PantherLake Intel® Firmware Support Package (FSP) Integration Guide

Tue Mar 10 2026 00:31:05

By using this document, in addition to any agreements you have with Intel, you accept the terms set forth below. 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.

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Do not finalize a design with this information.

The products described in this document 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.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

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

Any software source code reprinted in this document is furnished for informational purposes only and may only be used or copied and no license, express or implied, by estoppel or otherwise, to any of the reprinted source code is granted by this document.

[When the doc contains software source code for a special or limited purpose (such as informational purposes only), use the conditionalized Software Disclaimer tag. Otherwise, use the generic software source code disclaimer from the Legal page and include a copy of the software license or a hyperlink to its permanent location.]

This document contains information on products in the design phase of development. Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families. Go to: http://www.intel.com/products/processor_number/

Code Names are only for use by Intel to identify products, platforms, programs, services, etc. ("products") in development by Intel that have not been made commercially available to the public, i.e., announced, launched or shipped. They are never to be used as "commercial" names for products. Also, they are not intended to function as trademarks.

Intel, Intel Atom, [include any Intel trademarks which are used in this document] and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

Copyright ©Intel Corporation. All rights reserved.

1 Introduction 1

2 Overview 3

3 FSP Integration 5

4 FSP Dispatch Mode 15

4.1 Dispatch Mode Policy Init . . . 16

4.2 Config Blocks . . . 18

4.3 FSP Error Information . . . 23

4.4 Dispatch Mode Integration . . . 24

5 FSP API Mode 27

5.1 FSP APIs . . . 27

5.2 Reset Return Codes . . . 33

5.3 UPD Porting Guide . . . 33

6 Porting Recommendations 35

7 FSP Output 37

8 POST Codes 43

9 Todo List 51

10 Class Index 53

10.1 Class List . . . 53

11 File Index 63

11.1 File List . . . 63

12 Class Documentation 67

12.1 _CONFIG_BLOCK Struct Reference . . . 67

12.1.1 Detailed Description . . . 68

12.2 _CONFIG_BLOCK_HEADER Struct Reference . . . 68

12.2.1 Detailed Description . . . 69

12.3 _CONFIG_BLOCK_TABLE_STRUCT Struct Reference . . . 69

12.3.1 Detailed Description . . . 70

12.4 _DXE_ITBT_CONFIG Struct Reference . . . 70

12.4.1 Detailed Description . . . 71

12.5 _DXE_ITBT_GENERIC_CONFIG Struct Reference . . . 71

12.5.1 Detailed Description . . . 71

12.6 _EFI_MM_RESERVED_MMRAM_REGION Struct Reference . . . 71

12.6.1 Detailed Description . . . 72

12.6.2 Member Data Documentation . . . 72

12.6.2.1 MmramReservedSize . . . 72

12.6.2.2 MmramReservedStart . . . 72
12.7 _EFI_PEI_MP_SERVICES_PPI Struct Reference . . . 73
12.7.1 Detailed Description . . . 73
12.8 _ITBT_GENERIC_CONFIG Struct Reference . . . 73
12.8.1 Detailed Description . . . 74
12.8.2 Member Data Documentation . . . 74
12.8.2.1 ITbtForcePowerOnTimeoutInMs . . . 74
12.9 _ITBT_ROOTPORT_CONFIG Struct Reference . . . 74
12.9.1 Detailed Description . . . 74
12.10 _LIST_ENTRY Struct Reference . . . 75
12.10.1 Detailed Description . . . 75
12.11 _PEI_ITBT_CONFIG Struct Reference . . . 75
12.11.1 Detailed Description . . . 76
12.12 _PEI_PREMEM_SI_DEFAULT_POLICY_INIT_PPI Struct Reference . . . 76
12.12.1 Detailed Description . . . 76
12.13 _PEI_SI_DEFAULT_POLICY_INIT_PPI Struct Reference . . . 77
12.13.1 Detailed Description . . . 77
12.14 _SI_POLICY_STRUCT Struct Reference . . . 77
12.14.1 Detailed Description . . . 78
12.15 _SI_PREMEM_POLICY_STRUCT Struct Reference . . . 79
12.15.1 Detailed Description . . . 79
12.16 ADR_CONFIG Struct Reference . . . 80
12.16.1 Detailed Description . . . 81
12.17 ADR_GLOBAL_RESET_ENABLE Union Reference . . . 81
12.17.1 Detailed Description . . . 81
12.18 AMT_DXE_CONFIG Struct Reference . . . 82
12.18.1 Detailed Description . . . 83
12.19 AMT_PEI_CONFIG Struct Reference . . . 83
12.19.1 Detailed Description . . . 84
12.19.2 Member Data Documentation . . . 84
12.19.2.1 AmtEnabled . . . 84
12.19.2.2 WatchDogEnabled . . . 85
12.19.2.3 WatchDogTimerBios . . . 85
12.19.2.4 WatchDogTimerOs . . . 85
12.20 BCLK_CONFIG Struct Reference . . . 86
12.20.1 Detailed Description . . . 86
12.20.2 Member Data Documentation . . . 87
12.20.2.1 CpuBclkPllOn . . . 87
12.20.2.2 SocBclkPllOn . . . 87
12.21 BIOS_GUARD_CONFIG Struct Reference . . . 87
12.21.1 Detailed Description . . . 88
12.22 CNVI_CONFIG Struct Reference . . . 89

12.22.1 Detailed Description . . . 90
12.22.2 Member Data Documentation . . . 90
12.22.2.1 BtAudioOffload . . . 90
12.22.2.2 BtInterface . . . 91
12.22.2.3 Mode . . . 91
12.23 CNVI_PIN_MUX Struct Reference . . . 91
12.23.1 Detailed Description . . . 92
12.24 CNVI_PREMEM_CONFIG Struct Reference . . . 92
12.24.1 Detailed Description . . . 93
12.25 CPU_INIT_PREMEM_CONFIG Struct Reference . . . 93
12.25.1 Detailed Description . . . 95
12.25.2 Member Data Documentation . . . 95
12.25.2.1 ActiveCoreCount . . . 95
12.25.2.2 ActiveLpAtomCoreCount . . . 95
12.25.2.3 ActiveSmallCoreCount . . . 96
12.25.2.4 BootMaxFrequency . . . 96
12.25.2.5 CpuRatio . . . 96
12.25.2.6 CrashLogEnable . . . 96
12.25.2.7 CrashLogGprs . . . 97
12.25.2.8 DfdEnable . . . 97
12.25.2.9 FClkFrequency . . . 97
12.25.2.10 OcLock . . . 98
12.25.2.11 PeciSxReset . . . 98
12.25.2.12 PrmrrSize . . . 98
12.25.2.13 ReduceXecores . . . 99
12.25.2.14 SmmRelocationEnable . . . 99
12.25.2.15 TmeBypassCapability . . . 99
12.25.2.16 TmeEnable . . . 100
12.25.2.17 TsegSize . . . 100
12.25.2.18 VmxEnable . . . 100
12.26 CPU_POWER_DELIVERY_CONFIG Struct Reference . . . 101
12.26.1 Detailed Description . . . 103
12.26.2 Member Data Documentation . . . 104
12.26.2.1 DualTauBoost . . . 104
12.26.2.2 PowerLimit1 . . . 104
12.26.2.3 PowerLimit1Time . . . 104
12.26.2.4 PowerLimit2Power . . . 105
12.26.2.5 PowerLimit3 . . . 105
12.26.2.6 PowerLimit4 . . . 105
12.26.2.7 PowerLimit4Boost . . . 105
12.26.2.8 SkinTempControl . . . 106
12.26.2.9 TccActivationOffset . . . 106

12.26.2.10 ThETAlbattEnable . . . 106
12.27 CPU_POWER_MGMT_CUSTOM_CONFIG Struct Reference . . . 107
12.27.1 Detailed Description . . . 107
12.28 CPU_SECURITY_PREMEM_CONFIG Struct Reference . . . 108
12.28.1 Detailed Description . . . 109
12.28.2 Member Data Documentation . . . 109
12.28.2.1 BiosGuard . . . 109
12.28.2.2 GenerateNewTmeKey . . . 110
12.28.2.3 TdxEnable . . . 110
12.28.2.4 TseEnable . . . 110
12.28.2.5 Txt . . . 111
12.29 CPU_TEST_CONFIG Struct Reference . . . 111
12.29.1 Detailed Description . . . 112
12.29.2 Member Data Documentation . . . 112
12.29.2.1 ProcessorTraceMemSize . . . 113
12.30 DDI_CONFIGURATION Struct Reference . . . 113
12.30.1 Detailed Description . . . 114
12.31 DMI_EQ_PARAM Struct Reference . . . 114
12.31.1 Detailed Description . . . 115
12.32 DMI_HW_WIDTH_CONTROL Struct Reference . . . 115
12.32.1 Detailed Description . . . 115
12.33 DMI_LANE_CONFIG Struct Reference . . . 116
12.33.1 Detailed Description . . . 116
12.34 EFI_HOB_CPU Struct Reference . . . 116
12.34.1 Detailed Description . . . 117
12.34.2 Member Data Documentation . . . 117
12.34.2.1 Header . . . 117
12.35 EFI_HOB_FIRMWARE_VOLUME Struct Reference . . . 117
12.35.1 Detailed Description . . . 118
12.35.2 Member Data Documentation . . . 118
12.35.2.1 Header . . . 118
12.36 EFI_HOB_FIRMWARE_VOLUME2 Struct Reference . . . 118
12.36.1 Detailed Description . . . 119
12.36.2 Member Data Documentation . . . 119
12.36.2.1 Header . . . 119
12.37 EFI_HOB_FIRMWARE_VOLUME3 Struct Reference . . . 120
12.37.1 Detailed Description . . . 120
12.37.2 Member Data Documentation . . . 120
12.37.2.1 ExtractedFv . . . 121
12.37.2.2 FileName . . . 121
12.37.2.3 FvName . . . 121
12.37.2.4 Header . . . 121

12.38 EFI_HOB_GENERIC_HEADER Struct Reference . . . 122
12.38.1 Detailed Description . . . 122
12.39 EFI_HOB_GUID_TYPE Struct Reference . . . 122
12.39.1 Detailed Description . . . 123
12.39.2 Member Data Documentation . . . 123
12.39.2.1 Header . . . 123
12.40 EFI_HOB_HANDOFF_INFO_TABLE Struct Reference . . . 123
12.40.1 Detailed Description . . . 124
12.40.2 Member Data Documentation . . . 124
12.40.2.1 EfiMemoryTop . . . 124
12.40.2.2 Header . . . 124
12.40.2.3 Version . . . 125
12.41 EFI_HOB_MEMORY_ALLOCATION Struct Reference . . . 125
12.41.1 Detailed Description . . . 125
12.41.2 Member Data Documentation . . . 126
12.41.2.1 Header . . . 126
12.42 EFI_HOB_MEMORY_ALLOCATION_BSP_STORE Struct Reference . . . 126
12.42.1 Detailed Description . . . 127
12.42.2 Member Data Documentation . . . 127
12.42.2.1 Header . . . 127
12.43 EFI_HOB_MEMORY_ALLOCATION_HEADER Struct Reference . . . 127
12.43.1 Detailed Description . . . 128
12.43.2 Member Data Documentation . . . 128
12.43.2.1 MemoryBaseAddress . . . 128
12.43.2.2 MemoryType . . . 128
12.43.2.3 Name . . . 129
12.44 EFI_HOB_MEMORY_ALLOCATION_MODULE Struct Reference . . . 129
12.44.1 Detailed Description . . . 130
12.44.2 Member Data Documentation . . . 130
12.44.2.1 Header . . . 130
12.45 EFI_HOB_MEMORY_ALLOCATION_STACK Struct Reference . . . 130
12.45.1 Detailed Description . . . 131
12.45.2 Member Data Documentation . . . 131
12.45.2.1 Header . . . 131
12.46 EFI_HOB_MEMORY_POOL Struct Reference . . . 131
12.46.1 Detailed Description . . . 132
12.46.2 Member Data Documentation . . . 132
12.46.2.1 Header . . . 132
12.47 EFI_HOB_RESOURCE_DESCRIPTOR Struct Reference . . . 132
12.47.1 Detailed Description . . . 133
12.47.2 Member Data Documentation . . . 133
12.47.2.1 Header . . . 133

12.47.2.2 Owner . . . 133
12.48 EFI_HOB_UEFI_CAPSULE Struct Reference . . . 134
12.48.1 Detailed Description . . . 134
12.48.2 Member Data Documentation . . . 134
12.48.2.1 BaseAddress . . . 135
12.49 EFI_IP_ADDRESS Union Reference . . . 135
12.49.1 Detailed Description . . . 135
12.50 EFI_MAC_ADDRESS Struct Reference . . . 136
12.50.1 Detailed Description . . . 136
12.51 EFI_MMRAM_DESCRIPTOR Struct Reference . . . 136
12.51.1 Detailed Description . . . 136
12.51.2 Member Data Documentation . . . 136
12.51.2.1 CpuStart . . . 137
12.51.2.2 PhysicalStart . . . 137
12.51.2.3 RegionState . . . 137
12.52 EFI_PEI_HOB_POINTERS Union Reference . . . 138
12.52.1 Detailed Description . . . 138
12.53 EFI_TIME Struct Reference . . . 138
12.53.1 Detailed Description . . . 139
12.54 FIVR_EXT_RAIL_CONFIG Struct Reference . . . 139
12.54.1 Detailed Description . . . 139
12.54.2 Member Data Documentation . . . 139
12.54.2.1 SupportedVoltageStates . . . 140
12.54.2.2 Voltage . . . 140
12.55 FIVR_VCCIN_AUX_CONFIG Struct Reference . . . 140
12.55.1 Detailed Description . . . 140
12.55.2 Member Data Documentation . . . 141
12.55.2.1 LowToHighCurModeVolTranTime . . . 141
12.55.2.2 OffToHighCurModeVolTranTime . . . 141
12.55.2.3 RetToHighCurModeVolTranTime . . . 141
12.55.2.4 RetToLowCurModeVolTranTime . . . 142
12.56 FSP_ERROR_INFO Struct Reference . . . 142
12.56.1 Detailed Description . . . 142
12.56.2 Member Data Documentation . . . 143
12.56.2.1 ErrorType . . . 143
12.57 FSPM_ARCH_CONFIG_PPI Struct Reference . . . 143
12.57.1 Detailed Description . . . 143
12.58 FSPV_POSTMEM_CONFIG Struct Reference . . . 144
12.58.1 Detailed Description . . . 144
12.59 FSPV_PREMEM_CONFIG Struct Reference . . . 145
12.59.1 Detailed Description . . . 145
12.60 FUSA_CONFIG Struct Reference . . . 146

12.60.1 Detailed Description . . . 148
12.60.2 Member Data Documentation . . . 148
12.60.2.1 DisplayFusaConfigEnable . . . 149
12.61 GBE_CONFIG Struct Reference . . . 149
12.61.1 Detailed Description . . . 150
12.61.2 Member Data Documentation . . . 150
12.61.2.1 Enable . . . 150
12.62 GPIO_DXE_CONFIG Struct Reference . . . 151
12.62.1 Detailed Description . . . 151
12.62.2 Member Data Documentation . . . 152
12.62.2.1 HideGpioAcpiDevice . . . 152
12.62.2.2 HideHardGpioAcpiDevice . . . 152
12.62.2.3 HideVirtGpioAcpiDevice . . . 152
12.63 GUID Struct Reference . . . 152
12.63.1 Detailed Description . . . 153
12.64 HDA_LINK_DMIC Struct Reference . . . 153
12.64.1 Detailed Description . . . 153
12.65 HDA_LINK_HDA Struct Reference . . . 153
12.65.1 Detailed Description . . . 154
12.66 HDA_LINK_SNDW Struct Reference . . . 154
12.66.1 Detailed Description . . . 154
12.67 HDA_LINK_SSP Struct Reference . . . 154
12.67.1 Detailed Description . . . 155
12.68 HDA_MIC_PRIVACY Struct Reference . . . 155
12.68.1 Detailed Description . . . 156
12.69 HDA_SNDW_MULTILANE Struct Reference . . . 156
12.69.1 Detailed Description . . . 156
12.70 HDA_VERB_TABLE_HEADER Struct Reference . . . 156
12.70.1 Detailed Description . . . 157
12.71 HDAUDIO_CONFIG Struct Reference . . . 157
12.71.1 Detailed Description . . . 158
12.71.2 Member Data Documentation . . . 158
12.71.2.1 VerbTableEntryNum . . . 158
12.71.2.2 VerbTablePtr . . . 159
12.72 HDAUDIO_DXE_CONFIG Struct Reference . . . 159
12.72.1 Detailed Description . . . 160
12.73 HDAUDIO_PREMEM_CONFIG Struct Reference . . . 160
12.73.1 Detailed Description . . . 162
12.73.2 Member Data Documentation . . . 162
12.73.2.1 AudioLinkDmic . . . 162
12.73.2.2 AudioLinkHda . . . 162
12.73.2.3 AudioLinkSndw . . . 163

12.73.2.4 AudioLinkSsp . . . 163
12.73.2.5 SndwMultilane . . . 163
12.74 HOST_BRIDGE_PEI_CONFIG Struct Reference . . . 164
12.74.1 Detailed Description . . . 164
12.74.2 Member Data Documentation . . . 165
12.74.2.1 SkipPamLock . . . 165
12.75 HOST_BRIDGE_PREMEM_CONFIG Struct Reference . . . 165
12.75.1 Detailed Description . . . 167
12.76 HSIO_CONFIG Struct Reference . . . 167
12.76.1 Detailed Description . . . 168
12.77 HSIO_PARAMETERS Struct Reference . . . 168
12.77.1 Detailed Description . . . 170
12.77.2 Member Data Documentation . . . 170
12.77.2.1 HsioCtrlAdaptOffsetCfg . . . 170
12.78 HYBRID_GRAPHICS_CONFIG Struct Reference . . . 170
12.78.1 Detailed Description . . . 171
12.79 I2C_PIN_MUX Struct Reference . . . 171
12.79.1 Detailed Description . . . 171
12.80 I3C_MUX_GPIO_CONFIG Struct Reference . . . 172
12.80.1 Detailed Description . . . 172
12.81 IAX_PEI_CONFIG Struct Reference . . . 172
12.81.1 Detailed Description . . . 173
12.81.2 Member Data Documentation . . . 173
12.81.2.1 IaxEnable . . . 173
12.82 IEH_CONFIG Struct Reference . . . 174
12.82.1 Detailed Description . . . 174
12.83 IGPU_DXE_CONFIG Struct Reference . . . 175
12.83.1 Detailed Description . . . 176
12.84 IGPU_PEI_CONFIG Struct Reference . . . 176
12.84.1 Detailed Description . . . 177
12.85 IGPU_PEI_PREMEM_CONFIG Struct Reference . . . 177
12.85.1 Detailed Description . . . 179
12.85.2 Member Data Documentation . . . 179
12.85.2.1 IgdDvmt50PreAlloc . . . 179
12.85.2.2 InternalGraphics . . . 180
12.85.2.3 VgaInitControl . . . 180
12.86 IOM_AUX_ORI_GPIO_PAD_CONFIG Struct Reference . . . 180
12.86.1 Detailed Description . . . 181
12.87 IPU_PREMEM_CONFIG Struct Reference . . . 181
12.87.1 Detailed Description . . . 182
12.87.2 Member Data Documentation . . . 182
12.87.2.1 ImguClkOutEn . . . 182

12.87.2.2 IpuEnable . . . 182
12.88 IPv4_ADDRESS Struct Reference . . . 182
12.88.1 Detailed Description . . . 183
12.89 IPv6_ADDRESS Struct Reference . . . 183
12.89.1 Detailed Description . . . 183
12.90 ISH_CONFIG Struct Reference . . . 183
12.90.1 Detailed Description . . . 184
12.91 ISH_GP Struct Reference . . . 184
12.91.1 Detailed Description . . . 185
12.92 ISH_GPIO_CONFIG Struct Reference . . . 185
12.92.1 Detailed Description . . . 185
12.92.2 Member Data Documentation . . . 185
12.92.2.1 PadTermination . . . 185
12.92.2.2 PinMux . . . 186
12.93 ISH_I2C Struct Reference . . . 186
12.93.1 Detailed Description . . . 187
12.94 ISH_I2C_PIN_CONFIG Struct Reference . . . 187
12.94.1 Detailed Description . . . 187
12.95 ISH_PREMEM_CONFIG Struct Reference . . . 188
12.95.1 Detailed Description . . . 188
12.95.2 Member Data Documentation . . . 188
12.95.2.1 Enable . . . 189
12.96 ISH_SPI Struct Reference . . . 189
12.96.1 Detailed Description . . . 190
12.97 ISH_SPI_PIN_CONFIG Struct Reference . . . 190
12.97.1 Detailed Description . . . 191
12.98 ISH_UART Struct Reference . . . 191
12.98.1 Detailed Description . . . 192
12.99 ISH_UART_PIN_CONFIG Struct Reference . . . 192
12.99.1 Detailed Description . . . 193
12.100 LPSS_I2C_CONFIG Struct Reference . . . 193
12.100.1 Detailed Description . . . 193
12.101 LPSS_I2C_CONTROLLER_CONFIG Struct Reference . . . 194
12.101.1 Detailed Description . . . 194
12.101.2 Member Data Documentation . . . 194
12.101.2.1 PadTermination . . . 194
12.102 LPSS_I2C_PIN_MUX Struct Reference . . . 195
12.102.1 Detailed Description . . . 195
12.103 LPSS_I3C_CONFIG Struct Reference . . . 195
12.103.1 Detailed Description . . . 196
12.104 LPSS_I3C_DEVICE_CONFIG Struct Reference . . . 196
12.104.1 Detailed Description . . . 197

12.105 LPSS_SPI_CONFIG Struct Reference . . . 197
12.105.1 Detailed Description . . . 197
12.106 LPSS_SPI_DEVICE_CONFIG Struct Reference . . . 198
12.106.1 Detailed Description . . . 198
12.107 LPSS_SPI_PIN_MUX Struct Reference . . . 199
12.107.1 Detailed Description . . . 199
12.108 LPSS_UART_ATTRIBUTES Struct Reference . . . 199
12.108.1 Detailed Description . . . 200
12.109 LPSS_UART_CONFIG Struct Reference . . . 200
12.109.1 Detailed Description . . . 200
12.110 LPSS_UART_DEVICE_CONFIG Struct Reference . . . 201
12.110.1 Detailed Description . . . 201
12.111 LPSS_UART_PIN_MUX Struct Reference . . . 202
12.111.1 Detailed Description . . . 202
12.112 ME_DXE_CONFIG Struct Reference . . . 202
12.112.1 Detailed Description . . . 203
12.113 ME_PEI_CONFIG Struct Reference . . . 204
12.113.1 Detailed Description . . . 205
12.113.2 Member Data Documentation . . . 205
12.113.2.1 CseDataResilience . . . 205
12.113.2.2 MeUnconfigOnRtcClear . . . 205
12.114 ME_PEI_PREMEM_CONFIG Struct Reference . . . 206
12.114.1 Detailed Description . . . 207
12.114.2 Member Data Documentation . . . 207
12.114.2.1 SkipMbpHob . . . 207
12.115 MEMORY_CONFIG_NO_CRC Struct Reference . . . 208
12.115.1 Detailed Description . . . 210
12.115.2 Member Data Documentation . . . 210
12.115.2.1 ChannelToCkdQckMapping . . . 210
12.115.2.2 CkdAddressTable . . . 210
12.115.2.3 PhyClockToCkdDimm . . . 211
12.115.2.4 SerialDebugLevel . . . 211
12.115.2.5 SpdAddressTable . . . 212
12.116 MEMORY_CONFIGURATION Struct Reference . . . 212
12.116.1 Detailed Description . . . 213
12.117 MEMORY_DQDQS_MAPPING Struct Reference . . . 213
12.117.1 Detailed Description . . . 213
12.118 MEMORY_RCOMP Struct Reference . . . 213
12.118.1 Detailed Description . . . 214
12.119 MEMORY_SUB_SYSTEM_CONFIG Struct Reference . . . 214
12.119.1 Detailed Description . . . 215
12.120 MRC_ERROR_KEY_VALUE_ENTRY Struct Reference . . . 215

12.120.1 Detailed Description . . . 216
12.121 MUX_GPIO_CONFIG Struct Reference . . . 216
12.121.1 Detailed Description . . . 216
12.122 MUX_GPIO_PARAM Struct Reference . . . 216
12.122.1 Detailed Description . . . 217
12.122.2 Member Data Documentation . . . 217
12.122.2.1 PadTermination . . . 217
12.122.2.2 PinMux . . . 217
12.123 NPU_PEI_CONFIG Struct Reference . . . 218
12.123.1 Detailed Description . . . 218
12.123.2 Member Data Documentation . . . 219
12.123.2.1 NpuEnable . . . 219
12.124 OVERCLOCKING_PREMEM_CONFIG Struct Reference . . . 219
12.124.1 Detailed Description . . . 223
12.124.2 Member Data Documentation . . . 223
12.124.2.1 AdaptiveVoltage . . . 223
12.124.2.2 AtomClusterRatio . . . 223
12.124.2.3 Avx2VoltageScaleFactor . . . 224
12.124.2.4 ComputeDieSscEnable . . . 224
12.124.2.5 CoreMinRatio . . . 224
12.124.2.6 CoreVfConfigScope . . . 224
12.124.2.7 CpuBandgapRefMode . . . 225
12.124.2.8 CpuD2dRatio . . . 225
12.124.2.9 DisablePerCoreMask . . . 225
12.124.2.10 EcoreTvbTempThreshold0 . . . 225
12.124.2.11 EcoreTvbTempThreshold1 . . . 226
12.124.2.12 GranularityBins . . . 226
12.124.2.13 IaIccMax . . . 226
12.124.2.14 MaxOcRatio . . . 226
12.124.2.15 NegativeItd . . . 227
12.124.2.16 OcSupport . . . 227
12.124.2.17 OcTvb . . . 227
12.124.2.18 PcorePowerDensityThrottle . . . 227
12.124.2.19 PcoreTvbTempThreshold0 . . . 228
12.124.2.20 PcoreTvbTempThreshold1 . . . 228
12.124.2.21 PerCoreVoltageOffset . . . 228
12.124.2.22 PerEcoreCcpRatioDownBinAboveT0 . . . 228
12.124.2.23 PerEcoreCcpRatioDownBinAboveT1 . . . 229
12.124.2.24 PerPcoreGrRatioDownBinAboveT0 . . . 229
12.124.2.25 PerPcoreGrRatioDownBinAboveT1 . . . 229
12.124.2.26 PerPcoreRatioDownBinAboveT0 . . . 229
12.124.2.27 PerPcoreRatioDownBinAboveT1 . . . 230

12.124.2.28 ProcessVmaxLimit . . . 230
12.124.2.29 PvdMode . . . 230
12.124.2.30 PvdRatioThreshold . . . 230
12.124.2.31 RingDownBin . . . 231
12.124.2.32 SocDieSscEnable . . . 231
12.124.2.33 ThermalTurbo . . . 231
12.124.2.34 TjMaxOffset . . . 231
12.124.2.35 TvbConfigLimitSelect . . . 232
12.124.2.36 UnderVoltProtection . . . 232
12.124.2.37 VcciaBootVoltageSel . . . 232
12.124.2.38 VccsaBootVoltageSel . . . 232
12.124.2.39 VfPointOffset . . . 233
12.124.2.40 VfPointOffsetMode . . . 233
12.124.2.41 VoltageMode . . . 233
12.124.2.42 VoltageOffset . . . 233
12.124.2.43 VoltageOverride . . . 234
12.125 PCH_DCI_PREMEM_CONFIG Struct Reference . . . 234
12.125.1 Detailed Description . . . 235
12.125.2 Member Data Documentation . . . 235
12.125.2.1 DciClkEnable . . . 235
12.125.2.2 DciDbcMode . . . 235
12.125.2.3 DciEn . . . 236
12.125.2.4 KeepEarlyTrace . . . 236
12.126 PCH_DEVICE_INTERRUPT_CONFIG Struct Reference . . . 236
12.126.1 Detailed Description . . . 237
12.127 PCH_DMI_CONFIG Struct Reference . . . 237
12.127.1 Detailed Description . . . 238
12.127.2 Member Data Documentation . . . 238
12.127.2.1 DmiPowerReduction . . . 238
12.128 PCH_ESPI_CONFIG Struct Reference . . . 239
12.128.1 Detailed Description . . . 240
12.128.2 Member Data Documentation . . . 240
12.128.2.1 BmeHostDeviceEnabled . . . 240
12.128.2.2 LgmrEnable . . . 240
12.129 PCH_FIVR_CONFIG Struct Reference . . . 241
12.129.1 Detailed Description . . . 241
12.129.2 Member Data Documentation . . . 241
12.129.2.1 ExtVnnRailSx . . . 242
12.130 PCH_FLASH_PROTECTION_CONFIG Struct Reference . . . 242
12.130.1 Detailed Description . . . 243
12.131 PCH_GENERAL_CONFIG Struct Reference . . . 243
12.131.1 Detailed Description . . . 244

12.131.2 Member Data Documentation . . . 244
12.131.2.1 Crid . . . 244
12.131.2.2 LegacyIoLowLatency . . . 244
12.131.2.3 VtdEnabled . . . 245
12.132 PCH_GENERAL_PREMEM_CONFIG Struct Reference . . . 245
12.132.1 Detailed Description . . . 246
12.132.2 Member Data Documentation . . . 246
12.132.2.1 DisableResets . . . 246
12.132.2.2 GpioOverride . . . 247
12.132.2.3 IoeDebugEn . . . 247
12.132.2.4 PmcPrivacyConsent . . . 247
12.132.2.5 PmodeClkEn . . . 247
12.133 PCH_HSIO_PCIE_LANE_CONFIG Struct Reference . . . 248
12.133.1 Detailed Description . . . 248
12.134 PCH_HSIO_PCIE_PREMEM_CONFIG Struct Reference . . . 249
12.134.1 Detailed Description . . . 249
12.135 PCH_HSIO_SATA_PORT_LANE Struct Reference . . . 249
12.135.1 Detailed Description . . . 250
12.136 PCH_HSIO_SATA_PREMEM_CONFIG Struct Reference . . . 251
12.136.1 Detailed Description . . . 251
12.137 PCH_INTERRUPT_CONFIG Struct Reference . . . 252
12.137.1 Detailed Description . . . 253
12.138 PCH_IOAPIC_CONFIG Struct Reference . . . 253
12.138.1 Detailed Description . . . 254
12.138.2 Member Data Documentation . . . 254
12.138.2.1 Enable8254ClockGating . . . 254
12.138.2.2 Enable8254ClockGatingOnS3 . . . 255
12.139 PCH_LOCK_DOWN_CONFIG Struct Reference . . . 255
12.139.1 Detailed Description . . . 256
12.139.2 Member Data Documentation . . . 256
12.139.2.1 BiosInterface . . . 256
12.139.2.2 BiosLock . . . 257
12.139.2.3 GlobalSmi . . . 257
12.139.2.4 UnlockGpioPads . . . 257
12.140 PCH_LPC_PREMEM_CONFIG Struct Reference . . . 258
12.140.1 Detailed Description . . . 258
12.140.2 Member Data Documentation . . . 259
12.140.2.1 EnhancePort8xhDecoding . . . 259
12.140.2.2 LpcPmHAE . . . 259
12.141 PCH_P2SB_CONFIG Struct Reference . . . 260
12.141.1 Detailed Description . . . 260
12.141.2 Member Data Documentation . . . 260

12.141.2.1 SbAccessUnlock . . . 261
12.142 PCH_PCIE_CLOCK Struct Reference . . . 261
12.142.1 Detailed Description . . . 261
12.143 PCH_PCIE_CONFIG Struct Reference . . . 262
12.143.1 Detailed Description . . . 262
12.143.2 Member Data Documentation . . . 263
12.143.2.1 PcieClockGating . . . 263
12.143.2.2 PcieTbtPerfBoost . . . 263
12.144 PCH_PCIE_ROOT_PORT_CONFIG Struct Reference . . . 263
12.144.1 Detailed Description . . . 264
12.144.2 Member Data Documentation . . . 264
12.144.2.1 ExtSync . . . 264
12.144.2.2 MvcEnabled . . . 264
12.144.2.3 VppPort . . . 264
12.145 PCH_PCIE_RP_PREMEM_CONFIG Struct Reference . . . 265
12.145.1 Detailed Description . . . 265
12.145.2 Member Data Documentation . . . 266
12.145.2.1 RpEnabledMask . . . 266
12.146 PCH_PM_CONFIG Struct Reference . . . 266
12.146.1 Detailed Description . . . 268
12.146.2 Member Data Documentation . . . 268
12.146.2.1 C10DynamicThresholdAdjustment . . . 268
12.146.2.2 CppmFaEn . . . 268
12.146.2.3 CpuC10GatePinEnable . . . 269
12.146.2.4 DisableDsxAcPresentPulldown . . . 269
12.146.2.5 DisableEnergyReport . . . 269
12.146.2.6 DisableNativePowerButton . . . 269
12.146.2.7 GlobalResetMasksOverride . . . 270
12.146.2.8 LatchEventsC10Exit . . . 270
12.146.2.9 LpmS0ixSubStateEnable . . . 270
12.146.2.10 ModPhySusPgEnable . . . 270
12.146.2.11 OsIdleEnable . . . 271
12.146.2.12 PchPwrCycDur . . . 271
12.146.2.13 PciePllSsc . . . 271
12.146.2.14 PmcWdtTimerEn . . . 271
12.146.2.15 PsOnEnable . . . 272
12.146.2.16 PwrBtnOverridePeriod . . . 272
12.146.2.17 S0ixAutoDemotion . . . 272
12.146.2.18 SlpLanLowDc . . . 272
12.146.2.19 SlpStrchSusUp . . . 273
12.146.2.20 ThermTimerDelay . . . 273
12.146.2.21 Usb2PhySusPgEnable . . . 273

12.147 PCH_SATA_PORT_CONFIG Struct Reference . . . 273
12.147.1 Detailed Description . . . 274
12.147.2 Member Data Documentation . . . 274
12.147.2.1 Enable . . . 274
12.147.2.2 ZpOdd . . . 275
12.148 PCH_SMBUS_PREMEM_CONFIG Struct Reference . . . 275
12.148.1 Detailed Description . . . 276
12.148.2 Member Data Documentation . . . 276
12.148.2.1 Enable . . . 276
12.148.2.2 Header . . . 277
12.148.2.3 SpdWriteDisable . . . 277
12.149 PCH_TRACE_HUB_PREMEM_CONFIG Struct Reference . . . 277
12.149.1 Detailed Description . . . 278
12.150 PCH_WAKE_CONFIG Struct Reference . . . 278
12.150.1 Detailed Description . . . 278
12.150.2 Member Data Documentation . . . 279
12.150.2.1 PmeB0S5Dis . . . 279
12.151 PCH_WDT_PREMEM_CONFIG Struct Reference . . . 279
12.151.1 Detailed Description . . . 280
12.152 PCIE_COMMON_CONFIG Struct Reference . . . 280
12.152.1 Detailed Description . . . 280
12.152.2 Member Data Documentation . . . 281
12.152.2.1 ComplianceTestMode . . . 281
12.152.2.2 EnablePort8xhDecode . . . 281
12.152.2.3 RpFunctionSwap . . . 281
12.153 PCIE_DEVICE_OVERRIDE Struct Reference . . . 282
12.153.1 Detailed Description . . . 282
12.153.2 Member Data Documentation . . . 283
12.153.2.1 ForceLtrOverride . . . 283
12.153.2.2 L1sCommonModeRestoreTime . . . 283
12.153.2.3 L1sTpowerOnScale . . . 283
12.153.2.4 L1sTpowerOnValue . . . 284
12.153.2.5 L1SubstatesCapMask . . . 284
12.153.2.6 L1SubstatesCapOffset . . . 284
12.153.2.7 NonSnoopLatency . . . 284
12.153.2.8 SnoopLatency . . . 285
12.154 PCIE_EQ_PARAM Struct Reference . . . 285
12.154.1 Detailed Description . . . 285
12.155 PCIE_LINK_EQ_PLATFORM_SETTINGS Struct Reference . . . 285
12.155.1 Detailed Description . . . 286
12.155.2 Member Data Documentation . . . 287
12.155.2.1 EqPh23Bypass . . . 287

12.155.2.2 EqPh3Bypass . . . 287
12.155.2.3 LocalTxOverrideEn . . . 287
12.155.2.4 Ph2LocalTxOverridePreset . . . 288
12.155.2.5 Ph3NoOfPresetOrCoeff . . . 288
12.156 PCIE_RP_DXE_CONFIG Struct Reference . . . 288
12.156.1 Detailed Description . . . 289
12.156.2 Member Data Documentation . . . 289
12.156.2.1 PcieDeviceOverrideTablePtr . . . 289
12.157 PEI_DMI_CONFIG Struct Reference . . . 290
12.157.1 Detailed Description . . . 291
12.157.2 Member Data Documentation . . . 291
12.157.2.1 DmiGen3EqPh2Enable . . . 292
12.157.2.2 DmiGen3EqPh3Method . . . 292
12.157.2.3 DmiGen3ProgramStaticEq . . . 292
12.157.2.4 DmiGen3RxCtlePeaking . . . 293
12.157.2.5 DmiMaxLinkSpeed . . . 293
12.158 PMC_GLOBAL_RESET_MASK Union Reference . . . 293
12.158.1 Detailed Description . . . 293
12.159 PMC_INTERFACE_CONFIG Struct Reference . . . 294
12.159.1 Detailed Description . . . 294
12.160 PMC_LPM_S0IX_SUB_STATE_EN Union Reference . . . 294
12.160.1 Detailed Description . . . 294
12.160.2 Member Data Documentation . . . 294
12.160.2.1 S0i2p2En . . . 295
12.160.2.2 S0i3p3En . . . 295
12.160.2.3 S0i3p4En . . . 295
12.161 PPM_CUSTOM_RATIO_TABLE Struct Reference . . . 295
12.161.1 Detailed Description . . . 296
12.161.2 Member Data Documentation . . . 296
12.161.2.1 StateRatioMax16 . . . 296
12.162 PROTECTED_RANGE Struct Reference . . . 296
12.162.1 Detailed Description . . . 297
12.163 PSF_CONFIG Struct Reference . . . 297
12.163.1 Detailed Description . . . 298
12.163.2 Member Data Documentation . . . 298
12.163.2.1 DwbConfigDwbFlushThreshold . . . 298
12.164 RTC_CONFIG Struct Reference . . . 299
12.164.1 Detailed Description . . . 299
12.164.2 Member Data Documentation . . . 299
12.164.2.1 BiosInterfaceLock . . . 300
12.164.2.2 MemoryLock . . . 300
12.165 SA_ADDRESS_DECODE Struct Reference . . . 300

12.165.1 Detailed Description . . . 301
12.166 SATA_CONFIG Struct Reference . . . 301
12.166.1 Detailed Description . . . 302
12.166.2 Member Data Documentation . . . 302
12.166.2.1 Enable . . . 302
12.166.2.2 EsataSpeedLimit . . . 303
12.166.2.3 RaidDeviceId . . . 303
12.166.2.4 SataMode . . . 303
12.166.2.5 SpeedLimit . . . 303
12.166.2.6 ThermalThrottling . . . 304
12.167 SATA_THERMAL_THROTTLING Struct Reference . . . 304
12.167.1 Detailed Description . . . 305
12.168 SCS_SD_DLL Struct Reference . . . 305
12.168.1 Detailed Description . . . 305
12.169 SERIAL_IO_CONFIG Struct Reference . . . 305
12.169.1 Detailed Description . . . 306
12.170 SERIAL_IO_I2C_CONFIG Struct Reference . . . 306
12.170.1 Detailed Description . . . 307
12.170.2 Member Data Documentation . . . 307
12.170.2.1 PadTermination . . . 307
12.171 SERIAL_IO_I3C_CONFIG Struct Reference . . . 307
12.171.1 Detailed Description . . . 308
12.172 SERIAL_IO_SPI_CONFIG Struct Reference . . . 308
12.172.1 Detailed Description . . . 309
12.173 SERIAL_IO_UART_ATTRIBUTES Struct Reference . . . 309
12.173.1 Detailed Description . . . 309
12.174 SERIAL_IO_UART_CONFIG Struct Reference . . . 310
12.174.1 Detailed Description . . . 310
12.175 SI_CONFIG Struct Reference . . . 311
12.175.1 Detailed Description . . . 312
12.175.2 Member Data Documentation . . . 312
12.175.2.1 CustomizedSsid . . . 312
12.175.2.2 CustomizedSvid . . . 312
12.175.2.3 NumberOfSsidTableEntry . . . 313
12.175.2.4 SkipBiosDoneWhenFwUpdate . . . 313
12.175.2.5 SkipSsidProgramming . . . 313
12.175.2.6 SsidTablePtr . . . 313
12.176 SI_PREMEM_CONFIG Struct Reference . . . 314
12.176.1 Detailed Description . . . 314
12.176.2 Member Data Documentation . . . 315
12.176.2.1 PlatformDebugOption . . . 315
12.176.2.2 SkipOverrideBootModeWhenFwUpdate . . . 315

12.177 SPD_DATA_BUFFER Struct Reference . . . 315
12.177.1 Detailed Description . . . 316
12.178 SPD_OFFSET_TABLE Struct Reference . . . 316
12.178.1 Detailed Description . . . 316
12.179 SPI_CONFIG Struct Reference . . . 317
12.179.1 Detailed Description . . . 317
12.179.2 Member Data Documentation . . . 318
12.179.2.1 ExtendedBiosDecodeRangeBase . . . 318
12.179.2.2 ExtendedBiosDecodeRangeLimit . . . 318
12.180 SPI_PIN_MUX Struct Reference . . . 318
12.180.1 Detailed Description . . . 319
12.181 SVID_SID_VALUE Struct Reference . . . 319
12.181.1 Detailed Description . . . 319
12.182 SWIZZLE_CONFIG Struct Reference . . . 319
12.182.1 Detailed Description . . . 320
12.183 TCSS_IOM_PEI_CONFIG Struct Reference . . . 321
12.183.1 Detailed Description . . . 321
12.184 TCSS_IOM_PEI_PERMEM_CONFIG Struct Reference . . . 322
12.184.1 Detailed Description . . . 322
12.185 TCSS_MISC_PEI_CONFIG Struct Reference . . . 322
12.185.1 Detailed Description . . . 323
12.186 TCSS_PCIE_PEI_POLICY Struct Reference . . . 323
12.186.1 Detailed Description . . . 324
12.186.2 Member Data Documentation . . . 324
12.186.2.1 TcssTbtPerfBoost . . . 324
12.187 TCSS_PCIE_PORT_POLICY Struct Reference . . . 324
12.187.1 Detailed Description . . . 325
12.188 TCSS_PEI_CONFIG Struct Reference . . . 326
12.188.1 Detailed Description . . . 326
12.189 TCSS_PEI_PREMEM_CONFIG Struct Reference . . . 327
12.189.1 Detailed Description . . . 327
12.190 TCSS_USBTC_PEI_PERMEM_CONFIG Union Reference . . . 328
12.190.1 Detailed Description . . . 328
12.191 TELEMETRY_PEI_CONFIG Struct Reference . . . 328
12.191.1 Detailed Description . . . 329
12.192 TELEMETRY_PEI_PREMEM_CONFIG Struct Reference . . . 329
12.192.1 Detailed Description . . . 330
12.193 THC_CONFIG Struct Reference . . . 330
12.193.1 Detailed Description . . . 331
12.194 THC_DATA_SSID_CONFIG_HOB Struct Reference . . . 332
12.194.1 Detailed Description . . . 332
12.195 THC_HID_OVER_I2C Struct Reference . . . 333

12.195.1 Detailed Description . . . 334
12.195.2 Member Data Documentation . . . 334
12.195.2.1 AddressingMode . . . 334
12.196 THC_HID_OVER_SPI Struct Reference . . . 334
12.196.1 Detailed Description . . . 335
12.197 THC_PORT Struct Reference . . . 335
12.197.1 Detailed Description . . . 336
12.198 THC_PORT_SUBSYTEM_ID Struct Reference . . . 336
12.198.1 Detailed Description . . . 336
12.199 THC_RESET Struct Reference . . . 337
12.199.1 Detailed Description . . . 337
12.200 THC_WOT_PIN_CONFIG Struct Reference . . . 337
12.200.1 Detailed Description . . . 337
12.201 THERMAL_CONFIG Struct Reference . . . 338
12.201.1 Detailed Description . . . 338
12.201.2 Member Data Documentation . . . 338
12.201.2.1 DmiHaAWC . . . 339
12.201.2.2 PchHotLevel . . . 339
12.201.2.3 TTLevels . . . 339
12.202 THERMAL_THROTTLE_LEVELS Struct Reference . . . 339
12.202.1 Detailed Description . . . 340
12.202.2 Member Data Documentation . . . 340
12.202.2.1 TTLock . . . 340
12.202.2.2 TTState13Enable . . . 341
12.203 TRACE_HUB_CONFIG Struct Reference . . . 341
12.203.1 Detailed Description . . . 341
12.203.2 Member Data Documentation . . . 341
12.203.2.1 AetEnabled . . . 342
12.203.2.2 EnableMode . . . 342
12.203.2.3 MemReg0Size . . . 342
12.204 TRACE_HUB_PREMEM_CONFIG Struct Reference . . . 342
12.204.1 Detailed Description . . . 343
12.205 TSN_CONFIG Struct Reference . . . 343
12.205.1 Detailed Description . . . 344
12.206 TXT_PREMEM_CONFIG Struct Reference . . . 344
12.206.1 Detailed Description . . . 345
12.207 UART_PIN_MUX Struct Reference . . . 346
12.207.1 Detailed Description . . . 346
12.208 USB2_PHY_CONFIG Struct Reference . . . 346
12.208.1 Detailed Description . . . 347
12.208.2 Member Data Documentation . . . 347
12.208.2.1 Port . . . 347

12.209 USB2_PHY_PARAMETERS Struct Reference . . . 347
12.209.1 Detailed Description . . . 348
12.210 USB2_PORT_CONFIG Struct Reference . . . 348
12.210.1 Detailed Description . . . 349
12.210.2 Member Data Documentation . . . 349
12.210.2.1 OverCurrentPin . . . 349
12.211 USB3_HSIO_CONFIG Struct Reference . . . 349
12.211.1 Detailed Description . . . 350
12.212 USB3_PORT_CONFIG Struct Reference . . . 350
12.212.1 Detailed Description . . . 350
12.212.2 Member Data Documentation . . . 350
12.212.2.1 OverCurrentPin . . . 351
12.213 USB_CONFIG Struct Reference . . . 351
12.213.1 Detailed Description . . . 352
12.213.2 Member Data Documentation . . . 352
12.213.2.1 LtrOverrideEnable . . . 353
12.213.2.2 OverCurrentEnable . . . 353
12.213.2.3 PdoProgramming . . . 353
12.213.2.4 UaolEnable . . . 353
12.213.2.5 Usb31Speed . . . 354
12.213.2.6 XhciOcLock . . . 354
12.214 VMD_PEI_CONFIG Struct Reference . . . 354
12.214.1 Detailed Description . . . 356
12.215 XDCI_CONFIG Struct Reference . . . 356
12.215.1 Detailed Description . . . 356
12.215.2 Member Data Documentation . . . 356
12.215.2.1 Enable . . . 356

13 File Documentation **357**

13.1 AdrConfig.h File Reference . . . 357
13.1.1 Detailed Description . . . 357
13.2 AmtConfig.h File Reference . . . 358
13.2.1 Detailed Description . . . 359
13.3 Base.h File Reference . . . 359
13.3.1 Detailed Description . . . 364
13.3.2 Macro Definition Documentation . . . 364
13.3.2.1 _BASE_INT_SIZE_OF . . . 364
13.3.2.2 _INT_SIZE_OF . . . 364
13.3.2.3 ABS . . . 365
13.3.2.4 ADDRESS_IS_ALIGNED . . . 365
13.3.2.5 ALIGN_POINTER . . . 366
13.3.2.6 ALIGN_VALUE . . . 366

13.3.2.7 ALIGN_VALUE_ADDEND . . . 367
13.3.2.8 ALIGN_VARIABLE . . . 367
13.3.2.9 ALIGNOF . . . 368
13.3.2.10 ANALYZER_NORETURN . . . 368
13.3.2.11 ANALYZER_UNREACHABLE . . . 368
13.3.2.12 ARRAY_SIZE . . . 368
13.3.2.13 BASE_ARG . . . 369
13.3.2.14 BASE_CR . . . 369
13.3.2.15 ENCODE_ERROR . . . 370
13.3.2.16 ENCODE_WARNING . . . 370
13.3.2.17 FALSE . . . 371
13.3.2.18 IS_ALIGNED . . . 371
13.3.2.19 IS_POW2 . . . 372
13.3.2.20 MAX . . . 372
13.3.2.21 MIN . . . 372
13.3.2.22 NORETURN . . . 374
13.3.2.23 OFFSET_OF . . . 374
13.3.2.24 RETURN_ADDRESS . . . 375
13.3.2.25 RETURN_BUFFER_TOO_SMALL . . . 375
13.3.2.26 RETURN_ERROR . . . 375
13.3.2.27 RETURNS_TWICE . . . 376
13.3.2.28 SIGNATURE_16 . . . 376
13.3.2.29 SIGNATURE_32 . . . 376
13.3.2.30 SIGNATURE_64 . . . 377
13.3.2.31 TRUE . . . 378
13.3.2.32 UNREACHABLE . . . 378
13.3.2.33 VA_ARG . . . 378
13.3.2.34 VA_COPY . . . 379
13.3.2.35 VA_END . . . 379
13.3.2.36 VA_START . . . 379
13.3.3 Typedef Documentation . . . 380
13.3.3.1 BASE_LIST . . . 380
13.3.3.2 VA_LIST . . . 380
13.3.4 Function Documentation . . . 380
13.3.4.1 STATIC_ASSERT() . . . 380
13.4 BiosGuardConfig.h File Reference . . . 381
13.4.1 Detailed Description . . . 382
13.4.2 Typedef Documentation . . . 382
13.4.2.1 EC_CMD_LOCK . . . 382
13.4.2.2 EC_PROVISION_EAV . . . 383
13.5 CnviConfig.h File Reference . . . 383
13.5.1 Detailed Description . . . 384

13.6 ConfigBlock.h File Reference . . . 384
13.6.1 Detailed Description . . . 385
13.7 ConfigBlockLib.h File Reference . . . 386
13.7.1 Detailed Description . . . 387
13.7.2 Function Documentation . . . 387
13.7.2.1 AddConfigBlock() . . . 387
13.7.2.2 CreateConfigBlockTable() . . . 387
13.7.2.3 GetConfigBlock() . . . 388
13.8 CpuInitConfig.h File Reference . . . 388
13.8.1 Detailed Description . . . 389
13.8.2 Macro Definition Documentation . . . 389
13.8.2.1 CPU_INIT_CONFIG_REVISION . . . 389
13.9 CpuInitPreMemConfig.h File Reference . . . 389
13.9.1 Detailed Description . . . 390
13.10 CpuPowerDeliveryConfig.h File Reference . . . 390
13.10.1 Detailed Description . . . 391
13.10.2 Macro Definition Documentation . . . 391
13.10.2.1 MAX_CUSTOM_CTDP_ENTRIES . . . 391
13.11 CpuPowerMgmtBasicConfig.h File Reference . . . 391
13.11.1 Detailed Description . . . 392
13.11.2 Macro Definition Documentation . . . 392
13.11.2.1 CPU_POWER_MGMT_BASIC_CONFIG_REVISION . . . 392
13.12 CpuPowerMgmtCustomConfig.h File Reference . . . 392
13.12.1 Detailed Description . . . 393
13.13 CpuPowerMgmtTestConfig.h File Reference . . . 393
13.13.1 Detailed Description . . . 394
13.13.2 Macro Definition Documentation . . . 394
13.13.2.1 CPU_POWER_MGMT_TEST_CONFIG_REVISION . . . 394
13.13.3 Enumeration Type Documentation . . . 394
13.13.3.1 CUSTOM_POWER_UNIT . . . 394
13.14 CpuPowerMgmtVrConfig.h File Reference . . . 395
13.14.1 Detailed Description . . . 395
13.14.2 Macro Definition Documentation . . . 396
13.14.2.1 CPU_POWER_MGMT_VR_CONFIG_REVISION . . . 396
13.14.2.2 MAX_NUM_VRS . . . 396
13.15 CpuSecurityPreMemConfig.h File Reference . . . 396
13.15.1 Detailed Description . . . 397
13.16 CpuTestConfig.h File Reference . . . 397
13.16.1 Detailed Description . . . 397
13.17 DciConfig.h File Reference . . . 398
13.17.1 Detailed Description . . . 398
13.18 DmiConfig.h File Reference . . . 399

13.18.1 Detailed Description . . . 399
13.19 DxeITbtConfig.h File Reference . . . 399
13.19.1 Detailed Description . . . 400
13.19.2 Typedef Documentation . . . 401
13.19.2.1 DXE_ITBT_CONFIG . . . 401
13.20 EspiConfig.h File Reference . . . 401
13.20.1 Detailed Description . . . 401
13.21 FiaConfig.h File Reference . . . 402
13.21.1 Detailed Description . . . 402
13.22 FivrConfig.h File Reference . . . 402
13.22.1 Detailed Description . . . 403
13.23 FlashProtectionConfig.h File Reference . . . 403
13.23.1 Detailed Description . . . 404
13.24 FspErrorInfo.h File Reference . . . 404
13.24.1 Detailed Description . . . 405
13.25 FspFixedPcds.h File Reference . . . 405
13.25.1 Detailed Description . . . 405
13.26 FspmArchConfigPpi.h File Reference . . . 406
13.26.1 Detailed Description . . . 406
13.27 FspPerformance.h File Reference . . . 406
13.27.1 Detailed Description . . . 406
13.28 FspVConfig.h File Reference . . . 407
13.28.1 Detailed Description . . . 407
13.29 FusaConfig.h File Reference . . . 408
13.29.1 Detailed Description . . . 408
13.29.2 Macro Definition Documentation . . . 408
13.29.2.1 FUSA_CONFIG_REVISION . . . 409
13.30 GbeConfig.h File Reference . . . 409
13.30.1 Detailed Description . . . 409
13.31 GpioDevConfig.h File Reference . . . 409
13.31.1 Detailed Description . . . 410
13.32 HdAudioConfig.h File Reference . . . 410
13.32.1 Detailed Description . . . 411
13.33 HostBridgeConfig.h File Reference . . . 412
13.33.1 Detailed Description . . . 412
13.34 HsioConfig.h File Reference . . . 413
13.34.1 Detailed Description . . . 413
13.35 HsioPcieConfig.h File Reference . . . 413
13.35.1 Detailed Description . . . 414
13.36 HsioSataConfig.h File Reference . . . 414
13.36.1 Detailed Description . . . 414
13.37 HybridGraphicsConfig.h File Reference . . . 415

13.37.1 Detailed Description . . . 415
13.38 IaxConfig.h File Reference . . . 415
13.38.1 Detailed Description . . . 416
13.39 IehConfig.h File Reference . . . 416
13.39.1 Detailed Description . . . 417
13.40 IGpuConfig.h File Reference . . . 417
13.40.1 Detailed Description . . . 418
13.41 InterruptConfig.h File Reference . . . 418
13.41.1 Detailed Description . . . 419
13.41.2 Enumeration Type Documentation . . . 419
13.41.2.1 PCH_INT_PIN . . . 419
13.42 IoApicConfig.h File Reference . . . 420
13.42.1 Detailed Description . . . 420
13.43 IpuPreMemConfig.h File Reference . . . 420
13.43.1 Detailed Description . . . 421
13.44 IshConfig.h File Reference . . . 421
13.44.1 Detailed Description . . . 422
13.45 LockDownConfig.h File Reference . . . 422
13.45.1 Detailed Description . . . 423
13.46 LpcConfig.h File Reference . . . 423
13.46.1 Detailed Description . . . 423
13.47 LpssI2cConfig.h File Reference . . . 424
13.47.1 Detailed Description . . . 424
13.47.2 Enumeration Type Documentation . . . 425
13.47.2.1 LPSS_I2C_I2C_MODE . . . 425
13.48 LpssI3cConfig.h File Reference . . . 425
13.48.1 Detailed Description . . . 426
13.48.2 Enumeration Type Documentation . . . 426
13.48.2.1 I3C_MODE . . . 426
13.49 LpssSpiConfig.h File Reference . . . 427
13.49.1 Detailed Description . . . 427
13.49.2 Enumeration Type Documentation . . . 428
13.49.2.1 LPSS_SPI_MODE . . . 428
13.50 LpssUartConfig.h File Reference . . . 428
13.50.1 Detailed Description . . . 429
13.50.2 Enumeration Type Documentation . . . 429
13.50.2.1 LPSS_UART_MODE . . . 430
13.50.2.2 LPSS_UART_PG . . . 430
13.51 MeDxeConfig.h File Reference . . . 431
13.51.1 Detailed Description . . . 431
13.52 MemoryConfig.h File Reference . . . 432
13.52.1 Detailed Description . . . 433

13.52.2 Enumeration Type Documentation . . . 433
13.52.2.1 SA_SPD . . . 433
13.52.2.2 SPD_BOOT_MODE . . . 433
13.53 MemorySubSystemConfig.h File Reference . . . 434
13.53.1 Detailed Description . . . 434
13.54 MePeiConfig.h File Reference . . . 435
13.54.1 Detailed Description . . . 435
13.55 MpServices.h File Reference . . . 436
13.55.1 Detailed Description . . . 437
13.55.2 Typedef Documentation . . . 437
13.55.2.1 EFI_PEI_MP_SERVICES_ENABLEDISABLEAP . . . 437
13.55.2.2 EFI_PEI_MP_SERVICES_GET_NUMBER_OF_PROCESSORS . . . 438
13.55.2.3 EFI_PEI_MP_SERVICES_GET_PROCESSOR_INFO . . . 438
13.55.2.4 EFI_PEI_MP_SERVICES_STARTUP_ALL_APS . . . 439
13.55.2.5 EFI_PEI_MP_SERVICES_STARTUP_THIS_AP . . . 440
13.55.2.6 EFI_PEI_MP_SERVICES_SWITCH_BSP . . . 440
13.55.2.7 EFI_PEI_MP_SERVICES_WHOAMI . . . 441
13.56 NpuConfig.h File Reference . . . 441
13.56.1 Detailed Description . . . 442
13.57 OverclockingConfig.h File Reference . . . 443
13.57.1 Detailed Description . . . 443
13.58 P2sbConfig.h File Reference . . . 443
13.58.1 Detailed Description . . . 444
13.59 PchDmiConfig.h File Reference . . . 444
13.59.1 Detailed Description . . . 444
13.60 PchGeneralConfig.h File Reference . . . 445
13.60.1 Detailed Description . . . 445
13.60.2 Enumeration Type Documentation . . . 445
13.60.2.1 PCH_RESERVED_PAGE_ROUTE . . . 445
13.61 PchPcieRpConfig.h File Reference . . . 446
13.61.1 Detailed Description . . . 447
13.61.2 Enumeration Type Documentation . . . 447
13.61.2.1 PCH_PCIE_CLOCK_USAGE . . . 447
13.62 PcieConfig.h File Reference . . . 448
13.62.1 Detailed Description . . . 449
13.62.2 Enumeration Type Documentation . . . 449
13.62.2.1 PCIE_FORM_FACTOR . . . 449
13.62.2.2 PCIE_LINK_EQ_METHOD . . . 449
13.62.2.3 PCIE_LINK_EQ_MODE . . . 450
13.63 PeiDmiConfig.h File Reference . . . 450
13.63.1 Detailed Description . . . 451
13.64 PeiITbtConfig.h File Reference . . . 451

13.64.1 Detailed Description . . . 452
13.64.2 Typedef Documentation . . . 452
13.64.2.1 PEI_ITBT_CONFIG . . . 452
13.65 PeiITbtGenericStructure.h File Reference . . . 452
13.65.1 Detailed Description . . . 453
13.66 PeiPreMemSiDefaultPolicy.h File Reference . . . 453
13.66.1 Detailed Description . . . 454
13.67 PeiSiDefaultPolicy.h File Reference . . . 454
13.67.1 Detailed Description . . . 454
13.68 PiHob.h File Reference . . . 455
13.68.1 Detailed Description . . . 456
13.69 PiMultiPhase.h File Reference . . . 456
13.69.1 Detailed Description . . . 457
13.69.2 Macro Definition Documentation . . . 457
13.69.2.1 DXE_ERROR . . . 457
13.69.2.2 EFI_AUTH_STATUS_PLATFORM_OVERRIDE . . . 458
13.69.2.3 PI_ENCODE_ERROR . . . 458
13.69.2.4 PI_ENCODE_WARNING . . . 458
13.69.3 Typedef Documentation . . . 459
13.69.3.1 EFI_AP_PROCEDURE . . . 459
13.69.3.2 EFI_AP_PROCEDURE2 . . . 460
13.70 PmConfig.h File Reference . . . 460
13.70.1 Detailed Description . . . 461
13.70.2 Enumeration Type Documentation . . . 461
13.70.2.1 PCH_SLP_S4_MIN_ASSERT . . . 461
13.71 PsfConfig.h File Reference . . . 462
13.71.1 Detailed Description . . . 462
13.72 RtcConfig.h File Reference . . . 463
13.72.1 Detailed Description . . . 463
13.73 SataConfig.h File Reference . . . 464
13.73.1 Detailed Description . . . 464
13.74 ScsConfig.h File Reference . . . 465
13.74.1 Detailed Description . . . 465
13.75 SerialIoConfig.h File Reference . . . 466
13.75.1 Detailed Description . . . 466
13.76 SerialIoDevices.h File Reference . . . 467
13.76.1 Detailed Description . . . 468
13.76.2 Enumeration Type Documentation . . . 468
13.76.2.1 SERIAL_IO_I2C_MODE . . . 469
13.76.2.2 SERIAL_IO_I3C_MODE . . . 469
13.76.2.3 SERIAL_IO_SPI_MODE . . . 470
13.76.2.4 SERIAL_IO_UART_MODE . . . 470

13.76.2.5 SERIAL_IO_UART_PG . . . 471
13.77 SiConfig.h File Reference . . . 471
13.77.1 Detailed Description . . . 472
13.78 SiPolicyStruct.h File Reference . . . 473
13.78.1 Detailed Description . . . 473
13.78.2 Macro Definition Documentation . . . 474
13.78.2.1 SI_POLICY_REVISION . . . 474
13.78.2.2 SI_PREMEM_POLICY_REVISION . . . 474
13.79 SiPreMemConfig.h File Reference . . . 475
13.79.1 Detailed Description . . . 475
13.80 SmbusConfig.h File Reference . . . 476
13.80.1 Detailed Description . . . 476
13.81 SpiConfig.h File Reference . . . 477
13.81.1 Detailed Description . . . 477
13.82 SwizzleConfig.h File Reference . . . 478
13.82.1 Detailed Description . . . 478
13.83 TcssPeiConfig.h File Reference . . . 479
13.83.1 Detailed Description . . . 479
13.83.2 Function Documentation . . . 480
13.83.2.1 TcssConfigDump() . . . 480
13.84 TcssPeiPreMemConfig.h File Reference . . . 480
13.84.1 Detailed Description . . . 482
13.84.2 Enumeration Type Documentation . . . 482
13.84.2.1 TCSS_PORT_CAP . . . 482
13.84.3 Function Documentation . . . 482
13.84.3.1 TcssPreMemConfigDump() . . . 482
13.85 TelemetryPeiConfig.h File Reference . . . 484
13.85.1 Detailed Description . . . 484
13.86 ThcConfig.h File Reference . . . 485
13.86.1 Detailed Description . . . 486
13.86.2 Enumeration Type Documentation . . . 486
13.86.2.1 THC_MODE . . . 486
13.86.2.2 THC_PORT_ASSIGNMENT . . . 486
13.87 ThermalConfig.h File Reference . . . 487
13.87.1 Detailed Description . . . 487
13.88 TraceHubConfig.h File Reference . . . 487
13.88.1 Detailed Description . . . 488
13.89 TsnConfig.h File Reference . . . 488
13.89.1 Detailed Description . . . 489
13.89.2 Macro Definition Documentation . . . 489
13.89.2.1 TSN_CONFIG_REVISION . . . 489
13.90 TxtConfig.h File Reference . . . 490

13.90.1 Detailed Description 490
13.91 UefiBaseType.h File Reference 491
13.91.1 Detailed Description 493
13.91.2 Macro Definition Documentation 493
13.91.2.1 EFI_PAGES_TO_SIZE 493
13.91.2.2 EFI_SIZE_TO_PAGES 493
13.91.3 Typedef Documentation 494
13.91.3.1 EFI_IPv4_ADDRESS 494
13.91.3.2 EFI_IPv6_ADDRESS 494
13.92 Usb2PhyConfig.h File Reference 494
13.92.1 Detailed Description 495
13.93 Usb3HsioConfig.h File Reference 495
13.93.1 Detailed Description 496
13.94 UsbConfig.h File Reference 496
13.94.1 Detailed Description 497
13.95 VmdPeiConfig.h File Reference 497
13.95.1 Detailed Description 498
13.96 VtdConfig.h File Reference 498
13.96.1 Detailed Description 498
13.97 WatchDogConfig.h File Reference 499
13.97.1 Detailed Description 499

Index **501**

Chapter 1

Introduction

1.1 Purpose

The purpose of this document is to describe the steps required to integrate the Intel® Firmware Support Package (FSP) into a boot loader solution. It supports **PantherLake** platforms with **PantherLake** processor and **Panther↩ Lake** Platform Controller Hub (PCH).

