legacy Access Control (LAC) register in Device 0 configuration space. The same register controls mapping VGA I/O address ranges. The VGA I/O range is defined as addresses where A[9:0] are in the ranges 3B0h to 3BBh and 3C0h to 3DFh (inclusive of ISA address aliases – A[15:10] are not decoded). The function and interaction of these two bits is described below:

The following table shows the behavior for all combinations of MDA and VGA.

Table 6-4. MDA IO Transaction Mapping

MDAP	Range	Destination	Exceptions / Notes
0	MDA	OPI interface	
1	MDA	OPI interface	x3BCh – x3BEh will also go to OPI interface

The same registers control mapping of VGA I/O address ranges. The VGA I/O range is defined as addresses where A[9:0] are in the ranges 3B0h to 3BBh and 3C0h to 3DFh (inclusive of ISA address aliases – A[15:10] are not decoded).

MDA Present (MDAP): This bit controls the routing of processor-initiated transactions targeting MDA compatible I/O and memory address ranges. MDA resources are defined as the following:

Table 6-5. MDA Resources

Range Type	Address
Memory	0B0000h – 0B7FFFh
I/O	3B4h, 3B5h, 3B8h, 3B9h, 3BAh, 3BFh (Including ISA address aliases, A[15:10] are not used in decode)

Any I/O reference that includes the I/O locations listed above, or their aliases, will be forwarded to the OPI interface even if the reference includes I/O locations not listed above.

For I/O reads that are split into multiple DWord accesses, this decode applies to each DWord independently. For example, a read to x3B3h and x3B4h (quadword read to x3B0h with BE#=E7h) will result in a DWord read from PEG at 3B0h (BE#=Eh), and a DWord read from OPI at 3B4h (BE=7h). Since the processor will not issue I/O writes crossing the DWord boundary, this case does not exist for writes.

Summary of decode priority:

- Processor Graphics VGA, if enabled, gets:
  - 03C0h – 03CFh: always
  - 03B0h – 03BBh: if MSR[0]=0 (MSR is I/O register 03C2h)
  - 03D0h – 03DFh: if MSR[0]=1

Note: 03BCh – 03BFh never decodes to Processor Graphics; 3BCh – 3BEh are parallel port I/Os, and 3BFh is only used by true MDA devices.

- Else, if ISA Enable=1, OPI gets:
  - upper 768 bytes of each 1K block
- Else, IOBASE/IOLIMIT applies.

6.10 I/O Mapped Registers

The processor contains two registers that reside in the processor I/O address space - the Configuration Address (CONFIG_ADDRESS, port 0xCF8) Register and the Configuration Data (CONFIG_DATA, port 0xCFC) Register. The Configuration Address Register enables/disables the configuration space and determines what portion of configuration space is visible through the Configuration Data window.

6.10.1 CFC/CF8 IO Transactions

The processor claims IO transactions for VGA/Extended VGA found in the display/ graphics interface. It also claims the two 32-bit registers at port CF8h and CFCh used to access PCI configuration space.

6.10.2 Fixed I/O Address Ranges

I/O Address	Read Target	Write Target	Internal Unit (unless[E]: External)[2]	Separate Enable/Disable
CF8h	PCI Configuration Access Address	PCI Configuration Access Address	PCI Configuration Space	None
CFCh	PCI Configuration Access Data	PCI Configuration Access Data	PCI Configuration Space	None
3B0h - 3BBh	VGA Legacy Control & Status	VGA Legacy Control & Status	PCIe: VGA Legacy Control & Status	Yes.INTR_BCTRL
3C0h - 3DFh	VGA Legacy Control & Status	VGA Legacy Control & Status	PCIe: VGA Legacy Control & Status	Yes.INTR_BCTRL

6.11 PCH Address Mapping

The Functional Description includes the following topics:

- Fixed I/O Address Ranges
- Variable I/O Decode Ranges

6.11.1 Fixed I/O Address Ranges

OPI cycles that go to target ranges that are marked as Reserved will be handled by the PCH; writes are ignored and reads will return all 1s. The P2SB will claim many of the fixed I/O accesses and forward those transactions over sideband fabric to their functional target. Address ranges that are not listed or marked Reserved are NOT positively decoded by the PCH (unless assigned to one of the variable ranges) and will be internally terminated by the PCH.

Note: For each I/O range, there may be separate behavior for reads and writes. The following Table shows the Fixed I/O decode ranges from the processor perspective.

Table 6-6. Fixed I/O Ranges Decoded by PCH

I/O Address	Read Target	Write Target	Internal Unit (unless [E]: External)[2]	Separate Enable/Disable
20h – 21h	Interrupt Controller	Interrupt Controller	Interrupt	None
24h – 25h	Interrupt Controller	Interrupt Controller	Interrupt	None
28h – 29h	Interrupt Controller	Interrupt Controller	Interrupt	None
2Ch – 2Dh	Interrupt Controller	Interrupt Controller	Interrupt	None
2E-2F	Super I/O	Super I/O	[E] Forwarded to eSPI	Yes. IOE.SE
30h – 31h	Interrupt Controller	Interrupt Controller	Interrupt	None
34h – 35h	Interrupt Controller	Interrupt Controller	Interrupt	None
38h – 39h	Interrupt Controller	Interrupt Controller	Interrupt	None
3Ch – 3Dh	Interrupt Controller	Interrupt Controller	Interrupt	None
40h	Timer/Counter	Timer/Counter	8254 Timer	None
42h-43h	Timer/Counter	Timer/Counter	8254 Timer	None
4E-4F	Microcontroller	Microcontroller	[E] Forwarded to eSPI	Yes. IOE.ME2[2]
50h	Timer/Counter	Timer/Counter	8254 Timer	None
52h-53h	Timer/Counter	Timer/Counter	8254 Timer	None
60h	Keyboard Controller	Keyboard Controller	[E] Forwarded to eSPI	Yes, with 64h. IOE.KE[2]
61h	NMI Controller	NMI Controller	CPU I/F	None
62h	Microcontroller	Microcontroller	[E] Forwarded to eSPI	Yes, with 66h. IOE.ME12
63h	NMI Controller [1]	NMI Controller[1]	CPU I/F	Yes, alias to 61h. GIC.P61AE[3]
64h	Keyboard Controller	Keyboard Controller	[E] Forwarded to eSPI	Yes, with 60h. IOE.KE[2]
65h	NMI Controller [1]	NMI Controller [1]	CPU I/F	Yes, alias to 61h. GIC.P61AE[3]
66h	Microcontroller	Microcontroller	[E] Forwarded to eSPI	Yes, with 62h. IOE.ME12
67h	NMI Controller [1]	NMI Controller [1]	CPU I/F	Yes, alias to 61h. GIC.P61AE[3]
70h	RTC Controller	NMI and RTC Controller	RTC	None
71h	RTC Controller	RTC Controller	RTC	None

72h	RTC Controller	RTC Controller	RTC	None Alias to 70h if RC.UE[4]=0, else 72h
73h	RTC Controller	RTC Controller	RTC	None Alias to 71h if RC.UE[4]='0', else 73h
74h	RTC Controller	RTC Controller	RTC	None
75h	RTC Controller	RTC Controller	RTC	None
76h-77h	RTC Controller	NMI and RTC Controller	RTC	None Alias to 70h-71h if RC.UE[4]=0, else 76h-77h
80h[5]	eSPI or PCIe	eSPI or PCIe	Read: [E] eSPI or PCIe Write: [E] eSPI or [E] PCIe	None. PCIe if GCS.RPR[3]='1', else eSPI
84h - 86h	eSPI or PCIe	eSPI or PCIe	Read: [E] eSPI or PCIe Write: [E] eSPI or [E] PCIe	None. PCIe if GCS.RPR[3]='1', else eSPI
88h	eSPI or PCIe	eSPI or PCIe	Read: [E] eSPI or PCIe Write: [E] eSPI or [E] PCIe	None. PCIe if GCS.RPR[3]='1', else eSPI
8Ch - 8Eh	eSPI or PCIe	eSPI or PCIe	Read: [E] eSPI or PCIe Write: [E] eSPI or [E] PCIe	None. PCIe if GCS.RPR[3]='1', else eSPI
90h	eSPI	eSPI	Read: [E] eSPI Write: [E] eSPI	None. Alias to 80h
92h	Reset Generator	Reset Generator	CPU I/F	None
94h - 96h	eSPI	eSPI	Read: [E] eSPI Write: [E] eSPI	None. Alias to 84h - 86h
98h	eSPI	eSPI	Read: [E] eSPI Write: [E] eSPI	None. Alias to 88h
9Ch - 9Eh	eSPI	eSPI	Read: [E] eSPI Write: [E] eSPI	None. Alias to 8xh
A0h - A1h	Interrupt Controller	Interrupt Controller	Interrupt	None
A4h - A5h	Interrupt Controller	Interrupt Controller	Interrupt	None
A8h - A9h	Interrupt Controller	Interrupt Controller	Interrupt	None
ACh - ADh	Interrupt Controller	Interrupt Controller	Interrupt	None
B0h - B1h	Interrupt Controller	Interrupt Controller	Interrupt	None
B2h - B3h	Power Management	Power Management	Power Management	None
B4h - B5h	Interrupt Controller	Interrupt Controller	Interrupt	None
B8h - B9h	Interrupt Controller	Interrupt Controller	Interrupt	None
BCh - BDh	Interrupt Controller	Interrupt Controller	Interrupt	None

200-207h	Gameport Low	Gameport Low	[E] Forwarded to eSPI	Yes. IOE.LGE
208-20Fh	Gameport High	Gameport High	[E] Forwarded to eSPI	Yes. IOE.HGE
4D0h – 4D1h	Interrupt Controller	Interrupt Controller	Interrupt	None
CF9h	Reset Generator	Reset Generator	Interrupt controller	None

Notes:
1. Only if the Port 61 Alias Enable bit (GIC.P61AE) bit is set. Otherwise, the cycle is internally terminated by the PCH.
2. Refer to I/O Enables (IOE) register in Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors for Internet of Things (IoT) Applications, Datasheet, Volume 2 (Book 2 of 3), Mule Creek Canyon (Document Number: 636722)
3. Refer to General Control and Status (GCS) register in Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors for Internet of Things (IoT) Applications, Datasheet, Volume 2 (Book 2 of 3), Mule Creek Canyon (Document Number: 636722)
4. Refer to RTC Configuration (RC) register in Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors for Internet of Things (IoT) Applications, Datasheet, Volume 2 (Book 2 of 3), Mule Creek Canyon (Document Number: 636722)
5. This includes byte, word or double-word (DW) access at I/O address 80h

6.12 Variable I/O Decode Ranges

The following table shows the Variable I/O Decode Ranges. They are set using Base Address Registers (BARs) or other config bits in the various configuration spaces. The PnP software (PCI or ACPI) can use their configuration mechanisms to set and adjust these values.

Warning: The Variable I/O Ranges should not be set to conflict with the Fixed I/O Ranges. Unpredictable results if the configuration software allows conflicts to occur. The PCH does not perform any checks for conflicts.

Table 6-7. Variable I/O Decode Ranges

Range Name[1]	Mappable	Size (Bytes)	Target
ACPI	Anywhere in 64K I/O Space	256	Power Management
SMBus	Anywhere in 64K I/O Space	32	SMB Unit
TCO	Anywhere in 64K I/O Space	32	SMB Unit
Parallel Port	3 ranges in 64K I/O Space	8	eSPI
Serial Port 1	8 Ranges in 64K I/O Space	8	eSPI
Serial Port 2	8 Ranges in 64K I/O Space	8	eSPI
Serial Port 3	2 Ranges in 64K I/O Space	8	eSPI
Floppy Disk Controller	Anywhere in 64K I/O Space	8	eSPI
LPC Generic 1	Anywhere in 64K I/O Space	4 to 256 Bytes	eSPI CS0_N
LPC Generic 2	Anywhere in 64K I/O Space	4 to 256 Bytes	eSPI CS0_N
LPC Generic 3	Anywhere in 64K I/O Space	4 to 256 Bytes	eSPI CS0_N
LPC Generic 4	Anywhere in 64K I/O Space	4 to 256 Bytes	eSPI CS0_N
eSPI CS1 Generic 1	Anywhere in 64K I/O Space	4 to 256 Bytes	eSPI CS1_N
eSPI CS2 Generic 1	Anywhere in 64K I/O Space	4 to 256 Bytes	eSPI CS2_N
eSPI CS3 Generic 1	Anywhere in 64K I/O Space	4 to 256 Bytes	eSPI CS3_N
Serial ATA Index/Data Pair	Anywhere in 64K I/O Space	16	SATA Host Controller

PCI Express Root Ports	Anywhere in 64K I/O Space	I/O Base/Limit	PCI Express Root Ports 1-6
Note: All ranges are decoded directly from OPI.			

6.13 Memory Map

The following Table shows (from the Compute Die perspective) the memory ranges that the PCH will decode. Cycles that arrive from OPI that are not directed to any of the internal memory targets that decode directly from OPI will be initiator aborted. PCIe* cycles generated by external PCIe* initiators will be positively decoded unless they fall in the PCI-PCI bridge memory forwarding ranges (those addresses are reserved for PCI peer-to-peer traffic). Software must not attempt locks to the PCH’s memory-mapped I/O ranges.

Table 6-8. PCH Memory Decode Ranges (Compute Die Perspective)

Memory Range	Target	Dependency/Comments
000E 0000 - 000E FFFF	SPI	Bit 6 in BIOS Decode Enable Register is set.
000F 0000 - 000F FFFF	SPI	Bit 7 in BIOS Decode Enable Register is set.
FECX X000 - FECX X040	I/O(x)APIC inside PCH	XX controlled via APIC Range Select (ASEL) field and APIC Enable IOAC.AE bit.
FEC1 0000 - FEC1 7FFF	PCIe port 1	PCIe root port 1 I/OxApic Enable (PAE) is set
FEC1 8000 - FEC1 FFFF	PCIe port 2	PCIe root port 2 I/OxApic Enable (PAE) is set
FEC2 0000 - FEC2 7FFF	PCIe port 3	PCIe root port 3 I/OxApic Enable (PAE) is set
FEC2 8000 - FEC2 FFFF	PCIe port 4	PCIe root port 4 I/OxApic Enable (PAE) is set
FEC3 8000 - FEC3 FFFF	PCIe port 5	PCIe root port 5 I/OxApic Enable (PAE) is set
FEC3 8000 - FEC3 FFFF	PCIe port 6	PCIe root port 6 I/OxApic Enable (PAE) is set
FEC4 0000 - FEC4 7FFF	PCIe port 7	PCIe root port 7 I/OxApic Enable (PAE) is set
FEF0 0000 - FEFF FFFF	SPI	uCode Patch Region Enable UCPR.UPRE is set
FFC0 0000 - FFC7 FFFF FF80 0000 - FF87 FFFF	SPI	Bit 8 in BIOS Decode Enable Register is set
FFC8 0000 – FFCF FFFF FF88 0000 - FF8F FFFF	SPI	Bit 9 in BIOS Decode Enable Register is set
FFD0 0000 - FFD7 FFFF FF90 0000 - FF97 FFFF	SPI	Bit 10 in BIOS Decode Enable Register is set
FFD8 0000 – FFDF FFFF FF98 0000 - FF9F FFFF	SPI	Bit 11 in BIOS Decode Enable Register is set
FFE0 0000 - FFE7 FFFF FFA0 0000 - FFA7 FFFF	SPI	Bit 12 in BIOS Decode Enable Register is set
FFE8 0000 – FFEF FFFF FFA8 0000 – FFAF FFFF	SPI	Bit 13 in BIOS Decode Enable Register is set
FFF0 0000 - FFF7 FFFF FFB0 0000 - FFB7 FFFF	SPI	Bit 14 in BIOS Decode Enable Register is set
FFFC 0000 - FFFF	SPI	Always enabled. Refer to Section 6.13.1 on the Top-Block Swap
FFF8 0000 - FFFB FFFF FFB8 0000 - FFBF FFFF	SPI	Always enabled. Refer to Section 6.13.1 on the Top-Block Swap
FF70 0000 - FF7F FFFF FF30 0000 - FF3F FFFF	SPI	Bit 3 in BIOS Decode Enable Register is set

FF60 0000 - FF6F FFFF FF20 0000 - FF2F FFFF	SPI	Bit 2 in BIOS Decode Enable Register is set
FF50 0000 - FF5F FFFF FF10 0000 - FF1F FFFF	SPI	Bit 1 in BIOS Decode Enable Register is set
FF40 0000 - FF4F FFFF FF00 0000 - FF0F FFFF	SPI	Bit 0 in BIOS Decode Enable Register is set
FED0 X000 - FED0 X3FF	HPET	BIOS determines "fixed" location which is one of four 1KB ranges where X (in the first column) is 0h, 1h, 2h, or 3h.
FED4 0000 - FED4 7FFF	SPI or CSE (set by strap)	TPM and Trusted Mobile KBC
FED4 C000 - FED4 FFFF	PCH Internal (PSF Error Handler)	Always enabled
FED5 0000 - FED5 FFFF	CSE	Always enabled
FED6 0000 – FED6 1FFF	xHCI	NOT positively decoded in PCH(OPI/PSF)
FED7 0000 - FED7 4FFF	Internal Device	Security feature related
64kB (MBAR) anywhere in 64-bit address range	xHCI	Enable via standard PCI mechanism (D20:F0)
2MB (BAR) & 4kB (BAR1) anywhere in 64-bit address range	USB eXtensible Device Controller Interface (xDCI)	Enable via standard PCI mechanism (D20:F1)
16kB (HDAxBA), 4kB (SPCxBA) & 1MB (ADSPxBA) anywhere in 64-bit address range	Converged Audio, Video, Speech (cAVS) Controller	Enable via standard PCI mechanism (D31:F3)
64 KB anywhere in 4 GB range	eSPI CS0_N eSPI CS1_N eSPI CS2_N eSPI CS3_N	LPC Generic Memory Range. Enable via setting bit[0] of the ESPI_LGMR register (D31:F0:offset 98h). eSPI CS1 Generic Memory Range. Enable via setting bit[0] of the ESPI_CS1GMR1 register (D31:F0:offset A8h). eSPI CS2 Generic Memory Range. Enable via setting bit[0] of the ESPI_GMR1_EXT[0] register (D31:F0:offset 208h). eSPI CS3 Generic Memory Range. Enable via setting bit[0] of the ESPI_GMR1_EXT[1] register (D31:F0:offset 248h).
32B (SMBMBAR) anywhere in 64-bit address range	SM Bus	Enable via standard PCI mechanism (D31: F4)
32kB (MXTBA), 256B (MXPBA) & 512kB (ABAR) anywhere in 64-bit address range	SATA Controller (AHCI)	Enable via standard PCI mechanism (D23:F0)
Memory Base/Limit anywhere in 4 GB range	PCI Express Root Ports 1-7	Enable via standard PCI mechanism (D28:F[0:6])
Pre-fetchable Memory Base/ Limit anywhere in 64-bit address range	PCI Express Root Ports 1-7	Enable via standard PCI mechanism (D28:F[0:6])
16B (HECIx_MMIO_MBAR) anywhere in 64-bit address range	HECI #0, #1, #2, #3	Enable via standard PCI mechanism (D22:F[0:1,4:5])
16 MB (SBREG_BAR) anywhere in 64-bit address range	P2SB	Enable via standard PCI mechanism (D31:F1)
4kB (BAR & BAR1) anywhere in 64-bit address range	Intel® Serial I/O Controllers	Enable via standard PCI mechanism (D30:F[0:3], D25:F[0:2], D21:F[0:3], D18:F0, D16:F[1:0])

4kB (BAR & BAR1) anywhere in 64-bit address range	embedded Multi Media Card (eMMC) Controller	Enable via standard PCI mechanism (D26:F0)
4kB (BAR & BAR1) anywhere in 64-bit address range	Secure Digital (SD) & Secure Digital I/O Controller	Enable via standard PCI mechanism (D26:F1)
256kB (BAR0) & 4kB (BAR1) anywhere in 64-bit address range	Gigabit Ethernet Time Sensitive Networking (TSN) Controller	Enable via standard PCI mechanism (D30:F4, D29:F[1:2])
2048kB (BAR0) & 4kB (BAR1) anywhere in 64-bit address range	Intel® Programmable Services Engine (Intel® PSE)	Local Host to PSE (LH2OSE) IPC||Enable via standard PCI mechanism (D29:F0)
16kB (BAR0) & 4kB (BAR1) anywhere in 64-bit address range	Intel® PSE: Direct Memory Access (DMA) Controller	Enable via standard PCI mechanism (D29:F[3:5])
16kB (BAR0) & 4kB (BAR1) anywhere in 64-bit address range	Intel® PSE: Pulse Width Modulation (PWM) Controller	Enable via standard PCI mechanism (D29:F6)
8kB (BAR0) & 4kB (BAR1) anywhere in 64-bit address range	Intel® PSE: Inter-Integrated Circuit (I2C) Controller	Enable via standard PCI mechanism (D27:F[0:6] D24:F0)
64kB (BAR0) & 4kB (BAR1) anywhere in 64-bit address range	Intel® PSE: Controller Area Network (CAN) Controller	Enable via standard PCI mechanism (D24:F[1:2])
8kB (BAR0) & 4kB (BAR1) anywhere in 64-bit address range	Intel® PSE: Quadrature Encoder Peripheral (QEP) Controller	Enable via standard PCI mechanism (D24:F[3:6])
8kB anywhere in 64-bit address range	Intel® PSE: Serial Peripheral Interface (SPI) Controller BAR0	Enable

6.13.1 Boot-Block Update Scheme

The PCH supports a “Top-Block Swap” mode where PCH swaps the top block in the IFWI boot media (the boot block) with another location. This allows for safe update of the Boot Block (even if a power failure occurs). When the Top Swap Override hardware strap is set, the PCH will invert the appropriate address lines as selected in Boot Block Size (Top Swap Block Size (TBBS)) soft strap for the IFWI boot media.

The "Top-Block Swap" behavior is as described below. When the Top Swap Enable bit is 0, the PCH will not invert any address bit.

Table 6-9. Addressing Swapping

BOOT_BLOCK_SIZE Value	Accesses to	Being Directed to
000 (64KB)	FFFF_0000h - FFFF_FFFFh	FFFE_0000h - FFFE_FFFFh and vice versa
001 (128KB)	FFFE_0000h - FFFF_FFFFh	FFFC_0000h - FFFD_FFFFh and vice versa
010 (256KB)	FFFC_0000h - FFFF_FFFFh	FFF8_0000h - FFFB_FFFFh and vice versa
011 (512KB)	FFF8_0000h - FFFF_FFFFh	FFF0_0000h - FFF7_FFFFh and vice versa
100 (1MB)	FFF0_0000h - FFFF_FFFFh	FFE0_0000h - FFEF_FFFFh and vice versa
101 - 111	Reserved	Reserved

§ §

7 Graphics

7.1 Processor Graphics

The processor graphics is based on Generation 11 (GEN11-LP GT1) graphics core architecture that enables substantial gains in performance and lower-power consumption over prior generations. Gen 11 architecture supports up to 32 Execution Units (EUs) depending on the processor SKU.

The processor graphics architecture delivers high dynamic range of scaling to address segments spanning low power to high power, increased performance per watt, support for next generation of APIs. Gen 11 scalable architecture is partitioned by usage domains along Render/Geometry, Media, and Display. The architecture also delivers very low-power video playback. The new Graphics Architecture includes 3D compute elements, Multi-format HW assisted decode/encode pipeline, and Mid-Level Cache (MLC) for superior high definition playback, video quality, and improved 3D performance and media.

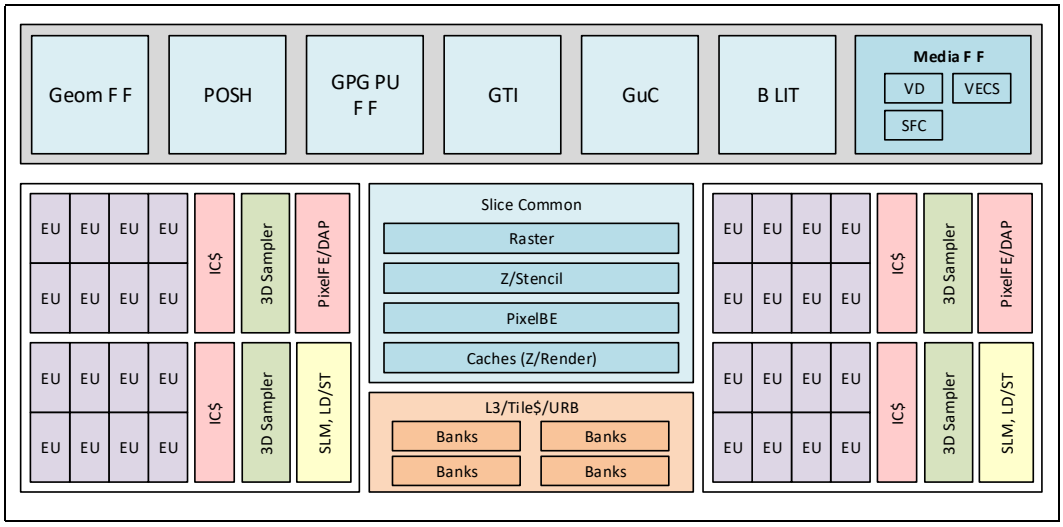
The Display Engine handles delivering the pixels to the screen. Graphics in System Agent (GSA) is the primary channel interface for display memory accesses and "PCI-like" traffic in and out.

7.1.1 Graphic Features

Below are listed of features supported in the processor:

- Microsoft* DirectX 12.1 compliant, OpenGL ES 3.1/3.0/2.0/1.1.
- OpenGL 4.5 supported
- OpenCL™ 1.2, Vulkan 1.0 APIs.
- Dedicated FIVR for Graphics.
- Intel® Virtualization Technology for Directed I/O (VT-d)
- 1280KB Coherent L3.
- Coarse Pixel Shading (CPS) – Pixel rate visibility at pixel granularity and sample rate shading at bigger-than-pixel granularity which mean less shading per group of pixels.
- Position-only Shading (POSh) – Reduce slides asset on idle by discarded vertices ahead compare traditional pipeline. POSh used less power and bandwidth at same performance.
- POSh Tile-Based Rendering (PTBR) – PBTR removed redundancy over POSh by parallel test against render target.
- GT Configuration 1x2x8 and 1x4x8.
- End to end memory compression not supported

Figure 7-1. Block Diagram



7.1.2 Media Support (Intel® QuickSync & Clear Video Technology HD)

Gen 11 implements multiple media video codecs in hardware.

7.1.2.1 Hardware Accelerated Video Decode and Encode

Gen 11 implements a high-performance and low-power HW acceleration for video decoding and encoding operations.

Table 7-1. Hardware Accelerated Video Decode and Encode

Codec Format	Decode Level	Encode Level
H.265/HEVC	M10P @ L5.1 8b/10b Up-to 4kp60 (3480x2160); (420) Up-to 4kp30 (3480x2160); (444)	M10P @ L5.1 8b/10b Up-to 4kp60 (3480x2160); (420) Up-to 4kp30 (3480x2160); (444)
VP9	Profile 0,1,2,3 8b/10b Up-to 4kp60 (3480x2160); (420) Up-to 4kp30 (3480x2160); (444)	Profile 0,1,2,3 8b/10b Up-to 4kp60 (3480x2160); (420) Up-to 4kp30 (3480x2160); (444)
H.264	MP, HP, CBP L5.2 8b/ Up-to 4kp60 (3480x2160); (420)	MP, HP, CBP 8b/ Up-to 4kp60 (3480x2160); (420)
VP8	8b/ Up-to 4kp60 (3480x2160);	Not Supported
WMV9/VC1	SP ML/MP HL/AP L4 and up to 4Kp60 (3480x2160); AP L3 Up-to 1920x1080p24 AP L4 Up to 2048x1536p24 MP HL Up to 1920x1080p30 SP ML Up to 352x288p15	Not Supported
MPEG-2	1080p60 (MP@HL and MP@ML)	Not Supported
VC-1	AP L3 8b/ Up to 1080p30	Not Supported
JPEG/MJPEG	850Mpps (420), 640Mpps (422), 428Mpps (444)	800Mpps (420), 600Mpps (422)

7.1.2.2 Hardware Accelerated Transcoding

Transcoding is a combination of video decode and encode. Using the above hardware capabilities can accomplish a high-performance transcode pipeline. There is not a dedicated API for transcoding.

The processor graphics supports the following transcoding features:

- Low-power and low-latency AVC, HEVC and VP9 encoder for video conferencing and Wireless Display applications.

- Lossless memory compression for media engine to reduce media power.
- Low power Scaler and Format Converter.

7.2 Registers

Please refer to Chapters 7 and 8 of the Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors for Internet of Things (IoT) Applications, Datasheet, Volume 2 (Book 1 of 3), Compute Die Registers Only (Document Number:635255), for a description of the registers associated with subject of this chapter.

§ §

8 Display

8.1 Display Technologies

Display Technologies Support

Technology	Standard
eDP 1.3	VESA Embedded DisplayPort Standard 1.3
MIPI DSI	MIPI Display Serial Interface (DSI) Specification Version 1.2
DisplayPort 1.4	VESA DisplayPort Standard 1.4 VESA DisplayPort PHY Compliance Test Specification 1.4 VESA DisplayPort Link Layer Compliance Test Specification 1.4
HDMI 2.0b	High-Definition Multimedia Interface Specification Version 2.0b

8.2 General Capabilities

Three simultaneous displays (Pipes A,B,C)

- 7 planes and 1 cursor per pipe
- Audio streams per pipe to go to external ports
- HDR support for 3 planes per pipe
- VESA DSC compression support for A, B and C
- Post-DSC joining for resolutions that require more bandwidth than one pipe can support
- Pipe A optimized for low power
- Support write back to DDR for Wireless displays, 1 display support
- 3 combo PHY Supports DSI/eDP/DP/HDMI
- AUX channels for Display ports and eDP
- Multi-stream support for Display ports
- PSR1, PSR2 and multi segmented operations, chip on glass for eDP

8.3 Display Features

Table 8-1. Display Features

Feature	MIPI-DSI	eDP	DP	HDMI
Numbers of Ports	1 (1x4)	1 (x4)	3 (x4)[1]	3 (x4)[1]
Maximum Resolution	1x4: 3200x2000 @ 60Hz (without compression, Multiple Active Displays), 4096x2160 @ 60Hz (with compression, Only Active Display)	4096 x 2160@ 60 Hz	4096 x 2160 @ 60 Hz	4096 x 2160 @ 60 Hz
Data Rate	2.5 GT/s	5.4 GT/s	5.4 GT/s (Without Re-Timer)	5.94 GT/s
Power gated during S0ix w/ display off	Yes	Yes	Yes	Yes
DRRS (Refresh reduction)	Yes (Panel command)	Yes	N/A	N/A
Self-Refresh with frame buffer in Panel	N/A	Yes (PSR)	N/A	N/A
Content-Based back light control	DPST6.0 LACE DPST	DPST6/CABC LACE DPST	N/A	N/A
HDCP 2.3	N/A	N/A	Yes	Yes
PAVP	AES-encrypted buffer, plan control, panic attack			
HD Audio	N/A	N/A	Yes	Yes
Compressed Audio	N/A	N/A	Yes	Yes
DSC (Display Stream Compression)	Yes	Yes	Yes	No

Note:

1. Processor can support 3 HDMI or 3 DP ports; see section 8.4 Port Configuration

8.4 Port Configuration

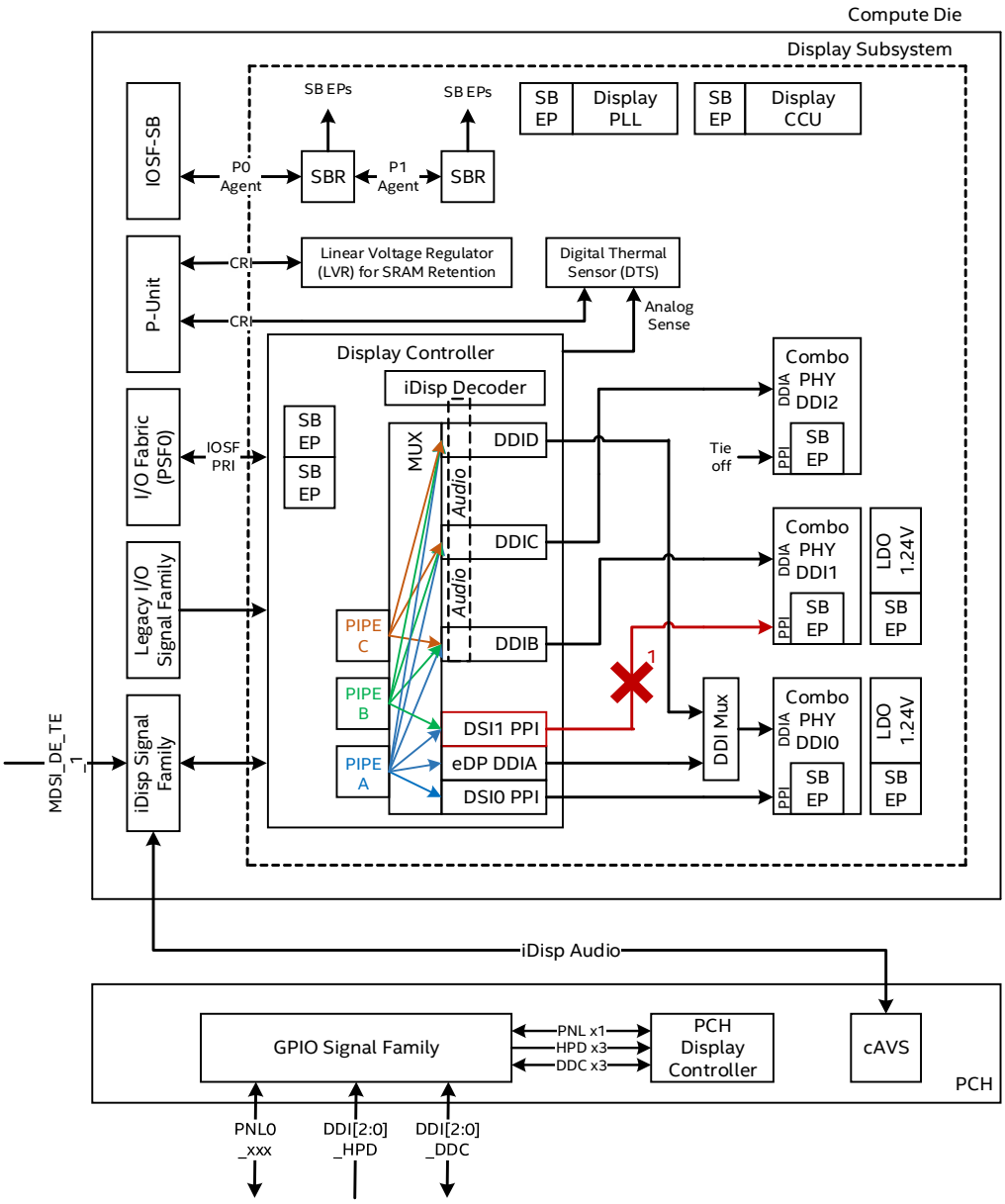
Either internal or external configuration is possible with each pipe. Only one configuration out of the list shown from the following table, is possible per port bases.

Table 8-2. Ports Availability

Combo PHY Port	Internal Port	External Display
DDI0 (Port A)	eDP with DDIA MIPIA with DSI0	HDMI with DDID DP with DDID
DDI1 (Port B)	N/A	HDMI with DDIB DP with DDIB
DDI2 (Port C)	N/A	HDMI with DDIC DP with DDIC

Notes:
1. PSR2 supported only on DDIA in single eDP mode

Figure 8-1. Display Subsystem Block Diagram



Note:

1. Due to limitations in the processor GOP driver, support for MIPI-DSI on Combo PHY Port DDI1 (Port B) is de-featured.

8.5 Display Interfaces

Table 8-3. Digital Display Signals

Package Pin	Dir.	eDP	MIPI DSI	DP	HDMI
DDI0_RCOMP	N/A	Common RCOMP for all PHYs			
DDI0_AUXP DDI0_AUXN	I/O	eDP Auxiliary Channel (AUX_CH)	MIPIA Data 0	DP0 Auxiliary Channel (AUX_CH)	NC
DDI0_TXN0 DDI0_TXP0	I/O	eDP Main Link, Lane 0 (ML_Lane 0)	MIPIA Data 1	DP0 Main Link, Lane 0 (ML_Lane 0)	TMDS0 Data2
DDI0_TXN1 DDI0_TXP1	O	eDP Main Link, Lane 1 (ML_Lane 1)	MIPIA Data 2	DP0 Main Link, Lane 1 (ML_Lane 1)	TMDS0 Data1
DDI0_TXN2 DDI0_TXP2	O	eDP Main Link, Lane 2 (ML_Lane 2)	MIPIA Clock	DP0 Main Link, Lane 2 (ML_Lane 2)	TMDS0 Data 0
DDI0_TXN3 DDI0_TXP3	O	eDP Main Link, Lane 3 (ML_Lane 3)	MIPIA Data 3	DP0 Main Link, Lane 3 (ML_Lane 3)	TMDS0 Clock
DDI1_AUXN DDI1_AUXP	I/O	NC	NC	DP1 Auxiliary Channel (AUX_CH)	NC
DDI1_TXN0 DDI1_TXP0	I/O	NC	NC	DP1 Main Link, Lane 0 (ML_Lane 0)	TMDS1 Data2
DDI1_TXN1 DDI1_TXP1	O	NC	NC	DP1 Main Link, Lane 1 (ML_Lane 1)	TMDS1 Data1
DDI1_TXN2 DDI1_TXP2	O	NC	NC	DP1 Main Link, Lane 2 (ML_Lane 2)	TMDS1 Data0
DDI1_TXN3 DDI1_TXP3	O	NC	NC	DP1 Main Link, Lane 3 (ML_Lane 3)	TMDS1 Clock
DDI2_AUXN DDI2_AUXP	I/O	NC	NC	DP2 Auxiliary Channel (AUX_CH)	NC
DDI2_TXN0 DDI2_TXP0	I/O	NC	NC	DP2 Main Link, Lane 0 (ML_Lane 0)	TMDS2 Data2
DDI2_TXN1 DDI2_TXP1	O	NC	NC	DP2 Main Link, Lane 1 (ML_Lane 1)	TMDS2 Data1
DDI2_TXN2 DDI2_TXP2	O	NC	NC	DP2 Main Link, Lane 2 (ML_Lane 2)	TMDS2 Data0
DDI2_TXN3 DDI2_TXP3	O	NC	NC	DP2 Main Link, Lane 3 (ML_Lane 3)	TMDS2 Clock

Table 8-4. Pin Mapping for PCH Die

			Usage Model		
Display Signals	**Dir.**	**Description**	**DDI0 eDP DDI2 DP/HDMI**	**DDI0 MIPIA DDI2 DP/HDMI**	**DDI0 DP/ HDMI DDI1 DP/ HDMI DDI2 DP/ HDMI**
DDI0_HPD	I/O	Panel 0 Reset or DDI0 Hot Plug Detection	eDP HPD	MIPIA Reset	DDI0 DP/HDMI HPD
DDI1_HPD	I/O	DDI1 Hot Plug Detection			DDI1 DP/HDMI HPD
DDI2_HPD	I	Dedicated DDI2 Hot Plug Detection	DDI2 DP/HDMI HPD	DDI2 DP/HDMI HPD	DDI2 DP/HDMI HPD
PNL0_VDDEN	O	Panel power control enable. This is used to control the VDC source of the panel logic	eDP Power Enable AVDD	MIPIA Power Enable AVDD	
PNL0_BKLTEN	O	Panel backlight enable control. This signal is used to gate power into the backlight circuitry	eDP Backlight Enable	MIPIA Backlight Enable	
PNL0_BKLTCTL	O	Panel brightness control. This is used as the PWM Clock input signall	eDP Backlight Control	MIPIA Backlight Control	
DDI0_D-DC_SCL	I/O	Panel 0 AVEE Power Enable or DDI0 DDC for HDMI or DP++		MIPIA Power Enable AVEE	DDI0 DDC Clk
DDI0_D-DC_SDA	I/O	Panel 0 VIO or DDI0 DDC Data for HDMI or DP++		MIPIA VIO	DDI0 DDC Data
DDI1_D-DC_SCL	I/O	DDI1 DDC Clock for HDMI or DP++			Port B DDC Clk
DDI1_D-DC_SDA	I/O	DDI1 DDC Data for HDMI or DP++			DDI1 DDC CTRL Data
DDI2_D-DC_SCL	I/O	Dedicated DDI2 DDC Clock for HDMI or DP++	DDI2 DDC Clk	DDI2 DDC Clk	DDI2 DDC Clk

			Usage Model		
Display Signals	Dir.	Description	DDI0 eDP DDI2 DP/HDMI	DDI0 MIPIA DDI2 DP/HDMI	DDI0 DP/ HDMI DDI1 DP/ HDMI DDI2 DP/ HDMI
DDI2_D-DC_SDA	I/O	Dedicated DDI2 DDC Data for HDMI od DP++	DDI2 DDC Data	DDI2 DDC Data	DDI2 DDC Data
PNL_MIS-C_DDI0	O	Same as DDI0_HPD used as MIPIA Reset			
MDSI_DE_TE_1	I/O	Tearing Effect from MIPI Panel 0		MIPIA TE	

Note: PNL_VDDEN, PNL_BLKLTEN, PNL_BKLTCTL can be left no connect if neither eDP or MIPI-DSI is not used.

8.6 Multi-Stream Transport (MST) Configuration

- The processor supports Multi-Stream Transport (MST), enabling multiple monitors to be used via a single DisplayPort connector.

8.7 Multiple Display Configurations

The following multiple display configuration modes are supported (with appropriate driver software):

- Single Display is a mode with one display port activated to display the output to one display device.
- Intel Display Clone is a mode with up to three display ports activated to drive the display content of same color depth setting but potentially different refresh rate and resolution settings to all the active display devices connected.
- Extended Desktop is a mode with up to three display ports activated to drive the content with potentially different color depth, refresh rate, and resolution settings on each of the active display devices connected.

The digital ports on the processor can be configured to support eDP/MIPI/DisplayPort/ HDMI.

8.8 High-bandwidth Digital Content Protection (HDCP)

HDCP is the technology for protecting high-definition content against unauthorized copy or unreceptive between a source (computer, digital set top boxes, and so on) and the sink (panels, monitor, and TVs). The processor supports both HDCP 2.3 and 1.4 for

4k Premium content protection over wired displays (HDMI, DVI, and DisplayPort). The HDCP 1.4/2.3 keys are integrated into the processor and customers are not required to physically configure or handle the keys.

8.9 Display Technologies

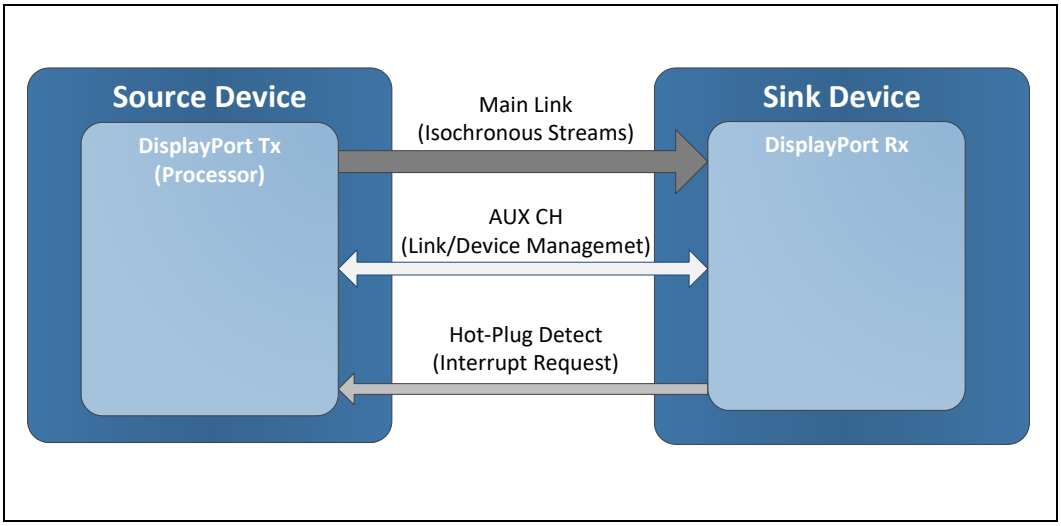
8.9.1 DisplayPort

The DisplayPort is a digital communication interface that uses differential signaling to achieve a high-bandwidth bus interface designed to support connections between PCs and monitors, projectors, and TV displays.

A DisplayPort consists of a Main Link (4 lanes), Auxiliary channel, and a Hot-Plug Detect signal. The Main Link is a unidirectional, high-bandwidth, and low-latency channel used for transport of isochronous data streams such as uncompressed video and audio. The Auxiliary Channel (AUX CH) is a half-duplex bidirectional channel used for link management and device control. The Hot-Plug Detect (HPD) signal serves as an interrupt request for the sink device.

The processor is designed in accordance to VESA DisplayPort specification.

Figure 8-2. DisplayPort Overview



- Support main link of 1, 2, or 4 data lanes.
- Aux channel for Link/Device management.
- Support up to 36 BPP (Bit Per Pixel).
- Support SSC.
- Support YCbCR 4:4:4, YCbCR 4:2:0, and RGB color format.
- Support MST (Multi-Stream Transport).
- Support VESA DSC 1.1.

- Adaptive sync.

8.9.1.1 Multi-Stream Transport (MST)

- The processor supports Multi-Stream Transport (MST), enabling multiple monitors to be used via a single DisplayPort connector.
- MST does not supported concurrent with DSC.
- Max MST DP supported resolution.

8.9.2 High-Definition Multimedia Interface (HDMI)

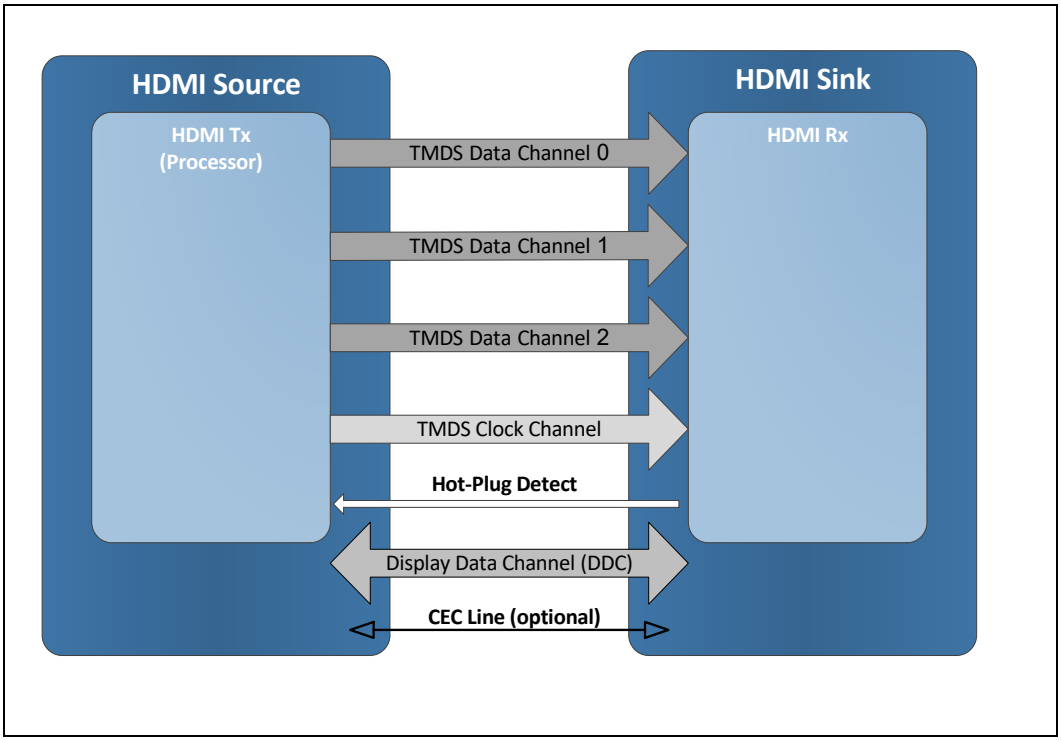
The High-Definition Multimedia Interface (HDMI) is provided for transmitting uncompressed digital audio and video signals from DVD players, set-top boxes, and other audio-visual sources to television sets, projectors, and other video displays. It can carry high-quality multi-channel audio data and all standard and high-definition consumer electronics video formats. The HDMI display interface connecting the processor and display devices uses transition minimized differential signaling (TMDS) to carry audiovisual information through the same HDMI cable. HDMI also needs an external component.

HDMI includes three separate communications channels: TMDS, Digital Display Channel (DDC), and the optional CEC (consumer electronics control). CEC is not supported on the processor. As shown in the following figure, the HDMI cable carries four differential pairs that make up the TMDS data and clock channels. These channels are used to carry video, audio, and auxiliary data. In addition, HDMI carries a VESA DDC. The DDC is used by an HDMI Source to determine the capabilities and characteristics of the Sink.

Audio, video, and auxiliary (control/status) data is transmitted across the three TMDS data channels. The video pixel clock is transmitted on the TMDS clock channel and is used by the receiver for data recovery on the three data channels. The digital display data signals driven natively through the PCH are AC coupled and needs level shifting to convert the AC coupled signals to the HDMI compliant digital signals.

The processor HDMI interface is designed in accordance with the High-Definition Multimedia Interface.

Figure 8-3. HDMI Overview



8.9.3 Digital Video Interface (DVI)

The processor Digital Ports can be configured to drive DVI-D. DVI uses TMDS for transmitting data from the transmitter to the receiver, which is similar to the HDMI protocol except for the maximum data rate (1.65 Gbps) audio and CEC. DVI requires an external component on the platform.

8.9.4 embedded DisplayPort (eDP)

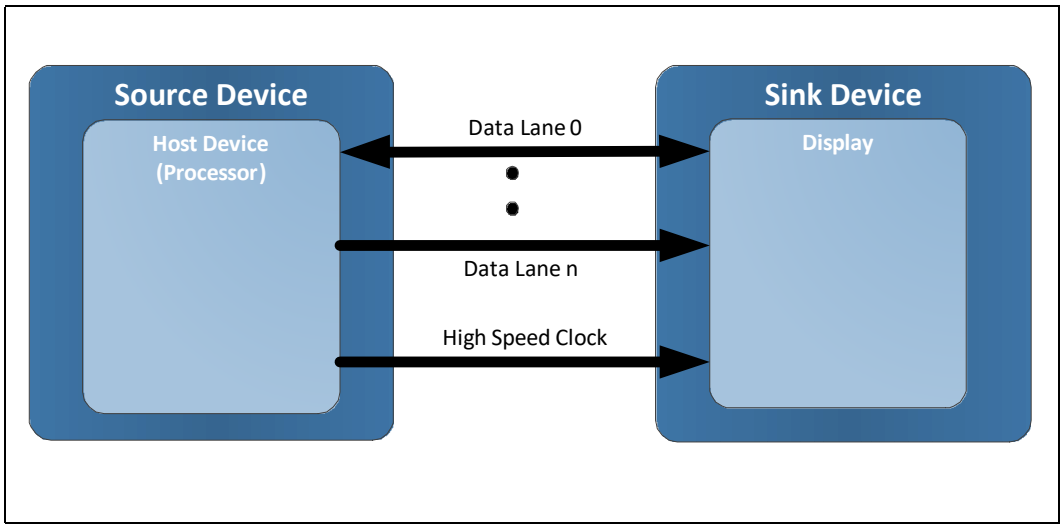
The embedded DisplayPort (eDP) is an embedded version of the DisplayPort standard oriented towards applications such as notebook and All-In-One PCs. Like DisplayPort, embedded DisplayPort also consists of a Main Link, Auxiliary channel, and an optional Hot-Plug Detect signal.

- Support Backlight PWM control signal.
- Support VESA DSC (Data Stream Compression)
- Support SSC
- Panel Self Refresh 1 & 2
- Adaptive sync

8.9.5 MIPI DSI

Display Serial Interface (DSI) specifies the interface between a host processor and peripheral such as display module. DSI is a high speed and high performance serial interface that offers efficient and low power connectivity between the processor and display module. The processor supports only single link interface.

Figure 8-4. MIPI DSI Overview



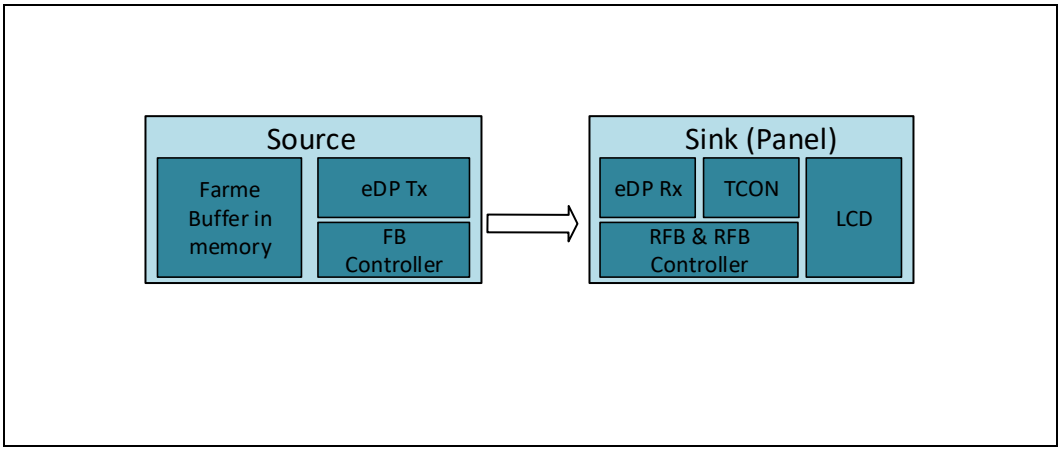
8.9.6 More Features of Display Controller

8.9.6.1 Panel Self Refresh (PSR)

PSR is an eDP feature that allows refresh to stop when the image is unchanging. Display Engine (DE) can disable the eDP link and stop reading pixels from memory. The panel stores the unchanging image in it's Remote Frame Buffer (RFB).

DE tracks image changes and automatically enters and exits PSR. Panel Self Refresh 2 (PSR2) adds several enhancements, including selective update.

Figure 8-5. Panel Self Refresh Diagram



8.9.7 Integrated Audio

HDMI and DisplayPort interfaces carry audio along with video.

The processor supports 3 High Definition audio streams on 3 digital ports simultaneously (the DMA controllers are in PCH).

The integrated audio processing is performed by the PCH, and delivered to the compute die using the on-package Serial Data Output (SDO) and Bit Clock (BCLK) signals.

The Serial Data Input (SDI) is used to carry responses back to the PCH

This HDA interface is not available for use with external CODECs.

Processor Supported Audio Formats over HDMI and DisplayPort

Audio Formats	HDMI	DisplayPort
AC-3 Dolby Digital	Yes	Yes
Dolby Digital Plus	Yes	Yes
DTS-HD	Yes	Yes
LPCM, 192 kHz/24 bit, 6 Channel	Yes	Yes
Dolby TrueHD, DTS-HD initiator Audio (Lossless Blu-Ray Disc Audio Format)	Yes	Yes

The processor will continue to support Silent stream. Silent stream is an integrated audio feature that enables short audio streams, such as system events to be heard over the HDMI and DisplayPort monitors. The processor supports silent streams over the HDMI and DisplayPort interfaces at 48kHz sample-rate two channel support.

8.10 PCH Display

Display support is divided between the compute die and PCH. The processor houses memory interface, display planes, pipes, and digital display interfaces/ports. The PCH integrates display side band signals comprising DDC/GMBUS bus, Hot-Plug Detect and

panel backlight control signals even though digital display interfaces are located on the compute die. There are three sets of Clock/Data, and Hot-Plug Detect signals on the PCH that correspond to DDI/ports. There is also two set of panel control signals that correspond to either eDP or two MIPI-DSI ports.

The Digital Display Channel (DDC) bus is used for communication between the host system and display. Three pairs of DDC (DDC_CLK and DDC_DATA) signals exist on the PCH that correspond to three digital ports on the processor. DDC follows I^2C protocol.

The Hot-Plug Detect (HPD) signal serves as an interrupt request for the sink device for DP, eDP and HDMI. It can be configured using the relevant Individual Voltage Select soft strap to be a 3.3V tolerant signal pin on the PCH.

The panel control signals serve as a way to control panel power and backlight brightness & power.

8.11 Panel Control Signals

Table 8-5. Panel Control Signals

Name	Type	Description
PNL0_VDDEN	O	**Panel Power Enable:** Panel power control enable. This signal is used to control the VDC source of the panel logic
PNL0_BKLTEN	O	**Backlight Enable:** Panel backlight enable control for eDP and MIPI-DSI. This signal is used to gate power into the backlight circuitry
PNL0_BKLTCTL	O	**Panel Backlight Brightness control:** Panel brightness control for eDP and MIPI-DSI. This signal is used as the PWM Clock input signal
Note: PNL_VDDEN, PNL_BKLTEN, PNL_BKLTCTL can be left as no connect if neither eDP or MIPI-DSI is not used		

8.12 Embedded DisplayPort (eDP) Signals

Table 8-6. Embedded DisplayPort Signals

Signal Name	Description	Dir.	Buffer Type	Link Type
DDI0_TXP0 DDI0_TXN0	**embedded DisplayPort Transmit:** differential pair	I/O	Combo PHY	Diff
DDI0_TXP[3:1] DDI0_TXN[3:1]	**embedded DisplayPort Transmit:** differential pair	O	Combo PHY	Diff
DDI0_AUXP DDI0_AUXN	**embedded DisplayPort Auxiliary:** Half-duplex, bidirectional channel consist of one differential pair.	O	Combo PHY	Diff

Table 8-6. Embedded DisplayPort Signals (Continued)

Signal Name	Description	Dir.	Buffer Type	Link Type
EDP_UTILS	**Embedded DisplayPort Utility:** Output control signal used for brightness correction of embedded LCD displays with backlight modulation. This pin will co-exist with functionality similar to existing BKLTCTL pin on PCH	N/A	Async CMOS	SE
DDI0_RCOMP	DDI IO Compensation resistor, supporting DP, eDP and HDMI channels.	N/A	A	SE
Note:	eDP implementation go along with additional sideband signals, for more information please refer to Section 8.10 - Section 8.12.			

8.13 MIPI DSI Signals

Table 8-7. MIPI DSI Signals

Signal Name	Description	Dir.	Buffer Type	Link Type
DDI0_TXP0 DDI0_TXN0	**DPHY Transmit:** differential pair	I/O	Combo PHY	Diff
DDI0_TXP[3:1] DDI0_TXN[3:1] DDI0_AUXP DDI0_AUXN	**DPHY Transmit:** differential pair **DPHY Clock:** differential pair	O	Combo PHY	Diff
MDSI_DE_TE_1	**Tearing Effect**	N/A	GPIO	SE
Note:	DSI implementation go along with additional sideband signals, for more information refer to Section 8.10 - Section 8.12.			

8.14 Digital Display Interface (DDI) Signals

Table 8-8. Display Interface Signals

Signal Name	Description	Dir.	Buffer Type	Link Type
DDI0_TXP0 DDI0_TXN0 DDI1_TXP0 DDI1_TXP0 DDI2_TXP0 DDI2_TXP0	**Digital Display Interface Transmit:** Differential Pairs	I/O	Combo PHY	Diff
DDI0_TXP[3:1] DDI0_TXN[3:1] DDI1_TXP[3:1] DDI1_TXN[3:1] DDI2_TXP[3:1] DDI2_TXN[3:1]	**Digital Display Interface Transmit:** Differential Pairs	O	Combo PHY	Diff
DDI0_AUXP DDI0_AUXN DDI1_AUXP DDI1_AUXN DDI2_AUXP DDI2_AUXN	**Digital Display Interface DisplayPort Auxiliary:** Half-duplex, bidirectional channel consist of one differential pair for each channel.	I/O	Combo PHY	Diff
Note: For DDC signals, refer to Section 8.10 - Section 8.12.				

§ §

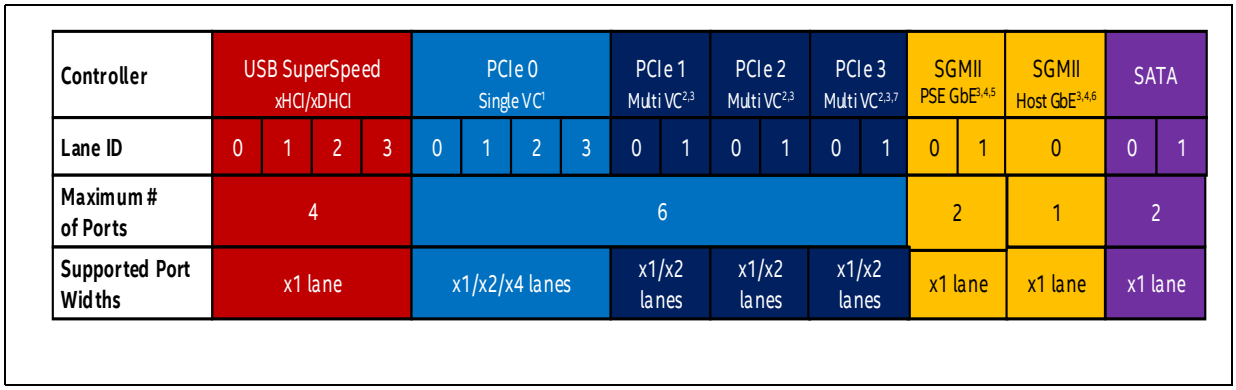
9 Flexible I/O

9.1 Acronyms

Acronyms	Description
USB	Universal Serial Bus
PCIE*	PCI Express* (Peripheral Component Interconnect Express*)
GbE	Gigabit Ethernet
SATA	Serial Advanced Technology Attachment
HSIO	High-Speed IO
VC	Virtual Channel

9.2 HSIO Controller (PCH)

Figure 9-1. HSIO Controller Port Configuration



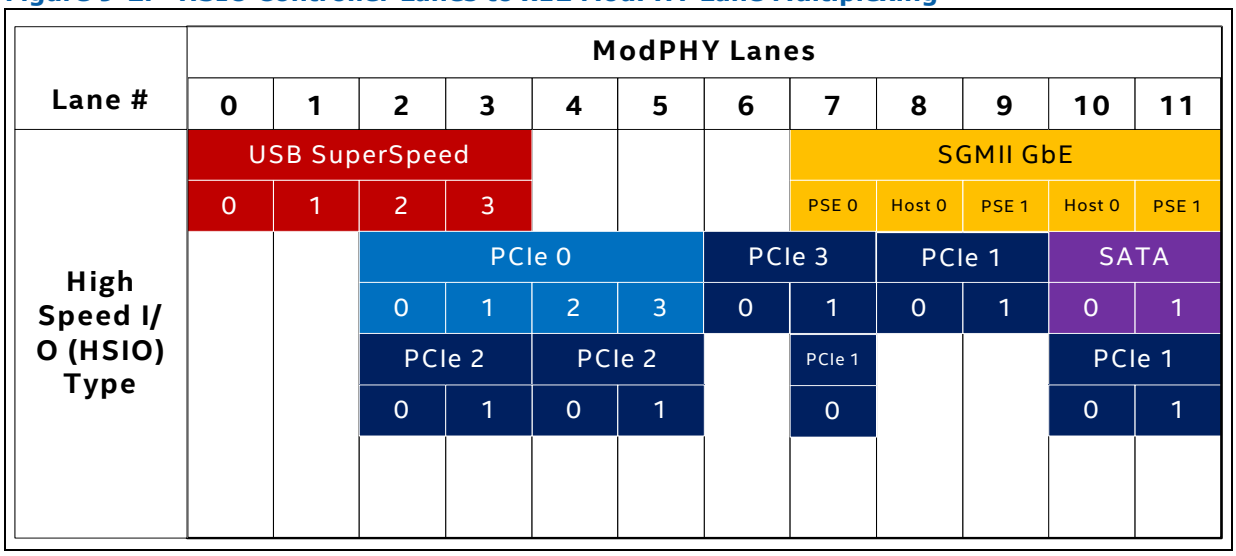
Notes:

1. Single VC (Virtual Channel) PCIe controller has 4 lanes and supports the use of one virtual channel only. This controller can support 1x4 or 2x2 or 1x2 and 2x1 or 4x1.
2. Multi VC PCIe controllers have 2 lanes each and support the use of two virtual channels. These controllers can support 1x2 or 1x1 (always the 1st lane) modes only. 2x1 mode is not supported. As such, for a M.2 connector that supports automatic detection of PCIe or SATA SSDs, it is recommended that ModPHY lane 10 be connected to pins 41, 43, 47 & 49 on the M.2 connector.
3. Controller has multiple (mutually exclusive) choices of which ModPHY lanes its lanes are multiplexed on.
4. GPIO-muxed RGMII interfaces for each Intel® Programmable Services Engine (Intel® PSE) GbE controller are also available.
5. Intel® Programmable Services Engine (Intel® PSE) GbE controllers can be assigned to either Intel® PSE or Host (IA processor) control.
6. Host GbE controller can only be assigned to Host (IA processor) control.

7. PCIe controller 1 (PCIe 1) will not support a x2 port width if PCIe lane 0 is multiplexed on ModPHY lane 7.
8. Lane reversal for PCIe 1, on ModPHY lanes 10 & 11, is supported by setting the DFLEXORM.ORMTC4 register field. Since this reversal happens on the ModPHY lanes, rather than at the PCIe 1 controller, the register field should not be set when either SATA or SGMII is configured on either of the ModPHY lanes.
9. Processors that support Intel® Time Coordinated Computing (TCC) are expected to have improved high bandwidth workload performance on PCH features such as PCI Express and Serial ATA.

Figure 9-2 shows High Speed I/O (HSIO) lane multiplexing in PCH.

Figure 9-2. HSIO Controller Lanes to x12 ModPHY Lane Multiplexing



The 12 HSIO lanes on PCH supports the following configurations:

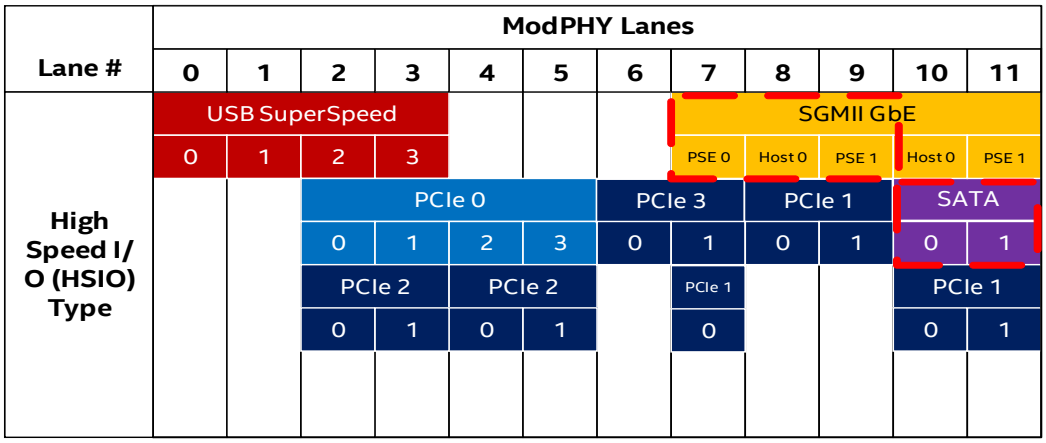
1. Up to 8 PCIe* Lanes
2. Up to 2 SATA Lanes
3. Up to 4 USB 3.1 Gen1/Gen2 Lanes
4. Up to 3 GbE Lanes

Notes: 1. Each GbE controller in the Intel® PSE can alternatively be mapped to a RGMII interface.

2. Because of an architectural limitation in the High-Speed I/O (HSIO) Phase-Locked Loop (PLL) design,only following configuration in the red dotted box is supported. This HSIO PLL limitation does not affect the availability of the PCIe* or USB* 3.1 functions.

Figure 9-3. Configuration SATA + SGMII GbE

Note: When SATA is used on ModPHY lanes 10 or 11, it is not possible to use SGMII GbE on ModPHY lanes 11 or 10.



9.3 Overview/Functional Description

Flexible Input/Output (I/O) is a technology that allows some of the PCH High Speed I/O (HSIO) lanes to be configured for connection to a Gigabit Ethernet (GbE) Controller, a PCIe* Controller, an Extensible Host Controller Interface (XHCI) USB 3.1 Controller, or an Advanced Host Controller Interface (AHCI) SATA Controller. Flexible I/O enables customers to optimize the allocation of the PCH HSIO interfaces to better meet the I/O needs of their system. The RGMII interface must be used if Gigabit Ethernet functionality is required in S0iX and S3-S5 system states.

Note: HSIO lanes are only active in a S0-state.

9.3.1 Flexible I/O Lane Selection

HSIO lane configuration and type is statically selected by soft straps.

Note: The HSIO lane soft strap configuration must match the platform hardware design and the corresponding processor controllers that are enabled in processor hardware, firmware and software. For example, PSE GbE controllers 0 & 1 must not be enabled on HSIO lanes 7, 9 & 11 in platforms that integrate either Intel® Celeron® Processor J6412 or Intel® Celeron® Processor N6210. It is the responsibility of the platform designers to configure the lane muxing and soft straps correctly without any conflict. The hardware behavior is undefined if this scenario ever happens.

9.3.2 PCIe*/SATA Lane Selection

In addition to static configuration via soft straps, Flexible I/O Lanes that have PCIe*/SATA multiplexing can be configured via SATAXPCIE signaling to support implementations like SATA Express or mSATA, where the port configuration is selected by the type of the add-in card that is used.

Note: Due to the muxing between SATA_LED_N signal and SATAXPCIE_0 signal, only one function is available at a time. If a design with M.2 slot that can automatically handle a

SATA (Port 0 at lane 10) or PCIe SSD is desired, then the SATA_LED_N signal is not available.

Note: SATAXPCIE_1 cannot be used for automatic detection of PCIe or SATA SSDs since PCIe 1 is not capable of 2 x1 ports. ModPHY lane 11 should always be statically assigned to be either PCIe or SATA, as required by the design.

9.4 Registers

Please refer to Chapter 29 of the Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors for Internet of Things (IoT) Applications, Datasheet, Volume 2 (Book 2 of 3), Mule Creek Canyon (Document Number: 636722), for a description of the registers associated with subject of this chapter.

§ §

10 Audio, Voice, and Speech

10.1 Feature Overview

The Converged Audio Voice Speech (cAVS) subsystem consists of a collection of controller, DSP, memory, and link interfaces that provides the audio experience to the platform. This subsystem provides streaming of audio from the host SW to external audio codecs, with the host CPU and/or DSP providing the audio enrichment. It may also be used as a host based sensor hub for managing various context info on the platform

The optional DSP can be enabled in the audio subsystem to provide low latency HW/FW acceleration for common audio and voice functions such as audio encode/decode, acoustic echo cancellation, noise cancellation, etc

The cAVS is fully backward compatible with the Intel HD Audio specification, with the controller implements a number of Output Stream DMA engines and Input Stream DMA engines for data transfers, as well as a Command Output DMA engine and a Response Input DMA engine for control transfers.

The cAVS also supports I2S audio codecs which are not Intel HD Audio standards. The General Purpose DMA engines has the ability to do simple data transfers or control transfers between system memory and the FIFO in the DSP I/O peripheral interfaces directly, however, these transfers are not optimized for power management.

10.2 Legacy Audio Interface - Signal Description

Table 10-1. Legacy Audio Signals

Signal Name	Description
Intel® High Definition Audio Interface	
HDA_RST_N	Initiator hardware reset to external codec(s)
HDA_SYNC	48 kHz fixed rate sample sync to the codec(s)
HDA_BCLK	24.000 MHz serial data clock generated by the Intel® HD Audio controller
HDA_SDO	Serial TDM data output to the codec(s)
HDA_SDI0	Serial TDM data inputs from the codec(s)
HDA_SDI1	
DMIC Interface	
DMIC_CLK_A0	Serial data clock generated by the PCH to the digital microphone module
DMIC_CLK_A1	
DMIC_CLK_B0	
DMIC_CLK_B1	
DMIC_DATA0	Serial data input from the digital microphone module
DMIC_DATA1	
I2S Interface	
AVS_I2S_MCLK1	I^2S Initiator Clock
AVS_I2S_MCLK2	
AVS_I2S0_SCLK AVS_I2S1_SCLK AVS_I2S2_SCLK AVS_I2S3_SCLK AVS_I2S4_SCLK AVS_I2S5_SCLK	I^2S Serial Bit Clocks for connection to I^2S devices Refer to Chapter 22, section 22.19.1 for more details on the transmission mode.
AVS_I2S0_TXD AVS_I2S1_TXD AVS_I2S2_TXD AVS_I2S3_TXD AVS_I2S4_TXD AVS_I2S5_TXD	I^2S Transmit Data (Serial Data Out) for connection to I^2S devices Refer to Chapter 22 section 22.19.1 for more details on the transmission mode.
AVS_I2S0_RXD AVS_I2S1_RXD AVS_I2S2_RXD AVS_I2S3_RXD AVS_I2S4_RXD AVS_I2S5_RXD	I^2S Receive Data (Serial Data In) for connection to I^2S devices Refer to Chapter 22, section 22.19.1 for more details on the transmission mode.
AVS_I2S0_SFRM AVS_I2S1_SFRM AVS_I2S2_SFRM AVS_I2S3_SFRM AVS_I2S4_SFRM AVS_I2S5_SFRM	I^2S Serial Frame for connection to I^2S devices Refer to Chapter 22, section 22.19.1 for more details on the transmission mode.

10.2.1 Key HW features of the AVS Subsystem

10.2.1.1 DSP

The DSP provides a mechanism for intercepting the rendering audio and voice streams (and tones) flowing through the controller's DMA engines and provides DSP enhancements to the audio. The same controller's DMA engines may also be used to download DSP function module at run-time, offering flexibility to the Audio DSP processing pipeline creation. The DSP also offers contextual processing using the sensor data obtained through the serial I/O interfaces (for example, I^2C, UART, SPI, and so on).

10.2.1.2 Memory

The central memory block for the cAVS is known as L2 local memory. All the HW based accelerators and DMA engines are able to access certain regions of this central memory as the audio stream buffer. The memory is also used as the working space for the DSP Core, and it can provide processing to the audio stream data or sensor data flowing through this central memory.

10.2.1.3 I/O Peripheral

The controller and DSP communicates with the external codec(s) over the audio I/O. These audio I/O connection to codec(s) include the Intel HD Audio serial link, the Intel iDisp Audio serial link, or the DSP I/O peripheral for proprietary interfaces (e.g. I^2S). Sensor devices may also be connected over the DSP I/O peripheral (for example, I^2C, and SPI).

Both the Intel HD Audio serial link and Intel iDisp Audio serial link are fully backward compatible with the legacy Intel HD Audio driver software stack.

10.3 Intel® High Definition Audio (Intel® HD Audio) Controller Capabilities

The Intel® HD Audio controller is the standard audio host controller widely adopted in the PC platform, with industrial standard Intel® HD Audio driver software available for Microsoft Windows* and many other Linux* based OS'es. Intel® HD Audio controller features are listed as follows:

- Supports data transfers, descriptor fetches, and DMA position writes using VC0 or VC1.
- Independent Bus Initiator logic for 16 general purpose DMA streams: 7 input and 9 output.
- Supports variable length stream slots.
- Supports up to:
  - 16 streams
  - 7 input
  - 7 system streams
  - 2 offload streams
  - 9 output
  - 2 system streams (dedicated)
  - 2 system / 2 offload streams (shared)

  - 2 offload streams (dedicated)
  - 1 feedback stream
  - 16 channels per stream
  - 32 bits/sample
  - 192 kHz sample rate
- Supports memory-based command/response transport.
- Three 8-channel universal DMA interfaces for transferring data between memory buffers and peripherals and between memories
- Supports optional Immediate Command/Response mechanism.
- Supports output and input stream synchronization.
- Supports global time synchronization.
- Supports MSI interrupt delivery.
- Support for ACPI D3 and D0 Device States.
- Supports Function Level Reset (FLR)
- Support Converged Platform Power Management (CPPM).
- Support 1 ms of buffering with all DMA running with maximum bandwidth.
- Support 10 ms of buffering with 1 output DMA & 1 input DMA running at 2 channels, 96 kHz, 16 bit audio.

The Input / Output Stream DMA can be individually put into coupled mode where the host and link portion of the DMA will be directed to the associated FIFO and flow-controlled automatically by HW; or put into de-coupled mode where the host and link portion of the DMA will be directed to the unique DSP buffers setup by DSP FW for inserting audio processing pipe stages.

10.3.1 Audio DSP Capabilities

The Audio DSP offload engine is an optional feature providing low power DSP functionality and offload the audio processing operation from host CPU. Audio DSP features are listed as follows:

- Audio DSP with 4 Tensilica* LX6+HiF3 cores for low power offloaded audio rendering and recording
  - 400 MHz operating frequency in S0
  - 120 MHz operating frequency in S0ix
  - 64KB L1 RAM
  - 768KB L2 SRAM
- Low power support for Intel® Wake on Voice (Intel® WOV)
- Low power audio playback with post processing
- Low power VoIP and circuit switch voice call with pre-processing
- Various DSP functions optionally provided by DSP Core firware: MP3, AAC, 3rd Party IP Algorithms, etc.

10.4 Direct Attached Digital Microphone (PDM) Interface

The direct attached digital microphone interface is an optional feature offering connections to PDM based digital microphone modules without the need of audio codecs. This provides the lowest possible platform power with the decimation functionality integrated into the audio host controller. Features for the digital microphone interface are listed as follows:

- Two DMIC PDM interfaces with each interface capable of supporting up to 2 digital MEMs microphones
- Low power always listening support for Intel® Wake on Voice (Intel® WOV)
- 2 PCM audio streams (with independent PCM sampling rate: 48 kHz or 16 kHz) per digital mic interface
- Ultrasound reception capable with higher frequency ranges between 3.84 MHz - 4.8 MHz.

10.5 I^2S/PCM Interface

The I^2S / PCM interface is an optional feature offering connection to the I^2S / PCM audio codecs. The I^2S / PCM audio codecs are widely adopted in the phone and tablet platforms as they are typically customized for low power application. The codec structure is typically unique per codec vendor implementation and requires vendor specific SW module for controlling the codec. These I^2S / PCM audio codecs will be enumerated based on ACPI table or OS specific static configuration information. The Audio DSP is required to be enabled in order to enable I^2S / PCM link as registers are only addressable through the Audio DSP and its FW. I^2S/PCM Interface features are listed as follows:

- Multiple I^2S/PCM ports to support multiple I^2S connections
- Can support 3 modes: Target Mode, Target Mode with Locally Generated Initiator Clock, or Initiator Mode. Refer to Section 1.1.1 for more information on initiator and target).
- I^2S audio playback up to 2 ch x 192 kHz x 24 bits
- I^2S audio capture up to2 ch x 192 kHz x 24 bits
- PCM audio playback up to 8 ch x 48 kHz x 24 bits
- PCM audio capture up to 8 ch x 48 kHz x 24 bits
- Support 3G / 4G modem codec
- Support BT codec HFP / HSP SCO at 8 / 16 kHz
- Support BT codec A2DP at 48 kHz
- Support FM radio codec

Note: In FD (Full Duplex) mode, processor's SDO/TXD pin will function as TXD and it is not used for transmission in HD-Tx mode. TXD is output pin.

Note: In HD (Half Duplex) mode, processor's SDI/RXD pin will function as TXD during Tx and RXD during Rx. RXD is inout pin.

10.6 References

Specification	Location
High Definition Audio Specification	http://www.intel.com/content/www/us/en/standards/high-definition-audio-specification.html

§ §

11 Universal Serial Bus (USB)

11.1 Overview

The PCH implements an xHCI USB controller which provides support for up to 10 USB 2.0 signal pairs and 4 USB 3.1 signal pairs.The xHCI controller supports wake up from sleep states S1-S4. The eXtensible Host Controller (xHCI) supports up to 64 devices for a max number of 2048 Asynchronous endpoints (Control/Bulk) or max number of 128 Periodic Endpoints (interrupt/isochronous).

Each walk-up USB 3.1 capable port contains one USB 2.0 signal pair and one USB 3.1 signal pair.

The USB subsystem also supports Dual Role Capability. The xHCI is paired with a standalone eXtensible Device Controller Interface (xDCI) to provide dual role functionality. Only one port can be connected (and active) to the device controller at one time. The USB subsystem incorporates a xDCI USB 3.0 device controller (5Gb/s) that supports all 32 endpoints (in both USB3 and USB2 modes) for maximum configurability.

The xDCI shares all USB ports with the host controller, with the ownership of the port being decided based the USB Power Delivery specification. Since all the ports support device mode, xDCI enabling must be extended by System BIOS. While the port is mapped to the device controller, the host controller Rx detection must always indicate a disconnected port.

11.1.1 USB Supported Features

- Device
  - D0i3 power gating
  - Wake capable on host initiated wakes when system is in Sx
  - Available on all ports
- Port Routing Control for Dual Role Capability

Table 11-1. USB Bandwidth Information

USB Interface	
Category	**Description**
USB 3.1 SuperSpeed Port	4 (1x Dual Role Configurable on any one port)
Peak USB 3.1 Speed	10 Gb/s (Host Role), 5 Gb/s (Device Role)
USB 2.0 Port	10 (1x Dual Role Configurable on any one port)
Direct Connect Interface (DCI)	USB 3.x USB 2.0
Peak USB 2.0 Speed	480Mb/s

Figure 11-1. USB 3.1/PCIe*/SATA Port Mapping

Refer to Chapter 9, "Flexible I/O"

11.1.2 USB Controllers Overview

Extensible Host Controller Interface (xHCI) is the interface specification that defines Host Controller for Universal Serial Bus (USB), which is capable of interfacing with USB 1.x, 2.x, and 3.x compatible devices.

In case that a device (e.g. USB mouse) was connected to the computer, the computer will work as Host and the xHCI will be activated inside the PCH.

Extensible Device Controller Interface (xDCI) is the interface specification that defines Device Controller for Universal Serial Bus (USB), which is capable of interfacing with USB 1.x, 2.x, and 3.x compatible devices

In case that the computer is connected as a device (e.g. tablet connected to desktop) to other computer then the xDCI controller will be activated inside the device will talk to the Host at the other computer.

Note: The PCH incorporates a USB3.0 device controller that allows data transfer of up to 5Gb/s, while USB 3.1 (data transfer of up to 10Gb/s) is not supported. The host controller supports both USB3.0 and USB3.1.

Table 11-2. Processor USB Specification

Protocol Name	Data Rate	USB3.0	USB3.1
Low - Speed	1.5 Mbps	Supported	Supported
Full - Speed	12 Mbps	Supported	Supported
High - Speed	480 Mbps	Supported	Supported
SuperSpeed	5 Gbps	Supported	Supported
SuperSpeed+	10 Gbps (xHCI only)	Not Supported	Supported
Note: USB2 ("Low/Full/High" speeds) implemented in PCH			

Table 11-3. Signal Description (Sheet 1 of 3)

Name	Type	Description
USB3_0_RXN **USB3_0_RXP**	I	**USB 3.1 Differential Receive Pair 1:** These are USB 3.1-based high-speed differential signals for Port #1 and the xHCI/xDCI. It should map to a USB connector with one of the OC (overcurrent).
USB3_0_TXN **USB3_0_TXP**	O	**USB 3.1 Differential Transmit Pair 1:** These are USB 3.1-based high-speed differential signals for Port #1 and the xHCI/xDCI. It should map to a USB connector with one of the OC (overcurrent).
USB3_1_RXN **USB3_1_RXP**	I	**USB 3.1 Differential Receive Pair 2:** These are USB 3.1-based high-speed differential signals for Port #2 and the xHCI/xDCI. It should map to a USB connector with one of the OC (overcurrent).
USB3_1_TXN **USB3_1_TXP**	O	**USB 3.1 Differential Transmit Pair 2:** These are USB 3.1-based high-speed differential signals for Port #2 and the xHCI/xDCI. It should map to a USB connector with one of the OC (overcurrent).
USB3_2_RXN **USB3_2_RXP**	I	**USB 3.1 Differential Receive Pair 3:** These are USB 3.1-based high-speed differential signals for Port #3 and the xHCI/xDCI. It should map to a USB-connector with one of the OC (overcurrent).
USB3_2_TXN **USB3_2_TXP**	O	**USB 3.1 Differential Transmit Pair 3:** These are USB 3.1-based high-speed differential signals for Port #3 and the xHCI/xDCI. It should map to a USB-connector with one of the OC (overcurrent).
USB3_3_RXN **USB3_3_RXP**	I	**USB 3.1 Differential Receive Pair 4:** These are USB 3.1-based high-speed differential signals for Port #4 and the xHCI/xDCI. It should map to a USB connector with one of the OC (overcurrent).

Table 11-3. Signal Description (Sheet 2 of 3)

Name	Type	Description
USB3_3_TXN USB3_3_TXP	O	**USB 3.1 Differential Transmit Pair 4:** These are USB 3.1-based high-speed differential signals for Port #4 and the xHCI/xDCI. It should map to a USB connector with one of the OC (overcurrent).
USB2_0_DP, USB2_0_DN	I/O	**USB 2.0 Port 1 Transmit/Receive Differential:** This USB 2.0 signal pair are routed to xHCI or xDCI and should map to a USB connector with one of the overcurrent OC.
USB2_1_DP, USB2_1_DN	I/O	**USB 2.0 Port 2 Transmit/Receive Differential:** This USB 2.0 signal pair are routed to xHCI or xDCI and should map to a USB connector with one of the overcurrent OC.
USB2_2_DP, USB2_2_DN	I/O	**USB 2.0 Port 3Transmit/Receive Differential:** This USB 2.0 signal pair are routed to xHCI or xDCI and should map to a USB connector with one of the overcurrent OC.
USB2_3_DP, USB2_3_DN	I/O	**USB 2.0 Port 4 Transmit/Receive Differential:** This USB 2.0 signal pair are routed to xHCI or xDCI and should map to a USB connector with one of the overcurrent OC.
USB2_4_DP, USB2_4_DN	I/O	**USB 2.0 Port 5 Transmit/Receive Differential:** This USB 2.0 signal pair are routed to xHCI or xDCI and should map to a USB connector with one of the overcurrent OC.
USB2_5_DP, USB2_5_DN	I/O	**USB 2.0 Port 6 Transmit/Receive Differential:** This USB 2.0 signal pair are routed to xHCI or xDCI and should map to a USB connector with one of the overcurrent OC.
USB2_6_DP, USB2_6_DN	I/O	**USB 2.0 Port 7 Transmit/Receive Differential:** This USB 2.0 signal pair are routed to xHCI or xDCI and should map to a USB connector with one of the overcurrent OC.
USB2_7_DP, USB2_7_DN	I/O	**USB 2.0 Port 8 Transmit/Receive Differential:** This USB 2.0 signal pair are routed to xHCI or xDCI and should map to a USB connector with one of the overcurrent OC.
USB2_8_DP, USB2_8_DN	I/O	**USB 2.0 Port 9 Transmit/Receive Differential:** This USB 2.0 signal pair are routed to xHCI or xDCI and should map to a USB connector with one of the overcurrent OC.

Table 11-3. Signal Description (Sheet 3 of 3)

Name	Type	Description
USB2_9_DP, USB2_9_DN	I/O	**USB 2.0 Port 10 Transmit/Receive Differential:** This USB 2.0 signal pair are routed to xHCI or xDCI and should map to a USB connector with one of the overcurrent OC.
USB2_OC0_N	I	**Overcurrent Indicators:** These signals set corresponding bits in the USB controller to indicate that an overcurrent condition has occurred.
USB2_OC1_N	I	**Overcurrent Indicators:** These signals set corresponding bits in the USB controller to indicate that an overcurrent condition has occurred.
USB2_OC2_N	I	**Overcurrent Indicators:** These signals set corresponding bits in the USB controller to indicate that an overcurrent condition has occurred.
USB2_OC3_N	I	**Overcurrent Indicators:** These signals set corresponding bits in the USB controller to indicate that an overcurrent condition has occurred.
USB2_RCOMP	I	USB Resistor Bias, analog connection points for an external resistor to ground.

Note: 1. Each USB port can only be assigned to one OC_N signal.

2. Intel recommends that no more than four USB ports are assigned to one OC_N signal

11.2 Integrated Pull-Ups and Pull-Down11.3 Registers

Signal	Resistor Type	Value	Notes
USB2_[9:0]_DN	Internal Pull-down	14.25–24.8 kohm	1
USB2_[9:0]_DP	Internal Pull-down	14.25–24.8 kohm	1
USB2_[9:0]_DP	Internal Pull-up	1.5 kohm	1,2
Note: 1. Series resistor (45 ohm ±10%) 2. Tolerance of ±5%			

11.3 Registers

Note: Please refer to Chapters 17 and 18 of the Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors for Internet of Things (IoT) Applications, Datasheet, Volume 2 (Book 2 of 3), Mule Creek Canyon (Document Number: 636722), for a description of the registers associated with subject of this chapter.

§ §

12 PCI Express

12.1 Acronyms

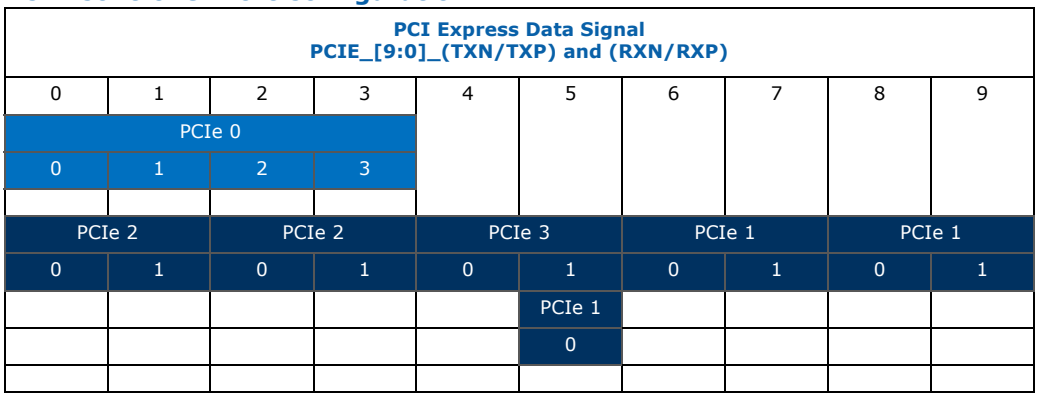
Acronyms	Description
PCIe*	PCI Express* (Peripheral Component Interconnect Express*)

12.2 Signal Description

Group	Signal Name	Description
Data	PCIE_[9:0]_TXN/TXP	PCI Express* Transmit Differential-Pair
	PCIE_[9:0]_RXN/RXP	PCI Express* Receive Differential-Pair
RCOMP	HSIO_RCOMPN and HSIO_RCOMPP	Impedance Compensation Inputs

12.3 I/O Signal Planes and States

Figure 12-1. PCIE Controller Port Configuration



Signal Name	Type	Power Plane	During Reset	Immediately After reset	S3/S4/S5	Deep Sx
PCIE_[9:0]_TXP PCIE_[9:0]_TXN	O	Primary	Internal Pull-down	Internal Pull-down	Internal Pull-down	Off
PCIE_[9:0]_RXP PCIE_[9:0]_RXN	I	Primary	Internal Pull-down	Internal Pull-down	Internal Pull-down	Off
HSIO_RCOMPP HSIO_RCOMPN	I	Primary	Undriven	Undriven	Undriven	Off

Note: PCIE_[9:0]_RXP\RXN pins transition from un-driven to internal pull-down during reset.
Note: Controllers PCIe 1 & PCIe 2 can only be active in one of the indicated locations shown in Figure 12-1. PCIe 1 can only be active in one out of PCIE_5, PCIE_[6:7] or PCIE_[8:9]. PCIe 2 can only be active in one out of PCIE_[0:1] or PCIE_[2:3].

12.4 PCI Express* Port Support Feature Details

Max Transfer Rate	Max Devices (Ports)	Max Lanes	PCIe* Gen Type	Encoding	Transfer Rate (MT/s)	Theoretical Max Bandwidth (GB/s)		
						x1	x2	x4
8 GT/s (Gen3)	6	8	1	8b/10b	2500	0.25	0.50	1.00
			2	8b/10b	5000	0.50	1.00	2.00
			3	128b/130b	8000	0.98	1.97	3.94

Notes:
1. Theoretical Maximum Bandwidth (GB/s) = ((Transfer Rate * Encoding * # PCIe Lane) /8)/1000
- Gen3 Example: = ((8000 * 128/130) * 4)/8)/1000 = 3.94 GB/s

12.5 Overview/Functional Description

There are 4 PCIe controllers in the PCH also known as Controller 0, 1, 2 and 3. Controller 0 is a x4 controller (single-VC) and controllers 1, 2 and 3 are x2 controllers - the same x2 controller (multi-VC) is instantiated 3 times. Below listed list of functionals supported by PCH PCIe:

- Interrupt Generation
- Up to 6 PCIe Ports and up to 8 PCIe* Lanes
- PCI Express* Power Management
- Latency Tolerance Reporting (LTR)
- Advanced Error Reporting
- Single Root I/O Virtualization (SR- IOV) Capability with Access Control Services (ACS) and Alternative Routing ID (ARI)
- SERR# Generation
- PCI Express* TX and RX Lane Polarity Inversion
- End-to-End PCI Express* Controller Lane Reversal (exclude x2 configuration)
- Dynamic Link Width Negotiation as a Target.
  - Refer to Section 1.1.1 for more information on target.
- Dynamic Speed Change
- 256B Maximum Data Payload Size
- PCIe* Subtractive Decode is not supported
  - PCI can still be supported via a PCIe*-to-PCI bridge. However, legacy PCI devices (such as PCMCIA or non-plug-and-play device) that need subtractive decode are not supported.
- Common RefClk RX Architecture support
- Two Virtual Channels (VCs) supported on x2 controllers only
- One Virtual Channel (VC) supported on all controllers.

12.5.1 Supported PCIe* Configurations:

12.5.1.1 PCIe* Controller Virtual Channel

The x2 PCIe Controller supports 2 Virtual Channel (VC) on all the cycle type for both Upstream and Downstream cycles.

VC1 is enabled through standard PCIe Virtual Channel capability and when VC1 is enabled, the PCIe Controller will autonomously exchange Flow Control initialization with the PCIe Device. Upon completion of the VC1 Flow Control, the PCIe Controller will start accepting VC1 transaction.

VC1 and VC0 transaction will be stored in independent Receive Queue to prevent any blocking transaction from VC0 or VC1 to block each other. Ordering within the same VC will still be applicable.

12.5.1.2 PCIe* Port Traffic Class

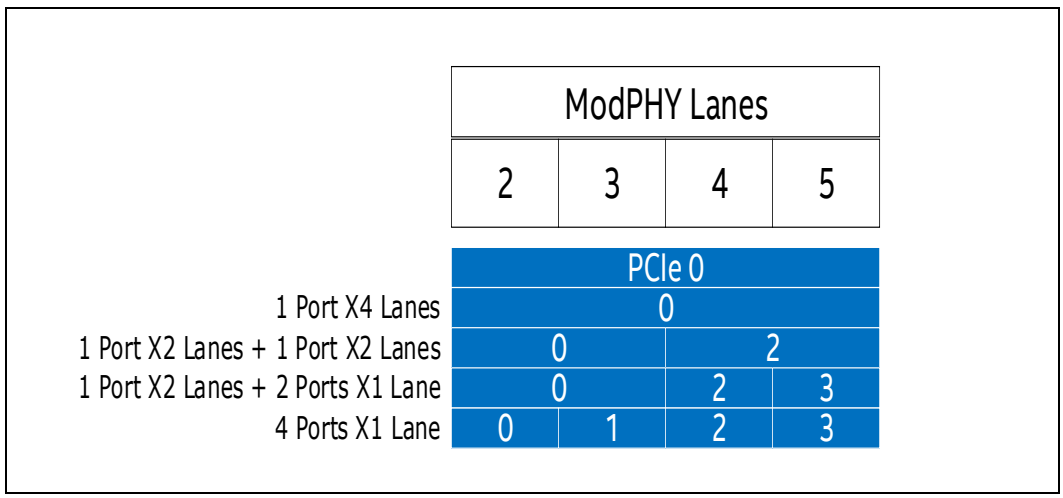
All incoming PCIe* Port transactions will be forwarded onto IOSF with a Traffic Class (TC) value of 0h, regardless of the traffic class of the received transaction.

As PCIe* requires that the same TC value that was sent in a request be returned in the corresponding completion, if the received traffic class was non-zero, actual TC for non-posted requests needs to be stored in the read completion sideways queue so that it can be paired up with the read completion.

12.5.1.3 Single Virtual Channel PCIe* Controller

- 1Port X4 Lanes or
- 2Ports X2 Lanes or
- 1Port X2 Lanes + 2Ports X1 Lane or
- 4Ports X1 Lane

Figure 12-2. Single Virtual Channel PCIe* Controller



12.5.1.4 Multi Virtual Channel PCIe* Controller

- 1 Port X2 Lanes or
- 1 Port X1 Lane

Note: 2 Ports X1 Lane is not supported.

12.5.2 Interrupt Generation

The root port generates interrupts on behalf of hot-plug, power management, link bandwidth management, Link Equalization Request and link error events, when enabled. These interrupts can either be pin-based, or can be Message Signal Interrupt (MSI), when enabled.

When an interrupt is generated using the legacy pin, the pin is internally routed to the processor interrupt controllers. The pin that is driven is based upon the setting of the STRPFUSECFG.PXIP configuration registers.

The following table summarizes interrupt behavior for MSI and wire-modes. In the table "bits" refers to the hot-plug and PME interrupt bits.

Interrupt Register	Wire-Mode Action	MSI Action
All bits 0	Wire inactive	No action
One or more bits set to 1	Wire active	Send message
One or more bits set to 1, new bit gets set to 1	Wire active	Send message
One or more bits set to 1, software clears some (but not all) bits	Wire active	Send message
One or more bits set to 1, software clears all bits	Wire inactive	No action
Software clears one or more bits, and one or more bits are set on the same clock	Wire active	Send message

12.5.3 PCI Express* Power Management

12.5.3.1 S3/S4/S5 Support

Software initiates the transition to S3/S4/S5 by performing an I/O write to the Power Management Control register in the processor. After the I/O write completion has been returned to the processor, the Power Management Controller will signal each root port to send a PME_Turn_Off message on the downstream link. The device attached to the link will eventually respond with a PME_TO_Ack followed by sending a PM_Enter_L23 DLLP (Data Link Layer Packet) request to enter L23. The Express ports and Power Management Controller take no action upon receiving a PME_TO_Ack. When all the Express port links are in state L23, the Power Management Controller will proceed with the entry into S3/S4/S5.

Prior to entering S3, software is required to put each device into $D3_{HOT}$. When a device is put into $D3_{HOT}$, it will initiate entry into a L1 link state by sending a PM_Enter_L1 DLLP. This under normal operating conditions when the root ports sends the PME_Turn_Off message, the link will be in state L1. However, when the root port is instructed to send the PME_Turn_Off message when link was in L1, it will still send the PME_Turn_Off message. Endpoints attached to the PCH can make no assumptions about the state of the link prior to receiving a PME_Turn_Off message.

12.5.3.2 Device Initiated PM_PME Message

When the system has returned to a working state from a previous low power state, a device requesting service will send a PM_PME message continuously, until acknowledged by the root port. The root port will take different actions depending upon whether this is the first PM_PME that has been received, or whether a previous message has been received but not yet serviced by the operating system.

If this is the first message received (RSTS.PS), the root port will set RSTS.PS, and log the PME Requester ID into RSTS.RID. If an interrupt is enabled using RCTL.PIE, an interrupt will be generated. This interrupt can be either a pin or an MSI if MSI is enabled using MC.MSIE.

If this is a subsequent message received (RSTS.PS is already set), the root port will set RSTS.PP. No other action will be taken.

When the first PME event is cleared by software clearing RSTS.PS, the root port will set RSTS.PS, clear RSTS.PP, and move the requester ID into RSTS.RID.

If RCTL.PIE is set, an interrupt will be generated. If RCTL.PIE is not set, a message will be sent to the power management controller so that a GPE can be set. If messages have been logged (RSTS.PS is set), and RCTL.PIE is later written from a 0b to a 1b, an interrupt will be generated. This last condition handles the case where the message was received prior to the operating system re-enabling interrupts after resuming from a low power state.

12.5.3.3 SMI/SCI Generation

Interrupts for power management events are not supported on legacy operating systems. To support power management on non-PCI Express aware operating systems, PM events can be routed to generate SCI. To generate SCI, MPC.PMCE must be set. When set, a power management event will cause SMSCS.PMCS to be set.

Additionally, BIOS workarounds for power management can be supported by setting MPC.PMME. When this bit is set, power management events will set SMSCS.PMMS, and SMI# will be generated. This bit will be set regardless of whether interrupts or SCI is enabled. The SMI# may occur concurrently with an interrupt or SCI.

12.5.3.4 Latency Tolerance Reporting (LTR)

The root port supports the extended Latency Tolerance Reporting (LTR) capability. LTR provides a means for device endpoints to dynamically report their service latency requirements for memory access to the root port. Endpoint devices should transmit a new LTR message to the root port each time its latency tolerance changes (and initially during boot). The PCH uses the information to make better power management decisions. The processor uses the worst case tolerance value communicated by the PCH to optimize C-state transitions. This results in better platform power management without impacting endpoint functionality.

Note: Endpoint devices that support LTR must implement the reporting and enable mechanism detailed in the PCI-SIG “Latency Tolerance Reporting Engineering Change Notice” (www.pcisig.com).

12.5.4 Port 8xh Decode

The PCIe* root ports will explicitly decode and claim I/O cycles within the 80h – 8Fh range when MPC.P8XDE is set. The claiming of these cycles are not subjected to standard PCI I/O Base/Limit and I/O Space Enable fields. This allows a POST-card to be connected to the Root Port either directly as a PCIe* device or through a PCIe* to PCI bridge as a PCI card.

Any I/O reads or writes will be forwarded to the link as it is. The device will need to be able to return the previously written value, on I/O read to these ranges. BIOS must ensure that at any one time, no more than one Root Port is enabled to claim Port 8xh cycles.

12.5.5 Advanced Error Reporting

The PCI Express* Root Ports each provide basic error handling, as well as Advanced Error Reporting (AER) as described in the latest PCI Express Base Specification

12.5.6 Single- Root I/O Virtualization (SR- IOV)

Alternative Routing ID Interpretation (ARI) and Access Control Services (ACS) are supported as part of the complementary technologies to enable SR-IOV capability.

12.5.6.1 Alternative Routing- ID Interpretation (ARI)

Alternative Routing-ID Interpretation (ARI) is a mechanism that can be used to extend the number of functions supported by a multi-function ARI device connected to the Root Port, beyond the conventional eight functions.

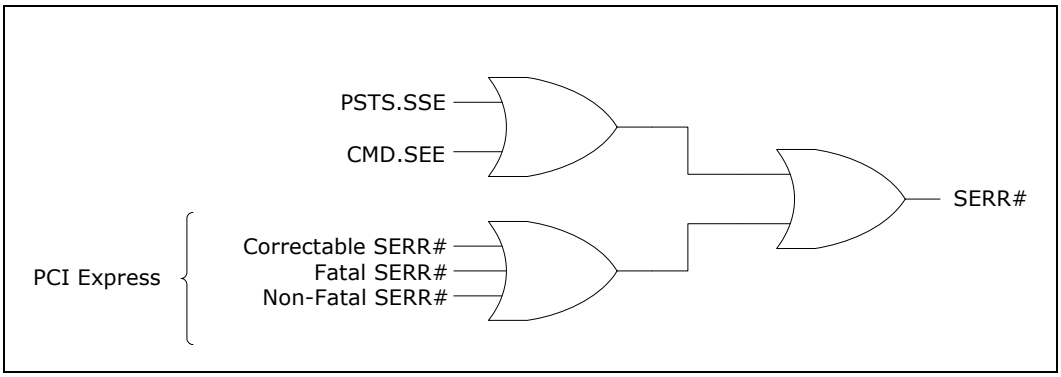
12.5.6.2 Access Control Services (ACS)

ACS is defined to control access between different Endpoints and between different Functions of a multi-function device. ACS defines a set of control points to determine whether a TLP should be routed normally, blocked, or redirected.

12.5.7 SERR# Generation

SERR# may be generated using two paths—through PCI mechanisms involving bits in the PCI header, or through PCI Express* mechanisms involving bits in the PCI Express capability structure.

Figure 12-1. Generation of SERR# to Platform



12.5.8 Hot-Plug

All PCIe* Root Ports support Express Card 1.0 based hot-plug that performs the following:

- Presence Detect and Link Active Changed Support
- Interrupt Generation Support

12.5.8.1 Presence Detection

When a module is plugged in and power is supplied, the physical layer will detect the presence of the device, and the root port sets SLSTS.PDS and SLSTS.PDC. If SLCTL.PDE and SLCTL.HPE are both set, the root port will also generate an interrupt.

When a module is removed (using the physical layer detection), the root port clears SLSTS.PDS and sets SLSTS.PDC. If SLCTL.PDE and SLCTL.HPE are both set, the root port will also generate an interrupt.

The interrupt is generated on an edge-event. For example, if SLSTS.PDC is already set, a change in SLSTS.PDS will not generate a new interrupt. Only SLSTS.PDC going from '0' to '1' will cause an interrupt to be generated.

12.5.8.2 SMI/SCI Generation

Interrupts for power-management events are not supported on legacy operating systems. To support power-management on non-PCI Express aware operating systems, power management events can be routed to generate SCI. To generate SCI, MPC.HPCE must be set. When set, enabled hot-plug events will cause SMSCS.HPCS to be set.

Additionally, BIOS workarounds for hot-plug can be supported by setting MPC.HPME. When this bit is set, hot-plug events can cause SMI status bits in SMSCS to be set. Supported hot-plug events and their corresponding SMSCS bit are:

- Presence Detect Changed – SMSCS.HPPDM
- Link Active State Changed – SMSCS.HPLAS

When any of these bits are set, SMI# will be generated. These bits are set regardless of whether interrupts or SCI is enabled for hot-plug events. The SMI# may occur concurrently with an interrupt or SCI.

12.5.9 PCI Express* Lane Polarity Inversion

The PCI Express* Base Specification requires polarity inversion to be supported independently by all receivers across a Link—each differential pair within each Lane of a PCIe* Link handles its own polarity inversion. Polarity inversion is applied, as needed, during the initial training sequence of a Lane. In other words, a Lane will still function correctly even if a positive (Tx+) signal from a transmitter is connected to the negative (Rx-) signal of the receiver. Polarity inversion eliminates the need to untangle a trace route to reverse a signal polarity difference within a differential pair and no special configuration settings are necessary in the PCH to enable it. It is important to note that polarity inversion does not imply direction inversion or direction reversal; that is, the Tx differential pair from one device must still connect to the Rx differential pair on the receiving device, per the PCIe* Base Specification. Polarity Inversion is not the same as "PCI Express* Controller Lane Reversal".

12.5.10 Precision Time Measurement (PTM)

Hardware protocol for precise coordination of events and timing information across multiple upstream and downstream devices using Transaction Layer Protocol (TLP) Message Requests. Minimizes timing translation errors resulting in the increased coordination of events across multiple components with very fine precision.

All of the PCH PCIe* Controllers and their assigned Root Ports support PTM where each Root Port can have PTM enabled or disabled individually from one another.

12.5.11 PCI Express* Controller Lane Reversal

For each PCIe* Controller we support end-to-end lane reversal across the four lanes mapped to a controller for the two motherboard PCIe* configurations listed below. Lane Reversal means that the most significant lane of a PCIe* Controller is swapped with the least significant lane of the PCIe* Controller while the inner lanes get swapped to preserve the data exchange sequence (order).

Notes: The lane reversal is not applicable for x2 controller (PCIe Controller 1, 2, 3).

PCI Express* Controller Lane Reversal is not the same as PCI Express* Lane Polarity Inversion.

Figure 12-3. PCI Express* Controller Lane Reversal

	PCIe* Lanes		PCIe* Down Device or Connector Lanes													
	PCIe Controller 0 (single VC) Default		Lane Reversal (option 1)		Lane Reversal (option 2)		Lane Reversal (option 3)		Lane Reversal (option 4)		Lane Reversal (option 5)		Lane Reversal (option 6)		Lane Reversal (option 7)	
Setting	Softstrap = N ORM1 = 0 ORM2 = 0		Softstrap = Y ORM1 = 0 ORM2 = 0		Softstrap = N ORM1 = 0 ORM2 = 1		Softstrap = N ORM1 = 1 ORM2 = 0		Softstrap = N ORM1 = 1 ORM2 = 1		Softstrap = Y ORM1 = 1 ORM2 = 0		Softstrap = Y ORM1 = 0 ORM2 = 1		Softstrap = Y ORM1 = 1 ORM2 = 1	
PCIe* Configuration	Port	Lane	Port	Lane	Port	Lane	Port	Lane	Port	Lane	Port	Lane	Port	Lane	Port	Lane
1 x4	1	0	1	3												
	1	1	1	2												
	1	2	1	1												
	1	3	1	0												
2 x2	1	0	3	1	1	0	1	1	1	1	3	0	3	1	3	0
	1	1	3	0	1	1	1	0	1	0	3	1	3	0	3	1
	3	0	1	0	3	1	3	0	3	1	1	1	1	0	1	0
	3	1	1	1	3	0	3	1	3	0	1	0	1	1	1	1
1 x2 + 2 x1	1	0	4	0												
	1	1	3	0												
	3	0	1	1												
	4	0	1	0												
4 x1	1	0	4	0												
	2	0	3	0												
	3	0	2	0												
	4	0	1	0												

12.6 Registers

Note: Please refer to chapter 11 of the Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors for Internet of Things (IoT) Applications, Datasheet, Volume 2 (Book 2 of 3), Mule Creek Canyon (Document Number: 636722), for a description of the registers associated with subject of this chapter.

§ §

13 Serial ATA (SATA)

13.1 Acronyms

Acronyms	Description
AHCI	Advanced Host Controller Interface
DMA	Direct Memory Access
DEVSLP	Device Sleep
IDE	Integrated Drive Electronics
RAID	Redundant Array of Independent Disks
SATA	Serial Advanced Technology Attachment

13.2 References

Specification	Location
Serial ATA Specification, Revision 3.2	https://www.sata-io.org
Serial ATA II: Extensions to Serial ATA 1.0, Revision 1.0	https://www.sata-io.org
Serial ATA II Cables and Connectors Volume 2 Gold	https://www.sata-io.org
Advanced Host Controller Interface Specification	http://www.intel.com/content/www/us/en/io/serial-ata/ahci.html

13.3 Overview

The PCH SATA controller support two modes of operation, AHCI mode using memory space. The PCH SATA controller no longer supports IDE legacy mode using I/O space. Therefore, AHCI software is required. The PCH SATA controller supports the Serial ATA Specification, Revision 3.2.

13.4 I/O Signal Planes and States

Signal Name	Power Plane	During Reset	Immediately after Reset	S3/S4/S5	Deep Sx
SATA_LED_N/ GP_E00	Primary	Undriven	Undriven	Undriven	Off
SATA_[1:0]_DEVSLP/ GP_E[04,08][1]	Primary	Undriven	Undriven	Undriven	Off
SATA_[1:0]_GP/ GP_E00, GP_G12[2]	Primary	Undriven	Undriven	Undriven	Off
SATA_[1:0]_RXP/RXN SATA_[1:0_TXP/TXN	Primary	Internal Pull-Down	Internal Pull-Down	Internal Pull-Down	Off

Signal Name	Power Plane	During Reset	Immediately after Reset	S3/S4/S5	Deep Sx

Note:
1. Pin defaults to GPIO mode. The pin state during and immediately after reset follows default GPIO mode pin state. The pin state for S0 to Deep Sx reflects assumption that GPIO Use Select register was programmed to native mode functionality. If GPIO Use Select register is programmed to GPIO mode, refer to Multiplexed GPIO (Defaults to GPIO Mode) section for the respective pin states in S0 to Deep Sx.
2. Pin defaults to Native mode as SATAXPCIEx depends on soft-strap.

13.5 Functional Description

The PCH SATA host controller (D23:F0) supports AHCI mode.

The PCH SATA controller does not support legacy IDE mode or combination mode.

The PCH SATA controller interacts with an attached mass storage device through a register interface that is compatible with an SATA AHCI host adapter. The host software follows existing standards and conventions when accessing the register interface and follows standard command protocol conventions.

13.5.1 SATA 6 Gb/s Support

The PCH SATA controller is SATA 6 Gb/s capable and supports 6 Gb/s transfers with all capable SATA devices. The PCH SATA controller also supports SATA 3 Gb/s and 1.5 Gb/s transfer capabilities.

13.5.2 SATA Feature Support

The PCH SATA controller is capable of supporting all AHCI 1.3 and AHCI 1.3.1, refer to the Intel web site on Advanced Host Controller Interface Specification for current specification status: http://www.intel.com/content/www/us/en/io/serial-ata/ahci.html.

For capability details, refer to PCH SATA controller register (D23:F0:Offset 00h CAP, and AHCI BAR PxCMD Offset 18h).

The PCH SATA controller does **not** support:

- Port Multiplier
- FIS Based Switching
- Command Based Switching
- IDE mode or combination mode
- Cold Presence Detect
- Function Level Reset (FLR)
- Redundant Array of Independent Disks (RAID).

13.5.3 Hot-Plug Operation

The PCH SATA controller supports Hot-Plug Surprise removal and Insertion Notification. An internal SATA port with a Mechanical Presence Switch can support PARTIAL and SLUMBER with Hot-Plug Enabled. Software can take advantage of power savings in the low power states while enabling Hot-Plug operation.

13.5.4 Power Management Operation

Power management of the PCH SATA controller and ports will cover operations of the host controller and the SATA link.

13.5.4.1 Power State Mappings

The D0 PCI Power Management (PM) state for device is supported by the PCH SATA controller.

SATA devices may also have multiple power states. SATA adopted 3 main power states from parallel ATA. The three device states are supported through ACPI. They are:

- **D0** – Device is working and instantly available.
- **D1** – Device enters when it receives a STANDBY IMMEDIATE command. Exit latency from this state is in seconds.
- **D3** – From the SATA device's perspective, no different than a D1 state, in that it is entered using the STANDBY IMMEDIATE command. However, an ACPI method is also called which will reset the device and then cut its power.

Each of these device states are subsets of the host controller's D0 state.

Finally, the SATA specification defines three PHY layer power states, which have no equivalent mappings to parallel ATA. They are:

- **PHY READY** – PHY logic and PLL are both on and in active state.
- **Partial** – PHY logic is powered up, and in a reduced power state. The link PM exit latency to active state maximum is 10 ns.
- **Slumber** – PHY logic is powered up, and in a reduced power state. The link PM exit latency to active state maximum is 10 ms.
- **Devslp** – PHY logic is powered down. The link PM exit latency from this state to active state maximum is 20 ms, unless otherwise specified by DETO in Identify Device Data Log page 08h (Refer SATA Rev3.2 Gold specification).

Since these states have much lower exit latency than the ACPI D1 and D3 states, the SATA controller specification defines these states as sub-states of the device D0 state.

13.5.4.2 Power State Transitions

13.5.4.2.1 Partial and Slumber State Entry/Exit

The partial and slumber states save interface power when the interface is idle. It would be most analogous to CLKRUN# (in power savings, not in mechanism), where the interface can have power saved while no commands are pending. The SATA controller defines PHY layer power management (as performed using primitives) as a driver operation from the host side, and a device proprietary mechanism on the device side. The SATA controller accepts device transition types, but does not issue any transitions as a host. All received requests from a SATA device will be ACKed.

When an operation is performed to the SATA controller such that it needs to use the SATA cable, the controller must check whether the link is in the Partial or Slumber states, and if so, must issue a COMWAKE to bring the link back online. Similarly, the SATA device must perform the same COMWAKE action.

Note: SATA devices shall not attempt to wake the link using COMWAKE/COMINIT when no commands are outstanding and the interface is in Slumber.

13.5.4.2.2 DEVSLP State Entry/Exit

Device Sleep (DEVSLP) is a host-controlled SATA interface power state. To support a hardware autonomous approach that is software agnostic Intel is recommending that BIOS configure the AHCI controller and the device to enable Device Sleep. This allows the AHCI controller and associated device to automatically enter and exit Device Sleep without the involvement of OS software.

To enter Device Sleep the link must first be in Slumber. By enabling HIPM (with Slumber) or DIPM on a Slumber capable device, the device/host link may enter the DevSleep Interface Power state.

The device must be DevSleep capable. Device Sleep is only entered when the link is in slumber, therefore when exiting the Device Sleep state, the device must resume with the COMWAKE out-of-band signal (and not the COMINIT out-of-band signal). Assuming Device Sleep was asserted when the link was in slumber, the device is expected to exit DEVSLP to the DR_Slumber state. Devices that do not support this feature will not be able to take advantage of the hardware automated entry to Device Sleep that is part of the AHCI 1.3.1 specification and supported by Intel platforms.

13.5.4.2.3 Device D1 and D3 States

These states are entered after some period of time when software has determined that no commands will be sent to this device for some time. The mechanism for putting a device in these states does not involve any work on the host controller, other then sending commands over the interface to the device. The command most likely to be used in ATA/ATAPI is the “STANDBY IMMEDIATE” command.

13.5.4.2.4 Host Controller $D3_{HOT}$ State

After the interface and device have been put into a low power state, the SATA host controller may be put into a low power state. This is performed using the PCI power management registers in configuration space. There are two very important aspects to Note when using PCI power management.

1. When the power state is D3, only accesses to configuration space are allowed. Any attempt to access the memory or I/O spaces will result in initiator abort.
2. When the power state is D3, no interrupts may be generated, even if they are enabled. If an interrupt status bit is pending when the controller transitions to D0, an interrupt may be generated.

When the controller is put into D3, it is assumed that software has properly shut down the device and disabled the ports. Therefore, there is no need to sustain any values on the port wires. The interface will be treated as if no device is present on the cable, and power will be minimized.

When returning from a D3 state, an internal reset will not be performed.

13.5.4.3 Low Power Platform Consideration

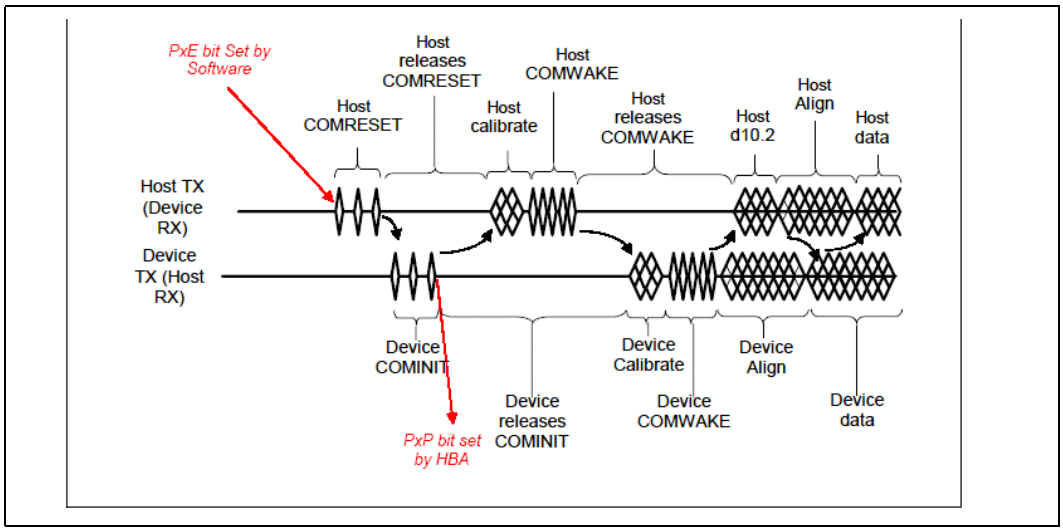
When low power feature is enabled, the Intel SATA controller may power off PLLs or OOB detection circuitry while in the Slumber link power state. As a result, a device initiated wake may not be recognized by the host. For example, when the low power feature is enabled it can prevent a Zero Power ODD (ZPODD) device from successfully communicating with the host on media insertion.

The SATA MPHY Dynamic Power Gating (PHYDPGEPx) can be enabled/disabled for each SATA ports.

13.5.5 SATA Device Presence

The flow used to indicate SATA device presence is shown in Figure 13-1. The 'PxE' bit refers to bits, depending on the port being checked and the 'PxP' bits refer to the bits, depending on the port being checked. If the PCS/PxP bit is set a device is present, if the bit is cleared a device is not present. If a port is disabled, software can check to see if a new device is connected by periodically re-enabling the port and observing if a device is present, if a device is not present it can disable the port and check again later. If a port remains enabled, software can periodically poll PCS.PxP to see if a new device is connected.

Figure 13-1. Flow for Port Enable/Device Present Bits



13.5.6 SATA LED

The SATA_LED_N output is driven whenever the BSY bit is set in any SATA port. The SATA_LED_N is an active-low open-drain output. When SATA_LED_N is low, the LED should be active. When SATA_LED_N is high, the LED should be inactive.

13.5.7 Advanced Host Controller Interface (AHCI) Operation

The PCH SATA controller provides hardware support for Advanced Host Controller Interface (AHCI), a standardized programming interface for SATA host controllers developed through a joint industry effort. Platforms supporting AHCI may take advantage of performance features such as port independent DMA Engines—each device is treated as a initiator—and hardware-assisted native command queuing.

AHCI defines transactions between the SATA controller and software and enables advanced performance and usability with SATA. Platforms supporting AHCI may take advantage of performance features such as no initiator/target (refer to Section 1.1.1 for more information on initiator and target) designation for SATA devices—each device is treated as a initiator—and hardware assisted native command queuing. AHCI also provides usability enhancements such as hot-plug and advanced power management. AHCI requires appropriate software support (such as, an AHCI driver) and for some features, hardware support in the SATA device or additional platform hardware. Visit the Intel web site for current information on the AHCI specification.

The PCH SATA controller supports all of the mandatory features of the *Serial ATA Advanced Host Controller Interface Specification*, Revision 1.3.1 and many optional features, such as hardware assisted native command queuing, aggressive power management, LED indicator support, and hot-plug through the use of interlock switch support (additional platform hardware and software may be required depending upon the implementation).

Note: For reliable device removal notification while in AHCI operation without the use of interlock switches (surprise removal), interface power management should be disabled for the associated port. See Section 7.3.1 of the *AHCI Specification* for more information.

13.6 Registers

Note: Please refer to the Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors for Internet of Things (IoT) Applications, Datasheet, Volume 2 (Book 2 of 3), Mule Creek Canyon (Document Number: 636722), for a description of the registers associated with subject of this chapter.

§ §

14 Host System Management Bus (SMBus) Controller

14.1 Functional Description

The Intel PCH provides an SMBus 2.0-compliant host controller. The host controller provides a mechanism for the CPU to initiate communications with SMB peripherals (targets). Refer to Section 1.1.1 for more information on target. The Intel PCH is also capable of operating in a mode in which it can communicate with I^2C compatible devices.

14.2 Signal Description

Name	Type	Description
SMB_CLK/ GPP_C00	I/OD	**SMBus Clock**. External Pull-up resistor is required
SMB_DATA/ GPP_C01	I/OD	**SMBus Data.** External Pull-up resistor is required.
SMB_ALERT_N/ GPP_C02	I/OD	**SMBus Alert:** This signal is used to wake the system or generate SMI#. External Pull-up resistor is required.

14.3 Host Controller

The PCH can perform SMBus messages with either Packet Error Checking (PEC) enabled or disabled. The actual PEC calculation and checking is performed in SW. The SMBus host controller logic can automatically append the CRC byte if configured to do so.

The SMBus Address Resolution Protocol (ARP) is supported by using the existing host controller commands through software, except for the Host Notify command (which is actually a received message).

The PCH SMBus host controller checks for parity errors as a target. If an error is detected, the detected parity error bit in the PCI Status Register is set.

14.3.1 Host Controller Operation Overview

The SMBus host controller is used to send commands to other SMBus target devices. Software sets up the host controller with an address, command, and, for writes, data and optional PEC; and then tells the controller to start. When the controller has finished transmitting data on writes, or receiving data on reads, it generates an SMI# or interrupt, if enabled.

The host controller supports 8 command protocols of the SMBus interface (see *System Management Bus (SMBus) Specification,* Version 2.0): Quick Command, Send Byte, Receive Byte, Write Byte/Word, Read Byte/Word, Process Call, Block Read/Write, and Block Write–Block Read Process Call.

The SMBus host controller requires that the various data and command fields be setup for the type of command to be sent. When software sets the START bit, the SMBus Host controller performs the requested transaction, and interrupts the processor (or generates an SMI#) when the transaction is completed. Once a START command has been issued, the values of the "active registers" (Host Control, Host Command, Transmit target Address, Data 0, Data 1) should not be changed or read until the interrupt status message (INTR) has been set (indicating the completion of the command). Any register values needed for computation purposes should be saved prior to issuing of a new command, as the SMBus host controller updates all registers while completing the new command.

Target functionality, including the Host Notify protocol, is available on the SMBus pins.

Using the SMB host controller to send commands to the PCH SMB target port is not supported.