1.2 Intended Audience

This document is targeted at all platform and system developers who need to consume FSP binaries in their boot loader solutions. This includes, but is not limited to: system BIOS developers, boot loader developers, system integrators, as well as end users.

1.3 Related Documents

- *Platform Initialization (PI) Specification v1.7* located at http://www.uefi.org/specifications
- *Intel® Firmware Support Package: External Architecture Specification (EAS) v2.1* located at https↩ ://cdrdv2.intel.com/v1/dl/getContent/611786
- *Boot Setting File Specification (BSF) v1.0* https://firmware.intel.com/sites/default/files/↩ BSF_1_0.pdf
- *Binary Configuration Tool for Intel® Firmware Support Package* available at http://www.intel.↩ com/fsp

1.4 Acronyms and Terminology

Acronym	Definition
BCT	Binary Configuration Tool
BDSM	Base Data Of Stolen Memory
BSF	Boot Setting File
BSP	Boot Strap Processor

Acronym	Definition
BWG	BIOS Writer's Guide
CAR	Cache As Ram
CRB	Customer Reference Board
DPR	DMA Protected Range
FIT	Firmware Interface Table
FSP	Firmware Support Package
FSP API	Firmware Support Package Interface
FW	Firmware
GTT	Graphics Translation Table
IED	Intel Enhanced Debug
IFWI	Integrated Firmware Image
IOT	Internal Observation Trace
MRC	Memory Reference Code (Memory Init code encapsulated by FSP-M)
MOT	Memory Observation Trace
PCH	Platform Controller Hub
PMC	Power Management Controller
PMRR	Protected Memory Range Reporting
REMAP	Remapped Memory Area
RVP	Reference and Validation Platform
SBSP	System BSP
SMI	System Management Interrupt
SMM	System Management Mode
SMRAM	System Management Mode RAM
SPI	Serial Peripheral Interface
TOLUD	Top of Low Usable Memory
TOUUD	Top of Upper Usable Memory
TSEG	Memory Reserved at the Top of Memory to be used as SMRAM
UPD	Updatable Product Data

Chapter 2

Overview

2.1 Technical Overview

The *Intel® Firmware Support Package (FSP)* provides chipset and processor initialization in a format that can easily be incorporated into many existing boot loaders.

The FSP will perform the necessary initialization steps as documented in the BWG including initialization of the CPU, memory controller, chipset and certain bus interfaces, if necessary.

FSP is not a stand-alone boot loader; therefore it needs to be integrated into a host boot loader to carry out other boot loader functions, such as: initializing non-Intel components, conducting bus enumeration, and discovering devices in the system and all industry standard initialization.

The FSP binary can be integrated easily into many different boot loaders, such as Coreboot, EDKII MinPlatform, Intel® Slim Bootloader, etc. and also into the embedded OS directly.

Below are some required steps for the integration:

- **Customizing** The static FSP configuration parameters are part of the FSP binary and can be customized by tools provided by Intel. This step is optional as configuration data may also be provided at runtime.
- **Rebasing** The FSP is not Position Independent Code (PIC) and the whole FSP has to be rebased if it is placed at a location which is different from the preferred address during build process.
- **Placing** Once the FSP binary is ready for integration, the boot loader build process needs to be modified to place this FSP binary at the specific rebasing location identified above.
- **Interfacing** The boot loader needs to add code to setup the operating environment for the FSP, call the FSP with correct parameters and parse the FSP output to retrieve the necessary information returned by the FSP.

2.2 FSP Distribution Package

- The FSP distribution package contains the following:
  - FSP Binary
  - FSP Integration Guide
  - BSF Configuration File
  - Data Structure Header Files
- The FSP configuration utility called BCT is available as a separate package. It can be downloaded from link mentioned in Section 1.3.

2.2.1 Package Layout

- **Docs (Auto generated)**
  - FSP_Integration_Guide.pdf
  - FSP_Integration_Guide.chm
- **Include**
  - FsptUpd.h, FspmUpd.h and FspsUpd.h (FSP UPD structure and related definitions)
  - GpioSampleDef.h (Sample enum definitions for GPIO table)
- FspBinPkg.dec (EDKII declaration file for package)
- Fsp.bsf (BSF file for configuring the data using BCT tool)
- Fsp.fd (FSP Binary)

Chapter 3

FSP Integration

3.1 Assumptions Used in this Document

The FSP for this platform is built with a preferred base address given by PcdFspAreaBaseAddress and so the reference code provided in the document assumes that the FSP is placed at this base address during the final boot loader build. Users may rebase the FSP binary at a different location with Intel's Binary Configuration Tool (BCT) or SplitFspBin.py before integrating to the boot loader.

For other assumptions and conventions, please refer to Chapter 9 and 10 of the FSP External Architecture Specification version 2.5.

3.2 Boot Flow

Please refer Chapter 7 in the FSP External Architecture Specification version 2.5 for Boot flow chart. The FSP for this SOC supports both Api and Dispatch mode, see Chapter 9 and 10 of the FSP External Architecture Specification version 2.5 for a detailed description of both Api and Dispatch mode.

3.2.1 FSP Early Boot Flow

High-level early FSP boot flow after the addition of FSP-O (containing Reset Vector code and SEC Core):

- After control is transferred to the reset vector, FSP early flow executes SEC Core to initialize TempRAM (FspSecEntry.nasm).
  - FSP consumes the BSP manifest to know the location of BSP and obtain FSP-T configuration parameters.
- FSP then calls BSP SEC core.
- In dispatch mode, BSP SEC Core can continue execution by reporting all FVs (including FSP-M - FSP needs to pass FSP-M location and size).
- In API mode, BSP SEC Core needs to call the FSP Memory Init API and continue.

3.3 FSP INFO Header

The FSP has an Information Header that provides critical information that is required by the bootloader to successfully interface with the FSP. The structure of the FSP Information Header is documented in the FSP External Architecture Specification version 2.5 with a HeaderRevision of 7.

3.4 FSP Image ID and Revision

FSP information header contains an Image ID field and an Image Revision field that provide the identification and revision information of the FSP binary. It is important to verify these fields while integrating the FSP as API parameters could change over different FSP IDs and revisions. All the FSP FV segments(FSP-T, FSP-M and FS↩ P-S) must have same FSP Image ID and revision number, using FV segments with different revision numbers in a single FSP image is not valid. The FSP API parameters documented in this integration guide are applicable for the Image ID and Revision specified as below.

The FSP ImageId string in the FSP information header is given by **PcdFspImageIdString** .The ImageRevision specifies the low-order bytes of the build number and revision while ExtendedImageRevision specifies the highorder bytes of the build number and revision, The FSP binary build number can be decoded as follows: "SiliconInitVersion Major|Minor|Revision|BuildNumber" (Ex: 0x020000241912) , where Build Number = (ExtendedImageRevision[7:0] << 8) | ImageRevision[7:0] Revision = (ExtendedImageRevision[15:8] << 8)| ImageRevision[15:8] Minor Version = ImageRevision[23:16] Major Version = ImageRevision[31:24] Please refer to section 5.1.1 Table 1 in the Intel(R) Firmware Support Package External Architecture Specification version 2.5.

3.5 FSP Global Data

FSP uses some amount of TempRam area to store FSP global data which contains some critical data like pointers to FSP information headers and UPD configuration regions, FSP/Bootloader stack pointers required for stack switching etc. HPET Timer register(2) PcdGlobalDataPointerAddress is reserved to store address of this global data, and hence boot loader should not use this register for any other purpose. If TempRAM initialization is done by boot loader, then HPET has to be initialized to the base so that access to the register will work fine.

3.6 Memory Map

Below diagram represents the memory map allocated by FSP including the FSP specific regions.

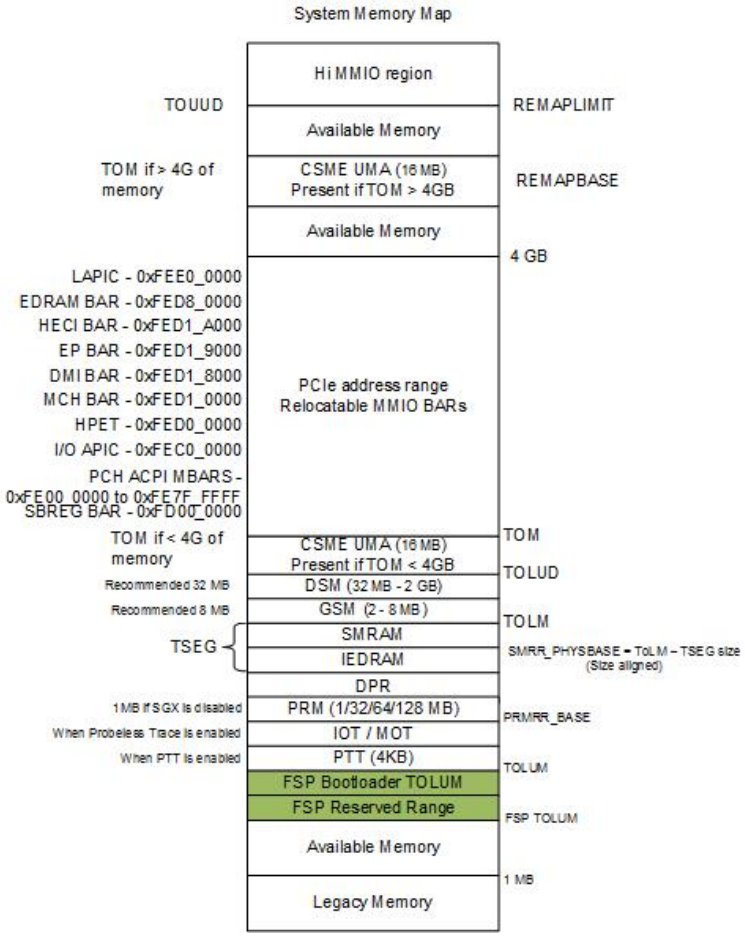


Figure 3.1 System Memory Map

3.7 Optional modules

For products using FSP in API mode, having SPI size constraints, the following modules can be safely removed to help save additional space. These modules are included in FSP binary primarily when FSP is used in dispatch mode:

- PtlPcdPchInitDxeFsp
- HeciInitDxeFsp
- SiInitDxeFsp
- CpuMpPei
- FspS3Notify
- FspEndOfPei2Peim
- GpioV2PtlPcdPpiInitPei
- PtlPcdPchInitFsp
- HeciInitApi

3.8 Reset vector in FSP

Reset vector code is now part of FSP-O. After verifying the IBB, Intel Root of Trust (RoT) such as BootGuard, transfers control to the FSP component (FSP-O) to execute the reset vector code and subsequently invokes other components from FSP and Bootloader. Once FSP-O gets control, it executes reset vector code followed by SEC core in the FSP scope. FSP-O then invokes FSP-T by retrieving FSP-T configuration from the BSP Manifest.

3.8.1 FSP Flash Layout

The below diagram represents the FSP flash layout:

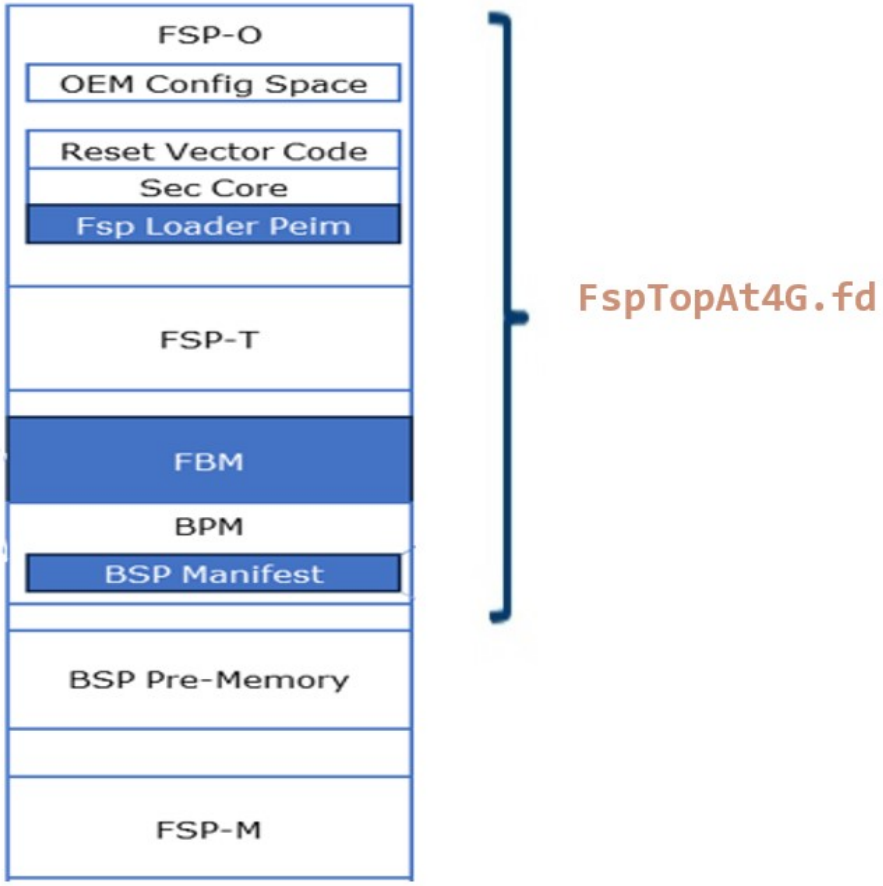


Figure 3.2 FSP Flash Layout

The FSP-O component is utilized in the FSP binary containing Reset Vector code and SEC Core.

- FSP-T is not optional anymore and executed as part of FSP early flow.
- FSP-O must be placed at a fixed location (4GB - FSP-O SIZE) in the Flash Map, and FSP-T right below FSP-O.
- FSP-T UPD as binary and report through BSP Manifest.
- FSP-O locates BPM and searches BSPM Element from BSSS field of BPM.
- If BSPM element is not found from BPM, it searches BSPM Element from OEM Config Space (4G-4K) as defined in previous generation.
- FSP-O retrieves FSP-T UPD data from BSPM Element

Note: Recommendation is to keep BSP Manifest inside the BPM.

3.8.2 FBM

The FBM is FSP Boot Manifest, FBM must be integrated by the OEMs and is provided by intel through BinPkg. OEMs must not modify any data within FBM while integrating and OEMs must build FIT table with the FIT entry (type 0x0D) to let FSP know the location of the FBM.

Note

The following requirements apply when using Signed FSP :

- Since FIT Pointer (4GB - 40h) is patched by OEMs, 4GB to (4GB - 4KB) region will not be included in the FBM (Firmware Boot Manifest). With the 4K region at the top of 4GB space continued to be supported for OEM use cases, the entire region of 4GB to (4GB-4KB) will be excluded from FSP Boot Manifest (FBM).
- This exclusion allows OEMs to continue the flexibility of patching the FIT pointer address in addition to having a dedicated region (4K) to use as data region for any special needs during early boot, mutually exclusive to Intel Chain of Trust via FBM.
- To maintain the integrity, OEMs must ensure that 4GB to (4GB - 4KB) region is reported to Intel RoT via BPM (Boot Policy Manifest). This is crucial when using the FSP signing feature to ensure that the ACM can verify the integrity of this region.
- When Booting with Signed FSP Binaries, If OEMs do not report 4GB to (4GB - 4KB) region in the BPM, the ACM will not allow the system to boot.
- **NEM Allocation and Pre-Memory IBB Size Limits:**
  - Boot Guard ACM performs NEM allocation for FSP-O/T, FSP-O then allocates NEM for FSP-M and BSP Pre-Memory components. BSP Pre-Memory information is reported through BSPM IBB segments for proper cache management.
  - All Pre-Memory components (FSP-O/T, FSP-M, and BSP Pre-Memory) should be placed contiguously in the top memory region near 4GB for efficient NEM allocation and optimal cache utilization.
  - **Platform-Specific Size Limits:**
    - **PantherLake (6MB LLC):**
      - FSP-M Compressed: Up to 3MB Pre-Memory
      - FSP-M Uncompressed: Up to 4.5MB Pre-Memory
    - **WildcatLake:**
      - 6MB LLC SKU: IBB size limit same as PantherLake FSP-M Uncompressed
      - 4MB LLC SKU: Up to 3.5MB Pre-Memory
  - Exceeding these platform-specific limits will result in compatibility issues due to insufficient cache capacity for Pre-Memory component execution.
  - ACM region placement must avoid any overlap with NEM mapped address space.

3.8.3 BSP Manifest

The BSPM Element (follows BSSS Structure Layout) is a data structure that represents BSP Manifest that contains information regarding BspEntryPoint, FSP-T configuration, FSP-M Base Address and other necessary elements. The structure definition is shown below.

```
typedef struct {
UINT8                       StructureID[8];
UINT8                       StructVersion;
UINT16                      SizeOfElement;
UINT8                       Reserved0;
//
// BSP Entrypoint for FSP to jump into
//
UINT32                      BspEntryPoint;
```

```
  //
  // FSP-T UPD
  //
  FSPT_UPD                    FsptUpd;
  //
  // 0: FSP loads FSP-M
  // 1: FSP skips loading FSP-M
  //
  UINT8                       FspmLoadingPolicy;
  UINT32                      FspmBaseAddress;
  UINT64                      TpmBaseAddress;
  //
  // FSP Signing Enabled:  BspSegmentCount and BspDigestData should describe BspPreMem segments and digest
  // FSP Signing Disabled: BspSegmentCount should be 0, BspDigestData should be NULL
  //
  UINT32                      BspSegmentCount;
  //
  // CMOS offset will be used to disable TXT if verification fails in BTG 0T and 3T case.
  //
  UINT8                       CmosOffset;
  //REGION_SEGMENT              BspSegmentArray[SegmentCount];
  //SHAX_HASH_STRUCTURE         BspDigestData;
} BSPM_ELEMENT;
```

OEMs can create a simple script that can generate the BSSS Structure that contains BSP Manifest information and map it to a location as mentioned in 3.8.1, which is reserved as part of FSP-O to allow OEMs to hold any data required for early initialization in the boot phase. While reporting Manifest in legacy way it is recommended to patch the manifest on a 32-byte boundary for performance reasons Refer the BsssGen tool inside PantherLakeBoard↩ Pkg\BoardSupport\Tools for more details.

3.8.3.1 FSP-T UPD Address Patching in BSPM Element

Care must be taken to make sure the right UPD Data offset is patched as there may be multiple FSP-T UPD structures in the binary. In the Intel reference implementation, the FSP-T component will contain default UPD values, but FvPreMemory contains the updated FSP-T UPD values. For example, MicrocodeRegionBase and Microcode↩ RegionSize have been updated during build. In this example, BsssGen.py searches for FSP-T UPD struct defined by "UPD_T" in FvPreMemory and patches this address in the BSP Manifest.

3.8.4 BIOS Build Flow

The below diagram represents the BIOS build flow:

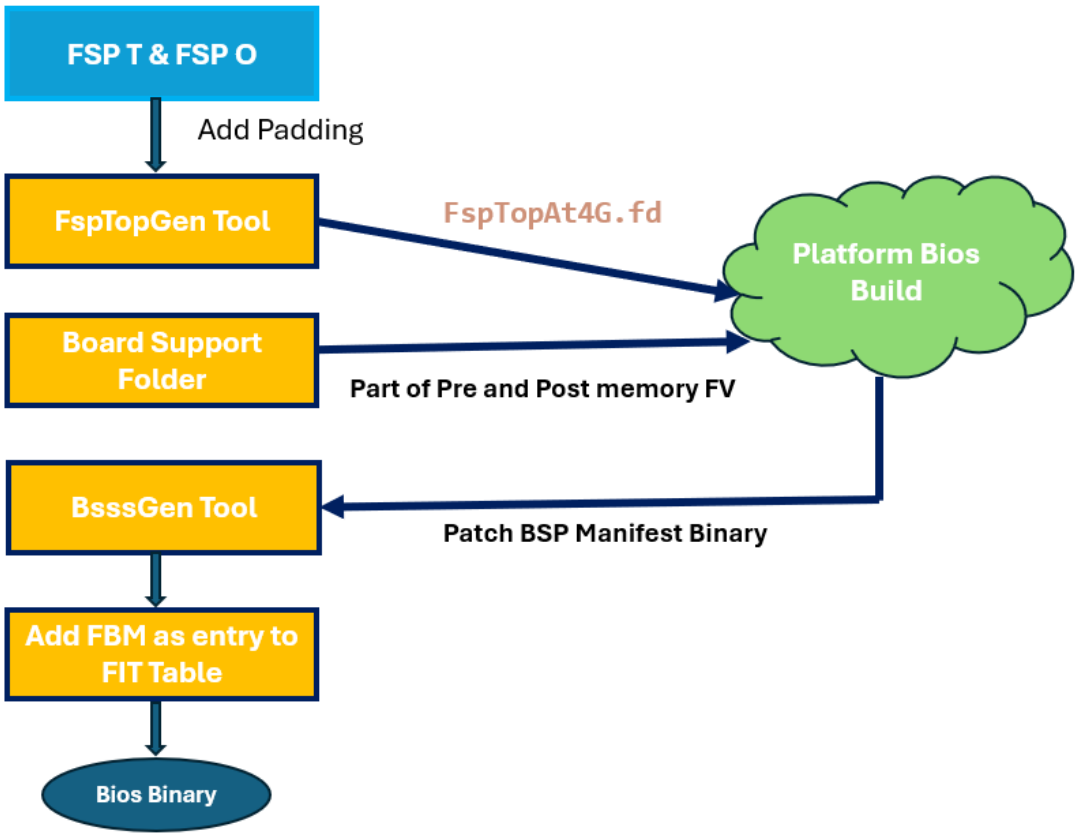


Figure 3.3 BIOS Build Flow

- FspTopGen.py:
  - The FspTopGen tool takes care of padding, FSP-T and FSP-0 to align it to 64 KB block size (as per platform requirement)
  - FSP-O component is placed at end of FSP-O FV
- PantherLakeBoardPkg\BoardSupport:
  - Folder contains drivers that are required to complete the FSP execution and new tools for FSP-O/T padding and manifest creation
- BsssGen.py:
  - Generates the Manifest binary that will be consumed by FSP-O
- FBM in FIT Table.
  - Using FitGen Tool add FBM as Entry to FIT table

3.9 FSP-M Compression

The PantherLake FSP binary introduces the ability to compress FSP-M resulting in significant size savings. In this generation, FSP-M is compressed, at build time, using LZMA compression algorithm. The decompression step takes place during boot time at a predefined address. Currently this feature is supported for FSP dispatch mode only.

3.9.1 FSP-M Compression/Decompression flow (Dispatch Mode)

The below diagram represents the FSP-M Compression/Decompression flow:

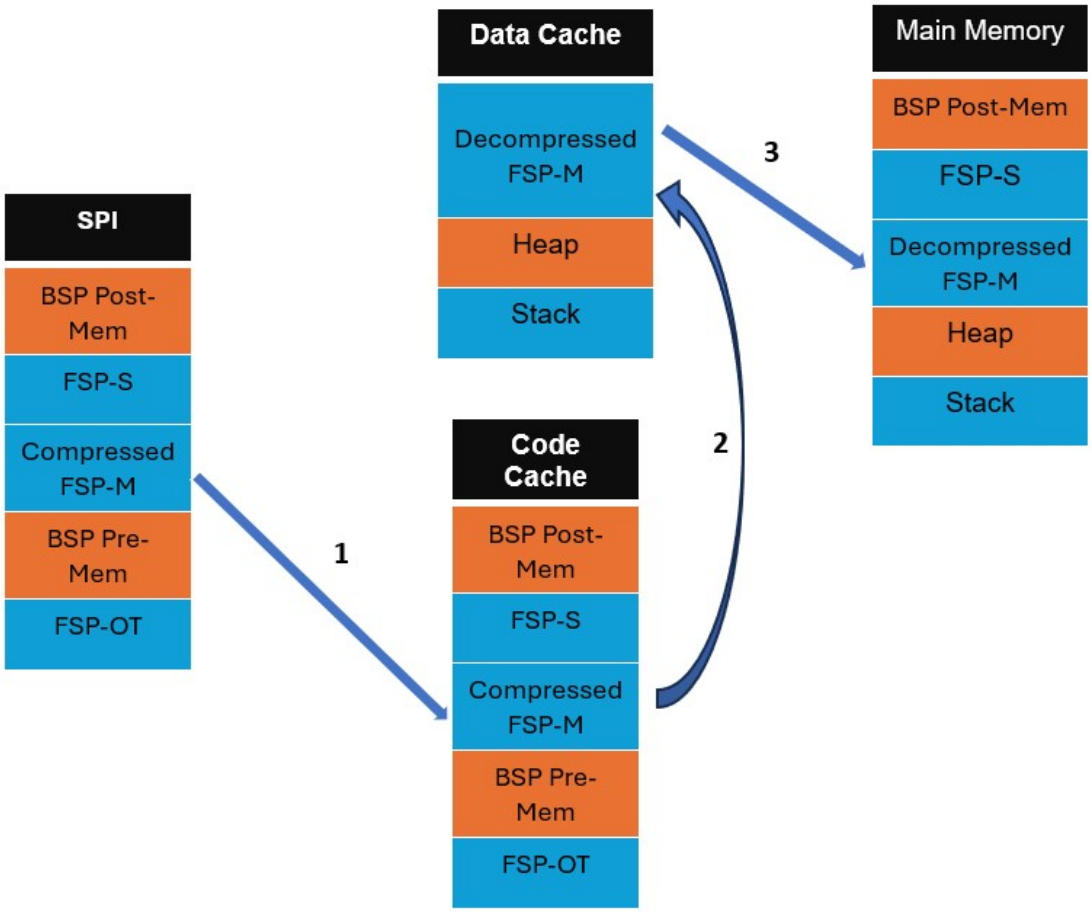


Figure 3.4 FSP-M Compression/Decompression flow

- FSP-M binary is integrated as FV in BoardPkg.fdf and is compressed using LZMA at build time.
- Build time rebase is not needed since PEI core is not used in FSP-M compression case
- The location of the compressed FSP-M component is passed to FSP-O via FspmBaseAddress in BSP Manifest (BSPM).
- The signed FSP flow, assumes all FSP components are uncompressed. As such, if FSP-M is compressed, it needs to be decompressed before it can be verified and measured by FSP-O.
  - FSP-O starts by decompressing FSP-M at a fixed address during boot time.
  - The Heap size needs of the Cache as RAM (CAR) need to be increased by 2MB (1 additional MTRR required) to hold the decompressed image.
  - The decompression region must be marked as write-back.
  - Finally, FSP-O verifies and measures (decompressed) FSP-M .
- The Runtime Rebase, introduced in signed FSP flow will now be performed to rebase FSP-M
- Once permanent memory is initialized, FSP-M FV and PPI pointers are migrated from CAR to memory.
  - Function pointers inside PPI Descriptors that are allocated in Heap will not be migrated.These PPI descriptors need to re-install separately.

  - The design requires to use platform supplied PEI core, because PEI core in FSP (i.e. BFV) is still compressed.
  - FSP-M will be migrated before Pei-Core re-entry. This is done to ensure that migration happens when Pei-Core is running from a cached location, helping to reduce migration time significantly.

3.9.2 API Mode Considerations

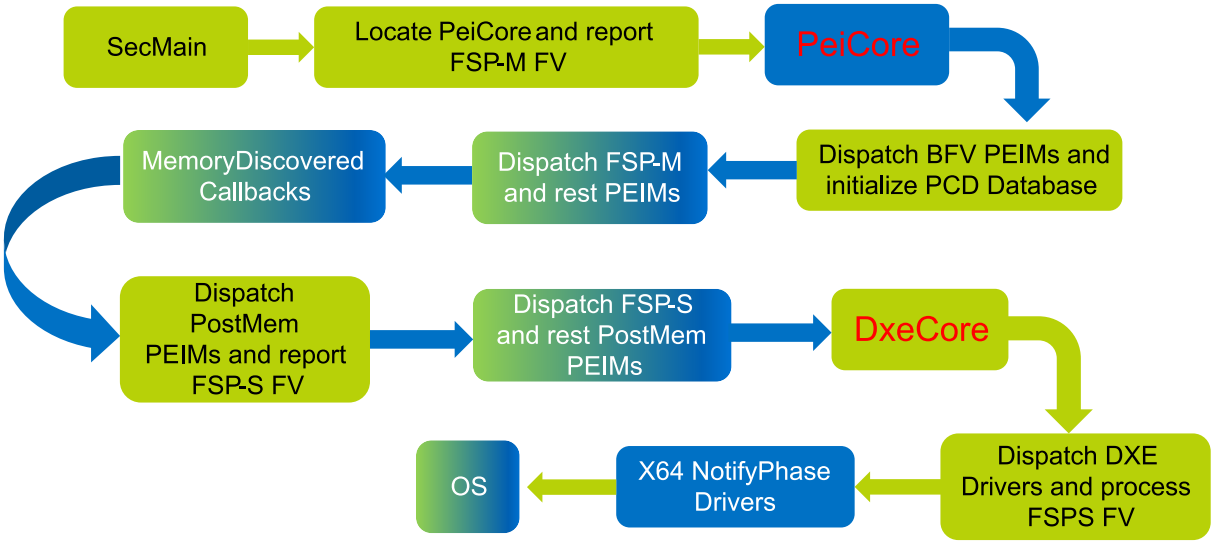
- Enabled early migration in dispatch mode, allowing migration of FSP-M to occur before PeiCore reentry.
  - PeiCore, which handles the migration of FSP-M, runs from BspPreMem
- In API mode, PeiCore operates directly from FSP-M, which means FSP-M migration can only happen after PeiCore reentry. This results in longer migration times in API mode because PeiCore runs from an uncached location.
- FSP-M is rebased at build time to the precise address where it will be decompressed, eliminating the need for a runtime rebase. This optimization reduces overhead and ensures faster execution.

Chapter 4

FSP Dispatch Mode

Overview

The FSP for this platform supports dispatch mode. Support for dispatch mode can be detected by checking if FSP_INFO_HEADER->ImageAttribute[BIT1] == 1. Dispatch mode is intended to enable FSP to integrate well in to UEFI bootloader implementations. Dispatch mode implements a boot flow that is as close to a standard UEFI boot flow as possible. In dispatch mode, the FSP is exposed as standard Firmware Volumes (FVs) directly to the bootloader. The PEIMs in these FVs are executed in the same PEI environment as the boot loader. In dispatch mode, the PPI database, PCD database, and HOB list are shared between the boot loader and the FSP.



Blue blocks are from the FSP binary and green blocks are from the bootloader. Blocks with mixed colors indicate that both bootloader and FSP modules are dispatched during that phase of the boot flow. In dispatch mode, the Notify↩ Phase() API is not used. Instead, FSP-S contains DXE drivers that implement the native callbacks on equivalent events for each of the NotifyPhase() invocations.

In dispatch mode, the PPI database and PCD database are used for providing policy data from the bootloader to FSP. Because these mechanisms provide a great deal of flexibility, dispatch mode does not constrain the method for passing policy data as strongly as API mode. The following sections describe the dispatch mode policy initialization flow used specifically for this platform.

4.1 Dispatch Mode Policy Init

For the plurality of platform designs, most of the policy options provided by FSP do not need to be modified by the bootloader.

To ease this common case, policy initialization has been broken in to two phases: 1. Policy Init and 2. Policy Update.

Policy Init creates the policy data structures with all default policy values pre-populated. Policy Init is implemented using the `factory method pattern`. The FSP provides an API to the bootloader for constructing the policy data structures. Because the factory method is provided by the FSP, backwards compatibility is made substantially easier. The FSP can be assured that the policy data structures match the definitions used at the time the FSP was compiled. As long as new fields are added to the bottom of policy structures, the bootloader may continue to use an older version of the policy data structures at compile time and retain compatibility with both new and old FSP binaries.

There are two factory methods, one for FSP-M and one for FSP-S. PeiPreMemPolicyInit() is provided by FSP-M, PeiPolicyInit() is provided by FSP-S. These methods are exposed to the bootloader using PPIs, PEI_PREMEM_SI_DEFAULT_POLICY_INIT_PPI for FSP-M and PEI_SI_DEFAULT_POLICY_INIT_PPI for FSP-S. The usage of PPIs ensures ABI stability between the FSP and the bootloader.

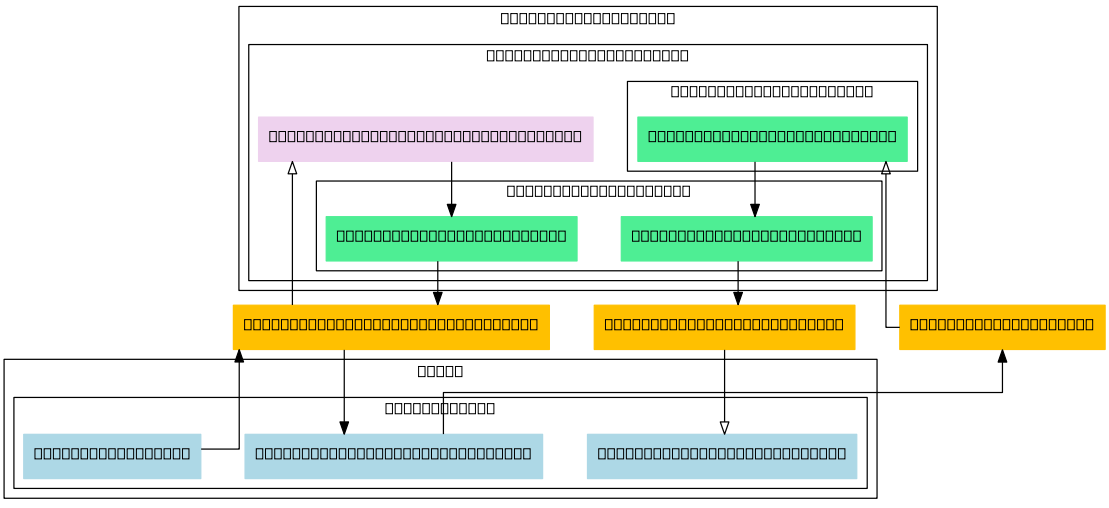
While Policy Init is implemented by the FSP, Policy Update is implemented by the bootloader. Policy Update uses a read-modify-write operation to apply any motherboard specific settings to the policy data while leaving the default values intact when appropriate. After Policy Update is done, FSP may run its SOC initialization code.

4.1.1 FSP-M Policy Initialization Flow

The follow graph shows the policy initialization flow for FSP-M.

Note

The hollow arrow heads in the flow diagram below indicate that action by the PEI Foundation is required for the flow to proceed.



Note

The function names and PEIMs used by the UEFI PI boot loader are implementation dependent and may vary from those shown here. The function names and PEIMs used by MinPlatform are provided as an example.

Execution of FSP-M begins after FspmWrapperInit() in IntelFsp2WrapperPkg/FspmWrapperPeim/FspmWrapperPeim.c installs a EFI_PEI_FIRMWARE_VOLUME_INFO_PPI instance for FSP-M. This causes the PEI foundation to become aware of the FV(s) in FSP-M, which schedules all PEIMs in FSP-M for dispatch by PEI. This results in the SiInitPreMem PEIM in FSP-M being executed.

One of the actions performed by the SiInitPreMem entry-point is calling the SiInitPrePolicy() function, which installs the PEI_PREMEM_SI_DEFAULT_POLICY_INIT_PPI. The DEPEX for MinPlatformPkg/PlatformInit/SiliconPolicyPei/SiliconPolicyPeiPreMem.inf requires PEI_PREMEM_SI_DEFAULT_POLICY_INIT_PPI to exist in the PPI database before SiliconPolicyPeiPreMem can run. While SiliconPolicyPeiPreMem.inf does not directly add PEI_PREMEM_SI_DEFAULT_POLICY_INIT_PPI to the DEPEX, SiliconPolicyPeiPreMem.inf does statically link against PantherLakeSiliconPkg/Library/PeiSiliconPolicyInitLib/PeiPreMemSiliconPolicyInitLib.inf, which adds PEI_PREMEM_SI_DEFAULT_POLICY_INIT_PPI to the DEPEX. After the SiInitPreMem entry-point completes the dependency will be satisfied, making SiliconPolicyPeiPreMem eligible for dispatch.

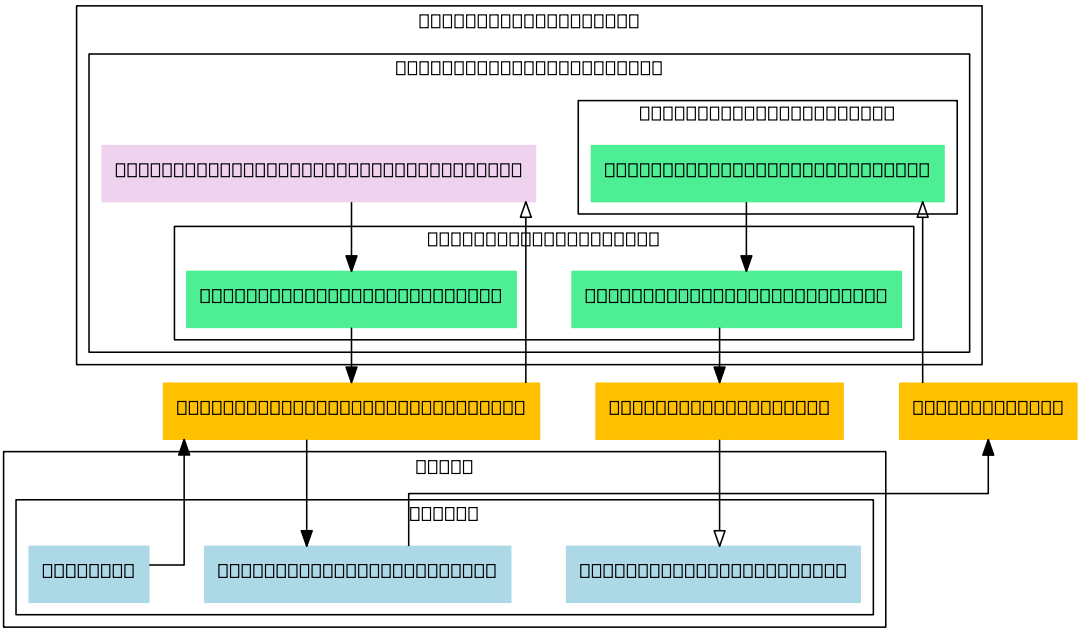
Next SiliconPolicyPeiPreMem PEIM will run. SiliconPolicyPeiPreMemEntryPoint() in MinPlatformPkg/PlatformInit/SiliconPolicyPei/SiliconPolicyPeiPreMem.c will call SiliconPolicyInitPreMem() in PantherLakeSiliconPkg/Library/PeiSiliconPolicyInitLib/PeiPolicyInitPreMem.c. This function will locate the instance of PEI_PREMEM_SI_DEFAULT_POLICY_INIT_PPI previously created by SiInitPreMem and use it to call PeiPreMemSiDefaultPolicyInit() in SiInitPreMem. PeiPreMemSiDefaultPolicyInit() will construct SI_PREMEM_POLICY_PPI, which contains all the policy data needed by FSP-M. SiliconPolicyInitPreMem() will then locate SI_PREMEM_POLICY_PPI and return it to SiliconPolicyPeiPreMemEntryPoint().

Next, SiliconPolicyPeiPreMemEntryPoint() takes the pointer to the policy data returned by SiliconPolicyInitPreMem() and passes it to SiliconPolicyUpdatePreMem(). SiliconPolicyUpdatePreMem() is part of SiliconPolicyUpdateLib, which is statically linked to SiliconPolicyPeiPreMem. SiliconPolicyUpdateLib is special since it is only piece of motherboard specific code in this flow. An example of SiliconPolicyUpdatePreMem() is seen in PantherlakeOpenBoardPkg/PantherlakeMRvp/Policy/Library/PeiSiliconPolicyUpdateLib/PeiSiliconPolicyUpdateLib.c. SiliconPolicyUpdatePreMem() uses a read-modify-write operation to apply any motherboard specific settings to the policy data while leaving the default values intact when appropriate.

After SiliconPolicyUpdatePreMem() applies any motherboard specific updates, SiliconPolicyPeiPreMemEntryPoint() calls SiliconPolicyDonePreMem() in PantherlakeSiliconPkg/Library/PeiSiliconPolicyInitLib/PeiPolicyInitPreMem.c. SiliconPolicyDonePreMem() dumps the policy data to the debug log (on a DEBUG build of MinPlatform only) and installs the SI_PREMEM_POLICY_READY_PPI. SI_PREMEM_POLICY_READY_PPI tells FSP-M that everything is ready and that it can run the SOC init code now. Installing SI_PREMEM_POLICY_READY_PPI causes the PEI Foundation to signal a PeiServices->NotifyPpi() callback previously set up by SiInitPreMem. This callback invokes SiInitPreMemOnPolicyReady() in SiInitPreMem, which runs MRC and all the other SOC init code in FSP-M.

4.1.1 FSP-S Policy Initialization Flow

Policy Initialization for FSP-S follows essentially the same flow as FSP-M. The primary difference is function and PPI names do not include the "PreMem" qualifier.



4.2 Config Blocks

Overview

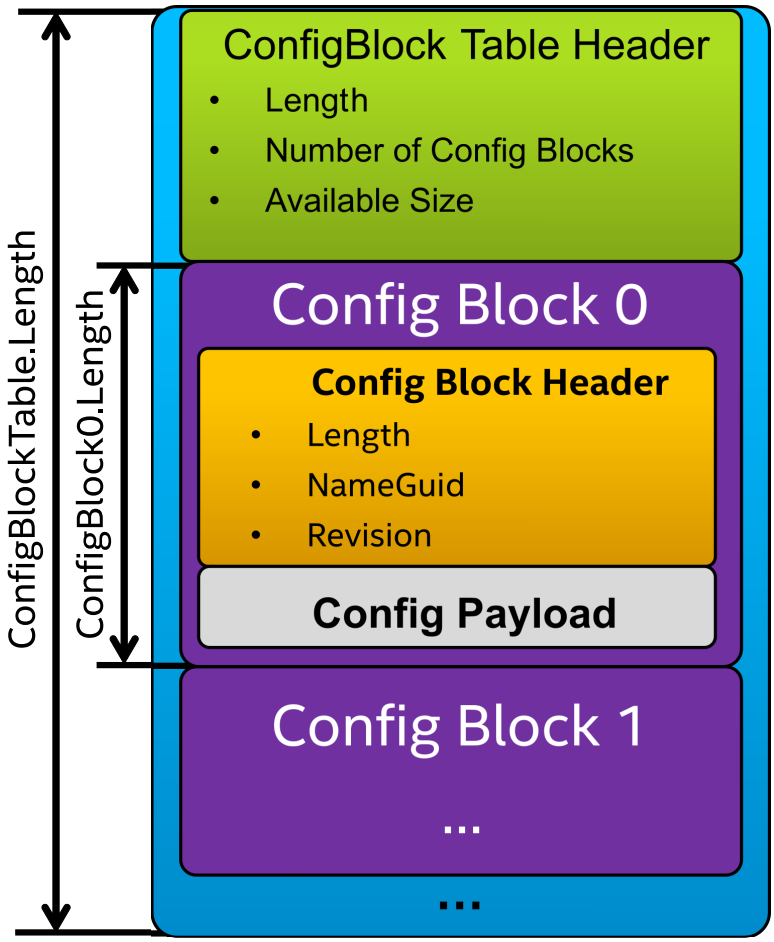
Config Blocks are a binary data serialization format with several unique properties that are useful for firmware implementations.

The serialization format is simple to implement in the C language, minimizing the code needed to implement it; this minimizes the size of the FSP binary. Second, the format is binary backwards compatible; enabling the bootloader to use an older version of the data structure at compile time and retain compatibility with both new and old FSP binaries. Third, the format is mutable in memory without requiring the data to be re-serialized. Finally, the format is position independent with no need for a base address or a rebase operation. This allows the data structure to be moved from pre-memory to post-memory without applying any fixups.

Data in SI_PREMEM_POLICY_PPI and SI_POLICY_PPI are stored using the Config Block format.

4.2.1 Config Block Data Format

In essence, Config Blocks are multiple C structures that are stored in a single continuous memory range using run length encoding.



The overall structure is termed a Config Block Table. The Config Block Table starts with a CONFIG_BLOCK_TABLE_HEADER structure. This header indicates the number of Config Blocks in the Config Block Table, the total size of the table, and the amount of free space remaining in the table. The table may be over-provisioned, which allows additional Config Blocks to be added after the table's initial creation.

The size of all Config Blocks must be an even multiple of 32-bits as all Config Blocks are DWORD aligned. When enumerating a Config Block Table one may assume that all Config Blocks are packed back-to-back, hence the enumeration algorithm can use the length of each config block to find the next config block in the table.

Each Config Block has a GUID that identifies which C structure the Config Block contains. All Config Blocks start with a standardized CONFIG_BLOCK_HEADER, which allows all the structures in the table to be enumerated even if the format of the data payload inside each Config Block is unknown.

```
typedef struct _CONFIG_BLOCK_HEADER {
  EFI_HOB_GUID_TYPE GuidHob;                    ///< Offset 0-23  GUID extension HOB header
  UINT8             Revision;                   ///< Offset 24    Revision of this config block
  UINT8             Attributes;                 ///< Offset 25    The main revision for config block
  UINT8             Reserved[2];                ///< Offset 26-27 Reserved for future use
} CONFIG_BLOCK_HEADER;
```

The CONFIG_BLOCK_HEADER uses the header of a GUID HOB to store its length and GUID. This makes it easy to copy Config Blocks to the HOB list. Copying Config Blocks to the HOB list is a common use case. Often the ACPI code uses policies that were initially supplied by a Config Block. Copying the Config Block to the HOB list makes it easy for DXE phase to copy any needed policies to the ACPI global NVS memory later.

The Revision field assists with binary backward compatibility. Whenever a new field is added to a Config Block, the new field is added to the bottom of the Config Block and the Revision is incremented by 1. Thus if a bootloader wishes to retain compatibility with new and old versions of the FSP binary, it may do so by either by only using fields that are present in Revision 1 of the Config Block or by checking the Revision number before modifying fields added since Revision 1.

The CONFIG_BLOCK_TABLE_HEADER builds upon the CONFIG_BLOCK_HEADER:

```
typedef struct _CONFIG_BLOCK_TABLE_STRUCT {
  CONFIG_BLOCK_HEADER           Header;          ///< Offset 0-27  GUID number for main entry of config
      block
  UINT8                         Rsvd0[2];        ///< Offset 28-29 Reserved for future use
  UINT16                        NumberOfBlocks;  ///< Offset 30-31 Number of config blocks (N)
  UINT32                        AvailableSize;   ///< Offset 32-35 Current config block table size
///
/// Individual Config Block Structures are added starting here
///
} CONFIG_BLOCK_TABLE_HEADER;
```

4.2.2 Config Block Library APIs

To assist in the creation and parsing of Config Blocks, Intel has provided an open source Config Block library in `IntelSiliconPkg/Library/BaseConfigBlockLib`. This library provides the following functions:

- GetConfigBlock()
- AddConfigBlock()
- CreateConfigBlockTable()

In most cases, bootloaders will only need to call GetConfigBlock().

4.2.3 Config Blocks used by FSP-M

On **PantherLake** platforms SI_PREMEM_POLICY_PPI contains a Config Block Table. PEI_PREMEM_SI_DEFAULT_POLICY_INIT_ will construct this Config Block Table with all the below Config Blocks pre-populated. Additionally, all the Config Blocks will be initialized with default policy values as described in section 4.1.

SI_PREMEM_POLICY_PPI contains the following Config Blocks:

Todo This list is updated for PantherLake

- CNVI_PREMEM_CONFIG
- CPU_INIT_PREMEM_CONFIG
- CPU_SECURITY_PREMEM_CONFIG
- FSPV_PREMEM_CONFIG
- HDAUDIO_PREMEM_CONFIG
- HOST_BRIDGE_PREMEM_CONFIG
- IGPU_PEI_PREMEM_CONFIG
- IPU_PREMEM_CONFIG
- ISH_PREMEM_CONFIG
- ME_PEI_PREMEM_CONFIG
- MEMORY_CONFIGURATION
- OVERCLOCKING_PREMEM_CONFIG

- PCH_DCI_PREMEM_CONFIG
- PCH_DMI_PREMEM_CONFIG
- PCH_GENERAL_PREMEM_CONFIG
- PCH_HSIO_PCIE_PREMEM_CONFIG
- PCH_SMBUS_PREMEM_CONFIG
- PCH_TRACE_HUB_PREMEM_CONFIG
- PCH_WDT_PREMEM_CONFIG
- PSMI_PEI_PREMEM_CONFIG
- TCSS_PEI_PREMEM_CONFIG
- TELEMETRY_PEI_PREMEM_CONFIG
- TRACE_HUB_PREMEM_CONFIG
- TXT_PREMEM_CONFIG
- VTD_CONFIG

4.2.4 Config Blocks used by FSP-S

On **PantherLake** platforms SI_POLICY_PPI contains a Config Block Table. PEI_SI_DEFAULT_POLICY_INIT_PPI will construct this Config Block Table with all the below Config Blocks pre-populated. Additionally, all the Config Blocks will be initialized with default policy values as described in section 4.1.

SI_POLICY_PPI contains the following Config Blocks:

Todo This list is updated for PantherLake

- AMT_PEI_CONFIG
- BIOS_GUARD_CONFIG
- CNVI_CONFIG
- CPU_INIT_CONFIG
- CPU_POWER_DELIVERY_CONFIG
- CPU_POWER_MGMT_BASIC_CONFIG
- CPU_POWER_MGMT_VR_CONFIG
- DMI_CONFIG
- FSPV_POSTMEM_CONFIG
- FUSA_CONFIG
- GBE_CONFIG
- GPIOV2_PEI_CONFIG
- GPIO_DXE_CONFIG
- HDAUDIO_CONFIG

- HDAUDIO_DXE_CONFIG
- HOST_BRIDGE_PEI_CONFIG
- HSIO_CONFIG
- HYBRID_GRAPHICS_CONFIG
- IAX_PEI_CONFIG
- IEH_CONFIG
- IGPU_DXE_CONFIG
- IGPU_PEI_CONFIG
- ISH_CONFIG
- LPSS_I2C_CONFIG
- LPSS_I3C_CONFIG
- LPSS_SPI_CONFIG
- LPSS_UART_CONFIG
- ME_DXE_CONFIG
- MEMORY_SUB_SYSTEM_CONFIG
- ME_PEI_CONFIG
- NPU_PEI_CONFIG
- PCH_DMI_CONFIG
- PCH_ESPI_CONFIG
- PCH_GENERAL_CONFIG
- PCH_INTERRUPT_CONFIG
- PCH_IOAPIC_CONFIG
- PCH_P2SB_CONFIG
- PCH_PCIE_CONFIG
- PCH_PM_CONFIG
- PCIE_RP_DXE_CONFIG
- PEI_DMI_CONFIG
- PEI_ITBT_CONFIG
- PSF_CONFIG
- RTC_CONFIG
- SATA_CONFIG
- SCS_SDCARD_CONFIG
- SERIAL_IO_CONFIG
- SPI_CONFIG
- SWIZZLE_CONFIG
- TCSS_PEI_CONFIG

- TELEMETRY_PEI_CONFIG
- THC_CONFIG
- THERMAL_CONFIG
- TSN_CONFIG
- USB_CONFIG
- VMD_PEI_CONFIG

4.3 FSP Error Information

In the case of a fatal error occurring during the execution of the FSP, it may not be possible for the FSP to continue.

If a fatal error that prevents the successful completion of the FSP occurs, the FSP may use FSP_ERROR_INFO to report this error to the bootloader. During PEI phase, (*PeiServices)->ReportStatusCode() shall be used to transmit this error notification to the bootloader. During DXE phase, EFI_STATUS_CODE_PROTOCOL.Report↩StatusCode() shall be used to transmit this error notification to the bootloader. The bootloader must ensure that ReportStatusCode() services are available before FSP-M begins execution.

```
typedef struct {
  EFI_STATUS_CODE_DATA    DataHeader;
  EFI_GUID                ErrorType;
  EFI_STATUS              Status;
} FSP_ERROR_INFO;
```

FSP_ERROR_INFO is provided as the optional EFI_STATUS_CODE_DATA parameter to ReportStatusCode(). EFI_STATUS_CODE_DATA provides a CallerId GUID, this CallerId combined with the ErrorType GUID describes the error to the bootloader. The FSP for this platform implements the following CallerId GUIDs:

CallerId	Description
{0x1f4dc7e9, 0x26ca, 0x4336, {0x8c, 0xe3, 0x39, 0x31, 0x03, 0xb5, 0xf3, 0xd7}}	ME
{0x98230916, 0xe632, 0x49ff, {0x81, 0x81, 0x55, 0xce, 0xe5, 0x10, 0x36, 0x89}}	System Agent
{0x5a47c211, 0x642f, 0x4f92, {0x9c, 0xb3, 0x7f, 0xeb, 0x93, 0xda, 0xdd, 0xba}}	MRC

If the CallerId parameter is not a GUID in the table above, then it should be the GUID that identifies the PEIM or DXE driver which was executing at the time of the error.

The following ErrorType GUIDs are implemented:

ErrorType	Description
{0x948585c4, 0x76a4, 0x45bb, {0xbe, 0x6c, 0x39, 0x61, 0xc3, 0xab, 0xde, 0x15}}	ME EOP failure
{0x8106a5cc, 0x30ba, 0x41cf, {0xa1, 0x78, 0x63, 0x38, 0x91, 0x11, 0xae, 0xb2}}	SA PEI GOP Init failure
{0x348cc7fe, 0x1e9a, 0x4c7a, {0x86, 0x28, 0xae, 0x48, 0x5b, 0x42, 0x10, 0xf0}}	SA PEI GOP GetMode failure
{0x5de1c071, 0x2c9c, 0x4a53, {0x80, 0x21, 0x4e, 0x80, 0xd2, 0x5d, 0x44, 0xa8}}	MRC training failure

When the FSP calls ReportStatusCode(), the Type parameter's EFI_STATUS_CODE_TYPE_MASK must be EFI↩_ERROR_CODE with the EFI_STATUS_CODE_SEVERITY_MASK >= EFI_ERROR_UNRECOVERED. The Value

and Instance parameters must be 0.

The bootloader must register a listener for this status code. This listener should check if DataHeader.↩ Type == STATUS_CODE_DATA_TYPE_FSP_ERROR_GUID to detect an FSP_ERROR_INFO notification. If an FSP_ERROR_INFO notification is encountered, the bootloader should assume that normal operation is no longer possible. In debug scenarios, this notification should be considered an ASSERT.

4.4 Dispatch Mode Integration

Dispatch Mode Integration Notes:

1. The FSP for this platform contains PEIMs compiled for the IA32 architecture. The boot loader therefore must utilize a PEI Foundation compiled for the IA32 architecture.

2. Since the FSP binary can be integrated into flash at any address, the boot loader has to report FSP FVs to the PEI and DXE dispatcher using PI specification defined mechanisms so PEIMs and DXE drivers inside the FSP Binary can be dispatched. FspmWrapperPeim and FspsWrapperPeim from IntelFsp2WrapperPkg can aid in implementing this.

3. For this platform, FSP-T, FSP-M, and FSP-S contain 1 FV each.

4. The FSP distribution package will include a DSC file which contains all DynamicEx PCDs consumed by the FSP binary. The boot loader should include the DSC during its build process so that any PCDs defined by this DSC file are included in the boot loader's PCD database. This enables the boot loader and FSP to share a single PCD database.