14.3.2 Command Protocols

In all of the following commands, the Host Status Register (offset 00h) is used to determine the progress of the command. While the command is in operation, the HBSY bit is set. If the command completes successfully, the INTR bit will be set in the Host Status Register. If the device does not respond with an acknowledge, and the transaction times out, the DERR bit is set.

If the software sets the KILL bit in the Host Control Register while the command is running, the transaction will stop and the FAILED bit will be set after Intel PCH forces a time-out. In addition, if KILL bit is set during the CRC cycle, both the CRCE and DERR bits will also be set.

Quick Command

When programmed for a Quick Command, the Transmit Target Address Register is sent.The PEC byte is never appended to the Quick Protocol. Software should force the PEC_EN bit to '0' when performing the Quick Command. Software must force I2C_EN set produces undefined results. Software must force the I2C_EN bit to 0 when running this command.

Send Byte/Receive Byte

For the Send Byte command, the Transmit Target Address and Device Command Registers are sent.The Receive Byte is similar to a Send Byte, the only difference being the direction of data transfer. When programmed for the receive byte command, the Transmit Target Address Register is sent. Software must force the I2C_EN bit to 0 when running this command.

Write Byte/Word

The first byte of a Write Byte/Word access is the command code. The next 1 or 2 bytes are the data to be written. When programmed for a Write Byte/Word command, the Transmit Target Address, Device Command, and Data0 Registers are sent. In addition, the Data1 Register is sent on a Write Word command. Software must force the I2C_EN bit to 0 when running this command.

Read Byte/Word

Reading data is slightly more complicated than writing data. First the PCH must write a command to the target device. Then it must follow that command with a repeated start condition to denote a read from that device's address. The target then returns 1 or 2 bytes of data. Software must force the I2C_EN bit to 0 when running this command.

When programmed for the read byte/word command, the Transmit Target Address and Device Command Registers are sent. Data is received into the DATA0 on the read byte, and the DAT0 and DATA1 registers on the read word.

Process Call

The process call is so named because a command sends data and waits for the target to return a value dependent on that data. The protocol is simply a Write Word followed by a Read Word, but without a second command or stop condition.

When programmed for the process call command, the Intel PCH transmits the Transmit Address, Device Command, and DATA0 and DATA1 registers. Data received from the device is stored in the DATA0 and DATA1 registers. The value written into bit 0 of the Transmit Target Address Register (SMBus Offset 04h) needs
to programmed to 0.

The Process Call command with I2C_EN set and either the PEC_EN or AAC bit set produces undefined results. Software must either force the I2C_EN bit or both PEC_EN and AAC bits to 0 when running this command.

Note: If the I2C_EN bit is set, then the Command field will not be sent.

Block Read/Write

The Intel PCH contains a 32-byte buffer for read and write data which can be enabled by setting bit ‘1’ of the Auxiliary Control register at offset 0Dh in I/O space, as opposed to a single byte of buffering. This 32-byte buffer is filled with write data before transmission, and filled with read data on reception.

In the PCH, the interrupt is generated only after a transmission or reception of 32 bytes, or when the entire byte count has been transmitted/received.
The byte count field is transmitted but ignored by the hardware as software will end the transfer after all bytes it cares about have been sent or received

For Block Write command software must either force the I2C_EN bit or both PEC_EN and AAC bits to 0 when running this command.

I²C* Read

This command allows the PCH to perform block reads to certain I²C devices, such as serial E²PROMs. The SMBus Block Read supports the 7-bit addressing mode only.

However, this does not allow access to devices using the I²C "Combined Format" that has data bytes after the address. Typically these data bytes correspond to an offset (address) within the serial memory chips.

Note: The I²C Read command with either PEC_EN or AAC it set produces undefined results. Software must force both PEC_EN and AAC bits to 0 when running this command
This new command is supported independent of the setting of the I2C_EN bit.
The value written into bit 0 of the Transmit Target Address Register (SMBus Offset 04h) must be 0.

The format that is used for the command is shown in Table 14-1.

Table 14-1. I²C* Multi-Byte Read

Bit	Description
1	Start
2:8	Target Address – 7 bits
9	Write
10	Acknowledge from target
11:18	Send DATA1 register
19	Acknowledge from target
20	Repeated Start
21:27	Target Address – 7 bits
28	Read
29	Acknowledge from target
30:37	Data byte 1 from target – 8 bits
38	Acknowledge
39:46	Data byte 2 from target – 8 bits
47	Acknowledge
–	Data bytes from target/Acknowledge
–	Data byte N from target – 8 bits
–	NOT Acknowledge
–	Stop

The PCH will continue reading data from the peripheral until the NAK is received.

Block Write–Block Read Process Call

The block write-block read process call is a two-part message. The call begins with a target address and a write condition. After the command code the host issues a write byte count (M) that describes how many more bytes will be written in the first part of the message. If a initiator has 6 bytes to send, the byte count field will have the value

6 (0000 0110b), followed by the 6 bytes of data. The write byte count (M) cannot be zero.

The second part of the message is a block of read data beginning with a repeated start condition followed by the target address and a Read bit. The next byte is the read byte count (N), which may differ from the write byte count (M). The read byte count (N) cannot be zero.
The combined data payload must not exceed 32 bytes. The byte length restrictions of this process call are summarized as follows:

- M ≥ 1 byte
- N ≥ 1 byte
- M + N ≤ 32 bytes

The read byte count does not include the PEC byte. The PEC is computed on the total message beginning with the first target address and using the normal PEC computational rules. It is highly recommended that a PEC byte be used with the Block Write-Block Read Process Call. Software must do a read to the command register (offset 2h) to reset the 32byte buffer pointer prior to reading the block data register.

Note: There is no STOP condition before the repeated START condition, and that a NACK signifies the end of the read transfer.

Note: E32B bit in the Auxiliary Control register must be set when using this protocol.

14.3.2.1 Bus Arbitration

Several initiators may attempt to get on the bus at the same time by driving the SMBDATA line low to signal a start condition. The Intel PCH must continuously monitor the SMBDATA line. When the Intel PCH is attempting to drive the bus to a '1' by letting go of the SMBDATA line, and it samples SMBDATA low, then some other initiator is driving the bus and the Intel PCH must stop transferring data.

If the Intel PCH loses arbitration, the condition is called a collision. The Intel PCH sets the BUS_ERR bit in the Host Status Register, and if enabled, generates an interrupt or SMI#. The CPU is responsible for restarting the transaction.

14.3.2.2 Clock Stretching

Some devices may not be able to handle their clock toggling at the rate that the Intel PCH as an SM Bus initiator would like. They have the capability of stretching the low time of the clock.
When the Intel PCH attempts to release the clock (allowing the clock to go high), the clock will remain low for an extended period of time.

The Intel PCH monitors the SM Bus clock line after it releases the bus to determine whether to enable the counter for the high time of the clock. While the bus is still low, the high time counter must not be enabled. Similarly, the low period of the clock can be stretched by an SMBus initiator if it is not ready to send or receive data.

14.3.2.3 Bus Timeout (PCH as SMBus initiator)

If there is an error in the transaction, such that an SM Bus device does not signal an acknowledge, or holds the clock lower than the allowed time-out time, the transaction will time out. The Intel PCH will discard the cycle, and set the DERR bit. The time out minimum is 25ms. The time-out counter inside the Intel PCH will start when the first t bit of data is transferred by the Intel PCH. The 25 ms will be a count of 800 RTC clocks.

The 25-ms Timeout counter will not count under the following conditions:

1. BYTE_DONE_STATUS bit (SMBus I/O Offset 00h, Bit 7) is set
2. The SECOND_TO_STS bit (TCO I/O Offset 06h, Bit 1) is not set (this indicates that the system has not locked up).

14.3.2.4 Interrupts/SMI#

The PCH SMBus controller uses PIRQB# as its interrupt pin. However, the system can alternatively be set up to generate SMI# instead of an interrupt, by setting the SMBUS_SMI_EN bit.

Table 14-2, Table 14-3 and Table 14-4 specify how the various enable bits in the SMBus function control the generation of the interrupt, Host and Target SMI, and Wake internal signals. The rows in the tables are additive, which means that if more than one row is true for a particular scenario then the Results for all of the activated rows will occur.

Table 14-2. Enable for SMB_ALERT_N

Event	INTREN (Host Control I/O Register, Offset 02h, Bit 0)	SMB_SMI_EN (Host Config Register, D31:F4:Offset 40h, Bit 1)	SMBALERT_DIS (Target Command I/O Register, Offset 11h, Bit 2)	Result
SMB_ALERT_N asserted low (always reported in SMBALERT_STS-Host Status Register, bit 5)	X	X	X	Wake generated (Depends on CSE setting)
	X	1	0	Target SMI# generated (SMBUS_SMI_STS)
	1	0	0	Interrupt generated

Table 14-3. Enables for SMBus Target Write and SMBus Host Events

Event	INTREN (Host Control I/O Register, Offset 02h, Bit 0)	SMB_SMI_EN (Host Config Register, D31:F4:Offset 40h, Bit 1)	Result
Target Write to Wake/SMI# Command	X	X	Wake generated when asleep. Target SMI# generated when awake (SMBUS_SMI_STS).
Target Write to SMLINK_Initiator_SMI Command	X	X	Target SMI# generated when in the S0 state (SMBUS_SMI_STS)
Any combination of Host Status Register [4:1] asserted	0	X	None
	1	0	Interrupt generated
	1	1	Host SMI# generated

Table 14-4. Enables for the Host Notify Command

HOST_NOTIFY_INTREN (target Control I/O Register, Offset 11h, Bit 0)	SMB_SMI_EN (Host Config Register, D31:F4:Off40h, Bit 1)	HOST_NOTIFY_WKEN (target Control I/O Register, Offset 11h, Bit 1)	Result
0	X	0	None
X	X	1	Wake generated (Depends on CSE setting)
1	0	X	Interrupt generated
1	1	X	Target SMI# generated (SMBUS_SMI_STS)

14.3.2.5 SMBus CRC Generation and Checking

If the AAC bit is set in the Auxiliary Control register, the PCH automatically calculates and drives CRC at the end of the transmitted packet for write cycles, and will check the CRC for read cycles. It will not transmit the contents of the PEC register for CRC. The PEC bit must not be set in the Host Control register if this bit is set, or unspecified behavior will result.

If the read cycle results in a CRC error, the DERR bit and the CRCE bit in the Auxiliary Status register at Offset 0Ch will be set.

14.3.3 SMBus Target Interface

The PCH SMBus target interface is accessed using the SMBus. The SMBus target logic will not generate or handle receiving the PEC byte and will only act as a Legacy Alerting Protocol Device.The target interface allows the PCH to decode cycles, and allows an external microcontroller to perform specific action.

Key features and capabilities include:

- Supports decode of three types of messages: Byte Write, Byte Read, and Host Notify
- Register for the receive target address. This is the address that the Intel PCH decodes. A default value is provided so that the target interface can be used without the CPU having to program this register.
- "Receive Target Data" register in the SMBus I/O space that includes the data written by the external microcontroller
- Registers that the external microcontroller can read to get the state of the Intel PCH
  - — Status bits to indicate that the SMBus target logic caused an interrupt or SMI#
  - — Bit 0 of the Target Status Register for the Host Notify command
  - — Bit 16 of the SMI Status Register for all others

Note: The external microcontroller should not attempt to access the PCH SMBus target logic until either:

- — 800 milliseconds after both: RTC_TEST_N is high and RTC_RST_N is high, OR
- — The PMC_PLTRST_N de-asserts

If a initiator leaves the clock and data bits of the SMBus interface at 1 for 50 µs or more in the middle of a cycle, the Intel PCH target logic's behavior is undefined. This is interpreted as an unexpected idle and should be avoided when performing management activities to the target logic.

14.3.3.1 Format of Target Write Cycle

The external initiator performs Byte Write commands to the Intel PCH SMBus Target I/F. The "Command" field (bits 11:18) indicate which register is being accessed. The Data field (bits 20:27) indicate the value that should be written to that register.

Table 14-5 has the values associated with the registers.

Table 14-5. Target Write Registers

Register	Function
0	Command Register. See Table 14-6 for valid values written to this register.
1–3	Reserved
4	Data Message Byte 0
5	Data Message Byte 1
6–7	Reserved
8	Reserved
9–FFh	Reserved
Note: The external microcontroller is responsible to make sure that it does not update the contents of the data byte registers until they have been read by the system CPU. The Intel PCH overwrites the old value with any new value received. A race condition is possible where the new value is being written to the register just at the time it is being read. The PCH will not attempt to cover this race condition (that is, unpredictable results in this case).	

Table 14-6. Command Types (Sheet 1 of 2)

Command Type	Description
0	Reserved
1	**WAKE/SMI#.** This command wakes the system if it is not already awake. If system is already awake, an SMI# is generated.
2	**Unconditional Powerdown.** This command sets the PWRBTNOR_STS bit, and has the same effect as the Powerbutton Override occurring.
3	**HARD RESET WITHOUT Power CYCLING:** This command causes a soft reset of the system (does not include cycling of the power supply). This is equivalent to a write to the CF9h register with Bits 2:1 set to 1, but Bit 3 set to 0.
4	**HARD RESET SYSTEM.** This command causes a hard reset of the system (including cycling of the power supply). This is equivalent to a write to the CF9h register with Bits 3:1 set to 1.
5	**Disable the TCO Messages.** This command will disable the PCH from sending Heartbeat and Event messages. Once this command has been completed, there is no method to re-enable the Heartbeat and Event Messages, until RSMRST# goes low and then high.
6	**WD RELOAD:** Reload watchdog timer.
7	Reserved

Table 14-6. Command Types (Sheet 2 of 2)

Command Type	Description
8	**SMLINK_Initiator_SMI.** When the Intel PCH detects this command type while in the S0 state, it sets the SMLINK_Initiator_SMI_STS bit. This command should only be used if the system is in an S0 state. If the message is received during S1–S5 states, the PCH acknowledges it, but the SMLINK_Initiator_SMI_STS bit does not get set. ***Note:*** It is possible that the system transitions out of the S0 state at the same time that the SMLINK_Initiator_SMI command is received. In this case, the SMLINK_Initiator_SMI_STS bit may get set but not serviced before the system goes to sleep. Once the system returns to S0, the SMI associated with this bit would then be generated. Software must be able to handle this scenario.
9–FFh	Reserved.

14.3.3.2 Format of Read Command

The external initiator performs Byte Read commands to the Intel PCH SMBus target interface. The "Command" field (bits 11:18) indicate which register is being accessed. The Data field (bits 30:37) contain the value that should be read from that register.

Table 14-7. Target Read Cycle Format

Bit	Description	Driven By	Comment
1	Start	External Microcontroller	
2–8	Target Address - 7 bits	External Microcontroller	Must match value in Receive Target Address register
9	Write	External Microcontroller	Always 0
10	ACK	Intel PCH	
11–18	Command code – 8 bits	External Microcontroller	Indicates which register is being accessed. See Table 14-8 for a list of implemented registers.
19	ACK	Intel PCH	
20	Repeated Start	External Microcontroller	
21–27	Target Address - 7 bits	External Microcontroller	Must match value in Receive Target Address register
28	Read	External Microcontroller	Always 1
29	ACK	Intel PCH	
30–37	Data Byte	Intel PCH	Value depends on register being accessed. See Table 14-8 for a list of implemented registers.
38	NOT ACK	External Microcontroller	
39	Stop	External Microcontroller	

Table 14-8. Data Values for Target Read Registers (Sheet 1 of 2)

Register	Bits	Description
0	7:0	Reserved
1	2:0	System Power State 000 = S0 001 = Reserved 010 = Reserved 011 = S3 100 = S4 101 = S5 110 = Reserved 111 = Reserved
	7:3	Reserved
2	3:0	Reserved
	7:4	Reserved
3	5:0	**Watchdog Timer current value** ***Note:*** The Watchdog Timer has 10 bits, but this field is only 6 bits. If the current value is greater than 3Fh, the Intel PCH will always report 3Fh in this field.
	7:6	Reserved
4	0	1 = **Intruder Detect** (INTRD_DET) bit is set. This indicates that the system cover has probably been opened.
	1	Reserved
	2	Reserved
	3	1 = **SECOND_TO_STS** bit set. This bit will be set after the second Timeout (SECOND_TO_STS bit) of the Watchdog Timer occurs.
	6:4	Reserved. Will always be 0, but software should ignore.
	7	**SMB_ALERT_N Status.** Reflects the value of the SMB_ALERT_N pin (when the pin is configured to SMB_ALERT_N). Valid only if SMBALERT_DISABLE = 0. Value always returns 1 if SMBALERT_DISABLE = 1.
5	0	**Reserved**
	1	**Battery Low Status.** '1' if the BATLOW# pin is a '0'.
	2	**SYS_PWROK Failure Status:** This bit will be 1 if the SYSPWR_FLR bit in the GEN_PMCON_2 register is set.
	3	**INIT# due to receiving Shutdown message:** This event is visible from the reception of the shutdown message until a platform reset is done if the Shutdown Policy Select bit (SPS) is configured to drive INIT#. When the SPS bit is configured to generate PLTRST# based on shutdown, this register bit will always return 0. Events on signal will not create a event message
	4	**LT Reset**: LT reset indication. Events on signal will not create a event message
	5	**POWER_OK_BAD:** Indicates the failure core power well ramp during boot/resume. This bit will be active if the SLP_S3# pin is de-asserted and PCH_PWROK pin is not asserted.
	6	**Thermal Trip**: This bit will shadow the state of processor Thermal Trip status bit (CTS)Events on signal will not create a event message
	7	Reserved: Default value is "X" ***Note:*** Software should not expect a consistent value when this bit is read through SMBUS
6	7:0	Contents of the Message 1 register.
7	7:0	Contents of the Message 2 register.
8	7:0	Contents of the WDSTATUS register.
9	7:0	Seconds of the RTC
A	7:0	Minutes of the RTC

Table 14-8. Data Values for Target Read Registers (Sheet 2 of 2)

Register	Bits	Description
B	7:0	Hours of the RTC
C	7:0	"Day of Week" of the RTC
D	7:0	"Day of Month" of the RTC
E	7:0	Month of the RTC
F	7:0	Year of the RTC
10h–FFh	7:0	Reserved

14.3.3.2.1 Behavioral Notes

The SMBus protocol always has either Start bit-Address-Write bit or Repeated Start bit-Address-Read bit. The Intel PCH is implemented such that the read/write bit in the repeated start phase is ignored with an assumption that the protocol always followed. In other words, if start-address-read occurs (which is illegal for SMBus byte read protocol), the Intel PCH will still grab the cycle. In another case, if a repeated start-address-write sequence occurs, then the cycle will continue as a target read.

14.3.3.3 Target Read of RTC Time Bytes

The PCH SMBus target interface allows external SMBus initiator to read the internal RTC's time byte registers.

The RTC time bytes are internally latched by the PCH's hardware whenever RTC time is not changing and SMBus is idle. This ensures that the time byte delivered to the target read is always valid and it does not change when the read is still in progress on the bus. The RTC time will change whenever hardware update is in progress, or there is a software write to the RTC time bytes.

The PCH SMBus target interface only supports Byte Read operation. The external SMBus initiator such as BMC will read the RTC time bytes one after another It is software's responsibility to check and manage the possible time rollover when subsequent time bytes are read.

For example, assuming the RTC time is 11 hours:59 minutes: 59 seconds. When BMC reads the hour as 11, and then proceeds to read the minute, it is possible that the rollover happens between the reads and the minute is read as 0. This results in 11 hours: 0 minute instead of
the correct time of 12 hours: 0 minute. Unless it is certain that rollover will not occur, software is required to detect the possible time rollover by reading multiple times such that the read time bytes can be adjusted accordingly if needed.

14.3.3.4 Format of Host Notify Command

The Intel PCH tracks and responds to the standard Host Notify command as specified in the *System Management Bus (SMBus) Specification,* Version 2.0. The host address for this command is fixed to 0001000b. If the Intel PCH already has data for a previously-received host notify command which has not been serviced yet by the host software (as indicated by the HOST_NOTIFY_STS bit), then it will NACK following the host address

byte of the protocol. This allows the host to communicate non-acceptance to the initiator and retain the host notify address and data values for the previous cycle until host software completely services the interrupt.

Note: Host software must always clear the HOST_NOTIFY_STS bit after completing any necessary reads of the address and data registers.

Table 14-9 shows the Host Notify format.

Table 14-9. Host Notify Format

Bit	Description	Driven By	Comment
1	Start	External initiator	
2:8	SMB Host Address – 7 bits	External initiator	Always 0001_000
9	Write	External initiator	Always 0
10	ACK (or NACK)	Intel PCH	Intel PCH NACKs if HOST_NOTIFY_STS is 1
11:17	Device Address – 7 bits	External initiator	Indicates the address of the initiator; loaded into the Notify Device Address Register
18	Unused – Always 0	External initiator	7-bit-only address; this bit is inserted to complete the byte
19	ACK	Intel PCH	
20:27	Data Byte Low	External initiator	Loaded into the Notify Data Low Byte Register
28	ACK	Intel PCH	
29:36	Data Byte High	External initiator	Loaded into the Notify Data High Byte Register
37	ACK	Intel PCH	
38	Stop	External initiator	

14.4 Registers

Note: Please refer to chapter 6 of the Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors for Internet of Things (IoT) Applications, Datasheet, Volume 2 (Book 2 of 3), Mule Creek Canyon (Document Number: 636722), for a description of the registers associated with subject of this chapter.

§ §

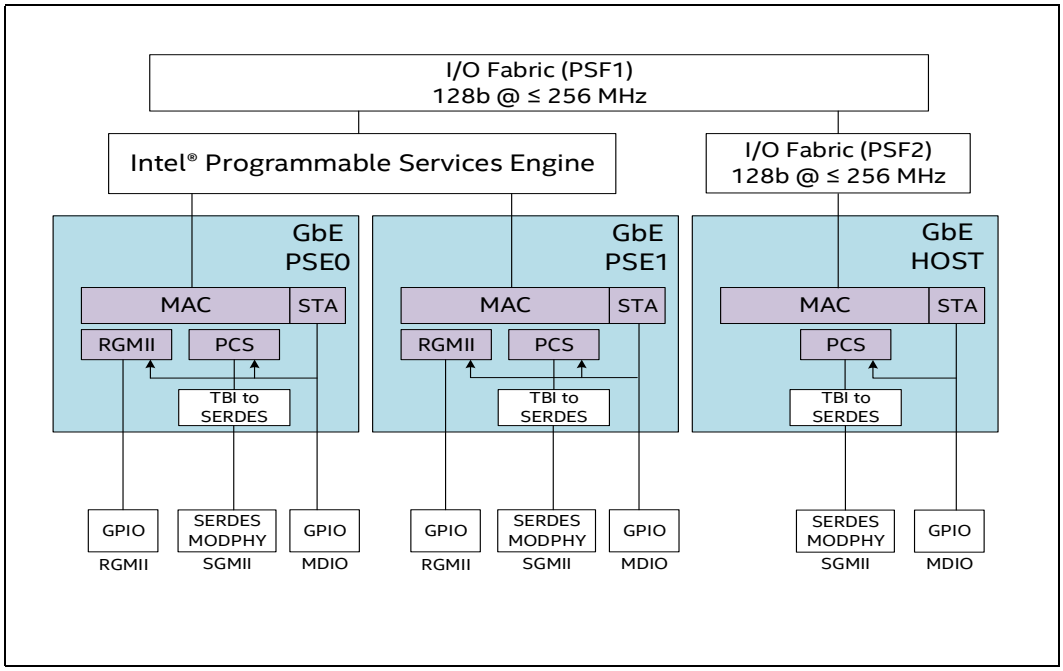
15 Gigabit Ethernet Controller and Time-Sensitive Networking

15.1 Overview

This chapter describes the behavior of the three Gigabit Ethernet (GbE) Controllers that reside in PCH including Intel® Programmable Services Engine (PSE). In Intel® PSE, there are two GbE Controllers. The GbE controller can operate at multiple speeds, 10/100/1000 Mbps (RGMII & SGMII) & 2500 Mbps (SGMII only) and in either full duplex or half duplex mode. Each integrated Time-Sensitive Networking (TSN) Ethernet Media Access Controller (MAC) has a unique 48-bit MAC Address. These MAC Addresses are located in a BIOS Sub-Region and are assigned by the customer using the Capsule Update Tool which runs in an OS.

The Figure 15-1 shows their locations in the PCH and Intel® PSE.

Figure 15-1. GbE-TSN MAC Placement



GbE PSE0 MAC and GbE PSE1 MAC are accessed by either the IA Processor cores through system software or the Intel® PSE's Arm* Cortex*-M7 Microcontroller through RTOS firmware. These two MAC devices are connected to the Intel® PSE rather than directly to the PCH IO Fabric (PSF). GbE PSE0 and GbE PSE1 support Serial Gigabit Media Independent Interface (SGMII) and Reduced Gigabit Media Independent Interface (RGMII). RGMII mode should be used when GbE operation is required in S0ix/

Sx mode, such as when ECMA-393 Network Proxy mode or Out Of Band Manageability support is required. RGMII clock in S0ix/Sx modes will come from PLL integrated inside Intel® PSE.

GbE HOST MAC is accessed by the IA Processor cores through system software via PCH IO Fabric (PSF2 and PSF1) and support SGMII interface only.

Each MAC has an IEEE Std 802.3 Station Management (STA) Entity that is accessible to software via the Memory Mapped IO (MMIO) registers to control the associated MDIO interface. See IEEE Std 802.3, Clause 22 and Clause 45 for MMIO registers. The Physical Coding Sublayer (PCS) module provides the sublayer circuitry between the GMII of the MAC and the Ten Bit Interface (TBI) of the SGMII SERrial-DESerial (SerDes) circuitry. See IEEE Std 802.3 Clause 35 for GMII and Clause 36 for TBI.

15.2 Features Description

The GbE features listed are same in all three GbE used in the PCH including Intel® PSE.

15.2.1 Ethernet Features Description

Each of the three GbE instances supports the following Ethernet features:

- 10Mbps/100Mbps/1Gbps RGMII mode through GPIO interface (only 1.8V support). RGMII mode is only applicable on GbE PSE0 and GbE PSE1.
- 10Mbps/100Mbps/1Gbps/2.5Gbps SGMII mode through SERDES interface
- MDIO (station management) interface
- GPIO - based input pin for interrupt signal from PHY status and wakes (magic packet from link partner)
- GPIO - based output signal pin to reset the SGMII PHY on the platform
- 8 TX queues and 8 RX queues with separate DMA channels and interrupts. Each TX/RX queue is 4KB for storing at least two normal packets with total of 64KB memory (TX+RX). Each queue size is programmable with TX queue size not to exceed 32KB and RX queue size not to exceed 32KB.
- Supports normal (1518/1522 bytes) and jumbo (9018 bytes) packets

Note: Jumbo packet support can be dependent on PHY compatibility and, if jumbo packet platform support is required, Intel recommends confirming support with the PHY vendor.

    - o On the receiving side, supports both cut-through and store and forward modes
    - o On the transmitting side, supports only store and forward mode

- Configurable DLL for clock centering on transmit in RGMII 1Gbps mode. The tap delays are adjustable in steps of 125ps and can sweep the entire eye. Implemented only on TX Direction. Refer to DLL Configuration (DLL_CFG) chapter in the Intel Atom® x6000E Series, and Intel® Pentium® and Celeron®

N and J Series Processors for Internet of Things (IoT) Applications, Datasheet, Volume 2 (Book 3 of 3), Intel Programmable Services Engine (Intel® PSE), to configure the DLL Delay Elements.

- Support for TCP/IP offloading:
  - Checksum Offload Engine (COE) that does Checksum insertion (on TX path) and detection (on RX path) for TCP/UDP/ICMP segments encapsulated in IPv4/ IPv6 datagrams.
  - TCP Segmentation Offload (TSO) Engine where large TCP packets are split into multiple small packets to save application bus cycles. Eight TX DMA Channels with separate 2KB memory (256 bytes per channel).
- Double VLAN support:
  - Insertion, replacement, or deletion of up to four VLAN tags on TX path
  - Packet filtering (layer-2) and stripping based on any one of four VLAN tags on RX path
- MAC Management Counters (MMC) for gathering statistics on the received and transmitted packets. Interrupts are generated for various events.
  - On TX: Jabber Timeout, No Carrier or Loss of Carrier, Late Collision, Packet Underflow, Excessive Deferral and Excessive Collision
  - On RX: CRC error, Runt packet (shorter than 64 bytes), Alignment error (in 10/100Mbps only), Length error (non-Type packet only), Out of Range (non-Type packet only, longer than 1518 bytes), GMII_RXER Input error
- Low power management
  - Magic packet wakes (from external PHY) through GPIO Interrupt when the entire Intel® PSE and SERDES are power gated. The SW driver will re-configure upon entry and the PMC does reset sequencing.
  - IEEE Std 802.3az-2010 Energy Efficient Ethernet (EEE) with automatic entry/exit when link is Idle – Low Power Idle mode. Both link TX/RX clocks can be clock gated.
- RX Filtering
  - 64 Address (SA/DA) based layer-2 perfect and Hash table filtering
  - Eight VLAN tag layer-2 filters
  - Two Layer-3 and Layer-4 frame filters
- Flexible RX Parser
  - All incoming RX packets are parsed as per the programmable instructions in the RX parser memory
  - 256B for non-express, 128B for express

  - 256 instructions of 32-bit data/mask
  - Match and Inverse Match
  - Drop or Accept packet
  - Packet Steering to a particular DMA Channel

Note: RX Filtering and Flexible RX Parser features are mutually exclusive. They shall not be enabled together.

- Reject ARP packets that don't belong to node (end station that is receiving ARP packets)
- Functional Level Reset (FLR)
- ECC protection on memories
  - Packet data
  - Control bits
  - Memory address
  - ECC is generated on (data + control bits + address) when is written into the memory
  - ECC is checked when data is read from the memory
  - 1-bit errors are detected and corrected
  - Two-bits error are only detected and reported via interrupt
- ECC Error Injection
  - 1-3 bits errors in the data and address
- Data Path Parity Protection
  - TX Data Path (from AXI primary to TX FIFO input)
  - RX Data Path (from RX FIFO output to AXI primary)
  - Parity Error Injection
- FSM protection
  - 1-3 bits errors in the data and address
- Timeout on certain interfaces
  - Control and Status Register (CSR) Interfaces, MDIO Interfaces, AXI Primary Interfaces, and AXI Secondary

Note: The Split Headers (SPH) function is not supported by the processor. Platform firmware & software shall not set any DMA_CH[0:7]_CONTROL.SPH register field to 0x1

15.2.2 TSN Features Description

TSN is a set of IEEE standards that are intended to ensure quality transmission of the time sensitive data over Ethernet networks. TSN standards are governed by an IEEE Std 802.1 task group driven by, in part, Avnu Alliance which is a consortium of organizations involved and invested in deterministic Ethernet Technology.

Table 15-1. TSN IEEE Standards

IEEE Standard	Description
IEEE Std 1588™-2008 v2	Precision Clock Synchronization Protocol for Networked Measurement and Control Systems
IEEE Std 802.1AS	A specific profile of IEEE Std 1588-2008. IEEE Std 802.1AS specifies the generalized Precision Time Protocol (gPTP). It provides a Layer 2 time synchronizing service
IEEE Std 802.1Qav 2009	Forwarding and Queueing Enhancements for Time-Sensitive Streams, which specifies the Credit Based Shaper
IEEE Std 802.1Qbu 2016	Frame Preemption. It allows a Bridge Port to suspend the transmission of non time critical frames while one or more time critical frames are transmitted
IEEE Std 802.3br 2015	Interspersed Express Traffic (IET)
IEEE Std 802.1Qbv 2015	Enhancements for Scheduled Traffic. It specifies time aware queue draining to schedule the transmission of frames relative to a known time scale

Note: Concurrent usage of the 802.1Qav and 802.1Qbv features is not supported.

In addition to IEEE Standards above, each of the GbE instances supports the following TSN features.

- Time Based Scheduling (TBS)
  - Time deterministic transmission of the packet according to per packet transmit time specified by users.
  - Separate Prefetch timer for each TX queue in TBS mode
- 16-deep Descriptor Prefetch buffer per each TX and RX queue to achieve line rate per queue.
- IEEE Std 1588™-2002/2008 timestamp support for PTP packets
  - 80-bit internal system timer that runs at 200MHz for high-precision one-step time stamping
  - 64-bit ART timer that runs at 19.2MHz with Time Synchronization support for local and system timer correlation
  - The ART timer and system timer values are captured with precision less than 5ns for timing correlation
- GPIO Auxiliary Timestamp Trigger input (IEEE Std 802.1AS)
- GPIO based Pulse Per Second output with programmable pulse width
- Each Control List of 1K entries for all Tx queues. 32x128(x8) memory for Control List to support IEEE Std 802.1Qbv

- Provision to route traffic on low latency on low fabric channel with traffic class based routing. Two virtual channels and traffic classes (TC) are supported. All express traffic is mapped to Virtual Channel-1(VC1) and best effort is mapped to Virtual Channel-0(VC0). Each queue is independently mapped to any of the supported VC/TC.

15.3 GbE Time-Stamping Logic

The Precision Time Protocol (PTP) over Ethernet is described in the IEEE Std 1588-2002 and 2008 versions. The subsystem provides the following features:

- IEEE Std 1588-2002 and 2008 formats
- Provides an option to take time snapshots of all packets or only PTP type packets
- Provides an option to take time snapshots of only event messages
- Provides an option to take the time snapshot based on the clock type: ordinary; boundary; end-to-end transparent; peer-to-peer transparent
- Provides an option to select the node to be a primary clock or secondary clock for ordinary and boundary clock
- Identifies the PTP message type, version, and PTP payload in the packets sent directly over Ethernet and sends the status
- Provides an option to measure sub-second time in digital or binary format
- Two time stamp sources, as follows:
  a) External timestamp
  b) Internal timestamp which is selected by software

15.4 GbE Cross-Timestamp Logic

Additional logic is included in the GbE design to provide time synchronization between the 64-bit timer Always Running Timer (ART) and the 80-bit system timer in the GbE Controller.

When system software sets the cross-timestamp enable bit in the GbE ART MDIO register, it captures simultaneous snapshots of the values of the GbE system timer and the ART. System software can then read the captured time values to establish a relationship between ART and GbE system timer; When the GbE system timer value is X, the ART value is Y.

15.5 External Interfaces

Two of the three MAC modules provide a choice of one of two types of interfaces to its external PHY. The PCH does not use any of the Precision Time Protocol (PTP) capture capabilities that may exist in the external PHY component. All the PTP events and timestamps are triggered in the MAC portion of the subsystem.

The selections for the GbE PSE0 and GbE PSE1 interfaces are Serial Gigabit Media-Independent Interface (SGMII) or Reduced Gigabit Media-Independent Interface (RGMII). For the GbE HOST the interface selection is SGMII.

15.5.1 Serial Gigabit Media-Independent Interface (SGMII)

Serial Gigabit Media-Independent Interface (SGMII) is a de-facto industry standards for achieving Ethernet LAN speeds of 10Mbps, 100Mbps, 1Gbps and 2.5Gbps. It consists of two sets of Current-Mode Logic (CML) differential signal using one of the multiplexed PCH ModPHY lanes (see Chapter 9 for further details). The design embeds the SGMII transmit clock in the transmit data and expects the SGMII receive clock to be embedded in the receive data. This eliminates four of the standard SGMII interface pins. For the controller's IEEE Std 802.3 Physical Sublayer configuration and management, it also provides two CMOS Management Data Input/Output (MDIO) interface signals. SGMII provides a reduced-pin implementation of GMII (IEEE Std 802.3, Clause 35) which would require 25 single-ended signals plus the two MDIO interface signals.

15.5.2 Reduced Gigabit Media-Independent Interface (RGMII)

Reduced Gigabit Media-Independent Interface (RGMII) is also a de-facto industry standard. It consists of 12 single-ended CMOS signals plus two MDIO signals, offering a reduced-pin implementation of IEEE-defined GMII.

15.5.3 Management Data Input/Output (MDIO)

The External PHY can be accessed and configured through Management Data Input/Output (MDIO) from the GbE controllers by SW/HW/BIOS. The IEEE Std 802.3 defines MDIO Management Interface which serves to access the Management registers of IEEE Std 802.3 complaint devices. This is a two-line interface including MDC (clock) and MDIO (bidirectional data).

Note: The interface supports MDIO operation as defined in IEEE Std 802.3 Clause 45 except at a signal voltage of 1.8V, not 1.2V.

15.6 Signal Description

15.6.1 SGMII Signals

Table 15-2. SGMII GbE LAN Signals

Group	Signal Name[1]	Type (Voltage Domain)	Direction	Description
GbE PSE0	PSE_GBE0_SGMII_TXP	CML differential signal (1.05V)	Output	Transmit P&N of the serial differential output
	PSE_GBE0_SGMII_TXN			
GbE PSE1	PSE_GBE1_SGMII_TXP			
	PSE_GBE1_SGMII_TXN			
GbE HOST	GBE_SGMII_TXP			
	GBE_SGMII_TXN			
GbE PSE0	PSE_GBE0_SGMII_RXP	CML differential signal (1.05V)	Input	Receive P&N of the serial differential input
	PSE_GBE0_SGMII_RXN			
GbE PSE1	PSE_GBE1_SGMII_RXP			
	PSE_GBE1_SGMII_RXN			
GbE HOST	GBE_SGMII_RXP			
	GBE_SGMII_RXN			

Note:
1. Refer to Chapter 9 for a description of how these signals are routed by the ModPHY lanes.

15.6.2 RGMII Signals

Table 15-3. RGMII Signals

Group	Signal Name	GPIO Type (Voltage Domain)[1]	Direction	Description
GbE PSE0	PSE_GBE0_RGMII_TXD[3:0]	CMOS (1.8V)	Output	Transmit Data
	PSE_GBE0_RGMII_TXCLK			Transmit Reference Clock
	PSE_GBE0_RGMII_TXCTL			Transmit Control
GbE PSE1	PSE_GBE1_RGMII_TXD[3:0]			Transmit Data
	PSE_GBE1_RGMII_TXCLK			Transmit Reference Clock
	PSE_GBE1_RGMII_TXCTL			Transmit Control
GbE PSE0	PSE_GBE0_RGMII_RXD[3:0]		Input	Receive Data
	PSE_GBE0_RGMII_RXCLK			Receive Reference Clock
	PSE_GBE0_RGMII_RXCTL			Receive Control
GbE PSE1	PSE_GBE1_RGMII_RXD[3:0]			Receive Data
	PSE_GBE1_RGMII_RXCLK			Receive Reference Clock
	PSE_GBE1_RGMII_RXCTL			Receive Control

Note:
1. The signals should be configured to 1.8V using the multiplexed GPIO's Individual Voltage Select soft strap.

15.6.3 MDIO Signals

Table 15-4. MDIO Signals

Group	Signal Name	GPIO Type (Voltage Domain)[1]	Direction	Description
GbE PSE0	PSE_GBE0_MDC	CMOS (1.8V)	Output	Management Data Clock This clock signal is driven by the LAN controllers to clock the serial MDIO data. The clock period is programmable.
GbE PSE1	PSE_GBE1_MDC			
GbE HOST	GBE_MDC			
GbE PSE0	PSE_GBE0_MDIO	Open Drain (1.8V)	Input/ Output	Management Data Input Output This signal is driven by either the LAN controller or the external PHY/Switch component during the MDIO transaction.
GbE PSE1	PSE_GBE1_MDIO			
GbE HOST	GBE_MDIO			
Note: 1. The signals should be configured to 1.8V using the multiplexed GPIO's Individual Voltage Select soft strap.				

15.6.4 Miscellaneous Signals

Table 15-5. Miscellaneous Signals (Sheet 1 of 2)

Group	Signal Name	GPIO Type (Voltage Domain)[1]	Direction	Description
GbE PSE0	PSE_GBE0_AUXTS	CMOS (1.8V)	Input	Auxiliary Time Stamp Trigger This edge-sensitive input signal triggers the storing of the time stamp into a 4x64 deep FIFO on its rising edge. If not used, this signal must be tied to GND through 20kOhm internal resistor. ***Note:*** This signal may also be called External Time Stamp Trigger (EXTTS) in other platform documentation
GbE PSE1	PSE_GBE1_AUXTS			
GbE HOST	GBE_AUXTS			
GbE PSE0	PSE_GBE0_INT[2]	CMOS (1.8V)	Input	Interrupt This input signal is driven by the external RGMII PHY or SGMII PHY device. It is also used for Wake On LAN (WOL). This WOL signal is also routed internally to the PCH Power Management Controller (PMC) when the in the S0ix power state. If not used under active-high mode (default), this signal must be tied to GND through 20kOhm internal resistor.
GbE PSE1	PSE_GBE1_INT[2]			
GbE HOST	GBE_INT[2]			
GbE PSE0	PSE_GBE0_RST_N	CMOS (1.8V)	Output	PHY Reset This output signal is used to reset the external RGMII or SGMII device. If not used, should be left as a No Connect.
GbE PSE1	PSE_GBE1_RST_N			
GbE HOST	GBE_RST_N			
GbE PSE0	PSE_GBE0_PPS	CMOS (1.8V)	Output	Pulse-Per-Second This output signal is generated as a pulse by the LAN controller each time its system timer indicates a new “seconds” value. If not used, should be left as a No Connect.
GbE PSE1	PSE_GBE1_PPS			
GbE HOST	GBE_PPS			

Table 15-5. Miscellaneous Signals (Sheet 2 of 2)

Group	Signal Name	GPIO Type (Voltage Domain)[1]	Direction	Description
Notes: 1. The signals should be configured to 1.8V using the multiplexed GPIO's Individual Voltage Select soft strap. 2. PCH input pin looks for an active-high Interrupt signal for the interrupt. If external PHY device is designed to produce an active-low Interrupt signal, the GbE controller has an inverter that must be enabled by programming the Global Configuration Register (GCR) MDIO register, PHY to MAC Interrupt Polarity (PHY2MAC_INTR_POL) bit. Refer to the Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors for Internet of Things (IoT) Applications, Datasheet, Volume 2 (Book 2 of 3), Mule Creek Canyon (Document Number: 636722), 10.3.2.1 Global Configuration Register – Address 00h for register details.				

15.7 GbE-TSN Interrupts and Message Signaled Interrupt

The GbE-TSN subsystem has the following 22 interrupts. 21 of these interrupts are routed through the Message Signaled Interrupt (MSI) and one comes from external PHY on the platform. The MSI has separate Vector Number. The GbE-TSN interrupts and MSI Vector Number are tabulated in Table 15-6.

Table 15-6. GbE-TSN interrupts and Message Signaled Interrupt (MSI) Vector Number

No	Interrupt Name	Direction	Description	MSI Vector No
1	Queue_TX0_IRQ	O	Per channel Transmit signal to host system	5'b00001
2	Queue_TX1_IRQ	O		5'b00011
3	Queue_TX2_IRQ	O		5'b00101
4	Queue_TX3_IRQ	O		5'b00111
5	Queue_TX4_IRQ	O		5'b01001
6	Queue_TX5_IRQ	O		5'b01011
7	Queue_TX6_IRQ	O		5'b01101
8	Queue_TX7_IRQ	O		5'b01111
9	Queue_RX0_IRQ	O	Per channel Receive signal to host system	5'b00000
10	Queue_RX1_IRQ	O		5'b00010
11	Queue_RX2_IRQ	O		5'b00100
12	Queue_RX3_IRQ	O		5'b00110
13	Queue_RX4_IRQ	O		5'b01000
14	Queue_RX5_IRQ	O		5'b01010
15	Queue_RX6_IRQ	O		5'b01100
16	Queue_RX7_IRQ	O		5'b01110
17	Correctable Error (CE) Interrupt	O	Safety interrupt for correctable errors	5'b11011
18	Uncorrectable Error (UE) Interrupt	O	Safety interrupt for uncorrectable errors	5'b11010
19	Low Power Idle (LPI) Interrupt	O	LPI RX exit interrupt output. This signal is high when the MAC receiver exits the LPI state. This is used for EEE and CSR clocks can be gated.	5'b11100

Table 15-6. GbE-TSN interrupts and Message Signaled Interrupt (MSI) Vector Number

No	Interrupt Name	Direction	Description	MSI Vector No
20	MAC_IRQ	O	Interrupt signal to the host system – generated by the MAC	5'b11101
21	PCS Interrupt	O	Interrupt signal to the host system – generated by the PCS	5'b11110
22	PHY MAC Interrupt	I	Interrupt from a PHY that is on the platform to MAC through a GPIO. The polarity of this signal is controlled by "phy2mac_intr_pol" bit in the GCR register. This interrupt also goes to PMC GPE register bit as Wake On LAN (WOL) interrupt. The PMC ignores any event on this signal during non-S0ix states.	Not Applicable

15.8 GbE TSN Register/Programming Differences Between GbE PSE MAC and GbE HOST MAC

This section describes the register/programming differences between GbE PSE MAC and GbE HOST MAC. Refer Table 15-7 for the details.

- BAR Number and Size
- DMA Channel to VC Mapping
- DMA Host Address Programming
- Power Management
- Snoop/No Snoop
- Device ID
- MMIO Register Space
- PHY Interface
- DLL Capability
- Proxy Mode
- Device Type
- Device Level Reset

Table 15-7. GbE TSN Register List Differences Between GbE PSE MAC and GbE HOST MAC

Item	GbE PSE (GbE PSE 0 & GbE PSE 1)	GbE Host	HW/SW Recommendation
BAR Number and Size	BAR0 – 256 KB: 64KB for GbE-TSN IP, 64KB Misc Logic, 128 KB for L2 SRAM exposure in proxy mode BAR1 – Used in ACPI mode, mapped to mainly CFG Spaced	BAR0 – 8K	Ignore ACPI Mode Program GbE PSE0 & GbE PSE1 accordingly for Proxy Mode (if used)

Table 15-7. GbE TSN Register List Differences Between GbE PSE MAC and GbE HOST MAC

Item	GbE PSE (GbE PSE 0 & GbE PSE 1)	GbE Host	HW/SW Recommendation
DMA Channel to VC Mapping	Address Space: MMIO Address Offset: Port0: 0x50210000, ... ,0x5021003C Port1: 0x50230000, ... ,0x5023003C	Address Space: MMIO MDIO Space. Address Offset: word address 0x1C, byte address 0x38 OCP Bridge was added, with Register Space for MMIO	Program per difference
DMA Host Address Programming[1]	Address Space: MMIO Address Offset: Port0: 0x50210100, ... ,0x5021023C Port1: 0x50230100, ... ,0x5023023C	Not Applicable	Make Host Address with in 4G space when owned by IA Processor cores
Power Management	D3: Address Space: PCI Config Address Offset: 0x84 D0i3: Address space: MMIO Address offset: 0x50210410	D3: Address Space: PCI Config Address Offset: 0x84 D0i3: Not Applicable	Enable Dx, Disable D0ix
Snoop/No Snoop	Supported	Not Supported	
Device ID[2]	GbE PSE 0[3]: 4BA0, 4BA1, 4BA2 GbE PSE 1[3]: 4BB0, 4BB1, 4BB2	GbE Host: 4B32	
MMIO Register Space[4]	DRAM ADDRESS_FILLIN (tied to 4GB) VC Mapping PCIe Snoop/NP Control D0i3 DLL Register: Offset 50230000 Proxy Mode Register	Register for: PMC ART Value, CFG Register for PHY PIPE Interface, General Purpose Global Control and Status	Refer to the Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors for Internet of Things (IoT) Applications, Datasheet, Volume 2 (Book 2 of 3) Mule Creek Canyon (Document Number: 636722), and Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors for Internet of Things (IoT) Applications, Datasheet, Volume 2 (Book 3 of 3), Intel® Programmable Services Engine (Intel® PSE) (Document Number: 636723), for the implications.
PHY Interface	SGMII and RGMII	SGMII only	
DLL Capability	DLL supported[5]	Not Applicable	
Proxy Mode	Proxy Mode	Not Available	
Device Type[6]	PCIe and ACPI Mode	PCIe Mode only	
Device Level Reset	MAC Soft Reset[7]	MAC Soft Reset[7] and FLR	

Table 15-7. GbE TSN Register List Differences Between GbE PSE MAC and GbE HOST MAC

Item	GbE PSE (GbE PSE 0 & GbE PSE 1)	GbE Host	HW/SW Recommendation
Notes: 1.Additional register needs to be programmed for upper Host address info. 2.Intel's Rule to assign unique ID for different instance of IP and provides placeholders for other values in case of future stepping and need to differentiate it the SW/FW. 3.The two GbE Controllers inside Intel® PSE are identical, different Device ID to identify their unique instantiation (4BA* & 4BB*). 4.Got differences in few Miscellaneous Registers. 5.Delay Logic Levels (DLL) are Programmable Delay Elements typically used for Serial Interfaces. Provides the flexibility to Adjust the Interfaces (Clock and Data) Delay to meet End to End Setup and Hold requirements. Only applicable on TSN with RGMII Interface. Relative Delays in steps of 125ps can be added between Clock and Data. Implemented only on TX Direction. Initialization and Programming of DLL Register done by BIOS. Refer to Chapter DLL Configuration (DLL_CFG) in the Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors for Internet of Things (IoT) Applications, Datasheet, Volume 2 (Book 2 of 3) Mule Creek Canyon (Document Number: 636722), to configure the DLL Delay Elements. 6.Intel recommends PCIE mode, no driver support for ACPI Mode. 7.MAC Soft Reset doesn't reset the CFG space. Customer can choose either option per instance, or just stay with MAC Soft Reset – which Intel Recommends.			

15.9 Supported System Configurations

Table 15-8 lists all the supported configurations, operating mode, and link partners for the LAN controllers. Selecting the desired PHY and system configuration is determined at power on through the use of soft strap and register configuration.

Table 15-8. Supported System Configurations

Connection	Speed	Electrical Interface	Third Party PHYs[1]
10BASE-T 100BASE-TX 1000BASE-T	10 Mb/s 100 Mb/s 1 Gb/s	RGMII/SGMII	Marvell* 88E1512
		SGMII	MaxLinear* GPY115
10BASE-T 100BASE-TX 1000BASE-T 2.5GBASE-T	10 Mb/s 100 Mb/s 1 Gb/s 2.5 Gb/s	SGMII	Marvell* 88E2110 MaxLinear* GPY211 MaxLinear* GPY215

Note:

1. Intel strongly recommends that only the PHYs listed be used in platform designs. Firmware and software incompatibilities may occur between other GbE PHYs and Intel's BKC due to per-vendor or per-model PHY-specific programming requirements.

15.10 Registers

Please refer to chapter 10 of the Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors for Internet of Things (IoT) Applications, Datasheet, Volume 2 (Book 2 of 3) Mule Creek Canyon (Document Number: 636722), and chapter 3 of the Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors for Internet of Things (IoT) Applications, Datasheet, Volume 2 (Book 3 of 3), Intel® Programmable Services Engine (Intel® PSE) (Document Number: 636723), for a description of the registers associated with subject of this chapter.

15.11 References

Specification	Location
IEEE Std 802.3-2015 Standard for Ethernet	https://standards.ieee.org/standard/802_3-2015.html
IEEE Std 802.1 AS-2011 Standard for Timing and Synchronization for Time-Sensitive Applications	https://standards.ieee.org/standard/802_1AS-2011.html
IEEE Std 1588 2008 Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems	https://standards.ieee.org/standard/1588-2008.html
IEEE Std 802.1Qav 2009 Standard for Forwarding and Queuing Enhancements for Time-Sensitive Streams	https://standards.ieee.org/standard/802_1Qav-2009.html
IEEE Std 802.1Qbu 2016 Standard for Frame Preemption	https://standards.ieee.org/standard/802_1Qbu-2016.html
IEEE Std 802.3br 2016 Standard for Specification and Management Parameters for Interspersing Express Traffic	https://standards.ieee.org/standard/802_3br-2016.html
IEEE Std 802.1Qbv 2015 Standard for Enhancements for Scheduled Traffic	https://standards.ieee.org/standard/802_1Qbv-2015.html
IEEE Std 802.3az 2010 Standard for Media Access Control Parameters, Physical Layers, and Management Parameters for Energy-Efficient Ethernet	https://standards.ieee.org/standard/802_3az-2010.html

§ §

16 Enhanced Serial Peripheral Interface (eSPI)

16.1 Functional Overview

The eSPI controller supports up to 4 devices. The Enhanced Serial Peripheral Interface (eSPI) is intended for connecting an EC to the platform.

eSPI operates at 1.8V only. This interface is not shared and distinct from the SPI interface used for flash device and TPM. The eSPI interface supports 14 MHz, 20 MHz, 33 MHz, and 50 MHz and up to Quad Mode with four chip selects.

The eSPI Target has an Alert Mode bit in its General Capabilities and Configuration register, which selects between the discrete and in-band Alert# indications. For a single Initiator – single Target configuration, the default value of this bit (in-band Alert#) works as-is. When two or more targets are present, this bit must be set to 1 by the eSPI Initiator to ensure that Alert# is signaled by discrete pins (one per target). Refer to Section 1.1.1 for more information on initiator and target.

16.1.1 Signal Description

Table 16-1. eSPI Signals

Signal Name	Group	Description
ESPI_IO[3:0]	Data	Bi-directional data signals used to transfer data between PCH and eSPI Target device
ESPI_CLK	Clock	eSPI Clock output from PCH
ESPI_CS[3:0]_N	Control	eSPI chip select
ESPI_RST0_N	Control	eSPI reset signal
ESPI_ALERT[3:0]_N	Control	eSPI alert signal

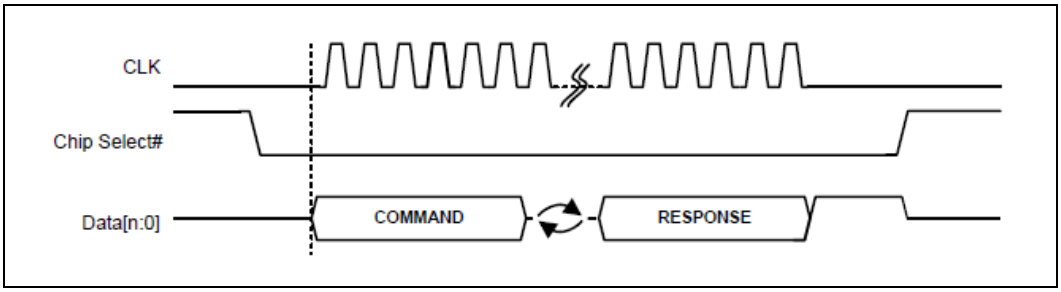
16.1.2 Operating Frequency

The eSPI controller supports 14 MHz, 20 MHz, 33 MHz, and 50 MHz. A target can support frequencies lower than the recommended maximum frequency (50 MHz). In addition, the target must support a minimum frequency of 20 MHz for default (reset) communication between the Initiator and Target.

16.1.3 Protocols

The following figure is an overview of the basic eSPI protocol.

Figure 16-1. Basic eSPI Protocol



An eSPI transaction consists of a Command phase driven by the Initiator, a turn-around phase (TAR), and a Response phase driven by the target.

A transaction is initiated by the PCH through the assertion of CS#, starting the clock and driving the command onto the data bus. The clock remains toggling until the complete response phase has been received from the target.

The serial clock must be low at the assertion edge of the CS# while ESPI_RESET# has been de-asserted. The first data is driven out from the PCH while the serial clock is still low and sampled on the rising edge of the clock by the target. Subsequent data is driven on the falling edge of the clock from the PCH and sampled on the rising edge of the clock by the target. Data from the target is driven out on the falling edge of the clock and is sampled on a falling edge of the clock by the PCH.

All transactions on eSPI are in multiple of 8 bits (one byte).

16.1.4 WAIT States from eSPI Target

There are situations when the target cannot predict the length of the command packet from the Initiator (PCH). For non-posted transactions, the target is allowed to respond with a limited number of WAIT states.

A WAIT state is a 1-byte response code. They must be the first set of response byte from the target after the TAR cycles.

16.1.5 In-Band Link Reset

In case the eSPI link may end up in an undefined state (for example when a CRC error is received from the target in a response to a Set_Configuration command), the PCH issues an In-Band Reset command that resets the eSPI link to the default configuration. This allows the controller to re-initialize the link and reconfigure the target.

16.1.6 Target Discovery

The controller does not perform discovery to confirm the presence of the target connection.

16.1.7 Multiple OOB Initiator

PCHs typically have multiple embedded processors such as the PMC and CSE. From an eSPI perspective, these are all classified as Out-of-Band (OOB) processors (as distinct from the Host processor). Since any such OOB processors may need to communicate with the eSPI device on the platform (e.g., EC. BMC), the eSPI controller implements dedicated OOB channel for each OOB processors including PMC and CSE to improve the interface performance and potentially enable new usage models.

16.1.8 Channels and Supported Transactions

An eSPI channel provides a means to allow multiple independent flows of traffic to share the same physical bus. Refer to the eSPI specification for more detail.

Each of the channels has its dedicated resources such as queue and flow control. There is no ordering requirement between traffic from different channels.

The number of types of channels supported by a particular eSPI Target is discovered through the GET_CONFIGURATION command issued by the PCH to the eSPI Target during initialization.

Table 16-2 summarizes the eSPI channels and supported transactions.

Table 16-2. eSPI Channels and Supported Transactions

CH #	Channel	Posted Cycles Supported	Non-Posted Cycles Supported
0	Peripheral	Memory Write, Completions	Memory Read, I/O Read/Write
1	Virtual Wire	Virtual Wire GET/PUT	N/A
2	Out-of-Band Message	SMBus Packet GET/PUT	N/A
3	Flash Access	N/A	Flash Read, Write, Erase
N/A	General	Register Accesses	N/A

16.1.8.1 Peripheral Channel (Channel 0) Overview

The Peripheral channel performs the following Functions:

- Target for PCI Device D31:F0: The eSPI controller duplicates the legacy LPC PCI Configuration space registers. These registers are mostly accessed via the BIOS, though some are accessed via the OS as well.
- Tunnel all Host to eSPI Target (EC/SIO) debug device accesses: these are the accesses that used to go over the LPC bus. These include various programmable and fixed I/O ranges as well as programmable Memory ranges. The programmable ranges and their enables reside in the PCI Configuration space.

Note: These accesses can only be routed to one CS_N signal at a time. The CS_N is selected using the eSPI CSx IO Routing Enables (ESPI_CSxIORE) registers.

Note: Only CS0_N has four Generic I/O Ranges assigned to it, which are configured using the ESPI_LGIR[3:0] registers. CS[1:3]_N have one Generic I/O Range assigned each, which are configured using the ESPI_CS1GIR1, ESPI_GIR1_EXT[0] & ESPI_GIR1_EXT[1]. An eSPI device requiring >1 Generic I/O range must be connected to CS0_N.

- Tunnel all accesses from the eSPI Target to the Host. These include Memory Reads and Writes.

16.1.8.2 Virtual Wire Channel (Channel 1) Overview

The Virtual Wire channel uses a standard message format to communicate several types of signals between the components on the platform.

- Sideband and GPIO Pins: System events and other dedicated signals between the PCH and eSPI Target. These signals are tunneled between the 2 components over eSPI.
- Serial IRQ Interrupts: Interrupts are tunneled from the eSPI Target to the PCH. Both edge and triggered interrupts are supported.

16.1.8.2.1 eSPI Virtual Wires (VW)

Table 16-3 summarizes the PCH virtual wires in eSPI mode.

Table 16-3. eSPI Virtual Wires (VW)

Virtual Wire	PCH Pin Direction	Reset Control	Pin Retained in PCH (For Use by Other Components)
SUS_STAT#	Output	ESPI_RST0_N	No
PLTRST#	Output	ESPI_RST0_N	Yes
PME# (eSPI Peripheral PME)	Input	ESPI_RST0_N	N/A
WAKE#	Input	ESPI_RST0_N	No
SMI#	Input	PMC_PLTRST_N	N/A
SCI#	Input	PMC_PLTRST_N	N/A
RCIN#	Input	PMC_PLTRST_N	No
Target_BOOT_LOAD_DONE	Input	ESPI_RST0_N	N/A
Target_BOOT_LOAD_STATUS	Input	ESPI_RST0_N	N/A
HOST_RST_WARN	Output	PMC_PLTRST_N	N/A
HOST_RST_ACK	Input	PMC_PLTRST_N	N/A
OOB_RST_WARN	Output	ESPI_RST0_N	N/A
OOB_RST_ACK	Input	ESPI_RST0_N	N/A
HOST_C10	Output	PMC_PLTRST_N	N/A
ERROR_NONFATAL	Input	ESPI_RST0_N	N/A
ERROR_FATAL	Input	ESPI_RST0_N	N/A

16.1.8.2.2 Interrupt Events

eSPI supports both level and edge-triggered interrupts. Refer to the eSPI Specification for details on the theory of operation for interrupts over eSPI.

The PCH eSPI controller will issue a message to the PCH interrupt controller when it receives an IRQ group in its VW packet, indicating a state change for that IRQ line number.

The eSPI Target can send multiple VW IRQ index groups in a single eSPI packet, up to the Operating Maximum VW Count programmed in its Virtual Wire Capabilities and Configuration Channel.

The eSPI controller acts only as a transport for all interrupt events generated from the Target. It does not maintain interrupt state, polarity or enable for any of the interrupt events.

16.1.8.3 Out-of-Band Channel (Channel 2) Overview

The Out-of-Band channel performs the following functions:

- Tunnel PCH Temperature Data to the eSPI Target: The eSPI controller stores the PCH temperature data internally and sends it to the target using a posted OOB message when a request is made to a specific destination address.
- Tunnel PCH RTC Time and Date Bytes to the eSPI Target: the eSPI controller captures this data internally at periodic intervals from the PCH RTC controller and sends it to the Target using a posted OOB message when a request is made to a specific destination address.

16.1.8.3.1 PCH Temperature Data Over eSPI OOB Channel

eSPI controller supports the transmitting of PCH thermal data to the eSPI Target. The thermal data consists of 1 byte of PCH temperature data that is transmitted periodically (~1 ms) from the thermal sensor unit.

The packet formats for the temperature request from the eSPI Target and the PCH response back are shown in Table 16-4 and Table 16-5.

Table 16-4. eSPI Target Request to PCH for PCH Temperature

Byte	7	6	5	4	3	2	1	0
0	eSPI Cycle Type: OOB Message = 21h							
1	Tag[3:0]				Length[11:8] = 0h			
2	Length[7:0] = 04h							
3	Destination Target Addr. =02h (PCH OOB HW Handler)							0
4	Command Code = 01h (Get_PCH_Temp)							
5	Byte Count =01h							
6	Source Target Address=0Fh(eSPI Target 0[EC])							1

Table 16-5. PCH Response to eSPI Target with PCH Temperature

Byte	7	6	5	4	3	2	1	0
0	eSPI Cycle Type: OOB Message = 21h							
1	Tag[3:0]				Length[11:8] = 0h			
2	Length[7:0] = 05h							
3	Destination Target Addr. =0Fh (eSPI Target 0[EC])							0
4	Command Code = 01h (Get_PCH_Temp)							
5	Byte Count =02h							
6	Source Target Address=02h(PCH OOB HW Handler)							1
7	PCH Temperature Data [7:0]							

16.1.8.3.2 PCH RTC Time/Date to EC Over eSPI OOB Channel

The PCH eSPI controller supports the transmitting of PCH RTC time/date to the eSPI Target. This allows the eSPI Target to synchronize with the PCH RTC system time. Moreover, using the OOB message channel allows reading of the internal time when the system is in Sx states.

The RTC time consists of 7 bytes: seconds, minutes, hours, day of week, day of month, month and year. The controller provides all the time/date bytes together in a single OOB message packet. This avoids the boundary condition of possible roll over on the RTC time bytes if each of the hours, minutes, and seconds bytes is read separately.

The packet formats for the RTC time/date request from the eSPI Target and the PCH response back to the device are shown in Table 16-4 and Table 16-5.

Table 16-6. eSPI Target Request to PCH for PCH RTC Time

Byte	7	6	5	4	3	2	1	0
0	eSPI Cycle Type: OOB Message = 21h							
1	Tag[3:0]				Length[11:8] = 0h			
2	Length[7:0] = 04h							
3	Destination Target Addr. =02h (PCH OOB HW Handler)							0
4	Command Code = 01h (Get_PCH_RTC_Time)							
5	Byte Count =01h							
6	Source Target Address=0Fh(eSPI Target 0[EC])							1

Table 16-7. PCH Response to eSPI Target with RTC Time

Byte	7	6	5	4	3	2	1	0
0	eSPI Cycle Type: OOB Message = 21h							
1	Tag[3:0]				Length[11:8] = 0h			
2	Length[7:0] = 0Ch							
3	Destination Target Addr. =0Fh (eSPI Target 0[EC])							0
4	Command Code = 02h (Get_PCH_RTC_Temp)							
5	Byte Count =02h							
6	Source Target Address=02h(PCH OOB HW Handler)							1
7	Reserved					DM	HF	DS
8	RTC Times: Seconds							
9	RTC Times: Minutes							
10	RTC Time: Hours							
11	RTC Time: Day of Week							
12	RTC Time: Day of Month							
13	RTC Time: Month							
14	RTC Time: Year							

Notes:
1. DS: Daylight Savings. A 1 indicates that Daylight Saving has been comprehended in the RTC time bytes. A 0 indicates that the RTC time bytes do not comprehend the Daylight Savings.
2. HF: Hour Format. A 1 indicates that the Hours byte is in the 24-hr format. A 0 indicates that the Hours byte is in the 12-hr format.
   In 12-hr format, the seventh bit represents AM when it is a 0 and PM when it is a 1.
3. DM: Data Mode. A 1 indicates that the time byte are specified in binary. A 0 indicates that the time bytes are in the Binary Coded Decimal (BCD) format.

16.1.9 Interface Configuration

The eSPI interface is enabled or disabled depending on the configuration of the 4-bit Boot Strap. Refer to Chapter 28 for further details.

Most configuration options for the eSPI interface, such as frequency & I/O mode, are available as Soft Straps.

Note: Neither Master Attached Flash Sharing (MAFS) nor Slave Attached Flash Sharing (SAFS) is supported.

16.2 Registers

Note: Please refer to chapter 2 of the Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors for Internet of Things (IoT) Applications, Datasheet, Volume 2 (Book 2 of 3), Mule Creek Canyon (Document Number: 636722), for a description of the registers associated with subject of this chapter.

§ §

17 Serial Peripheral Interface (SPI) Flash Memory and TPM Only

17.1 Acronyms

Acronyms	Description
CLK	Clock
CS	Chip Select
FCBA	Flash Component Base Address
FIBA	Flash Initialization Base Address
FLA	Flash Linear Address
FMBA	Flash Initiator Base Address
FPSBA	Flash PCH Strap Base Address
FRBA	Flash Region Base Address
MDTBA	MIP Descriptor Table Base Address
MISO	Initiator In Target Out
MOSI	Initiator Out Target In
TPM	Trusted Platform Module
CSE	Converge Security Engine

Note: Refer to Section 1.1.1 for more information on initiator and target.

17.2 Feature Overview

The FSPI interface consists of 3 Chip Select signals. It is allowing up to two flash memory devices (CS0_N and CS1_N) and one TPM device (CS2_N) to be connected to the PCH. The FSPI interface supports either 1.8V or 3.3V

A SPI flash device supporting SFDP (Serial Flash Discovery Parameter) is required for all PCH designs. A SPI flash device with a valid descriptor MUST be attached directly to the PCH. The POR capacity of the SPI Flash device is 64MB.

17.3 Signal Description

Name	Type	Description
FSPI_MOSI_IO0	Data	SPI serial output data from PCH to the SPI flash device. This Pin will also function as Input during Dual and Quad I/O operation
FSPI_MISO_IO1	Data	SPI serial input data from the SPI flash device to PCH. This Pin will also function as Output during Dual and Quad I/O operation
FSPI_IO2	Data	SPI serial Input/Output data to comprehend the support for the Quad I/O operation
FSPI_IO3	Data	SPI serial Input/Output data to comprehend the support for the Quad I/O operation
FSPI_CLK	Clock	SPI Clock output from PCH
FSPI_CS0_N	Chip Select	SPI chip select 0
FSPI_CS1_N	Chip Select	SPI chip select 1 signal is used as the second chip select, when 2 flash devices are used. Do not use when only one SPI flash is used.
FSPI_CS2_N	Chip Select	Chip Select 2 is dedicated to support TPM on SPI.

17.4 Integrated Pull-Ups and Pull-Downs

Signal	Resistor Type	Value	Notes
FSPI_MOSI_IO0	Pull-Up	20k ± 30%	
FSPI_MISO_IO1	Pull-Up	20k ± 30%	
FSPI_IO2	Pull-Up	20k ± 30%	
FSPI_IO3	Pull-Up	20k ± 30%	
FSPI_CLK	Pull-Up	20k ± 30%	
FSPI_CS0_N	Pull-Up	20k ± 30%	
FSPI_CS1_N	Pull-Up	20k ± 30%	
FSPI_CS2_N	Pull-Up	20k ± 30%	

Note: The internal pull-up is disabled when PMC_RSMRST_N is asserted (during reset) and only enabled after PMC_RSMRST_N de-assertion

17.5 I/O Signal Planes and States

Signal Name	Power Plane	During Reset (See Note 1)	Immediately after Reset	S3/S4/S5
FSPI_MOSI_IO0	Primary	Pulled Low	Driven Low	Driven Low
FSPI_MISO_IO1	Primary	Pulled High (See Note 2)	Pulled High	Pulled High
FSPI_IO2	Primary	Pulled High	Pulled High	Pulled High
FSPI_IO3	Primary	Pulled High	Pulled High	Pulled High
FSPI_CLK	Primary	Active	Active	Driven Low
FSPI_CS0_N	Primary	Driven High	Driven High	Driven High
FSPI_CS1_N	Primary	Pulled High	Driven High	Driven High
FSPI_CS2_N	Primary	Driven High	Driven High	Driven High

Notes:
1. During reset refers to when PMC_RSMRST_N is asserted.
2. FSPI_MOSI_IO0 also functions as a strap pin. The actual pin state during Reset is dependent on the platform Pull-up/Pull-down resistor.

17.6 Functional Description

17.6.1 FSPI for Flash

17.6.1.1 Overview

The Serial Peripheral Interface (FSPI) supports 2 SPI flash devices via 2 chip select signals (FSPI_CS0_N and FSPI_CS1_N).The maximum size of flash supported is determined by the SFDP-discovered addressing capability of each device. Each component can be up to 16 MB (32 MB total addressable) using 3-byte addressing. Each component can be up to 64 MB (128 MB total addressable) using 4-byte addressing.

PCH drives the interface clock at either 20 MHz, 33 MHz, or 50 MHz and will function with flash devices that support at least one of these frequencies.

A SPI flash device supporting SFDP (Serial Flash Discovery Parameter) is required for all PCH designs. A SPI flash device with a valid descriptor MUST be attached directly to the PCH.

The PCH supports fast read which consist of:

1. Dual Output Fast Read (Single Input Dual Output)
2. Dual I/O Fast Read (Dual Input Dual Output)
3. Quad Output Fast Read (Single Input Quad Output)
4. Quad I/O Fast Read (Quad Input Quad Output)

17.6.1.2 Operational Modes

The SPI Controller has two operational modes: Descriptor mode and Dnx Mode.

17.6.1.2.1 DnX Mode

- 20 MHz, single I/O, 03h read instruction, with option to enable higher throughput
- Read SFDP (Serial Flash Discoverable Parameters) from both devices, use SFDP to determine flash device sizes and number of components
- Up to two components are supported in DnX mode. They may be any size. Their size is discovered via SFDP.
- Only CSE is allowed to access flash
- All descriptor and register based protections are disabled when DnX mode is active
- DnX mode takes precedence over fdopss (flash descriptor security override), i.e. register security is turned off if both DnX and fdopss are asserted
- Only CSE h/w and s/w sequencing are allowed, not direct read

17.6.1.2.2 Descriptor Mode

Descriptor Mode is required to enable many features of the processor:

- Converged Security Engine
- Secure Boot
- PCI Express* root port configuration
- Supports for two SPI components using two separate chip select pins
- Hardware enforced security restricting Initiator accesses to different regions
- Soft Strap region providing the ability to use Flash NVM to remove the need for pull-up/pull-down resistors for strapping processor features
- Support for the SPI Fast Read instruction and frequencies greater than 20 MHz
- Support for Single Input, Dual Output Fast reads
- Use of standardized Flash instruction set

17.6.1.2.3 SPI Flash Regions

In Descriptor Mode the Flash is divided into separate regions.

Table 17-1. SPI Flash Regions

Region	Content
0	Flash Descriptor
1	BIOS
2	Converged Security Engine
3	RSVD
4	Platform Data
5	RSVD

Only two Initiators can access the regions: Host processor running BIOS code and the Intel CSE (Converged Security Engine).

The Flash Descriptor and CSE region are the only required regions. The Flash Descriptor has to be in region 0 and region 0 must be located in the first sector of Device 0 (Offset 0). All other regions can be organized in any order.

Regions can extend across multiple components, but must be contiguous.

17.6.1.2.4 Flash Region Sizes

SPI flash space requirements differ by platform and configuration. The Flash Descriptor requires one 4-KB or larger block. The amount of flash space consumed is dependent on the erase granularity of the flash part and the platform requirements for the CSE and BIOS regions. The CSE region contains firmware to support CSE capabilities.

Table 17-2. Region Size Versus Erase Granularity of Flash Components

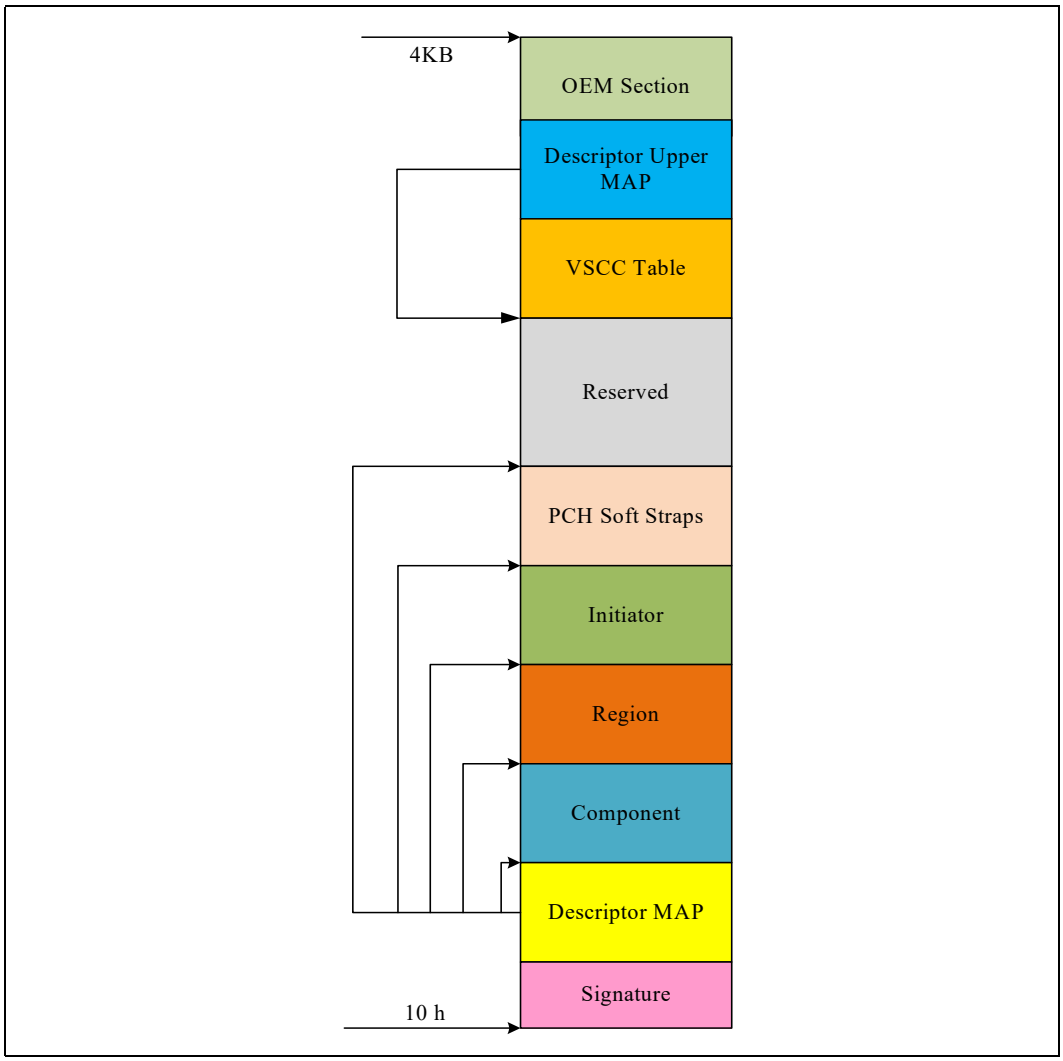
Region	Size with 4-KB Blocks	Size with 8-KB Blocks	Size with 64-KB Blocks
Descriptor	4 KB	8 KB	64 KB
BIOS	Varies by Platform	Varies by Platform	Varies by Platform
Intel® CSE	Varies by Platform	Varies by Platform	Varies by Platform

17.6.1.3 Descriptor

The bottom sector of the flash component 0 contains the Flash Descriptor. The maximum size of the Flash Descriptor is 4 KB. If the block/sector size of the SPI flash device is greater than 4 KB, the flash descriptor will only use the first 4 KB of the first block. The flash descriptor requires its own block at the bottom of memory (00h). The information stored in the Flash Descriptor can only be written during the manufacturing process as its read/write permissions must be set to read only when the computer leaves the manufacturing floor.

The Flash Descriptor is made up of eleven sections as shown in Figure 17-1.

Figure 17-1. Flash Descriptor Regions



- The **OEM** Section is 256 bytes reserved at the top of the Flash Descriptor for use by OEM.
- The **Descriptor Upper MAP** determines the length and base address of the Management Engine VSCC Table.
- The **VSCC Table** holds the JEDEC ID and the VSCC information of the entire SPI Flash supported by the NVM image.
- The **Reserved** region between the top of the processor strap section and the bottom of the OEM Section is reserved for future chipset usages.
- The **PCH Soft Straps** section contains processor and PCH configurable parameters.

- The **Initiator** region contains the security settings for the flash, granting read/write permissions for each region and identifying each initiator by a requestor ID.
- The **Region** section points to the three other regions as well as the size of each region.
- The **Component** section has information about the SPI flash in the system including: the number of components, density of each, invalid instructions (such as chip erase), and frequencies for read, fast read and write/erase instructions.
- The **Descriptor Map** has pointers to the other five descriptor sections as well as the size of each.
- The **Signature** selects Descriptor Mode as well as verifies if the flash is programmed and functioning. The data at the bottom of the flash (offset 10h) must be 0FF0A55Ah in order to be in Descriptor mode.

17.6.1.3.1 DnX Support

The expectation is that when the platform fails to boot the user will force a re-boot into DnX mode. If the descriptor is invalid but the DnX mode indication is false, then the desired behavior is for the flash controller to allow the CSE to come up and run using the old non-descriptor mode restrictions, however no flash controller behavior is guaranteed.

17.6.1.3.2 Descriptor Initiator Region

The initiator region defines read and write access setting for each region of the SPI device. The initiator region recognizes two initiators: BIOS and CSE. Each initiator is only allowed to do direct reads of its primary regions.

Table 17-3. Region Access Control Table

Initiator Read/Write Access		
Region Name	**Processor and BIOS**	**Intel® CSE**
BIOS	Read/Write	N/A
CSE	N/A	Read/Write

17.6.1.3.3 Flash Descriptor CPU Complex Soft Strap Section

Region Name	Starting Address
Signature	10h
Component FCBA	30h
Regions FRBA	40h
Initiators FMBA	80h

Region Name	Starting Address
PCH Straps FPSBA	100h
MDTBA	C00h
PMC Straps	C14h
CPU Straps	C2Ch
Intel® CSE Straps	C3Ch
Register Init FIBA	340h

17.6.1.4 Flash Access

There are two types of accesses: Direct Access and Program Register Accesses.

17.6.1.4.1 Direct Access

- Initiators are allowed to do direct read only of their primary region
- The BIOS or CSE virtual read address is converted into the SPI Flash Linear Address (FLA) using the Flash Descriptor Region Base/Limit registers

Direct Access Security

- Requester ID of the device must match that of the primary Requester ID in the Initiator Section
- Calculated Flash Linear Address must fall between primary region base/limit. If it does not, the cycle will not be run on the SPI bus, a completion with not data will be synthesized and returned with an Unsupported Request completion status and the AEL (Access Error Log) register error bit will be set
- Direct Write is not allowed with the exception of SPI TPM accesses
- Direct Read Cache contents are reset to 0's on a read from a different initiator

17.6.1.4.2 Program Register Access

- Program Register Accesses are not allowed to cross a 4-KB boundary and can not issue a command that might extend across two components
- Software programs the FLA corresponding to the region desired
  - Software must read the devices Primary Region Base/Limit address to create a FLA.

Register Access Security

- Only primary region initiators can access the registers. If the initiator ID is not valid, the cycle will not be run on the SPI bus, a a completion with no data will be synthesized and returned with an Unsupported Request completion status and the AEL (Access Error Log) register error bit will be set

17.6.2 FSPI Support for TPM

The PCH's FSPI flash controller supports a discrete TPM on the platform via its dedicated FSPI_CS2_N signal. The platform must have no more than 1 TPM.

SPI controller supports accesses to SPI TPM at 20 MHz, 33 MHz and 50 MHz depending on the PCH soft strap. 20 MHz is the reset default, a valid PCH soft strap setting overrides the requirement for the 20 MHz. SPI TPM device must support a clock of 20 MHz. It may, but is not required to support a frequency greater than 20 MHz.

The SPI controller does have an integrated interrupt signal for the TPM.

17.6.2.1 TPM Address and Cycle Decode

The TPM address range is FED4_0000h through FED4_7FFFh in LT memory space. Note that this is not the same as the regular memory space. LT memory space is accessed using the LTR/LTW transactions only. Host-space memory transactions to the above address are to the regular memory space; the flash controller relies on OPI source decode to only forward valid LTW/LTR cycles to TPM on SPI.

When enabled for TPM and the LTR/LTW cycle type required input is true, the SPI controller will decode as follows:

SPI will decode LTW transactions on the IOSF primary host root space to the address range FED4_0000h through FED4_7FFFh and send the cycle down SPI as a TPM write, using the TPM-SPI protocol.

SPI will decode LTR transactions on the IOSF primary host root space to the address range FED4_0000h through FED4_7FFFh and send the cycle down SPI as a TPM read, using the TPM-SPI protocol.

17.6.2.2 TPM Protocol on SPI

Though the SPI controller supports 26-bit addressing on SPI, only 24-address bits are sent to the TPM. The FEh byte is dropped. For example, an incoming address of FED4_4024h would be sent on the SPI as D4_4024h.

Access to the SPI TPM is always with single-address input and single output at the single data rate. Dual-output, dual-I/O, quad-output and quad-I/O operations are not supported on CS2_N with the TPM.

The following rules apply for the data transfer:

- Data is shifted Most Significant (MS) Bit first and Least Significant (LS) Byte first.
- The address and command are shifted MS Bit first for the entire field.
- The 24-bit address shifts A23 first, then A22, A21..., A0.
- Initiator and Target both drive data on the falling edge of the clock.
- The TPM accesses always get a 24-bit address that is the offset from FE00_0000h.
- Asserting the SPI_TPM_CS_N is an indication that the processor did a full decode and the cycle is in the FED4_xxxxh range.
- Only SPI mode 0 is supported (CPHA = 0, CPOL = 0).

It is legal to transmit any number of bytes from 1 to 64. Zero length reads or writes are not allowed. If the transfer is less than 4B, then the corresponding bits are left out. For example, on a 2B write, the last transfer on MOSI is Data[8] (LS Bit of the MS Byte).

There is no status byte for the transfer. If the write to the TPM failed, the TPM would not honor what was received, and software/driver will understand the software command did not succeed and perform the appropriate recovery mechanisms. If the read fails, then the processor would return all FFs for the data, which would signal a failure to the driver.

17.6.2.3 TPM Flow Control on SPI

Note: This description uses idealized, zero-delay timing for illustrative purposes.

For the TPM operation, it involves hardware generating transactions directly to the TPM with minimum or no software involvement. As such, there needs to be a simple flow control mechanism on SPI because hardware cannot poll busy bits or use other software mechanisms. Therefore the following flow control is allowed by the TPM.

SPI protocol doesn't have a defined flow control mechanism. Thus a new flow control mechanism is being created for the TPM on SPI.

The flow control is on a transaction basis and not on a byte basis. For example, a read or write to the data register can be at most 4B in length today, moving to 8B or 64B in the future. The TPM will accept the write data when it has the full size of buffer available to be written (1-64B), or provide the read data when it has the full amount of data (1-64B) ready to deliver, again based on the size of the transaction. The overhead of allowing flow control between each byte is too high with almost no benefit.

Since the specification allows for larger sizes of transactions in the future, the processor will have limited, if any, hardware checking on accesses to the TPM address space. If the processor receives transaction for any size from 1B to 64B that doesn't cross a 64B boundary, it must issue that transaction on SPI as received. The processor must accept all transactions of any length on any address boundary to FED4_0000h to FED4_4FFFh, as long as they don't cross a 64B boundary.

The TPM transaction on SPI consists of 1B of command, 3B of address, followed by write data from the processor or read data from the TPM. The TPM may insert wait states after the 4B of command and address have been received.

The mechanism to insert wait states is as follows. For a read to the TPM, the command and address are driven on MOSI and the TPM responds with data on MISO. With no wait states, the TPM would drive data on the next falling clock edge after the falling clock edge that the processor drove the last address bit. The flow control mechanism added for the TPM is that the processor will monitor the MISO pin in the same clock window that A[0] (the last address bit) is valid.

The TPM receives the address, where address bit[2] is captured a clock and a half before it has to drive the flow control bit. For reads,

17.7 VCCSPI Voltage (3.3V or 1.8V) Selection

The VCCSPI voltage (3.3V or 1.8V) is selected via a strap on GP_DSW11:

0 = SPI voltage is 3.3V

1 = SPI voltage is 1.8V

17.8 Registers

Note: Please refer to chapter 7 of the Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors for Internet of Things (IoT) Applications, Datasheet, Volume 2 (Book 2 of 3), Mule Creek Canyon (Document Number: 636722), for a description of the registers associated with subject of this chapter.

§ §

18 SIO (LPSS)

18.1 Intel® Serial I/O Universal Asynchronous Receiver/Transmitter (UART) Controllers

18.1.1 Overview

The PCH implements three independent SIO UART interfaces, UART0, UART1 and UART2. Each UART interface is a 4-wire interface supporting up to 3.8Mbit/s.The interfaces can be used in the low-speed, full-speed, and high-speed modes. The UART communicates with serial data ports that conform to the RS-232 interface protocol.

18.1.2 UART Signal Descriptions

Table 18-1. UART Signals

Group	Signal Name	Description
Data	SIO_UART[2:0]_RXD	Receive Data Input signals
	SIO_UART[2:0]_TXD	Transmit Data Output signals
Control	SIO_UART[2:0]_RTS_N	Request to Send signals
	SIO_UART[2:0]_CTS_N	Clear to Send signals

18.1.3 Feature Overview

The UART interfaces support the following features:

- Up to 3.8 Mbits/s Auto Flow Control mode as specified in the 16750 standard
- Transmitter Holding Register Empty (THRE) interrupt mode
- 64-byte TX and 64-byte RX host controller FIFOs
- DMA support with 64-byte DMA FIFO per channel (up to 32-byte burst)
- Functionality based on the 16550 industry standards
- Programmable character properties, such as number of data bits per character (5-8), optional parity bit (with odd or even select) and number of stop bits (1, 1.5, or 2)
- Line break generation and detection
- DMA signaling with two programmable modes
- Prioritized interrupt identification
- Programmable FIFO enable/disable
- Programmable serial data baud rate
- Modem and status lines are independently controlled

Notes: SIR mode is not supported.

18.1.4 UART Baud Rate Generation

The generated Baud rate depends on the input serial clock frequency and the applied clock division. The UART controller provides a rudimentary clock divider which lacks the precision to generate any Baud rate with a low error rate for a given clock frequency. The SIO block provides an optional clock divider circuit which applies an M:N ratio to the input clock (100MHz) before the controller. This 15-bit division circuit provides < 0.01% error margin for all clock frequencies.

The formula to calculate the generated Baud rate with the M:N ratio is:

Baud Rate = (f_input*M/N)/(16*Divisor)

For a f_input value of 100MHz, a variable M/N ratio and a fixed Divisor (DLH = 0, DLL = 1) is recommended. The variable M can be calculated as follows:

M = (Baud Rate*16*N)/100MHz

(M should be rounded to be closest integer, if necessary)

For example, a target Baud rate of 3 Mbps @ 100 MHz requires a divider ratio of 0.48, which implies that M should be set to 0.48 of N. To achieve the lowest error margin, M and N should be assigned full 15-bit values, 15720 and 32750 respectively, in this example.

The M and N divider values are programmable. The register programming sequence to use the M:N clock divider is defined in Table 18-2.

Table 18-2. Baud Rate Programming

Register.Field	Offset	Bits	Value	Access
DLL.DLL	0x000	7:0	1	MMIO
REG DLH.DLH	0x004	7:0	0	MMIO
REG CLOCKS.N_VAL	0x200	30:16	0:M	MMIO
REG CLOCKS.M_VAL	0x200	15:1	0:N	MMIO

Note: The M/N ratio cannot be changed when clock gating is enabled for the controller.

18.1.5 Functional Description

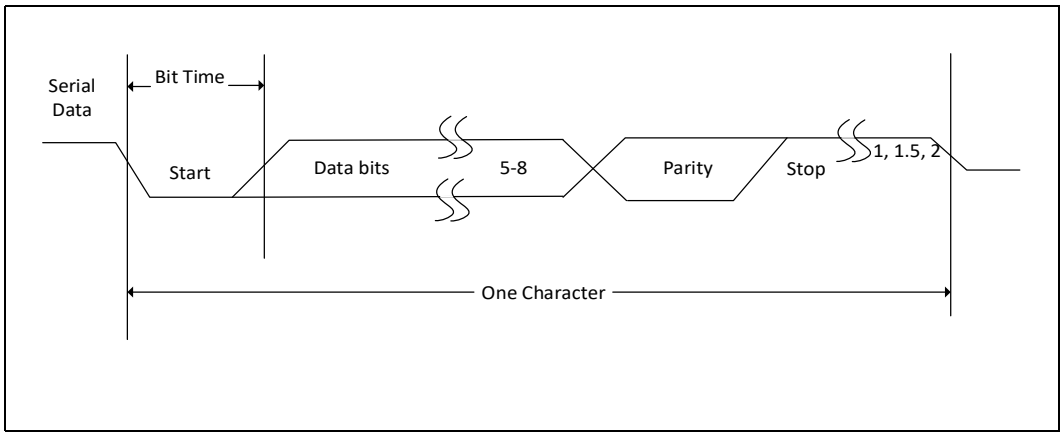
The UART transmits and receives data in bit frames as shown in Figure 18-1. Each data frame is between 7 and 12 bits long, depending on the size of data programmed and if parity and stop bits are enabled.The frame begins with a start bit that is represented by a high-to-low transition.Next, 5 to 8 bits of data are transmitted, beginning with the least significant bit. An optional parity bit follows, which is set if even parity is enabled and an odd number of ones exist within the data byte; or, if odd parity is enabled and

the data byte contains an even number of ones. The data frame ends with one, one-and-one-half, or two stop bits (as programmed by users), which is represented by one or two successive bit periods of a logic one.

18.1.6 UART Serial (RS-232) Protocols Overview

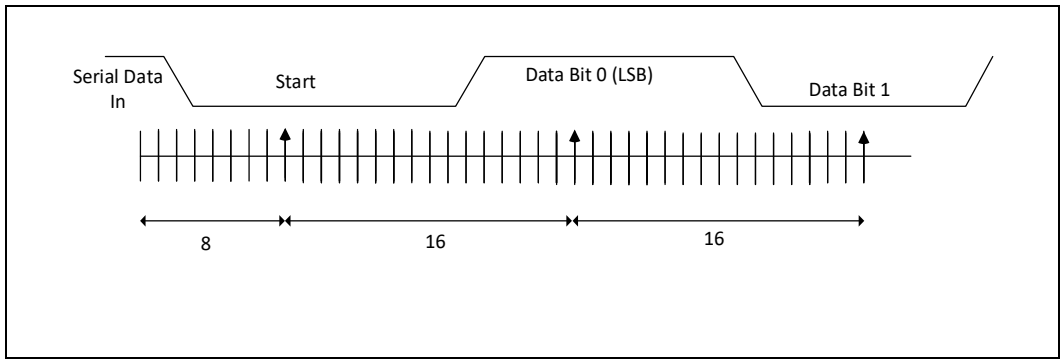
The serial communication between the UART host controller and the selected device is asynchronous, Start and Stop bits are used on the serial data to synchronize the two devices. The structure of serial data accompanied by Start and Stop bits is referred to as a character. An additional parity bit may be added to the serial character. This bit appears after the last data bit and before the stop bit(s) in the character structure to provide the UART Host Controller with the ability to perform simple error checking on the received data.

Figure 18-1. UART Serial Protocol



The UART Host Controller Line Control Register (LCR) is used to control the serial character characteristics. The individual bits of the data word are sent after the Start bit, starting with the least significant bit (LSB). These are followed by the optional parity bit, followed by the Stop bit(s), which can be 1, 1.5, or 2. The Stop bit duration implemented by UART host controller may appear longer due to idle time inserted between characters for some configurations and baud clock divisor values in the transmit direction. All bit in the transmission (with exception to the half stop bit when 1.5 stop bits are used) are transmitted for exactly the same time duration (which is referred to as Bit Period or Bit Time). One Bit Time equals to 16 baud clocks. To ensure stability on the line, the receiver samples the serial input data at approximately the midpoint of the Bit Time once the start bit has been detected.

Figure 18-2. UART Receiver Serial Data Sample Points



18.1.7 16550 8-bit Addressing - Debug Driver Compatibility

The UART controller is not compatible with legacy UART 16550 debug-port drivers. The controller operates in 32-bit addressing mode only and UART 16550 legacy drivers only operate with 8-bit (byte) addressing. In order to provide compatibility with standard in-box legacy UART drivers a 16550 Legacy Driver mode has been implemented in the controller that will convert 8-bit addressed accesses from the 16550 legacy driver to the 32-bit addressing that the controller supports. The control of this mode is over IOSF SB through the GPPRVRW7 register (offset 0x618).

Notes:

1) The UART 16550 8-bit Legacy mode only operates with PIO transactions. DMA transactions are not supported in this mode.

2) When operating in the UART 16550 8-bit Legacy mode only the UART controller registers are accessible. Access to other address regions of the SIO block related to UART should be disabled by the BIOS programming PCICFGCTRL:PCI_CFG_DIS (bit 0) to 1. UART0 (Device 30:Function 0) must not be used for legacy mode since PCI_CFG_DIS=1 of Function 0 may block initialization of the other functions. To access address regions outside of the UART Host control the UART 16550 8-bit Legacy mode must be disabled first.

3) After changing any of the GPPRVRW7.UART[2:0]_BYTE_ADDR_EN register bits, the firmware or software must immediately issue an MMIO Read transaction to a UARTnBAR0 + Offset Register (For example: 0x0F8, the read data can be discarded). This MUST BE done in order for the UART 16550 8-bit Legacy Mode to become active or inactive.

4) Power managing the device is not expected to be a function of the legacy driver and the debug UART must be configured to be functional before OS handoff. This means the controller will remain in D0 when configured for debug. It is invalid to program the controller to D3 when the 8-bit aligned mode is configured. This means that S0ix entry, if conditioned on power gating of SIO, would not occur.

18.1.8 DMA Controller

The UART controllers have an integrated DMA controller. Each channel contains a 64-byte FIFO. Max. burst size supported is 32 bytes.

18.1.8.1 DMA Transfer and Setup Modes

The DMA can operate in the following modes:

1. Memory to peripheral transfers. This mode requires that the peripheral control the flow of the data to itself.

2. Peripheral to memory transfer. This mode requires that the peripheral control the flow of the data from itself.

The DMA supports the following modes for programming:

1. Direct programming. Direct register writes to DMA registers to configure and initiate the transfer.

2. Descriptor based linked list. The descriptors will be stored in memory (such as DDR or SRAM). The DMA will be informed with the location information of the descriptor. DMA initiates reads and programs its own register. The descriptors can form alinked list for multiple blocks to be programmed.

3. Scatter Gather mode.

18.1.8.2 Channel Control

- The source transfer width and destination transfer width are programmable. It can vary to 1 byte, 2 bytes, and 4 bytes.
- Burst size is configurable per channel for source and destination. The number is a power of 2 and can vary between 1,2,4,...,128. this number times the transaction width gives the number of bytes that will be transferred per burst.
- Individual Channel enables. If the channel is not being used, then it should be clock gated.
- Programmable Block size and Packing/Unpacking. Block size of the transfer is programmable in bytes. the block size is not be limited by the source or destination transfer widths.
- Address incrementing modes: The DMA has a configurable mechanism for computing the source and destination addresses for the next transfer within the current block. The DMA supports incrementing addresses and constant addresses.
- Flexibility to configure any hardware handshake sideband interface to any of the DMA channels.
- Early termination of a transfer on a particular channel.

18.1.9 Reset

Each host controller has an independent rest associated with it. Control of these resets is accessed through the Reset Register. Each host controller and DMA will be in reset state once powered off and require SW (BIOS or driver) to write into specific reset register to bring the controller from reset state into operational mode.

18.1.10 Power Management

18.1.10.1 Device Power Down Support

In order to power down peripherals connected to PCH UART bus, the idle, configured state of the I/O signals must be retained to avoid transitions on the bus that can affect the connected powered peripheral. Connected devices are allowed to remain in the D0 active or D2 low power states when the bus is powered off (power gated). The PCH HW will prevent any transitions on the serial bus signals during a power gate event.

18.1.10.2 Latency Tolerance Reporting (LTR)

Latency Tolerance Reporting is used to allow the system to optimize internal power states based on dynamic data, comprehending the current platform activity and service latency requirements. The UART bus architecture, however, does not provide the architectural means to define dynamic latency tolerance messaging. Therefore, the interface supports this by reporting its service latency requirements to the platform power management controller via LTR registers. The controller's latency tolerance reporting can be managed by one of the two following schemes. The platform integrator must choose the correct scheme for managing latency tolerance reporting based on the platform, OS and usage.

1. Platform/HW Default Control. This scheme is used for usage models in which the controller's state correctly informs the platform of the current latency requirements. In this scheme, the latency requirement is a function of the controller state. The latency for transmitting data to/from its connected device at a given rate while the controller is active is representative of the active latency requirements. On the other hand if the device is not transmitting or receiving data and idle, there is no expectation for end to end latency.

2. Driver Control. This scheme is used for usage models in which the controller state does not inform the platform correctly of the current latency requirements. If the FIFOs of the connected device are much smaller than the controller FIFOs, or the connected device's end to end traffic assumptions are much smaller than the latency to restore the platform from low power state, driver control should be used.

18.1.11 Interrupts

UART interface has an interrupt line which is used to notify the driver that service is required. When an interrupt occurs, the device driver needs to read both the host controller and DMA status and TX completion interrupt registers to identify the interrupt source. Clearing the interrupt is done with the corresponding interrupt register in the host controller or DMA.

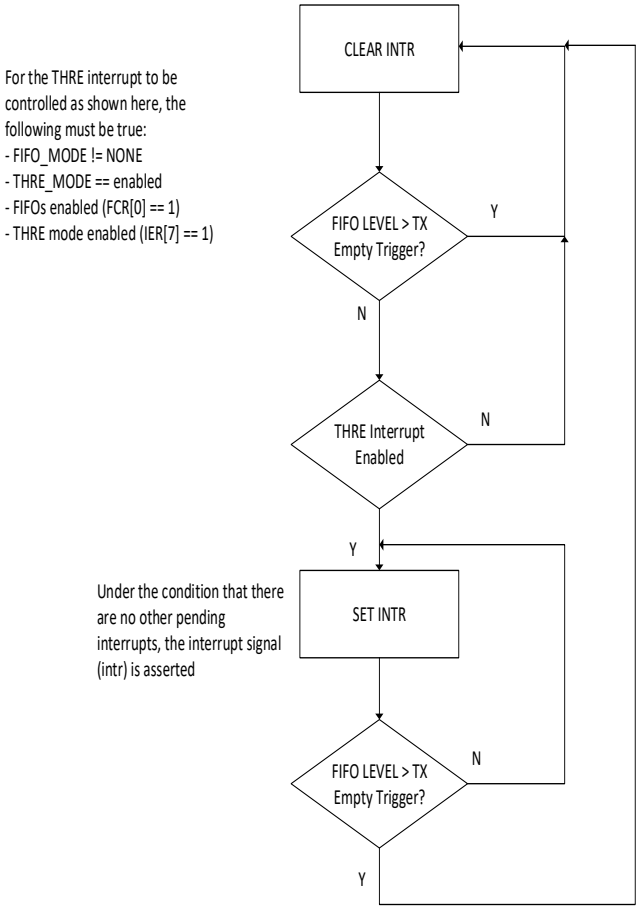
All interrupts are active high and their behavior is level interrupt. Controller interrupts are enabled using the IER (Interrupt Enable Register) and read using the IIR (Interrupt Identification Register)

18.1.12 Error Handling

Errors that might occur on the external UART signals are comprehended by the host controller and reported to the interface host controller driver through the MMIO registers.

18.1.13 Programmable THRE Interrupt

Programmable THRE Interrupt mode can be enabled using the Interrupt Enable Register (IER[7]).When FIFOs and THRE mode are implemented and enabled, the THRE Interrupts and DMA controllers are active at, and below, a programmed transmitter FIFO empty threshold level, as opposed to empty, as shown in the pseudo-code below. Clear THRE interrupt (By IIR register read or write to TX FIFO above the threshold)



The threshold level is programmed into FCR[5:4]. Available empty thresholds are: Empty, 2, ¼ & ½. Selection of the best threshold value depends on the system's ability to begin a new transmission sequence in a timely manner. However, one of these thresholds should be optimal for increasing system performance by preventing the transmitter FIFO from running empty. In addition to the interrupt change, the Line Status Register (LSR[5]) also switches from indicating that the transmitter FIFO is empty to the FIFO being full. This allows software to fill the FIFO for each transmit sequence by polling LSR[5] before writing another character. The flow then allows the transmitter FIFO to be filled whenever an interrupt occurs and there is data to transmit,

rather than waiting until the FIFO is completely empty. Waiting until the FIFO is empty causes a reduction in performance whenever the system is too busy to respond immediately. Further system efficiency is achieved when this mode is enabled in combination with Auto Flow Control.

18.1.14 Auto Flow Control

The controller can be configured to have a 16750-compatible Auto RTS and Auto CTS serial data flow control mode available. This mode can be enabled by setting the MCR.AFCE register bit to 1, as long as the controller FIFOs are enabled.

18.1.14.1 Auto RTS

Auto RTS becomes active when the following occurs:

•Auto Flow Control is selected during configuration

•RTS (MCR[1] bit and MCR[5]bit are both set)

•FIFOs are enabled (FCR[0]) bit is set)

When Auto RTS is enabled, the RTS_N output is forced inactive (high) when the receiver FIFO level reaches the threshold set by FCR[7:6], but only if the RTC flow-control trigger is disabled. Otherwise, the RTS_N output is forced inactive (high) when the FIFO is almost full, where “almost full” refers to two available slots in the FIFO. When RTS_N is connected to the CTS_N input of another UART device, the other UART stops sending serial data until the receiver FIFO has available space; that is, until it is completely empty.

The selectable receiver FIFO threshold values are: 1, 1/4, 1/2, 2

Since one additional character can be transmitted to the controller after RTS_N has become inactive—due to data already having entered the transmitter block in the other UART—setting the threshold to “2 less than full” allows maximum use of the FIFO with a safety zone of one character.

Once the receiver FIFO becomes completely empty by reading the Receiver Buffer Register (RBR), RTS_N again becomes active (low), signalling the other UART to continue sending data.

18.1.14.2 Auto CTS

Auto CTS becomes active when the following occurs:

•Auto Flow Control is selected during configuration

•AFCE (MCR[5] bit = 1)

•FIFOs are enabled through FIFO Control Register FCR[0] bit

When Auto CTS is enabled (active), controller transmitter is disabled whenever the CTS_N input becomes inactive (high); this prevents overflowing the FIFO of the receiving UART. If the CTS_N input is not inactivated before the middle of the last stop bit, another character is transmitted before the transmitter is disabled. While the transmitter is disabled, the transmitter FIFO can still be written to, and even overflowed.

18.1.15 Registers

Note: Please refer to the Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors for Internet of Things (IoT) Applications, Datasheet, Volume 2 (Book 2 of 3), Mule Creek Canyon (Document Number: 636722), for a description of the registers associated with subject of this chapter.

18.2 Intel® Serial I/O Inter-Integrated Circuit (I^2C) Controllers

18.2.1 Feature Overview

The PCH implements eight I^2C controllers for eight independent I^2C interfaces, I2C0-I2C7. Each interface is a two-wire serial interface consisting of a serial data line (SDA) and a serial clock (SCL).

The I^2C interfaces support the following features:

- Speed: standard mode (up to 100 Kb/s), fast mode (up to 400 Kb/s), fast mode plus (up to 1 MB/s) and High speed mode (up to 3.4 Mb/s).
- 1.8V or 3.3V support (depending on the voltage configured to the I^2C signal group)
- initiator I^2C operation only
- 7-bit or 10-bit addressing
- 7-bit or 10-bit combined format transfers
- Bulk transmit mode
- Ignoring CBUS addresses (an older ancestor of I^2C used to share the I^2C bus)
- Interrupt or polled-mode operation
- Bit and byte waiting at all bus speed
- Component parameters for configurable software driver support
- Programmable SDA hold time (t_{HD}; DAT)
- DMA support with 64-byte DMA FIFO per channel (up to 32-byte burst)
- 64-byte Tx FIFO and 64-byte Rx FIFO
- SW controlled serial data line (SDA) and serial clock (SCL)

Notes:

1. The controllers must only be programmed to operate in initiator mode only. I^2C target mode is not supported. Refer to Section 1.1.1 for more information on initiator and target.
2. I^2C multi initiators is not supported.

3. Simultaneous configuration of Fast Mode and Fast Mode Plus/High speed mode is not supported.
4. I^2C General Call is not supported.

18.2.2 Signal Description

Table 18-3. Signal Description

Group	Signal Name	Description
Clock	SIO_I2C[7:0]_SCL	I^2C clock signals
Data	SIO_I2C[7:0]_SDA	I^2C data signals

18.2.3 Functional Description

For more information on the I^2C protocols and command formats, refer to the industry I^2C specification. Below is a simplified description of I^2C bus operation:

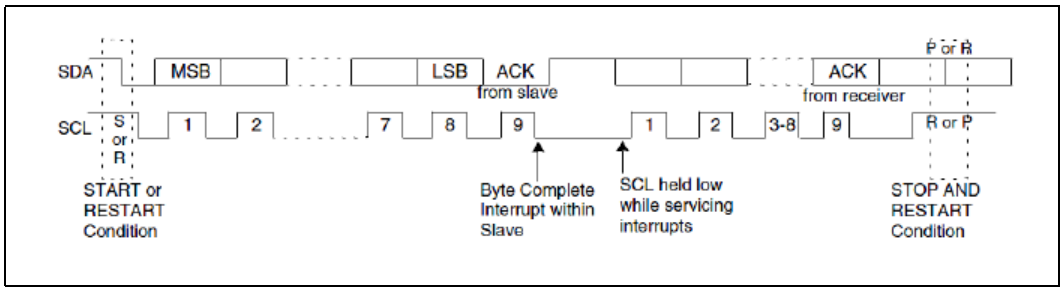
- The initiator generates a START condition, signaling all devices on the bus to listen for data.
- The initiator writes a 7-bit address, followed by a read/write bit to select the target device and to define whether it is a transmitter or a receiver.
- The target device sends an acknowledge bit over the bus. The initiator must read this bit to determine whether the addressed target device is on the bus.
- Depending on the value of the read/write bit, any number of 8-bit messages can be transmitted or received by the initiator. These messages are specific to the I^2C device used. After 8 message bits are written to the bus, the transmitter will receive an acknowledge bit. This message and acknowledge transfer continues until the entire message is transmitted.
- The message is terminated by the initiator with a STOP condition. This frees the bus for the next initiator to begin communications. When the bus is free, both data and clock lines are high.

18.2.3.1 Bus Speed Modes

The four supported bus speed modes defined by the I^2C protocol are

- I^2C High Speed (HS) mode ≤ 3.4 Mbps
- I^2C Fast Mode+ ≤ 1 Mbps
- I^2C Fast Mode ≤ 400 kbps
- I^2C Standard ≤ 100 kbps

Figure 18-3. Data Transfer on the I²C Bus



18.2.3.2 Combined Formats

The PCH I²C controllers support mixed read and write combined format transactions in both 7-bit and 10-bit addressing modes.

The PCH controllers do not support mixed address and mixed address format (which means a 7-bit address transaction followed by a 10-bit address transaction or vice versa) combined format transaction.

To initiate combined format transfers, IC_CON.IC_RESTSART_EN should be set to 1. With this value set and operating as a initiator, when the controller completes an I²C transfer, it checks the transmit FIFO and executes the next transfer. If the direction of this transfer differs from the previous transfer, the combined format is used to issue the transfer. If the transmit FIFO is empty when the current I²C transfer completes, a STOP is issued and the next transfer is issued following a START condition.

18.2.3.3 I²C Setup/Hold Time

The I²C protocol specifies a minimum SDA hold time. Timing delays between the initiator and target devices must be compensated for by dynamically adjusting the SDA hold time in the I²C host controller, to maintain a constant logic value until the SCL transitions. The IC_SDA_HOLD register extends the initiator's SDA hold time, using separate values for the initiator being a transmitter and a receiver. This register contains these separate values in the RX[23:16] and TX[15:0] bitfields. There are minimum values supported by the controller: a initiator device must be >1 and a target device must be > 7. The hold value granularity is the input clock (ic_clk=133MHz) period. The programmed SDA hold time must not exceed the low portion of the SCL period and must meet the minimum duration specified by the I²C protocol.

18.2.4 DMA Controller

The I²C controllers have an integrated DMA controller.

18.2.4.1 DMA Transfer and Setup Modes

The DMA can operate in the following modes:

1. Memory to peripheral transfers. This mode requires the peripheral to control the flow of the data to itself.
2. Peripheral to memory transfer. This mode requires the peripheral to control the flow of the data from itself.

The DMA supports the following modes for programming:

1. Direct programming. Direct register writes to DMA registers to configure and initiate the transfer.
2. Descriptor based linked list. The descriptors will be stored in memory (such as DDR or SRAM). The DMA will be informed with the location information of the descriptor. DMA initiates reads and programs its own register. The descriptors can form a linked list for multiple blocks to be programmed.
3. Scatter Gather mode.

18.2.4.2 Channel Control

- The source transfer width and destination transfer width is programmable. The width can be programmed to 1, 2, or 4 bytes.
- Burst size is configurable per channel for source and destination. The number is a power of 2 and can vary between 1,2,4,...,128. This number times the transaction width gives the number of bytes that will be transferred per burst.
- Individual channel enables. If the channel is not being used, then it should be clock gated.
- Programmable Block size and Packing/Unpacking. Block size of the transfer is programmable in bytes. The block size is not be limited by the source or destination transfer widths.
- Address incrementing modes: The DMA has a configurable mechanism for computing the source and destination addresses for the next transfer within the current block. The DMA supports incrementing addresses and constant addresses.
- Flexibility to configure any hardware handshake sideband interface to any of the DMA channels
- Early termination of a transfer on a particular channel.

18.2.5 Reset

Each host controller has an independent reset associated with it. Control of these resets is accessed through the Reset Register.

Each host controller and DMA will be in reset state once powered ON and require SW (BIOS or driver) to write into specific reset register to bring the controller from reset state into operational mode.

Note: To avoid a potential I^2C peripheral deadlock condition where the reset goes active in the middle of a transaction, the I^2C controller must be idle before a reset can be initiated.

18.2.6 Power Management

18.2.6.1 Device Power Down Support

To power down peripherals connected to PCH I^2C bus, the idle configured state of the I/O signals is retained to avoid voltage transitions on the bus that can affect the connected powered peripheral. Connected devices are allowed to remain in the D0 active or D2 low power states when I^2C bus is powered off (power gated). The PCH HW will prevent any transitions on the serial bus signals during a power gate event.

18.2.6.2 Latency Tolerance Reporting (LTR)

Latency Tolerance Reporting is used to allow the system to optimize internal power states based on dynamic data, comprehending the current platform activity and service latency requirements. The interface supports this by reporting its service latency requirements to the platform power management controller using LTR registers.

The controller's latency tolerance reporting can be managed by one of the two following schemes. The platform integrator must choose the correct scheme for managing latency tolerance reporting based on the platform, OS and usage.

1. Platform/HW Default Control. This scheme is used for usage models in which the controller's state correctly informs the platform of the current latency requirements.
2. Driver Control. This scheme is used for usage models in which the controller state does not inform the platform correctly of the current latency requirements. If the FIFOs of the connected device are much smaller than the controller FIFOs, or the connected device's end to end traffic assumptions are much smaller than the latency to restore the platform from low power state, driver control should be used.

18.2.7 Interrupts

I^2C interface has an interrupt line which is used to notify the driver that service is required.

When an interrupt occurs, the device driver needs to read the host controller, DMA interrupt status and TX completion interrupt registers to identify the interrupt source. Clearing the interrupt is done with the corresponding interrupt register in the host controller or DMA.

All interrupts are active high and their behavior is level triggered.

18.2.8 Error Handling

Errors that might occur on the external I^2C signals are comprehended by the I^2C host controller and reported to the I^2C bus driver through the MMIO registers.

18.2.9 I^2C Clock Period

The I²C initiator protocol depends on proper I/O timing to operate in a designated speed mode. The I²C host controller provides high and low SCL count registers (HCNT & LCNT) that calibrate the serial clock for the data rate.

The formulas to calculate the high and low clock counts are:

$$HCNT=(T\_SCL(high)-RC\ Delay)\cdot f\_input$$

$$LCNT=(T\_SCL(low)-RC\ Delay)\cdot f\_input$$

(where f_input=133MHz) (where RC delay is calculated by multiplying any output impedance by the capacitance load).

For example, High Speed mode (3.4 Mbps), an input clock of 133 MHz, an RC delay of 100ns (1 kΩ*100pF), and a 50% SCL duty cycle requires:

HCNT = LCNT =(((1/3.4M)*50%)-0.1µs)*133MHz = 7

The host controller programmable registers are labeled based on the bus speed mode, for example IC_HS_SCL_HCNT is the SCL high count for High speed mode. For the example above, the programmed values are:

IS_HS_SCL_HCNT = IS_HS_SCL_LCNT =7

18.2.10 Reference

Specification	Location
The I^2C Bus Specification, Version 5	www.nxp.com/documents/user_manual/UM10204.pdf?

18.2.11 Registers

Note: Please refer to chapter 14 of the Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors for Internet of Things (IoT) Applications, Datasheet, Volume 2 (Book 2 of 3), Mule Creek Canyon (Document Number: 636722), for a description of the registers associated with subject of this chapter.

18.3 Serial Peripheral Interface (SIO SPI)

18.3.1 Feature Overview

The PCH's three SIO SPI interfaces use SPI serial protocols for transferring data over short distances between many devices.

The interfaces [2:0] support 2 devices each and consists of 5 wires: a clock (CLK), 2 chip selects (CS0 and CS1) and two data lines (MOSI and MISO).

The PCH SIO SPI supports full-duplex and half-duplex modes. The interface operates in initiator mode only and supports serial bit rates up to 25Mb/s. Serial data formats may range from 4 to 32 bits in length.

18.3.2 Signal Description

Name	Type	Description
SIO_SPI[2:0]_CLK	O	**SIO SPI Clock:** SPI Clock signals
SIO_SPI[2:0]_CS0_N	O	**SIO SPI Chip Select 0:** SPI chip select 0 signals
SIO_SPI[2:0]_CS1_N	O	**SIO SPI Chip Select 1:** SPI chip select 1 signals
SIO_SPI[2:0]_MOSI	O	**SIO SPI Initiator OUT Target IN** data output signal.
SIO_SPI[2:0]_MISO	I	**SIO SPI Initiator IN Target OUT** data input signal.

18.3.3 Functional description

The SIO SPI controllers can only be set to operate as a initiator.

The processor or DMA accesses data through the SIO SPI port's transmit and receive

64 entry FIFOs. A processor access takes the form of programmed I/O, transferring one FIFO entry per access. Processor accesses must always be 32 bits wide. Processor writes to the FIFOs are 32 bits wide, but the PCH will ignore all bits beyond the programmed FIFO data size. Processor reads to the FIFOs are also 32 bits wide, but the receive data written into the Receive FIFO is stored with '0' in the most significant bits (MSB) down to the programmed data size.

The FIFOs can also be accessed by DMA, which must be in multiples of 1, 2, or 4 bytes, depending upon the EDSS value, and must also transfer one FIFO entry per access. For writes, the SIO SPI controller takes the data from the transmit FIFO, serializes it, and sends it over the serial wire to the external peripheral. Receive data from the external peripheral on the serial wire is converted to parallel words and stored in the receive FIFO.

A programmable FIFO trigger threshold, when exceeded, generates an interrupt or DMA service request that, if enabled, signals the processor or DMA respectively to empty the Receive FIFO or to refill the Transmit FIFO.
The SIO SPI controller, as a initiator, provides the clock signal and controls the chip select line. Commands codes as well as data values are serially transferred on the data signals. The PCH asserts a chip select line to select the corresponding peripheral device with which it wants to communicate. The clock line is brought to the device whether it is selected or not. The clock serves as synchronization of the data communication.

18.3.4 Interface Frequency

The interface frequency depends on the input serial clock frequency and the applied clock division. The SPI controller provides a rudimentary clock divider which lacks the precision to generate any frequency with a low error rate for a given input clock frequency. The SIO block provides an optional clock divider circuit which applies an M:N ratio to the input clock before the controller. This 15-bit division circuit provides < 0.01% error margin for all interface frequencies.

18.3.4.1 SPI Controller Divider

The formula to calculate the interface frequency with the controller's divider is:

Frequency = f_input/(SSCR0.SCR+1)

(Where f_input = 100MHz & SCR = 3 to 99)

The value of SSCR0.SCR for a specific frequency can be calculated as:

SSCR0.SCR = (f_input/Frequency)-1

Note: The M:N ratio must be configured to be 1:1 in this configuration.

18.3.4.2 SIO M/N Divider

The formula to calculate the interface frequency with the M:N ratio is:

Frequency = f_input*(M/N)
(SSCR0.SCR must be configured to be 0 in this configuration)

For a f_input value of 100MHz, the variable M can be calculated as follows:

M = (Frequency*N)/100M
(M should be rounded to be closest integer, if necessary)

For example, a target frequency of 25MHz requires a divider ratio of 0.25, which implies that M should be set to 0.25 of N. To achieve the lowest error margin, M and N should be assigned full 15-bit values, 8190 and 32760 respectively, in this example. The M and N divider values are programmable. The register programming sequence to use the M:N clock divider is defined in Table 18-4.

Table 18-4. Frequency Programming

Register.Field	Offset	Bits	Value	Access
CLOCK_PARMAS.M_VAL	0x200	30:16	0:M	MMIO
CLOCK_PARMAS.N_VAL	0x200	15:1	0:N	MMIO

Note: The M/N ratio cannot be changed when clock gating is enabled for the controller.

18.3.5 DMA controller

The SIO SPI controllers have an integrated DMA controller.

18.3.5.1 DMA Transfer and Setup Modes

The DMA can operate in the following modes:

- Memory to peripheral transfers. This mode requires that the peripheral control the flow of the data to itself.
- Peripheral to memory transfer. This mode requires that the peripheral control the flow of the data from itself.

The DMA supports the following modes for programming:

- Direct programming. Direct register writes to DMA registers to configure and initiate the transfer.
- Descriptor based linked list. The descriptors will be stored in memory. The DMA will be informed with the location information of the descriptor. DMA initiates reads and programs its own register. The descriptors can form a linked list for multiple blocks to be programmed.
- Scatter Gather mode

18.3.5.2 Channel Control

- The source transfer width and destination transfer width are programmable. The width can be programmed to 1, 2, or 4 bytes.
- Burst size is configurable per channel for source and destination. The number is a power of 2 and can vary between 1,2,4,...,128. this number times the transaction width gives the number of bytes that will be transferred per burst.
- Individual Channel enables. If the channel is not being used, then it should be clock gated.
- Programmable Block size and Packing/Unpacking. Block size of the transfer is programmable in bytes. the block size is not limited by the source or destination transfer widths.
- Address incrementing modes: The DMA has a configurable mechanism for computing the source and destination addresses for the next transfer within the current block. The DMA supports incrementing addresses and constant addresses.
- Flexibility to configure any hardware handshake sideband interface to any of the DMA channels.
- Early termination of a transfer on a particular channel.

18.3.6 Reset

Each host controller has an independent rest associated with it. Control of these resets is accessed through the Reset Register.
Each host controller and DMA will be in reset state once powered ON and require SW (BIOS or driver) to write into the corresponding reset register to bring the controller from reset state into operational mode.

18.3.7 Power Management

18.3.7.1 Device Power Down Support

In order to power down peripherals connected to the PCH SIO SPI bus, the idle configured state of the I/O signals must be retained to avoid transitions on the bus that can affect the connected powered peripheral. Connected devices are allowed to remain in the D0 active or D2 low power states when the bus is powered off (power gated). The PCH HW will prevent any transitions on the serial bus signals during a power gate event.

18.3.7.2 Latency Tolerance Reporting (LTR)

Latency Tolerance Reporting is used to allow the system to optimize internal power states based on dynamic data, comprehending the current platform activity and service latency requirements. However, the SIO SPI bus architecture does not provide the architectural means to define dynamic latency tolerance messaging. Therefore, the interface supports this by reporting its service latency requirements to the platform power management controller via LTR registers. The controller’s latency tolerance reporting can be managed by one of the two following schemes. The platform integrator must choose the correct scheme for managing latency tolerance reporting based on the platform, OS and usage.

- Platform/HW Default Control. This scheme is used for usage models in which the controller's state correctly informs the platform of the current latency requirements. In this scheme, the latency requirement is a function of the controller state. The latency for transmitting data to/from its connected device at a given rate while the controller is active is representative of the active latency requirements. On the other hand if the device is not transmitting or receiving data and idle, there is no expectation for end to end latency.
- Driver Control. This scheme is used for usage models in which the controller state does not inform the platform correctly of the current latency requirements. If the FIFOs of the connected device are much smaller than the controller FIFOs, or the connected device's end-to-end traffic assumptions are much smaller than the latency to restore the platform from low power state, driver control should be used.

18.3.8 Interrupts

SIO SPI interface has an interrupt line which is used to notify the driver that service is required. When an interrupt occurs, the device driver needs to read both the host controller and DMA interrupt status and transmit completion interrupt registers to identify the interrupt source. Clearing the interrupt is done with the corresponding interrupt register in the host controller or DMA. All interrupts are active high and their behavior is level interrupt.

18.3.9 Error Handling

Errors that might occur on the external SIO SPI signals are comprehended by the host controller and reported to the interface host controller driver through the MMIO registers.

18.3.10 SPI Mode Support

Four modes are supported for data latching, based on the configured clock polarity and phase. The clock polarity is configured using the SSCR1.SPO register field and the clock phase is configured using the SSCR1.SPH register field.

Mode	SPO	SPH	Clock Polarity In Idle State	Data Sampled On Clock	Data Output On Clock
0	0	0	Low	Rising Edge	Falling Edge
1	0	1	Low	Falling Edge	Rising Edge
2	1	0	High	Falling Edge	Rising Edge
3	1	1	High	Rising Edge	Falling Edge

18.3.11 Registers

Note: Please refer to chapter 9 of the Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors for Internet of Things (IoT) Applications,

Datasheet, Volume 2 (Book 2 of 3), Mule Creek Canyon (Document Number: 636722), for a description of the registers associated with subject of this chapter.

§ §

19 Storage

19.1 embedded Multi Media Card (eMMC*)

19.1.1 Overview

The eMMC* is a universal data storage and communication media. It is designed to cover a wide area of applications such as smart phones, tablets, computers, cameras, and so on. PCH supports only 1.8V operating devices and PCH supports eMMC* version 5.1.

19.1.1.1 Key Features Supported

- HW Command Queuing support complaint to eMMC* v5.1 specification
- Support enhanced Strobe for HS400 mode @1.8V
- Both ADMA2/DMA and Non-DMA mode of operation
- Transfers the data in 1 bit, 4 bit and 8 bit mode
- support 64b address
- Cyclic Redundancy Check CRC7 for command and CRC16 for data integrity
- Support for Tx Path tuning and retention of DLL delay values

19.1.2 Signals Description

Table 19-1. eMMC Signal Descriptions

Name	Type	Description
EMMC_CMD	I/O	eMMC* Command/Response
EMMC_DATA[7:0]	I/O	eMMC* Data
EMMC_RCLK	I	eMMC* Receive Clock (Data Strobe)
EMMC_CLK	O	eMMC* Clock
EMMC_RCOMP	I/O	External reference (200 Ohm+/- 1% pull down to ground)
EMMC_RST_N	O	Reset

19.1.3 Functional Description

The Controller handles eMMC* Protocol at transmission, packing data, adding cyclic redundancy check (CRC), start/end bit, and checking for transaction format correctness. Main supported features are listed below.

The eMMC* main use case is to connect an on board external storage device.

19.1.3.1 eMMC* 5.1 Command Queuing

Command Queuing (CQ) definition for eMMC* includes new commands for issuing tasks to the device, for ordering the execution of previously issued tasks & for additional task management function. The host controller with CQ can queue up to 32 commands to the device and the device selects and indicates one of the queued commands to host for service.

The host controller implements additional logic for handling a door-bell based DMA for the 32 descriptor / task list and manages the entire CQ flow which includes:

- Fetch and send the tasks/commands to device using existing logic
- Maintains context of each queued command
- Periodically read the device queue status & indicates completion of task to SW.
- Implements interrupt coalescing to reduce burden on software ISR.

19.1.3.2 eMMC* 5.1 Enhanced Strobe

Enhanced Strobe Mode for HS400 improves upon the HS400 mode interface speed increase that was first defined in eMMC* version 5.0, by facilitating faster synchronization between the host and the device.

Refer JEDEC eMMC* 5.1 specification for additional information.

19.1.3.3 eMMC* Working Modes

The following table shows the working modes of eMMC*. Since the processor uses a base clock of 200MHz, the actual throughput for the processor will vary as indicated.

Table 19-2. eMMC* Working Modes

eMMC* Mode	Data Rate	Clock Frequency	Max. Data Throughput	Max. Data Throughput (Actual)
Compatibility	Single	0 – 26 MHz	26 MB/s	25 MB/s
High Speed SDR	Single	0 – 52 MHz	52 MB/s	50 MB/s
High Speed DDR	Dual	0 – 52 MHz	104MB/s	100 MB/s
HS200	Single	0 - 200 MHz	200 MB/s	200 MB/s
HS400	Dual	0 - 200 MHz	400 MB/s	400 MB/s

19.2 Secure Digital eXtended Capacity (SDXC)

The SDXC controller is to connect to an external detachable storage and/or I/O devices. It supports SD Card specification version 3.01 and SDIO specification version 3.0.

19.2.1 SDXC Signal Description

Table 19-3. SDXC Signals

Group	Signal Name	Description
Clock	SD_SDIO_CLK	SDXC Clock signal
Data	SD_SDIO_D[3:0]	SDXC Data signals
Command	SD_SDIO_CMD	SDXC Command signal
Control	SD_SDIO_CD_N	SD Card detect
	SD_SDIO_WP	SD card write protect
Power Enable	SD_SDIO_PWR_EN_N	SD card power enable 3.3V

19.2.2 Key Features Supported

- Support SD 3.01 @ 1.8V Signaling (UHS-1@ SDR 104/50/25/12 & DDR50)
- Support SD 3.01 @ 3.3V Signaling (Default Speed Mode/High Speed Mode)
- Support Cyclic Redundancy Check CRC7 for command and CRC16 for data integrity
- Support Card Detection (Insertion / Removal) (SD memory card only)
- Suport D1-line wake from S0/D0i3 (To enable SDIO v3.00 on SD Removable card slot)

19.2.3 Functional Description

The SDXC controller handles SD Protocol at transmission, packing data, adding cyclic redundancy check (CRC), start/end bit, and checking for transaction format correctness. The main use case for SDXC is to connect to an external detachable storage and /or I/O device. Both 1.8V and 3.3V signaling is supported. Additional information can be obtained from the specifications stated above.The following chart maps the working modes of SDXC.

Table 19-4. SD Working Modes

SDXC Mode	Data Rate	Clock Frequency	Maximum Data Throughput
Default Speed/SDR12	Single	0 – 25 MHz	12.5 MB/s
High Speed/SDR25	Single	0 – 50 MHz	25 MB/s
SDR50	Single	0 – 100 MHz	50 MB/s
DDR50	Dual	0 – 50 MHz	50 MB/s
SDR104	Single	0 – 200 MHz	100 MB/s

§ §

20 Clocking

20.1 Integrated Clock Controller (ICC)

The processor requires several single-ended and differential clocks to synchronize signal operations and data propagations system wide between many interfaces and across multiple clock domains. The PCH generates and provides this complete system clocking solution through its Integrated Clock Controller (ICC).

The external clock sources for the PCH are 38.4MHz crystal clock and 32KHz RTC clock.

20.2 PCH ICC Clocking

The PCH ICC Hardware includes the following clocking.

- **"iSCLK" (See Figure 20-1)**
  - Main PLL = Clocks generated from this PLL are non-SSC clocks.
  - OC PLL = OCPLL is SSC enabled and supports under-clocking for Memory Adaptive Clocking Technology (memACT).
  - IOTG PLL = This PLL is SSC enabled.
- **"modPHY (Modular Physical Layer)" (See Figure 20-2)**
  - USB 3.1/ Gen2 PCIe PLL = It generates the 100MHz SSC reference clock to SATA PLL, OPI PLL, Gen3 PCIe PLL, CPU PLL and external PCIE devices.
  - MIPI PLL = Provides clocking support for high speed serial data rate on the MIPI MPHY interfaces.
  - SATA PLL = This PLL generates 300MHz core clock to the SATA controller in the core and generates 125MHz/312.5MHz core clock to the Integrated 2.5GbE controller in the core.
  - Gen3 PCIe PLL = This PLL generates 500MHz link clock to the PCIe controllers in the core.
- **"Intel® Programmable Services Engine (Intel® PSE)" (See Figure 20-3)**
  - This PLL is used to generate clocks for the following use cases such as clock to the microcontroller during S0,RGMII support and PTP timers.

Figure 20-1. Internal Clock Diagram - "iSCLK"

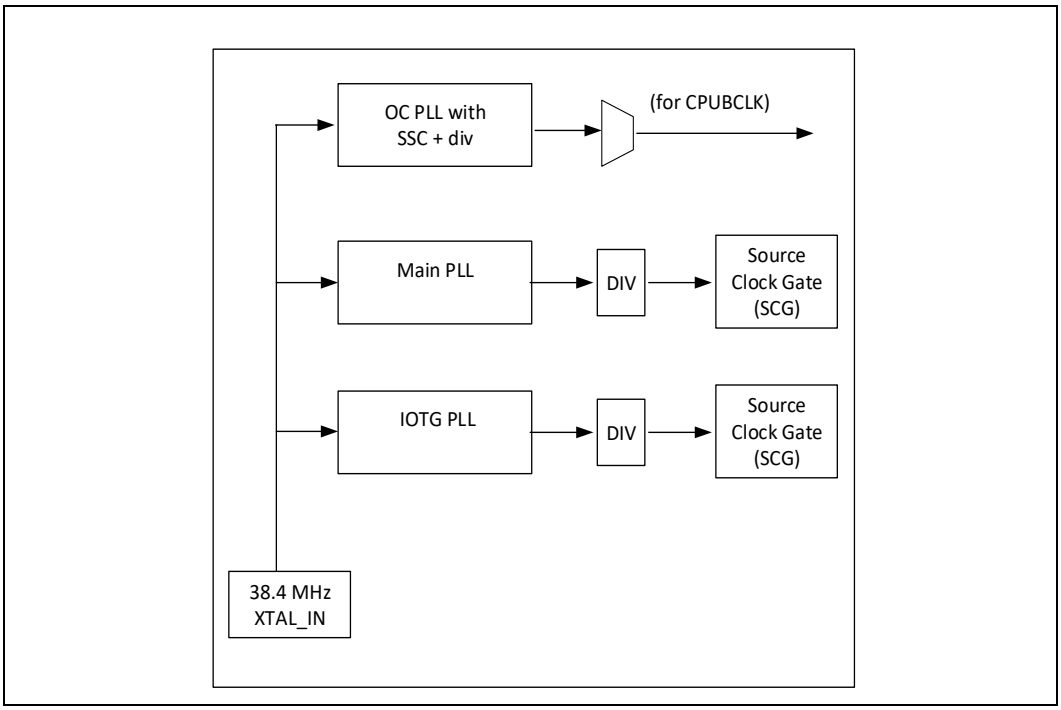
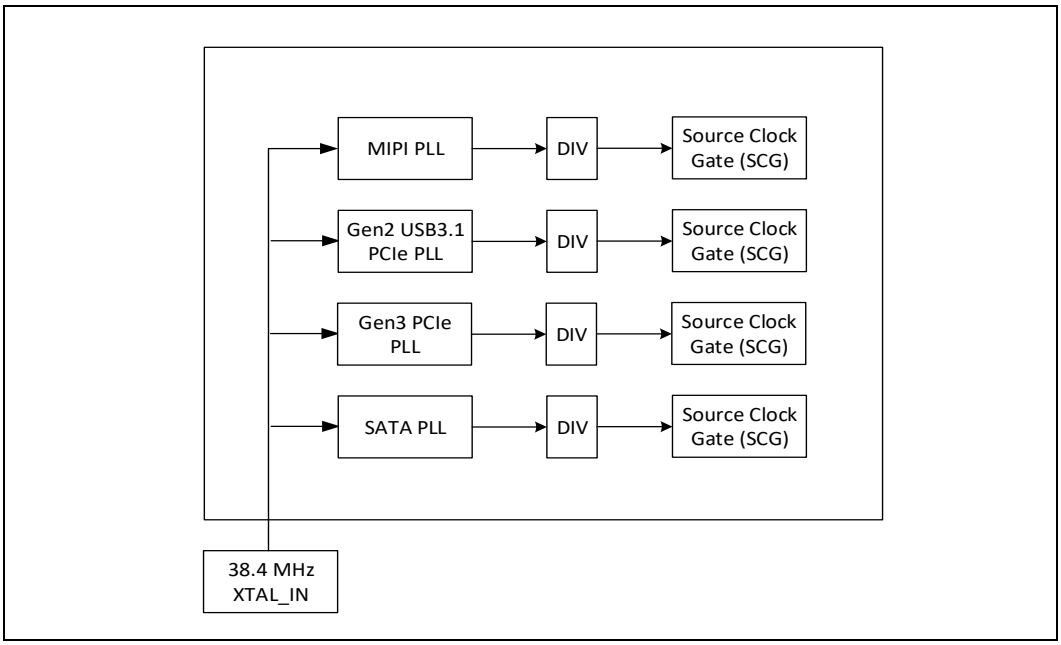


Figure 20-2. Internal Clock Diagram - "modPHY"



Note: Phase Lock Loop (PLL). Hardware control systems used to generate stable output clock frequencies.

Figure 20-3. PSE_Clocking

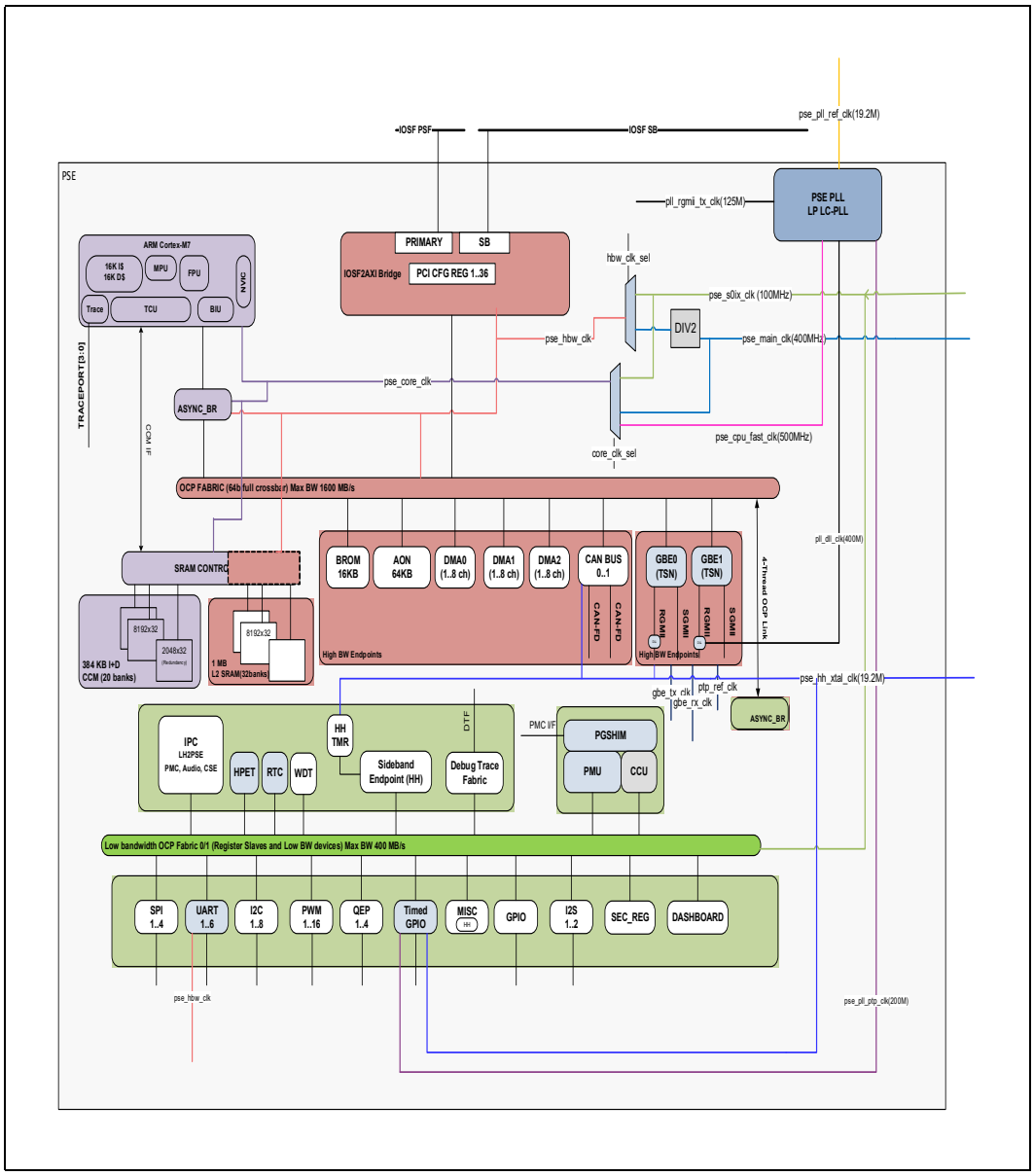


Figure 20-4. PSE_GBe Clocking

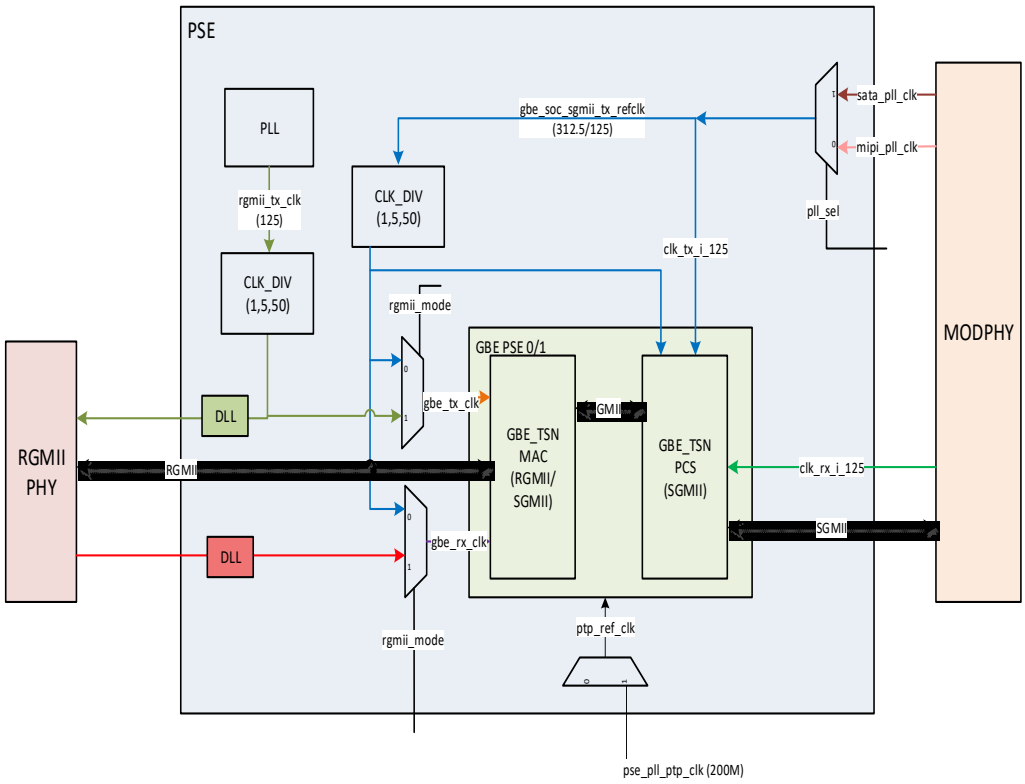


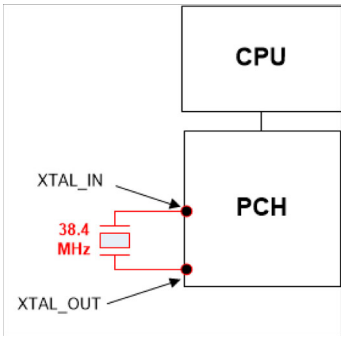
Table 20-1. Intel ® PSE Clock Distribution

IP	Frequency (MHz)	Description
GPIO0	100	GPIO0 controller clock
GPIO1	100	GPIO1 controller clock
HBW_FABRIC	200/100	High Bandwidth fabric clock. 200MHz = Host devices active (not in D3 or D0i3) 100MHz = Host devices inactive (D3 or D0i3) Note: FW needs to switch the clock using HBW_CLK_SEL.CLK_SELECT
GbE CSR clock	Same as HBW_FABRIC clock	Ethernet controller Control and Status registers Interface clock
HPET	32.768K	HPET clock
12C*[7:0]	100	I^2C controller clock
I2S*[1:0]	200	I^2S controller clock

IP	Frequency (MHz)	Description
PERI_FABRIC	100	Low Speed Peripheral fabric clock
PWM*[1:0]	100	PWM controller clock
QEP*[3:0]	100	QEP controller clock
SPI*[3:0]	100	SPI controller clock
TGPIO*[1:0]	19.2	Local ART configuration TGPIOCTLx.Timer Select = 0b00
	200	TMT_0,TMT_1,TMT_2 configuration TGPIOCTLx.Timer Select = 0b01/0b10/0b11
UART*[5:0]	200/100	UART controller clock, same as HBW_FABRIC
WDT	100	Watchdog timer clock
SRAM_L2	200/100	L2 SRAM clock, same as HBW_FABRIC
ARM (Cortex-M7) core	500/400/100	Core clock 400MHz = Host devices active (not in D3 or D0i3) 100MHz = Host devices inactive (D3 or D0i3) Note: FW needs to switch the clock using CORE_CLK_SEL.CLK_SELECT
CCM	500/400/100	CCM clock, same as Core clock
TRACECLK	200/100	Debug Trace clock, same as HBW_FABRIC

20.3 PCH ICC XTAL Input Configuration

Figure 20-5. PCH ICC XTAL Input Configuration



20.4 Summary of Clock Signal

Interface	Clock Signal	Clock Frequency
Memory - DDR4	DDR_[1:0]_CLK_DP/DN	Up to 1600 MHz
Memory - LPDDR4	LP4_[3:0]_CLK_DP/DN	Up to 2133 MHz
PCIe	PCIE_CLK[5:0]_DP, PCIE_CLK[5:0]_DN	100 MHz
Storage - eMMC	EMMC_CLK	2, 50, 200 MHz
Display - HDMI*DDC	DDI[2:0]_DDC_SCL DDI[2:0]_DDC_SDA	100kHz
Display - HDMI*	DDI[2:0]_TXN3 DDI[2:0]_TXP3 (TMDS[2:0] Clock)	19.2 MHz, 24 MHz, 38.4 MHz
Display - MIPI*DSI	DDI0_TXN2 DDI0_TXP2 MIPIA Clock	19.2MHz reference (CLK Frequency 300-1066 MHz)
Audio - HD Audio	HDA_BCLK	6, 12, 24 MHz
Audio Codec/Analog Microphone - I2S	AVS_I2S_MCLK[2:1] AVS_I2S[5:0]_SCLK	BCLK = 12.288 MHz MCLK = 19.2 MHz
Audio - Digital Microphone	DMIC_CLK_A0/B0 DMIC_CLK_A1/B1	12MHz
SIO (LPSS) - I^2C	SIO_I2C[7:0]_SDA SIO_I2C[7:0]_SCL	100 KHz, 400 KHz, 1 MHz, 3.4 MHz
SIO (LPSS) - SPI	SIO_SPI[2:0]_CLK	Up to 25 MHz

Interface	Clock Signal	Clock Frequency
SMBus	SMB_CLK	Maximum 100 KHz
SUSCLK	PMC_SUSCLK	32.768 KHz
XTAL Source - XTAL Clock	XTAL_IN XTAL_OUT	38.4 MHz
XTAL Source - RTC Clock	RTC_X[1,2]	32.768 KHz
FAST SPI	FSPI_CLK	20, 33 and 50MHz
ISI I2C	ISI_I2CS_SCL ISI_I2CS_SDA	3.4 MHz
SMLink	SMB_CLK	100 KHz
eSPI	ESPI_CLK	14 MHz, 20 MHz, 33 MHz and 50 MHz
SVID	SVID_CLK	26.25 MHz
JTAG	PCH_JTAG_TCK	100 MHz
SD Card/ SDIO	SD_SDIO_CLK	200 MHz
RGMII	PSE_GBE0_RGMII_RXCLK PSE_GBE0_RGMII_RXCTL PSE_GBE1_RGMII_RXCLK PSE_GBE1_RGMII_RXCTL	125 MHz

20.5 Registers

Please refer to the Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors for Internet of Things (IoT) Applications, Datasheet, Volume 2 (Book 1-3), for a description of the registers associated with other interface clocks.

§ §

21 General Purpose Input and Output (GPIO)

21.1 Overview

The General Purpose Input/Output (GPIO) signals are grouped into multiple groups.

The high level features of GPIO:

- Configurable 3.3V or 1.8V voltage
- Configurable as an input or output signal.
- Configurable GPIO pad ownership by host or Intel® Programmable Services Engine (PSE).
- SCI (GPE) and IOAPIC interrupt capable on most GPIs
- NMI and SMI capability capable (on selected GPIs).

21.2 Pad Grouping, Muxing, and Capabilities

Pads are grouped to families, and families grouped into communities. The type of buffer for each pad, along with the capabilities.

21.2.1 Buffer capabilities

While every buffer has the same set of registers, not all registers are applicable for all buffer types.

For example, there are options in each GPIO's DW0.TERM register field for 1k & 5k wpu (weak pull-up) and wpd (weak pull-down) resistors but these options are not supported and should not be used.

21.3 Functional Description

21.3.1 Programmable Hardware Debouncer

Hardware debounce capability is supported on GP_DSW3/PWRBTN# pad. The capability can be used to filter signal from switches and buttons if needed.

The period can be programmed from 8 to 32768 times of the RTC clock by programming the Pad Configuration DW2 register. At 32 kHz RTC clock, the debounce period is 244us to 1s.

21.3.2 Configurable GPIO Voltage

Except for all pads in GP_DSW (3.3V only), GP_S (1.8V Only), and GP_V (1.8V Only) groups, all other GPIO pads support per-pad configurable voltage, which allows control selection of 1.8V or 3.3V for each pad. The configuration is done via soft straps.

Before soft straps are loaded, the default voltage of each pin depends on its default as input(GP_In) or output(GP_Out).

- Input: 1.8V and 3.3V.
- Output: the pin drives 3.3V via a ~20K pull-up.

Warning: GPIO pad voltage configuration must be set correctly depending on device connected to it; otherwise, damage to the PCH or the device may occur.

21.3.3 Integrated Pull-ups and Pull-downs

All GPIOs have programmable internal pull-up (20kOhm nominal) and pull-down (20kOhm nominal) resistor which are off by default. The internal pull-up/pull-down for each GPIO can be enabled by BIOS programming the corresponding PAD_CFG_DW0 register. Refer to the Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors for Internet of Things (IoT) Applications, Datasheet, Volume 2 (Book 1-3), for more register information.

21.3.4 SCI / SMI# and NMI

SCI capability is available on all GPIOs except GP_S, while SMI and NMI capability is available on only selected GPIOs.

Below are the PCH GPIOs that can be routed to generate SMI# or NMI:

- GP_B14, GP_B20, GP_B23
- GP_C[23:22]
- GP_D[4:0]
- GP_E[8:0], GP_E[16:13]

21.3.5 Time-Aware GPIO

The PCH has two (2) Time-Aware GPIO controllers. Time-Aware GPIOs are muxed on GPIO pins as a native function (PMC_TGPIO). Each Time-Aware GPIO can be independently configured as an input or an output.

21.3.5.1 Input Event Capture

When the Time-Aware GPIO hardware is configured for input, an input event triggers hardware capture of the 19.2MHz Always Running Timer (ART) counter in the Time Capture (TGPIOTCV) register. The input event type is selected using the Event Polarity (EP) field of the Control (TGPIOCTL) register. There are three event types: rising edge, falling edge, or both rising and falling edges (toggle edge).

These three input event types are shown in Figure 21-1 – Figure 21-3. The level associated with an input edge/event must be asserted for a period of at least three ART clock ticks in order for the event to be recognized.

Figure 21-1. Input Capture Rising Edge

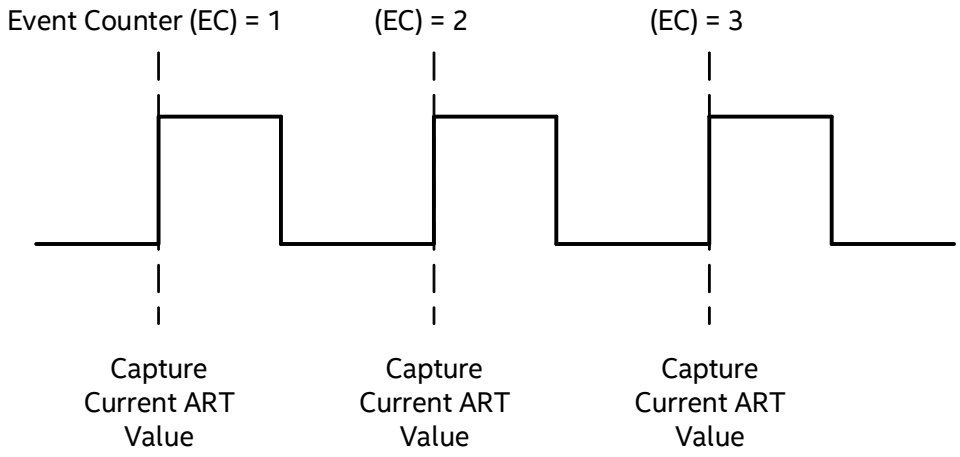


Figure 21-2. Input Capture Falling Edge

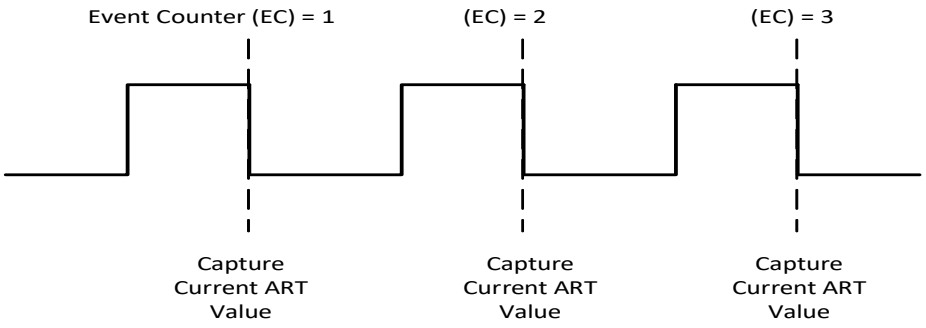
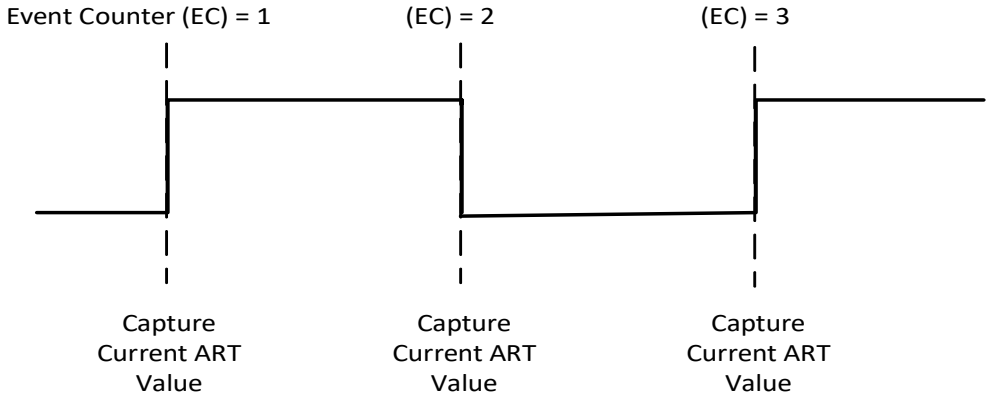


Figure 21-3. Input Capture Both (Toggle) Edge(s)



21.3.5.2 Output Event Generation

When the Time-Aware GPIO hardware is configured for output, an output event is triggered if the ART counter matches the software programmed time in the Comparator (TGPIOCOMP) register. If periodic mode is enabled, periodic output events are generated based on the programmed time interval (in units of ART time) in the Periodic Interval Value (TGPIOPIV) register. The programmed interval must be three (3) or more ART clock ticks.

The output event type is selected using the Event Polarity (EP) field of the Control (TGPIOCTL) register. There are three event types: rising pulse, falling pulse, or a single toggle edge. When rising pulse is selected, the output signal is disabled by default and the event enables output for a short interval. When falling pulse is selected, the output signal is enabled by default and the event disables output for a short interval. The interval for both rising and falling pulses is two (2) ART clock ticks. Rising and falling output pulse types are shown in Figure 21-4 and Figure 21-5.

Figure 21-4. Output Generation Rising Pulse

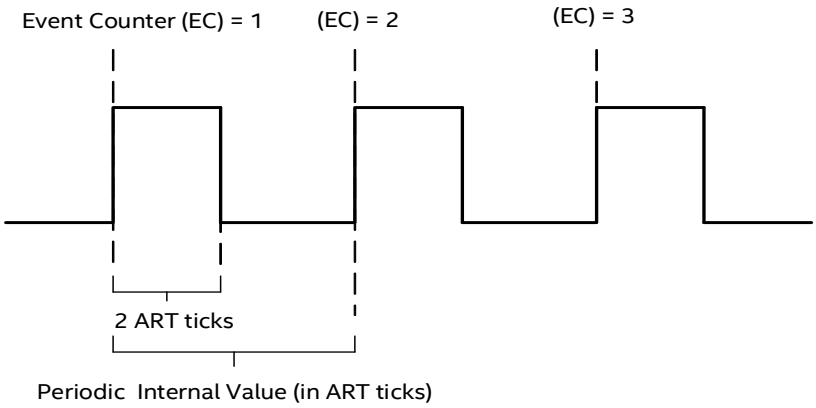
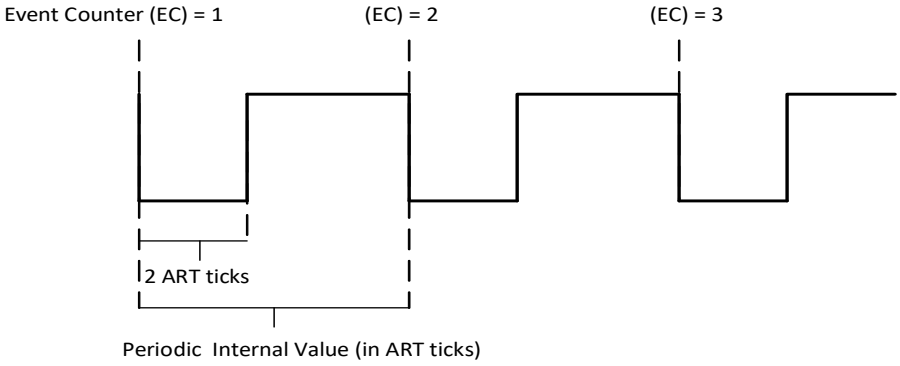
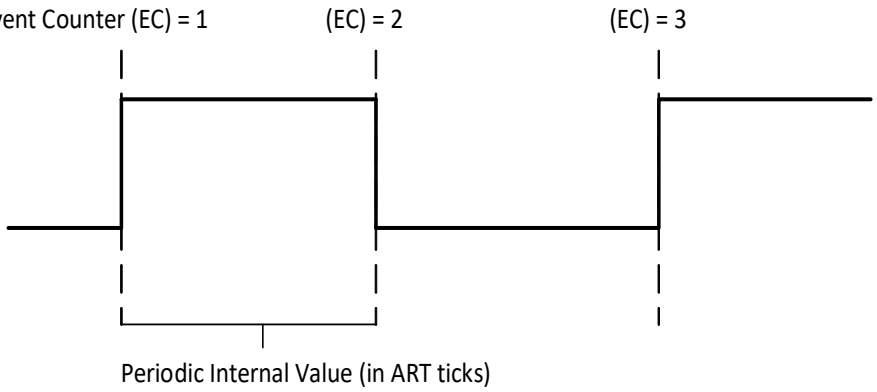


Figure 21-5. Output Generation Falling Pulse



When toggle edge is selected, the event triggers a single edge changing the signal from enabled to disabled or disabled to enabled depending on the current output state. Toggle edge output is shown in Figure 21-6. The output event also triggers capture of the current ART time in the Time Capture (TGPIOTCV) register in the same way externally driven input events are captured.

Figure 21-6. Output Generation Toggle Edge



21.3.5.3 Input / Output Event Counting

The Time-Aware GPIO also supports an event counter. When Time-Aware GPIO is configured as an input, the event counter increments by one (1) for every input event triggered. When Time-Aware GPIO is configured as output, the event counter increments by one (1) for every output event generated. The event count can be read using the Event Counter Capture (TGPIOECCV) register. The Event Counter Capture (TGPIOECCV) and Time Capture (TGPIOTCV) register values correspond to the same event if the TGPIOTCV0_31_0 register is read first to lock the values in both registers.

When Time-Aware GPIO is configured for input the event count is used to determine if the software missed an event. Input event counting is shown in Figure 21-1 – Figure 21-3.

When Time-Aware GPIO is configured for periodic output, the event count is used to determine the average output event period in terms of ART when software modulates the output period. The cumulative average period can be calculated by dividing the ART delta from the Time Capture register by the event count delta from the Event Counter Capture register. Output event counting is shown in Figure 21-4 – Figure 21-6.

21.3.5.4 TGPIO Usage

The principle usage of the Time-Aware GPIO hardware is to synchronize time. The Linux* OS system clock by default uses the CPU Time Stamp Counter (TSC) as its hardware clock source. The TSC is directly related to the ART clock. Time-Aware GPIO event times are translated to system time using the relation between ART and the system clock.

Synchronized periodic signals are used to align clocks between connected devices. A pulse-per-second (PPS) signal is an example of such a periodic signal. A PPS signal is a 1 Hz square wave aligned to the system clock of the transmitter. The receiver uses the PPS signal to align its clock to the system clock of the transmitting device. Other

frequencies, such as 1 kHz, may also be used. Each cycle of the signal provides a synchronizing event to align the receiver clock, and as a result, a higher frequency signal may result in more precise synchronization.

The Time-Aware GPIO hardware can be configured to generate a synchronization signal (for example, PPS) that is aligned with the system clock. External devices that accept a synchronization input signal use this to align their internal clocks to the system clock.

The Time-Aware GPIO hardware can be configured to capture an input synchronization signal. Each cycle, software uses the Time-Aware GPIO hardware to compute the offset between the systemclock and the transmitter clock. Software uses the offset to adjust the system clock to track the transmitter clock.

More information regarding the relationship between ART and TSC and the ART frequency can be found in chapters 17.17.4 "Invariant Timekeeping" and 18.7.3 "Determining the Processor Base Frequency" of the *Intel Software Developer's Manual (SDM),* respectively.

Note: When Time-Aware GPIO is enabled, the crystal oscillator driving ART will not be shut down because the crystal clock is needed for the Time-Aware GPIO operation. As a result, PMC_SLP_S0_N will not be asserted. This affects platform power because S0ix active idle states cannot be reached. For optimum power saving performance, software should only enable Time-Aware GPIO when needed and disable it, using the control register, when Time-Aware GPIO functionality is not required.

21.3.6 Shared RCOMP

GPIO supports shared RCOMP feature which reduces the number of required on board precision resistors. Instead of one to one connection, all the 5 RCOMP circuits *can* share the same external precision resistor.

During power-up, the first calibration will start automatically.

21.3.7 Glitch-free Operations

For some products, there is a requirement to make sure pad state is glitch-free and does not cause unexpected toggling due to for instance:

— Platform undergoes power up sequence

— Software changes the configuration settings like Pad Mode

21.3.7.1 Power up Sequencing

Glitch-free pad state on power up sequence is governed by the GPIO buffer if its I/O voltage rails ramp up before the GPIO controller voltage rail. The GPIO buffer contains internal voltage detection logic to check if the GPIO controller voltage rail has reached a threshold (0.5v at typical corner). If it hasn't, GPIO keep its TX driver tri-stated and enables a 20k ohm weak pull down. If the GPIO I/O voltage rail(s) ramp after the GPIO controller's, then the GPIO controller must guarantee the glitch-free pad state on power up.

21.3.8 Pad Driver Impedance

The pad driver impedance (also called buffer drive strength) for some GPIOs, or the native function(s) multiplexed on those GPIOs, can be modified to optimize DC & AC characteristics of TX signals. Control is exposed through three register fields that are available for each GPIO family.

Note: A GPIO family supports pad driver impedance modification if the register field is indicated as being RW (Read/Write) accessible.

FAM_CFG_Reg_xxx.STRSEL

FAM_RCOMP_A_DW0_Reg_xxx.PSTR

Note: This modifies the logic high driver impedance

FAM_RCOMP_A_DW0_Reg_xxx.NSTR (Logic low driver)

Note: This modifies the logic low driver impedance

21.3.8.1 Pad Driver Impedance Modification

Increasing the value of any of these fields decreases the driver impedance (increasing the driver strength) but the change does not vary proportionally with the field value. For this reason, it is necessary to take an experimental & iterative approach to find an optimal configuration while monitoring the electrical signal integrity.

21.3.8.1.1 Coarse Pad Driver Impedance Modification

If a RW accessible FAM_CFG_Reg_xxx.STRSEL register is available, sweep through all possible STRSEL values while monitoring signal integrity. Identify the minimum & maximum STRSEL values where DC and AC specifications are met and then identify the median value.

21.3.8.1.2 Coarse Pad Driver Impedance Modification

If a RW accessible FAM_RCOMP_A_DW0_Reg_xxx.PSTR/NSTR register is available, using the median STRSEL value, sweep through NSTR/PSTR values in increments of five while measuring signal integrity. Identify the minimum and maximum NSTR/PSTR values, where DC and AC specifications are met and then identify the median value.

21.3.9 I/O Standby

Except for GPIOs in GP_DSW, all GPIOs can be configured to be in a specific state while the processor is in a non-S0 ACPI state. This GPIO state is configured by two register fields that are unique to each configurable GPIO: PAD_CFG_DW1_GPPC_x_yy.IOSSTATE & PAD_CFG_DW1_GPPC_x_yy.IOSTERM. These fields default to 0h for all configurable GPIOs and a different value can be enabled by BIOS programming.

21.3.9.1 I/O Standby State (IOSSTATE)

I/O Standby State defines which state the GPIO should be parked in when the processor is in a non-S0 ACPI state.

IOSSTATE	Description
0	Latch last value driven on TX, TX Enable and RX Enable
1	Drive 0 with RX disabled and RX drive 0 internally
2	Drive 0 with RX disabled and RX drive 1 internally
3	Drive 1 with RX disabled and RX drive 0 internally
4	Drive 1 with RX disabled and RX drive 1 internally
5	Drive 0 with RX enabled
6	Drive 1 with RX enabled
7	Hi-Z with RX drive 0 internally
8	Hi-Z with RX drive 1 internally
9	TX Disabled and RX Enabled (i.e. wake or interrupt)
15	IO Standby signal is masked for this pad. In this mode, a pad operates as if IO Standby has not been asserted.
Others	Reserved

21.3.9.2 I/O Standby Termination (IOSTERM)

I/O Standby Termination defines the behavior of the integrated internal pull-up and pull-down resistors when the processor is in a non-S0 ACPI state. The value of the pull-up/pull-down is determined in the PAD_CFG_DW1_GPPC_x_yy.TERM register field

IOSSTERM	Description
0	Latch last value driven on TX, TX Enable and RX Enable
1	Drive 0 with RX disabled and RX drive 0 internally
2	Drive 0 with RX disabled and RX drive 1 internally
3	Drive 1 with RX disabled and RX drive 0 internally

21.4 GPIO Multiplexing Table

Table 21-1. GPIO Multiplexing Table (Sheet 1 of 42)

GPIO (EDS)	EDS Signal Name	Default Function	Default Termination	I/O Voltage Support	Native Function 1 (EDS)	Native Function 2 (EDS)	Native Function 3 (EDS)	Native Function 4 (EDS)	Native Function 5 (EDS)	Native Function 6 (EDS)	Native Function 7 (EDS)	Native Function 8 (EDS)	Pin State During Reset	Pin State Immediately After Reset	SCI/GPE IS	SMI	NMI	Output power sequence deglitch	GPIO Community #
GP_DSW00	GP_DSW00/ PMC_BATLOW_N	Native F1	None (BATLOW Disabled) 20K PU (BATLOW Enabled)	3.3V	PMC_BATLOW_N								Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	2
GP_DSW01	GP_DSW01/ PMC_ACPRESENT	Native F1	Native	3.3V	PMC_ACPRESENT								Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	2
GP_DSW10	GP_DSW10/ PMC_SLP_S5_N	Native F1/GPOut	None	3.3V	PMC_SLP_S5_N								Z	Z	Yes	No	No	Hi-Z output with internal 20K PD during power sequencing	2
GP_DSW11	GP_DSW11	Native F1/GPOut	None	3.3V	RSVD								Z	Z	Yes	No	No	Hi-Z output with internal 20K PD during power sequencing	2
GP_DSW02	GP_DSW02	Native F1	Native	3.3V	RSVD								Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	2
GP_DSW03	GP_DSW03/ PMC_PWRBTN_N	Native F1	20K PU	3.3V	PMC_PWRBTN_N								Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	2
GP_DSW04	GP_DSW04/ PMC_SLP_S3_N	Native F1/GPOut	None	3.3V	PMC_SLP_S3_N								Z	Z	Yes	No	No	Hi-Z output with internal 20K PD during power sequencing	2

Table 21-1. GPIO Multiplexing Table (Sheet 2 of 42)

GPIO (EDS)	EDS Signal Name	Default Function	Default Termination	I/O Voltage Support	Native Function 1 (EDS)	Native Function 2 (EDS)	Native Function 3 (EDS)	Native Function 4 (EDS)	Native Function 5 (EDS)	Native Function 6 (EDS)	Native Function 7 (EDS)	Native Function 8 (EDS)	Pin State During Reset	Pin State Immediately After Reset	SCI/GPE IS	SMI	NMI	Output power sequence deglitch	GPIO Community #
GP_DSW05	GP_DSW05/ PMC_SLP_S4_N	Native F1/GP-Out	None	3.3V	PMC_SLP_S4_N								Z	Z	Yes	No	No	Hi-Z output with internal 20K PD during power sequencing	2
GP_DSW07	GP_DSW07	GP-Out	None	3.3V									Z	Z	Yes	No	No	None	2
GP_DSW08	GP_DSW08/ PMC_SUSCLK	Native F1	None	3.3V	PMC_-SUSCLK								Z	Z	Yes	No	No	Hi-Z output with internal 20K PD during power sequencing	2
GP_DSW09	GP_DSW09	Native F1/GP-Out	None	3.3V	RSVD								Z	Z	Yes	No	No	Hi-Z output with internal 20K PD during power sequencing	2
GP_R00	GP_R00/HDA_BCLK/ AVS_I2S0_SCLK/ PSE_I2S0_SCLK	Native F1	None	1.8V/ 3.3V	HDA_BCLK	AVS_I2S0_SCLK	PSE_I2S0_SCLK	RSVD					Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	5
GP_R01	GP_R01/ HDA_SYNC/ AVS_I2S0_SFRM/ PSE_I2S0_SFRM	Native F1	Native	1.8V/ 3.3V	HDA_SYNC	AVS_I2S0_SFRM	PSE_I2S0_SFRM						Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	5
GP_R02	GP_R02/ HDA_SDO/ AVS_I2S0_TXD/ PSE_I2S0_TXD/ DMIC_CLK_B0	Native F1	Native	1.8V/ 3.3V	HDA_SDO	AVS_I2S0_TXD	PSE_I2S0_TXD	RSVD	DMIC_CLK_B0				Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	5
GP_R03	GP_R03/HDA_SDI0/ AVS_I2S0_RXD/ PSE_I2S0_RXD/ DMIC_CLK_B1	Native F1	Native	1.8V/ 3.3V	HDA_SDI0	AVS_I2S0_RXD	PSE_I2S0_RXD	RSVD	DMIC_CLK_B1				Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	5

Table 21-1. GPIO Multiplexing Table (Sheet 3 of 42)

GPIO (EDS)	EDS Signal Name	Default Function	Default Termination	I/O Voltage Support	Native Function 1 (EDS)	Native Function 2 (EDS)	Native Function 3 (EDS)	Native Function 4 (EDS)	Native Function 5 (EDS)	Native Function 6 (EDS)	Native Function 7 (EDS)	Native Function 8 (EDS)	Pin State During Reset	Pin State Immediately After Reset	SCI/GPE IS	SMI	NMI	Output power sequence deglitch	GPIO Community #
GP_R04	GP_R04/ HDA_RST_N/ DMIC_CLK_A1	Native F1	None	1.8V/ 3.3V	HDA_RST_N				DMIC_CLK_A1				Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	5
GP_R05	GP_R05/HDA_SDI1/ AVS_I2S1_RXD/ DMIC_DATA1	GP-In	None	1.8V/ 3.3V	HDA_SDI1	AVS_I2S1_RXD			DMIC_DATA1				Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	5
GP_R06	GP_R06/ AVS_I2S1_TXD/ DMIC_CLK_A0	GP-In	None	1.8V/ 3.3V		AVS_I2S1_TXD			DMIC_CLK_A0				Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	5
GP_R07	GP_R07/ AVS_I2S1_SFRM/DMIC_DATA0	GP-In	None	1.8V/ 3.3V		AVS_I2S1_SFRM			DMIC_DATA0				Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	5
GP_S00	GP_S00	Native F1	20K PD	1.8V									Z	Z	No	No	No	Hi-Z output with no internal termination during power sequencing	3
GP_S01	GP_S01	Native F1	20K PD	1.8V									Z	Z	No	No	No	Hi-Z output with no internal termination during power sequencing	3

Table 21-1. GPIO Multiplexing Table (Sheet 4 of 42)

GPIO (EDS)	EDS Signal Name	Default Function	Default Termination	I/O Voltage Support	Native Function 1 (EDS)	Native Function 2 (EDS)	Native Function 3 (EDS)	Native Function 4 (EDS)	Native Function 5 (EDS)	Native Function 6 (EDS)	Native Function 7 (EDS)	Native Function 8 (EDS)	Pin State During Reset	Pin State Immediately After Reset	SCI/GPE IS	SMI	NMI	Output power sequence deglitch	GPIO Community #
GP_A00	GP_A00/PSE_GBE0_RGMII_TXD3	GP-In	None	1.8V/3.3V	PSE_GBE0_RGMII_TXD3								Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	3
GP_A01	GP_A01/PSE_GBE0_RGMII_TXD2	GP-In	None	1.8V/3.3V	PSE_GBE0_RGMII_TXD2								Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	3
GP_A10	GP_A10/PSE_GBE0_RGMII_RXD0	GP-In	None	1.8V/3.3V	PSE_GBE0_RGMII_RXD0								Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	3
GP_A11	GP_A11/PSE_GBE1_RGMII_TXD3	GP-In	None	1.8V/3.3V	PSE_GBE1_RGMII_TXD3								Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	3
GP_A12	GP_A12/PSE_GBE1_RGMII_TXD2	GP-In	None	1.8V/3.3V	PSE_GBE1_RGMII_TXD2								Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	3
GP_A13	GP_A13/PSE_GBE1_RGMII_TXD1	GP-In	None	1.8V/3.3V	PSE_GBE1_RGMII_TXD1								Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	3

Table 21-1. GPIO Multiplexing Table (Sheet 5 of 42)

GPIO (EDS)	EDS Signal Name	Default Function	Default Termination	I/O Voltage Support	Native Function 1 (EDS)	Native Function 2 (EDS)	Native Function 3 (EDS)	Native Function 4 (EDS)	Native Function 5 (EDS)	Native Function 6 (EDS)	Native Function 7 (EDS)	Native Function 8 (EDS)	Pin State During Reset	Pin State Immediately After Reset	SCI/GPE IS	SMI	NMI	Output power sequence deglitch	GPIO Community #
GP_A14	GP_A14/PSE_GBE1_RGMII_TXD0	GP-In	None	1.8V/3.3V	PSE_GBE1_RGMII_TXD0								Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	3
GP_A15	GP_A15/PSE_GBE1_RGMII_TXCLK	GP-In	None	1.8V/3.3V	PSE_GBE1_RGMII_TXCLK								Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	3
GP_A16	GP_A16/PSE_GBE1_RGMII_TXCTL	GP-In	None	1.8V/3.3V	PSE_GBE1_RGMII_TXCTL								Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	3
GP_A17	GP_A17/PSE_GBE1_RGMII_RXCLK	GP-In	None	1.8V/3.3V	PSE_GBE1_RGMII_RXCLK								Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	3
GP_A18	GP_A18/PSE_GBE1_RGMII_RXCTL	GP-In	None	1.8V/3.3V	PSE_GBE1_RGMII_RXCTL								Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	3
GP_A19	GP_A19/PSE_GBE1_RGMII_RXD3/AVS_I2S5_SCLK	GP-In	None	1.8V/3.3V	PSE_GBE1_RGMII_RXD3	AVS_I2S5_SCLK							Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	3

Table 21-1. GPIO Multiplexing Table (Sheet 6 of 42)

GPIO (EDS)	EDS Signal Name	Default Function	Default Termination	I/O Voltage Support	Native Function 1 (EDS)	Native Function 2 (EDS)	Native Function 3 (EDS)	Native Function 4 (EDS)	Native Function 5 (EDS)	Native Function 6 (EDS)	Native Function 7 (EDS)	Native Function 8 (EDS)	Pin State During Reset	Pin State Immediately After Reset	SCI/GPE IS	SMI	NMI	Output power sequence deglitch	GPIO Community #
GP_A02	GP_A02/PSE_GBE0_RGMII_TXD1	GP-In	None	1.8V/3.3V	PSE_GBE0_RGMII_TXD1								Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	3
GP_A20	GP_A20/PSE_GBE1_RGMII_RXD2/AVS_I2S5_SFRM	GP-In	None	1.8V/3.3V	PSE_GBE1_RGMII_RXD2	AVS_I2S5_SFRM							Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	3
GP_A21	GP_A21/PSE_GBE1_RGMII_RXD1/AVS_I2S5_TXD	GP-In	None	1.8V/3.3V	PSE_GBE1_RGMII_RXD1	AVS_I2S5_TXD							Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	3
GP_A22	GP_A22/PSE_GBE1_RGMII_RXD0/AVS_I2S5_RXD	GP-In	None	1.8V/3.3V	PSE_GBE1_RGMII_RXD0	AVS_I2S5_RXD							Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	3
GP_A23	GP_A23/PSE_GBE0_RGMII_RXCTL	GP-In	None	1.8V/3.3V	PSE_GBE0_RGMII_RXCTL								Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	3
GP_A03	GP_A03/PSE_GBE0_RGMII_TXD0	GP-In	None	1.8V/3.3V	PSE_GBE0_RGMII_TXD0								Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	3
GP_A04	GP_A04/PSE_GBE0_RGMII_TXCLK	GP-In	None	1.8V/3.3V	PSE_GBE0_RGMII_TXCLK								Z	Z	Yes	No	No	Hi-Z output with internal 20K PD during power sequencing	3

Table 21-1. GPIO Multiplexing Table (Sheet 7 of 42)

GPIO (EDS)	EDS Signal Name	Default Function	Default Termination	I/O Voltage Support	Native Function 1 (EDS)	Native Function 2 (EDS)	Native Function 3 (EDS)	Native Function 4 (EDS)	Native Function 5 (EDS)	Native Function 6 (EDS)	Native Function 7 (EDS)	Native Function 8 (EDS)	Pin State During Reset	Pin State Immediately After Reset	SCI/GPE IS	SMI	NMI	Output power sequence deglitch	GPIO Community #
GP_A05	GP_A05/PSE_GBE0_RGMII_TXCTL	GP-In	None	1.8V/3.3V	PSE_GBE0_RGMII_TXCTL								Z	Z	Yes	No	No	Hi-Z output with internal 20K PD during power sequencing	3
GP_A06	GP_A06/PSE_GBE0_RGMII_RXCLK	GP-In	None	1.8V/3.3V	PSE_GBE0_RGMII_RXCLK								Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	3
GP_A07	GP_A07/PSE_GBE0_RGMII_RXD3	GP-In	None	1.8V/3.3V	PSE_GBE0_RGMII_RXD3								Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	3
GP_A08	GP_A08/PSE_GBE0_RGMII_RXD2	GP-In	None	1.8V/3.3V	PSE_GBE0_RGMII_RXD2								Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	3
GP_A09	GP_A09/PSE_GBE0_RGMII_RXD1	GP-In	None	1.8V/3.3V	PSE_GBE0_RGMII_RXD1								Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	3
GP_B00	GP_B00/PMC_CORE_VID0	Native F1	None	1.8V/3.3V	PMC_CORE_VID0								Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0
GP_B01	GP_B01/PMC_CORE_VID1	Native F1	None	1.8V/3.3V	PMC_CORE_VID1								Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0

Table 21-1. GPIO Multiplexing Table (Sheet 8 of 42)

GPIO (EDS)	EDS Signal Name	Default Function	Default Termination	I/O Voltage Support	Native Function 1 (EDS)	Native Function 2 (EDS)	Native Function 3 (EDS)	Native Function 4 (EDS)	Native Function 5 (EDS)	Native Function 6 (EDS)	Native Function 7 (EDS)	Native Function 8 (EDS)	Pin State During Reset	Pin State Immediately After Reset	SCI/GPE IS	SMI	NMI	Output power sequence deglitch	GPIO Community #
GP_B10	GP_B10/ SIO_I2C5_SCL/ PSE_I2C2_SCL/ ESPI_ALERT3_N	Native F4/GP-In	None (eSPI Disabled) 20K PU (eSPI Enabled)	1.8V/ 3.3V	SIO_I2C5_SCL	PSE_I2C2_SCL		ESPI_ALERT3_N					Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0
GP_B11	GP_B11/ PMC_ALERT_N/ PSE_TGPIO06	GP-In	None	1.8V/ 3.3V	PMC_ALERT_N							PSE_TGPIO06	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0
GP_B12	GP_B12/ PMC_SLP_S0_N	Native F1	None	1.8V/ 3.3V	PMC_SLP_S0_N								Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0
GP_B13	GP_B13/ PMC_PLTRST_N	Native F1	None	1.8V/ 3.3V	PMC_PLTRST_N								Z	Z	Yes	No	No	Hi-Z output with internal 20K PD during power sequencing	0
GP_B14	GP_B14/SPKR/ PMC_TGPIO1/ SIO_S-PI0_CS1_N/ PSE_S-PI2_CS1_N	GP-Out	None	1.8V/ 3.3V	SPKR	PMC_TGPIO1	SIO_SPI0_CS1_N	PSE_S-PI2_CS1_N					Z	Z	Yes	Yes	Yes	None	0
GP_B15	GP_B15/SIO_S-PI0_CS0_N/ PSE_S-PI2_CS0_N/ ESPI_CS1_N	Native F5/GP-In	None (eSPI Disabled) 20K PU (eSPI Enabled)	1.8V/ 3.3V	SIO_S-PI0_CS0_N		PSE_SPI2_CS0_N		ESPI_CS1_N				Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0

Table 21-1. GPIO Multiplexing Table (Sheet 9 of 42)

GPIO (EDS)	EDS Signal Name	Default Function	Default Termination	I/O Voltage Support	Native Function 1 (EDS)	Native Function 2 (EDS)	Native Function 3 (EDS)	Native Function 4 (EDS)	Native Function 5 (EDS)	Native Function 6 (EDS)	Native Function 7 (EDS)	Native Function 8 (EDS)	Pin State During Reset	Pin State Immediately After Reset	SCI/GPE IS	SMI	NMI	Output power sequence deglitch	GPIO Community #
GP_B16	GP_B16/SIO_SPI0_CLK/PSE_SPI2_CLK	GP-In	None	1.8V/3.3V	SIO_SPI0_CLK		PSE_SPI2_CLK						Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0
GP_B17	GP_B17/SIO_SPI0_MISO/PSE_SPI2_MISO	GP-In	None	1.8V/3.3V	SIO_SPI0_MISO		PSE_SPI2_MISO						Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0
GP_B18	GP_B18/SIO_SPI0_MOSI/PSE_SPI2_MOSI	GP-Out	None	1.8V/3.3V	SIO_SPI0_MOSI		PSE_SPI2_MOSI						Z	Z	Yes	No	No	None	0
GP_B19	GP_B19/SIO_SPI1_CS0_N/PSE_SPI3_CS0_N/ESPI_CS2_N	Native F5/GP-In	None (eSPI Disabled) 20K PU (eSPI Enabled)	1.8V/3.3V	SIO_SPI1_CS0_N		PSE_SPI3_CS0_N		ESPI_CS2_N				Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0
GP_B02	GP_B02/PMC_VRALERT_N/ESPI_ALERT2_N/PSE_TGPIO25	Native F4/GP-In	None (eSPI Disabled) 20K PU (eSPI Enabled)	1.8V/3.3V	PMC_VRALERT_N			ESPI_ALERT2_N				PSE_TGPIO25	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0
GP_B20	GP_B20/SIO_SPI1_CLK/PSE_SPI3_CLK	GP-In	None	1.8V/3.3V	SIO_SPI1_CLK	RSVD	PSE_SPI3_CLK						Z	Z	Yes	Yes	Yes	Hi-Z output with no internal termination during power sequencing	0
GP_B21	GP_B21/SIO_SPI1_MISO/PSE_SPI3_MISO	GP-In	None	1.8V/3.3V	SIO_SPI1_MISO	RSVD	PSE_SPI3_MISO						Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0

Table 21-1. GPIO Multiplexing Table (Sheet 10 of 42)

GPIO (EDS)	EDS Signal Name	Default Function	Default Termination	I/O Voltage Support	Native Function 1 (EDS)	Native Function 2 (EDS)	Native Function 3 (EDS)	Native Function 4 (EDS)	Native Function 5 (EDS)	Native Function 6 (EDS)	Native Function 7 (EDS)	Native Function 8 (EDS)	Pin State During Reset	Pin State Immediately After Reset	SCI/GPE IS	SMI	NMI	Output power sequence deglitch	GPIO Community #
GP_B22	GP_B22/SIO_SPI1_MOSI/ PSE_SPI3_MOSI	GP-Out	None	1.8V/ 3.3V	SIO_SPI1_MOSI		PSE_SPI3_MOSI						Z	Z	Yes	No	No	None	0
GP_B23	GP_B23/PCH-HOT_N/SIO_SPI1_CS1_N/ PSE_SPI3_CS1_N/ PSE_TGPIO28	GP-Out	None	1.8V/ 3.3V		PCH-HOT_N	SIO_SPI1_CS1_N	PSE_SPI3_CS1_N				PSE_TGPIO28	Z	Z	Yes	Yes	Yes	None	0
GP_B03	GP_B03/ CPU_GP2/ ESPI_ALERT0_N/ PSE_TGPIO26	NativeF4/GP-In	None (eSPI Disabled) 20K PU (eSPI Enabled)	1.8V/ 3.3V	CPU_GP2			ESPI_ALERT0_N				PSE_TGPIO26	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0
GP_B04	GP_B04/ CPU_GP3/ ESPI_ALERT1_N/ PSE_TGPIO27	NativeF4/GP-In	None (eSPI Disabled) 20K PU (eSPI Enabled)	1.8V/ 3.3V	CPU_GP3			ESPI_ALERT1_N				PSE_TGPIO27	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0
GP_B05	GP_B05/ PSE_I2C0_SCL/ PSE_TGPIO06	GP-In	None	1.8V/ 3.3V	PSE_I2C0_SCL							PSE_TGPIO06	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0
GP_B06	GP_B06/ PSE_I2C0_SDA/ PSE_TGPIO07	GP-In	None	1.8V/ 3.3V	PSE_I2C0_SDA							PSE_TGPIO07	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0
GP_B07	GP_B07/ PSE_I2C1_SCL/ PSE_TGPIO08	GP-In	None	1.8V/ 3.3V	PSE_I2C1_SCL							PSE_TGPIO08	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0

Table 21-1. GPIO Multiplexing Table (Sheet 11 of 42)

GPIO (EDS)	EDS Signal Name	Default Function	Default Termination	I/O Voltage Support	Native Function 1 (EDS)	Native Function 2 (EDS)	Native Function 3 (EDS)	Native Function 4 (EDS)	Native Function 5 (EDS)	Native Function 6 (EDS)	Native Function 7 (EDS)	Native Function 8 (EDS)	Pin State During Reset	Pin State Immediately After Reset	SCI/GPE IS	SMI	NMI	Output power sequence deglitch	GPIO Community #
GP_B08	GP_B08/ PSE_I2C1_SDA/ PSE_TGPIO09	GP-In	None	1.8V/ 3.3V	PSE_I2C1_SDA							PSE_TGPIO09	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0
GP_B09	GP_B09/ SIO_I2C5_SDA/ PSE_I2C2_SDA/ ESPI_CS3_N	Native F4/GP-In	None (eSPI Disabled) 20K PU (eSPI Enabled)	1.8V/ 3.3V	SIO_I2C5_SDA	PSE_I2C2_SDA		ESPI_CS3_N					Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0
GP_C00	GP_C00/SMB_-CLK/ PSE_I2C3_SCL/ PSE_TGPIO18	Native F1	None	1.8V/ 3.3V	SMB_-CLK	PSE_I2C3_SCL						PSE_TGPIO18	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4
GP_C01	GP_C01/SMB_-DATA/ PSE_I2C3_SDA/ PSE_TGPIO19	Native F1	None	1.8V/ 3.3V	SMB_-DATA	PSE_I2C3_SDA						PSE_TGPIO19	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4
GP_C10	GP_C10/ PSE_TGPIO05	GP-In	None	1.8V/ 3.3V	PSE_TGPIO05			RSVD					Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4
GP_C11	GP_C11/PSE_H-SUART0_RE	GP-In	None	1.8V/ 3.3V	PSE_H-SUART0_RE			RSVD					Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4

Table 21-1. GPIO Multiplexing Table (Sheet 12 of 42)

GPIO (EDS)	EDS Signal Name	Default Function	Default Termination	I/O Voltage Support	Native Function 1 (EDS)	Native Function 2 (EDS)	Native Function 3 (EDS)	Native Function 4 (EDS)	Native Function 5 (EDS)	Native Function 6 (EDS)	Native Function 7 (EDS)	Native Function 8 (EDS)	Pin State During Reset	Pin State Immediately After Reset	SCI/GPE IS	SMI	NMI	Output power sequence deglitch	GPIO Community #
GP_C12	GP_C12/ PSE_UART0_RXD/ SIO_UART1_RXD	GP-In	None	1.8V/ 3.3V	PSE_UART0_RXD			SIO_UART1_RXD					Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4
GP_C13	GP_C13/ PSE_UART0_TXD/ SIO_UART1_TXD	GP-In	None	1.8V/ 3.3V	PSE_UART0_TXD			SIO_UART1_TXD					Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4
GP_C14	GP_C14/ PSE_UART0_RTS_N/PSE_HSUART0_DE/ SIO_UART1_RTS_N	GP-In	None	1.8V/ 3.3V	PSE_UART0_RTS_N	PSE_HSUART0_DE		SIO_UART1_RTS_N					Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4
GP_C15	GP_C15/ PSE_UART0_CTS_N/ SIO_UART1_CTS_N	GP-In	None	1.8V/ 3.3V	PSE_UART0_CTS_N			SIO_UART1_CTS_N					Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4
GP_C16	GP_C16/GBE_MDIO/ PSE_UART3_RXD/ SIO_I2C0_SDA	GP-In	None	1.8V/ 3.3V	GBE_MDIO		PSE_UART3_RXD	SIO_I2C0_SDA					Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4
GP_C17	GP_C17/ GBE_MDC/ PSE_UART3_TXD/ SIO_I2C0_SCL	GP-In	None	1.8V/ 3.3V	GBE_MDC		PSE_UART3_TXD	SIO_I2C0_SCL					Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4

Table 21-1. GPIO Multiplexing Table (Sheet 13 of 42)

GPIO (EDS)	EDS Signal Name	Default Function	Default Termination	I/O Voltage Support	Native Function 1 (EDS)	Native Function 2 (EDS)	Native Function 3 (EDS)	Native Function 4 (EDS)	Native Function 5 (EDS)	Native Function 6 (EDS)	Native Function 7 (EDS)	Native Function 8 (EDS)	Pin State During Reset	Pin State Immediately After Reset	SCI/GPE IS	SMI	NMI	Output power sequence deglitch	GPIO Community #
GP_C18	GP_C18/ PSE_I2C4_SDA/ SML_DATA0/ SIO_I2C1_SDA	GP-In	None	1.8V/ 3.3V	PSE_I2C4_SDA		SML_-DATA0	SIO_I2C1_SDA					Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4
GP_C19	GP_C19/ PSE_I2C4_SCL/ SML_CLK0/ SIO_I2C1_SCL	GP-In	None	1.8V/ 3.3V	PSE_I2C4_SCL		SML_-CLK0	SIO_I2C1_SCL					Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4
GP_C02	GP_C02/PSE_P-WM00/ SMB_ALERT_N/ PSE_TGPIO29	Native F2/GP-Out	None	1.8V/ 3.3V	PSE_P-WM00	SMB_ALERT_N						PSE_TGPIO29	Z	Z	Yes	No	No	None	4
GP_C20	GP_C20/ PSE_UART4_RXD/ SIO_UART2_RXD	GP-In	None	1.8V/ 3.3V	PSE_UART4_RXD		RSVD	SIO_UART2_RXD					Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4
GP_C21	GP_C21/ PSE_UART4_TXD/ SIO_UART2_TXD	GP-In	None	1.8V/ 3.3V	PSE_UART4_TXD		RSVD	SIO_UART2_TXD					Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4
GP_C22	GP_C22/ PSE_UART4_RTS_N/ISI_SPIM_-MOSI/ SIO_UART2_RTS_N	GP-In	None	1.8V/ 3.3V	PSE_UART4_RTS_N	ISI_SPIM_-MOSI	RSVD	SIO_UART2_RTS_N					Z	Z	Yes	Yes	Yes	Hi-Z output with no internal termination during power sequencing	4
GP_C23	GP_C23/ PSE_UART4_CTS_N/ISI_SPIS_-MOSI/ SIO_UART2_CTS_N	GP-In	None	1.8V/ 3.3V	PSE_UART4_CTS_N	ISI_SPIS_-MOSI	RSVD	SIO_UART2_CTS_N					Z	Z	Yes	Yes	Yes	Hi-Z output with no internal termination during power sequencing	4

Table 21-1. GPIO Multiplexing Table (Sheet 14 of 42)

GPIO (EDS)	EDS Signal Name	Default Function	Default Termination	I/O Voltage Support	Native Function 1 (EDS)	Native Function 2 (EDS)	Native Function 3 (EDS)	Native Function 4 (EDS)	Native Function 5 (EDS)	Native Function 6 (EDS)	Native Function 7 (EDS)	Native Function 8 (EDS)	Pin State During Reset	Pin State Immediately After Reset	SCI/GPE IS	SMI	NMI	Output power sequence deglitch	GPIO Community #
GP_C03	GP_C03/PSE_GBE0_MDC/ PSE_HSUART3_EN	GP-In	None	1.8V/ 3.3V	PSE_GBE0_MDC		PSE_HSUART3_EN			RSVD			Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4
GP_C04	GP_C04/PSE_GBE0_MDIO/ PSE_UART3_RTS_N/PSE_HSUART3_DE	GP-In	None	1.8V/ 3.3V	PSE_GBE0_MDIO		PSE_UART3_RTS_N	PSE_HSUART3_DE		RSVD			Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4
GP_C05	GP_C05/PSE_PWM01/ PSE_UART3_CTS_N/ SML_ALERT0_N/ PSE_TGPIO30	GP-Out	None	1.8V/ 3.3V	PSE_PWM01		PSE_UART3_CTS_N			SML_ALERT0_N		PSE_TGPIO30	Z	Z	Yes	No	No	None	4
GP_C06	GP_C06/PSE_GBE1_MDC	GP-In	None	1.8V/ 3.3V	PSE_GBE1_MDC								Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4
GP_C07	GP_C07/PSE_GBE1_MDIO/ PSE_HSUART3_RE	GP-In	None	1.8V/ 3.3V	PSE_GBE1_MDIO		PSE_HSUART3_RE						Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4
GP_C08	GP_C08/ PSE_TGPIO04/ DNX_FORCE_RELOAD	NativeF2	20K PD	1.8V/ 3.3V	PSE_TGPIO04	DNX_FORCE_RELOAD		RSVD					Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4

Table 21-1. GPIO Multiplexing Table (Sheet 15 of 42)

GPIO (EDS)	EDS Signal Name	Default Function	Default Termination	I/O Voltage Support	Native Function 1 (EDS)	Native Function 2 (EDS)	Native Function 3 (EDS)	Native Function 4 (EDS)	Native Function 5 (EDS)	Native Function 6 (EDS)	Native Function 7 (EDS)	Native Function 8 (EDS)	Pin State During Reset	Pin State Immediately After Reset	SCI/GPE IS	SMI	NMI	Output power sequence deglitch	GPIO Community #
GP_C09	GP_C09/PSE_HSUART0_EN	GP-In	None	1.8V/3.3V	PSE_HSUART0_EN			RSVD					Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4
GP_D00	GP_D00/PSE_QEPA0/PSE_SPI1_CS0_N/PSE_TGPIO32	GP-In	None	1.8V/3.3V	PSE_QEPA0		RSVD	PSE_SPI1_CS0_N				PSE_TGPIO32	Z	Z	Yes	Yes	Yes	Hi-Z output with no internal termination during power sequencing	1
GP_D01	GP_D01/PSE_QEPB0/PSE_SPI1_CLK/PSE_TGPIO33	GP-In	None	1.8V/3.3V	PSE_QEPB0		RSVD	PSE_SPI1_CLK				PSE_TGPIO33	Z	Z	Yes	Yes	Yes	Hi-Z output with no internal termination during power sequencing	1
GP_D10	GP_D10/PSE_SPI0_CLK/SIO_SPI2_CLK/PSE_TGPIO11	GP-In	None	1.8V/3.3V	PSE_SPI0_CLK						SIO_SPI2_CLK	PSE_TGPIO11	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_D11	GP_D11/PSE_SPI0_MISO/SIO_SPI2_MISO/PSE_TGPIO12	GP-In	None	1.8V/3.3V	PSE_SPI0_MISO						SIO_SPI2_MISO	PSE_TGPIO12	Z	Z	Yes	No	No	None	1
GP_D12	GP_D12/PSE_SPI0_MOSI/SIO_SPI2_MOSI/PSE_TGPIO13	GP-In	None	1.8V/3.3V	PSE_SPI0_MOSI						SIO_SPI2_MOSI	PSE_TGPIO13	Z	Z	Yes	No	No	None	1
GP_D13	GP_D13/PSE_QEPA1/PSE_TGPIO37	GP-In	None	1.8V/3.3V	PSE_QEPA1		RSVD					PSE_TGPIO37	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1

Table 21-1. GPIO Multiplexing Table (Sheet 16 of 42)

GPIO (EDS)	EDS Signal Name	Default Function	Default Termination	I/O Voltage Support	Native Function 1 (EDS)	Native Function 2 (EDS)	Native Function 3 (EDS)	Native Function 4 (EDS)	Native Function 5 (EDS)	Native Function 6 (EDS)	Native Function 7 (EDS)	Native Function 8 (EDS)	Pin State During Reset	Pin State Immediately After Reset	SCI/GPE IS	SMI	NMI	Output power sequence deglitch	GPIO Community #
GP_D14	GP_D14/ PSE_QEPB1/ PSE_TGPIO38	GP-In	None	1.8V/ 3.3V	PSE_QEPB1		RSVD					PSE_TGPIO38	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_D15	GP_D15/PSE_PWM03/SIO_SPI2_CS1_N/ PSE_SPI0_CS1_N/ PSE_TGPIO39	GP-In	None	1.8V/ 3.3V	PSE_PWM03	SIO_SPI2_CS1_N	RSVD	PSE_SPI0_CS1_N				PSE_TGPIO39	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_D16	GP_D16/ PSE_QEPI1/ PSE_TGPIO40	GP-In	None	1.8V/ 3.3V	PSE_QEPI1							PSE_TGPIO40	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_D17	GP_D17/PSE_PWM04/ ISI_SPIM_MOSI/ PSE_TGPIO41	GP-In	None	1.8V/ 3.3V	PSE_PWM04	ISI_SPIM_MOSI						PSE_TGPIO41	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_D18	GP_D18/PSE_PWM05/ ISI_SPIS_MOSI/ PSE_TGPIO42	GP-In	None	1.8V/ 3.3V	PSE_PWM05	ISI_SPIS_MOSI						PSE_TGPIO42	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_D19	GP_D19/ AVS_I2S_MCLK1/PSE_TGPIO43	GP-In	None	1.8V/ 3.3V	AVS_I2S_MCLK1							PSE_TGPIO43	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1

Table 21-1. GPIO Multiplexing Table (Sheet 17 of 42)

GPIO (EDS)	EDS Signal Name	Default Function	Default Termination	I/O Voltage Support	Native Function 1 (EDS)	Native Function 2 (EDS)	Native Function 3 (EDS)	Native Function 4 (EDS)	Native Function 5 (EDS)	Native Function 6 (EDS)	Native Function 7 (EDS)	Native Function 8 (EDS)	Pin State During Reset	Pin State Immediately After Reset	SCI/GPE IS	SMI	NMI	Output power sequence deglitch	GPIO Community #
GP_D02	GP_D02/ PSE_QEPI0/ PSE_SPI1_MISO/ PSE_TGPIO34	GP-In	None	1.8V/ 3.3V	PSE_QEPI0		RSVD	PSE_SPI1_MISO				PSE_TGPIO34	Z	Z	Yes	Yes	Yes	Hi-Z output with no internal termination during power sequencing	1
GP_D03	GP_D03/PSE_PWM06/PSE_SPI1_MOSI/ PSE_TGPIO35	GP-In	None	1.8V/ 3.3V	PSE_PWM06		RSVD	PSE_SPI1_MOSI				PSE_TGPIO35	Z	Z	Yes	Yes	Yes	Hi-Z output with no internal termination during power sequencing	1
GP_D04	GP_D04/PSE_PWM02/PSE_SPI1_CS1_N/ PSE_TGPIO36	GP-In	None	1.8V/ 3.3V	PSE_PWM02		RSVD	PSE_SPI1_CS1_N				PSE_TGPIO36	Z	Z	Yes	Yes	Yes	Hi-Z output with no internal termination during power sequencing	1
GP_D05	GP_D05/PCIE_CLKREQ0_N	GP-In	None	1.8V/ 3.3V	PCIE_CLKREQ0_N								Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_D06	GP_D06/PCIE_CLKREQ1_N	GP-In	None	1.8V/ 3.3V	PCIE_CLKREQ1_N								Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_D07	GP_D07/PCIE_CLKREQ2_N	GP-In	None	1.8V/ 3.3V	PCIE_CLKREQ2_N								Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1

Table 21-1. GPIO Multiplexing Table (Sheet 18 of 42)

GPIO (EDS)	EDS Signal Name	Default Function	Default Termination	I/O Voltage Support	Native Function 1 (EDS)	Native Function 2 (EDS)	Native Function 3 (EDS)	Native Function 4 (EDS)	Native Function 5 (EDS)	Native Function 6 (EDS)	Native Function 7 (EDS)	Native Function 8 (EDS)	Pin State During Reset	Pin State Immediately After Reset	SCI/GPE IS	SMI	NMI	Output power sequence deglitch	GPIO Community #
GP_D08	GP_D08/PCIE_CLKREQ3_N	GP-In	None	1.8V/3.3V	PCIE_CLKREQ3_N								Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_D09	GP_D09/PSE_SPI0_CS0_N/ SIO_SPI2_CS0_N/ PSE_TGPIO10	GP-In	None	1.8V/3.3V	PSE_SPI0_CS0_N						SIO_SPI2_CS0_N	PSE_TGPIO10	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_E00	GP_E00/ SATA_LED_N/ SATAXPCIE_0/ SATA_0_GP	Native F2/GP-In	None (SATAXPCIE Disabled) 20K PU (SATAXPCIE Enabled)	1.8V/3.3V	SATA_LED_N	SATAXPCIE_0	RSVD	RSVD	RSVD	RSVD	SATA_0_GP		Z	Z	Yes	Yes	Yes	Hi-Z output with no internal termination during power sequencing	4
GP_E01	GP_E01	GP-In	None	1.8V/3.3V		RSVD	RSVD	RSVD	RSVD	RSVD	RSVD		Z	Z	Yes	Yes	Yes	Hi-Z output with no internal termination during power sequencing	4
GP_E10	GP_E10	GP-In	None	1.8V/3.3V		RSVD	RSVD	RSVD		RSVD	RSVD		Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4
GP_E11	GP_E11	GP-In	None	1.8V/3.3V		RSVD	RSVD	RSVD		RSVD	RSVD		Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4

Table 21-1. GPIO Multiplexing Table (Sheet 19 of 42)

GPIO (EDS)	EDS Signal Name	Default Function	Default Termination	I/O Voltage Support	Native Function 1 (EDS)	Native Function 2 (EDS)	Native Function 3 (EDS)	Native Function 4 (EDS)	Native Function 5 (EDS)	Native Function 6 (EDS)	Native Function 7 (EDS)	Native Function 8 (EDS)	Pin State During Reset	Pin State Immediately After Reset	SCI/GPE IS	SMI	NMI	Output power sequence deglitch	GPIO Community #
GP_E12	GP_E12	GP-In	None	1.8V/3.3V		RSVD	RSVD	RSVD		RSVD	RSVD		Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4
GP_E13	GP_E13	GP-In	None	1.8V/3.3V		RSVD	RSVD	RSVD		RSVD	RSVD		Z	Z	Yes	Yes	Yes	Hi-Z output with no internal termination during power sequencing	4
GP_E14	GP_E14/ DDI0_HPD/ PNL_MISC_DDI0/PSE_TGPIO19	GP-In	None	1.8V/3.3V	DDI0_HPD	PNL_MISC_DDI0	RSVD	RSVD		RSVD		PSE_TGPIO19	Z	Z	Yes	Yes	Yes	Hi-Z output with no internal termination during power sequencing	4
GP_E15	GP_E15/ PSE_I2S0_RXD/ PSE_CAN0_TX/ PSE_TGPIO17	Native F2	None	1.8V/3.3V	PSE_I2S0_RXD	RSVD	RSVD	RSVD		RSVD	PSE_CAN0_TX	PSE_TGPIO17	Z	Z	Yes	Yes	Yes	Hi-Z output with no internal termination during power sequencing	4
GP_E16	GP_E16/ PSE_I2S0_TXD/ PSE_CAN0_RX/ PSE_TGPIO16	Native F2	None	1.8V/3.3V	PSE_I2S0_TXD	RSVD	RSVD	RSVD	RSVD	RSVD	PSE_CAN0_RX	PSE_TGPIO16	Z	Z	Yes	Yes	Yes	Hi-Z output with no internal termination during power sequencing	4
GP_E17	GP_E17/PNL1_VDDEN/PNL_MISC_DDI2/ PSE_PWM11/ PSE_TGPIO46	GP-In	None	1.8V/3.3V	PNL1_VDDEN	RSVD	RSVD	RSVD	PNL_MISC_DDI2	RSVD	PSE_PWM11	PSE_TGPIO46	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4

Table 21-1. GPIO Multiplexing Table (Sheet 20 of 42)

GPIO (EDS)	EDS Signal Name	Default Function	Default Termination	I/O Voltage Support	Native Function 1 (EDS)	Native Function 2 (EDS)	Native Function 3 (EDS)	Native Function 4 (EDS)	Native Function 5 (EDS)	Native Function 6 (EDS)	Native Function 7 (EDS)	Native Function 8 (EDS)	Pin State During Reset	Pin State Immediately After Reset	SCI/GPE IS	SMI	NMI	Output power sequence deglitch	GPIO Community #
GP_E18	GP_E18/ DDI0_DDC_SDA/PSE_PWM12/ PSE_TGPIO23	Native F5	Native	1.8V/ 3.3V	DDI0_DDC_SDA				RSVD		PSE_PWM12	PSE_TGPIO23	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4
GP_E19	GP_E19/ DDI0_DDC_SCL/ PSE_PWM13/ PSE_TGPIO24	Native F5	Native	1.8V/ 3.3V	DDI0_DDC_SCL				RSVD		PSE_PWM13	PSE_TGPIO24	Z	Z	Yes	No	No	None	4
GP_E02	GP_E02	GP-In	None	1.8V/ 3.3V		RSVD	RSVD	RSVD	RSVD	RSVD	RSVD		Z	Z	Yes	Yes	Yes	Hi-Z output with no internal termination during power sequencing	4
GP_E20	GP_E20/ PSE_I2S0_SCLK/ PSE_CAN1_TX/ PSE_TGPIO14	GP-In	None	1.8V/ 3.3V	PSE_I2S0_SCLK						PSE_CAN1_TX	PSE_TGPIO14	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4
GP_E21	GP_E21/ PSE_I2S0_SFRM/PSE_CAN1_RX/ PSE_TGPIO15	GP-In	None	1.8V/ 3.3V	PSE_I2S0_SFRM						PSE_CAN1_RX	PSE_TGPIO15	Z	Z	Yes	No	No	None	4
GP_E22	GP_E22/PNL1_BKLTCTL/PSE_PWM14/ PSE_TGPIO18	GP-In	None	1.8V/ 3.3V	PNL1_BKLTCTL						PSE_PWM14	PSE_TGPIO18	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4
GP_E23	GP_E23/PNL1_BKLTEN/PSE_PWM15/ PSE_TGPIO19	GP-Out	None	1.8V/ 3.3V	PNL1_BKLTEN						PSE_PWM15	PSE_TGPIO19	Z	Z	Yes	No	No	Hi-Z output with internal 20K PD during power sequencing	4

Table 21-1. GPIO Multiplexing Table (Sheet 21 of 42)

GPIO (EDS)	EDS Signal Name	Default Function	Default Termination	I/O Voltage Support	Native Function 1 (EDS)	Native Function 2 (EDS)	Native Function 3 (EDS)	Native Function 4 (EDS)	Native Function 5 (EDS)	Native Function 6 (EDS)	Native Function 7 (EDS)	Native Function 8 (EDS)	Pin State During Reset	Pin State Immediately After Reset	SCI/GPE IS	SMI	NMI	Output power sequence deglitch	GPIO Community #
GP_E03	GP_E03/ DDI1_HPD/ PNL_MISC_DDI1/CPU_GP0/ PSE_TGPIO15	GP-In	None	1.8V/ 3.3V	DDI1_HPD	PNL_MISC_DDI1	RSVD	RSVD	RSVD	RSVD	CPU_GP0	PSE_TGPIO15	Z	Z	Yes	Yes	Yes	Hi-Z output with no internal termination during power sequencing	4
GP_E04	GP_E04/ SATA_0_DEVSLP /PSE_PWM08/ PSE_TGPIO44	GP-In	None	1.8V/ 3.3V	SATA_0_DEVSLP		RSVD	RSVD	RSVD	RSVD	PSE_PWM08	PSE_TGPIO44	Z	Z	Yes	Yes	Yes	Hi-Z output with no internal termination during power sequencing	4
GP_E05	GP_E05/ DDI1_DDC_SDA/PSE_PWM09/ PSE_TGPIO17	GP-In	None	1.8V/ 3.3V	DDI1_DDC_SDA		RSVD	RSVD	RSVD	RSVD	PSE_PWM09	PSE_TGPIO17	Z	Z	Yes	Yes	Yes	Hi-Z output with no internal termination during power sequencing	4
GP_E06	GP_E06/PSE_PWM10/PSE_TGPIO18	GP-Out	None	1.8V/ 3.3V		RSVD	RSVD	RSVD	RSVD	RSVD	PSE_PWM10	PSE_TGPIO18	Z	Z	Yes	Yes	Yes	Hi-Z output with no internal termination during power sequencing	4
GP_E07	GP_E07/ DDI1_DDC_SCL/ CPU_GP1/ PSE_TGPIO16	GP-In	None	1.8V/ 3.3V	DDI1_DDC_SCL		RSVD	RSVD	RSVD	RSVD	CPU_GP1	PSE_TGPIO16	Z	Z	Yes	Yes	Yes	Hi-Z output with no internal termination during power sequencing	4
GP_E08	GP_E08/ SATA_1_DEVSLP /PSE_TGPIO45	GP-In	None	1.8V/ 3.3V		SATA_1_DEVSLP	RSVD	RSVD		RSVD	RSVD	PSE_TGPIO45	Z	Z	Yes	Yes	Yes	Hi-Z output with no internal termination during power sequencing	4

Table 21-1. GPIO Multiplexing Table (Sheet 22 of 42)

GPIO (EDS)	EDS Signal Name	Default Function	Default Termination	I/O Voltage Support	Native Function 1 (EDS)	Native Function 2 (EDS)	Native Function 3 (EDS)	Native Function 4 (EDS)	Native Function 5 (EDS)	Native Function 6 (EDS)	Native Function 7 (EDS)	Native Function 8 (EDS)	Pin State During Reset	Pin State Immediately After Reset	SCI/GPE IS	SMI	NMI	Output power sequence deglitch	GPIO Community #
GP_E09	GP_E09/ USB2_OC0_N	GP-In	None	1.8V/ 3.3V	USB2_OC0_N		RSVD	RSVD		RSVD			Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4
GP_F00	GP_F00/ SIO_UART0_RTS_N	Native F1	None	1.8V/ 3.3V	RSVD	SIO_UART0_RTS_N							Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4
GP_F01	GP_F01/ SIO_UART0_RXD	Native F1	20K PU	1.8V/ 3.3V	RSVD	SIO_UART0_RXD							Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4
GP_F10	GP_F10/ PSE_I2S1_SFRM/ AVS_I2S4_SFRM/PSE_TGPIO15	GP-Out	None	1.8V/ 3.3V	PSE_I2S1_SFRM				AVS_I2S4_SFRM			PSE_TGPIO15	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4
GP_F11	GP_F11/ PSE_TRACECLK/ ISI_TRACECLK/ PSE_TGPIO49	Native F5	None	1.8V/ 3.3V			RSVD	PSE_TRACECLK	ISI_TRACECLK			PSE_TGPIO49	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4
GP_F12	GP_F12/ AVS_I2S4_TXD/ PSE_TRACESWO/ISI_TRACESWO	Native F5	None	1.8V/ 3.3V	RSVD	AVS_I2S4_TXD	RSVD	PSE_TRACESWO	ISI_TRACESWO				Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4

Table 21-1. GPIO Multiplexing Table (Sheet 23 of 42)

GPIO (EDS)	EDS Signal Name	Default Function	Default Termination	I/O Voltage Support	Native Function 1 (EDS)	Native Function 2 (EDS)	Native Function 3 (EDS)	Native Function 4 (EDS)	Native Function 5 (EDS)	Native Function 6 (EDS)	Native Function 7 (EDS)	Native Function 8 (EDS)	Pin State During Reset	Pin State Immediately After Reset	SCI/GPE IS	SMI	NMI	Output power sequence deglitch	GPIO Community #
GP_F13	GP_F13/ AVS_I2S4_SFRM/ PSE_SWDIO/ ISI_SWDIO	Native F5	None	1.8V/ 3.3V	RSVD	AVS_I2S4_SFRM	RSVD	PSE_SWDIO	ISI_SWDIO				Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4
GP_F14	GP_F14/ AVS_I2S4_RXD/ PSE_TRACEDATA_1/ ISI_TRACEDATA_1	Native F5	None	1.8V/ 3.3V	RSVD	AVS_I2S4_RXD	RSVD	PSE_TRACEDATA_1	ISI_TRACEDATA_1				Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4
GP_F15	GP_F15/ PSE_TRACEDATA_2/ ISI_TRACEDATA_2	Native F5	None	1.8V/ 3.3V	RSVD		RSVD	PSE_TRACEDATA_2	ISI_TRACEDATA_2				Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4
GP_F16	GP_F16/ AVS_I2S4_SCLK/ PSE_SWCLK/ ISI_SWCLK	Native F5	None	1.8V/ 3.3V	RSVD	AVS_I2S4_SCLK	RSVD	PSE_SWCLK	ISI_SWCLK				Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4
GP_F17	GP_F17/ PSE_TRACEDATA_3/ ISI_TRACEDATA_3/ PSE_TGPIO50	Native F5	None	1.8V/ 3.3V			RSVD	PSE_TRACEDATA_3	ISI_TRACEDATA_3			PSE_TGPIO50	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4
GP_F18	GP_F18/ PSE_I2S1_TXD/ AVS_I2S4_TXD/ PSE_TGPIO16	GP-In	None	1.8V/ 3.3V	PSE_I2S1_TXD				AVS_I2S4_TXD			PSE_TGPIO16	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4

Table 21-1. GPIO Multiplexing Table (Sheet 24 of 42)

GPIO (EDS)	EDS Signal Name	Default Function	Default Termination	I/O Voltage Support	Native Function 1 (EDS)	Native Function 2 (EDS)	Native Function 3 (EDS)	Native Function 4 (EDS)	Native Function 5 (EDS)	Native Function 6 (EDS)	Native Function 7 (EDS)	Native Function 8 (EDS)	Pin State During Reset	Pin State Immediately After Reset	SCI/GPE IS	SMI	NMI	Output power sequence deglitch	GPIO Community #
GP_F19	GP_F19/ PSE_I2S1_RXD/ AVS_I2S4_RXD/ PSE_TGPIO17	GP-In	None	1.8V/ 3.3V	PSE_I2S1_RXD				AVS_I2S4_RXD			PSE_TGPIO17	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4
GP_F02	GP_F02/ SIO_UART0_TXD	Native F1	None	1.8V/ 3.3V	RSVD	SIO_UART0_TXD							Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4
GP_F20	GP_F20	Native F1	None	1.8V/ 3.3V	RSVD								Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4
GP_F21	GP_F21	Native F1	None	1.8V/ 3.3V	RSVD								Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4
GP_F22	GP_F22/ PMC_VNN_CTRL	GP-In	None	1.8V/ 3.3V	PMC_VNN_CTRL								Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4
GP_F23	GP_F23/ PMC_V1P05_CTRL	GP-In	None	1.8V/ 3.3V	PMC_V1P05_CTRL								Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4

Table 21-1. GPIO Multiplexing Table (Sheet 25 of 42)

GPIO (EDS)	EDS Signal Name	Default Function	Default Termination	I/O Voltage Support	Native Function 1 (EDS)	Native Function 2 (EDS)	Native Function 3 (EDS)	Native Function 4 (EDS)	Native Function 5 (EDS)	Native Function 6 (EDS)	Native Function 7 (EDS)	Native Function 8 (EDS)	Pin State During Reset	Pin State Immediately After Reset	SCI/GPE IS	SMI	NMI	Output power sequence deglitch	GPIO Community #
GP_F03	GP_F03/ SIO_UART0_CTS_N	Native F1	20K PU	1.8V/ 3.3V	RSVD	SIO_UART0_CTS_N							Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4
GP_F04	GP_F04	Native F1	None	1.8V/ 3.3V	RSVD								Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4
GP_F05	GP_F05/ PSE_TGPIO14	Native F2	None	1.8V/ 3.3V	RSVD	RSVD	RSVD	RSVD				PSE_TGPIO14	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4
GP_F06	GP_F06/ PSE_TGPIO47	GP-In	None	1.8V/ 3.3V	RSVD							PSE_TGPIO47	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4
GP_F07	GP_F07/ PSE_I2S1_SCLK/ AVS_I2S4_S-CLK/PSE_TG-PIO14	GP-Out	None	1.8V/ 3.3V	PSE_I2S1_SCLK				AVS_I2S4_SCLK			PSE_TGPIO14	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4
GP_F08	GP_F08/ AVS_I2S_MC-LK2/PSE_TRACE-DATA_0/ ISI_TRACE-DATA_0/ PSE_TGPIO48	Native F5	None	1.8V/ 3.3V	AVS_I2S_MC-LK2			PSE_TRACE-DATA_0	ISI_TRACE-DATA_0			PSE_TGPIO48	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	4

Table 21-1. GPIO Multiplexing Table (Sheet 26 of 42)

GPIO (EDS)	EDS Signal Name	Default Function	Default Termination	I/O Voltage Support	Native Function 1 (EDS)	Native Function 2 (EDS)	Native Function 3 (EDS)	Native Function 4 (EDS)	Native Function 5 (EDS)	Native Function 6 (EDS)	Native Function 7 (EDS)	Native Function 8 (EDS)	Pin State During Reset	Pin State Immediately After Reset	SCI/GPE IS	SMI	NMI	Output power sequence deglitch	GPIO Community #
GP_F09	GP_F09	Native F1	None	1.8V/3.3V									Z	Z	Yes	No	No	Hi-Z output with internal 20K PD during power sequencing	4
GP_G00	GP_G00/SD_SDIO_CMD	GP-In	None	1.8V/3.3V	SD_SDIO_CMD								Missing	Missing	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0
GP_G01	GP_G01/SD_SDIO_D0	GP-In	None	1.8V/3.3V	SD_SDIO_D0								Missing	Missing	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0
GP_G10	GP_G10/ AVS_I2S2_RXD/ DMIC_DATA1	GP-In	None	1.8V/3.3V	AVS_I2S2_RXD			DMIC_DATA1					Missing	Missing	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0
GP_G11	GP_G11/ AVS_I2S3_SCLK/DMIC_DATA0/ PSE_TGPIO07	GP-In	None	1.8V/3.3V	AVS_I2S3_SCLK			DMIC_DATA0				PSE_TGPIO07	Missing	Missing	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0
GP_G12	GP_G12/ AVS_I2S3_SFRM/ SATA_1_GP/ SATAXPCIE_1/ DMIC_DATA1/ PSE_TGPIO31	Native F3/GP-In	None (SATAXPCIE Disabled) 20K PU (SATAXPCIE Enabled)	1.8V/3.3V	AVS_I2S3_SFRM	SATA_1_GP	SATAXPCIE_1	DMIC_DATA1				PSE_TGPIO31	Missing	Missing	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0

Table 21-1. GPIO Multiplexing Table (Sheet 27 of 42)

GPIO (EDS)	EDS Signal Name	Default Function	Default Termination	I/O Voltage Support	Native Function 1 (EDS)	Native Function 2 (EDS)	Native Function 3 (EDS)	Native Function 4 (EDS)	Native Function 5 (EDS)	Native Function 6 (EDS)	Native Function 7 (EDS)	Native Function 8 (EDS)	Pin State During Reset	Pin State Immediately After Reset	SCI/GPE IS	SMI	NMI	Output power sequence deglitch	GPIO Community #
GP_G13	GP_G13/ AVS_I2S3_TXD/ DMIC_CLK_B0/ PSE_TGPIO08	GP-In	None	1.8V/ 3.3V	AVS_I2S3_TXD			DMIC_CLK_B0				PSE_TGPIO08	Missing	Missing	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0
GP_G14	GP_G14/ AVS_I2S3_RXD/ DMIC_CLK_B1/ PSE_TGPIO09	GP-In	None	1.8V/ 3.3V	AVS_I2S3_RXD			DMIC_CLK_B1				PSE_TGPIO09	Missing	Missing	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0
GP_G15	GP_G15/ ESPI_IO0	Native F1/GP-In	None (eSPI Disabled) 20K PU (eSPI Enabled)	1.8V/ 3.3V	ESPI_IO0								Missing	Missing	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0
GP_G16	GP_G16/ ESPI_IO1	Native F1/GP-In	None (eSPI Disabled) 20K PU (eSPI Enabled)	1.8V/ 3.3V	ESPI_IO1								Missing	Missing	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0
GP_G17	GP_G17/ ESPI_IO2	Native F1/ Native F2	None (eSPI Disabled) 20K PU (eSPI Enabled)	1.8V/ 3.3V	ESPI_IO2	Reserved							Missing	Missing	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0
GP_G18	GP_G18/ ESPI_IO3	Native F1/ Native F2	20K PU	1.8V/ 3.3V	ESPI_IO3	Reserved							Missing	Missing	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0

Table 21-1. GPIO Multiplexing Table (Sheet 28 of 42)

GPIO (EDS)	EDS Signal Name	Default Function	Default Termination	I/O Voltage Support	Native Function 1 (EDS)	Native Function 2 (EDS)	Native Function 3 (EDS)	Native Function 4 (EDS)	Native Function 5 (EDS)	Native Function 6 (EDS)	Native Function 7 (EDS)	Native Function 8 (EDS)	Pin State During Reset	Pin State Immediately After Reset	SCI/GPE IS	SMI	NMI	Output power sequence deglitch	GPIO Community #
GP_G19	GP_G19/ AVS_I2S1_SCLK	GP-In	None	1.8V/ 3.3V	AVS_I2S1_S-CLK								Missing	Missing	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0
GP_G02	GP_G02/SD_S-DIO_D1	GP-In	None	1.8V/ 3.3V	SD_S-DIO_D1								Missing	Missing	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0
GP_G20	GP_G20/ ESPI_CS0_N	Native F1/GP-In	None (eSPI Disabled) 20K PU (eSPI Enabled)	1.8V/ 3.3V	ESPI_CS0_N								Missing	Missing	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0
GP_G21	GP_G21/ESPI_-CLK	Native F1/GP-In	None (eSPI Disabled) 20K PD (eSPI Enabled)	1.8V/ 3.3V	ESPI_-CLK								Missing	Missing	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0
GP_G22	GP_G22/ ESPI_RST0_N	Native F1/GP-In	None	1.8V/ 3.3V	ESPI_RST0_N								Missing	Missing	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0
GP_G23	GP_G23/SD_S-DIO_WP	GP-In	None	1.8V/ 3.3V	SD_S-DIO_WP								Missing	Missing	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0

Table 21-1. GPIO Multiplexing Table (Sheet 29 of 42)

GPIO (EDS)	EDS Signal Name	Default Function	Default Termination	I/O Voltage Support	Native Function 1 (EDS)	Native Function 2 (EDS)	Native Function 3 (EDS)	Native Function 4 (EDS)	Native Function 5 (EDS)	Native Function 6 (EDS)	Native Function 7 (EDS)	Native Function 8 (EDS)	Pin State During Reset	Pin State Immediately After Reset	SCI/GPE IS	SMI	NMI	Output power sequence deglitch	GPIO Community #
GP_G03	GP_G03/SD_SDIO_D2	GP-In	None	1.8V/3.3V	SD_SDIO_D2								Missing	Missing	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0
GP_G04	GP_G04/SD_SDIO_D3	GP-In	None	1.8V/3.3V	SD_SDIO_D3								Missing	Missing	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0
GP_G05	GP_G05/SD_SDIO_CD_N	GP-In	None	1.8V/3.3V	SD_SDIO_CD_N								Missing	Missing	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0
GP_G06	GP_G06/SD_SDIO_CLK	GP-In	None	1.8V/3.3V	SD_SDIO_CLK								Missing	Missing	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0
GP_G07	GP_G07/AVS_I2S2_SCLK/DMIC_CLK_A0	GP-In	None	1.8V/3.3V	AVS_I2S2_SCLK			DMIC_CLK_A0					Missing	Missing	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0
GP_G08	GP_G08/AVS_I2S2_SFRM/DMIC_DATA0	Native F2	None	1.8V/3.3V	AVS_I2S2_SFRM	RSVD		DMIC_DATA0					Missing	Missing	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0
GP_G09	GP_G09/	Native	None	1.8V/	AVS_I2	RSVD	RSVD	DMIC_					Missing	Missing	Yes	No	No	Hi-Z output with	0

Table 21-1. GPIO Multiplexing Table (Sheet 30 of 42)

GPIO (EDS)	EDS Signal Name	Default Function	Default Termination	I/O Voltage Support	Native Function 1 (EDS)	Native Function 2 (EDS)	Native Function 3 (EDS)	Native Function 4 (EDS)	Native Function 5 (EDS)	Native Function 6 (EDS)	Native Function 7 (EDS)	Native Function 8 (EDS)	Pin State During Reset	Pin State Immediately After Reset	SCI/GPE IS	SMI	NMI	Output power sequence deglitch	GPIO Community #
GP_H00	GP_H00/PSE_GBE1_INT/ PSE_UART5_RXD	GP-Out	None	1.8V/ 3.3V	PSE_GBE1_INT			PSE_UART5_RXD					Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_H01	GP_H01/PSE_GBE1_RST_N/ PSE_UART5_TXD	GP-In	None	1.8V/ 3.3V	PSE_GBE1_RST_N			PSE_UART5_TXD					Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_H10	GP_H10/PCIE_CLKREQ4_N/ PSE_PWM14	GP-In	None	1.8V/ 3.3V	PCIE_CLKREQ4_N			PSE_PWM14					Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_H11	GP_H11/PCIE_CLKREQ5_N/ PSE_PWM15	GP-In	None	1.8V/ 3.3V	PCIE_CLKREQ5_N			PSE_PWM15					Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_H12	GP_H12/ PSE_UART1_RXD/ M2_SKT2_CFG0/ PSE_TGPIO51	GP-In	None (M2_SKT_CFG Disabled) 20K PU (M2_SKT_CFG Enabled)	1.8V/ 3.3V	PSE_UART1_RXD			M2_SKT2_CFG0				PSE_TGPIO51	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_H13	GP_H13/ PSE_UART1_TXD/ M2_SKT2_CFG1/ PSE_TGPIO52	GP-In	None (M2_SKT_CFG Disabled) 20K PU (M2_SKT_CFG Enabled)	1.8V/ 3.3V	PSE_UART1_TXD			M2_SKT2_CFG1				PSE_TGPIO52	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1

Table 21-1. GPIO Multiplexing Table (Sheet 31 of 42)

GPIO (EDS)	EDS Signal Name	Default Function	Default Termination	I/O Voltage Support	Native Function 1 (EDS)	Native Function 2 (EDS)	Native Function 3 (EDS)	Native Function 4 (EDS)	Native Function 5 (EDS)	Native Function 6 (EDS)	Native Function 7 (EDS)	Native Function 8 (EDS)	Pin State During Reset	Pin State Immediately After Reset	SCI/GPE IS	SMI	NMI	Output power sequence deglitch	GPIO Community #
GP_H14	GP_H14/ M2_SKT2_CFG2/ PSE_TGPIO53	GP-In	None (M2_SKT_CFG Disabled) 20K PU (M2_SKT_CFG Enabled)	1.8V/ 3.3V				M2_SKT2_CFG2				PSE_TGPIO53	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_H15	GP_H15/ PSE_UART1_CTS_N/ M2_SKT2_CFG3/ PSE_TGPIO54	GP-In	None (M2_SKT_CFG Disabled) 20K PU (M2_SKT_CFG Enabled)	1.8V/ 3.3V	PSE_UART1_CTS_N			M2_SKT2_CFG3				PSE_TGPIO54	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_H16	GP_H16/ PCIE_LNK_DOWN/DDI2_DDC_SCL	GP-In	None	1.8V/ 3.3V	PCIE_LNK_DOWN	DDI2_DDC_SCL							Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_H17	GP_H17/SD_SDIO_PWR_EN_N	GP-In	None	1.8V/ 3.3V	SD_SDIO_PWR_EN_N								Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_H18	GP_H18/ PMC_CPU_C10_GATE_N	Native F1	None	1.8V/ 3.3V	PMC_CPU_C10_GATE_N								Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_H19	GP_H19/ DDI2_DDC_SDA/ PMC_TGPIO0/ PSE_TGPIO20	GP-In	None	1.8V/ 3.3V		DDI2_DDC_SDA		PMC_TGPIO0				PSE_TGPIO20	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1

Table 21-1. GPIO Multiplexing Table (Sheet 32 of 42)

GPIO (EDS)	EDS Signal Name	Default Function	Default Termination	I/O Voltage Support	Native Function 1 (EDS)	Native Function 2 (EDS)	Native Function 3 (EDS)	Native Function 4 (EDS)	Native Function 5 (EDS)	Native Function 6 (EDS)	Native Function 7 (EDS)	Native Function 8 (EDS)	Pin State During Reset	Pin State Immediately After Reset	SCI/GPE IS	SMI	NMI	Output power sequence deglitch	GPIO Community #
GP_H02	GP_H02/PSE_GBE1_AUXTS/PSE_UART5_RTS_N	GP-Out	None	1.8V/3.3V	PSE_GBE1_AUXTS			PSE_UART5_RTS_N					Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_H20	GP_H20/PSE_PWM07/DDI2_HPD/PSE_TGPIO55	GP-In	None	1.8V/3.3V	PSE_PWM07	DDI2_HPD		RSVD				PSE_TGPIO55	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_H21	GP_H21/PSE_HSUART1_DE/PSE_UART1_RTS_N/PSE_TGPIO56	GP-In	None	1.8V/3.3V	PSE_HSUART1_DE	PSE_UART1_RTS_N		RSVD				PSE_TGPIO56	Z	Z	Yes	No	No	None	1
GP_H22	GP_H22/PSE_HSUART1_RE/PSE_TGPIO57	GP-In	None	1.8V/3.3V	PSE_HSUART1_RE			RSVD				PSE_TGPIO57	Z	Z	Yes	No	No	None	1
GP_H23	GP_H23/PSE_HSUART1_EN/PSE_TGPIO58	GP-In	None	1.8V/3.3V	PSE_HSUART1_EN			RSVD				PSE_TGPIO58	Z	Z	Yes	No	No	None	1
GP_H03	GP_H03/PSE_GBE1_PPS/PSE_UART5_CTS_N/PSE_TGPIO21	GP-In	None	1.8V/3.3V	PSE_GBE1_PPS			PSE_UART5_CTS_N				PSE_TGPIO21	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_H04	GP_H04/SIO_I2C2_SDA/PSE_PWM08/PSE_TGPIO10	GP-In	None	1.8V/3.3V	SIO_I2C2_SDA			PSE_PWM08				PSE_TGPIO10	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_H05	GP_H05/SIO_I2C2_SCL/PSE_PWM09/PSE_TGPIO11	GP-In	None	1.8V/3.3V	SIO_I2C2_SCL			PSE_PWM09				PSE_TGPIO11	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1

Table 21-1. GPIO Multiplexing Table (Sheet 33 of 42)

GPIO (EDS)	EDS Signal Name	Default Function	Default Termination	I/O Voltage Support	Native Function 1 (EDS)	Native Function 2 (EDS)	Native Function 3 (EDS)	Native Function 4 (EDS)	Native Function 5 (EDS)	Native Function 6 (EDS)	Native Function 7 (EDS)	Native Function 8 (EDS)	Pin State During Reset	Pin State Immediately After Reset	SCI/GPE IS	SMI	NMI	Output power sequence deglitch	GPIO Community #
GP_H06	GP_H06/ SIO_I2C3_SDA/ PSE_I2C5_SDA/ PSE_PWM10	GP-In	None	1.8V/ 3.3V	SIO_I2C3_SDA	PSE_I2C5_SDA		PSE_PWM10					Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_H07	GP_H07/ SIO_I2C3_SCL/ PSE_I2C5_SCL/ PSE_PWM11	GP-In	None	1.8V/ 3.3V	SIO_I2C3_SCL	PSE_I2C5_SCL		PSE_PWM11					Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_H08	GP_H08/ SIO_I2C4_SDA/ PSE_PWM12	GP-In	None	1.8V/ 3.3V	SIO_I2C4_SDA	RSVD		PSE_PWM12					Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_H09	GP_H09/ SIO_I2C4_SCL/ PSE_PWM13	GP-In	None	1.8V/ 3.3V	SIO_I2C4_SCL	RSVD		PSE_PWM13					Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_T00	GP_T00/ PSE_QEPA2/ SIO_I2C6_SDA/ PSE_TGPIO08	GP-In	None	1.8V/ 3.3V	PSE_QEPA2		SIO_I2C6_SDA					PSE_TGPIO08	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0
GP_T01	GP_T01/ PSE_QEPB2/ SIO_I2C6_SCL/ PSE_TGPIO09	GP-In	None	1.8V/ 3.3V	PSE_QEPB2		SIO_I2C6_SCL					PSE_TGPIO09	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0

Table 21-1. GPIO Multiplexing Table (Sheet 34 of 42)

GPIO (EDS)	EDS Signal Name	Default Function	Default Termination	I/O Voltage Support	Native Function 1 (EDS)	Native Function 2 (EDS)	Native Function 3 (EDS)	Native Function 4 (EDS)	Native Function 5 (EDS)	Native Function 6 (EDS)	Native Function 7 (EDS)	Native Function 8 (EDS)	Pin State During Reset	Pin State Immediately After Reset	SCI/GPE IS	SMI	NMI	Output power sequence deglitch	GPIO Community #
GP_T10	GP_T10/PSE_HSUART2_RE	GP-In	None	1.8V/3.3V	PSE_HSUART2_RE	RSVD							Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0
GP_T11	GP_T11/ USB2_OC3_N/ PSE_TGPIO06	GP-In	None	1.8V/3.3V	USB2_OC3_N	RSVD						PSE_TGPIO06	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0
GP_T12	GP_T12/ PSE_UART2_RXD/ SIO_UART0_RXD	GP-In	None	1.8V/3.3V	PSE_UART2_RXD	SIO_UART0_RXD							Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0
GP_T13	GP_T13/ PSE_UART2_TXD/ SIO_UART0_TXD	GP-In	None	1.8V/3.3V	PSE_UART2_TXD	SIO_UART0_TXD							Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0
GP_T14	GP_T14/ PSE_UART2_RTS_N/ SIO_UART0_RTS_N/PSE_HSUART2_DE	GP-In	None	1.8V/3.3V	PSE_UART2_RTS_N	SIO_UART0_RTS_N	PSE_HSUART2_DE						Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0
GP_T15	GP_T15/ PSE_UART2_CTS_N/ SIO_UART0_CTS_N	GP-In	None	1.8V/3.3V	PSE_UART2_CTS_N	SIO_UART0_CTS_N							Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0

Table 21-1. GPIO Multiplexing Table (Sheet 35 of 42)

GPIO (EDS)	EDS Signal Name	Default Function	Default Termination	I/O Voltage Support	Native Function 1 (EDS)	Native Function 2 (EDS)	Native Function 3 (EDS)	Native Function 4 (EDS)	Native Function 5 (EDS)	Native Function 6 (EDS)	Native Function 7 (EDS)	Native Function 8 (EDS)	Pin State During Reset	Pin State Immediately After Reset	SCI/GPE IS	SMI	NMI	Output power sequence deglitch	GPIO Community #
GP_T02	GP_T02/ PSE_QEPI2/ SIO_I2C7_SDA/ PSE_TGPIO07	GP-In	None	1.8V/ 3.3V	PSE_QEPI2		SIO_I2C7_SDA					PSE_TGPIO07	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0
GP_T03	GP_T03/ SIO_I2C7_SCL/ PSE_TGPIO06	GP-In	None	1.8V/ 3.3V			SIO_I2C7_SCL					PSE_TGPIO06	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0
GP_T04	GP_T04/PSE_GBE0_INT	GP-In	None	1.8V/ 3.3V	PSE_GBE0_INT			RSVD					Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0
GP_T05	GP_T05/PSE_GBE0_RST_N	GP-In	None	1.8V/ 3.3V	PSE_GBE0_RST_N			RSVD					Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0
GP_T06	GP_T06/PSE_GBE0_AUXTS/ USB2_OC1_N	GP-In	None	1.8V/ 3.3V	PSE_GBE0_AUXTS	USB2_OC1_N		RSVD					Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0
GP_T07	GP_T07/PSE_GBE0_PPS/ PSE_TGPIO59	GP-In	None	1.8V/ 3.3V	PSE_GBE0_PPS			RSVD				PSE_TGPIO59	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0

Table 21-1. GPIO Multiplexing Table (Sheet 36 of 42)

GPIO (EDS)	EDS Signal Name	Default Function	Default Termination	I/O Voltage Support	Native Function 1 (EDS)	Native Function 2 (EDS)	Native Function 3 (EDS)	Native Function 4 (EDS)	Native Function 5 (EDS)	Native Function 6 (EDS)	Native Function 7 (EDS)	Native Function 8 (EDS)	Pin State During Reset	Pin State Immediately After Reset	SCI/GPE IS	SMI	NMI	Output power sequence deglitch	GPIO Community #
GP_T08	GP_T08/ USB2_OC2_N/ PSE_TGPIO22	GP-In	None	1.8V/ 3.3V	USB2_OC2_N	RSVD						PSE_TGPIO22	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0
GP_T09	GP_T09/PSE_HSUART2_EN	GP-In	None	1.8V/ 3.3V	PSE_HSUART2_EN	RSVD							Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	0
GP_U00	GP_U00/ GBE_INT/ PSE_I2C6_SCL	GP-In	None	1.8V/ 3.3V	GBE_INT	PSE_I2C6_SCL	RSVD						Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_U01	GP_U01/ GBE_RST_N/ PSE_I2C6_SDA	GP-In	None	1.8V/ 3.3V	GBE_RST_N	PSE_I2C6_SDA	RSVD						Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_U10	GP_U10/ ISI_SPIS_MISO/ ISI_I2CS_SDA/ PSE_TGPIO12	GP-In	None	1.8V/ 3.3V	ISI_SPIS_MISO	ISI_I2CS_SDA	RSVD					PSE_TGPIO12	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_U11	GP_U11/ PSE_QEPB3/ PSE_TGPIO11	GP-In	None	1.8V/ 3.3V	PSE_QEPB3		RSVD					PSE_TGPIO11	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1

Table 21-1. GPIO Multiplexing Table (Sheet 37 of 42)

GPIO (EDS)	EDS Signal Name	Default Function	Default Termination	I/O Voltage Support	Native Function 1 (EDS)	Native Function 2 (EDS)	Native Function 3 (EDS)	Native Function 4 (EDS)	Native Function 5 (EDS)	Native Function 6 (EDS)	Native Function 7 (EDS)	Native Function 8 (EDS)	Pin State During Reset	Pin State Immediately After Reset	SCI/GPE IS	SMI	NMI	Output power sequence deglitch	GPIO Community #
GP_U12	GP_U12/ ISI_CHX_OK-NOK_0	Native F1	20K PD	1.8V/ 3.3V	ISI_CHX_OK-NOK_0		RSVD						Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_U13	GP_U13/ ISI_CHX_OK-NOK_1	Native F1	20K PD	1.8V/ 3.3V	ISI_CHX_OK-NOK_1		RSVD						Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_U14	GP_U14/ ISI_CHX_RLY_SWTCH	GP-In	None	1.8V/ 3.3V	ISI_CHX_RLY_SWTCH		RSVD						Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_U15	GP_U15/ ISI_CHX_PMIC_EN/ PSE_TGPIO13	GP-In	None	1.8V/ 3.3V	ISI_CHX_PMIC_EN		RSVD					PSE_TGPIO13	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_U16	GP_U16/ISI_OK-NOK_0	Native F1	None	1.8V/ 3.3V	ISI_OK-NOK_0		RSVD						Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_U17	GP_U17/ISI_OK-NOK_1	Native F1	None	1.8V/ 3.3V	ISI_OK-NOK_1		RSVD						Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1

Table 21-1. GPIO Multiplexing Table (Sheet 38 of 42)

GPIO (EDS)	EDS Signal Name	Default Function	Default Termination	I/O Voltage Support	Native Function 1 (EDS)	Native Function 2 (EDS)	Native Function 3 (EDS)	Native Function 4 (EDS)	Native Function 5 (EDS)	Native Function 6 (EDS)	Native Function 7 (EDS)	Native Function 8 (EDS)	Pin State During Reset	Pin State Immediately After Reset	SCI/GPE IS	SMI	NMI	Output power sequence deglitch	GPIO Community #
GP_U18	GP_U18/ ISI_ALERT_N	Native F1	None	1.8V/ 3.3V	ISI_ALERT_N		RSVD						Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_U19	GP_U19/ PSE_QEPI3/ PSE_TGPIO12	GP-In	None	1.8V/ 3.3V	PSE_QEPI3		RSVD					PSE_TGPIO12	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_U02	GP_U02/ GBE_PPS/ PSE_I2C7_SCL	GP-In	None	1.8V/ 3.3V	GBE_PPS	PSE_I2C7_SCL	RSVD						Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_U03	GP_U03/ GBE_AUXTS/ PSE_I2C7_SDA	GP-In	None	1.8V/ 3.3V	GBE_AUXTS	PSE_I2C7_SDA	RSVD						Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_U04	GP_U04/ ISI_SPIM_CS/ PSE_S-PI1_CS0_N	GP-In	None	1.8V/ 3.3V	ISI_SPIM_CS		RSVD	PSE_S-PI1_CS0_N					Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_U05	GP_U05/ ISI_SPIM_SCLK/ PSE_SPI1_CLK	GP-In	None	1.8V/ 3.3V	ISI_SPIM_SCLK		RSVD	PSE_S-PI1_-CLK					Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1

Table 21-1. GPIO Multiplexing Table (Sheet 39 of 42)

GPIO (EDS)	EDS Signal Name	Default Function	Default Termination	I/O Voltage Support	Native Function 1 (EDS)	Native Function 2 (EDS)	Native Function 3 (EDS)	Native Function 4 (EDS)	Native Function 5 (EDS)	Native Function 6 (EDS)	Native Function 7 (EDS)	Native Function 8 (EDS)	Pin State During Reset	Pin State Immediately After Reset	SCI/GPE IS	SMI	NMI	Output power sequence deglitch	GPIO Community #
GP_U06	GP_U06/ ISI_SPIM_MISO/ PSE_SPI1_MISO	GP-In	None	1.8V/ 3.3V	ISI_SPIM_MISO		RSVD	PSE_SPI1_MISO					Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_U07	GP_U07/ PSE_QEPA3/ PSE_SPI1_MOSI/PSE_TGPIO10	GP-In	None	1.8V/ 3.3V	PSE_QEPA3		RSVD	PSE_SPI1_MOSI				PSE_TGPIO10	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_U08	GP_U08/ ISI_SPIS_CS/ PSE_TGPIO10	GP-In	None	1.8V/ 3.3V	ISI_SPIS_CS		RSVD					PSE_TGPIO10	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_U09	GP_U09/ ISI_SPIS_SCLK/ ISI_I2CS_SCL/ PSE_TGPIO11	GP-In	None	1.8V/ 3.3V	ISI_SPIS_SCLK	ISI_I2CS_SCL	RSVD					PSE_TGPIO11	Z	Z	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_V00	GP_V00/ EMMC_CMD	GP-In	None	1.8V	EMMC_CMD								Missing	Missing	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_V01	GP_V01/EMMC_DATA0	GP-In	None	1.8V	EMMC_DATA0								Missing	Missing	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1

Table 21-1. GPIO Multiplexing Table (Sheet 40 of 42)

GPIO (EDS)	EDS Signal Name	Default Function	Default Termination	I/O Voltage Support	Native Function 1 (EDS)	Native Function 2 (EDS)	Native Function 3 (EDS)	Native Function 4 (EDS)	Native Function 5 (EDS)	Native Function 6 (EDS)	Native Function 7 (EDS)	Native Function 8 (EDS)	Pin State During Reset	Pin State Immediately After Reset	SCI/GPE IS	SMI	NMI	Output power sequence deglitch	GPIO Community #
GP_V02	GP_V02/EMMC_DATA1	GP-In	None	1.8V	EMMC_DATA1								Missing	Missing	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_V03	GP_V03/EMMC_DATA2	GP-In	None	1.8V	EMMC_DATA2								Missing	Missing	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_V04	GP_V04/EMMC_DATA3	GP-In	None	1.8V	EMMC_DATA3								Missing	Missing	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_V05	GP_V05/EMMC_DATA4	GP-In	None	1.8V	EMMC_DATA4								Missing	Missing	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_V06	GP_V06/EMMC_DATA5	GP-In	None	1.8V	EMMC_DATA5								Missing	Missing	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_V07	GP_V07/EMMC_DATA6	GP-In	None	1.8V	EMMC_DATA6								Missing	Missing	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1

Table 21-1. GPIO Multiplexing Table (Sheet 41 of 42)

GPIO (EDS)	EDS Signal Name	Default Function	Default Termination	I/O Voltage Support	Native Function 1 (EDS)	Native Function 2 (EDS)	Native Function 3 (EDS)	Native Function 4 (EDS)	Native Function 5 (EDS)	Native Function 6 (EDS)	Native Function 7 (EDS)	Native Function 8 (EDS)	Pin State During Reset	Pin State Immediately After Reset	SCI/GPE IS	SMI	NMI	Output power sequence deglitch	GPIO Community #
GP_V08	GP_V08/EMMC_DATA7	GP-In	None	1.8V	EMMC_DATA7								Missing	Missing	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_V09	GP_V09/EMMC_RCLK	GP-In	None	1.8V	EMMC_RCLK								Missing	Missing	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_V10	GP_V10/EMMC_CLK	GP-In	None	1.8V	EMMC_CLK								Missing	Missing	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_V11	GP_V11/EMMC_RST_N	GP-In	None	1.8V	EMMC_RST_N								Missing	Missing	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_V12	GP_V12/PSE_TGPIO00	GP-In	None	1.8V	PSE_TGPIO00								Missing	Missing	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_V13	GP_V13/PSE_TGPIO01	GP-In	None	1.8V	PSE_TGPIO01								Missing	Missing	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1

Table 21-1. GPIO Multiplexing Table (Sheet 42 of 42)

GPIO (EDS)	EDS Signal Name	Default Function	Default Termination	I/O Voltage Support	Native Function 1 (EDS)	Native Function 2 (EDS)	Native Function 3 (EDS)	Native Function 4 (EDS)	Native Function 5 (EDS)	Native Function 6 (EDS)	Native Function 7 (EDS)	Native Function 8 (EDS)	Pin State During Reset	Pin State Immediately After Reset	SCI/GPE IS	SMI	NMI	Output power sequence deglitch	GPIO Community #
GP_V14	GP_V14/ PSE_TGPIO02	GP-In	None	1.8V	PSE_TGPIO02								Missing	Missing	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1
GP_V15	GP_V15/ PSE_TGPIO03	GP-In	None	1.8V	PSE_TGPIO03								Missing	Missing	Yes	No	No	Hi-Z output with no internal termination during power sequencing	1

21.5 Registers

Please refer to chapter 21 of the Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors for Internet of Things (IoT) Applications, Datasheet, Volume 2 (Book 2 of 3), Mule Creek Canyon (Document Number: 636722), for description of the registers associated with subject of this chapter.

§ §

22 Intel® Programmable Services Engine (Intel® PSE)

22.1 Overview

The Intel® Programmable Services Engine (Intel® PSE) is designed as an Asymmetric Multi-Processing (AMP) system, comprising both an IA Processor core on the compute die, and an Arm* Cortex*-M7 core based subsystem on the Platform Controller Hub. The Arm* Cortex*-M7 core shall primarily be used by the IA Processor core as a target (Refer to Section 1.1.1 for more information on target) co-processor for handling firmware defined scenarios including, for example, real-time/latency-sensitive applications, real-time industrial communication protocols or low-power industrial sensing solutions.

In addition to the Arm* Cortex*-M7 core, the Intel® PSE also implements a number of interfaces as described later in this chapter. Ownership of these interfaces can be assigned to either the Arm* Cortex*-M7 or the IA Processor but not both.

The Intel® PSE is designed to support following functions:

- Acquisition/sampling of sensor data.
- Provide I/O interfaces catering to Industrial and Automotive platforms, such as RS-485, Ethernet and CANBUS.
- Low power operation through clock and power gating of the Intel® PSE blocks.
- The ability to operate independently when the host platform is in a low power state.
- Provide the capability to expose Intel® PSE I/O interfaces (such as Ethernet and CANbus) to IA Processor drivers (running on ATOM). Note that the ownership of I/O interfaces between Intel® PSE and the IA Processor are mutually exclusive.
- Provide the capability to expose each I/O controller as an independent single or multi-function PCI device.
- Provide a host interface for BIOS/Intel® PSE drivers to communicate with Intel® PSE FW.