   - A NULL library (FspPcdListLibNull.inf) is included in the FSP distribution package. This library should be included in one of the boot loader's PEIMs. This ensures all DynamicEx PCDs used by the FSP are included in the boot loader's PCD database. One can fulfill this requirement by including the following code snippet in ∗BoardPkg.dsc:

```
IntelFsp2WrapperPkg/FspmWrapperPeim/FspmWrapperPeim.inf {
  <LibraryClasses>
  !if gIntelFsp2WrapperTokenSpaceGuid.PcdFspModeSelection == 0
    #
    # In FSP Dispatch mode below dummy library should be linked to bootloader PEIM
    # to build all DynamicEx PCDs that FSP consumes into bootloader PCD database.
    #
    NULL|$(PLATFORM_FSP_BIN_PACKAGE)/Library/FspPcdListLib/FspPcdListLibNull.inf
  !endif
}
```

5. The boot loader must provide at minimum 256KB of stack and 160KB of HOB heap to execute FSP on this platform.

6. In dispatch mode, the boot loader should not use FSP API calls described in chapter 5 of this document or chapter 9 of the FSP External Architecture Specification version 2.5. The TempRamInit API is the only exception, it is supported in both API mode and dispatch mode. All other APIs (MemoryInit, SiliconInit, etc.) should not be invoked.

7. For dispatch mode, FSP contains x64 DXE drivers to replace the NotifyPhase API. This eliminates thunking from 64bit to 32bit when using FSP dispatch mode. The boot loader should remove S3EndOfPeiNotify and FspWrapperNotifyDxe since they are no longer used in dispatch mode.

8. EFI_PEI_CORE_FV_LOCATION_PPI should be installed by the boot loader's SEC phase. EFI_PEI_C↩ ORE_FV_LOCATION_PPI.PeiCoreFvLocation should point to the first Firmware Volume (FV) in FSP-M so the PeiCore inside FSP will be invoked. If EFI_PEI_CORE_FV_LOCATION_PPI is not installed or PeiCore cannot be found at the address specified by EFI_PEI_CORE_FV_LOCATION_PPI.PeiCoreFvLocation, the PeiCore from the Boot Firmware Volume (BFV) will be invoked instead.

9. FSP-S requires multi-threaded code to complete silicon initialization on this platform. FSP-S includes the `UefiCpuPkg/CpuMpPei/CpuMpPei.inf` PEIM to provide multiprocessing. The bootloader can choose to either use the MP_SERVICES provided by the FSP for all PEIMs or the bootloader may provide an alternative implementation. In either case, only one MP_SERVICES implementation can be active at once. If the bootloader wishes to not use the FSP provided MP_SERVICES, then the bootloader must install the MP_SERVICES_PPI before installing the FV_INFO_PPI for FSP-S. If MP_SERVICES_PPI already exists, then FSP-S will use it and not execute its own implementation.

10. If you are using the Intel Reference BIOS, some EDK2 overrides may be required for Dispatch Mode, please refer to override folders in reference code or the override EDK2 repo for more details. For open source MinPlatform based firmware, no EDK2 overrides are required.

11. FSPM_ARCH_CONFIG_PPI->NvsBufferPtr is now a universal policy option (FSP Dispatch Mode and EDK2 native.) To enable the fast MRC training flow, the boot loader or platform code must to install this PPI to restore the previous MRC training data (SA_MISC_PEI_PREMEM_CONFIG->S3DataPtr is obsolete).

12. Policy initialization Flow Changes:

    - PEIMs from FSP-M/FSP-S should be dispatched early in PEI to produce the *DefaultPolicyInit* PPIs. -> Bootloader consumes the *DefaultPolicyInit* PPIs produced by the FSP binary to create the policy PPIs with default settings. -> Bootloader then locates and updates the policy PPIs as needed. -> Bootloader installs the *PolicyReadyPpi* after policy updates are completed. This signals to the FSP that silicon initialization may proceed.
    - Bootloader shall consume two PPIs produced by FSP binary to create policy PPIs with default settings. These PPIs are:
      - PEI_PREMEM_SI_DEFAULT_POLICY_INIT_PPI
      - PEI_SI_DEFAULT_POLICY_INIT_PPI
    - The bootloader shall call the two functions below after the bootloader has completed any needed policy updates:
      - SiPreMemInstallPolicyReadyPpi()
      - SiInstallPolicyReadyPpi()

13. Debug message handling in dispatch mode:

    - Before the ReportStatusCode service is ready, a debug built FSP will send debug messages using the FSP-T UPD configuration (passed as FSP-T API input parameter). FSP-T is recommended to be used regardless FSP API mode or Dispatch mode.
    - Once the ReportStatusCode service is ready, a debug built FSP will send debug messages using the ReportStatusCode service.
    - It is recommended that bootloader register a StatusCode listener immediately after the ReportStatus↩ Code service is ready. It is important to register this listener as soon as possible so that all debug messages sent by the FSP are captured.
    - Please refer to section 10.4.12 in the Intel(R) Firmware Support Package External Architecture Specification v2.5 for details about the ReportStatusCode debug message format.

14. Enable or Disable FSP S3BootScript functionality: Since edk2-stable201911 release the PiDxeS3Boot↩ ScriptLib supports enabling or disabling S3BootScript functionality by PCD, and FSP consumes this PCD as DynamicEx type to enable or disable internal S3BootScript functionality. Bootloader must configure below PCD before entering DXE phase or before executing FSP DXE drivers.

    - gEfiMdeModulePkgTokenSpaceGuid.PcdAcpiS3Enable = TRUE if S3BootScript should be enabled.
    - gEfiMdeModulePkgTokenSpaceGuid.PcdAcpiS3Enable = FALSE if S3BootScript should be disabled.

Chapter 5

FSP API Mode

Overview

This release of the FSP supports the all APIs required by the FSP External Architecture Specification version 2.5. These APIs are only used when running the FSP in API mode. In Dispatch mode, these APIs are not used (with the exception of TempRamInit.) The FSP information header contains the address offset for these APIs. Register usage is described in the FSP External Architecture Specification version 2.5. Any usage not described by the specification is described in the individual sections below.

The sections below will highlight any changes that are specific to this FSP release.

5.1 FSP APIs

5.1.1 TempRamInit API

Please refer Chapter 9.7 in the FSP External Architecture Specification version 2.5 for complete details including the prototype, parameters and return value details for this API.

TempRamInit does basic early initialization primarily setting up temporary RAM using cache. It returns ECX pointing to beginning of temporary memory and EDX pointing to end of temporary memory + 1. The total temporary ram currently available is given by FspTemporaryRamSize (FSPT_ARCH2_UPD) starting from the base address of PcdTemporaryRamBase, Out of the total temporary memory available, the last PcdFspReservedBufferSize bytes of space are reserved by the FSP for TempRamInit if temporary RAM initialization is done by the FSP. Any remaining space from **TemporaryRamBase**(ECX) to TemporaryRamBase+TemporaryRamSize-FspReserved↩ BufferSize∗∗ (EDX) is available for both bootloader and FSP use.

As defined in FSP External Architecture Specification version 2.5, If BIT4 in the ImageAttribute field of the FSP_I↩ NFO_HEADER is set, bootloader can pass temporary RAM size as input to FspTempRamInit() API. Value passed by Bootloader shall not exceed the maximum temporary Ram size defined. If BIT4 in the ImageAttribute field of the FSP_INFO_HEADER is not set, bootloader must pass a value of 0 or FSP defined default value as FspTemporary↩ RamSize. If bootloader passes a value of 0 as FspTemporaryRamSize then FspTempRamInit () API will use FSP defined default value. Both the maximum temporary size and intel defined default value is same that is 1 MB.

TempRamInit∗∗ also sets up the code caching of the region passed in through CodeCacheAddress and Length, which are input parameters to TempRamInitApi. if 0 is passed in for Base, the base used will be (4 GB - 1 - CodeCacheSize).

Note

: When programming MTRRs CodeCacheSize will be reduced, if the LLC size on the current processor is smaller than the requested size.

It is a requirement for Firmware to have a Firmware Interface Table (FIT). The FIT contains pointers to each microcode update. The microcode update is loaded for all logical processors before executing the reset vector. If more than one microcode update for the CPU is present, the microcode update with the latest revision is loaded.

FSPT_UPD.MicrocodeRegionBase∗∗ and **FSPT_UPD.MicrocodeRegionSize** are input parameters to TempRamInit API. If these values are 0, FSP will not attempt to update microcode. If MicrocodeRegionBase != 0 and MicrocodeRegionSize != 0, then FSP will search that region for microcode updates. If a newer microcode update revision is found in the region, FSP will load it.

TempRamInit∗∗ will program MTRRs values to provide the following memory map:

Memory range	Cache Attribute
0xFEF00000 - 0xFEFFEFFF	Write back
CodeCacheAddress - CodeCacheSize	Write protect

5.1.2 FspMemoryInit API

Please refer to Chapter 9.8 in the FSP External Architecture Specification version 2.5 for the prototype, parameters and return value details for this API.

The variable **FspmUpdDataPtr** is a pointer to the **FSPM_UPD** structure which is described in the header file FspmUpd.h.

The bootloader must pass valid a CAR region for FSP through **FSPM_UPD.FspmArchUpd.StackBase** and **FSPM_UPD.FspmArchUpd.StackSize** UPDs.

The FSP for this platform will run FspMemoryInit top of the stack provided by the bootloader instead of establishing a separate stack as described by the FSP External Architecture Specification version 2.1/2.2. The memory region provided by the **FSPM_UPD.FspmArchUpd.StackBase** and **FSPM_UPD.FspmArchUpd.StackSize** UPDs is used to establish a HOB heap. The names **StackBase** and **StackSize** can be confusing since they are **NOT** used for stack. These names were retained for backwards compatibility with FSP v2.0.

Below are the heap and stack requirements for FSP on this platform:

HOB Heap requirement:

HOB Heap	UPD	Setting
Base	FSPM_UPD.FspmArchUpd.StackBase	Any non-conflict CAR region (0xFEF17F00 as default)
Size	FSPM_UPD.FspmArchUpd.StackSize	at least 160KB

Stack requirement: FSP's stack usage starts from the current stack pointer which will be set by bootloader and FSP will continue to use the same stack pointer. The minimum stack size requirement for FSP-M is 256KB.

The bootloader must ensure that sufficient stack space is available to fulfill the FSP-M minimum stack size requirement at the point in execution where FspMemoryInit() is called. The stack allocated by the bootloader must be large enough for both FSP-M as well as any other parent function calls that are still on the stack at the point when FspMemoryInit() is called.

After FspMemoryInit() is completed, permanent memory is available. After this point, the memory pressure experienced early in boot is eliminated. Accordingly, right before FspMemoryInit() exits, any data that needs to be retained for later use by FspSiliconInit() will be copied to permanent memory. FspSiliconInit() will then execute on a second stack.

The base address of HECI device (Bus 0, Device 22, Function 0) is required to be initialized prior to calling FspMemoryInit(). The default address is programmed to 0xFED1A000.

FspMemoryInit() will calculate the memory map by taking into account the size of several memory regions: TSEG, IED, GTT, BDSM, ME stolen, Uncore PMRR, IOT, MOT, DPR, REMAP, TOLUD, TOUUD. These memory regions may not be initialized by FspMemoryInit(), but space will be reserved for them.

5.1.3 TempRamExit API

Please refer to Chapter 9.9 in the FSP external Architecture Specification version 2.5 for the prototype, parameters and return value details for this API.

If Boot Loader initializes the Temporary RAM (CAR) and skip calling **TempRamInit API**, it is expected that bootloader must skip calling this API and bootloader will tear down the temporary memory area setup in the cache and bring the cache to normal mode of operation.

The FSP for this platform doesn't have any input parameters for this API. The value of *TempRamExitParamPtr* should be NULL.

At the end of *TempRamExit* the original code and data cache are disabled. FSP will reconfigure all MTRRs as described in the table below. These MTRR values optimize performance in most scenarios. If the boot loader wishes to configure the MTRRs differently, they can be reprogrammed immediately after this API call.

Memory range	Cache Attribute
0xFF000000 - 0xFFFFFFFF (Flash region)	Write protect
0x00000000 - Top of Low Memory	Write back
0x7E000000 - 0x7FFFFFFF	Write back
xxxx - xxxx	x ∗Note1
0x100000000 - Top of High Memory	Write back ∗Note2

Stack requirement: 4KB of free stack space should be provided to execute **TempRamExit**.

Note1: Certain silicon features require specific cache types for specific memory ranges. These ranges will be configured by FSP when such features are enabled.

Note2: In some cases MTRRs might not be enough to cover all desired regions, in this case memory regions need to be adjusted for better alignment (e.g., adjust MmioSize or MmioSizeAdjustment UPD) Covering flash region and above 4GB memory is another case which may consume more MTRRs, when there is not enough MTRR available FSP will only cover above 4GB memory partially. In this case the boot loader can optimize MTRRs to remove flash from the cached regions after all needed data is loaded from flash and before booting the OS.

5.1.4 FspSiliconInit API

Please refer to Chapter 9.10 in the FSP external Architecture Specification version 2.5 for the prototype, parameters and return value details for this API.

The variable *FspsUpdDataPtr* is a pointer to the **FSPS_UPD** structure which is described in the header file FspsUpd.h.

It is expected that the boot loader will adjust MTRRs for SBSP if needed after **TempRamExit** but before entering **FspSiliconInit**. If the MTRRs are not programmed properly, boot performance can be impacted.

The region of 0x5_8000 - 0x5_8FFF is used by FspSiliconInit for starting APs. If this data is important to bootloader, then bootloader needs to preserve it before calling FspSiliconInit.

FspSiliconInit requires multi-threaded code to complete silicon initialization on this platform. FSP includes the `UefiCpuPkg/CpuMpPei/CpuMpPei.inf` PEIM to provide multiprocessing. Accordingly, the boot loader must expect FSP to perform an INIT-SIPI-SIPI during **FspSiliconInit** and **NotifyPhase** which will take control of all APs in the system and restart them. This will kill any background threads that were running on the APs. In some cases, this may not be desirable. As an alternative, the boot loader may provide an instance of the EFI_PEI_MP_SERVICES_PPI through the `FspsUpd->FspsConfig.CpuMpPpi` UPD for the FSP to use instead of the built-in implementation. **This is entirely optional**, if callbacks from FSP to the boot loader are not desired, one may set `FspsUpd->FspsConfig.CpuMpPpi == NULL`.

In summary, there are two methods that FSP can use for performing multiprocessing:

1. Using the multiprocessing services built-in to the FSP.
2. Using the boot loader's multiprocessing services.

Using method #1, the boot loader will set `FspsUpd->FspsConfig.CpuMpPpi == NULL`. If this UPD is NULL, then FSP will use its built-in MP services implementation for multiprocessing. Accordingly, the boot loader must expect FSP to perform an INIT-SIPI-SIPI during **FspSiliconInit** and **NotifyPhase** which will take control of all APs in the system and restart them. This will kill any background threads that were running on the APs.

Note: If bootloader needs to take control of APs back, a full AP re-initialization is required after FSP-S completed and control has been transferred back to bootloader.

Using method #2, the boot loader will set `FspsUpd->FspsConfig.CpuMpPpi != NULL`. If this UPD is not NULL, it must point to an instance of EFI_PEI_MP_SERVICES_PPI. When the FSP needs to perform multiprocessing, it will use the EFI_PEI_MP_SERVICES_PPI instance provided by the boot loader to do so. Accordingly, the boot loader must expect the function pointers in EFI_PEI_MP_SERVICES_PPI to be invoked in the middle of the execution of **FspSiliconInit** and **NotifyPhase**.

It is a requirement for the bootloader to have a Firmware Interface Table (FIT), which contains pointers to each microcode. The microcode is loaded for all cores before reset vector. If more than one microcode update for the CPU is present, the latest revision is loaded.

MicrocodeRegionBase and MicrocodeRegionSize are both input parameters to TempRamInit and UPD for SiliconInit API. UPD has priority and will be searched for a later revision than TempRamInit. If MicrocodeRegionBase and MicrocodeRegionSize values are 0, FSP will not attempt to update the microcode. If a microcode region is passed, FSP will search that region for microcode updates. If a newer microcode update revision is found in the region, FSP will load it.

FspSiliconInit initializes PCH audio. This includes selecting the HD Audio verb table and initializing the audio CODEC.

FspSiliconInit initializes the following PCH features: HECI, USB, HSIO, Integrated Sensor Hub, Camera, PCI Express, DMI, Vt-d.

FspSiliconInit initializes the following CPU features: XD, VMX, AES, IED, HDC, x(2)APIC, Intel® Processor Trace, Three Strike Counter, Machine Check, Cache Pre-fetchers, Core PMRR, and Power Management.

FspSiliconInit initializes Internal Graphics. During this initialization, FSP publishes the EFI_PEI_GRAPHICS_INFO_HOB and the EFI_PEI_GRAPHICS_DEVICE_INFO_HOB. These HOBs will not be published during S3 resume as FSP will not initialize the internal graphics frame buffer during S3 resume.

FspSiliconInit programs SA BARs: MchBar, DmiBar, EpBar, GdxcBar, EDRAM (if supported). Please refer to section 2.8 (MemoryMap) for the corresponding Bar values. GttMmadr (0xDF000000) and GmAdr(0xC0000000) are temporarily programmed and cleared after use in FSP.

Stack requirement: 4KB of free stack space should be provided to execute *FspSiliconInit*.

5.1.5 FspMultiPhaseSiInit API

Please refer Chapter 9.11 in the FSP External Architecture Specification version 2.5 for the prototype, parameters and return value details for this API.

If the FspMultiPhaseSiInit API is enabled (by setting FSPS_ARCH2_UPD->EnableMultiPhaseSiliconInit UPD to 1 before the FspSiliconInit API is invoked), the bootloader should call this API with the MultiPhaseAction parameter set to 0 to get the NumberOfPhases supported by this platform. FSP will return control back to the bootloader NumberOfPhases times and then the bootloader should call FspMultiPhaseSiInit API with the MultiPhaseAction parameter set to 1 by NumberOfPhases times.

Note

: Disable FSPS_ARCH2_UPD->EnableMultiPhaseSiliconInit when we disable FSP_M_CONFIG->Tcss↩ XhciEn.

This platform supports the following MultiPhaseSiInit phase(s):

Phase	FSP return point	Purpose
1	After TCSS initialization completed	for TCSS board specific initialization from bootloader side

5.1.6 NotifyPhase API

Please refer Chapter 9.13 in the FSP External Architecture Specification version 2.5 for the prototype, parameters and return value details for this API.

Stack requirement: 4KB of free stack space should be provided to execute *NotifyPhase*.

5.1.6.1 PostPciEnumeration Notification

This phase *EnumInitPhaseAfterPciEnumeration* is to be called after PCI enumeration but before execution of third party code such as option ROMs. It includes powering down unused PCH SATA ports, locking registers in PMC, DMI, TCO, and SPI Flash (DLOCK + WRSDIS + FLOCKDN). It also includes enabling bus mastering for eSPI connected devices.

5.1.6.2 ReadyToBoot Notification

This phase *EnumInitPhaseReadyToBoot* is to be called before giving control to the operating system. It includes some final initialization steps recommended by the BWG, including power management settings, and sending ME Message EOP (End of Post).

5.1.6.3 EndOfFirmware Notification

This phase *EnumInitPhaseEndOfFirmware* is to be called before the firmware/preboot environment transfers management of all system resources to the OS or next level execution environment. It includes final locking of chipset registers.

5.1.7 FSP Events API

Please refer Chapter 9.5 in the FSP External Architecture Specification version 2.5 for the prototype, parameters and return value details for these APIs.

This platform supports FSP Events which re-direct debug messages to event handlers provided by bootloader. To enable this function the bootloader must provide event handler function pointers via UPDs.

FSP phase	UPD for bootloader to pass handler function pointer
FSP-T	FSPT_ARCH2_UPD -> FspDebugHandler
FSP-M	FSPM_ARCH2_UPD -> FspEventHandler
FSP-S	FSPS_ARCH2_UPD -> FspEventHandler

5.2 Reset Return Codes

As per FSP External Architecture Specification version 2.0/2.1/2.2/2.4/2.5, any reset required in the FSP flow will be reported by returning one of the FSP_STATUS_RESET_REQUIRED∗ return codes.

It is the boot loader's responsibility to reset the system according to the reset type requested.

Below table specifies the return status returned by FSP API and the requested reset type.

FSP_STATUS_RESET_REQUIRED Code	Reset Type requested
0x4000000000000001ULL	Cold Reset
0x4000000000000002ULL	Warm Reset
0x4000000000000003ULL	Global Reset - Puts the system through a Global Reset through HECI or a Full Reset through PCH
0x4000000000000004ULL	Reserved
0x4000000000000005ULL	Reserved
0x4000000000000006ULL	Reserved
0x4000000000000007ULL	Reserved
0x4000000000000008ULL	Reserved

5.3 UPD Porting Guide

Recommended values for UPDs:

UPD	Dependency	Description	Value
CstateLatencyControl1Irtl	Server platform	Server platform should has different setting	0x6B
PchPcieHsioRxSetCtleEnable	Board design	Different board requires different value	tune
PchPcieHsioRxSetCtle	Board design	Different board requires different value	tune
PchSataHsioRxGen3EqBoostMag↩Enable	Board design	Different board requires different value	tune
PchSataHsioRxGen3EqBoostMag	Board design	Different board requires different value	tune
PchSataHsioTxGen1DownscaleAmp↩Enable	Board design	Different board requires different value	tune
PchSataHsioTxGen1DownscaleAmp	Board design	Different board requires different value	tune
PchSataHsioTxGen2DownscaleAmp↩Enable	Board design	Different board requires different value	tune
PchSataHsioTxGen2DownscaleAmp	Board design	Different board requires different value	tune
PchNumRsvdSmbusAddresses	Board design	Different board requires different value	tune
RsvdSmbusAddressTablePtr	Board design	Different board requires different value	tune
BiosSize	Board design	Different board requires different value	tune

Chapter 6

Porting Recommendations

Here are some notes and recommendations for adapting an existing boot loader to FSP.

6.1 Locking PAM register

FSP 2.0 introduced the EndOfFirmware Notify phase callback which is the recommended place for locking PAM registers. Accordingly, by default FSP locks the PAM registers during the EndOfFirmware Notify phase. If the EndOf↩ Firmware Notify phase is too early to lock PAM registers, then the PAM locking code inside the FSP can be disabled by setting the UPD -> FSP_S_TEST_CONFIG -> SkipPamLock or SA policy -> _SI_PREMEM_POLICY_STRUCT -> SA_MISC_PEI_CONFIG -> SkipPamLock. If the PAM locking code inside the FSP is skipped, then the boot loader must lock the PAM registers before booting the OS. by programming one PCI config space register as below.

The PAM registers must be locked in all boot paths including S3 resume. To lock the PAM registers, program the following lock bit:

```
MmioOr32 (B0: D0: F0: Register 0x80, BIT0)
```

6.2 Locking SMRAM register

It is recommended that SMRAM be locked before any third party code (e.g. OpROM) execution. The point in execution where third party OpROMs begin executing can vary depending on the boot loader implementation. To provide flexibility, the FSP by default will not lock it. The boot loader should lock SMRAM by programming the following lock bit before any third party OpROM execution. Additionally, SMRAM should be locked before the **EnumInitPhaseReadyToBoot** notify phase is called. During S3 resume, the lock bit should be set right before the OS wake vector.

```
PciOr8 (B0: D0: F0: Register 0x88, BIT4);
Note: This register must be programmed using the legacy CF8/CFC PCI access mechanism. (MMIO access will not
       work)
```

6.3 Locking SMI register

It is recommended that the global SMI bit is locked before any third party code (e.g. OpROM) execution. SMM initialization flows may vary depending on boot loader implementation details. Accordingly, FSP will not lock it by default. The boot loader is responsible for locking the following registers after SMM configuration is complete. Set AcpiBase + 0x30[0] to 1b to enable global SMI. Set PMC PCI offset A0h[4] = 1b to lock SMI.

6.4 Verify below settings are correct for your platforms

PMC PCI configuration space is not PCI specification compliant. The FSP will hide the PMC controller to avoid external software or OS from corrupting the BAR addresses. FSP will program the PMC controller IO and MMIO BAR's with below addresses. Please use these addresses in the boot loader code instead of reading BAR addresses from the PMC controller.

Register	Values
ABASE	0x1800
PWRMBASE	0xFE000000
PCIEXBAR_BASE_ADDRESS	0xE0000000

Note

:

- Boot Loader can use a different value for PCIEXBAR_BASE_ADDRESS either by modifying the UPD (under FSP-T) or by overriding the PCIEXBAR (B0:D0:F0:R60h) before calling FspMemoryInit.
- Boot Loader should avoid using conflicting address when reprogramming PCIEXBAR_BASE_ADDRESS than the recommended one.

6.5 Locking SPI DMA Engine

-The SPI-DMA is initialized early to enable access to SPI flash memory for reading Firmware Volumes. By default, SPI-DMA is locked after Silicon Initialization (FSP-S). Intel recommends locking SPI-DMA after Silicon Initialization for platform security.

Chapter 7

FSP Output

The FSP builds a series of data structures called Hand-Off-Blocks (HOBs) as it progresses through initializing the silicon.

Please refer to the Platform Initialization (PI) Specification - Volume 3: Shared Architectural Elements specification for PI Architectural HOBs. Please refer Chapter 11 in the FSP External Architecture Specification version 2.5 for details about FSP Architectural HOBs.

The sections below describe the HOBs not covered in the above two specifications.

7.1 Resource Descriptor HOB

Below resource HOBs with specific owner GUID are publicized to specify memory ranges used by FSP. Bootloader shall check and not consume these ranges to avoid memory collision.

7.1.1 SMRAM Resource Descriptor HOB

The FSP will report the system SMRAM T-SEG range through a generic resource HOB if T-SEG is enabled. The owner field of the HOB identifies the owner as T-SEG.

```
#define FSP_HOB_RESOURCE_OWNER_TSEG_GUID  \
{ 0xd038747c, 0xd00c, 0x4980, { 0xb3, 0x19, 0x49, 0x01, 0x99, 0xa4, 0x7d, 0x55 } }
```

7.1.2 TraceHub Resource Descriptor HOB

TraceHub Resource HOB is reported when TraceHub is enabled or PlatformDebugConsent UPD is 2 (Enabled) to reserve memory above 4G for TraceHub use.

```
#define FSP_HOB_RESOURCE_OWNER_TRACEHUB_GUID  \
{ 0x5fb35909, 0x5a1c, 0x4a31, { 0xad, 0xaf, 0x57, 0x7b, 0x54, 0x68, 0x26, 0x3f } }
```

7.1.3 IGPU Flat CCS Memory Resource Descriptor HOB

FSP reports the IGPU Flat CCS Memory Resource Descriptor HOB when the MemoryBandwidthCompression UPD is enabled. FSP allocates and reserves an extra memory region specifically for Integrated Graphics Processing Unit (IGPU) Compression Control Surface (CCS) functionality. The bootloader must identify this HOB by its owner GUID and ensure this memory region is marked as reserved in the system memory map to prevent operating system access.

```
#define FSP_HOB_RESOURCE_OWNER_IGPU_FLAT_CCS_MEMORY_GUID \
{ 0x8a9f1579, 0xea72, 0x4bb5, { 0x90, 0x69, 0x54, 0x9a, 0x1b, 0xf7, 0xc4, 0xfd } }
```

7.2 SMBIOS INFO HOB

The FSP will report the SMBIOS through a HOB with below GUID. This information can be consumed by the bootloader to produce the SMBIOS tables. These structures are included as part of MemInfoHob.h, SmbiosCacheInfoHob.h, SmbiosProcessorInfoHob.h, & FirmwareVersionInfoHob.h

```
#define SI_MEMORY_INFO_DATA_HOB_GUID \
{ 0x9b2071d4, 0xb054, 0x4e0c, { 0x8d, 0x09, 0x11, 0xcf, 0x8b, 0x9f, 0x03, 0x23 } };
typedef struct {
  MrcDimmStatus Status;                    ///< See MrcDimmStatus for the definition of this field.
  UINT8         DimmId;
  UINT32        DimmCapacity;              ///< DIMM size in MBytes.
  UINT16        MfgId;
  UINT8         ModulePartNum[20];         ///< Module part number for DDR3 is 18 bytes however for DDR4 20
      bytes as per JEDEC Spec, so reserving 20 bytes
  UINT8         RankInDimm;                ///< The number of ranks in this DIMM.
  UINT8         SpdDramDeviceType;         ///< Save SPD DramDeviceType information needed for SMBIOS
      structure creation.
  UINT8         SpdModuleType;             ///< Save SPD ModuleType information needed for SMBIOS structure
      creation.
  UINT8         SpdModuleMemoryBusWidth;   ///< Save SPD ModuleMemoryBusWidth information needed for SMBIOS
      structure creation.
  UINT8         SpdSave[MAX_SPD_SAVE_DATA]; ///< Save SPD Manufacturing information needed for SMBIOS
      structure creation.
} DIMM_INFO;
typedef struct {
  UINT8         Status;                    ///< Indicates whether this channel should be used.
  UINT8         ChannelId;
  UINT8         DimmCount;                 ///< Number of valid DIMMs that exist in the channel.
  MRC_CH_TIMING Timing[MAX_PROFILE];       ///< The channel timing values.
  DIMM_INFO     Dimm[MAX_DIMM];            ///< Save the DIMM output characteristics.
} CHANNEL_INFO;
typedef struct {
  UINT8            Status;                 ///< Indicates whether this controller should be used.
  UINT16           DeviceId;               ///< The PCI device id of this memory controller.
  UINT8            RevisionId;             ///< The PCI revision id of this memory controller.
  UINT8            ChannelCount;           ///< Number of valid channels that exist on the controller.
  CHANNEL_INFO     Channel[MAX_CH];        ///< The following are channel level definitions.
} CONTROLLER_INFO;
typedef struct {
  EFI_HOB_GUID_TYPE EfiHobGuidType;
  UINT8             Revision;
  UINT16            DataWidth;
  /// As defined in SMBIOS 3.0 spec
  /// Section 7.18.2 and Table 75
  UINT8             DdrType;               ///< DDR type: DDR3, DDR4, or LPDDR3
  UINT32            Frequency;             ///< The system's common memory controller frequency in MT/s.
  /// As defined in SMBIOS 3.0 spec
  /// Section 7.17.3 and Table 72
  UINT8             ErrorCorrectionType;
  SiMrcVersion      Version;
  UINT32            FreqMax;
  BOOLEAN           EccSupport;
  UINT8             MemoryProfile;
  UINT32            TotalPhysicalMemorySize;
  BOOLEAN           XmpProfileEnable;
  UINT8             Ratio;
  UINT8             RefClk;
  UINT32            VddVoltage[MAX_PROFILE];
  CONTROLLER_INFO   Controller[MAX_NODE];
} MEMORY_INFO_DATA_HOB;
#define  SI_MEMORY_PLATFORM_DATA_HOB \
  { 0x6210d62f, 0x418d, 0x4999, { 0xa2, 0x45, 0x22, 0x10, 0x0a, 0x5d, 0xea, 0x44 } }
typedef struct {
  UINT8             Revision;
  UINT8             Reserved[3];
  UINT32            BootMode;
  UINT32            TsegSize;
  UINT32            TsegBase;
  UINT32            PrmrrSize;
  UINT32            PrmrrBase;
  UINT32            GttBase;
  UINT32            MmioSize;
  UINT32            PciEBaseAddress;
} MEMORY_PLATFORM_DATA;
typedef struct {
  EFI_HOB_GUID_TYPE    EfiHobGuidType;
  MEMORY_PLATFORM_DATA Data;
  UINT8                *Buffer;
} MEMORY_PLATFORM_DATA_HOB;
#define SMBIOS_CACHE_INFO_HOB_GUID \
 { 0xd805b74e, 0x1460, 0x4755, {0xbb, 0x36, 0x1e, 0x8c, 0x8a, 0xd6, 0x78, 0xd7} }

///
/// SMBIOS Cache Info HOB Structure
```

```
///
typedef struct {
  UINT16     ProcessorSocketNumber;
  UINT16     NumberOfCacheLevels;         ///< Based on Number of Cache Types L1/L2/L3
  UINT8      SocketDesignationStrIndex;   ///< String Index in the string Buffer. Example "L1-CACHE"
  UINT16     CacheConfiguration;          ///< Format defined in SMBIOS Spec v3.0 Section7.8 Table36
  UINT16     MaxCacheSize;                ///< Format defined in SMBIOS Spec v3.0 Section7.8.1
  UINT16     InstalledSize;               ///< Format defined in SMBIOS Spec v3.0 Section7.8.1
  UINT16     SupportedSramType;           ///< Format defined in SMBIOS Spec v3.0 Section7.8.2
  UINT16     CurrentSramType;             ///< Format defined in SMBIOS Spec v3.0 Section7.8.2
  UINT8      CacheSpeed;                  ///< Cache Speed in nanoseconds. 0 if speed is unknown.
  UINT8      ErrorCorrectionType;         ///< ENUM Format defined in SMBIOS Spec v3.0 Section 7.8.3
  UINT8      SystemCacheType;             ///< ENUM Format defined in SMBIOS Spec v3.0 Section 7.8.4
  UINT8      Associativity;               ///< ENUM Format defined in SMBIOS Spec v3.0 Section 7.8.5
  ///String Buffer - each string terminated by NULL "0x00"
  ///String buffer terminated by double NULL "0x0000"
} SMBIOS_CACHE_INFO;
#define SMBIOS_PROCESSOR_INFO_HOB_GUID \
  { 0xe6d73d92, 0xff56, 0x4146, {0xaf, 0xac, 0x1c, 0x18, 0x81, 0x7d, 0x68, 0x71} }

///
/// SMBIOS Processor Info HOB Structure
///
typedef struct {
  UINT16     TotalNumberOfSockets;
  UINT16     CurrentSocketNumber;
  UINT8      ProcessorType;               ///< ENUM defined in SMBIOS Spec v3.0 Section 7.5.1
  ///This info is used for both ProcessorFamily and ProcessorFamily2 fields
  ///See ENUM defined in SMBIOS Spec v3.0 Section 7.5.2
  UINT16     ProcessorFamily;
  UINT8      ProcessorManufacturerStrIndex; ///< Index of the String in the String Buffer
  UINT64     ProcessorId;                 ///< ENUM defined in SMBIOS Spec v3.0 Section 7.5.3
  UINT8      ProcessorVersionStrIndex;    ///< Index of the String in the String Buffer
  UINT8      Voltage;                     ///< Format defined in SMBIOS Spec v3.0 Section 7.5.4
  UINT16     ExternalClockInMHz;          ///< External Clock Frequency. Set to 0 if unknown.
  UINT16     CurrentSpeedInMHz;           ///< Snapshot of current processor speed during boot
  UINT8      Status;                      ///< Format defined in the SMBIOS Spec v3.0 Table 21
  UINT8      ProcessorUpgrade;            ///< ENUM defined in SMBIOS Spec v3.0 Section 7.5.5
  ///This info is used for both CoreCount & CoreCount2 fields
  /// See detailed description in SMBIOS Spec v3.0 Section 7.5.6
  UINT16     CoreCount;
  ///This info is used for both CoreEnabled & CoreEnabled2 fields
  ///See detailed description in SMBIOS Spec v3.0 Section 7.5.7
  UINT16     EnabledCoreCount;
  ///This info is used for both ThreadCount & ThreadCount2 fields
  /// See detailed description in SMBIOS Spec v3.0 Section 7.5.8
  UINT16     ThreadCount;
  UINT16     ProcessorCharacteristics;    ///< Format defined in SMBIOS Spec v3.0 Section 7.5.9
  /// String Buffer - each string terminated by NULL "0x00"
  /// String buffer terminated by double NULL "0x0000"
} SMBIOS_PROCESSOR_INFO;
#define SMBIOS_FIRMWARE_VERSION_INFO_HOB_GUID \
  { 0x798e722e, 0x15b2, 0x4e13, { 0x8a, 0xe9, 0x6b, 0xa3, 0x0f, 0xf7, 0xf1, 0x67 }}

///
/// Firmware Version Structure
///
typedef struct {
  UINT8                         MajorVersion;
  UINT8                         MinorVersion;
  UINT8                         Revision;
  UINT16                        BuildNumber;
} FIRMWARE_VERSION;

///
/// Firmware Version Information Structure
///
typedef struct {
  UINT8                         ComponentNameIndex;        ///< Offset 0   Index of Component Name
  UINT8                         VersionStringIndex;        ///< Offset 1   Index of Version String
  FIRMWARE_VERSION              Version;                   ///< Offset 2-6 Firmware version
} FIRMWARE_VERSION_INFO;

///
/// The Smbios structure header.
///
typedef struct {
  UINT8                         Type;
  UINT8                         Length;
  UINT16                        Handle;
} SMBIOS_STRUCTURE;
///
/// Firmware Version Information HOB Structure
///
typedef struct {
  SMBIOS_STRUCTURE            Header;         ///< SMBIOS structure header
  UINT8                       Count;          ///< Number of FVI entries in this structure
```

```
  INTEL_FIRMWARE_VERSION_INFO Fvi[1];         ///< FVI structure(s)
///
/// FIRMWARE_VERSION_INFO structures followed by the null terminated string buffer
///
} FIRMWARE_VERSION_INFO_HOB;
```

7.3 CHIPSETINIT INFO HOB

The FSP will report the ChipsetInit CRC through a HOB with below GUID. This information can be consumed by the bootloader to check if ChipsetInit CRC is matched between BIOS and ME. These structures are included as part of FspsUpd.h

```
#define CHIPSETINIT_INFO_HOB_GUID \
{ 0xc1392859, 0x1f65, 0x446e, { 0xb3, 0xf5, 0x84, 0x35, 0xfc, 0xc7, 0xd1, 0xc4 }}

///
/// The ChipsetInit Info structure provides the information of ME ChipsetInit CRC and BIOS ChipsetInit CRC.
///
typedef struct {
  UINT8             Revision;
  UINT8             Rsvd[3];
  UINT16            MeChipInitCrc;
  UINT16            BiosChipInitCrc;
} CHIPSET_INIT_INFO;
```

7.4 FSP_ERROR_INFO_HOB

In the case of an error occurring during the execution of the FSP, the FSP may produce this HOB which describes the error in more detail. FSP_ERROR_INFO_HOB is only used in FSP API Mode. In FSP Dispatch Mode, FSP may call ReportStatusCode() and provide a FSP_ERROR_INFO structure using the PI status code services.

```
#define FSP_ERROR_INFO_HOB_GUID \
{0x611e6a88, 0xadb7, 0x4301, { 0x93, 0xff, 0xe4, 0x73, 0x04, 0xb4, 0x3d, 0xa6 }}
typedef struct {
  EFI_HOB_GUID_TYPE     GuidHob;
  EFI_STATUS_CODE_TYPE  Type;
  EFI_STATUS_CODE_VALUE Value;
  UINT32                Instance;
  EFI_GUID              CallerId;
  EFI_GUID              ErrorType;
  UINT32                Status;
} FSP_ERROR_INFO_HOB;
```

7.5 FSP_PERFORMANCE_HOB

FSP Performance HOB is produced to record the performance data of FSP modules. FSP Performance HOB stores the module name and timestamps. The identifiers present in the FSP_PERFORMANCE enum variable can be used to identify the corresponding performance entries of the Modules. The FspMaxPerf field in the enum variable denotes the total number of entries.

```
#define FSP_PERFORMANCE_HOB_GUID  \
  { 0xaa4d127a, 0x6c1f, 0x4833, {0xa4, 0x6c, 0x07, 0xf2, 0x27, 0x14, 0x71, 0x69} }
typedef struct {
  EFI_PHYSICAL_ADDRESS    Handle;
  CHAR8                   Token[STRING_SIZE];  ///< Measured token string name.
  CHAR8                   Module[STRING_SIZE]; ///< Module string name.
  UINT64                  StartTimeStamp;      ///< Start time point.
  UINT64                  EndTimeStamp;        ///< End time point.
} FSP_PERFORMANCE_HOB;
//
// Performance Entries currently present
//
// 0. uGOP      -> Identifier = FspuGopPerf
// 1. uGOP Exit -> Identifier = FspuGopExitPerf
//
// FspMaxPerf -> Holds the total count of entries
//
typedef enum {
  FspuGopPerf,
```

```
  FspuGopExitPerf,
  FspMaxPerf
} FSP_PERFORMANCE;
```

The FSP for this platform implements the following CallerId GUIDs:

CallerId	Description
{0x1f4dc7e9, 0x26ca, 0x4336, {0x8c, 0xe3, 0x39, 0x31, 0x03, 0xb5, 0xf3, 0xd7}}	ME
{0x98230916, 0xe632, 0x49ff, {0x81, 0x81, 0x55, 0xce, 0xe5, 0x10, 0x36, 0x89}}	System Agent
{0x5a47c211, 0x642f, 0x4f92, {0x9c, 0xb3, 0x7f, 0xeb, 0x93, 0xda, 0xdd, 0xba}}	MRC

The following ErrorType GUIDs are implemented:

ErrorType	Description
{0x948585c4, 0x76a4, 0x45bb, {0xbe, 0x6c, 0x39, 0x61, 0xc3, 0xab, 0xde, 0x15}}	ME EOP failure
{0x8106a5cc, 0x30ba, 0x41cf, {0xa1, 0x78, 0x63, 0x38, 0x91, 0x11, 0xae, 0xb2}}	SA PEI GOP Init failure
{0x348cc7fe, 0x1e9a, 0x4c7a, {0x86, 0x28, 0xae, 0x48, 0x5b, 0x42, 0x10, 0xf0}}	SA PEI GOP GetMode failure
{0x5de1c071, 0x2c9c, 0x4a53, {0x80, 0x21, 0x4e, 0x80, 0xd2, 0x5d, 0x44, 0xa8}}	MRC training failure

Chapter 8

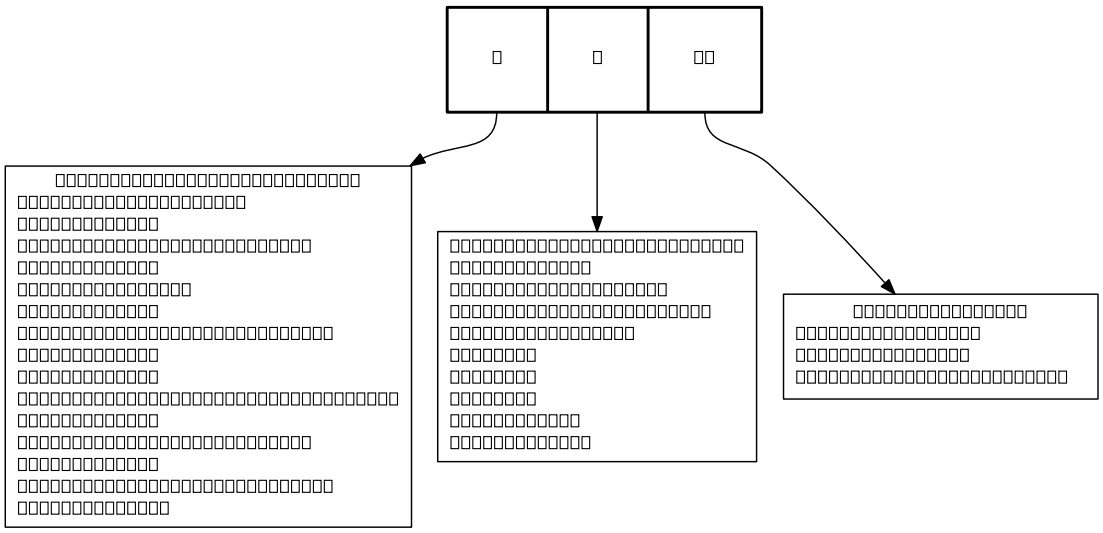
POST Codes

The FSP outputs 16-bit POST codes to indicate which API and which module is currently executing.

Bit Range	Description
Bit15 - Bit12 (X)	used to indicate the phase/api under which the code is executing
Bit11 - Bit8 (Y)	used to indicate the module
Bit7 (ZZ bit 7)	reserved for error
Bit6 - Bit0 (ZZ)	individual codes

8.1 POST Code Info

The diagram below illustrates how sub-fields are encoded in the POST codes produced by the FSP.



8.1.1 TempRamInit API Status Codes (0xFxxx)

PostCode	Module	Description
0x0000	FSP	TempRamInit API Entry (The change in upper byte is due to not enabling of the Port81 early in the boot)
0x007F	FSP	TempRamInit API Exit

8.1.2 FspMemoryInit API Status Codes (0xDxxx)

PostCode	Module	Description
0xD800	FSP	FspMemoryInit API Entry
0xD87F	FSP	FSpMemoryInit API Exit
0xDA00	SA	Pre-Mem SaInit Entry
0xDA02	SA	OverrideDev0Did Start
0xDA04	SA	OverrideDev2Did Start
0xDA06	SA	Programming SA Bars
0xDA08	SA	Install SA HOBs
0xDA0A	SA	Reporting SA PCIe code version
0xDA0C	SA	SaSvInit Start
0xDA10	SA	Initializing DMI
0xDA15	SA	Initialize TCSS PreMem
0xDA1F	SA	Initializing DMI/OPI Max PayLoad Size
0xDA20	SA	Initializing SwitchableGraphics
0xDA30	SA	Initializing SA PCIe
0xDA3F	SA	Programming PEG credit values Start
0xDA40	SA	Initializing DMI Tc/Vc mapping
0xDA42	SA	CheckOffboardPcieVga
0xDA44	SA	CheckAndInitializePegVga
0xDA50	SA	Initializing Graphics
0xDA52	SA	Initializing System Agent Overclocking
0xDA7F	SA	Pre-Mem SaInit Exit
0xDB00	PCH	Pre-Mem PchInit Entry
0xDB02	PCH	Pre-Mem Disable PCH fused controllers
0xDB15	PCH	Pre-Mem SMBUS configuration
0xDB48	PCH	Pre-Mem PchOnPolicyInstalled Entry
0xDB49	PCH	Pre-Mem Program HSIO
0xDB4A	PCH	Pre-Mem DCI configuration
0xDB4C	PCH	Pre-Mem Host DCI enabled
0xDB4D	PCH	Pre-Mem Trace Hub - Early configuration
0xDB4E	PCH	Pre-Mem Trace Hub - Device disabled
0xDB4F	PCH	Pre-Mem TraceHub - Programming MSR
0xDB50	PCH	Pre-Mem Trace Hub - Power gating configuration
0xDB51	PCH	Pre-Mem Trace Hub - Power gating Trace Hub device and locking HSWPGCR1 register
0xDB52	PCH	Pre-Mem Initialize HPET timer
0xDB55	PCH	Pre-Mem PchOnPolicyInstalled Exit
0xDB7F	PCH	Pre-Mem PchInit Exit
0xDC00	CPU	CPU Pre-Mem Entry
0xDC0F	CPU	CpuAddPreMemConfigBlocks Done
0xDC20	CPU	CpuOnPolicyInstalled Start
0xDC11	CPU	XmmInit Start
0xDC3F	CPU	TxtInit Start

PostCode	Module	Description
0xDC12	CPU	Init CPU Straps
0xDC5F	CPU	Init Overclocking
0xDC1F	CPU	CPU Pre-Mem Exit
0x**55	SA	MRC_MEM_INIT_DONE
0x**D5	SA	MRC_MEM_INIT_DONE_WITH_ERRORS
0xDD00	SA	MRC_INITIALIZATION_START
0xDD10	SA	MRC_CMD_PLOT_2D
0xDD1B	SA	MRC_FAST_BOOT_PERMITTED
0xDD1C	SA	MRC_RESTORE_NON_TRAINING
0xDD1D	SA	MRC_PRINT_INPUT_PARAMS
0xDD1E	SA	MRC_SET_OVERRIDES_PSPD
0xDD20	SA	MRC_SPD_PROCESSING
0xDD21	SA	MRC_SET_OVERRIDES
0xDD22	SA	MRC_MC_CAPABILITY
0xDD23	SA	MRC_MC_CONFIG
0xDD24	SA	MRC_MC_MEMORY_MAP
0xDD25	SA	MRC_JEDEC_INIT_LPDDR3
0xDD26	SA	MRC_RESET_SEQUENCE
0xDD27	SA	MRC_PRE_TRAINING
0xDD28	SA	MRC_EARLY_COMMAND
0xDD29	SA	MRC_SENSE_AMP_OFFSET
0xDD2A	SA	MRC_READ_MPR
0xDD2B	SA	MRC_RECEIVE_ENABLE
0xDD2C	SA	MRC_JEDEC_WRITE_LEVELING
0xDD2D	SA	MRC_LPDDR_LATENCY_SET_B
0xDD2E	SA	MRC_WRITE_TIMING_1D
0xDD2F	SA	MRC_READ_TIMING_1D
0xDD30	SA	MRC_DIMM_ODT
0xDD31	SA	MRC_EARLY_WRITE_TIMING_2D
0xDD32	SA	MRC_WRITE_DS
0xDD33	SA	MRC_WRITE_EQ
0xDD34	SA	MRC_EARLY_READ_TIMING_2D
0xDD35	SA	MRC_READ_ODT
0xDD36	SA	MRC_READ_EQ
0xDD37	SA	MRC_READ_AMP_POWER
0xDD38	SA	MRC_WRITE_TIMING_2D
0xDD39	SA	MRC_READ_TIMING_2D
0xDD3A	SA	MRC_CMD_VREF
0xDD3B	SA	MRC_WRITE_VREF_2D
0xDD3C	SA	MRC_READ_VREF_2D
0xDD3D	SA	MRC_POST_TRAINING
0xDD3E	SA	MRC_LATE_COMMAND
0xDD3F	SA	MRC_ROUND_TRIP_LAT
0xDD40	SA	MRC_TURN_AROUND
0xDD41	SA	MRC_CMP_OPT
0xDD42	SA	MRC_SAVE_MC_VALUES
0xDD43	SA	MRC_RESTORE_TRAINING
0xDD44	SA	MRC_RMT_TOOL
0xDD45	SA	MRC_WRITE_SR
0xDD46	SA	MRC_DIMM_RON
0xDD47	SA	MRC_RCVEN_TIMING_1D

PostCode	Module	Description
0xDD48	SA	MRC_MR_FILL
0xDD49	SA	MRC_PWR_MTR
0xDD4A	SA	MRC_DDR4_MAPPING
0xDD4B	SA	MRC_WRITE_VOLTAGE_1D
0xDD4C	SA	MRC_EARLY_RDMPR_TIMING_2D
0xDD4D	SA	MRC_FORCE_OLTM
0xDD50	SA	MRC_MC_ACTIVATE
0xDD51	SA	MRC_RH_PREVENTION
0xDD52	SA	MRC_GET_MRC_DATA
0xDD53	SA	Reserved
0xDD58	SA	MRC_RETRAIN_CHECK
0xDD5A	SA	MRC_SA_GV_SWITCH
0xDD5B	SA	MRC_ALIAS_CHECK
0xDD5C	SA	MRC_ECC_CLEAN_START
0xDD5D	SA	MRC_DONE
0xDD5F	SA	MRC_CPGC_MEMORY_TEST
0xDD60	SA	MRC_TXT_ALIAS_CHECK
0xDD61	SA	MRC_ENG_PERF_GAIN
0xDD68	SA	MRC_MEMORY_TEST
0xDD69	SA	MRC_FILL_RMT_STRUCTURE
0xDD70	SA	MRC_SELF_REFRESH_EXIT
0xDD71	SA	MRC_NORMAL_MODE
0xDD7D	SA	MRC_SSA_PRE_STOP_POINT
0xDD7F	SA	MRC_SSA_STOP_POINT, MRC_INITIALIZATION_END
0xDD90	SA	MRC_CMD_PLOT_2D_ERROR
0xDD9B	SA	MRC_FAST_BOOT_PERMITTED_ERROR
0xDD9C	SA	MRC_RESTORE_NON_TRAINING_ERROR
0xDD9D	SA	MRC_PRINT_INPUT_PARAMS_ERROR
0xDD9E	SA	MRC_SET_OVERRIDES_PSPD_ERROR
0xDDA0	SA	MRC_SPD_PROCESSING_ERROR
0xDDA1	SA	MRC_SET_OVERRIDES_ERROR
0xDDA2	SA	MRC_MC_CAPABILITY_ERROR
0xDDA3	SA	MRC_MC_CONFIG_ERROR
0xDDA4	SA	MRC_MC_MEMORY_MAP_ERROR
0xDDA5	SA	MRC_JEDEC_INIT_LPDDR3_ERROR
0xDDA6	SA	MRC_RESET_ERROR
0xDDA7	SA	MRC_PRE_TRAINING_ERROR
0xDDA8	SA	MRC_EARLY_COMMAND_ERROR
0xDDA9	SA	MRC_SENSE_AMP_OFFSET_ERROR
0xDDAA	SA	MRC_READ_MPR_ERROR
0xDDAB	SA	MRC_RECEIVE_ENABLE_ERROR
0xDDAC	SA	MRC_JEDEC_WRITE_LEVELING_ERROR
0xDDAD	SA	MRC_LPDDR_LATENCY_SET_B_ERROR
0xDDAE	SA	MRC_WRITE_TIMING_1D_ERROR
0xDDAF	SA	MRC_READ_TIMING_1D_ERROR
0xDDB0	SA	MRC_DIMM_ODT_ERROR
0xDDB1	SA	MRC_EARLY_WRITE_TIMING_ERROR
0xDDB2	SA	MRC_WRITE_DS_ERROR
0xDDB3	SA	MRC_WRITE_EQ_ERROR
0xDDB4	SA	MRC_EARLY_READ_TIMING_ERROR
0xDDB5	SA	MRC_READ_ODT_ERROR

PostCode	Module	Description
0xDDB6	SA	MRC_READ_EQ_ERROR
0xDDB7	SA	MRC_READ_AMP_POWER_ERROR
0xDDB8	SA	MRC_WRITE_TIMING_2D_ERROR
0xDDB9	SA	MRC_READ_TIMING_2D_ERROR
0xDDBA	SA	MRC_CMD_VREF_ERROR
0xDDBB	SA	MRC_WRITE_VREF_2D_ERROR
0xDDBC	SA	MRC_READ_VREF_2D_ERROR
0xDDBD	SA	MRC_POST_TRAINING_ERROR
0xDDBE	SA	MRC_LATE_COMMAND_ERROR
0xDDBF	SA	MRC_ROUND_TRIP_LAT_ERROR
0xDDC0	SA	MRC_TURN_AROUND_ERROR
0xDDC1	SA	MRC_CMP_OPT_ERROR
0xDDC2	SA	MRC_SAVE_MC_VALUES_ERROR
0xDDC3	SA	MRC_RESTORE_TRAINING_ERROR
0xDDC4	SA	MRC_RMT_TOOL_ERROR
0xDDC5	SA	MRC_WRITE_SR_ERROR
0xDDC6	SA	MRC_DIMM_RON_ERROR
0xDDC7	SA	MRC_RCVEN_TIMING_1D_ERROR
0xDDC8	SA	MRC_MR_FILL_ERROR
0xDDC9	SA	MRC_PWR_MTR_ERROR
0xDDCA	SA	MRC_DDR4_MAPPING_ERROR
0xDDCB	SA	MRC_WRITE_VOLTAGE_1D_ERROR
0xDDCC	SA	MRC_EARLY_RDMPR_TIMING_2D_ERROR
0xDDCD	SA	MRC_FORCE_OLTM_ERROR
0xDDD0	SA	MRC_MC_ACTIVATE_ERROR
0xDDD1	SA	MRC_RH_PREVENTION_ERROR
0xDDD2	SA	MRC_GET_MRC_DATA_ERROR
0xDDD3	SA	Reserved
0xDDD8	SA	MRC_RETRAIN_CHECK_ERROR
0xDDDA	SA	MRC_SA_GV_SWITCH_ERROR
0xDDDB	SA	MRC_ALIAS_CHECK_ERROR
0xDDDC	SA	MRC_ECC_CLEAN_ERROR
0xDDDD	SA	MRC_DONE_WITH_ERROR
0xDDDF	SA	MRC_CPGC_MEMORY_TEST_ERROR
0xDDE0	SA	MRC_TXT_ALIAS_CHECK_ERROR
0xDDE1	SA	MRC_ENG_PERF_GAIN_ERROR
0xDDE8	SA	MRC_MEMORY_TEST_ERROR
0xDDE9	SA	MRC_FILL_RMT_STRUCTURE_ERROR
0xDDF0	SA	MRC_SELF_REFRESH_EXIT_ERROR
0xDDF1	SA	MRC_MRC_NORMAL_MODE_ERROR
0xDDFD	SA	MRC_SSA_PRE_STOP_POINT_ERROR
0xDDFE	SA	MRC_NO_MEMORY_DETECTED

8.1.3 TempRamExit API Status Codes (0xBxxx)

PostCode	Module	Description
0xB800	FSP	TempRamExit API Entry
0xB87F	FSP	TempRamExit API Exit

8.1.4 FspSiliconInit API Status Codes (0x9xxx)

PostCode	Module	Description
0x9800	FSP	FspSiliconInit API Entry
0x987F	FSP	FspSiliconInit API Exit
0x9A00	SA	PostMem SaInit Entry
0x9A01	SA	DeviceConfigure Start
0x9A02	SA	UpdateSaHobPostMem Start
0x9A03	SA	Initializing Pei Display
0x9A04	SA	PeiGraphicsNotifyCallback Entry
0x9A05	SA	CallPpiAndFillFrameBuffer
0x9A06	SA	GraphicsPpiInit
0x9A07	SA	GraphicsPpiGetMode
0x9A08	SA	FillFrameBufferAndShowLogo
0x9A0F	SA	PeiGraphicsNotifyCallback Exit
0x9A14	SA	Initializing SA IPU device
0x9A20	SA	Initializing PciExpressInitPostMem
0x9A22	SA	Initializing ConfigureNorthIntelTraceHub
0xDA51	SA	Initializing Vtd
0x9A31	SA	Initializing TCSS
0x9A33	SA	Initializing Pavp
0x9A34	SA	PeiInstallSmmAccessPpi Start
0x9A36	SA	EdramWa Start
0x9A4F	SA	Post-Mem SaInit Exit
0x9A50	SA	SaSecurityLock Start
0x9A5F	SA	SaSecurityLock End
0x9A60	SA	SaSResetComplete Entry
0x9A61	SA	Set BIOS_RESET_CPL to indicate all configurations complete
0x9A62	SA	SaSvInit2 Start
0x9A63	SA	IGpuPmInit Start
0x9A64	SA	SaPciPrint Start
0x9A6F	SA	SaSResetComplete Exit
0x9A70	SA	SaS3ResumeAtEndOfPei Callback Entry
0x9A7F	SA	SaS3ResumeAtEndOfPei Callback Exit
0x9B00	PCH	Post-Mem PchInit Entry
0x9B04	PCH	Post-Mem Tune the USB 2.0 high-speed signals quality
0x9B05	PCH	Post-Mem Tune the USB 3.0 signals quality
0x9B06	PCH	Post-Mem Performs configuration of PCH xHCI SSIC
0x9B07	PCH	Post-Mem Configure PCH xHCI after init
0x9B08	PCH	Post-Mem Configures PCH USB device (xDCI)
0x9B0A	PCH	Post-Mem DMI/OP-DMI configuration
0x9B0B	PCH	Post-Mem Initialize P2SB controller
0x9B0C	PCH	Post-Mem IOAPIC initialization
0x9B0D	PCH	Post-Mem PCH devices interrupt configuration
0x9B0E	PCH	Post-Mem HD Audio initization
0x9B0F	PCH	Post-Mem HD Audio Codec enumeration
0x9B10	PCH	Post-Mem HD Audio Codec not detected
0x9B13	PCH	Post-Mem SCS initization
0x9B14	PCH	Post-Mem ISH initization
0x9B15	PCH	Post-Mem Configure SMBUS power management