22.2 Functional Description

The Intel® PSE consists of the following key components:

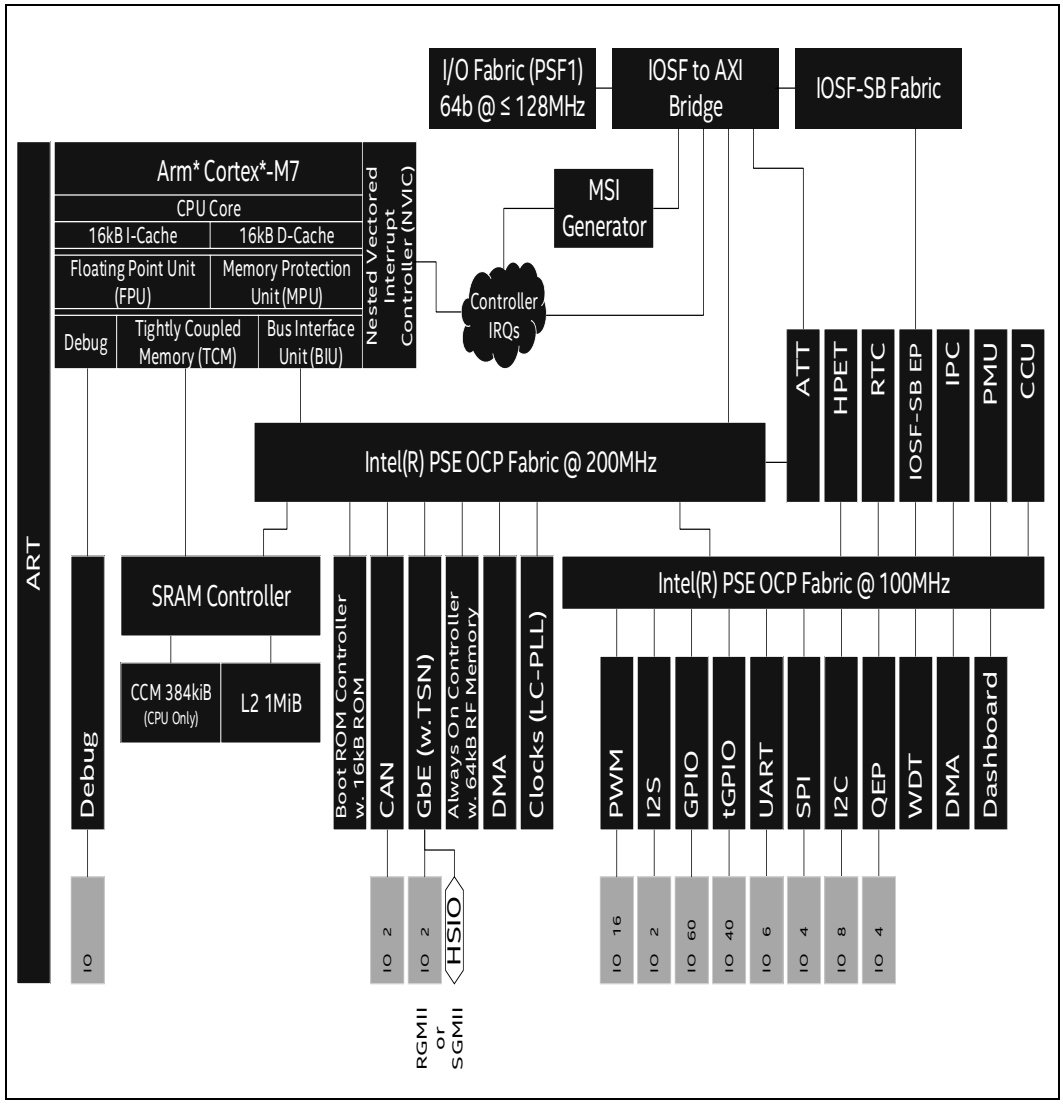
- Arm* Cortex*-M7 core with Core Coupled Memory (CCM) configuration
  - 384 KB CCM SRAM space for code and data with 64KB of SRAM for repair/redundancy
  - 1MB L2 SRAM
  - 16KB L1 ICache and 16KB DCache
- Interfaces to time-sensitive peripheral controllers (supporting time stamping using the ART): Ethernet, CAN & TGPIO
- General purpose controllers like I^2C, SPI, PWM, UART, GPIO, I^2S & QEP

- An Intel On-chip System Fabric (IOSF) connection to the IA Processor IOSF backbone (and hence to main memory)
- Timers such as time-synchronization support local ART, watchdog, real-time clock (RTC), and high-performance event timer (HPET)
- Two IOSF sideband interfaces
- Out of Band signals for clock and wakeup control
- Inter-Processor Communications (IPC) to Host & PMC

22.3 Block Diagram

The block diagram shows the interconnections of all of the Intel® PSE blocks.

Figure 22-1. Intel® PSE Block Diagram



22.4 Intel® PSE Resources Required

Intel® PSE requires a variety of resources within the rest of the processor.

Table 22-1. List of Arm* Cortex*-M7 resources required

Resource	Description
External Interfaces	Peripheral signals that are owned by the Intel® PSE are muxed with GPIO pins that are owned by the Host CPU. I^2C, SPI, UART, PWM, QEP, GbE, CAN, TGPIO are blocks within Intel® PSE configurable from GPIO cluster.
Soft Straps	Various soft straps are required and provisioned for configuration of Intel® PSE.
Protected Region of Main Memory	Region to which the Intel® PSE can read/write and no other device can read/write other than the secure agent (refers to CSE/BIOS). This is the IMR (Isolated Memory Region) area allocated for Intel® PSE/secure-agent access.
System Memory Space	Intel® PSE is a multi-function device supporting 36 PCI device-functions. Each function has pre-defined MMIO space and is enabled and mapped via its corresponding PCI configuration space.
PCI configuration space	Region within the processor's PCI configuration space on Bus 0.
Interrupt	Intel® PSE asserts IRQx (BIOS configured) to ATOM. Intel® PSE also sends MSI (with multi-message capability) to ATOM. The interrupt delivery mechanism (MSI vs IRQ) is to be setup by BIOS by programming the interrupt delivery register.

22.5 Arm* Cortex*-M7 Subsystem

22.5.1 Overview

The Arm* Cortex*-M7 is a highly efficient high-performance, embedded processor that features low interrupt latency, low-cost debug, and has backwards compatibility with existing Cortex-M profile processors.

The Arm* Cortex*-M7 will run pre-compiled firmware that will be provisioned as part of the IFWI (Integrated Firmware Image) present on either the SPI or flash storage device(s). The firmware defines the persona (function) of the Intel® PSE & the interfaces owned by the Arm* Cortex*-M7.

22.5.2 Features Overview

The main features of the Arm* Cortex*-M7 core include:

- An in-order issue, super-scalar pipeline with dynamic branch prediction.
- DSP extensions.
- The Armv7-M Thumb instruction set, defined in the Armv7-M Architecture Reference Manual.
- Banked Stack Pointer (SP).
- Hardware integer divide instructions, SDIV and UDIV.

- Handler and Thread modes.
- Thumb and Debug states.
- Automatic processor state saving and restoration for low-latency Interrupt Service Routine (ISR) entry and exit.
- Support for Arm* Cortex*-M7 big-endian byte-invariant or little-endian accesses.
- Support for Arm* Cortex*-M7 unaligned accesses.
- Low-latency interrupt processing achieved by:
  - A Nested Vectored Interrupt Controller (NVIC) closely integrated with the Arm* Cortex*-M7 core.
  - Supporting exception-continuable instructions, such as LDM, LDMDB, STM, STMDB, PUSH, POP and VLDM, VSTM, VPUSH, VPOP.
- A memory system, that includes the MPU (that can be configured to protect regions of memory) and Harvard data and instruction cache with ECC protection.
- A Floating Point Unit (FPU).
- Low-power features including architectural clock gating and sleep mode.
- AXI to AHB bridge for legacy memory system support.

22.6 L2 SRAM

22.6.1 Overview

The Intel® PSE supports 1MB of L2 SRAM with the following features:

- 32kB bank size
- Full 1MB usable by Arm* Cortex*-M7 core for Intel® PSE firmware that is larger than the 384kB Core Coupled Memory SRAM.
- 128kB per Intel® PSE GbE controller usable by the IA Processor only if enabled by Intel® PSE firmware, for network proxy functionality
- SEC-DED ECC protection for single bit error correction and double bit error detection. Errors are reported to the Arm* Cortex*-M7 core as a NMI and memory address.
- 200MHz clock frequency
- Bank level power gating in D0i3

22.7 Clock Control Unit (CCU) and PLL

PSE is integrating Low Power LCPLL to generate the following clocks:

- 125 MHz RGMII mode clock for GBE TSN
- 500 Mhz fast clock for PSE CPU
- 400 Mhz clock for RGMII DLL reference clock
- 200 MHz clock for TMT timer in TGPIO

LCPLL integrated in the ungated domain of the OSE main partition. The LC PLL will be powered up by a separate power rail (V1P05_IS) and a reference clock of 38.4 Mhz (lcpll_ref_xtal_clk). The PLL is controlled by PSE FW through configuration registers in

PMU which has the interface to the PLL for the power up sequence captured in the PLL specification. The PLL also has a Configuration Register Interface (CRI), which will be accessible to PSE FW through fabric.

The PLL in OSE will be configured to generate a 6 GHz clock which is further divided to 500 Mhz, 400 Mhz, 200 MHz and 125 MHz clocks using a post divider Hard IP. These clocks are further used by the Clock Control Unit to generate the clocks listed above.

If the IP is owned by PSE, then the clock gate enable configuration register for that IP is present in CCU. For all HOST owned IPs, the clock gate enable configuration registers are part of the MMIO space of the device register accessible to the HOST driver. When IP ownership is NULL, then IP is permanently clock gated.

If the IP is owned by PSE, then the soft reset enable configuration register for that IP is present in PSE CCU. Similarly, for all HOST owned IPs, the soft reset enable configuration registers are part of the IP MMIO address accessible only to the HOST driver.

More details on clocking can be found in Chapter 20.

22.8 Power Management Unit (PMU)

The PMU is partitioned into two, the PMU AON and PMU gated:

1. **PMU AON**
   The PMU AON module implements the AON clock gate FSM along with the associated Synchronizer module, wake record logic and the glitch filters for the GPIO wakes.
   The SRAM PG FSM is also implemented in this module to allow for the SRAMs to be powered up even during IP accessible power gating when required. This would be required for D0i2 and Dxi2 power states.
2. **PMU Gated**
   The PMU gated module implements Functional clock gate FSM along with associated wake logic, glitch filter logic and synchronization logic.

22.9 Address Translation Table (ATT)

To support proxy mode use-case, 128KB of L2 SRAM is exposed to Host (GBE driver) contiguous with the GBE MMIO space. This is achieved by opening up ATT entries for the GBE IP and L2 SRAM via the ATT

- ATT does address translation on fixed MMIO address ranges
- 0x8020_0000 – 0x8024_0000 (256KB range for GBE0)
- 0x8040_0000 – 0x8044_0000 (256KB range for GBE1)

22.10 AON Controller

22.10.1 Overview

- Light-weight scalable controller that sits on the PSE OCP fabric
- Supports 64KB of always on RF memory.

- Supports 4 bits odd parity (1 bit per byte).
- {PAR[3], DATA[31:24], PAR[2], DATA[23:16], PAR[1], DATA[15:8], PAR[0], DATA[7:0]} is how the data and parity bits are written/read from memory
- Supports PAR_ERROR output on parity error detection

Note: AON memory has to be explicitly scrubbed (write 0) by ROM/PSE-FW before it can be used. There is no HW based scrubbing logic for the AON memory.

22.11 Timer

22.11.1 Functional Description

The Intel® PSE includes several timers and time-related functions, as shown in the following table:

Table 22-2. Timers

Timer	Clock Source Options	Description
HPET Periodic Timer	32 kHz	One of the timers available in the HPET. This generates an interrupt at a periodic rate. The Intel® PSE firmware can use this scheduling periodic events, such as a sensor that must be sampled every 20 milliseconds
HPET One Shot (two instances)	32 kHz	Two of the timers available in the HPET. Each can cause the HPET interrupt at a specific number of clocks in the future (countdown style).
Watchdog Timer	100 MHz	The Watchdog Timer generates an interrupt if the Intel® PSE firmware fails to reload the timer within the specific amount of time and generates a reset if the firmware fails to service the WDT interrupt after a specific time. This can detect some lockups in the firmware. The second catastrophic WDT timer interrupt is routed to PMC to potentially create a global reset, if deemed necessary. **Note:** This clock stops when the func_dk is gated to ARM.
Time Synchronization Local ART	19.2 MHz	Time Synchronization local copy of the global ART timer.
Real Time Clock	32 kHz	The Real Time Clock is used by software to find the number of clock ticks since the platform has booted, and may be used for timestamps of events and to track the passage of times. There are no controls to stop/reset/restart this timer and there are no provisions for interrupts associated with this counter. The RTC, however, is NOT within the Intel® PSE. ARM can access RTC via initiating Private Control Register read/writes to 0x70/71, by using the Sideband endpoint within the Intel® PSE.

22.12 I/O Ownership and Interrupts

- Many Intel Atom® x6000E Series processors, Intel® Pentium® and Celeron® N and J Series processors I/O interfaces can be owned by either the Arm* Cortex*-M7 or the Local Host (IA Processor) and interrupts need to be driven accordingly.

- Interrupts to Local Host (IA Processor) can be driven as assertIRQ/deassertIRQ messages on IOSF Sideband Fabric or as a MSI message.
- Interrupts from each I/O interface are appropriately qualified based on ownership and routed appropriately.
- For I/O interfaces that have multiple interrupts, a MMIO register must be programmed by the device driver to select the appropriate interrupts that need to be routed as MSI multi-message interrupt.
- Table 22-3 shows the Intel® PSE device ownership and interrupt routing IA mechanism, where OWNERSHIP_DEVx[2:0] bits devices the ownership (Intel® PSE or host). For example if OWNERSHIP_DEVx[2:0]=0b000, then device ownership is with Intel® PSE and interrupts are routed to CORTEX-M7 ARM NVIC. If OWNERSHIP_DEVx[2:0]=0b001 then device ownership is with the IA host. Also interrupts to the IA host can be further classified as MSI message or IRQ to APIC on the basis of INTR_DELIVERY_DEVx bit.
- Ownership and Interrupt Delivery bits are present in the Inter Process Communications (IPC) to the Local Host (IA Processor).
- A G3 power cycle is required when a change in ownership occurs. **Note:** This requirement does not apply to the network proxy function.

Table 22-3. Intel® PSE Interrupt Routing

Ownership_DEVx	Device Owner	INTR_DELIVERY_DEVx	Interrupt Destination	Interrupt Mechanism
0b000	Intel® PSE (ARM)	X	ARM	Wire to NVIC
0b001	IA	0b0	IA	MSI Message
		0b1	IA	IRQ to IOAPIC
0b010-0b111	NA	NA	NA	NA

Table 22-4. Intel® PSE ARM Interrupt And MSI Vector Mapping (Sheet 1 of 3)

Module Name	Interrupt Name	ARM IRQ No.	MSI Vector No.
IPC	LH channel_interrupt	0	5'b00000
	PMC channel_interrupt	2	N/A
DASHBOARD *	Interrupt	5	N/A
DMA	DMA0_Interrupt	11	5'b00000
	DMA1_Interrupt	12	5'b00000
	DMA2_Interrupt	13	5'b00000
FABRIC	Fabric_interrupt	14	N/A
GPIO	GPIO0_interrupt	15	5'b00000
	GPIO1_interrupt	16	5'b00000
TIME-AWARE GPIO	GPIO0_timed_interrupt	17	5'b00001
	GPIO1_timed_interrupt	18	5'b00001
HPET	HPET0_intr	19	N/A
	HPET1_intr	20	N/A
	HPET2_intr	21	N/A
SBEP *	SBEP_intr	22	N/A
SRAMC	SRAM Register Interrupt	23	N/A
	SRAM Correctable Error Interrupt	24	N/A
WDT	Watchdog Interrupt	25	N/A
I^2C	I^2C0_intr	26	5'b00000
	I^2C1_intr	27	5'b00000
	I^2C2_intr	28	5'b00000
	I^2C3_intr	29	5'b00000
	I^2C4_intr	30	5'b00000
	I^2C5_intr	31	5'b00000
	I^2C6_intr	32	5'b00000
	I^2C7_intr	33	5'b00000
SPI	SPI0_intr	34	5'b00000
	SPI1_intr	35	5'b00000
	SPI2_intr	36	5'b00000
	SPI3_intr	37	5'b00000
UART	UART0_intr	38	5'b00000
	UART1_intr	39	5'b00000

Table 22-4. Intel® PSE ARM Interrupt And MSI Vector Mapping (Sheet 2 of 3)

Module Name	Interrupt Name	ARM IRQ No.	MSI Vector No.
	UART2_intr	40	5'b00000
	UART3_intr	41	5'b00000
	UART4_intr	42	5'b00000
	UART5_intr	43	5'b00000
PWM	Timer0_intr	44	5'b00000
	Timer1_intr	45	5'b00001
QEP	QEP0_intr	46	5'b00000
	QEP1_intr	47	5'b00000
	QEP2_intr	48	5'b00000
	QEP3_intr	49	5'b00000
I^2S	I2S0_intr	51	5'b00000
	I2S1_intr	52	5'b00000
CAN	CAN0_int	53	5'b00000
	CAN1_int	54	5'b00000
GBE0	NETPROX_IRQ	55	5'b11000
	CE - Correctable Error Intr	56	5'b11011
	UE - Uncorrectable Error Intr	57	5'b11010
	PCS Interrupt	58	5'b11110
	MAC_IRQ	59	5'b11101
	Queue_TX0_IRQ	60	5'b00001
	Queue_TX1_IRQ	61	5'b00011
	Queue_TX2_IRQ	62	5'b00101
	Queue_TX3_IRQ	63	5'b00111
	Queue_TX4_IRQ	64	5'b01001
	Queue_TX5_IRQ	65	5'b01011
	Queue_TX6_IRQ	66	5'b01101
	Queue_TX7_IRQ	67	5'b01111
	Queue_RX0_IRQ	68	5'b00000
	Queue_RX1_IRQ	69	5'b00010
	Queue_RX2_IRQ	70	5'b00100
	Queue_RX3_IRQ	71	5'b00110
	Queue_RX4_IRQ	72	5'b01000
	Queue_RX5_IRQ	73	5'b01010
	Queue_RX6_IRQ	74	5'b01100
	Queue_RX7_IRQ	75	5'b01110

Table 22-4. Intel® PSE ARM Interrupt And MSI Vector Mapping (Sheet 3 of 3)

GBE1	NETPROX_IRQ	76	5'b11000
	CE - Correctable Error Intr	77	5'b11011
	UE - Uncorrectable Error Intr	78	5'b11010
	PCS Interrupt	79	5'b11110
	MAC_IRQ	80	5'b11101
	Queue_TX0_IRQ	81	5'b00001
	Queue_TX1_IRQ	82	5'b00011
	Queue_TX2_IRQ	83	5'b00101
	Queue_TX3_IRQ	84	5'b00111
	Queue_TX4_IRQ	85	5'b01001
	Queue_TX5_IRQ	86	5'b01011
	Queue_TX6_IRQ	87	5'b01101
	Queue_TX7_IRQ	88	5'b01111
	Queue_RX0_IRQ	89	5'b00000
	Queue_RX1_IRQ	90	5'b00010
	Queue_RX2_IRQ	91	5'b00100
	Queue_RX3_IRQ	92	5'b00110
	Queue_RX4_IRQ	93	5'b01000
	Queue_RX5_IRQ	94	5'b01010
	Queue_RX6_IRQ	95	5'b01100
	Queue_RX7_IRQ	96	5'b01110
PLL	PLL_lock	97	N/A
ARM	CTIIRQ[0]	98	N/A
ARM	CTIIRQ[1]	99	N/A
ARM	ICDET[0]	100	N/A
ARM	ICDET[2]	101	N/A
ARM	DCDET[0]	102	N/A
ARM	DCDET[2]	103	N/A
CRU *	CRU Interrupt	104	N/A
GBE0	LPI Interrupt	105	5'b11100
GBE1	LPI Interrupt	106	5'b11100

*DASHBOARD: Intel® PSE integrates dashboard block to support SIIP boot-flow. The dashboard block has set of registers that are used by BIOS to communicate with Intel® PSE ROM/FW for Intel® PSE boot.

*SBEP: Sideband Endpoint, allows messaging to/from peer agents like Boot Prep from PMC, Time Sync from ART

*CRU: Clock and Reset Unit, part of Clock Control Unit

22.13 Controller Area Network (CAN) Bus Controller

22.13.1 Overview

The CANBUS controller performs communication according to ISO 11898-1 (identical to Bosch CAN protocol specification 2.0 part A,B) and according to ISO 11898-4 (Time-triggered communication on CAN).

In addition the controller supports communication according to CAN FD protocol specification 1.0. The CAN FD option can be used together with event-triggered CAN communication.

The messages are stored in a parity protected RAM connected to the controller.

All functions concerning the handling of messages are implemented by the Rx Handler and the Tx Handler. The Rx Handler manages message acceptance filtering, the transfer of received messages from the controller to the Message RAM as well as providing receive message status information. The Tx Handler is responsible for the transfer of transmit messages from the Message RAM to the controller as well as providing transmit status information. It implements all functions concerning the time schedule and the global system time.

Acceptance filtering is implemented by a combination of up to 128 filter elements where each one can be configured as range, as a bit mask, or as a dedicated ID filter.

22.13.1.1 Ownership Allocation

- CAN0 ownership can be controlled using OWNERSHIP_DEV20 field in Ownership Control 2 register
- CAN1 ownership can be controlled using OWNERSHIP_DEV21 field in Ownership Control 2 register

22.13.2 Key Features

- 2 CAN nodes
  - — CAN FD supported (up to 64B message size for FD-long)
  - — Both CAN instances has a full sized message RAM
- Parity protection for the message RAMs, with error injection functionality
- CAN disable inputs to prevent access to all CSRs and message RAM
- Per CAN instances interrupt output
- CAN Error Logging
- AUTOSAR optimized
- SAE J1939 optimized
- Improved acceptance filtering
- Programmable loop-back test mode

22.13.3 Functional Description

22.13.3.1 Bit Rate Configuration

The CAN serial bit rate for each CAN instance is configured by software through the TTCAN_CSR.BTP (standard CAN bit time) and TTCAN_CSR.FBTP (CAN FD fast bit time) registers.

Max bit rate supported is 5Mbps.

22.13.3.2 Message RAM

The message RAM is used to store the CAN message filter elements, receive and transmit FIFO's and buffers, the transmit event FIFO and the TTCAN trigger memory. The CAN controller IP itself contains very little data storage, it uses the message RAM as its local storage.

The message RAM memory is organized in 32b words, with two extra bits (total of 34b per address location) for storing parity. The parity bits are not visible to software, but is instead under the control of the DBY_CAN parity logic (2.4).

The registers CTL_CSR.MSG_RAM_SIZE allow software to detect the size of the CAN[0/1] message RAM.

Figure below show the size and breakdown of the MSG_RAM allocation for CAN0/1.

Configuration of TTCAN_CSR registers space controls which region of the message RAM they use for which CAN function (i.e. RX_FIFO0, TX Buffers, and so on).

Table 22-5. Size and breakdown of the MSG_RAM allocation for CAN[0/1] message RAM

	Numb. elements	Size words	Bytes	Bits
11b FILTER	128	128	512	4096
29b FILTER	64	128	512	4096
RX FIFO 0	64	1152	4608	36864
RX FIFO 1	64	1152	4608	36864
RX BUFFER	64	1152	4608	36864
TX EVENT FIFO	32	64	256	2048
TX BUFFERS	32	576	2304	18432
TRIGGER MEM	64	128	512	4096
SUM		4480	17920	143360

22.13.4 Operating Modes

22.13.4.1 Software Initialization

Software initialization is started by setting bit CCCR.INIT, either by software or by a hardware reset, when an uncorrected bit error was detected in the Message RAM, or by going Bus_Off. While CCCR.INIT is set, message transfer from and to the CAN bus is stopped, the status of the CAN bus output is recessive (HIGH). The counters of the Error Management Logic EML are unchanged. Setting CCCR.INIT does not change any configuration register. Resetting CCCR.INIT finishes the software initialization.

Afterwards the Bit Stream Processor BSP synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (? Bus_Idle) before it can take part in bus activities and start the message transfer.

22.13.4.2 Normal Operation

The module default operating mode after hardware reset is event-driven CAN communication without time triggers (TTOCF.OM = “00”). It is required that both CCCR.INIT and CCCR.CCE are set before the TT Operation Mode can be changed.

Once the controller is initialized and CCCR.INIT is reset to zero, the controller synchronizes itself to the CAN bus and is ready for communication.

After passing the acceptance filtering, received messages including Message ID and DLC are stored into a dedicated Rx Buffer or into Rx FIFO 0 or Rx FIFO 1.

For messages to be transmitted dedicated Tx Buffers or a Tx FIFO can be initialized or updated. Automated transmission on reception of remote frames is not implemented.

22.13.4.3 CAN FD Operation

There are two variants in the CAN FD frame format, first the CAN FD frame without bit rate switching where the data field of a CAN frame may be longer than 8 bytes. The second variant is the CAN FD frame where control field, data field, and CRC field of a CAN frame are transmitted with a higher bit rate than the beginning and the end of the frame.

The CAN operation mode is enabled by programming CCCR.CME. In case CCCR.CME = “01” transmission of long CAN FD frames and reception of long and fast CAN FD frames is enabled. With CCCR.CME = “10”/”11” transmission and reception of long and fast CAN FD frames is enabled. CCCR.CME can only be changed while CCCR.INIT and CCCR.CCE are both set.

When initialization is left (CCCR.INIT set to '0'), the CAN FD protocol option is inactive, it has to be requested by writing to CCCR.CMR.

A mode change requested by writing to CCCR.CMR will be executed next time the CAN protocol controller FSM reaches idle phase between CAN frames. Upon this event CCCR.CMR is reset to “00” and the status flags CCCR.FDBS and CCCR.FDO are set accordingly. In case the requested CAN operation mode is not enabled, the value written to CCCR.CMR is retained until it is overwritten by the next mode change request. Default is CAN operation according to ISO11898-1.

It is not necessary to change the CAN operation mode after system startup. A mode change during CAN operation is only recommended under the following conditions:

- The failure rate in the CAN FD data phase is significant higher than in the CAN FD arbitration phase. In this case disable the CAN FD bit rate switching option for transmissions.
- During system startup all nodes are transmitting according to ISO11898-1 until it is verified that they are able to communicate in CAN FD format. If this is true, all nodes switch to CAN FD operation.
- End-of-line programming in case not all nodes are CAN FD capable. Non CAN FD nodes are held in silent mode until programming has completed. Then all nodes switch back to CAN communication according ISO11898-1.

When CCCR.CME != “00”, received CAN FD frames are interpreted according to the CAN FD Protocol Specification. The reserved bit in CAN frames with 11-bit identifiers and the first reserved bit in CAN frames with 29-bit identifiers will be decoded as EDL bit. EDL = recessive signifies a CAN FD frame, EDL = dominant signifies a standard CAN frame. In a CAN FD frame, the two bits following EDL, r0 and BRS, decide whether the bit rate inside of this CAN FD frame is switched. A CAN FD bit rate switch is signified by r0 = dominant and BRS = recessive. The coding of r0 = recessive is reserved for future expansion of the protocol.

Reception of CAN frames according to ISO 11898-1 is possible in all CAN operation modes.

The status bits CCCR.FDO and CCCR.FDBS indicate the format of transmitted frames. When CCCR.FDO is set, frames will be transmitted in CAN FD format with EDL = recessive. When both CCCR.FDO and CCCR.FDBS are set, frames will be transmitted in CAN FD format with bit rate switching and both bits EDL and BRS = recessive.

In the CAN FD format, the coding of the DLC differs from the standard CAN format. The DLC codes 0 to 8 have the same coding as in standard CAN, the codes 9 to 15, which in standard CAN all code a data field of 8 bytes, are coded according to the following table.

Table 22-6. Coding of DLS in CAN FD

DLC	9	10	11	12	13	14	15
Number of Data bytes	12	16	20	24	32	48	64

In CAN FD frames, the bit timing will be switched inside the frame, after the BRS (Bit Rate Switch) bit, if this bit is recessive. Before the BRS bit, in the CAN FD arbitration phase, the standard CAN bit timing is used as defined by the Bit Timing & Prescaler Register BTP. In the following CAN FD data phase, the fast CAN bit timing is used as defined by the Fast Bit Timing & Prescaler Register FBTP. The bit timing is switched back from the fast timing at the CRC delimiter or when an error is detected, whichever occurs first.

The maximum configurable bit rate in the CAN FD data phase depends on the CAN clock frequency. Example: with a CAN clock frequency of 20MHz and the shortest configurable bit time of 4 Time Quanta (TQ), the bit rate in the data phase is 5 Mbit/s.

In both data frame formats, CAN FD long and CAN FD fast, the value of the bit ESI (Error Status Indicator) is determined by the transmitter's error state at the start of the transmission. If the transmitter is error passive, ESI is transmitted recessive, else it is transmitted dominant.

22.13.4.4 Transceiver Delay Compensation

During the data phase of a CAN FD transmission only one node is transmitting, all others are receivers. The length of the bus line has no impact. When transmitting via pin CAN controller transmitter the protocol controller receives the transmitted data from its local CAN transceiver via pin CAN controller receiver. The received data is delayed by the CAN transceiver's loop delay. In case this delay is greater than TSEG1 (time segment before sample point), a bit error is detected. In order to enable a data phase bit time that is even shorter than the transceiver loop delay, the delay compensation is introduced. Without transceiver delay compensation, the bit rate in the data phase of a CAN FD frame is limited by the transceivers loop delay.

22.13.4.5 Description

The CAN FD protocol unit has implemented a delay compensation mechanism to compensate the CAN transceiver's loop delay, thereby enabling transmission with higher bit rates during the CAN FD data phase independent of the delay of a specific CAN transceiver.

Within each CAN FD frame, the transmitter measures the delay between the data transmitted at pin CAN controller transmitter and the data received at pin CAN controller receiver. The measurement is done once, at the falling edge of bit EDL to bit r0. The delay is measured in CAN clock frequency periods.

A secondary sample point position is calculated by adding a configurable transceiver delay compensation offset FBTP.TDCO to the measured transceiver delay. This transceiver delay compensation value TEST.TDCV is the sum of the measured transceiver delay and the transceiver delay compensation offset. The transceiver delay compensation offset is chosen to adjust the secondary sample point inside the bit time (e.g. half of the bit time in the data phase). The position of the secondary sample point is rounded down to the next integer number of time quanta t_q.

To check for bit errors during the data phase, the delayed transmit data is compared against the received data at the secondary sample point. If a bit error is detected at the secondary sample point, the transmitter will react to this bit error at the next following regular sample point. During arbitration phase the delay compensation is always disabled.

For the transceiver delay compensation the following boundary conditions have to be considered:

- The sum of the measured delay from CAN controller transmitter to CAN controller receiver and the configured transceiver delay compensation offset FBTP.TDCO has to be less than 3 bit times in the data phase.
- The sum of the measured delay from CAN controller transmitter to CAN controller receiver and the configured transceiver delay compensation offset FBTP.TDCO has to be less or equal 63 CAN controller clock periods. In case this sum exceeds 63 CAN controller clock periods, the maximum value of 63 periods is used for transceiver delay compensation.

The actual delay compensation value is monitored by reading TEST.TDCV.

22.13.4.6 Configuration and Status

Compensation for the transceiver loop delay by the CAN controller is enabled via FBTP.TDC. The transceiver delay compensation offset is configured via FBTP.TDCO. The actual delay compensation value applied by the CAN controller's protocol engine can be read from TEST.TDCV.

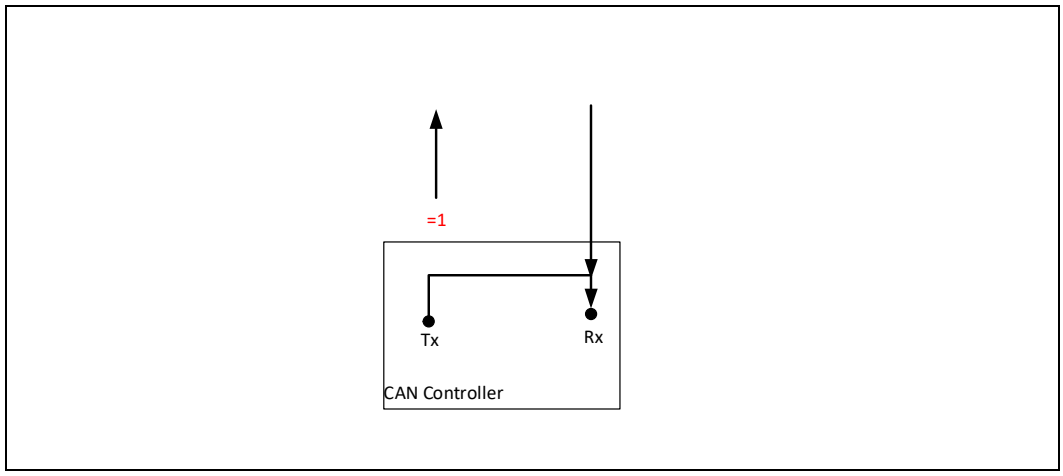
22.13.4.7 Bus monitoring mode

The CAN controller is set in Bus Monitoring Mode by programming CCCR.MON to one or when error level S3 (TTOST.EL = "11") is entered. In Bus Monitoring Mode (see ISO11898-1, 10.12 Bus monitoring), the CAN controller is able to receive valid data frames and valid remote frames, but cannot start a transmission. In this mode, it sends only recessive bits on the CAN bus, if the CAN controller is required to send a dominant

bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the CAN controller monitors this dominant bit, although the CAN bus may remain in recessive state. In Bus Monitoring Mode register TXBRP is held in reset state.

The Bus Monitoring Mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits. The following figure shows the connection of signals PSE_CANx_TX and PSE_CANx_RX to the CAN controller in Bus Monitoring Mode.

Figure 22-2. Bus monitoring mode



22.13.4.8 Disable Automatic Retransmission

According to the CAN Specification, the CAN controller provides means for automatic retransmission of frames that have lost arbitration or that have been disturbed by errors during transmission. By default automatic retransmission is enabled.

22.13.4.8.1 Frame Transmission in DAR Mode

In DAR mode all transmissions are automatically canceled after they started on the CAN bus. A Tx Buffer's Tx Request Pending bit TXBRP.TRPx is reset after successful transmission, when a transmission has not yet been started at the point of cancellation, has been aborted due to lost arbitration, or when an error occurred during frame transmission.

- Successful transmission:
  Corresponding Tx Buffer Transmission Occurred bit TXBTO.TOx set
  Corresponding Tx Buffer Cancellation Finished bit TXBCF.CFx not set
- Successful transmission in spite of cancellation:
  Corresponding Tx Buffer Transmission Occurred bit TXBTO.TOx set
  Corresponding Tx Buffer Cancellation Finished bit TXBCF.CFx set
- Arbitration lost or frame transmission disturbed:
  Corresponding Tx Buffer Transmission Occurred bit TXBTO.TOx not set
  Corresponding Tx Buffer Cancellation Finished bit TXBCF.CFx set

In case of a successful frame transmission, and if storage of Tx events is enabled, a Tx Event FIFO element is written with Event Type ET = "10" (transmission in spite of cancellation).

22.13.4.9 Test Modes

To enable write access to register TEST, bit CCCR.TEST has to be set to one. This allows the configuration of the test modes and test functions.

Four output functions are available for the CAN transmit pin PSE_CANx_TX by programming TEST.TX. Additionally to its default function - the serial data output - it can drive the CAN Sample Point signal to monitor the CAN controller's bit timing and it can drive constant dominant or recessive values. The actual value at pin PSE_CANx_RX can be read from TEST.RX. Both functions can be used to check the CAN bus' physical layer.

Due to the synchronization mechanism between CAN clock and Host clock domain, there may be a delay of several Host clock periods between writing to TEST.TX until the new configuration is visible at output pin PSE_CANx_TX. This applies also when reading input pin PSE_CANx_RX via TEST.RX.

Note: Test modes should be used for production tests or self test only. The software control for pin PSE_CANx_TX interferes with all CAN protocol functions. It is not recommended to use test modes for application.

External Loop Back Mode

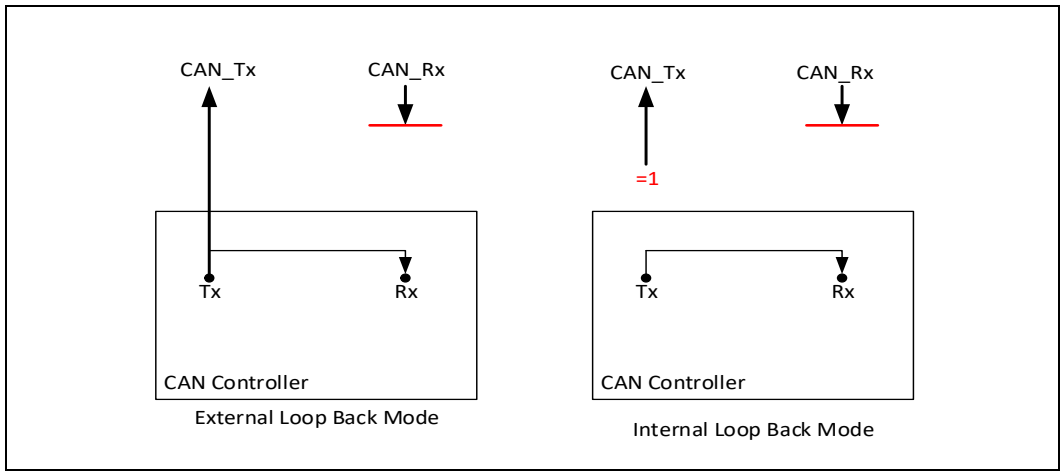
The CAN controller can be set in External Loop Back Mode by programming TEST.LBCK to one. In Loop Back Mode, the CAN controller treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into an Rx Buffer or an Rx FIFO. Figure 22-3 shows the connection of signals PSE_CANx_TX and PSE_CANx_RX to the CAN controller in External Loop Back Mode.

This mode is provided for hardware self-test. To be independent from external stimulation, the CAN controller ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/ remote frame) in Loop Back Mode. In this mode the CAN controller performs an internal feedback from its Tx output to its Rx input. The actual value of the PSE_CANx_RX input pin is disregarded by the CAN controller. The transmitted messages can be monitored at the PSE_CANx_TX pin.

Internal Loop Back Mode

Internal Loop Back Mode is entered by programming bits TEST.LBCK and CCCR.MON to one. This mode can be used for a "Hot Selftest", meaning the CAN controller can be tested without affecting a running CAN system connected to the pins PSE_CANx_TX and PSE_CANx_RX. In this mode pin PSE_CANx_RX is disconnected from the CAN controller and pin PSE_CANx_TX is held recessive (held high). The following figure shows the connection of PSE_CANx_TX and PSE_CANx_RX to the CAN controller in case of Internal Loop Back Mode.

Figure 22-3. Internal Loop Back Mode



22.13.5 Rx Handling

The Rx Handler controls the acceptance filtering, the transfer of received messages to the Rx Buffers or to one of the two Rx FIFOs, as well as the Rx FIFO's Put and Get Indices.

22.13.5.1 Acceptance Filtering

The CAN controller offers the possibility to configure two sets of acceptance filters, one for standard identifiers and one for extended identifiers. These filters can be assigned to an Rx Buffer or to Rx FIFO 0,1. For acceptance filtering each list of filters is executed from element #0 until the first matching element. Acceptance filtering stops at the first matching element. The following filter elements are not evaluated for this message.

The main features are:

- Each filter element can be configured as
  - — range filter (from - to)
  - — filter for one or two dedicated IDs
  - — classics bit mask filter
- Each filter element is configurable for acceptance or rejection filtering
- Each filter element can be enabled / disabled individually
- Filters are checked sequentially, execution stops with the first matching filter element

Related configuration registers are:

- Global Filter Configuration GFC
- Standard ID Filter Configuration SIDFC
- Extended ID Filter Configuration XIDFC
- Extended ID AND Mask XIDAM

Depending on the configuration of the filter element (SFEC/EFEC) a match triggers one of the following actions:

- Store received frame in FIFO 0 or FIFO 1
- Store received frame in Rx Buffer
- Store received frame in Rx Buffer and generate pulse at filter event pin
- Reject received frame
- Set High Priority Message interrupt flag IR.HPM
- Set High Priority Message interrupt flag IR.HPM and store received frame in FIFO 0 or FIFO 1

Acceptance filtering is started after the complete identifier has been received. After acceptance filtering has completed, and if a matching Rx Buffer or Rx FIFO has been found, the Message Handler starts writing the received message data in portions of 32 bit to the matching Rx Buffer or Rx FIFO. If the CAN protocol controller has detected an error condition (e.g. CRC error), this message is discarded with the following impact on the affected Rx Buffer or Rx FIFO:

Rx Buffer

New Data flag of matching Rx Buffer is not set, but Rx Buffer (partly) overwritten with received data. For error type see PSR.LEC respectively PSR.FLEC.

Rx FIFO

Put index of matching Rx FIFO is not updated, but related Rx FIFO element (partly) overwritten with received data. For error type see PSR.LEC respectively PSR.FLEC. In case the matching Rx FIFO is operated in overwrite mode, the boundary conditions described in Section 22.13.5.2.2 have to be considered.

Note: When an accepted message is written to one of the two Rx FIFOs, or into an Rx Buffer, the unmodified received identifier is stored independent of the filter(s) used. The result of the acceptance filter process is strongly depending on the sequence of configured filter elements.

22.13.5.1.1 Range Filter

The filter matches for all received frames with Message IDs in the range defined by SF1ID/SF2ID resp. EF1ID/EF2ID.

There are two possibilities when range filtering is used together with extended frames:

EFT = “00”: The Message ID of received frames is ANDed with the Extended ID AND Mask (XIDAM) before the range filter is applied

EFT = “11”: The Extended ID AND Mask (XIDAM) is not used for range filtering

22.13.5.1.2 Filter for specific IDs

A filter element can be configured to filter for one or two specific Message IDs. To filter for one specific Message ID, the filter element has to be configured with SF1ID = SF2ID resp. EF1ID = EF2ID.

22.13.5.1.3 Classic Bit Mask Filter

Classic bit mask filtering is intended to filter groups of Message IDs by masking single bits of a received Message ID.With classic bit mask filtering SF1ID/EF1ID is used as Message ID filter, while SF2ID/EF2ID is used as filter mask.

A zero bit at the filter mask will mask out the corresponding bit position of the configured ID filter, e.g. the value of the received Message ID at that bit position is not relevant for acceptance filtering. Only those bits of the received Message ID where the corresponding mask bits are one are relevant for acceptance filtering.

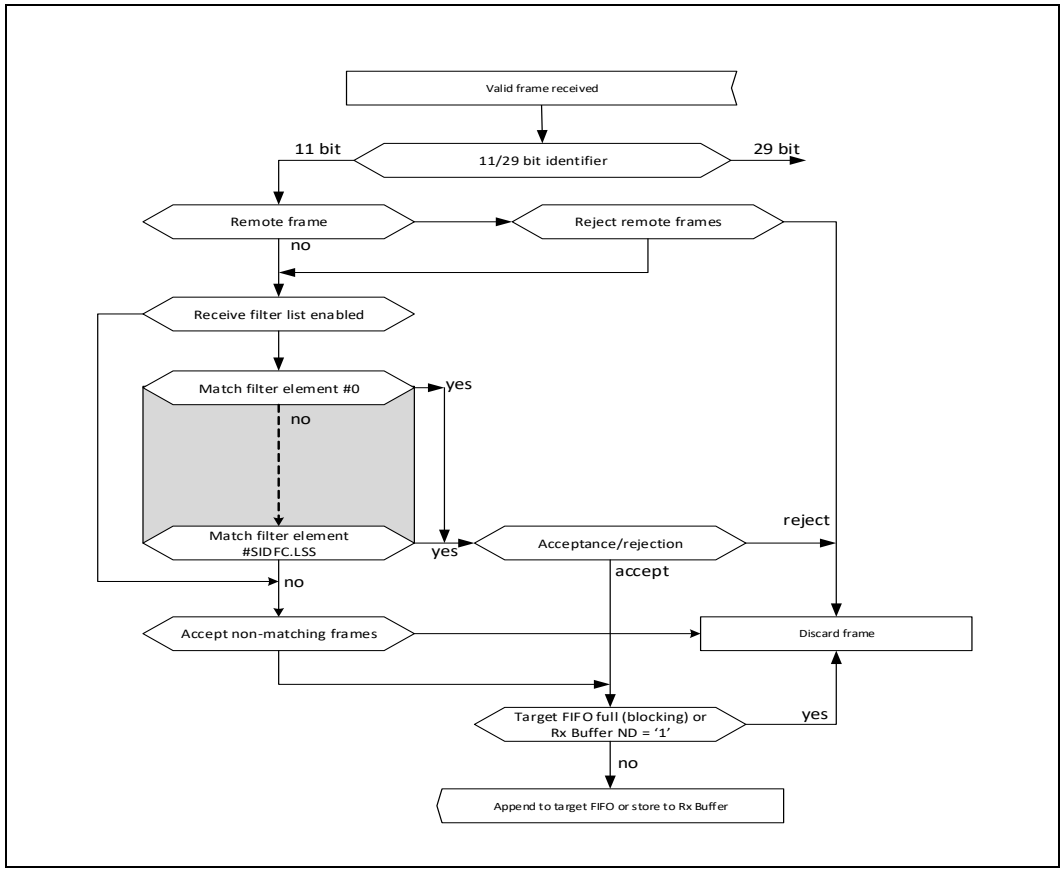
In case all mask bits are one, a match occurs only when the received Message ID and the Message ID filter are identical. If all mask bits are zero, all Message IDs match.

22.13.5.1.4 Standard Message ID Filtering

Figure 22-4 shows the flow for standard Message ID (11-bit Identifier) filtering. The Standard Message ID Filter element is described in Datasheet Volume 2, (Book 3).

Controlled by the Global Filter Configuration GFC and the Standard ID Filter Configuration SIDFC Message ID, Remote Transmission Request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements.

Figure 22-4. Standard Message ID Filter Path



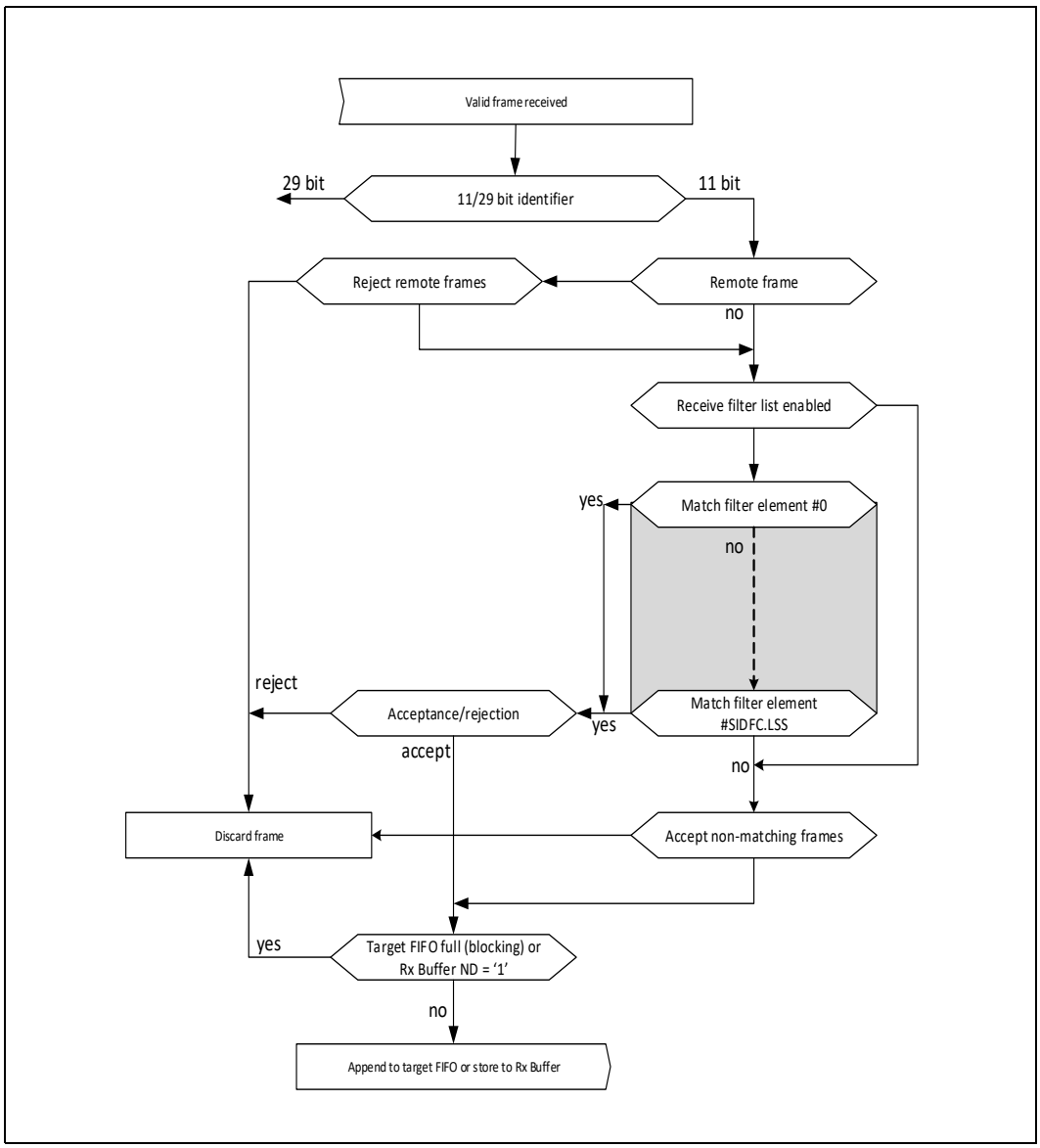
22.13.5.1.5 Extended Message ID Filtering

Figure 22-5 shows the flow for extended Message ID (29-bit Identifier) filtering. The Extended Message ID Filter element is described in Datasheet Volume 2, (Book 3).

Controlled by the Global Filter Configuration GFC and the Extended ID Filter Configuration XIDFC Message ID, Remote Transmission Request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements.

The Extended ID AND Mask XIDAM is ANDed with the received identifier before the filter list is executed.

Figure 22-5. Extended Message ID Filtering



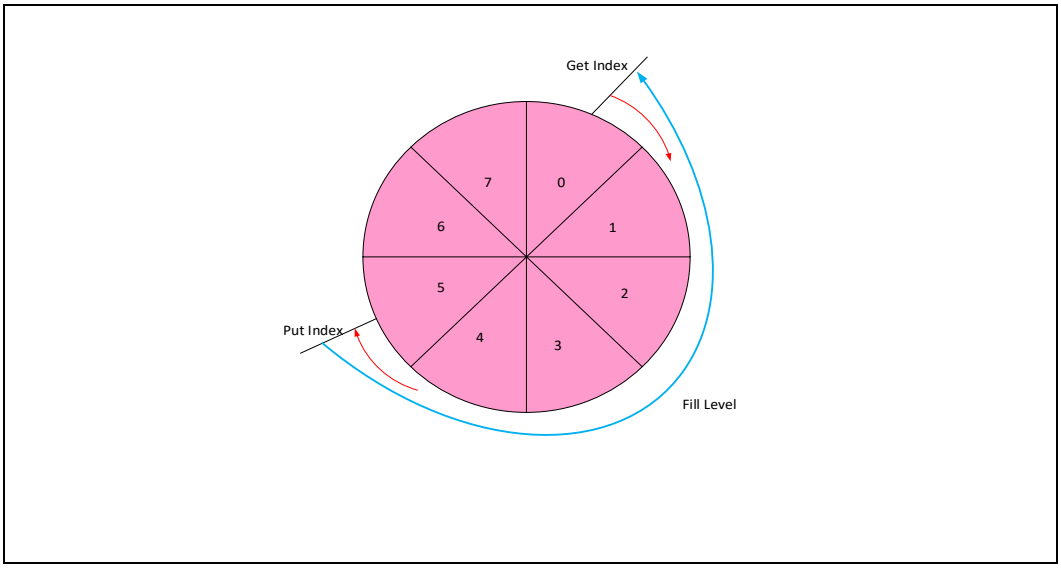
22.13.5.2 Rx FIFOs

Rx FIFO 0 and Rx FIFO 1 can be configured to hold up to 64 elements each. Configuration of the two Rx FIFOs is done via registers RXF0C and RXF1C.

Received messages that passed acceptance filtering are transferred to the Rx FIFO as configured by the matching filter element. For a description of the filter mechanisms available for Rx FIFO 0 and Rx FIFO 1, see Section 22.13.5.1. The Rx FIFO element is described in Datasheet Volume 2, (Book 3).

To avoid an Rx FIFO overflow, the Rx FIFO watermark can be used. When the Rx FIFO fill level reaches the Rx FIFO watermark configured by RXFnC.FnWM, interrupt flag IR.RFnW is set. When the Rx FIFO Put Index reaches the Rx FIFO Get Index an Rx FIFO Full condition is signalled by RXFnS.FnF. In addition interrupt flag IR.RFnF is set.

Figure 22-6. Rx FIFO Status



When reading from an Rx FIFO, Rx FIFO Get Index RXFnS.FnGI • FIFO Element Size has to be added to the corresponding Rx FIFO start address RXFnC.FnSA.

Table 22-7. Rx buffer/FIFO Element Size

RXESC.RBDS[2:0] RXESC.FnDS[2:0]	Data Field [bytes]	FIFO Element Size [RAM words]
000	8	4
001	12	5
010	16	6
011	20	7
100	24	8
101	32	10
110	48	14
111	64	18

22.13.5.2.1 Rx FIFO Blocking Mode

The Rx FIFO blocking mode is configured by RXFnC.FnOM = '0'. This is the default operation mode for the Rx FIFOs.

When an Rx FIFO full condition is reached (RXFnS.FnPI = RXFnS.FnGI), no further messages are written to the corresponding Rx FIFO until at least one message has been read out and the Rx FIFO Get Index has been incremented. An Rx FIFO full condition is signalled by RXFnS.FnF = '1'. In addition interrupt flag IR.RFnF is set.

In case a message is received while the corresponding Rx FIFO is full, this message is discarded and the message lost condition is signalled by RXFnS.RFnL = '1'. In addition interrupt flag IR.RFnL is set.

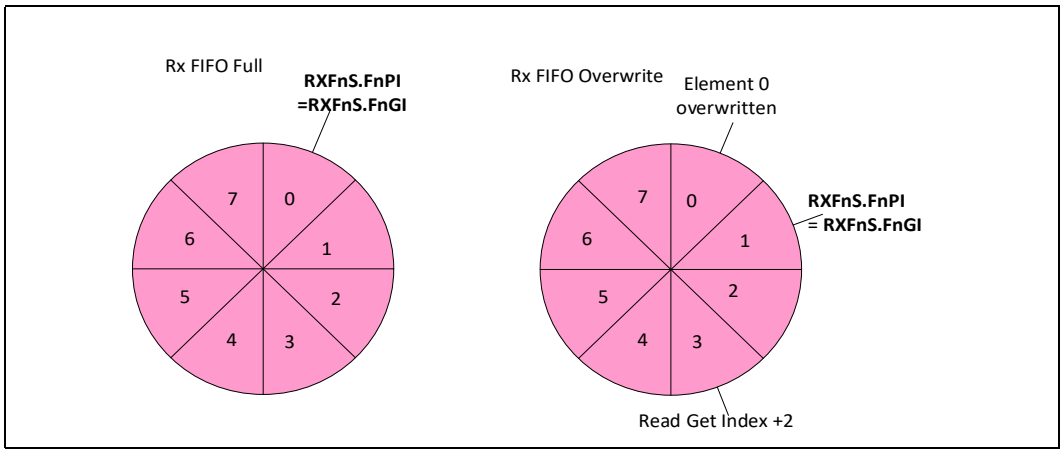
22.13.5.2.2 Rx FIFO Overwrite Mode

The Rx FIFO overwrite mode is configured by RXFnC.FnOM = '1'.

When an Rx FIFO full condition (RXFnS.FnPI = RXFnS.FnGI) is signalled by RXFnS.FnF = '1', the next message accepted for the FIFO will overwrite the oldest FIFO message. Put and get index are both incremented by one.

When an Rx FIFO is operated in overwrite mode and an Rx FIFO full condition is signalled, reading of the Rx FIFO elements should start at least at get index + 1. The reason for that is, that it might happen, that a received message is written to the Message RAM (put index) while the CPU is reading from the Message RAM (get index). In this case inconsistent data may be read from the respective Rx FIFO element. Adding an offset to the get index when reading from the Rx FIFO avoids this problem. The offset depends on how fast the CPU accesses the Rx FIFO. The following figure shows an offset of two with respect to the get index when reading the Rx FIFO. In this case the two messages stored in element 1 and 2 are lost.

Figure 22-7. Rx FIFO Overflow Handling



After reading from the Rx FIFO, the number of the last element read has to be written to the Rx FIFO Acknowledge Index RXFnA.FnA. This increments the get index to that element number. In case the put index has not been incremented to this Rx FIFO element, the Rx FIFO full condition is reset (RXFnS.FnF = '0').

22.13.5.3 Dedicated Rx Buffers

The CAN controller supports up to 64 dedicated Rx Buffers. The start address of the dedicated Rx Buffer section is configured via RXBC.RBSA.

For each Rx Buffer a Standard or Extended Message ID Filter Element with SFEC / EFEC = "111" and SFID2 / EFID2[10:9] = "00" has to be configured.

After a received message has been accepted by a filter element, the message is stored into the Rx Buffer in the Message RAM referenced by the filter element. The format is the same as for an Rx FIFO element. In addition the flag IR.DRX (Message stored in Dedicated Rx Buffer) in the interrupt register is set.

Table 22-8. Example Filter Configuration for Rx Buffers

Filter Element	SFID1[10:0] EFID1[28:0]	SFID2[10:9] EFID2[10:9]	SFID2[5:0] EFID2[5:0]
0	ID message 1	00	00 0000
1	ID message 2	00	00 0001
2	ID message 3	00	00 0010

After the last word of a matching received message has been written to the Message RAM, the respective New Data flag in register NDAT1,2 is set. As long as the New Data flag is set, the respective Rx Buffer is locked against updates from received matching frames. The New Data flags have to be reset by the Host by writing a '1' to the respective bit position.

While an Rx Buffer's New Data flag is set, a Message ID Filter Element referencing this specific Rx Buffer will not match, causing the acceptance filtering to continue. Following Message ID Filter Elements may cause the received message to be stored into another Rx Buffer, or into an Rx FIFO, or the message may be rejected, depending on filter configuration.

22.13.5.3.1 Rx Buffer Handling

- Reset interrupt flag IR.DRX
- Read New Data registers
- Read messages from Message RAM
- Reset New Data flags of processed messages

22.13.6 Tx Handling

The Tx Handler handles transmission requests for the dedicated Tx Buffers and the Tx FIFO. It controls the transfer of transmit messages to the CAN Core, the Put and Get Indices, and the Tx Event FIFO. Up to 32 Tx Buffers can be set up for message transmission. The Tx Buffer element is described in Datasheet Volume 2, (Book 3).

The Tx Handler starts a Tx scan to check for the highest priority pending Tx request (Tx Buffer with lowest Message ID) when the Tx Buffer Request Pending register TXBRP is updated, or when a transmission has been started.

22.13.6.1 Transmit Pause

The transmit pause feature is intended for use in CAN systems where the CAN message identifiers are (permanently) specified to specific values and cannot easily be changed. These message identifiers may have a higher CAN arbitration priority than other defined messages, while in a specific application their relative arbitration priority should be inverse. This may lead to a case where one ECU sends a burst of CAN messages that cause another ECU's CAN messages to be delayed because that other messages have a lower CAN arbitration priority.

If e.g. CAN ECU-1 has the transmit pause feature enabled and is requested by its application software to transmit four messages, it will, after the first successful message transmission, wait for two CAN bit times of bus idle before it is allowed to start the next requested message. If there are other ECUs with pending messages, those messages are started in the idle time, they would not need to arbitrate with the next message of ECU-1. After having received a message, ECU-1 is allowed to start its next transmission as soon as the received message releases the CAN bus.

The transmit pause feature is controlled by bit CCCR.TXP. If the bit is set, the CAN controller will, each time it has successfully transmitted a message, pause for two CAN bit times before starting the next transmission. This enables other CAN nodes in the network to transmit messages even if their messages have lower prior identifiers. Default is transmit pause disabled (CCCR.TXP = '0').

This feature looses up burst transmissions coming from a single node and it protects against "babbling idiot" scenarios where the application program erroneously requests too many transmissions.

22.13.6.2 Dedicated Tx buffers

Dedicated Tx Buffers are intended for message transmission under complete control of the Host CPU. Each Dedicated Tx Buffer is configured with a specific Message ID. In case that multiple Tx Buffers are configured with the same Message ID, the Tx Buffer with the lowest buffer number is transmitted first.

If the data section has been updated, a transmission is requested by an Add Request via TXBAR.ARn. The requested messages arbitrate internally with messages from an optional Tx FIFO and externally with messages on the CAN bus, and are sent out according to their Message ID.

A Dedicated Tx Buffer allocates Element Size 32-bit words in the Message RAM (see Table 22-10). Therefore the start address of a dedicated Tx Buffer in the Message RAM is calculated by adding transmit buffer index (0…31) • Element Size to the Tx Buffer Start Address TXBC.TBSA.

Table 22-9. Tx Buffer/FIFO Element Size

TXESC.TBDS[2:0]	Data Field [bytes]	Element Size [RAM words]
000	8	4
001	12	5
010	16	6
011	20	7
100	24	8

Table 22-9. Tx Buffer/FIFO Element Size

TXESC.TBDS[2:0]	Data Field [bytes]	Element Size [RAM words]
101	32	10
110	48	14
111	64	18

22.13.6.3 Tx FIFO

Tx FIFO operation is configured by programming TXBC.TFQM to '0'. Messages stored in the Tx FIFO are transmitted starting with the message referenced by the Get Index TXFQS.TFGI. After each transmission the Get Index is incremented cyclically until the Tx FIFO is empty. The Tx FIFO enables transmission of messages with the same Message ID from different Tx Buffers in the order these messages have been written to the Tx FIFO. The CAN controller calculates the Tx FIFO Free Level TXFQS.TFFL as difference between Get and Put Index. It indicates the number of available (free) Tx FIFO elements.

New transmit messages have to be written to the Tx FIFO starting with the Tx Buffer referenced by the Put Index TXFQS.TFQPI. An Add Request increments the Put Index to the next free Tx FIFO element. When the Put Index reaches the Get Index, Tx FIFO Full (TXFQS.TFQF = '1') is signalled.

In this case no further messages should be written to the Tx FIFO until the next message has been transmitted and the Get Index has been incremented.

When a single message is added to the Tx FIFO, the transmission is requested by writing a '1' to the TXBAR bit related to the Tx Buffer referenced by the Tx FIFO's Put Index.

When multiple (n) messages are added to the Tx FIFO, they are written to n consecutive Tx Buffers starting with the Put Index. The transmissions are then requested via TXBAR. The Put Index is then cyclically incremented by n. The number of requested Tx buffers should not exceed the number of free Tx Buffers as indicated by the Tx FIFO Free Level.

When a transmission request for the Tx Buffer referenced by the Get Index is cancelled, the Get Index is incremented to the next Tx Buffer with pending transmission request and the Tx FIFO Free Level is recalculated. When transmission cancellation is applied to any other Tx Buffer, the Get Index and the FIFO Free Level remain unchanged.

A Tx FIFO element allocates Element Size 32-bit words in the Message RAM (see Table 22-10). Therefore the start address of the next available (free) Tx FIFO Buffer is calculated by adding Tx FIFO/Queue Put Index TXFQS.TFQPI (0…31) • Element Size to the Tx Buffer Start Address TXBC.TBSA.

22.13.6.4 Transmit Cancellation

The CAN controller supports transmit cancellation. This feature is especially intended for gateway applications and AUTOSAR based applications. To cancel a requested transmission from a Dedicated Tx Buffer the Host has to write a '1' to the corresponding bit position (=number of Tx Buffer) of register TXBCR. Transmit cancellation is not intended for Tx FIFO operation.

Successful cancellation is signalled by setting the corresponding bit of register TXBCF to '1'.

In case a transmit cancellation is requested while a transmission from a Tx Buffer is already ongoing, the corresponding TXBRP bit remains set as long as the transmission is in progress. If the transmission was successful, the corresponding TXBTO and TXBCF bits are set. If the transmission was not successful, it is not repeated and only the corresponding TXBCF bit is set.

Note: In case a pending transmission is canceled immediately before this transmission could have been started, there follows a short time window where no transmission is started even if another message is also pending in this node. This may enable another node to transmit a message which may have a lower priority than the second message in this node.

22.13.6.5 Tx Event Handling

To support Tx event handling the CAN controller has implemented a Tx Event FIFO. After the CAN controller has transmitted a message on the CAN bus, Message ID and timestamp are stored in a Tx Event FIFO element. To link a Tx event to a Tx Event FIFO element, the Message Marker from the transmitted Tx Buffer is copied into the Tx Event FIFO element.

The Tx Event FIFO can be configured to a maximum of 32 elements. The Tx Event FIFO element is described in Datasheet Volume 2, (Book 3).

When a Tx Event FIFO full condition is signalled by IR.TEFF, no further elements are written to the Tx Event FIFO until at least one element has been read out and the Tx Event FIFO Get Index has been incremented. In case a Tx event occurs while the Tx Event FIFO is full, this event is discarded and interrupt flag IR.TEFL is set.

To avoid a Tx Event FIFO overflow, the Tx Event FIFO watermark can be used. When the Tx Event FIFO fill level reaches the Tx Event FIFO watermark configured by TXEFC.EFWM, interrupt flag IR.TEFW is set.

When reading from the Tx Event FIFO, two times the Tx Event FIFO Get Index TXEFS.EFGI has to be added to the Tx Event FIFO start address TXEFC.EFSA.

22.13.7 CAN Cross-Timestamping Flows

CANBUS IP supports cross-timestamping of its local timer (synchronized on CAN network) with ART.

Software sets the TTCAN_TIMESTAMP_CTL[CAPTURE_LXTS] bit that triggers the cross-timestamping logic. The SWT pin of the CAN IP is also triggered with this pulse.

Software must poll for TTCAN_TIMESTAMP_CTL[LXTS_VALID] bit to be set which indicates cross-timestamp is complete.

ART Snapshot can be read from TTCAN_LOCAL_TIMESTAMP_LOW/HIGH registers. The TTCAN local timer snapshot is captured in the TTCPT (TT capture time) register.

22.13.8 Signal Description

Table 22-10. CANBUS Signal

Signal Name	Type	Description
PSE_CAN0_TX PSE_CAN1_TX	O	CAN Transmit
PSE_CAN0_RX PSE_CAN1_RX	I	CAN Receive

22.13.9 Registers

Please refer to chapter 7 of the Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors for Internet of Things (IoT) Applications, Datasheet, Volume 2 (Book 3 of 3), Intel® Programmable Services Engine (Intel® PSE) (Document Number: 636723), for a description of the registers associated with subject of this chapter.

22.14 I^2C Controller

22.14.1 Overview

I^2C is a two-wire, bi-directional serial bus that provides simple and efficient method of data transmission over a short distance between many devices. I^2C is used typically for connecting the Intel® PSE to external sensor devices, such as accelerometers, gyroscopes, ambient light sensors, and so on.

There are 8 instances of I^2C controllers. These controllers are completely independent of each other: they do not share any pins, memory spaces, or interrupts. They can be independently clock gated.

The Intel® PSE I^2C host controllers share the same general specifications:

- Initiator Mode Only (all peripherals must be target devices)
- Support for the following operating speeds:
  - Standard mode:100 kbps
  - Fast Mode: 400 kbps
  - Fast Mode Plus:1000 kbps
  - High Speed Mode:3400 kbps with max. 100pf load
- Arbitration and clock synchronization
- Support for both 7-bit and 10-bit addressing formats on the I^2C bus
- FIFO of 64 bytes with programmable watermarks/thresholds
- DMA HW hook for Tx/Rx FIFO fill/drain.

22.14.1.1 Ownership Allocation

- I^2C0 ownership can be controlled using OWNERSHIP_DEV0 field in Ownership Control 0 register

- I^{2}C1 ownership can be controlled using OWNERSHIP_DEV1 field in Ownership Control 0 register
- I^{2}C2 ownership can be controlled using OWNERSHIP_DEV2 field in Ownership Control 0 register
- I^{2}C3 ownership can be controlled using OWNERSHIP_DEV3 field in Ownership Control 0 register
- I^{2}C4 ownership can be controlled using OWNERSHIP_DEV4 field in Ownership Control 0 register
- I^{2}C5 ownership can be controlled using OWNERSHIP_DEV5 field in Ownership Control 0 register
- I^{2}C6 ownership can be controlled using OWNERSHIP_DEV6 field in Ownership Control 0 register
- I^{2}C7 ownership can be controlled using OWNERSHIP_DEV7 field in Ownership Control 0 register

22.14.2 Features

I2C Controller have the following features:

- Two-wire I2C serial interface C consists of a serial data line (SDA) and a serial clock (SCL)
- Three speeds:
  - — Standard mode (0 to 100 Kb/s)
  - — Fast mode (≤ 400 Kb/s) or fast mode plus (≤ 1000 Kb/s)1
  - — High-speed mode (≤ 3.4 Mb/s)
- Clock synchronization
- Initiator I2C operation
- 7- or 10-bit addressing
- 7- or 10-bit combined format transfers
- Bulk transmit mode
- Ignores CBUS addresses (an older ancestor of I2C that used to share the I2C bus)
- Transmit and receive buffers
- Interrupt or polled-mode operation
- Handles Bit and Byte waiting at all bus speeds
- Simple software interface consistent with DesignWare APB peripherals
- Component parameters for configurable software driver support
- DMA handshaking interface compatible with the DMA Controller handshaking interface
- Programmable SDA hold time (tHD;DAT)
- Bus clear feature
- Device ID feature
- SMBus/PMBus support
- SMBus Target detects and responds to ARP commands.

- Ultra-Fast mode support
- UDID feature support

The I2C Controller requires external hardware components as support in order to be compliant in an I2C system.

22.14.3 Functional Description

The I^2C bus is a two-wire serial interface, consisting of a serial data line (SDA) and a serial clock (SCL). These wires carry information between the devices connected to the bus. Each device is recognized by a unique address and can operate as either a "transmitter" or "receiver," depending on the function of the device. Devices can also be considered as initiators or targets when performing data transfers. An initiator is a device that starts a data transfer on the bus and generates the clock signals to permit that transfer. At that time, any device addressed is considered a target.

The I^2C controller module can operate in standard mode (with data rates 0 to 100 Kb/s), fast mode (with data rates less than or equal to 400 Kb/s), fast mode plus (with data rates less than or equal to 1000 Kb/s), and high-speed mode (with data rates less than or equal to 3.4 Mb/s).

The I^2C controller can communicate with devices only of these modes as long as they are attached to the bus. Additionally, high-speed mode and fast mode devices are downward compatible. For instance, high-speed mode devices can communicate with fast mode and standard mode devices in a mixed-speed bus system; fast mode devices can communicate with standard mode devices in 0 to 100 Kb/s I^2C bus system. However:

- Standard mode devices are not upward compatible and should not be incorporated in a fast-mode I^2C bus system as they cannot follow the higher transfer rate and unpredictable states would occur.

An example of high-speed mode devices are LCD displays and high capacity EEPROMs. These devices typically need to transfer large amounts of data. Most maintenance and control applications, the common use for the I²C bus, typically operate at 100 kHz (in standard and fast modes).

An example of high speed mode devices are LED controllers and other devices that do not need feedback. These devices typically need to transfer large amounts of data greater than 1Mhz.

Any I^2C Controller device can be attached to an I²C-bus and every device can talk with any initiator, passing information back and forth. There needs to be at least one initiator on the bus but there can be multiple initiators, which require them to arbitrate for ownership. Multiple initiators and arbitration are explained later in this chapter.

22.14.4 I^2C Behavior

I^2C controller can be initiator only. The initiator is responsible for generating the clock and controlling the transfer of data. The target is responsible for either transmitting or receiving data to/from the initiator. The acknowledgment of data is sent by the device that is receiving data, which can be either an initiator or a target. As mentioned previously, the I^2C protocol also allows multiple initiators to reside on the I^2C bus and uses an arbitration procedure to determine bus ownership.

Each target has a unique address that is determined by the system designer. When an initiator wants to communicate with a target, the initiator transmits a START/RESTART condition that is then followed by the target's address and a control bit (R/W) to determine if the initiator wants to transmit data or receive data from the target. The target then sends an acknowledge (ACK) pulse after the address.

If the initiator (initiator-transmitter) is writing to the target (target-receiver), the receiver gets one byte of data. This transaction continues until the initiator terminates the transmission with a STOP condition. If the initiator is reading from a target (initiator-receiver), the target transmits (target-transmitter) a byte of data to the initiator, and the initiator then acknowledges the transaction with the ACK pulse. This transaction continues until the initiator terminates the transmission by not acknowledging (NACK) the transaction after the last byte is received, and then the initiator issues a STOP condition or addresses another target after issuing a RESTART condition.

The I^2C controller is a synchronous serial interface. The SDA line is a bidirectional signal and changes only while the SCL line is low, except for STOP, START, and RESTART conditions. The output drivers are open-drain or open-collector to perform wire-AND functions on the bus. The maximum number of devices on the bus is limited by only the maximum capacitance specification of 400 pF. Data is transmitted in byte packages.

22.14.5 I^2C Protocol

22.14.5.1 Start and Stop Condition

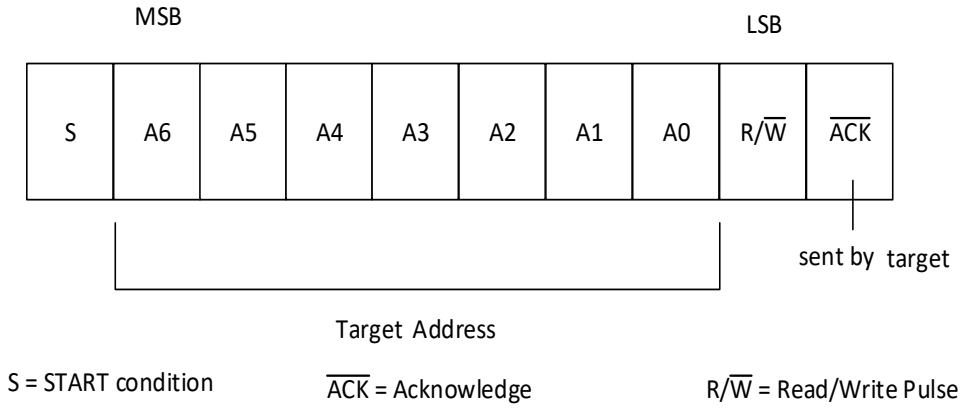
When the bus is idle, both the SCL and SDA signals are pulled high through external pull-up resistors on the bus. When the initiator wants to start a transmission on the bus, the initiator issues a START condition. This is defined to be a high-to-low transition of the SDA signal while SCL is 1. When the initiator wants to terminate the transmission, the initiator issues a STOP condition. This is defined to be a low-to-high transition of the SDA line while SCL is 1.

22.14.5.2 Addressing Target Protocol

22.14.5.2.1 7-bit Address Format

During the 7-bit address format, the first seven bits (bits 7:1) of the first byte set the target address and the LSB bit (bit 0) is the R/W bit as shown in the following figure. When bit 0 (R/W) is set to 0, the initiator writes to the target. When bit 0 (R/W) is set to 1, the initiator reads from the target.

Figure 22-8. 7-bit Address Format



22.14.5.2.2 10-bit Address Format

During 10-bit addressing, two bytes are transferred to set the 10-bit address. The transfer of the first byte contains the following bit definition. The first five bits (bits 7:3) notify the targets that this is a 10-bit transfer followed by the next two bits (bits 2:1), which set the targets address bits 9:8, and the LSB bit (bit 0) is the R/W bit. The second byte transferred sets bits 7:0 of the target address. The following figure shows the 10-bit address format.

Figure 22-9. 10-bit address format

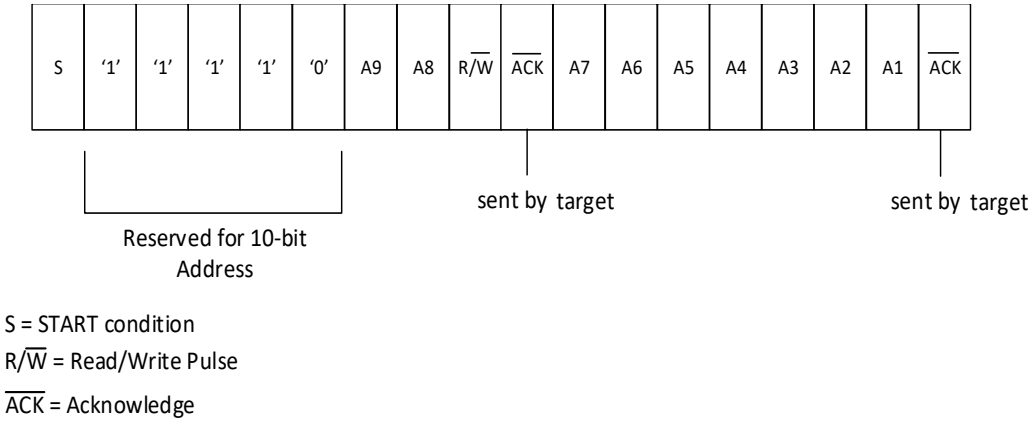


Table 22-11. I²C Definition of Bits in First Byte

Target Address	R/W Bit	Description
0000 000	0	General Call Address. I²C controller places the data in the receive buffer and issues a General Call interrupt.
0000 000	1	START byte

Table 22-11. I²C Definition of Bits in First Byte

Target Address	R/W Bit	Description
0000 001	x	CBUS address.
0000 010	x	Reserved.
0000 011	x	Reserved.
0000 1XX	x	High-speed initiator code
1111 1XX	x	Reserved.
1111 0XX	x	10-bit target addressing

I²C Controller does not restrict you from using these reserved addresses. However, if you use these reserved addresses, you may run into incompatibilities with other I²C components.

22.14.5.3 Transmitting and Receiving Protocol

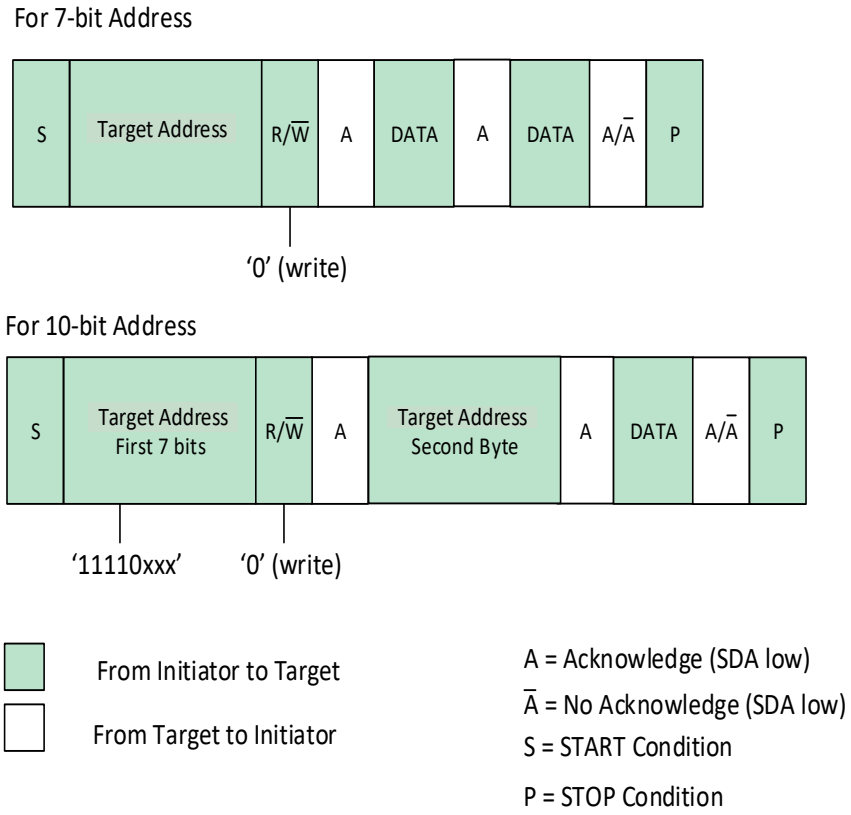
The initiator can start data transmission and reception to/from the bus, acting as either a initiator-transmitter or initiator-receiver. A target responds to requests from the initiator to either transmit data or receive data to/from the bus, acting as either a target-transmitter or target-receiver, respectively.

22.14.5.3.1 Initiator-Transmitter and Target-Receiver

All data is transmitted in byte format, with no limit on the number of bytes transferred per data transfer. After the initiator sends the address and R/W bit or the initiator transmits a byte of data to the target, the target-receiver must respond with the acknowledge signal (ACK). When a target-receiver does not respond with an ACK pulse, the initiator aborts the transfer by issuing a STOP condition. The target must leave the SDA line high so that the initiator can abort the transfer.

If the initiator-transmitter is transmitting data as shown in the following figure, then the target-receiver responds to the initiator-transmitter with an acknowledge pulse after every byte of data is received.

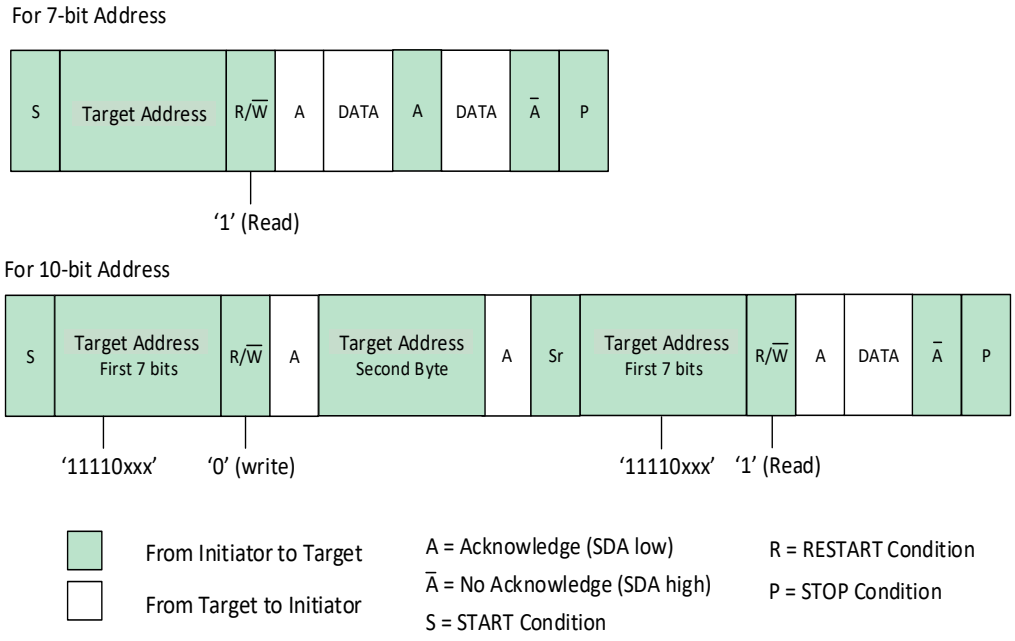
Figure 22-10. Initiator-Transmitter Protocol



22.14.5.3.2 Initiator-Receiver and Target-Transmitter

If the initiator is receiving data as shown in the following figure, then the initiator responds to the target-transmitter with an acknowledge pulse after a byte of data has been received, except for the last byte. This is the way the initiator-receiver notifies the target-transmitter that this is the last byte. The target-transmitter relinquishes the SDA line after detecting the No Acknowledge (NACK) so that the initiator can issue a STOP condition.

Figure 22-11.Initiator-Receiver Protocol



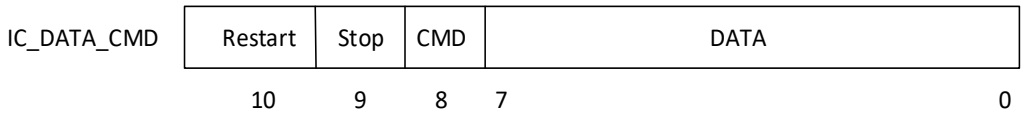
When an initiator does not want to relinquish the bus with a STOP condition, the initiator can issue a RESTART condition. This is identical to a START condition except it occurs after the ACK pulse. Operating in initiator mode, the I^2C controller can then communicate with the same target using a transfer of a different direction.

22.14.6 Tx FIFO Management

The component does not generate a STOP if the Tx FIFO becomes empty; in this situation the component holds the SCL line low, stalling the bus until a new entry is available in the Tx FIFO. A STOP condition is generated only when the user specifically requests it by setting bit 9 (Stop bit) of the command written to IC_DATA_CMD register.

The following figure shows the bits in the IC_DATA_CMD register.

Figure 22-12.IC_DATA_CMD register if IC_EMPTYFIFO_HOLD_MASTER_EN= 1



DATA – Read/Write field; data related from slave is read from this field; data to be sent to slave is written to this field.
CMD – Write-only field; this bit determines whether transfer to be carried out is Read (CMD-1) or Write (CMD-0)
Stop – Write-only field; this bit determines whether STOP is generated after data byte is sent or received
Restart – Write-only field; this bit determines whether RESTART (or STOP followed by START in case of restart capability is not enabled) is generated before data byte is sent or received

22.14.7 Multiple Initiator Arbitration

The I^2C controller bus protocol allows multiple initiators to reside on the same bus. If there are two initiators on the same I²C-bus, there is an arbitration procedure if both try to take control of the bus at the same time by generating a START condition at the same time. Once an initiator (for example, a microcontroller) has control of the bus, no other initiator can take control until the first initiator sends a STOP condition and places the bus in an idle state.

Arbitration takes place on the SDA line, while the SCL line is 1. The initiator, which transmits a 1 while the other initiator transmits 0, loses arbitration and turns off its data output stage. The initiator that lost arbitration can continue to generate clocks until the end of the byte transfer. If both initiators are addressing the same target device, the arbitration could go into the data phase. Upon detecting that it has lost arbitration to another initiator, the I^2C controller will stop generating SCL.

For high-speed mode, the arbitration cannot go into the data phase because each initiator is programmed with a unique high-speed initiator code. This 8-bitcode is defined by the system designer and is set by writing to the High Speed Initiator Mode Code Address Register, IC_HS_MADDR. Because the codes are unique, only one initiator can win arbitration, which occurs by the end of the transmission of the high-speed initiator code.

Control of the bus is determined by address or initiator code and data sent by competing initiators, so there is no central initiator nor any order of priority on the bus.

Arbitration is not allowed between the following conditions:

- A RESTART condition and a data bit
- A STOP condition and a data bit
- A RESTART condition and a STOP condition
- Targets are not involved in the arbitration process.

22.14.8 Clock Synchronization

When two or more initiators try to transfer information on the bus at the same time, they must arbitrate and synchronize the SCL clock. All initiators generate their own clock to transfer messages. Data is valid only during the high period of SCL clock. Clock synchronization is performed using the wired-AND connection to the SCL signal. When the initiator transitions the SCL clock to 0, the initiator starts counting the low time of the SCL clock and transitions the SCL clock signal to 1 at the beginning of the next clock period. However, if another initiator is holding the SCL line to 0, then the initiator goes into a HIGH wait state until the SCL clock line transitions to 1.

All initiators then count off their high time, and the initiator with the shortest high time transitions the SCL line to 0. The initiators then counts out their low time and the one with the longest low time forces the other initiator into a HIGH wait state. Therefore, a synchronized SCL clock is generated. Optionally, targets may hold the SCL line low to slow down the timing on the I^2C bus.

22.14.9 Operation Mode

22.14.9.1 Initiator Mode Operation

22.14.9.1.1 Initial Configuration

The target address and address format can be changed dynamically without having to disable I^2C controller.

The procedures are very similar and are only different with regard to where the IC_10BITADDR_MASTER bit is set (either bit 4 of IC_CON register or bit 12 of IC_TAR register).

To use the I^2C controller as a initiator, perform the following steps:

1. Disable the I^2C controller by writing 0 to bit 0 of the IC_ENABLE register.
2. Write to the IC_CON register to set the maximum speed mode supported for target operation (bits 2:1).
3. Write to the IC_TAR register the address of the I^2C device to be addressed. It also indicates whether a General Call or a START BYTE command is going to be performed by I^2C. The desired speed of the I^2C controller initiator-started transfers, either 7-bit or 10-bit addressing, is controlled by the IC_10BITADDR_MASTER bit field (bit 12).
4. Only applicable for high-speed mode transfers. Write to the IC_HS_MADDR register the desired initiator code for the I^2C controller. The initiator code is programmer-defined.
5. Enable the I^2C controller by writing a 1 to bit 0 of the IC_ENABLE register.
6. Now write the transfer direction and data to be sent to the IC_DATA_CMD register. If the IC_DATA_CMD register is written before the I^2C controller is enabled, the data and commands are lost as the buffers are kept cleared when I^2C controller is not enabled.

Dynamic IC_TAR or IC_10BITADDR_MASTER Update

The I^2C controller supports dynamic updating of the IC_TAR (bits 9:0) and IC_10BITADDR_MASTER (bit 12) bit fields of the IC_TAR register. You can dynamically write to the IC_TAR register provided the software ensures that there are no other

commands in the Tx FIFO that use the existing TAR address. If the software does not ensure this, then IC_TAR should be re-programmed only if the following conditions are met:

- I^2C Controller is not enabled (IC_ENABLE[0]=0);

OR

- I^2C controller is enabled (IC_ENABLE[0]=1); AND
- I^2C controller is NOT engaged in any Initiator (tx, rx) operation (IC_STATUS[5]=0); AND
- I^2C controller is enabled to operate in Initiator mode (IC_CON[0]=1); AND
- There are NO entries in the Tx FIFO (IC_STATUS[2]=1);1

You can change the TAR address dynamically without losing the bus, only if the following conditions are met.

- P2C controller is enabled (IC_ENABLE[0]=1); AND
- IC_EMPTYFIFO_HOLD_MASTER_EN configuration parameter is set to 1; AND
- I^2C controller is enabled to operate in Initiator mode (IC_CON[0]=1); AND
- There are NO entries in the Tx FIFO and the Initiator is in HOLD state (IC_INTR_STAT[13]=1)

Initiator Transmit and Initiator Receive

The I^2C controller supports switching back and forth between reading and writing dynamically. To transmit data, write the data to be written to the lower byte of the I^2C Rx/Tx Data Buffer and Command Register (IC_DATA_CMD). The CMD bit [8] should be written to 0 for I^2C write operations. Subsequently, a read command may be issued by writing "don't cares" to the lower byte of the IC_DATA_CMD register, and a 1 should be written to the CMD bit. The I^2C controller initiator continues to start transfers as long as there are commands present in the transmit FIFO. If the transmit FIFO becomes empty.depending on the value of IC_EMPTYFIFO_HOLD_MASTER_EN, the initiator either inserts a STOP condition after completing the current transfers, or it checks to see if IC_DATA_CMD[9] is set to 1.

- If set to 1, it issues a STOP condition after completing the current transfer.
- If set to 0, it holds SCL low until next command is written to the transmit FIFO.

22.14.9.1.2 Disabling I^2C controller

The register IC_ENABLE_STATUS is added to allow software to unambiguously determine when the hardware has completely shutdown in response to bit 0 of the IC_ENABLE register being set from 1 to 0.

1. If the software or application is aware the I^2C controller is not using the TAR address for the pending commands in the Tx FIFO, then it is possible to update the TAR address even while the Tx FIFO has entries (IC_STATUS[2]= 0). Only one register is required to be monitored, as opposed to monitoring two registers (IC_STATUS and IC_RAW_INTR_STAT).

22.14.9.2 Procedure

1. Define a timer interval (ti2c_poll) equal to the 10 times the signaling period for the highest I^2C transfer speed used in the system and supported by I^2C controller. For example, if the highest I^2C transfer mode is 400 kb/s, then this ti2c_poll is 25us.

2. Define a maximum time-out parameter, MAX_T_POLL_COUNT, such that if any repeated polling operation exceeds this maximum value, an error is reported.
3. Execute a blocking thread/process/function that prevents any further I^2C initiator transactions to be started by software, but allows any pending transfers to be completed.
4. The variable POLL_COUNT is initialized to zero.
5. Set bit 0 of the IC_ENABLE register to 0.
6. Read the IC_ENABLE_STATUS register and test the IC_EN bit (bit 0). Increment POLL_COUNT by one. If POLL_COUNT >= MAX_T_POLL_COUNT, exit with the relevant error code.
7. If IC_ENABLE_STATUS[0] is 1, then sleep for ti2c_poll and proceed to the previous step. Otherwise, exit with a relevant success code.

22.14.9.3 Aborting I^2C Transfers

The ABORT control bit of the IC_ENABLE register allows the software to relinquish the I^2C bus before completing the issued transfer commands from the Tx FIFO. In response to an ABORT request, the controller issues the STOP condition over the I^2C bus, followed by Tx FIFO flush. Aborting the transfer is allowed only in initiator mode of operation.

1. Stop filling the Tx FIFO (IC_DATA_CMD) with new commands
2. When operating in DMA mode, disable the transmit DMA by setting TDMAE to 0.
3. Set bit 1 of the IC_ENABLE register (ABORT) to 1.
4. Wait for the M_TX_ABRT interrupt.
5. Read the IC_TX_ABRT_SOURCE register to identify the source as ABRT_USER_ABRT.

22.14.9.4 Spike Suppression

The I2C Controller contains programmable spike suppression logic that match requirements imposed by the I2C Bus Specification for SS/FS (tSP, Table 9) and high speed (tSP, Table 11).

This logic is based on counters that monitor the input signals (SCL and SDA), checking if they remain stable for a predetermined amount of I2C Controller Clock (ic_clk) cycles before they are sampled internally. There is one separate counter for each signal (SCL and SDA). The number of ic_clk cycles can be programmed and should be calculated taking into account the frequency of ic_clk and the relevant spike length specification.

Each counter is started whenever its input signal changes its value. Depending on the behavior of the input signal, one of the following scenarios occurs:

- The input signal remains unchanged until the counter reaches its count limit value. When this happens, the internal version of the signal is updated with the input value, and the counter is reset and stopped. The counter is not restarted until a new change on the input signal is detected.
- The input signal changes again before the counter reaches its count limit value. When this happens, the counter is reset and stopped, but the internal version of the signal is not updated. The counter remains stopped until a new change on the input signal is detected.

22.14.9.5 DMA Controller Interface

The I2C Controller has an optional built-in DMA capability that can be selected at configuration time; it has a handshaking interface to a DMA Controller to request and control transfers. The APB bus is used to perform the data transfer to or from the DMA. While the I2C Controller DMA operation is designed in a generic way to fit any DMA controller as easily as possible, it is designed to work seamlessly, and best used, with the DMA Controller. The settings of the DMA Controller that are relevant to the operation of the I2C Controller are discussed here, mainly bit fields in the DMA Controller channel control register, CTLx, where x is the channel number.

Note: When the I2C Controller interfaces to the DMA Controller, the DMA Controller is always a flow controller; that is, it controls the block size. This must be programmed by software in the DMA Controller. The DMA Controller always transfers data using DMA burst transactions if possible, for efficiency. The relevant DMA settings are discussed in the following sections. The DMA output dma_finish is a status signal to indicate that the DMA block transfer is complete. I2C Controller does not use this status signal, and therefore does not appear in the I/O port list.

Note: Standard synchronization logic (two flip-flops in series) is implemented upstream of the spike suppression logic and is not affected in any way by the contents of the spike length registers or the operation of the spike suppression logic; the two operations (synchronization and spike suppression) are completely independent. Because the SCL and SDA inputs are asynchronous to ic_clk, there is one ic_clk cycle uncertainty in the sampling of these signals; that is, depending on when they occur relative to the rising edge of ic_clk, spikes of the same original length might show a difference of one ic_clk cycle after being sampled.

Note: Spike suppression is symmetrical; that is, the behavior is exactly the same for transitions from 0 to 1 and from 1 to 0.

22.14.9.5.1 Enabling the DMA Controller Interface

To enable the DMA Controller interface on the I2C Controller, a write to the DMA Control Register (IC_DMA_CR) is required. Writing a 1 into the TDMAE bit field of IC_DMA_CR register enables the I2C Controller transmit handshaking interface. Writing a 1 into the RDMAE bit field of the IC_DMA_CR register enables the I2C Controller receive handshaking interface.

22.14.9.5.2 Overview of Operation

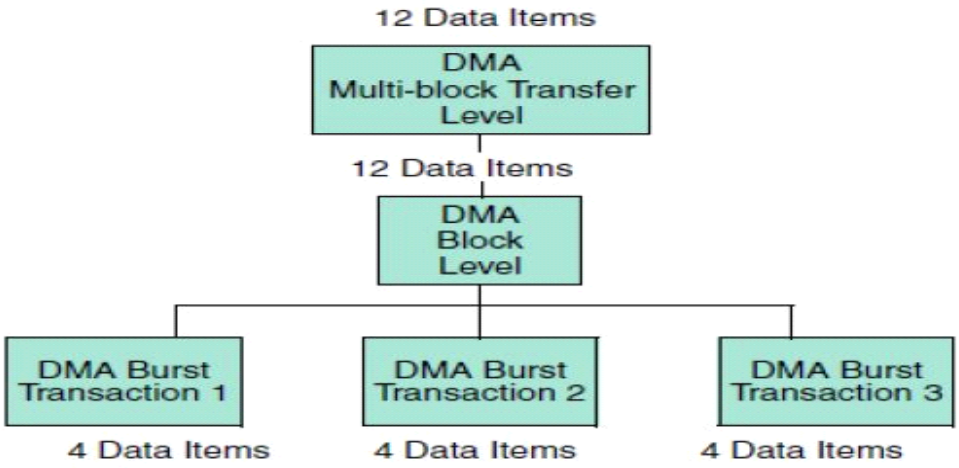
As a block flow control device, the DMA Controller is programmed by the processor with the number of data items (block size) that are to be transmitted or received by I2C Controller; this is programmed into the BLOCK_TS field of the DMA Controller CTLx register.

The block is broken into a number of transactions, each initiated by a request from the I2C Controller. The DMA Controller must also be programmed with the number of data items (in this case, I2C Controller FIFO entries) to be transferred for each DMA request. This is also known as the burst transaction length and is programmed into the SRC_MSIZE/DEST_MSIZE fields of the DMA Controller CTLx register for source and destination, respectively.

Figure 22-13 shows a single block transfer, where the block size programmed into the DMA Controller is 12 and the burst transaction length is set to 4. In this case, the block size is a multiple of the burst transaction length. Therefore, the DMA block transfer consists of a series of burst transactions. If the I2C Controller makes a transmit request

to this channel, four data items are written to the I2C Controller TX FIFO. Similarly, if the I2C Controller makes a receive request to this channel, four data items are read from the I2C Controller RX FIFO. Three separate requests must be made to this DMA channel before all 12 data items are written or read.

Figure 22-13.Breakdown of DMA Transfer into Burst Transactions



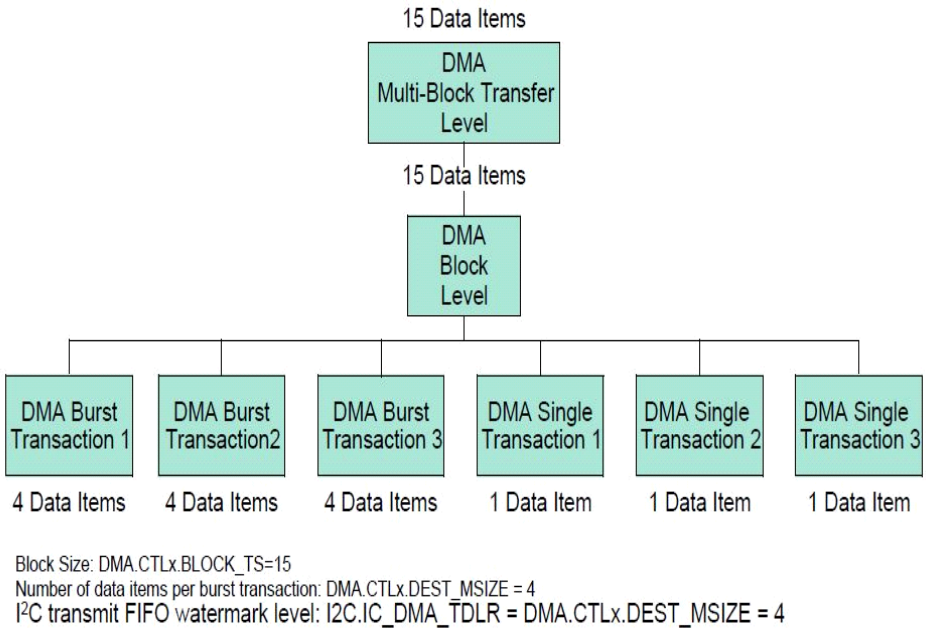
Block Size: DMA.CTLx.BLOCK_TS=12

Number of data items per source burst transaction: DMA.CTLx.SRC_MSIZE = 4

I2C receive FIFO watermark level: I2C.DMARDLR + 1 = DMA.CTLx.SRC_MSIZE = 4

When the block size programmed into the DMA Controller is not a multiple of the burst transaction length, as shown in Figure 22-14, a series of burst transactions followed by single transactions are needed to complete the block transfer.

Figure 22-14.Breakdown of DMA Transfer into Single and Burst Transactions



22.14.9.5.3 Transmit Watermark Level and Transmit FIFO Underflow

During I2C Controller serial transfers, transmit FIFO requests are made to the DMA Controller whenever the number of entries in the transmit FIFO is less than or equal to the DMA Transmit Data Level Register (IC_DMA_TDLR) value; this is known as the watermark level. The DMA Controller responds by writing a burst of data to the transmit FIFO buffer, of length CTLx.DEST_MSIZE.

If IC_EMPTYFIFO_HOLD_MASTER_EN parameter is set to 0, data should be fetched from the DMA often enough for the transmit FIFO to perform serial transfers continuously; that is, when the FIFO begins to empty another DMA request should be triggered. Otherwise, the FIFO runs out of data causing a STOP to be inserted on the I2C bus. To prevent this condition, you must set the watermark level correctly.

22.14.9.5.4 Choosing the Transmit Watermark Level

Consider the example where the assumption is made:

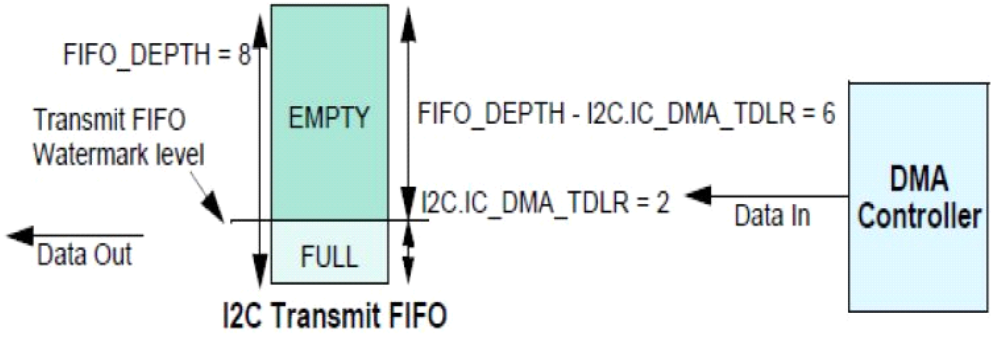
DMA.CTLx.DEST_MSIZE = FIFO_DEPTH - I2C.IC_DMA_TDLR

Here the number of data items to be transferred in a DMA burst is equal to the empty space in the Transmit FIFO. Consider two different watermark level settings.

Case 1: IC_DMA_TDLR = 2

- Transmit FIFO watermark level = I2C.IC_DMA_TDLR = 2
- DMA.CTLx.DEST_MSIZE = FIFO_DEPTH - I2C.IC_DMA_TDLR = 6
- I2C transmit FIFO_DEPTH = 8
- DMA.CTLx.BLOCK_TS = 30

Figure 22-15.Case 1 Watermark Levels



Therefore, the number of burst transactions needed equals the block size divided by the number of data items per burst:

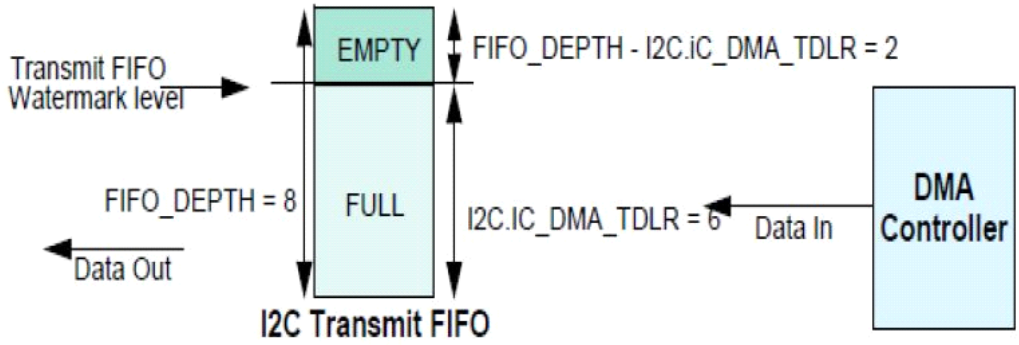
DMA.CTLx.BLOCK_TS/DMA.CTLx.DEST_MSIZE = 30/6 = 5

The number of burst transactions in the DMA block transfer is 5. But the watermark level, I2C.IC_DMA_TDLR, is quite low. Therefore, the probability of an I2C underflow is high where the I2C serial transmit line needs to transmit data, but where there is no data left in the transmit FIFO. This occurs because the DMA has not had time to service the DMA request before the transmit FIFO becomes empty.

Case 2: IC_DMA_TDLR = 6

- Transmit FIFO watermark level = I2C.IC_DMA_TDLR = 6
- DMA.CTLx.DEST_MSIZE = FIFO_DEPTH - I2C.IC_DMA_TDLR = 2
- I2C transmit FIFO_DEPTH = 8
- DMA.CTLx.BLOCK_TS = 30

Figure 22-16.Case 2 Watermark Levels



Number of burst transactions in Block:

DMA.CTLx.BLOCK_TS/DMA.CTLx.DEST_MSIZE = 30/2 = 15

In this block transfer, there are 15 destination burst transactions in a DMA block transfer. But the watermark level, I2C.IC_DMA_TDLR, is high. Therefore, the probability of an I2C underflow is low because the DMA controller has plenty of time to service the destination burst transaction request before the I2C transmit FIFO becomes empty.

Thus, the second case has a lower probability of underflow at the expense of more burst transactions per block. This provides a potentially greater amount of bus bursts per block and worse bus utilization than the former case.

Therefore, the goal in choosing a watermark level is to minimize the number of transactions per block, while at the same time keeping the probability of an underflow condition to an acceptable level. In practice, this is a function of the ratio of the rate at which the I2C transmits data to the rate at which the DMA can respond to destination burst requests.

For example, promoting the channel to the highest priority channel in the DMA, and promoting the DMA initiator interface to the highest priority initiator in the bus layer, increases the rate at which the DMA controller can respond to burst transaction requests. This in turn allows you to decrease the watermark level, which improves bus utilization without compromising the probability of an underflow occurring.

22.14.9.5.5 Selecting DEST_MSIZE and Transmit FIFO Overflow

As can be seen from Figure 22-16, programming DMA.CTLx.DEST_MSIZE to a value greater than the watermark level that triggers the DMA request may cause overflow when there is not enough space in the I2C transmit FIFO to service the destination burst request. Therefore, the following equation must be adhered to in order to avoid overflow:

DMA.CTLx.DEST_MSIZE <= I2C.FIFO_DEPTH - I2C.IC_DMA_TDLR (1)

In Case 2: IC_DMA_TDLR = 6, the amount of space in the transmit FIFO at the time the burst request is made is equal to the destination burst length, DMA.CTLx.DEST_MSIZE. Thus, the transmit FIFO may be full, but not overflowed, at the completion of the burst transaction.

Therefore, for optimal operation, DMA.CTLx.DEST_MSIZE should be set at the FIFO level that triggers a transmit DMA request; that is:

DMA.CTLx.DEST_MSIZE = I2C.FIFO_DEPTH - I2C.IC_DMA_TDLR (2)

This is the setting used in Figure 22-14.

Adhering to equation (2) reduces the number of DMA bursts needed for a block transfer, and this in turn improves bus utilization.

Note: The transmit FIFO is not full at the end of a DMA burst transfer if the I2C has successfully transmitted one data item or more on the I2C serial transmit line during the transfer.

22.14.9.5.6 Receive Watermark Level and Receive FIFO Overflow

During I2C Controller serial transfers, receive FIFO requests are made to the DMA Controller whenever the number of entries in the receive FIFO is at or above the DMA Receive Data Level Register; that is, IC_DMA_RDLR+1. This is known as the watermark level. The DMA Controller responds by fetching a burst of data from the receive FIFO buffer of length CTLx.SRC_MSIZE.

Data should be fetched by the DMA often enough for the receive FIFO to accept serial transfers continuously; that is, when the FIFO begins to fill, another DMA transfer is requested. Otherwise, the FIFO fills with data (overflow). To prevent this condition, you must correctly set the watermark level.

22.14.9.5.7 Choosing the Receive Watermark Level

Similar to choosing the transmit watermark level described earlier, the receive watermark level, IC_DMA_RDLR+1, should be set to minimize the probability of overflow, as shown in Figure 22-17**.** It is a trade-off between the number of DMA burst transactions required per block versus the probability of an overflow occurring.

22.14.9.5.8 Selecting SRC_MSIZE and Receive FIFO Underflow

As can be seen in Figure 22-17, programming a source burst transaction length greater than the watermark level may cause underflow when there is not enough data to service the source burst request. Therefore, equation 3 following must be adhered to avoid underflow.

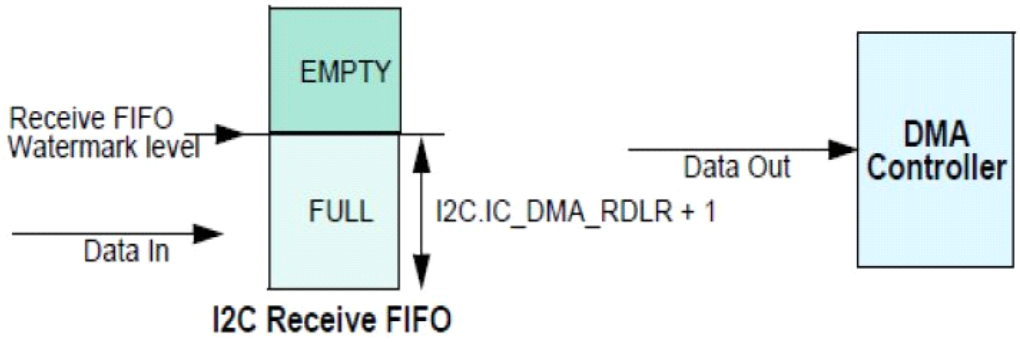
If the number of data items in the receive FIFO is equal to the source burst length at the time the burst request is made – DMA.CTLx.SRC_MSIZE – the receive FIFO may be emptied, but not underflowed, at the completion of the burst transaction. For optimal operation, DMA.CTLx.SRC_MSIZE should be set at the watermark level; that is:

DMA.CTLx.SRC_MSIZE = I2C.IC_DMA_RDLR + 1 (3)

Adhering to equation (3) reduces the number of DMA bursts in a block transfer, which in turn can avoid underflow and improve bus utilization.

The receive FIFO is not empty at the end of the source burst transaction if the I2C has successfully received one data item or more on the I2C serial receive line during the burst.

Figure 22-17.I2C Receive FIFO



22.14.10 IC_CLK Frequency Configuration

The input clock to I^2C controller is 100MHz. When the I^2C controller is configured as a Standard (SS), Fast (FS)/Fast-Mode Plus (FM+), or High Speed (HS) initiator, the *CNT registers must be set before any I^2C bus transaction can take place in order to ensure proper I/O timing. The *CNT registers are:

- IC_SS_SCL_HCNT
- IC_SS_SCL_LCNT
- IC_FS_SCL_HCNT
- IC_FS_SCL_LCNT
- IC_HS_SCL_HCNT
- IC_HS_SCL_LCNT

22.14.11 Signal Description

Table 22-12. I^2C Signal Description

Signal Name	Type	Description
PSE_I2C[7:0]_SCL	iod	Serial Clock
PSE_I2C[7:0]_SDA	iod	Serial Data

22.14.12 Registers

Please refer to chapter 7 of the Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors for Internet of Things (IoT) Applications, Datasheet, Volume 2 (Book 3 of 3), Intel® Programmable Services Engine (Intel® PSE) (Document Number: 636723), for a description of the registers associated with subject of this chapter.

22.15 UART Controller

22.15.1 Overview

All UART have four-wire RS-232, bi-directional point-to-point connection between the Intel® PSE and a peripheral. Four out of six UARTs have 3 additional signals that add RS-485 mode support.

22.15.1.1 Ownership Allocation

- UART0 ownership can be controlled using OWNERSHIP_DEV8 field in Ownership Control 1 register
- UART1 ownership can be controlled using OWNERSHIP_DEV9 field in Ownership Control 1 register

- UART2 ownership can be controlled using OWNERSHIP_DEV10 field in Ownership Control 1 register
- UART3 ownership can be controlled using OWNERSHIP_DEV11 field in Ownership Control 1 register
- UART4 ownership can be controlled using OWNERSHIP_DEV12 field in Ownership Control 1 register
- UART5 ownership can be controlled using OWNERSHIP_DEV13 field in Ownership Control 1 register

22.15.2 Features

The UART has the following Capabilities:

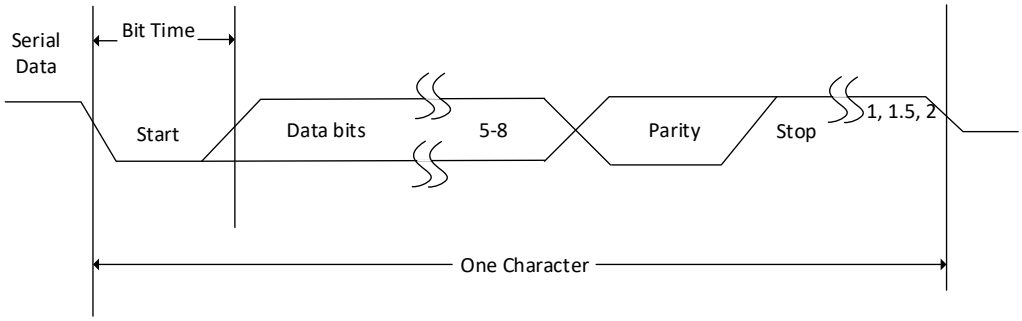
- Operate in strictly compliant 16550 mode
- Operate in 16750 auto flow control model. RTS and CTS will be automatically managed until there is data in RX FIFO or space in TX FIFO
- Support RS-232 mode (UART[5:0]) & RS-485 mode (UART[3:0])
- Support operating in odd, even and none parity modes
- Support 1, 1.5 and 2 stop bits as per standard
- Support HW based flow control and SW based flow control
- Support for operating speeds up to 10 Mb/s in S0, 6.25Mbps in S0ix
- Support for auto flow control using the RTS#/CTS# signals
- 64 byte FIFO
- DMA support to allow direct transfer to the Intel® PSE local SRAM without intervention by the Arm* Cortex*-M7. This saves interrupts on packets that are longer than the FIFO or when there are back-to-back packets to send or receive.
- Multi drop support in RS-485 mode

22.15.3 Functional Description

22.15.3.1 UART (RS-232) Serial Protocol

Because the serial communication between the UART controller and a selected device is asynchronous, additional bits (start and stop) are added to the serial data to indicate the beginning and end. Utilizing these bits allows two devices to be synchronized. This structure of serial data—accompanied by start and stop bits—is referred to as a character, as shown in the following figure.

Figure 22-18.Serial Data Format



An additional parity bit can be added to the serial character. This bit appears after the last data bit and before the stop bit(s) in the character structure in order to provide the UART controller with the ability to perform simple error checking on the received data.

The UART controller Line Control Register is used to control the serial character characteristics. The individual bits of the data word are sent after the start bit, starting with the least significant bit (LSB). These are followed by the optional parity bit, followed by the stop bit(s), which can be 1, 1.5, or 2.

All the bits in the transmission are transmitted for exactly the same time duration; the exception to this is the half-stop bit when 1.5 stop bits are used. This duration is referred to as a Bit Period or Bit Time; one Bit Time equals sixteen baud clocks.

To ensure stability on the line, the receiver samples the serial input data at approximately the midpoint of the Bit Time once the start bit has been detected. Because the exact number of baud clocks is known for which each bit was transmitted, calculating the midpoint for sampling is not difficult; that is, every sixteen baud clocks after the midpoint sample of the start bit.

Together with serial input debouncing, this sampling helps to avoid the detection of false start bits. Short glitches are filtered out by debouncing, and no transition is detected on the line. If a glitch is wide enough to avoid filtering by debouncing, a falling edge is detected. However, a start bit is detected only if the line is again sampled low after half a bit time has elapsed.

22.15.3.2 9-bit Data Transfer (only RS-485 mode)

The UART controller can be configured to have 9-bit data transfer in both transmit and receive mode. The 9th bit in the character appears after the 8th bit and before the parity bit in the character.

By enabling 9-bit data transfer mode, UART controller can be used in multi-drop systems where one initiator is connected to multiple targets in a system. The initiator communicates with one of the targets. When the initiator wants to transfer a block of data to a target, it first sends an address byte to identify the target.

The differentiation between the address/data byte is done based on the 9th bit in the incoming character. If the 9th bit is set to 0, then the character represents a data byte. If the 9th bit is set to 1, then the character represents address byte. All the target systems compare the address byte with their own address and only the target (in which

the address has matched) is enabled to receive data from the initiator. The initiator then starts transmitting data bytes to the target. The non-addressed target systems ignore the incoming data until a new address byte is received.

Configuration of the UART controller for 9-bit data transfer does the following:

- LCR_EXT[0] bit is used to enable or disable the 9-bit data transfer.
- LCR_EXT[1] bit is used to choose between hardware and software based address match in the case of receive.
- LCR_EXT[3] bit is used to choose between hardware and software based address transmission.
- TAR and RAR registers are used to transmit address and to match the received address, respectively.
- THR, RBR, STHR and SRBR registers are of 9-bit which is used to do the data transfers in 9-bit mode.
- LSR[8] bit is used to indicate the address received interrupt.

22.15.3.3 Transmit Mode

UART controller supports two types of transmit modes:

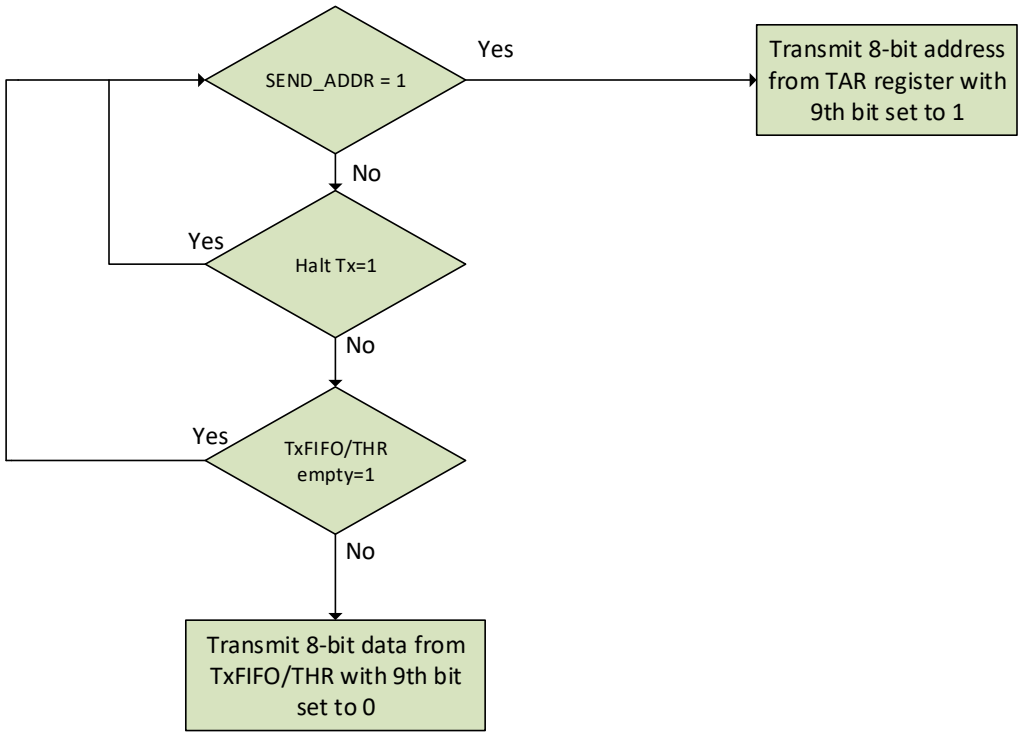
- Transmit Mode 0 (when (LCR_EXT[3]) is set to 0)
- Transmit Mode 1 (when (LCR_EXT[3]) is set to 1)

22.15.3.3.1 Transmit Mode 0

In transmit mode 0, the address is programmed in the Transmit Address Register (TAR) register and data is written into the Transmit Holding Register (THR) or the Shadow Transmit Holding Register (STHR). The 9th bit of the THR and STHR register is not applicable in this mode.

Figure 22-19 illustrates the transmission of address and data based on SEND_ADDR (LCR_EXT[2]), Halt Tx, and TxFIFO/THR empty conditions.

Figure 22-19.Auto Address Transmit Flow Chart



The address of the target to which the data is to be transmitted is programmed in the TAR register.

You must enable the SEND_ADDR (LCR_EXT[2]) bit to transmit the target address present in the TAR register on the serial UART line with 9th data bit set to 1 to indicate that the address is being sent to the target. The UART controller clears the SEND_ADDR bit after the address character starts transmitting on the UART line.

The data required to transmit to the target is programmed through Transmit Holding Register (THR).

The data is transmitted on the UART line with 9th data bit set to 0 to indicate data is being sent to the target.

If the application is required to fill the data bytes in the TxFIFO before sending the address on the UART line (before setting LCR_EXT[2]=1), then it is recommended to set the "Halt Tx" to 1 such that UART controller does not start sending out the data in the TxFIFO as data byte. Once the TxFIFO is filled, then program SEND_ADDR (LCR_EXT[2]) to 1 and then set "Halt Tx" to 0.

22.15.3.3.2 Transmit Mode 1

In transmit mode 1, THR and STHR registers are of 9-bit wide and both address and data are programmed through the THR and STHR registers. The UART controller does not differentiate between address and data, and both are taken from the TxFIFO. The

SEND_ADDR (LCR_EXT[2]) bit and Transmit address register (TAR) are not applicable in this mode. The software must pack the 9th bit with 1/0 depending on whether address/data has to be sent.

22.15.3.4 Receive Mode

The UART controller supports two receive modes:

- Hardware Address Match Receive Mode (when ADDR_MATCH (LCR_EXT[1]) is set to 1)
- Software Address Match Receive Mode (when ADDR_MATCH (LCR_EXT[1]) is set to 0)

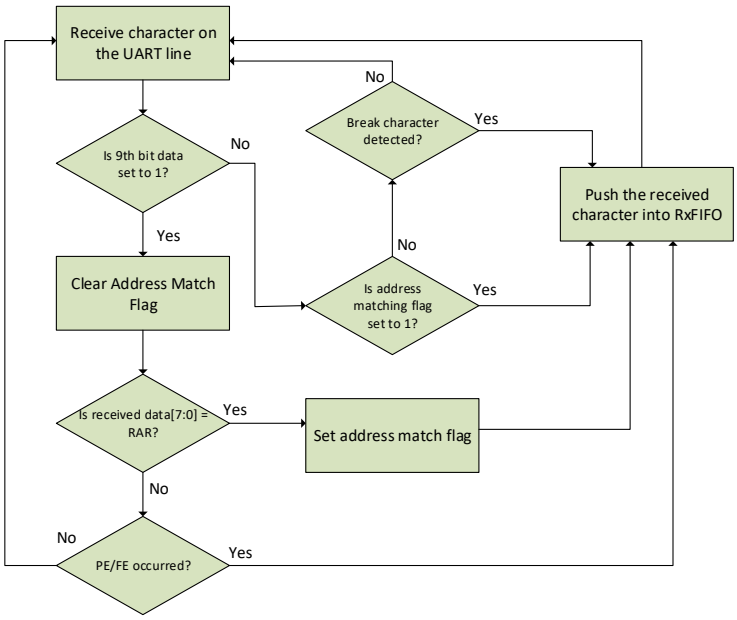
22.15.3.4.1 Hardware Address Match Receive Mode

In the hardware address match receive mode, the UART controller matches the received character with the address programmed in the Receive Address register (RAR), if the 9th bit of the received character is set to 1.

If the received address is matched with the programmed address in RAR register, then subsequent data bytes (with 9th bit set to 0) are pushed into the RxFIFO. If the address matching fails, then UART controller discards further data characters until a matching address is received.

The following figure illustrates the flow chart for the reception of data bytes based on the address matching feature.

Figure 22-20.Hardware Address Match Receive Mode



UART controller receives the character irrespective of whether the 9th bit data is set to 1. If 9th bit of the received character is set to 1, then it clears internal address match flag and then compares the received 8-bit character information with the address programmed in the RAR register.

If the received address character matches with the address programmed in the RAR register, then the address match flag is set to 1 and the received character is pushed to the RxFIFO in FIFO-mode or to RBR register in non-FIFO mode and the ADDR_RCVD bit in LSR register is set to indicate that the address has been received.

In case of parity or if a framing error is found in the received address character and if the address is not matched with the RAR register, then the received address character is still pushed to RxFIFO or RBR register with ADDR_RCVD and PE/FE error bit set to 1.

The subsequent data bytes (9th bit of received character is set to 0) are pushed to the Rx_FIFO in FIFO mode or to the RBR register in non-FIFO mode until the new address character is received.

If any break character is received, UART controller treats it as a special character and pushes to the RxFIFO or RBR register based on the FIFO_MODE irrespective of address match flag.

22.15.3.4.2 Software Address Match Receive Mode

In this mode of operation, the UART controller does not perform the address matching for the received address character (9th bit data set to 1) with the RAR register. The UART controller always receives the 9-bit data and pushes in to RxFIFO in FIFO mode or to the RBR register in non-FIFO mode. The user must compare the address whenever address byte is received and indicated through ADDR_RCVD bit in the Line

Status register. The user can flush/reset the RxFIFO in case of address not matched through 'RCVR FIFO Reset' bit in FIFO control register (FCR).

22.15.3.5 RS-485 Serial Protocol

The difference between the RS-232 and RS-485 standards is its use of a balanced line for transmission. This usage is also known as the differential format that sends the same signal on two separate lines with phase delay and then compares the signals at the end, subtracts any noise, and adds them to regain signal strength. This process allows the RS-485 standard to be viable over significantly longer distances than its short range RS-232 counterpart.

Note: The processor receive and transmit signals that are not differential and an external transceiver is needed to convert them into differential pairs.

UART controller supports the RS-485 serial protocol that enables transfer of serial data using the RS-485 interface. The driver enable (PSE_HSUARTx_DE) and receiver enable (PSE_HSUARTx_RE) signals are generated for enabling the RS-485 interface support in the external transceiver. The de and re signals are hardware generated and the assertion/de-assertion times for these signals are programmable. The active level of these signals are configurable.

Configuration of the UART controller for RS-485 interface does the following:

1. Bit 0 of the Transceiver Control Register (TCR) enables or disables the RS-485 mode which is indicated to the transceiver by the PSE_HSUARTx_EN signal.

2. Bit 1 and bit 2 of TCR are used to select the polarity of PSE_HSUARTx_RE and PSE_HSUARTx_DE signals.
3. Bit [4:3] of the TCR selects the type of transfer in RS-485 mode.
4. Driver output enable (DE_EN) and Receiver output enable (RE_EN) registers are used for software control of DE and RE signals.
5. Driver output Enable Timing (DET) register is used to program the assertion and deassertion timings of DE signal.
6. TurnAround Timing (TAT) register is used to program the turnaround time from PSE_HSUARTx_DE to PSE_HSUARTx_RE and PSE_HSUARTx_RE to PSE_HSUARTx_DE.

22.15.3.5.1 PSE_HSUARTx_DE Assertion and De-assertion Timing

The assertion and deassertion timings of the PSE_HSUARTx_DE signal are controlled through the DET register:

- PSE_HSUARTx_DE assertion time (DET[7:0]): The assertion time is the time between the activation of the PSE_HSUARTx_DE signal and the beginning of the START bit. The value represented is in terms of serial clock cycles.
- PSE_HSUARTx_DE de-assertion time (DET[15:8]): The de-assertion time is the time between the end of the last stop bit, in a transmitted character, and the de-activation of the PSE_HSUARTx_DE signal. The value represented is in terms of serial clock cycles.

Hardware ensures that these values are met for PSE_HSUARTx_DE assertion and PSE_HSUARTx_DE deassertion before/after active data transmission.

22.15.3.5.2 RS-485 Modes

UART controller consists of the following RS-485 modes based on the XFER_MODE field in the Transceiver.

Control Register (TCR) register:

- Full Duplex Mode. In this mode, XFER_MODE of TCR is set to 0.
- Software-Controlled Half Duplex Mode . In this mode, XFER_MODE of TCR is set to 1.
- Hardware-Controlled Half Duplex Mode . In this mode, XFER_MODE of TCR is set to 2

Full Duplex Mode

The full duplex mode supports both transmit and receive transfers simultaneously.

In Full Duplex mode, the PSE_HSUARTx_DE signal:

- Goes active if both these conditions are satisfied:
  - When the DE Enable (DE_EN[0]) field of Driver Output Enable Register is set to 1.
  - Transmitter Holding Register is not empty in non-FIFO mode or transmitter FIFO is not empty in FIFO mode.
- Goes inactive if both these conditions are satisfied

  - When the current ongoing transmitting serial transfer is completed.
  - Either DE Enable (DE_EN[0]) of Driver Output Enable Register is set to 0, transmitter FIFO is empty in FIFO mode or Transmitter Holding Register is empty in non-FIFO mode.

In Full Duplex mode, the PSE_HSUARTx_RE signal:

- Goes active when RE Enable (RE_EN[0]) of Receiver Output Enable Register is set to 1.
- Goes inactive when RE Enable (RE_EN[0]) of Receiver Output Enable Register is set to 0.

The user can choose when to transmit or when to receive. Both PSE_HSUARTx_RE and PSE_HSUARTx_DE can be simultaneously asserted or de-asserted at any time. UART controller does not impose any turnaround time between transmit and receive ('PSE_HSUARTx_DE to PSE_HSUARTx_RE') or receive to transmit ('PSE_HSUARTx_RE to PSE_HSUARTx_DE') in this mode. This mode can directly be used in full duplex operation where separate differential pair of wires is present for transmit and receive.

Software-Controlled Half Duplex Mode

The software-controlled half duplex mode supports either transmit or receive transfers at a time but not both simultaneously. The switching between transmit to receive or receive to transmit is through programming the Driver output enable (DE_EN) and Receiver output enable (RE_EN) registers.

In software-controlled Half Duplex mode, the PSE_HSUARTx_DE signal:

- Goes active if the following conditions are satisfied:
  - The DE Enable (DE_EN[0]) field of the Driver Output Enable Register is set to 1.
  - Transmitter Holding Register is not empty in non-FIFO mode or transmitter FIFO is not empty in FIFO mode.
  - If any receive transfer is ongoing, then the signal waits until receive has finished, and after the turnaround time counter ('PSE_HSUARTx_RE to PSE_HSUARTx_DE') has elapsed.
- Goes inactive if the following conditions are satisfied:
  - The current ongoing transmitting serial transfer is completed.
  - The DE Enable (DE_EN[0]) field of Driver Output Enable Register is set to 0.
  - Either transmitter FIFO is empty in FIFO mode or Transmitter Holding Register is empty in non- FIFO mode.

In software-controlled half duplex mode, the PSE_HSUARTx_RE signal:

- Goes active if the following conditions are satisfied:
  - When RE Enable (RE_EN[0]) field of Receiver Output Enable Register is set to 1.
  - If any transmit transfer is ongoing, then the signal waits until transmit has finished and after the turnaround time counter ('PSE_HSUARTx_DE to PSE_HSUARTx_RE') has elapsed.
- Goes in-active under the following conditions:
  - The current ongoing receive serial transfer is completed.
  - When RE Enable (RE_EN[0]) of Receiver Output Enable Register is set to 0.

The user must enable either DE or RE but not both at any point of time. As PSE_HSUARTx_RE and PSE_HSUARTx_DE signals are mutually exclusive, the user must ensure that both of them are not programmed to be active at any point of time.

In this mode, the hardware ensures that a proper turnaround time is maintained while switching from PSE_HSUARTx_RE to PSE_HSUARTx_DE or from PSE_HSUARTx_DE to PSE_HSUARTx_RE (value of turnaround is obtained from the TAT register, in terms of serial clock cycles).

Hardware-Controlled Half Duplex Mode

The hardware-controlled half duplex mode supports either transmit or receive transfers at a time but not both simultaneously. If both 'DE Enable' and 'RE Enable' bits of Driver output enable (DE_EN) and Receiver output enable (RE_EN) registers are enabled, the switching between transmit to receive or receive to transmit is automatically done by the hardware based on the empty condition of Tx-FIFO.

In hardware-controlled half duplex mode, the PSE_HSUARTx_DE signal:

- Goes active if the following conditions are satisfied:
  - The DE Enable (DE_EN[0]) field of Driver Output Enable Register is set to 1.
  - Transmitter Holding Register is not empty in non-FIFO mode or transmitter FIFO is not empty in FIFO mode.
  - If any receive transfer is ongoing, then the signal waits until receive is finished and after the turnaround time counter ('PSE_HSUARTx_RE to PSE_HSUARTx_DE') has elapsed.
- Goes inactive if the following conditions are satisfied
  - The current ongoing transmitting serial transfer is completed.
  - Either transmitter FIFO is empty in FIFO mode or Transmitter Holding Register is empty in non-FIFO mode or the DE Enable (DE_EN[0]) of Driver output Enable Register is set to 0.

In hardware-controlled half duplex mode, the PSE_HSUARTx_DE signal:

When RE Enable (RE_EN[0]) field of Receiver Output Enable Register is set to 1.

Either transmitter FIFO is empty in FIFO mode or Transmitter Holding Register is empty in non- FIFO mode.

If any transmit transfer is ongoing, then the signal waits until transmit is finished and after the turnaround time counter ('PSE_HSUARTx_DE to PSE_HSUARTx_RE') has elapsed.

- Goes inactive under the following conditions:
  - The current ongoing receive serial transfer has completed.
  - Either transmitter FIFO is non-empty in FIFO mode or Transmitter Holding Register is non empty in non-FIFO mode or the RE Enable (RE_EN[0]) of Receiver output Enable Register is set to 0.

In this mode, the hardware ensures that a proper turnaround time is maintained while switching from 'PSE_HSUARTx_RE' to 'PSE_HSUARTx_DE' or from 'PSE_HSUARTx_DE' to 'PSE_HSUARTx_RE' (value of turnaround is obtained from the TAT register, in terms of serial clock cycles) as shown in Figure 22-41 and Figure 22-42.

22.15.4 Fractional Baud Rate Support

UART controller supports fractional baud rate that enables a user to program the fractional part of the divisor value to generate fractional baud rate that results in reduced frequency error. The UART interface usage has been evolving to include ever increasing baud rate speeds. The UART controller needs to be software configurable to handle the baud rates within 2% frequency error.

Serial clock operating frequency (Input serial clock is 200MHz in S0 mode and 100MHz in S0ix mode.

The baud rate is determined by the following factors:

- Serial clock operating frequency (sclk in Asynchronous serial clock implementation or pclk in single clock implementation)
- The desired baud rate.
- The baud rate generator divisor value, DIVISOR (composed of DLH & DLL registers).
- The acceptable Baud-rate error, %ERROR

The equation to calculate the baud rate is as follows:

Equation (1)

$$\text{Baud Rate} = \frac{\text{Serial Clock Operating Frequency}}{(16 \times \text{DIVISOR})}$$

Where,

DIVISOR – Number (in hexadecimal) to program the DLL and DLH.

Serial clock frequency – sclk = 200MHz (S0 state) or 100 MHz (S0ix state)
pclk = 100MHz

From Equation (1), DIVISOR can be calculated as:

$$\text{DIVISOR} = \frac{\text{Serial Clock Operating Frequency}}{(16 \times \text{Baud Rate})}$$

Also from Equation (1), it can also be shown that:

$$\text{Serial clock frequency} = \text{Baud Rate} \times 16 \times \text{DIVISOR}$$

The Error between the Baud rate and Baud rate (selected) is given as:

$$\text{Percentage ERROR} = \frac{|\text{Baud Rate} - \text{Baud Rate (selected)}|}{\text{Baud Rate}} \times 100$$

Configuration of the UART controller for Fractional Baud Rate does the following:

The configurable parameter DLF_SIZE is used to choose the width of the register that stores fractional part of the divisor.

The fractional value of the divisor is programmed in the Divisor Latch Fraction Register (DLF) register. The fractional value is computed by using the (Divisor Fraction value)/(2^DLF_SIZE) formula. The following table shows fractional values when the DLF_SIZE=4.

Table 22-13. Divisor Latch Fractional Values

DLF Value	Fraction	Fractional Value
0000	0/16	0.0000
0001	1/16	0.0625
0010	2/16	0.125
0011	3/16	0.1875
0100	4/16	0.25
0101	5/16	0.3125
0110	6/16	0.375
0111	7/16	0.4375
1000	8/16	0.5
1001	9/16	0.5625
1010	10/16	0.625
1011	11/16	0.6875
1100	12/16	0.75
1101	13/16	0.8125
1110	14/16	0.875
1111	15/16	0.9375

The programmable fractional baud rate divisor enables a finer resolution of baud clock than the conventional integer divider. The programmable fractional baud clock divider allows for the programmability of both an integer divisor as well as fractional component. The average frequency of the baud clock from the fractional baud rate divisor is dependent upon both the integer divisor and the fractional component, thereby providing a finer resolution to the average frequency of the baud clock.

Equation (2)

$$\text{Baud Rate Divisor} = \frac{\text{Serial Clock Frequency}}{(16 \times \text{Required Baud Rate})} = \text{BRD}_\text{I} + \text{BRD}_\text{F}$$

Where,

BRDI - Integer part of the divisor.

BRDF - Fractional part of the divisor.

22.15.4.1 Fractional Division Used to Generate Baud Clock

Fractional division of clock is used by the N/N+1 divider, where N is the integer part of the divisor. N/N+1 division works on the basis of achieving the required average timing over a long period by alternating the division between two numbers. If N=1 and ratio of N/N+1 is same, which means equal number of divide by 1 and divide by 2 over a period of time, average time period would come out to be divided by 1.5. Varying the ratio of N/N+1 any value can be achieved above 1 and below 2.

22.15.4.2 Calculating the Fractional Value Error PI

Following is a sample for calculating the fractional value error.

Consider the following values:

- Required Baud Rate (RBR) = 4454400
- Serial Clock (SCLK) = 200MHz
- DLF_SIZE = 4

Then, as per equation (2), Baud Rate Divisor (BRD) is as follows:

$$BRD = \frac{200}{16 \times 4454400} = 2.8062140804$$

The integer and fractional parts are as follows:

- Integer part (BRDI) = 2
- Fractional part (BRDF) = 0.8062140804

Therefore, Baud Rate Divisor Latch Fractional Value (DLF) is as follows:

$$DLF = BRD_F \times 2^{DLF\_SIZE} = 0.8062140804 \times 16 = 12.899425286 = 13\ (roundoff\ value)$$

The Generated Baud Rate Divider (GD) is as follows:

$$GD = BRD_I + \frac{DLF}{2^{DLF\_SIZE}} = 2 + \frac{13}{16} = 2.8125$$

Therefore, the Generated Baud Rate (GBR) is as follows:

$$GBR = \frac{Serial\ Clock}{(16 \times GD)} = \frac{200}{16 \times 2.8125} = 4444444.44$$

Now the error is calculated as follows:

$$Error = \frac{GBR - RBR}{RBR} = 0.002234$$

The error percentage is as follows:

$$Error\ \% = 0.002234 \times 100 = 0.223$$

22.15.5 FIFO Support

When FIFO Access mode has been selected it can be enabled with the FIFO Access Register (FAR[0]). Once enabled, the control portions of the transmit and receive FIFOs are reset and the FIFOs are treated as empty.

Data can be written to the transmit FIFO as normal; however no serial transmission occurs in this mode— normal operation halted—and thus no data leave the FIFO. The data that has been written to the transmit FIFO can be read back with the Transmit FIFO Read (TFR) register, which when read gives the current data at the top of the transmit FIFO.

Similarly, data can be read from the receive FIFO as normal. Since the normal operation of the UART controller is halted in this mode, data must be written to the receive FIFO so the data can be read back.

Data is written to the receive FIFO using the Receive FIFO Write (RFW) register. The upper two bits of the 10-bit register are used to write framing error and parity error detection information to the receive FIFO, as follows:

- RFW[9] indicates framing error
- RFW[8] indicates parity error

Although these bits cannot be read back through the Receive Buffer Register, they can be checked by reading the Line Status Register and checking the corresponding bits when the data in question is at the top of the receive FIFO.

22.15.6 Interrupts

Assertion of the UART controller interrupt output signal, a positive-level interrupt, occurs whenever one of the several prioritized interrupt types are enabled and active.

When an interrupt occurs, the initiator accesses the IIR register.

The following interrupt types can be enabled with the IER register:

- Receiver Error
- Receiver Data Available
- Character Timeout (in FIFO mode only)
- Transmitter Holding Register Empty at/below threshold (in Programmable THRE interrupt mode)
- Modem Status
- Busy Detect Indication

22.15.7 Auto Flow Control

The UART Controller can be configured to have a 16750-compatible Auto RTS and Auto CTS serial data flow control mode available; if FIFOs are not implemented, this mode cannot be selected. When Auto Flow Control is not selected, none of the corresponding logic is implemented and the mode cannot be enabled, reducing overall gate counts. When Auto Flow Control mode is selected, it can be enabled with the Modem Control Register (MCR[5]).

Auto RTS and Auto CTS are described as follows:

- Auto RTS ¨C Becomes active when the following occurs:
  - — RTS (MCR[1] bit and MCR[5]bit are both set)
  - — FIFOs are enabled (FCR[0]) bit is set)
  - — SIR mode is disabled (MCR[6] bit is not set)

When Auto RTS is enabled, the rts_n output is forced inactive (high) when the receiver FIFO level reaches the threshold set by FCR[7:6], but only if the RTC flow-control trigger is disabled. Otherwise, the rts_n output is forced inactive (high) when the FIFO is almost full, where almost full refers to two available slots in the FIFO. When rts_n is connected to the cts_n input of another UART device, the other UART stops sending serial data until the receiver FIFO has available space; that is, until it is completely empty.

The selectable receiver FIFO threshold values are:

- 1
- 1/4
- 1/2
- 2 less than full

Since one additional character can be transmitted to the UART Controller after rts_n has become inactive—due to data already having entered the transmitter block in the other UART—setting the threshold to “2 less than full” allows maximum use of the FIFO with a safety zone of one character.

Once the receiver FIFO becomes completely empty by reading the Receiver Buffer Register (RBR), rts_n again becomes active (low), signaling the other UART to continue sending data.

Note: Even if everything else is selected and the correct MCR bits are set, if the FIFOs are disabled through FCR[0] or the UART is in SIR mode (MCR[6] is set to 1), Auto Flow Control is also disabled. When Auto RTS is not implemented or disabled, rts_n is controlled solely by MCR[1].

- **Auto CTS** - becomes active when the following occurs:
  - AFCE (MCR[5] bit = 1)
  - FIFOs are enabled through FIFO Control Register FCR[0] bit
  - SIR mode is disabled (MCR[6] bit = 0)

When Auto CTS is enabled (active), the UART Controller transmitter is disabled whenever the cts_n input becomes inactive (high); this prevents overflowing the FIFO of the receiving UART.

If the cts_n input is not inactivated before the middle of the last stop bit, another character is transmitted before the transmitter is disabled. While the transmitter is disabled, the transmitter FIFO can still be written to, and even overflowed.

Therefore, when using this mode, the following happens:

- UART status register can be read to check if transmit FIFO is full (USR[1] set to 0)
- Current FIFO level can be read using TFL register
- Programmable THRE Interrupt mode must be enabled to access "FIFO full" status using Line Status Register (LSR)

When using the "FIFO full" status, software can poll this before each write to the Transmitter FIFO; for details, see "Section 22.15.3.5". When the cts_n input becomes active (low) again, transmission resumes.

Note: When everything else is selected, if the FIFOs are disabled using FCR[0], Auto Flow Control is also disabled. When Auto CTS is not implemented or disabled, the transmitter is unaffected by cts_n.

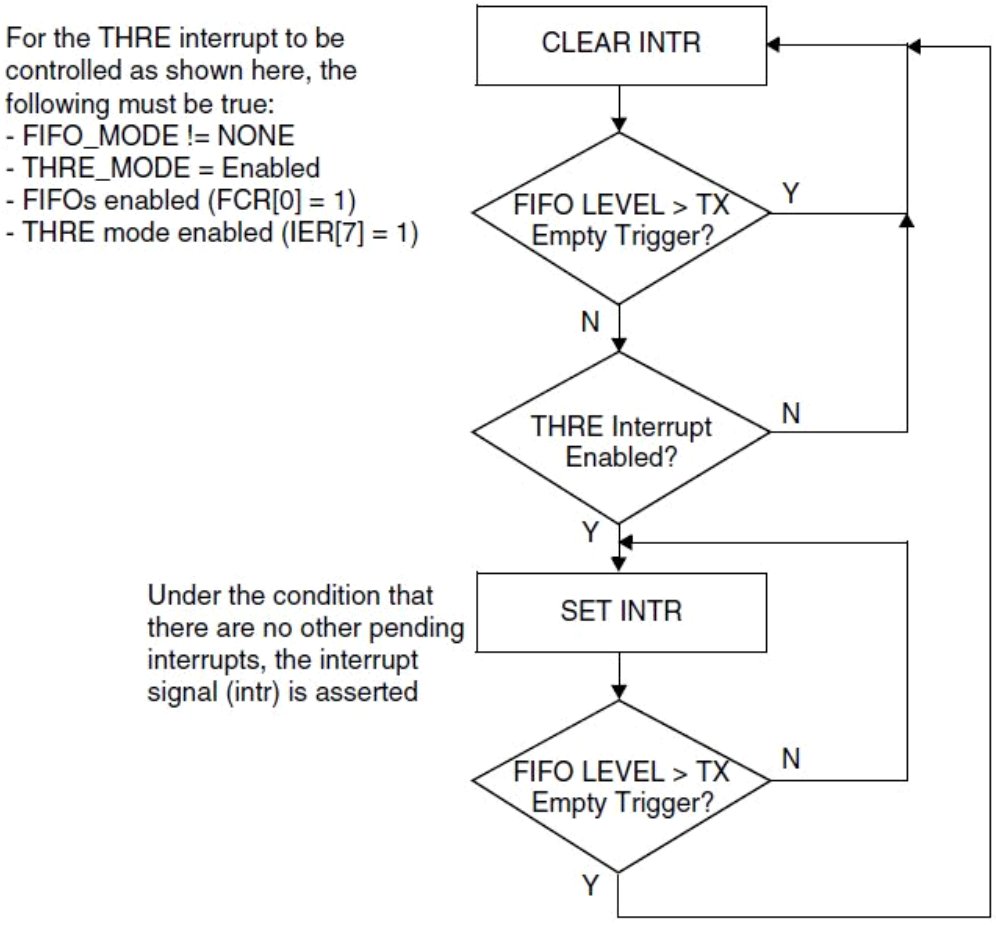
22.15.8 Programmable THRE Interrupt

The UART Controller can be configured for a Programmable THRE Interrupt mode in order to increase system performance;

- When Programmable THRE Interrupt mode is selected, it can be enabled using the Interrupt Enable Register (IER[7]).

When FIFOs and THRE mode are enabled, the THRE Interrupts and dma_tx_req_n are active at, and below, a programmed transmitter FIFO empty threshold level, as opposed to empty, as shown in the flowchart in Figure 22-21.

Figure 22-21.Flowchart of Interrupt Generation for Programmable THRE Interrupt Mode



The threshold level is programmed into FCR[5:4]. Available empty thresholds are:

- empty
- 2
- 1/4
- 1/2

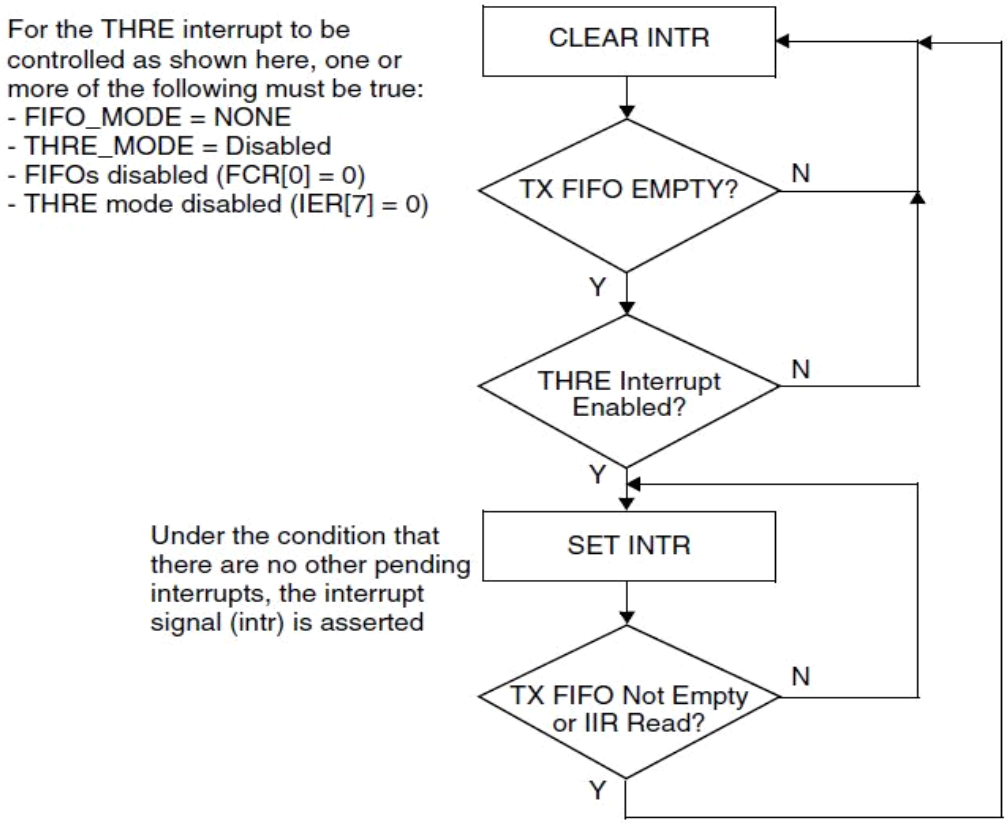
Selection of the best threshold value depends on the system's ability to begin a new transmission sequence in a timely manner. However, one of these thresholds should be optimal for increasing system performance by preventing the transmitter FIFO from running empty.

In addition to the interrupt change, the Line Status Register (LSR[5]) also switches from indicating that the transmitter FIFO is empty to the FIFO being full. This allows software to fill the FIFO for each transmit sequence by polling LSR[5] before writing another character. The flow then allows the transmitter FIFO to be filled whenever an interrupt occurs and there is data to transmit, rather than waiting until the FIFO is

completely empty. Waiting until the FIFO is empty causes a reduction in performance whenever the system is too busy to respond immediately. Further system efficiency is achieved when this mode is enabled in combination with Auto Flow Control.

Even if everything else is selected and enabled, if the FIFOs are disabled using the FCR[0] bit, the Programmable THRE Interrupt mode is also disabled. When not selected or disabled, THRE interrupts and the LSR[5] bit function normally, signifying an empty THR or FIFO. Figure 22-22 illustrates the flowchart of THRE interrupt generation when not in programmable THRE interrupt mode.

Figure 22-22.Flowchart of Interrupt generation when not in Programmable THRE Interrupt Mode



22.15.9 DMA Modes

The UART controller uses two DMA channels. One for transmit data and one for receive data. There are two

DMA modes:

- mode 0 . bit 3 of FIFO Control Register set to 0
- mode 1 . bit 3 of FIFO Control Register set to 1

22.15.9.1 DMA Mode 0

DMA mode 0 supports single DMA data transfers at a time.

In mode 0, the DMA Transmit request:

- Goes active under the following conditions:
  - When Transmitter Holding Register is empty in non-FIFO mode
  - When transmitter FIFO is empty in FIFO mode with Programmable THRE interrupt mode disabled
  - When transmitter FIFO is at or below programmed threshold with Programmable THRE interrupt mode enabled
- Goes inactive when:
  - Single character has been written into Transmitter Holding Register or transmitter FIFO with Programmable THRE interrupt mode disabled
  - Transmitter FIFO is above threshold with Programmable THRE interrupt mode enabled

In mode 0, the DMA Receive request:

- Goes active when single character is available in Receiver FIFO or Receive Buffer Register
- Goes inactive when Receive Buffer Register or Receiver FIFO are empty, depending on FIFO mode

22.15.9.2 DMA Mode 1

DMA mode 1 supports multi-DMA data transfers, where multiple transfers are made continuously until the receiver FIFO has been emptied or the transmit FIFO has been filled.

In mode 1, the DMA Transmit request signal is asserted:

- When transmitter FIFO is empty with Programmable THRE interrupt mode disabled
- When transmitter FIFO is at or below programmed threshold with Programmable THRE interrupt mode enabled

In mode 1, the DMA Transmit request signal is de-asserted when the transmitter FIFO is completely full.

In mode 1, the DMA Receive request signal is asserted:

- When Receiver FIFO is at or above programmed trigger level
- When character timeout has occurred; ERBFI does not need to be set

In mode 1, the DMA Receive request signal is de-asserted when the receiver FIFO becomes empty.

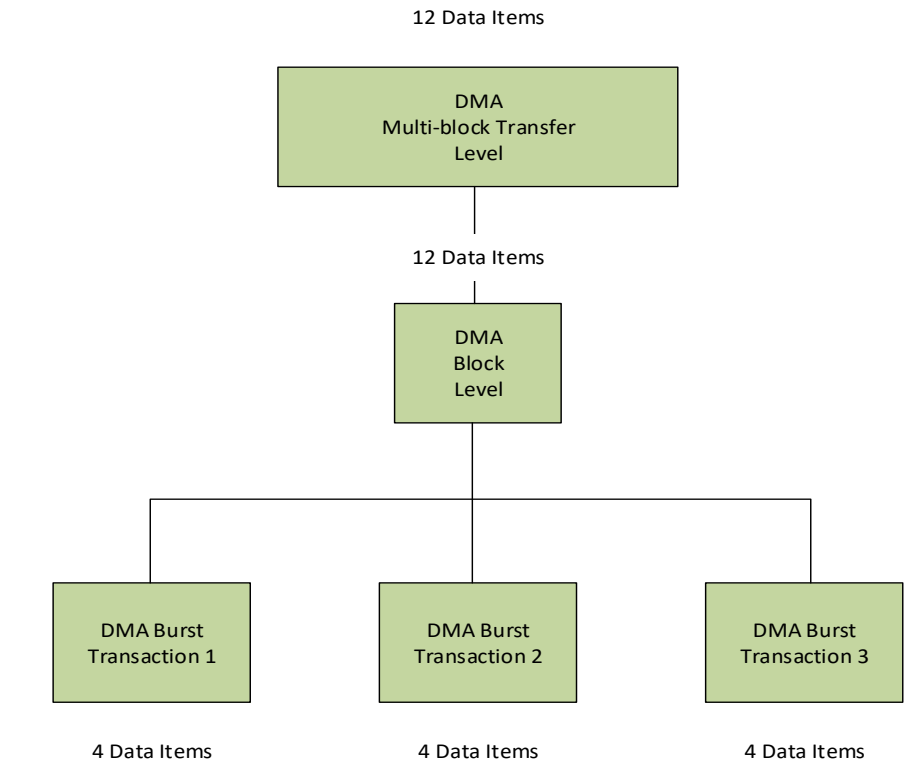
22.15.9.3 Example DMA Flow

As a block flow control device, the DMA Controller is programmed by the processor with the number of data items (block size) that are to be transmitted or received by the UART controller; this is programmed into the BLOCK_TS field of the CTLx register.

The block is broken into a number of transactions, each initiated by a request from the UART controller. The DMA Controller must also be programmed with the number of data items (in this case, UART controller FIFO entries) to be transferred for each DMA request. This is also known as the burst transaction length, and is programmed into the SRC_MSIZE/DEST_MSIZE fields of the DMA controller's CTLx register for source and destination, respectively.

The following figure shows a single block transfer, where the block size programmed into the DMA Controller is 12 and the burst transaction length is set to 4.

Figure 22-23.Breakdown of DMA Transfer into Burst Transaction

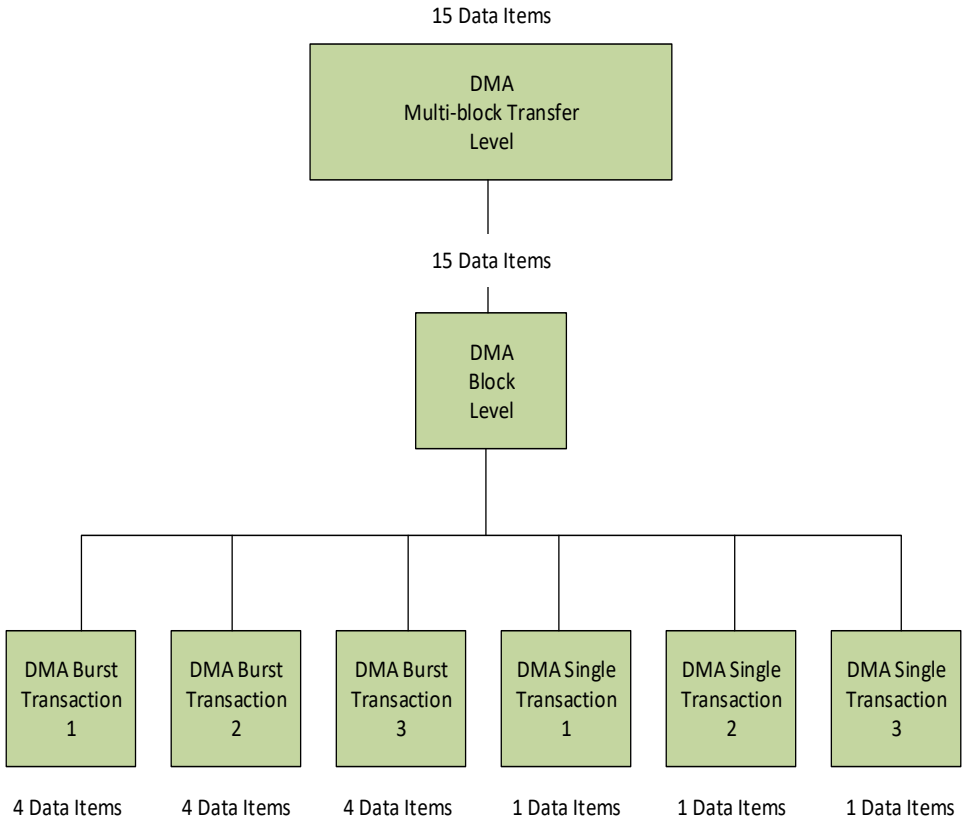


Block Size : DMA.CTLx.BLOCK_TS=12
Number of data items per source burst transaction : DMA.CTLx.SRC_MSIZE = 4
For a FIFO depth of 16: UART.FCR[7:6] = 01 = FIFO ¼ full = DMA.CTLx.SRC_MSIZE

In this case, the block size is a multiple of the burst transaction length. Therefore, the DMA block transfer consists of a series of burst transactions. If the UART controller makes a transmit request to this channel, four data items are written to the UART controller transmit FIFO. Similarly, if the UART controller makes a receive request to this channel, four data items are read from the UART controller receive FIFO. Three separate requests must be made to this DMA channel before all twelve data items are written or read.

When the block size programmed into the DMA Controller is not a multiple of the burst transaction length, as shown in the following figure, a series of burst transactions followed by single transactions are needed to complete the block transfer.

Figure 22-24.Breakdown of DMA Transfer into Single and Burst Transactions



Block Size : DMA.CTLx.BLOCK_TS=15
Number of data items per burst transaction : DMA.CTLx.DEST_MSIZE = 4
For a FIFO depth of 16: UART.FCR[5:4] = 10 = FIFO ¼ full = 4 = DMA.CTLx.DEST_MSIZE

22.15.9.4 Transmit Watermark Level and Transmit FIFO Underflow

During UART controller serial transfers, transmit FIFO requests are made to the DMA controller whenever the number of entries in the transmit FIFO is less than or equal to the decoded level of the Transmit Empty Trigger (TET) of the FCR register (bits 5:4); this is known as the watermark level. The DMA controller responds by writing a burst of data to the transmit FIFO buffer, of length CTLx.DEST_MSIZE.

Data should be fetched from the DMA often enough for the transmit FIFO to perform serial transfers continuously; that is, when the FIFO begins to empty, another DMA request should be triggered. Otherwise the FIFO runs out of data (underflow). To prevent this condition, you must set the watermark level correctly.

22.15.9.5 Choosing Watermark Level

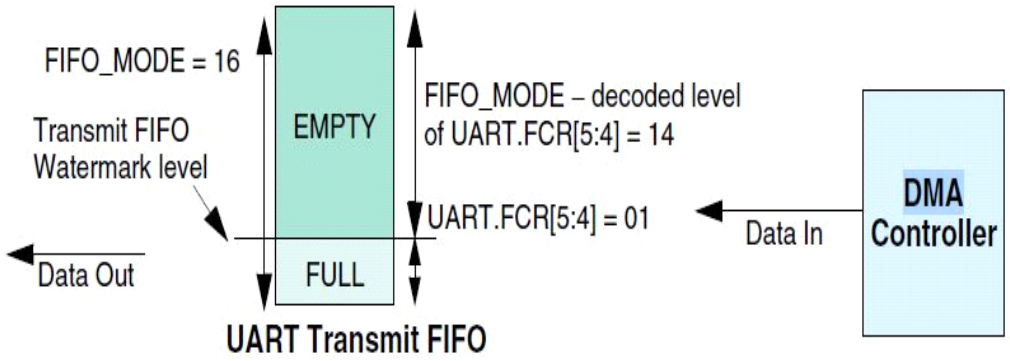
Consider the example where the following assumption is made:

DMA.CTLx.DEST_MSIZE = FIFO_DEPTH - UART.FCR[5:4]

The number of data items to be transferred in a DMA burst is equal to the empty space in the Transmit FIFO. Consider two different watermark level settings.

22.15.9.5.1 Case 1: FCR[5:4] = 01 — decodes to 2

Figure 22-25.Case 1 Watermark Levels



- Transmit FIFO watermark level = decoded level of UART.FCR[5:4] = 2
- DMA.CTLx.DEST_MSIZE = FIFO_MODE . UART.FCR[5:4] = 14
- UART transmit FIFO_MODE = 16
- DMA.CTLx.BLOCK_TS = 56

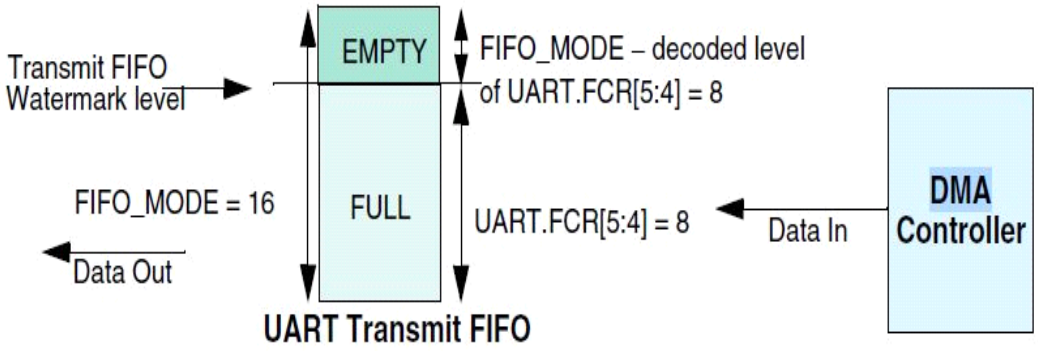
Therefore, the number of burst transactions needed equals the block size divided by the number of data items per burst:

DMA.CTLx.BLOCK_TS/DMA.CTLx.DEST_MSIZE = 56/14 = 4

The number of burst transactions in the DMA block transfer is 4., but the watermark level—decoded level of UART.FCR[5:4]—is quite low. Therefore, the probability of a UART underflow is high where the UART serial transmit line needs to transmit data, but where there is no data left in the transmit FIFO. This occurs because the DMA has not had time to service the DMA request before the transmit FIFO becomes empty.

22.15.9.5.2 Case 2: FCR[5:4] = 11 — FIFO 1/2 full (decodes to 8)

Figure 22-26.Case 2 Watermark Levels



- Transmit FIFO watermark level = decoded level of UART.FCR[5:4] = 8
- DMA.CTLx.DEST_MSIZE = FIFO_MODE - UART.FCR[5:4] = 8
- UART transmit FIFO_MODE = 16
- DMA.CTLx.BLOCK_TS = 56

Number of burst transactions in Block:

DMA.CTLx.BLOCK_TS/DMA.CTLx.DEST_MSIZE = 56/8 = 7

In this block transfer, there are seven destination burst transactions in a DMA block transfer, but the watermark level.decoded level of UART.FCR[5:4] is high. Therefore, the probability of a UART underflow is low because the DMA controller has enough time to service the destination burst transaction request before the UART transmit FIFO becomes empty.

Thus, the second case has a lower probability of underflow at the expense of more burst transactions per block. This provides a potentially greater amount of bus bursts per block and worse bus utilization than Case 1.

Therefore, the goal in choosing a watermark level is to minimize the number of transactions per block, while at the same time keeping the probability of an underflow condition to an acceptable level. In practice, this is a function of the ratio of:

rate of UART data transmission: rate of DMA response to destination burst requests

For example, both of the following increases the rate at which the DMA controller can respond to burst transaction requests:

- Promoting channel to highest priority channel in DMA
- Promoting DMA initiator interface to highest priority initiator in bus layer

This in turn enables the user to decrease the watermark level, which improves bus utilization without compromising the probability of an underflow occurring.

22.15.9.6 Selecting DEST_MSIZE and Transmit FIFO Overflow

As can be seen from Figure 22-26, programming DMA.CTLx.DEST_MSIZE to a value greater than the watermark level that triggers the DMA request can cause overflow when there is not enough space in the UART transmit FIFO to service the destination burst request. Therefore, use the following in order to avoid overflow:

DMA.CTLx.DEST_MSIZE <= UART.FIFO_DEPTH - decoded level of UART.FCR[5:4] (1)

In Case 2: FCR[5:4] = 11 — FIFO 1/2 full (decodes to 8), the amount of space in the transmit FIFO at the time the burst request is made is equal to the destination burst length, DMA.CTLx.DEST_MSIZE. Thus, the transmit FIFO can be full, but not overflowed, at the completion of the burst transaction.

Therefore, for optimal operation, DMA.CTLx.DEST_MSIZE should be set at the FIFO level that triggers a transmit DMA request; that is:

DMA.CTLx.DEST_MSIZE = UART.FIFO_DEPTH - decoded level of UART.FCR[5:4] (2)

This is the setting used in Figure 22-39.

Adhering to equation (2) reduces the number of DMA bursts needed for a block transfer, which in turn improves bus utilization.

Adhering to equation (2) reduces the number of DMA bursts needed for a block transfer, which in turn improves bus utilization.

22.15.9.7 Receive Watermark Level and Receive FIFO Overflow

During UART controller serial transfers, receive FIFO requests are made to the DMA controller whenever the number of entries in the receive FIFO is at or above the decoded level of Receiver Trigger (RT) of the FCR[7:6]. This is known as the watermark level. The DMA Controller responds by fetching a burst of data from the receive FIFO buffer of length CTLx.SRC_MSIZE.

Data should be fetched by the DMA often enough for the receive FIFO to accept serial transfers continuously; that is, when the FIFO begins to fill, another DMA transfer is requested. Otherwise, the FIFO fills with data (overflow). To prevent this condition, you must correctly set the watermark level.

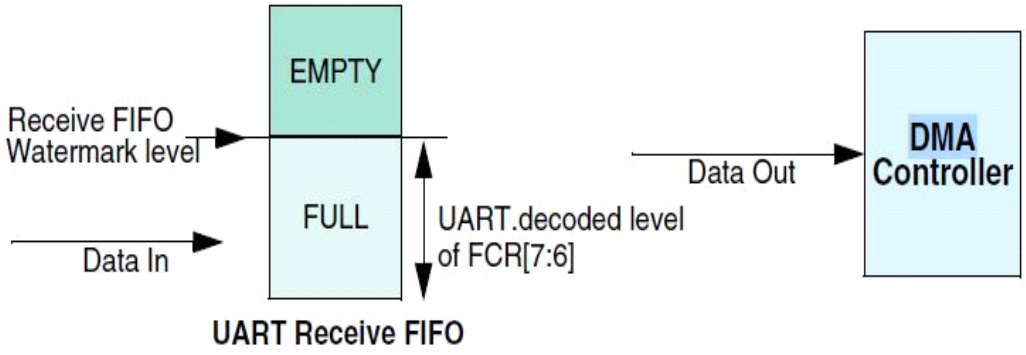
22.15.9.8 Choosing Receive Watermark Level

Similar to choosing the transmit watermark level described earlier, the receive watermark level—decoded level of FCR[7:6]—should be set to minimize the probability of overflow. It is a trade-off between the number of DMA burst transactions required per block versus the probability of an overflow occurring.

22.15.9.9 Selecting SRC_MSIZE and Receive FIFO Underflow

As can be seen in Figure 22-27, programming a source burst transaction length greater than the watermark level can cause underflow when there is not enough data to service the source burst request. Therefore, equation (3) below must be adhered to in order to avoid underflow.

Figure 22-27.UART Receive FIFO



If the number of data items in the receive FIFO is equal to the source burst length at the time the burst request is made – DMA.CTLx.SRC_MSIZE – the receive FIFO can be emptied, but not underflowed, at the completion of the burst transaction. For optimal operation, DMA.CTLx.SRC_MSIZE should be set at the watermark level; that is:

DMA.CTLx.SRC_MSIZE = decoded level of FCR[7:6] (3)

Adhering to equation (3) reduces the number of DMA bursts in a block transfer, and this in turn can improve bus utilization.

Note: The receive FIFO is not empty at the end of the source burst transaction if the UART has successfully received one data item or more on the UART serial receive line during the burst.

22.15.10 Signal Description

Table 22-14. UART Signal Description (Sheet 1 of 2)

Signal Name	Type	Description
PSE_UART0_TXD	O	Transmit
PSE_UART0_RXD	I	Receive
PSE_UART0_RTS_N	O	Ready to Send
PSE_UART0_CTS_N	I	Clear to Send
PSE_HSUART0_DE	O	Driver Enabled (DE) signal to external transceiver
PSE_HSUART0_RE	O	Receiver Enabled (RE) signal to external transceiver
PSE_HSUART0_EN	O	RS-232 (Logic Low)/RS-485 (Logic High) mode enable to external transceiver
PSE_UART1_TXD	O	Transmit
PSE_UART1_RXD	I	Receive
PSE_UART1_RTS_N	O	Ready to Send
PSE_UART1_CTS_N	I	Clear to Send
PSE_HSUART1_DE	O	Driver Enabled (DE) signal to external transceiver

Table 22-14. UART Signal Description (Sheet 2 of 2)

Signal Name	Type	Description
PSE_HSUART1_RE	O	Receiver Enabled (RE) signal to external transceiver
PSE_HSUART1_EN	O	RS-232 (Logic Low)/RS-485 (Logic High) mode enable to external transceiver
PSE_UART2_TXD	O	Transmit
PSE_UART2_RXD	I	Receive
PSE_UART2_RTS_N	O	Ready to Send
PSE_UART2_CTS_N	I	Clear to Send
PSE_HSUART2_DE	O	Driver Enabled (DE) signal to external transceiver
PSE_HSUART2_RE	O	Receiver Enabled (RE) signal to external transceiver
PSE_HSUART2_EN	O	RS-232 (Logic Low)/RS-485 (Logic High) mode enable to external transceiver
PSE_UART3_TXD	O	Transmit
PSE_UART3_RXD	I	Receive
PSE_UART3_RTS_N	O	Ready to Send
PSE_UART3_CTS_N	I	Clear to Send
PSE_HSUART3_DE	O	Driver Enabled (DE) signal to external transceiver
PSE_HSUART3_RE	O	Receiver Enabled (RE) signal to external transceiver
PSE_HSUART3_EN	O	RS-232 (Logic Low)/RS-485 (Logic High) mode enable to external transceiver
PSE_UART4_TXD	O	Transmit
PSE_UART4_RXD	I	Receive
PSE_UART4_RTS_N	O	Ready to Send
PSE_UART4_CTS_N	I	Clear to Send
PSE_UART5_TXD	O	Transmit
PSE_UART5_RXD	I	Receive
PSE_UART5_RTS_N	O	Ready to Send
PSE_UART5_CTS_N	I	Clear to Send

22.15.11 Registers

Please refer to chapter 12 of the Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors for Internet of Things (IoT) Applications, Datasheet, Volume 2 (Book 3 of 3), Intel® Programmable Services Engine (Intel® PSE) (Document Number: 636723), for a description of the registers associated with subject of this chapter.

22.16 SPI Controller

22.16.1 Overview

SPI is a four-wire, bi-directional serial bus that provides simple and efficient method of data transmission over a short distance between many devices. SPI is used typically for connecting the Intel® PSE to external sensor devices, such as combo accelerometers, gyroscopes, compass devices.

22.16.1.1 Ownership Allocation

- SPI0 ownership can be controlled using OWNERSHIP_DEV14 field in Ownership Control 1 register
- SPI1 ownership can be controlled using OWNERSHIP_DEV15 field in Ownership Control 1 register
- SPI2 ownership can be controlled using OWNERSHIP_DEV16 field in Ownership Control 2 register
- SPI3 ownership can be controlled using OWNERSHIP_DEV17 field in Ownership Control 2 register

22.16.2 Features

The SPI controller support includes:

- SPI supports Motorola SPI protocols.
- Four controllers supporting two targets each.
- 4 combinations of polarity and phase - also referred to as modes
- Full-duplex and half-duplex mode of operation
- Programmable SPI clock frequency range with max of 50MHz (1.8v operation), 25Mhz (3.3v operation)
- FIFO of 64 bytes with programmable watermarks/thresholds
- PIO Transfers by reading/writing to the FIFO memory mapped register
- DMA HW hook based transfer
  - — DMA channel moves the data from the SPI RX FIFO by performing a read to a memory-mapped register, upon data available indication.
  - — DMA channel moves the data to the SPI TX FIFO by performing a write to a memory-mapped register, upon space available indication.
- Programmable character length (2 to 16 bits)
- Programmable clock phase (delay or no delay)
- Programmable clock polarity (high or low)

22.16.3 Functional Description

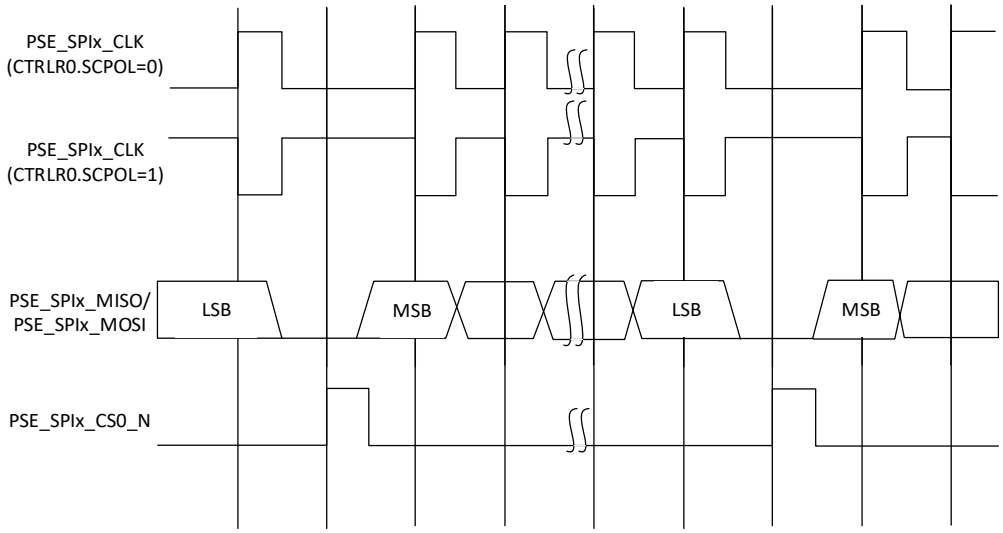
22.16.3.1 Connection Interfaces

The following interfaces are valid only when the frame format (FRF) is set to Motorola SPI.

The clock polarity (SCPOL) configuration parameter determines whether the inactive state of the serial clock is high or low.

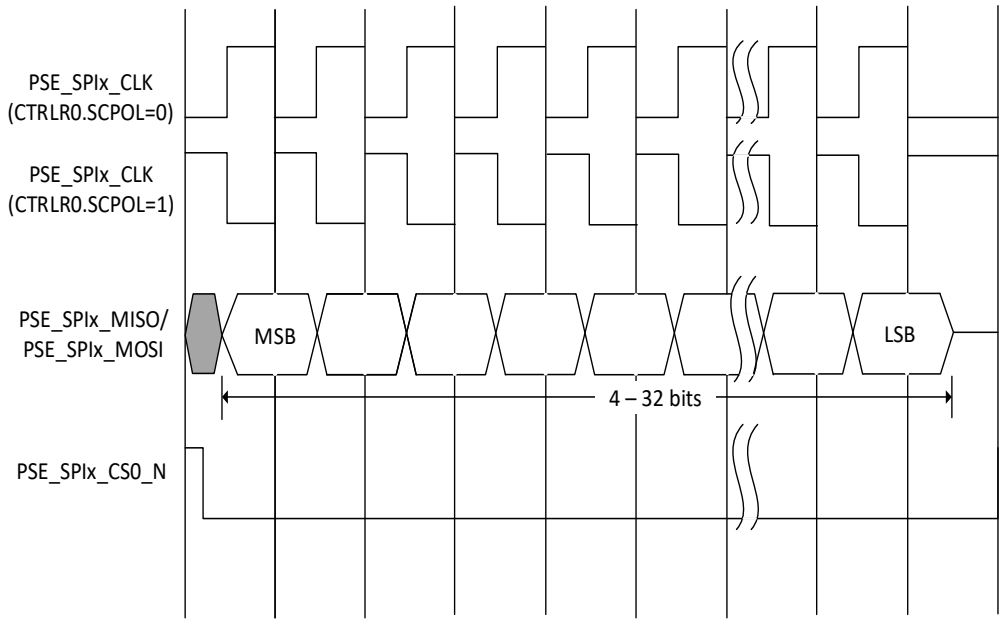
When the configuration parameter SCPH = 0, data transmission begins on the falling edge of the target select signal. The first data bit is captured by the initiator and target peripherals on the first edge of the serial clock; data are propagated on the second edge of the serial clock.

Figure 22-28.Serial Format Continuous Transfers (SCPH = 0) when SSI_SCPH0_SSTOGGLE = 1



When the configuration parameter SCPH = 1, both initiator and target peripherals begin transmitting data on the first serial clock edge after the target select line is activated. The first data bit is captured on the second (trailing) serial clock edge. Data are propagated by the initiator and target peripherals on the leading edge of the serial clock.

Figure 22-29.SPI Serial Format (SCPH=1)



22.16.3.2 Transfer Mode

When transferring data on the serial bus, the SPI controller operates in the modes discussed in this section. The transfer mode (TMOD) is set by writing to control register 0 (CTRLR0).

22.16.3.3 Transmit and Receive

When TMOD = 2‘b00, both transmit and receive logic are valid. The data transfer occurs as normal according to the selected frame format (serial protocol). Transmit data are popped from the transmit FIFO and sent through the txd line to the target device, which replies with data on the rxd line. The receive data from the target device is moved from the receive shift register into the receive FIFO at the end of each data frame.

Note: Chip select (PSE_SPI_CSx_N) will be automatically de-asserted when TX FIFO is empty.

22.16.3.4 Transmit only

When TMOD = 2‘b01, the receive data are invalid and should not be stored in the receive FIFO. The data transfer occurs as normal, according to the selected frame format (serial protocol). Transmit data are popped from the transmit FIFO and sent through the txd line to the target device, which replies with data on the rxd line. At the end of the data frame, the receive shift register does not load its newly received data into the receive FIFO. The data in the receive shift register is overwritten by the next transfer. You should mask interrupts originating from the receive logic when this mode is entered.

22.16.3.5 Receive Only

When TMOD = 2'b10, the transmit data are invalid. When configured as a target, the transmit FIFO is never popped in Receive Only mode. The txd output remains at a constant logic level during the transmission.

The data transfer occurs as normal according to the selected frame format (serial protocol). The receive data from the target device is moved from the receive shift register into the receive FIFO at the end of each data frame. You should mask interrupts originating from the transmit logic when this mode is entered.

22.16.3.6 EEPROM Memory Read

When TMOD = 2'b11, the transmit data is used to transmit an opcode and/or an address to the EEPROM device. Typically this takes three data frames (8-bit opcode followed by 8-bit upper address and 8-bit lower address). During the transmission of the opcode and address, no data is captured by the receive logic (as long as the SPI controller initiator is transmitting data on its txd line, data on the rxd line is ignored). The SPI controller initiator continues to transmit data until the transmit FIFO is empty. Therefore, you should ONLY have enough data frames in the transmit FIFO to supply the opcode and address to the EEPROM. If more data frames are in the transmit FIFO than are needed, then read data is lost.

When the transmit FIFO becomes empty (all control information has been sent), data on the receive line (rxd) is valid and is stored in the receive FIFO; the txd output is held at a constant logic level. The serial transfer continues until the number of data frames received by the SPI controller initiator matches the value of the NDF field in the CTRLR1 register + 1.

22.16.4 Clocking

When SPI controller is configured as a initiator device, the maximum frequency of the bit-rate clock (PSE_SPIx_CLK) is one-half the frequency of SCLK_IN (100MHz). This allows the shift control logic to capture data on one clock edge of PSE_SPIx_CLK and propagate data on the opposite edge.

The PSE_SPIx_CLK line toggles only when an active transfer is in progress. At all other times it is held in an inactive state, as defined by the serial protocol under which it operates.

The frequency of PSE_SPIx_CLK can be derived from the following equation:

$$F_{PSE\_SPIx\_CLK} = \frac{F_{SCLK\_IN}}{SCKDV}$$

SCKDV is a bit field in the programmable register BAUDR, holding any even value in the range 0 to 65,534. If SCKDV is 0, then PSE_SPIx_CLK is disabled.

When SPI controller is configured as a target device, the minimum frequency of operation depends on the operation of the target peripheral.

If the target device is receive only, the maximum frequency supported is SCLK_IN/6 i.e. 100MHz/6 = 16.6MHz.

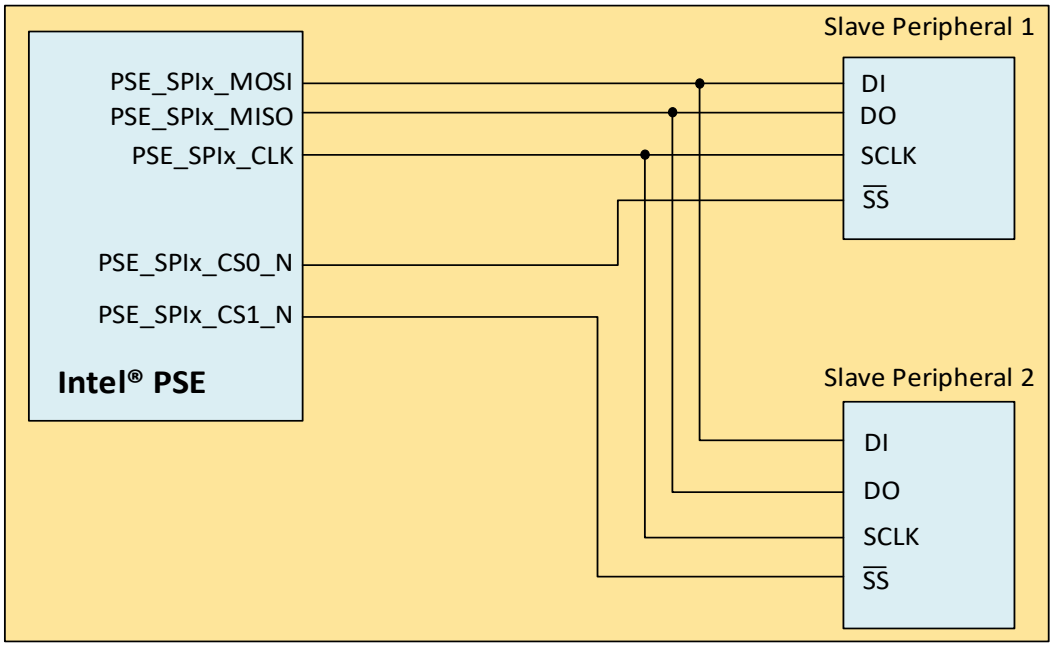
If the target device is transmit and receive, the maximum frequency supported is SCLK_IN/12 i.e 100MHz/12 = 8.33MHz.

22.16.5 Operations Mode

22.16.5.1 Serial Initiator Mode

This mode enables serial communication with serial-target peripheral devices. When configured as a serial-initiator device, the SPI controller starts and controls all serial transfers. The following figure shows an example of the SPI controller configured as a serial initiator with all other devices on the serial bus configured as serial targets.

Figure 22-30.SPI controller Configured as Initiator Device



The serial bit-rate clock, generated and controlled by the SPI controller, is driven out on the PSE_SPIx_CLK line. When the SPI controller is disabled (SSI_EN = 0), no serial transfers can occur and PSE_SPIx_CLK is held in "inactive" state, as defined by the serial protocol under which it operates.

22.16.6 Transmit and Receive FIFO Buffers

The FIFO buffers have 64 entries depth The width of both transmit and receive FIFO buffers is fixed at 16 Each data entry in the FIFO buffers contains a single data frame. It is impossible to store multiple data frames in a single FIFO location; for example, you may not store two 8-bit data frames in a single FIFO location. If an 8-bit data frame is required, the upper 8-bits of the FIFO entry are ignored or unused when the serial shifter transmits the data.

The transmit FIFO is loaded by write commands to the SPI controller data register (DR). Data are popped (removed) from the transmit FIFO by the shift control logic into the transmit shift register. The transmit FIFO generates a FIFO empty interrupt request

(SR.TFE & IMR.TXEIM) when the number of entries in the FIFO is less than or equal to the FIFO threshold value. The threshold value, set through the programmable register TXFTLR, determines the level of FIFO entries at which an interrupt is generated. The threshold value allows you to provide early indication to the processor that the transmit FIFO is nearly empty. A transmit FIFO overflow interrupt (IMR.TXOIM) is generated if you attempt to write data into an already full transmit FIFO.

Data are popped from the receive FIFO by read commands to the SPI controller data register (DR). The receive FIFO is loaded from the receive shift register by the shift control logic. The receive FIFO generates a FIFO-full interrupt request (SR.RFF & IMR.RXFIM) when the number of entries in the FIFO is greater than or equal to the FIFO threshold value plus 1. The threshold value, set through programmable register RXFTLR, determines the level of FIFO entries at which an interrupt is generated.

The threshold value allows you to provide early indication to the processor that the receive FIFO is nearly full. A receive FIFO overrun interrupt (IMR.RXOIM) is generated when the receive shift logic attempts to load data into a completely full receive FIFO. However, this newly received data are lost. A receive FIFO underflow interrupt (IMR.RXUIM) is generated if you attempt to read from an empty receive FIFO. This alerts the processor that the read data are invalid.

The following table provides description for different Transmit FIFO Threshold values.

Table 22-15. Transmit FIFO Threshold (TFT) Decode Values

TFT Value	Description
0000_0000	Transmit FIFO empty interrupt request is asserted when 0 data entries are present in transmit FIFO
0000_0001	Transmit FIFO empty interrupt request is asserted when 1 or less data entry is present in transmit FIFO
0000_0010	Transmit FIFO empty interrupt request is asserted when 2 or less data entries are present in transmit FIFO
0000_0011	Transmit FIFO empty interrupt request is asserted when 3 or less data entries are present in transmit FIFO
...	...
...	...
1111_1100	Transmit FIFO empty interrupt request is asserted when 252 or less data entries are present in transmit FIFO
1111_1101	Transmit FIFO empty interrupt request is asserted when 253 or less data entries are present in transmit FIFO
1111_1110	Transmit FIFO empty interrupt request is asserted when 254 or less data entries are present in transmit FIFO
1111_1111	Transmit FIFO empty interrupt request is asserted when 255 or less data entries are present in transmit FIFO

Table 22-16. Receive FIFO Threshold (TFT) Decode Values

RFT Value	Description
0000_0000	Receive FIFO full interrupt request is asserted when 1 or more data entry is present in receive FIFO
0000_0001	Receive FIFO full interrupt request is asserted when 2 or more data entries are present in receive FIFO

Table 22-16. Receive FIFO Threshold (TFT) Decode Values

RFT Value	Description
0000_0010	Receive FIFO full interrupt request is asserted when 3 or more data entries are present in receive FIFO
0000_0011	Receive FIFO full interrupt request is asserted when 4 or more data entries are present in receive FIFO
...	...
...	...
1111_1100	Receive FIFO full interrupt request is asserted when 253 or more data entries are present in receive FIFO
1111_1101	Receive FIFO full interrupt request is asserted when 254 or more data entries are present in receive FIFO
1111_1110	Receive FIFO full interrupt request is asserted when 255 or more data entries are present in receive FIFO
1111_1111	Receive FIFO full interrupt request is asserted when 256 data entries are present in receive FIFO

22.16.7 DMA Controller Interface

The SPI controller uses two DMA channels, one for the transmit data and one for the receive data. The SPI controller has these DMA registers:

- DMACR. Control register to enable DMA operation.
- DMATDLR. Register to set the transmit the FIFO level at which a DMA request is made.
- DMARDLR. Register to set the receive FIFO level at which a DMA request is made.

To enable the DMA Controller interface on the SPI controller, you must write the DMA Control Register (DMACR). Writing a 1 into the TDMAE bit field of DMACR register enables the SPI controller transmit handshaking interface. Writing a 1 into the RDMAE bit field of the DMACR register enables the SPI controller receive handshaking interface.

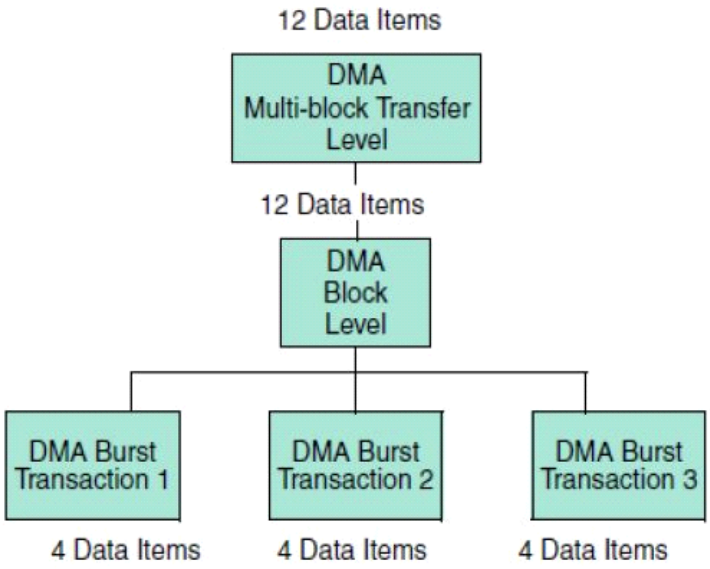
22.16.7.1 Overview of Operation

As a block flow control device, the DMA Controller is programmed by the processor with the number of data items (block size) that are to be transmitted or received by the SPI controller; this is programmed into the BLOCK_TS field of the CTLx register.

The block is broken into a number of transactions, each initiated by a request from the SPI controller. The DMA Controller must also be programmed with the number of data items (in this case, SPI controller FIFO entries) to be transferred for each DMA request. This is also known as the burst transaction length, and is programmed into the SRC_MSIZE/DEST_MSIZE fields of the DMA Controller's CTLx register for source and destination, respectively.

The following figure shows a single block transfer, where the block size programmed into the DMA Controller is 12 and the burst transaction length is set to 4. In this case, the block size is a multiple of the burst transaction length; therefore, the DMA block transfer consists of a series of burst transactions.

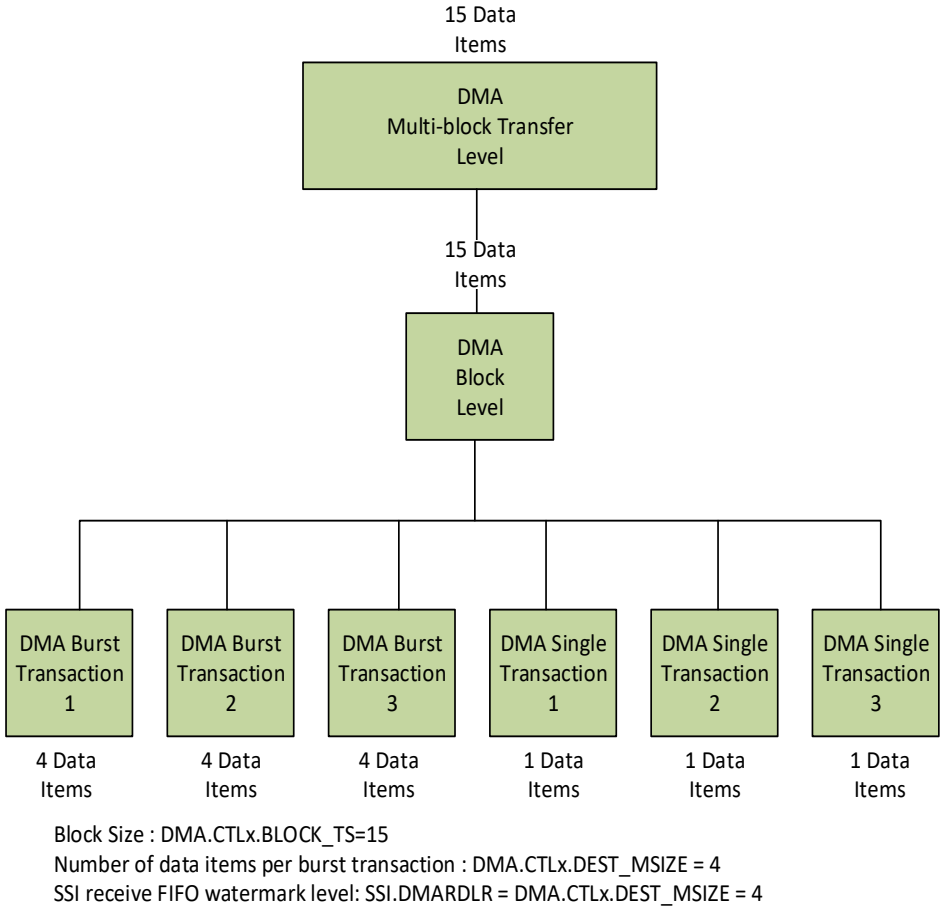
Figure 22-31.Breakdown of DMA Transfer into Burst Transactions



If the SPI controller makes a transmit request to this channel, four data items are written to the SPI controller transmit FIFO. Similarly, if the SPI controller makes a receive request to this channel, four data items are read from the SPI controller receive FIFO. Three separate requests must be made to this DMA channel before all 12 data items are written or read.

When the block size programmed into the DMA Controller is not a multiple of the burst transaction length, as shown in Figure 22-32, a series of burst transactions followed by single transactions are needed to complete the block transfer.

Figure 22-32.Breakdown of DMA Transfer into Single and Burst Transactions



22.16.7.2 Transmit Watermark Level and Transmit FIFO Underflow

During SPI controller serial transfers, transmit FIFO requests are made to the DMA Controller whenever the number of entries in the transmit FIFO is less than or equal to the DMA Transmit Data Level Register (DMATDLR) value; this is known as the watermark level. The DMA Controller responds by writing a burst of data to the transmit FIFO buffer, of length CTLx.DEST_MSIZE.

Data should be fetched from the DMA often enough for the transmit FIFO to perform serial transfers continuously; that is, when the FIFO begins to empty another DMA request should be triggered. Otherwise the FIFO will run out of data (underflow). To prevent this condition, the user must set the watermark level correctly.

22.16.7.3 Choosing the Transmit Watermark Level

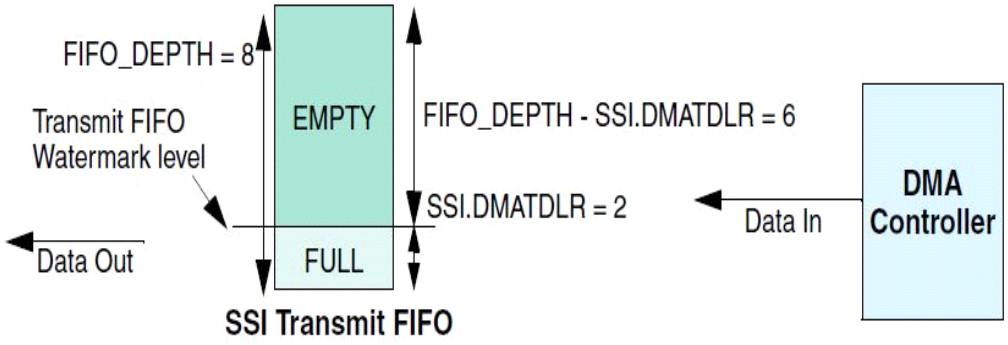
Consider the example where the assumption is made:

DMA.CTLx.DEST_MSIZE = FIFO_DEPTH - SSI.DMATDLR

Here the number of data items to be transferred in a DMA burst is equal to the empty space in the Transmit FIFO. Consider two different watermark level settings.

22.16.7.3.1 Case 1: DMATDLR = 2

Figure 22-33.Case 1 Watermark Levels



- Transmit FIFO watermark level = SSI.DMATDLR = 2
- DMA.CTLx.DEST_MSIZE = FIFO_DEPTH - SSI.DMATDLR = 6
- SSI transmit FIFO_DEPTH = 8
- DMA.CTLx.BLOCK_TS = 30

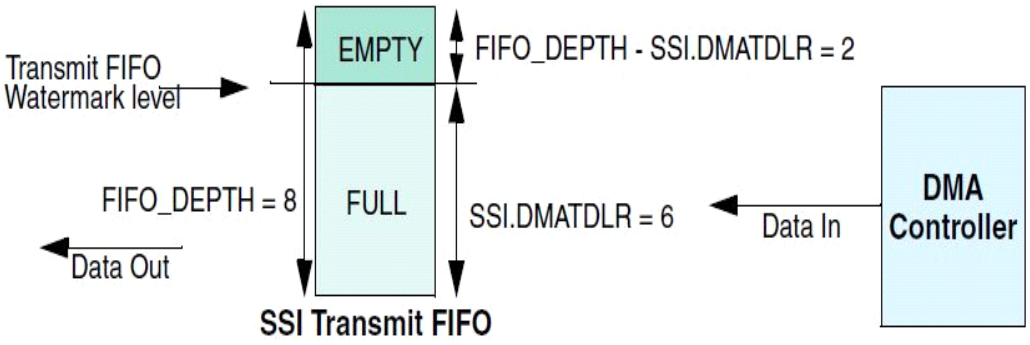
Therefore, the number of burst transactions needed equals the block size divided by the number of data items per burst:

DMA.CTLx.BLOCK_TS/DMA.CTLx.DEST_MSIZE = 30/6 = 5

The number of burst transactions in the DMA block transfer is 5. But the watermark level, SSI.DMATDLR, is quite low. Therefore, the probability of an SPI underflow is high where the SPI serial transmit line needs to transmit data, but where there is no data left in the transmit FIFO. This occurs because the DMA has not had time to service the DMA request before the transmit FIFO becomes empty.

22.16.7.3.2 DMATDLR = 6

Figure 22-34.Case 2 Watermark Levels



- Transmit FIFO watermark level = SSI.DMATDLR = 6
- DMA.CTLx.DEST_MSIZE = FIFO_DEPTH - SSI.DMATDLR = 2
- SPI transmit FIFO_DEPTH = 8
- DMA.CTLx.BLOCK_TS = 30

Therefore, the number of burst transactions needed equals the block size divided by the number of data items per burst:

DMA.CTLx.BLOCK_TS/DMA.CTLx.DEST_MSIZE = 30/2 = 15

In this block transfer, there are 15 destination burst transactions in a DMA block transfer. But the watermark level, SSI.DMATDLR, is high. Therefore, the probability of an SPI underflow is low because the DMA controller has plenty of time to service the destination burst transaction request before the SPI transmit FIFO becomes empty.

Thus, the second case has a lower probability of underflow at the expense of more burst transactions per block. This provides a potentially greater amount of bus bursts per block and worse bus utilization than the former case.

Therefore, the goal in choosing a watermark level is to minimize the number of transactions per block, while at the same time keeping the probability of an underflow condition to an acceptable level. In practice, this is a function of the ratio of the rate at which the SPI transmits data to the rate at which the DMA can respond to destination burst requests.

For example, promoting the channel to the highest priority channel in the DMA, and promoting the DMA initiator interface to the highest priority initiator in the bus layer, increases the rate at which the DMA controller can respond to burst transaction requests. This in turn allows you to decrease the watermark level, which improves bus utilization without compromising the probability of an underflow occurring.

22.16.7.4 Selecting DEST_MSIZE and Transmit FIFO Overflow

As can be seen from Figure 22-34, programming DMA.CTLx.DEST_MSIZE to a value greater than the watermark level that triggers the DMA request may cause overflow when there is not enough space in the SPI transmit FIFO to service the destination burst request. Therefore, the following equation must be adhered to in order to avoid overflow:

DMA.CTLx.DEST_MSIZE <= SSI.FIFO_DEPTH - SSI.DMATDLR (1)

In Case 2: DMATDLR = 6, the amount of space in the transmit FIFO at the time the burst request is made is equal to the destination burst length, DMA.CTLx.DEST_MSIZE. Thus, the transmit FIFO may be full, but not overflowed, at the completion of the burst transaction.

Therefore, for optimal operation, DMA.CTLx.DEST_MSIZE should be set at the FIFO level that triggers a transmit DMA request; that is:

DMA.CTLx.DEST_MSIZE = SSI.FIFO_DEPTH - SSI.DMATDLR (2)

This is the setting used in Figure 22-32.

Adhering to equation (2) reduces the number of DMA bursts needed for a block transfer, and this in turn improves bus utilization.

Note: The transmit FIFO will not be full at the end of a DMA burst transfer if the SPI has successfully transmitted one data item or more on the SPI serial transmit line during the transfer.

22.16.7.5 Receive Watermark Level and Receive FIFO Overflow

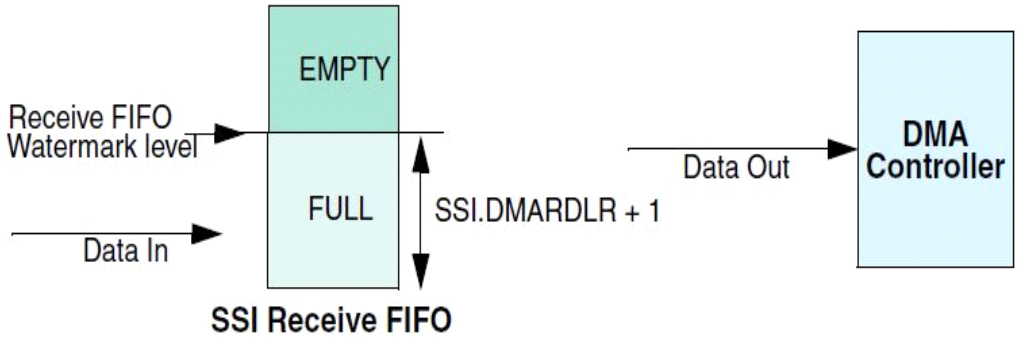
During SPI controller serial transfers, receive FIFO requests are made to the DMA Controller whenever the number of entries in the receive FIFO is at or above the DMA Receive Data Level Register; that is, DMARDLR+1. This is known as the watermark level. The DMA Controller responds by fetching a burst of data from the receive FIFO buffer of length CTLx.SRC_MSIZE.

Data should be fetched by the DMA often enough for the receive FIFO to accept serial transfers continuously; that is, when the FIFO begins to fill, another DMA transfer is requested. Otherwise, the FIFO will fill with data (overflow). To prevent this condition, the user must correctly set the watermark level.

22.16.7.6 Choosing the Receive Watermark Level

Similar to choosing the transmit watermark level described earlier, the receive watermark level, DMARDLR+1, should be set to minimize the probability of overflow, as shown in Figure 22-35. It is a trade off between the number of DMA burst transactions required per block versus the probability of an overflow occurring.

Figure 22-35.SPI Controller Receive FIFO



22.16.7.7 Selecting SRC_MSIZE and Receive FIFO Underflow

As seen in Figure 22-35, programming a source burst transaction length greater than the watermark level may cause underflow when there is not enough data to service the source burst request. Therefore, equation (3) below must be adhered to avoid underflow.

If the number of data items in the receive FIFO is equal to the source burst length at the time the burst request is made – DMA.CTLx.SRC_MSIZE – the receive FIFO may be emptied, but not underflowed, at the completion of the burst transaction. For optimal operation, DMA.CTLx.SRC_MSIZE should be set at the watermark level; that is:

DMA.CTLx.SRC_MSIZE = SSI.DMARDLR + 1 (3)

Adhering to equation (3) reduces the number of DMA bursts in a block transfer, and this in turn can improve bus utilization.

Note: The receive FIFO will not be empty at the end of the source burst transaction if the SPI has successfully received one data item or more on the SPI serial receive line during the burst.

22.16.8 SPI Interface Tuning Guidance

The following are steps to tune the SPI-based communication with end devices to have correct reading of written data, especially at higher frequencies:

1. Identify the frequency that causes a shifting behavior when reading data:
   a. Write with a known good frequency, to an address.
   b. Read the data with a known good frequency.
   c. Increase the frequency until the value starts shifting.
2. After identifying the frequency, start increasing the RX_SAMPLE_DLY value one by one until the correct data are read, or get the upper limit by sweeping until the data start to shift to the other direction. This way, you can have a range of working RX_SAMPLE_DLY values.
3. Read the data by decreasing the frequency with the new RX_SAMPLE_DLY value(s) to see if a lower frequency is affected.

22.16.9 SPI Interrupts

The SPI controller interrupts are described as follows:

- Transmit FIFO Empty Interrupt (SR.TFE & IMR.TXEIM). Set when the transmit FIFO is equal to or below its threshold value and requires service to prevent an under-run. The threshold value, set through a software-programmable register, determines the level of transmit FIFO entries at which an interrupt is generated. This interrupt is cleared by hardware when data are written into the transmit FIFO buffer, bringing it over the threshold level.
- Transmit FIFO Overflow Interrupt (IMR.TXOIM). Set when an access attempts to write into the transmit FIFO after it has been completely filled. When set, data written from the bus is discarded. This interrupt remains set until you read the transmit FIFO overflow interrupt clear register (TXOICR).
- Receive FIFO Full Interrupt (SR.RFF & IMR.RXFIM). Set when the receive FIFO is equal to or above its threshold value plus 1 and requires service to prevent an overflow. The threshold value, set through a software-programmable register, determines the level of receive FIFO entries at which an interrupt is generated. This interrupt is cleared by hardware when data are read from the receive FIFO buffer, bringing it below the threshold level.
- Receive FIFO Overflow Interrupt (IMR.RXOIM). Set when the receive logic attempts to place data into the receive FIFO after it has been completely filled. When set, newly received data are discarded. This interrupt remains set until you read the receive FIFO overflow interrupt clear register (RXOICR).
- Receive FIFO Underflow Interrupt (IMR.RXUIM). Set when an access attempts to read from the receive FIFO when it is empty. When set, zeros are read back from the receive FIFO. This interrupt remains set until you read the receive FIFO underflow interrupt clear register (RXUICR).
- Multi-Initiator Contention Interrupt (IMR.MSTIM). Present only when the SPI controller component is configured as a serial-initiator device. The interrupt is set when another serial initiator on the serial bus selects the SPI controller initiator as

a serial-target device and is actively transferring data. This informs the processor of possible contention on the serial bus. This interrupt remains set until you read the multi-initiator interrupt clear register (MSTICR).

- Combined Interrupt Request. OR'ed result of all the above interrupt requests after masking. To mask this interrupt signal, you must mask all other SPI controller interrupt requests.

22.16.10 Signal Description

Table 22-17. SPI Signal Description

Signal Name	Type	Description
PSE_SPI_CS0_N [3:0]	O	SPI Chip Select
PSE_SPI_CS1_N [3:0]	O	SPI Chip Select
PSE_SPI_CLK [3:0]	O	SPI Clock
PSE_SPI_MOSI [3:0]	O	SPI Master Out, Slave In
PSE_SPI_MISO [3:0]	I	SPI Master In, Slave Out

22.16.11 Registers

Please refer to chapter 10 of the Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors for Internet of Things (IoT) Applications, Datasheet, Volume 2 (Book 3 of 3), Intel® Programmable Services Engine (Intel® PSE) (Document Number: 636723), for a description of the registers associated with subject of this chapter.

22.17 GPIO Controller

22.17.1 Overview

The Intel® PSE GPIO are multiplexed with non Intel® PSE GPIOs and other native functions, in some cases.

- The assignment of specific pins as Intel® PSE GPIO is performed by the secure agent like BIOS, using native function (PMode bits in Pad Configuration Register DW0)
- Intel® PSE has 2 instances of GPIO controllers - each supporting 30 pins, meaning TGPIO00-TGPIO29 belong to GPIO controller 0 and TGPIO30-TGPIO59 belong to GPIO controller 1. Also, 20 of these pins per GPIO controller is muxed with Time-Aware GPIO

The GPIO can be individually configured to be:

- Input or Output using the GPIO Pin Direction Register (GPDR)
- If configured as an output: Driven High or low or tri-state using the GPIO Pin-Output Set Register / GPIO Pin-Output Clear Registers (GPSRx/GPCRx)
- Bits 29:0 of each configuration register (GPDR, GPSR, GPCR, GPLR etc) are mapped to 30 GPIO pins (TGPIO29:TGPIO00 for GPIO controller 0 and TGPIO59:TGPIO30 for GPIO controller 1)

- If configured as an input: cause an interrupt to the Arm* Cortex*-M7 as either Active high or active low (Edge or level triggered).

22.17.1.1 Ownership Allocation

- GPIO0 ownership can be controlled using OWNERSHIP_DEV22 field in Ownership Control 2 register
- GPIO1 ownership can be controlled using OWNERSHIP_DEV23 field in Ownership Control 2 register

22.17.2 Functional Description

The GPIO controller controls GPIO Capable pins on the platform. There are three common functions for the GPIO capable pins which include General Purpose IO, Wakes, or Interrupts. This section describes these three use cases.

22.17.2.1 GPIO Capable Pin as GPIO

To use a pin as GPIO then the PMode mux must be switched to the GPIO Controller mode which is the default for most pins. Once in GPIO mode the GPIO Pin Direction, GPIO Pin Output Set, and GPIO Pin Output Clear registers control the state of the pin.

The GPIO Pin Direction registers (GPDRx) are used to program the GPIO pins as inputs or outputs. For a pin configured as an output, write to the GPIO Pin Output Set register (GPSRx) to set the pin high. Write to the GPIO Pin Output Clear register (GPCx) to clear the pin to a low level. Writes to GPDRx and GPSRx can take place whether the pin is configured as an input or an output. If a pin is configured as an input the programmed output state occurs when the pin is reconfigured as an output.

To validate the state of a GPIO pin, read the GPIO Pin Level register (GPLRx). Software can read this register at any time to confirm the state of a pin, even if the pin is configured as an output.

To detect either a rising or a falling edge on each GPIO pin, use the GPIO Rising-Edge Detect Enable registers (GRERx) and GPIO Falling Edge Detect Enable registers (GFERx) to enable the respective edge detect mechanism. Note that the edge detect logic has a built in back to back flop which acts as a debounce. Please refer to the GPIO Glitch Filter section for details.

22.17.2.2 GPIO Capable Pin as Interrupt

A GPIO pin can be used as an external interrupt either the Arm* Cortex*-M7 or the IA Processor. This capability is supported by unmasking a GPIO capable pin in the GPIO Interrupt Mask Register (GIMR). Depending on standby mode there is a minimum time for which an interrupt has to be asserted which are dictated by the GPIO Glitch Filter (discussed in section Section 22.17.2.4) update accordingly.

All GPIO Interrupt Sources are mapped into one GPIO Controller interrupt going to either the Arm* Cortex*-M7 or the IA Processor.

22.17.2.3 GPIO Capable Pin as Wake

All GPIO capable pins are wake capable. It is required is to unmask the specific GPIO capable pin as a Wake or Interrupt in the GPIO Wake Mask Register (GWMR). When an edge is detected on a wake enabled pin, a bit is set in the GPIO Wake Source Register (GWSR). These wake signals can be configured to wake up the system, details of which can be found in "Power management" chapter.

22.17.2.4 GPIO Glitch Filter

The GPIO Glitch Filter is used to ensure a signal with sufficient width enters the edge detection logic and hence used for wake or interrupts. The Glitch Filter drops pulses that are not at least 3 clock periods wide and therefore filters out small glitches.

When a GPIO capable pin is in GPI mode the input signals enters the glitch filter by default before reaching the edge detection registers. The glitch filter will filter out any signal pulses that are smaller than 30ns when the GPIO runs on a 100 MHz clock (clock period 10 ns). Any pulse shorter than 60 ns will not reach the GPIO edge detection logic. Any pulse longer than 60 ns will be detected by the GPIO edge detection logic.

There is a bypass mode to bypass the Glitch filter and this is controlled through the GPIO Glitch Filter Bypass register (GGFR). The GPIO Glitch Filter Bypass registers could configure the bypass of this glitch filter logic such that the pin inputs directly trigger the edge-detection register.

22.17.3 Registers

Please refer to chapter 11 of Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors for Internet of Things (IoT) Applications, Datasheet, Volume 2 (Book 3 of 3), Intel® Programmable Services Engine (Intel® PSE) (Document Number: 636723), for a description of the registers associated with subject of this chapter.

22.18 Time-Aware GPIO

22.18.1 Overview

The Time-Aware GPIO module in Intel® PSE provides the following functions:

- Pulse and Waveform generation on Output pins.
- Pulse width modulation.
- Recording the time at which specific events occur.
- Event Counters to record input events.
- Three Rate and Offset Compensated Timers per Time-Aware GPIO controller block. These timers are referred to as Tunable Monotonous Timers or TMTs.
- Time stamping events in synchronization with the global time base distributed from the ART as part of the Time Synchronization Protocol.
- Time stamping events in synchronization with global working or system time bases distributed through Precision Time Protocol (PTP).
  - — Note only one TMT can be used to cross-timestamp against various IP timers.

- Time slice capability by using a TMT to driver a selectable number of GPIO controllers.
- Interrupt capability when specific events occur.
  - Interrupts are coalesced.
- Note: There are two instances of the Time-Aware GPIO controllers in Intel® PSE. Each instance has 20 Time-Aware GPIO signals.

22.18.2 Functional Description

There are 2 TGPIO blocks. Any instance of the Time-Aware GPIO block can be configured to select from one of four Timer sources to Input Time Stamp Counter Input. The four sources are the Local ART value and the three Tunable Monotonous Timers (TMT_0, TMT_1 & TMT_2). Selection is made by software using the Time-Aware GPIOx Control register TGPIOCTLx.Timer Select. "x" in here varies from 0 to 19.

There are two clock sources, namely the 19.2MHz clock or the 200 MHz TMT clock. Depending on the setting of the TGPIOCTLx.Timer_Mux field the hardware will select the correct clock source. When the Local ART value is selected the 19.2MHz crystal clock must be used. For all other TMT timers the hardware selects the 200 MHz TMT clock.

22.18.3 Tunable Monotonous Timer (TMT)

The Time-Aware GPIO block implements several TMT counters. Each of these counters are tunable for rate and offset. Rate tuning allows the counter to be slowed or speeded up by changing the increment or decrement value. Offset correction allows the counter to be adjusted in 1ns increments.

The TMT is a 96-bit register is composed of: TMTR, TMTL and TMTH registers: The TMTR register holds the sub ns fraction, the TMTL register holds the ns fraction and the TMTH register holds the second fraction of the time (note that the upper two bits of the TMTL register are always zero and the max value of this register is 999,999,999 dec). When synchronized the TMT registers defines the absolute time relative to PTP "epoch" which is January 1st 1970 00:00:00 International Atomic Time (TAI).

- Initial Setting - Setting the initial time is done by direct write access to the TMTL and TMTH registers.

  Software should first set the TMTL register and then set the TMTH register. Setting the TMTR register is meaningless while it represents sub ns units. It is recommended to disable the timer at programming time using TGPIOCTLx.EN.
- Run Time - During run time the SYSTIM timer value in the TMTR, TMTL and TMTR registers, is updated periodically each clock cycle according to the following formula:
  - Define: Timer Increment = 5ns (TMT Clock Period) +/- TIMINCA.Incvalue (Tunable Monotonous Timer Increment Attributes.Increment Value) * (2 to 32) nsec. Add or subtract the TIMINCA.Incvalue is defined by TIMINCA.ISGN (Tunable Monotonous Timer Increment Attributes.Increment Sign) (while 0b means Add and 1b means Subtract)
  - Then: TMT = TMT + INC_TIME
- Reading the TMT register by software is done by the following sequence:
  - Read the TMTR register

  - — Read the TMTL register
  - — Read the TMTH register

Dynamic update of TMT registers can be done by using the TIMADJ registers by the following flow.

  - — Write the Tadjust value and its Sign to the TIMADJ register (the Sign bit indicates if the Tadjust value should be added or subtracted)
  - — Following the write access to the TIMADJ register, the hardware repeats the following two steps at each TMT Clock Period as long as the Tadjust > zero.

- TMT = TMT+ INC_TIME +/- 1 nsec. Add or subtract 1 nsec is defined by TIMADJ.Sign (while 0bmeans Add and 1b means Subtract) Tadjust = Tadjust - 1 nsec

Note: The TMT timer is incremented monotonically at all times. When updating the TMT by the TIMADJ and concurrent non-zero TIMINCA, the TMT is incremented each clock by steps in the range of TMT Clock Period - 1.5ns up to TMT Clock Period + 1.5ns. For a 200MHz clock the range is 3.5ns up to 6.5ns units.

  - — As shown above, the time adjustment might take multiple clocks. Software might write a new value to the TIMADJ register before the hardware completed the previous adjustment. In such a case, the new value written by software, overrides the above equation. If such a race is not desired, the software could check that the previous adjustment is completed by one of the following methods:

- Wait enough time before accessing the TIMADJ register which guarantees that the previous update procedure is completed.
- Poll the matched TGPIORIS.TADJ_TMT_GLOBAL_CMPLT/ TGPIORIS.TADJ_TMT_WORKING_CMPLT flag which is set by the hardware each time the update procedure is completed.
- Enable the TADJ interrupt by setting the TADJ flag in the Time-Aware GPIO Raw Interrupt Status (TGPIORIS/TGPIOMSC/TGPIOMIS) registers. The TADJ interrupt indicates that the hardware completed the adjustment procedure. This method is unlikely to be used in nominal operation since the expected adjustments are in the sub s range.

1. Each TMT operates at 200MHz.
2. Any TMT can be selected to be used by any Time-Aware GPIO controller as a counter source.
3. Each TMT can be cross-timestamped against:
   a. ART

When the clock source is the 200MHz TMT clock, the Timer input is one of the TMT registers. For this case the COMPV and PIV registers must be programmed to match the Seconds and Nano-seconds fields of the TMT. The lower 30 bits of the COMPV/PIV represent the nanosecond field (max value is 0x3B9A C9FF) and the upper 32 bits (63:32) represent the Seconds field. Bits 31:30 are zero and don’t care. The PIV bits [63:29] must be zero when timer source is TMT. When lower 30 bits overflows (exceeds 0x3B9A C9FF) indicating that at least 1 second has been counted then the upper bits of COMP[63:32] are incremented.

22.18.4 Cross-Timestamp Support

This section describes how cross-timestamps are captured between the various counters in the Time-Aware GPIO block and the Local ART counter Intel® PSE. Cross-timestamps allows software to establish a known relationship between Tunable Monotonous Timer and Local ART time base.

When a cross-timestamp is captured in relation to the local ART value, it is referred to as the Local Cross-Timestamp.

One set of cross-timestamps are captured in the Time-Aware GPIO block for TMT_0, TMT_1, and TMT_2.

1. Cross-Timestamp between local ART and TMT_0, TMT_1 and TMT_2

Software enables the snapshot by setting TSC_CTL[0/2/4] for TMT0_vs_ART, TMT1_vs_ART and TMT2_vs_ART respectively.

Software must poll for TSC_STATUS[0/2/4] to be correspondingly set indicating snapshot is valid. There is no interrupt on snapshot.

22.18.5 Interrupts

Each Time-Aware GPIO instance can generate an interrupt. The interrupt is controlled by the following registers:

- Time-Aware GPIO Raw Interrupt Status Register
- Time-Aware GPIO Interrupt Mask Control register
- Time-Aware GPIO Masked Interrupt Status register
- Time-Aware GPIO Interrupt Clear register

Each of the outputs of the Time-Aware GPIO Masked Interrupt Status registers can be connected to the IOAPIC however to reduce the number of interrupts the Time-Aware GPIO block coalesces interrupts to a single interrupt.

22.18.6 Usage Model

This section briefly describes some example usage models for the Time-Aware GPIO.

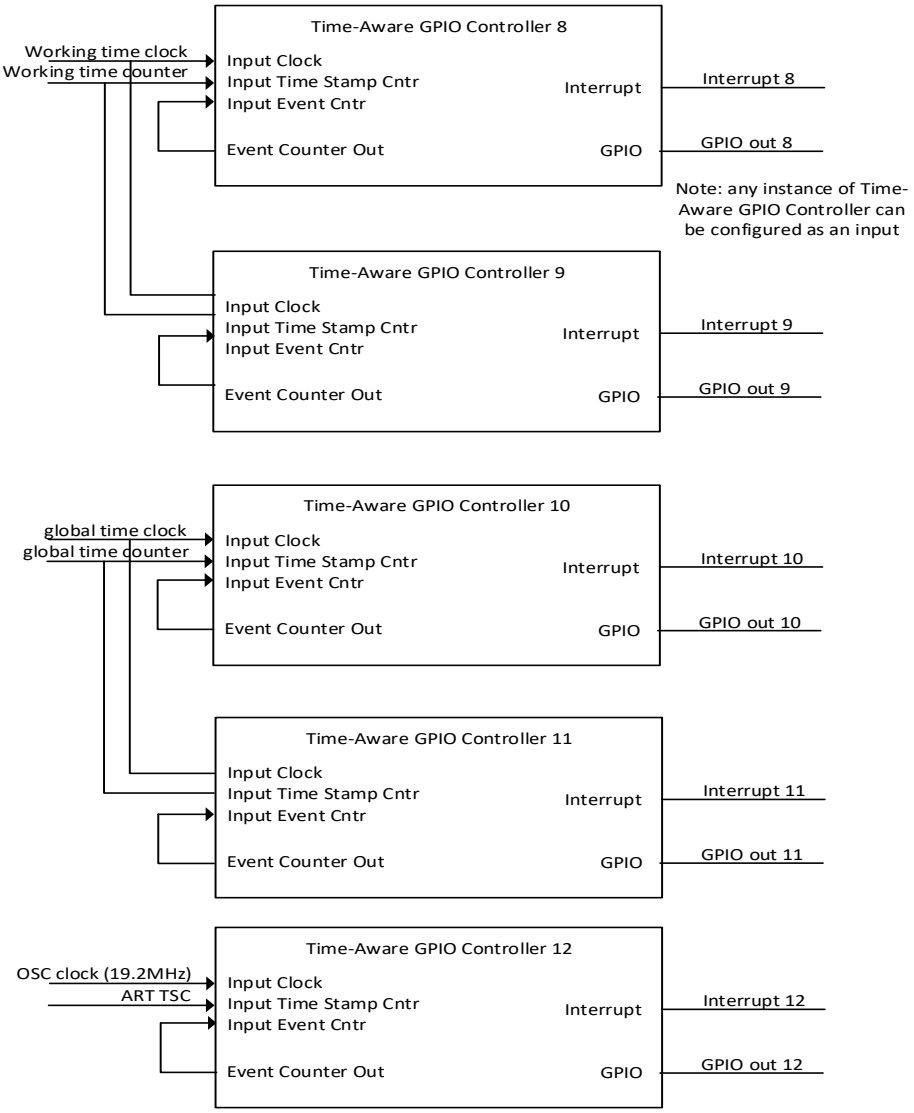
22.18.6.1 Sync Out

Multiple GPIO pins synced to the Always Running Time (ART) can be used to trigger a pulse over a defined time period ranging from 1us to 1s. These timed pulses can be used to synchronize legacy (not supporting PTM or PTP) peripherals.

Low Jitter is a requirement for the Sync Out pins. Jitter is a deviation or displacement in the pulse in terms of pulse width and phase timing (do the rise and fall times move around). The jitter target is 10ns.

Different Time sources besides the ART can be used to drive the Sync Out pins. This Time-Aware GPIO module provides a TMT (Tunable Monotonous Timer) which can be rate and offset tuned by software to match a time base that is common to a network.

Figure 22-36.Sync out configuration

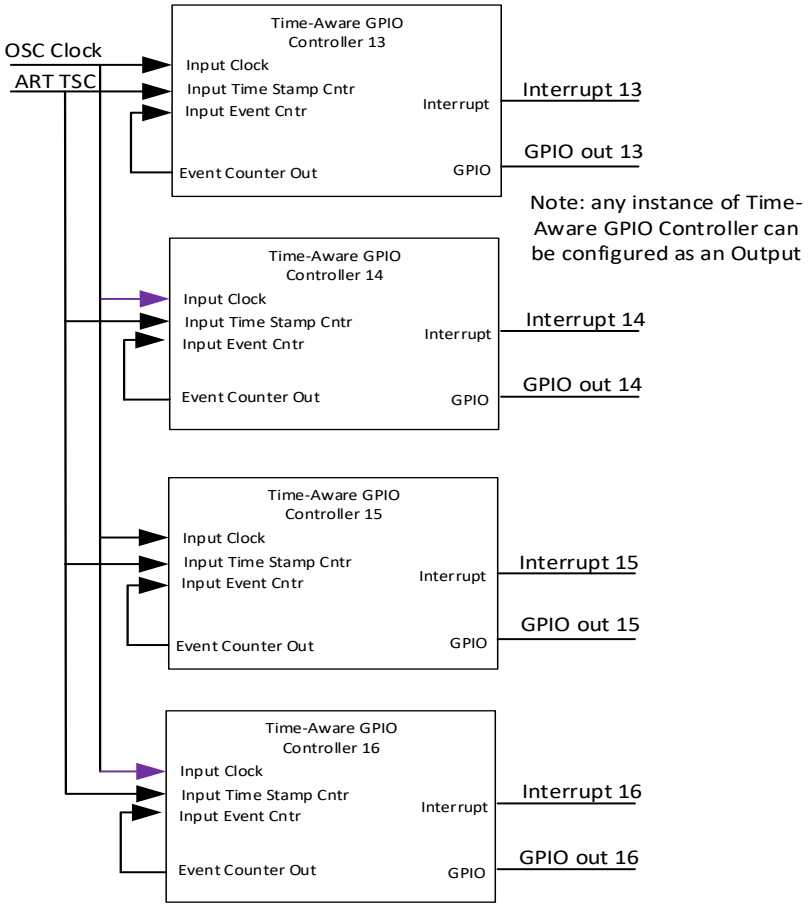


Working time counter/Global time counter refers to application usage based names for TMT counters.

22.18.6.2 Sync In

Time-Aware GPIO pins that are configured as input pins are used to record events that are triggered by external hardware. The Time-Aware GPIO pins can be configured to count pulses and trigger an interrupt when a certain number of events have occurred or trigger an interrupt as soon as there is a transition on the input. Timestamps can be attached to these events to allow software to synchronize when these events occurred.

Figure 22-37.Sync In Configuration

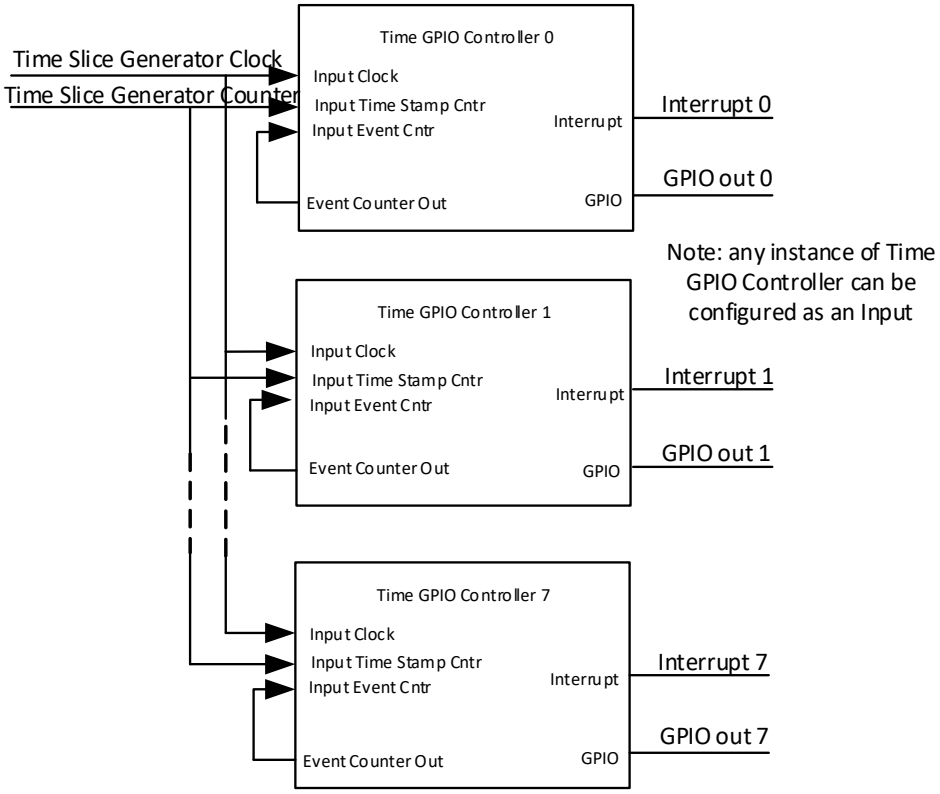


22.18.6.3 Time Slice Generator

The Time Slice Generator is used to generate a sequence of repeating pulses from up to 8 Time-Aware GPIO pins driven from a common TMT. The train of pulses from the ganged GPIO pins are typically offset from each other. The pulses can divide up an isochronous cycle in to multiple sub cycles which can be used to control and or synchronize external hardware. Each TSG pin should be capable of generating an interrupt.

The jitter requirement is 10ns or better for the TSG. Note that the jitter between the TSG pins should also be low.

Figure 22-38.Time Slice Generator controller



Time Slice Generator Counter refers to application usage based names for TMT counters.

22.18.7 TGPIO/GPIO Configuration

Intel® PSE has 2 instances of GPIO and 2 instances of Time-Aware GPIO modules. These are clubbed together into 2 clusters

- Cluster 0 -> GPIO0 + Time-Aware GPIO0
- Cluster 1 -> GPIO1 + Time-Aware GPIO1
- Below shows how the GPIO and TGPIO are muxed
  - — 20-pins from each Time-Aware GPIO and 30-pins from each GPIO are muxed together. The mux selection register, TGPIO_MUX_SEL[1:0] is implemented in IPC.

Table 22-18. GPIO an TGPIO Muxed

TGPIO_MUX_SEL[1:0]	MUXED_GPIO_OUT[29:0]
00	{GPIO[29:0]}
01	{GPIO[9:0], TGPIO[29:10]}
10	{TGPIO[19:0], GPIO[29:20]}
11	{TGPIO[9:0], GPIO[19:10], TGPIO[29:20]}

— Therefore, from each cluster 30-pins are coming out, package balls TGPIO0-29 are mapped to one controller (cluster 0) and TGPIO30-59 are mapped to other controller (cluster 1).

22.18.8 TGPIO/GPIO Signal Description

Table 22-19. TGPIO/GPIO Signal Description

Signal Name	Type	Description
PSE_TGPIO[59:00]	I/O	Input/Output

22.18.9 Registers

Please refer to chapter 11 of the Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors for Internet of Things (IoT) Applications, Datasheet, Volume 2 (Book 3 of 3), Intel® Programmable Services Engine (Intel® PSE) (Document Number: 636723), for a description of the registers associated with subject of this chapter.

22.19 I^2S Controller

22.19.1 Overview

- The I^2S controllers are only supported for usage by the Arm* Cortex*-M7 in the Intel® PSE.
- The I^2S controllers provides a three wire serial audio interface Receiver and transmitter compliant with Phillips I^2S specification
- Support Left and Right justified audio modes
- Support max resolution of 32 bit and min resolution of 12bit
- Supports 8-channel TDM per interface. Each interface supports full duplex Tx and Rx capability
- Supports interface to DMA controller to allow transfer of audio samples directly to memory from I^2S FIFO, which is 128Bytes for TX and 128Bytes for RX
- Supports audio sampling frequencies up to 192KHz. Minimum sampling rate of 8KHz with frequency granularity if <=0.025Hz
- Supports audio sampling rates of 11.025KHz, 12KHz, 16KHz, 18.9KHz, 22.05KHz, 24KHz, 32KHz, 37.8KHz, 44.1KHz and 48KHz
- Supports maximum of 8-channels 16bit audio @ sample rate of 48KHz

- Multi-channel support with 8-channels, 16-bit audio with sample rate of 48KHz multiplexed over single interface in TDM mode
- Support adjusting sampling frequency while controller is in use
- Support independent enable/disable of Rx and Tx paths without interrupting operation of the other path
- Support edge triggered frame sync in target mode

22.19.1.1 Ownership Allocation

- I^2S0 ownership can be controlled using OWNERSHIP_DEV31 field in Ownership Control 3 register
- I^2S1 ownership can be controlled using OWNERSHIP_DEV32 field in Ownership Control 4 register

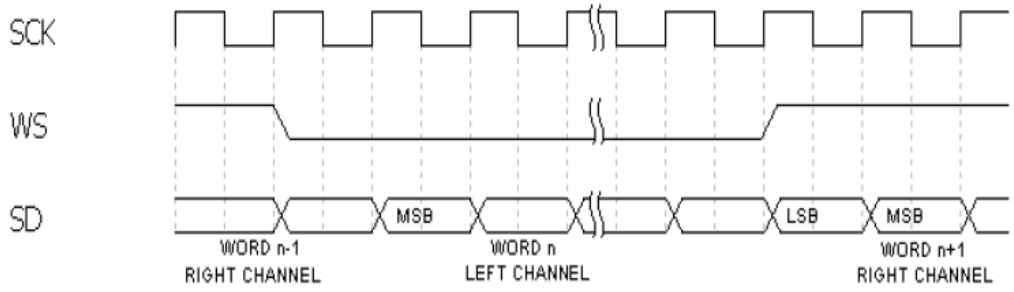
22.19.2 Functional Description

The I^2S bus uses four wires to transfer information between devices connected to the bus:

- SCK - PSE_I2S0_SCLK
- WS - PSE_I2S0_SFRM
- SDI - PSE_I2S0_RXD
- SDO - PSE_I2S0_TXD

Audio data and control signals are transferred separately. Audio data coming from two channels is time-multiplexed to the data signal. According to the Philips I^2S bus specification revised June 1996, left channel data is transmitted when WS = 0, while right channel data is transmitted when WS = 1. Serial data is transmitted in two's complement with the MSB first. The transmitter always sends the MSB of the next word one clock period after WS changes. The basic I^2S Philips interface timing is illustrated on the following figure.

Figure 22-39.Example of basic transmission for I^2S bus



The controller includes a Register block. The block is used to change the operation mode, configure data transmission data rate, resolution, etc. The information about current state of the I^2S controller and the data FIFOs associated with the I^2S controller.

The controller can operate as a transmitter, as a receiver or as a full-duplex mode transceiver. The mode of operation can be set using DIR_CFG bit in the I^2S_CTRL register.

Channel width (number of bits in audio channel) can take the following values: 8, 12, 16, 18, 20, 24, 28 or 32. Samples resolution can vary between 2 up to 32. In full-duplex mode transmission the sample resolution is set separately for transmitting and receiving.

The SFR (Special Functions Registers) block comprises set of registers that provide status monitoring and control over the I^2S I/O channel and FIFOs. It also includes interrupt generation unit as well as DMA data transfer support unit.

22.19.3 I^2S Bus Interface

The controller provides set of audio interface configuration options that are programmed through SFR control registers. These options provide wide configuration possibilities of the audio interface. The controller's most common operation modes are presented below and suitable register subset values for described audio interface modes are listed in the following table.

Table 22-20. Audio interface models

Audio interface	Register/Bit	Value	Comment
Standard I^2S mode	I2S_CTRL/sck_polar	0	Interface meets the I^2S digital audio interface specification defined by Philips® I^2S bus specification, revised June 1996
	I2S_CTRL/ws_polar	0	
	I2S_CTRL/ws_mode	1	
	I2S_CTRL/data_ws_del	1	
	I2S_CTRL/audio_mode	0	
	I2S_CTRL/mono_mode	0	
	I2S_CTRL/data_align	0	
	I2S_CTRL/data_order	0	
Left-justified I^2S mode	I2S_CTRL/sck_polar	0	The MSB of the digital audio sample is available on the rising edge of the bit clock signal directly following the rising edge of the word select signal.
	I2S_CTRL/ws_polar	1	
	I2S_CTRL/ws_mode	1	
	I2S_CTRL/data_ws_del	0	
	I2S_CTRL/audio_mode	0	
	I2S_CTRL/mono_mode	0	
	I2S_CTRL/data_align	0	
	I2S_CTRL/data_order	0	

Table 22-20. Audio interface models

Audio interface	Register/Bit	Value	Comment
Right-justified I^2S mode	I2S_CTRL/sck_polar	0	The LSB of the digital audio sample is available on the rising edge of the bit clock signal preceding the falling edge of the word select signal. If sample width is smaller than the channel time slot the preceding serial data bits are filled with zeros.
	I2S_CTRL/ws_polar	1	
	I2S_CTRL/ws_mode	1	
	I2S_CTRL/data_ws_del	0	
	I2S_CTRL/audio_mode	0	
	I2S_CTRL/mono_mode	0	
	I2S_CTRL/data_align	1	
	I2S_CTRL/data_order	0	

Regardless of the audio interface configuration the controller may operate in one of the following transmission modes:

- Initiator Transmitter Mode - Serial data is transmitted through PSE_I2Sx_TXD output while PSE_I2Sx_SCLK and PSE_I2Sx_SFRM output are driven by internally generated serial clock and word select, respectively. The controller is responsible for generating the serial clock signal and the word select signal. Serial clock and word select signals are synchronized together.
- Initiator Receiver Mode - Serial data is received from PSE_I2Sx_RXD input while PSE_I2Sx_SCLK and PSE_I2Sx_SFRM output are driven by internally generated serial clock and word select, respectively. The controller is responsible for generating the serial clock signal and the word select signal. Serial clock and word select signals are synchronized together.
- Target Transmitter Mode - Serial data is transmitted through PSE_I2Sx_TXD output while serial clock and word select are received through PSE_I2Sx_SCLK and PSE_I2Sx_SFRM, respectively. The serial clock and the word select signals must be provided by the external initiator device on the I^2S bus. To fulfill all the timing requirements mentioned in the Philips specification revised June 1996, the I^2S bus control signals should be synchronized together.
- Target Receiver Mode - Serial data, serial clock and word select are received through PSE_I2Sx_RXD, PSE_I2Sx_SCLK and PSE_I2Sx_SFRM, respectively. The serial clock and the word select signals must be provided by the external initiator device on the I^2S bus. To fulfill all the timing requirements mentioned in the Philips specification revised June 1996, the I^2S bus control signals should be synchronized together.
- Initiator Full-duplex Mode - Serial data is transmitted through PSE_I2Sx_TXD output and simultaneously received from PSE_I2Sx_RXD input while PSE_I2Sx_SCLK and PSE_I2Sx_SFRM outputs are driven by internally generated serial clock and word select, respectively. The controller is responsible for generating the serial clock signal and the word select signal. Serial clock and word select signals are synchronized together.
- Target Full-duplex Mode - Serial data is transmitted through PSE_I2Sx_TXD output and simultaneously received from PSE_I2Sx_RXD input while PSE_I2Sx_SCLK and PSE_I2Sx_SFRM outputs are driven by internally generated serial clock and word select, respectively. The serial clock and the word select signals are sampled from the PSE_I2Sx_SCLK and PSE_I2Sx_SFRM inputs and must be provided by the external initiator device on the I^2S bus.

22.19.4 Time Division Multiplexing Audio Interface

Time Division Multiplexed (TDM) audio interface provides possibility of transferring multiple digital audio channels within one physical interface common for all channels.

In TDM mode, a sequence of audio samples (one for each channel) creates one audio frame. All frame samples are transferred via serial data line sequentially in appropriate time slots. The I^2S word select signal provides frame synchronization functionality. PSE_I2Sx_SFRM rising edge indicates the beginning of the first channel time slot in consecutive audio frames. Concerning the particular solution the PSE_I2Sx_SFRM signal may back to the default level after one I^2S serial clock cycle or after certain number of channels time slots, but the WS signal has to be at the default level during the last channel time slot.

The controller operating in TDM interface mode can support up to 8 audio channels. There is a dedicated SFR register to control the additional transmission properties for TDM mode and one more to control TDM channels in full-duplex mode. PSE_I2Sx_SCLK polarity, PSE_I2Sx_SFRM polarity and time slot with reference to PSE_I2Sx_SFRM delay can be changed through the I^2S_CTRL register as for other I^2S transmission modes.

The TDM transmission mode is activated with tdm_en bit from TDM_CTRL register. The TDM mode control register additionally provides the chn_no field which configures the number of supported channels in one audio frame. Each channel from 0…chn_no-1 range can be disabled. When audio channel for its corresponding time slot is disabled, the serial data line is assigned to low logic level. Note that audio samples for only active channels are read from/written to the FIFO memory in transmitter/receiver mode respectively. In full-duplex mode each active channel have another two bits in the TDM_FD_DIR register to enable transmit and/or receive at the channel time slot. In half-duplex mode channels transmit/receive direction setting is specified by dir_cfg bit at I^2S_CTRL register.

While the I^2S word select (PSE_I2Sx_SFRM) signal is used for frame synchronization the ws_mode configuration filed provided through I^2S_CTRL register defines PSE_I2Sx_SFRM signal format.

22.19.5 Sample Rate

In the initiator mode I2S_SRATE register determine the PSE_I2Sx_SCLK and PSE_I2Sx_SFRM signals frequencies for the transmitter and the receiver mode respectively. Sample set of this register values corresponding to the most commonly used and standardized audio sample frequencies are collected in the following table. Values in this table are calculated for clk clock frequency fclk = 100 MHz.

Table 22-21. Example of audio settings and sample rate

Audio Settings	I^2S Rate	I^2S_SRATE register sample_rate
Single (mono) channel, 8 kHz, 8-bit samples	64 kbit/s	1563 (61B)
Single (mono) channel, 16 kHz, 16-bit samples	256 kbit/s	391 (187)
8 (stereo) channels, 48 kHz, 16-bit samples	12.288 Mbit/s	9 (9)

22.19.6 Control Settings

Special Functions Registers (SFRs) can be reset (set to the default state) in two ways. They are always reset when the Arm* Cortex*-M7 core resets. The other way is to set sfr_rst bit to '0' in the I2S_CTRL register.

Note: This bit has to be explicitly cleared before SFRs are used.

There are two bits in the I2S_CTRL register and one in I2S_CTRL_FDR that, when set to the '0' state, reset the I^2S transceiver or FIFO (i2s_en, fifo_rst and rfifo_rst bits). Since these reset signals have to be re-synchronized to the internal controller clock, it takes 2-3 clk clock cycles before the actual reset takes place. This has to be taken into account, particularly in the case of communication with the FIFO.

Other bits in the I2S_CTRL register as well as in the I2S_SRATE, I2S_SRES, FIFO_AEMPTY, FIFO_AFULL, TDM_CTRL (if implemented) registers act as configuration fields. It is important that configuration fields influencing the given module do not change when corresponding reset bit is '1'. In other words, in order to provide safe operation of the core configuration fields should be only changed when the corresponding reset bit is '0'.

The I2S_CTRL[24] register (i2s_stb) controls gating the I^2S transceiver clock. When the i2s_stb = 0 the I^2S transceiver clock is enabled. When the i2s_stb = 1 the I2S transceiver clock is disabled. The I^2S transceiver is also disabled when the i2s_stb = 1 (then the i2s_en_r register is cleared). The I2S_CTRL[24] register is connected to the strobe output. An external synchronizer is required to proper gating the I^2S transceiver clock.

22.19.7 Status and Interrupts

Some bits in the I2S_STAT register (Table 22-22) serve as events indicators. When '1' they cause generation of an interrupt request (see also masking below). I2S_STAT register bits that indicate transmit channel(s) or receive channel(s) events are tdata_underr, rdata_ovrerr, respectively. When read and '1' they indicate occurrence of the underrun condition in the transmit channel(s) or overrun condition in the receive channel(s), respectively.

Table 22-22. I^2S Status Register

Bit	Number	Description
tdata_underr	0	Indicates data underrun event, active HIGH. This flag is set to HIGH when TX underrun arises (rising edge of 'underrun' condition). Updated on the rising edge of the clock. This bit can trigger the interrupt.
rdata_ovrerr	1	Indicates data overrun error for transceiver in receiver mode, active HIGH. Sampled and updated on the rising edge of the clock. Writing LOW value to this bit resets the transceiver and the FIFO. The transceiver configuration is preserved. This bit can trigger the interrupt.

Table 22-22. I²S Status Register

Bit	Number	Description
fifo_empty	2	FIFO empty flag. Active HIGH. This flag is set to HIGH when FIFO is empty. When full-duplex mode is active, this bit is a flag only for transmitter FIFO empty condition. This bit can trigger the interrupt.
fifo_aempty	3	FIFO almost empty flag. Active HIGH. This flag is set to HIGH when FIFO becomes almost empty (rising edge of 'almost empty' condition). When full-duplex mode is active, this bit is a flag only for transmitter FIFO almost empty condition. This bit can trigger the interrupt.
fifo_full	4	FIFO full flag. Active HIGH. This flag is set to HIGH when FIFO is full. When full-duplex mode is active, this bit is a flag only for transmitter FIFO full condition. This bit can trigger the interrupt.
fifo_afull	5	FIFO almost full flag. Active HIGH. This flag is set to HIGH when FIFO becomes almost full (rising edge of 'almost full' condition). When full-duplex mode is active, this bit is a flag only for transmitter FIFO almost full condition. This bit can trigger the interrupt.
-	6-15	Not implemented. Return '0' when read.
rfifo_empty	16	Receive FIFO empty flag in full-duplex mode. Active HIGH. This flag is set to HIGH when RX FIFO is empty. This bit can trigger the interrupt.
rfifo_aempty	17	Receive FIFO almost empty flag in full-duplex mode. Active HIGH. This flag is set to HIGH when RX FIFO becomes almost empty (rising edge of 'almost empty' condition). This bit can trigger the interrupt.
rfifo_full	18	Receive FIFO full flag in full-duplex mode. Active HIGH. This flag is set to HIGH when RX FIFO is full. This bit can trigger the interrupt.
rfifo_afull	19	Receive FIFO almost full flag in full-duplex mode. Active HIGH. This flag is set to HIGH when RX FIFO becomes almost full (rising edge of 'almost full' condition). This bit can trigger the interrupt.
-	others	Not implemented. Return '0' when read.

22.19.8 Signal Description

Table 22-23. I^2S Signal Description

Signal Name	Type	Description
PSE_I2S0_TXD PSE_I2S1_TXD	O	I^2S Transmit
PSE_I2S0_RXD PSE_I2S1_RXD	I	I^2S Receive
PSE_I2S0_SFRM PSE_I2S1_SFRM	I/O	I^2S Frame Clock
PSE_I2S0_SCLK PSE_I2S1_SCLK	I/O	I^2S Clock

22.19.9 Clocking

I^2S bus clock is derived using 100MHz clock fed to M/N divider. Need to program I2S0_MNDIV_M_VALUE, I2S0_MNDIV_N_VALUE, and I2S0_MNDIV_ENABLE.

Note: If I2S0_MNDIV_ENABLE bit is set then I2S_SRATE divider is disabled. I2S_SRATE divider can only be used to generate limited sampling rates.

22.19.10 Registers

Please refer to chapter 13 of the Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors for Internet of Things (IoT) Applications, Datasheet, Volume 2 (Book 3 of 3), Intel® Programmable Services Engine (Intel® PSE) (Document Number: 636723), for a description of the registers associated with subject of this chapter.

22.20 Pulse Width Modulation (PWM)

PWM has the following features:

- 2 instances of PWM are part of a single PCI function to Host. MSI multi-message is implemented with 2 vectors to support independent interrupts from each PWM instance towards Host.
- Up to eight programmable timers each
- 32 bits timer width
- Support for two operation modes: free-running and user-defined count
- Support for independent clocking of timers

22.20.0.1 Ownership Allocation

- PWM ownership can be controlled using OWNERSHIP_DEV33 field in Ownership Control 4 register

22.20.1 Functional Description

22.20.1.1 Timer Operation

Each PWM block implements eight identical but separately-programmable timers. Timers count down from a programmed value and generate an interrupt when the count reaches zero.