PostCode	Module	Description
0x9B16	PCH	Post-Mem Reserved
0x9B17	PCH	Post-Mem Performing global reset
0x9B18	PCH	Post-Mem Reserved
0x9B19	PCH	Post-Mem Reserved
0x9B40	PCH	Post-Mem OnEndOfPEI Entry
0x9B41	PCH	Post-Mem Initialize Thermal controller
0x9B42	PCH	Post-Mem Configure Memory Throttling
0x9B47	PCH	Post-Mem OnEndOfPEI Exit
0x9B4D	PCH	Post-Mem Trace Hub - Memory configuration
0x9B4E	PCH	Post-Mem Trace Hub - MSC0 configured
0x9B4F	PCH	Post-Mem Trace Hub - MSC1 configured
0x9B7F	PCH	Post-Mem PchInit Exit
0x9C00	CPU	CPU Post-Mem Entry
0x9C09	CPU	CpuAddConfigBlocks Done
0xDC13	CPU	SetCpuStrapAndEarlyPowerOnConfig Start
0xDC14	CPU	SetCpuStrapAndEarlyPowerOnConfig Reset
0xDC15	CPU	SetCpuStrapAndEarlyPowerOnConfig Done
0x9C10	CPU	CpuInit Start
0x9C17	CPU	CollectProcessorFeature Start
0x9C12	CPU	ProgramProcessorFeature Start
0x9C13	CPU	ProgramProcessorFeature Done
0x9C20	CPU	CpuInitPreResetCpl Start
0x9C21	CPU	ProcessorsPrefetcherInitialization Start
0x9C31	CPU	InitRatl Start
0x9C40	CPU	ConfigureSvidVrs Start
0x9C24	CPU	ConfigurePidSettings Start
0x9C33	CPU	SetBootFrequency Start
0xDC20	CPU	CpuOcInitPreMem Start
0x9C27	CPU	CpuOcInit Reset
0x9C20	CPU	BiosGuardInit Start
0x9C21	CPU	BiosGuardInit Reset
0x9C3F	CPU	CpuInitPreResetCpl Done
0x9C62	CPU	ApSafePostMicrocodePatchInit Start
0x9C43	CPU	InitializeCpuDataHob Start
0x9C44	CPU	InitializeCpuDataHob Done
0x9C1F	CPU	CpuInit Done
0x9C50	CPU	S3InitializeCpu Start
0x9C55	CPU	MpRendezvousProcedure Start
0x9C56	CPU	MpRendezvousProcedure Done
0x9C69	CPU	S3InitializeCpu Done
0x9C70	CPU	CpuPowerMgmtInit Start
0x9C71	CPU	InitPpm
0x9C7F	CPU	CPU Post-Mem Exit
0x9C60	CPU	ReloadMicrocodePatch Start
0x9C61	CPU	ReloadMicrocodePatch Done
0x9C6F	CPU	ApSafePostMicrocodePatchInit Done

8.1.5 NotifyPhase API Status Codes (0x6xxx)

PostCode	Module	Description
0x6800	FSP	NotifyPhase API Entry
0x687F	FSP	NotifyPhase API Exit

Chapter 9

Todo List

Page Config Blocks

This list is updated for PantherLake

This list is updated for PantherLake

Chapter 10

Class Index

10.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

_CONFIG_BLOCK
Config Block . . . 67
_CONFIG_BLOCK_HEADER
Config Block Header . . . 68
_CONFIG_BLOCK_TABLE_STRUCT
Config Block Table Header . . . 69
_DXE_ITBT_CONFIG
ITBT DXE configuration
Revision 1: . . . 70
_DXE_ITBT_GENERIC_CONFIG
ITBT Generic Data Structure to be used cross to RP and controller in DXE phase . . . 71
_EFI_MM_RESERVED_MMRAM_REGION
Structure describing a MMRAM region which cannot be used for the MMRAM heap . . . 71
_EFI_PEI_MP_SERVICES_PPI
This PPI is installed by some platform or chipset-specific PEIM that abstracts handling multiprocessor support . . . 73
_ITBT_GENERIC_CONFIG
ITBT Controller Data Structure to be used cross to RP and controller to be shared by CONFIG_BLOCK and HOB . . . 73
_ITBT_ROOTPORT_CONFIG
ITBT RootPort Data Structure . . . 74
_LIST_ENTRY
_LIST_ENTRY structure definition . . . 75
_PEI_ITBT_CONFIG
ITBT PEI configuration
Revision 1: . . . 75
_PEI_PREMEM_SI_DEFAULT_POLICY_INIT_PPI
This PPI provides function to install default silicon policy . . . 76
_PEI_SI_DEFAULT_POLICY_INIT_PPI
This PPI provides function to install default silicon policy . . . 77
_SI_POLICY_STRUCT
SI Policy PPI
All SI config block change history will be listed here

_SI_PREMEM_POLICY_STRUCT
SI Policy PPI in Pre-Mem
All SI config block change history will be listed here

79
ADR_CONFIG
ADR Configuration **Revision 1**: - Initial version . . . 80
ADR_GLOBAL_RESET_ENABLE
ADR Source Enable . . . 81
AMT_DXE_CONFIG
AMT Dxe Configuration Structure . . . 82
AMT_PEI_CONFIG
AMT Pei Configuration Structure . . . 83
BCLK_CONFIG
BCLK configuration . . . 86
BIOS_GUARD_CONFIG
BIOS Guard Configuration Structure . . . 87
CNVI_CONFIG
The CNVI_CONFIG block describes the expected configuration of the CNVi IP . . . 89
CNVI_PIN_MUX
CNVi signals pin muxing settings . . . 91
CNVI_PREMEM_CONFIG
The CNVI_PREMEM_CONFIG block describes the expected configuration of the CNVi IP . . . 92
CPU_INIT_PREMEM_CONFIG
CPU Config Library PreMemory Configuration Structure . . . 93
CPU_POWER_DELIVERY_CONFIG
CPU Power Delivery Configuration Structure . . . 101
CPU_POWER_MGMT_CUSTOM_CONFIG
CPU Power Management Custom Configuration Structure . . . 107
CPU_SECURITY_PREMEM_CONFIG
CPU Security PreMemory Configuration Structure . . . 108
CPU_TEST_CONFIG
CPU Test Configuration Structure . . . 111
DDI_CONFIGURATION
This structure configures the Native GPIOs for DDI port per VBT settings . . . 113
DMI_EQ_PARAM
Represent lane specific Dmi Gen3 equalization parameters . . . 114
DMI_HW_WIDTH_CONTROL
This structure allows to customize DMI HW Autonomous Width Control for Thermal and Mechanical spec design . . . 115
DMI_LANE_CONFIG
The PCH_DMI_LANE_CONFIG block describes specific lane configuration . . . 116
EFI_HOB_CPU
Describes processor information, such as address space and I/O space capabilities . . . 116
EFI_HOB_FIRMWARE_VOLUME
Details the location of firmware volumes that contain firmware files . . . 117
EFI_HOB_FIRMWARE_VOLUME2
Details the location of a firmware volume that was extracted from a file within another firmware volume . . . 118
EFI_HOB_FIRMWARE_VOLUME3
Details the location of a firmware volume that was extracted from a file within another firmware volume . . . 120
EFI_HOB_GENERIC_HEADER
Describes the format and size of the data inside the HOB . . . 122
EFI_HOB_GUID_TYPE
Allows writers of executable content in the HOB producer phase to maintain and manage HOBs with specific GUID . . . 122

EFI_HOB_HANDOFF_INFO_TABLE
Contains general state information used by the HOB producer phase . . . 123
EFI_HOB_MEMORY_ALLOCATION
Describes all memory ranges used during the HOB producer phase that exist outside the HOB list 125
EFI_HOB_MEMORY_ALLOCATION_BSP_STORE
Defines the location of the boot-strap processor (BSP) BSPStore ("Backing Store Pointer Store") 126
EFI_HOB_MEMORY_ALLOCATION_HEADER
EFI_HOB_MEMORY_ALLOCATION_HEADER describes the various attributes of the logical memory allocation . . . 127
EFI_HOB_MEMORY_ALLOCATION_MODULE
Defines the location and entry point of the HOB consumer phase . . . 129
EFI_HOB_MEMORY_ALLOCATION_STACK
Describes the memory stack that is produced by the HOB producer phase and upon which all post-memory-installed executable content in the HOB producer phase is executing . . . 130
EFI_HOB_MEMORY_POOL
Describes pool memory allocations . . . 131
EFI_HOB_RESOURCE_DESCRIPTOR
Describes the resource properties of all fixed, nonrelocatable resource ranges found on the processor host bus during the HOB producer phase . . . 132
EFI_HOB_UEFI_CAPSULE
Each UEFI capsule HOB details the location of a UEFI capsule . . . 134
EFI_IP_ADDRESS
16-byte buffer aligned on a 4-byte boundary . . . 135
EFI_MAC_ADDRESS
32-byte buffer containing a network Media Access Control address . . . 136
EFI_MMRAM_DESCRIPTOR
Structure describing a MMRAM region and its accessibility attributes . . . 136
EFI_PEI_HOB_POINTERS
Union of all the possible HOB Types . . . 138
EFI_TIME
EFI Time Abstraction: Year: 1900 - 9999 Month: 1 - 12 Day: 1 - 31 Hour: 0 - 23 Minute: 0 - 59 Second: 0 - 59 Nanosecond: 0 - 999,999,999 TimeZone: -1440 to 1440 or 2047 . . . 138
FIVR_EXT_RAIL_CONFIG
Structure for V1p05/Vnn VR rail configuration . . . 139
FIVR_VCCIN_AUX_CONFIG
Structure for VCCIN_AUX voltage rail configuration . . . 140
FSP_ERROR_INFO
FSP Error Information Block . . . 142
FSPM_ARCH_CONFIG_PPI
This PPI provides FSP-M Arch Config PPI . . . 143
FSPV_POSTMEM_CONFIG
The FSPV_POSTMEM_CONFIG block describes policies related to FSP Validation . . . 144
FSPV_PREMEM_CONFIG
The FSPV_PREMEM_CONFIG block describes policies related to FSP Validation . . . 145
FUSA_CONFIG
The FUSA_CONFIG block describes the expected configuration of Functional Safety . . . 146
GBE_CONFIG
PCH intergrated GBE controller configuration settings . . . 149
GPIO_DXE_CONFIG
This structure contains the DXE policies which are related to GPIO device . . . 151
GUID
128 bit buffer containing a unique identifier value . . . 152
HDA_LINK_DMIC
HD Audio DMIC Interface Policies . . . 153
HDA_LINK_HDA
HD Audio Link Policies . . . 153
HDA_LINK_SNDW
HD Audio SNDW Interface Policies . . . 154

HDA_LINK_SSP
HD Audio SSP Interface Policies . . . 154
HDA_MIC_PRIVACY
HD Audio Link Policies . . . 155
HDA_SNDW_MULTILANE
HD Audio SNDW Multilane Policies . . . 156
HDA_VERB_TABLE_HEADER
Azalia verb table header Every verb table should contain this defined header and followed by azalia verb commands . . . 156
HDAUDIO_CONFIG
This structure contains the policies which are related to HD Audio device (cAVS) . . . 157
HDAUDIO_DXE_CONFIG
This structure contains the DXE policies which are related to HD Audio device (cAVS) . . . 159
HDAUDIO_PREMEM_CONFIG
This structure contains the premem policies which are related to HD Audio device (cAVS) . . . 160
HOST_BRIDGE_PEI_CONFIG
This configuration block describes HostBridge settings in Post-Mem . . . 164
HOST_BRIDGE_PREMEM_CONFIG
This configuration block describes HostBridge settings in PreMem . . . 165
HSIO_CONFIG
The HSIO_CONFIG block provides HSIO message related settings . . . 167
HSIO_PARAMETERS
This structure describes USB3 Port N configuration parameters . . . 168
HYBRID_GRAPHICS_CONFIG
This Configuration block configures CPU PCI Express 0/1/2 RTD3 GPIOs & Root Port . . . 170
I2C_PIN_MUX
I2C signals pin muxing settings . . . 171
I3C_MUX_GPIO_CONFIG
I3C GPIO settings . . . 172
IAX_PEI_CONFIG
IAX PEI configuration **Revision 1**: . . . 172
IEH_CONFIG
The IEH_CONFIG block describes the expected configuration of the PCH Integrated Error Handler . . . 174
IGPU_DXE_CONFIG
This configuration block is to configure IGD related variables used in DXE . . . 175
IGPU_PEI_CONFIG
This configuration block is to configure IGD related variables used in PostMem PEI . . . 176
IGPU_PEI_PREMEM_CONFIG
This Configuration block is to configure GT related PreMem data/variables . . . 177
IOM_AUX_ORI_GPIO_PAD_CONFIG
The IOM_AUX_ORI_GPIO_PAD_CONFIG describes IOM TypeC port map GPIO pin . . . 180
IPU_PREMEM_CONFIG
IPU PreMem configuration
Revision 1: . . . 181
IPv4_ADDRESS
4-byte buffer . . . 182
IPv6_ADDRESS
16-byte buffer . . . 183
ISH_CONFIG
The ISH_CONFIG block describes Integrated Sensor Hub device . . . 183
ISH_GP
Struct contains GPIO pins assigned and signal settings of GP . . . 184
ISH_GPIO_CONFIG
ISH GPIO settings . . . 185
ISH_I2C
Struct contains GPIO pins assigned and signal settings of I2C . . . 186

ISH_I2C_PIN_CONFIG
I2C signals settings . . . 187
ISH_PREMEM_CONFIG
Premem Policy for Integrated Sensor Hub device . . . 188
ISH_SPI
Struct contains GPIO pins assigned and signal settings of SPI . . . 189
ISH_SPI_PIN_CONFIG
SPI signals settings . . . 190
ISH_UART
Struct contains GPIO pins assigned and signal settings of UART . . . 191
ISH_UART_PIN_CONFIG
UART signals settings . . . 192
LPSS_I2C_CONFIG
The LPSS_I2C_CONFIG block provides the configurations to set the Serial IO controllers . . . 193
LPSS_I2C_CONTROLLER_CONFIG
Serial IO I2C Controller Configuration . . . 194
LPSS_I2C_PIN_MUX
I2C signals pin muxing settings . . . 195
LPSS_I3C_CONFIG
The LPSS_I3C_CONFIG block provides the configurations to set the Serial IO controllers . . . 195
LPSS_I3C_DEVICE_CONFIG
The LPSS_I3C_DEVICE_CONFIG provides the configurations for Serial IO I3C controller . . . 196
LPSS_SPI_CONFIG
The LPSS_SPI_CONFIG block provides the configurations to set the LPSS SPI controllers **Revision 1:** . . . 197
LPSS_SPI_DEVICE_CONFIG
The LPSS_SPI_CONFIG provides the configurations to set the Serial IO SPI controller . . . 198
LPSS_SPI_PIN_MUX
SPI signals pin muxing settings . . . 199
LPSS_UART_ATTRIBUTES
UART Settings . . . 199
LPSS_UART_CONFIG
The LPSS_UART_CONFIG block provides the configurations to set the LPSS UART controllers **Revision 1:** . . . 200
LPSS_UART_DEVICE_CONFIG
Lpss UART Controller Configuration . . . 201
LPSS_UART_PIN_MUX
UART signals pin muxing settings . . . 202
ME_DXE_CONFIG
ME Dxe Configuration Structure . . . 202
ME_PEI_CONFIG
ME Pei Post-Memory Configuration Structure . . . 204
ME_PEI_PREMEM_CONFIG
ME Pei Pre-Memory Configuration Structure . . . 206
MEMORY_CONFIG_NO_CRC
Memory Configuration The contents of this structure are not CRC'd by the MRC for option change detection . . . 208
MEMORY_CONFIGURATION
Memory Configuration The contents of this structure are CRC'd by the MRC for option change detection . . . 212
MEMORY_DQDQS_MAPPING
DqDqs Mapping . . . 213
MEMORY_RCOMP
Rcomp Policies . . . 213
MEMORY_SUB_SYSTEM_CONFIG
This Configuration block configures Memory SubSystem **Revision 1**: . . . 214
MRC_ERROR_KEY_VALUE_ENTRY
MRC Error Key Value Entry structure maps an MRC error code to an error message string . . . 215

MUX_GPIO_CONFIG
I3C GPIO settings . . . 216
MUX_GPIO_PARAM
SD GPIO settings . . . 216
NPU_PEI_CONFIG
NPU PEI configuration
Revision 1: . . . 218
OVERCLOCKING_PREMEM_CONFIG
Overclocking Configuration Structure . . . 219
PCH_DCI_PREMEM_CONFIG
The PCH_DCI_PREMEM_CONFIG block describes policies related to Direct Connection Interface (DCI) . . . 234
PCH_DEVICE_INTERRUPT_CONFIG
The PCH_DEVICE_INTERRUPT_CONFIG block describes interrupt pin, IRQ and interrupt mode for PCH device . . . 236
PCH_DMI_CONFIG
The PCH_DMI_CONFIG block describes the expected configuration of the PCH for DMI . . . 237
PCH_ESPI_CONFIG
This structure contains the policies which are related to ESPI . . . 239
PCH_FIVR_CONFIG
The PCH_FIVR_CONFIG block describes FIVR settings . . . 241
PCH_FLASH_PROTECTION_CONFIG
The PCH provides a method for blocking writes and reads to specific ranges in the SPI flash when the Protected Ranges are enabled . . . 242
PCH_GENERAL_CONFIG
PCH General Configuration **Revision 2**: - Add VtdEnabled field . . . 243
PCH_GENERAL_PREMEM_CONFIG
PCH General Pre-Memory Configuration **Revision 1**: - Initial version . . . 245
PCH_HSIO_PCIE_LANE_CONFIG
The PCH_HSIO_PCIE_LANE_CONFIG describes HSIO settings for PCIe lane . . . 248
PCH_HSIO_PCIE_PREMEM_CONFIG
The PCH_HSIO_PCIE_CONFIG block describes the configuration of the HSIO for PCIe lanes . 249
PCH_HSIO_SATA_PORT_LANE
The PCH_HSIO_SATA_PORT_LANE describes HSIO settings for SATA Port lane . . . 249
PCH_HSIO_SATA_PREMEM_CONFIG
The PCH_HSIO_SATA_CONFIG block describes the HSIO configuration of the SATA controller 251
PCH_INTERRUPT_CONFIG
The PCH_INTERRUPT_CONFIG block describes interrupt settings for PCH . . . 252
PCH_IOAPIC_CONFIG
The PCH_IOAPIC_CONFIG block describes the expected configuration of the PCH IO APIC, it's optional and PCH code would ignore it if the BdfValid bit is not TRUE . . . 253
PCH_LOCK_DOWN_CONFIG
The PCH_LOCK_DOWN_CONFIG block describes the expected configuration of the PCH for security requirement . . . 255
PCH_LPC_PREMEM_CONFIG
This structure contains the policies which are related to LPC . . . 258
PCH_P2SB_CONFIG
This structure contains the policies which are related to P2SB device . . . 260
PCH_PCIE_CLOCK
PCH_PCIE_CLOCK describes PCIe source clock generated by PCH . . . 261
PCH_PCIE_CONFIG
The PCH_PCIE_CONFIG block describes the expected configuration of the PCH PCI Express controllers **Revision 1**: . . . 262
PCH_PCIE_ROOT_PORT_CONFIG
The PCH_PCI_EXPRESS_ROOT_PORT_CONFIG describe the feature and capability of each PCH PCIe root port . . . 263

PCH_PCIE_RP_PREMEM_CONFIG
The PCH_PCIE_RP_PREMEM_CONFIG block describes early configuration of the PCH PCI Express controllers **Revision 1**: . . . 265
PCH_PM_CONFIG
The PCH_PM_CONFIG block describes expected miscellaneous power management settings 266
PCH_SATA_PORT_CONFIG
This structure configures the features, property, and capability for each SATA port . . . 273
PCH_SMBUS_PREMEM_CONFIG
The SMBUS_CONFIG block lists the reserved addresses for non-ARP capable devices in the platform . . . 275
PCH_TRACE_HUB_PREMEM_CONFIG
PCH Trace Hub PreMem Configuration Contains Trace Hub settings for PCH side tracing **Revision 1**: - Initial version . . . 277
PCH_WAKE_CONFIG
This structure allows to customize PCH wake up capability from S5 or DeepSx by WOL, LAN, PCIE wake events . . . 278
PCH_WDT_PREMEM_CONFIG
This policy clears status bits and disable watchdog, then lock the WDT registers . . . 279
PCIE_COMMON_CONFIG
PCIe Common Config . . . 280
PCIE_DEVICE_OVERRIDE
PCIe device table entry entry . . . 282
PCIE_EQ_PARAM
Represent lane specific PCIe Gen3 equalization parameters . . . 285
PCIE_LINK_EQ_PLATFORM_SETTINGS
PCIe Link EQ Platform Settings . . . 285
PCIE_RP_DXE_CONFIG
The PCIE_RP_DXE_CONFIG block describes the expected configuration of the PCH PCI Express controllers in DXE phase . . . 288
PEI_DMI_CONFIG
The PEI_DMI_CONFIG block describes the expected configuration of the DMI . . . 290
PMC_GLOBAL_RESET_MASK
Description of Global Reset Trigger/Event Mask register . . . 293
PMC_INTERFACE_CONFIG
The PMC_INTERFACE_CONFIG block describes interaction between BIOS and PMC . . . 294
PMC_LPM_S0IX_SUB_STATE_EN
Low Power Mode Enable config . . . 294
PPM_CUSTOM_RATIO_TABLE
This structure is used to describe the custom processor ratio table desired by the platform . . . 295
PROTECTED_RANGE
Protected Flash Range . . . 296
PSF_CONFIG
PSF Configuration **Revision 1**: - Initial version . . . 297
RTC_CONFIG
The RTC_CONFIG block describes the expected configuration of RTC configuration . . . 299
SA_ADDRESS_DECODE
SA memory address decode . . . 300
SATA_CONFIG
The SATA_CONFIG block describes the expected configuration of the SATA controllers . . . 301
SATA_THERMAL_THROTTLING
This structure lists PCH supported SATA thermal throttling register setting for customization . . . 304
SCS_SD_DLL
This structre holds the DLL configuration register values that will be programmed by RC if EnableCustomDlls field is set to TRUE . . . 305
SERIAL_IO_CONFIG
The SERIAL_IO_CONFIG block provides the configurations to set the Serial IO controllers . . . 305
SERIAL_IO_I2C_CONFIG
Serial IO I2C Controller Configuration . . . 306

SERIAL_IO_I3C_CONFIG
The SERIAL_IO_I3C_CONFIG provides the configurations for Serial IO I3C controller 307
SERIAL_IO_SPI_CONFIG
The SERIAL_IO_SPI_CONFIG provides the configurations to set the Serial IO SPI controller . 308
SERIAL_IO_UART_ATTRIBUTES
UART Settings 309
SERIAL_IO_UART_CONFIG
Serial IO UART Controller Configuration 310
SI_CONFIG
The Silicon Policy allows the platform code to publish a set of configuration information that the RC drivers will use to configure the silicon hardware 311
SI_PREMEM_CONFIG
The Silicon PreMem Policy allows the platform code to publish a set of configuration information that the RC drivers will use to configure the silicon hardware 314
SPD_DATA_BUFFER
SPD Data Buffer 315
SPD_OFFSET_TABLE
SPD Offset Table 316
SPI_CONFIG
Basic configuration for option features of PCH SPI Flash controller 317
SPI_PIN_MUX
SPI signals pin muxing settings 318
SVID_SID_VALUE
Subsystem Vendor ID / Subsystem ID 319
SWIZZLE_CONFIG
This config block describes how to configure Interrupt Swizzling functionality in EAH 319
TCSS_IOM_PEI_CONFIG
The TCSS_IOM_PEI_CONFIG block describes IOM settings for TCSS 321
TCSS_IOM_PEI_PERMEM_CONFIG
The TCSS_IOM_PEI_PERMEM_CONFIG block describes IOM settings for TCSS 322
TCSS_MISC_PEI_CONFIG
The TCSS_MISC_PEI_CONFIG block describes MISC settings for TCSS 322
TCSS_PCIE_PEI_POLICY
TCSS_PCIE_PEI_POLICY describes PCIe port settings for TCSS 323
TCSS_PCIE_PORT_POLICY
The TCSS_PCIE_PORT_POLICY block describes PCIe settings for TCSS 324
TCSS_PEI_CONFIG
The TCSS_PEI_CONFIG block describes TCSS settings for SA 326
TCSS_PEI_PREMEM_CONFIG
This configuration block describes TCSS settings 327
TCSS_USBTC_PEI_PERMEM_CONFIG
The TCSS_USBTC_PEI_PERMEM_CONFIG block describes IOM settings for TCSS 328
TELEMETRY_PEI_CONFIG
This configuration block describes Telemetry settings in PostMem 328
TELEMETRY_PEI_PREMEM_CONFIG
This configuration block describes Telemetry settings in PreMem 329
THC_CONFIG
THC_CONFIG block provides the configurations forTouch Host Controllers 330
THC_DATA_SSID_CONFIG_HOB
THC SSID Data Hob 332
THC_HID_OVER_I2C
THC Hid Over I2C mode related settings 333
THC_HID_OVER_SPI
THC Hid Over SPI mode related settings 334
THC_PORT
Port Configuration structure required for each Port that THC might use 335
THC_PORT_SUBSYTEM_ID
Structure to hold Sub System ID for port 336

THC_RESET
THC Reset Pad related settings . . . 337
THC_WOT_PIN_CONFIG
Structure to hold Wake On Touch pin configuration for GPIO resource in ACPI . . . 337
THERMAL_CONFIG
The THERMAL_CONFIG block describes the expected configuration of the Thermal IP block . 338
THERMAL_THROTTLE_LEVELS
This structure lists PCH supported throttling register setting for custimization . . . 339
TRACE_HUB_CONFIG
TRACE_HUB_CONFIG block describes TraceHub settings . . . 341
TRACE_HUB_PREMEM_CONFIG
Trace Hub PreMem Configuration Contains Trace Hub settings **Revision 1**: - Initial version . . 342
TSN_CONFIG
TSN Config settings . . . 343
TXT_PREMEM_CONFIG
CPU TXT PreMemory Configuration Structure . . . 344
UART_PIN_MUX
UART signals pin muxing settings . . . 346
USB2_PHY_CONFIG
This structure holds info on how to tune electrical parameters of USB2 ports based on board layout . . . 346
USB2_PHY_PARAMETERS
This structure configures per USB2 AFE settings . . . 347
USB2_PORT_CONFIG
This structure configures per USB2.0 port settings like enabling and overcurrent protection . . 348
USB3_HSIO_CONFIG
Structure for holding USB3 tuning parameters . . . 349
USB3_PORT_CONFIG
This structure configures per USB3.x port settings like enabling and overcurrent protection . . 350
USB_CONFIG
This member describes the expected configuration of the USB controller, Platform modules may need to refer Setup options, schematic, BIOS specification to update this field . . . 351
VMD_PEI_CONFIG
This configuration block is to configure VMD related variables used in PostMem PEI . . . 354
XDCI_CONFIG
The XDCI_CONFIG block describes the configurations of the xDCI Usb Device controller . . . 356

Chapter 11

File Index

11.1 File List

Here is a list of all documented files with brief descriptions:

AdrConfig.h
ADR policy . . . 357
AmtConfig.h
AMT Config Block for PEI/DXE phase . . . 358
Base.h
Root include file for Mde Package Base type modules . . . 359
BiosGuardConfig.h
CPU BIOS Guard Config Block . . . 381
CnviConfig.h
CNVi policy . . . 383
ConfigBlock.h
Header file for Config Block Lib implementation . . . 384
ConfigBlockLib.h
Header file for Config Block Lib implementation . . . 386
CpuInitConfig.h
CPU Initialization Config Block . . . 388
CpuInitPreMemConfig.h
CPU Initialization PreMemory Config Block . . . 389
CpuPowerDeliveryConfig.h
CPU Power Delivery Config Block . . . 390
CpuPowerMgmtBasicConfig.h
CPU Power Management Basic Config Block . . . 391
CpuPowerMgmtCustomConfig.h
CPU Power Managment Custom Config Block . . . 392
CpuPowerMgmtTestConfig.h
CPU Power Management Test Config Block . . . 393
CpuPowerMgmtVrConfig.h
CPU Power Management VR Config Block . . . 395
CpuSecurityPreMemConfig.h
CPU Security PreMemory Config Block . . . 396
CpuTestConfig.h
CPU Test Config Block . . . 397
DciConfig.h
Dci policy . . . 398
DmiConfig.h
Config block definition for PCH DMI . . . 399

DxeITbtConfig.h
Headr file for iTBT DXE Policy . . . 399
EspiConfig.h
Espi policy . . . 401
FiaConfig.h
PCH FIA MUX config block containing user configuration options . . . 402
FivrConfig.h
PCH FIVR policy . . . 402
FlashProtectionConfig.h
FlashProtection policy . . . 403
FspErrorInfo.h
FSP Error Information to describe errors inside FSP that bootloader may take some actions to handle those error scenarios . . . 404
FspFixedPcds.h
This file lists all FixedAtBuild PCDs referenced in FSP integration guide . . . 405
FspmArchConfigPpi.h
Header file for FSP-M Arch Config PPI . . . 406
FspPerformance.h
Library for Fsp Performance Lib Config file . . . 406
FspVConfig.h
FspV Config block . . . 407
FusaConfig.h
Functional Safety policy . . . 408
GbeConfig.h
Gigabit Ethernet policy . . . 409
GpioDevConfig.h
GPIO device policy . . . 409
HdAudioConfig.h
HDAUDIO policy . . . 410
HostBridgeConfig.h
Configurations for HostBridge . . . 412
HsioConfig.h
HSIO policy . . . 413
HsioPcieConfig.h
HSIO pcie policy . . . 413
HsioSataConfig.h
Hsio Sata policy . . . 414
HybridGraphicsConfig.h
Hybrid Graphics policy definitions . . . 415
IaxConfig.h
IAX policy definitions . . . 415
IehConfig.h
Integrated Error Handler policy . . . 416
IGpuConfig.h
Policy definition for Internal Graphics Config Block . . . 417
InterruptConfig.h
Interrupt policy . . . 418
IoApicConfig.h
IoApic policy . . . 420
IpuPreMemConfig.h
IPU policy definitions . . . 420
IshConfig.h
ISH policy . . . 421
LockDownConfig.h
Lock down policy . . . 422
LpcConfig.h
Lpc policy . . . 423

LpssI2cConfig.h
I2C config defines . . . 424
LpssI3cConfig.h
I3C config defines . . . 425
LpssSpiConfig.h
LPSS SPI config defines . . . 427
LpssUartConfig.h
LPSS UART config defines . . . 428
MeDxeConfig.h
ME config block for DXE phase . . . 431
MemoryConfig.h
Policy definition of Memory Config Block Used by projects: PantherLake . . . 432
MemorySubSystemConfig.h
Policy definition for Memory SubSystem Config Block . . . 434
MePeiConfig.h
ME config block for PEI phase . . . 435
MpServices.h
This file declares UEFI PI Multi-processor PPI . . . 436
NpuConfig.h
NPU policy definitions . . . 441
OverclockingConfig.h
Overclocking Config Block . . . 443
P2sbConfig.h
P2sb policy . . . 443
PchDmiConfig.h
DMI policy . . . 444
PchGeneralConfig.h
PCH General policy . . . 445
PchPcieRpConfig.h
PCH Pcie root port policy . . . 446
PcieConfig.h
PCIe Config Block . . . 448
PeiDmiConfig.h
Config block defination for DMI . . . 450
PeiITbtConfig.h
Header file for TBT PEI Policy . . . 451
PeiITbtGenericStructure.h
ITBT Policy definition to be referred in both PEI and DXE phase . . . 452
PeiPreMemSiDefaultPolicy.h
This file defines the function to initialize default silicon policy PPI . . . 453
PeiSiDefaultPolicy.h
This file defines the function to initialize default silicon policy PPI . . . 454
PiHob.h
HOB related definitions in PI . . . 455
PiMultiPhase.h
Include file matches things in PI for multiple module types . . . 456
PmConfig.h
Power Management policy . . . 460
PsfConfig.h . . . 462
RtcConfig.h
RTC policy . . . 463
SataConfig.h
Sata policy . . . 464
ScsConfig.h
Scs policy . . . 465
SerialIoConfig.h
Serial IO policy . . . 466

SerialIoDevices.h
Serial IO policy . . . 467
SiConfig.h
Si Config Block . . . 471
SiPolicyStruct.h
Intel reference code configuration policies . . . 473
SiPreMemConfig.h
Si Config Block PreMem . . . 475
SmbusConfig.h
Smbus policy . . . 476
SpiConfig.h
PCH SPI Flash Controller config block . . . 477
SwizzleConfig.h
Config block for interrupt swizzle functionality configuration . . . 478
TcssPeiConfig.h
TCSS PEI policy . . . 479
TcssPeiPreMemConfig.h
TCSS PEI PreMem policy . . . 480
TelemetryPeiConfig.h
Configurations for Telemetry . . . 484
ThcConfig.h
Touch Host Controller policy . . . 485
ThermalConfig.h
Thermal policy . . . 487
TraceHubConfig.h
Configurations for TraceHub . . . 487
TsnConfig.h
TSN Config policy . . . 488
TxtConfig.h
TXT PreMemory Config Block . . . 490
UefiBaseType.h
Defines data types and constants introduced in UEFI . . . 491
Usb2PhyConfig.h
USB2 PHY configuration policy . . . 494
Usb3HsioConfig.h
USB3 Mod PHY configuration policy . . . 495
UsbConfig.h
Common USB policy shared between PCH and CPU Contains general features settings for xHCI and xDCI . . . 496
VmdPeiConfig.h
VMD PEI policy VMD Config Block . . . 497
VtdConfig.h
VT-d policy definitions . . . 498
WatchDogConfig.h
WatchDog policy . . . 499

Chapter 12

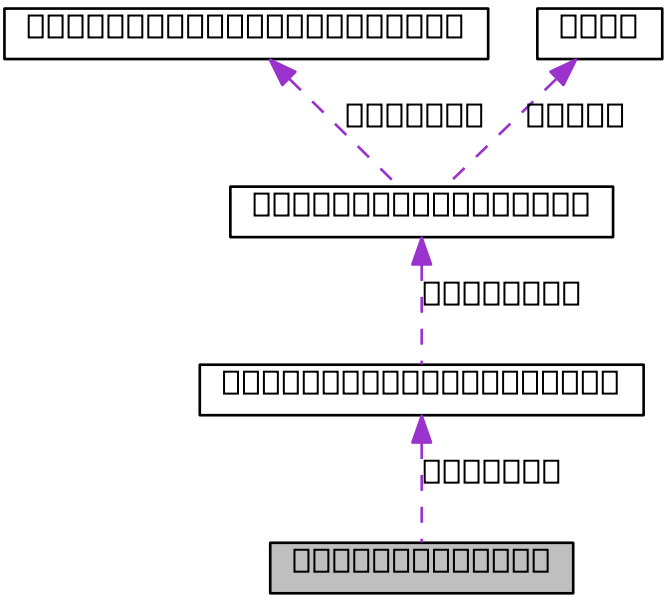
Class Documentation

12.1 _CONFIG_BLOCK Struct Reference

Config Block.

`#include <ConfigBlock.h>`

Collaboration diagram for _CONFIG_BLOCK:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Offset 0-27 Header of config block.

12.1.1 Detailed Description

Config Block.

Definition at line 45 of file ConfigBlock.h.

The documentation for this struct was generated from the following file:

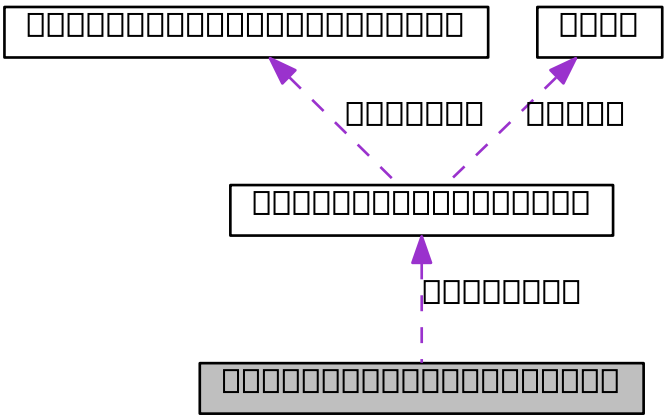
- ConfigBlock.h

12.2 _CONFIG_BLOCK_HEADER Struct Reference

Config Block Header.

```
#include <ConfigBlock.h>
```

Collaboration diagram for _CONFIG_BLOCK_HEADER:



Public Attributes

- EFI_HOB_GUID_TYPE GuidHob

  Offset 0-23 GUID extension HOB header.
- UINT8 Revision

  Offset 24 Revision of this config block.
- UINT8 Attributes

  Offset 25 The main revision for config block.
- UINT8 Reserved [2]

  Offset 26-27 Reserved for future use.

12.2.1 Detailed Description

Config Block Header.

Definition at line 35 of file ConfigBlock.h.

The documentation for this struct was generated from the following file:

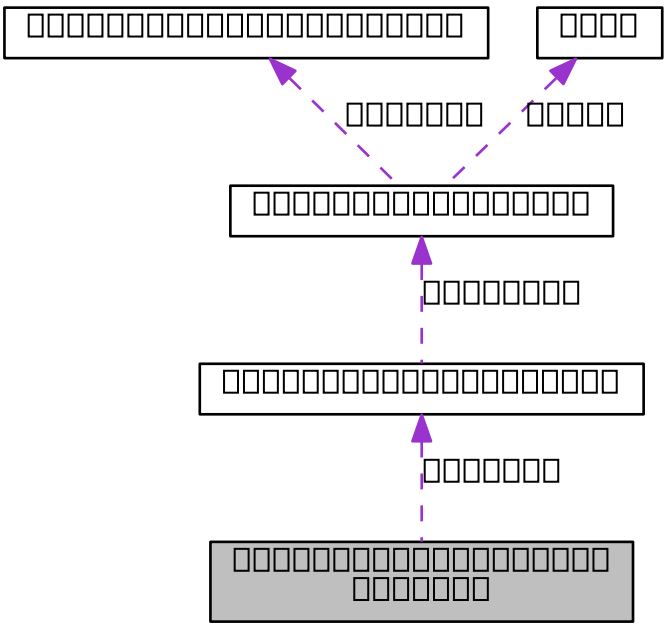
- ConfigBlock.h

12.3 _CONFIG_BLOCK_TABLE_STRUCT Struct Reference

Config Block Table Header.

`#include <ConfigBlock.h>`

Collaboration diagram for _CONFIG_BLOCK_TABLE_STRUCT:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Offset 0-27 GUID number for main entry of config block.
- UINT8 Rsvd0 [2]

  Offset 28-29 Reserved for future use.
- UINT16 NumberOfBlocks

  Offset 30-31 Number of config blocks (N)
- UINT32 AvailableSize

  Offset 32-35 Current config block table size.

12.3.1 Detailed Description

Config Block Table Header.

Definition at line 55 of file ConfigBlock.h.

The documentation for this struct was generated from the following file:

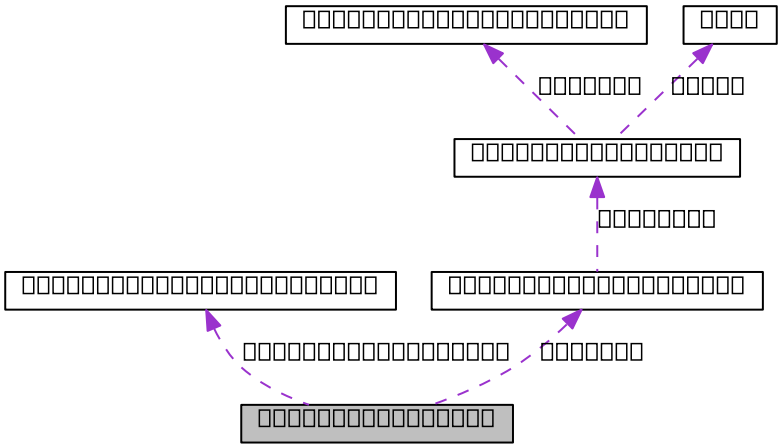
- ConfigBlock.h

12.4 _DXE_ITBT_CONFIG Struct Reference

ITBT DXE configuration
Revision 1:

```
#include <DxeITbtConfig.h>
```

Collaboration diagram for _DXE_ITBT_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header
  Offset 0-27 Config Block Header.
- DXE_ITBT_GENERIC_CONFIG ITbtGenericConfig
  ITbt Generic Information.
- TBT_RESOURCE_CONFIG ITbtResourceConfig [MAX_ITBT_PCIE_PORT]
  iTbt Resource Configuration
- UINT8 RsvdBytes54 [1]
  Reserved.
- UINT16 PcieRtd3LinkActiveTimeout
  PCIE RTD3 exit polling Link Active timeout and delay in PS0 before return to OS, unit is ms ***100***

12.4.1 Detailed Description

ITBT DXE configuration
Revision 1:

- Initial version. **Revision 2**:
- Add ITbtVproDockSupport in DXE_ITBT_GENERIC_CONFIG structure.

Definition at line 49 of file DxeITbtConfig.h.

The documentation for this struct was generated from the following file:

- DxeITbtConfig.h

12.5 _DXE_ITBT_GENERIC_CONFIG Struct Reference

ITBT Generic Data Structure to be used cross to RP and controller in DXE phase.

`#include <DxeITbtConfig.h>`

Public Attributes

- UINT8 ITbtRtd3
  *ITBT RTD3. - 0: Disable; **1: Enable**.*
- UINT8 ITbtVproDockSupport
  *ITBT vPro Dock support. - **0: not support**; 1: supported.*
- UINT16 ITbtRtd3ExitDelay
  *ITBT RTD3 delay before exit, unit is ms **0**.*

12.5.1 Detailed Description

ITBT Generic Data Structure to be used cross to RP and controller in DXE phase.

Definition at line 36 of file DxeITbtConfig.h.

The documentation for this struct was generated from the following file:

- DxeITbtConfig.h

12.6 _EFI_MM_RESERVED_MMRAM_REGION Struct Reference

Structure describing a MMRAM region which cannot be used for the MMRAM heap.

`#include <PiMultiPhase.h>`

Public Attributes

- EFI_PHYSICAL_ADDRESS MmramReservedStart

  Starting address of the reserved MMRAM area, as it appears while MMRAM is open.
- UINT64 MmramReservedSize

  Number of bytes occupied by the reserved MMRAM area.

12.6.1 Detailed Description

Structure describing a MMRAM region which cannot be used for the MMRAM heap.

Definition at line 146 of file PiMultiPhase.h.

12.6.2 Member Data Documentation

12.6.2.1 MmramReservedSize

```
UINT64 _EFI_MM_RESERVED_MMRAM_REGION::MmramReservedSize
```

Number of bytes occupied by the reserved MMRAM area.

A size of zero indicates the last MMRAM area.

Definition at line 156 of file PiMultiPhase.h.

12.6.2.2 MmramReservedStart

```
EFI_PHYSICAL_ADDRESS _EFI_MM_RESERVED_MMRAM_REGION::MmramReservedStart
```

Starting address of the reserved MMRAM area, as it appears while MMRAM is open.

Ignored if MmramReservedSize is 0.

Definition at line 151 of file PiMultiPhase.h.

The documentation for this struct was generated from the following file:

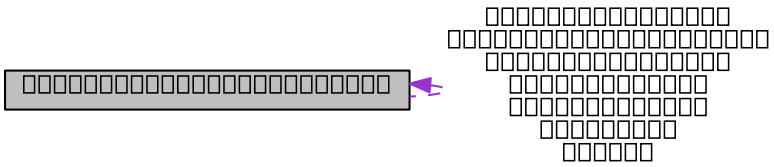
- PiMultiPhase.h

12.7 _EFI_PEI_MP_SERVICES_PPI Struct Reference

This PPI is installed by some platform or chipset-specific PEIM that abstracts handling multiprocessor support.

`#include <MpServices.h>`

Collaboration diagram for _EFI_PEI_MP_SERVICES_PPI:



12.7.1 Detailed Description

This PPI is installed by some platform or chipset-specific PEIM that abstracts handling multiprocessor support.

Definition at line 265 of file MpServices.h.

The documentation for this struct was generated from the following file:

- MpServices.h

12.8 _ITBT_GENERIC_CONFIG Struct Reference

ITBT Controller Data Structure to be used cross to RP and controller to be shared by CONFIG_BLOCK and HOB.

`#include <PeiITbtGenericStructure.h>`

Public Attributes

- UINT16 ITbtForcePowerOnTimeoutInMs

  Timeout value for forcing power iTBT controller on every boot/reboot/Sx exit as a precondition for execution of following mailbox communication.
- UINT16 ITbtConnectTopologyTimeoutInMs

  Timeout value while sending connect topology mailbox command in order to bring all connected TBT devices are available on PCIe before BIOS will enumerate them in BDS ***(Test) default is 5000 ms***
- UINT8 ITbtSecurityLevel

  iTbt Security Level ***Deprecated***
- UINT8 ITbtPcieTunnelingForUsb4

  Disable/Enable PCIe tunneling for USB4. ***default is enable***
- UINT8 Usb4CmMode

  USB4 CM mode.
- UINT8 Reserved [1]

  Reserved for DWORD alignment.

12.8.1 Detailed Description

ITBT Controller Data Structure to be used cross to RP and controller to be shared by CONFIG_BLOCK and HOB.

Definition at line 38 of file PeiITbtGenericStructure.h.

12.8.2 Member Data Documentation

12.8.2.1 ITbtForcePowerOnTimeoutInMs

```
UINT16 _ITBT_GENERIC_CONFIG::ITbtForcePowerOnTimeoutInMs
```

Timeout value for forcing power iTBT controller on every boot/reboot/Sx exit as a precondition for execution of following mailbox communication.

After applying Force Power Thunderbolt BIOS shall poll for iTBT readiness for mailbox communication If TBT cable is disconnected, iTBT microcontrollers are in lower power state. To ensure successful mailbox execution, independently on presence of TBT cable, TBT BIOS shall bring iTBT microcontrollers up by applying Force Power. iTBT microcontrollers will wake up either due to TBT cable presence or Force Power event. **(Test) default is 500 ms**

Definition at line 50 of file PeiITbtGenericStructure.h.

The documentation for this struct was generated from the following file:

- PeiITbtGenericStructure.h

12.9 _ITBT_ROOTPORT_CONFIG Struct Reference

iTBT RootPort Data Structure

```
#include <PeiITbtGenericStructure.h>
```

Public Attributes

- UINT8 ITbtPcieRootPortEn
  Disable/Enable iTBT PCIe Root Port.
- UINT8 Reserved [3]
  Reserved for DWORD alignment.

12.9.1 Detailed Description

iTBT RootPort Data Structure

Definition at line 30 of file PeiITbtGenericStructure.h.

The documentation for this struct was generated from the following file:

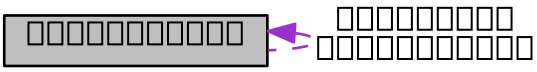
- PeiITbtGenericStructure.h

12.10 _LIST_ENTRY Struct Reference

_LIST_ENTRY structure definition.

```
#include <Base.h>
```

Collaboration diagram for _LIST_ENTRY:



12.10.1 Detailed Description

_LIST_ENTRY structure definition.

Definition at line 247 of file Base.h.

The documentation for this struct was generated from the following file:

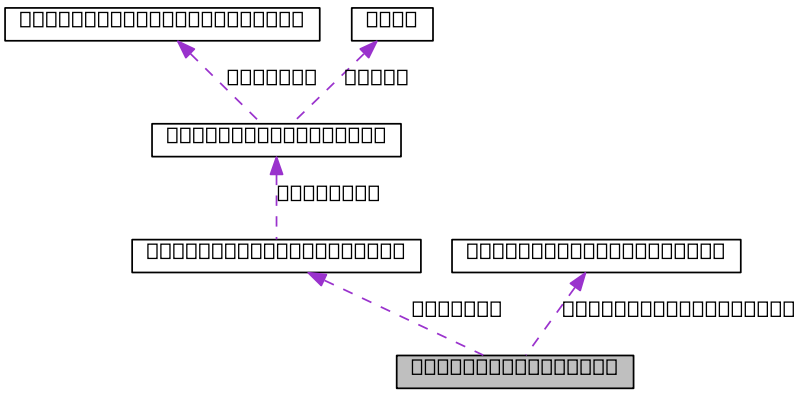
- Base.h

12.11 _PEI_ITBT_CONFIG Struct Reference

ITBT PEI configuration
Revision 1:

```
#include <PeiITbtConfig.h>
```

Collaboration diagram for _PEI_ITBT_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Offset 0-27 Config Block Header.
- ITBT_GENERIC_CONFIG ITbtGenericConfig

  ITbt Common Configuration.
- UINT16 ITbtDmaLtr [MAX_HOST_ITBT_DMA_NUMBER]

  iTbt Host controller DMA LTR value
- UINT8 RsvdBytes [4]

  Reserved for future use.

12.11.1 Detailed Description

ITBT PEI configuration
Revision 1:

- Initial version. **Revision 2**:
- Add ITbtPcieTunnelingForUsb4, deprecated ITbtSecurityLevel.

Definition at line 45 of file PeiITbtConfig.h.

The documentation for this struct was generated from the following file:

- PeiITbtConfig.h

12.12 _PEI_PREMEM_SI_DEFAULT_POLICY_INIT_PPI Struct Reference

This PPI provides function to install default silicon policy.

`#include <PeiPreMemSiDefaultPolicy.h>`

Public Attributes

- PEI_PREMEM_POLICY_INIT PeiPreMemPolicyInit

  PeiPreMemPolicyInit()

12.12.1 Detailed Description

This PPI provides function to install default silicon policy.

Definition at line 41 of file PeiPreMemSiDefaultPolicy.h.

The documentation for this struct was generated from the following file:

- PeiPreMemSiDefaultPolicy.h

12.13 _PEI_SI_DEFAULT_POLICY_INIT_PPI Struct Reference

This PPI provides function to install default silicon policy.

`#include <PeiSiDefaultPolicy.h>`

Public Attributes

- PEI_POLICY_INIT PeiPolicyInit

  PeiPolicyInit()

12.13.1 Detailed Description

This PPI provides function to install default silicon policy.

Definition at line 41 of file PeiSiDefaultPolicy.h.

The documentation for this struct was generated from the following file:

- PeiSiDefaultPolicy.h

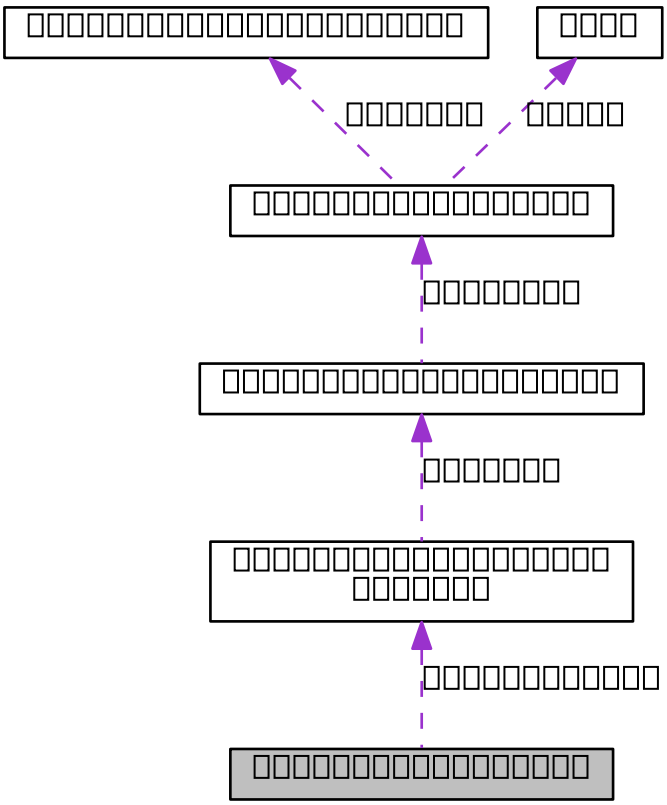
12.14 _SI_POLICY_STRUCT Struct Reference

SI Policy PPI
All SI config block change history will be listed here

.

`#include <SiPolicyStruct.h>`

Collaboration diagram for _SI_POLICY_STRUCT:



Public Attributes

- CONFIG_BLOCK_TABLE_HEADER TableHeader
  Config Block Table Header.

12.14.1 Detailed Description

SI Policy PPI
All SI config block change history will be listed here

.

- **Revision 1**:
  - Initial version.

Definition at line 71 of file SiPolicyStruct.h.

The documentation for this struct was generated from the following file:

- SiPolicyStruct.h

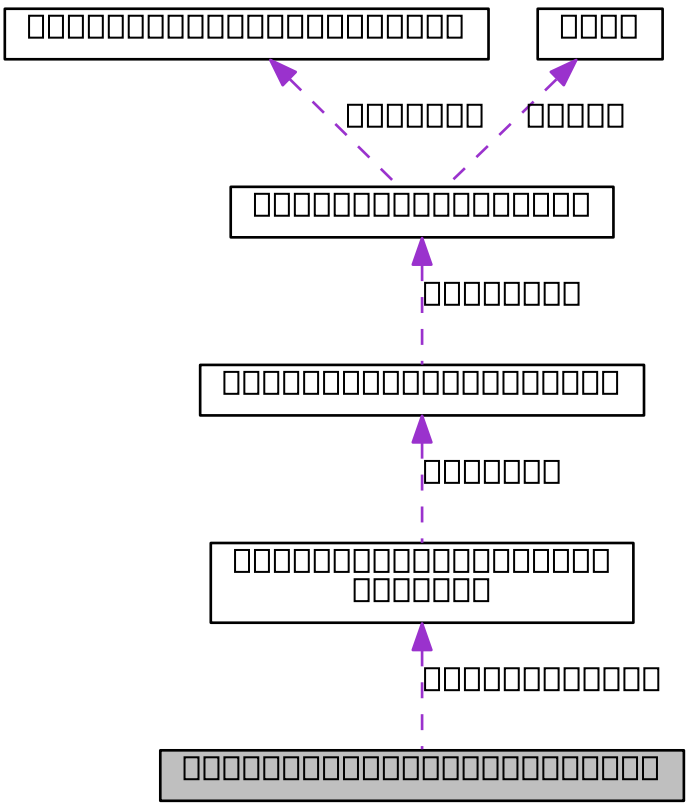
12.15 _SI_PREMEM_POLICY_STRUCT Struct Reference

SI Policy PPI in Pre-Mem
All SI config block change history will be listed here

.

```
#include <SiPolicyStruct.h>
```

Collaboration diagram for _SI_PREMEM_POLICY_STRUCT:



Public Attributes

- CONFIG_BLOCK_TABLE_HEADER TableHeader
  Config Block Table Header.

12.15.1 Detailed Description

SI Policy PPI in Pre-Mem
All SI config block change history will be listed here

.

- **Revision 1**:
  - Initial version.

Definition at line 57 of file SiPolicyStruct.h.

The documentation for this struct was generated from the following file:

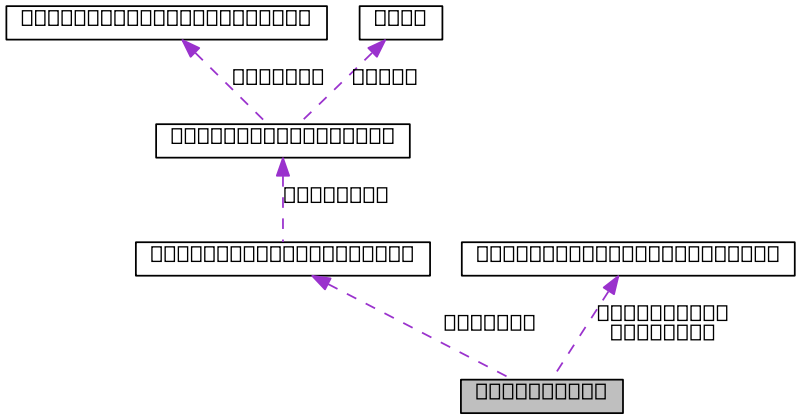
- SiPolicyStruct.h

12.16 ADR_CONFIG Struct Reference

ADR Configuration **Revision 1**: - Initial version.

`#include <AdrConfig.h>`

Collaboration diagram for ADR_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT32 AdrEn: 2

  Determine if Adr is enabled - 0: PLATFORM_POR, 1: FORCE_ENABLE, 2: FORCE_DISABLE.
- UINT32 AdrTimerEn: 2

  Determine if Adr timer options are enabled - 0: PLATFORM_POR, 1: FORCE_ENABLE, 2: FORCE_DISABLE.
- UINT32 AdrTimer1Val: 8

  Determines the Timeout value used for the ADR timer 1. A value of zero bypasses the timer.
- UINT32 AdrMultiplier1Val: 3

  Specifies the tick frequency upon which the timer 1 will increment. ADR_TIMER_SCALE should be used to encode values.
- UINT32 AdrTimer2Val: 8

Determines the Timeout value used for the ADR timer 2. A value of zero bypasses the timer.

- UINT32 AdrMultiplier2Val: 3

  Specifies the tick frequency upon which the timer 2 will increment. ADR_TIMER_SCALE should be used to encode values.

- UINT32 AdrHostPartitionReset: 2

  Determine if Host Partition Reset is enabled - 0: PLATFORM_POR, 1: FORCE_ENABLE, 2: FORCE_DISABLE.

- UINT32 AdrPlatAckEn: 2

  Determine if Platform Acknowledge is enabled - 0: PLATFORM_POR, 1: FORCE_ENABLE, 2: FORCE_DISABLE.

- ADR_GLOBAL_RESET_ENABLE AdrSrcOff

  Determine which ADR sources are disabled. The code perform clearing bits in ADR Global Reset Enable register. This is first step.

- ADR_GLOBAL_RESET_ENABLE AdrSrcOn

  Determine which ADR sources are enabled. The code perform setting bits in ADR Global Reset Enable register. This is second step.

12.16.1 Detailed Description

ADR Configuration **Revision 1**: - Initial version.

Definition at line 84 of file AdrConfig.h.

The documentation for this struct was generated from the following file:

- AdrConfig.h

12.17 ADR_GLOBAL_RESET_ENABLE Union Reference

ADR Source Enable.

```
#include <AdrConfig.h>
```

12.17.1 Detailed Description

ADR Source Enable.

Definition at line 46 of file AdrConfig.h.

The documentation for this union was generated from the following file:

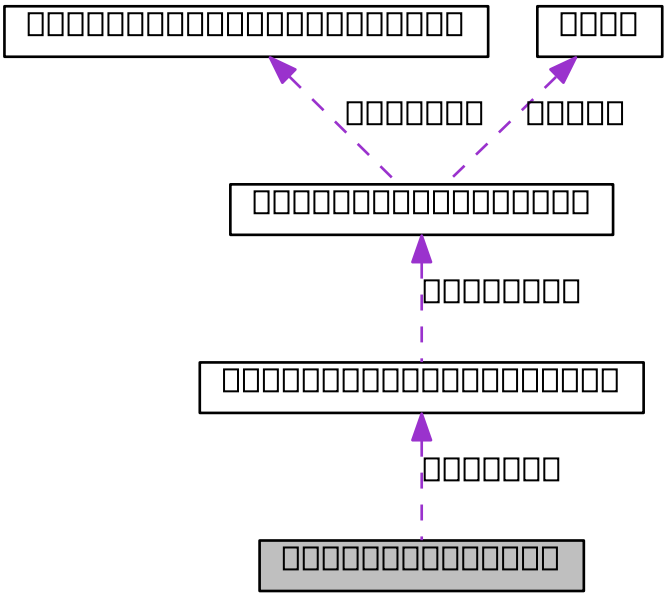
- AdrConfig.h

12.18 AMT_DXE_CONFIG Struct Reference

AMT Dxe Configuration Structure.

```
#include <AmtConfig.h>
```

Collaboration diagram for AMT_DXE_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT32 CiraRequest: 1

  *Trigger CIRA boot. **0: No CIRA request**; 1: Trigger CIRA request.*
- UINT32 UnConfigureMe: 1

  *OEMFlag Bit 15: Unconfigure ME with resetting MEBx password to default. **0: No**; 1: Un-configure ME without password.*
- UINT32 UsbProvision: 1

  *Enable/Disable of AMT USB Provisioning. **0: Disable**; 1: Enable.*
- UINT32 RsvdBits: 29

  Reserved for future use & Config block alignment.

12.18.1 Detailed Description

AMT Dxe Configuration Structure.

Revision 1:

- Initial version.

Definition at line 81 of file AmtConfig.h.