The initial value for each timer - that is, the value from which it counts down - is loaded into the timer using the appropriate load count register (TimerNLoadCount). Two events can cause a timer to load the initial count from its TimerNLoadCount register:

- Timer is enabled after being reset or disabled
- Timer counts down to 0

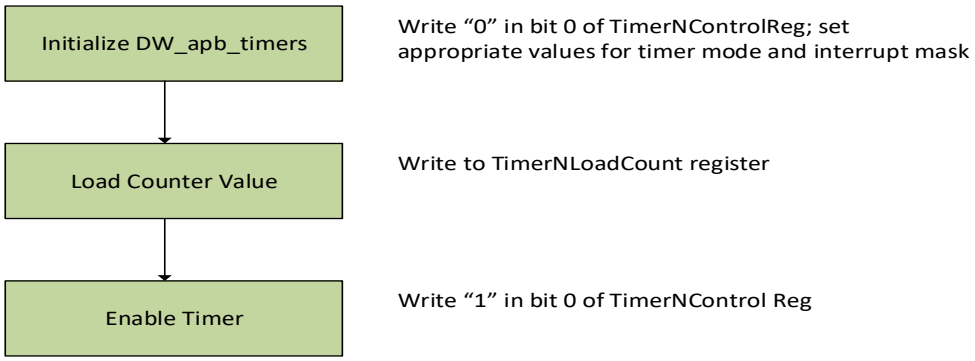
All interrupt status registers and end-of-interrupt registers can be accessed at any time.

22.20.1.2 Timers Usage Flow

The procedure illustrated in the following figure, is a basic flow to follow when programming the PWM controller. More advanced functions are discussed later in this chapter.

- Initialize the timer through the TimerNControlReg register (where N is in the range 1 to 8):
  - Disable the timer by writing a “0” to the timer enable bit (bit 0); accordingly, the timer_en output signal is de-asserted
  - Program the timer mode-user-defined or free-running-by writing a “0” or “1,” respectively, to the timer mode bit (bit 1).
  - Set the interrupt mask as either masked or not masked by writing a “1” or “0,” respectively, to the timer interrupt mask bit (bit 2).
- Load the timer counter value into the TimerNLoadCount register (where N is in the range 1 to 8).
- Enable the timer by writing a “1” to bit 0 of TimerNControlReg.

Figure 22-40.Timers Usage Flow Diagram



22.20.1.3 Enabling and Disabling a Timer

22.20.1.3.1 Enabling a Timer

If you want to enable a timer, you write a “1” to bit 0 of its TimerNControlReg register.

22.20.1.3.2 Disabling a Timer

To disable a timer, write a “0” to bit 0 of its TimerNControlReg register.

When a timer is enabled and running, its counter decrements on each rising edge of its clock signal, timer_N_clk. When a timer transitions from disabled to enabled, the current value of its TimerNLoadCount register is loaded into the timer counter on the next rising edge of timer_N_clk.

When the timer enable bit is de-asserted and the timer stops running, the timer counter and any associated registers in the timer clock domain, such as the toggle register, are asynchronously reset.

When the timer enable bit is asserted, then a rising edge on the timer_en signal is used to load the initial value into the timer counter. A “0” is always read back when the timer is not enabled; otherwise, the current value of the timer (TimerNCurrentValue register) is read back.

22.20.1.4 Loading a Timer Countdown Value

When a timer counter is enabled after being reset or disabled, the count value is loaded from the TimerNLoadCount register; this occurs in both free-running and user-defined count modes.

When a timer counts down to 0, it loads one of two values, depending on the timer operating mode:

- User-defined count mode - Timer loads the current value of the TimerNLoadCount register. Use this mode if you want a fixed, timed interrupt. Designate this mode by writing a “1” to bit 1 of TimerNControlReg.
- The value that is loaded to the timer - when it counts down to 0 - alternates between the value of the TimerNLoadCount register and the TimerNLoadCount2 register.
- Free-running mode - Timer loads the maximum value, which is 2^(32 - 1) bits, all of which are loaded with 1s. The timer counter wrapping to its maximum value allows time to reprogram or disable the timer before another interrupt occurs. Use this mode if you want a single timed interrupt. Designate this mode by writing a "0" to bit 1 of TimerNControlReg.

22.20.1.5 Generating Toggled Outputs

You can configure a timer to generate an output that toggles whenever the timer counter reaches 0. If the register bit TimerNControlReg[4] (TIMER_PWM bit) is set to 1, the HIGH and LOW periods of the toggle outputs can be controlled separately by programming the TimerNLoadCount2 and TimerNLoadCount registers.

The toggle output from each of the timers can be pulse-width modulated. The pulse widths of the toggle outputs are controlled as follows:

- HIGH period = (TimerNLoadCount2 + 1) * timer_N_clk clock period

- LOW period = (TimerNLoadCount + 1) * timer_N_clk clock period

* timer_N_clk clock period = 10ns

22.20.1.6 Interrupts

The TimerNIntStatus and TimerNEOI registers handle interrupts in order to ensure safe operation of the interrupt clearing.

22.20.2 Signal Description

Table 22-24. PWM Signal Description

Signal Name	Type	Description
PSE_PWM [15:0]	O	Output

22.20.3 Registers

Please refer to chapter 5 of the Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors for Internet of Things (IoT) Applications, Datasheet, Volume 2 (Book 3 of 3), Intel® Programmable Services Engine (Intel® PSE) (Document Number: 636723), for description of the registers associated with subject of this chapter.

22.21 Quadrature Encoder Peripheral (QEP)

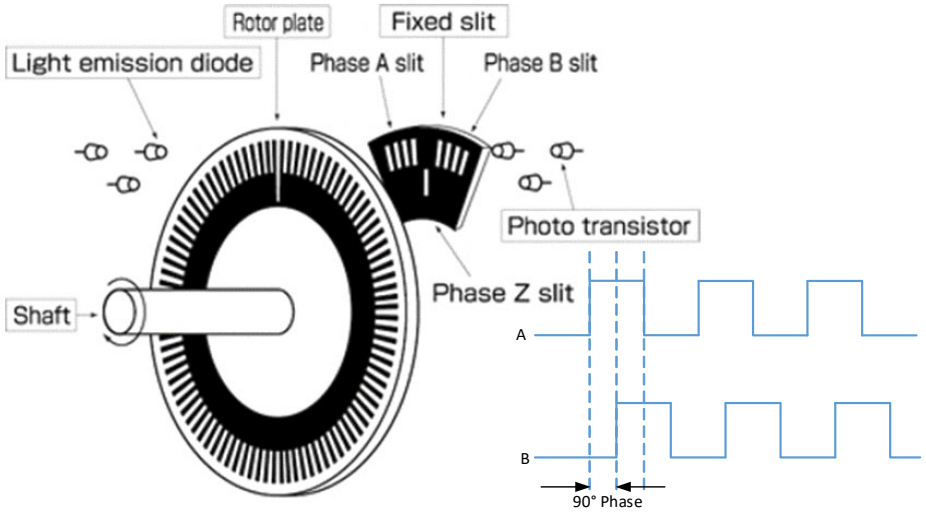
22.21.1 Overview

A quadrature encoder peripheral (QEP) is a peripheral designed to interface to a shaft encoder fitted to either a motor shaft or a user interface dial, as a means to measure rotational speed, direction, and (optionally) absolute angle of rotation. This interface has become popular due to its ability to be electrically isolated from the monitored system, and the absence of frictional components which would wear over time (such as potentiometers.)

The principle of operation of an encoder is shown in the following figure. A slotted wheel is mounted on the shaft through which a constant light beam is passing. Phototransistors detect the change in light level as the slits align with a fixed slotted plate mounted in front of them. The “Phase B” slits are slightly offset from the Phase A slits, resulting in two square wave signals of identical frequency (dependent on speed of rotation) but offset by 90 degrees. The direction of rotation can be inferred from which rising edge leads the other.

An optional third slit, called Phase Z or “Index”, generates a single pulse at one location on the wheel which allows for calculating the absolute rotation angle.

Figure 22-41.Block Diagram of QEP



The number of slits on the wheel can vary, with the resulting phase signal frequency being up to the order of 1 MHz. Figure 22-41 is an example of the physical construction of an encoder; other realizations are possible, but the principle of operation is the same.

In addition to Quadrature decoding, the controller adds a "Capture Compare" capability which enables the processor to capture the time at which an input signal changes level. By capturing several transition in a row, the CPU can measure an input signal's period or pulse length. This is unrelated to quadrature decoding and the controller can be configured to perform either function, but not both simultaneously.

22.21.1.1 Feature

The QEP function supports the following features:

- QEP function is used to measure rotational speed, direction and angle of rotation of an external motor shaft.
- Capture function to measure input signal 'Period' or 'Duty Cycle'.
- Either function can be enabled (but not both simultaneously).
- Connects to either 2 or 3 digital external inputs via pin mux.
  - — Inputs referred to as phase A, phase B and Index.
  - — Phase A and B inputs used to measure speed and direction.
  - — Index input is optional and is used to measure absolute angle of rotation (of say a dial).
- Maximum toggle frequency of phase A and B inputs supported is 10MHz. This applies to both QEP and Capture functions
- Inputs are debounced/filtered, via configurable 'filter clock' to ensure clean transition captured.

  - Number of clock cycles that input signal has to be steady before a transition is detected is configurable. Up to 20 msec of signal 'noise' can be rejected.
- Input transition detection can be configured as 'lo/hi' or 'hi/lo' for QEP.
- Capture function operates on phase A input only, but can be configured to operate on 'lo/hi' or 'hi/lo' or both hi and lo transitions.
  - Capture 'free running counter' clock source (QEPCAPDIV) is configurable. Free running counter to be configured to run at greater than or equal to 16 x input signal frequency.
  - 8 deep fifo to capture timestamps of up to 8 transition events.
  - Interrupt on detection of configurable depth of fifo events.
- Status and control register.
- Interrupt control.
- Watchdog timer to detect 'stall' event when the QEP function is enabled. Generates interrupt when the timer reaches the watchdog comparator value. Activity is indicated by the QEP 'load' signal asserting.

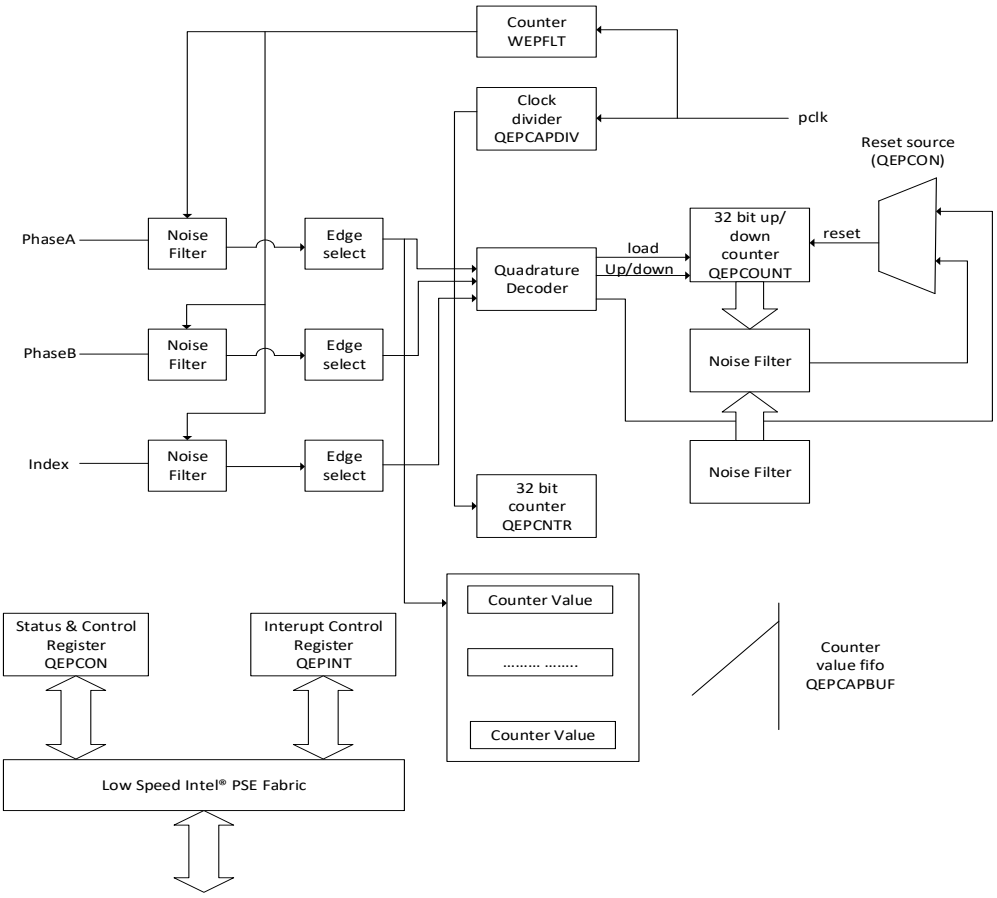
22.21.1.2 Ownership Allocation

- QEP0 ownership can be controlled using OWNERSHIP_DEV27 field in Ownership Control 3 register
- QEP1 ownership can be controlled using OWNERSHIP_DEV28 field in Ownership Control 3 register
- QEP2 ownership can be controlled using OWNERSHIP_DEV29 field in Ownership Control 3 register
- QEP3 ownership can be controlled using OWNERSHIP_DEV30 field in Ownership Control 3 register

22.21.2 Functional Description

The following figure shows the entire block diagram of a controller. In the following subsections each functional block will be explained in detail.

Figure 22-42.Controller Block Diagram



Each input signal can propagate through a programmable noise filter block. This noise filter is able to remove noise spikes typically seen in motion monitoring applications. When the QEPCON.FLTx bit is 1, signal pulses shorter or equal to QEPFLT.MAX_COUNT+2 peripheral clock periods (10ns, since IP clock = 100MHz) will be ignored. The programmable counter value QEPFLT allows to reject glitches in the phase signals up to 20ms. Given the minimum period of the peripheral clock Tpclk=10ns, the size N of the QEPFLT counter is:

$$N = [log_2 \left(\frac{20ms}{T_{pclk}} \right)]$$

Noise filters can be enabled/disabled depending on the value assigned to the QEPCON.FLT_EN bit in the Control and Status Register space.

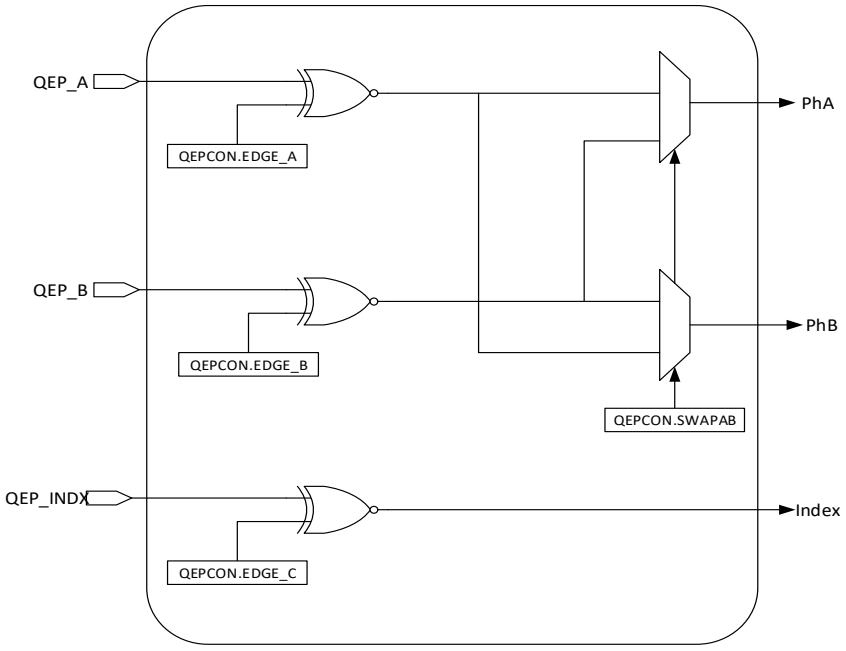
22.21.2.1 Edge Selection and Phase Swapping

An individually configurable Edge Select block allows control over rising or falling edge detection on each input pins, removing the need for platform logic inversion. This feature is controlled by the QEPCON.EDGEx bit, which will invert the pulse coming from the filter when it's set to 0.

Another field of the QEPCON register gives the possibility to swap the quadrature phases. This might correct possible miswirings in the platform implementation. When SWPAB=0, Phase A and Phase inputs will be fed to the A and B input ports of the quadrature counter, respectively. The capture block also will use the edges of Phase A channel. When SWPAB=1, Phase A and Phase B inputs will be interchanged, resulting in an inversion of count direction and in capturing information related to Phase B signal.

The following figure illustrates the edge selection and phase swapping logic.

Figure 22-43.Edge Selection and Phase Swapping Block Diagram



22.21.2.2 Quadrature Decoder

The quadrature Decoder outputs a "Load" pulse for each transition and an "up/down" indication to a 32bit up/down counter. It's employed only when the peripheral is configured to work as QEP.

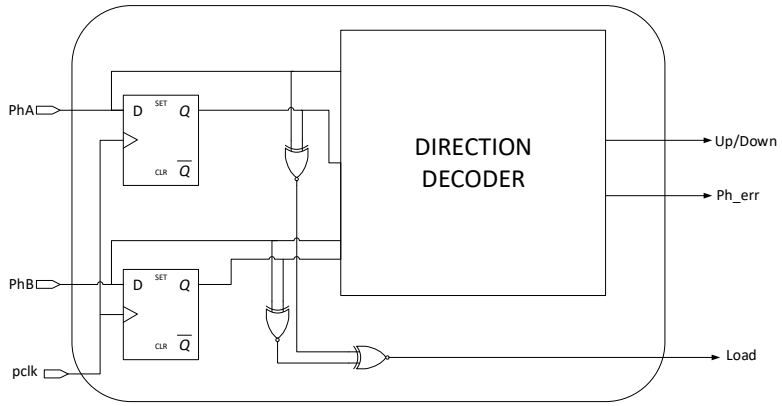
The generated "Load" signal provides a count information which has 4x times the resolution of quadrature phase signals. To do so a pulse generated on every rising/ falling edge of both Phase A and B inputs. This limits the phase frequency f_{Ph} to not violate this expression:

$$f_{ph} \leq \frac{f_{pclk}}{8}$$

where fpclk is the frequency of the peripheral clock, fpclk=100MHz. However, 1MHz is the maximum phases frequency the design is required to support. The "Up/Down" signal indicates if the counter has to increment or decrement its count value when a "Load" pulse is detected. The generation of this signal is based on the phase relationship between the quadrature pulses.

The following figure shows the functional block diagram of the decoder.

Figure 22-44.Quadrature Decoder Block Diagram

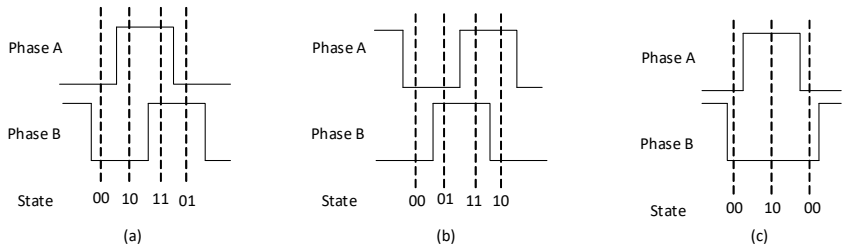


22.21.2.3 Direction Decoding

The direction decoding is able to detect the leading sequence between the incoming PhA and PhB pulses. This allows to determine whether the up/down counter needs to be incremented or decremented on the next "Load" pulse.

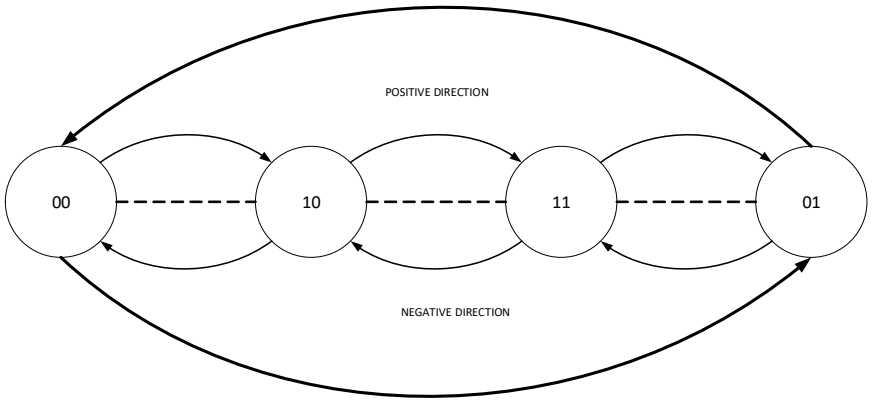
The Up/Down output is high when PhA edges lead PhB ones, as shown in Figure 22-7(a), is low when PhA edges lag PhB ones, as illustrated in Figure 22-7(b), and toggles when a pair of consecutive edges on one phase signal occurs when the other phase is stable, as in the example of Figure 22-7(c). Consequently, the position counter will be incremented on every "Load" pulse if Up/Down is high, decremented if Up/Down is low.

Figure 22-45.Phase relationship example between PhA and PhB signals



The implemented decoding logic determines the positive/negative level of the direction output from the current 'state' of the phases and the one assumed one pclk period earlier. The state diagram is shown in the following figure.

Figure 22-46.State diagram for direction decoding



22.21.2.4 Phase Error

As it's possible to see, some state transitions not allowed by the state machine, like the transition from '00' to '11' or from '10' to '01'. As PhA and PhB are expected to be quadrature inputs, the decoder does not consider scenarios where an edge transition occurs on both the phase signals in the same clock cycle. If this happens, the Up/Down output maintains its previous value and a phase error flag is asserted and exposed to CSR space as a status bit.

22.21.2.5 Position Counter (QEP)

A 32-bit up/down counter is implemented to track the encoder position using the outputs provided by the quadrature decoder. Based on the direction, the count value is incremented or decremented on every Load pulse.

The processor can read the current count value stored in the QEPCOUNT register. This register can be also cleared by software issuing a write access when the peripheral is disabled. On operating condition, the counter can be automatically reset either by the detection of the index pulse or by the counter reaching a particular value defined in the register QEPMAX. The QEPCON.COUNT_RST_MODE register field enables one of these two operating modes.

22.21.2.6 Position Counter Reset on Index Event

In the case when QEPCON.COUNT_RST_MODE is set to 0, the index input is used to align the position count to the absolute position of the encoder calculated from the location of the index slit. In this operating mode, the QEPMAX register needs to be programmed to the number of phase edges in a full revolution of the encoder wheel. Every time a pulse of the index is detected, the count value is overridden with the following value on the correspondent Load pulse:

- '0' if the Up/Down input is equal to 1
- QEPMAX if the Up/Down input is equal to 0

Generally the index marker can be aligned to any transition of Phase A or B and its width can range between 90 degrees and 540 degrees.

In order to have a continuous counting when the direction changes close to the index slit, the index pulse needs to be gated to A and B phases before being sent to the counter. The QEPCON.INDX_GATING register field allows the user to set the state of Phase A and Phase B signals at which the index pulse will be gated.

22.21.2.7 Position Counter Reset on Maximum Position

When the QEPCON.COUNT_RST_MODE is set to 1, the position counter is limited to operate in the range between '0' and the maximum value programmed in the QEPMAX register. If the QEPMAX is set to its full-scale value, the counter works as a general purpose 32-bit timer/counter. The reset event occurs every time the current count value QEPCOUNT matches one of the range boundaries, zero or QEPMAX, in particular:

- When the position counter reaches the QEPMAX value with Up/Down equal to 1, the count value is reset to '0' on the next Load pulse
- When the position counter reaches '0' with Up/Down equal to 0, the count value is reset to QEPMAX on the next Load pulse

The index pulse is not sensed in this mode.

22.21.2.8 Watchdog Timer

A free running 32-bit counter clocked by the peripheral clock is implemented to work as watchdog timer when the module operates as QEP. Its purpose is to detect stall events and issue a dedicated interrupt. The watchdog timer is reset to zero every time a "Load" pulse is generated by the quadrature decoder. A 32-bit watchdog comparator register stores the timeout value QEPWDT.TOP and is writeable by the processor when the peripheral is disabled. When the watchdog timer reaches the value held in the watchdog comparator, the WDT_INTR maskable interrupt is generated.

22.21.2.9 Period/Duty Cycle Counter (Capture Compare)

After noise filtering, edge selection and possibly phase swapping, the PhA signal enters a "Capture Compare" functional block, which records the current value of a free running counter QEPCNTR into a FIFO QEPCAPBUF when the selected edge is detected. A field within the QEPCON register called CAP_MODE sets the FIFO to store the QEPCNTR value on every rising edge or on both edges of PhA. This allows to select different capture options as detailed in the following table.

Table 22-25. Capture compare options

Register settings	Capture compare option
QEPCON.EDGE_A = 1 QEPCON.SWPAB = 0 QEPCON. CAP_MODE = 0	Capture on every rising edge of Phase A
QEPCON.EDGE_A = 0 QEPCON.SWPAB = 0 QEPCON. CAP_MODE = 0	Capture on every falling edge of Phase A
QEPCON.SWPAB = 0 QEPCON. CAP_MODE = 1	Capture on both edges of Phase A
QEPCON.EDGE_B = 1 QEPCON.SWPAB = 1 QEPCON. CAP_MODE = 0	Capture on every rising edge of Phase B
QEPCON.EDGE_B = 0 QEPCON.SWPAB = 1 QEPCON. CAP_MODE = 0	Capture on every falling edge of Phase B
QEPCON.SWPAB = 1 QEPCON. CAP_MODE = 1	Capture on both edges of Phase B

The 32-bit counter is clocked through a postscaler (defined by the register QEPCAPDIV) derived from the peripheral clock. The allowed dividing values are: 1,2,4,8,16,32,64,128. Its count value is readable by the processor through the QEPCNTR register.

The Capture FIFO has a width of 32-bit and can store 8 data entries. The number of entries to store in the FIFO is specified in the QEPCON.FIFO_THRE register field. This 3-bit register field allows to set the "high water mark" threshold at which the FIFO is considered as full and a dedicated interrupt is issued.

The field decode values are specified in the following table.

Table 22-26. FIFO_THRE Decode

FIFO_THRE Value	Description
000	1 data entry
001	2 data entry
010	3 data entry
011	4 data entry
101	5 data enty
100	6 data entry
110	7 data entry
111	8 data entry

Once that number of samples has been received an optional interrupt may be generated, and no further entries will be written to the FIFO until the contents have been read out by the processor through the QEPCAPBUF register. The FIFO can be read while the module is enabled. A FIFO_EMPTY status bit is available in the QEPCON register.

22.21.3 Interrupt

Each controller module has several interrupt sources that can be independently masked. After masking, these active high level interrupts are combined (ORed) onto a single interrupt output. Disabling the peripheral through the QEPCON.EN bit will cause the interrupt flags to be synchronously cleared. The lists below are the interrupt sources (QEPINT_STAT hold the following bits):

- FIFOCRIT Capture Function Event Fifo Critical Interrupt. The number of entries in the 'Capture' fifo has reached the configured threshold level.
- FIFOENTRY Capture Function Event Fifo Entry Interrupt. An entry has been added to the 'Capture' fifo.
- QEPDIR QDP Function Change Of Direction Detected Interrupt. The Quadrature Decoder has detected that the attached motor (or user dial device) has changed direction
- QEPRST QEPCOUNT Reset detect Interrupt; The QEPCOUNT can be reset by one of 3 configurable methods: Index Input detect event, comparator match event or counter 'roll-over' event.
- WDT Watchdog Timeout Interrupt. The Watchdog Timer value has reached the watchdog comparator value.

22.21.4 Usage/ Application Flow

The peripheral provides two different functional modes which are not related and only one of the functions can be supported at a time. The operating mode is selected through the QEPCON.OP_MODE configuration bit. The two usages are explained below.

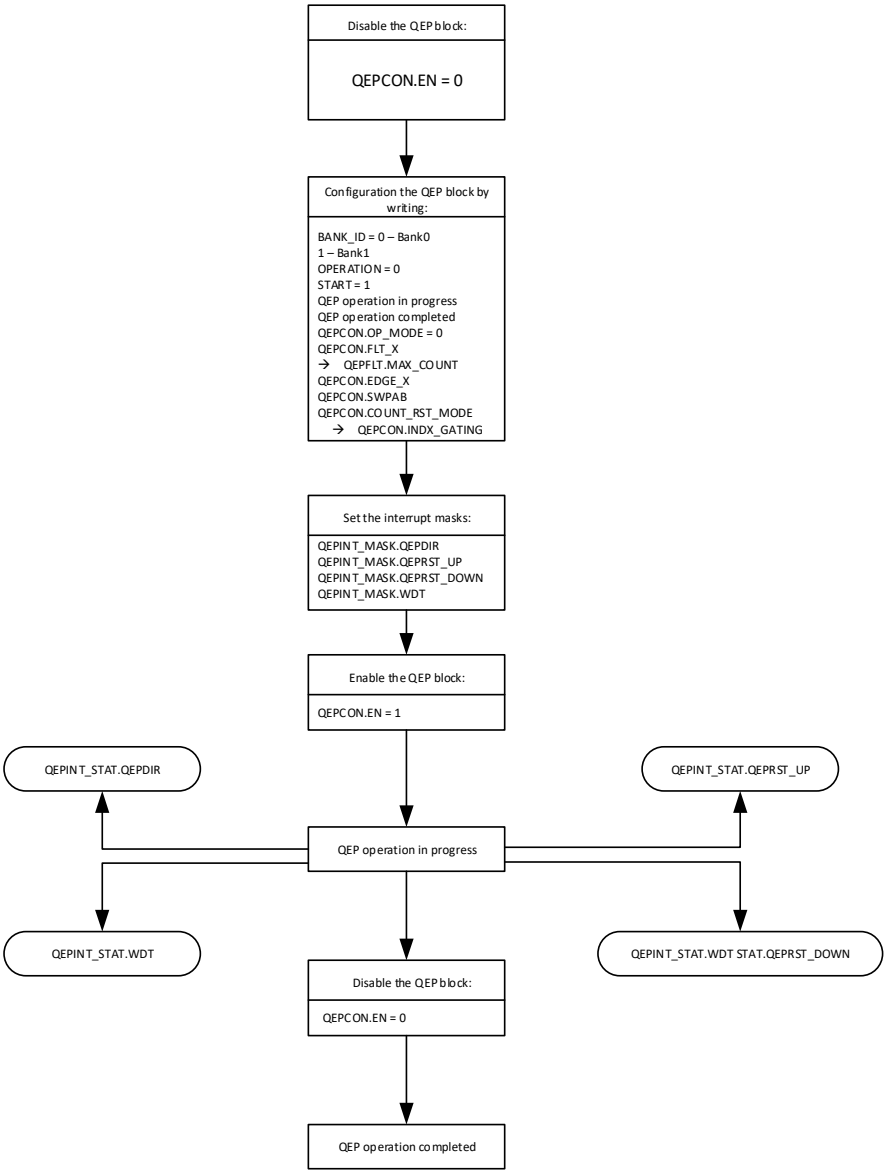
22.21.4.1 Quadrature Encoder Peripheral

The primary function of QEP is to interface to an external shaft encoder fitted to either a motor shaft or a user interface dial, as a way of measuring rotational speed, direction and absolute angle of rotation.

Note: This function is also referred to as "Quadrature Encoder Peripheral" or QEP in the literature on this topic.

The flow diagram in the following figure shows an overview of programming the module to use the QEP functionality.

Figure 22-47.Software Flow Diagram for QEP Functionality

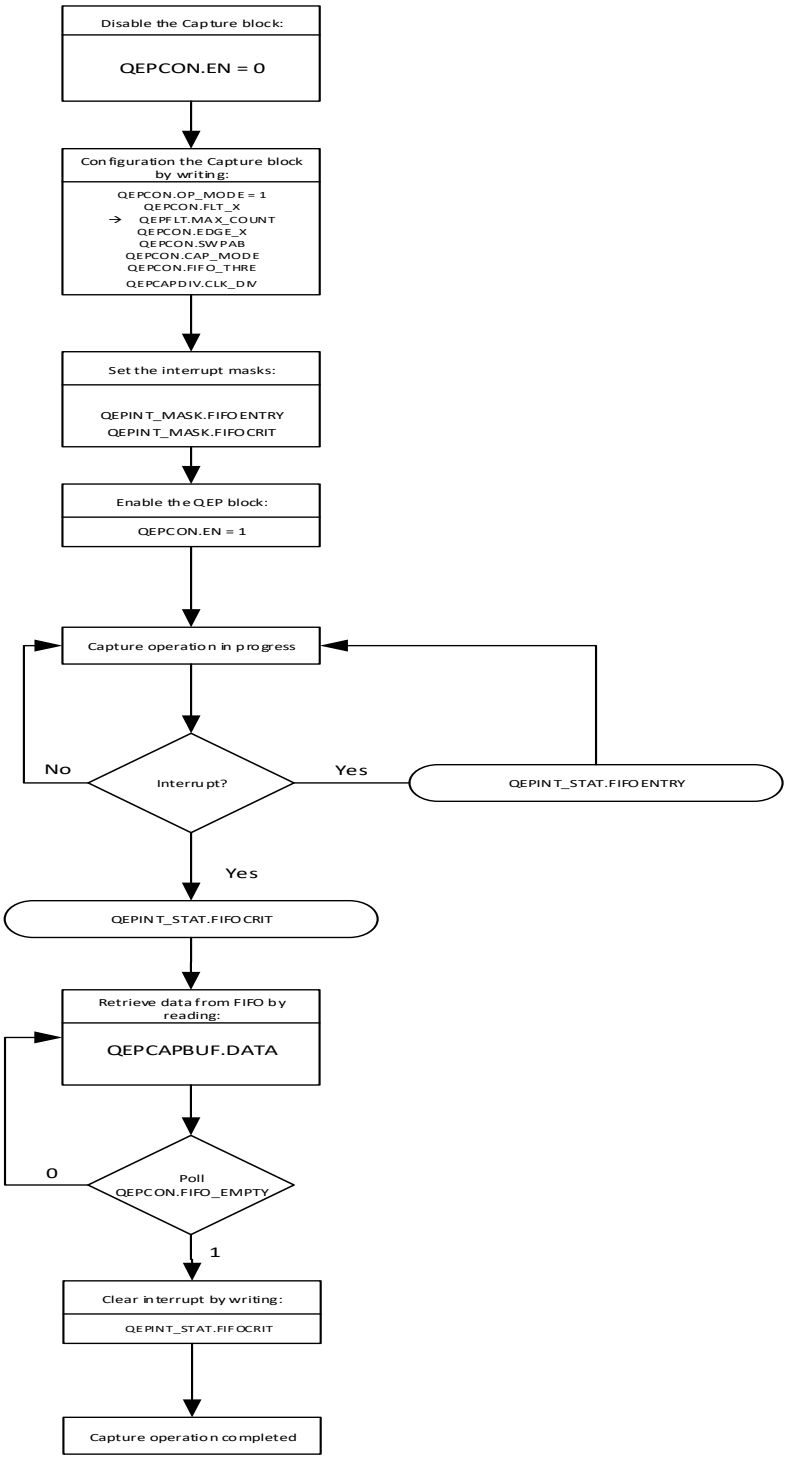


22.21.4.2 Capture Compare

A secondary function of this module is to ‘capture’ a series of ‘timestamps’ of the time that an input (digital) signal changes state from HI to LO (or vice-versa or indeed any state change). By capturing a series of such transitions in sequence, the CPU can measure the signals period.

The following figure shows a typical software flow to follow when programming the module to work as Capture Compare block.

Figure 22-48.Software flow diagram for Capture Compare Functionality



22.21.5 Signal Description

Table 22-27. QEP Signal Description

Signal Name	Type	Description
PSE_QEP_A [3:0]	I	Phase A signal input
PSE_QEP_B [3:0]	I	Phase B signal input
PSE_QEP_I [3:0]	I	Index signal input

22.21.6 Registers

Please refer to chapter 9 of Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors for Internet of Things (IoT) Applications, Datasheet, Volume 2 (Book 3 of 3), Intel® Programmable Services Engine (Intel® PSE) (Document Number: 636723), for a description of the registers associated with subject of this chapter.

22.22 Time Synchronous Support

22.22.1 Time Synchronous Implementation

Intel® PSE has three instances of the Time Synchronous timer running on XTAL, AON and RTC clocks simultaneously and are synced with the PMC Always Running Time (ART) during time synchronization flow. If the Intel® PSE enters D0ix power states, then the XTAL clock is gated. In these cases, the HW updates the XTAL clock timer value with the AON clock timer when exiting the D0ix state. This would help avoid the Intel® PSE FW to initiate the entire Time Synchronous flow in case of D0ix entry and exit. However, there is an error of one XTAL clock period possible during every entry and exit and could accumulate after multiple D0ix entry and exit. If this error accumulation cannot be tolerated, then the Intel® PSE FW needs to re-sync all the timers with the PMC ART using the Time Synchronous flow.

When Intel® PSE enters deeper power states like D0i2 and D0i3, the AON clock can be optionally gated. In these scenarios, the RTC clock Time Synchronous timer is the only timer operational and hence ensures the Time Synchronous time is kept with a lower accuracy of RTC period. During D0i2/3 exit, HW updates the AON clock timer with the RTC timer value. This update could also have an error of one AON clock.

22.22.2 Time Synchronous Support In The Arm* Cortex*-M7

The Arm* Cortex*-M7 does not support reading Time Synchronous time using a default instruction. For firmware to read the Time Synchronous value, it must do a MMIO read to the Time Synchronous timer block in the Time-Aware GPIO.

22.23 DMA

22.23.1 Overview

The Intel® PSE supports 3 DMA engines with the following features:

- 8-channel DMA

- All channels have 2-cache line size data buffer
- 8 outstanding reads and 2 outstanding non-posted writes supported
- Link-list and Direct modes of operation supported
- Out of band DMA completion interrupt wire routed to ARM/IA based on ownership
- Support for HW DMA handshake mode for I^2C, SPI, UART, I^2S modules
- DMA HW can support up to 64b addressing
- Support for clock gating
- DMA will support channel disable mode to enable exception handling
- DMA irrespective of PSE/IA ownership can copy data into main memory (DRAM) but it has some restrictions (Refer Table 22-28)

22.23.1.1 DMA Capabilities

The following table shows the DMA capabilities and restrictions wen owned by the Arm* Cortex*-M7 and when owned by the IA Processor (Local Host):

Table 22-28. DMA Capabilities/Restrictions

DMA Engine	RS0	RS1	RS3	VC0	VC1	S/NS	Descriptors	Payload	Peripheral<->SRAM*	Peripheral<->DRAM	SRAM*<->SRAM*	SRAM*<->DRAM	DRAM<->DRAM
ARM owned DMA0/1/2	Y*	Y*	Y*	Y*	Y*	Y*	SRAM	SRAM/ DRAM	Y	N	Y	Y	Y
HOST owned DMA0/1/2	Y	N	N	Y	Y	Y	DRAM	DRAM	N	Y	N	N	Y
ARM owned GbE0/1	Y	N	N	Y	Y	Y	SRAM/DRAM	SRAM/ DRAM*	Y	Y	N	N	N
HOST owned GbE0/1	Y	N	N	Y	Y	Y	DRAM	DRAM	N	Y	N	N	N

Y* -> All channels must be same RS/VC

SRAM* -> L2, CCM, AONRF

ARM Owned GbE - Both payload and descriptors have to be in SRAM or DRAM

S/NS - Snooping/Non-Snooping

22.23.2 Functional Description

22.23.2.1 Hardware Handshake with DMA

DMAs support HW HS (Hand Shake) signals with IO controllers. There are 8 channels in each DMA, each channel capable of supporting HW HS. Thus, at a given time, there can be a max total of 24 HW HS protocols ongoing on all of the DMAs. But, there are >24 IO controllers which are capable of HW HS support. Thus, DMA controller implements a DMA HW HS crossbar to allocate ANY given IO controller's HW HS signaling to be allocated to any of the DMA's channels. This is done via “DMA_XBAR_SELx”.

Peripheral Assignment

The table below shows the peripheral assignments. This assignment holds for the DMA handshake routine. IA Processor (Local Host) firmware or Intel® PSE firmware will program the DEVFUNC/Direction in the DMA_XBAR_SELx to allocate a particular peripheral's handshake signal to a corresponding DMA channel. If the incorrect DEVFUNC is programmed, then the DMA_REQ will be driven to 0.

Table 22-29. DMA Hardware Handshake Peripheral Assignments

HW Handshake	DEVFUNC	Direction	Internal Index
I^2C0 RX	D8	0	0
I^2C0 TX	D8	1	1
I^2C1 RX	D9	0	2
I^2C1 TX	D9	1	3
I^2C2 RX	DA	0	4
I^2C2 TX	DA	1	5
I^2C3 RX	DB	0	6
I^2C3 TX	DB	1	7
I^2C4 RX	DC	0	8
I^2C4 TX	DC	1	9
I^2C5 RX	DD	0	10
I^2C5 TX	DD	1	11
I^2C6 RX	DE	0	12
I^2C6 TX	DE	1	13
I^2C7 RX	C0	0	14
I^2C7 TX	C0	1	15
UART0 RX	88	0	16
UART0 TX	88	1	17
UART1 RX	89	0	18
UART1 TX	89	1	19
UART2 RX	8A	0	20
UART2 TX	8A	1	21
UART3 RX	8B	0	22
UART3 TX	8B	1	23
UART4 RX	8C	0	24
UART4 TX	8C	1	25
UART5 RX	8D	0	26
UART5 TX	8D	1	27
SPI0 RX	98	0	28
SPI0 TX	98	1	29
SPI1 RX	99	0	30
SPI1 TX	99	1	31
SPI2 RX	9A	0	32
SPI2 TX	9A	1	33
SPI3 RX	9B	0	34
SPI3 TX	9B	1	35
I2S0 RX	8E	0	36
I2S0 TX	8E	1	37
I2S1 RX	8F	0	38
I2S1 TX	8F	1	39

For each DMA there are 8 hardware handshake select registers - one corresponding to each channel. Each register is 17-bit with lower 8 bits holding the DEVFUNC number and bit 16 determining the direction of transfer: 0 - RX and 1 - TX.

The DMA driver needs to program this register with the DEVFUNC values corresponding to the peripheral that is requesting for the DMA channel.

22.23.2.2 DMA Transfer and Setup Modes

The DMA can operate in the following modes:

- Memory to memory transfers - This assumes that there is always space available in the destination. This transfer has to be completed in the most optimized way. It should utilize the maximum burst size allowed with the maximum transfer widths on the source and destination sides.
- Memory to peripheral transfers - This mode requires that the peripheral control the flow of the data to itself. A separate sideband interface is defined for this purpose.
- Peripheral to Memory transfers - This mode requires that the peripheral control the flow of the data from itself. The same sideband interface will be used.

The DMA supports the following modes for programming:

- Direct programming - Direct register writes to DMA registers to configure and initiate the transfer. For more information please refer to DIRECT PROGRAMMED TRANSFERS.
- Descriptor based linked lists - The descriptors will be stored in memory (DRAM or SRAM or elsewhere). The DMA will be informed with the location information of the descriptor. DMA initiates reads and program its own registers. The descriptors can form a linked list for multiple blocks to be programmed.

22.23.2.3 Types of DMA Transfers

Direct Programmed Transfers

Direct programmed transfers simply mean that the DMA Registers will be programmed by the CPU directly. For more information on how to program the DMA using this mode refer to COMMON DMA USAGE FLOWS.

Descriptor Based Linked List Transfers

Descriptor based transfers (also referred to as linked list based transfers or multi block transfers involve setting up a linked list in memory. The format of the linked list is as follows (or a variation of this)

Offset 0x18:[31:0]	Write back for DSTATn
Offset 0x14:[31:0]	Write back for SSTATn
Offset 0x10:[31:0]	CTL_HIn
Offset 0x0C:[31:0]	CTL_LOn
Offset 0x08:[31:0]	LLPn
Offset 0x04:[31:0]	DARn
Offset 0x0:[31:0]	SARn

Once the channel doing the Descriptor based transfer is enabled, the DMA initiates a read to the address in the Channel Linked List Pointer Register. It programs its own registers based on the received data and begins the transfer. If the Pointer to next linked list element is "32'h 00000000" then the DMA assumes that this is the last block of the transfer. If not, then it fetches the next descriptor and continues with the transfer. Hence for the latter case, all relevant fields in the CTL_HI/LO registers needs to be programmed in the memory descriptor itself (i.e. LLi.CTL_HI/LO) and not in the physical registers (CTL_HI/LO). Also, fields such as channel class/weight should not be varied from one descriptor blocks to another in linked-list-multi-block transfer for a certain channel.

At the end of each block the DMA can be configured to generate an interrupt indicating completion of the block. Alternatively it can generate an interrupt only at the end of the complete transfer.

22.23.2.4 Common DMA Usage Flows

Transfers are set up by programming fields of the CTLx and CFGx registers for that channel.

The following table lists the parameters that are investigated in the following examples. The effects of these parameters on the flow of the block transfer are highlighted in the examples that follow.

Table 22-30. Parameters Used for DMA Setup

CTL_LOn.TT_FC	Transfer type and flow control
CTL_HIn.BLOCK_TS	Block Transfer size
CTL_LOn.SRC_TR_WIDTH	Encoded Source transfer width
CTL_LOn.DST_TR_WIDTH	Encoded Destination transfer width
CTL_LOn.SRC_MSIZE	Source burst transaction length
CTL_LOn.DST_MSIZE	Destination burst transaction length

The DMA Usage flows are categorized with examples as follows:

1. Memory to Memory transfers - Direct programming
   a. Basic Block transfer example.
2. Memory to Memory transfers - Multi Block Transfers
3. Memory to Peripheral transfers - Direct programming
4. Peripheral to Memory transfers - Direct programming

All of the above modes assume that the DMA is the flow controller. Peripheral flow control is not supported by this DMA. For flow control definition refer to Basic Definitions.

The following definitions are used in the examples that follow:

Source single transaction size in bytes

src_single_size_bytes = (2 ^ CTL_LOn.SRC_TR_WIDTH)

Source burst transaction size in bytes

src_burst_size_bytes = (2 ^ CTL_LOn.SRC_MSIZE) * src_single_size_bytes

Destination single transaction size in bytes

dst_single_size_bytes = (2 ^ CTL_LOn.DST_TR_WIDTH)

Destination burst transaction size in bytes

dst_burst_size_bytes = (2 ^ CTL_LOn.DEST_MSIZE) * dst_single_size_bytes

Block size in bytes

blk_size_bytes = CTL_HIn.BLOCK_TS

Memory to Memory Transfers – Direct Programming

The following programming is required to initiate a memory to memory transfer:

1. Check DMA channel usage bits to figure out which channels are currently in use and identifying a free channel to be used for the current transfer.
2. The identified DMA channel is put in to a memory to memory transfer mode.
3. The channel is also put in a Direct programming mode for a single block
4. Source address is programmed.
5. Destination address is programmed.
6. Block size is programmed.
7. Unmasking of Interrupt status bit for the identified DMA Channel is done so that the DMA can assert a level interrupt line at the end of the block transfer.
8. Enabling the channel to begin the transfer.
9. On receiving the interrupt the DMA status register is read to understand which channel completed.
10. The busy bit indicating the “channel in use” status is cleared automatically.

Basic Block Transfer Example

For a basic DMA transfer from the source to the destination the following parameters will control the transfer.

Table 22-31. Basic Block Transfer Example Settings

Parameter	Description
CTL_LOn.TT_FC = 3’b000	Memory to Memory transfer
CTL_HIn.BLOCK_TS = 48	Block size = 48 bytes
CTL_LOn.SRC_TR_WIDTH = 3’b010	4 Bytes per DW
CTL_LOn.DST_TR_WIDTH = 3’b010	4 Bytes per DW
CTL_LOn.SRC_MSIZE = 3’b010	Source burst length = 4 DW
CTL_LOn.DEST_MSIZE = 3’b010	Destination burst length = 4 DW

The transfer will result in reads and writes with Burst-Length set to 4DW with total transfer for 4DW*4 Bytes per DW = 16 Bytes. Therefore to transfer 48 bytes, 3 reads and 3 writes will be required to complete the block transfer.

Memory to Memory Transfers - Multi Block Transfers

The following programming is required to initiate a linked list based memory to memory transfer:

A linked list is setup in memory. It has to be setup at a Dword aligned address in memory. The format to be followed is described in "Descriptor Based Linked List Transfers". The Figure 22-49 shows how the linked list can be setup in memory to define multiblock transfers.

Check DMA channel usage bits to figure out which channels are currently in use and identifying a free channel to be used for the current transfer.

The identified DMA channel is put in to a memory to memory transfer mode.

The first location of the Linked list is put in the LLP[n]: Linked List address Register. Please notice that having a non-zero value in this register is an indication to the DMA controller that it needs to fetch the descriptor from the memory address pointed to by LLP[n].

The DMA channel is programmed to either provide interrupts at the end of each block or at the end of the entire transfer.

The DMA channel is enabled.

The CPU reads the status registers on receiving the interrupts.

On completion of the transfer the "channel in use" status bit is cleared automatically.

For details of the various modes and register programming please refer to the following table and associated flowchart in the following figure.

*** Flowchart in the following figure is supposed to be an aid in understanding the following table. However, in case there is a conflict/inconsistency between the flowchart and the table (or any other reference), the flowchart reflects the real implementation of the DMA and hence overrules any conflict or inconsistency.

Table 22-32. Programming of Transfer Types and Channel Register Update Method

Transfer type	LLP.LOC = 0	LLP_SRC_EN (CTL_LOn)	RELOAD_SRC (CFG_LOn)	LLP_DST_EN (CTL_LOn)	RELOAD_DST (CFG_LOn)	Update Methods			
						CTL_LOn, LLPn	SARn	DARn	Write Back
Single-block or last transfer of multi-block	Yes	0	0	0	0	None, user reprograms	None	None	No
Auto-reload multi-block transfer with continuous SAR	Yes	0	0	0	1	CTL_LOn, LLPn are reloaded from initial values	C	AR	No
Auto-reload multi-block transfer with continuous DAR	Yes	0	1	0	0	CTL_LOn, LLPn are reloaded from initial values	AR	C	No
Auto-reload multi-block transfer	Yes	0	1	0	1	CTL_LOn, LLPn are reloaded from initial values	AR	AR	No
Single-block or Last transfer of multi-block	No	0	0	0	0	None, user reprograms	None	None	Yes
Linked list multi-block transfer with continuous SAR	No	0	0	1	0	CTL_LOn, LLPn loaded from next Linked List item	C	LL	Yes
Linked list multi-block transfer with auto-reload SAR	No	0	1	1	0	CTL_LOn, LLPn loaded from next Linked List item	AR	LL	Yes
Linked list multi-block transfer with continuous DAR	No	1	0	0	0	CTL_LOn, LLPn loaded from next Linked List item	LL	C	Yes
Linked list multi-block transfer with auto-reload DAR	No	1	0	0	1	CTL_LOn, LLPn loaded from next Linked List item	LL	AR	Yes
Linked list multi-block transfer	No	1	0	1	0	CTL_LOn, LLPn loaded from next Linked List item	LL	LL	Yes

1. C = Contiguous
2. AR = Auto Reload
3. LL = Linked List

Figure 22-49.DMA Multi-Block and Update Flowchart

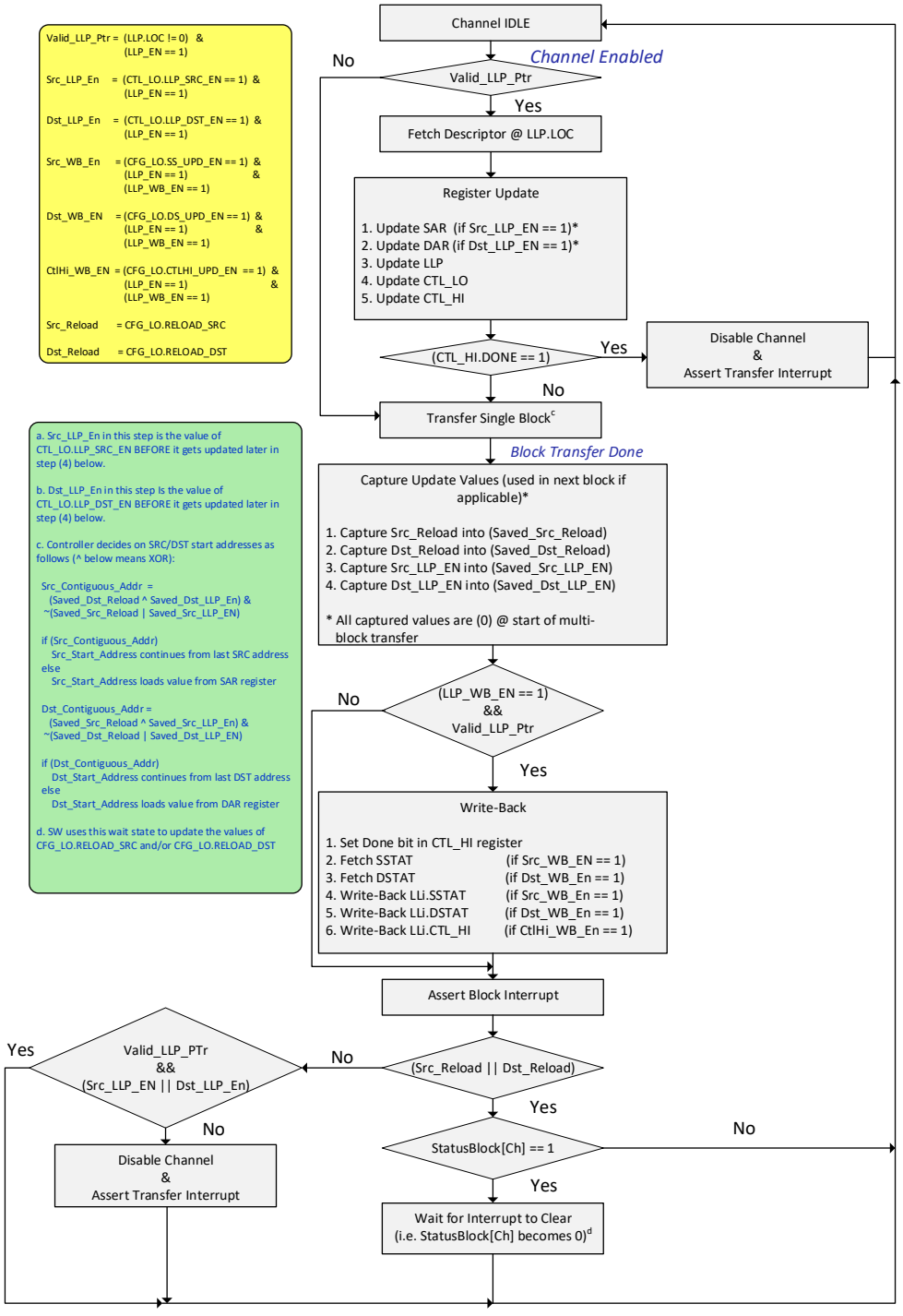
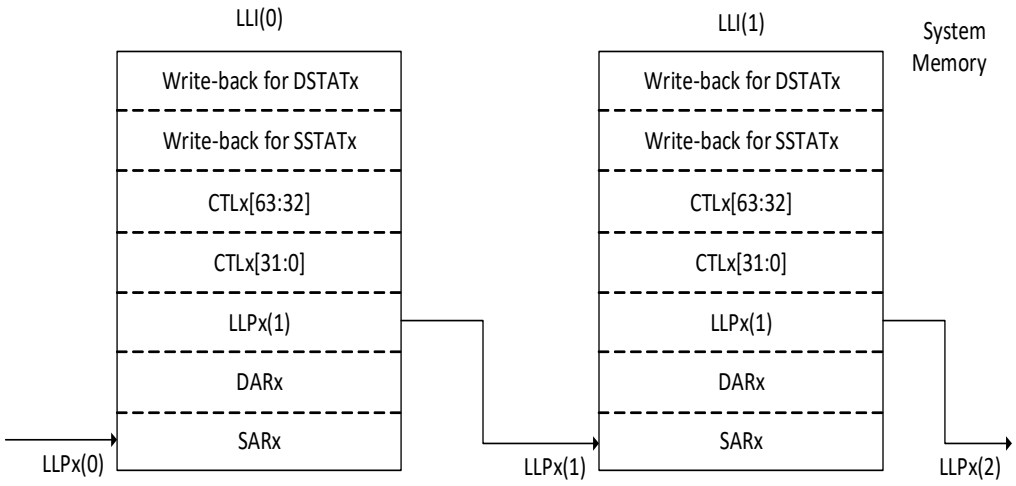


Figure 22-50.Multi-Block Transfer Setup using Linked Lists



Suspension of Transfers between Blocks

At the end of every block transfer, an end-of-block interrupt is asserted if:

1. Interrupts are enabled, CTL_LOn.INT_EN = 1, and
2. The channel block interrupt is unmasked, MaskBlock[n] = 1, where n is the channel number.

Note: The block-complete interrupt is generated at the completion of the block transfer to the destination.

For rows 6, 8, and 10 of Table 22-37, the DMA transfer does not stall between block transfers. For example, at the end-of-block N, the DMA automatically proceeds to block N + 1.

For rows 2, 3, 4, 7, and 9, the DMA transfer automatically stalls after the end-of-block interrupt is asserted, if the end-of-block interrupt is enabled and unmasked.

The DMA does not proceed to the next block transfer until a write to the ClearBlock[n] block interrupt clear register, done by software to clear the channel block-complete interrupt, is detected by hardware.

For rows 2, 3, 4, 7, and 9, the DMA transfer does not stall if either:

- Interrupts are disabled, or
- The channel block interrupt is masked, MaskBlock[n] = 0, where n is the channel number.

Channel suspension between blocks is used to ensure that the end-of-block ISR (interrupt service routine) of the next-to-last block is serviced before the start of the final block commences. This ensures that the ISR has cleared the CFG_LOn.RELOAD_SRC and/or CFG_LOn.RELOAD_DST bits before completion of the final block. The reload bits CFG_LOn.RELOAD_SRC and/or CFG_LOn.RELOAD_DST should be cleared in the end-of-block ISR for the next-to-last block transfer.

Ending Multi-Block Transfers

All multi-block transfers must end as shown in either Row 1 or Row 5 of the preceding table. At the end of every block transfer, the DMA samples the row number, and if the DMA is in the Row 1 or Row 5 state, then the previous block transferred was the last block and the DMA transfer is terminated.

Note: Rows 1 and 5 are used for single-block transfers or terminating multi-block transfers. Ending in the Row 5 state enables status fetch and write-back for the last block. Ending in the Row 1 state disables status fetch and write-back for the last block.

For rows 2, 3, and 4, (LLPn = 0 and CFG_LOn.RELOAD_SRC and/or CFG_LOn.RELOAD_DST is set), multi-block DMA transfers continue until both the CFG_LOn.RELOAD_SRC and CFG_LOn.RELOAD_DST registers are cleared by software. They should be programmed to 0 in the end-of-block interrupt service routine that services the next-to-last block transfer; this puts the DMA into the Row 1 state.

For rows 6, 8, and 10 of (both CFG_LOn.RELOAD_SRC and CFG_LOn.RELOAD_DST cleared), the user must set up the last block descriptor in memory so that both LLI.CTL_LOn.LLP_SRC_EN and LLI.CTL_LOn.LLP_DST_EN are 0. If the LLI.LLPn register of the last block descriptor in memory is non-zero, then the DMA transfer is terminated in Row 5. If the LLI.LLPn register of the last block descriptor in memory is 0, then the DMA transfer is terminated in Row 1.

For rows 7 and 9, the end-of-block interrupt service routine that services the next-to-last block transfer should clear the CFG_LOn.RELOAD_SRC and CFG_LOn.RELOAD_DST reload bits. The last block descriptor in memory should be set up so that both the LLI.CTL_LOn.LLP_SRC_EN and LLI.CTL_LOn.LLP_DST_EN registers are 0. If the LLI.LLPn register of the last block descriptor in memory is non-zero, then the DMA transfer is terminated in Row 5. If the LLI.LLPn register of the last block descriptor in memory is 0, then the DMA transfer is terminated in Row 1.

Note: The only allowed transitions between the rows of the preceding table are from any row into Row 1 or Row 5. As already stated, a transition into row 1 or row 5 is used to terminate the DMA transfer; all other transitions between rows are not allowed. Software must ensure that illegal transitions between rows do not occur between blocks of a multi-block transfer. For example, if block N is in row 10, then the only allowed rows for block N +1 are rows 10, 5, or 1.

If interrupts are disabled, user can poll for the transfer complete raw interrupt status register (RawTfr[n], n = channel number) until it is set by hardware, in order to detect when the transfer is complete. Note that if this polling is used, the software must ensure that the transfer complete interrupt is cleared by writing to the Interrupt Clear register, ClearTfr[n], before the channel is enabled.

Programming examples for Multiblock transfers are explained in detail in the Programming examples section.

Memory to Peripheral Transfers - Direct Programming

These notes apply to Peripheral to Memory transfers – Direct programming as well.

The following programming is required to initiate a Memory to Peripheral transfer and Peripheral to Memory transfers.

1. Check DMA channel usage bits to figure out which channels are currently in use and identifying a free channel to be used for the current transfer.

2. The identified DMA channel is put in to a memory to peripheral transfer mode.
3. The hardware handshake interface (one of the 16 per DMA) to which the transmit side of the peripheral is connected to is identified. The identified DMA channel is programmed to use the signals from this hardware handshake interface. To see details of the registers where this programming is done, refer to Per Channel Register Space
4. For Peripheral to Memory transfers the hardware handshake interface should be connected to the Receive side of the peripheral interface.
5. The source and the destination addresses are programmed as in the Memory to Memory transfer case. The control register is also programmed.
6. Some peripherals might require a non-incrementing address whereas some require incrementing ones.
7. For more details on how a peripheral and DMA should be programmed in conjunction when doing a Memory to Peripheral transfer please refer to Hardware Handshaking
8. The DMA interrupts are unmasked, and the DMA channel is enabled.
9. On completion of the transfer the “channel in use” status bit is cleared automatically.

22.23.3 Registers

Please refer to chapter 4 of the Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors for Internet of Things (IoT) Applications, Datasheet, Volume 2 (Book 3 of 3), Intel® Programmable Services Engine (Intel® PSE) (Document Number: 636723), for a description of the registers associated with subject of this chapter.

§ §

23 Intel® Safety Island (Intel® SI)

23.1 Feature Overview

Intel® Safety Island (Intel® SI) is designed to be SIL2 and SC3 compliant per IEC 61508:2010.

Intel® SI is based on an Arm* Cortex M3 micro controller coupled with fRCPU_armcm3, an asymmetric lock-step monitor. Intel® SI implements a number of interfaces for error reporting to the system, communication to external platform components and for monitoring of a second processor in support of a 1oo2D safety architecture. OEM can change Intel® SI configuration parameters, to update configurations in Intel® SI, new image needs to be created and downloaded in Intel® SI.

Intel® SI functions as a dedicated diagnostic IP which collects safety related errors originating from different blocks/parts of processor and signals to the system using dedicated pins. The major use cases of Intel® SI are (but not limited to) listed below:

- Intel® SI takes action after fault detection and communicates with system
- Intel® SI logs error information to storage for all errors occurring within the power-on to power-off interval
- Intel® SI monitors fatal, non-fatal, corrected and advisory errors reported within the processor
- Intel® SI controls or executes on demand proof test diagnostics.
- Intel® SI controls or executes diagnostic measures.
- Intel® SI detects violations of Worst Case and Best Case Execution Time.
- Intel® SI reaction and diagnostics can be configured in the field.

Figure 23-1. Intel® SI Block Diagram

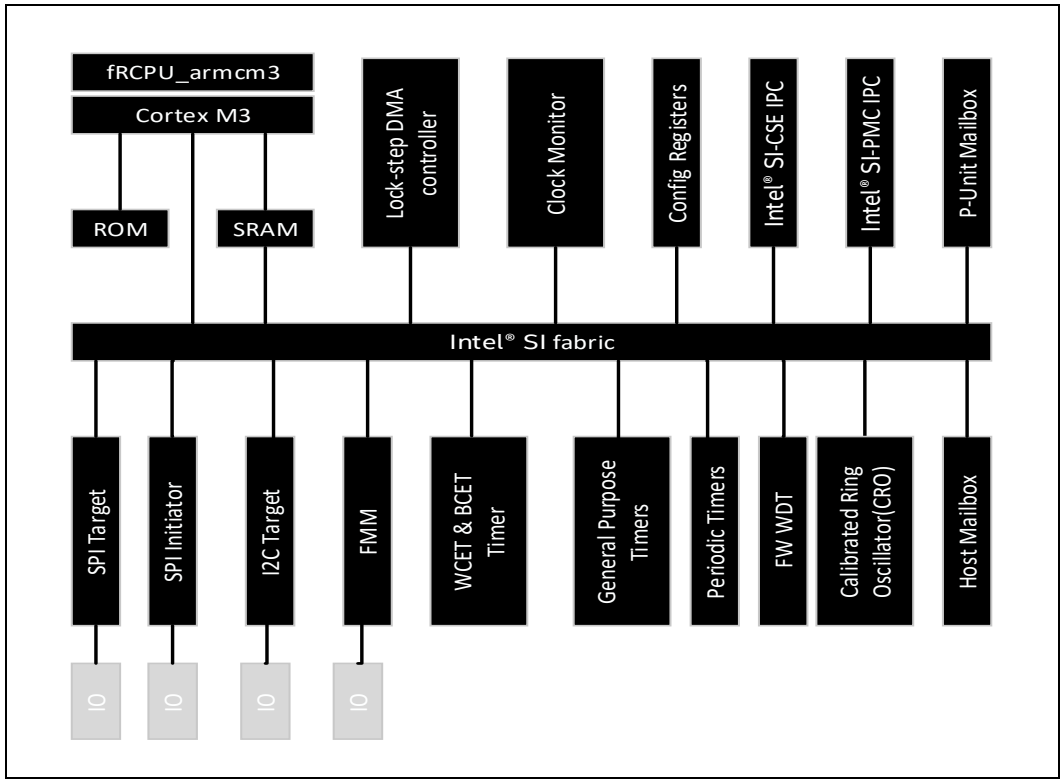


Table 23-1. Signal Description

Signal Name	Type	Description
ISI_OK (ISI_OKNOK_0)	O	**OK/NOK**: Antivalent signals for error indication to the system. The system shall transition to safe state for any of following OK/NOK states: 0/0 (power off) 0/1 (error state) 1/1 (reset state) The system is in normal operating state "OK" for the following OK/NOK states. 1/0 (no fault)
ISI_NOK (ISI_OKNOK_1)		
ISI_CHX_OK (ISI_CHX_OKNOK_0)	I	**1oo2D:** OK of other Intel Atom® x6000E Series processors, Intel® Pentium® and Celeron® N and J Series processors channel
ISI_CHX_NOK (ISI_CHX_OKNOK_1)		**1oo2D:** NOK of other Intel Atom® x6000E Series processors, Intel® Pentium® and Celeron® N and J Series processors channel
ISI_ALERT_N	O	Indicates that a correctable error occurred. In a "smart system" this signal is typically routed to the MCU and the MCU is responsible to ask for additional diagnostic info over SPI. Intel® SI de-asserts Alert# on receiving the diagnostic info request. For basic systems the signal is not used and stays active once the first correctable error is detected. Active low signal
ISI_CHX_RLY_SWITCH	O	**1oo2D**: Active high, Platform level relay switch override 1'b1: Override the Relay Switch and turn OFF the switch 1'b0: no override
ISI_CHX_PMIC_EN	O	**1oo2D**: Active high, PMIC Enable override 1'b1: Override the PMIC enable and turn OFF the PMIC 1'b0: no override
ISI_SPIM_CS	O	SPI Initiator Chip Select Output Enable
ISI_SPIM_SCLK	O	SPI Initiator Clock
ISI_SPIM_MOSI	O	Initiator SPI: Initiator Out / Target In
ISI_SPIM_MISO	I	Initiator SPI: Initiator In Target Out
ISI_SPIS_CS	I	SPI Target Chip Select
ISI_SPIS_SCLK	I	SPI Target Clock
ISI_SPIS_MOSI	I	Target SPI: Initiator Out / Target In
ISI_SPIS_MISO	O	Target SPI: Initiator In Target Out
ISI_I2CS_SCL	I/O	I^2C Target Serial Clock Line
ISI_I2CS_SDA	I/O	I^2C Target Serial Data Line
Note: Refer to **Section 1.1.1** for more information on initiator and target		

23.1.1 Functional Description

Intel® SI has the following key components:

- An Arm Cortex M3 based microcontroller with a SIL3 certified asymmetric lockstep monitor (fRCPU_armcm3).
- Boot ROM and SRAM are both protected by SEC-DED ECC
- Calibrated Ring Oscillator (CRO) which will generate the clock for the Intel® SI.

- A DMA engine operating in lock-step that fetches Intel® SI's FW during boot and for data transfer between Intel® SI and system memory during runtime.
- HW based WDT to monitor FW timeout.
- General purpose timers.
- Clock monitors for processor RTC & XTAL clocks as well as Intel® SI's functional clock.
- Parity protected configuration registers.
- An error collection hub that receives errors from processor, classifies them into severity levels and reports them to the system or an external MCU by asserting OKNOK[1:0] or ISI_ALERT_N accordingly.
- RTOS timer used by the FW for WCET (Worst Case Execution Time) & BCET (Best Case Execution Time) checks.
- 61bit programmable periodic timer with a triple redundant counter.
- General purpose timers.
- Interfaces to an external platform component (I^2C target, SPI initiator, SPI target)
- IOSF-primary and IOSF-SB interfaces
- Separate IPC with CSE and PMC
- Mailbox for communication with Punit and Host.

23.1.2 Clock Monitoring

The Intel® SI provides clock monitoring for the processor XTAL and RTC clocks, using the Intel® SI clock as a reference with cross-monitoring:

- monitor the frequency drift in the target clock, to be within a fixed percentage (PPM)
- check if the target clock is running or dead
- check if the clock valid signal is always asserted during the clock enabled period
- error injection with corresponding alarm masking capability for the above checks

23.2 Error Reporting

Intel® SI collects errors reported in the processor, classifies them and reports them to the system according to their classification.

23.2.1 Fault Management Module (FMM) HUB

FMM module is primarily responsible for collecting alarm signals from Compute Die, PCH and the peripherals within the Intel® SI subsystem. It classifies the received alarms according to configurable severity, generates OK_NOK, ALERT indication to the safety MCU. It also supports the following functions

- Generates configurable interrupts to Processor Sub System based on severity levels.
- Supports HW based parity checks for all the configuration registers.
- Supports Alarm injection to artificially introduce any kind of alarms based on specific register configuration.

HW Loopback status register for OK_NOK, ALERT signals.

23.2.1.1 Error Classification

Errors generated by the detection of faults by hardware diagnostic measures are classified by the FMM into the following types:

- **Un-correctable Fatal errors**, Non-recoverable and uncorrected errors. Intel® SI triggers a fatal error by driving OKNOK[1:0] to 2'b01
- **Un-correctable Non-fatal errors** are signaled to the software which is accountable to recover from the failure condition. If SW fails to recover from the failure condition within a DTI, Intel® SI triggers a fatal error by driving OKNOK[1:0] to 2'b01
- **Correctable errors** are corrected (by hardware) on the fly without any specific actions for the software. Correctable errors are always signaled by ALERT# which is controlled by the Intel® SI. Intel® SI tracks unusual occurrences of correctable errors and when unusual occurrences are detected, a fatal error is triggered by Intel® SI driving OKNOK[1:0] to 2'b01.

23.2.2 Error reporting from Intel® SI to system

Intel® SI reports errors to the system using the error pins ALERT_N and OKNOK[1:0] (refer to Figure 23-1 Signal Description). In addition, Intel® SI can send a message to host SW for user defined action; send a message over SPI initiator for a MCU or equivalent to monitor error conditions; or trigger an interrupt to the host processor.

23.2.3 Internal Processor error measures reporting to Intel® SI

For legacy reasons the processor supports a number of logic blocks for reporting errors and error information to the operating system or external pins.

To properly serve functional safety use cases, all diagnostic measures reported via legacy errors are internally routed to Intel® SI which drives OKNOK[1:0] or ALERT# accordingly. Some measures don’t use legacy error pins to report to Intel® SI. Intel provides STLs that have to be executed every DTI (Diagnostic Test Interval) to read error status registers and report errors to Intel® SI.

Below is the list of legacy error pins routed internally to Intel® SI:

- CATERR_N: For reporting compute die catastrophic error.
- ERR [0]: For reporting PCH die correctable errors.
- ERR [1]: For reporting PCH die non-fatal errors
- ERR [2]: For reporting PCH die fatal errors.

Table 23-2. Legacy Error Reporting Logic

Error Reporting Mechanism	Blocks that Use this to Report Errors
Machine Check Architecture (MCA)	Intel Atom® cores, BIU, PTR /CCF, Punit, Parts of Memory Subsystem
Advanced Error Reporting (AER) as defined in the PCI Express* Base Specification, Revision 3.0	PCIe Root Ports, Root Complex Integrated Endpoints, GBe Controllers, SATA
Global Error Logic	PCIe Root Ports (e.g., external PCIe ports), PCIe Endpoints behind the Virtual Root Port (e.g., Gbe controller), PCIE Root Complex Integrated Endpoints, Legacy PCI devices (e.g., eSPI)
Local Error Logic	Many IPs have local registers which also expose errors. Intel will supply STLs to monitor these registers and report errors to Intel® SI.
PROCHOT and THRMTRIP_N	THRMTRIP_N triggers on catastrophic over temperature conditions PROCHOT triggers when an internal temperature sensor detects a temperature equivalent to the rated junction temperature. These pins are routed to the system directly. Intel® SI also redundantly monitors the internal temperature sensors and drives OKNOK[1:0] to 2'b01 if any of the temperature sensors detects a temperature equivalent to THRMTRIP_N.

23.3 Integrated Pull-Ups and Pull - Downs

The input signals ISI_CHX_OKNOK [1:0] requires enabling of the integrated GPIO weak 20KΩ pull-down resistor. ISI_ALERT_N requires an external pull-up of 4.7KΩ.

Table 23-3. Integrated Pull-Ups and Pull-Downs

Signal	Resistor Type	Value
ISI_CHX_OK (ISI_CHX_OKNOK_0) ISI_CHX_NOK (ISI_CHX_OKNOK_1)	Integrated GPIO weak pull-down enabled	20KΩ
ISI_ALERT_N	external pull up	4.7KΩ

23.4 I/O Signal Planes and States

Table 23-4. I/O Signal Planes and States

Signal	Power Plane	During Reset	Immediately After Reset	S3/S4/S5
ISI_OK (ISI_OKNOK_0) ISI_NOK (ISI_OKNOK_1)	Primary	Driven to 2b'11	Driven to 2b'11	N/A
ISI_CHX_OK (ISI_CHX_OKNOK_0) ISI_CHX_NOK (ISI_CHX_OKNOK_1)	Primary	Pull-Down	Pull-Down	N/A
ISI_ALERT_N	Primary	Driven to 1b'1	Driven to 1b'1	N/A
ISI_CHX_RLY_SWITCH	Primary	Driven to 1b'0	Driven to 1b'0	N/A
ISI_CHX_PMIC_EN	Primary	Driven to 1b'0	Driven to 1b'0	N/A
ISI_I2CS_SCL	Primary	Undriven	Undriven	N/A
ISI_I2CS_SDA	Primary	Undriven	Undriven	N/A

Note: Intel® SI does not support low power states

23.5 Registers

Please refer to chapter 19 of the Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors for Internet of Things (IoT) Applications, Datasheet, Volume 2 (Book 2 of 3), Mule Creek Canyon (Document Number: 636722), for a description of the registers associated with subject of this chapter.

§ §

24 Functional Safety (FuSa)

24.1 Overview

Functional Safety (FuSa) is a requirement for Industrial applications which have to comply with IEC61508 and other Functional Safety standards. Such standards address possible hazards caused by malfunctioning behavior of electrical or electronic devices due to random HW failures and systematic failures. Random HW failures can occur unpredictably during the lifetime of a hardware component, while systematic failures are directly related to the process used for designing and developing a component and therefore can affect both hardware and software. Functional Safety standards provide a framework of requirements and techniques to determine and reach the appropriate safety integrity. IEC61508 uses the SIL (Safety Integrity Level) notation, ranging from SIL1 to SIL4, to specify the necessary requirements that a Functional Safety application and related components have to fulfill to avoid unreasonable risk.

Other standards like the ISO13849 introduce further notations and requirements for domain specific safety applications.

24.2 Processor and FuSa Safety Package

The processor is designed to meet requirements for SIL2 as per IEC61508, and for Cat3/Pl d as per ISO13849, using a single chip solution. It is also designed to meet requirements for SIL3 and Cat4/Ple using a two chip solution.

Note: FuSa is only available on SKU 11 and 12 on processor.

§ §

25 Primary to Sideband Bridge (P2SB)

25.1 Overview

The processor incorporates a wide variety of devices and functions. The registers within these devices are mainly accessed through PCI configuration space and IO/MMIO space. Some devices also have registers that are distributed within the PCH Private Configuration Space at individual endpoints (Target Port IDs) which are only accessible through the PCH Sideband Interface. Refer to Section 1.1.1 for more information on target.

These Compute Die and PCH Private Configuration Space Registers can be addressed via SBREG_BAR or through SBI Index Data pair programming.

Table 25-1. Private Configuration Space Register Target Port IDs (Sheet 1 of 2)

IOSF-SB Endpoint Name	Compute Die Local Port ID (Hex)	PCH Local Port ID (Hex)
iSCLK	-	0xAD
On Package Interface (OPI)	-	0xAC
ModPHY (Lanes 3:0)	-	0xAB
ModPHY (Lanes 7:4)	-	0xAA
ModPHY (Lanes 11:8)	-	0xA9
Universal Serial Bus (USB)	-	0xCA
Serial ATA (SATA)	-	0xD9
Converged Audio Voice Speech (cAVS)	-	0xD7
Enhanced Serial Peripheral Interface (eSPI) Serial Peripheral Interface (SPI) Flash Memory and TPM Only	-	0x72
Flex I/O Adapter (FIA)	-	0xCF
PCIe 0 Single VC	-	0x80
PCIe 1 Multi VC	-	0x81
PCIe 2 Multi VC	-	0x82
PCIe 3 Multi VC	-	0x83
SIO (LPSS)	-	0xCB
Secure Digital eXtended Capacity (SDXC)	-	0x53
embedded Multi Media Card (eMMC*)	-	0x51
Host System Management Bus (SMBus) Controller	-	0xC6
Primary to Sideband Bridge (P2SB)	-	0xC5

Table 25-1. Private Configuration Space Register Target Port IDs (Sheet 2 of 2)

IOSF-SB Endpoint Name	Compute Die Local Port ID (Hex)	PCH Local Port ID (Hex)
Real Time Clock (RTC)	-	0xC3
PCH Display	-	0xA0
Integrated Clock Controller (ICC)	-	0xDC
GbE HOST	-	0xDA

25.2 Integrated Error Handler

25.2.1 Overview

The Integrated Error Handler (IEH) is part of the Primary to Sideband Bridge (P2SB) device and processes & forwards PCI/PCIe errors from other devices within the PCH to the ITSS over the IOSF-SB bus. Once received, the ITSS may optionally initiate an NMI or SMI to the compute die.

Note: In most platforms, PCI/PCIe errors are handled directly by the Interrupt and Timer Sub-System (ITSS) or equivalent functional block.

Additionally, for FuSa SKUs only, the IEH generates three internal error signals for consumption by the Intel® SI (Safety Island).

1. Fatal Error
2. Non-Fatal Error
3. Correctable Error

25.2.2 Error Sources

The IEH can receive errors from 16 devices within the PCH. The error source is identified within the IEH using each device's Port ID, as described in the following table.

Table 25-2. Error Sources

Device	Port ID	Error Type
IEH	C2h	SERR#
xHCI	70h	SERR#
xDCI	71h	SERR#
eSPI, FSPI	72h	SERR#
PCIe 0	80h	SERR#
PCIe 1	81h	SERR#
PCIe 2	82h	SERR#
Intel® THB	6h	SERR#
SMBus[1]	C6h	SERR#

Table 25-2. Error Sources

SATA	D9h	SERR#
GbE Host	DAh	SERR#
Intel® PSE	BEh	PCIE_ERR
cAVS	D7h	SERR#
Intel® CSE	90h	PCIE_ERR
PMC	CCh	SERR#
OPI	88h	PCIE_ERR & SERR#

Note: 1. The SMBus controller can also generate a SERR# in response to a SERR# message received over the On Package Interface (OPI) from the compute die.

25.3 Registers

Please refer to chapter 3 of Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors for Internet of Things (IoT) Applications, Datasheet, Volume 2 (Book 2 of 3), Mule Creek Canyon (Document Number: 636722), for a description of the registers associated with subject of this chapter.

§ §

26 Legacy Interfaces

26.1 8254 Timers

The PCH contains two counters that have fixed uses. All registers and functions associated with these counters are in the Primary well. The 8254 unit is clocked by a 1.193 MHz periodic timer tick, which is functional only in S0 states. The 1.193MHz periodic timer tick is generated off the 38.4MHz xtal clock utilizing distribution masking of clock high time of the xtal clock.

Counter 0, System Timer

This counter functions as the system timer by controlling the state of IRQ0 and is typically programmed for Mode 3 operation (square wave output). The counter produces a square wave with a period equal to the product of the counter period (838 ns) and the initial count value. The counter loads the initial count value 1 counter period after software writes the count value to the counter I/O address. The counter initially asserts IRQ0 and decrements the count value by two each counter period. The counter negates IRQ0 when the count value reaches 0. It then reloads the initial count value and again decrements the initial count value by two each counter period. The counter then asserts IRQ0 when the count value reaches 0, reloads the initial count value, and repeats the cycle, alternately asserting and negating IRQ0.

Counter 2, Speaker Tone

This counter provides the speaker tone and is typically programmed for Mode 3 operation (square wave output). The counter provides a speaker frequency equal to the counter clock frequency (1.193 MHz) divided by the initial count value. The speaker must be enabled by a write to port 061h.

Note: 8254 clock gating should be disabled when using the SPKR signal, in order to ensure reliable operation, by the BIOS setting the ITSSPRC.CGE8254 register bit to 0h

26.1.1 Timer Programming

The counter/timers are programmed in the following steps:

1. Write a control word to select a counter.
2. Write an initial count for that counter.
3. Load the least and/or most significant bytes (as required by Control Word)of the 16-bit counter.
4. Repeat with other counters.

Only two conventions need to be observed when programming the counters. First, for each counter, the control word must be written before the initial count is written. Second, the initial count must follow the count format specified in the control word (least significant byte only, most significant byte only, or least significant byte, and then most significant byte).

A new initial count may be written to a counter at any time without affecting the counter's programmed mode. Counting is affected as described in the mode definitions. The new count must follow the programmed count format.

If a counter is programmed to read/write two-byte counts, the following precaution applies – a program must not transfer control between writing the first and second byte to another routine which also writes into that same counter. Otherwise, the counter will be loaded with an incorrect count.

The Timer Control Word (TCW) Register at I/O Port 43h controls the operation of all two counters. Several commands are available:

- **Control Word Command.** Specifies which counter to read or write, the operating mode, and the count format (binary or BCD).
- **Counter Latch Command.** Latches the current count so that it can be read by the system. The countdown process continues.
- **Read Back Command.** Reads the count value, programmed mode, the current state of the OUT pins, and the state of the Null Count Flag of the selected counter.

Table 26-1 lists the six operating modes for the interval counters.

Table 26-1. Counter Operating Modes

Mode	Function	Description
0	Termination at the end of count	Output is 0. When count goes to 0, output goes to 1 and stays at 1 until counter is reprogrammed.
1	Hardware retriggerable one-shot	Output is 0. When count goes to 0, output goes to 1 for one clock time.
2	Rate generator (divide by n counter)	Output is 1. Output goes to 0 for one clock time, then back to 1 and counter is reloaded.
3	Square wave output	Output is 1. Output goes to 0 when counter rolls over, and counter is reloaded. Output goes to 1 when counter rolls over, and counter is reloaded, and so on
4	Software triggered strobe	Output is 1. Output goes to 0 when count expires for one clock time.
5	Hardware triggered strobe	Output is 1. Output goes to 0 when count expires for one clock time.

26.1.2 Reading from the Interval Timer

It is often desirable to read the value of a counter without disturbing the count in progress. There are three methods for reading the counters—a simple read operation, counter Latch command, and the Read-Back command. Each is explained below.

With the simple read and counter latch command methods, the count must be read according to the programmed format; specifically, if the counter is programmed for two byte counts, two bytes must be read. The two bytes do not have to be read one right after the other. Read, write, or programming operations for other counters may be inserted between them.

26.1.2.1 Simple Read

The first method is to perform a simple read operation. The counter is selected through Port 40h (Counter 0) or 42h (Counter 2).

Note: Performing a direct read from the counter does not return a determinate value, because the counting process is asynchronous to read operations. However, in the case of Counter 2, the count can be stopped by writing to the GATE bit in Port 61h.

26.1.2.2 Counter Latch Command

The Counter Latch command, written to Port 43h, latches the count of a specific counter at the time the command is received. This command is used to ensure that the count read from the counter is accurate, particularly when reading a two-byte count. The count value is then read from each counter's Count register as was programmed by the Control register.

The count is held in the latch until it is read or the counter is reprogrammed. The count is then unlatched. This allows reading the contents of the counters on the fly without affecting counting in progress. Multiple Counter Latch Commands may be used to latch more than one counter. Counter Latch commands do not affect the programmed mode of the counter in any way.

If a Counter is latched and then, some time later, latched again before the count is read, the second Counter Latch command is ignored. The count read is the count at the time the first Counter Latch command was issued.

26.1.2.3 Read Back Command

The Read Back command, written to Port 43h, latches the count value, programmed mode, and current states of the OUT pin and Null Count flag of the selected counter or counters. The value of the counter and its status may then be read by I/O access to the counter address.

The Read Back command may be used to latch multiple counter outputs at one time. This single command is functionally equivalent to several counter latch commands, one for each counter latched. Each counter's latched count is held until it is read or reprogrammed. Once read, a counter is unlatched. The other counters remain latched until they are read. If multiple count Read Back commands are issued to the same counter without reading the count, all but the first are ignored.

The Read Back command may additionally be used to latch status information of selected counters. The status of a counter is accessed by a read from that counter's I/O port address. If multiple counter status latch operations are performed without reading the status, all but the first are ignored.

Both count and status of the selected counters may be latched simultaneously. This is functionally the same as issuing two consecutive, separate Read Back commands. If multiple count and/or status Read Back commands are issued to the same counters without any intervening reads, all but the first are ignored.

If both count and status of a counter are latched, the first read operation from that counter returns the latched status, regardless of which was latched first. The next one or two reads, depending on whether the counter is programmed for one or two type counts, returns the latched count. Subsequent reads return unlatched count.

26.1.3 Registers

Please refer to chapter 22 of the Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors for Internet of Things (IoT) Applications, Datasheet, Volume 2 (Book 2 of 3), Mule Creek Canyon (Document Number: 636722), for a description of the registers associated with subject of this chapter.

26.2 I/O APIC

The IO Advanced Programmable Interrupt Controller (APIC) is used to support line interrupts more flexibly than the 8259 PIC. Line interrupts are routed to it from multiple sources, including the P2SB, various PCH controllers and the compute die. These line based interrupts are then used to generate interrupt messages targeting the local APIC in the processor.

26.2.1 Feature Overview

- 120 interrupt lines — IRQ0-119
- Edge or level trigger mode per interrupt
- Active low or high polarity per interrupt
- Works with local APIC in processor via MSIs
- MSIs can target specific processor core
- Established APIC programming model
- Registers mapped to fixed I/O locations
- Uses Messages to CPU to indicate interrupt
- RTE accessed via indirect addressing scheme, with re-mappable address for Vt-d support.
- Interrupts issued as MSI, complete with vector, processor address and status information.
- Supports multiprocessor systems.

26.2.2 Functional Description

The I/O APIC contains indirectly accessed I/O APIC registers and normal memory mapped registers. There are three memory mapped registers:

• Index Register (IDX)

• Window Register (WDW)

• End Of Interrupt Register (EOI)

The Index register selects an indirect I/O APIC register (ID/VS/RTE[n]) to appear in the Window register.

The Window register is used to read or write the indirect register selected by the Indexregister.

The EOI register is written to by the Local APIC in the processor. The I/O APIC compares the lower eight bits written to the EOI register to the Vector set for each interrupt (RTE.VCT). All interrupts that match this vector will have their RTE.RIRR register cleared. All other EOI register bits are ignored.

26.2.3 Indirect I/O APIC Registers

These registers are selected with the IDX register, and read/wr register. Accessing these registers must be done as DW request behavior will result. Software should not attempt to write to reserved registers. Reserved registers may return non-zero values when read.

Note: There is one pair of redirection (RTE) registers per interrupt lin bit RTE register.

Note: Specified offsets should be placed in IDX, not added to IDX.

26.2.4 Bus: Device: Function for IOxAPIC

The VT-d architecture extension requires Interrupt Messages to go through the similar Address Remapping as any other memory requests. This is to allow domain isolation for interrupts such that a device assigned in one domain is not allowed to generate interrupts to another domain.

The Address Remapping for VT-d is based on the Bus:Device:Function field associated with the requests. Hence, it is required for the internal IOxAPIC to initiate the Interrupt Messages using a unique Bus:Device:Function.

PCH supports BIOS programmable unique Bus:Device:Function for the internal IOxAPIC. The Bus:Device:Function field does not change the IOxAPIC functionality in anyway, nor promoting IOxAPIC as a stand-alone PCI device. The field is only used by the IOxAPIC in the following:

- As the Requester ID when initiating Interrupt Messages to the CPU
- As the Completer ID when responding to the reads targeting the IOxAPIC's Memory-Mapped I/O registers

The register for the programmable IOxAPIC's Bus:Device:Function is resided under the Device 31:Function 0 configuration space. This is meeting the robustness and security requirement as Hypervisor owns the device's configuration space in the VT-d environment and thus will provide the appropriate protection against any possible attack.

26.2.5 Registers

Please refer to chapter 24 of the Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors for Internet of Things (IoT) Applications, Datasheet, Volume 2 (Book 2 of 3), Mule Creek Canyon (Document Number: 636722), for a description of the registers associated with subject of this chapter.

26.3 8259 Programmable Interrupt Controller (PIC)

26.3.1 Overview

The ISA compatible interrupt controller (PIC) incorporates the functionality of two 8259 interrupt controllers. The following table shows how the cores are connected:

8259	8259 Input	Connected Pin / Function
Initiator	0	Internal Timer / Counter 0 output or Multimedia Timer #0
	1	IRQ1 from configurable sources including eSPI, GPIO, internal ACPI devices.
	2	Target controller INTR output
	3	IRQ3 from configurable sources including PIRQx, eSPI,GPIO, internal ACPI devices.
	4	IRQ4 from configurable sources including PIRQx, eSPI,GPIO, internal ACPI devices.
	5	IRQ5 from configurable sources including PIRQx, eSPI,GPIO, internal ACPI devices.
	6	IRQ6 from configurable sources including PIRQx, eSPI,GPIO, internal ACPI devices.
	7	IRQ7 from configurable sources including PIRQx, eSPI,GPIO, internal ACPI devices.
Target	0	Inverted IRQ8# from internal RTC or HPET
	1	IRQ9 from configurable sources including PIRQx, eSPI,GPIO, internal ACPI devices, SCI, TCO.
	2	IRQ10 from configurable sources including PIRQx, eSPI,GPIO, internal ACPI devices, SCI, TCO.
	3	IRQ11 from configurable sources including PIRQx, eSPI,GPIO, internal ACPI devices, SCI, TCO or HPET #2.
	4	IRQ12 from configurable sources including PIRQx, eSPI, GPIO, internal ACPI devices, SCI, TCO or HPET #3.
	5	IRQ13 from configurable sources including PIRQx, eSPI, GPIO,internal ACPI devices.
	6	IRQ14 from configurable sources including PIRQx, eSPI,GPIO, internal ACPI devices.
	7	IRQ15 from configurable sources including PIRQx, eSPI,GPIO, internal ACPI devices.
Notes: Refer to Section 1.1.1 for more information on initiator and target		

The processor cascades the target controller onto the initiator controller through initiator controller interrupt input 2. This means there are only 15 possible interrupts for the processor PIC. Interrupts can be programmed individually to be edge or level, except for IRQ0, IRQ2 and IRQ8#.

Note: Active-low interrupt sources (such as a PIRQ#) are inverted inside the processor. In the following descriptions of the 8259s, the interrupt levels are in reference to the signals at the internal interface of the 8259s, after the required inversions have occurred. Therefore, the term “high” indicates “active,” which means “low” on an originating PIRQ#.

26.3.2 Functional Description

26.3.2.1 Interrupt Handling

26.3.2.1.1 Generating Interrupts

The PIC interrupt sequence involves three bits, from the IRR, ISR, and IMR, for each interrupt level. These bits are used to determine the interrupt vector returned, and status of any other pending interrupts. The fowwing table defines the IRR, ISR, and IMR.

Table 26-2. Interrupt Status Registers

Bit	Description
IRR	**Interrupt Request Register.** This bit is set on a low to high transition of the interrupt line in edge mode, and by an active high level in level mode.
ISR	**Interrupt Service Register.** This bit is set, and the corresponding IRR bit cleared, when an interrupt acknowledge cycle is seen, and the vector returned is for that interrupt.
IMR	**Interrupt Mask Register.** Determines whether an interrupt is masked. Masked interrupts will not generate INTR.

26.3.2.1.2 Acknowledging Interrupts

The processor generates an interrupt acknowledge cycle that is translated into a Interrupt Acknowledge Cycle to the processor. The PIC translates this command into two internal INTA# pulses expected by the 8259 controllers. The PIC uses the first internal INTA# pulse to freeze the state of the interrupts for priority resolution. On the second INTA# pulse, the initiator or target sends the interrupt vector to the processor with the acknowledged interrupt code. This code is based upon the ICW2.IVBA bits, combined with the ICW2.IRL bits representing the interrupt within that controller.

Note: References to ICWx and OCWx registers are relevant to both the initiator and target 8259 controllers.

26.3.2.1.3 Content of Interrupt Vector Byte

Initiator, Target Interrupt	Bits[7:3]	Bits[2:0]
IRQ7,15	ICW2.IVBA	111
IRQ6,14		110
IRQ5,13		101
IRQ4,12		100
IRQ3,11		011
IRQ2,10		010
IRQ1,9		001
IRQ0,8		000

26.3.2.1.4 Hardware/Software Interrupt Sequence

1. One or more of the Interrupt Request lines (IRQ) are raised high in edge mode, or seen high in level mode, setting the corresponding IRR bit.

2. The PIC sends INTR active to the processor if an asserted interrupt is not masked.

3. The processor acknowledges the INTR and responds with an interrupt acknowledge cycle.

4. Upon observing the special cycle, the processor converts it into the two cycles that the internal 8259 pair can respond to. Each cycle appears as an interrupt acknowledge pulse on the internal INTA# pin of the cascaded interrupt controllers.

5. Upon receiving the first internally generated INTA# pulse, the highest priority ISR bit is set and the corresponding IRR bit is reset. On the trailing edge of the first pulse, a target identification code is broadcast by the initiator to the target on a private, internal three bit wide bus. The target controller uses these bits to determine if it must respond with an interrupt vector during the second INTA# pulse.

6. Upon receiving the second internally generated INTA# pulse, the PIC returns the interrupt vector. If no interrupt request is present because the request was too short in duration, the PIC returns vector 7 from the initiator controller.

7. This completes the interrupt cycle. In AEOI mode the ISR bit is reset at the end ofthe second INTA# pulse. Otherwise, the ISR bit remains set until an appropriate EOI command is issued at the end of the interrupt subroutine.

26.3.2.2 Initialization Command Words (ICWx)

Before operation can begin, each 8259 must be initialized. In the processor, this is a four byte sequence. The four initialization command words are referred to by their acronyms: ICW1, ICW2, ICW3, and ICW4.

The base address for each 8259 initialization command word is a fixed location in the I/O memory space: 20h for the initiator controller, and A0h for the target controller.

26.3.2.3 ICWI

A write to the initiator or target controller base address with data bit 4 equal to 1 is interpreted as a write to ICW1. Upon sensing this write, the PIC expects three more byte writes to 21h for the initiator controller, or A1h for the target controller, to complete the ICW sequence.

A write to ICW1 starts the initialization sequence during which the following automatically occur:

1. Following initialization, an interrupt request (IRQ) input must make a low-to-high transition to generate an interrupt.

2. The Interrupt Mask Register is cleared.

3. IRQ7 input is assigned priority 7.

4. The target mode address is set to 7.

5. Special mask mode is cleared and Status Read is set to IRR.

26.3.2.4 ICW2

The second write in the sequence (ICW2) is programmed to provide bits [7:3] of the interrupt vector that will be released during an interrupt acknowledge. A different base is selected for each interrupt controller.

26.3.2.5 ICW3

The third write in the sequence (ICW3) has a different meaning for each controller.

- For the initiator controller, ICW3 is used to indicate which IRQ input line is used tocascade the target controller. Within the processor, IRQ2 is used. Therefore, MICW3.CCCis set to a 1, and the other bits are set to 0s.
- For the target controller, ICW3 is the target identification code used during an interrupt acknowledge cycle. On interrupt acknowledge cycles, the initiator controller broadcasts a code to the target controller if the cascaded interrupt won arbitration on the initiator controller. The target controller compares this identification code to the value stored in its ICW3, and if it matches, the target controller assumes responsibility for broadcasting the interrupt vector.

26.3.2.6 ICW4

The final write in the sequence (ICW4) must be programmed for both controllers. At the very least, ICW4.MM must be set to a 1 to indicate that the controllers are operating in an Intel Architecture-based system.

26.3.2.7 Operation Command Words (OCW)

These command words reprogram the Interrupt controller to operate in various interrupt modes.

• OCW1 masks and unmasks interrupt lines.

• OCW2 controls the rotation of interrupt priorities when in rotating priority mode, and controls the EOI function.

• OCW3 sets up ISR/IRR reads, enables/disables the special mask mode (SMM), and enables/disables polled interrupt mode.

26.3.2.8 Modes of Operation

26.3.2.8.1 Fully Nested Mode

In this mode, interrupt requests are ordered in priority from 0 through 7, with 0 being the highest. When an interrupt is acknowledged, the highest priority request is determined and its vector placed on the bus. Additionally, the ISR for the interrupt is set. This ISR bit remains set until: the processor issues an EOI command immediately before returning from the service routine; or if in AEOI mode, on the trailing edge of the second INTA#. While the ISR bit is set, all further interrupts of the same or lower priority are inhibited, while higher levels generate another interrupt.

Interrupt priorities can be changed in the rotating priority mode.

26.3.2.8.2 Special Fully-Nested Mode

This mode is used in the case of a system where cascading is used, and the priority has to be conserved within each target. In this case, the special fully-nested mode is programmed to the initiator controller. This mode is similar to the fully-nested mode with the following exceptions:

• When an interrupt request from a certain target is in service, this target is not locked out from the initiator's priority logic and further interrupt requests from higher priority interrupts within the target are recognized by the initiator and initiate interrupts to the processor. In the normal-nested mode, a target is masked out when its request is in service.

• When exiting the Interrupt Service routine, software has to check whether the interrupt serviced was the only one from that target. This is done by sending a NonSpecific EOI command to the target and then reading its ISR. If it is 0, a non-specific EOI can also be sent to the initiator.

26.3.2.8.3 Automatic Rotation Mode (Equal Priority Devices)

In some applications, there are a number of interrupting devices of equal priority. Automatic rotation mode provides for a sequential 8-way rotation. In this mode, a device receives the lowest priority after being serviced. In the worst case, a device requesting an interrupt has to wait until each of seven other devices are serviced at most once. There are two ways to accomplish automatic rotation using OCW2.REOI; the Rotation on Non-Specific EOI Command (OCW2.REOI=101b) and the rotate in automatic EOI mode which is set by (OCW2.REOI=100b).

26.3.2.8.4 Specific Rotation Mode (Specific Priority)

Software can change interrupt priorities by programming the bottom priority. For example, if IRQ5 is programmed as the bottom priority device, then IRQ6 is the highest priority device. The Set Priority Command is issued in OCW2 to accomplish this, where: OCW2.REOI=11xb, and OCW2.ILS is the binary priority level code of the bottom priority device. In this mode, internal status is updated by software control during OCW2. However, it is independent of the EOI command. Priority changes can be executed during an EOI command by using the Rotate on Specific EOI Command in OCW2 (OCW2.REOI=111b) and OCW2.ILS=IRQ level to receive bottom priority.

26.3.2.8.5 Poll Mode

Poll mode can be used to conserve space in the interrupt vector table. Multiple interrupts that can be serviced by one interrupt service routine do not need separate vectors if the service routine uses the poll command. Poll mode can also be used to expand the number of interrupts. The polling interrupt service routine can call the appropriate service routine, instead of providing the interrupt vectors in the vector table. In this mode, the INTR output is not used and the microprocessor internal Interrupt Enable flip-flop is reset, disabling its interrupt input. Service to devices is achieved by software using a Poll command. The Poll command is issued by setting OCW3.PMC. The PIC treats its next I/O read as an interrupt acknowledge, sets the appropriate ISR bit if there is a request, and reads the priority level. Interrupts are frozen from the OCW3 write to the I/O read. The byte returned during the I/O read contains a 1 in Bit 7 if there is an interrupt, and the binary code of the highest priority level in Bits 2:0.

26.3.2.8.6 Edge and Level Triggered Mode

In ISA systems this mode is programmed using ICW1.LTIM, which sets level or edge for the entire controller. In the processor, this bit is disabled and a register for edge and level triggered mode selection, per interrupt input, is included. This is the Edge/Level control

Registers ELCR1 and ELCR2. If an ELCR bit is 0, an interrupt request will be recognized by a low-to-high transition on the corresponding IRQ input. The IRQ input can remain high without generating another interrupt. If an ELCR bit is 1, an interrupt request will be recognized by a high level on the corresponding IRQ input and there is no need for an edge detection. The interrupt request must be removed before the EOI command is issued to prevent a second interrupt from occurring.

In both the edge and level triggered modes, the IRQ inputs must remain active until after the falling edge of the first internal INTA#. If the IRQ input goes inactive before this time, a default IRQ7 vector is returned.

26.3.2.8.7 End of Interrupt (EOI) Operations

An EOI can occur in one of two fashions: by a command word write issued to the PIC before returning from a service routine, the EOI command; or automatically when the ICW4.AEOI bit is set to 1.

26.3.2.8.8 Normal End of Interrupt

In normal EOI, software writes an EOI command before leaving the interrupt service routine to mark the interrupt as completed. There are two forms of EOI commands: Specific and Non-Specific. When a Non-Specific EOI command is issued, the PIC clears the highest ISR bit of those that are set to 1. Non-Specific EOI is the normal mode of operation of the PIC within the processor, as the interrupt being serviced currently is the interrupt entered with the interrupt acknowledge. When the PIC is operated in modes that preserve the fully nested structure, software can determine which ISR bit to clear by issuing a Specific EOI. An ISR bit that is masked is not cleared by a Non-Specific EOI if the PIC is in the special mask mode. An EOI command must be issued for both the initiator and target controller.

26.3.2.8.9 Automatic End of Interrupt Mode

In this mode, the PIC automatically performs a Non-Specific EOI operation at the trailing edge of the last interrupt acknowledge pulse. From a system standpoint, this mode should be used only when a nested multi-level interrupt structure is not required within a single PIC. The AEOI mode can only be used in the initiator controller and not the target controller.

Note: Both the initiator and target PICs have an AEOI bit: MICW4.AEOI and SICW4.AEOI respectively. Only the MICW4.AEOI bit should be set by software. The SICW4.AEOI bit should not be set by software.

26.3.2.9 Masking Interrupts

26.3.2.9.1 Masking on an Individual Interrupt Request

Each interrupt request can be masked individually by the Interrupt Mask Register (IMR). This register is programmed through OCW1. Each bit in the IMR masks one interrupt channel. Masking IRQ2 on the initiator controller masks all requests for service from the target controller.

26.3.2.9.2 Special Mask Mode

Some applications may require an interrupt service routine to dynamically alter the system priority structure during its execution under software control. For example, the routine may wish to inhibit lower priority requests for a portion of its execution but enable some of them for another portion.

The special mask mode enables all interrupts not masked by a bit set in the Mask register. Normally, when an interrupt service routine acknowledges an interrupt without issuing an EOI to clear the ISR bit, the interrupt controller inhibits all lower priority requests. In the special mask mode, any interrupts may be selectively enabled by loading the Mask Register with the appropriate pattern.

The special mask mode is set by OCW3.ESMM=1b & OCW3.SMM=1b, and clearedwhere OCW3.ESMM=1b & OCW3.SMM=0b.

26.3.3 Registers

Please refer to chapter 26 of the Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors for Internet of Things (IoT) Applications, Datasheet, Volume 2 (Book 2 of 3), Mule Creek Canyon (Document Number: 636722), for a description of the registers associated with subject of this chapter.

26.4 Real-Time Real Time Clock (RTC)

26.4.1 Functional Description

The Real Time Clock (RTC) module provides a battery backed-up date and time keeping device with two banks of static RAM with 128 bytes each, although the first bank has 114 bytes for general purpose usage. The real time clock has 256bytes of battery-backed RAM.

Three interrupt features are available: time of day alarm with once a second to once a month range, periodic rates of 122us – 500 ms, and end of update cycle notification. Seconds, minutes, hours, days, day of week, month, and year are counted. Daylight savings compensation is optional.

The hour is represented in twelve or twenty-four hour format, and data can be represented in BCD or binary format. The design is functionally compatible with the Motorola MS146818B. The time keeping comes from a 32.768-KHz oscillating source, which is divided to achieve an update every second. The lower 14 bytes on the lower RAM block has very specific functions. The first ten are for time and date information. The next four (0Ah to 0Dh) are registers, which configure and report RTC functions.

The time and calendar data should match the data mode (BCD or binary) and hour mode (12 or 24 hour) as selected in register B. Programmer must make sure that data stored in these locations is within the reasonable values ranges and represents a possible date and time. The exception to these ranges is to store a value of C0–FFh in the Alarm bytes to indicate a don't care situation. All Alarm conditions must match to trigger an Alarm Flag, which could trigger an Alarm Interrupt if enabled.

The SET bit in register B must be 1 while programming these locations to avoid clashes with an update cycle. Access to time and date information is done through the RAM locations. If a RAM read from the ten time and date bytes is attempted during an update cycle, the value read do not necessarily represent the true contents of those locations. Any RAM writes under the same conditions are ignored.

The leap year determination for adding a 29th day to February does not take into account the end-of-the-century exceptions. The logic simply assumes that all years divisible by 4 are leap years. According to the Royal Observatory Greenwich, years that are divisible by 100 are typically not leap years. In every fourth century (years divisible by 400, like 2000), the 100-year-exception is over-ridden and a leap-year occurs.

The year 2100 will be the first time in which the current RTC implementation would incorrectly calculate the leap-year.

26.4.2 Signal Description

Name	Type	Description
RTC_X1	I	**Crystal Input 1**: This signal is connected to the 32.768 kHz crystal (max 50K ohm ESR). If using an external oscillator, the RTC_X1 VIH must be within the range of 0.8V to 1.2V (1.2V max) and VIL must be within range of -0.2V to 0.2V (0.2V max).
RTC_X2	O	**Crystal Input 2**: This signal is connected to the 32.768 kHz crystal (max 50K ohm ESR). If using an external oscillator, RTC_X2 should be left floating.
RTC_TEST_N	I	**RTC Reset:** When asserted, this signal resets non-CSE register bits in the RTC well. ***Notes:*** 1. Unless CMOS is being cleared (only to be done in the G3 power state) with a jumper, the RTC_TEST_N input must always be high when all other RTC power planes are on. 2. In the case where the RTC battery is dead or missing on the platform, the RTC_TEST_N pin must rise before the DSW_PWROK pin. 3. The specifications provided for the use of an external oscillator has not been validated on hardware and are provided, as-is and without warranty, only as a suggestion to aid platform design decisions. Validation is the sole responsibility of the platform designer.

Name	Type	Description
RTC_RST_N	I	**Secondary RTC Reset :When asserted, this signal resets CSE register bits in the RTC well.** 1. The RTC_RST_N input must always be high when all other RTC power planes are on. 2. In the case where the RTC battery is dead or missing on the platform, the RTC_RST_N pin must rise before the DSW_PWROK pin. 3. RTC_RST_N and RTC_TEST_N should not be shorted together.
PMC_SUSCLK	O	**Suspend Mode Clock**: Output of the RTC circuit (32.768 kHz) that can be used by the platform board. The output has a duty cycle as low as 30% or as high as 70%.
PMC_RSMRST_N	I	**Resume Well Reset**: This signal is used for resetting the resume power plane logic. This signal must be asserted for at least 10ms after the suspend power wells are valid. When de-asserted, this signal is an indication that the suspend power wells are stable.

26.4.3 Update Cycles

An update cycle occurs once a second, if the RTC_SET bit of register B is not asserted and the divide chain is properly configured. During this procedure, the stored time and date are incremented, overflow is checked, a matching alarm condition is checked, and the time and date are rewritten to the RAM locations.

The update cycle will start at least 488 µs after the UIP bit of register A is asserted, and the entire cycle does not take more than 1984 µs to complete. The time and date RAM locations (0–9) are disconnected from the external bus during this time.

To avoid update and data corruption conditions, external RAM access to these locations can safely occur upon the detection of either of two conditions. When a updated-ended interrupt is detected, almost 999 ms is available to read and write the valid time and date data. If the UIP bit of Register A is detected to be low, there is at least 488 µs before the update cycle begins.

Note: The overflow conditions for leap years and daylight savings adjustments are based on more than one date or time item. To ensure proper operation when adjusting the time, the new time and data values should be set at least two seconds before one of these conditions (leap year, daylight savings time adjustments) occurs.

26.4.4 Interrupts

The real-time clock interrupt is internally routed within the PCH to both the I/O APIC and the 8259. It is mapped to interrupt vector 8. This interrupt does not leave the PCH prior to connection to the interrupt controller, nor is it shared with any other interrupt. The High Performance Event Timers (HPET) can also be mapped to IRQ8#; in this case, the RTC interrupt is blocked.

26.4.5 Lockable Ranges

The RTC battery-backed RAM supports two 8-byte ranges that can be locked via the PCI config space. If the locking bits are set, the corresponding range in the RAM will not be readable or writable. A write cycle to those locations will have no effect. A read cycle to those locations will not return the location’s actual value (resultant value is undefined).

Once a range is locked, the range can be unlocked only by a hard reset, which will invoke the BIOS and allow it to relock the RAM range.

26.4.6 Century Rollover

The hardware detects the case when the year rolls over from 99 to 00 (e.g., a rollover from December 31, 1999, 11:59:59 p.m. to 12:00:00 a.m. on January 1st, 2000). Upon detecting the rollover, the PCH sets the NEWCENTURY_STS bit. If the system is in an S0 state, this causes an SMI#. The SMI# handler can update registers in the RTC RAM that are associated with the century value.

26.4.7 Clearing Battery - Backed RTC CMOS RAM

Clearing CMOS RAM in an processor-based platform can be done by using a jumper on RTC_TEST_N or a GPI. Implementations should not attempt to clear CMOS by using a jumper to pull VCC_RTC_3P3 low.

26.4.8 Using RTC_TEST_N to Clear the RTC CMOS RAM

A jumper on RTC_TEST_N can be used to clear CMOS values. When RTC_TEST_N is low, the GEN_PMCON1.RPS register bit will be set. BIOS can monitor the state of this bit, and manually clear the RTC CMOS array once the system is booted. The normal position will cause RTC_TEST_N to be pulled up through a weak pull-up resistor. This RTC_TEST_N jumper technique allows the jumper to be moved and then replaced—all while the system is powered off. Then, once booted, the GEN_PMCON1.RPS bit can be detected in the set state.

26.4.9 Using a GPI to clear CMOS

A jumper on a GPI can also be used to clear CMOS values. BIOS should detect the setting of this GPI on system boot-up, and manually clear the CMOS array.

Note: The GPI strap technique to clear CMOS requires multiple steps to implement. The system is booted with the jumper in new position, then powered back down. The jumper is replaced back to the normal position, then the system is rebooted again.

Warning: Do not implement a jumper on VCC_RTC_3P3 to clear CMOS.

26.4.10 Clearing Battery Backed RTC Registers

Clearing battery backed RTC registers the platform can be done by using a jumper on RTC_TEST_N. Implementations should not attempt to clear the registers by using a jumper to pull VCC_RTC_3P3 low. A jumper on RTC_TEST_N pulled to ground can be used to reset the state of those battery backed RTC register configuration bits that reside in the RTC power well to their default state. The register in the Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors for

Internet of Things (IoT) Applications, Datasheet, Volume 2 (Book 2 of 3), Mule Creek Canyon, (Document Number: 636722) is reset by RTC_TEST_N is BUC (Backed Up Control). The register fields in the Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors for Internet of Things (IoT) Applications, Datasheet, Volume 2 (Book 2 of 3), Mule Creek Canyon (Document Number: 636722) are reset by RTC_TEST_N are Register_B.AIE, Register_C.AF, TSTS1.NEWCENTURY_STS and TSTS2.INTRD_DET.

A jumper on RTC_RST_N pulled to ground can be used to reset the state of those battery backed CSE register configuration bits that reside in the RTC power well to their default state. It is expected in normal operation, however, that RTC_RST_N must only be asserted when the processor is in a G5 state (All rails removed including VCC_RTC_3P3).

26.4.11 External RTC Circuitry

The PCH implements an internal oscillator circuit that is sensitive to voltage changes in VCC_RTC_3P3.

Note: Capacitors used in the RC delay circuits for RTC_RST_N & RTC_TEST_N should be evaluated with regards to aging, voltage and temperature characteristics to ensure reliable operation in the intended operating environment.

Table 26-3. RTC Crystal Requirements

Parameter	Specification
Frequency	32.768 KHz
Typical Tolerance	20 ppm or better
ESR	≤ 50 KΩ

Table 26-4. External Crystal Oscillator Requirement

Parameter	Specification
Frequency	32.768 KHz
Typical Tolerance	20 ppm or better
Voltage Swing	0 to 1.0Vp-p (±5%)

26.4.12 Registers

Please refer to chapter 27 of the Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors for Internet of Things (IoT) Applications, Datasheet, Volume 2 (Book 2 of 3), Mule Creek Canyon (Document Number: 636722), for a description of the registers associated with subject of this chapter.

26.5 System Management

26.5.1 Signal Description

Name	Type	Description
SML_DATA0	I/O	**System Management Link 0 Data:** SMBus link to USB Type-C Power Delivery (PD) Controller. External Pull-up resistor required.
SML_CLK0	I/O	**System Management Link 0 Clock:** SMBus link to USB Type-C Power Delivery (PD) Controller. External Pull-up resistor required.
SML_ALERT0_N	I/O	**System Management Link 0 Alert:** This signal is used to wake the system or generate SMI#. External Pull-up resistor required. **Note**: Do not use for PD controller, use PMC_ALERT_N for USB-C PD controller interrupt request instead.
INTRUDER_N	I	Detect input from a switch if the system cover is removed. Can generate a TCO SMI#.

Note: SML_DATA0 and SML_CLK0 should be configured to 3.3V using the multiplexed GPIO's Individual Voltage Select soft strap.

26.5.2 Feature Overview

The processor provides various functions to make a system easier to manage and to lower the Total Cost of Ownership (TCO) of the system. Features and functions can be augmented using external A/D converters and GPIOs, as well as an external micro controller.

The following features and functions are supported by the PCH:

- Various Error detection (such as ECC Errors) indicated by host controller:
  - — Can generate SMI#, SCI, SERR, SMI, or TCO interrupt

26.5.3 Theory of Operation

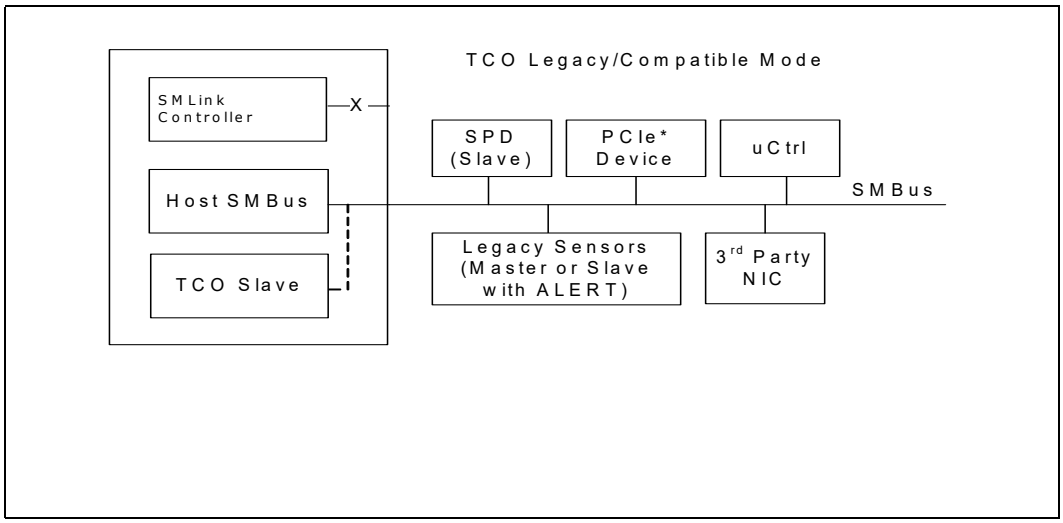
The System Management functions are designed to allow the system to diagnose failing subsystems. The intent of this logic is that some of the system management functionality can be provided without the aid of an external microcontroller.

26.5.4 TCO Modes

26.5.4.1 TCO Compatible Mode

In TCO Legacy/Compatible mode, only the host SMBus is used. The TCO target is connected to the host SMBus internally by default.

Figure 26-1. TCO Compatible Mode SMBus Configuration



In TCO Legacy/Compatible mode the PCH can function directly with an external LAN controller or equivalent external LAN controller to report messages to a network management console without the aid of the system processor. This is crucial in cases where the processor is malfunctioning or cannot function due to being in a low-power state. Table 26-5 includes a list of events that will report messages to the network management console.

Table 26-5. Event Transitions that Cause Messages

Event	Assertion?	Deassertion?	Comments
Watchdog Timer Expired	Yes	NA	"Hung S0" state entered
SMB_ALERT_N	Yes	Yes	Must be in "Hung S0" state
PMC_BATLOW_N	Yes	Yes	Must be in "Hung S0" state
CPU_PWR_FLR	Yes	No	"Hung S0" state entered

26.5.5 Handling an Intruder

The processor has an input signal, INTRUDER_N, that can be attached to a switch that is activated by the system's case being open. This input has a 2 RTC clock debounce. If INTRUDER_N goes active (after the debouncer), this will set the INTRD_DET bit in the TCO2_STS Register (TSTS2). Note that INTRUDER_N can go active in any power state.

The INTRD_SEL bits in the TCO2_CNT Register (TCTL2) can enable the Intel PCH to cause an SMI# . The BIOS or interrupt handler can then cause a transition to the S5 state by writing to the PM1_CNT.SLP_EN bitS.

26.5.6 SMLink Support for USB Type-C Power Delivery Controller

The processor has a SMLink interface, comprised of the SML_DATA0 & CLK0 signals, that can be attached to a USB Type-C PD (Power Delivery) controller, which acts as the Type-C Port Manager for the system. The SMLink interface is controlled by the processor's PMC (Power Management Controller) and has a interrupt based interaction with the external PD controller.

The PD controller's USB Type-C interrupt request is serviced by PMC through the following sequence:

• The PD controller's interrupt request is presented to PMC through PMC_ALERT_N pin assertion.

• The PMC sets the interrupt acknowledge bit in the PD controller.

• The PMC acquires the USB Type-C connection attributes from the PD controller.

• The PMC set the IRQ acknowledge register bit in the PD controller (if required).

• The PMC processes the USB Type-C connection request.

The following PD controller behaviors are expected:

• PD controller is expected to keep the PMC_ALERT_N signal de-asserted if it is not ready.

• PD controller is expected to assert the PMC_ALERT_N signal when the content of the Data Status register changed, and de-assert the PMC_ALERT_N pin when the "I2C_INT_ACK" bit in the Data Control register is set.

• PD controller is allowed to change the Data Status register content regardless of the PMC_ALERT_N state.

• PD controller is expected to clear the "HPD_IRQ" bit in the Data Status register when the "HPD_IRQ_ACK" bit in the Data Control register is set.

Notes:

1. Only one PD controller is supported by the PMC.

2. When PD controller support is enabled by soft strap, the PMC will automatically access the SMLink interface during early boot despite the corresponding package balls defaulting to a GP-In. If it does not read logic high on SML_DATA0, the processor fails to boot.

26.5.7 TCO Watchdog Timer

The TCO Watchdog Timer is a countdown timer used by software to detect software hangs. First TCO Watchdog Timer timeout can generate SMI# after a programmable time:

• The programmable 10-bit timer is decremented approximately every 0.6 seconds and allows timeouts ranging from 1.2 seconds to 613.8 seconds. The timeout value is programmable in the TCO_TMR Register (TTMR).

• OS software can periodically reload the processor timer using the TCO_RLD Register (TRLD). The first TCO Watchdog Timer timeout causes an SMI# allowing SMM based recovery from OS lockup.

Second hard-coded TCO Watchdog Timer timeout to generate a system reboot by the processor asserting the active low PMC_PLTRST_N signal:

• This second timer is used only after the first TCO Watchdog Timer timeout occurs.

• The SMI# handler must reload the first timer within 2.4 seconds after it times out to prevent the second timer from causing a system reboot. A timeout here assumes to be from a CPU or hardware error and reason to generate a system reset and reboot.

• Option to prevent the system from rebooting after the second timeout via the "No Reboot" Hard Pin Strap. See Chapter 28 for further details.

26.5.8 Registers

Please refer to Chapter 28 of EDS Volume 2 (Book 2 of 4) for a description of the registers associated with subject of this chapter.

26.6 High Precision Event Timer (HPET)

26.6.1 Overview

This function provides a set of timers that can be used by the operating system. The timers are defined such that the operating system may assign specific timers to be used directly by specific applications. Each timer can be configured to cause a separate interrupt.

The PCH provides eight timers. The timers are implemented as a single counter with a set of comparators. Each timer has its own comparator and value register. The counter increases monotonically. Each individual timer can generate an interrupt when the value in its value register matches the value in the main counter.

Timer 0 supports periodic interrupts.

The registers associated with these timers are mapped to a range in memory space (much like the I/O APIC). However, it is not implemented as a standard PCI function. The BIOS reports to the operating system the location of the register space using ACPI. The hardware can support an assignable decode space; however, BIOS sets this space prior to handing it over to the operating system. It is not expected that the operating system will move the location of these timers once it is set by BIOS.

26.6.1.1 Timer Accuracy

The timers are accurate over any 1-ms period to within 0.05% of the time specified in the timer resolution fields.

Within any 100-microsecond period, the timer reports a time that is up to two ticks too early or too late. Each tick is less than or equal to 100 ns; thus, this represents an error of less than 0.2%.

The timer is monotonic. It does not return the same value on two consecutive reads (unless the counter has rolled over and reached the same value).

The main counter uses the PCH's 38.4 XTAL as its clock. The accuracy of the main counter is as accurate as the crystal that is used in the system.The XTAL clock frequency is determined by the pin strap that is sampled on PMC_RSMRST_N.

26.6.1.2 Timer Off-load

The PCH supports a timer off-load feature that allows the HPET timers to remain operational during very low power S0 operational modes when the 38.4MHz clock is disabled. The clock source during this off-load is the Real Time Clock's 32.768kHz clock. This clock is calibrated against the 38.4MHz clock during boot time to an accuracy that ensures the error introduced by this off-load is less than 10 ppb (.000001%).

When the 38.4MHz clock is active, the 64-bit counter will increment by one each cycle of the 38.4MHz clock when enabled. When the 38.4MHz clock is disabled, the timer is maintained using the RTC clock. The long-term (> 1 msec) frequency drift allowed by the HPET specification is 500 ppm. The off-load mechanism ensures that it contributes < 1ppm to this, which will allow this specification to be easily met given the clock crystal accuracies required for other reasons.

Timer off-load is prevented when there are HPET comparators active.

The HPET timer in the PCH runs typically on the 38.4 MHz crystal clock and is off-loaded to the 32 kHz clock once the processor enters C10. This is the state where there are no C10 wake events pending and when the off-load calibrator is not running. HPET timer re-uses this 28-bit calibration value calculated by PMC when counting on the 32-kHz clock. During C10 entry, PMC sends an indication to HPET to off-load and keeps the indication active as long as the processor is in C10 on the 32 kHz clock. The HPET counter will be off-loaded to the 32 kHz clock domain to allow the 38.4 MHz clock to shut down when it has no active comparators.