The documentation for this struct was generated from the following file:

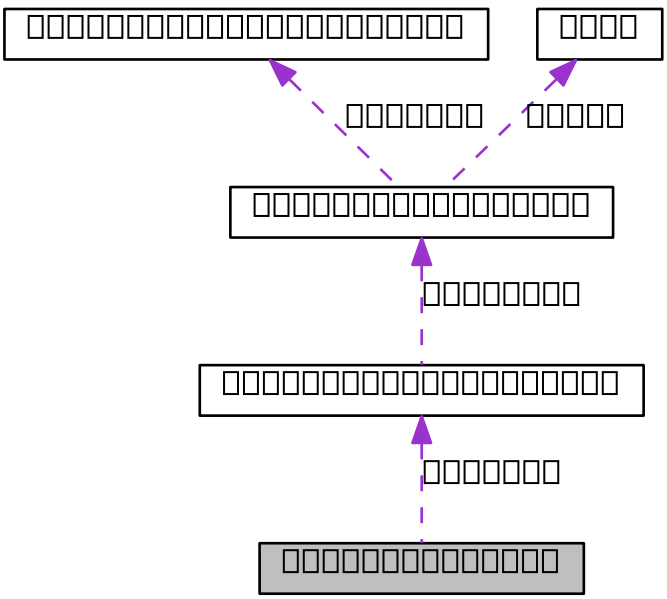
- AmtConfig.h

12.19 AMT_PEI_CONFIG Struct Reference

AMT Pei Configuration Structure.

```
#include <AmtConfig.h>
```

Collaboration diagram for AMT_PEI_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT32 AmtEnabled: 1

  Enable or Disable Intel Active Management Technology feature.
- UINT32 WatchDogEnabled: 1

  ME WatchDog timer feature.
- UINT32 AmtSolEnabled: 1

  Serial Over Lan retrieved from Mebx. The default value depends on CSME/AMT. 0: Disable, ***1: Enable***
- UINT32 RsvdBits: 29

  Reserved for future use & Config block alignment.
- UINT16 WatchDogTimerOs

  OS WatchDog Timer ***0: Disable*** *OS WDT won't be started after stopping BIOS WDT even if WatchDogEnabled is 1.*
- UINT16 WatchDogTimerBios

  BIOS WatchDog Timer ***0: Disable*** *BIOS WDT won't be started even if WatchDogEnabled is 1.*

12.19.1 Detailed Description

AMT Pei Configuration Structure.

Revision 1:

- Initial version.

Definition at line 41 of file AmtConfig.h.

12.19.2 Member Data Documentation

12.19.2.1 AmtEnabled

```
UINT32 AMT_PEI_CONFIG::AmtEnabled
```

Enable or Disable Intel Active Management Technology feature.

If disabled, all Intel AMT features, including Alert Standard Format features, will not be supported. 0: Disable **1: Enable.**

Definition at line 49 of file AmtConfig.h.

12.19.2.2 WatchDogEnabled

```
UINT32 AMT_PEI_CONFIG::WatchDogEnabled
```

ME WatchDog timer feature.

If disabled, below WatchDogTimerOs/WatchDogTimerBios will be irrelevant. See WatchDogTimerOs and Watch↩ DogTimerBios description. **0: Disable** 1: Enable ME WDT if corresponding timer value is not zero.

Definition at line 56 of file AmtConfig.h.

12.19.2.3 WatchDogTimerBios

```
UINT16 AMT_PEI_CONFIG::WatchDogTimerBios
```

BIOS WatchDog Timer **0: Disable** BIOS WDT won't be started even if WatchDogEnabled is 1.

Non zero value - The BIOS WDT is set according to the value and started if WatchDogEnabled is 1.

Definition at line 71 of file AmtConfig.h.

12.19.2.4 WatchDogTimerOs

```
UINT16 AMT_PEI_CONFIG::WatchDogTimerOs
```

OS WatchDog Timer **0: Disable** OS WDT won't be started after stopping BIOS WDT even if WatchDogEnabled is 1.

Non zero value - OS WDT will be started after stopping BIOS WDT if WatchDogEnabled is 1. The timer is set according to the value.

Definition at line 65 of file AmtConfig.h.

The documentation for this struct was generated from the following file:

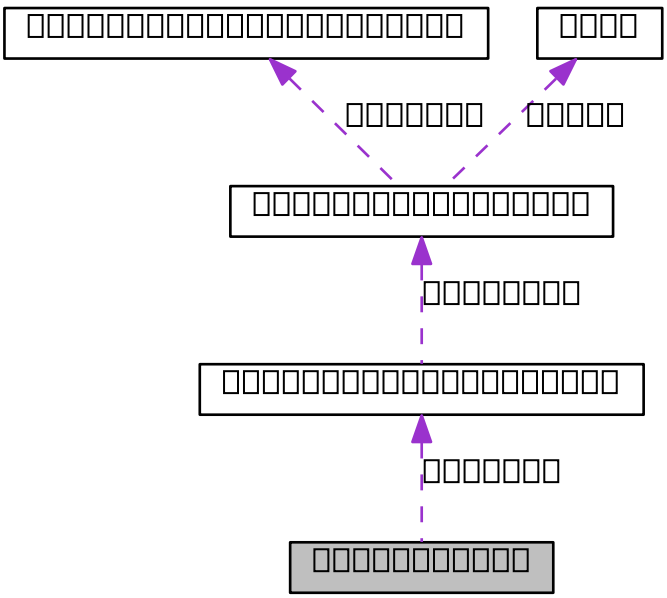
- AmtConfig.h

12.20 BCLK_CONFIG Struct Reference

BCLK configuration.

```
#include <PmConfig.h>
```

Collaboration diagram for BCLK_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT8 SocBclkPllOn

  Controls enable status of SOC BCLK.
- UINT8 CpuBclkPllOn

  Controls enable status of CPU BCLK.

12.20.1 Detailed Description

BCLK configuration.

Only available on projects with SOC and CPU BCLK supported.

Definition at line 461 of file PmConfig.h.

12.20.2 Member Data Documentation

12.20.2.1 CpuBclkPllOn

`UINT8 BCLK_CONFIG::CpuBclkPllOn`

Controls enable status of CPU BCLK.

0: HW default, 1: Force enable, 2: Force disable

Definition at line 473 of file PmConfig.h.

12.20.2.2 SocBclkPllOn

`UINT8 BCLK_CONFIG::SocBclkPllOn`

Controls enable status of SOC BCLK.

0: HW default, 1: Force enable, 2: Force disable

Definition at line 468 of file PmConfig.h.

The documentation for this struct was generated from the following file:

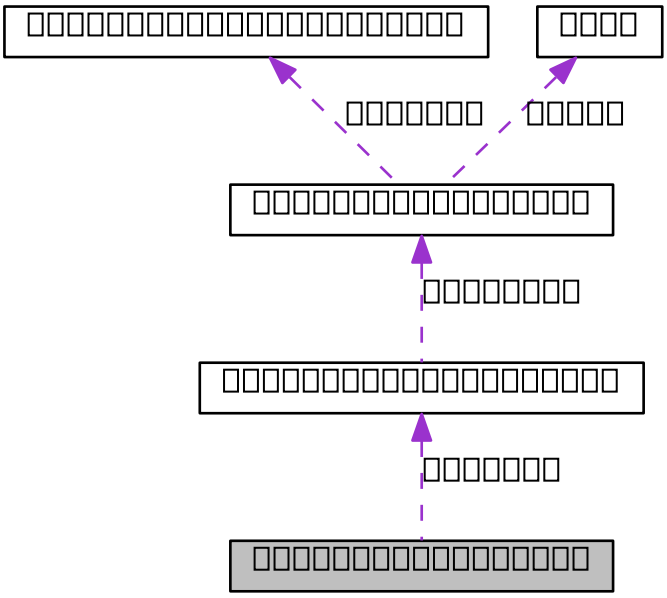
- PmConfig.h

12.21 BIOS_GUARD_CONFIG Struct Reference

BIOS Guard Configuration Structure.

`#include <BiosGuardConfig.h>`

Collaboration diagram for BIOS_GUARD_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- BIOSGUARD_ATTRIBUTES BiosGuardAttr

  BIT1 - EC Present, BIT2 - EC BIOS Guard protection, BIT3 - Descriptor Override policy, BIT4 - Flash wearout protection, BIT5 - FTU enable.
- UINT64 BgpdtHash [BIOSGUARD_TOTAL_HASH_MSRS]

  Hash of the BGPDT supported by BIOS Guard module PLAT_FRMW_PROT_HASH_0/1/2 MSR. Total HW HASH MSRS supported for BIOS Guard 2.2 = 6.
- EFI_PHYSICAL_ADDRESS BiosGuardModulePtr

  Pointer to the BIOS Guard Module.
- EFI_PHYSICAL_ADDRESS EcProvisionEav

  Pointer to the EC function Provision Ephemeral Auth Value.
- EFI_PHYSICAL_ADDRESS EcBiosGuardCmdLock

  Pointer to the EC function Provision Lock.

12.21.1 Detailed Description

BIOS Guard Configuration Structure.

Platform policies for BIOS Guard Configuration for all processor security features configuration. Platform code can pass relevant configuration data through this structure.

Note

Optional. These policies will be ignored if CPU_SECURITY_PREMEM_CONFIG -> BiosGuard is disabled, or PeiBiosGuardLibNull is used.

Revision 1:

- Initial version.

Definition at line 44 of file BiosGuardConfig.h.

The documentation for this struct was generated from the following file:

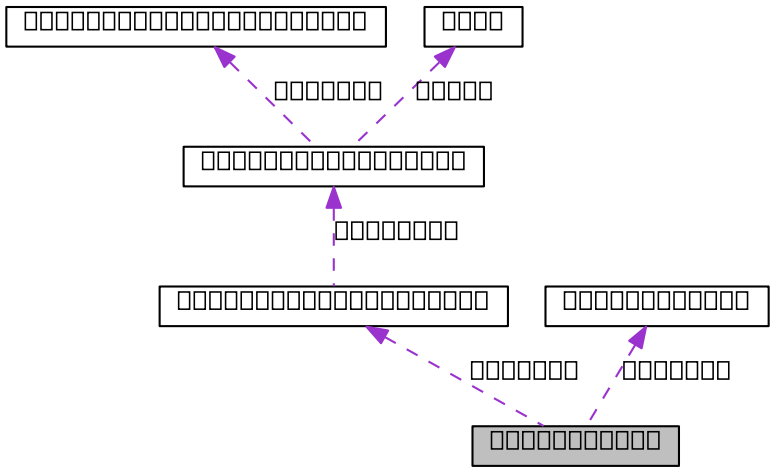
- BiosGuardConfig.h

12.22 CNVI_CONFIG Struct Reference

The CNVI_CONFIG block describes the expected configuration of the CNVi IP.

```
#include <CnviConfig.h>
```

Collaboration diagram for CNVI_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT32 Mode: 1

  This option allows for automatic detection of Connectivity Solution.
- UINT32 BtCore: 1

  The option to turn ON or OFF the BT Core. 0: Disabled, ***1: Enabled***
- UINT32 BtAudioOffload: 1

  The option to enable or disable BT Audio Offload.
- UINT32 WifiCore: 1

  The option to turn ON or OFF the Wi-Fi Core. 0: Disabled, ***1: Enabled***
- UINT32 BtInterface: 2

  This option allows configuration of BT interface to USB or PCIe Use CNVI_BT_INTERFACE enum for selection: ***CnviBtIfUsb = BT over USB interface*** *CnviBtIfPci = BT over PCIe interface.*
- CNVI_PIN_MUX PinMux

  CNVi PinMux Configuration RESET#/CLKREQ to CRF, can have two alternative mappings, depending on board routing requirements.

12.22.1 Detailed Description

The CNVI_CONFIG block describes the expected configuration of the CNVi IP.

Revision 1: - Initial version. **Revision 2**: - Add DdrRfim. **Revision 3**: - Add WifiCore. **Revision 4**: - Move DdrRfim to Pre-Mem config. **Revision 5**: - Add WwanCoex. **Revision 6**: - Add SkipBtPreInit.

Definition at line 79 of file CnviConfig.h.

12.22.2 Member Data Documentation

12.22.2.1 BtAudioOffload

```
UINT32 CNVI_CONFIG::BtAudioOffload
```

The option to enable or disable BT Audio Offload.

0: Disabled, **1: Enabled**

Note

This feature supports from Intel(R) Wireless-AX 22560

Definition at line 95 of file CnviConfig.h.

12.22.2.2 BtInterface

```
UINT32 CNVI_CONFIG::BtInterface
```

This option allows configuration of BT interface to USB or PCIe Use CNVI_BT_INTERFACE enum for selection: **CnviBtIfUsb = BT over USB interface** CnviBtIfPci = BT over PCIe interface.

Note

PCIe interface supports from Blazar generation

Definition at line 105 of file CnviConfig.h.

12.22.2.3 Mode

```
UINT32 CNVI_CONFIG::Mode
```

This option allows for automatic detection of Connectivity Solution.

Auto Detection assumes that CNVi will be enabled when available; Disable allows for disabling CNVi. CnviMode↩ Disabled = Disabled, **CnviModeAuto = Auto Detection**

Definition at line 88 of file CnviConfig.h.

The documentation for this struct was generated from the following file:

- CnviConfig.h

12.23 CNVI_PIN_MUX Struct Reference

CNVi signals pin muxing settings.

```
#include <CnviConfig.h>
```

Public Attributes

- UINT32 RfReset

  RF_RESET# Pin mux configuration. Refer to GPIO_∗_MUXING_CNVI_RF_RESET_∗.
- UINT32 Clkreq

  CLKREQ Pin mux configuration. Refer to GPIO_∗_MUXING_CNVI_∗_CLKREQ_∗.

12.23.1 Detailed Description

CNVi signals pin muxing settings.

If signal can be enable only on a single pin then this parameter is ignored by RC. Refer to GPIO_∗_MUXING_C↩ NVI_∗ in GpioPins∗.h for supported settings on a given platform

Definition at line 64 of file CnviConfig.h.

The documentation for this struct was generated from the following file:

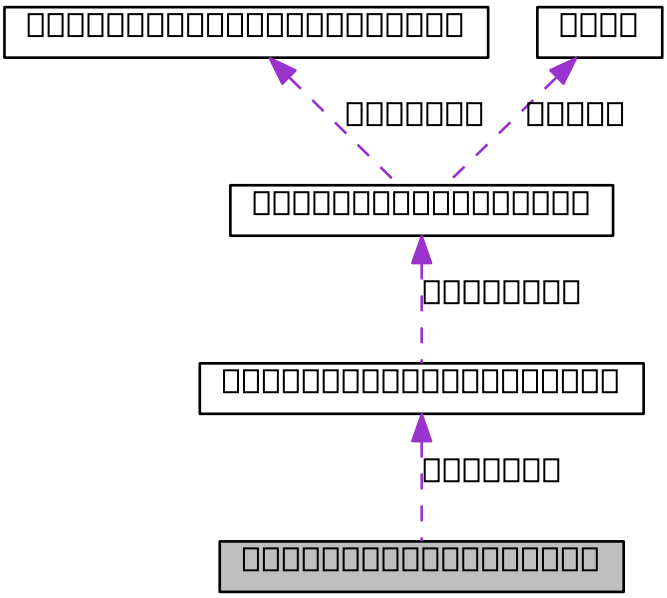
- CnviConfig.h

12.24 CNVI_PREMEM_CONFIG Struct Reference

The CNVI_PREMEM_CONFIG block describes the expected configuration of the CNVi IP.

`#include <CnviConfig.h>`

Collaboration diagram for CNVI_PREMEM_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT8 DdrRfim

  The option to enable or disable DDR RFI Mitigation. 0: Disabled, ***1: Enabled***

12.24.1 Detailed Description

The CNVI_PREMEM_CONFIG block describes the expected configuration of the CNVi IP.

Revision 1: - Initial version.

Definition at line 122 of file CnviConfig.h.

The documentation for this struct was generated from the following file:

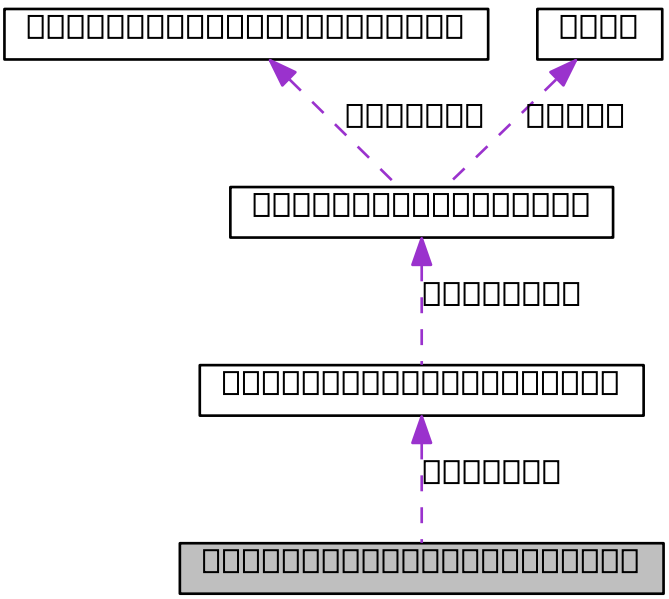
- CnviConfig.h

12.25 CPU_INIT_PREMEM_CONFIG Struct Reference

CPU Config Library PreMemory Configuration Structure.

```
#include <CpuInitPreMemConfig.h>
```

Collaboration diagram for CPU_INIT_PREMEM_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT32 BootMaxFrequency: 1

  Boot Maximum Frequency.
- UINT32 BistOnReset: 1

  ***(Test)** Enable or Disable BIST on Reset; **0: Disable**; 1: Enable.*
- UINT32 VmxEnable: 1

  Enable or Disable Virtual Machine Extensions (VMX) feature.
- UINT32 FClkFrequency: 2

  Processor Early Power On Configuration FCLK setting.
- UINT32 CrashLogEnable: 1

  Enable or Disable CrashLog feature.
- UINT32 TmeEnable: 1

  Enable or Disable Total Memory Encryption (TME) feature.
- UINT32 DebugInterfaceEnable: 2

  *Enable or Disable processor debug features; 0: Disable; 1: Enable; **2: No Change**.*
- UINT32 DebugInterfaceLockEnable: 1

  *Lock or Unlock debug interface features; 0: Disable; **1: Enable**.*
- UINT32 PeciSxReset: 1

  Enables a mailbox command to resolve rare PECI related Sx issues.
- UINT32 CrashLogGprs: 2

  Enable or Disable CrashLog GPRs dump.
- UINT32 OcLock: 1

  Set OC lock bit in MSR 0x194[20] will lock the OC mailbox for OC configration settings and Turbo Ratio Limit.
- UINT8 CpuRatio

  CpuRatio - Max non-turbo ratio (Flexible Ratio Boot) is set to CpuRatio.
- UINT8 ActiveCoreCount

  Number of big cores in processor to enable.
- UINT8 ActiveSmallCoreCount

  Number of small cores in processor to enable.
- UINT8 ActiveLpAtomCoreCount

  Number of enabled LP Atom cores.
- UINT8 DfdEnable

  Determine whether to enable or disable DFD.
- UINT32 PrmrrSize

  PRMRR Size.
- UINT8 TmeBypassCapability

  Enable or Disable Total Memory Encryption (TME) Bypass Mode If TME encryption bypass is enabled, then transactions with KeyID0 Bypass the encryption in CCE.
- UINT32 TsegSize

  Size of TSEG in bytes.
- UINT8 SmmRelocationEnable

  Enable or disable SmmRelocation.
- UINT8 ReduceXecores

  Enable or disable ReduceXecores.

12.25.1 Detailed Description

CPU Config Library PreMemory Configuration Structure.

Revision 1:

- Initial version.

Definition at line 36 of file CpuInitPreMemConfig.h.

12.25.2 Member Data Documentation

12.25.2.1 ActiveCoreCount

`UINT8 CPU_INIT_PREMEM_CONFIG::ActiveCoreCount`

Number of big cores in processor to enable.

- 0: Disable all cores for Hybrid CPU; Active all cores for Non-Hybrid CPU.
- 1: 1 core
- 2: 2 cores
- 3: 3 cores

Definition at line 114 of file CpuInitPreMemConfig.h.

12.25.2.2 ActiveLpAtomCoreCount

`UINT8 CPU_INIT_PREMEM_CONFIG::ActiveLpAtomCoreCount`

Number of enabled LP Atom cores.

Default is set to supported LP Atom core number.

- 0 - all core disabled
- 1 - enable 1 core
- 2 - enable 2 cores
- 3 - enable 3 cores
- 4 - enable 4 cores
- 0xFF - enable all cores
- others - reserved for future use

Definition at line 136 of file CpuInitPreMemConfig.h.

12.25.2.3 ActiveSmallCoreCount

`UINT8 CPU_INIT_PREMEM_CONFIG::ActiveSmallCoreCount`

Number of small cores in processor to enable.

- 0: Disable all cores for Hybrid CPU; Active all cores for Non-Hybrid CPU.
- 1: 1 core
- 2: 2 cores
- 3: 3 cores

Definition at line 123 of file CpuInitPreMemConfig.h.

12.25.2.4 BootMaxFrequency

`UINT32 CPU_INIT_PREMEM_CONFIG::BootMaxFrequency`

Boot Maximum Frequency.

- 0: Disable
- **1: Enable**

Definition at line 43 of file CpuInitPreMemConfig.h.

12.25.2.5 CpuRatio

`UINT8 CPU_INIT_PREMEM_CONFIG::CpuRatio`

CpuRatio - Max non-turbo ratio (Flexible Ratio Boot) is set to CpuRatio.

0: Disabled If disabled, doesn't override max-non turbo ratio.

Definition at line 104 of file CpuInitPreMemConfig.h.

12.25.2.6 CrashLogEnable

`UINT32 CPU_INIT_PREMEM_CONFIG::CrashLogEnable`

Enable or Disable CrashLog feature.

- 0: Disable
- **1: Enable**

Definition at line 65 of file CpuInitPreMemConfig.h.

12.25.2.7 CrashLogGprs

`UINT32 CPU_INIT_PREMEM_CONFIG::CrashLogGprs`

Enable or Disable CrashLog GPRs dump.

- **0: Disable**
- 1: Gprs Enabled, Smm Gprs Enabled 2: Gprs Enabled, Smm Gprs Disabled

Definition at line 91 of file CpuInitPreMemConfig.h.

12.25.2.8 DfdEnable

`UINT8 CPU_INIT_PREMEM_CONFIG::DfdEnable`

Determine whether to enable or disable DFD.

DFD is enabled from cold boot.

- **0 : DFD disabled**
- 1 : DFD enabled

Definition at line 143 of file CpuInitPreMemConfig.h.

12.25.2.9 FClkFrequency

`UINT32 CPU_INIT_PREMEM_CONFIG::FClkFrequency`

Processor Early Power On Configuration FCLK setting.

- **0: 800 MHz (ULT/ULX)**.
- **1: 1 GHz (DT/Halo)**. Not supported on ULT/ULX.
- 2: 400 MHz.
- 3: Reserved.

Definition at line 59 of file CpuInitPreMemConfig.h.

12.25.2.10 OcLock

`UINT32 CPU_INIT_PREMEM_CONFIG::OcLock`

Set OC lock bit in MSR 0x194[20] will lock the OC mailbox for OC configration settings and Turbo Ratio Limit.

- 0: Disable
- **1: Enable (Lock)**

Definition at line 98 of file CpuInitPreMemConfig.h.

12.25.2.11 PeciSxReset

`UINT32 CPU_INIT_PREMEM_CONFIG::PeciSxReset`

Enables a mailbox command to resolve rare PECI related Sx issues.

Note

This should only be used on systems that observe PECI Sx issues.

- **0: Disable**
- 1: Enable

Definition at line 83 of file CpuInitPreMemConfig.h.

12.25.2.12 PrmrrSize

`UINT32 CPU_INIT_PREMEM_CONFIG::PrmrrSize`

PRMRR Size.

- **Software Control : 0x0**
- 2MB : 0x200000
- 4MB : 0x400000
- 16 MB : 0x1000000

Definition at line 152 of file CpuInitPreMemConfig.h.

12.25.2.13 ReduceXecores

```
UINT8 CPU_INIT_PREMEM_CONFIG::ReduceXecores
```

Enable or disable ReduceXecores.

- 0: Disable (Default)
- 1: Enable

Definition at line 181 of file CpuInitPreMemConfig.h.

12.25.2.14 SmmRelocationEnable

```
UINT8 CPU_INIT_PREMEM_CONFIG::SmmRelocationEnable
```

Enable or disable SmmRelocation.

For Coreboot customer it must be disabled.

- 0: Disable
- 1: Enable (Default)

Definition at line 175 of file CpuInitPreMemConfig.h.

12.25.2.15 TmeBypassCapability

```
UINT8 CPU_INIT_PREMEM_CONFIG::TmeBypassCapability
```

Enable or Disable Total Memory Encryption (TME) Bypass Mode If TME encryption bypass is enabled, then transactions with KeyID0 Bypass the encryption in CCE.

- 0: Disable (Default)
- 1: Enable

Definition at line 160 of file CpuInitPreMemConfig.h.

12.25.2.16 TmeEnable

```
UINT32 CPU_INIT_PREMEM_CONFIG::TmeEnable
```

Enable or Disable Total Memory Encryption (TME) feature.

- 0: Disable
- **1: Enable**

Definition at line 72 of file CpuInitPreMemConfig.h.

12.25.2.17 TsegSize

```
UINT32 CPU_INIT_PREMEM_CONFIG::TsegSize
```

Size of TSEG in bytes.

(Must be power of 2) **0x400000**: 4MB for Release build 0x1000000 : 16MB for Debug build

Definition at line 168 of file CpuInitPreMemConfig.h.

12.25.2.18 VmxEnable

```
UINT32 CPU_INIT_PREMEM_CONFIG::VmxEnable
```

Enable or Disable Virtual Machine Extensions (VMX) feature.

- 0: Disable
- **1: Enable**

Definition at line 51 of file CpuInitPreMemConfig.h.

The documentation for this struct was generated from the following file:

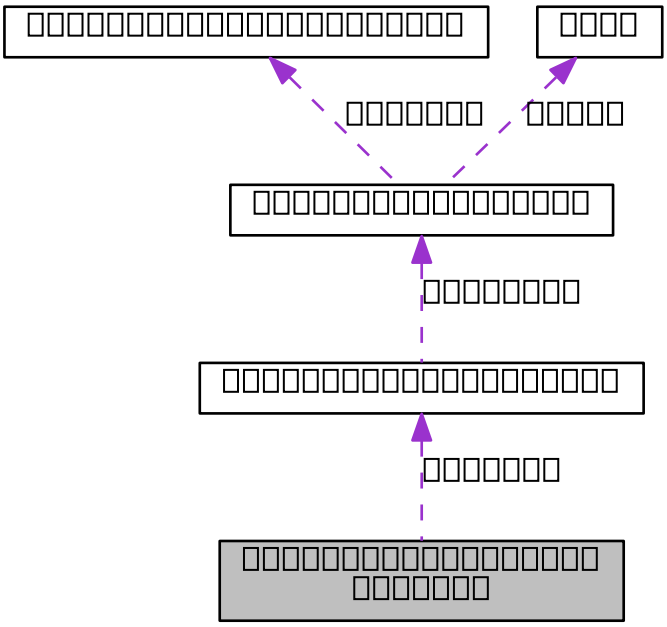
- CpuInitPreMemConfig.h

12.26 CPU_POWER_DELIVERY_CONFIG Struct Reference

CPU Power Delivery Configuration Structure.

`#include <CpuPowerDeliveryConfig.h>`

Collaboration diagram for CPU_POWER_DELIVERY_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT32 PowerLimit2: 1

  *Enable or Disable short duration Power Limit (PL2). 0: Disable; **1: Enable***
- UINT32 PowerLimit3DutyCycle: 8

  *Package PL3 Duty Cycle. Specifies the PL3 duty cycle percentage, Range 0-100. **Default: 0**.*
- UINT32 PowerLimit3Lock: 1

  *Package PL3 MSR 615h lock; **0: Disable**; 1: Enable (Lock).*
- UINT32 PowerLimit4Lock: 1

  *Package PL4 MSR 601h lock; **0: Disable**; 1: Enable (Lock).*
- UINT32 TccOffsetLock: 1

  *Tcc Offset Lock for Runtime Average Temperature Limit (RATL) to lock temperature target MSR 1A2h; 0: Disabled; **1: Enabled (Lock)**.*
- UINT32 ApplyConfigTdp: 1

  *Switch TDP applied setting based on non-cTDP or TDP; 0: non-cTDP; **1: cTDP**.*
- UINT32 DualTauBoost: 1

Dual Tau Base Performance Boost support.

- UINT32 ResponseMode: 1

  *ResponseMode - PL3 CONTROL MSR 0x615 BIT 16, it indicates how quickly Pcode reduces power consumption to a low (safe) level; **0: Gradual Power Reduction**; 1: Aggressive Power Reduction.*
- UINT32 PlatformAtxTelemetryUnit: 1

  PMC_SUS_SPARE_GCR_0 bit 0, set by BIOS to indicate the telemetry type for Psys support.
- UINT32 RsvdBits: 16

  Reserved for future use.
- UINT16 PowerLimit1

  Package Long duration turbo mode power limit (PL1).
- UINT16 PowerLimit2Power

  Package Short duration turbo mode power limit (PL2).
- UINT16 PowerLimit3

  Package PL3 power limit.
- UINT16 PowerLimit4

  Package PL4 power limit.
- UINT32 PowerLimit1Time

  Package Long duration turbo mode power limit (PL1) time window in seconds.
- UINT32 PowerLimit3Time

  Package PL3 time window. Range from 3ms to 64ms.
- UINT32 PsysPowerLimit1: 1

  MSR 0x65C[15]: PL1 Enable activates the PL1 value to limit average platform power.
- UINT32 PsysPowerLimit1Time: 8

  MSR 0x65C[23:17]: PL1 timewindow in seconds.
- UINT32 PsysPowerLimit2: 1

  MSR 0x65C[47]: PL2 Enable activates the PL2 value to limit average platform power.
- UINT32 ConfigTdpLock: 1

  *Lock the ConfigTdp mode settings from runtime changes; **0: Disable**; 1: Enable.*
- UINT32 ConfigTdpBios: 1

  *Configure whether to load Configurable TDP SSDT; **0: Disable**; 1: Enable.*
- UINT32 DisableVrThermalAlert: 1

  *Enable or Disable VR Thermal Alert; **0: Disable**; 1: Enable.*
- UINT32 ThermalMonitor: 1

  *Enable or Disable Thermal Monitor; 0: Disable; **1: Enable**.*
- UINT32 ConfigTdpLevel: 8

  *Configuration for boot TDP selection; **0: TDP Nominal**; 1: TDP Level1; 2: TDP Level2.*
- UINT32 AcDcPowerState: 1

  AC/DC Power state.
- UINT32 RsvdBits1: 9

  Reserved for future use.
- UINT16 PsysPowerLimit1Power

  MSR 0x65C[14:0]: Platform PL1 power. Units are 125 milliwatt.
- UINT16 PsysPowerLimit2Power

  MSR 0x65C[46:32]]: Platform PL2 power. Units are 125 milliwatt.
- UINT32 CustomPowerLimit1Time: 8

  Short term Power Limit time window value for custom cTDP level.
- UINT32 CustomTurboActivationRatio: 8

  Turbo Activation Ratio for custom cTDP level.
- UINT32 RsvdBits2: 16

  Bits reserved for DWORD alignment.
- UINT16 CustomPowerLimit1

Short term Power Limit value for custom cTDP level. Units are 125 milliwatt.

- UINT16 CustomPowerLimit2

  Long term Power Limit value for custom cTDP level. Units are 125 milliwatt.
- UINT16 PowerLimit4Boost

  Package Power Limit 4 Boost configuration.
- UINT8 TccActivationOffset

  TCC Activation Offset.
- UINT16 IsysCurrentLimitL1

  This field indicated the current limitiation of L1. Units of measurements are 1/8 A. ***Default: 0x0****. Range 0-0x7FFF.*
- UINT8 IsysCurrentLimitL1Enable

  *This bits enables disables ISYS_CURRENT_LIMIT_L1 algorithm.****Default: 0****.*
- UINT8 IsysCurrentL1Tau

  Specifies the time window used to calculate average current for ISYS_L1. ***Default: 0x0****. Range 0-255.*
- UINT16 IsysCurrentLimitL2

  This field indicated the current limitiation of L2. Units of measurements are 1/8 A. ***Default: 0x0****. Range 0-0x7FFF.*
- UINT8 IsysCurrentLimitL2Enable

  *This bits enables disables ISYS_CURRENT_LIMIT_L2 algorithm.****Default: 0****.*
- UINT16 VsysMax

  *Specifies Vsys Max defined in 1/1000 increments.****Default: 0x0*** *. Range 0-65535.*
- UINT8 ThETAIbattEnable

  ThETA is name given to this feature which stands for Thermally Equal Turbo-boost Algorithm.
- UINT8 SkinTempControl

  SoC firmware based Skin Thermal Control feature programmed via MMIO register.
- UINT8 SkinTargetTemp [SKIN_TEMP_CONTROL_SENSOR]

  Target temperature is limit to which the control mechanism is regulating.It is defined in 1/2 C increments.Range is 0-122.5 C. For a 0.5 C temperature, enter 1. 0 = AUTO.
- UINT8 SkinTempControlEnable [SKIN_TEMP_CONTROL_SENSOR]

  *Enables the skin temperature control for MMIO register.****0: Disable****; 1: Enable.*
- UINT8 SkinControlLoopGain [SKIN_TEMP_CONTROL_SENSOR]

  Sets the aggressiveness of control loop where 0 is graceful, favors performance on expense of temperature overshoots and 7 is aggressive, favors tight regulation over performance. Range is 0 to 7.
- UINT8 SkinTempOverrideEnable [SKIN_TEMP_CONTROL_SENSOR]

  *When set, Pcode will use TEMPERATURE_OVERRIDE values instead of reading from corresponding sensor.****0: Disable****; 1: Enable.*
- UINT8 SkinMinPerformanceLevel [SKIN_TEMP_CONTROL_SENSOR]

  Minimum Performance level below which the STC limit will not throttle. 0 - all levels of throttling allowed incl. survivability actions. 256 - no throttling allowed.
- UINT8 SkinTempOverride [SKIN_TEMP_CONTROL_SENSOR]

  Allows SW to override the input temperature.Pcode will use this value instead of the sensor temperature.EC control is not impacted.Units: 0.5C.Values are 0 to 255 which represents 0C - 122.5C range.

12.26.1 Detailed Description

CPU Power Delivery Configuration Structure.

Revision 1:

- Initial version.

Definition at line 44 of file CpuPowerDeliveryConfig.h.

12.26.2 Member Data Documentation

12.26.2.1 DualTauBoost

`UINT32 CPU_POWER_DELIVERY_CONFIG::DualTauBoost`

Dual Tau Base Performance Boost support.

This will improve platform performance for desktop sku. When enabled, it exposes 2 MMIO PL1 and MMIO Tau registers, and also provides the default value.

Note

: This is only applicable for Desktop 35W/65W/125W sku.

: When this is enabled, it will override the settings in policys PowerLimit1, PowerLimit1Time, PowerLimit2↩ Power. **0: Disable**; 1: Enable

Definition at line 59 of file CpuPowerDeliveryConfig.h.

12.26.2.2 PowerLimit1

`UINT16 CPU_POWER_DELIVERY_CONFIG::PowerLimit1`

Package Long duration turbo mode power limit (PL1).

Default is the TDP power limit of processor. Units are 125 milliwatt.

Definition at line 70 of file CpuPowerDeliveryConfig.h.

12.26.2.3 PowerLimit1Time

`UINT32 CPU_POWER_DELIVERY_CONFIG::PowerLimit1Time`

Package Long duration turbo mode power limit (PL1) time window in seconds.

Used in calculating the average power over time. Mobile: **28s** Desktop: **8s** Range: 0 - 128s

Definition at line 94 of file CpuPowerDeliveryConfig.h.

12.26.2.4 PowerLimit2Power

```
UINT16 CPU_POWER_DELIVERY_CONFIG::PowerLimit2Power
```

Package Short duration turbo mode power limit (PL2).

Allows for short excursions above TDP power limit. Default = 1.25 $*$ TDP Power Limit. Units are 125 milliwatt.

Definition at line 75 of file CpuPowerDeliveryConfig.h.

12.26.2.5 PowerLimit3

```
UINT16 CPU_POWER_DELIVERY_CONFIG::PowerLimit3
```

Package PL3 power limit.

PL3 is the CPU Peak Power Occurences Limit. **Default: 0**. Range 0-65535. Units are 125 milliwatt.

Definition at line 80 of file CpuPowerDeliveryConfig.h.

12.26.2.6 PowerLimit4

```
UINT16 CPU_POWER_DELIVERY_CONFIG::PowerLimit4
```

Package PL4 power limit.

PL4 is a Preemptive CPU Package Peak Power Limit, it will never be exceeded. Power is premptively lowered before limit is reached. **Default: 0**. Range 0-65535. Units are 125 milliwatt.

Definition at line 86 of file CpuPowerDeliveryConfig.h.

12.26.2.7 PowerLimit4Boost

```
UINT16 CPU_POWER_DELIVERY_CONFIG::PowerLimit4Boost
```

Package Power Limit 4 Boost configuration.

Default: 0. Units are 125 milliwatt. Valid Range 0 to 63000 in step size of 125/1000 Watt. The value 0 means disable.

Definition at line 119 of file CpuPowerDeliveryConfig.h.

12.26.2.8 SkinTempControl

```
UINT8 CPU_POWER_DELIVERY_CONFIG::SkinTempControl
```

SoC firmware based Skin Thermal Control feature programmed via MMIO register.

- Configures these MMIO option to set 3 sensors: thresholds, gain, min perf
- Controls setting enable/disable sensor temperature and ranges Enable or Disable STC (Skin Thermal Control) Feature. **0: Disable**; 1: Enable.

Definition at line 147 of file CpuPowerDeliveryConfig.h.

12.26.2.9 TccActivationOffset

```
UINT8 CPU_POWER_DELIVERY_CONFIG::TccActivationOffset
```

TCC Activation Offset.

Offset from factory set TCC activation temperature at which the Thermal Control Circuit must be activated. TCC will be activated at (TCC Activation Temperature - TCC Activation Offset), in degrees Celcius. **Default: 0**, causing TCC to activate at TCC Activation temperature.

Note

The policy is recommended for validation purpose only.

Definition at line 127 of file CpuPowerDeliveryConfig.h.

12.26.2.10 ThETAIbattEnable

```
UINT8 CPU_POWER_DELIVERY_CONFIG::ThETAIbattEnable
```

ThETA is name given to this feature which stands for Thermally Equal Turbo-boost Algorithm.

Ibatt is Ibatt battery current. Enable or Disable ThETA Ibatt Feature. **0: Disable**; 1: Enable.

Definition at line 140 of file CpuPowerDeliveryConfig.h.

The documentation for this struct was generated from the following file:

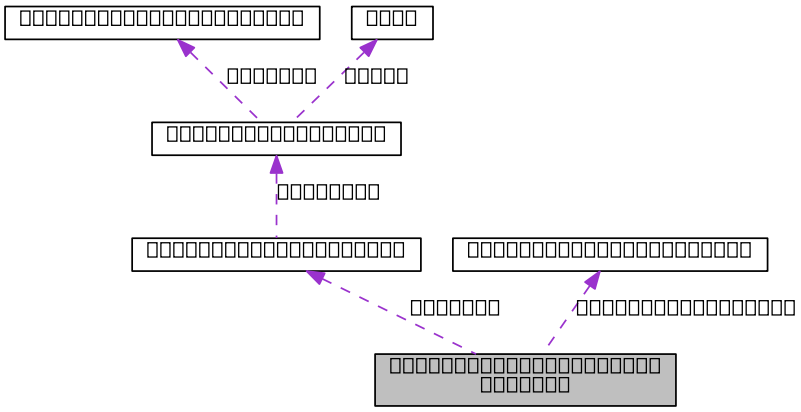
- CpuPowerDeliveryConfig.h

12.27 CPU_POWER_MGMT_CUSTOM_CONFIG Struct Reference

CPU Power Management Custom Configuration Structure.

`#include <CpuPowerMgmtCustomConfig.h>`

Collaboration diagram for CPU_POWER_MGMT_CUSTOM_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- PPM_CUSTOM_RATIO_TABLE CustomRatioTable

  Custom Processor Ratio Table Instance.

12.27.1 Detailed Description

CPU Power Management Custom Configuration Structure.

Revision 1:

- Initial version.

Definition at line 62 of file CpuPowerMgmtCustomConfig.h.

The documentation for this struct was generated from the following file:

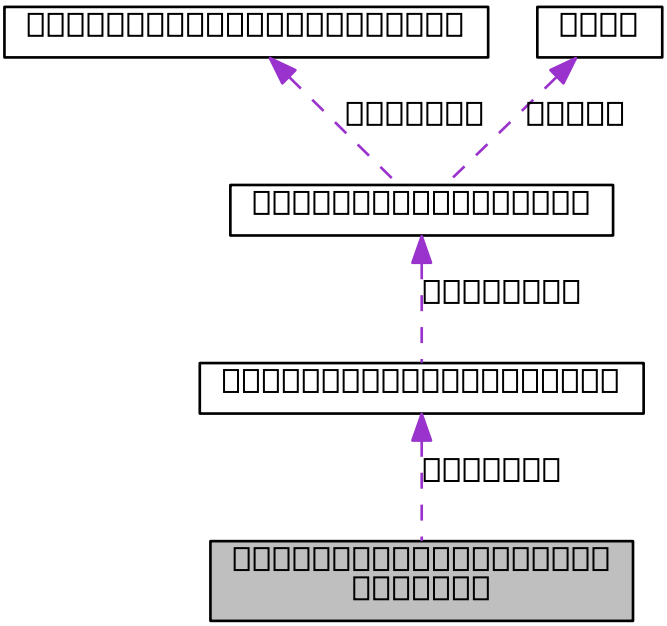
- CpuPowerMgmtCustomConfig.h

12.28 CPU_SECURITY_PREMEM_CONFIG Struct Reference

CPU Security PreMemory Configuration Structure.

`#include <CpuSecurityPreMemConfig.h>`

Collaboration diagram for CPU_SECURITY_PREMEM_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT16 BiosSize

  Flash information for BIOS Guard: BIOS Size in KB.
- UINT8 Reserved [2]

  Reserved for future use.
- UINT32 BiosGuard: 1

  *Enable or Disable BIOS Guard; 0: Disable; **1: Enable**.*
- UINT32 BiosGuardToolsInterface: 1

  *BIOS Guard Tools Interface; **0: Disable**, 1:Enable.*
- UINT32 Txt: 1

  *Enable or Disable Trusted Execution Technology; **0: Disable**; 1: Enable.*
- UINT32 SkipStopPbet: 1

  ***(Test)** Skip Stop PBET Timer; **0: Disable**; 1: Enable.*
- UINT32 ResetAux: 1

  ***(Test)** Reset Auxiliary content, **0: Disabled**, 1: Enabled*

- UINT32 TxtAcheckRequest: 1

  (Test) *AcheckRequest* ***0: Disabled****, 1: Enabled. When Enabled, it will call Acheck regardless of crashcode value*
- UINT32 TseEnable: 1

  Enable or Disable Total Storage Encryption (TSE) feature.
- UINT32 TdxEnable: 1

  Enable or Disable Trust Domain Extension (TDX)
- UINT32 GenerateNewTmeKey: 1

  Generate new TME key regardless of the type of reset.
- UINT32 TxtPowerdownRequest: 1

  (Test) *PowerRequest* ***0: Disabled****, 1: Enabled. When Enabled, it will set Powerdown request in TPM and reset the platform*
- UINT64 TmeExcludeBase

  TME exclude base.
- UINT64 TmeExcludeSize

  TME exclude size.

12.28.1 Detailed Description

CPU Security PreMemory Configuration Structure.

Revision 1:

- Initial version. **Revision 2**:
- Add TmeExcludeBase, TmeExcludeSize, GenerateNewTmeKey.

Definition at line 38 of file CpuSecurityPreMemConfig.h.

12.28.2 Member Data Documentation

12.28.2.1 BiosGuard

```
UINT32 CPU_SECURITY_PREMEM_CONFIG::BiosGuard
```

Enable or Disable BIOS Guard; 0: Disable; **1: Enable**.

- This is an optional feature and can be opted out.
- If this policy is set to Disabled, the policies in the BIOS_GUARD_CONFIG will be ignored.
- If PeiBiosGuardLibNull is used, this policy will have no effect.

Definition at line 48 of file CpuSecurityPreMemConfig.h.

12.28.2.2 GenerateNewTmeKey

```
UINT32 CPU_SECURITY_PREMEM_CONFIG::GenerateNewTmeKey
```

Generate new TME key regardless of the type of reset.

- **0: do not generate**
- 1: generate

Definition at line 78 of file CpuSecurityPreMemConfig.h.

12.28.2.3 TdxEnable

```
UINT32 CPU_SECURITY_PREMEM_CONFIG::TdxEnable
```

Enable or Disable Trust Domain Extension (TDX)

- 0: Disable (Default)
- 1: Enable

Definition at line 72 of file CpuSecurityPreMemConfig.h.

12.28.2.4 TseEnable

```
UINT32 CPU_SECURITY_PREMEM_CONFIG::TseEnable
```

Enable or Disable Total Storage Encryption (TSE) feature.

- **0: Disable**
- 1: Enable

Definition at line 66 of file CpuSecurityPreMemConfig.h.

12.28.2.5 Txt

```
UINT32 CPU_SECURITY_PREMEM_CONFIG::Txt
```

Enable or Disable Trusted Execution Technology; **0: Disable**; 1: Enable.

- This is an optional feature and can be opted out.
- If this policy is set to Disabled, the policies in the CPU_TXT_PREMEM_CONFIG will be ignored.
- If PeiTxtLibNull is used, this policy will have no effect.

Definition at line 56 of file CpuSecurityPreMemConfig.h.

The documentation for this struct was generated from the following file:

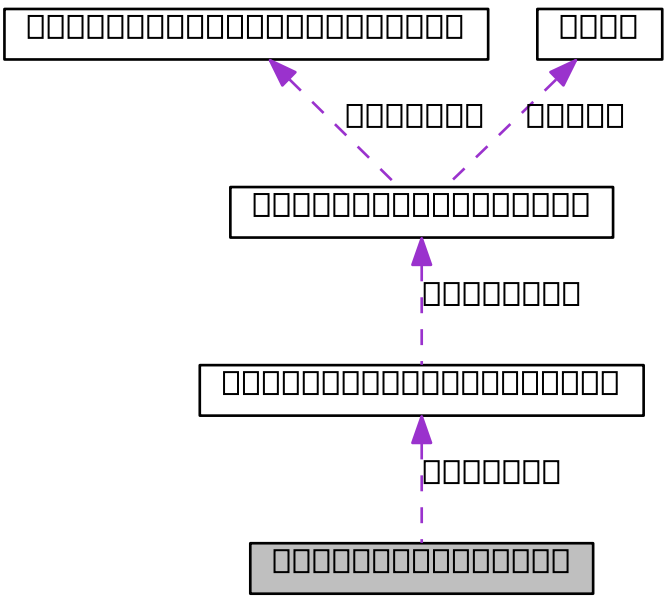
- CpuSecurityPreMemConfig.h

12.29 CPU_TEST_CONFIG Struct Reference

CPU Test Configuration Structure.

```
#include <CpuTestConfig.h>
```

Collaboration diagram for CPU_TEST_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT32 MlcStreamerPrefetcher: 1

  *Enable or Disable MLC Streamer Prefetcher; 0: Disable; **1: Enable**.*
- UINT32 MlcSpatialPrefetcher: 1

  *Enable or Disable MLC Spatial Prefetcher; 0: Disable; **1: Enable**.*
- UINT32 MonitorMwaitEnable: 1

  *Enable or Disable Monitor /MWAIT instructions; 0: Disable; **1: Enable**.*
- UINT32 MachineCheckEnable: 1

  *Enable or Disable initialization of machine check registers; 0: Disable; **1: Enable**.*
- UINT32 ProcessorTraceOutputScheme: 1

  *Control on Processor Trace output scheme; **0: Single Range Output**; 1: ToPA Output.*
- UINT32 ProcessorTraceEnable: 1

  *Enable or Disable Processor Trace feature; **0: Disable**; 1: Enable.*
- UINT32 ProcessorTraceBspOnly: 1

  *Processor Trace for all cores or BSP only; **0: all core**; 1: BSP only.*
- UINT32 ProcessorTraceTimingPacket: 1

  *Enable or Disable Processor Trace timing packet (CYC, TSC); **0: Disable**; 1: Enable.*
- UINT32 ThreeStrikeCounter: 1

  *Enable or Disable Three strike counter; 0: Disable; **1: Enable**.*
- UINT32 RsvdBits: 23

  Reserved for future use.
- UINT32 ProcessorTraceMemSize

  Memory size per thread for Processor Trace.

12.29.1 Detailed Description

CPU Test Configuration Structure.

Revision 1:

- Initial version.

Definition at line 36 of file CpuTestConfig.h.

12.29.2 Member Data Documentation

12.29.2.1 ProcessorTraceMemSize

```
UINT32 CPU_TEST_CONFIG::ProcessorTraceMemSize
```

Memory size per thread for Processor Trace.

Processor Trace requires 2^N alignment and size in bytes per thread, from 4KB to 128MB.

- **0xFF: Disabled**, 0:4k, 0x1:8k, 0x2:16k, 0x3:32k, 0x4:64k, 0x5:128k, 0x6:256k, 0x7:512k, 0x8:1M, 0x9:2M, 0xa:4M. 0xb:8M, 0xc:16M, 0xd:32M, 0xe:64M, 0xf:128M

Definition at line 54 of file CpuTestConfig.h.

The documentation for this struct was generated from the following file:

- CpuTestConfig.h

12.30 DDI_CONFIGURATION Struct Reference

This structure configures the Native GPIOs for DDI port per VBT settings.

```
#include <IGpuConfig.h>
```

Public Attributes

- UINT8 DdiPortBConfig

  The Configuration of DDI port A, this settings must match VBT's settings. DdiPortDisabled - No LFP is connected on DdiPortA, ***DdiPortEdp - Set DdiPortA to eDP***
- UINT8 DdiPortAHpd

  The Configuration of DDI port B, this settings must match VBT's settings. DdiPortDisabled - No LFP is connected on DdiPortB, ***DdiPortEdp - Set DdiPortB to eDP***
- UINT8 DdiPortBHpd

  The HPD setting of DDI Port A, this settings must match VBT's settings. ***DdiHpdDisable - Disable HPD****, DdiHpd↩ Enable - Enable HPD.*
- UINT8 DdiPortCHpd

  The HPD setting of DDI Port B, this settings must match VBT's settings. DdiHpdDisable - Disable HPD, ***DdiHpd↩ Enable - Enable HPD***
- UINT8 DdiPort1Hpd

  The HPD setting of DDI Port C, this settings must match VBT's settings. ***DdiHpdDisable - Disable HPD****, DdiHpd↩ Enable - Enable HPD.*
- UINT8 DdiPort2Hpd

  The HPD setting of DDI Port 1, this settings must match VBT's settings. ***DdiHpdDisable - Disable HPD****, DdiHpd↩ Enable - Enable HPD.*
- UINT8 DdiPort3Hpd

  The HPD setting of DDI Port 2, this settings must match VBT's settings. ***DdiHpdDisable - Disable HPD****, DdiHpd↩ Enable - Enable HPD.*
- UINT8 DdiPort4Hpd

  The HPD setting of DDI Port 3, this settings must match VBT's settings. ***DdiHpdDisable - Disable HPD****, DdiHpd↩ Enable - Enable HPD.*
- UINT8 DdiPortADdc

The HPD setting of DDI Port 4, this settings must match VBT's settings. ***DdiHpdDisable - Disable HPD****, DdiHpd↩ Enable - Enable HPD.*

- UINT8 DdiPortBDdc

  The DDC setting of DDI Port A, this settings must match VBT's settings. ***DdiDisable - Disable DDC****, DdiDdcEnable - Enable DDC.*

- UINT8 DdiPortCDdc

  The DDC setting of DDI Port B, this settings must match VBT's settings. DdiDisable - Disable DDC, ***DdiDdcEnable - Enable DDC***

- UINT8 DdiPort1Ddc

  The DDC setting of DDI Port C, this settings must match VBT's settings. ***DdiDisable - Disable DDC****, DdiDdcEnable - Enable DDC.*

- UINT8 DdiPort2Ddc

  The DDC setting of DDI Port 1, this settings must match VBT's settings. ***DdiDisable - Disable DDC****, DdiDdcEnable - Enable DDC.*

- UINT8 DdiPort3Ddc

  The DDC setting of DDI Port 2, this settings must match VBT's settings. ***DdiDisable - Disable DDC****, DdiDdcEnable - Enable DDC.*

- UINT8 DdiPort4Ddc

  The DDC setting of DDI Port 3, this settings must match VBT's settings. ***DdiDisable - Disable DDC****, DdiDdcEnable - Enable DDC.*

12.30.1 Detailed Description

This structure configures the Native GPIOs for DDI port per VBT settings.

Definition at line 136 of file IGpuConfig.h.

The documentation for this struct was generated from the following file:

- IGpuConfig.h

12.31 DMI_EQ_PARAM Struct Reference

Represent lane specific Dmi Gen3 equalization parameters.

`#include <PeiDmiConfig.h>`

Public Attributes

- UINT8 Cm

  Coefficient C-1.

- UINT8 Cp

  Coefficient C+1.

- UINT8 Rsvd0 [2]

  Reserved bytes.

12.31.1 Detailed Description

Represent lane specific Dmi Gen3 equalization parameters.

Definition at line 47 of file PeiDmiConfig.h.

The documentation for this struct was generated from the following file:

- PeiDmiConfig.h

12.32 DMI_HW_WIDTH_CONTROL Struct Reference

This structure allows to customize DMI HW Autonomous Width Control for Thermal and Mechanical spec design.

`#include <ThermalConfig.h>`

Public Attributes

- UINT32 DmiTsawEn: 1
  DMI Thermal Sensor Autonomous Width Enable.
- UINT32 SuggestedSetting: 1
  *0: Disable; **1: Enable** suggested representative values*
- UINT32 RsvdBits0: 6
  Reserved bits.
- UINT32 TS0TW: 3
  *Thermal Sensor 0 Target Width (**DmiThermSensWidthx8**)*
- UINT32 TS1TW: 3
  *Thermal Sensor 1 Target Width (**DmiThermSensWidthx4**)*
- UINT32 TS2TW: 3
  *Thermal Sensor 2 Target Width (**DmiThermSensWidthx2**)*
- UINT32 TS3TW: 3
  *Thermal Sensor 3 Target Width (**DmiThermSensWidthx1**)*
- UINT32 RsvdBits1: 12
  Reserved bits.

12.32.1 Detailed Description

This structure allows to customize DMI HW Autonomous Width Control for Thermal and Mechanical spec design.

When the SuggestedSetting is enabled, the customized values are ignored. Look at DMI_THERMAL_SENSOR_↩ TARGET_WIDTH for possible values

Definition at line 69 of file ThermalConfig.h.

The documentation for this struct was generated from the following file:

- ThermalConfig.h

12.33 DMI_LANE_CONFIG Struct Reference

The PCH_DMI_LANE_CONFIG block describes specific lane configuration.

`#include <PchPcieRpConfig.h>`

Public Attributes

- UINT32 DmiTranCoOverEn: 2
  Enable/Disable Lane Transmitter Coefficient.
- UINT32 Rsvd1: 6
  Reserved.
- UINT32 DmiTranCoOverPostCur: 6
  Lane Transmitter Post-Cursor Coefficient Override.
- UINT32 Rsvd2: 2
  Reserved.
- UINT32 DmiTranCoOverPreCur: 6
  Lane Transmitter Pre-Cursor Coefficient Override.
- UINT32 Rsvd3: 2
  Reserved.
- UINT32 DmiUpPortTranPreset: 4
  Upstream Port Lane Transmitter Preset.
- UINT32 Rsvd4: 4
  Reserved.

12.33.1 Detailed Description

The PCH_DMI_LANE_CONFIG block describes specific lane configuration.

Definition at line 183 of file PchPcieRpConfig.h.

The documentation for this struct was generated from the following file:

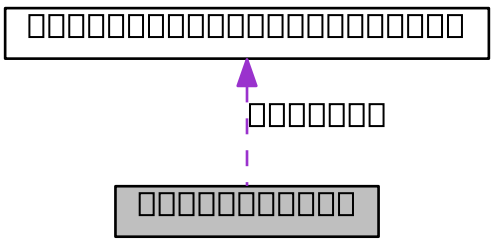
- PchPcieRpConfig.h

12.34 EFI_HOB_CPU Struct Reference

Describes processor information, such as address space and I/O space capabilities.

`#include <PiHob.h>`

Collaboration diagram for EFI_HOB_CPU:



Public Attributes

- EFI_HOB_GENERIC_HEADER Header

  The HOB generic header.
- UINT8 SizeOfMemorySpace

  Identifies the maximum physical memory addressability of the processor.
- UINT8 SizeOfIoSpace

  Identifies the maximum physical I/O addressability of the processor.
- UINT8 Reserved [6]

  This field will always be set to zero.

12.34.1 Detailed Description

Describes processor information, such as address space and I/O space capabilities.

Definition at line 440 of file PiHob.h.

12.34.2 Member Data Documentation

12.34.2.1 Header

```
EFI_HOB_GENERIC_HEADER EFI_HOB_CPU::Header
```

The HOB generic header.

Header.HobType = EFI_HOB_TYPE_CPU.

Definition at line 444 of file PiHob.h.

The documentation for this struct was generated from the following file:

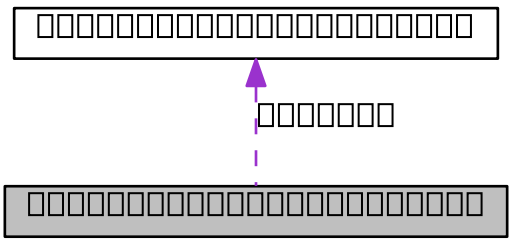
- PiHob.h

12.35 EFI_HOB_FIRMWARE_VOLUME Struct Reference

Details the location of firmware volumes that contain firmware files.

```
#include <PiHob.h>
```

Collaboration diagram for EFI_HOB_FIRMWARE_VOLUME:



Public Attributes

- EFI_HOB_GENERIC_HEADER Header

  The HOB generic header.
- EFI_PHYSICAL_ADDRESS BaseAddress

  The physical memory-mapped base address of the firmware volume.
- UINT64 Length

  The length in bytes of the firmware volume.

12.35.1 Detailed Description

Details the location of firmware volumes that contain firmware files.

Definition at line 357 of file PiHob.h.

12.35.2 Member Data Documentation

12.35.2.1 Header

```
EFI_HOB_GENERIC_HEADER EFI_HOB_FIRMWARE_VOLUME::Header
```

The HOB generic header.

Header.HobType = EFI_HOB_TYPE_FV.

Definition at line 361 of file PiHob.h.

The documentation for this struct was generated from the following file:

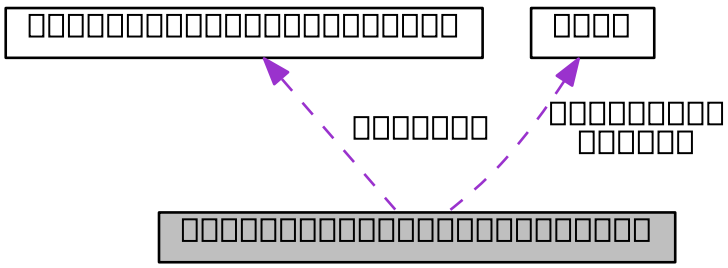
- PiHob.h

12.36 EFI_HOB_FIRMWARE_VOLUME2 Struct Reference

Details the location of a firmware volume that was extracted from a file within another firmware volume.

```
#include <PiHob.h>
```

Collaboration diagram for EFI_HOB_FIRMWARE_VOLUME2:



Public Attributes

- EFI_HOB_GENERIC_HEADER Header

  The HOB generic header.
- EFI_PHYSICAL_ADDRESS BaseAddress

  The physical memory-mapped base address of the firmware volume.
- UINT64 Length

  The length in bytes of the firmware volume.
- EFI_GUID FvName

  The name of the firmware volume.
- EFI_GUID FileName

  The name of the firmware file that contained this firmware volume.

12.36.1 Detailed Description

Details the location of a firmware volume that was extracted from a file within another firmware volume.

Definition at line 376 of file PiHob.h.