26.6.1.2.1 Theory of operation

The Off-loadable Timer Block consists of a 64b fast clock counter and an 82b slow clock counter. During fast clock mode the counter increments by one on every rising edge of the fast clock. During slow clock mode, the 82-bit slow clock counter will increment by the value provided by the Off-load Calibrator.

The Off-loadable Timer will accept an input to tell it when to switch to the slow RTC clock mode and provide an indication of when it is using the slow clock mode. The switch will only take place on the slow clock rising edge, so for the 32 kHz RTC clock the maximum delay is around 30 microseconds to switch to or from slow clock mode. Both of these flags will be in the fast clock domain.

When transitioning from fast clock to slow clock, the fast clock value will be loaded into the upper 64b of the 82b counter, with the 18 LSBs set to zero. The actual transition through happens in two stages to avoid metastability. There is a fast clock sampling of the slow clock through a double flop synchronizer. Following a request to transition to the slow clock, the edge of the slow clock is detected and this causes the fast clock value to park. At this point the fast clock can be gated. On the next rising edge of the slow clock, the parked fast clock value (in the upper 64b of an 82b value) is added to the value from the Off-load Calibrator. On subsequent edges while in slow clock mode the slow clock counter increments its count by the value from the Off-load Calibrator.

When transitioning from slow clock to fast clock, the fast clock waits until it samples a rising edge of the slow clock through its synchronizer and then loads the upper 64b of the slow clock value as the fast count value. It then de-asserts the indication that slow clock mode is active. The 32 kHz clock counter no longer counts. The 64-bit MSB will be over-written when the 32 kHz counter is reloaded once conditions are met to enable the 32 kHz HPET counter but the 18-bit LSB is retained and it is not cleared out during the next reload cycle to avoid losing the fractional part of the counter.

After initiating a transition from fast clock to slow clock and parking the fast counter value, the fast counter no longer tracks. This means if a transition back to fast clock is requested before the entry into off-load slow clock mode completes, the Off-loadable Timer must wait until the next slow clock edge to restart. This case effectively performs the fast clock to slow clock and back to fast clock on the same slow clock edge.

26.6.1.3 Interrupt Mapping

The interrupts associated with the various timers have several interrupt mapping options. When reprogramming the HPET interrupt routing scheme (LEG_RT_CNF bit in the General Config Register), a spurious interrupt may occur. This is because the other source of the interrupt (8254 timer) may be asserted. Software should mask interrupts prior to clearing the LEG_RT_CNF bit.

26.6.1.3.1 Mapping Option #1 (Legacy Replacement Option)

In this case, the Legacy Replacement Rout bit (LEG_RT_CNF) is set. This forces the mapping found in Table 26-6.

Table 26-6. Legacy Replacement Routing

Timer	8259 Mapping	APIC Mapping	Comment
0	IRQ0	IRQ2	In this case, the 8254 timer will not cause any interrupts
1	IRQ8	IRQ8	In this case, the RTC will not cause any interrupts.
2 and 3	Per IRQ Routing Field.	Per IRQ Routing Field	
4, 5, 6, 7	not available	not available	
Note: The Legacy Option does not preclude delivery of IRQ0/IRQ8 using processor interrupts messages.			

26.6.1.3.2 Mapping Option #2 (Standard Option)

In this case, the Legacy Replacement Rout bit (LEG_RT_CNF) is 0. Each timer has its own routing control. The interrupts can be routed to various interrupts in the 8259 or I/O APIC. A capabilities field indicates which interrupts are valid options for routing. If a timer is set for edge-triggered mode, the timers should not be shared with any legacy interrupts.

For the PCH, the only supported interrupt values are as follows:

Timer 0 and 1: IRQ20, 21, 22, and 23 (I/O APIC only).

Timer 2: IRQ11 (8259 or I/O APIC) and IRQ20, 21, 22, and 23 (I/O APIC only).

Timer 3: IRQ12 (8259 or I/O APIC) and IRQ 20, 21, 22, and 23 (I/O APIC only).

Note: Interrupts from Timer 4, 5, 6, 7 can only be delivered through direct FSB interrupt messages.

Note: System architecture changes since the HPET specification 1.0 was released have made some of the terminology used obsolete. In particular the reference to a Front Side Bus (FSB) has no relevance to current platforms, as this interface is no longer in use. For consistency with the HPET specification though, the FSB and specifically the FSB Interrupt Delivery terminology has been maintained. Where the specification refers to FSB, this should be read as ‘processor message interface’; independent of the physical attach mechanism.

26.6.1.3.3 Mapping Option #3 (Processor Message Option)

In this case, the interrupts are mapped directly to processor messages without going to the 8259 or I/O (x) APIC. To use this mode, the interrupt must be configured to edge-triggered mode. The Tn_PROCMSG_EN_CNF bit must be set to enable this mode.

When the interrupt is delivered to the processor, the message is delivered to the address indicated in the Tn_PROCMSG_INT_ADDR field. The data value for the write cycle is specified in the Tn_PROCMSG_INT_VAL field.

Note: The FSB interrupt deliver option has HIGHER priority and is mutually exclusive to the standard interrupt delivery option. Thus, if the TIMERn_FSB_EN_CNF bit is set, the interrupts will be delivered via the FSB, rather than via the APIC or 8259.

The FSB interrupt delivery can be used even when the legacy mapping is used.

For the Intel PCH HPET implementation, the direct FSB interrupt delivery mode is supported, besides via 8259 or I/O APIC.

26.6.1.4 Periodic Versus Non-Periodic Modes

26.6.1.4.1 Non-Periodic Mode

This mode can be thought of as creating a one-shot.

When a timer is set up for non-periodic mode, it will generate an interrupt when the value in the main counter matches the value in the timer's comparator register. Another interrupt will be generated when the main counter matches the value in the timer's comparator register after a wrap around.

During run-time, the value in the timer's comparator value register will not be changed by the hardware. Software can of course change the value.

The Timer 0 Comparator Value register cannot be programmed reliably by a single 64-bit write in a 32-bit environment **except** if only the periodic rate is being changed during run-time. If the actual Timer 0 Comparator Value needs to be reinitialized, then the following software solution will always work regardless of the environment:

- Set TIMER0_VAL_SET_CNF bit
- Set the lower 32 bits of the Timer0 Comparator Value register
- Set TIMER0_VAL_SET_CNF bit
- Set the upper 32 bits of the Timer0 Comparator Value register

Timer 0 is configurable to 32- (default) or 64-bit mode, whereas Timers 1:7 only support 32-bit mode.

Warning: Software must be careful when programming the comparator registers. If the value written to the register is not sufficiently far in the future, then the counter may pass the value before it reaches the register and the interrupt will be missed. The BIOS should pass a data structure to the operating system to indicate that the operating system should not attempt to program the periodic timer to a rate faster than 5 microseconds.

All of the timers support non-periodic mode.

Refer to Section 2.3.9.2.1 of the *IA-PC HPET Specification* for more details of this mode.

26.6.1.4.2 Periodic Mode

When a timer is set up for periodic mode, the software writes a value in the timer's comparator value register. When the main counter value matches the value in the timer's comparator value register, an interrupt can be generated. The hardware will then automatically increase the value in the comparator value register by the last value written to that register.

To make the periodic mode work properly, the main counter is typically written with a value of 0 so that the first interrupt occurs at the right point for the comparator. If the main counter is not set to 0, interrupts may not occur as expected.

During run-time, the value in the timer's comparator value register can be read by software to find out when the next periodic interrupt will be generated (not the rate at which it generates interrupts). Software is expected to remember the last value written to the comparator's value register (the rate at which interrupts are generated).

If software wants to change the periodic rate, it should write a new value to the comparator value register. At the point when the timer's comparator indicates a match, this new value will be added to derive the next matching point.

If the software resets the main counter, the value in the comparator's value register needs to reset as well. This can be done by setting the **TIMERn_VAL_SET_CNF** bit. Again, to avoid race conditions, this should be done with the main counter halted. The following usage model is expected:

1. Software clears the ENABLE_CNF bit to prevent any interrupts
2. Software Clears the main counter by writing a value of 00h to it.
3. Software sets the TIMER0_VAL_SET_CNF bit.
4. Software writes the new value in the TIMER0_COMPARATOR_VAL register

Software sets the ENABLE_CNF bit to enable interrupts.

Note: As the timer period approaches zero, the interrupts associated with the periodic timer may not get completely serviced before the next timer match occurs. Interrupts may get lost and/or system performance may be degraded in this case.

Each timer is NOT required to support the periodic mode of operation. A capabilities bit indicates if the particular timer supports periodic mode. he reason for this is that supporting the periodic mode adds a significant amount of gates.

For the Intel PCH, only timer 0 will support the periodic mode. This saves a substantial number of gates.

26.6.1.5 Enabling the Timers

The BIOS or operating system PnP code should route the interrupts. This includes the Legacy Rout bit, Interrupt Rout bit (for each timer), and interrupt type (to select the edge or level type for each timer).

The Device Driver code should do the following for an available timer:

1. Set the Overall Enable bit (Offset 10h, bit 0).
2. Set the timer type field (selects one-shot or periodic).
3. Set the interrupt enable.
4. Set the comparator value.

26.6.1.6 Interrupt Levels

Interrupts directed to the internal 8259s are active high.

If the interrupts are mapped to the 8259 or I/O APIC and set for level-triggered mode, they can be shared with legacy interrupts. They may be shared although it is unlikely for the operating system to attempt to do this.

If more than one timer is configured to share the same IRQ (using the TIMERn_INT_ROUT_CNF fields), then the software must configure the timers to level-triggered mode. Edge-triggered interrupts cannot be shared.

26.6.1.7 Handling Interrupts

Section 2.4.6 of the IA-PC HPET Specification describes handling interrupts.

26.6.1.8 Issues Related to 64-Bit Timers with 32-Bit Processors

Section 2.4.7 of the IA-PC HPET Specification describes issues related to 64-bit timers with 32-bit processors.

26.6.2 References

Specification	Location
IA-PC HPET (High Precision Event Timers) Specification, Revision 1.0a	http://www.intel.com/content/dam/www/public/us/en/documents/technical-specifications/software-developers-hpet-spec-1-0a.pdf

26.6.3 Registers

Please refer to Chapter 25 of EDS Volume 2 (Book 2 of 4) for a description of the registers associated with subject of this chapter.

26.7 Processor Interface

26.7.1 Functional Description

This section provides additional behavioral descriptions of the functionality that interfaces between the PCH and the Compute Die.

26.7.1.1 INIT# Functionality

INIT# will be active (asserted) internally for 16 PCI clocks based on any one of several events and is handled as edge triggered. When any of these events occur, INIT# will be detected as active and an “assert” message will be sent for INIT#.

Table 26-7. Cause of INIT#

Cause of INIT# Going Active	Comment
Shutdown special cycle message from the CPU.	INIT# assertion based on value of Shutdown Policy Select register (GIC.SDPS).
PORT92 register write, where INIT_NOW (bit 0) transitions from a 0 to a 1.	

RST_CNT register write, where SYS_RST (bit 1) is a 0 and RST_CPU (bit 2) transitions from 0 to 1.	

For systems that enable the Secure Boot, INIT# must be prevented from being issued upstream to the compute die to cause a CPU-Only Reset. Allowing INIT# to trigger CPU-Only Reset for Secure Boot enabled systems exposes security vulnerabilities such as cache reset attacks.

26.7.1.2 NMI Functionality

Non-Maskable Interrupts (NMIs) can be generated by several sources:

Table 26-8. Cause of NMI

Cause of NMI	Comment
SERR# assertion	Transmitted by IEH (Integrated Error Handler)
NMI_NOW bit assertion (TCO1_CNT register in SMBus controller)	Transmitted by SMBus Controller.
GPIO when configured as General Purpose Input (GPI) and routed as NMI (by GPIROUTNMI register field).	Enabled by GPI_NMI_EN_GPPC_x_0 registers per GPIO
IBECC error is reported by memory controller	

Note: In contrast to most other platforms, SERR# assertion directly from most devices within the PCH will not be the cause of an NMI. This is because PCI/PCIe errors are handled by the IEH (Integrated Error Handler) instead. The IEH will transmit a SERR# assertion instead.

26.7.2 Registers

Please refer to chapter 23 of the Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors for Internet of Things (IoT) Applications, Datasheet, Volume 2 (Book 2 of 3), Mule Creek Canyon (Document Number: 636722), for a description of the registers associated with subject of this chapter.

§ §

27 Pin Strap

The following signals are used for static configuration. They are sampled at the rising edge of PMC_DSW_PWROK, PMC_RSMRST_N, or PMC_PCH_PWROK to select configuration and then revert later to their normal usage. To invoke the associated mode, the signal should meet both set up and hold time of 1us, with respect to the rising edge of the sampling signal.

Table 27-1. Pin Straps (Sheet 1 of 4)

Signal	Usage	When Sampled	Comment
GP_B14/SPKR/ PMC_TGPIO1/ SIO_SPI0_CS1_ N/ PSE_SPI2_CS1_ N	Top Swap Override	Rising edge of PMC_P-CH_PWROK	The strap has a 20 kohm ± 30% internal pull-down. 0 = **Disable** "Top Swap" mode. (Default) 1 = **Enable** "Top Swap" mode. This inverts an address on access to SPI, so the processor believes it fetches the alternate boot block instead of the original boot-block. PCH will invert the appropriate address lines (A16, A17, or A18) as selected in Top Swap Block size soft strap ***Notes:*** 1. The internal pull-down is disabled after PMC_PCH_PWROK is high. 2. Software will not be able to clear the Top Swap bit until the system is rebooted. 3. The status of this strap is readable using the Top Swap bit (Bus0, Device31, Function0, offset DCh, bit4). 4. This signal is in the primary well.
GP_B18/ SIO_SPI0_MOSI/ PSE_SPI2_MOSI	No Reboot	Rising edge of PMC_P-CH_PWROK	The strap has a 20 kohm ± 30% internal pull-down. 0 = **Disable** "No Reboot" mode. (Default) 1 = **Enable** "No Reboot" mode (PCH will disable the TCO Timer system reboot feature). This function is useful when running ITP. ***Notes:*** 1. The internal pull-down is disabled after PMC_PCH_PWROK is high. 2. This signal is in the primary well.
GP_C02/ PSE_PWM00/ SMB_ALERT_N/ PSE_TGPIO29	Reserved	Rising edge of PMC_RSMRST _N	This strap has a 20 kohm ± 30% internal pull-down. 0 = Default 1 = Reserved ***Notes:*** 1. The internal pull-down is disabled after PMC_RSMRST_N de-asserts. 2. This signal is in the primary well.
CFG_00	Reserved	Between PMC_P-CH_PWROK assertion & PMC_PLTRST_ N de-assertion	External pull-up is required. Recommend 1kΩ to VCCIO. This strap should sample HIGH. There should NOT be any on-board device driving it to opposite direction during strap sampling.
CFG_01	Reserved		
CFG_09	Reserved		
CFG_10	Reserved		
CFG_12	Reserved		
CFG_13	Reserved		

Table 27-1. Pin Straps (Continued) (Sheet 2 of 4)

<table>
<tr><th>Signal</th><th>Usage</th><th>When Sampled</th><th>Comment</th></tr>
<tr><td>GP_C05/
PSE_PWM01/
PSE_UART3_CTS_N/
PSE_TGPIO30</td><td>Boot Strap 0</td><td>Rising edge of PMC_RSMRST_N</td><td>This strap has a 20 kohm ± 30% internal pull-down.
This is bit 0 (LSB) of a total of 4-bit encoded pin straps for boot configuration.
This strap is used in conjunction with Boot Strap 1,2,3, (on GPP_H00, GPP_H01, GPP_H02 respectively).
<table><tr><th>GP_C05</th><th>GP_H00</th><th>GP_H01</th><th>GP_H02</th><th>Boot Strap Configuration</th></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>IFWI boot from SPI (eSPI enabled)</td></tr><tr><td>0</td><td>1</td><td>0</td><td>0</td><td>IFWI boot from SPI (eSPI disabled)</td></tr></table>1. The internal pull-down is disabled after PMC_RSMRST_N de-asserts.
2. This signal is in the primary well.</td></tr>
<tr><td>FSPI_MOSI_IO0</td><td>Reserved</td><td>Rising edge of PMC_RSMRST_N</td><td>External pull-up is required. Recommend 20K if pulled up to 3.3V /1.8V.
This strap should sample HIGH. There should NOT be any on-board device driving it to opposite direction during strap sampling.</td></tr>
<tr><td>GP_B23/
PCHHOT_N/
SIO_S-PI1_CS1_N/
PSE_S-PI3_CS1_N/
PSE_TGPIO28</td><td>Reserved</td><td>Rising edge of PMC_RSMRST_N</td><td>This strap has a 20 kohm ± 30% internal pull-down.
0 = Default
1 = Reserved
Notes:
1. The internal pull-down is disabled after PMC_RSMRST_N de-asserts.
2. When used as PCHHOT_N and strap low, a 150K pull-up is needed to ensure it does not override the internal pull-down strap sampling.
3. This signal is in the primary well.</td></tr>
<tr><td>FSPI_IO2</td><td>Reserved</td><td>Rising edge of PMC_RSMRST_N</td><td>External pull-up is required. Recommend 20K if pulled up to 3.3V /1.8V.
This strap should sample HIGH. There should NOT be any on-board device driving it to opposite direction during strap sampling.</td></tr>
<tr><td>FSPI_IO3</td><td>Reserved</td><td>Rising edge of PMC_RSMRST_N</td><td>External pull-up is required. Recommend 20K if pulled up to 3.3V /1.8V.
This strap should sample HIGH. There should NOT be any on-board device driving it to opposite direction during strap sampling.</td></tr>
<tr><td>GP_R02/
HDA_SDO/
AVS_I2S0_TXD/
PSE_I2S0_TXD/
DMIC_CLK_B0</td><td>Flash Descriptor Security Override</td><td>Rising edge of PMC_P-CH_PWROK</td><td>This strap has a 20 kohm ± 30% internal pull-down.
0 = Enable security measures defined in the Flash Descriptor. (Default)
1 = Disable Flash Descriptor Security (override). This strap should only be asserted high using external Pull-up in manufacturing/debug environments ONLY.
Notes:
1. The internal pull-down is disabled after PMC_PCH_PWROK is high.
2. This signal is in the primary well.</td></tr>
</table>

Table 27-1. Pin Straps (Continued) (Sheet 3 of 4)

Signal	Usage	When Sampled	Comment
GP_E06/ PSE_PWM10/ PSE_TGPIO18	Reserved	Rising edge of PMC_RSMRST_N	External pull-up is required. The pull-up rail will be V1P8A or V3P3A, depending on what I/O voltage the GPIO is configured to. This strap should sample HIGH. There should NOT be any on-board device driving it to opposite direction during strap sampling.
GP_E19/ DDI0_DDC_SCL/ PSE_PWM13/ PSE_TGPIO24	GP_E18/ E19 pins VCC configu-ration	Rising edge of PMC_RSMRST_N	This strap has a 20 kohm ± 30% internal pull-down. 0 = GP_E18/E19 DDC pins at 1.8V 1 = GP_E18/E19 DDC pins at 3.3V ***Notes:*** 1. An external pull-up resistor is required if the muxed DDC signals are to be connected directly to the HDMI or DP++ connector. 2. The internal pull-down is disabled after PMC_RSMRST_N de-asserts. 3. This signal is in the primary well. 4. This strap will become RSVD in future (targeted BKC Beta release), DDC pin voltage is configurable via soft strap.
DBG_PMODE	Reserved	Rising edge of PMC_RSMRST_N	This strap has a 20 kohm ± 30% internal pull-up. This strap should sample high. There should NOT be any on-board device driving it to opposite direction during strap sampling. ***Notes:*** 1. The internal pull-up is disabled after PMC_RSMRST_N de-asserts. 2. This signal is in the primary well.
GP_F00/ SIO_UART0_RTS_N	Reserved	Rising edge of PMC_RSMRST_N	This strap has a 20 kohm ± 30% internal pull-down. This strap should not be pulled high. 0 = Default 1 = Reserved ***Notes:*** 1. The internal pull-down is disabled after PMC_RSMRST_N de-asserts. 2. This signal is in the primary well.
GP_F02/ SIO_UART0_TXD	Reserved	Rising edge of PMC_RSMRST_N	A weak external pull-up is required. The pull-up rail will be V1P8A or V3P3A, depending on what I/O voltage the GPIO is configured to.
GP_F07/ PSE_I2S1_SCLK/ AVS_I2S4_SCLK/ PSE_TGPIO14	Reserved	Rising edge of PMC_RSMRST_N	This strap has a 20 kohm ± 30% internal pull-down. This strap should sample LOW. There should NOT be any on-board device driving it to opposite direction during strap sampling. ***Notes:*** 1. The internal pull-down is disabled after PMC_RSMRST_N de-asserts. 2. This signal is in the primary well.
GP_H00/PSE_G-BE1_INT/ PSE_UART5_RXD	Boot Strap 1	Rising edge of PMC_RSMRST_N	This strap has a 20 kohm ± 30% internal pull-down. This is bit 1 of a total of 4-bit encoded pin straps for boot configuration. Refer to Boot Strap 0 (on GPP_C5) for the encoding. ***Notes:*** 1. The internal pull-down is disabled after PMC_RSMRST_N de-asserts. 2. This signal is in the primary well.

Table 27-1. Pin Straps (Continued) (Sheet 4 of 4)

Signal	Usage	When Sampled	Comment
GP_H01/PSE_G-BE1_RST_N/ PSE_UART5_TXD	Boot Strap 2	Rising edge of PMC_RSMRST_N	This strap has a 20 kohm ± 30% internal pull-down. This is bit 2 of a total of 4-bit encoded pin straps for boot configuration. Refer to Boot Strap 0 (on GPP_C5) for the encoding. ***Notes:*** 1. The internal pull-down is disabled after PMC_RSMRST_N de-asserts. 2. This signal is in the primary well.
GP_H02/PSE_G-BE1_AUXTS/ PSE_UART5_RTS_N	Boot Strap 3	Rising edge of PMC_RSMRST_N	This strap has a 20 kohm ± 30% internal pull-down. This is bit 3 of a total of 4-bit encoded pin straps for boot configuration. Refer to Boot Strap 0 (on GPP_C5) for the encoding. ***Notes:*** 1. The internal pull-down is disabled after PMC_RSMRST_N de-asserts. 2. This signal is in the primary well.
GP_DSW11	SPI Operation Voltage Select	Rising edge of PMC_ DSW_PWROK	There is no internal pull-up or pull-down on the strap. An external resistor is required. 0 = SPI voltage is 3.3V (100 kohm pull-down to GND) 1 = SPI voltage is 1.8V (4.7 kohm pull-up to VCC_3P3A_DSW) ***Note:*** The pull-down value of 100kOhm is used to reduce processor VCC_3P3A_DSW power consumption. Designs that use the previously specified value of 4.7 kOhm pull-down may continue to do so.
GP_DSW07	Skip RTC Clock Stabili-zation Delay	Rising edge of PMC_DSW_ PWROK	This strap has a 20 kohm ± 30% internal pull-down. 0 = Do not skip 95ms delay between PMC_DSW_PWROK de-assertion and PMC_SLP_SUS_N de-assertion. (Default) 1 = Skip 95ms delay between PMC_DSW_PWROK de-assertion and PMC_SLP_SUS_N de-assertion. This should only be used when an external 32kHz oscillator is used to supply the RTC clock rather than a crystal Notes: 1. The internal pull-down is disabled after PMC_DSW_PWROK asserts. 2. This signal is in the Deep Sx well.
GP_F10/ PSE_I2S1_SFRM / AVS_I2S4_SFRM /PSE_TGPIO15	Reserved	Rising edge of PMC_RSMRST_N	This strap has a 20 kohm ± 30% internal pull-down. 0 = Default 1 = Reserved Notes: 1. The internal pull-down is disabled after PMC_RSMRST_N de-asserts. 2. This signal is in the primary well.

§ §

28 Test and Debug

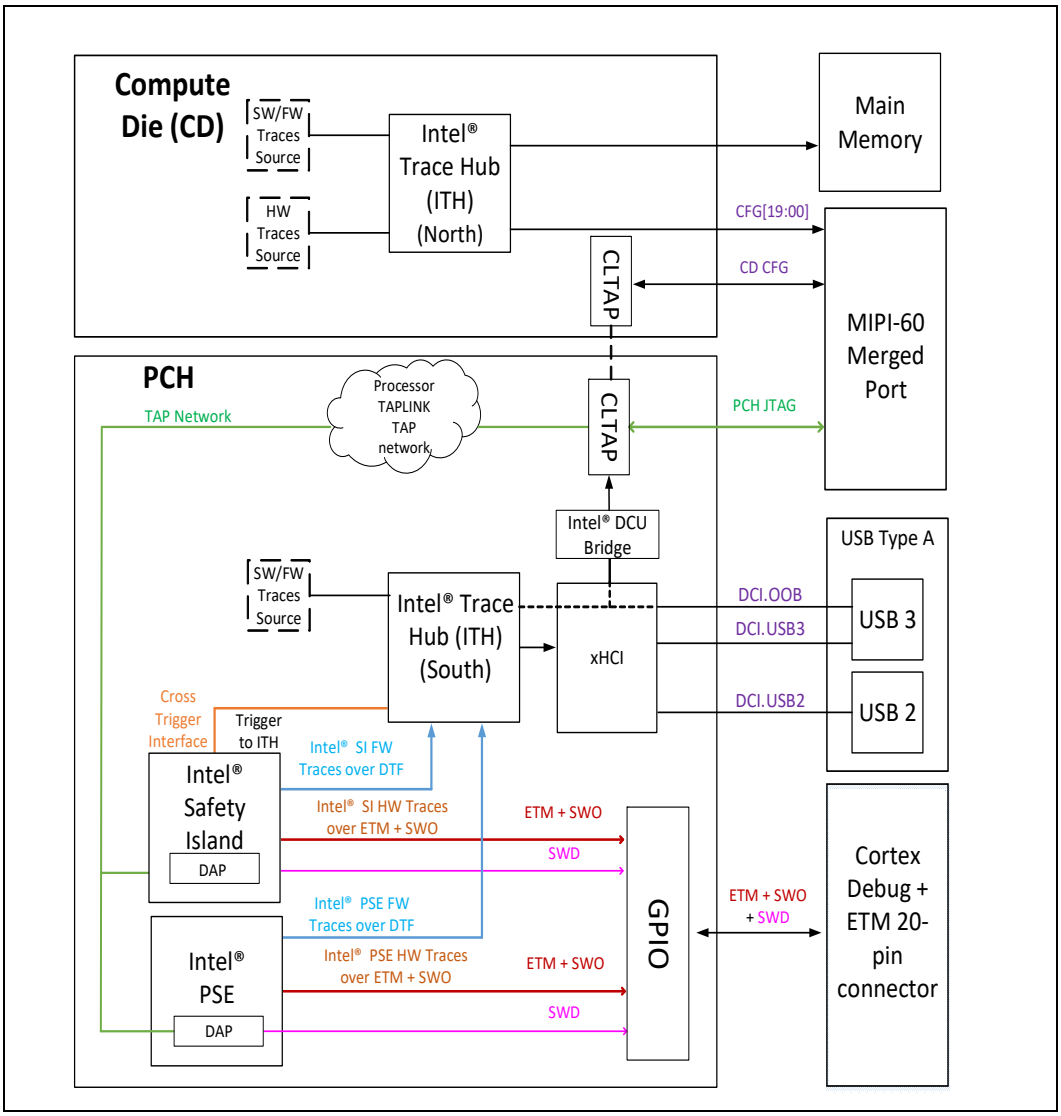
28.1 Debug Capability and Technologies

Figure 28-1 shows high level connectivity for debug and trace for Intel® and Arm*. It shows different source of traces and possible destinations.

In Compute Die, multiple have trace sources stream the data to Intel® Trace Hub (North) and then either to DRAM or PTI Signals (CFG[00:19]). The Compute Die TAP link is connected to the PCH package to be exposed on the Merged JTAG port or via Intel® DCI.

The PCH is the package contains the debug interfaces visible to customers. Besides the standard open-chassis interfaces such as PCH JTAG, it includes DCI.OOB (4-wire), DCI.USB2 and DCI.USB3 interfaces too. DCI.OOB (2-wire) is not supported. In addition, there is a SWJ-DP (Serial Wire JTAG Debug Port) Access Port, running in SWD mode for Arm* debug and trace capability.

Figure 28-1. Overall Debug Capability



28.1.1 Intel® Processor Trace

Intel® Processor Trace (Intel® PT) is a tracing capability added to Intel® Architecture, for use in software debug and profiling. Intel® PT provides the capability for precise software control flow and timing information, with limited impact to software execution. This provides enhanced ability to debug software crashes, hangs, or other anomalies, as well as responsiveness and short-duration performance issues.

28.1.2 JTAG

This section contains information regarding the testability signals that provides access to JTAG, Run-control, system control, and observation resources. JTAG (TAP) port is compatible with the IEEE Standard Test Access Port and Boundary Scan Architecture 1149.1 and 1149.6 Specification, as detailed per device in each BSDL file. JTAG Pin definitions are from IEEE Standard Test Access Port and Boundary Scan Architecture (IEEE Std. 1149.1-2013). MIPI-60 Debug Port (Connector) provides access to JTAG Port. JTAG may also be accessible via Intel® DCI for closed chassis debug usage.

28.1.3 Intel® Trace Hub

Intel® Trace Hub is a debug architecture that unifies hardware and software system visibility. Intel® Trace Hub is not merely intended for hardware debug or software debug, but full system debug. This includes debugging hardware and software as they interact and produce complex system behavior. Intel® Trace Hub defines new features and also leverages some existing debug technologies to provide a complete framework for hardware and software co-debug, software development and tuning, as well as overall system performance optimization.

Intel® Trace Hub is a set of silicon features with supported software API. The primary purpose is to collect trace data from different sources in the system and combine them into a single output stream with time-correlated to each other. Intel® Trace Hub uses common hardware interface for collecting time-correlated system traces through standard destinations (see list below). Intel® Trace Hub adopts industry standard (MIPI* STPv2) debug methodology for system debug and software development.

There are multiple trace sources are supported in the platform.

- BIOS
- AET (Architecture Event Trace)
- Power Management Event Trace
- Hardware Traces
- SW and FW Traces
- Windows* ETW (for driver or application)

There are multiple destinations to receive trace data from the Intel® Trace Hub:

- Intel® Direct Connect Interface (Intel® DCI)
  - — DCI.OOB
  - — DCI.USB2/3
- System Memory
- MIPI-60 Debug Port. The PTI signals are connected to the MIPI-60 port.

28.1.4 Intel® Direct Connect Interface (DCI)

Intel® DCI is a Intel debug tool-transport technology that connect the host to the System-Under-Test (SUT) using the USB2 and USB3 port. The advantage is debug functions and trace can be connected using existing USB2 and USB3 ports, rather than

the usual additional connectors. This enables easy connectors to production systems where debug connectors have been removed. The Intel® DCI connection supports JTAG and processor run-control debug, validation, system trace extraction, DMA read and write to system memory, OS Debug and scripting. Intel® DCI is implemented using two primary transport topologies: Intel® DCI.OOB (Direct Connect Interface Out of Band) and Intel® DCI.USB2/USB3. Formally, these topologies were called Boundary Scan to Side Band (BDSB) and Debug Class (DbC) respectively.

28.1.5 Arm* CoreSight Architecture

Arm* CoreSight Architecture is a technology that can be used to debug and trace software that runs on the Arm* Cortex*-M7 (Intel® PSE) and Arm* Cortex*-M3 (Intel® SI) Microcontroller. CoreSight debug and trace is fully supported by Arm* Development Studio for bring up and optimization of the microcontroller firmware. Arm* Debug Interface v5(ADIv5) defines a standard debug interface to debug components in the microcontroller. A SWJ-DP (Serial Wire JTAG Debug Port) Debug Access Port, running in SWD mode) and an EMT (Embedded Trace Marcocell) Trace Source are implemented. Besides Arm* Development Studio, another IDE that is supported for Arm* Cortex*-M7 (Intel® PSE) and Arm* Cortex*-M3 (Intel® SI) Microcontroller debug is Eclipse.

Note: Refer to Arm* CoreSight Architecture in Arm* website, for more information. Available at: https://developer.arm.com/architectures/cpu-architecture/debug-visibility-and-trace/coresight-architecture.

28.1.5.1 Serial Wire Debug (SWD)

Serial Wire Debug (SWD) is available through the Arm* Debug Connector to debug only the Arm* Cortex*-M7 (Intel® PSE) and Arm* Cortex*-M3 (Intel® SI) Microcontroller as an alternative interface to the JTAG via the MIPI60 Debug Port. SWD uses a standard single bi-directional data signal and clock signal as defined in the Arm* Debug Interface Architecture Specification ADIv5. As a standard interface for Arm* Cortex*-M7 (Intel® PSE) and Arm* Cortex*-M3 (Intel® SI) Microcontroller-based device, the software developer can count on a wide choice of inter operable tools from Arm* and third party tool vendors.

28.1.5.2 Embedded Trace Macrocell (ETM)

Embedded Trace Macrocell (ETM) is a real-time trace module providing instruction and data tracing of a processor. Hard real-time debugging requires close interaction with the processor. Tracing shall provide a chronological picture of a system’s inner working up to, starting from or in the vicinity of an event, mainly to guide in understanding a faulty program.

28.1.6 Platform CrashLog

The CrashLog feature is intended for use by system builders (OEMs) as a means to triage and perform first level debug of failures. Additionally, CrashLog enables the BIOS or the OS to collect data on failures with the intent to collect and classify the data as well as analyze failure trends.

CrashLog is a mechanism to collect debug information into a single location and then allow access to that data via multiple methods, including the BIOS and OS of the failing system.

CrashLog is initiated by a Crash Data Detector on observation of error conditions (TCO watchdog timeout, machine check exceptions, etc.). Crash Data Detector notifies the Crash Data Requester of the error condition in order for the Crash Data Requester to collect Crash Data from several different IPs and/or Crash Nodes and stores the data to the Crash Data Storage (on-die SRAM) prior to the reset.

After the system has rebooted, the Crash Data Collector reads the Crash Data from the Crash Data Storage and makes the data available to either to software and/or back to a central server to track error frequency and trends.

28.2 Signal Description

28.2.1 JTAG, DBG_PMODE, CFG & BPM_N Signal Description

Table 28-1. JTAG, DBG_PMODE, CFG and BPM_N Testability Signal

Signal Name	IO Direction[1]	Description
CPU_JTAG_TCK	IN	Test Clock Input (TCK): The test clock input provides the clock for the Scan-Chain 0 JTAG test logic
PCH_JTAGX	IN	Pin used to support Merged Debug Port topology
PCH_JTAG_TCK	IN	Test Clock Input (TCK): The test clock input provides the clock for the Scan-Chain 1 JTAG test logic
CPU_JTAG_TMS	IN	Test Mode Select (TMS): The signal is decoded by the Test Access Port (TAP) controller to control test operations
PCH_JTAG_TMS		
CPU_JTAG_TDI	IN	Test Data Input (TDI): Serial test instructions and data are received by the test logic at TDI
PCH_JTAG_TDI		
CPU_JTAG_TDO	OUT	Test Data Output (TDO): TDO is the serial output for test instructions and data from the test logic defined in this standard
PCH_JTAG_TDO		
CPU_JTAG_TRST_N	IN	Test Reset (active low)
PCH_JTAG_TRST_N		
DBG_PMODE	IN/OUT	Debug Power Mode Indicator. Signal is used to transmit Compute Die and PCH power/reset information to the debug tool
CFG[00:19]	IN/OUT	CFG (Parallel Trace Interface) signals are used for Compute Die Tracing
BPM[3:0]_N	OUT	Outputs from the processor that indicate the status of breakpoints and programmable counters used for monitoring processor performance.

Note:
1. Directions are specified at Processor

28.2.2 SWD & ETM Signal Description

Table 28-2. SWD & ETM Signal Description

Signal Name	IO Direction	Description
PSE_SWDIO	IN	Serial Wire Data Input/Output for Data I/O Pin
ISI_SWDIO		
PSE_SWCLK	IN	Serial Wire Clock for Clock Pin
ISI_SWCLK		
PSE_TRACESWO	IN/OUT	Serial Wire Trace Out for Instrumentation Trace Macrocell (ITM) traces
ISI_TRACESWO		
PSE_TRACEDATA [3:0]	IN/OUT	ETM Trace Data for Arm* Tracing
ISI_TRACEDATA [3:0]		
PSE_TRACECLK	IN	ETM Trace Clock for Arm* Tracing
ISI_TRACECLK		

28.3 Intel® Atom Debug and Tool

28.3.1 Open Chassis Debug

The MIPI-60 Debug Port is used for traditional open-chassis debug in platform. It provides connections for the following signal groups, TAP, (JTAG and run-control (PREQ/PRDY)), Parallel Trace (Compute Die PTI, CFG) and Miscellaneous Design signals (I2C and various debug "Hook" Signals). The Compute Die and PCH JTAG ports are compatible with the IEEE Standard Test Access Port and Boundary Scan Architecture 1149.1 and 1149.6 Specification. Both Compute Die and PCH TAP are merged in order to share the same MIPI-60 Debug Port.

Lauterbach TRACE32 is a third party debug tool set supported via the MIPI-60 Debug Port. Information regarding the available tools can be found here: https://www.lauterbach.com/frames.html?tools_intel.html

28.3.2 Closed Chassis Debug

Closed Chassis Debug uses USB interfaces as connection method and is based on Intel® DCI. Intel® DCI is implemented using two primary transport topologies: Intel® DCI.OOB (Direct Connect Interface Out of Band) and Intel® DCI.USB2 and DCI.USB3. Formerly, these topologies were called Boundary Scan to SideBand (BSSB) and Debug Class (DbC) respectively.

Intel® System Studio is a Software Tools suite for System and IoT Development. Further information on how to run Intel® DCI through Intel® System Studio System is available at https://software.intel.com/en-us/articles/system-debugging-via-direct-connect-interfacedci-of-intel-system-debug. It can be used to run Intel® DCI by using its component tool called Intel® System Debugger. Details information of the tool is available at https://software.intel.com/en-us/system-studio.

Note: Intel® DCI and USB 3.1 Gen2 are mutually exclusive.

In summary, Intel® DCI supports capabilities as below:

- Closed Chassis Debug at S0 & Sx State
- JTAG Access & Run-control (Probe Mode)
- System Tracing with Intel® Trace Hub

Debug host software that support Intel® DCI is:

- Intel® System Studio (ISS)
- TRACE32 by Lauterbach

28.3.2.1 Intel® DCI.OOB (Out of Band)

Intel® DCI.OOB was developed to provide an alternate path to convey controls and data to or from Intel® Trace Hub by connecting physically to the target through a USB 3.1 Gen2 port over Type A receptacle. Refer to Section 1.1.1 for more information on target. Intel® DCI.OOB provides an alternate side band path around the USB 3.1 controller, so that the embedded logic can be accessed, even when the USB 3.1 controller is not alive (such as in low power states) or is malfunctioning. This path does not rely on USB 3.1 Gen2 protocol, link layer, or physical layer, because the xHCI functions are generally not available in such conditions.

Instead, this path relies on a special adapter that was developed by Intel called Intel® SVT Closed Chassis Adapter (CCA). It is a simple data transformation device. This adapter works together with debug host software and the embedded logic, contain a back-pressure scheme that makes both sides tolerant of overflow and starvation conditions, which is equivalent of USB 3.1 Gen2 link layer. This path also use native Intel® DCI packet protocol instead of USB 3.1 Gen2 protocol.

Besides Intel® SVT CCA, Lauterbach is an example of Third Party Vendor (TPV) solution. User may use a specific Lauterbach hardware and software configuration to connect between the Debug Host System and the Target Platform. It need Debug Host System (Lauterbach CombiProbe) to be in Downstream Facing Port (DFP) mode for Intel® DCI.OOB support in S0ix and Debug Host System (Lauterbach CombiProbe v2) can be DFP or Upstream Facing Port (UFP) for Intel® DCI.OOB supports in S0. In Closed Chassis Debug, Lauterbach debug tools-set is planned as below.

Lauterbach Intel® DCI.OOB - CombiProbe v2 and DCI.OOB Whisker USB Cable via TRACE32

Further information is available in Debugging via Intel® DCI User Guide from Lauterbach at http://www2.lauterbach.com/pdf/dci_intel_user.pdf.

28.3.2.2 Intel® DCI.USB2 & DCI.USB3

Intel® DCI.USB2 & DCI.USB3 is a USB hosted Intel® DCI transport and the higher USB bandwidths, multiple parallel pipes or endpoints and BULK-mode data-integrity and retry-recovery mechanisms built into the protocol. Supported USB endpoints include: DFx for JTAG/Run-control IA cores in system; General Purpose 1 (GP1) for Kernel Mode Debug (KMD); General Purpose 2 (GP2) for Direct Memory Access (DMA) to system memory; Trace (TRC) for streaming of live tracker.

Intel® DCI.USB2 and DCI.USB3 supports multiple USB transport modes.

- Intel® DCI.USB2 - Provides limited ~35MB/s usable bandwidth, but extends USB hosting to cover early-boot and low power Sx and S0ix states.

- Intel® DCI.USB3 - Provides an increase in S0 bandwidth up to ~800MB/s usable bandwidth (generally limited further by host and host SW).

28.3.3 Debug Considerations

28.3.3.1 Debug During Multiple Contiguous System State Transitions

Delayed Authentication Mode (DAM) must be disabled via soft strap when the system is being put through multiple contiguous S0 > S4 > S0 or S0 > S3 > S0 system state transitions.

28.4 Arm* Debug and Tool

28.4.1 Open Chassis Debug

Arm* enables two use-models of debug and trace from Arm* which are Arm* only debug and trace use Serial Wire Debug (SWD) & Embedded Trace Macrocell (ETM) and Arm* debug via JTAG (TAP).

28.4.1.1 Arm* Debug and Trace through SWD and ETM

Arm* Cortex-M processor-based devices use Arm* CoreSight technology. The Arm* CoreSight feature is available via SWD and ETM interfaces using standard low-cost target connector. In processor, Arm* only debug and trace is enabled via an Arm* Cortex 20 Pin Debug Connector connecting for debug through SWD and tracing through ETM. If Arm* FW and SW traces are not required, 10 Pin Debug Connector is sufficient for Arm* debug. Information about Arm* Target Connectors can be reached at Arm* KEIL website: http://www.keil.com/support/man/docs/ulink2/ulink2_hw_connectors.htm

ULINKpro is an example of third party tool for Arm*-based subsystem debug capability. The tool connects via the Arm* Cortex 20 Pin Debug Connector. It supports for downloaded programs to target hardware, single-step and examine memory and register. Information about this tool can be reached at Arm* KEIL website: http://www.keil.com/support/man/docs/ulinkpro/. This tool can be used with IDE Arm* Development Studio.

OpenOCD is another way to debug Arm* via the Arm* Cortex 20 Pin Debug Connector. The OpenOCD aims to provide debugging for embedded target devices. It doesn't support on tracing capability. This tool can be used with Eclipse IDE. It requires below adapters and cable for functioning on CRB.

- Arm*-USB-OCD Adapter
- Arm*-JTAG-SWD Adapter
- J-LINK 19-Pin Cortex M Adapter with 20 pins cable

28.4.1.2 Arm* Debug via JTAG (TAP)

In this use-case, Arm* debug is enabled via MIPI-60 Debug Port (for Intel® Safety Island only) connecting to the JTAG (TAP) interface. Refer Section 28.4.3 to understand the JTAG (TAP) link and SWD selection mechanism.

28.4.2 Closed Chassis Debug

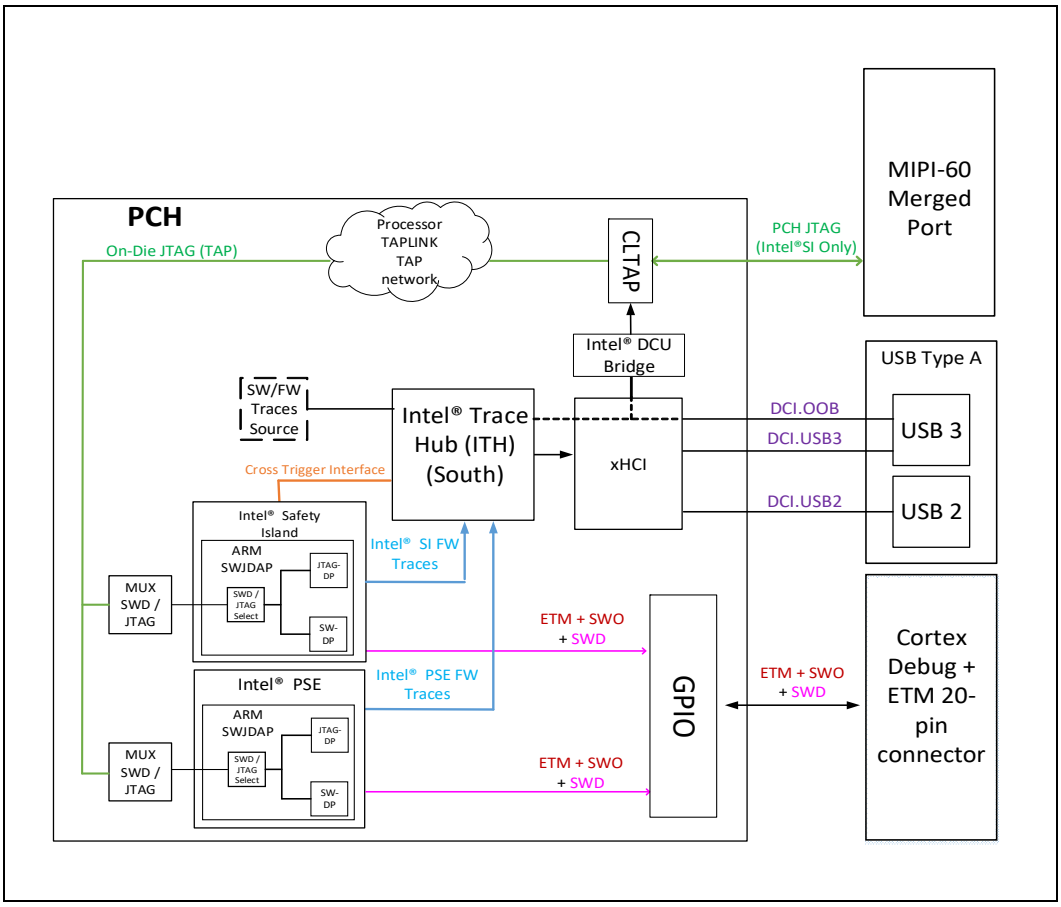
In Closed Chassis Debug, Arm* debug via JTAG (TAP) is enabled through Intel® DCI in the PCH. The Arm* cores send FW traces via Debug Trace Fabric (DTF) to Intel® Trace Hub. The JTAG (TAP) interface gets embedded in Intel network and accessed only through CLTAP via Intel® DCI.OOB.

28.4.3 JTAG (TAP) and Serial Wire Debug selection

Arm* supports debug access via standard JTAG (TAP) and SWD. A signal (CLTAP) at the interface of a DAP is used to perform the selection. The default debug access mode is through SWD and the value will be ‘0’.

Note: Since the same registers are used for the configuration of Intel® PSE and Intel® Safety Island (SI) debug access, it is not possible to use different debug access selections concurrently. For example, it is not possible to concurrently use JTAG (TAP) for Intel® SI and SWD for Intel® PSE.

Figure 28-2. Switching Flows Between JTAG and SWD



28.4.3.1 Switching from SWD to JTAG for Intel® PSE

To access Arm*-DAP from SW

- Setup up the GPIO to Native Function 4 to use the GPIO's (GP_F[8,11:17]) for SW access

Note: Refer to Chapter 21, "General Purpose Input and Output (GPIO)", for GPIO Multiplexing Table (Native Function).

To switch Arm*-DAP from Serial Wire Debug to JTAG (via Intel® CLTAP)

- Write OSE_TAP2OCP_TAP.ARM_JTAG_SW_SWITCH_VAL = 1 to select JTAG mode
- Write OSE_TAP2OCP_TAP.ARM_RST_OVERRIDE_VAL = 1 to apply reset
- Write OSE_TAP2OCP_TAP.ARM_RST_OVERRIDE_VAL = 0 to release reset

To switch Arm*-DAP from JTAG (via Intel CLTAP) to Serial Wire Debug

- Write OSE_TAP2OCP_TAP.ARM_JTAG_SW_SWITCH_VAL = 0 to select Serial Wire Debug mode
- Write OSE_TAP2OCP_TAP.ARM_RST_OVERRIDE_VAL = 1 to apply reset
- Write OSE_TAP2OCP_TAP.ARM_RST_OVERRIDE_VAL = 0 to release reset

28.4.3.2 Switching from SWD to JTAG for Intel® Safety Island

To access Arm*-DAP from SW

- Setup up the GPIO to Native Function 5 to use the GPIO's (GP_F[8,11:17]) for SW access

Note: Refer to Chapter 21, "General Purpose Input and Output (GPIO)", for GPIO Multiplexing Table (Native Function).

To switch Arm*-DAP from Serial Wire Debug to JTAG (via Intel® CLTAP)

- Write OSE_TAP2OCP_TAP.ARM_JTAG_SW_SWITCH_VAL = 1 to select JTAG mode
- Write OSE_TAP2OCP_TAP.ARM_RST_OVERRIDE_VAL = 1 to apply reset
- Write OSE_TAP2OCP_TAP.ARM_RST_OVERRIDE_VAL = 0 to release reset

To switch Arm*-DAP from JTAG (via Intel® CLTAP) to Serial Wire Debug

- Write OSE_TAP2OCP_TAP.ARM_JTAG_SW_SWITCH_VAL = 0 to select Serial Wire Debug mode
- Write OSE_TAP2OCP_TAP.ARM_RST_OVERRIDE_VAL = 1 to apply reset
- Write OSE_TAP2OCP_TAP.ARM_RST_OVERRIDE_VAL = 0 to release reset

To switch SWJ-DP from JTAG (via Intel® CLTAP) to SWD operation:

- Send more than 50 SWCLKTCK cycles with SWDIOTMS=1. This ensures that both SWD and JTAG are in their reset states.
- Send the 16-bit JTAG-to-SWD select sequence on SWDIOTMS
- Send more than 50 SWCLKTCK cycles with SWDIOTMS=1. This ensures that if SWJ-DP was already in SWD mode, before sending the select sequence, the SWD goes to line reset.
- Perform a READID to validate that SWJ-DP has switched to SWD operation.
- The 16-bit JTAG-to-SWD select sequence is defined to be 0b0111100111100111, MSB first. This can be represented as 16'h79E7 if transmitted MSB first or 16'hE79E if transmitted LSB first.

To switch SWJ-DP from SWD to JTAG (via Intel® CLTAP) operation:

- Send more than 50 SWCLKTCK cycles with SWDIOTMS=1. This ensures that both SWD and JTAG are in their reset states.
- Send the 16-bit SWD-to-JTAG select sequence on SWDIOTMS.
- Send at least five SWCLKTCK cycles with SWDIOTMS=1. This ensures that if SWJ-DP was already in JTAG mode before sending the select sequence, it goes into the TLR state.
- Set the JTAG-DP IR to READID and shift out the DR to read the ID.

- The 16-bit JTAG-to-SWD select sequence is defined to be 0b0011110011100111, MSB first. This can be represented as 16'h3CE7 if transmitted MSB first or 16'hE73C if transmitted LSB first.

28.5 Debug Interface Availability

Table 28-3. Debug Interface Availability

Power State	JTAG	CFG [19:00] Signals	DCI.OOB (4 -wire)	DCI.USB2	DCI.USB3	Intel® PSE[1]	Intel® Safety Island[1]
S0	YES	YES	YES	YES	YES	YES	YES
S0i2.0	YES	YES	YES	YES	NO	YES	NO
S0i3.0	YES	YES	YES	YES	NO	NO	NO
Note: 1. JTAG, SWD, and ETM interfaces.							

28.6 References

Specification	Location
IEEE Standard Test Access Port and Boundary Scan Architecture	http://standards.ieee.org/findstds/standard/1149.1-2013.html
IEEE Standard for Boundary-Scan Testing of Advanced Digital Networks	https://standards.ieee.org/standard/1149_6-2015.html

§ §

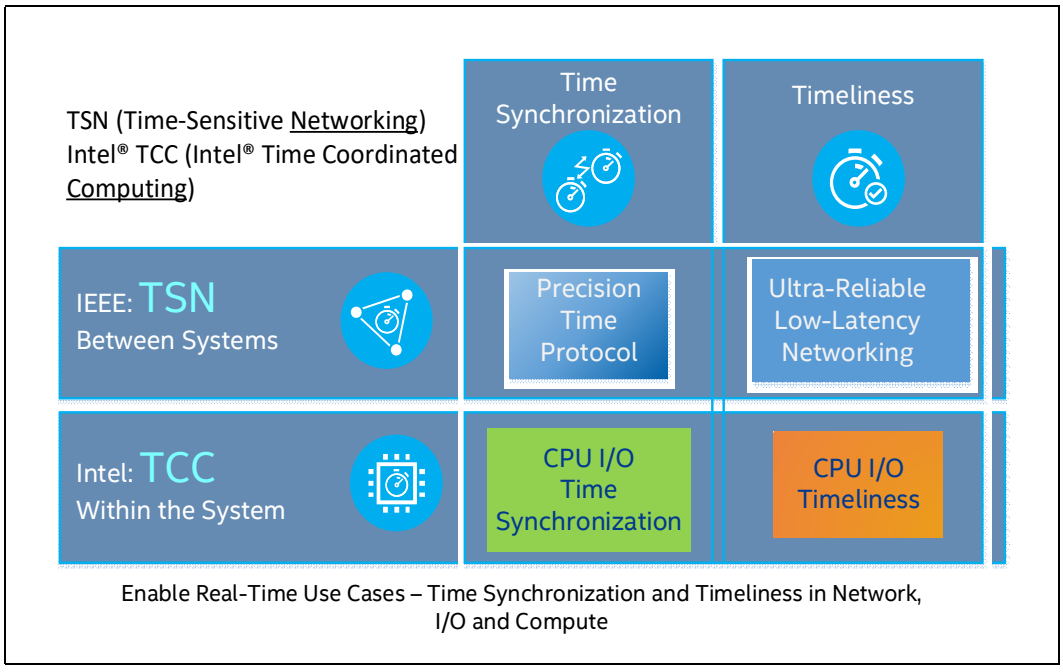
29 Intel® Time Coordinated Computing

29.1 Intel® Time Coordinated Computing Overview

Intel® Time Coordinated Computing (Intel® TCC) is ushering in a new era of coordination of computation scheduling to reduce jitter and improve performance of time-sensitive applications. Intel® TCC is not an IP block but a set of capabilities available across multiple IP blocks of the processor. Intel® TCC enhances performance along two vectors: Time Synchronization and Timeliness. Timeliness is traditionally referred to as "Real-Time" systems that are focused on the optimization. Time Synchronization provides a hardware mechanism to precisely determine how various IP specific clocks are related. Real-Time provides a hardware mechanism to minimize the latency of data packets from one IP block to another IP block.

This chapter describes the key features and requirements that compose Intel® TCC, split into the respective performance vectors. For each feature, the motivation, description, software interface (if any), hardware dependencies (if any) and formal requirements are listed.

Figure 29-1. Intel® TCC Features within System and TSN between Systems



29.2 Intel® Time Coordinated Computing Features

Intel® TCC helps real-time applications meet performance metrics, such as worst-case execution time (WCET). In addition, it keeps all software and hardware time-synchronized.

Intel® TCC performance is measured by the temporal determinism afforded to the critical tasks. The following section outlines the platform capabilities that enable temporal determinism.

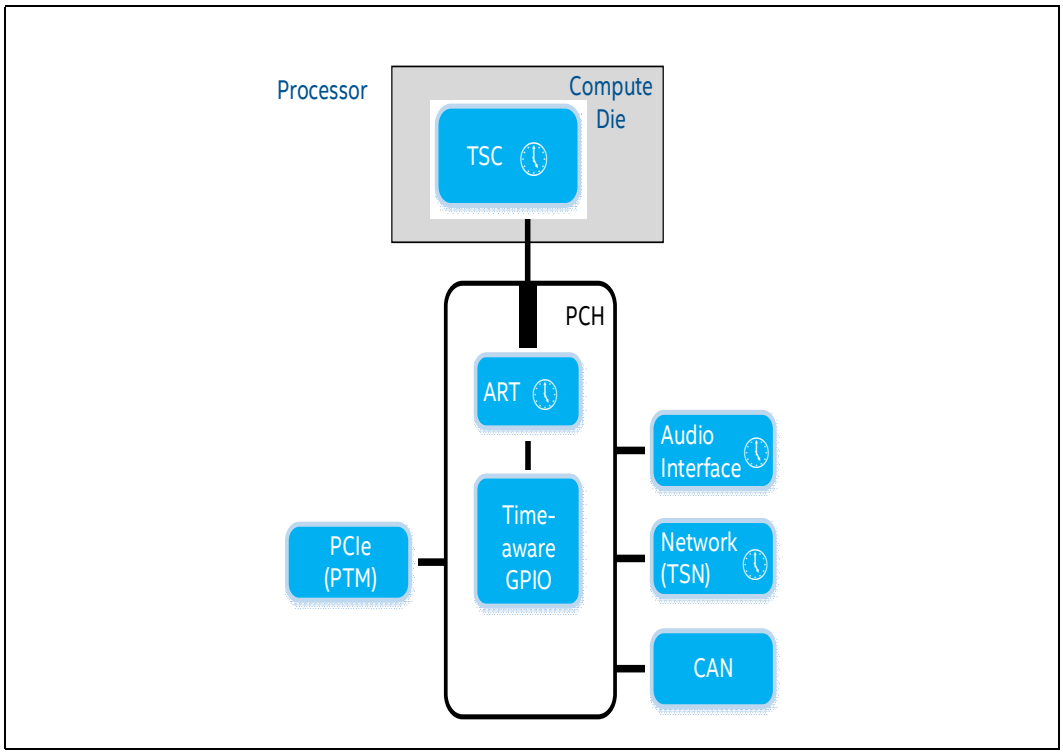
29.2.1 Time Synchronization Features

The processor provides a common timekeeping framework, based on the Always Running Timer (ART) defined in Intel® 64 Architecture and IA-32 Architectures Software Developer's Manual. ART runs at the core crystal clock frequency. The ART clock is at 19.2MHz. This makes it possible for software to calculate the precise time relationship between numerous subsystems, including the CPU’s time-stamp counter (TSC), the network device, time-aware GPIOs (TGPIO), and other various IPs. This processor support makes deep sub-microsecond time-correlation / synchronization possible, which is a requirement of coordinated computing, sensing, and actuation.

Processor Time Synchronization supports the following:

- Global time reference based on Always Running Timer (ART)
- IEEE Standard 802.1AS, the TSN standard for time synchronization across wired Ethernet network
- Global time synchronization for the following local time bases with respect to ART:
  - GbE Time-Sensitive Networking (TSN)
  - High Definition Audio/Audio DSP
  - PCI Express Root-ports Precision Time Measurement (PCIe PTM)
  - Controller Area Network (CAN)
- Audio cross-timestamp
- Time-aware GPIO

Figure 29-2. Platform Time Synchronization



29.2.1.1 CPU Time Stamp Counter to I/O Time Synchronization

The CPU Time Stamp Counter (TSC) Root in the Power Management Controller (PMC) is the global referenced time in the Time Synchronization architecture. The software can access the CPU TSC through the Read Time Stamp Counter (RDTSC) assembly instruction. RDTSC is based on the ART counter, which is the root of time in the system. The CPU TSC is derived from ART. ART is discoverable in the CPUID leaf 15H, and the ART relationship to all other clock domains is maintained in software.

Using TSN the ART clock is extended outside of the platform. Refer to Section 29.2.1.3 for more information about IEEE Standard 802.1AS network time synchronization. The relation between the network device timestamp clock and ART (also TSC) is determined using cross-timestamp hardware.

The relation between Audio hardware clock and ART (also TSC) is determined using cross-timestamp hardware. Refer to Section 29.2.1.4 for more information about audio cross-timestamp.

The time-aware GPIO interface logic is driven by ART. Edge events (one shot and pulse train) can be sampled or driven with respect to the ART clock. Refer to Section 29.2.1.5 for details on time-aware GPIO.

29.2.1.2 Off-Chip Time Synchronization via PCIe* PTM

Besides supporting known hardware, ART also extends to PCIe* peripherals using Precision Time Measurement (PTM). PTM provides a hardware mechanism for PCIe devices to correlate clock domains between an end point and the root complex.

29.2.1.3 IEEE Standard 802.1AS Network Time Synchronization

The IEEE Standard 802.1AS TSN standard extends time synchronization across multiple platforms for time-sensitive applications such as audio and video when these platforms are Ethernet connected. The integrated GbE MAC supports the IEEE Standard 802.1AS generalized Precision Time Protocol (gPTP) event messages for network path delay measurement between two time-aware systems.

Time Synchronization utilizes the standard IEEE Standard 802.1AS capability as the base feature to enable cross-platform time synchronization. At the high level, the IEEE Standard 802.1AS implements the following:

- Select the best clock source on the network using the algorithm specified in IEEE Standard 802.1AS
- Determine the propagation delay through the network
- Calculate the offset between the selected clock source and the local clock

The IEEE Standard 802.1AS timer runs on its Local Time base. It supports a 19.2MHz crystal oscillator clock determined by a pin-strap based on platform configurations where it does not require any PLL to be running. The crystal oscillator continues running when IEEE Standard 802.1AS time synchronization is enabled such that a valid time stamp is always available when gPTP event messages are transmitted or received.

29.2.1.4 Audio Cross-Timestamp

For Time Synchronization, the audio software stack handles the scheduled audio cross-timestamp such that an audio stream plays in-sync with multiple audio/video components in a network distributed system. Using audio hardware-assisted time synchronization, software can determine the relation between the selected best clock source on the network using the algorithm specified in IEEE Standard 802.1AS and the local audio clock, by determining the appropriate start time across the network. Any error is corrected using audio sample rate conversion by performing linear regression approximation to calculate variance. Software may utilize other information available to achieve the best audio cross-timestamp accuracy.

Before the scheduled start of the stream is carried out, software is expected to trigger a time synchronization request such that offset tracking between the Local Time base and the ART is intact and does not roll over.

29.2.1.5 Time-Aware GPIO

The PCH has two Time-Aware GPIO (TGPIO) controllers. Each controller can be independently configured to generate or capture timestamped events. TGPIO events are timestamped using the Always Running Timer (ART) clock. TGPIOs provide the following functions:

- Generate an edge or pulse at a future ART clock value
- Generate periodic pulse or square wave output with a software programmed interval

- TGPIO to capture the current ART value when an edge (rising or falling) is detected
- Modulate the periodic output interval to generate an average interval that is an arbitrary fraction of the ART period and aligned with the system clock

The I/O from the TGPIO controllers are routed on the board via the PMC_TGPIO0 signal and PMC_TGPIO1 signal which are connected to pins AN11 and CJ27, respectively.

Intel® PSE supports 40 time-aware GPIOs as native function muxed on GPIO pins. Time Synchronization supports time-aware GPIO events. The purpose of time-aware GPIO is to extend time synchronization through GPIO events to comprehend interfaces that do not support time synchronization natively. By using time-aware GPIO, these interfaces are able to utilize the time synchronization infra-structure. It provides the linkage between the ART and the interface devices through the GPIO pin such that devices are time-aware.

The time-aware GPIO has the following functional behavior:

- Based on global ART timer. There is no separate local GPIO time domain.
- Time-aware GPIO can be an input or an output.
- The polarity of the time-aware GPIO event is software configurable as rising edge, falling edge, or toggle edge.
- As an input, a time-aware GPIO input event triggers the hardware to capture the ART time in the Time Capture register. The time-aware GPIO input event must be asserted for a period of at least 2 crystal oscillator clocks in order for the event to be recognized.
- As an output, a match between the ART time and the software programmed time value triggers the hardware to generate a time-aware GPIO output event and capture the ART time in the Time Capture register. If periodic mode is enabled, hardware generates the periodic time-aware GPIO events based on the programmed interval. The time-aware GPIO output event is asserted by hardware for a period of at least 2 crystal oscillator clocks.
- Time-aware GPIO supports event counter. When time-aware GPIO is configured as input, the event counter increments by 1 for every input event triggered. When time-aware GPIO is configured as output, the event counter increments by 1 for every output event generated. The event counter provides the correlation to associate the time-aware GPIO event (the N^{th} event) with the captured ART time. The event counter value is captured when a read to the Time Capture register occurs.
- Time-aware GPIO registers are defined as 32-bit (Double-Word) registers. For a 64-bit value that spans across 2 registers, software must account for possible rollover when reading the 2 registers one after another.
- Time-aware GPIO configuration bits must be programmed appropriately (i.e., static) before the time-aware GPIO Enable bit is set to '1'.
- Before reading the Event Counter Capture register, software must first perform a read to the Time Capture register such that these two values correspond with each other despite being obtained by 2 separate register reads one after another.

For more information Intel® PSE on time-aware GPIO, please refer to Section 21.3.5, “Time-Aware GPIO” and Section 22.18, “Time-Aware GPIO”.

29.2.2 Real-Time Features for I/O

The processor provides hardware mechanisms to improve the worst-case, which aims to put an upper bound on the latency for data coming into the system. The following features are required to provide a bounded latency for incoming transactions.

29.2.2.1 Upstream Virtual Channels

The processor provides support for upstream virtual channels through the PCIe and processor fabric. I/O devices are expected to make use of the PCIe base specification definition around quality of service mechanisms for data movement, specifically those related to Traffic Class. Upstream transactions that are differentiated by Traffic Class have the ability to be mapped to a separate virtual channel in an attempt to provide a low latency path to the coherent domain.

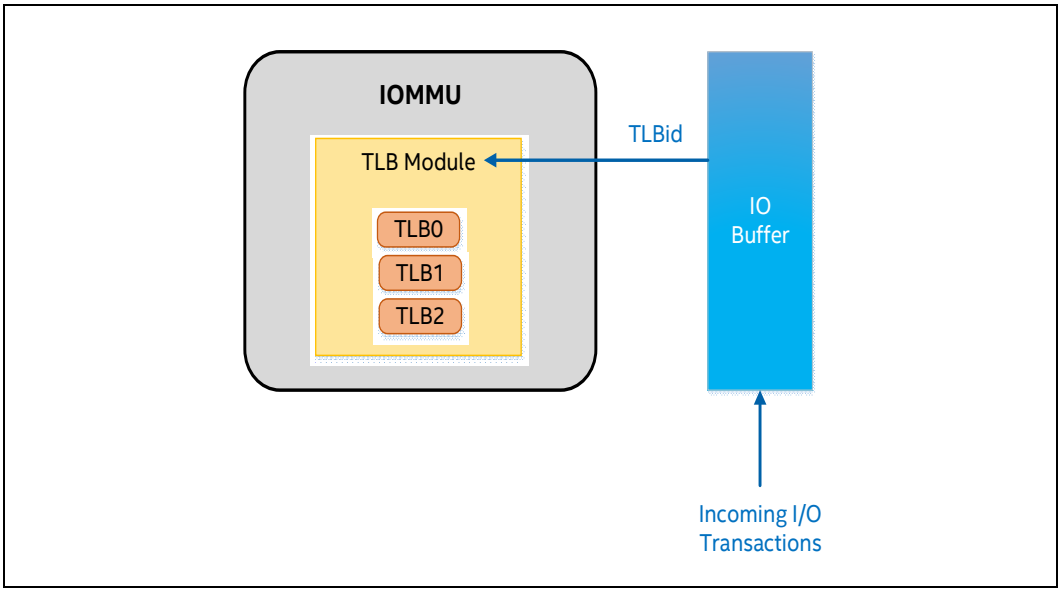
The Single Virtual Channel (VC) PCIe0 controller supports 1 virtual channel. The Multi VC PCIe1, PCIe2 and PCIe3 controllers support 2 virtual channels, VC0 (best effort) and VC1 (high priority).

29.2.2.2 Dedicated IOTLB

The processor provides support for a dedicated I/O Translation Lookaside Buffer (IOTLB) to cache critical address translations for real-time I/O traffic.

When workload consolidation is implemented with virtualization, the I/O transactions of real-time workloads running in a virtual machine are subject to increased jitter as a result of a page walk by the I/O Memory Management Unit (IOMMU). To avoid additional page walk latency on all I/O transactions, an IOTLB is implemented in the IOMMU to cache the guest physical address to host physical address translation. There are no Quality of Service (QoS) mechanisms available on this cache, and therefore translations for time-critical addresses are subject to eviction as a result of concurrent best effort traffic.

Figure 29-3. IOTLB Usage



By providing a dedicated IOTLB for time-critical transactions, it is possible to minimize the additional latency and jitter previously introduced by the IOMMU. The processor specifies a TLBid based on the PCIe traffic class used, avoiding contention on the cache resources from concurrent best effort traffic.

29.2.3 Real-Time Features for Compute Die

The processor provides mechanisms to promote the reduction of execution jitter, leading to a more deterministic computing environment.

29.2.3.1 Multiple Outstanding MMIO

The processor provides support for tracking multiple downstream non-posted transactions simultaneously targeting the Memory Mapped I/O subsystem.

29.2.3.2 Alignment Check Exception on Split Lock

The processor provides support to generate an Alignment Check Exception (#AC) when an application attempts to issue a split lock.

When the LOCK prefix is used on an unaligned operand, there is the potential for the operand to span multiple cache lines. In the case where an atomic operation is necessary across two cache lines, a bus lock is executed. A bus lock stops all cores and I/Os from initiating transactions, resulting in an increase in latency. This increased latency can be significant and many real-time applications cannot tolerate the jitter that a split lock introduces.

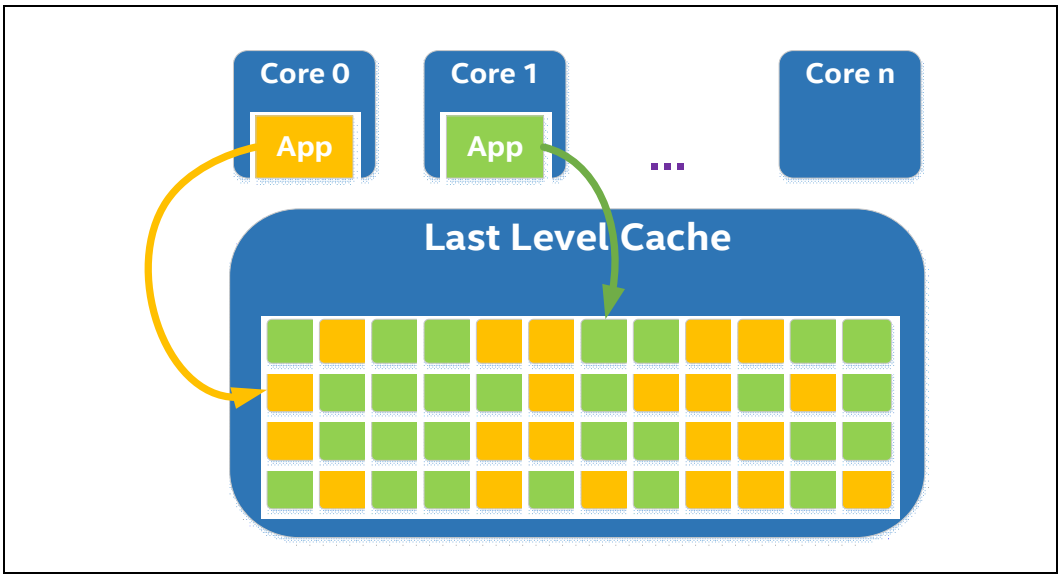
29.2.3.3 L2 and LLC Cache QoS

The processor provides support for Cache Allocation Technology (CAT), a Quality of Service (QoS) feature under the Intel® Resource Director Technology (Intel® RDT) portfolio.

On Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors for IoT Applications that support Intel® Time Coordinated Computing (Intel® TCC), a model-specific, non-architectural version of Level 3 (L3) CAT is supported. Support for this feature will not be enumerated via CPUID leafs as indicated in the Intel Software Developer Manuals. For details on how to use L3 CAT, including information on the number of classes of service, selecting the active class of service, and more, please consult the Cache Allocation Technology chapter in the Real-Time Tuning Guide for Intel Atom® x6000E Series Processors.

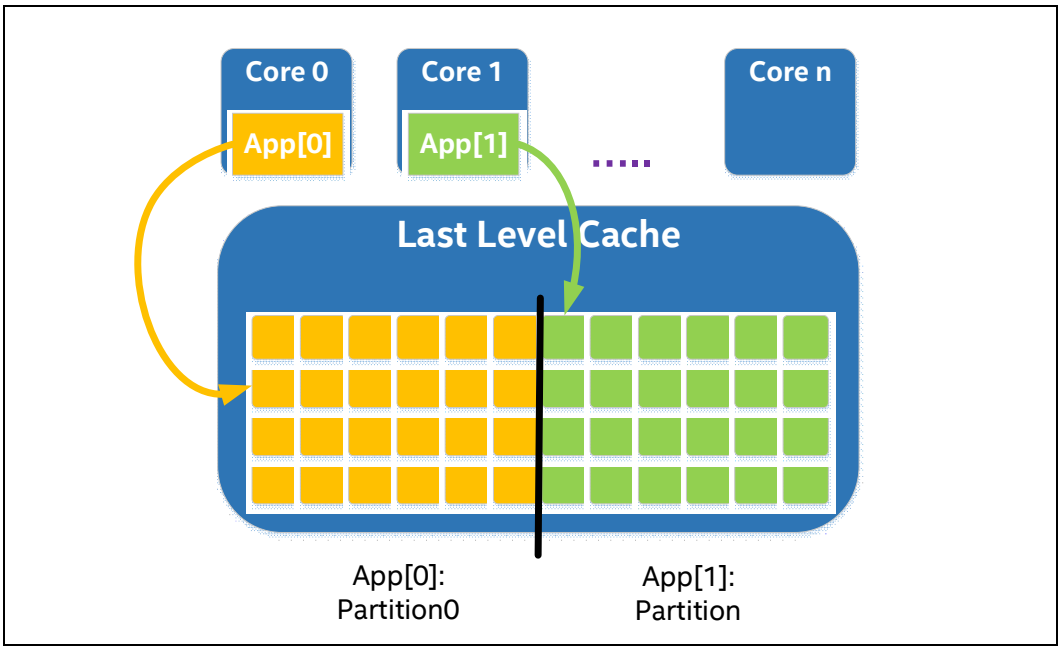
CAT helps address shared cache resource contention by providing software control of where data is allocated into the Level 2 (L2) cache and last-level cache (LLC), enabling isolation and prioritization of key applications. The LLC may also be referred to as a third level cache, L3. Without CAT, cache resources are shared between applications.

Figure 29-4. LLC without Cache QoS



With CAT, cache resources can be partitioned. This partitioning leads to improved performance determinism.

Figure 29-5. LLC with Cache QoS



29.3 Intel® TCC Tools

Intel® TCC Tools provide application and middleware support, including APIs, tools, and sample applications that enable developers to access certain Intel® TCC features. Intel® TCC Tools also enable developers to analyze the behavior of real-time applications. For more information about Intel® TCC Tools beyond the scope of this document, see the Intel® TCC Tools product page listed in Intel® Developer Zone (IDZ).

§ §

30 Global Device IDs

30.1 Overview

This chapter lists the Global Device IDs for the Processor.

30.2 PCH Global Device IDs

Table 30-1 lists the Global Device IDs for the Platform Controller Hub (PCH). The Global Device IDs comprise of PCI Device ID, Device number, Function number and Description.

Table 30-1. PCH Global Device IDs (Sheet 1 of 5)

Device ID	Bus#	Device#	Function#	Description
4B00	0	31	0	Enhanced Serial Peripheral Interface (eSPI) Controller
4B01	0	31	0	Reserved
4B20	0	31	1	Primary to Sideband Bridge (P2SB)
4B21	0	31	2	Power Management Controller (PMC)
4B55	0	31	3	Converged Audio, Video, Speech (cAVS) Controller
4B56	0	31	3	cAVS Controller
4B57	0	31	3	cAVS Controller
4B58	0	31	3	cAVS Controller
4B59	0	31	3	cAVS Controller
4B5A	0	31	3	cAVS Controller
4B5B	0	31	3	cAVS Controller
4B5C	0	31	3	cAVS Controller
4B23	0	31	4	System Management Bus (SMBus) Controller
4B24	0	31	5	Serial Peripheral Interface (SPI) Controller for Flash & TPM
4B26	0	31	7	Intel® Trace Hub
4B28	0	30	0	Intel® Serial I/O: Universal Asynchronous Receiver/ Transmitter (UART) Controller #0
4B29	0	30	1	Intel® Serial I/O: UART Controller #1
4B2A	0	30	2	Intel® Serial I/O: Serial Peripheral Interface (SPI) Controller #0
4B2B	0	30	3	Intel® Serial I/O: SPI Controller #1
4B30	0	30	4	Reserved

Table 30-1. PCH Global Device IDs (Sheet 2 of 5)

Device ID	Bus#	Device#	Function#	Description
4B32	0	30	4	Gigabit Ethernet TSN Controller (SGMII: 1Gb & 2.5Gb Mode). ***Note:*** SGMII 1Gb & 2.5Gb Modes share the same Device ID. The GCR.LINK_MODE register field shall be used to configure operation at 1Gb or 2.5Gb.
4B68	0	30	6	High Precision Event Timer (HPET)
4B69	0	30	7	I/O Advanced Programmable Interrupt Controller (IOAPIC)
4BB3	0	29	0	Intel® Programmable Services Engine (Intel® PSE): Local Host to PSE (LH2OSE) IPC
4BA0	0	29	1	Intel® Programmable Services Engine (Intel® PSE):Gigabit Ethernet TSN Controller #0 (RGMII: 1Gb Mode) **Note**: The two GbE Controllers inside Intel® PSE are identical, different DEVICE ID to identify their unique instantiation (4BA* & 4BB*)
4BA1	0	29	1	Intel® PSE: Gigabit Ethernet TSN Controller #0 (SGMII: 1Gb Mode) **Note**: The two GbE Controllers inside Intel® PSE are identical, different DEVICE ID to identify their unique instantiation (4BA* & 4BB*)
4BA2	0	29	1	Intel® PSE: Gigabit Ethernet TSN Controller #0 (SGMII: 2.5Gb Mode) **Note**: The two GbE Controllers inside Intel® PSE are identical, different DEVICE ID to identify their unique instantiation (4BA* & 4BB*)
4BB0	0	29	2	Intel® PSE: Gigabit Ethernet Time Sensitive Networking (TSN) Controller #1 (RGMII: 1Gb Mode) **Note**: The two GbE Controllers inside Intel® PSE are identical, different DEVICE ID to identify their unique instantiation (4BA* & 4BB*)
4BB1	0	29	2	Intel® PSE: Gigabit Ethernet TSN Controller #1 (SGMII: 1Gb Mode) **Note**: The two GbE Controllers inside Intel® PSE are identical, different DEVICE ID to identify their unique instantiation (4BA* & 4BB*)
4BB2	0	29	2	Intel® PSE: Gigabit Ethernet TSN Controller #1 (SGMII: 2.5Gb Mode) **Note**: The two GbE Controllers inside Intel® PSE are identical, different DEVICE ID to identify their unique instantiation (4BA* & 4BB*)
4BB4	0	29	3	Intel® PSE: Direct Memory Access (DMA) Controller #0
4BB5	0	29	4	Intel® PSE: DMA Controller #1
4BB6	0	29	5	Intel® PSE: DMA Controller #2
4BB7	0	29	6	Intel® PSE: Pulse Width Modulation (PWM) Controller
4BB8	0	29	7	Reserved
4B38	0	28	0	PCI Express* (PCIe*) Root Port #0 (PCIe 0, Single VC)
4B39	0	28	1	PCIe* Root Port #1 (PCIe 0, Single VC)
4B3A	0	28	2	PCIe* Root Port #2 (PCIe 0, Single VC)

Table 30-1. PCH Global Device IDs (Sheet 3 of 5)

Device ID	Bus#	Device#	Function#	Description
4B3B	0	28	3	PCIe* Root Port #3 (PCIe 0, Single VC)
4B3C	0	28	4	PCIe* Root Port #4 (PCIe 1, Multi VC)
4B3D	0	28	5	PCIe* Root Port #5 (PCIe 2, Multi VC)
4B3E	0	28	6	PCIe* Root Port #6 (PCIe 3, Multi VC)
4BB9	0	27	0	Intel® PSE: Inter-Integrated Circuit (I^2C) Controller #0
4BBA	0	27	1	Intel® PSE: I^2C Controller #1
4BBB	0	27	2	Intel® PSE I^2C Controller #2
4BBC	0	27	3	Intel® PSE: I^2C Controller #3
4BBD	0	27	4	Intel® PSE I^2C Controller #4
4BBE	0	27	5	Intel® PSE: I^2C Controller #5
4BBF	0	27	6	Intel® PSE I^2C Controller #6
4B47	0	26	0	embedded Multi Media Card (eMMC) Controller
4B48	0	26	1	Secure Digital (SD) & Secure Digital I/O Controller
4B49	0	26	2	Reserved
4B4A	0	26	3	Intel® Safety Island (Intel® SI) Controller
4B4B	0	25	0	Intel® Serial I/O: Inter-Integrated Circuit (I^2C) Controller #4
4B4C	0	25	1	Intel® Serial I/O I^2C Controller #5
4B4D	0	25	2	Intel® Serial I/O: Universal Asynchronous Receiver/ Transmitter (UART) Controller #2
4BC0	0	24	0	Intel® Programmable Services Engine (Intel® PSE): I^2C Controller #7
4BC1	0	24	1	Intel® PSE: Controller Area Network (CAN) Controller #0
4BC2	0	24	2	Intel® PSE: CAN Controller #1
4BC3	0	24	3	Intel® PSE: Quadrature Encoder Peripheral, (QEP) Controller #0
4B81	0	24	4	Intel® PSE: QEP Controller #1
4B82	0	24	5	Intel® PSE: QEP Controller #2
4B83	0	24	6	Intel® PSE: QEP Controller #3
4B60	0	23	0	SATA Controller (AHCI)
4B62	0	23	0	Reserved
4B63	0	23	0	Reserved
4B70	0	22	0	Intel® Converged Security Engine (Intel® CSE): Host Embedded Controller Interface (HECI) #0
4B71	0	22	1	Intel® CSE: HECI #1
4B74	0	22	4	Intel® CSE: HECI #2
4B75	0	22	5	Intel® CSE: HECI #3
4B76	0	22	7	Reserved
4B78	0	21	0	Intel® Serial I/O: Inter-Integrated Circuit (I^2C) Controller #0
4B79	0	21	1	Intel® Serial I/O: I^2C Controller #1

Table 30-1. PCH Global Device IDs (Sheet 4 of 5)

Device ID	Bus#	Device#	Function#	Description
4B7A	0	21	2	Intel® Serial I/O: I2C Controller #2
4B7B	0	21	3	Intel® Serial I/O: I2C Controller #3
4B7C	0	21	7	Reserved
4B7D	0	20	0	USB eXtensible Host Controller Interface (xHCI)
4B7E	0	20	1	USB eXtensible Device Controller Interface (xDCI)
4B7F	0	20	2	Reserved
4B2C	0	20	3	Reserved
4B33	0	20	3	Reserved
4B34	0	20	3	Reserved
4B35	0	20	3	Reserved
4B36	0	20	3	Reserved
4B6A	0	20	4	Reserved
4B84	0	19	0	Intel® PSE: Serial Peripheral Interface (SPI) Controller #0
4B85	0	19	1	Intel® PSE: SPI Controller #1
4B86	0	19	2	Intel® PSE: SPI Controller #2
4B87	0	19	3	Intel® PSE: SPI Controller #3
4B88	0	19	4	Intel® PSE: General Purpose Input Output (GPIO) Controller #0
4B89	0	19	5	Intel® Programmable Services Engine (Intel® PSE): GPIO Controller #1
4B37	0	18	0	Intel® Serial I/O: SPI Controller #2
4B6C	0	18	3	Intel® Converged Security Engine (Intel® CSE): UMA Access
4B40	0	18	4	Reserved
4B6B	0	18	4	Intel® CSE: Intel® PTT DMA Controller
4B41	0	18	5	Reserved
4B42	0	18	6	Reserved
4B43	0	18	7	Reserved
4B96	0	17	0	Intel® PSE: Universal Asynchronous Receiver/ Transmitter (UART) Controller #0
4B97	0	17	1	Intel® PSE: UART Controller #1
4B98	0	17	2	Intel® PSE: UART Controller #2
4B99	0	17	3	Intel® PSE: UART Controller #3
4B9A	0	17	4	Intel® PSE: UART Controller #4
4B9B	0	17	5	Intel® PSE: UART Controller #5
4B9C	0	17	6	Intel® PSE: Inter-Integrated Circuit Sound (I2S) Controller #0
4B9D	0	17	7	Intel® PSE: I2S Controller #1
4B44	0	16	0	Intel® Serial I/O: Inter-Integrated Circuit (I2C) Controller #6
4B45	0	16	1	Intel® Serial I/O: I2C Controller #7
4B46	0	16	2	Reserved

Table 30-1. PCH Global Device IDs (Sheet 5 of 5)

Device ID	Bus#	Device#	Function#	Description
4B4E	0	16	2	Reserved
4B4F	0	16	2	Reserved
4B50	0	16	2	Reserved
4B2F	0	16	5	Integrated Error Handler (IEH)
4B5D	0	16	6	Reserved
4B5E	0	16	7	Reserved
4B64				Reserved
4B65				Reserved
4B66				Reserved
4B67				Reserved

30.3 PCH ACPI IDs

Table 30-2 lists the assigned ACPI IDs for the Platform Controller Hub (PCH). The ACPI IDs comprise of ACPI ID and Description.

Table 30-2. ACPI IDs

ACPI ID	Description
INTC1020	General Purpose Input Output (GPIO) Controller
INTC1021-1022	Timed General Purpose Input Output (GPIO) Controller

30.4 Compute Die Global Device ID

Table 30-3 lists the Global Device IDs for the Compute Die. The Global Device IDs comprise of PCI Device ID, Device number, Function number and Description.

Table 30-3. Compute Die Global Device ID (Sheet 1 of 3)

Device ID	Bus#	Device#	Function#	Description
4502	0	0	0	Reserved
4504	0	0	0	Reserved
4506	0	0	0	Reserved
4508	0	0	0	Reserved
450A	0	0	0	Reserved
450C	0	0	0	Reserved
450E	0	0	0	Reserved
4510	0	0	0	Reserved
4512	0	0	0	Processor Transaction Router (Processor SKU 3)
451E	0	0	0	Processor Transaction Router (Processor SKU 3A)

Table 30-3. Compute Die Global Device ID (Sheet 2 of 3)

Device ID	Bus#	Device#	Function#	Description
4514	0	0	0	Processor Transaction Router (Processor SKU 5)
4516	0	0	0	Processor Transaction Router (Processor SKU 8/8PU)
4518	0	0	0	Processor Transaction Router (Processor SKU 12)
451A	0	0	0	Reserved
4520	0	0	0	Reserved
4522	0	0	0	Processor Transaction Router (Processor SKU 1)
4538	0	0	0	Processor Transaction Router (Processor SKU 1A)
4524	0	0	0	Reserved
4526	0	0	0	Processor Transaction Router (Processor SKU 4)
4528	0	0	0	Processor Transaction Router (Processor SKU 6)
452A	0	0	0	Processor Transaction Router (Processor SKU 7)
452C	0	0	0	Processor Transaction Router (Processor SKU 9/9PU)
452E	0	0	0	Processor Transaction Router (Processor SKU 10)
4530	0	0	0	Reserved
4532	0	0	0	Processor Transaction Router (Processor SKU 11)
4534	0	0	0	Reserved
4536	0	0	0	Reserved
453A	0	0	0	Processor Transaction Router (Processor SKU 2)
4540	0	2	0	Reserved
4541	0	2	0	Reserved
4550	0	2	0	Reserved
4555	0	2	0	GPU (16 Execution Unit (EU) SKU)
4560-456F	0	2	0	Reserved
4570	0	2	0	GPU (32 Execution Unit (EU) Super SKU)
4571	0	2	0	GPU (32 Execution Unit (EU) SKU)

Table 30-3. Compute Die Global Device ID (Sheet 3 of 3)

Device ID	Bus#	Device#	Function#	Description
4503	0	4	0	Dynamic Platform and Thermal Framework (DPTF)
4511	0	8	0	Gaussian Mixture Model and Neural Network Accelerator (GNA)
4529	0	9	0	Intel® Trace Hub

§ §

31 Processor Ball Map and Pin Location

Table 31-1. Processor Ball Names (Sheet 1 of 39)

Ball Number	Ball Name
A100	VSS
A103	JTAG_PREQ_N
A105	VSS
A108	VSS
A110	RSVD
A112	RSVD
A12	RSVD
A15	USB2_7_DN
A17	VSS
A20	USB2_3_DP
A23	USB2_1_DP
A26	VSS
A28	USB2_4_DP
A31	USB2_6_DP
A34	VSS
A36	USB3_1_TXP
A39	USB3_1_RXN
A4	RSVD
A41	USB3_0_RXP
A44	PCIE_0_RXP/USB3_2_RXP
A47	PCIE_0_TXN/USB3_2_TXN
A50	PCIE_1_TXP/USB3_3_TXP
A53	PCIE_2_TXP
A55	PCIE_4_TXN
A58	PCIE_5_TXP/PSE_GBE0_SGMII_TXP
A6	RSVD
A61	PCIE_7_TXN/PSE_GBE1_SGMII_TXN
A64	PCIE_8_TXP/SATA_0_TXP/GBE_SGMII_TXP
A66	RSVD
A69	VSS
A72	VCCIN_AUX
A75	VCCIN_AUX
A78	VCC_IN_SFR
A80	VCC_OUT_STG
A83	VSS

Table 31-1. Processor Ball Names (Sheet 2 of 39)

Ball Number	Ball Name
A86	RSVD
A89	RSVD
A9	VSS
A92	VSS
A94	PCH_JTAG_TMS
A97	PCH_JTAG_TRST_N
AA107	VCCIN
AA109	VCCIN
AA111	VCCIN
AA113	VCCIN
AA2	GP_U02/GBE_PPS/PSE_I2C7_SCL
AA4	GP_U12/ISI_CHX_OKNOK_0
AA6	PMC_SYS_PWROK
AB102	VCCIN
AB104	VCCIN
AB106	VCCIN
AB11	VSS
AB110	VCCIN
AB14	VSS
AB18	GP_C21/PSE_UART4_TXD/SIO_UART2_TXD
AB22	VSS
AB25	VSS
AB29	VSS
AB32	RSVD
AB36	RSVD
AB39	PROC_OPIRCOMP
AB43	VSS
AB46	GP_A06/PSE_GBE0_RGMII_RXCLK
AB50	GP_A05/PSE_GBE0_RGMII_TXCTL
AB53	GP_A02/PSE_GBE0_RGMII_TXD1
AB58	GP_A00/PSE_GBE0_RGMII_TXD3
AB61	VCC_OUT_FET_1P05A
AB65	VSS
AB69	VSS
AB72	VSS
AB76	RSVD
AB8	VSS
AB80	VSS
AB83	VSS
AB86	VCCIN

Table 31-1. Processor Ball Names (Sheet 3 of 39)

Ball Number	Ball Name
AB89	VCCIN
AB91	VCCIN
AB93	VCCIN
AB95	VCCIN
AB97	VCCIN
AB99	VCCIN
AC1	VCCA_CLKLDO_1P8
AC108	VCCIN
AC112	VCCIN
AC3	GP_U03/GBE_AUXTS/PSE_I2C7_SDA
AC63	RSVD
AC67	VSS
AC71	VSS
AC74	VSS
AC78	VSS
AD11	GP_U15/ISI_CHX_PMIC_EN/PSE_TGPIO13
AD13	GP_U09/ISI_SPIS_SCLK/ISI_I2CS_SCL/ PSE_TGPIO11
AD15	GP_U11/PSE_QEPB3/PSE_TGPIO11
AD17	VSS
AD19	GP_U08/ISI_SPIS_CS/PSE_TGPIO10
AD2	VSS
AD21	GP_U01/GBE_RST_N/PSE_I2C6_SDA
AD24	GP_U18/ISI_ALERT_N
AD26	GP_U17/ISI_OKNOK_1
AD28	GP_U19/PSE_QEPI3/PSE_TGPIO12
AD30	VSS
AD32	VCC_1P8A
AD35	VSS
AD37	RSVD
AD4	GP_U00/GBE_INT/PSE_I2C6_SCL
AD40	RSVD
AD43	VSS
AD46	VCC_OUT_FIVR_1P05A
AD48	VSS
AD50	VCCIO
AD53	VCCIO
AD56	VCCIO
AD59	VSS
AD61	RSVD
AD7	GP_U16/ISI_OKNOK_0

Table 31-1. Processor Ball Names (Sheet 4 of 39)

Ball Number	Ball Name
AD80	VCCIN
AD82	VCCIN
AD85	VCCIN
AD88	VCCIN
AD9	GP_U14/ISI_CHX_RLY_SWTCH
AE1	GP_U06/ISI_SPIM_MISO/PSE_SPI1_MISO
AE102	VCCIN
AE104	VCCIN
AE106	VCCIN
AE110	VCCIN
AE113	VCCIN
AE3	GP_U10/ISI_SPIS_MISO/ISI_I2CS_SDA/ PSE_TGPIO12
AE69	VSS
AE72	RSVD
AE75	RSVD
AE77	VSS
AE84	VCCIN
AE86	VCCIN
AE89	VCCIN
AE91	VCCIN
AE93	VCCIN
AE95	VCCIN
AE97	VCCIN
AE99	VCCIN
AF32	VCC_1P8A
AF35	VCC_1P8A
AF37	VSS
AF40	RSVD
AF43	VSS
AF46	VCC_OUT_FIVR_1P05A
AF48	VCC_OUT_FIVR_1P05A
AF50	VSS
AF53	VCCIO
AG101	VCCIN
AG103	VCCIN
AG108	VCCIN
AG110	VCCIN
AG112	VCCIN
AG2	GP_D14/PSE_QEPB1/PSE_TGPIO38
AG4	GP_D08/PCIE_CLKREQ3_N

Table 31-1. Processor Ball Names (Sheet 5 of 39)

Ball Number	Ball Name
AG56	VCCIO
AG59	VSS
AG61	RSVD
AG69	RSVD
AG72	VSS
AG81	VCCIN
AG84	VCCIN
AG86	VCCIN
AG89	VCCIN
AG94	VCCIN
AG96	VCCIN
AH1	RSVD
AH102	VSS
AH104	VSS
AH106	VCCIN
AH3	GP_D17/PSE_PWM04/ISI_SPIM_MOSI/ PSE_TGPIO41
AH91	VSS
AH93	VSS
AH95	VSS
AH97	VSS
AH99	VSS
AJ107	VSS
AJ109	VSS
AJ11	GP_D11/PSE_SPI0_MISO/SIO_SPI2_MISO/ PSE_TGPIO12
AJ13	GP_D10/PSE_SPI0_CLK/SIO_SPI2_CLK/ PSE_TGPIO11
AJ15	GP_D09/PSE_SPI0_CS0_N/SIO_SPI2_CS0_N/ PSE_TGPIO10
AJ17	VSS
AJ19	GP_D04/PSE_PWM02/PSE_SPI1_CS1_N/ PSE_TGPIO36
AJ21	GP_D02/PSE_QEPI0/PSE_SPI1_MISO/ PSE_TGPIO34
AJ24	GP_D03/PSE_PWM06/PSE_SPI1_MOSI/ PSE_TGPIO35
AJ26	GP_D00/PSE_QEPA0/PSE_SPI1_CS0_N/ PSE_TGPIO32
AJ28	VSS
AJ30	VCC_1P8A
AJ32	VCC_1P8A
AJ35	VCC_1P8A
AJ37	VCC_1P8A

Table 31-1. Processor Ball Names (Sheet 6 of 39)

Ball Number	Ball Name
AJ40	VSS
AJ43	VCC_OUT_1P05A
AJ46	VCC_OUT_1P05A
AJ48	VCC_OUT_FIVR_1P05A
AJ50	VCC_OUT_FIVR_1P05A
AJ53	VSS
AJ56	VCCIO
AJ59	VCCIO
AJ61	RSVD
AJ69	VSS
AJ7	GP_D18/PSE_PWM05/ISI_SPIS_MOSI/ PSE_TGPIO42
AJ72	VSS
AJ81	VSS
AJ84	VSS
AJ86	VCCIN
AJ89	VCCIN
AJ9	GP_D12/PSE_SPI0_MOSI/SIO_SPI2_MOSI/ PSE_TGPIO13
AK111	VSS
AK113	VSS
AK2	GP_D13/PSE_QEPA1/PSE_TGPIO37
AK4	VSS
AK75	VSS
AK77	VCCIN
AL1	GP_D16/PSE_QEPI1/PSE_TGPIO40
AL106	LP4_0_DQ15/DDR_0_DQ15
AL110	LP4_0_DQ13/DDR_0_DQ13
AL3	GP_D06/PCIE_CLKREQ1_N
AL32	VCC_1P8A
AL35	VCC_1P8A
AL37	VCC_1P8A
AL40	VSS
AL43	RSVD
AL46	VCC_OUT_1P05A
AL48	VCC_OUT_FIVR_1P05A
AL50	VCC_OUT_FIVR_1P05A
AL53	VSS
AL56	VCCIO
AL59	VCCIO
AL61	RSVD

Table 31-1. Processor Ball Names (Sheet 7 of 39)

Ball Number	Ball Name
AL64	VSS
AL66	VSS
AL69	VSS
AL72	VCCIN
AL81	VCCIN
AL84	VCCIN
AL86	VCCIN
AL89	VCCIN
AM102	LP4_0_DQ05/DDR_0_DQ05
AM104	VSS
AM108	LP4_0_DQ14/DDR_0_DQ14
AM112	LP4_0_DQ11/DDR_0_DQ11
AM2	GP_D05/PCIE_CLKREQ0_N
AM4	GP_D19/AVS_I2S_MCLK1/PSE_TGPIO43
AM75	RSVD
AM77	VCCIN
AM91	VSS
AM93	LP4_0_DQ00/DDR_0_DQ00
AM95	LP4_0_DQ01/DDR_0_DQ01
AM97	LP4_0_DQS0_DN/DDR_0_DQS0_DN
AM99	LP4_0_DQ03/DDR_0_DQ03
AN11	GP_H19/DDI2_DDC_SDA/PMC_TGPIO0/ PSE_TGPIO20
AN13	GP_H15/PSE_UART1_CTS_N/M2_SKT2_CFG3/ PSE_TGPIO54
AN15	GP_H11/PCIE_CLKREQ5_N/PSE_PWM15
AN17	VSS
AN19	GP_H22/PSE_HSUART1_RE/PSE_TGPIO57
AN21	GP_H20/PSE_PWM07/DDI2_HPD/PSE_TGPIO55
AN24	GP_H17/SD_SDIO_PWR_EN_N
AN26	GP_H23/PSE_HSUART1_EN/PSE_TGPIO58
AN28	VSS
AN30	VCC_3P3A
AN32	VSS
AN35	VCC_1P8A
AN37	VCC_1P8A
AN39	VSS
AN41	VCCIN_AUX
AN43	VCCIN_AUX
AN45	VCCIN_AUX
AN47	VSS

Table 31-1. Processor Ball Names (Sheet 8 of 39)

Ball Number	Ball Name
AN49	VCCIN_AUX
AN52	VCCIN_AUX
AN54	VCCIN_AUX
AN56	VSS
AN59	VSS
AN61	RSVD_NCTF
AN64	VSS
AN7	GP_D01/PSE_QEPB0/PSE_SPI1_CLK/PSE_TGPIO33
AN72	VCCIN
AN74	VCCIN
AN81	VCCIN
AN84	VCCIN
AN86	VSS
AN89	VCCIN
AN9	GP_H18/PMC_CPU_C10_GATE_N
AP1	VCC_OUT_1P24A
AP106	VSS
AP110	LP4_0_DQS1_DN/DDR_0_DQS1_DN
AP113	VSS
AP3	VSS
AP36	VSS
AR102	LP4_0_DQ07/DDR_0_DQ07
AR104	VSS
AR108	LP4_0_DQS1_DP/DDR_0_DQS1_DP
AR2	GP_D15/PSE_PWM03/SIO_SPI2_CS1_N/ PSE_SPI0_CS1_N/PSE_TGPIO39
AR32	VCC_3P3A
AR35	VCC_3P3A
AR4	GP_D07/PCIE_CLKREQ2_N
AR49	VCCIN_AUX
AR52	VCCIN_AUX
AR54	VCCIN_AUX
AR56	VCCIN_AUX
AR59	VCCIN_AUX_VCCSENSE
AR61	RSVD_NCTF
AR64	VSS
AR72	VCCIN
AR74	VCCIN
AR81	VCCIN
AR84	VCCIN
AR86	VSS

Table 31-1. Processor Ball Names (Sheet 9 of 39)

Ball Number	Ball Name
AR89	VCCIN_SENSE
AR91	VSS
AR93	LP4_0_DQ02/DDR_0_DQ02
AR95	LP4_0_DQ04/DDR_0_DQ04
AR97	LP4_0_DQS0_DP/DDR_0_DQS0_DP
AR99	LP4_0_DQ06/DDR_0_DQ06
AT1	GP_V14/PSE_TGPIO02
AT106	LP4_0_DQ10/DDR_0_DQ10
AT110	LP4_0_DQ08/DDR_0_DQ08
AT112	LP4_0_DQ12/DDR_0_DQ12
AT3	GP_V12/PSE_TGPIO00
AT36	VCC_3P3A
AU11	GP_H14/M2_SKT2_CFG2/PSE_TGPIO53
AU13	GP_H12/PSE_UART1_RXD/M2_SKT2_CFG0/ PSE_TGPIO51
AU15	VSS
AU17	GP_H13/PSE_UART1_TXD/M2_SKT2_CFG1/ PSE_TGPIO52
AU19	GP_H10/PCIE_CLKREQ4_N/PSE_PWM14
AU21	GP_H09/SIO_I2C4_SCL/PSE_PWM13
AU24	GP_H08/SIO_I2C4_SDA/PSE_PWM12
AU26	VSS
AU28	VCC_3P3A
AU30	VCC_3P3A
AU7	GP_H21/PSE_HSUART1_DE/PSE_UART1_RTS_N/ PSE_TGPIO56
AU9	GP_H16/PCIE_LNK_DOWN/DDI2_DDC_SCL
AV102	VSS
AV104	VSS
AV107	VSS
AV109	VSS
AV111	LP4_0_DQ09/DDR_0_DQ09
AV113	VSS
AV2	GP_V11/EMMC_RST_N
AV32	VCC_3P3A
AV35	VCC_3P3A
AV4	GP_V13/PSE_TGPIO01
AV49	VCCIN_AUX
AV52	VCCIN_AUX
AV54	VCCIN_AUX
AV56	VCCIN_AUX
AV59	VCCIN_AUX_VSSSENSE

Table 31-1. Processor Ball Names (Sheet 10 of 39)

Ball Number	Ball Name
AV61	RSVD
AV64	VSS
AV72	VCCIN
AV74	RSVD
AV81	VSS
AV84	VCCIN
AV86	VCCIN
AV89	VSSIN_SENSE
AV91	VSS
AV93	VSS
AV95	VSS
AV97	VSS
AV99	VSS
AW1	VCC_RTC_3P3
AW106	LP4_0_DQ31/DDR_0_DQ31
AW3	VSS
AY108	LP4_0_DQ30/DDR_0_DQ30
AY110	LP4_0_DQ29/DDR_0_DQ29
AY32	VCC_3P3A
AY35	VCC_3P3A
AY37	VCC_BYP_1P05
AY46	VCCIN_AUX
AY49	VCCIN_AUX
AY52	VCCIN_AUX
AY54	VCCIN_AUX
AY56	VCCIN_AUX
AY59	VSS
AY61	RSVD
AY64	VSS
AY72	VCCIN
AY74	VSS
AY81	RSVD
AY84	VCCIN
AY86	VCCIN
AY89	RSVD
B10	HSIO_RCOMPP
B101	CPU_JTAG_TRST_N
B105	VSS
B107	SVID_ALERT_N
B109	RSVD

Table 31-1. Processor Ball Names (Sheet 11 of 39)

Ball Number	Ball Name
B111	RSVD
B113	RSVD
B13	VSS
B16	USB2_9_DN
B19	USB2_5_DP
B2	RSVD
B22	VSS
B24	USB2_0_DP
B27	USB2_2_DP
B30	VSS
B33	USB2_8_DP
B35	USB3_1_TXN
B37	USB3_0_TXP
B4	VSS
B40	USB3_0_RXN
B43	PCIE_0_RXN/USB3_2_RXN
B46	PCIE_1_RXP/USB3_3_RXP
B48	PCIE_1_TXN/USB3_3_TXN
B51	PCIE_2_TXN
B54	PCIE_3_TXP
B57	PCIE_5_TXN/PSE_GBE0_SGMII_TXN
B6	RSVD
B60	PCIE_6_TXP/GBE_SGMII_TXP
B62	PCIE_8_TXN/SATA_0_TXN/GBE_SGMII_TXN
B65	PCIE_9_TXP/SATA_1_TXP/PSE_GBE1_SGMII_TXP
B68	VCCST_OVERRIDE
B71	VCCIN_AUX
B73	VCCIN_AUX
B76	VCC_OUT_FIVR_1P05A
B79	VCC_IN_ST
B8	RSVD
B82	VCC_IN_STG
B85	RSVD
B87	VSS
B90	RSVD
B93	PCH_JTAGX
B96	VSS
B98	CPU_JTAG_TDO
BA102	LP4_0_DQ21/DDR_0_DQ21
BA104	VSS

Table 31-1. Processor Ball Names (Sheet 12 of 39)

Ball Number	Ball Name
BA11	GP_H05/SIO_I2C2_SCL/PSE_PWM09/ PSE_TGPIO11
BA112	LP4_0_DQ27/DDR_0_DQ27
BA13	GP_H04/SIO_I2C2_SDA/PSE_PWM08/ PSE_TGPIO10
BA15	VSS
BA17	GP_H03/PSE_GBE1_PPS/PSE_UART5_CTS_N/ PSE_TGPIO21
BA19	GP_H02/PSE_GBE1_AUXTS/PSE_UART5_RTS_N
BA2	GP_V04/EMMC_DATA3
BA21	GP_H01/PSE_GBE1_RST_N/PSE_UART5_TXD
BA24	GP_H00/PSE_GBE1_INT/PSE_UART5_RXD
BA26	VSS
BA28	VCC_PGPPR
BA30	VCC_PGPPR
BA4	GP_V15/PSE_TGPIO03
BA7	GP_H06/SIO_I2C3_SDA/PSE_I2C5_SDA/ PSE_PWM10
BA9	GP_H07/SIO_I2C3_SCL/PSE_I2C5_SCL/ PSE_PWM11
BA91	VSS
BA93	LP4_0_DQ16/DDR_0_DQ16
BA95	LP4_0_DQ17/DDR_0_DQ17
BA97	LP4_0_DQS2_DN/DDR_0_DQS2_DN
BA99	LP4_0_DQ19/DDR_0_DQ19
BB1	VSS
BB106	VSS
BB110	LP4_0_DQS3_DN/DDR_0_DQS3_DN
BB113	VSS
BB3	GP_V01/EMMC_DATA0
BB32	VSS
BB35	VCC_3P3A
BB37	VCC_3P3A
BB46	VSS
BB49	VCCIN_AUX
BB52	VSS
BB54	VCCIN_AUX
BB56	VCCIN_AUX
BB59	VSS
BB61	VSS
BB64	VSS
BB72	RSVD
BB74	VCCIN

Table 31-1. Processor Ball Names (Sheet 13 of 39)

Ball Number	Ball Name
BB81	RSVD
BB84	VCCIN
BB86	VSS
BB89	VSS
BC2	GP_V07/EMMC_DATA6
BC4	GP_V08/EMMC_DATA7
BD102	LP4_0_DQ22/DDR_0_DQ22
BD104	LP4_0_DQ23/DDR_0_DQ23
BD108	LP4_0_DQS3_DP/DDR_0_DQS3_DP
BD110	LP4_0_DQ24/DDR_0_DQ24
BD112	LP4_0_DQ28/DDR_0_DQ28
BD32	VSS
BD35	VCC_BYP_VNN
BD37	VSS
BD46	VCCIN_AUX
BD49	VCCIN_AUX
BD52	VCCIN_AUX
BD54	VCCIN_AUX
BD56	VCC_IN_STG
BD59	VCC_IN_STG
BD61	VCC_IN_STG
BD64	VCC_IN_STG
BD67	VSS
BD70	VSS
BD73	RSVD
BD76	VSS
BD79	VCCIN
BD81	VCCIN
BD84	VCCIN
BD86	VSS
BD89	DDR_1_VREF_CA
BD91	VSS
BD93	LP4_0_DQ18/DDR_0_DQ18
BD95	LP4_0_DQ20/DDR_0_DQ20
BD97	LP4_0_DQS2_DP/DDR_0_DQS2_DP
BD99	VSS
BE1	GP_V02/EMMC_DATA1
BE106	LP4_0_DQ26/DDR_0_DQ26
BE3	GP_V05/EMMC_DATA4
BF102	VSS

Table 31-1. Processor Ball Names (Sheet 14 of 39)

Ball Number	Ball Name
BF104	VSS
BF107	VSS
BF109	VSS
BF11	GP_DSW09
BF13	GP_DSW08/PMC_SUSCLK
BF15	GP_DSW07
BF17	VSS
BF19	GP_DSW05/PMC_SLP_S4_N
BF2	GP_V10/EMMC_CLK
BF21	GP_DSW04/PMC_SLP_S3_N
BF24	RSVD
BF26	RSVD
BF28	VCC_3P3A_DSW
BF30	VCC_3P3A_DSW
BF32	VCC_OUT_1P05A
BF35	VCC_BYP_VNN
BF37	VSS
BF4	VSS
BF40	RSVD
BF43	RSVD
BF46	VCCIN_AUX
BF49	VCCIN_AUX
BF52	VCCIN_AUX
BF54	VCCIN_AUX
BF56	VSS
BF59	RSVD
BF61	VCC_IN_STG
BF64	VCC_IN_STG
BF67	VSS
BF7	GP_DSW11
BF70	VSS
BF73	VSS
BF76	VCCIN
BF79	VCCIN
BF81	VCCIN
BF84	VCCIN
BF86	VCCIN
BF89	DDR_0_VREF_CA
BF9	GP_DSW10/PMC_SLP_S5_N
BF91	VSS

Table 31-1. Processor Ball Names (Sheet 15 of 39)

Ball Number	Ball Name
BF93	VSS
BF95	VSS
BF97	VSS
BF99	VSS
BG111	LP4_0_DQ25/DDR_0_DQ25
BG113	VSS
BH1	GP_V00/EMMC_CMD
BH106	LP4_0_CKE0/DDR_0_CKE0
BH110	LP4_0_CKE1
BH3	GP_V03/EMMC_DATA2
BH32	VCC_OUT_1P05A
BH35	VSS
BH37	VSS
BH40	VSS
BH43	VSS
BH46	VSS
BH49	VCCIN_AUX
BH52	VCCIN_AUX
BH54	VSS
BH56	VSS
BH59	RSVD
BH61	VCC_IN_STG
BH64	VCC_IN_STG
BH67	VSS
BH70	VSS
BH73	VSS
BH76	VSS
BH79	VSS
BH81	VSS
BH84	VSS
BH86	VSS
BH89	VSS
BJ108	LP4_0_CA0/DDR_0_MA05
BJ112	LP4_0_CA1/DDR_0_MA09
BJ12	VSS
BJ14	VSS
BJ16	VSS
BJ18	VSS
BJ2	VSS
BJ20	VSS

Table 31-1. Processor Ball Names (Sheet 16 of 39)

Ball Number	Ball Name
BJ22	VSS
BJ24	VSS
BJ27	VSS
BJ4	GP_V09/EMMC_RCLK
BJ7	VSS
BJ9	VSS
BK1	VCCDSW_OUT_1P05
BK10	GP_DSW03/PMC_PWRBTN_N
BK102	LP4_1_CKE0
BK104	VSS
BK13	GP_DSW02
BK15	VSS
BK17	GP_DSW01/PMC_ACPRESENT
BK19	GP_DSW00/PMC_BATLOW_N
BK21	VSS
BK3	GP_V06/EMMC_DATA5
BK30	VCC_OUT_1P05A
BK32	RSVD
BK35	RSVD
BK38	RSVD
BK41	VSS
BK44	VSS
BK48	VCC_IN_FUSE_V1P05A
BK51	VCC_IN_FUSE_V1P05A
BK54	VDDQ
BK57	VDDQ
BK6	FSPI_CS0_N
BK60	VDDQ
BK63	VDDQ
BK66	VDDQ
BK68	VDDQ
BK72	VDDQ
BK74	VDDQ
BK77	VDDQ
BK8	FSPI_CS1_N
BK80	VDDQ
BK83	VDDQ
BK86	VDDQ
BK89	VDDQ
BK91	VSS

Table 31-1. Processor Ball Names (Sheet 17 of 39)

Ball Number	Ball Name
BK93	LP4_1_CS0
BK95	LP4_1_CS1/DDR_0_CS1_N
BK97	LP4_1_CA0/DDR_0_MA13
BK99	LP4_1_CA1/DDR_0_MA15_CAS_N
BL106	VSS
BL110	VSS
BL113	VSS
BL24	VSS
BL27	VSS
BM108	DDR_0_BG1
BM2	RSVD
BM30	VCC_OUT_1P05A
BM32	RSVD
BM35	RSVD
BM38	RSVD
BM4	EMMC_RCOMP
BM41	VSS
BM44	PROC_POPIRCOMP
BM48	VCC_IN_FUSE_V1P05A
BM51	VCC_IN_FUSE_V1P05A
BM54	VDDQ
BM57	VDDQ
BM60	VDDQ
BM63	VDDQ
BM66	VDDQ
BM68	VDDQ
BM72	VDDQ
BM74	VDDQ
BM77	VDDQ
BM80	VDDQ
BM83	VDDQ
BM86	VDDQ
BM89	VDDQ
BN1	RSVD
BN102	LP4_1_CKE1/DDR_0_CKE1
BN104	VSS
BN106	DDR_0_ACT_N
BN110	DDR_0_MA11
BN112	DDR_0_ALERT_N
BN24	VSS

Table 31-1. Processor Ball Names (Sheet 18 of 39)

Ball Number	Ball Name
BN27	GP_T04/PSE_GBE0_INT
BN3	RSVD
BN91	VSS
BN93	LP4_1_CA4/DDR_0_BA0
BN95	LP4_1_CA5/DDR_0_MA02
BN97	LP4_1_CA3/DDR_0_MA16_RAS_N
BN99	LP4_1_CA2/DDR_0_MA14_WE_N
BP10	FSPI_MISO_IO1
BP13	FSPI_IO2
BP15	VSS
BP17	FSPI_IO3
BP19	FSPI_CS2_N
BP21	GP_R00/HDA_BCLK/AVS_I2S0_SCLK/ PSE_I2S0_SCLK
BP30	VSS
BP32	VSS
BP35	VSS
BP38	VSS
BP41	VSS
BP44	VSS
BP48	VSS
BP51	VSS
BP54	VDDQ
BP57	VSS
BP6	FSPI_CLK
BP60	VSS
BP63	VSS
BP67	VSS
BP70	VSS
BP73	VSS
BP76	VSS
BP8	FSPI_MOSI_IO0
BP80	VSS
BP83	VSS
BP86	VSS
BR100	VSS
BR104	VSS
BR107	VSS
BR109	DDR_0_MA12
BR111	DDR_0_MA04
BR2	RSVD

Table 31-1. Processor Ball Names (Sheet 19 of 39)

Ball Number	Ball Name
BR24	GP_T07/PSE_GBE0_PPS/PSE_TGPIO59
BR27	GP_T11/USB2_OC3_N/PSE_TGPIO06
BR4	VSS
BR91	VSS
BR94	VSS
BR97	VSS
BT1	VSS
BT102	LP4_1_DQ02/DDR_0_DQ34
BT113	VSS
BT3	RSVD
BT30	VSS
BT32	LP4_3_DQ23/DDR_1_DQ55
BT35	LP4_3_DQ18/DDR_1_DQ50
BT38	VSS
BT41	LP4_3_DQ07/DDR_1_DQ39
BT44	LP4_3_DQ02/DDR_1_DQ34
BT48	VSS
BT51	VSS
BT54	VSS
BT57	VSS
BT60	LP4_3_CA1/DDR_1_MA15_CAS_N
BT63	DDR_1_ACT_N
BT67	VSS
BT70	LP4_2_DQ23/DDR_1_DQ23
BT73	LP4_2_DQ18/DDR_1_DQ18
BT76	VSS
BT80	LP4_2_DQ07/DDR_1_DQ07
BT83	LP4_2_DQ02/DDR_1_DQ02
BT86	VSS
BT89	LP4_1_DQ23/DDR_0_DQ55
BT92	LP4_1_DQ18/DDR_0_DQ50
BT96	VSS
BT99	LP4_1_DQ07/DDR_0_DQ39
BU106	DDR_0_MA01
BU108	DDR_0_MA00
BU110	DDR_0_MA03
BU2	RSVD
BU24	GP_T06/PSE_GBE0_AUXTS/USB2_OC1_N
BU27	GP_T15/PSE_UART2_CTS_N/SIO_UART0_CTS_N
BU4	RSVD

Table 31-1. Processor Ball Names (Sheet 20 of 39)

Ball Number	Ball Name
BV10	GP_R05/HDA_SDI1/AVS_I2S1_RXD/DMIC_DATA1
BV102	LP4_1_DQ00/DDR_0_DQ32
BV112	DDR_0_MA10
BV13	GP_R06/AVS_I2S1_TXD/DMIC_CLK_A0
BV15	VSS
BV17	GP_R04/HDA_RST_N/DMIC_CLK_A1
BV19	GP_R07/AVS_I2S1_SFRM/DMIC_DATA0
BV21	GP_R01/HDA_SYNC/AVS_I2S0_SFRM/ PSE_I2S0_SFRM
BV30	VSS
BV32	LP4_3_DQ22/DDR_1_DQ54
BV35	LP4_3_DQ16/DDR_1_DQ48
BV38	VSS
BV41	LP4_3_DQ06/DDR_1_DQ38
BV44	LP4_3_DQ00/DDR_1_DQ32
BV48	VSS
BV51	LP4_3_CA4/DDR_1_BA0
BV54	LP4_3_CKE0
BV57	VSS
BV6	GP_R03/HDA_SDI0/AVS_I2S0_RXD/ PSE_I2S0_RXD/DMIC_CLK_B1
BV60	LP4_3_CA0/DDR_1_MA13
BV63	DDR_1_ALERT_N
BV67	VSS
BV70	LP4_2_DQ22/DDR_1_DQ22
BV73	LP4_2_DQ16/DDR_1_DQ16
BV76	VSS
BV8	GP_R02/HDA_SDO/AVS_I2S0_TXD/ PSE_I2S0_TXD/DMIC_CLK_B0
BV80	LP4_2_DQ06/DDR_1_DQ06
BV83	LP4_2_DQ00/DDR_1_DQ00
BV86	VSS
BV89	LP4_1_DQ22/DDR_0_DQ54
BV92	LP4_1_DQ16/DDR_0_DQ48
BV96	VSS
BV99	LP4_1_DQ06/DDR_0_DQ38
BW1	RSVD
BW113	VSS
BW24	VSS
BW27	VSS
BW3	RSVD
BY102	LP4_1_DQS0_DN/DDR_0_DQS4_DN

Table 31-1. Processor Ball Names (Sheet 21 of 39)

Ball Number	Ball Name
BY106	VSS
BY110	VSS
BY2	RSVD
BY30	VSS
BY32	LP4_3_DQS2_DP/DDR_1_DQS6_DP
BY35	LP4_3_DQS2_DN/DDR_1_DQS6_DN
BY38	VSS
BY4	VSS
BY41	LP4_3_DQS0_DP/DDR_1_DQS4_DP
BY44	LP4_3_DQS0_DN/DDR_1_DQS4_DN
BY48	VSS
BY51	DDR_1_ODT0
BY54	DDR_1_ODT1
BY57	VSS
BY60	DDR_1_MA10
BY63	DDR_1_MA12
BY67	VSS
BY70	LP4_2_DQS2_DP/DDR_1_DQS2_DP
BY73	LP4_2_DQS2_DN/DDR_1_DQS2_DN
BY76	VSS
BY80	LP4_2_DQS0_DP/DDR_1_DQS0_DP
BY83	LP4_2_DQS0_DN/DDR_1_DQS0_DN
BY86	VSS
BY89	LP4_1_DQS2_DP/DDR_0_DQS6_DP
BY92	LP4_1_DQS2_DN/DDR_0_DQS6_DN
BY96	VSS
BY99	LP4_1_DQS0_DP/DDR_0_DQS4_DP
C100	CPU_JTAG_TCK
C103	MDSI_DE_TE_2
C12	RSVD
C15	USB2_7_DP
C17	VSS
C20	USB2_3_DN
C23	USB2_1_DN
C26	VSS
C28	USB2_4_DN
C31	USB2_6_DN
C34	VSS
C36	USB3_0_TXN
C39	USB3_1_RXP

Table 31-1. Processor Ball Names (Sheet 22 of 39)

Ball Number	Ball Name
C41	VSS
C44	PCIE_1_RXN/USB3_3_RXN
C47	PCIE_0_TXP/USB3_2_TXP
C50	VSS
C53	PCIE_3_TXN
C55	PCIE_4_TXP
C58	PCIE_6_TXN/GBE_SGMII_TXN
C61	PCIE_7_TXP/PSE_GBE1_SGMII_TXP
C64	PCIE_9_TXN/SATA_1_TXN/PSE_GBE1_SGMII_TXN
C66	PROC_PWR_GD
C69	VCCIN_AUX
C72	VCCIN_AUX
C75	VCCIN_AUX
C78	VCC_OUT_FIVR_1P05A
C80	VCC_OUT_STG
C83	RSVD
C86	RSVD
C89	RSVD
C92	RSVD
C94	PCH_JTAG_TDI
C97	CPU_JTAG_TMS
CA10	GP_T12/PSE_UART2_RXD/SIO_UART0_RXD
CA108	LP4_0_CLK_DP/DDR_0_CLK0_DP
CA112	LP4_0_CA5/DDR_0_BG0
CA13	GP_T09/PSE_HSUART2_EN
CA15	VSS
CA17	GP_T01/PSE_QEPB2/SIO_I2C6_SCL/PSE_TGPIO09
CA19	GP_T08/USB2_OC2_N/PSE_TGPIO22
CA21	GP_T05/PSE_GBE0_RST_N
CA24	GP_T14/PSE_UART2_RTS_N/SIO_UART0_RTS_N/ PSE_HSUART2_DE
CA27	GP_T02/PSE_QEPI2/SIO_I2C7_SDA/PSE_TGPIO07
CA6	GP_T13/PSE_UART2_TXD/SIO_UART0_TXD
CA8	GP_T10/PSE_HSUART2_RE
CB1	RSVD
CB102	LP4_1_DQ01/DDR_0_DQ33
CB110	LP4_0_CLK_DN/DDR_0_CLK0_DN
CB3	RSVD
CB30	VSS
CB32	VSS
CB35	LP4_3_DQ17/DDR_1_DQ49

Table 31-1. Processor Ball Names (Sheet 23 of 39)

Ball Number	Ball Name
CB38	VSS
CB41	LP4_3_DQ05/DDR_1_DQ37
CB44	LP4_3_DQ01/DDR_1_DQ33
CB48	VSS
CB51	LP4_3_CA5/DDR_1_MA02
CB54	LP4_3_CKE1/DDR_1_CKE1
CB57	VSS
CB60	LP4_3_CS1/DDR_1_CS1_N
CB63	DDR_1_MA11
CB67	VSS
CB70	LP4_2_DQ21/DDR_1_DQ21
CB73	LP4_2_DQ17/DDR_1_DQ17
CB76	VSS
CB80	LP4_2_DQ05/DDR_1_DQ05
CB83	LP4_2_DQ01/DDR_1_DQ01
CB86	VSS
CB89	LP4_1_DQ21/DDR_0_DQ53
CB92	LP4_1_DQ17/DDR_0_DQ49
CB96	VSS
CB99	LP4_1_DQ05/DDR_0_DQ37
CC106	LP4_0_CA4/DDR_0_MA07
CC113	LP4_0_CA2/DDR_0_MA06
CC2	RSVD
CC24	GP_T03/SIO_I2C7_SCL/PSE_TGPIO06
CC27	GP_T00/PSE_QEPA2/SIO_I2C6_SDA/PSE_TGPIO08
CC4	RSVD
CD1	PMC_PCH_PWROK
CD102	LP4_1_DQ04/DDR_0_DQ36
CD108	DDR_0_ODT0
CD111	LP4_0_CA3/DDR_0_MA08
CD3	VSS
CD30	VSS
CD32	LP4_3_DQ21/DDR_1_DQ53
CD35	LP4_3_DQ20/DDR_1_DQ52
CD38	VSS
CD41	LP4_3_DQ03/DDR_1_DQ35
CD44	LP4_3_DQ04/DDR_1_DQ36
CD48	VSS
CD51	LP4_3_CA3/DDR_1_MA16_RAS_N
CD54	LP4_3_CA2/DDR_1_MA14_WE_N