12.36.2 Member Data Documentation

12.36.2.1 Header

```
EFI_HOB_GENERIC_HEADER EFI_HOB_FIRMWARE_VOLUME2::Header
```

The HOB generic header.

Header.HobType = EFI_HOB_TYPE_FV2.

Definition at line 380 of file PiHob.h.

The documentation for this struct was generated from the following file:

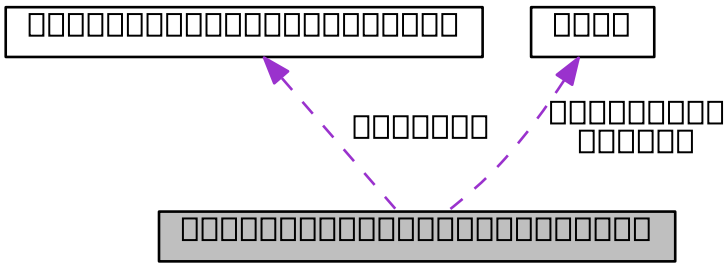
- PiHob.h

12.37 EFI_HOB_FIRMWARE_VOLUME3 Struct Reference

Details the location of a firmware volume that was extracted from a file within another firmware volume.

`#include <PiHob.h>`

Collaboration diagram for EFI_HOB_FIRMWARE_VOLUME3:



Public Attributes

- EFI_HOB_GENERIC_HEADER Header
  The HOB generic header.
- EFI_PHYSICAL_ADDRESS BaseAddress
  The physical memory-mapped base address of the firmware volume.
- UINT64 Length
  The length in bytes of the firmware volume.
- UINT32 AuthenticationStatus
  The authentication status.
- BOOLEAN ExtractedFv
  TRUE if the FV was extracted as a file within another firmware volume.
- EFI_GUID FvName
  The name of the firmware volume.
- EFI_GUID FileName
  The name of the firmware file that contained this firmware volume.

12.37.1 Detailed Description

Details the location of a firmware volume that was extracted from a file within another firmware volume.

Definition at line 403 of file PiHob.h.

12.37.2 Member Data Documentation

12.37.2.1 ExtractedFv

```
BOOLEAN EFI_HOB_FIRMWARE_VOLUME3::ExtractedFv
```

TRUE if the FV was extracted as a file within another firmware volume.

FALSE otherwise.

Definition at line 424 of file PiHob.h.

12.37.2.2 FileName

```
EFI_GUID EFI_HOB_FIRMWARE_VOLUME3::FileName
```

The name of the firmware file that contained this firmware volume.

Valid only if IsExtractedFv is TRUE.

Definition at line 434 of file PiHob.h.

12.37.2.3 FvName

```
EFI_GUID EFI_HOB_FIRMWARE_VOLUME3::FvName
```

The name of the firmware volume.

Valid only if IsExtractedFv is TRUE.

Definition at line 429 of file PiHob.h.

12.37.2.4 Header

```
EFI_HOB_GENERIC_HEADER EFI_HOB_FIRMWARE_VOLUME3::Header
```

The HOB generic header.

Header.HobType = EFI_HOB_TYPE_FV3.

Definition at line 407 of file PiHob.h.

The documentation for this struct was generated from the following file:

- PiHob.h

12.38 EFI_HOB_GENERIC_HEADER Struct Reference

Describes the format and size of the data inside the HOB.

`#include <PiHob.h>`

Public Attributes

- UINT16 HobType

  Identifies the HOB data structure type.
- UINT16 HobLength

  The length in bytes of the HOB.
- UINT32 Reserved

  This field must always be set to zero.

12.38.1 Detailed Description

Describes the format and size of the data inside the HOB.

All HOBs must contain this generic HOB header.

Definition at line 36 of file PiHob.h.

The documentation for this struct was generated from the following file:

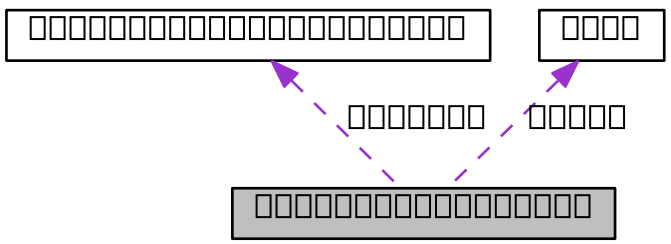
- PiHob.h

12.39 EFI_HOB_GUID_TYPE Struct Reference

Allows writers of executable content in the HOB producer phase to maintain and manage HOBs with specific GUID.

`#include <PiHob.h>`

Collaboration diagram for EFI_HOB_GUID_TYPE:



Public Attributes

- EFI_HOB_GENERIC_HEADER Header
  The HOB generic header.
- EFI_GUID Name
  A GUID that defines the contents of this HOB.

12.39.1 Detailed Description

Allows writers of executable content in the HOB producer phase to maintain and manage HOBs with specific GUID.

Definition at line 340 of file PiHob.h.

12.39.2 Member Data Documentation

12.39.2.1 Header

```
EFI_HOB_GENERIC_HEADER EFI_HOB_GUID_TYPE::Header
```

The HOB generic header.

Header.HobType = EFI_HOB_TYPE_GUID_EXTENSION.

Definition at line 344 of file PiHob.h.

The documentation for this struct was generated from the following file:

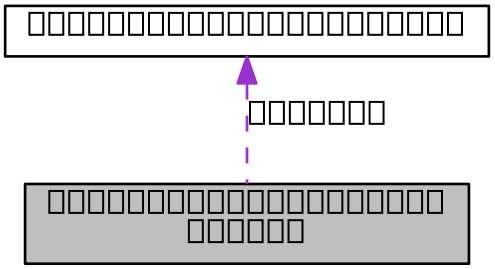
- PiHob.h

12.40 EFI_HOB_HANDOFF_INFO_TABLE Struct Reference

Contains general state information used by the HOB producer phase.

```
#include <PiHob.h>
```

Collaboration diagram for EFI_HOB_HANDOFF_INFO_TABLE:



Public Attributes

- EFI_HOB_GENERIC_HEADER Header

  The HOB generic header.
- UINT32 Version

  The version number pertaining to the PHIT HOB definition.
- EFI_BOOT_MODE BootMode

  The system boot mode as determined during the HOB producer phase.
- EFI_PHYSICAL_ADDRESS EfiMemoryTop

  The highest address location of memory that is allocated for use by the HOB producer phase.
- EFI_PHYSICAL_ADDRESS EfiMemoryBottom

  The lowest address location of memory that is allocated for use by the HOB producer phase.
- EFI_PHYSICAL_ADDRESS EfiFreeMemoryTop

  The highest address location of free memory that is currently available for use by the HOB producer phase.
- EFI_PHYSICAL_ADDRESS EfiFreeMemoryBottom

  The lowest address location of free memory that is available for use by the HOB producer phase.
- EFI_PHYSICAL_ADDRESS EfiEndOfHobList

  The end of the HOB list.

12.40.1 Detailed Description

Contains general state information used by the HOB producer phase.

This HOB must be the first one in the HOB list.

Definition at line 60 of file PiHob.h.

12.40.2 Member Data Documentation

12.40.2.1 EfiMemoryTop

```
EFI_PHYSICAL_ADDRESS EFI_HOB_HANDOFF_INFO_TABLE::EfiMemoryTop
```

The highest address location of memory that is allocated for use by the HOB producer phase.

This address must be 4-KB aligned to meet page restrictions of UEFI.

Definition at line 79 of file PiHob.h.

12.40.2.2 Header

```
EFI_HOB_GENERIC_HEADER EFI_HOB_HANDOFF_INFO_TABLE::Header
```

The HOB generic header.

Header.HobType = EFI_HOB_TYPE_HANDOFF.

Definition at line 64 of file PiHob.h.

12.40.2.3 Version

```
UINT32 EFI_HOB_HANDOFF_INFO_TABLE::Version
```

The version number pertaining to the PHIT HOB definition.

This value is four bytes in length to provide an 8-byte aligned entry when it is combined with the 4-byte BootMode.

Definition at line 70 of file PiHob.h.

The documentation for this struct was generated from the following file:

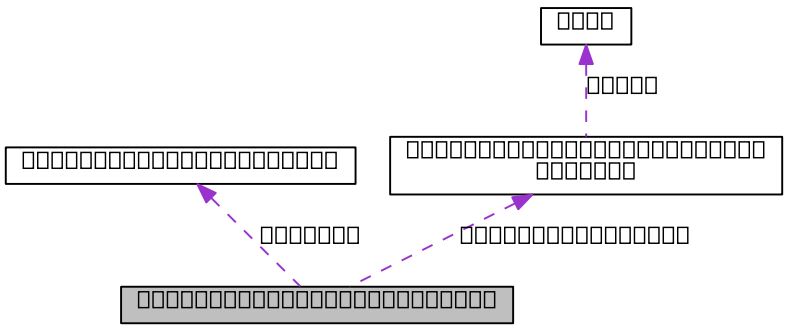
- PiHob.h

12.41 EFI_HOB_MEMORY_ALLOCATION Struct Reference

Describes all memory ranges used during the HOB producer phase that exist outside the HOB list.

```
#include <PiHob.h>
```

Collaboration diagram for EFI_HOB_MEMORY_ALLOCATION:



Public Attributes

- EFI_HOB_GENERIC_HEADER Header

  The HOB generic header.
- EFI_HOB_MEMORY_ALLOCATION_HEADER AllocDescriptor

  An instance of the EFI_HOB_MEMORY_ALLOCATION_HEADER that describes the various attributes of the logical memory allocation.

12.41.1 Detailed Description

Describes all memory ranges used during the HOB producer phase that exist outside the HOB list.

This HOB type describes how memory is used, not the physical attributes of memory.

Definition at line 144 of file PiHob.h.

12.41.2 Member Data Documentation

12.41.2.1 Header

```
EFI_HOB_GENERIC_HEADER EFI_HOB_MEMORY_ALLOCATION::Header
```

The HOB generic header.

Header.HobType = EFI_HOB_TYPE_MEMORY_ALLOCATION.

Definition at line 148 of file PiHob.h.

The documentation for this struct was generated from the following file:

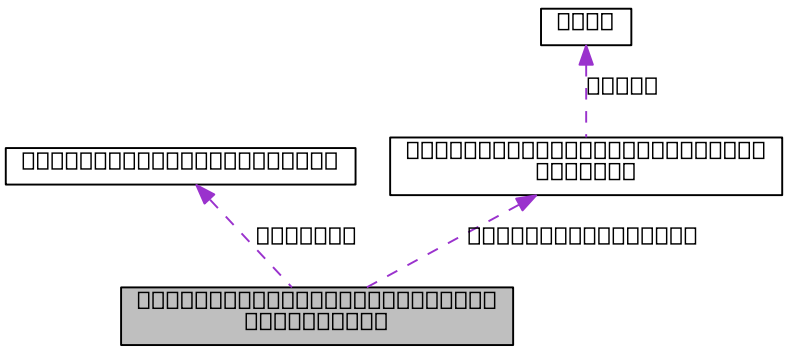
- PiHob.h

12.42 EFI_HOB_MEMORY_ALLOCATION_BSP_STORE Struct Reference

Defines the location of the boot-strap processor (BSP) BSPStore ("Backing Store Pointer Store").

```
#include <PiHob.h>
```

Collaboration diagram for EFI_HOB_MEMORY_ALLOCATION_BSP_STORE:



Public Attributes

- EFI_HOB_GENERIC_HEADER Header

  The HOB generic header.
- EFI_HOB_MEMORY_ALLOCATION_HEADER AllocDescriptor

  An instance of the EFI_HOB_MEMORY_ALLOCATION_HEADER that describes the various attributes of the logical memory allocation.

12.42.1 Detailed Description

Defines the location of the boot-strap processor (BSP) BSPStore ("Backing Store Pointer Store").

This HOB is valid for the Itanium processor family only register overflow store.

Definition at line 183 of file PiHob.h.

12.42.2 Member Data Documentation

12.42.2.1 Header

```
EFI_HOB_GENERIC_HEADER EFI_HOB_MEMORY_ALLOCATION_BSP_STORE::Header
```

The HOB generic header.

Header.HobType = EFI_HOB_TYPE_MEMORY_ALLOCATION.

Definition at line 187 of file PiHob.h.

The documentation for this struct was generated from the following file:

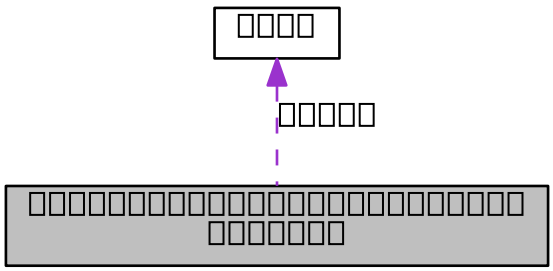
- PiHob.h

12.43 EFI_HOB_MEMORY_ALLOCATION_HEADER Struct Reference

EFI_HOB_MEMORY_ALLOCATION_HEADER describes the various attributes of the logical memory allocation.

```
#include <PiHob.h>
```

Collaboration diagram for EFI_HOB_MEMORY_ALLOCATION_HEADER:



Public Attributes

- EFI_GUID Name

  A GUID that defines the memory allocation region's type and purpose, as well as other fields within the memory allocation HOB.
- EFI_PHYSICAL_ADDRESS MemoryBaseAddress

  The base address of memory allocated by this HOB.
- UINT64 MemoryLength

  The length in bytes of memory allocated by this HOB.
- EFI_MEMORY_TYPE MemoryType

  Defines the type of memory allocated by this HOB.
- UINT8 Reserved [4]

  Padding for Itanium processor family.

12.43.1 Detailed Description

EFI_HOB_MEMORY_ALLOCATION_HEADER describes the various attributes of the logical memory allocation.

The type field will be used for subsequent inclusion in the UEFI memory map.

Definition at line 104 of file PiHob.h.

12.43.2 Member Data Documentation

12.43.2.1 MemoryBaseAddress

```
EFI_PHYSICAL_ADDRESS EFI_HOB_MEMORY_ALLOCATION_HEADER::MemoryBaseAddress
```

The base address of memory allocated by this HOB.

Type EFI_PHYSICAL_ADDRESS is defined in AllocatePages() in the UEFI 2.0 specification.

Definition at line 119 of file PiHob.h.

12.43.2.2 MemoryType

```
EFI_MEMORY_TYPE EFI_HOB_MEMORY_ALLOCATION_HEADER::MemoryType
```

Defines the type of memory allocated by this HOB.

The memory type definition follows the EFI_MEMORY_TYPE definition. Type EFI_MEMORY_TYPE is defined in AllocatePages() in the UEFI 2.0 specification.

Definition at line 131 of file PiHob.h.

12.43.2.3 Name

```
EFI_GUID EFI_HOB_MEMORY_ALLOCATION_HEADER::Name
```

A GUID that defines the memory allocation region's type and purpose, as well as other fields within the memory allocation HOB.

This GUID is used to define the additional data within the HOB that may be present for the memory allocation HOB. Type EFI_GUID is defined in InstallProtocolInterface() in the UEFI 2.0 specification.

Definition at line 112 of file PiHob.h.

The documentation for this struct was generated from the following file:

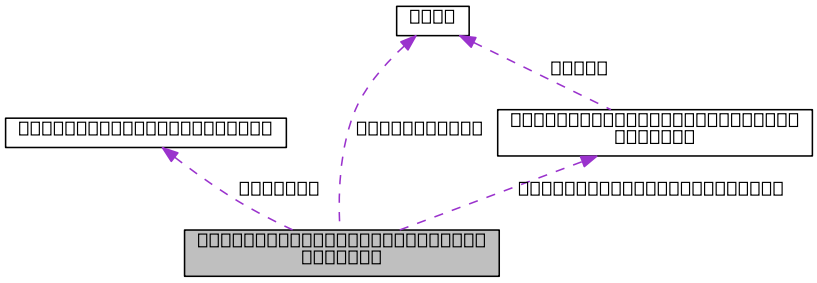
- PiHob.h

12.44 EFI_HOB_MEMORY_ALLOCATION_MODULE Struct Reference

Defines the location and entry point of the HOB consumer phase.

```
#include <PiHob.h>
```

Collaboration diagram for EFI_HOB_MEMORY_ALLOCATION_MODULE:



Public Attributes

- EFI_HOB_GENERIC_HEADER Header

  The HOB generic header.
- EFI_HOB_MEMORY_ALLOCATION_HEADER MemoryAllocationHeader

  An instance of the EFI_HOB_MEMORY_ALLOCATION_HEADER that describes the various attributes of the logical memory allocation.
- EFI_GUID ModuleName

  The GUID specifying the values of the firmware file system name that contains the HOB consumer phase component.
- EFI_PHYSICAL_ADDRESS EntryPoint

  The address of the memory-mapped firmware volume that contains the HOB consumer phase firmware file.

12.44.1 Detailed Description

Defines the location and entry point of the HOB consumer phase.

Definition at line 198 of file PiHob.h.

12.44.2 Member Data Documentation

12.44.2.1 Header

```
EFI_HOB_GENERIC_HEADER EFI_HOB_MEMORY_ALLOCATION_MODULE::Header
```

The HOB generic header.

Header.HobType = EFI_HOB_TYPE_MEMORY_ALLOCATION.

Definition at line 202 of file PiHob.h.

The documentation for this struct was generated from the following file:

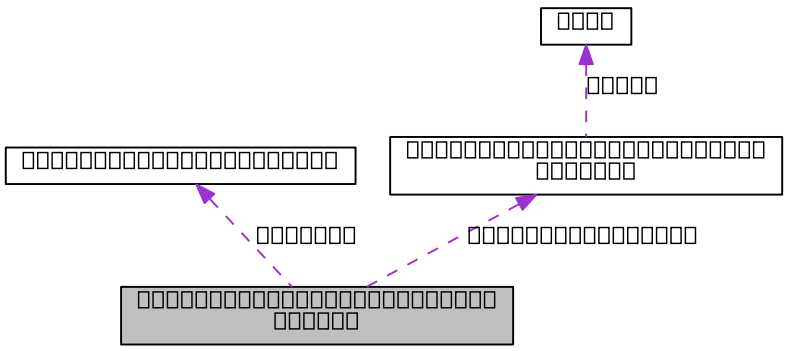
- PiHob.h

12.45 EFI_HOB_MEMORY_ALLOCATION_STACK Struct Reference

Describes the memory stack that is produced by the HOB producer phase and upon which all post-memory-installed executable content in the HOB producer phase is executing.

```
#include <PiHob.h>
```

Collaboration diagram for EFI_HOB_MEMORY_ALLOCATION_STACK:



Public Attributes

- EFI_HOB_GENERIC_HEADER Header

  The HOB generic header.
- EFI_HOB_MEMORY_ALLOCATION_HEADER AllocDescriptor

  An instance of the EFI_HOB_MEMORY_ALLOCATION_HEADER that describes the various attributes of the logical memory allocation.

12.45.1 Detailed Description

Describes the memory stack that is produced by the HOB producer phase and upon which all post-memory-installed executable content in the HOB producer phase is executing.

Definition at line 165 of file PiHob.h.

12.45.2 Member Data Documentation

12.45.2.1 Header

```
EFI_HOB_GENERIC_HEADER EFI_HOB_MEMORY_ALLOCATION_STACK::Header
```

The HOB generic header.

Header.HobType = EFI_HOB_TYPE_MEMORY_ALLOCATION.

Definition at line 169 of file PiHob.h.

The documentation for this struct was generated from the following file:

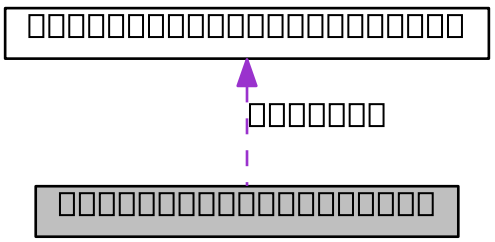
- PiHob.h

12.46 EFI_HOB_MEMORY_POOL Struct Reference

Describes pool memory allocations.

```
#include <PiHob.h>
```

Collaboration diagram for EFI_HOB_MEMORY_POOL:



Public Attributes

- EFI_HOB_GENERIC_HEADER Header

  The HOB generic header.

12.46.1 Detailed Description

Describes pool memory allocations.

Definition at line 462 of file PiHob.h.

12.46.2 Member Data Documentation

12.46.2.1 Header

```
EFI_HOB_GENERIC_HEADER EFI_HOB_MEMORY_POOL::Header
```

The HOB generic header.

Header.HobType = EFI_HOB_TYPE_MEMORY_POOL.

Definition at line 466 of file PiHob.h.

The documentation for this struct was generated from the following file:

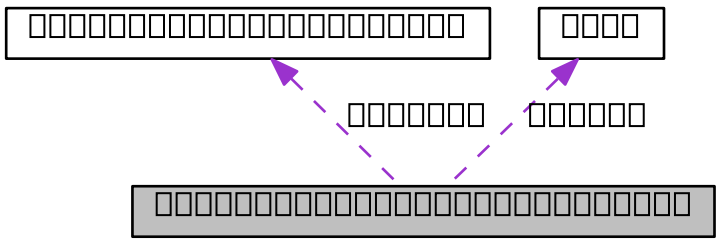
- PiHob.h

12.47 EFI_HOB_RESOURCE_DESCRIPTOR Struct Reference

Describes the resource properties of all fixed, nonrelocatable resource ranges found on the processor host bus during the HOB producer phase.

```
#include <PiHob.h>
```

Collaboration diagram for EFI_HOB_RESOURCE_DESCRIPTOR:



Public Attributes

- EFI_HOB_GENERIC_HEADER Header

  The HOB generic header.
- EFI_GUID Owner

  A GUID representing the owner of the resource.
- EFI_RESOURCE_TYPE ResourceType

  The resource type enumeration as defined by EFI_RESOURCE_TYPE.
- EFI_RESOURCE_ATTRIBUTE_TYPE ResourceAttribute

  Resource attributes as defined by EFI_RESOURCE_ATTRIBUTE_TYPE.
- EFI_PHYSICAL_ADDRESS PhysicalStart

  The physical start address of the resource region.
- UINT64 ResourceLength

  The number of bytes of the resource region.

12.47.1 Detailed Description

Describes the resource properties of all fixed, nonrelocatable resource ranges found on the processor host bus during the HOB producer phase.

Definition at line 308 of file PiHob.h.

12.47.2 Member Data Documentation

12.47.2.1 Header

```
EFI_HOB_GENERIC_HEADER EFI_HOB_RESOURCE_DESCRIPTOR::Header
```

The HOB generic header.

Header.HobType = EFI_HOB_TYPE_RESOURCE_DESCRIPTOR.

Definition at line 312 of file PiHob.h.

12.47.2.2 Owner

```
EFI_GUID EFI_HOB_RESOURCE_DESCRIPTOR::Owner
```

A GUID representing the owner of the resource.

This GUID is used by HOB consumer phase components to correlate device ownership of a resource.

Definition at line 317 of file PiHob.h.

The documentation for this struct was generated from the following file:

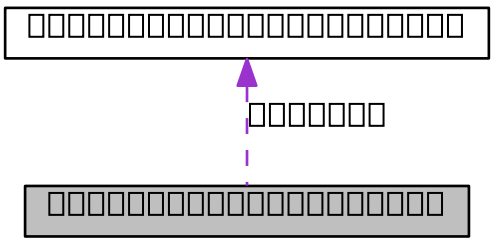
- PiHob.h

12.48 EFI_HOB_UEFI_CAPSULE Struct Reference

Each UEFI capsule HOB details the location of a UEFI capsule.

`#include <PiHob.h>`

Collaboration diagram for EFI_HOB_UEFI_CAPSULE:



Public Attributes

- EFI_HOB_GENERIC_HEADER Header

  The HOB generic header where Header.HobType = EFI_HOB_TYPE_UEFI_CAPSULE.
- EFI_PHYSICAL_ADDRESS BaseAddress

  The physical memory-mapped base address of an UEFI capsule.

12.48.1 Detailed Description

Each UEFI capsule HOB details the location of a UEFI capsule.

It includes a base address and length which is based upon memory blocks with a EFI_CAPSULE_HEADER and the associated CapsuleImageSize-based payloads. These HOB's shall be created by the PEI PI firmware sometime after the UEFI UpdateCapsule service invocation with the CAPSULE_FLAGS_POPULATE_SYSTEM_TABLE flag set in the EFI_CAPSULE_HEADER.

Definition at line 476 of file PiHob.h.

12.48.2 Member Data Documentation

12.48.2.1 BaseAddress

```
EFI_PHYSICAL_ADDRESS EFI_HOB_UEFI_CAPSULE::BaseAddress
```

The physical memory-mapped base address of an UEFI capsule.

This value is set to point to the base of the contiguous memory of the UEFI capsule. The length of the contiguous memory in bytes.

Definition at line 487 of file PiHob.h.

The documentation for this struct was generated from the following file:

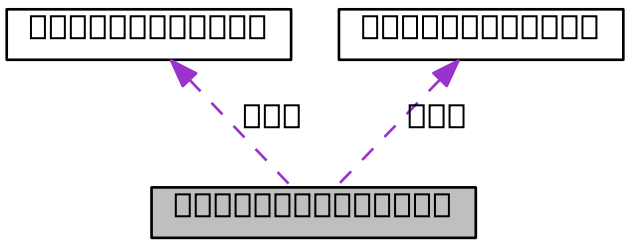
- PiHob.h

12.49 EFI_IP_ADDRESS Union Reference

16-byte buffer aligned on a 4-byte boundary.

```
#include <UefiBaseType.h>
```

Collaboration diagram for EFI_IP_ADDRESS:



12.49.1 Detailed Description

16-byte buffer aligned on a 4-byte boundary.

An IPv4 or IPv6 internet protocol address.

Definition at line 103 of file UefiBaseType.h.

The documentation for this union was generated from the following file:

- UefiBaseType.h

12.50 EFI_MAC_ADDRESS Struct Reference

32-byte buffer containing a network Media Access Control address.

`#include <UefiBaseType.h>`

12.50.1 Detailed Description

32-byte buffer containing a network Media Access Control address.

Definition at line 95 of file UefiBaseType.h.

The documentation for this struct was generated from the following file:

- UefiBaseType.h

12.51 EFI_MMRAM_DESCRIPTOR Struct Reference

Structure describing a MMRAM region and its accessibility attributes.

`#include <PiMultiPhase.h>`

Public Attributes

- EFI_PHYSICAL_ADDRESS PhysicalStart
  Designates the physical address of the MMRAM in memory.
- EFI_PHYSICAL_ADDRESS CpuStart
  Designates the address of the MMRAM, as seen by software executing on the processors.
- UINT64 PhysicalSize
  Describes the number of bytes in the MMRAM region.
- UINT64 RegionState
  Describes the accessibility attributes of the MMRAM.

12.51.1 Detailed Description

Structure describing a MMRAM region and its accessibility attributes.

Definition at line 116 of file PiMultiPhase.h.

12.51.2 Member Data Documentation

12.51.2.1 CpuStart

```
EFI_PHYSICAL_ADDRESS EFI_MMRAM_DESCRIPTOR::CpuStart
```

Designates the address of the MMRAM, as seen by software executing on the processors.

This address may or may not match PhysicalStart.

Definition at line 127 of file PiMultiPhase.h.

12.51.2.2 PhysicalStart

```
EFI_PHYSICAL_ADDRESS EFI_MMRAM_DESCRIPTOR::PhysicalStart
```

Designates the physical address of the MMRAM in memory.

This view of memory is the same as seen by I/O-based agents, for example, but it may not be the address seen by the processors.

Definition at line 122 of file PiMultiPhase.h.

12.51.2.3 RegionState

```
UINT64 EFI_MMRAM_DESCRIPTOR::RegionState
```

Describes the accessibility attributes of the MMRAM.

These attributes include the hardware state (e.g., Open/Closed/Locked), capability (e.g., cacheable), logical allocation (e.g., allocated), and pre-use initialization (e.g., needs testing/ECC initialization).

Definition at line 138 of file PiMultiPhase.h.

The documentation for this struct was generated from the following file:

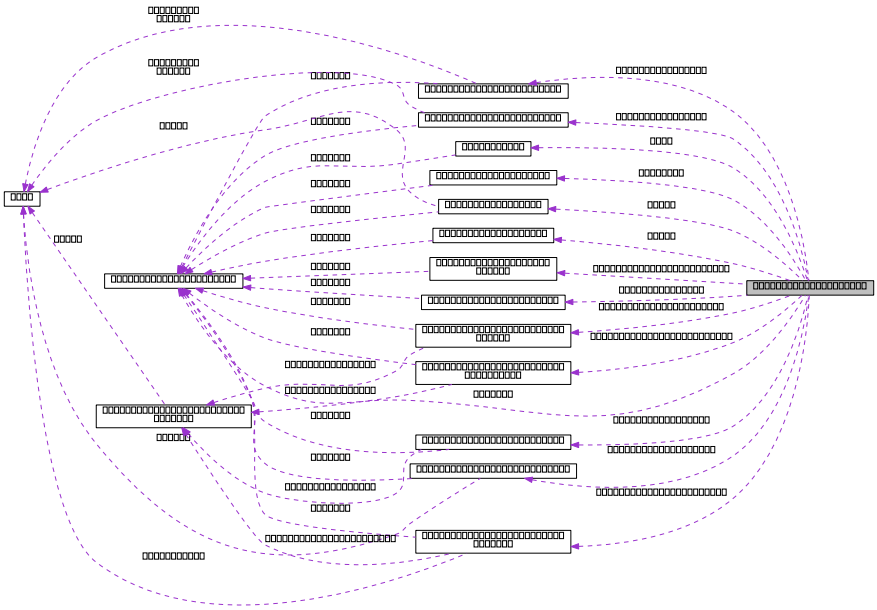
- PiMultiPhase.h

12.52 EFI_PEI_HOB_POINTERS Union Reference

Union of all the possible HOB Types.

`#include <PiHob.h>`

Collaboration diagram for EFI_PEI_HOB_POINTERS:



12.52.1 Detailed Description

Union of all the possible HOB Types.

Definition at line 494 of file PiHob.h.

The documentation for this union was generated from the following file:

- PiHob.h

12.53 EFI_TIME Struct Reference

EFI Time Abstraction: Year: 1900 - 9999 Month: 1 - 12 Day: 1 - 31 Hour: 0 - 23 Minute: 0 - 59 Second: 0 - 59 Nanosecond: 0 - 999,999,999 TimeZone: -1440 to 1440 or 2047.

`#include <UefiBaseType.h>`

12.53.1 Detailed Description

EFI Time Abstraction: Year: 1900 - 9999 Month: 1 - 12 Day: 1 - 31 Hour: 0 - 23 Minute: 0 - 59 Second: 0 - 59 Nanosecond: 0 - 999,999,999 TimeZone: -1440 to 1440 or 2047.

Definition at line 68 of file UefiBaseType.h.

The documentation for this struct was generated from the following file:

- UefiBaseType.h

12.54 FIVR_EXT_RAIL_CONFIG Struct Reference

Structure for V1p05/Vnn VR rail configuration.

`#include <FivrConfig.h>`

Public Attributes

- UINT32 EnabledStates: 6

  *Mask to enable the usage of external VR rail in specific S0ix or Sx states Use values from FIVR_RAIL_SX_STATE The default is **FivrRailDisabled**.*
- UINT32 Voltage: 11

  VR rail voltage value that will be used in S0i2/S0i3 states.
- UINT32 RsvdBits1: 15

  UINT32 Alignement.
- UINT32 CtrlRampTmr: 8

  This register holds the control hold off values to be used when changing the rail control for external bypass value in us.
- UINT32 SupportedVoltageStates: 4

  Mask to set the supported configuration in VR rail.
- UINT32 IccMaximum: 16

  *VR rail Icc Maximum Value Granularity of this setting is 1mA and maximal possible value is 500mA The default is **0mA** .*
- UINT32 RsvdBits2: 4

  UINT32 Alignment.

12.54.1 Detailed Description

Structure for V1p05/Vnn VR rail configuration.

Definition at line 55 of file FivrConfig.h.

12.54.2 Member Data Documentation

12.54.2.1 SupportedVoltageStates

```
UINT32 FIVR_EXT_RAIL_CONFIG::SupportedVoltageStates
```

Mask to set the supported configuration in VR rail.

Use values from FIVR_RAIL_SUPPORTED_VOLTAGE

Definition at line 83 of file FivrConfig.h.

12.54.2.2 Voltage

```
UINT32 FIVR_EXT_RAIL_CONFIG::Voltage
```

VR rail voltage value that will be used in S0i2/S0i3 states.

This value is given in 2.5mV increments (0=0mV, 1=2.5mV, 2=5mV...) The default for Vnn is set to **420 - 1050 mV**.

Definition at line 68 of file FivrConfig.h.

The documentation for this struct was generated from the following file:

- FivrConfig.h

12.55 FIVR_VCCIN_AUX_CONFIG Struct Reference

Structure for VCCIN_AUX voltage rail configuration.

```
#include <FivrConfig.h>
```

Public Attributes

- UINT8 LowToHighCurModeVolTranTime

  Transition time in microseconds from Low Current Mode Voltage to High Current Mode Voltage.
- UINT8 RetToHighCurModeVolTranTime

  Transition time in microseconds from Retention Mode Voltage to High Current Mode Voltage.
- UINT8 RetToLowCurModeVolTranTime

  Transition time in microseconds from Retention Mode Voltage to Low Current Mode Voltage.
- UINT32 OffToHighCurModeVolTranTime: 11

  Transition time in microseconds from Off (0V) to High Current Mode Voltage.

12.55.1 Detailed Description

Structure for VCCIN_AUX voltage rail configuration.

Definition at line 102 of file FivrConfig.h.

12.55.2 Member Data Documentation

12.55.2.1 LowToHighCurModeVolTranTime

```
UINT8 FIVR_VCCIN_AUX_CONFIG::LowToHighCurModeVolTranTime
```

Transition time in microseconds from Low Current Mode Voltage to High Current Mode Voltage.

Voltage transition time required by motherboard voltage regulator when PCH changes the VCCIN_AUX regulator set point from the low current mode voltage and high current mode voltage. This field has 1us resolution. When value is 0 PCH will not transition VCCIN_AUX to low current mode voltage. The default is **0xC** .

Definition at line 111 of file FivrConfig.h.

12.55.2.2 OffToHighCurModeVolTranTime

```
UINT32 FIVR_VCCIN_AUX_CONFIG::OffToHighCurModeVolTranTime
```

Transition time in microseconds from Off (0V) to High Current Mode Voltage.

Voltage transition time required by motherboard voltage regulator when PCH changes the VCCIN_AUX regulator set point from 0V to the high current mode voltage. This field has 1us resolution. 0 = Transition to 0V is disabled Setting this field to 0 sets VCCIN_AUX as a fixed rail that stays on in all S0 & Sx power states after initial start up on G3 exit The default is **0x96** .

Definition at line 143 of file FivrConfig.h.

12.55.2.3 RetToHighCurModeVolTranTime

```
UINT8 FIVR_VCCIN_AUX_CONFIG::RetToHighCurModeVolTranTime
```

Transition time in microseconds from Retention Mode Voltage to High Current Mode Voltage.

Voltage transition time required by motherboard voltage regulator when PCH changes the VCCIN_AUX regulator set point from the retention mode voltage to high current mode voltage. This field has 1us resolution. When value is 0 PCH will not transition VCCIN_AUX to retention voltage. The default is **0x36** .

Definition at line 121 of file FivrConfig.h.

12.55.2.4 RetToLowCurModeVolTranTime

```
UINT8 FIVR_VCCIN_AUX_CONFIG::RetToLowCurModeVolTranTime
```

Transition time in microseconds from Retention Mode Voltage to Low Current Mode Voltage.

Voltage transition time required by motherboard voltage regulator when PCH changes the VCCIN_AUX regulator set point from the retention mode voltage to low current mode voltage. This field has 1us resolution. When value is 0 PCH will not transition VCCIN_AUX to retention voltage. The default is **0x2B** .

Definition at line 131 of file FivrConfig.h.

The documentation for this struct was generated from the following file:

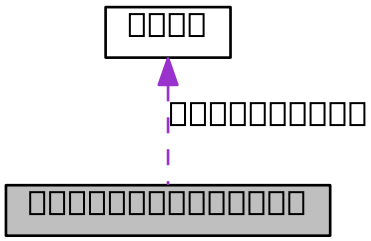
- FivrConfig.h

12.56 FSP_ERROR_INFO Struct Reference

FSP Error Information Block.

```
#include <FspErrorInfo.h>
```

Collaboration diagram for FSP_ERROR_INFO:



Public Attributes

- EFI_GUID ErrorType

  A GUID identifying the nature of the fatal error.
- EFI_STATUS Status

  A code describing the error encountered.

12.56.1 Detailed Description

FSP Error Information Block.

Definition at line 43 of file FspErrorInfo.h.

12.56.2 Member Data Documentation

12.56.2.1 ErrorType

```
EFI_GUID FSP_ERROR_INFO::ErrorType
```

A GUID identifying the nature of the fatal error.

This GUID is platform speciic.

Definition at line 48 of file FspErrorInfo.h.

The documentation for this struct was generated from the following file:

- FspErrorInfo.h

12.57 FSPM_ARCH_CONFIG_PPI Struct Reference

This PPI provides FSP-M Arch Config PPI.

```
#include <FspmArchConfigPpi.h>
```

12.57.1 Detailed Description

This PPI provides FSP-M Arch Config PPI.

Definition at line 36 of file FspmArchConfigPpi.h.

The documentation for this struct was generated from the following file:

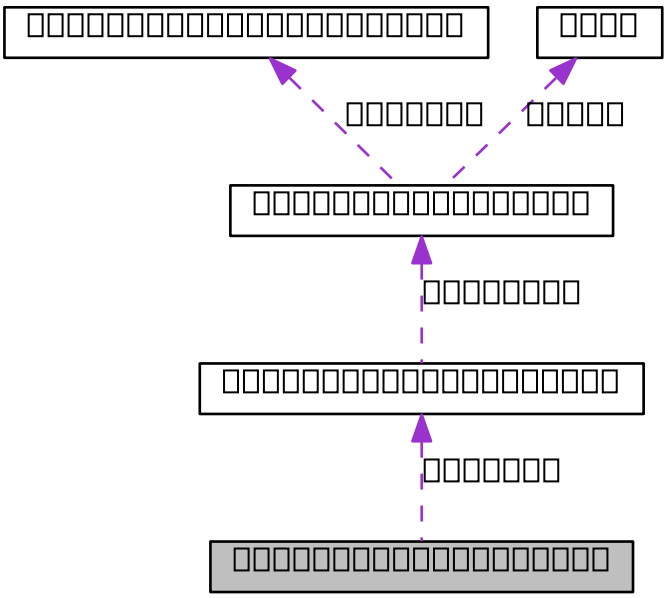
- FspmArchConfigPpi.h

12.58 FSPV_POSTMEM_CONFIG Struct Reference

The FSPV_POSTMEM_CONFIG block describes policies related to FSP Validation.

`#include <FspVConfig.h>`

Collaboration diagram for FSPV_POSTMEM_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header
  Config Block Header.
- UINT32 TestId
  FSP Validation Test Config Structure for FSPM.

12.58.1 Detailed Description

The FSPV_POSTMEM_CONFIG block describes policies related to FSP Validation.

Revision 1:

- Initial version.

Definition at line 69 of file FspVConfig.h.

The documentation for this struct was generated from the following file:

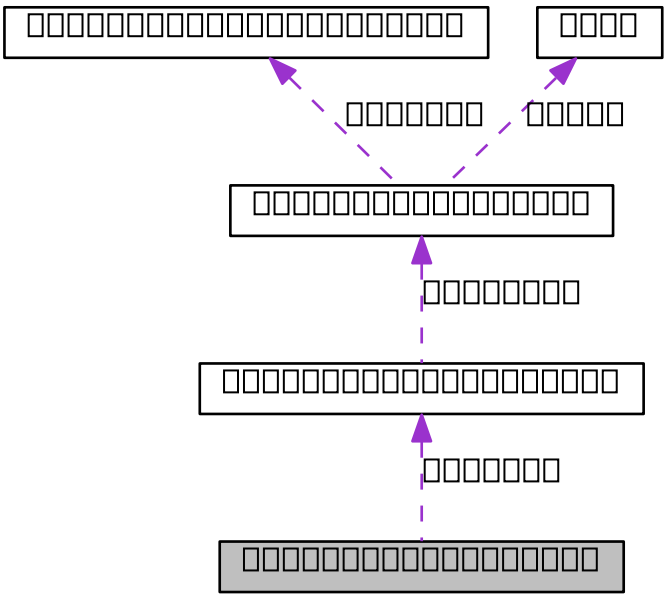
- FspVConfig.h

12.59 FSPV_PREMEM_CONFIG Struct Reference

The FSPV_PREMEM_CONFIG block describes policies related to FSP Validation.

`#include <FspVConfig.h>`

Collaboration diagram for FSPV_PREMEM_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header
  Config Block Header.
- UINT32 TestId
  FSP Validation Test Config Structure for FSPM.

12.59.1 Detailed Description

The FSPV_PREMEM_CONFIG block describes policies related to FSP Validation.

Revision 1:

- Initial version.

Definition at line 54 of file FspVConfig.h.

The documentation for this struct was generated from the following file:

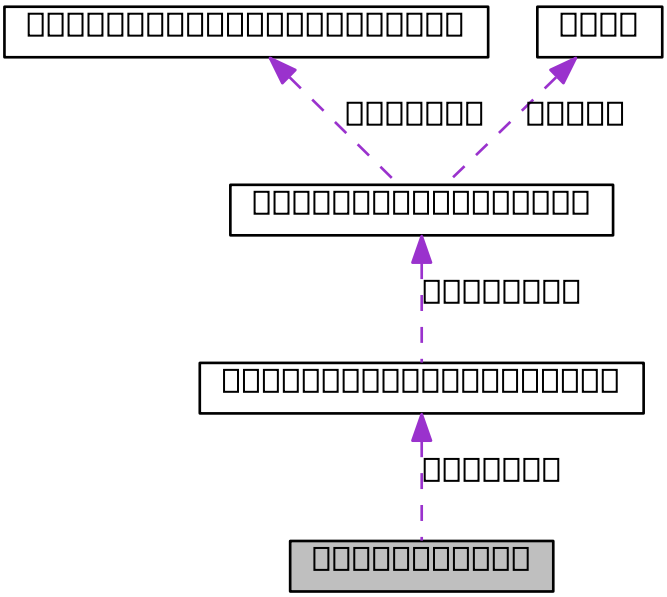
- FspVConfig.h

12.60 FUSA_CONFIG Struct Reference

The FUSA_CONFIG block describes the expected configuration of Functional Safety.

`#include <FusaConfig.h>`

Collaboration diagram for FUSA_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT32 DisplayFusaConfigEnable: 1

  *Enable/Disable Fusa feature on Display Fusa configuration on Display **0: Disable**; 1: Enable.*
- UINT32 GraphicFusaConfigEnable: 1

  Enable/Disable FuSa feature on GT: 0=Disable, 1=Enable <deprecated>
- UINT32 OpioFusaConfigEnable: 1

  Enable/Disable FuSa feature on OPIO: 0=Disable, 1=Enable <deprecated>
- UINT32 PsfFusaConfigEnable: 1

  Enable/Disable FuSa feature on PSF: 0=Disable, 1=Enable <deprecated>
- UINT32 IopFusaConfigEnable: 1

  Enable/Disable FuSa feature on IOP: 0=Disable, 1=Enable <deprecated>
- UINT32 IopFusaMcaCheckEnable: 1

  Enable/Disable FuSa feature on IOP MCA Check Enable: 0=Disable, 1=Enable <deprecated>
- UINT32 FusaConfigEnable: 1

  Enable/Disable FuSa feature: 0=Disable, 1=Enable.
- UINT32 FusaRunStartupArrayBist: 1

Enabling this will execute startup array test during boot <deprecated>

- UINT32 FusaRunStartupScanBist: 1

  Enabling this will execute startup scan test during boot <deprecated>
- UINT32 FusaRunPeriodicArrayBist: 1

  Enabling this will execute periodic array test during boot <deprecated>
- UINT32 FusaRunPeriodicScanBist: 1

  Enabling this will execute periodic scan test during boot <deprecated>
- UINT32 Module0Lockstep: 2

  Enable/Disable Lockstep for Atom module 0, which has 4 cores; 0: Disable lockstep; 1: Enable lockstep for Core 0 with Core 1, Core 2 with Core 3; 2: Enable lockstep for Core 0 with Core 1; 3: Enable lockstep for Core 2 with Core 3.
- UINT32 Module1Lockstep: 2

  Enable/Disable Lockstep for Atom module 1, which has 4 cores; 0: Disable lockstep; 1: Enable lockstep for Core 0 with Core 1, Core 2 with Core 3; 2: Enable lockstep for Core 0 with Core 1; 3: Enable lockstep for Core 2 with Core 3.
- UINT32 FusaRunStartupArrayBistMod0: 1

  Enabling this will execute startup array test Module 0 during boot <deprecated>
- UINT32 FusaRunStartupArrayBistMod1: 1

  Enabling this will execute startup array test Module 1 during boot <deprecated>
- UINT32 FusaRunStartupScanBistMod0: 1

  Enabling this will execute startup scan test Module 0 during boot <deprecated>
- UINT32 FusaRunStartupScanBistMod1: 1

  Enabling this will execute startup scan test Module 1 during boot <deprecated>
- UINT32 FusaRunPeriodicArrayBistMod0: 1

  Enabling this will execute periodic array test Module 0 during boot <deprecated>
- UINT32 FusaRunPeriodicArrayBistMod1: 1

  Enabling this will execute periodic array test Module 1 during boot <deprecated>
- UINT32 FusaRunPeriodicScanBistMod0: 1

  Enabling this will execute periodic scan test Module 0 during boot <deprecated>
- UINT32 FusaRunPeriodicScanBistMod1: 1

  Enabling this will execute periodic scan test Module 1 during boot <deprecated>
- UINT32 RsvdBits0: 9

  Reserved bits <deprecated>
- UINT32 FusaStartupPatternAddr

  Fusa Startup Pattern File Address.
- UINT32 FusaPeriodicPatternAddr

  <deprecated>
- UINT32 LpcLockstep: 2

  Enable/Disable Lockstep for Atom Lpc module, which has 4 cores; 0: Disable lockstep; 1: Enable lockstep for Core 0 with Core 1, Core 2 with Core 3; 2: Enable lockstep for Core 0 with Core 1; 3: Enable lockstep for Core 2 with Core 3.
- UINT32 RsvdBits1: 30

  Reserved Bits.
- UINT32 FusaCfiParityIDIBE2E: 1

  Enable/Disable IDI-B E2E CFI Parity.
- UINT32 FusaCfiParityCCFE2E_SANTA1: 1

  Enable/Disable CCF E2E SANTA1 CFI Parity.
- UINT32 FusaCfiParityCCFE2E_SANTA0: 1

  Enable/Disable CCF E2E SANTA0 CFI Parity.
- UINT32 FusaCfiParityHBOE2E_HBO0: 1

  Enable/Disable HBO E2E HBO0 CFI Parity.
- UINT32 FusaCfiParityHBOE2E_HBO1: 1

  Enable/Disable HBO E2E HBO1 CFI Parity.
- UINT32 FusaCfiParityPBE2E_AXI2CFI_IPU: 1

Enable/Disable PB E2E AXI2CFI_IPU CFI Parity.

- UINT32 FusaCfiParityPBE2E_AXI2CFI_VPU: 1

  Enable/Disable PB E2E AXI2CFI_VPU CFI Parity.
- UINT32 FusaCfiParityPBE2E_ICXL2CFI_MEDIA: 1

  Enable/Disable PB E2E ICXL2CFI_MEDIA CFI Parity.
- UINT32 FusaCfiParityPBE2E_ICXL2CFI_GT: 1

  Enable/Disable PB E2E ICXL2CFI_GT CFI Parity.
- UINT32 FusaCfiParityPBE2E_ICXL2CFI_IAX: 1

  Enable/Disable PB E2E ICXL2CFI_IAX CFI Parity.
- UINT32 FusaCfiParityPBE2E_DNI2CFI: 1

  Enable/Disable PB E2E DNI2CFI CFI Parity.
- UINT32 FusaCfiParitySVTUE2E: 1

  Enable/Disable SVTU E2E CFI Parity.
- UINT32 FusaCfiParitySNCUE2E: 1

  Enable/Disable SNCU E2E CFI Parity.
- UINT32 FusaCfiParityPUNITE2E: 1

  Enable/Disable PUNIT E2E CFI Parity.
- UINT32 FusaCfiParityIOCE2E: 1

  Enable/Disable IOC E2E CFI Parity.
- UINT32 FusaCfiParityIOCCEE2E: 1

  Enable/Disable IOCCE E2E CFI Parity.
- UINT32 FusaCfiParityIVTUE2E: 1

  Enable/Disable IVTU E2E CFI Parity.
- UINT32 FusaCfiParityCCEE2E_CCE0: 1

  Enable/Disable CCE E2E CCE0 CFI Parity.
- UINT32 FusaCfiParityCCEE2E_CCE1: 1

  Enable/Disable CCE E2E CCE1 CFI Parity.
- UINT32 IsFusaDiagnosticMode: 1

  To indicate BIOS is in Fusa Diagnostics Mode.
- UINT32 RsvdBits3: 12

  Reserved Bits.
- UINT8 RsvdByte1 [4]

  Reserved Byte.

12.60.1 Detailed Description

The FUSA_CONFIG block describes the expected configuration of Functional Safety.

Definition at line 45 of file FusaConfig.h.

12.60.2 Member Data Documentation

12.60.2.1 DisplayFusaConfigEnable

```
UINT32 FUSA_CONFIG::DisplayFusaConfigEnable
```

Enable/Disable Fusa feature on Display Fusa configuration on Display **0: Disable**; 1: Enable.

Enable/Disable Fusa feature on IOP MCA Check Fusa configuration on IOP MCA Check **0: Disable**; 1: Enable. Enable/Disable FuSa feature on DE: 0=Disable, 1=Enable <deprecated>

Definition at line 53 of file FusaConfig.h.

The documentation for this struct was generated from the following file:

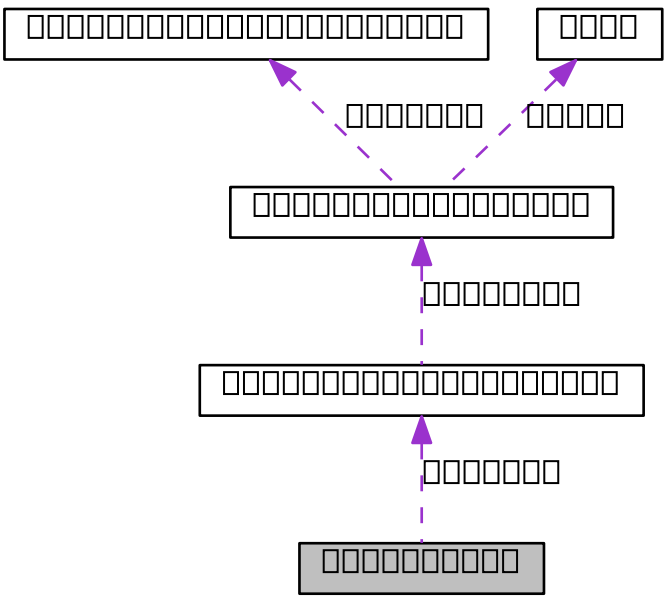
- FusaConfig.h

12.61 GBE_CONFIG Struct Reference

PCH intergrated GBE controller configuration settings.

```
#include <GbeConfig.h>
```

Collaboration diagram for GBE_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT32 Enable: 1

  *Determines if enable PCH internal GBE, 0: Disable; **1: Enable**.*
- UINT32 LtrEnable: 1

  ***0: Disable**; 1: Enable LTR capabilty of PCH internal LAN.*
- UINT32 PchWOLFastSupport: 1

  Enables bit B_PCH_ACPI_GPE0_EN_127_96_PME_B0 during PchLanSxCallback in PchLanSxSmm.
- UINT32 RsvdBits0: 29

  Reserved bits.

12.61.1 Detailed Description

PCH intergrated GBE controller configuration settings.

Definition at line 32 of file GbeConfig.h.

12.61.2 Member Data Documentation

12.61.2.1 Enable

`UINT32 GBE_CONFIG::Enable`

Determines if enable PCH internal GBE, 0: Disable; **1: Enable**.

When Enable is changed (from disabled to enabled or from enabled to disabled), it needs to set LAN Disable regsiter, which might be locked by FDSWL register. So it's recommendated to issue a global reset when changing the status for PCH Internal LAN.

Definition at line 40 of file GbeConfig.h.

The documentation for this struct was generated from the following file:

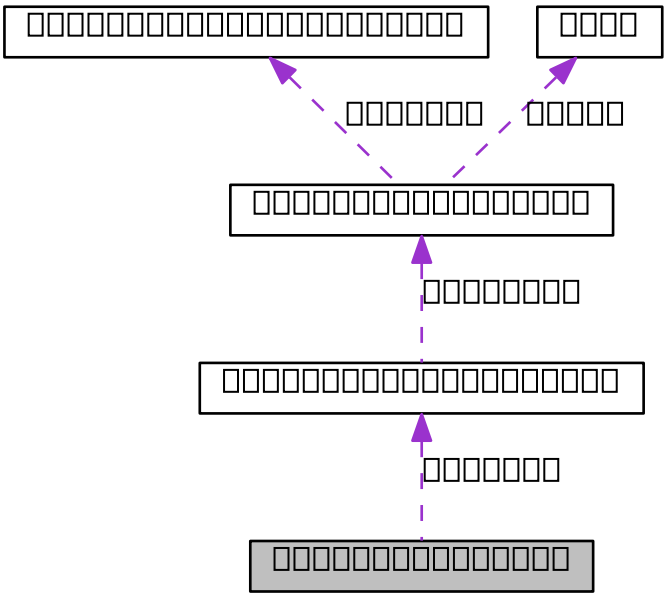
- GbeConfig.h

12.62 GPIO_DXE_CONFIG Struct Reference

This structure contains the DXE policies which are related to GPIO device.

`#include <GpioDevConfig.h>`

Collaboration diagram for GPIO_DXE_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header
  Config Block Header.
- UINT32 HideGpioAcpiDevice: 1
  If GPIO ACPI device is not used by OS it can be hidden.
- UINT32 HideHardGpioAcpiDevice: 1
  Only hide Hard GPIOs ACPI device 'HideGpioAcpiDevice' set to 1 can override this flag.
- UINT32 HideVirtGpioAcpiDevice: 1
  Only hide Virtual GPIOs ACPI device 'HideGpioAcpiDevice' set to 1 can override this flag.
- UINT32 RsvdBits: 29
  Reserved bits.

12.62.1 Detailed Description

This structure contains the DXE policies which are related to GPIO device.

Revision 1:

- Inital version.

Definition at line 36 of file GpioDevConfig.h.

12.62.2 Member Data Documentation

12.62.2.1 HideGpioAcpiDevice

```
UINT32 GPIO_DXE_CONFIG::HideGpioAcpiDevice
```

If GPIO ACPI device is not used by OS it can be hidden.

In such case no other device exposed to the system can reference GPIO device in one of its resources through GpioIo(..) or GpioInt(..) ACPI descriptors. **0: Disable**; 1: Enable

Definition at line 44 of file GpioDevConfig.h.

12.62.2.2 HideHardGpioAcpiDevice

```
UINT32 GPIO_DXE_CONFIG::HideHardGpioAcpiDevice
```

Only hide Hard GPIOs ACPI device 'HideGpioAcpiDevice' set to 1 can override this flag.

0: Disable; 1: Enable

Definition at line 50 of file GpioDevConfig.h.

12.62.2.3 HideVirtGpioAcpiDevice

```
UINT32 GPIO_DXE_CONFIG::HideVirtGpioAcpiDevice
```

Only hide Virtual GPIOs ACPI device 'HideGpioAcpiDevice' set to 1 can override this flag.

0: Disable; 1: Enable

Definition at line 56 of file GpioDevConfig.h.

The documentation for this struct was generated from the following file:

- GpioDevConfig.h

12.63 GUID Struct Reference

128 bit buffer containing a unique identifier value.

```
#include <Base.h>
```

12.63.1 Detailed Description

128 bit buffer containing a unique identifier value.

Unless otherwise specified, aligned on a 64 bit boundary.

Definition at line 213 of file Base.h.

The documentation for this struct was generated from the following file:

- Base.h

12.64 HDA_LINK_DMIC Struct Reference

HD Audio DMIC Interface Policies.

`#include <HdAudioConfig.h>`

Public Attributes

- UINT32 Enable: 1

  *HDA DMIC interface enable. When enabled related pins will be switched to native mode: **0: Disable**; 1: Enable.*
- UINT32 DmicClockSelect: 2

  *DMIC link clock select: **0: Both**, 1: ClkA, 2: ClkB; default is "Both".*
- HDA_DMIC_PIN_MUX PinMux

  Pin mux configuration.

12.64.1 Detailed Description

HD Audio DMIC Interface Policies.

Definition at line 132 of file HdAudioConfig.h.

The documentation for this struct was generated from the following file:

- HdAudioConfig.h

12.65 HDA_LINK_HDA Struct Reference

HD Audio Link Policies.

`#include <HdAudioConfig.h>`

Public Attributes

- UINT32 Enable: 1

  *HDA interface enable. When enabled related pins will be switched to native mode: **0: Disable**; 1: Enable.*
- UINT8 SdiEnable [PCH_MAX_HDA_SDI]

  *HDA SDI signal enable. When enabled related SDI pins will be switched to appropriate native mode: **0: Disable**; 1: Enable.*
- UINT8 Reserved [(4 -(PCH_MAX_HDA_SDI % 4)) % 4]

  Padding for SDI enable table.

12.65.1 Detailed Description

HD Audio Link Policies.

Definition at line 122 of file HdAudioConfig.h.

The documentation for this struct was generated from the following file:

- HdAudioConfig.h

12.66 HDA_LINK_SNDW Struct Reference

HD Audio SNDW Interface Policies.

```
#include <HdAudioConfig.h>
```

Public Attributes

- UINT32 Enable: 1

  *HDA SNDW interface enable. When enabled related pins will be switched to native mode: **0: Disable**; 1: Enable.*

12.66.1 Detailed Description

HD Audio SNDW Interface Policies.

Definition at line 151 of file HdAudioConfig.h.

The documentation for this struct was generated from the following file:

- HdAudioConfig.h

12.67 HDA_LINK_SSP Struct Reference

HD Audio SSP Interface Policies.

```
#include <HdAudioConfig.h>
```

Public Attributes

- UINT32 Enable: 1

  *HDA SSP interface enable. When enabled related pins will be switched to native mode: **0: Disable**; 1: Enable.*
- HDA_SSP_PIN_MUX PinMux

  Pin mux configuration.

12.67.1 Detailed Description

HD Audio SSP Interface Policies.

Definition at line 142 of file HdAudioConfig.h.

The documentation for this struct was generated from the following file:

- HdAudioConfig.h

12.68 HDA_MIC_PRIVACY Struct Reference

HD Audio Link Policies.

```
#include <HdAudioConfig.h>
```

Public Attributes

- UINT32 Mode: 2

  *Microphone Privacy Mode: **0: No Microphone Privacy Support**; 1: HW Managed Microphone Privacy; 2: FW Managed Microphone Privacy; 3: Force Microphone Mute.*
- UINT32 Deglitch: 1

  *Microphone Privacy Deglitch **0: Disable**; 1: Enable.*
- UINT32 HwModeSoundWire0: 1

  *Microphone Privacy applied for SoundWire Link number 0 in HW Mode: **0: Disable**; 1: Enable.*
- UINT32 HwModeSoundWire1: 1

  *Microphone Privacy applied for SoundWire Link number 1 in HW Mode: **0: Disable**; 1: Enable.*
- UINT32 HwModeSoundWire2: 1

  *Microphone Privacy applied for SoundWire Link number 2 in HW Mode: **0: Disable**; 1: Enable.*
- UINT32 HwModeSoundWire3: 1

  *Microphone Privacy applied for SoundWire Link number 3 in HW Mode: **0: Disable**; 1: Enable.*
- UINT32 HwModeSoundWire4: 1

  *Microphone Privacy applied for SoundWire Link number 4 0i0n HW Mode: **0: Disable**; 1: Enable.*
- UINT32 HwModeDmic: 1

  *Microphone Privacy applied for DMIC in HW Mode: **0: Disable**; 1: Enable.*
- UINT32 Timeout

  Indicates the time-out duration to wait before forcing the actual microphone privacy DMA data zeroing. Unit in number of RTC clocks.