Table 31-1. Processor Ball Names (Sheet 24 of 39)

Ball Number	Ball Name
CD57	VSS
CD60	LP4_3_CS0
CD63	DDR_1_BG1
CD67	VSS
CD70	LP4_2_DQ19/DDR_1_DQ19
CD73	LP4_2_DQ20/DDR_1_DQ20
CD76	VSS
CD80	LP4_2_DQ03/DDR_1_DQ03
CD83	LP4_2_DQ04/DDR_1_DQ04
CD86	VSS
CD89	LP4_1_DQ19/DDR_0_DQ51
CD92	LP4_1_DQ20/DDR_0_DQ52
CD96	VSS
CD99	LP4_1_DQ03/DDR_0_DQ35
CE10	GP_B02/PMC_VRALERT_N/ESPI_ALERT2_N/ PSE_TGPIO25
CE106	VSS
CE13	GP_B01/PMC_CORE_VID1
CE15	VSS
CE17	GP_B03/CPU_GP2/ESPI_ALERT0_N/PSE_TGPIO26
CE19	GP_B12/PMC_SLP_S0_N
CE21	GP_B22/SIO_SPI1_MOSI/PSE_SPI3_MOSI
CE24	GP_B00/PMC_CORE_VID0
CE27	GP_B20/SIO_SPI1_CLK/PSE_SPI3_CLK
CE6	GPIO_RCOMP
CE8	GP_B16/SIO_SPI0_CLK/PSE_SPI2_CLK
CF102	VSS
CF107	DDR_0_PAR
CF109	VSS
CF111	LP4_0_CS1
CF113	VSS
CF2	RTC_RST_N
CF30	VSS
CF32	LP4_3_DQ19/DDR_1_DQ51
CF35	VSS
CF38	VSS
CF4	PMC_RSMRST_N
CF41	VSS
CF44	VSS
CF48	VSS
CF51	VSS

Table 31-1. Processor Ball Names (Sheet 25 of 39)

Ball Number	Ball Name
CF54	VSS
CF57	VSS
CF60	VSS
CF63	VSS
CF67	VSS
CF70	VSS
CF73	VSS
CF76	VSS
CF80	VSS
CF83	VSS
CF86	VSS
CF89	VSS
CF92	VSS
CF96	VSS
CF99	VSS
CG1	VSS
CG105	VSS
CG24	GP_B13/PMC_PLTRST_N
CG27	GP_B19/SIO_SPI1_CS0_N/PSE_SPI3_CS0_N/ ESPI_CS2_N
CH107	VSS
CH109	LP4_1_CLK_DP/DDR_0_CLK1_DP
CH111	LP4_1_CLK_DN/DDR_0_CLK1_DN
CH113	VSS
CH2	RTC_X2
CH30	VSS
CH32	VSS
CH4	INTRUDER_N
CJ10	GP_B23/PCHHOT_N/SIO_SPI1_CS1_N/ PSE_SPI3_CS1_N/PSE_TGPIO28
CJ102	VSS
CJ104	VSS
CJ13	GP_B04/CPU_GP3/ESPI_ALERT1_N/PSE_TGPIO27
CJ15	VSS
CJ17	GP_B15/SIO_SPI0_CS0_N/PSE_SPI2_CS0_N/ ESPI_CS1_N
CJ19	GP_B21/SIO_SPI1_MISO/PSE_SPI3_MISO
CJ21	GP_B17/SIO_SPI0_MISO/PSE_SPI2_MISO
CJ24	GP_B18/SIO_SPI0_MOSI/PSE_SPI2_MOSI
CJ27	GP_B14/SPKR/PMC_TGPIO1/SIO_SPI0_CS1_N/ PSE_SPI2_CS1_N
CJ36	VSS

Table 31-1. Processor Ball Names (Sheet 26 of 39)

Ball Number	Ball Name
CJ39	LP4_3_DQ26/DDR_1_DQ58
CJ42	LP4_3_DQ31/DDR_1_DQ63
CJ44	VSS
CJ47	LP4_3_DQ10/DDR_1_DQ42
CJ50	LP4_3_DQ15/DDR_1_DQ47
CJ53	VSS
CJ56	LP4_2_CS0/DDR_1_CS0_N
CJ59	DDR_1_MA01
CJ6	GP_B10/SIO_I2C5_SCL/PSE_I2C2_SCL/ ESPI_ALERT3_N
CJ61	VSS
CJ64	LP4_2_CA4/DDR_1_MA07
CJ67	LP4_2_CKE0/DDR_1_CKE0
CJ70	VSS
CJ73	LP4_2_DQ26/DDR_1_DQ26
CJ76	LP4_2_DQ31/DDR_1_DQ31
CJ79	VSS
CJ8	GP_B09/SIO_I2C5_SDA/PSE_I2C2_SDA/ ESPI_CS3_N
CJ82	LP4_2_DQ10/DDR_1_DQ10
CJ85	LP4_2_DQ15/DDR_1_DQ15
CJ87	VSS
CJ90	LP4_1_DQ26/DDR_0_DQ58
CJ93	LP4_1_DQ31/DDR_0_DQ63
CJ96	VSS
CJ99	LP4_1_DQ10/DDR_0_DQ42
CK1	VSS
CK100	LP4_1_DQS1_DN/DDR_0_DQS5_DN
CK106	DDR_0_ODT1
CK109	LP4_0_CS0/DDR_0_CS0_N
CK111	DDR_0_BA1
CK113	VSS
CK2	RTC_X1
CK31	GP_B08/PSE_I2C1_SDA/PSE_TGPIO09
CK4	VSS
CK40	VSS
CK49	VSS
CK57	VSS
CK66	VSS
CK75	VSS
CK83	VSS

Table 31-1. Processor Ball Names (Sheet 27 of 39)

Ball Number	Ball Name
CK92	VSS
CL104	LP4_1_DQ15/DDR_0_DQ47
CL35	VCCSFR_OC
CL38	LP4_3_DQ24/DDR_1_DQ56
CL43	LP4_3_DQ30/DDR_1_DQ62
CL46	LP4_3_DQ08/DDR_1_DQ40
CL51	LP4_3_DQ14/DDR_1_DQ46
CL55	DDR_1_PAR
CL60	DDR_1_MA00
CL64	LP4_2_CA2/DDR_1_MA06
CL69	LP4_2_CA0/DDR_1_MA05
CL72	LP4_2_DQ24/DDR_1_DQ24
CL77	LP4_2_DQ30/DDR_1_DQ30
CL81	LP4_2_DQ08/DDR_1_DQ08
CL86	LP4_2_DQ14/DDR_1_DQ14
CL89	LP4_1_DQ24/DDR_0_DQ56
CL94	LP4_1_DQ30/DDR_0_DQ62
CL98	LP4_1_DQ08/DDR_0_DQ40
CM1	RSVD
CM10	VSS
CM100	LP4_1_DQS1_DP/DDR_0_DQS5_DP
CM102	LP4_1_DQ14/DDR_0_DQ46
CM105	RSVD
CM107	LP4__VTT_CTL/DDR_VTT_CTL
CM109	VSS
CM111	LP4_RCOMP0/DDR_RCOMP0
CM113	VSS
CM13	GP_B11/PMC_ALERT_N/PSE_TGPIO06
CM16	GP_G22/ESPI_RST0_N
CM19	VSS
CM2	PMC_DSW_PWROK
CM22	GP_G14/AVS_I2S3_RXD/DMIC_CLK_B1/ PSE_TGPIO09
CM24	GP_G19/AVS_I2S1_SCLK
CM27	VSS
CM30	GP_G01/SD_SDIO_D0
CM33	GP_G03/SD_SDIO_D2
CM4	PMC_SLP_SUS_N
CM40	LP4_3_DQS3_DN/DDR_1_DQS7_DN
CM42	LP4_3_DQ29/DDR_1_DQ61
CM49	LP4_3_DQS1_DN/DDR_1_DQS5_DN

Table 31-1. Processor Ball Names (Sheet 28 of 39)

Ball Number	Ball Name
CM51	LP4_3_DQ13/DDR_1_DQ45
CM57	LP4_3_CLK_DN/DDR_1_CLK1_DN
CM59	DDR_1_MA04
CM6	RSVD
CM66	LP4_2_CLK_DN/DDR_1_CLK0_DN
CM68	LP4_2_CA1/DDR_1_MA09
CM75	LP4_2_DQS3_DN/DDR_1_DQS3_DN
CM77	LP4_2_DQ29/DDR_1_DQ29
CM8	PMC_WAKE_N
CM83	LP4_2_DQS1_DN/DDR_1_DQS1_DN
CM85	LP4_2_DQ13/DDR_1_DQ13
CM92	LP4_1_DQS3_DN/DDR_0_DQS7_DN
CM94	LP4_1_DQ29/DDR_0_DQ61
CN12	GP_B05/PSE_I2C0_SCL/PSE_TGPIO06
CN15	VSS
CN17	GP_G21/ESPI_CLK
CN20	GP_G18/ESPI_IO3
CN23	VSS
CN26	GP_G10/AVS_I2S2_RXD/DMIC_DATA1
CN29	GP_G00/SD_SDIO_CMD
CN31	VSS
CN34	VCCSFR_OC
CN36	VSS
CN38	LP4_3_DQ25/DDR_1_DQ57
CN44	VSS
CN47	LP4_3_DQ09/DDR_1_DQ41
CN53	VSS
CN55	LP4_2_CS1
CN61	VSS
CN64	LP4_2_CA5/DDR_1_BG0
CN70	VSS
CN73	LP4_2_DQ25/DDR_1_DQ25
CN79	VSS
CN81	LP4_2_DQ09/DDR_1_DQ09
CN87	VSS
CN90	LP4_1_DQ25/DDR_0_DQ57
CN96	VSS
CN98	LP4_1_DQ09/DDR_0_DQ41
CP100	VSS
CP102	LP4_1_DQ13/DDR_0_DQ45

Table 31-1. Processor Ball Names (Sheet 29 of 39)

Ball Number	Ball Name
CP105	VSS
CP107	LP4_RCOMP2/DDR_RCOMP2
CP109	LP4_RCOMP1/DDR_RCOMP1
CP111	VSS
CP2	RSVD
CP37	LP4_3_DQ28/DDR_1_DQ60
CP4	RTC_TEST_N
CP40	LP4_3_DQS3_DP/DDR_1_DQS7_DP
CP42	LP4_3_DQ27/DDR_1_DQ59
CP46	LP4_3_DQ12/DDR_1_DQ44
CP48	LP4_3_DQS1_DP/DDR_1_DQS5_DP
CP51	LP4_3_DQ11/DDR_1_DQ43
CP54	DDR_1_BA1
CP57	LP4_3_CLK_DP/DDR_1_CLK1_DP
CP59	DDR_1_MA03
CP6	VSS
CP63	LP4_2_CA3/DDR_1_MA08
CP66	LP4_2_CLK_DP/DDR_1_CLK0_DP
CP68	LP4_2_CKE1
CP72	LP4_2_DQ28/DDR_1_DQ28
CP74	LP4_2_DQS3_DP/DDR_1_DQS3_DP
CP77	LP4_2_DQ27/DDR_1_DQ27
CP8	PMC_DRAM_RESET_N
CP80	LP4_2_DQ12/DDR_1_DQ12
CP83	LP4_2_DQS1_DP/DDR_1_DQS1_DP
CP85	LP4_2_DQ11/DDR_1_DQ11
CP89	LP4_1_DQ28/DDR_0_DQ60
CP91	LP4_1_DQS3_DP/DDR_0_DQS7_DP
CP94	LP4_1_DQ27/DDR_0_DQ59
CP97	LP4_1_DQ12/DDR_0_DQ44
CR1	VSS
CR10	VCC_RTC_EXT
CR113	RSVD
CR13	GP_B06/PSE_I2C0_SDA/PSE_TGPIO07
CR16	GP_G15/ESPI_IO0
CR19	GP_G17/ESPI_IO2
CR22	GP_G13/AVS_I2S3_TXD/DMIC_CLK_B0/ PSE_TGPIO08
CR24	GP_G12/AVS_I2S3_SFRM/SATA_1_GP/ SATAXPCIE_1/DMIC_DATA1/PSE_TGPIO31
CR27	GP_G08/AVS_I2S2_SFRM/DMIC_DATA0

Table 31-1. Processor Ball Names (Sheet 30 of 39)

Ball Number	Ball Name
CR30	GP_G23/SD_SDIO_WP
CR33	GP_G04/SD_SDIO_D3
CT101	LP4_1_DQ11/DDR_0_DQ43
CT104	VSS
CT107	VSS
CT110	VSS
CT112	RSVD
CT12	GP_B07/PSE_I2C1_SCL/PSE_TGPIO08
CT15	GP_G16/ESPI_IO1
CT17	GP_G20/ESPI_CS0_N
CT2	RSVD
CT20	GP_G09/AVS_I2S2_TXD/DMIC_CLK_A1
CT23	GP_G11/AVS_I2S3_SCLK/DMIC_DATA0/ PSE_TGPIO07
CT26	GP_G07/AVS_I2S2_SCLK/DMIC_CLK_A0
CT29	GP_G06/SD_SDIO_CLK
CT31	GP_G05/SD_SDIO_CD_N
CT34	GP_G02/SD_SDIO_D1
CT36	VDDQ
CT39	VSS
CT4	VSS
CT44	VDDQ
CT48	VSS
CT53	VDDQ
CT57	VSS
CT6	VSS
CT61	VDDQ
CT65	VSS
CT70	VDDQ
CT74	VSS
CT79	VDDQ
CT82	VSS
CT87	VDDQ
CT9	VSS
CT91	VSS
CT96	VDDQ
D1	RSVD
D10	RSVD
D101	JTAG_PRDY_N
D105	SVID_DATA
D107	SVID_CLK

Table 31-1. Processor Ball Names (Sheet 31 of 39)

Ball Number	Ball Name
D109	PROCHOT_N
D111	VSS
D113	RSVD
D13	VSS
D16	USB2_9_DP
D19	USB2_5_DN
D2	VSS
D22	VSS
D24	USB2_0_DN
D27	USB2_2_DN
D30	VSS
D33	USB2_8_DN
D35	VSS
D37	VSS
D4	USB2_RCOMP
D40	VSS
D43	VSS
D46	VSS
D48	VSS
D51	VSS
D54	VSS
D57	VSS
D6	RSVD
D60	VSS
D62	VSS
D65	VSS
D68	VCCST_PWRGD
D71	VCCIN_AUX
D73	VCCIN_AUX
D76	VCCIN_AUX
D79	VCC_OUT_FIVR_1P05A
D8	HSIO_RCOMPN
D82	VCC_IN_STG
D85	RSVD
D87	RSVD
D90	EDP_UTILS/MDSI_DE_TE_1
D93	PCH_JTAG_TCK
D96	PCH_JTAG_TDO
D98	CPU_JTAG_TDI
E1	RSVD

Table 31-1. Processor Ball Names (Sheet 32 of 39)

Ball Number	Ball Name
E107	VSS
E109	DDI0_RCOMP
E111	RSVD
E113	VSS
E2	RSVD
E4	GP_F02/SIO_UART0_TXD
F101	DDI1_TXN1
F104	VSS
F11	GP_F13/AVS_I2S4_SFRM/PSE_SWDIO/ISI_SWDIO
F14	GP_F17/PSE_TRACEDATA_3/ISI_TRACEDATA_3/ PSE_TGPIO50
F18	GP_C18/PSE_I2C4_SDA/SML_DATA0/ SIO_I2C1_SDA
F22	GP_C13/PSE_UART0_TXD/SIO_UART1_TXD
F25	GP_C02/PSE_PWM00/SMB_ALERT_N/PSE_TGPIO29
F29	GP_E13
F32	GP_E01
F36	GP_E07/DDI1_DDC_SCL/CPU_GP1/PSE_TGPIO16
F39	GP_E03/DDI1_HPD/PNL_MISC_DDI1/CPU_GP0/ PSE_TGPIO15
F43	GP_E10
F46	GP_E15/PSE_I2S0_RXD/PSE_CAN0_TX/ PSE_TGPIO17
F50	PCIE_3_RXN
F53	PCIE_4_RXN
F58	PCIE_6_RXN/GBE_SGMII_RXN
F61	PCIE_5_RXN/PSE_GBE0_SGMII_RXN
F65	VSS
F68	VSS
F72	VSS
F75	CFG_RCOMP
F8	GP_F20
F80	CFG_09
F83	CFG_15
F87	CFG_14
F90	CFG_13
F95	CATERR_N
F98	VSS
G108	VSS
G110	DDI0_TXN0
G112	DDI0_TXP0
G2	RSVD

Table 31-1. Processor Ball Names (Sheet 33 of 39)

Ball Number	Ball Name
G5	GP_F08/AVS_I2S_MCLK2/PSE_TRACEDATA_0/ ISI_TRACEDATA_0/PSE_TGPIO48
H1	RSVD
H101	DDI1_TXP1
H104	DDI0_TXP2
H11	GP_F12/AVS_I2S4_TXD/PSE_TRACESWO/ ISI_TRACESWO
H113	VSS
H14	GP_F16/AVS_I2S4_SCLK/PSE_SWCLK/ISI_SWCLK
H18	GP_C19/PSE_I2C4_SCL/SML_CLK0/SIO_I2C1_SCL
H22	GP_C12/PSE_UART0_RXD/SIO_UART1_RXD
H25	GP_C00/SMB_CLK/PSE_I2C3_SCL/PSE_TGPIO18
H29	GP_E02
H3	RSVD
H32	GP_E22/PNL1_BKLTCTL/PSE_PWM14/PSE_TGPIO18
H36	GP_E23/PNL1_BKLTEN[1]/PSE_PWM15/PSE_TGPIO19 ***Note:*** 1. Not used.
H39	GP_E04/SATA_0_DEVSLP/PSE_PWM08/ PSE_TGPIO44
H43	GP_E18/DDI0_DDC_SDA/PSE_PWM12/ PSE_TGPIO23
H46	GP_E09/USB2_OC0_N
H50	PCIE_3_RXP
H53	PCIE_4_RXP
H58	PCIE_6_RXP/GBE_SGMII_RXP
H61	PCIE_5_RXP/PSE_GBE0_SGMII_RXP
H65	PCIE_CLK5_DP
H68	PCIE_CLK0_DP
H72	PCIE_CLK2_DN
H75	ICLK_BIASREF
H8	GP_F01/SIO_UART0_RXD
H80	VSS
H83	CFG_10
H87	CFG_11
H90	CFG_18
H95	THRMTRIP_N
H98	DDI1_TXP3
J109	DDI0_TXP1
J111	DDI0_TXN1
J6	VSS
K101	VSS

Table 31-1. Processor Ball Names (Sheet 34 of 39)

Ball Number	Ball Name
K104	DDI0_TXN2
K11	GP_F10/PSE_I2S1_SFRM/AVS_I2S4_SFRM/ PSE_TGPIO15
K14	GP_F15/PSE_TRACEDATA_2/ISI_TRACEDATA_2
K18	GP_C16/GBE_MDIO/PSE_UART3_RXD/ SIO_I2C0_SDA
K2	RSVD
K22	GP_C20/PSE_UART4_RXD/SIO_UART2_RXD
K25	GP_C11/PSE_HSUART0_RE
K29	GP_E08/SATA_1_DEVSLP/PSE_TGPIO45
K32	GP_E21/PSE_I2S0_SFRM/PSE_CAN1_RX/ PSE_TGPIO15
K36	GP_E05/DDI1_DDC_SDA/PSE_PWM09/ PSE_TGPIO17
K39	GP_E12
K4	RSVD
K43	GP_E17/PNL1_VDDEN[1]/PNL_MISC_DDI2/ PSE_PWM11/PSE_TGPIO46 ***Note:*** 1. Not used.
K46	GP_E14/DDI0_HPD/PNL_MISC_DDI0/PSE_TGPIO19
K50	VSS
K53	VSS
K58	VSS
K61	VSS
K65	PCIE_CLK5_DN
K68	PCIE_CLK0_DN
K72	PCIE_CLK2_DP
K75	VSS
K8	GP_F22/PMC_VNN_CTRL
K80	CFG_16
K83	CFG_08
K87	VSS
K90	CFG_19
K95	VSS
K98	DDI1_TXN3
L1	XTAL_IN
L108	VSS
L110	DDI2_TXP3
L112	DDI2_TXN3
L3	VSS
M101	DDI0_TXP3
M104	VSS

Table 31-1. Processor Ball Names (Sheet 35 of 39)

Ball Number	Ball Name
M11	GP_F11/PSE_TRACECLK/ISI_TRACECLK/ PSE_TGPIO49
M113	VSS
M14	GP_F14/AVS_I2S4_RXD/PSE_TRACEDATA_1/ ISI_TRACEDATA_1
M18	GP_C17/GBE_MDC/PSE_UART3_TXD/ SIO_I2C0_SCL
M22	GP_C05/PSE_PWM01/PSE_UART3_CTS_N/ SML_ALERT0_N/PSE_TGPIO30
M25	GP_C10/PSE_TGPIO05
M29	GP_E20/PSE_I2S0_SCLK/PSE_CAN1_TX/ PSE_TGPIO14
M32	GP_E19/DDI0_DDC_SCL/PSE_PWM13/ PSE_TGPIO24
M36	GP_E06/PSE_PWM10/PSE_TGPIO18
M39	GP_E11
M43	GP_E16/PSE_I2S0_TXD/PSE_CAN0_RX/ PSE_TGPIO16
M46	GP_E00/SATA_LED_N/SATAXPCIE_0/SATA_0_GP
M50	PCIE_2_RXN
M53	PCIE_9_RXN/SATA_1_RXN/PSE_GBE1_SGMII_RXN
M58	PCIE_7_RXN/PSE_GBE1_SGMII_RXN
M61	PCIE_8_RXN/SATA_0_RXN/GBE_SGMII_RXN
M65	VSS
M68	VSS
M72	VSS
M75	PCIE_CLK3_DP
M8	GP_F21
M80	CFG_17
M83	CFG_12
M87	CFG_06
M90	VSS
M95	DDI1_TXP2
M98	VSS
N109	DDI2_TXP2
N111	DDI2_TXN2
N2	XTAL_OUT
N4	VSS
N6	GP_F07/PSE_I2S1_SCLK/AVS_I2S4_SCLK/ PSE_TGPIO14
P1	DBG_PMODE
P101	DDI0_TXN3
P104	DDI0_AUXP
P11	VSS

Table 31-1. Processor Ball Names (Sheet 36 of 39)

Ball Number	Ball Name
P14	VSS
P18	VSS
P22	VSS
P25	VSS
P29	VSS
P3	VSS
P32	VSS
P36	VSS
P39	VSS
P43	VSS
P46	VSS
P50	PCIE_2_RXP
P53	PCIE_9_RXP/SATA_1_RXP/PSE_GBE1_SGMII_RXP
P58	PCIE_7_RXP/PSE_GBE1_SGMII_RXP
P61	PCIE_8_RXP/SATA_0_RXP/GBE_SGMII_RXP
P65	VSS
P68	PCIE_CLK1_DN
P72	PCIE_CLK4_DP
P75	PCIE_CLK3_DN
P8	GP_F05/PSE_TGPIO14
P80	VSS
P83	VSS
P87	CFG_05
P90	CFG_01
P95	DDI1_TXN2
P98	DDI1_TXP0
R108	VSS
R110	DDI2_TXP0
R112	DDI2_TXN0
T101	DDI1_AUXP
T104	DDI0_AUXN
T11	GP_F19/PSE_I2S1_RXD/AVS_I2S4_RXD/ PSE_TGPIO17
T113	VSS
T14	GP_F09
T18	GP_C06/PSE_GBE1_MDC
T2	PNL0_BKLTCTL
T22	GP_C04/PSE_GBE0_MDIO/PSE_UART3_RTS_N/ PSE_HSUART3_DE
T25	GP_C15/PSE_UART0_CTS_N/SIO_UART1_CTS_N
T29	GP_S00

Table 31-1. Processor Ball Names (Sheet 37 of 39)

Ball Number	Ball Name
T32	GP_A21/PSE_GBE1_RGMII_RXD1/AVS_I2S5_TXD
T36	GP_A19/PSE_GBE1_RGMII_RXD3/AVS_I2S5_SCLK
T39	GP_A13/PSE_GBE1_RGMII_TXD1
T4	PNL0_BKLTEN
T43	GP_A11/PSE_GBE1_RGMII_TXD3
T46	GP_A10/PSE_GBE0_RGMII_RXD0
T50	VSS
T53	VSS
T58	VSS
T61	VSS
T65	VSS
T68	PCIE_CLK1_DP
T72	PCIE_CLK4_DN
T75	VSS
T8	GP_F23/PMC_V1P05_CTRL
T80	CFG_00
T83	BPM2_N
T87	CFG_07
T90	CFG_02
T95	DDI1_TXN0
T98	DDI1_AUXN
U1	GP_U07/PSE_QEPA3/PSE_SPI1_MOSI/ PSE_TGPIO10
U109	DDI2_AUXP
U111	DDI2_AUXN
U3	PNL0_VDDEN
U6	GP_F06/PSE_TGPIO47
V102	VSS
V104	VSS
V106	VSS
V11	GP_F18/PSE_I2S1_TXD/AVS_I2S4_TXD/ PSE_TGPIO16
V14	GP_F00/SIO_UART0_RTS_N
V18	GP_C07/PSE_GBE1_MDIO/PSE_HSUART3_RE
V2	GP_U05/ISI_SPIM_SCLK/PSE_SPI1_CLK
V22	GP_C03/PSE_GBE0_MDC/PSE_HSUART3_EN
V25	GP_C23/PSE_UART4_CTS_N/ISI_SPIS_MOSI/ SIO_UART2_CTS_N
V29	GP_S01
V32	GP_A22/PSE_GBE1_RGMII_RXD0/AVS_I2S5_RXD
V36	GP_A20/PSE_GBE1_RGMII_RXD2/AVS_I2S5_SFRM
V39	GP_A14/PSE_GBE1_RGMII_TXD0

Table 31-1. Processor Ball Names (Sheet 38 of 39)

Ball Number	Ball Name
V4	GP_U13/ISI_CHX_OKNOK_1
V43	GP_A12/PSE_GBE1_RGMII_TXD2
V46	GP_A23/PSE_GBE0_RGMII_RXCTL
V50	GP_A08/PSE_GBE0_RGMII_RXD2
V53	GP_A03/PSE_GBE0_RGMII_TXD0
V58	PSE_GBE0_RGM0_RCOMP
V61	VCC_OUT_FET_1P05A
V65	VCC_OUT_FET_1P05A
V68	VSS
V72	VSS
V75	VSSIO_Sense
V8	GP_F03/SIO_UART0_CTS_N
V80	CFG_03
V83	BPM1_N
V87	CFG_04
V90	VSS
V93	VSS
V95	VSS
V97	VSS
V99	VSS
W108	VSS
W110	DDI2_TXP1
W112	DDI2_TXN1
Y1	GP_U04/ISI_SPIM_CS/PSE_SPI1_CS0_N
Y11	GP_F04
Y14	GP_C08/PSE_TGPIO04/DNX_FORCE_RELOAD
Y18	GP_C09/PSE_HSUART0_EN
Y22	GP_C01/SMB_DATA/PSE_I2C3_SDA/PSE_TGPIO19
Y25	GP_C14/PSE_UART0_RTS_N/PSE_HSUART0_DE/ SIO_UART1_RTS_N
Y29	GP_C22/PSE_UART4_RTS_N/ISI_SPIM_MOSI/ SIO_UART2_RTS_N
Y3	VSS
Y32	GP_A18/PSE_GBE1_RGMII_RXCTL
Y36	GP_A17/PSE_GBE1_RGMII_RXCLK
Y39	GP_A16/PSE_GBE1_RGMII_TXCTL
Y43	GP_A15/PSE_GBE1_RGMII_TXCLK
Y46	GP_A09/PSE_GBE0_RGMII_RXD1
Y50	GP_A07/PSE_GBE0_RGMII_RXD3
Y53	GP_A04/PSE_GBE0_RGMII_TXCLK
Y58	GP_A01/PSE_GBE0_RGMII_TXD2

Table 31-1. Processor Ball Names (Sheet 39 of 39)

Ball Number	Ball Name
Y61	VCC_OUT_FET_1P05A
Y65	VCC_OUT_FET_1P05A
Y68	VCC_AGSH
Y72	VCC_AGSH
Y75	VCCIO_Sense
Y8	PMC_SYS_RESET_N
Y80	BPM3_N
Y83	BPM0_N
Y87	VSS

§ §

32 Package Information

32.1 Package Mechanical Drawing - Non IHS

Figure 32-1. Package Mechanical drawing - Part 1 of 2

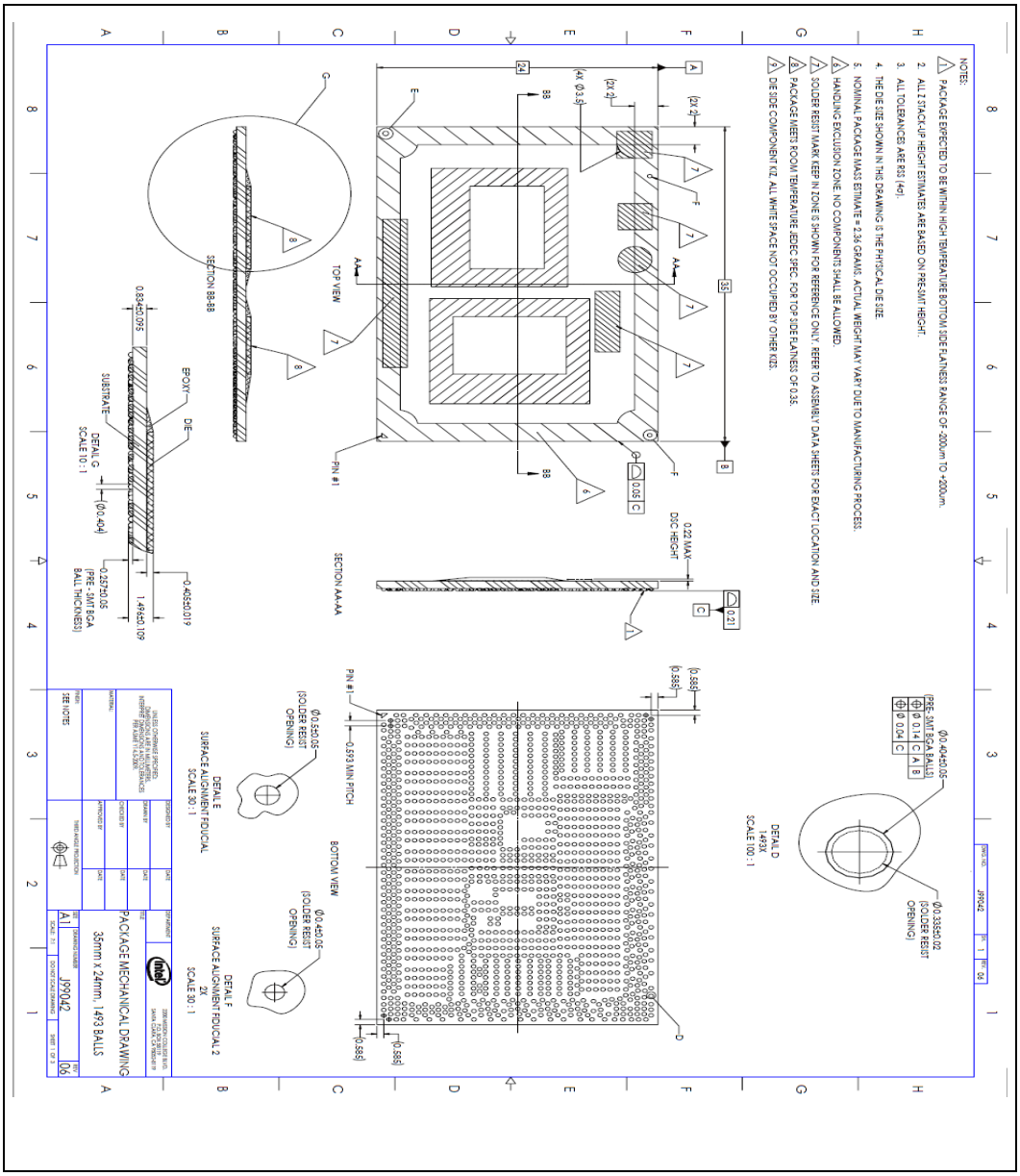
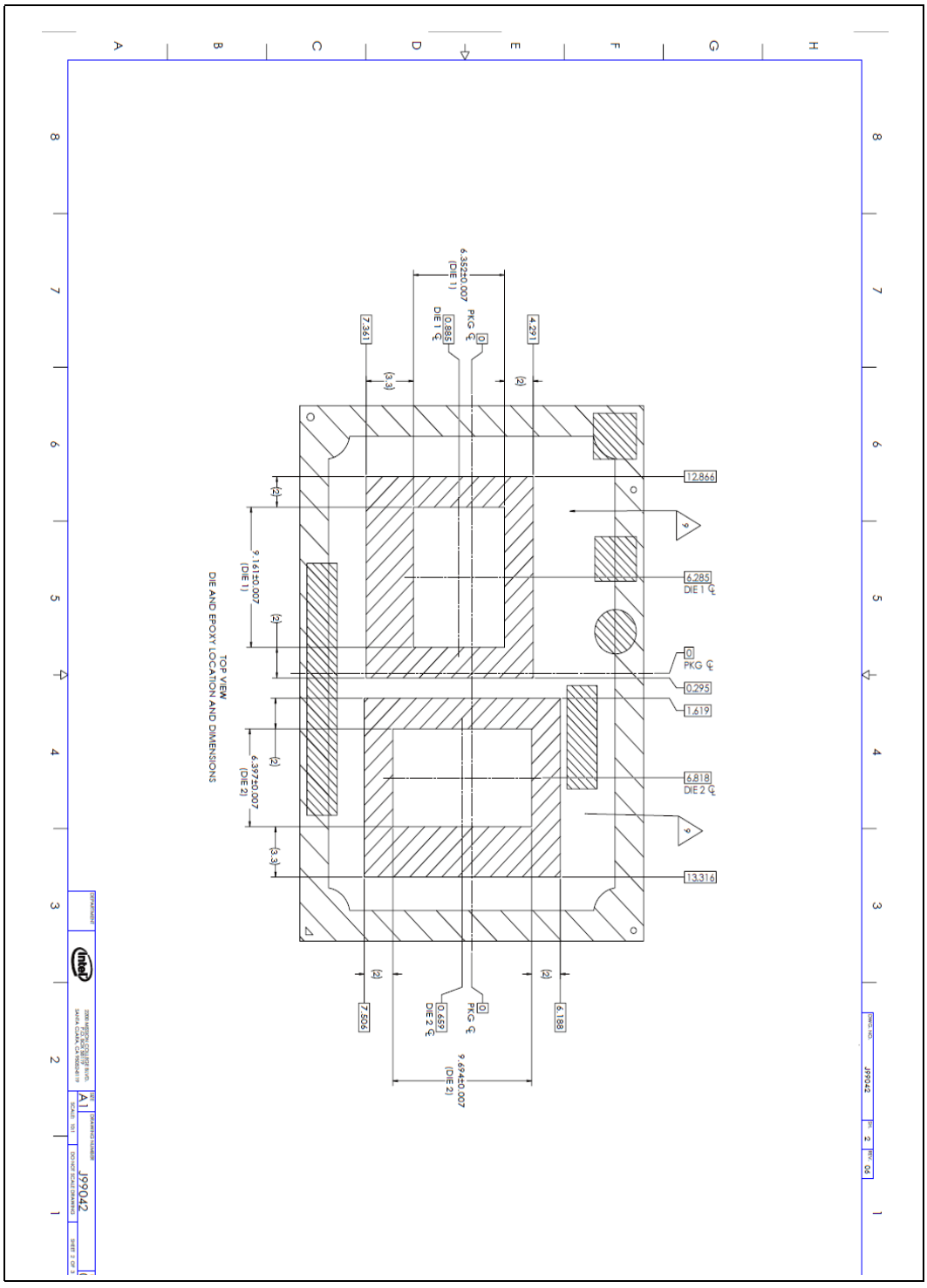
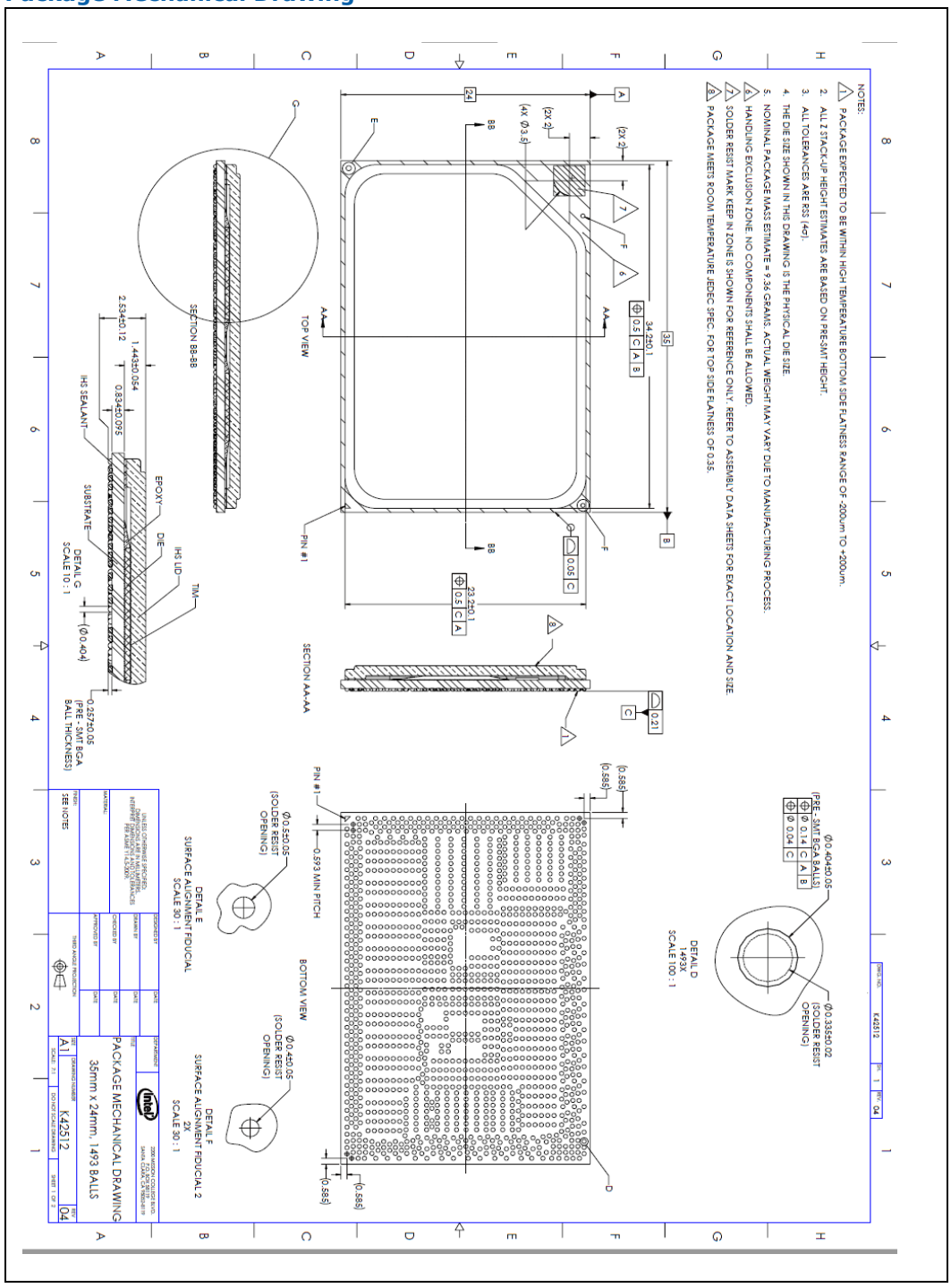


Figure 32-2. Package Mechanical drawing - Part 2 of 2



32.2 Package Mechanical Drawing - IHS

Figure 32-3. Package Mechanical Drawing



§ §

33 Processor Transaction Router (PTR)

33.1 Overview

The Processor Transaction Router (sometimes known as the System Agent) is a central hub that routes transactions between the CPU cores, Gen 11LP graphics controller, the memory controller and the I/O Fabric (PSFx). It also includes up to 4MB of LLC (Last Level Cache) that is shared between the CPU cores and the Gen 11LP graphics controller.

33.2 I/O Port (IOP)

33.2.1 Overview

The I/O Port (IOP) is part of the Processor Transaction Router (PTR) that translates I/O Fabric (PSFx) transactions to transactions within the PTR & transactions to the memory controller and vice versa.

It also implements:

- IOMMU (Input/Output Memory Management Unit) required for processor VT-d support
- IBECC (Inband ECC) error reporting

33.2.1.1 IOMMU (Input/Output Memory Management Unit) required for processor VT-d support

The IOMMU provides DMA address translation and device isolation facilities, so that a particular device is only allowed to perform DMA transactions to and from certain memory areas (designated by the IOMMU) and cannot access the rest of the system memory address space.

Due to the involvement of the IOMMU, the physical address the hardware uses may not be the real physical address, but instead a (completely arbitrary) input-output virtual address (IOVA) assigned to the hardware by the IOMMU. The IOMMU takes care of address translation, so the hardware never notices the difference between the two.

33.2.1.2 IBECC (Inband ECC) Error Reporting

Uncorrectable and correctable ECC errors are required to be reported to the PCH and compute die. The information in this section is in addition to Section 5.3.5 and describes the processor-level error reporting flow.

1. IBECC controller sends an error message to the IOP

2. IOP receives the error message from the IBECC controller and

i. Sets either the IBECC_UC or IBECC_COR field, as applicable, in the ERRSTS_0_0_0_PCI register to 1h
ii. Sends an error message to the PCH if the IBECC_UC or IBECC_COR field, as applicable, in the ERRCMD_0_0_0_PCI register is set to 1h

- **Note:** It is possible, but not supported, for the IOP to send a SMI or SCI to the PCH instead of an SERR by setting the IBECC_UC or IBECC_COR field, as applicable, in the SMICMD_0_0_0_PCI or SCICMD_0_0_0_PCI registers

3. ITSS in PCH receives the error message and sends an NMI to the IA CPU core

4. The IA CPU core receives the NMI and runs an NMI handler (if implemented by the OS (Operating System)) to check the NMI_STS_CNT register settings.

Note: An IBECC error will cause SERR_NMI_STS to be set.

5. If SERR_NMI_STS is set, the IA CPU core should then run the OS EDAC (Error Detection and Correction) framework driver (if implemented by the OS) to read the ERRSTS_0_0_0_PCI register to determine the cause of the NMI

Note: An IBECC error will be indicated by either the IBECC_UC or IBECC_COR field being set.

6. If either the IBECC_UC or IBECC_COR fields is set, the OS EDAC framework driver (if implemented by the OS) should then:

i. Read the IBECC ECC_ERR_LOG register

ii. Clear the IBECC_UC & IBECC_COR fields

iii. Clear the MERRSTS & CERRSTS fields in the IBECC ECC_ERR_LOG register

§ §

34 Machine Check Architecture (MCA)

34.1 Overview

The processor implements a machine-check architecture that provides a mechanism for detecting and reporting hardware (machine) errors, such as: system bus errors, parity errors, cache errors, and TLB errors. It consists of a set of model-specific registers (MSRs) that are used to set up machine checking and additional banks of MSRs used for recording errors that are detected. The processor signals the detection of an uncorrected machine-check error by generating a machine-check exception (#MC), which is an abort class exception. The implementation of the machine-check architecture does not ordinarily permit the processor to be restarted reliably after generating a machine-check exception. However, the machine-check-exception handler can collect information about the machine-check error from the machine-check MSRs. The processor can report information on corrected machine-check errors and deliver a programmable interrupt for software to respond to MC errors, referred to as corrected machine-check error interrupt (CMCI).

See Chapter 15 of the Intel® 64 and IA-32 Architectures Software Developer's Manual for further information regarding Machine Check Architecture.

34.2 Machine Check Architecture (MCA) MSR Addresses

Table 34-1 lists the MSR addresses and Figure 34-1, "Processor Core, Module, and Compute Die Machine Check Registers" shows the core, module, and compute die with respect to each Machine Check register.

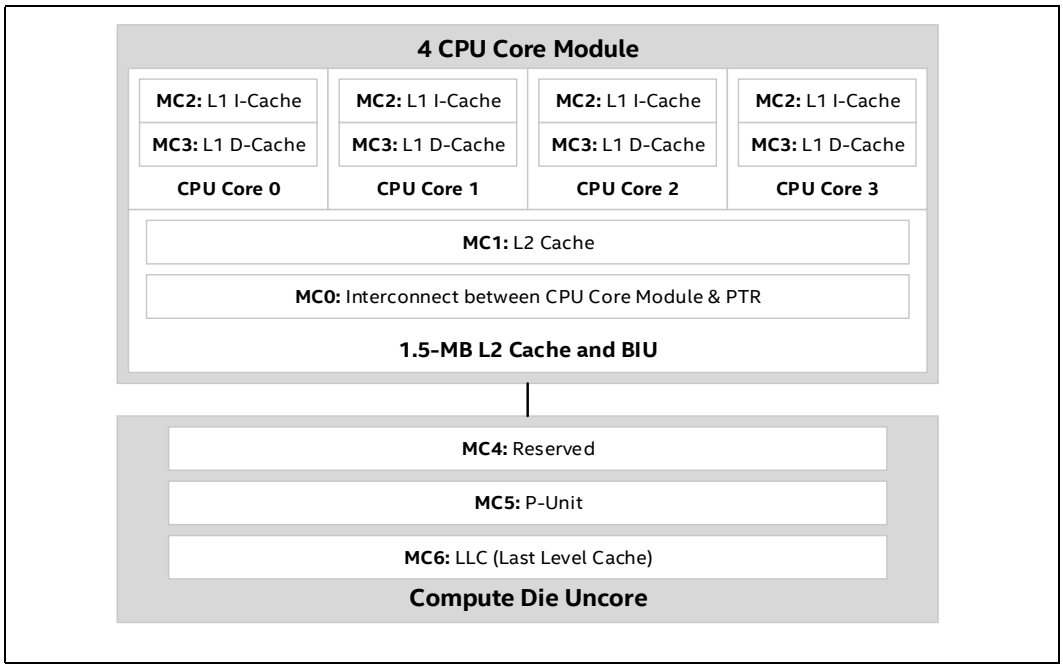
Table 34-1. Processor Machine Check MSR Address (Sheet 1 of 2)

Register Type	Function	Register	Register Address
MC Bank 0	Interconnect between CPU Core modules & Processor Transaction Router	IA32_MC0_CTL	400h
		IA32_MC0_STATUS	401h
		IA32_MC0_ADDR	402h
		IA32_MC0_MISC	403h
		IA32_MC0_CTL2	280h
MC Bank 1	CPU L2 Cache	IA32_MC1_CTL	404h
		IA32_MC1_STATUS	405h
		IA32_MC1_ADDR	406h
		IA32_MC1_MISC	407h
		IA32_MC1_CTL2	281h

Table 34-1. Processor Machine Check MSR Address (Sheet 2 of 2)

Register Type	Function	Register	Register Address
MC Bank 2	CPU L1 Instruction Cache	IA32_MC2_CTL	408h
		IA32_MC2_STATUS	409h
		IA32_MC2_ADDR	40Ah
		IA32_MC2_MISC	40Bh
		IA32_MC2_CTL2	282h
MC Bank 3	CPU L1 Data Cache	IA32_MC3_CTL	40Ch
		IA32_MC3_STATUS	40Dh
		IA32_MC3_ADDR	40Eh
		IA32_MC3_MISC	40Fh
		IA32_MC3_CTL2	283h
MC Bank 4	Reserved	UNCORE_MC04_CTL	410h
		UNCORE_MC04_STATUS	411h
		UNCORE_MC04_ADDR	412h
		UNCORE_MC04_MISC	413h
		UNCORE_MC04_CTL2	284h
MC Bank 5	P-Unit	UNCORE_MC05_CTL	414h
		UNCORE_MC05_STATUS	415h
		UNCORE_MC05_ADDR	416h
		UNCORE_MC05_MISC	417h
		UNCORE_MC05_CTL2	285h
MC Bank 6	LLC (Last Level Cache)	UNCORE_MC06_CTL	418h
		UNCORE_MC06_STATUS	419h
		UNCORE_MC06_ADDR	41Ah
		UNCORE_MC06_MISC	41Bh
		UNCORE_MC06_CTL2	286h

Figure 34-1. Processor Core, Module, and Compute Die Machine Check Registers



34.3 Registers

Note: Please refer to chapter 2 of the Intel Atom® x6000E Series, and Intel® Pentium® and Celeron® N and J Series Processors for Internet of Things (IoT) Applications, Datasheet, Volume 2 (Book 3 of 3), Intel® Programmable Services Engine (Intel® PSE), for a description of the registers associated with the subject of this chapter.

§ §

35 Intel® Converged Security Engine (Intel® CSE)

35.1 Overview

The Processor is the next generation Intel Atom® CPU product targeting key IoT markets. The processor provides new security features embodied within the secure engine (Intel® CSE) such as the enhanced Intel® Boot Guard Device Protection, enhanced cryptographic key sizes and algorithm - RSA 3072 and SHA 384 and content protection services.

The Intel® CSE 15.40 is the security engine that resides in the PCH for running firmware-related applications such as Intel® Boot Guard Device Protection, Intel® Platform Trust Technology (Intel® PTT), and content protection.

35.1.1 Supported Features

Requirement:	Description/Details:
SPI Flash Firmware Image Size	SPI flash memory size shall support: • 16MB • 32MB • 64MB • Flash memory size shall be based on OEM IFWI configuration.
SPI Flash Memory	SPI is supported as boot media to store/boot IFWI.
RPMC Binding	In the case of SPI with RPMC support, Intel® CSE supports data anti-replay protection using a Replay Protected Monotonic Counter. **Note:** Replay Protected Monotonic Counter is a part inside the SPI device used to securely store data and accessible by CSE only.
RPMC Re-Binding	Intel®CSE shall support RPMC re-binding with different SPI device, meaning that if the SPI flash is replaced, a new SPI will be bound to the PCH.
FPF Provisioning	Intel® CSE FW shall use Field Programmable Fuses provisioning to allow fusing of manufacturing setting such as OEM key hash used to sign the OEM KM, selection of boot source, and so on.
Debug Capabilities	Intel® CSE FW shall allow logging firmware debug messages utilizing NPK infrastructure over USB 2.0. **Note**: In a production platform, this capability can be enabled using a debug OEM unlock token.

Requirement:	Description/Details:
Secure Tokens	Intel® CSE FW shall support secure/debug tokens to allow OEMs to unlock the SoC for debugging purposes. Secure/debug tokens are required to be signed by OEM per key provisioned to Intel® CSE in OEM KM. Token characterizations support is as follows: § **Authenticity**: The token can only be signed by trusted entities Intel® and OEM. § **Anti-Replay**: A token cannot be replayed on a device once it has been used and the token's expiration properties have reached. § **Integrity**: An attacker should not be able to tamper with a token to make it available for multiple platforms. Also, an attacker should not be able to tamper with a token to alter the purpose of the token or the data payload of the token. § **Expiration**: The token signing server may enforce limited lifespan from token generation start. § **Tied to Specific Part**: The token signing server authorizes specific device for debug unlock.
Intel® CSE Configuration	Intel® CSE shall support configuration using FPF Fuses, HW/SW straps and firmware setting in CSE NVM/Flash.
Cryptography	Intel® CSE shall support 3072-bit RSA, SHA-384, 256-bit AES, and 384-bit ECC for CSE security services, to be compliant with the NSA and NIST's cryptography recommendation for transitional period to quantum computing. For example, firmware image signing uses 3072-bit RSA with SHA-384.
On-Die Certification Authority	Intel® CSE shall generate and renews CSE application certificates using On-die Certification Authority which provides certificates to the following CSE applications: • Content Protection • PTT • DAL • BUP Note: Intel® CSE on-die Certification Authority is endorsed by iKGF but it issues certificates to CSE applications independent of iKGF Intel server.
Authentication of FW components	Depending on OEM choice and Boot Guard policy in FPF, Intel® CSE firmware shall authenticate Intel® and OEM IP's.
Fuse Emulation	Intel® CSE shall support UEP ("Unified Emulation Partition ") flash partition to emulate fuses prior to EOM. Such capability allows easier integration and simplifies the debugging process by OEM.
Android* TEE	Intel® CSE shall support HW unique seed and attestation keybox provisioning for TEE implementation in Android* in SPI boot media only.

Requirement:	Description/Details:
Flexible CSE Manufacturing Flow	Intel® CSE shall support a flexible EOM flow which allows customers to decide which operations they want to perform in the manufacturing line and which operations can be performed outside of the manufacturing line. Nevertheless, committing the FPFs is a must and should be taking place on the manufacturing line as recommended by Intel. Other operations, such as locking of CSE configurations and locking of the SPI descriptor, can be performed at a later stage outside the manufacturing line through BIOS calls to CSE or by proprietary OEM tools as long as operations are performed pre-EOP.
Anti-rollback	Intel® CSE shall support Anti-Rollback feature which prevent downgrading to a lower firmware SVN. When improving security vulnerability in FW, SVN (Security Vulnerability Number) is increased. In order to prevent downgrading to a lower firmware SVN. ARB which is based on HW stores permanent the minimal SVN value in Field Programmable Fuses (FPFs) which mitigate risk of physical downgrade to a lower SVN. Intel®CSE shall also support ARBSVN mechanism using Key-Manifest for entities that don't have dedicated ARB FPF such as some host loaded modules.
FUSA	When Proof test request is set by Intel® Safety Island (Intel® SI), Intel® CSE shall drive PCH and CPU pre-BIOS Safely tests, and report test results to ISI.
SPI Descriptor Version	Intel® CSE shall support creating a version field in the SPI descriptor that will be updated each time the layout of the SPI descriptor changes.
SPI Descriptor Signature check	Intel® CSE shall support signature verification of SPI descriptor region and halt boot if signature verification fails. Note: This feature protects physical access attacks tampering the descriptor.
Firmware Measurements report to platform	Intel® CSE shall measure firmware binaries in the system boot flash device. It allows the host software to measure Intel® CSE firmware, based on that, the system can meet the requirements of TCG PC Client Specific Implementation Specification.
CSE IP Authentication Status	Intel® CSE shall report to flash log for any IP Authentication failure.
No-coin cell battery designs	Intel® CSE shall support no-coin cell battery designs.

§ §

36 Electrical Specifications

Note: Refer to Section 3.4 and Section 3.5 for power rail electrical specifications.

36.1 Crystal Specifications

Table 36-1. Integrated Clock Crystal Specification

Parameter	Values	Units	Max/Min Range
Frequency	38.4	MHz	
Frequency Tolerance	≤ 100	PPM	
Operating Temperature	-40 to 85	°C	
Series Resistance	≤ 30	Ω	
Aging rate	<10	PPM	Within 10 years
Crystal Drive Level	≥100	µW	
Crystal load capacitance	10 - 12	pF	
Crystal shunt capacitance	≤ 3	pF	

Notes:

1. Customers should verify that the vendor's published specifications in the component data sheet meet the required conditions for frequency, frequency tolerance, temperature, oscillation mode and load capacitance as specified in the respective data sheet.
2. Perform conformance testing and EMC (FCC and EN) testing in real systems.
3. Independently measure the component's electrical parameters in real systems. Measure frequency at a test output to avoid test probe loading effects. Check that the measured behavior is consistent from sample to sample and that measurements meet the published specifications. For crystals, it is also important to examine startup behavior while varying system voltage and temperature.
4. Crystal must be AT cut, at fundamental frequency, parallel resonance mode.

Table 36-2. RTC Crystal Specification

Symbol	Parameter	Minimum	Maximum	Units	Notes/Figure
FRTC	Frequency		32.768	kHz	Nominal
TPPM	Crystal frequency tolerance	-20	20	ppm	
PDRIVE	Crystal drive load	0.1	0.5	µW	Nominal
CLOAD	Crystal Load Capacitance		12.5	pF	12 pF Nominal
CSHUNT	Crystal shunt Capacitance		1.3	pF	
C1/C2	Load Capacitance tolerance	-10	10	%	
RTC Crystal PPM	Crystal Tolerance Cut Accuracy Maximum	35 ppm (@ 25 °C ±3 °C)		ppm	Total of crystal cut accuracy, frequency variations due to temperature, parasitics, load capacitance variations and aging is recommended to be less than 90 ppm.
	Temp Stability Maximum	30 ppm (10 – 70 °C)		ppm	
	Aging Maximum	5 ppm		ppm	

Table 36-2. RTC Crystal Specification

Symbol	Parameter	Minimum	Maximum	Units	Notes/Figure
Notes: 1. Capacitors used in RC Delay circuit for each signals should be evaluated with regards to aging, voltage and temperature characteristic to ensure reliable operation in the intended operating environment.					

36.2 Storage Conditions

This section specifies absolute maximum and minimum storage temperature and humidity limits for given time durations. Failure to adhere to the specified limits could result in physical damage to the component. If this is suspected, Intel recommends a visual inspection to determine possible physical damage to the silicon or surface components.

Table 36-3. Storage Conditions (PC Client Only)

Symbol	Parameter	Min	Max
$T_{ABSOLUTE\ STORAGE}$	Device storage temperature range should not exceed for any length of time	-25°C	125°C
$T_{SHORT\ TERM\ STORAGE}$	The ambient storage temperature and time for up to 72 hours.	-20°C	85°C
$T_{SUSTAINED\ STORAGE}$	The ambient storage temperature and time for up to 30 months.	-5°C	40°C
$RH_{SUSTAINED\ STORAGE}$	The maximum device storage relative humidity for up to 30 months.	N/A	60%@ 24°C
Notes: 1. Specified temperatures are not to exceed values based on data collected. Exceptions for surface mount re-flow are specified by the applicable JEDEC* standard. Non-adherence may affect processor reliability. 2. Component product device storage temperature qualification methods may follow JESD22-A119 (low temperature) and JESD22-A103 (high temperature) standards when applicable for volatile memory. 3. Component stress testing is conducted in conformance with JESD22-A104. 4. The JEDEC* J-JSTD-020 moisture level rating and associated handling practices apply to all moisture sensitive devices removed from the moisture barrier bag.			

Table 36-4. Storage Conditions (Embedded and Indu Only)

Symbol	Parameter	Min	Max
$T_{ABSOLUTE\ STORAGE}$	Device storage temperature range should not exceed for any length of time	-25°C	125°C
$T_{SHORT\ TERM\ STORAGE}$	The ambient storage temperature and time for up to 72 hours.	-20°C	85°C
$T_{SUSTAINED\ STORAGE}$	The ambient storage temperature and time for up to 66 months.	-5°C	40°C
$RH_{SUSTAINED\ STORAGE}$	The maximum device storage relative humidity for up to 66 months.	N/A	60%@ 24°C

Table 36-4. Storage Conditions (Embedded and Indu Only)

Notes:

1. Specified temperatures are not to exceed values based on data collected. Exceptions for surface mount re-flow are specified by the applicable JEDEC* standard. Non-adherence may affect processor reliability.
2. Component product device storage temperature qualification methods may follow JESD22-A119 (low temperature) and JESD22-A103 (high temperature) standards when applicable for volatile memory.
3. Component stress testing is conducted in conformance with JESD22-A104.
4. The JEDEC* J-JSTD-020 moisture level rating and associated handling practices apply to all moisture sensitive devices removed from the moisture barrier bag.

36.2.1 Post Board-Attach

The storage condition limits for the component once attached to the application board are not specified. Intel does not conduct component-level certification assessments post board-attach given the multitude of attach methods, socket types, and board types used by customers. Provided as general guidance only, board-level Intel-branded products are specified and certified to meet the following temperature and humidity limits:

- Non-Operating Temperature Limit: -40 °C to 70 °C
- Humidity: 50% to 90%, non-condensing with a maximum wet-bulb of 28 °C

36.3 DC Specifications

Platform reference voltages are specified at DC only. VCC measurements should be made with respect to the supply voltages specified Section 3.4 and Section 3.5.

Note: **VIH/OH Max and VIL/OL minimum values are bounded by VCC and VSS.**

Note: **Care should be taken to read all notes associated to each parameter.**

Note: **Processor output timing specification, Tco, is measured in a tester environment with a test load. Customer should validate and ensure processor output signal meets the required input setup/hold specification for the device.**

36.3.1 Single-Ended Signal DC Characteristics

Table 36-5. Single-Ended Signal DC Characteristics as Inputs or Outputs (Sheet 1 of 11)

Type	Symbol	Parameter	Minimum	Maximum	Unit	Condition	Notes
Note: For GPIO pads (GP) listed in the Associated Signals below, all functions that are multiplexed on GPIO pads will have the same DC characteristics as the GPIO pads. Refer to the GPIO chapter for the muxed functions on a specific GPIO pad.							
Associated Signals: GP_R00/HDA_BCLK/AVS_I2S0_SCLK/PSE_I2S0_SCLK; **GP_R01**/HDA_SYNC/ AVS_I2S0_SFRM/PSE_I2S0_SFRM; **GP_R02**/HDA_SDO/AVS_I2S0_TXD/PSE_I2S0_TXD/DMIC_CLK_B0; **GP_R03**/HDA_SDI0/AVS_I2S0_RXD/PSE_I2S0_RXD/DMIC_CLK_B1; **GP_R04**/HDA_RST_N/DMIC_CLK_A1 **GP_R05**/HDA_SDI1/AVS_I2S1_RXD/DMIC_DATA1; **GP_R06**/AVS_I2S1_TXD/DMIC_CLK_A0 **GP_R07**/AVS_I2S1_SFRM/DMIC_DATA0; **GP_S00**; **GP_S01**							
3.3V Operation ***Note:*** GP_S GPIOs are not 3.3V capable ***Note:*** VCC = VCC_3P3A							
Input	V_{IH}	Input High Voltage Threshold	0.75 x VCC		V		1ms rise Vpad
	V_{IL}	Input Low Voltage Threshold		0.25 x VCC	V		1ms rise Vpad
	I_{IL}	Input Leakage Current	-12	12	µA		
	C_{IN}	Input Pin Capacitance		13	pF		
Output	V_{OH}	Output High Voltage Threshold	VCC - 0.45V		V		1ms rise Vpad
	V_{OL}	Output Low Voltage Threshold		0.45	V		1ms rise Vpad
	I_{OH}	Output High Current		3	mA		
	I_{OL}	Output Low Current	-3		mA		
	R_{pu}	WPU 20K Resistance	15k	27k	Ω		
	R_{pd}	WPD 20K Resistance	15k	27k	Ω		
1.8V Operation ***Note:*** VCC = VCC_1P8A							
Input	V_{IH}	Input High Voltage Threshold	0.75 x VCC		V		1ms rise Vpad
	V_{IL}	Input Low Voltage Threshold		0.25 x VCC	V		1ms rise Vpad
	I_{IL}	Input Leakage Current	-12	12	µA		
	C_{IN}	Input Pin Capacitance		13	pF		

Table 36-5. Single-Ended Signal DC Characteristics as Inputs or Outputs (Sheet 2 of 11)

Type	Symbol	Parameter	Minimum	Maximum	Unit	Condition	Notes
Output	V_{OH}	Output High Voltage Threshold	VCC - 0.45V		V		1ms rise Vpad
	V_{OL}	Output Low Voltage Threshold		0.45	V		1ms rise Vpad
	I_{OH}	Output High Current		3	mA		
	I_{OL}	Output Low current	-3		mA		
	R_{pu}	WPU 20K Resistance	15k	27k	Ω		
	R_{pd}	WPD 20K Resistance	15k	27k	Ω		
Notes: 1. For GPIO supported voltages, refer to the GPIO chapter.							
Type	**Symbol**	**Parameter**	**Minimum**	**Maximum**	**Unit**	**Condition**	**Notes**
Note: For GPIO pads (GP) listed in the Associated Signals below, all functions that are multiplexed on GPIO pads will have the same DC characteristics as the GPIO pads. Refer to the GPIO chapter for the muxed functions on a specific GPIO pad.							
Associated Signals: GP_A00/PSE_GBE0_RGMII_TXD3;**GP_A01**/PSE_GBE0_RGMII_TXD2;**GP_A10**/PSE_GBE0_RGMII_RXD0;**GP_A11**/PSE_GBE1_RGMII_TXD3;**GP_A12**/PSE_GBE1_RGMII_TXD2;**GP_A13**/PSE_GBE1_RGMII_TXD1;**GP_A14**/PSE_GBE1_RGMII_TXD0;**GP_A15**/PSE_GBE1_RGMII_TXCLK;**GP_A16**/PSE_GBE1_RGMII_TXCTL;**GP_A17**/PSE_GBE1_RGMII_RXCLK; **GP_A18**/PSE_GBE1_RGMII_RXCTL;**GP_A19**/PSE_GBE1_RGMII_RXD3/AVS_I2S5_SCLK; **GP_A02**/PSE_GBE0_RGMII_TXD1;**GP_A20**/PSE_GBE1_RGMII_RXD2/AVS_I2S5_SFRM;**GP_A21**/PSE_GBE1_RGMII_RXD1/AVS_I2S5_TXD;**GP_A22**/PSE_GBE1_RGMII_RXD0/AVS_I2S5_RXD;**GP_A23**/PSE_GBE0_RGMII_RXCTL;**GP_A03**/PSE_GBE0_RGMII_TXD0;**GP_A04**/PSE_GBE0_RGMII_TXCLK;**GP_A05**/PSE_GBE0_RGMII_TXCTL;**GP_A06**/PSE_GBE0_RGMII_RXCLK; **GP_A07**/PSE_GBE0_RGMII_RXD3;**GP_A08**/PSE_GBE0_RGMII_RXD2;**GP_A09**/PSE_GBE0_RGMII_RXD1;**GP_G00**/SD_SDIO_CMD;**GP_G01**/SD_SDIO_D0;**GP_G10**/AVS_I2S2_RXD/DMIC_DATA1;**GP_G11**/AVS_I2S3_SCLK/DMIC_DATA0/PSE_TGPIO07;**GP_G12**/AVS_I2S3_SFRM/SATA_1_GP/SATAXPCIE_1/DMIC_DATA1/PSE_TGPIO31;**GP_G13**/AVS_I2S3_TXD/DMIC_CLK_B0/PSE_TGPIO08;**GP_G14**/AVS_I2S3_RXD/DMIC_CLK_B1/PSE_TGPIO09;**GP_G15**/ESPI_IO0;**GP_G16**/ESPI_IO1;**GP_G17**/ESPI_IO2;**GP_G18**/ESPI_IO3;**GP_G19**/AVS_I2S1_SCLK;**GP_G02**/SD_SDIO_D1;**GP_G20**/ESPI_CS0_N;**GP_G21**/ESPI_CLK;**GP_G22**/ESPI_RST0_N;**GP_G23**/SD_SDIO_WP;**GP_G03**/SD_SDIO_D2 **GP_G04**/SD_SDIO_D3;**GP_G05**/SD_SDIO_CD_N;**GP_G06**/SD_SDIO_CLK;**GP_G07**/AVS_I2S2_SCLK/DMIC_CLK_A0;**GP_G08**/AVS_I2S2_SFRM/DMIC_DATA0;**GP_G09**/AVS_I2S2_TXD/DMIC_CLK_A1							
3.3V Operation ***Note:*** VCC = VCC_3P3A.							
Input	V_{IH}	Input High Voltage Threshold	0.75 x VCC		V		1ms rise Vpad
	V_{IL}	Input Low Voltage Threshold		0.25 x VCC	V		1ms rise Vpad
	I_{IL}	Input Leakage Current	-14	14	µA	VCC/VSS	
	C_{IN}	Input Pin Capacitance		14	pF		

Table 36-5. Single-Ended Signal DC Characteristics as Inputs or Outputs (Sheet 3 of 11)

Type	Symbol	Parameter	Minimum	Maximum	Unit	Condition	Notes
Output	V_{OH}	Output High Voltage Threshold	VCC - 0.45V		V		1ms rise Vpad
	V_{OL}	Output Low Voltage Threshold		0.45	V		1ms rise Vpad
	I_{OH}	Output High Current		3	mA		
	I_{OL}	Output Low Current	-3		mA		
	R_{pu}	WPU 20K Resistance	15k	27k	Ohm		
	R_{pd}	WPD 20K Resistance	15k	27k	Ohm		
1.8V Operation ***Note:*** VCC = VCC_1P8A							
Input	V_{IH}	Input High Voltage Threshold	0.75 x VCC		V		1ms rise Vpad
	V_{IL}	Input Low Voltage Threshold		0.25 x VCC	V		1ms rise Vpad
	I_{IL}	Input Leakage Current	-14	14	µA	VCC/VSS	
	C_{IN}	Input Pin Capacitance		14	pF		
Output	V_{OH}	Output High Voltage Threshold	VCC -0.45V		V		1ms rise Vpad
	V_{OL}	Output Low Voltage Threshold		0.45	V		1ms rise Vpad
	I_{OH}	Output High Current		3	mA		
	I_{OL}	Output Low Current	-3		mA		
	R_{pu}	WPU 20K Resistance	15k	27k	Ohm		
	R_{pd}	WPD 20K Resistance	15k	27k	Ohm		
Type	**Symbol**	**Parameter**	**Minimum**	**Maximum**	**Unit**	**Condition**	**Notes**

Note: For GPIO pads (GP) listed in the Associated Signals below, all functions that are multiplexed on GPIO pads will have the same DC characteristics as the GPIO pads. Refer to the GPIO chapter for the muxed functions on a specific GPIO pad.

Table 36-5. Single-Ended Signal DC Characteristics as Inputs or Outputs (Sheet 4 of 11)

Type	Symbol	Parameter	Minimum	Maximum	Unit	Condition	Notes
Associated Signals: GP_E00/SATA_LED_N/SATAXPCIE_0/SATA_0_GP; **GP_E01**; **GP_E10**;**GP_E11**;**GP_E12**;**GP_E13** ;**GP_E14**/DDI0_HPD/PNL_MISC_DDI0/PSE_TGPIO19;**GP_E15**/PSE_I2S0_RXD/PSE_CAN0_TX/PSE_TGPIO17;**GP_E16**/PSE_I2S0_TXD/PSE_CAN0_RX/PSE_TGPIO16;**GP_E17**/PNL1_VDDEN/PNL_MISC_DDI2/PSE_PWM11/PSE_TGPIO46;**GP_E18**/DDI0_DDC_SDA/PSE_PWM12/PSE_TGPIO23; **GP_E19**/DDI0_DDC_SCL/PSE_PWM13/PSE_TGPIO24; **GP_E02** ;**GP_E20**/PSE_I2S0_SCLK/PSE_CAN1_TX/PSE_TGPIO14 **GP_E21**/PSE_I2S0_SFRM/PSE_CAN1_RX/PSE_TGPIO15;**GP_E22**/PNL1_BKLTCTL/PSE_PWM14/PSE_TGPIO18;**GP_E23**/PNL1_BKLTEN/PSE_PWM15/PSE_TGPIO19; **GP_E03**/DDI1_HPD/PNL_MISC_DDI1/CPU_GP0/PSE_TGPIO15;**GP_E04**/SATA_0_DEVSLP/PSE_PWM08/PSE_TGPIO44;**GP_E05**/DDI1_DDC_SDA/PSE_PWM09/PSE_TGPIO17;**GP_E06**/PSE_PWM10/PSE_TGPIO18;**GP_E07**/DDI1_DDC_SCL/CPU_GP1/PSE_TGPIO16;**GP_E08**/SATA_1_DEVSLP/PSE_TGPIO45;**GP_E09**/USB2_OC0_N;**GP_F00**/SIO_UART0_RTS_N;**GP_F01**/SIO_UART0_RXD;**GP_F10**/PSE_I2S1_SFRM/AVS_I2S4_SFRM/PSE_TGPIO15 **GP_F11**/PSE_TRACECLK/ISI_TRACECLK/PSE_TGPIO49;**GP_F12**/AVS_I2S4_TXD/PSE_TRACESWO/ISI_TRACESWO;**GP_F13**/AVS_I2S4_SFRM/PSE_SWDIO/ISI_SWDIO;**GP_F14**/AVS_I2S4_RXD/PSE_TRACEDATA_1/ISI_TRACEDATA_1;**GP_F15**/PSE_TRACEDATA_2/ISI_TRACEDATA_2;**GP_F16**/AVS_I2S4_SCLK/PSE_SWCLK/ISI_SWCLK;**GP_F17**/PSE_TRACEDATA_3/ISI_TRACEDATA_3/PSE_TGPIO50;**GP_F18**/PSE_I2S1_TXD/AVS_I2S4_TXD/PSE_TGPIO16;**GP_F19**/PSE_I2S1_RXD/AVS_I2S4_RXD/PSE_TGPIO17 **GP_F02**/SIO_UART0_TXD;**GP_F20;GP_F21;GP_F22**/PMC_VNN_CTRL;**GP_F23**/PMC_V1P05_CTRL;**GP_F03**/SIO_UART0_CTS_N;**GP_F04;GP_F05**/PSE_TGPIO14;**GP_F06**/PSE_TGPIO47 **GP_F07**/PSE_I2S1_SCLK/AVS_I2S4_SCLK/PSE_TGPIO14;**GP_F08**/AVS_I2S_MCLK2/PSE_TRACEDATA_0/ISI_TRACEDATA_0/PSE_TGPIO48;**GP_F09**							
3.3V Operation ***Note:*** VCC = VCC_3P3A							
Input	V_{IH}	Input High Voltage Threshold	0.75 x VCC		V		1ms rise Vpad
	V_{IL}	Input Low Voltage Threshold		0.25 x VCC	V		1ms rise Vpad
	I_{IL}	Input Leakage Current	-14	14	µA	VCC/VSS	
	C_{IN}	Input Pin Capacitance		14	pF		
Output	V_{OH}	Output High Voltage Threshold	VCC - 0.45V		V		1ms rise Vpad
	V_{OL}	Output Low Voltage Threshold		0.45	V		1ms rise Vpad
	I_{OH}	Output High Current		3	mA		
	I_{OL}	Output Low Current	-3		mA		
	R_{pu}	WPU 20K Resistance	15k	27k	Ohm		
	R_{pd}	WPD 20K Resistance	15k	27k	Ohm		
1.8V Operation ***Note:*** VCC = VCC_1P8A							

Table 36-5. Single-Ended Signal DC Characteristics as Inputs or Outputs (Sheet 5 of 11)

Type	Symbol	Parameter	Minimum	Maximum	Unit	Condition	Notes
Input	V_{IH}	Input High Voltage Threshold	0.75 x VCC		V		1ms rise Vpad
	V_{IL}	Input Low Voltage Threshold		0.25 x VCC	V		1ms rise Vpad
	I_{IL}	Input Leakage Current	-14	14	µA	VCC/VSS	
	C_{IN}	Input Pin Capacitance		14	pF		
Output	V_{OH}	Output High Voltage Threshold	VCC - 0.45V		V		1ms rise Vpad
	V_{OL}	Output Low Voltage Threshold		0.45	V		1ms rise Vpad
	I_{OH}	Output High Current		3	mA		
	I_{OL}	Output Low Current	-3		mA		
	R_{pu}	WPU 20K Resistance	15k	27k	Ohm		
	R_{pd}	WPD 20K Resistance	15k	27k	Ohm		
Type	**Symbol**	**Parameter**	**Minimum**	**Maximum**	**Unit**	**Condition**	**Notes**
1. **Note:** For GPIO pads (GP) listed in the Associated Signals below, all functions that are multiplexed on GPIO pads will have the same DC characteristics as the GPIO pads. Refer to the GPIO chapter for the muxed functions on a specific GPIO pad.							
Associated Signals: GP_V00/EMMC_CMD;**GP_V01**/EMMC_DATA0;**GP_V10**/EMMC_CLK;**GP_V11**/EMMC_RST_N;**GP_V12**/PSE_TGPIO00;**GP_V13**/PSE_TGPIO01;**GP_V14**/PSE_TGPIO02;**GP_V15**/PSE_TGPIO03;**GP_V02**/EMMC_DATA1;**GP_V03**/EMMC_DATA2;**GP_V04**/EMMC_DATA3;**GP_V05**/EMMC_DATA4 **GP_V06**/EMMC_DATA5;**GP_V07**/EMMC_DATA6;**GP_V08**/EMMC_DATA7;**GP_V09**/EMMC_RCLK							
1.8V Operation ***Note:*** VCC = VCC_1P8A							
Input	V_{IH}	Input High Voltage Threshold	0.65 x VCC		V		1ms rise Vpad
	V_{IL}	Input Low Voltage Threshold		0.35 x VCC	V		1ms rise Vpad
	I_{IL}	Input Leakage Current	-5	5	µA	VCC/VSS	
	C_{IN}	Input Pin Capacitance		5	pF		

Table 36-5. Single-Ended Signal DC Characteristics as Inputs or Outputs (Sheet 6 of 11)

Type	Symbol	Parameter	Minimum	Maximum	Unit	Condition	Notes
Output	V_{OH}	Output High Voltage Threshold	VCC -0.45V		V		1ms rise Vpad
	V_{OL}	Output Low Voltage Threshold		0.45	V		1ms rise Vpad
	I_{OH}	Output High Current		3	mA		
	I_{OL}	Output Low Current	-3		mA		
	R_{pu}	WPU 20K Resistance	12k	28k	Ohm		
	R_{pd}	WPD 20K Resistance	12k	28k	Ohm		
Type	**Symbol**	**Parameter**	**Minimum**	**Maximum**	**Unit**	**Condition**	**Notes**
1. **Note:** For GPIO pads (GP) listed in the Associated Signals below, all functions that are multiplexed on GPIO pads will have the same DC characteristics as the GPIO pads. Refer to the GPIO Chapter for the muxed functions on a specific GPIO pad.							
Associated Signals: GP_DSW00/PMC_BATLOW_N;**GP_DSW01**/PMC_ACPRESENT;**GP_DSW10**/ PMC_SLP_S5_N;**GP_DSW11;GP_DSW02;GP_DSW03**/PMC_PWRBTN_N;**GP_DSW04**/PMC_SLP_S3_N **GP_DSW05**/PMC_SLP_S4_N;**GP_DSW07;GP_DSW08**/PMC_SUSCLK;**GP_DSW09**							
3.3V Operation ***Note:*** VCC = VCC_3P3A							
Input	V_{IH}	Input High Voltage Threshold	0.75 x VCC		V		1ms rise Vpad
	V_{IL}	Input Low Voltage Threshold		0.25 x VCC	V		1ms rise Vpad
	I_{IL}	Input Leakage Current	-10	10	µA	VCC/VSS	
	C_{IN}	Input Pin Capacitance		14	pF		
Output	V_{OH}	Output High Voltage Threshold	VCC -0.45V		V		1ms rise Vpad
	V_{OL}	Output Low Voltage Threshold		0.45	V		1ms rise Vpad
	I_{OH}	Output High Current		3	mA		
	I_{OL}	Output Low Current	-3		mA		
	R_{pu}	WPU 20K Resistance	13k	27k	Ohm		
	R_{pd}	WPD 20K Resistance	13k	27k	Ohm		
Type	**Symbol**	**Parameter**	**Minimum**	**Maximum**	**Unit**	**Condition**	**Notes**
1. **Note:** For GPIO pads (GP) listed in the Associated Signals below, all functions that are multiplexed on GPIO pads will have the same DC characteristics as the GPIO pads. Refer to the GPIO Chapter for the muxed functions on a specific GPIO pad.							

Table 36-5. Single-Ended Signal DC Characteristics as Inputs or Outputs (Sheet 7 of 11)

Type	Symbol	Parameter	Minimum	Maximum	Unit	Condition	Notes
Associated Signals: PNL0_BKLTEN/L_BKLTCTL/PNL0_VDDEN/PMC_SYS_PWROK/PMC_SYS_RESET_N							
3.3V Operation ***Note:*** VCC = VCC_3P3A							
Input	V_{IH}	Input High Voltage Threshold	0.75 x VCC		V		1ms rise Vpad
	V_{IL}	Input Low Voltage Threshold		0.25 x VCC	V		1ms rise Vpad
	I_{IL}	Input Leakage Current	-10	10	µA	VCC/VSS	
	C_{IN}	Input Pin Capacitance		14	pF		
Output	V_{OH}	Output High Voltage Threshold	VCC - 0.45V		V		Only 50ohm mode
	V_{OL}	Output Low Voltage Threshold		0.45	V		Only 50ohm mode
	I_{OH}	Output High Current		3	mA		
	I_{OL}	Output Low Current	-3		mA		
	R_{pu}	WPU 20K Resistance	13k	27k	Ohm		
	R_{pd}	WPD 20K Resistance	13k	27k	Ohm		
Note: For GPIO pads (GP) listed in the Associated Signals below, all functions that are multiplexed on GPIO pads will have the same DC characteristics as the GPIO pads. Refer to the GPIO chapter for the muxed functions on a specific GPIO pad.							
Associated Signals: **GP_B00**/PMC_CORE_VID0;**GP_B01**/PMC_CORE_VID1;**GP_B10**/SIO_I2C5_SCL/PSE_I2C2_SCL/ESPI_ALERT3_N;**GP_B11**/PMC_ALERT_N/PSE_TGPIO06;**GP_B12**/PMC_SLP_S0_N **GP_B13**/PMC_PLTRST_N;**GP_B14**/SPKR/PMC_TGPIO1/SIO_SPI0_CS1_N/PSE_SPI2_CS1_N **GP_B15**/SIO_SPI0_CS0_N/PSE_SPI2_CS0_N/ESPI_CS1_N;**GP_B16**/SIO_SPI0_CLK/PSE_SPI2_CLK **GP_B17**/SIO_SPI0_MISO/PSE_SPI2_MISO;**GP_B18**/SIO_SPI0_MOSI/PSE_SPI2_MOSI;**GP_B19**/SIO_SPI1_CS0_N/PSE_SPI3_CS0_N/ESPI_CS2_N;**GP_B02**/PMC_VRALERT_N/ESPI_ALERT2_N/PSE_TGPIO25 **GP_B20**/SIO_SPI1_CLK/PSE_SPI3_CLK;**GP_B21**/SIO_SPI1_MISO/PSE_SPI3_MISO **GP_B22**/SIO_SPI1_MOSI/PSE_SPI3_MOSI;**GP_B23**/PCHHOT_N/SIO_SPI1_CS1_N/PSE_SPI3_CS1_N/PSE_TGPIO28;**GP_B03**/CPU_GP2/ESPI_ALERT0_N/PSE_TGPIO26;**GP_B04**/CPU_GP3/ESPI_ALERT1_N/PSE_TGPIO27;**GP_B05**/PSE_I2C0_SCL/PSE_TGPIO06;**GP_B06**/PSE_I2C0_SDA/PSE_TGPIO07 **GP_B07**/PSE_I2C1_SCL/PSE_TGPIO08;**GP_B08**/PSE_I2C1_SDA/PSE_TGPIO09;**GP_B09**/SIO_I2C5_SDA/PSE_I2C2_SDA/ESPI_CS3_N;**GP_T00**/PSE_QEPA2/SIO_I2C6_SDA/PSE_TGPIO08 **GP_T01**/PSE_QEPB2/SIO_I2C6_SCL/PSE_TGPIO09;**GP_T10**/PSE_HSUART2_RE **GP_T11**/USB2_OC3_N/PSE_TGPIO06,**GP_T12**/PSE_UART2_RXD/SIO_UART0_RXD **GP_T13**/PSE_UART2_TXD/SIO_UART0_TXD;**GP_T14**/PSE_UART2_RTS_N/SIO_UART0_RTS_N/PSE_HSUART2_DE;**GP_T15**/PSE_UART2_CTS_N/SIO_UART0_CTS_N;**GP_T02**/PSE_QEPI2/SIO_I2C7_SDA/PSE_TGPIO07;**GP_T03**/SIO_I2C7_SCL/PSE_TGPIO06;**GP_T04**/PSE_GBE0_INT;**GP_T05**/PSE_GBE0_RST_N **GP_T06**/PSE_GBE0_AUXTS/USB2_OC1_N;**GP_T07**/PSE_GBE0_PPS/PSE_TGPIO59;**GP_T08**/USB2_OC2_N/PSE_TGPIO22;**GP_T09**/PSE_HSUART2_EN							
3.3V Operation VCC = VCC_3P3A							

Table 36-5. Single-Ended Signal DC Characteristics as Inputs or Outputs (Sheet 8 of 11)

Type	Symbol	Parameter	Minimum	Maximum	Unit	Condition	Notes
Input	V_{IH}	Input High Voltage Threshold	0.75 x VCC		V		1ms rise Vpad
	V_{IL}	Input Low Voltage Threshold		0.25 x VCC	V		1ms rise Vpad
	I_{IL}	Input Leakage Current	-12	12	µA	VCC/VSS	
	C_{IN}	Input Pin Capacitance		10	pF		
Output	V_{OH}	Output High Voltage Threshold	VCC -0.45V		V		1ms rise Vpad
	V_{OL}	Output Low Voltage Threshold		0.45	V		1ms rise Vpad
	I_{OH}	Output High Current		3	mA		
	I_{OL}	Output Low Current	-3		mA		
	R_{pu}	WPU 20K Resistance	13k	27k	Ohm		
	R_{pd}	WPD 20K Resistance	13k	27k	Ohm		
1.8V Operation ***Note:*** VCC = VCC_1P8A							
Input	V_{IH}	Input High Voltage Threshold	0.75 x VCC		V		1ms rise Vpad
	V_{IL}	Input Low Voltage Threshold		0.25 x VCC	V		1ms rise Vpad
	I_{IL}	Input Leakage Current	-12	12	µA	VCC/VSS	
	C_{IN}	Input Pin Capacitance		10	pF		
Output	V_{OH}	Output High Voltage Threshold	VCC -0.45V		V		Only 50 ohm mode
	V_{OL}	Output Low Voltage Threshold		0.45	V		Only 50 ohm mode
	I_{OH}	Output High Current		3	mA		
	I_{OL}	Output Low Current	-3		mA		
	R_{pu}	WPU 20K Resistance	13k	27k	Ohm		
	R_{pd}	WPD 20K Resistance	13k	27k	Ohm		

Table 36-5. Single-Ended Signal DC Characteristics as Inputs or Outputs (Sheet 9 of 11)

Type	Symbol	Parameter	Minimum	Maximum	Unit	Condition	Notes
Type	Symbol	Parameter	Minimum	Maximum	Unit	Condition	Notes
1. **Note:** For GPIO pads (GP) listed in the Associated Signals below, all functions that are multiplexed on GPIO pads will have the same DC characteristics as the GPIO pads. Refer to the GPIO Chapter for the muxed functions on a specific GPIO pad.							
Associated Signals: GP_C00/SMB_CLK/PSE_I2C3_SCL/PSE_TGPIO18;**GP_C01**/SMB_DATA/PSE_I2C3_SDA/PSE_TGPIO19;**GP_C10**/PSE_TGPIO05;**GP_C11**/PSE_HSUART0_RE;**GP_C12**/PSE_UART0_RXD/SIO_UART1_RXD;**GP_C13**/PSE_UART0_TXD/SIO_UART1_TXD;**GP_C14**/PSE_UART0_RTS_N/PSE_HSUART0_DE/SIO_UART1_RTS_N;**GP_C15**/PSE_UART0_CTS_N/SIO_UART1_CTS_N;**GP_C16**/GBE_MDIO/PSE_UART3_RXD/SIO_I2C0_SDA;**GP_C17**/GBE_MDC/PSE_UART3_TXD/SIO_I2C0_SCL;**GP_C18**/PSE_I2C4_SDA/SML_DATA0/SIO_I2C1_SDA;**GP_C19**/PSE_I2C4_SCL/SML_CLK0/SIO_I2C1_SCL;**GP_C02**/PSE_PWM00/SMB_ALERT_N/PSE_TGPIO29;**GP_C20**/PSE_UART4_RXD/SIO_UART2_RXD;**GP_C21**/PSE_UART4_TXD/SIO_UART2_TXD; **GP_C22**/PSE_UART4_RTS_N/ISI_SPIM_MOSI/SIO_UART2_RTS_N; **GP_C23**/PSE_UART4_CTS_N/ISI_SPIS_MOSI/SIO_UART2_CTS_N;**GP_C03**/PSE_GBE0_MDC/PSE_HSUART3_EN;**GP_C04**/PSE_GBE0_MDIO/PSE_UART3_RTS_N/PSE_HSUART3_DE;**GP_C05**/PSE_PWM01/PSE_UART3_CTS_N/SML_ALERT0_N/PSE_TGPIO30;**GP_C06**/PSE_GBE1_MDC;**GP_C07**/PSE_GBE1_MDIO/PSE_HSUART3_RE;**GP_C08**/PSE_TGPIO04/DNX_FORCE_RELOAD;**GP_C09**/PSE_HSUART0_EN;**GP_D00**/PSE_QEPA0/PSE_SPI1_CS0_N/PSE_TGPIO32;**GP_D01**/PSE_QEPB0/PSE_SPI1_CLK/PSE_TGPIO33;**GP_D10**/PSE_SPI0_CLK/SIO_SPI2_CLK/PSE_TGPIO11;**GP_D11**/PSE_SPI0_MISO/SIO_SPI2_MISO/PSE_TGPIO12;**GP_D12**/PSE_SPI0_MOSI/SIO_SPI2_MOSI/PSE_TGPIO13;**GP_D13**/PSE_QEPA1/PSE_TGPIO37;**GP_D14**/PSE_QEPB1/PSE_TGPIO38;**GP_D15**/PSE_PWM03/SIO_SPI2_CS1_N/PSE_SPI0_CS1_N/PSE_TGPIO39;**GP_D16**/PSE_QEPI1/PSE_TGPIO40;**GP_D17**/PSE_PWM04/ISI_SPIM_MOSI/PSE_TGPIO41;**GP_D18**/PSE_PWM05/ISI_SPIS_MOSI/PSE_TGPIO42;**GP_D19**/AVS_I2S_MCLK1/PSE_TGPIO43;**GP_D02**/PSE_QEPI0/PSE_SPI1_MISO/PSE_TGPIO34;**GP_D03**/PSE_PWM06/PSE_SPI1_MOSI/PSE_TGPIO35;**GP_D04**/PSE_PWM02/PSE_SPI1_CS1_N/PSE_TGPIO36;**GP_D05**/PCIE_CLKREQ0_N;**GP_D06**/PCIE_CLKREQ1_N;**GP_D07**/PCIE_CLKREQ2_N;**GP_D08**/PCIE_CLKREQ3_N;**GP_D09**/PSE_SPI0_CS0_N/SIO_SPI2_CS0_N/PSE_TGPIO10;**GP_H00**/PSE_GBE1_INT/PSE_UART5_RXD,**GP_H01**/PSE_GBE1_RST_N/PSE_UART5_TXD;**GP_H10**/PCIE_CLKREQ4_N/PSE_PWM14;**GP_H11**/PCIE_CLKREQ5_N/PSE_PWM15;**GP_H12**/PSE_UART1_RXD/M2_SKT2_CFG0/PSE_TGPIO51;**GP_H13**/PSE_UART1_TXD/M2_SKT2_CFG1/PSE_TGPIO52;**GP_H14**/M2_SKT2_CFG2/PSE_TGPIO53;**GP_H15**/PSE_UART1_CTS_N/M2_SKT2_CFG3/PSE_TGPIO54;**GP_H16**/PCIE_LNK_DOWN/DDI2_DDC_SCL;**GP_H17**/SD_SDIO_PWR_EN_N;**GP_H18**/PMC_CPU_C10_GATE_N;**GP_H19**/DDI2_DDC_SDA/PMC_TGPIO0/PSE_TGPIO20;**GP_H02**/PSE_GBE1_AUXTS/PSE_UART5_RTS_N;**GP_H20**/PSE_PWM07/DDI2_HPD/PSE_TGPIO55;**GP_H21**/PSE_HSUART1_DE/PSE_UART1_RTS_N/PSE_TGPIO56;**GP_H22**/PSE_HSUART1_RE/PSE_TGPIO57;**GP_H23**/PSE_HSUART1_EN/PSE_TGPIO58;**GP_H03**/PSE_GBE1_PPS/PSE_UART5_CTS_N/PSE_TGPIO21;**GP_H04**/SIO_I2C2_SDA/PSE_PWM08/PSE_TGPIO10;**GP_H05**/SIO_I2C2_SCL/PSE_PWM09/PSE_TGPIO11;**GP_H06**/SIO_I2C3_SDA/PSE_I2C5_SDA/PSE_PWM10;**GP_H07**/SIO_I2C3_SCL/PSE_I2C5_SCL/PSE_PWM11;**GP_H08**/SIO_I2C4_SDA/PSE_PWM12;**GP_H09**/SIO_I2C4_SCL/PSE_PWM13;**GP_U00**/GBE_INT/PSE_I2C6_SCL;**GP_U01**/GBE_RST_N/PSE_I2C6_SDA;**GP_U10**/ISI_SPIS_MISO/ISI_I2CS_SDA/PSE_TGPIO12;**GP_U11**/PSE_QEPB3/PSE_TGPIO11;**GP_U12**/ISI_CHX_OKNOK_0;**GP_U13**/ISI_CHX_OKNOK_1;**GP_U14**/ISI_CHX_RLY_SWTCH;**GP_U15**/ISI_CHX_PMIC_EN/PSE_TGPIO13;**GP_U16**/ISI_OKNOK_0;**GP_U17**/ISI_OKNOK_1;**GP_U18**/ISI_ALERT_N;**GP_U19**/PSE_QEPI3/PSE_TGPIO12;**GP_U02**/GBE_PPS/PSE_I2C7_SCL;**GP_U03**/GBE_AUXTS/PSE_I2C7_SDA;**GP_U04**/ISI_SPIM_CS/PSE_SPI1_CS0_N;**GP_U05**/ISI_SPIM_SCLK/PSE_SPI1_CLK;**GP_U06**/ISI_SPIM_MISO/PSE_SPI1_MISO;**GP_U07**/PSE_QEPA3/PSE_SPI1_MOSI/PSE_TGPIO10;**GP_U08**/ISI_SPIS_CS/PSE_TGPIO10;**GP_U09**/ISI_SPIS_SCLK/ISI_I2CS_SCL/PSE_TGPIO11							
3.3V Operation ***Notes:*** VCC = VCC_3P3A							
Input	V_{IH}	Input High Voltage Threshold	0.75 x VCC		V		1ms rise Vpad
	V_{IL}	Input Low Voltage Threshold		0.25 x VCC	V		1ms rise Vpad
	I_{IL}	Input Leakage Current	-12	12	µA	VCC/VSS	
	C_{IN}	Input Pin Capacitance		10	pF		

Table 36-5. Single-Ended Signal DC Characteristics as Inputs or Outputs (Sheet 10 of 11)

Type	Symbol	Parameter	Minimum	Maximum	Unit	Condition	Notes
Output	V_{OH}	Output High Voltage Threshold	VCC - 0.45V		V		
	V_{OL}	Output Low Voltage Threshold		0.45	V		
	I_{OH}	Output High Current		3	mA		
	I_{OL}	Output Low Current	-3		mA		
	R_{pu}	WPU 20K Resistance	13k	27k	Ohm		
	R_{pd}	WPD 20K Resistance	13k	27k	Ohm		
1.8V Operation ***Note:*** VCC = VCC_1P8A							
Input	V_{IH}	Input High Voltage Threshold	0.75 x VCC		V		1ms rise Vpad
	V_{IL}	Input Low Voltage Threshold		0.25 x VCC	V		1ms rise Vpad
	I_{IL}	Input Leakage Current	-12	12	µA	VCC/VSS	
	C_{IN}	Input Pin Capacitance		10	pF		
Output	V_{OH}	Output High Voltage Threshold	VCC - 0.45V		V		
	V_{OL}	Output Low Voltage Threshold		0.45	V		
	I_{OH}	Output High Current		3	mA		
	I_{OL}	Output Low Current	-3		mA		
	R_{pu}	WPU 20K Resistance	13k	27k	Ohm		
	R_{pd}	WPD 20K Resistance	13k	27k	Ohm		
Note: The Signals Below Utilize GPIO Buffers But Do Not Have a GPIO Signal Multiplexed With them.							
Associated Signals: PCH_JTAG_TDO; PCH_JTAGX; PCH_JTAG_TRST_N; PCH_JTAG_TDI; PCH_JTAG_TMS PCH_JTAG_TCK; THRMTRIP_N, VCCST_OVERRIDE, PROCHOT_N, CATERR_N, PROC_PWR_GD							
1.05V Operation ***Notes:*** VCC = VCC_IN_ST.							

Table 36-5. Single-Ended Signal DC Characteristics as Inputs or Outputs (Sheet 11 of 11)

Type	Symbol	Parameter	Minimum	Maximum	Unit	Condition	Notes
Input	VIH	Input High Voltage Threshold	JTAG: 0.8 x VCC Non-JTAG: 0.7 x VCC		V		
	VIL	Input Low Voltage Threshold		JTAG: 0.3x VCC Non-JTAG: 0.3 x VCC	V		
	IIL	Input Leakage Current	-10	10	µA		
	CIN	Input Pin Capacitance		2	pF		
Output	V_{OH} (All JTAG except TDO & TMS)	Output High Voltage Threshold	0.75		V		
	V_{OL} (ALL JTAG except TDO & TMS))	Output Low Voltage Threshold		0.25	V		
	IOH (VCCST_ OVERRIDE, PROC_P-WR_GD)	Output High Current		-5	mA		
	IOL (VCCST_ OVERRIDE, PROC_P-WR_GD)	Output Low Current	5		mA		
	V_{OH} (VCCST_OVERRIDE, PROC_P-WR_GD)	Output High Voltage Threshold	0.79		V		
	V_{OL} (VCCST_OVERRIDE, PROC_P-WR_GD)	Output Low Voltage Threshold		0.26	V		
	R_{pu}	WPU 20K Resistance	14k	26k	Ohm		
	R_{pd}	WPD 20K Resistance	14k	26k	Ohm		
Note:	THRMTRIP_N, PROCHOT_N and CATERR_N are input signals. Refer to Table 36-7 for output values.						

36.3.2 CMOS DC Specifications (Compute Die)

Table 36-6. CMOS Signal Group DC Specifications

Associated Signals: SVID_ALERT_N, VCCST_PWRGD; Open Drain (EDP_UTILS,MDSI_DE_TE_1, PROC_PWR_GD					
Symbol	**Parameter**	**Minimum**	**Maximum**	**Units**	**Notes/Figure**
V_{IL}	Input Low Voltage	-	Vcc*0.3	V	2
V_{IH}	Input High Voltage	Vcc*0.7	-	V	2, 4
R_{ON}	Buffer on Resistance	20	70	Ω	-
I_{LI}	Input Leakage Current	-	±150	µA	3

Notes:

1. Unless otherwise noted, all specifications in this table apply to all processor frequencies.
2. The Vcc referred to in these specifications refers to instantaneous VCCIO or VCC_IN_STG.
3. For VIN between “0” V and Vcc. Measured when the driver is tri-stated.
4. VIH may experience excursions above Vcc. However, input signal drivers should comply with the signal quality specifications.
5. Refer the processor I/O Buffer Models for I/V characteristics

36.3.3 GTL and OD DC Specification (Compute Die)

Table 36-7. GTL Signal Group and Open Drain (OD) Signal Group DC Specifications

Associated Signals: GTL: BPM[3:0]_N, JTAG_PREQ_N, CPU_JTAG_TMS, CPU_JTAG_TRST_N, CPU_JTAG_TDI, CPU_JTAG_TCK, PROCHOT_N, CFG_[15:00], CFG_[18,16], CFG_[19,17], SVID_DATA

Open Drain: JTAG_PRDY_N, CPU_JTAG_TDO, CATERR_N, THRMTRIP_N, PROCHOT_N, SVID_CLK, SVID_DATA

Symbol	Parameter	Minimum	Maximum	Units	Notes/ Figure
VIL	Input Low Voltage (PROC_JTAG, except PROC_JTAG_TCK, PROC_JTAG_TRST_N)	-	0.6*Vcc	V	2
VIH	Input High Voltage (PROC_JTAG, except PROC_JTAG_TCK, PROC_JTAG_TRST_N)	0.72*Vcc	-	V	2, 4
VIL	Input Low Voltage (PROC_JTAG_TCK,PROC_JTAG_TRST_N)	-	0.3*Vcc	V	2
VIH	Input High Voltage (PROC_JTAG_TCK,PROC_JTAG_TRST_N)	0.7*Vcc	-	V	2, 4
VOL	Output Low Voltage (CATERR_N, THRMTRIP_N)		0.26	V	6
IOL	Output Low Current (CATERR_N, THRMTRIP_N)		14.6	mA	6
VHYSTERESIS	Hysteresis Voltage	0.2*Vcc	-	V	-
RON	Buffer on Resistance (TDO, THRMTRIP_N and CATERR_N) - OD	7	17	Ω	-
RON PD	Buffer on Resistance (THRMTRIP_N and CATERR_N)		17	Ω	
VIL	Input Low Voltage (other GTL)	-	0.6*Vcc	V	2
VIH	Input High Voltage (other GTL)	0.72*Vcc	-	V	2, 4
RON	Buffer on Resistance (BPM, other GTL)	16	24	Ω	-
RON	Buffer on Resistance (PROCHOT_N) - OD	12	28	Ω	-
ILI	Input Leakage Current	-	±250	µA	3

Notes:
1. Unless otherwise noted, all specifications in this table apply to all processor frequencies.
2. The Vcc referred to in these specifications refers to instantaneous VCCIO or VCC_IN_STG.
3. For VIN between 0V and Vcc. Measured when the driver is tri-stated.
4. VIH and VOH may experience excursions above Vcc. However, input signal drivers should comply with the signal quality specifications.
5. Refer the processor I/O Buffer Models for I/V characteristics.
6. VOH and IOH specification does NOT apply to Open Drain signals.

36.3.4 Display Port* Transmitter DC Specification

Table 36-8. Display Port* Transmitter DC Specification

Symbol	Parameter	Minimum	Maximum	Units	Notes/Figure
VIL	Aux Input Low Voltage	-	0.8	V	

Symbol	Parameter	Minimum	Maximum	Units	Notes/Figure
VIH	Aux Input High Voltage	2.25	3.6	V	
VOL	Output Low Voltage	-	0.25*VCCIO	V	1
VOH	Output High Voltage	0.75*VCCIO	-	V	1
ZTX-DIFF-DC	DC Differential Tx Impedance	100	120	Ω	
Notes:					
1. VOL and VOH levels depends on the level chosen by the Platform.					

36.3.5 HDMI* DC Specification

Table 36-9. HDMI* DC Specification

Symbol	Parameter	Minimum	Maximum	Units	Notes/ Figure
VOL	DDI[0:2]_TXP[3:0] Output Low Voltage DDI[0:2]_TXN[3:0] Output Low Voltage	-	0.25*VCCIO	V	1
VOH	DDI[0:2]_TXP[3:0] Output High Voltage DDI[0:2]_TXN[3:0] Output High Voltage	0.75*VCCIO	-	V	1
ZTX-DIFF-DC	DC Differential Tx Impedance	100	120	Ω	
Notes:					
1. VOL and VOH levels depends on the level chosen by the Platform.					

36.3.6 Embedded Display Port* DC Specification

Table 36-10. Embedded Display Port* DC Specification

Symbol	Parameter	Minimum	Maximum	Units	Notes/Figure
V_{OL}	EDP_UTILS Output Low Voltage	—	—	0.1*VCCIO	V
V_{OH}	EDP_UTILS Output High Voltage	0.9*VCCIO	—	—	V
R_{UP}	EDP_UTILS Internal pull-up	45	—	—	Ω
R_{DOWN}	EDP_UTILS Internal pull-down	45	—	—	Ω

36.3.7 MIPI*-DSI DC Specification

Table 36-11. MIPI*-DSI DC Specification

Symbol	Parameter	Minimum	Maximum	Units	Notes/Figure
ILEAK	Pin Leakage current	-10	10	µA	6
VGNDSH	Ground Shift	-50	50	mV	
V_{CMTX}	HS transmit static common-mode voltage	150	250	mV	1

Symbol	Parameter	Minimum	Maximum	Units	Notes/Figure
$\|V_{CMTX(1,0)}\|$	VCMTX mismatch when output is differential-1 or differential-0	-	5	mV	2
$\|V_{OD}\|$	HS transmit differential voltage	140	270	mV	1
$\|\Delta V_{OD}\|$	VOD mismatch when output is Differential-1 or Differential-0	-	14	mV	2
V_{OHHS}	HS output high voltage	-	360	mV	1
Z_{OS}	Single-ended output impedance	40	62.5	Ω	
ΔZ_{OS}	Single-ended output impedance mismatch	-	10	%	
V_{OH}	Thevenin output high level	1.1	1.3	V	3
		0.95	1.3	V	4
V_{OL}	Thevenin output low level	-50	50	mV	
Z_{OLP}	Output impedance of LP transmitter	110		Ω	5
V_{IH}	Logic 1 input voltage	880		mV	
V_{IL}	Logic 0 input voltage, not in ULP state		550	mV	
V_{HYST}	Input hysteresis	25		mV	
V_{IHCD}	Logic 1 Contention threshold	450		mV	
V_{ILCD}	Logic 0 Contention threshold		200	mV	

Notes:

1. Value when driving into load impedance anywhere in the ZID range.

2. A transmitter should minimize ΔVOD and ΔVCMTX(1,0) in order to minimize radiation, and optimize signal integrity.

3. Applicable when the supported data rate <= 1.5 Gbps.
4. Applicable when the supported data rate > 1.5 Gbps.
5. Though no maximum value for ZOLP is specified, the LP transmitter output impedance shall ensure the TRLP/TFLP specification is met.
6. When the pad voltage is in the signal voltage range from VGNDSH (voltage ground shift), MIN to VOH + VGNDSH,MAX and the Lane Module is in LP receive mode.

36.3.8 Memory Specifications

36.3.8.1 DDR4 DC Specification

Table 36-12. DDR4 Signal Group DC Specifications (Sheet 1 of 2)

Symbol	Parameter	Minimum	Maximum	Units	Notes[1]
VIL	Input Low Voltage		0.68*VDDQ	V	2, 3, 4,9
VIH	Input High Voltage	0.82*VDDQ		V	2, 3, 4,9,13
IIL	Input Leakage Current (DQ, CK) 0 V 0.2*Vddq 0.8*Vddq	-	1.1	mA	9
RON_UP(DQ)	Data Buffer pull-up Resistance	25	60	Ω	5, 11
RON_DN(DQ)	Data Buffer pull-down Resistance	26	75	Ω	5, 11
RODT(DQ)	On-die termination equivalent resistance for data signals	25	Hi-Z	Ω	6, 11
VODT(DC)	On-die termination DC working point (driver set to receive mode)	0.7*VDDQ	0.8*VDDQ	V	12
RON_UP(CK)	Clock Buffer pull-up Resistance	25	60	Ω	5,11
RON_DN(CK)	Clock Buffer pull-down Resistance	25	75	Ω	5,11
RON_UP(CMD)	Command Buffer pull-up Resistance	23	50	Ω	5,11
RON_DN(CMD)	Command Buffer pull-down Resistance	24	57	Ω	5,11
RON_UP(CTL)	Control Buffer pull-up Resistance	23	50	Ω	5,11
RON_DN(CTL)	Control Buffer pull-Down Resistance	24	57	Ω	5,11
RON_UP (DDR_VTT_CTL)	System Memory Power Gate Control Buffer Pull-up Resistance	45	125	Ω	
RON_DN (DDR_VTT_CTL)	System Memory Power Gate Control Buffer Pull- down Resistance	40	130	Ω	
DDR0_VREF_DQ DDR1_VREF_DQ DDR_VREF_CA	VREF output voltage	Trainable	Trainable	V	10
DDR_RCOMP[0]	ODT resistance compensation	99	101	Ω	8
DDR_RCOMP[1]	Data resistance compensation	99	101	Ω	8
DDR_RCOMP[2]	Command resistance compensation	99	101	Ω	8

Table 36-12. DDR4 Signal Group DC Specifications (Sheet 1 of 2)

Symbol	Parameter	Minimum	Maximum	Units	Notes[1]

Notes:
1. Unless otherwise noted, all specifications in this table apply to all processor frequencies.Timing specifications only depend on the operating frequency of the memory channel and not the maximum rated frequency.
2. V_{IL} is defined as the maximum voltage level at a receiving agent that will be interpreted as a logical low value.
3. V_{IH} is defined as the minimum voltage level at a receiving agent that will be interpreted as a logical high value.
4. V_{IH} and V_{OH} may experience excursions above V_{DDQ}. However, input signal drivers should comply with the signal quality specifications.
5. Pull up/down resistance after compensation (assuming ±5% COMP inaccuracy).
Note: BIOS power training may change these values significantly based on margin/power trade-off.
6. ODT values after COMP (assuming ±5% inaccuracy). BIOS MRC can reduce ODT strength towards.
7. The minimum and maximum values for these signals are programmable by BIOS to one of the two sets.
8. DDR_RCOMP resistance should be provided on the system board with 1% resistors. SM_RCOMP[x] resistors are to VSS. Values are pre-silicon estimations and are subject to change.
9. PMC_DRAM_RESET_N must have a maximum of 15 ns rise or fall time over VDDQ * 0.30 ±100 mV and the edge must be monotonic.
10. DDR_[1:0]_VREF_CA is defined as VDDQ/2 for DDR4.
11. R_{ON} tolerance is preliminary and might be subject to change.
12. Max-min range is correct but center point is subject to change during MRC boot training.
13. Processor may be damaged if V_{IH} exceeds the maximum voltage for extended periods.

36.3.8.2 LPDDR4/x Memory Controller DC Specification

Table 36-13. LPDDR4/x DC Specifications

Symbol	Parameter	Minimum	Maximum	Units	Notes
VIL	Input Low Voltage		0.08*VDDQ	V	2, 3, 4,9
VIH	Input High Voltage	0.35*VDDQ		V	2, 3, 4,9,13
IIL	Input Leakage Current(DQ, CK) 0 V 0.2*VDDQ 0.8*VDDQ	-	1	mA	9
$R_{ON\_UP(DQ)}$	Data Buffer pull-up Resistance	25 (LP4x:23)	60 (LP4x:58)	Ω	5,11
$R_{ON\_DN(DQ)}$	Data Buffer pull-down Resistance	25 (LP4x:26)	72 (LP4x:85)	Ω	5,11
$R_{ODT(DQ)}$	On-die termination equivalent resistance for data signals	28 (LP4x:26)	Hi-Z	Ω	6, 11
$V_{ODT(DC)}$	On-die termination DC working point (driver set to receive mode)	0.15*Vddq (LP4x: 0.25* Vddq)	0.25*Vddq (LP4x:0.35 * Vddq)	V	-
$R_{ON\_UP(CK)}$	Clock Buffer pull-up Resistance	24 (LP4x:30)	60 (LP4x:59)	Ω	5, 11
$R_{ON\_DN(CK)}$	Clock Buffer pull-down Resistance	28	92 (LP4x:94)	Ω	5, 11
$R_{ON\_UP(CMD)}$	Command Buffer pull-up Resistance	26	50	Ω	5, 11
$R_{ON\_DN(CMD)}$	Command Buffer pull-down Resistance	22 (LP4x:20)	67	Ω	5, 11
$R_{ON\_UP(CTL)}$	Control Buffer pull-up Resistance	26	50	Ω	5, 11
$R_{ON\_DN(CTL)}$	Control Buffer pull-down Resistance	22 (LP4x:20)	67	Ω	5, 11
DDR0_VREF_DQ DDR1_VREF_DQ DDR_VREF_CA	VREF output voltage	Trainable	Trainable	V	10
DDR_RCOMP[0]	ODT resistance compensation	99	101	Ω	8
DDR_RCOMP[1]	Data resistance compensation	99	101	Ω	8
DDR_RCOMP[2]	Command resistance compensation	99	101	Ω	8

Table 36-13. LPDDR4/x DC Specifications

Symbol	Parameter	Minimum	Maximum	Units	Notes
Notes: 1. Unless otherwise noted, all specifications in this table apply to all processor frequencies.Timing specifications only depend on the operating frequency of the memory channel and not the maximum rated frequency. 2. VIL is defined as the maximum voltage level at a receiving agent that will be interpreted as a logical low value. 3. VIH is defined as the minimum voltage level at a receiving agent that will be interpreted as a logical high value. 4. VIH and VOH may experience excursions above VDDQ. However, input signal drivers should comply with the signal quality specifications. 5. Pull up/down resistance after compensation (assuming ±5% COMP inaccuracy). Note that BIOS power training may change these values significantly based on margin/power trade-off. 6. ODT values after COMP (assuming ±5% inaccuracy). BIOS MRC can reduce ODT strength towards 7. The minimum and maximum values for these signals are programmable by BIOS to one of the two sets. 8. LP4_RCOMP resistance should be provided on the system board with 1% resistors. SM_RCOMP[x] resistors are to VSS. Values are pre-silicon estimations and are subject to change. 9. PMC_DRAM_RESET_N must have a maximum of 15 ns rise or fall time over VDDQ * 0.30 ±100 mV and the edge must be monotonic. 10. SM_VREF is defined as VDDQ/2 for LPDDR4/LPDDR4x. 11. RON tolerance is preliminary and might be subject to change. 12. Max-min range is correct but center point is subject to change during MRC boot training. 13. Processor may be damaged if VIH exceeds the maximum voltage for extended periods.					

36.3.9 USB

36.3.9.1 USB 2.0 DC Specification (Low-Speed (LS)/Full-Speed (FS)/High Speed (HS)

Table 36-14. USB 2.0 Host DC Specification

Symbol	Parameter	Minimum	Maximum	Units	Notes/Figure
VDI	Differential Input Sensitivity (LS & FS Only)	0.2	-	V	VDI = | USBPx[P] – USBPx[N] |
VCM	Differential Common Mode Range (LS & FS Only)	0.8	2.5	V	Includes VDI range
VSE	Single-Ended Receiver Threshold (LS & FS Only)	0.8	2	V	
VCRS	Output Signal Crossover Voltage (LS & FS Only)	1.3	2	V	10
VOL	Output Low Voltage (LS & FS Only)	-	0.4	V	At Iol = 2mA
VOH	Output High Voltage (LS & FS Only)	3.3 - 0.5	-	V	At Ioh = -2 mA
VHSSQ	High-speed squelch detection threshold (differential signal amplitude) (HS Only)	100	150	mV	
VHSDSC	High speed disconnect detection threshold (differential signal amplitude) (HS Only)	525	625	mV	
VHSCM	High-speed data signaling common mode voltage range (guideline for receiver) (HS Only)	-50	500	mV	
VHSOI	High-speed idle level (HS Only)	-10	10	mV	
VHSOH	High-speed data signaling high (HS Only)	360	440	mV	
VHSOL	High-speed data signaling low (HS Only)	-10	10	mV	
VCHIRPJ	Chirp J level (differential voltage) (HS Only)	700	1100	mV	
VCHIRPK	Chirp K level (differential voltage) (HS Only)	-900	-500	mV	

36.3.9.2 USB 3.1 DC Specificatio

Table 36-15. USB 3.1 Interface DC Specification

Symbol	Parameter	Minimum	Maximum	Units	Notes/Figure
VTX-DIFF-PP	Differential Peak to Peak Output Voltage	0.8	1.2	V	
VTX-DIFF-PP- LOW	Low power differential Peak to Peak Output Voltage	0.4	1.2	V	
VTX_CM-Acp-p	TX AC Common Mode Output Voltage (5GT/s)	-	100	mV	
ZTX-DIFF-DC	DC Differential TX Impedance	72	120	Ohm	
VRX_CM-ACp	AC peak Common Mode Input Voltage	-	150	mV	
VRX-DIFF p-p	Differential Input Peak to Peak Voltage	0.1	1.2	V	

36.3.10 PCIe* Specification

36.3.10.1 PCIe* DC Specification

Symbol	Parameter	Minimum	Maximum	Units
VTX-DIFF P-P	Differential Peak to Peak Output Voltage [Gen 1]	0.8	1.2	V
VTX-DIFF P-P - Low	Low power differential Peak to Peak Output Voltage [Gen 1]	0.4	1.2	V
VTX_CM-ACp	TX AC Common Mode Output Voltage (2.5 GT/s) [Gen 1]	-	20	mV
ZTX-DIFF-DC	DC Differential TX Impedance [Gen 1]	80	120	Ohm
VRX-DIFF p-p	Differential Input Peak to Peak Voltage [Gen 1]	0.175	1.2	V
VRX_CM-ACp	AC peak Common Mode Input Voltage [Gen 1]	-	150	mV
VTX-DIFF P-P	Differential Peak to Peak Output Voltage [Gen 2]	0.8	1.2	V
VTX-DIFF P-P - Low	Low power differential Peak to Peak Output Voltage [Gen 2]	0.4	1.2	V
VTX_CM-ACp	TX AC Common Mode Output Voltage (5 GT/s) [Gen 2]	-	150	mVPP
ZTX-DIFF-DC	DC Differential TX Impedance [Gen 2]	80	120	Ohm
VRX-DIFF p-p	Differential Input Peak to Peak Voltage [Gen 2]	0.12	1.2	V
VRX_CM-ACp	AC peak Common Mode Input Voltage [Gen 2]	-	150	mV
VTX-DIFF P-P	Differential Peak to Peak Output Voltage [Gen 3]	0.8	1.3	V
VTX-DIFF P-P - Low	Low power differential Peak to Peak Output Voltage [Gen 3]	0.4	1.2	V
VTX_CM-ACp	TX AC Common Mode Output Voltage (5 GT/s) [Gen 3]	-	150	mVPP
ZTX-DIFF-DC	DC Differential TX Impedance [Gen 3]	80	120	Ohm
VRX-DIFF p-p	Differential Input Peak to Peak Voltage [Gen 3]	Refer to Stressed Voltage Eye Parameters Table in PCIe* Gen 3 industry specifications.	-	V
VRX_CM-ACp	AC peak Common Mode Input Voltage [Gen 3]	-	150	mV

36.3.11 SATA Specification

Table 36-16. SATA DC Specification

Symbol	Spec	Min Spec	Max Spec	Unit
VIMIN-Gen1i	Minimum Input Voltage - 1.5Gb/s internal SATA	325	-	mVdiff p-p
VIMAX-Gen1i	Maximum Input Voltage - 1.5Gb/s internal SATA	-	600	mVdiff p-p
VOMIN-Gen1i	Minimum Output Voltage 1.5Gb/s internal	400	-	mVdiff p-p
VOMAX-Gen1i	Maximum Output Voltage 1.5Gb/s internal	-	600	mVdiff p-p
VIMIN-Gen2i	Minimum Input Voltage - 3.0Gb/s internal SATA	275	-	mVdiff p-p
VIMAX-Gen2i	Maximum Input Voltage - 3.0Gb/s internal SATA	-	750	mVdiff p-p
VOMIN-Gen2i	Minimum Output Voltage 3.0Gb/s internal	400	-	mVdiff p-p
VOMAX-Gen2i	Maximum Output Voltage 3.0Gb/s internal	-	700	mVdiff p-p
VIMIN-Gen3i	Minimum Input Voltage - 6.0Gb/s internal SATA	240	-	mVdiff p-p
VIMAX-Gen3i	Maximum Input Voltage - 6.0Gb/s internal SATA	-	1000	mVdiff p-p
VOMIN-Gen3i	Minimum Output Voltage 6.0Gb/s internal SATA	200	-	mVdiff p-p
VOMAX-Gen3i	Maximum Output Voltage 6.0Gb/s internal SATA	-	900	mVdiff p-p

§ §

37 Terminology

Table 37-1. Terminology (Sheet 1 of 4)

Term	Description
ACPI	Advanced Configuration & Power Interface
AHB	Advanced High-performance Bus
AMP	Asymmetric Multi-Processing
APB	Advanced Peripheral Bus
ARP	Address Resolution Protocol
ARP	Address Resolution Protocol
ART	Always Running Timer
ART	Always Running Timer
ARU	Always Running Unit
ASIL	Automotive Safety Integrity Levels
ATB	Advanced Trace Bus
ATT	Address Translation Table
BAR	Base Address Register
BCD	Binary Coded Decimal
BCET	Best Case Execution Timers
BDSL	Boundary Scan Description Language
BIOS	Basic Input Output System
BSSB	Boundary Scan Side Band (also known as OOB)
CAN	Control Area Network
CCU	Clock Control Unit
CFIO	Configuration Flexible IO
CMOS	Complementary Metal Oxide Semiconductor. A manufacturing process used to produce electronics circuits, but in reference to RTC is used interchangeably as the RTC's RAM i.e. clearing CMOS meaning to clear RTC RAM.
COE	Checksum Offload Engine
CRC	Cyclic Redundancy Check
CSE	Converged Security Engine
DAP	Debug Access Port
DbC	Debug Class
DCI	Debug Connect Interface
DCLS	Dual Core Lock Step
DFP	Downstream Facing Port
DLL	Delay Logic Levels
DMA	Direct Memory Access
DNX_FORCE_RELOAD	During Pre-boot phase,to flash SPI via DnX (USB): DNX force reload is used (button press).
DTF	DFT Trace Fabric
ECC	Error Correction Code

Table 37-1. Terminology (Sheet 2 of 4)

Term	Description
ESR	Equivalent Series Resistance. Resistive element in a circuit such as a clock crystal.
ETM	Embedded Trace Macrocell
EU	Execution Unit
FF	Fixed Function
FIA	Flex IO Adapter
FLR	Function Level Reset
FMM_Hub	Fault Management Module Hub
FPU	Floating Point Unit
FTTI	Failure Tolerant Time Interval
FW	Firmware
GbE	Gigabit Ethernet
GEOM	Geometry
GPG	General Purpose Graphic
GPI	General Purpose Input
GTI	Graphic Technology Interface
HIP	Hard Intellectual Property
HW	Hardware
I2C	Inter Integrated Circuit
IC$	Instruction Cache
ICMP	Internet Control Message Protocol
IEEE	Institute of Electrical and Electronics Engineers
IEH	Integrated Error Handler
Intel® PSE	Intel® Programmable Services Engine
Intel® TCC	Intel® Time Coordinated Computing
IOSF	Intel On-Chip System Fabric
IOSF-P	IOSF Primary
IOSF-SB	IOSF Sideband
IPC	Inter-Processor Communication
IRQ	Interrupt Request (usually wire)
ISI	Intel Safety Island
ISR	Interrupt Service Routine
JTAG	Joint Test Action Group
LBIST	Logic Built-In Self-Test
LD/ST	Load/Store
LFM	Latent Fault Metric
LPSS	Low Power Subsystem
MAC	Media Access Control
MBIST	Memory Built-In Self-Test
MCA	Machine Check Architecture
MCTP	Management Component Transport Protocol

Table 37-1. Terminology (Sheet 3 of 4)

Term	Description
MCU	Micro-Controller Unit
MDIO	Management Data Input/Output
MON	(Clock/Voltage) Monitor
MPU	Memory Protection Unit
MSI	Message Signaled Interrupt
NVIC	Nested Vectored Interrupt Controller
OCP	Open Core Protocol
OOB	Out Of Band (also known as BSSB)
OOB	Out Of Band
OS	Operating System
PCH	Platform Controller Hub
PCI	Peripheral Component Interconnect
PCS	Physical Coding Sublayer
PEC	Package Error Checking
PGCB	Power Gating Control Block
PMC	Power Management Controller - dedicated microprocessor running power management firmware
PMHF	Probabilistic Metric for (Random) Hardware Failures
POSH	Position Only Shading
PPM	Parts Per Million. Used to provide crystal accuracy or as a frequency variation indicator
PU	Processing Unit
QEP	Quadrature Encoder Peripheral
RAM	Random Access Memory
RGMII	Reduced Gigabit Media Independent Interface
RS	Root Space
SAI	Security Attribute of Initiator
SB-ATT	IOSF-SB Address Translation Table
SCI	Sycamore Island
SECDED	Single Error Correction Double Error Detection
SFF	Safe Failure Fraction
SGMII	Serial Gigabit Media Independent Interface
SIL	Safety Integrity Levels
SIP	Soft Intellectual Property
SLM	Shared Local Memory
SM	Safety Mechanisms
SMA	Station Management Agent
SMBus	System Management Bus
SoC	System On Chip
SPFM	Single Point Fault Metric
SPI	Serial Peripheral Interface

Table 37-1. Terminology (Sheet 4 of 4)

Term	Description
SR	Safety Related
STA	Station Management
STL	Software Test Library
SW	Software
SWD	Serial Wire Debug
TBI	Ten Bit Interfaces
TC	Traffic Class
TCM	Tightly Coupled Memory
TCP	Transmission Control Protocol
TGP	Tiger Lake PCH
TOUUD	Top of Upper Usable DRAM
TPIU	Trace Port Interface Unit
TSN	Time-Sensitive Networking
TSO	TCP Segmentation Offload
UART	Universal Asynchronous Receiver Transmitter
UDP	User Datagram Protocol
UFP	Upstream Facing Port
VD	Video Display
WDT	Watchdog Timer
WIC	Wake-u Interrupt Controller

§ §