12.68.1 Detailed Description

HD Audio Link Policies.

Definition at line 106 of file HdAudioConfig.h.

The documentation for this struct was generated from the following file:

- HdAudioConfig.h

12.69 HDA_SNDW_MULTILANE Struct Reference

HD Audio SNDW Multilane Policies.

`#include <HdAudioConfig.h>`

Public Attributes

- UINT32 Enable: 2

  *HDA SNDW Multilane enablement: **0: Disable**, 1: Two lines enabled, 2: Three lines enabled, 3: Four Lines enabled.*
- UINT32 SndwInterface: 3

  *HDA SNDW Interface assigment to a Multiline: **0: Sndw Interface 0**, 1: Sndw Interface 1, 2: Sndw Interface 2, etc.*
- UINT32 RsvdBits0: 27

  Reserved bits 0.
- HDA_SNDW_MULTILANE_PIN_MUX PinMux

  Pin mux configuration.

12.69.1 Detailed Description

HD Audio SNDW Multilane Policies.

Definition at line 159 of file HdAudioConfig.h.

The documentation for this struct was generated from the following file:

- HdAudioConfig.h

12.70 HDA_VERB_TABLE_HEADER Struct Reference

Azalia verb table header Every verb table should contain this defined header and followed by azalia verb commands.

`#include <HdAudioConfig.h>`

Public Attributes

- UINT16 VendorId

  Codec Vendor ID.
- UINT16 DeviceId

  Codec Device ID.
- UINT8 RevisionId

  Revision ID of the codec. 0xFF matches any revision.
- UINT8 SdiNum

  SDI number, 0xFF matches any SDI.
- UINT16 DataDwords

  Number of data DWORDs following the header.

12.70.1 Detailed Description

Azalia verb table header Every verb table should contain this defined header and followed by azalia verb commands.

Definition at line 58 of file HdAudioConfig.h.

The documentation for this struct was generated from the following file:

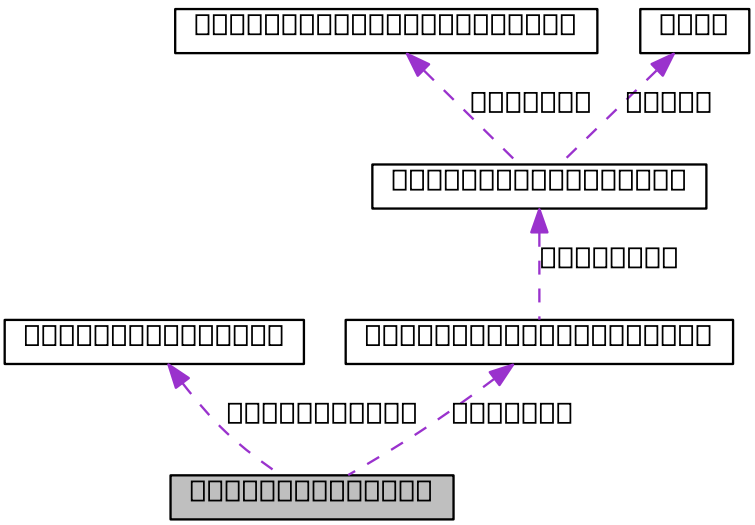
- HdAudioConfig.h

12.71 HDAUDIO_CONFIG Struct Reference

This structure contains the policies which are related to HD Audio device (cAVS).

`#include <HdAudioConfig.h>`

Collaboration diagram for HDAUDIO_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT32 Pme: 1

  *Azalia wake-on-ring, **0: Disable**; 1: Enable.*
- UINT32 CodecSxWakeCapability: 1

  *Capability to detect wake initiated by a codec in Sx (eg by modem codec), **0: Disable**; 1: Enable.*
- UINT32 HdAudioLinkFrequency: 4

  *HDA-Link frequency (PCH_HDAUDIO_LINK_FREQUENCY enum): **2: 24MHz**, 1: 12MHz, 0: 6MHz.*
- UINT32 RsvdBits0: 26

  Reserved bits 0.
- UINT8 VerbTableEntryNum

  Number of the verb table entry defined in VerbTablePtr.
- UINT8 RsvdBytes184 [7]

  Reserved bytes, align to multiple 4.
- UINTN VerbTablePtr

  Pointer to a verb table array.
- HDA_MIC_PRIVACY MicPrivacy

  HdAudio Microphone Privacy Settings.

12.71.1 Detailed Description

This structure contains the policies which are related to HD Audio device (cAVS).

Revision 1:

- Inital version.

Definition at line 173 of file HdAudioConfig.h.

12.71.2 Member Data Documentation

12.71.2.1 VerbTableEntryNum

```
UINT8 HDAUDIO_CONFIG::VerbTableEntryNum
```

Number of the verb table entry defined in VerbTablePtr.

Each entry points to a verb table which contains HDAUDIO_VERB_TABLE structure and verb command blocks.

Definition at line 183 of file HdAudioConfig.h.

12.71.2.2 VerbTablePtr

```
UINTN HDAUDIO_CONFIG::VerbTablePtr
```

Pointer to a verb table array.

This pointer points to 32bits address, and is only eligible and consumed in post mem phase. Each entry points to a verb table which contains HDAUDIO_VERB_TABLE structure and verb command blocks. The prototype of this is: HDAUDIO_VERB_TABLE **VerbTablePtr;

Definition at line 192 of file HdAudioConfig.h.

The documentation for this struct was generated from the following file:

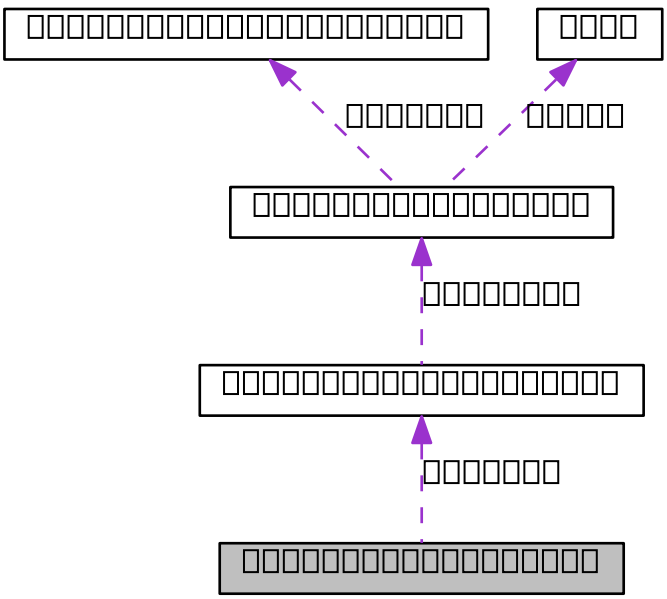
- HdAudioConfig.h

12.72 HDAUDIO_DXE_CONFIG Struct Reference

This structure contains the DXE policies which are related to HD Audio device (cAVS).

```
#include <HdAudioConfig.h>
```

Collaboration diagram for HDAUDIO_DXE_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- HDAUDIO_SNDW_CONFIG SndwConfig [PCH_MAX_HDA_SNDW_LINK_NUM]

  SNDW configuration for exposed via SNDW ACPI tables:
- UINT32 DspFeatureMask

  *Bitmask of supported DSP features: [BIT0] - WoV; [BIT1] - BT Sideband; [BIT2] - Codec VAD; [BIT5] - BT Intel HFP; [BIT6] - BT Intel A2DP [BIT7] - DSP based speech pre-processing disabled; [BIT8] - 0: Intel WoV, 1: Windows Voice Activation [BIT9] - BT Intel Low Energy;[BIT9] - 0 BT Intel Low Energy is not supported , 1: BT Intel Low Energy is supported Default is **zero**.*
- HDAUDIO_DISC_BT_OFFLOAD HdaDiscBtOffload

  Discrete BT HCI Audio Offload Support.
- UINT32 SoundFreqPoolSelect: 1

  Deprecated: SoundWire supported frequency pool select has been moved to HDAUDIO_SNDW_CONFIG to enable per-link configuration of each SoundWire interface.

12.72.1 Detailed Description

This structure contains the DXE policies which are related to HD Audio device (cAVS).

Revision 1:

- Inital version. **Revision 2:**
- Add IoControlEnabled **Revision 3:**
- Remove IoControlEnabled **Revision 4:**
- Add SoundFreqPoolSelect and RsvdBits1
- Add SoundWire properties controls

Definition at line 309 of file HdAudioConfig.h.

The documentation for this struct was generated from the following file:

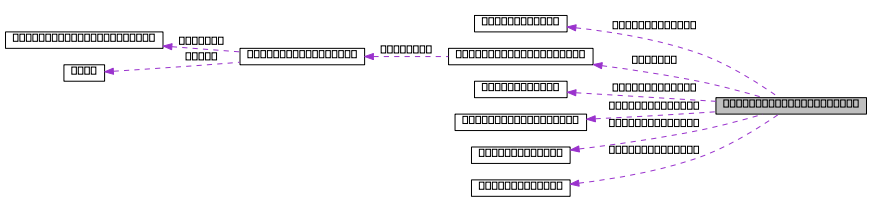
- HdAudioConfig.h

12.73 HDAUDIO_PREMEM_CONFIG Struct Reference

This structure contains the premem policies which are related to HD Audio device (cAVS).

`#include <HdAudioConfig.h>`

Collaboration diagram for HDAUDIO_PREMEM_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.

- UINT32 Enable: 1

  Intel HD Audio (Azalia) enablement: 0: Disable, ***1: Enable***

- UINT32 DspEnable: 1

  DSP enablement: 0: Disable; ***1: Enable***

- UINT32 DspUaaCompliance: 1

  Universal Audio Architecture compliance for DSP enabled system: ***0: Not-UAA Compliant (Intel SST driver supported only)****, 1: UAA Compliant (HDA Inbox driver or SST driver supported)*

- UINT32 IDispLinkFrequency: 4

  iDisp-Link frequency (PCH_HDAUDIO_LINK_FREQUENCY enum): ***4: 96MHz****, 3: 48MHz*

- UINT32 IDispLinkTmode: 3

  iDisp-Link T-Mode (PCH_HDAUDIO_IDISP_TMODE enum): 0: 2T, 1: 1T, 2: 4T, ***3: 8T****, 4: 16T*

- UINT32 IDispCodecDisconnect: 1

  iDisplay Audio Codec disconnection, ***0: Not disconnected, enumerable****; 1: Disconnected SDI, not enumerable*

- UINT32 PowerGatingSupported: 1

  Power Gating supported: ***0: Not supported****, 1: Supported.*

- UINT32 SoundWireClockSelect: 1

  SoundWire clock source select: 0: XTAL, 1: Audio PLL.

- UINT32 RsvdBits: 19

  Reserved bits 0.

- HDA_LINK_HDA AudioLinkHda

  Audio Link Mode configuration bitmask.

- HDA_LINK_DMIC AudioLinkDmic [PCH_MAX_HDA_DMIC_LINK_NUM]

  DMIC link enablement: 0: Disable; ***1: Enable****.*

- HDA_LINK_SSP AudioLinkSsp [PCH_MAX_HDA_SSP_LINK_NUM]

  I2S/SSP link enablement: ***0: Disable****; 1: Enable.*

- HDA_LINK_SNDW AudioLinkSndw [PCH_MAX_HDA_SNDW_LINK_NUM]

  SoundWire link enablement: ***0: Disable****; 1: Enable.*

- HDA_SNDW_MULTILANE SndwMultilane [PCH_MAX_HDA_SNDW_MULTILANE_NUM]

  Multilane SoundWire enablement.

- UINT16 ResetWaitTimer

  (Test) *The delay timer after Azalia reset, the value is number of microseconds. Default is* ***600****.*

- UINT8 PowerClockGating

  Power/Clock Gating (PG/CG): 0: POR (Enable), 1: Enable, 2: Disable.

- UINT8 LowFreqLinkClkSrc

  Low Frequency Link Clock Source (LFLCS): 0: POR (Enable), 1: Enable (XTAL), 2: Disable (Audio PLL)

- UINT32 SubSystemIds

  Value for SID and SVID. If its set to 0 then default value is used.

- HDAUDIO_DISC_BT_OFFLOAD HdaDiscBtOffload

  Discrete BT HCI Audio Offload Support.

12.73.1 Detailed Description

This structure contains the premem policies which are related to HD Audio device (cAVS).

Revision 1:

- Inital version. **Revision 2:**
- Add DmicClockSelect **Revision 3:**
- Add AudioFpgaSocVer **Revision 4:**
- Remove AudioFpgaSocVer **Revision 5:**
- Remove VcType **Revision 6:**
- Add HdaDiscBtOffload

Definition at line 221 of file HdAudioConfig.h.

12.73.2 Member Data Documentation

12.73.2.1 AudioLinkDmic

```
HDA_LINK_DMIC HDAUDIO_PREMEM_CONFIG::AudioLinkDmic[PCH_MAX_HDA_DMIC_LINK_NUM]
```

DMIC link enablement: 0: Disable; **1: Enable**.

DMIC0 LKF: Muxed with SNDW2/SNDW4.

Definition at line 247 of file HdAudioConfig.h.

12.73.2.2 AudioLinkHda

```
HDA_LINK_HDA HDAUDIO_PREMEM_CONFIG::AudioLinkHda
```

Audio Link Mode configuration bitmask.

Allows to configure enablement of the following interfaces: HDA-Link, DMIC, SSP, SoundWire. HDA-Link enablement: 0: Disable; **1: Enable**.

Definition at line 242 of file HdAudioConfig.h.

12.73.2.3 AudioLinkSndw

HDA_LINK_SNDW HDAUDIO_PREMEM_CONFIG::AudioLinkSndw[PCH_MAX_HDA_SNDW_LINK_NUM]

SoundWire link enablement: **0: Disable**; 1: Enable.

SNDW2 LKF: Muxed with DMIC0/DMIC1. SNDW3 LKF: Muxed with DMIC1. SNDW4 LKF: Muxed with DMIC0.

Definition at line 261 of file HdAudioConfig.h.

12.73.2.4 AudioLinkSsp

HDA_LINK_SSP HDAUDIO_PREMEM_CONFIG::AudioLinkSsp[PCH_MAX_HDA_SSP_LINK_NUM]

I2S/SSP link enablement: **0: Disable**; 1: Enable.

SSP0/1 LKF: Muxed with HDA.

Note

Since the I2S/SSP2 pin set contains pads which are also used for CNVi purpose, enabling AudioLinkSsp2 is exclusive with CNVi is present.

Definition at line 254 of file HdAudioConfig.h.

12.73.2.5 SndwMultilane

HDA_SNDW_MULTILANE HDAUDIO_PREMEM_CONFIG::SndwMultilane[PCH_MAX_HDA_SNDW_MULTILANE_NUM]

Multilane SoundWire enablement.

The index of Sndw interface to be used depends on used platform.

Definition at line 266 of file HdAudioConfig.h.

The documentation for this struct was generated from the following file:

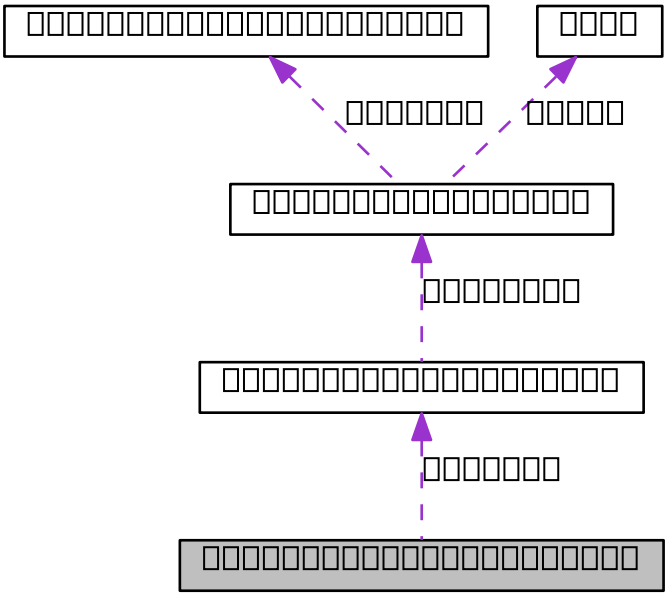
- HdAudioConfig.h

12.74 HOST_BRIDGE_PEI_CONFIG Struct Reference

This configuration block describes HostBridge settings in Post-Mem.

`#include <HostBridgeConfig.h>`

Collaboration diagram for HOST_BRIDGE_PEI_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Offset 0-27 Config Block Header.
- UINT8 Device4Enable

  Offser 28 :This policy is used to control enable or disable System Agent Thermal device (0,4,0). ***0=FALSE****, 1=TRUE.*
- UINT8 SkipPamLock

  Offset 29 :To skip PAM register locking.
- UINT8 Reserved [2]

  Offset 31 Reserved for future use.

12.74.1 Detailed Description

This configuration block describes HostBridge settings in Post-Mem.

Revision 1:

- Initial version.

Definition at line 89 of file HostBridgeConfig.h.

12.74.2 Member Data Documentation

12.74.2.1 SkipPamLock

```
UINT8 HOST_BRIDGE_PEI_CONFIG::SkipPamLock
```

Offset 29 :To skip PAM register locking.

Note

It is still recommended to set PCI Config space B0: D0: F0: Offset 80h[0]=1 in platform code even Silicon code skipped this.
0=All PAM registers will be locked in Silicon code, 1=Skip lock PAM registers in Silicon code.

Definition at line 92 of file HostBridgeConfig.h.

The documentation for this struct was generated from the following file:

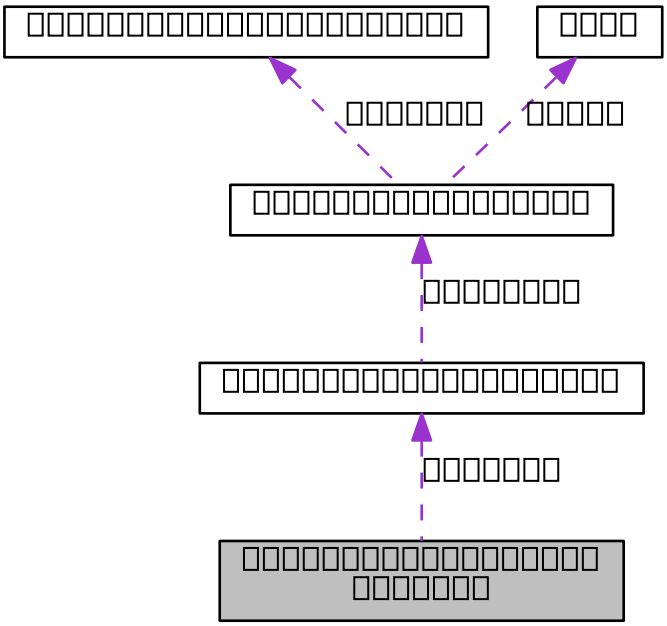
- HostBridgeConfig.h

12.75 HOST_BRIDGE_PREMEM_CONFIG Struct Reference

This configuration block describes HostBridge settings in PreMem.

```
#include <HostBridgeConfig.h>
```

Collaboration diagram for HOST_BRIDGE_PREMEM_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Offset 0-27 Config Block Header.
- UINT8 RsvdBytes51 [4]

  Reserved.
- UINT64 MchBar

  Offset 28 Address of System Agent MCHBAR: ***0xFEDC0000***
- UINT64 RegBar

  Offset 36 Address of System Agent REGBAR: ***0xF0000000***
- UINT32 MmioSize

  Offset 44 : Size of reserved MMIO space for PCI devices
  0=AUTO*, 512=512MB, 768=768MB, 1024=1024MB, 1280=1280MB, 1536=1536MB, 1792=1792MB, 2048=2048MB, 2304=2304MB, 2560=2560MB, 2816=2816MB, 3072=3072MB*
  When AUTO mode selected, the MMIO size will be calculated by required MMIO size from PCIe devices detected.
- UINT32 MmioSizeAdjustment

  Offset 48 Increase (given positive value) or Decrease (given negative value) the Reserved MMIO size when Dynamic Tolud/AUTO mode enabled (in MBs): ***0=no adjustment***
- UINT32 ApicLocalAddress

  Offset 52 Address of System Agent ApicLocalAddress: ***0xFEE00000***
- UINT32 NvmeHcPeiMmioBase

  Offset 56 Address of System Agent NvmeHcPeiMmioBase: ***0xA0000000***
- UINT32 NvmeHcPeiMmioLimit

  Offset 60 Address of System Agent NvmeHcPeiMmioLimit: ***0xA1000000***
- UINT32 AhciPeiMmioBase

  Offset 64 Address of System Agent AhciPeiMmioBase: ***0xD1000000***
- UINT32 AhciPeiMmioLimit

  Offset 68 Address of System Agent AhciPeiMmioLimit: ***0xD10FFFFF***
- UINT16 EcExtraIoBase

  Offset 72 Address of System Agent EcExtraIoBase: ***0x6A0***
- UINT16 SioBaseAddress

  Offset 74 Address of System Agent SioBaseAddress: ***0x0680***
- UINT32 VmdCfgBarBar

  Offset 76 Address of System Agent VmdCfgBarBase: ***0xA0000000***
- UINT32 VmdMemBar2Bar

  Offset 80 Address of System Agent VmdMemBar2Base: ***0xA4000000***
- UINT32 VmdMemBar1Bar

  Offset 84 Address of System Agent VmdMemBar1Base: ***0xA2000000***
- UINT64 SafBar

  Offset 88 Address of System Agent SafBar: ***0x4150000000***
- UINT8 EnableAbove4GBMmio

  Offset 96 Enable/disable above 4GB MMIO resource support: 0=Disable, ***1=Enable***
- UINT8 CridEnable

  Offset 97 For Platforms supporting Intel(R) SIPP, this policy is use control enable/disable Compatibility Revision ID (CRID) feature: ***0=FALSE****, 1=TRUE.*
- UINT8 RsvdBytes77 [5]

  Offset 98-97 Reserved to ensure config block size is a multiple of DWORDs.

12.75.1 Detailed Description

This configuration block describes HostBridge settings in PreMem.

Revision 1:

- Initial version. **Revision 2**:
- Added GdxcBaseAddress, EdramBaseAddress, EpBaseAddress, ApicLocalAddress, NvmeHcPeiMmioBase.
- Added NvmeHcPeiMmioLimit, AhciPeiMmioBase, AhciPeiMmioLimit, EcExtraIoBase, NvmeHcPeiMmioBase, SioBaseAddress.
- Added VmdCfgBarBar, VmdMemBar2Bar, VmdMemBar1Bar. **Revision 3**:
- Removed GdxcBaseAddress, EdramBaseAddress, EpBaseAddress. **Revision 4**:
- Added CridEnable. **Revision 5**:
- Resized MchBar, RegBar **Revision 6**:
- Added FabricGVDisable.

Definition at line 51 of file HostBridgeConfig.h.

The documentation for this struct was generated from the following file:

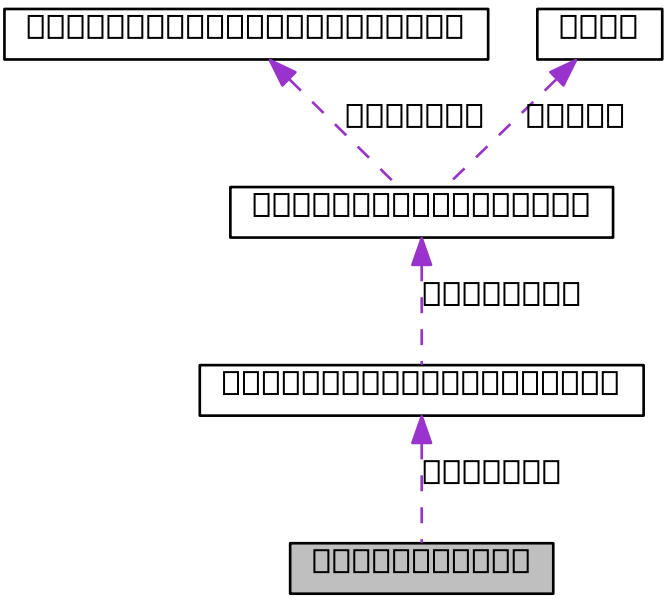
- HostBridgeConfig.h

12.76 HSIO_CONFIG Struct Reference

The HSIO_CONFIG block provides HSIO message related settings.

`#include <HsioConfig.h>`

Collaboration diagram for HSIO_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT8 RsvdBytes35 [4]

  Reserved.
- VOID * ChipsetInitBinPtr

  Policy used to point to the Base (+ OEM) ChipsetInit binary used to sync between BIOS and CSME.
- UINT32 ChipsetInitBinLen

  Policy used to indicate the size of the Base (+ OEM) ChipsetInit binary used to sync between BIOS and CSME.
- UINT8 RsvdBytes38 [4]

  Reserved.
- VOID * NphyBinPtr

  Policy used to point to the OEM Nphy binary used to sync between BIOS and CSME.
- UINT32 NphyBinLen

  Policy used to indicate the size of the OEM Nphy binary used to sync between BIOS and CSME.
- UINT8 RsvdBytes41 [4]

  Reserved.
- VOID * SynpsPhyBinPtr

  Policy used to point to the OEM Synophsys Phy binary used to sync between BIOS and CSME.
- UINT32 SynpsPhyBinLen

  Policy used to indicate the size of the OEM Synophsys Phy binary used to sync between BIOS and CSME.
- UINT8 RsvdBytes44 [4]

  Reserved.

12.76.1 Detailed Description

The HSIO_CONFIG block provides HSIO message related settings.

Definition at line 33 of file HsioConfig.h.

The documentation for this struct was generated from the following file:

- HsioConfig.h

12.77 HSIO_PARAMETERS Struct Reference

This structure describes USB3 Port N configuration parameters.

```
#include <Usb3HsioConfig.h>
```

Public Attributes

- UINT8 HsioTxDownscaleAmp

  USB 3.0 TX Output Downscale Amplitude Adjustment (orate01margin) HSIO_TX_DWORD8[21:16] ***Default = 00h***
- UINT8 HsioTxDeEmph

  USB 3.0 TX Output -3.5dB De-Emphasis Adjustment Setting (ow2tapgen2deemph3p5) HSIO_TX_DWORD5[21:16] ***Default = 29h*** *(approximately -3.5dB De-Emphasis)*
- UINT8 HsioCtrlAdaptOffsetCfg

  Signed Magnatude number added to the CTLE code.
- UINT8 HsioFilterSelN

  LFPS filter select for n (filter_sel_n_2_0) HSIO_RX_DWORD51 [29:27] 0h:1.6ns 1h:2.4ns 2h:3.2ns 3h:4.0ns 4h:4.8ns 5h:5.6ns 6h:6.4ns ***Default = 0h***
- UINT8 HsioFilterSelP

  LFPS filter select for p (filter_sel_p_2_0) HSIO_RX_DWORD51 [26:24] 0h:1.6ns 1h:2.4ns 2h:3.2ns 3h:4.0ns 4h:4.8ns 5h:5.6ns 6h:6.4ns ***Default = 0h***
- UINT8 HsioOlfpsCfgPullUpDwnRes

  Controls the input offset (olfpscfgpullupdwnres_sus_usb_2_0) HSIO_RX_DWORD51 [2:0] 000 Prohibited 001 45K 010 Prohibited 011 31K 100 36K 101 36K 110 36K 111 36K ***Default = 3h***
- UINT8 HsioTxDeEmphEnable

  Enable the write to USB 3.0 TX Output -3.5dB De-Emphasis Adjustment, ***0: Disable****; 1: Enable.*
- UINT8 HsioTxDownscaleAmpEnable

  Enable the write to USB 3.0 TX Output Downscale Amplitude Adjustment, ***0: Disable****; 1: Enable.*
- UINT8 HsioCtrlAdaptOffsetCfgEnable

  Enable the write to Signed Magnatude number added to the CTLE code, ***0: Disable****; 1: Enable.*
- UINT8 HsioFilterSelNEnable

  Enable the write to LFPS filter select for n, ***0: Disable****; 1: Enable.*
- UINT8 HsioFilterSelPEnable

  Enable the write to LFPS filter select for p, ***0: Disable****; 1: Enable.*
- UINT8 HsioOlfpsCfgPullUpDwnResEnable

  Enable the write to olfpscfgpullupdwnres, ***0: Disable****; 1: Enable.*
- UINT8 HsioTxRate3UniqTran

  USB 3.0 TX Output - Unique Transition Bit Scale for rate 3 (rate3UniqTranScale) HSIO_TX_DWORD9[6:0] ***Default = 4Ch***
- UINT8 HsioTxRate2UniqTran

  USB 3.0 TX Output -Unique Transition Bit Scale for rate 2 (rate2UniqTranScale) HSIO_TX_DWORD9[14:8] ***Default = 4Ch***
- UINT8 HsioTxRate1UniqTran

  USB 3.0 TX Output - Unique Transition Bit Scale for rate 1 (rate1UniqTranScale) HSIO_TX_DWORD9[22:16] ***Default = 4Ch***
- UINT8 HsioTxRate0UniqTran

  USB 3.0 TX Output - Unique Transition Bit Scale for rate 0 (rate0UniqTranScale) HSIO_TX_DWORD9[30:24] ***Default = 4Ch***
- UINT8 HsioTxRate3UniqTranEnable

  Enable the write to USB 3.0 TX Unique Transition Bit Mode for rate 3, ***0: Disable****; 1: Enable.*
- UINT8 HsioTxRate2UniqTranEnable

  Enable the write to USB 3.0 TX Unique Transition Bit Mode for rate 2, ***0: Disable****; 1: Enable.*
- UINT8 HsioTxRate1UniqTranEnable

  Enable the write to USB 3.0 TX Unique Transition Bit Mode for rate 1, ***0: Disable****; 1: Enable.*
- UINT8 HsioTxRate0UniqTranEnable

  Enable the write to USB 3.0 TX Unique Transition Bit Mode for rate 0, ***0: Disable****; 1: Enable.*

12.77.1 Detailed Description

This structure describes USB3 Port N configuration parameters.

Definition at line 35 of file Usb3HsioConfig.h.

12.77.2 Member Data Documentation

12.77.2.1 HsioCtrlAdaptOffsetCfg

`UINT8 HSIO_PARAMETERS::HsioCtrlAdaptOffsetCfg`

Signed Magnatude number added to the CTLE code.

(ctle_adapt_offset_cfg_4_0) HSIO_RX_DWORD25 [20:16] Ex: -1 – 1_0001. +1: 0_0001 **Default = 0h**

Definition at line 54 of file Usb3HsioConfig.h.

The documentation for this struct was generated from the following file:

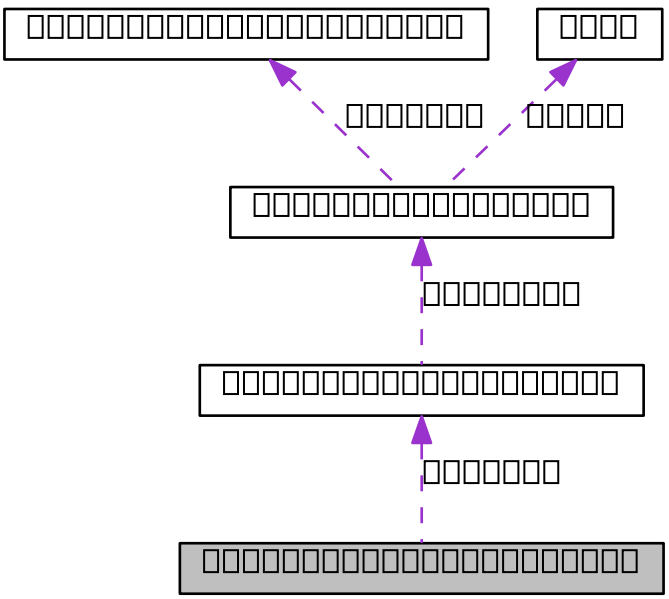
- Usb3HsioConfig.h

12.78 HYBRID_GRAPHICS_CONFIG Struct Reference

This Configuration block configures CPU PCI Express 0/1/2 RTD3 GPIOs & Root Port.

`#include <HybridGraphicsConfig.h>`

Collaboration diagram for HYBRID_GRAPHICS_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Offset 0-27 Config Block Header.
- UINT16 HgSubSystemId

  Offset 54 Hybrid Graphics Subsystem ID: ***2212***

12.78.1 Detailed Description

This Configuration block configures CPU PCI Express 0/1/2 RTD3 GPIOs & Root Port.

Hybrid Gfx uses the same GPIOs & Root port as PCI Express 0/1/2 RTD3. **Revision 1**:

- Initial version. **Revision 2**:
- Add HgSlot Policy: PEG or PCH Slot Slection for Hybrid Graphics **Revision 3**:
- Removed HgSlot Policy: As there is no selection between PEG or PCH Slot for Hybrid Graphics

Definition at line 38 of file HybridGraphicsConfig.h.

The documentation for this struct was generated from the following file:

- HybridGraphicsConfig.h

12.79 I2C_PIN_MUX Struct Reference

I2C signals pin muxing settings.

```
#include <SerialIoDevices.h>
```

Public Attributes

- UINT32 Sda

  SDA Pin mux configuration. Refer to GPIO_∗_MUXING_SERIALIO_I2Cx_SDA_∗.
- UINT32 Scl

  SCL Pin mux configuration. Refer to GPIO_∗_MUXING_SERIALIO_I2Cx_SCL_∗.

12.79.1 Detailed Description

I2C signals pin muxing settings.

If signal can be enable only on a single pin then this parameter is ignored by RC. Refer to GPIO_∗_MUXING_SE↩ RIALIO_I2Cx_∗ in GpioPins∗.h for supported settings on a given platform

Definition at line 198 of file SerialIoDevices.h.

The documentation for this struct was generated from the following file:

- SerialIoDevices.h

12.80 I3C_MUX_GPIO_CONFIG Struct Reference

I3C GPIO settings.

`#include <LpssI3cConfig.h>`

Public Attributes

- UINT32 PadTermination

  GPIO signals pin muxing settings.

12.80.1 Detailed Description

I3C GPIO settings.

Definition at line 56 of file LpssI3cConfig.h.

The documentation for this struct was generated from the following file:

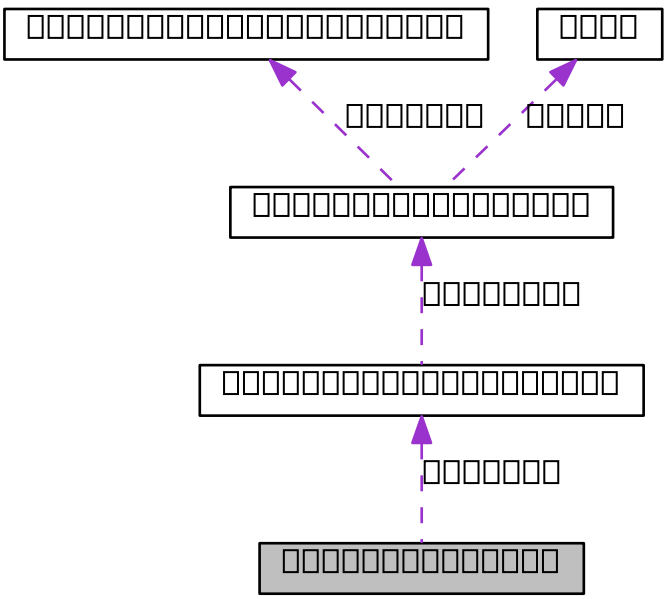
- LpssI3cConfig.h

12.81 IAX_PEI_CONFIG Struct Reference

IAX PEI configuration **Revision 1**:

`#include <IaxConfig.h>`

Collaboration diagram for IAX_PEI_CONFIG:



Public Attributes

- UINT8 IaxEnable

  Offset 0-27 : Config Block Header.
- UINT8 Reserved [3]

  Offset 29 : Reserved for 4 bytes of alignment.

12.81.1 Detailed Description

IAX PEI configuration **Revision 1**:

- Initial version.

Definition at line 35 of file IaxConfig.h.

12.81.2 Member Data Documentation

12.81.2.1 IaxEnable

```
UINT8 IAX_PEI_CONFIG::IaxEnable
```

Offset 0-27 : Config Block Header.

Offset 28 It controls enabling/disabling IAX device if supported and not fused off. If FALSE, all other polices can be ignored. **1=TRUE**; 0=FALSE.

Definition at line 44 of file IaxConfig.h.

The documentation for this struct was generated from the following file:

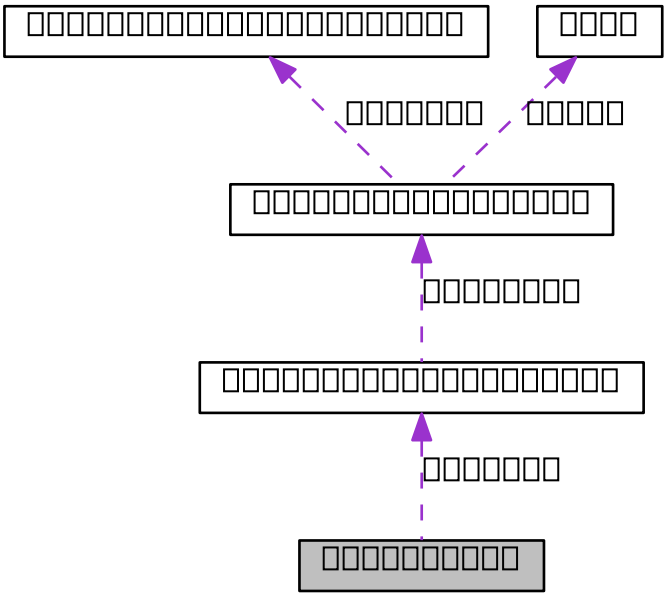
- IaxConfig.h

12.82 IEH_CONFIG Struct Reference

The IEH_CONFIG block describes the expected configuration of the PCH Integrated Error Handler.

`#include <IehConfig.h>`

Collaboration diagram for IEH_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header
  Config Block Header.
- UINT32 Mode: 1
  *IEH mode **0: Bypass Mode**; 1: Enable.*
- UINT32 RsvdBits0: 31
  Reserved bits.

12.82.1 Detailed Description

The IEH_CONFIG block describes the expected configuration of the PCH Integrated Error Handler.

Definition at line 39 of file IehConfig.h.

The documentation for this struct was generated from the following file:

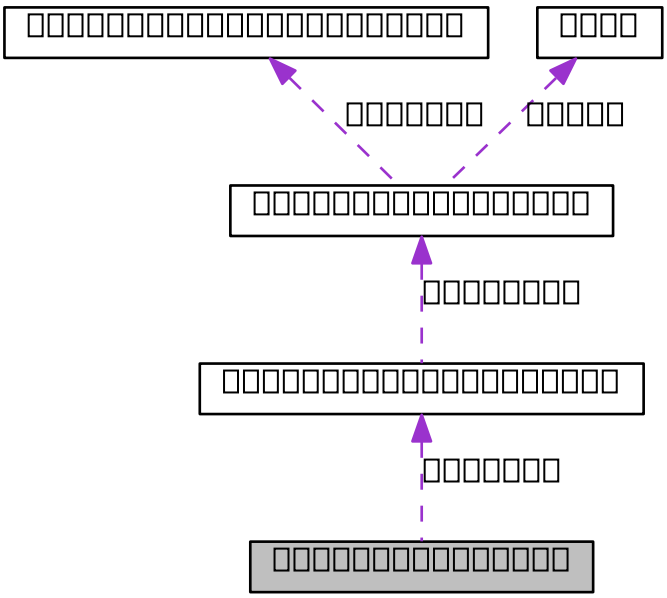
- IehConfig.h

12.83 IGPU_DXE_CONFIG Struct Reference

This configuration block is to configure IGD related variables used in DXE.

`#include <IGpuConfig.h>`

Collaboration diagram for IGPU_DXE_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Offset 0-27: Config Block Header.
- UINT32 Size

  Offset 28: This field gives the size of the GOP VBT Data buffer.
- EFI_PHYSICAL_ADDRESS VbtAddress

  Offset 32: This field points to the GOP VBT data buffer.
- CHAR16 GopVersion [0x10]

  Offset 40: This field holds the GOP Driver Version. It is an Output Protocol and updated by the Silicon code.
- UINT32 BCM1 [MAX_BCLM_ENTRIES]

  Offset 72: IGD Backlight Brightness Level Duty cycle Mapping Table for LFP1.
- UINT32 BCM2 [MAX_BCLM_ENTRIES]

  Offset 192: IGD Backlight Brightness Level Duty cycle Mapping Table for LFP2.
- UINT8 PlatformConfig

  *Offset 312: This field gives the Platform Configuration Information (0=Platform is S0ix Capable for ULT SKUs only, **1=Platform is not S0ix Capable**, 2=Force Platform is S0ix Capable for All SKUs)*
- UINT8 AlsEnable

  *Offset 313: Ambient Light Sensor Enable: **0=Disable**, 2=Enable.*
- UINT8 Reserved [6]

  Offset 314 : Reserved for QWORD alignment.

12.83.1 Detailed Description

This configuration block is to configure IGD related variables used in DXE.

If Intel IGPU Device is not supported or disabled, all policies will be ignored. The data elements should be initialized by a Platform Module.
Revision 1:

- Initial version.

Definition at line 303 of file IGpuConfig.h.

The documentation for this struct was generated from the following file:

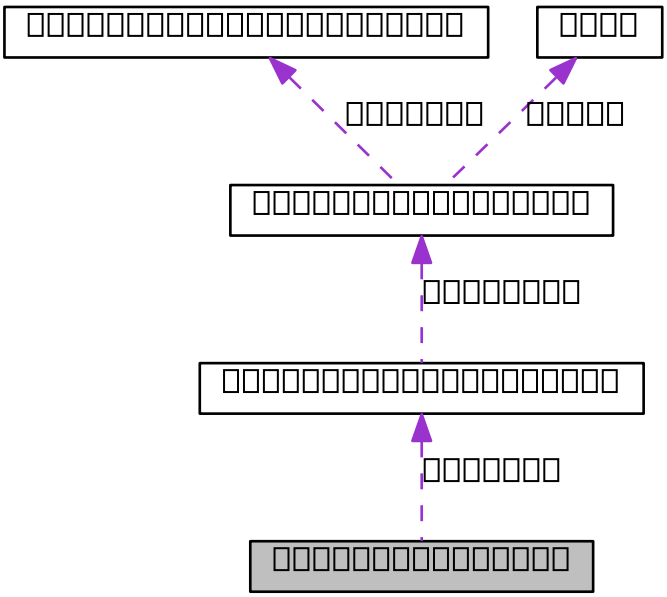
- IGpuConfig.h

12.84 IGPU_PEI_CONFIG Struct Reference

This configuration block is to configure IGD related variables used in PostMem PEI.

```
#include <IGpuConfig.h>
```

Collaboration diagram for IGPU_PEI_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Offset 0-27 Config Block Header.
- UINT8 RenderStandby

  *Offset 28 :**(Test)** This field is used to enable or disable RC6 (Render Standby): 0=FALSE, **1=TRUE***
- UINT8 PavpEnable

  *Offset 29 :IGD PAVP TRUE/FALSE: 0=FALSE, **1=TRUE***
- UINT8 MediaStandby

  *Offset 30 :This field is used to enable or disable MC6 (Media Standby): 0=FALSE, **1=TRUE***

12.84.1 Detailed Description

This configuration block is to configure IGD related variables used in PostMem PEI.

If Intel IGPU Device is not supported, all policies can be ignored. **Revision 1**:

- Initial version.

Definition at line 284 of file IGpuConfig.h.

The documentation for this struct was generated from the following file:

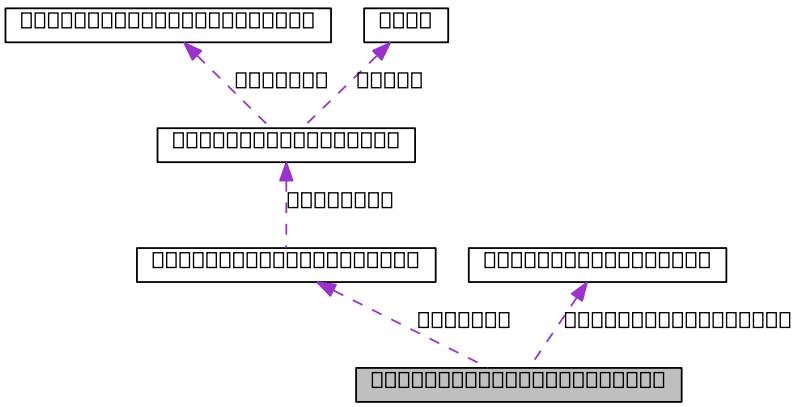
- IGpuConfig.h

12.85 IGPU_PEI_PREMEM_CONFIG Struct Reference

This Configuration block is to configure GT related PreMem data/variables.

`#include <IGpuConfig.h>`

Collaboration diagram for IGPU_PEI_PREMEM_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Offset 0-27 Config Block Header.
- UINT8 PrimaryDisplay

  *Offset 28 Selection of the primary display device: 0=IGPU, 1=PEG, 2=PCIe Graphics on PCH, **3=AUTO**, 4=Switchable Graphics*
  When AUTO mode selected, the priority of display devices is: PCIe Graphics on PCH > PEG > IGPU.
- UINT8 InternalGraphics

  Offset 29 Intel IGPU Support.
- UINT16 IgdDvmt50PreAlloc

  Offset 30 Pre-allocated memory for IGPU
  0 = 0MB,1 or 247 = 32MB,
  2 = 64MB, 3 = 128MB,
  240 = 4MB, 241 = 8MB,
  242 = 12MB, 243 = 16MB,
  244 = 20MB, 245 = 24MB,
  246 = 28MB, 248 = 36MB,
  249 = 40MB, 250 = 44MB,
  251 = 48MB, 252 = 52MB,
  *253 = 56MB, **254 = 60MB**,*
  Note: enlarging pre-allocated memory for IGPU may need to reduce MmioSize because of 4GB boundary limitation
- UINT8 PanelPowerEnable

  *Offset 32 :**(Test)** Control for enabling/disabling VDD force bit (Required only for early enabling of eDP panel): 0=FALSE, **1=TRUE***
- UINT8 MemoryBandwidthCompression

  *Offset 33 This policy is used to enable/disable Memory Bandwidth Compression **0- Disable**, 1- Enable.*
- UINT16 DeltaT12PowerCycleDelay

  *Offset 34 Power Cycle Delay required for eDP as per VESA standard.0 - 0 ms, **0xFFFF - Auto calculate to max 500 ms**< >*
- UINT32 VbtSize

  Size of VBT data.
- UINT64 GttMmAdr

  *Offset 36 Temp Address of Graphics GTTMMADR: Default is **0xAF000000***
- UINT64 LMemBar

  *Offset 44 Temp Address of Graphics LMEMBAR: Default is **0xB0000000***
- DDI_CONFIGURATION DdiConfiguration

  Offset 52 DDI configuration, need to match with VBT settings.
- UINT8 OemT12DelayOverride

  *Offset 68 :Oem T12 Delay Override **Disable**< >**,Enable-Enable T12 Time.***
- UINT8 IGpuGsm2Size

  Offset 68 {0:2GB, 1:4GB, 2:6GB, 3:8GB, 4:10GB, 5:12GB, 6:14GB, 7:16GB, 8:18GB, 9:20GB, 10:22GB, 11:24GB, 12:26GB, 13:28GB, 14:30GB, 15:32GB, 0xFF/Other Value:No Allocation}.
- UINT8 TestRsvd0 [4]

  Offset 69 Reserved for 8 bytes alignment.
- BOOLEAN LidStatus

  *Lid Status: 0=Close, **1=Open***
- UINT8 VgaInitControl

  BIT0 - 0 : Disable VGA Support, 1 : Enable VGA Support BIT1 - 0 : VGA Text Mode 3, 1 : VGA Graphics Mode 12 BIT2 - 0 : No Extended SOL Support (Disable VGA after Memory training done), 1 : Extended SOL Support BIT3 - 0 : VGA Init During Display Init, 1 - VGA Init During MRC Cold Boot.
- VOID * VgaMessage

  Pointer to Message which should be displayed.
- VOID * VbtPtr

Address of the Graphics Configuration Table.

- VGA_GRAPHICS_MODE12_INFO GraphicsMode12Info

  Offset 82 VGA Mode 12 Information.

- UINT8 * GraphicsMode12FontPtr

  Pointer to VGA Mode 12 Font Data.

- MRC_ERROR_KEY_VALUE_TABLE * MrcErrorKeyValueTablePtr

  Pointer to MRC Error Key Value Table.

- UINT8 Reserved [16]

  Reserved space for future use.

12.85.1 Detailed Description

This Configuration block is to configure GT related PreMem data/variables.

Revision 1:

- Initial version.

Definition at line 185 of file IGpuConfig.h.

12.85.2 Member Data Documentation

12.85.2.1 IgdDvmt50PreAlloc

```
UINT16 IGPU_PEI_PREMEM_CONFIG::IgdDvmt50PreAlloc
```

Offset 30 Pre-allocated memory for IGPU
0 = 0MB,1 or 247 = 32MB,
2 = 64MB, 3 = 128MB,
240 = 4MB, 241 = 8MB,
242 = 12MB, 243 = 16MB,
244 = 20MB, 245 = 24MB,
246 = 28MB, 248 = 36MB,
249 = 40MB, 250 = 44MB,
251 = 48MB, 252 = 52MB,
253 = 56MB, **254 = 60MB**,
Note: enlarging pre-allocated memory for IGPU may need to reduce MmioSize because of 4GB boundary limitation

Offset 30 >

Definition at line 220 of file IGpuConfig.h.

12.85.2.2 InternalGraphics

```
UINT8 IGPU_PEI_PREMEM_CONFIG::InternalGraphics
```

Offset 29 Intel IGPU Support.

It controls enabling/disabling IGPU device. When AUTO mode selected, IGPU will be turned off when external graphics detected. If FALSE, all other polices can be ignored. **2 = AUTO**; 0 = FALSE; 1 = TRUE.

Definition at line 204 of file IGpuConfig.h.

12.85.2.3 VgaInitControl

```
UINT8 IGPU_PEI_PREMEM_CONFIG::VgaInitControl
```

BIT0 - 0 : Disable VGA Support, 1 : Enable VGA Support BIT1 - 0 : VGA Text Mode 3, 1 : VGA Graphics Mode 12 BIT2 - 0 : No Extended SOL Support (Disable VGA after Memory training done), 1 : Extended SOL Support BIT3 - 0 : VGA Init During Display Init, 1 - VGA Init During MRC Cold Boot.

BIT4 - 0 : Enable Progress Bar, 1 : Disable Progress Bar BIT5 - 0 : VGA Mode 12 16 Color Support, 1 : VGA Mode 12 Monochrome Black and White Support BIT6-7 - 0 : No Higher Cd Clock, 1 : 442 MHz, 2 : 461 MHz, 3 : Reserved Offset 74 VGA Init Control

Definition at line 242 of file IGpuConfig.h.

The documentation for this struct was generated from the following file:

- IGpuConfig.h

12.86 IOM_AUX_ORI_GPIO_PAD_CONFIG Struct Reference

The IOM_AUX_ORI_GPIO_PAD_CONFIG describes IOM TypeC port map GPIO pin.

```
#include <TcssPeiPreMemConfig.h>
```

Public Attributes

- UINT32 ReceptacleSbu1BiasCtrl

  SBU1 Pin number that is for IOM indicate the pull up pin from TypeC port.
- UINT32 ReceptacleSbu2BiasCtrl

  SBU2 Pin number that is for IOM indicate the pull down pin from TypeC port.
- UINT32 AuxIsoCtrl

  AUX ISO control pin GPIO number that is IOM indicate the AUX active from SOC.

12.86.1 Detailed Description

The IOM_AUX_ORI_GPIO_PAD_CONFIG describes IOM TypeC port map GPIO pin.

Those GPIO setting for DP Aux Orientation Bias Control when the TypeC port didn't have re-timer. IOM needs to know SBU1 and SBU2 pins for Bias control

Definition at line 79 of file TcssPeiPreMemConfig.h.

The documentation for this struct was generated from the following file:

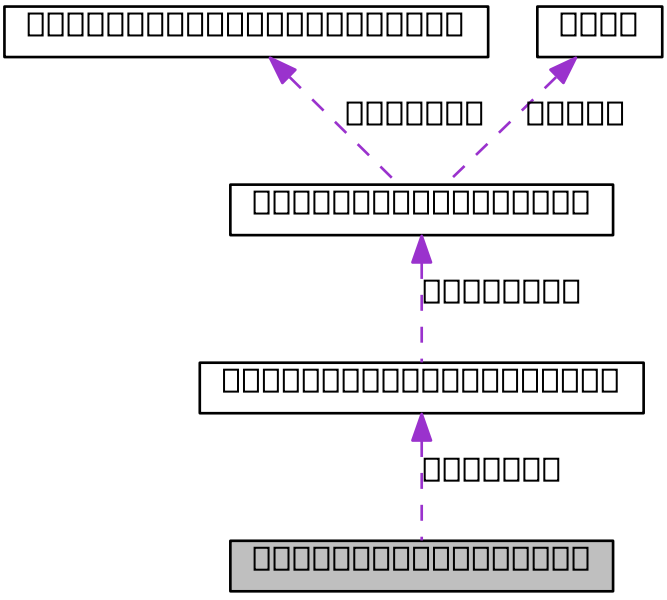
- TcssPeiPreMemConfig.h

12.87 IPU_PREMEM_CONFIG Struct Reference

IPU PreMem configuration
Revision 1:

```
#include <IpuPreMemConfig.h>
```

Collaboration diagram for IPU_PREMEM_CONFIG:



Public Attributes

- UINT8 IpuEnable

  Config Block Header.
- UINT8 ImguClkOutEn [GPIO_IMGUCLK_NUMBER_OF_PINS]

  It enable the IMGU CLKOUT.

12.87.1 Detailed Description

IPU PreMem configuration
Revision 1:

- Initial version.

Definition at line 48 of file IpuPreMemConfig.h.

12.87.2 Member Data Documentation

12.87.2.1 ImguClkOutEn

```
UINT8 IPU_PREMEM_CONFIG::ImguClkOutEn[GPIO_IMGUCLK_NUMBER_OF_PINS]
```

It enable the IMGU CLKOUT.

TRUE FALSE

Definition at line 62 of file IpuPreMemConfig.h.

12.87.2.2 IpuEnable

```
UINT8 IPU_PREMEM_CONFIG::IpuEnable
```

Config Block Header.

(Test) It enables the SA IPU Device if supported and not fused off. If FALSE, all other policies in this config block will be ignored. **1=TRUE**; 0=FALSE.

Definition at line 56 of file IpuPreMemConfig.h.

The documentation for this struct was generated from the following file:

- IpuPreMemConfig.h

12.88 IPv4_ADDRESS Struct Reference

4-byte buffer.

```
#include <Base.h>
```

12.88.1 Detailed Description

4-byte buffer.

An IPv4 internet protocol address.

Definition at line 223 of file Base.h.

The documentation for this struct was generated from the following file:

- Base.h

12.89 IPv6_ADDRESS Struct Reference

16-byte buffer.

```
#include <Base.h>
```

12.89.1 Detailed Description

16-byte buffer.

An IPv6 internet protocol address.

Definition at line 230 of file Base.h.

The documentation for this struct was generated from the following file:

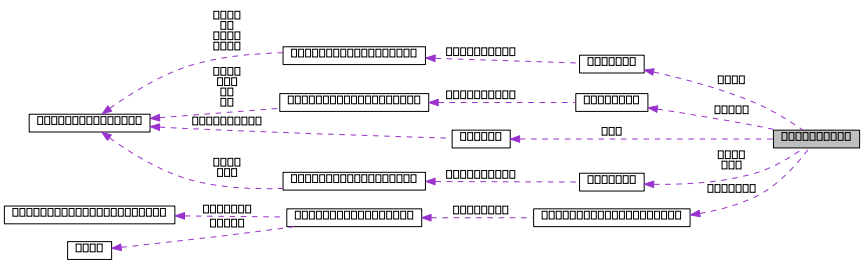
- Base.h

12.90 ISH_CONFIG Struct Reference

The ISH_CONFIG block describes Integrated Sensor Hub device.

```
#include <IshConfig.h>
```

Collaboration diagram for ISH_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT32 PdtUnlock: 1

  *ISH PDT Unlock Msg: **0: False** 1: True.*
- UINT32 MsiInterrupt: 1

  *ISH MSI Interrupts: **0: False** 1: True.*
- UINT32 RsvdBits0: 30

  Reserved Bits.

12.90.1 Detailed Description

The ISH_CONFIG block describes Integrated Sensor Hub device.

Definition at line 122 of file IshConfig.h.

The documentation for this struct was generated from the following file:

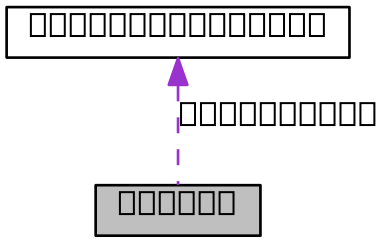
- IshConfig.h

12.91 ISH_GP Struct Reference

Struct contains GPIO pins assigned and signal settings of GP.

`#include <IshConfig.h>`

Collaboration diagram for ISH_GP:



Public Attributes

- UINT32 Enable: 1

  *ISH GP GPIO pins assigned: **0: False** 1: True.*
- UINT32 RsvdBits0: 31

  Reserved Bits.

12.91.1 Detailed Description

Struct contains GPIO pins assigned and signal settings of GP.

Definition at line 113 of file IshConfig.h.

The documentation for this struct was generated from the following file:

- IshConfig.h

12.92 ISH_GPIO_CONFIG Struct Reference

ISH GPIO settings.

`#include <IshConfig.h>`

Public Attributes

- UINT32 PinMux
  GPIO signals pin muxing settings.
- UINT32 PadTermination
  GPIO Pads Internal Termination.

12.92.1 Detailed Description

ISH GPIO settings.

Definition at line 34 of file IshConfig.h.

12.92.2 Member Data Documentation

12.92.2.1 PadTermination

`UINT32 ISH_GPIO_CONFIG::PadTermination`

GPIO Pads Internal Termination.

For more information please see Platform Design Guide. Check GPIO_ELECTRICAL_CONFIG for reference

Definition at line 46 of file IshConfig.h.

12.92.2.2 PinMux

```
UINT32 ISH_GPIO_CONFIG::PinMux
```

GPIO signals pin muxing settings.

If signal can be enable only on a single pin then this parameter should be set to 0. Refer to GPIO_∗_MUXI↩ NG_ISH_∗x_∗ in GpioPins∗.h for supported settings on a given platform GPIO Pin mux configuration. Refer to GPIO_∗_MUXING_ISH_∗x_MOSI_∗

Definition at line 40 of file IshConfig.h.

The documentation for this struct was generated from the following file:

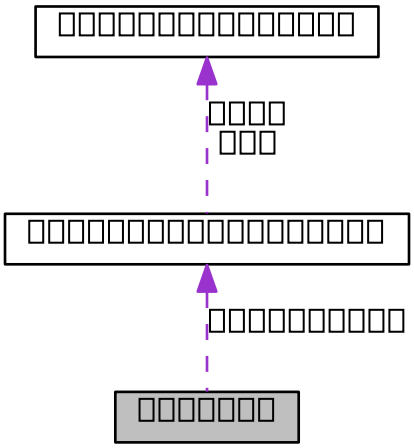
- IshConfig.h

12.93 ISH_I2C Struct Reference

Struct contains GPIO pins assigned and signal settings of I2C.

```
#include <IshConfig.h>
```

Collaboration diagram for ISH_I2C:



Public Attributes

- UINT32 Enable: 1

  ISH I2C GPIO pins assigned: ***0: False*** *1: True.*
- UINT32 RsvdBits0: 31

  Reserved Bits.

12.93.1 Detailed Description

Struct contains GPIO pins assigned and signal settings of I2C.

Definition at line 103 of file IshConfig.h.

The documentation for this struct was generated from the following file:

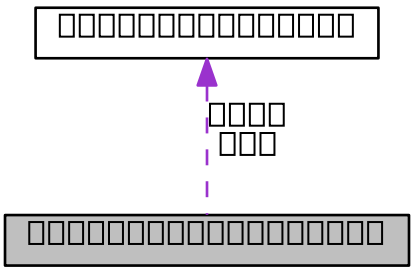
- IshConfig.h

12.94 ISH_I2C_PIN_CONFIG Struct Reference

I2C signals settings.

`#include <IshConfig.h>`

Collaboration diagram for ISH_I2C_PIN_CONFIG:



Public Attributes

- ISH_GPIO_CONFIG Sda
  SDA Pin configuration.
- ISH_GPIO_CONFIG Scl
  SCL Pin configuration.

12.94.1 Detailed Description

I2C signals settings.

Definition at line 74 of file IshConfig.h.

The documentation for this struct was generated from the following file:

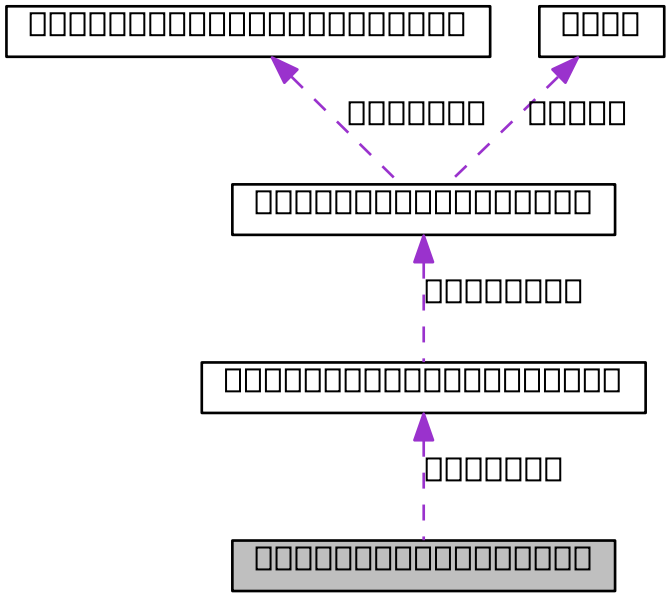
- IshConfig.h

12.95 ISH_PREMEM_CONFIG Struct Reference

Premem Policy for Integrated Sensor Hub device.

`#include <IshConfig.h>`

Collaboration diagram for ISH_PREMEM_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header
  Config Block Header.
- UINT32 Enable: 1
  *ISH Controler 0: Disable; **1: Enable**.*
- UINT32 RsvdBits0: 31
  Reserved Bits.

12.95.1 Detailed Description

Premem Policy for Integrated Sensor Hub device.

Definition at line 138 of file IshConfig.h.

12.95.2 Member Data Documentation

12.95.2.1 Enable

```
UINT32 ISH_PREMEM_CONFIG::Enable
```

ISH Controler 0: Disable; **1: Enable**.

For Desktop sku, the ISH POR should be disabled. **0:Disable** .

Definition at line 144 of file IshConfig.h.

The documentation for this struct was generated from the following file:

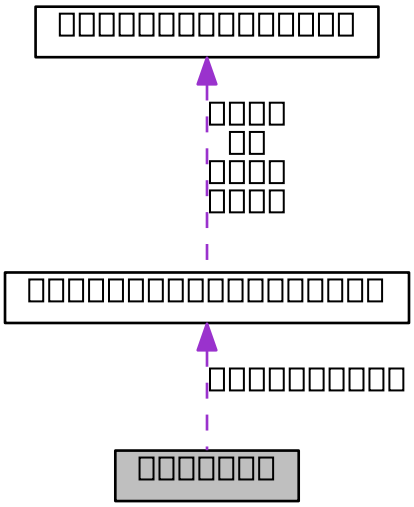
- IshConfig.h

12.96 ISH_SPI Struct Reference

Struct contains GPIO pins assigned and signal settings of SPI.

```
#include <IshConfig.h>
```

Collaboration diagram for ISH_SPI:



Public Attributes

- UINT8 Enable

  *ISH SPI GPIO pins assigned: **0: False** 1: True.*
- UINT8 CsEnable [PCH_MAX_ISH_SPI_CS_PINS]

  *ISH SPI CS pins assigned: **0: False** 1: True.*
- UINT16 RsvdField0

  Reserved field.

12.96.1 Detailed Description

Struct contains GPIO pins assigned and signal settings of SPI.

Definition at line 83 of file IshConfig.h.

The documentation for this struct was generated from the following file:

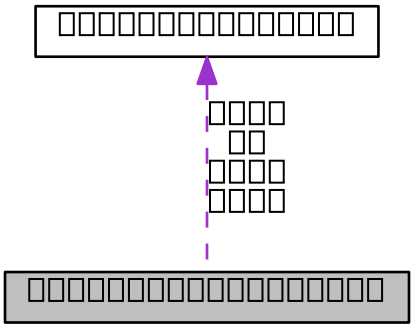
- IshConfig.h

12.97 ISH_SPI_PIN_CONFIG Struct Reference

SPI signals settings.

```
#include <IshConfig.h>
```

Collaboration diagram for ISH_SPI_PIN_CONFIG:



Public Attributes

- ISH_GPIO_CONFIG Mosi

  MOSI Pin configuration.
- ISH_GPIO_CONFIG Miso

  MISO Pin configuration.
- ISH_GPIO_CONFIG Clk

  CLK Pin configuration.
- ISH_GPIO_CONFIG Cs [PCH_MAX_ISH_SPI_CS_PINS]

  CS Pin configuration.

12.97.1 Detailed Description

SPI signals settings.

Definition at line 52 of file IshConfig.h.

The documentation for this struct was generated from the following file:

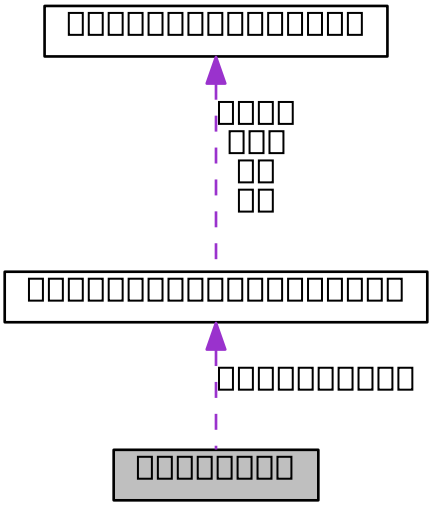
- IshConfig.h

12.98 ISH_UART Struct Reference

Struct contains GPIO pins assigned and signal settings of UART.

```
#include <IshConfig.h>
```

Collaboration diagram for ISH_UART:



Public Attributes

- UINT32 Enable: 1

  ISH UART GPIO pins assigned: ***0: False*** *1: True.*
- UINT32 RsvdBits0: 31

  Reserved Bits.

12.98.1 Detailed Description

Struct contains GPIO pins assigned and signal settings of UART.

Definition at line 94 of file IshConfig.h.

The documentation for this struct was generated from the following file:

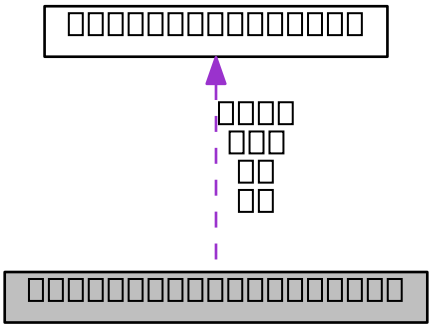
- IshConfig.h

12.99 ISH_UART_PIN_CONFIG Struct Reference

UART signals settings.

```
#include <IshConfig.h>
```

Collaboration diagram for ISH_UART_PIN_CONFIG:



Public Attributes

- ISH_GPIO_CONFIG Rx
  RXD Pin configuration.
- ISH_GPIO_CONFIG Tx
  TXD Pin configuration.
- ISH_GPIO_CONFIG Rts
  RTS Pin configuration.
- ISH_GPIO_CONFIG Cts
  CTS Pin configuration.

12.99.1 Detailed Description

UART signals settings.

Definition at line 63 of file IshConfig.h.

The documentation for this struct was generated from the following file:

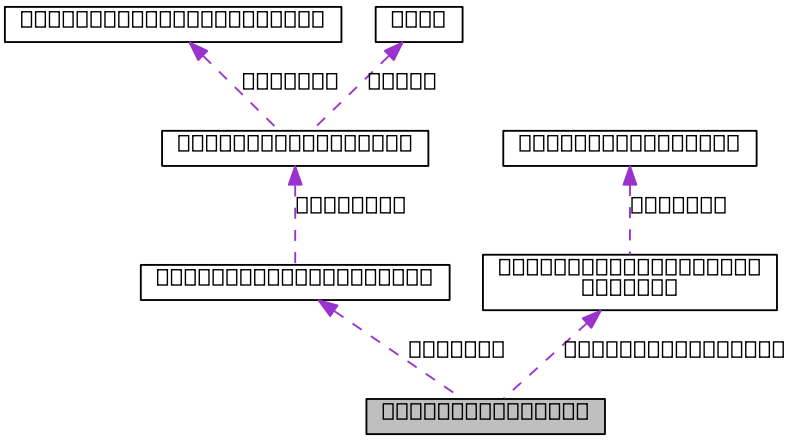
- IshConfig.h

12.100 LPSS_I2C_CONFIG Struct Reference

The LPSS_I2C_CONFIG block provides the configurations to set the Serial IO controllers.

`#include <LpssI2cConfig.h>`

Collaboration diagram for LPSS_I2C_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header
  Config Block Header.
- LPSS_I2C_CONTROLLER_CONFIG I2cDeviceConfig [PCH_MAX_SERIALIO_I2C_CONTROLLERS]
  I2C Device Configuration.

12.100.1 Detailed Description

The LPSS_I2C_CONFIG block provides the configurations to set the Serial IO controllers.

Revision 1:

- Inital version.

Definition at line 86 of file LpssI2cConfig.h.

The documentation for this struct was generated from the following file:

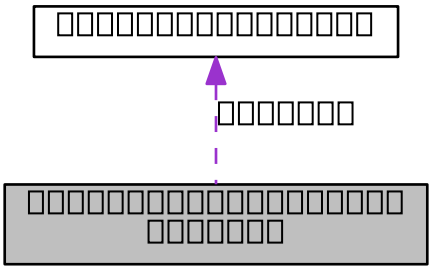
- LpssI2cConfig.h

12.101 LPSS_I2C_CONTROLLER_CONFIG Struct Reference

Serial IO I2C Controller Configuration.

`#include <LpssI2cConfig.h>`

Collaboration diagram for LPSS_I2C_CONTROLLER_CONFIG:



Public Attributes

- UINT8 PadTermination

  LpssI2cPci see LPSS_I2C_I2C_MODE
- LPSS_I2C_PIN_MUX PinMux

  I2C pin mux configuration.

12.101.1 Detailed Description

Serial IO I2C Controller Configuration.

Definition at line 65 of file LpssI2cConfig.h.

12.101.2 Member Data Documentation

12.101.2.1 PadTermination

`UINT8 LPSS_I2C_CONTROLLER_CONFIG::PadTermination`

LpssI2cPci see LPSS_I2C_I2C_MODE

I2C Pads Internal Termination. For more information please see Platform Design Guide. Supported values (check GPIO_ELECTRICAL_CONFIG for reference): **GpioTermNone: No termination**, GpioTermWpu1K: 1kOhm weak pull-up, GpioTermWpu5K: 5kOhm weak pull-up, GpioTermWpu20K: 20kOhm weak pull-up

Definition at line 76 of file LpssI2cConfig.h.

The documentation for this struct was generated from the following file:

- LpssI2cConfig.h

12.102 LPSS_I2C_PIN_MUX Struct Reference

I2C signals pin muxing settings.

`#include <LpssI2cConfig.h>`

Public Attributes

- UINT32 Sda

  SDA Pin mux configuration. Refer to GPIO_∗_MUXING_SERIALIO_I2Cx_SDA_∗.
- UINT32 Scl

  SCL Pin mux configuration. Refer to GPIO_∗_MUXING_SERIALIO_I2Cx_SCL_∗.

12.102.1 Detailed Description

I2C signals pin muxing settings.

If signal can be enable only on a single pin then this parameter is ignored by RC. Refer to GPIO_∗_MUXING_SE↩ RIALIO_I2Cx_∗ in GpioPins∗.h for supported settings on a given platform

Definition at line 57 of file LpssI2cConfig.h.

The documentation for this struct was generated from the following file:

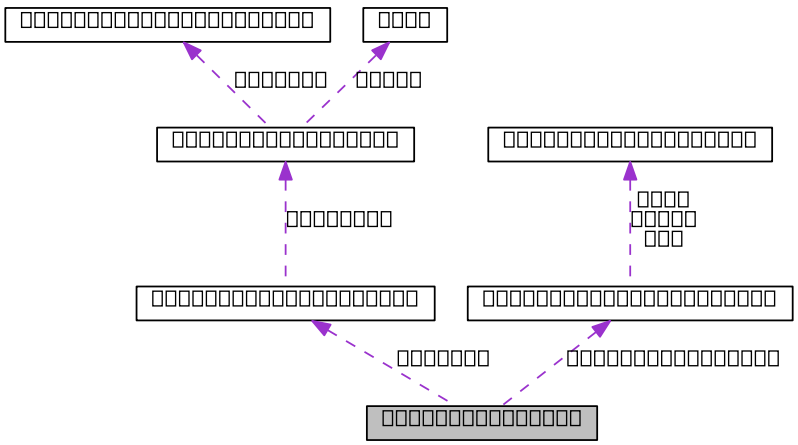
- LpssI2cConfig.h

12.103 LPSS_I3C_CONFIG Struct Reference

The LPSS_I3C_CONFIG block provides the configurations to set the Serial IO controllers.

`#include <LpssI3cConfig.h>`

Collaboration diagram for LPSS_I3C_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- LPSS_I3C_DEVICE_CONFIG I3cDeviceConfig [PCH_MAX_SERIALIO_I3C_CONTROLLERS]

  I3C Device Configuration.

12.103.1 Detailed Description

The LPSS_I3C_CONFIG block provides the configurations to set the Serial IO controllers.

Revision 1:

- Inital version.

Definition at line 78 of file LpssI3cConfig.h.

The documentation for this struct was generated from the following file:

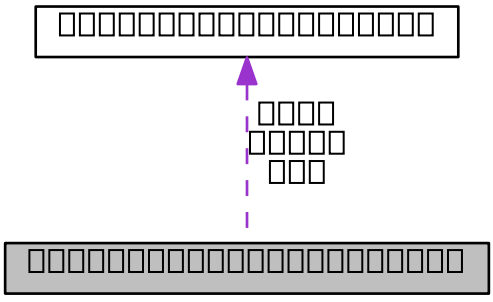
- LpssI3cConfig.h

12.104 LPSS_I3C_DEVICE_CONFIG Struct Reference

The LPSS_I3C_DEVICE_CONFIG provides the configurations for Serial IO I3C controller.

`#include <LpssI3cConfig.h>`

Collaboration diagram for LPSS_I3C_DEVICE_CONFIG:



Public Attributes

- UINT8 Mode

  <b<I3cPci ***see I3C_MODE***
- UINT8 Reserved [3]

  Reserved bytes.

12.104.1 Detailed Description

The LPSS_I3C_DEVICE_CONFIG provides the configurations for Serial IO I3C controller.

Definition at line 64 of file LpssI3cConfig.h.

The documentation for this struct was generated from the following file:

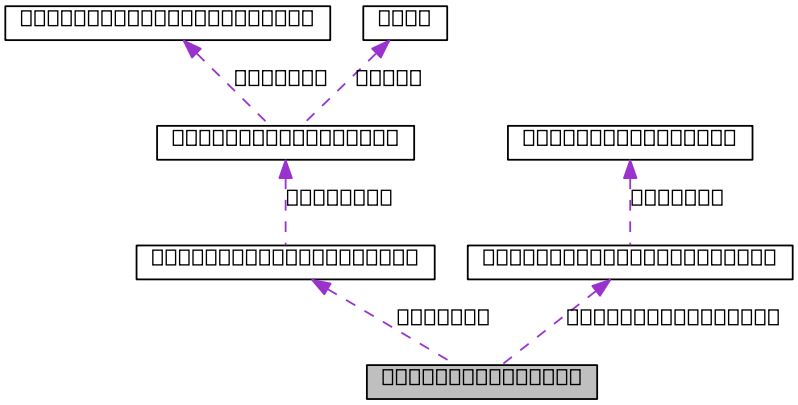
- LpssI3cConfig.h

12.105 LPSS_SPI_CONFIG Struct Reference

The LPSS_SPI_CONFIG block provides the configurations to set the LPSS SPI controllers **Revision 1:**

```
#include <LpssSpiConfig.h>
```

Collaboration diagram for LPSS_SPI_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header
  Config Block Header.
- LPSS_SPI_DEVICE_CONFIG SpiDeviceConfig [PCH_MAX_LPSS_SPI_CONTROLLERS]
  LPSS SPI Device Configuration.

12.105.1 Detailed Description

The LPSS_SPI_CONFIG block provides the configurations to set the LPSS SPI controllers **Revision 1:**

- Inital version.

Definition at line 94 of file LpssSpiConfig.h.

The documentation for this struct was generated from the following file:

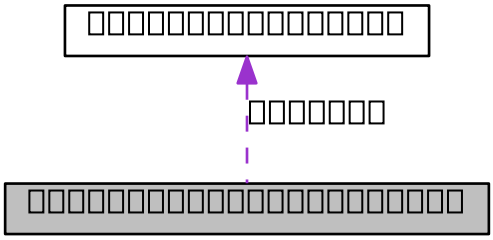
- LpssSpiConfig.h

12.106 LPSS_SPI_DEVICE_CONFIG Struct Reference

The LPSS_SPI_CONFIG provides the configurations to set the Serial IO SPI controller.

`#include <LpssSpiConfig.h>`

Collaboration diagram for LPSS_SPI_DEVICE_CONFIG:



Public Attributes

- UINT8 Mode

  LpssSpiPci *see LPSS_SPI_MODE*
- UINT8 DefaultCsOutput

  0 = CS0 *CS1, CS2, CS3. Default CS used by the SPI HC*
- UINT8 CsPolarity [PCH_MAX_LPSS_SPI_CHIP_SELECTS]

  Selects SPI ChipSelect signal polarity, 0 = low ***1 = High***
- UINT8 CsEnable [PCH_MAX_LPSS_SPI_CHIP_SELECTS]

  0 = Enable *1 = Disable. Based on this setting GPIO for given SPIx CSx will be configured in Native mode*
- UINT8 CsMode

  0 = HW Control *1 = SW Control. Sets Chip Select Control mode Hardware or Software.*
- UINT8 CsState

  0 = CS is set to low *1 = CS is set to high*
- LPSS_SPI_PIN_MUX PinMux

  SPI Pinmux configuration.

12.106.1 Detailed Description

The LPSS_SPI_CONFIG provides the configurations to set the Serial IO SPI controller.

Definition at line 78 of file LpssSpiConfig.h.

The documentation for this struct was generated from the following file:

- LpssSpiConfig.h

12.107 LPSS_SPI_PIN_MUX Struct Reference

SPI signals pin muxing settings.

`#include <LpssSpiConfig.h>`

Public Attributes

- UINT32 Cs [PCH_MAX_LPSS_SPI_CHIP_SELECTS]

  CS Pin mux configuration. Refer to GPIO_∗_MUXING_SERIALIO_SPIx_CS_∗.
- UINT32 Clk

  CLK Pin mux configuration. Refer to GPIO_∗_MUXING_SERIALIO_SPIx_CLK_∗.
- UINT32 Miso

  MISO Pin mux configuration. Refer to GPIO_∗_MUXING_SERIALIO_SPIx_MISO_∗.
- UINT32 Mosi

  MOSI Pin mux configuration. Refer to GPIO_∗_MUXING_SERIALIO_SPIx_MOSI_∗.

12.107.1 Detailed Description

SPI signals pin muxing settings.

If signal can be enable only on a single pin then this parameter is ignored by RC. Refer to GPIO_∗_MUXING_SE↩ RIALIO_SPIx_∗ in GpioPins∗.h for supported settings on a given platform

Definition at line 68 of file LpssSpiConfig.h.

The documentation for this struct was generated from the following file:

- LpssSpiConfig.h

12.108 LPSS_UART_ATTRIBUTES Struct Reference

UART Settings.

`#include <LpssUartConfig.h>`

Public Attributes

- UINT32 BaudRate

  115200 *Max 6000000 MdePkg.dec PcdUartDefaultBaudRate*
- UINT8 Parity

  1 - No Parity *see EFI_PARITY_TYPE MdePkg.dec PcdUartDefaultParity*
- UINT8 DataBits

  8 *MdePkg.dec PcdUartDefaultDataBits*
- UINT8 StopBits

  1 - One Stop Bit *see EFI_STOP_BITS_TYPE MdePkg.dec PcdUartDefaultStopBits*
- UINT8 AutoFlow

  FALSE *IntelFrameworkModulePkg.dsc PcdIsaBusSerialUseHalfHandshake*

12.108.1 Detailed Description

UART Settings.

Definition at line 94 of file LpssUartConfig.h.

The documentation for this struct was generated from the following file:

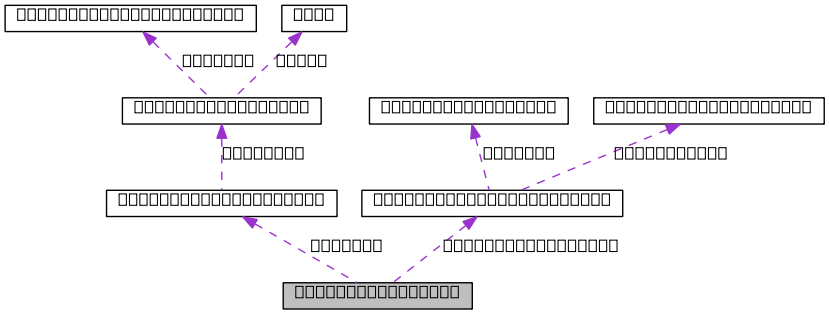
- LpssUartConfig.h

12.109 LPSS_UART_CONFIG Struct Reference

The LPSS_UART_CONFIG block provides the configurations to set the LPSS UART controllers **Revision 1:**

`#include <LpssUartConfig.h>`

Collaboration diagram for LPSS_UART_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header
  Config Block Header.
- LPSS_UART_DEVICE_CONFIG UartDeviceConfig [PCH_MAX_LPSS_UART_CONTROLLERS]
  LPSS UART Device Configuration.

12.109.1 Detailed Description

The LPSS_UART_CONFIG block provides the configurations to set the LPSS UART controllers **Revision 1:**

- Inital version.

Definition at line 133 of file LpssUartConfig.h.

The documentation for this struct was generated from the following file:

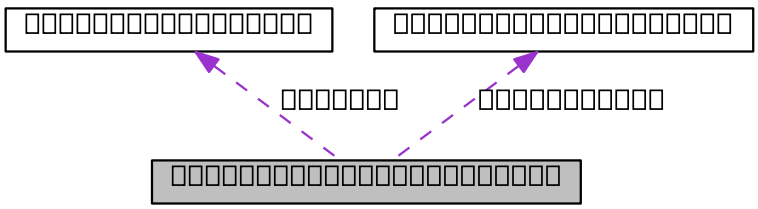
- LpssUartConfig.h

12.110 LPSS_UART_DEVICE_CONFIG Struct Reference

Lpss UART Controller Configuration.

```
#include <LpssUartConfig.h>
```

Collaboration diagram for LPSS_UART_DEVICE_CONFIG:



Public Attributes

- LPSS_UART_ATTRIBUTES Attributes

  see LPSS_UART_ATTRIBUTES
- LPSS_UART_PIN_MUX PinMux

  UART pin mux configuration.
- UINT8 Mode

  LpssUartPci *see LPSS_UART_MODE*
- UINT8 DBG2

  FALSE *If TRUE adds UART to DBG2 table and overrides UartPg to LpssUartPgDisabled*
- UINT8 PowerGating

  LpssUartPgAuto *Applies to Hidden/COM/SkipInit see LPSS_UART_PG*
- UINT8 DebugPowerGating

  Applies to ACPI DBG2 specific case
- UINT8 DmaEnable

  TRUE *Applies to LpssUartPci only. Informs OS driver to use DMA, if false it will run in PIO mode*

12.110.1 Detailed Description

Lpss UART Controller Configuration.

Definition at line 117 of file LpssUartConfig.h.

The documentation for this struct was generated from the following file:

- LpssUartConfig.h

12.111 LPSS_UART_PIN_MUX Struct Reference

UART signals pin muxing settings.

`#include <LpssUartConfig.h>`

Public Attributes

- UINT32 Rx

  RXD Pin mux configuration. Refer to GPIO_*_MUXING_SERIALIO_UARTx_RXD_*.
- UINT32 Tx

  TXD Pin mux configuration. Refer to GPIO_*_MUXING_SERIALIO_UARTx_TXD_*.
- UINT32 Rts

  RTS Pin mux configuration. Refer to GPIO_*_MUXING_SERIALIO_UARTx_RTS_*.
- UINT32 Cts

  CTS Pin mux configuration. Refer to GPIO_*_MUXING_SERIALIO_UARTx_CTS_*.

12.111.1 Detailed Description

UART signals pin muxing settings.

If signal can be enable only on a single pin then this parameter is ignored by RC. Refer to GPIO_*_MUXING_SE↩ RIALIO_UARTx_* in GpioPins*.h for supported settings on a given platform

Definition at line 107 of file LpssUartConfig.h.

The documentation for this struct was generated from the following file:

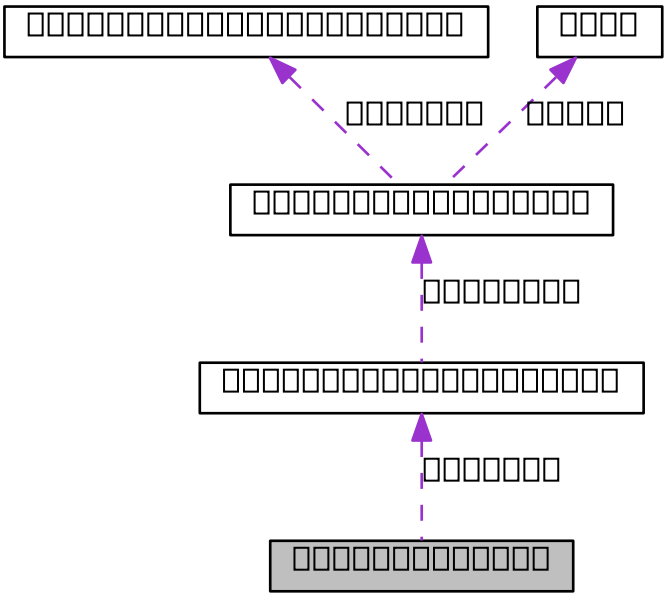
- LpssUartConfig.h

12.112 ME_DXE_CONFIG Struct Reference

ME Dxe Configuration Structure.

`#include <MeDxeConfig.h>`

Collaboration diagram for ME_DXE_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT32 MeFwDowngrade: 1

  ***0: Disabled**; 1: Enabled - Enable/Disable ME FW downgrade*
- UINT32 CoreBiosDoneEnabled: 1

  *0: Disabled; **1: Enabled** - Enable/Disable Sending notification that BIOS is starting to run 3rd party code*
- UINT32 AutoArbSvnCommit: 1

  *0: Disabled; **1: Enabled** - Enable/Disable attempt to perform Anti-Rollback SVN on each boot*
- UINT32 RsvdBits: 29

  Reserved for future use & Config block alignment.

12.112.1 Detailed Description

ME Dxe Configuration Structure.

Revision 1:

- Initial version.

Definition at line 35 of file MeDxeConfig.h.

The documentation for this struct was generated from the following file:

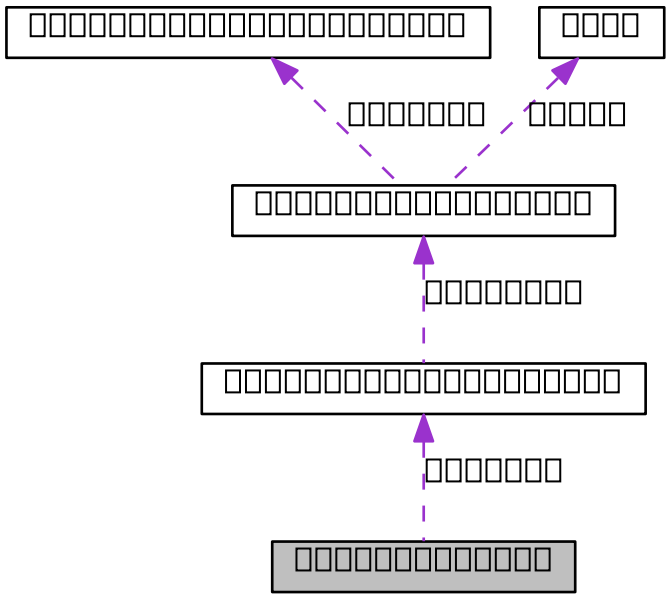
- MeDxeConfig.h

12.113 ME_PEI_CONFIG Struct Reference

ME Pei Post-Memory Configuration Structure.

`#include <MePeiConfig.h>`

Collaboration diagram for ME_PEI_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT32 EndOfPostMessage: 2

  0: Disabled; 1: Send in PEI; ***2: Send in DXE*** *- Send EOP at specific phase.*
- UINT32 DisableD0I3SettingForHeci: 1

  (Test) 0: Disable*; 1: Enable - Enable/Disable D0i3 for HECI.*
- UINT32 MeUnconfigOnRtcClear: 2

  Enable/Disable Me Unconfig On Rtc Clear.
- UINT32 MctpBroadcastCycle: 1

  (Test) 0: Disable*; 1: Enable - Program registers for MCTP Cycle.*
- UINT32 CseDataResilience: 2

  Enable/Disable CSME Data Resilience BIOS support.
- UINT32 SseCommunication: 2

  (Test) 0: POR; 1: Enable; 2: Disable - Enable/Disable SSE/SSE++ Devices.
- UINT32 PseEomFlowEnable: 1

  0: Disable*; 1: Enable - PSE EOM flow enable*
- UINT32 RsvdBits: 21

  Reserved for future use & Config block alignment.

12.113.1 Detailed Description

ME Pei Post-Memory Configuration Structure.

Revision 1:

- Initial version.

Definition at line 91 of file MePeiConfig.h.

12.113.2 Member Data Documentation

12.113.2.1 CseDataResilience

`UINT32 ME_PEI_CONFIG::CseDataResilience`

Enable/Disable CSME Data Resilience BIOS support.

It will force loading ME data default.

- 0: Disable
- **1: Enable** BIOS will send Data Clear HECI command in PEI if MFS failure is detected.
- 2: Enable and defer - BIOS will defer Data Clear HECI command to DXE if MFS failure is detected. but defer the data clear command to later boot phase

Definition at line 113 of file MePeiConfig.h.

12.113.2.2 MeUnconfigOnRtcClear

`UINT32 ME_PEI_CONFIG::MeUnconfigOnRtcClear`

Enable/Disable Me Unconfig On Rtc Clear.

If enabled, BIOS will send MeUnconfigOnRtcClearDisable Msg with parameter 0. It will cause ME to unconfig if RTC is cleared.

- 0: Disable
- **1: Enable**
- 2: Cmos is clear, status unkonwn
- 3: Reserved

Definition at line 104 of file MePeiConfig.h.

The documentation for this struct was generated from the following file:

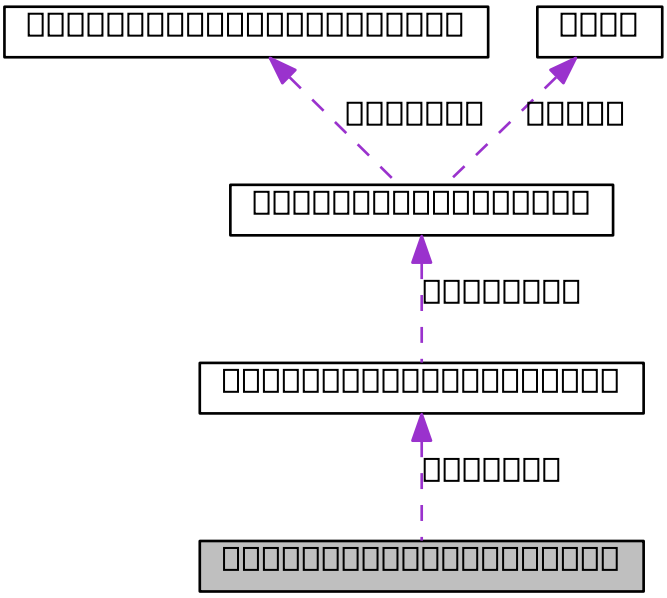
- MePeiConfig.h

12.114 ME_PEI_PREMEM_CONFIG Struct Reference

ME Pei Pre-Memory Configuration Structure.

`#include <MePeiConfig.h>`

Collaboration diagram for ME_PEI_PREMEM_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT32 HeciTimeouts: 1

  *0: Disable; **1: Enable** - HECI Send/Receive Timeouts.*
- UINT32 DidInitStat: 2

  ***(Test) 0: Disabled** 1: ME DID init stat 0 - Success 2: ME DID init stat 1 - No Memory in Channels 3: ME DID init stat 2 - Memory Init Error*
- UINT32 DisableCpuReplacedPolling: 1

  ***(Test) 0: Set to 0 to enable polling for CPU replacement** 1: Set to 1 will disable polling for CPU replacement*
- UINT32 DisableMessageCheck: 1

  ***(Test) 0: ME BIOS will check each messages before sending** 1: ME BIOS always sends messages without checking*
- UINT32 SkipMbpHob: 1

  ***(Test)** The SkipMbpHob policy determines whether ME BIOS Payload data will be requested during boot.*
- UINT32 HeciCommunication: 2

  ***(Test) 0: POR**; 1: Enable; 2: Disable - Enable/Disable HECI1.*
- UINT32 HeciCommunication2: 2

***(Test) 0: POR**; 1: Enable; 2: Disable - Enable/Disable HECI2.*

- UINT32 HeciCommunication3: 2

  ***(Test) 0: POR**; 1: Enable; 2: Disable - Enable/Disable HECI3.*

- UINT32 KtDeviceEnable: 2

  ***(Test) 0: POR**; 1: Enable; 2: Disable - Enable/Disable KT Device.*

- UINT32 SkipCpuReplacementCheck: 1

  ***(Test) 0: Disable**; 1: Enable - Enable/Disable to skip CPU replacement check.*

- UINT32 RsvdBits:17

  Reserved for future use & Config block alignment.

12.114.1 Detailed Description

ME Pei Pre-Memory Configuration Structure.

Revision 1:

- Initial version.

Definition at line 38 of file MePeiConfig.h.

12.114.2 Member Data Documentation

12.114.2.1 SkipMbpHob

```
UINT32 ME_PEI_PREMEM_CONFIG::SkipMbpHob
```

(Test) The SkipMbpHob policy determines whether ME BIOS Payload data will be requested during boot.

If set to 1, BIOS will not send Get MBP message and not create MBP Hob. **0: ME BIOS will send Get MBP message and create HOB for MBP data** 1: ME BIOS will not send Get MBP message

Definition at line 68 of file MePeiConfig.h.

The documentation for this struct was generated from the following file:

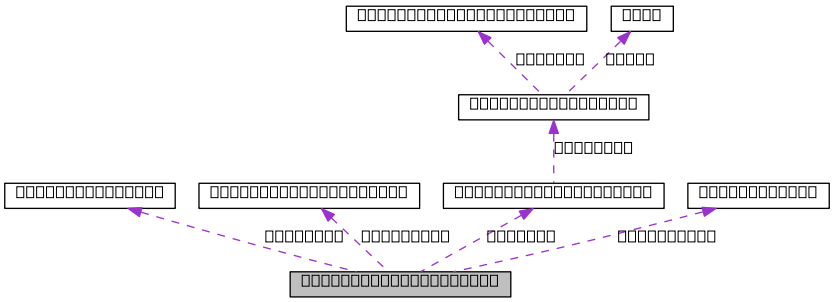
- MePeiConfig.h

12.115 MEMORY_CONFIG_NO_CRC Struct Reference

Memory Configuration The contents of this structure are not CRC'd by the MRC for option change detection.

`#include <MemoryConfig.h>`

Collaboration diagram for MEMORY_CONFIG_NO_CRC:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Offset 0-27 Config Block Header.
- UINT8 RsvdBytes151 [4]

  Offset 28-31 Reserved for alignment.
- SPD_DATA_BUFFER * SpdData

  Offset 32 Memory SPD data, will be used by the MRC when SPD SmBus address is zero.
- MEMORY_DQDQS_MAPPING * DqDqsMap

  Offset 40 LPDDR DQ bit and DQS byte swizzling between CPU and DRAM.
- MEMORY_RCOMP * RcompData

  Offset 48 DDR RCOMP resistors and target values.
- UINT64 PlatformMemorySize

  Offset 56 The minimum platform memory size required to pass control into DXE.
- UINT32 CleanMemory:1

  *Offset 64 Ask MRC to clear memory content: **FALSE=Do not Clear Memory**; TRUE=Clear Memory.*
- UINT8 SerialDebugLevel

  Sets the serial debug message level
  0x00 = Disabled
  0x01 = Errors only
  0x02 = Errors and Warnings
  0x03 = Errors, Warnings, and Info
  0x04 = Errors, Warnings, Info, and Events
  0x05 = Displays Memory Init Execution Time Summary only
  .
- UINT8 MemTestOnWarmBoot

  *Offset 69 Run Base Memory Test On WarmBoot: 0=Disabled, **1=Enabled***
- UINT8 DisableMrcRetraining

  Offset 70 Enable/Disable DisableMrcRetraining.
- UINT8 SpdAddressTable [MEM_CFG_MAX_SOCKETS]

  Offset 71 Memory DIMMs' SPD address for reading SPD data.

- UINT8 MrcBdatEnable

  *Offset 87 This field enables the generation of the BIOS DATA ACPI Tables: **0=FALSE**, 1=TRUE.*
- UINT8 MrcBdatTestType

  *Offset 88 When BdatEnable is set to TRUE, this option selects the type of data which will be populated in the BIOS Data ACPI Tables: **0=RMT**, 1=RMT Per Bit, 2=Margin 2D.*
- UINT8 SafeLoadingBiosEnableState

  Offset 89 Safe Loading Bios Enable State.
- UINT8 PprRecoveryStatusEnable

  Offset 90 Ppr Recovery Status Enable.
- UINT8 TsegMemoryTestStatus

  Offset 91 Tseg Memory Test Status.
- UINT8 MrcPprStatus

  Offset 92 Mrc Ppr Status.
- UINT8 RetryCount

  Offset 93 TSEG RetryCount.
- UINT8 ChannelToCkdQckMapping [MEM_CFG_MAX_DDR5_SOCKETS]

  /< Offset 94 CKD QCK to Controller/Channel/DIMM Mapping.
- UINT8 PhyClockToCkdDimm [MEM_CFG_MAX_DDR5_SOCKETS]

  /< Offset 102 DDRIO to CKD Clock.
- UINT8 RsvdBytes106 [2]

  Offset 110 Reserved bytes for struct natural alignment.
- MRC_PPR_ENTRY_INFO PprEntryInfo [PPR_REQUEST_MAX]

  Offset 112 PPR Repair Info.
- MRC_PPR_ENTRY_ADDRESS PprEntryAddress [PPR_REQUEST_MAX]

  Offset 128 PPR Repair Memory Address.
- MRC_PPR_TEST_TYPE PprTestType

  Offset 144 PPR Test Type, enable or disable memory tests in the PPR flow.
- MRC_PPR_REPAIR_TYPE PprRepairType

  Offset 152 PPR Repair Type: 0 = Do not repair, 1 = Soft PPR, 2 = Hard PPR.
- UINT8 PprRunOnce

  Offset 156 PPR Run Once, used to un-set PPR Policies after PPR execution is completed.
- UINT8 PprErrorInjection

  Offset 157 PPR Error Injection.
- UINT8 PprForceRepair

  Offset 158 PPR Force Repair Some Rows.
- UINT8 PprRetryLimit

  Offset 159 PPR Retry Limit.
- UINT8 CAVrefCtlOffset

  Offset 160.
- UINT8 VrefCtlOffset

  Offset 161.
- UINT8 ClkPiCodeOffset

  Offset 162.
- UINT8 RcvEnOffset

  Offset 163.
- UINT8 RxDqsOffset

  Offset 164.
- UINT8 TxDqOffset

  Offset 165.
- UINT8 TxDqsOffset

  Offset 166.

- UINT8 VrefOffset
  Offset 167.
- UINT8 CntrlrMask
  Offset 168.
- UINT8 ChMask
  Offset 169.
- UINT8 CkdAddressTable [MEM_CFG_MAX_SOCKETS]
  /< Offset 170 Memory DIMMs' CKD address for read/write CKD data.
- UINT8 RetrainToWorkingChannel
  Offset 327 Disable Failing Channel during Cold Boot and Retrain To Working Channel.
- UINT8 Reserved [5]
  Offset 186 Reserved bytes for struct alignment.

12.115.1 Detailed Description

Memory Configuration The contents of this structure are not CRC'd by the MRC for option change detection.

Revision 1: - Initial version. **Revision 5**: - Added CkdAddressTable

Definition at line 146 of file MemoryConfig.h.

12.115.2 Member Data Documentation

12.115.2.1 ChannelToCkdQckMapping

`UINT8 MEMORY_CONFIG_NO_CRC::ChannelToCkdQckMapping[MEM_CFG_MAX_DDR5_SOCKETS]`

/< Offset 94 CKD QCK to Controller/Channel/DIMM Mapping.

This mapping is aligned with CkdAddressTable[16], but skipping channel 2/3 (LPDDR) instances. Encoding: 0 - Q↩ CK[0:1]_A 1 - QCK[0:1]_B Keep as 0 for unused since we won't be using the instance if it's non-CKD/unpopulated, so the encoding doesn't matter. Array Index Mapping 0 - Controller 0 Channel 0 Dimm 0 1 - Controller 0 Channel 0 Dimm 1 2 - Controller 0 Channel 1 Dimm 0 3 - Controller 0 Channel 1 Dimm 1 4 - Controller 1 Channel 0 Dimm 0 5 - Controller 1 Channel 0 Dimm 1 6 - Controller 1 Channel 1 Dimm 0 7 - Controller 1 Channel 1 Dimm 1

Definition at line 212 of file MemoryConfig.h.

12.115.2.2 CkdAddressTable

`UINT8 MEMORY_CONFIG_NO_CRC::CkdAddressTable[MEM_CFG_MAX_SOCKETS]`

/< Offset 170 Memory DIMMs' CKD address for read/write CKD data.

0 - Controller 0 Channel 0 Dimm 0 - DDR5 -— LPDDR5 1 - Controller 0 Channel 0 Dimm 1 - DDR5 2DPC 2 - Controller 0 Channel 1 Dimm 0 - DDR5 -— LPDDR5 3 - Controller 0 Channel 1 Dimm 1 - DDR5 2DPC 4 - Controller 0 Channel 2 Dimm 0 ---------— LPDDR5 5 - Controller 0 Channel 2 Dimm 1 ----------— 6 - Controller 0 Channel 3 Dimm 0 ---------— LPDDR5 7 - Controller 0 Channel 3 Dimm 1 ----------— 8 - Controller 1 Channel 0 Dimm 0 - DDR5 -— LPDDR5 9 - Controller 1 Channel 0 Dimm 1 - DDR5 2DPC 10 - Controller 1 Channel 1 Dimm 0 - DDR5 -— LPDDR5 11 - Controller 1 Channel 1 Dimm 1 - DDR5 2DPC 12 - Controller 1 Channel 2 Dimm 0 ---------— LPDDR5 13 - Controller 1 Channel 2 Dimm 1 ----------— 14 - Controller 1 Channel 3 Dimm 0 ---------— LPDDR5 15 - Controller 1 Channel 3 Dimm 1 ----------—

Definition at line 282 of file MemoryConfig.h.

12.115.2.3 PhyClockToCkdDimm

```
UINT8 MEMORY_CONFIG_NO_CRC::PhyClockToCkdDimm[MEM_CFG_MAX_DDR5_SOCKETS]
```

/< Offset 102 DDRIO to CKD Clock.

This mapping is aligned with CkdAddressTable[16], but skipping channel 2/3 (LPDDR) instances. This is a 0-based index which identifies the Phy Clock this Controller/Channel/DIMM is using. For Channels mapping to the same CKD DIMM, the encoding will be the same. The following requirements exist for selecting a CKD Clock:

1. There must be two instances of the clock requested and no more. A CKD DIMM must service two channels
2. The clock selected must be in the channel that is servicing the CKD DIMM
3. This policy must have two Channels of the same DIMM mapping to the same Phy Clock.
4. In the case of CKD and Non-CKD mixed 2DPC systems: a. If the CKD DIMM is DIMM 0, user must map Rank 0 or 1 from either Channel attached to the DIMM. If the CKD DIMM is DIMM 1, user must map Rank 2 or 3 from either Channel attached to the CKD DIMM. b. If the Non-CKD DIMM is in DIMM 0, Rank 0 must be routed to the DIMM Rank 0 and Rank 1 must be routed to the DIMM Rank 1. If the Non-CKD DIMM is in DIMM 1, Rank 2 must be routed to DIMM Rank 2, and Rank 3 must be routed to DIMM Rank 3. All ranks must be from the Channel the Non-CKD DIMM is attached. Encoding: a - xCLKy where a is the array index, x is Phy channel, and y is Phy Rank. 0 - 0CLK0, 4 - 1CLK0, 8 - 2CLK0, 12 - 3CLK0 1 - 0CLK1, 5 - 1CLK1, 9 - 2CLK1, 13 - 3CLK1 2 - 0CLK2, 6 - 1CLK2, 10 - 2CLK2, 14 - 3CLK2 3 - 0CLK3, 7 - 1CLK3, 11 - 2CLK3, 15 - 3CLK3 Keep as 0 for unused since we won't be using the instance if it's non-CKD/unpopulated, so the encoding doesn't matter. Array Index Mapping: 0 - Controller 0 Channel 0 Dimm 0 1 - Controller 0 Channel 0 Dimm 1 2 - Controller 0 Channel 1 Dimm 0 3 - Controller 0 Channel 1 Dimm 1 4 - Controller 1 Channel 0 Dimm 0 5 - Controller 1 Channel 0 Dimm 1 6 - Controller 1 Channel 1 Dimm 0 7 - Controller 1 Channel 1 Dimm 1

Definition at line 242 of file MemoryConfig.h.

12.115.2.4 SerialDebugLevel

```
UINT8 MEMORY_CONFIG_NO_CRC::SerialDebugLevel
```

Sets the serial debug message level
0x00 = Disabled
0x01 = Errors only
0x02 = Errors and Warnings
0x03 = Errors, Warnings, and Info
0x04 = Errors, Warnings, Info, and Events
0x05 = Displays Memory Init Execution Time Summary only
.

Offset 68

Definition at line 164 of file MemoryConfig.h.

12.115.2.5 SpdAddressTable

```
UINT8 MEMORY_CONFIG_NO_CRC::SpdAddressTable[MEM_CFG_MAX_SOCKETS]
```

Offset 71 Memory DIMMs' SPD address for reading SPD data.

0 - Controller 0 Channel 0 Dimm 0 - DDR5 -— LPDDR5 1 - Controller 0 Channel 0 Dimm 1 - DDR5 2DPC 2 - Controller 0 Channel 1 Dimm 0 - DDR5 -— LPDDR5 3 - Controller 0 Channel 1 Dimm 1 - DDR5 2DPC 4 - Controller 0 Channel 2 Dimm 0 ---------— LPDDR5 5 - Controller 0 Channel 2 Dimm 1 ----------— 6 - Controller 0 Channel 3 Dimm 0 ---------— LPDDR5 7 - Controller 0 Channel 3 Dimm 1 ----------— 8 - Controller 1 Channel 0 Dimm 0 - DDR5 -— LPDDR5 9 - Controller 1 Channel 0 Dimm 1 - DDR5 2DPC 10 - Controller 1 Channel 1 Dimm 0 - DDR5 -— LPDDR5 11 - Controller 1 Channel 1 Dimm 1 - DDR5 2DPC 12 - Controller 1 Channel 2 Dimm 0 ---------— LPDDR5 13 - Controller 1 Channel 2 Dimm 1 ----------— 14 - Controller 1 Channel 3 Dimm 0 ---------— LPDDR5 15 - Controller 1 Channel 3 Dimm 1 ----------—

Definition at line 186 of file MemoryConfig.h.

The documentation for this struct was generated from the following file:

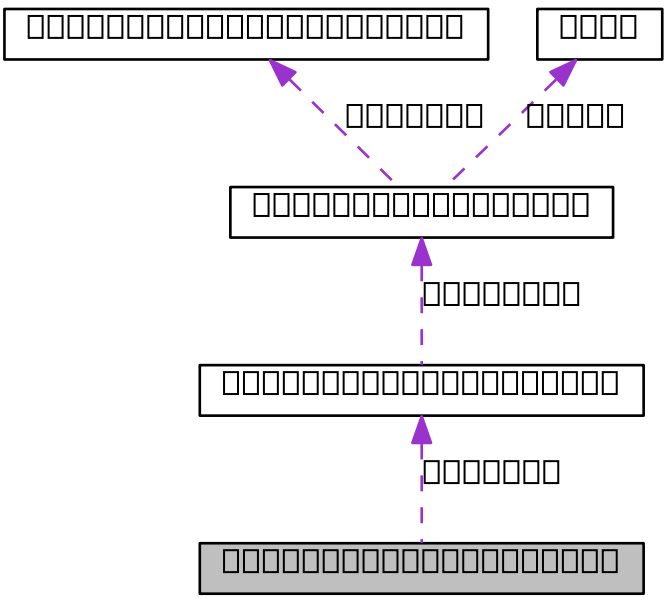
- MemoryConfig.h

12.116 MEMORY_CONFIGURATION Struct Reference

Memory Configuration The contents of this structure are CRC'd by the MRC for option change detection.

```
#include <MemoryConfig.h>
```

Collaboration diagram for MEMORY_CONFIGURATION:



Public Attributes

- CONFIG_BLOCK_HEADER Header
  Offset 0-27 Config Block Header.

12.116.1 Detailed Description

Memory Configuration The contents of this structure are CRC'd by the MRC for option change detection.

Definition at line 137 of file MemoryConfig.h.

The documentation for this struct was generated from the following file:

- MemoryConfig.h

12.117 MEMORY_DQDQS_MAPPING Struct Reference

DqDqs Mapping.

```
#include <MemoryConfig.h>
```

Public Attributes

- UINT8 DqsMapCpu2Dram [MEM_CFG_MAX_CONTROLLERS][MEM_CFG_MAX_CHANNELS][MEM_C↩FG_NUM_BYTES_MAPPED]
  DqsMapCpu2Dram.
- UINT8 DqMapCpu2Dram [MEM_CFG_MAX_CONTROLLERS][MEM_CFG_MAX_CHANNELS][MEM_CF↩G_NUM_BYTES_MAPPED][8]
  DqMapCpu2Dram.

12.117.1 Detailed Description

DqDqs Mapping.

Definition at line 93 of file MemoryConfig.h.

The documentation for this struct was generated from the following file:

- MemoryConfig.h

12.118 MEMORY_RCOMP Struct Reference

Rcomp Policies.

```
#include <MemoryConfig.h>
```

Public Attributes

- UINT16 RcompResistor

  Offset 0: Reference RCOMP resistors on motherboard ~ 100 ohms.
- UINT16 RcompTarget [MAX_RCOMP_TARGETS]

  Offset 1: RCOMP target values for DqOdt, DqDrv, CmdDrv, CtlDrv, ClkDrv.

12.118.1 Detailed Description

Rcomp Policies.

Definition at line 102 of file MemoryConfig.h.

The documentation for this struct was generated from the following file:

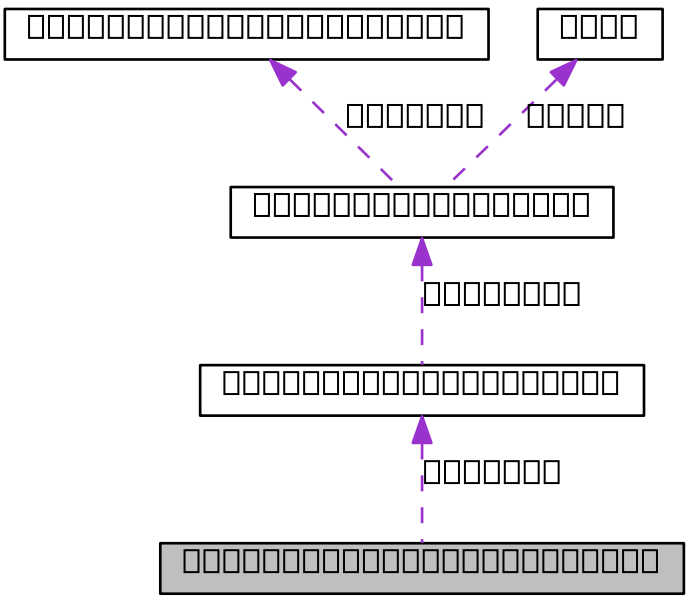
- MemoryConfig.h

12.119 MEMORY_SUB_SYSTEM_CONFIG Struct Reference

This Configuration block configures Memory SubSystem **Revision 1**:

```
#include <MemorySubSystemConfig.h>
```

Collaboration diagram for MEMORY_SUB_SYSTEM_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Offset 0-27 Config Block Header.
- UINT32 AcpiReservedMemorySize

  Offset 28 The Size of a Reserved memory buffer allocated in previous boot for S3 resume used. Originally it is retrieved from AcpiVariableCompatibility variable.
- UINT64 AcpiReservedMemoryBase

  Offset 32 The Base address of a Reserved memory buffer allocated in previous boot for S3 resume used. Originally it is retrieved from AcpiVariableCompatibility variable.
- UINT8 MemMapOverlapCheckSupport

  Offset 40 MemoryOverlap check Enable/Disable.
- UINT8 RsvdBytes38 [7]

  Reserved for future use.

12.119.1 Detailed Description

This Configuration block configures Memory SubSystem **Revision 1**:

- Initial version.

Definition at line 33 of file MemorySubSystemConfig.h.

The documentation for this struct was generated from the following file:

- MemorySubSystemConfig.h

12.120 MRC_ERROR_KEY_VALUE_ENTRY Struct Reference

MRC Error Key Value Entry structure maps an MRC error code to an error message string.

```
#include <IGpuConfig.h>
```

Public Attributes

- UINT32 Key

  MRC POST code (16-bit value, 0xFFFF = fallback key)
- CONST CHAR8 * Value

  MRC error message string.

12.120.1 Detailed Description

MRC Error Key Value Entry structure maps an MRC error code to an error message string.

Maximum message length (excluding null terminator):

- Exact match messages (specific POST code keys): 80 characters
- Fallback message (key 0xFFFF): 73 characters (+ "0xXXXX" appended automatically)

Definition at line 169 of file IGpuConfig.h.

The documentation for this struct was generated from the following file:

- IGpuConfig.h

12.121 MUX_GPIO_CONFIG Struct Reference

I3C GPIO settings.

```
#include <SerialIoDevices.h>
```

Public Attributes

- UINT32 PadTermination
  GPIO signals pin muxing settings.

12.121.1 Detailed Description

I3C GPIO settings.

Definition at line 247 of file SerialIoDevices.h.

The documentation for this struct was generated from the following file:

- SerialIoDevices.h

12.122 MUX_GPIO_PARAM Struct Reference

SD GPIO settings.

```
#include <ScsConfig.h>
```

Public Attributes

- UINT32 PinMux

  GPIO signals pin muxing settings.
- UINT32 PadTermination

  GPIO Pads Internal Termination.

12.122.1 Detailed Description

SD GPIO settings.

Definition at line 64 of file ScsConfig.h.

12.122.2 Member Data Documentation

12.122.2.1 PadTermination

```
UINT32 MUX_GPIO_PARAM::PadTermination
```

GPIO Pads Internal Termination.

For more information please see Platform Design Guide. Check GPIO_ELECTRICAL_CONFIG for reference

Definition at line 76 of file ScsConfig.h.

12.122.2.2 PinMux

```
UINT32 MUX_GPIO_PARAM::PinMux
```

GPIO signals pin muxing settings.

If signal can be enable only on a single pin then this parameter should be set to 0. Refer to GPIO_∗_MUXING_S↩DCARD_∗x_∗ in GpioPins∗.h for supported settings on a given platform

Definition at line 70 of file ScsConfig.h.

The documentation for this struct was generated from the following file:

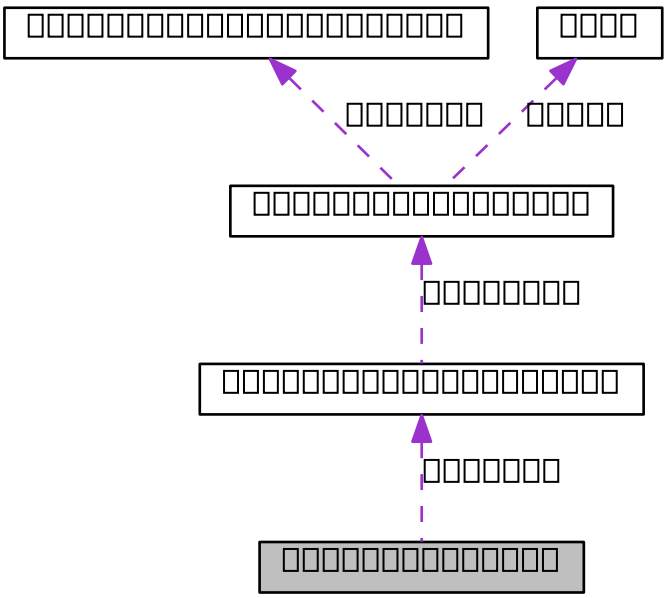
- ScsConfig.h

12.123 NPU_PEI_CONFIG Struct Reference

NPU PEI configuration
Revision 1:

```
#include <NpuConfig.h>
```

Collaboration diagram for NPU_PEI_CONFIG:



Public Attributes

- UINT8 NpuEnable

  Offset 0-27 : Config Block Header.
- UINT8 Reserved [3]

  Offset 29 : Reserved for 4 bytes of alignment.

12.123.1 Detailed Description

NPU PEI configuration
Revision 1:

- Initial version.

Definition at line 34 of file NpuConfig.h.

12.123.2 Member Data Documentation

12.123.2.1 NpuEnable

```
UINT8 NPU_PEI_CONFIG::NpuEnable
```

Offset 0-27 : Config Block Header.

Offset 28 It controls enabling/disabling NPU device if supported and not fused off. If FALSE, all other polices can be ignored. **1=TRUE**; 0=FALSE.

Definition at line 43 of file NpuConfig.h.

The documentation for this struct was generated from the following file:

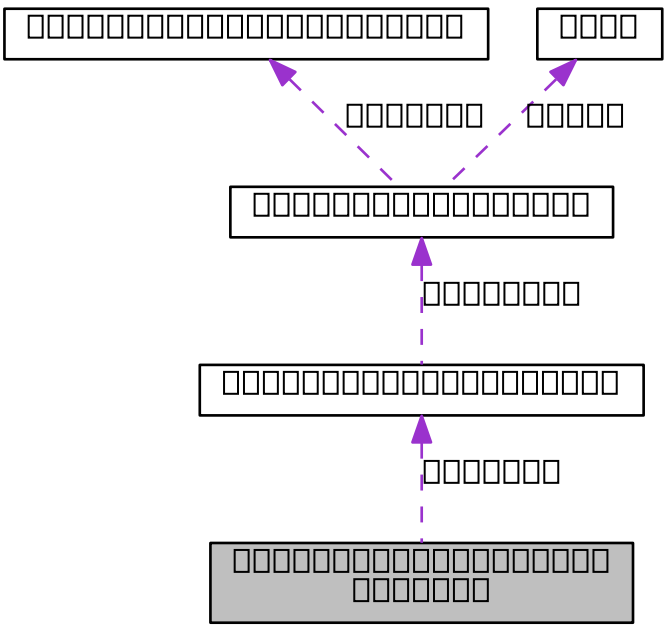
- NpuConfig.h

12.124 OVERCLOCKING_PREMEM_CONFIG Struct Reference

Overclocking Configuration Structure.

```
#include <OverclockingConfig.h>
```

Collaboration diagram for OVERCLOCKING_PREMEM_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT8 OcSupport

  Overclocking support.
- UINT8 PerCoreRatioOverride

  Enable or disable Per Core PState OC supported by writing OCMB 0x1D to program new favored core ratio to each Core. ***0: Disable****, 1: enable.*
- UINT8 RingDownBin

  Ring Downbin enable/disable.
- UINT8 ThermalTurbo

  This service controls the Intel Thermal Velocity Boost (TVB) feature enabling.
- UINT8 NegativeItd

  This service controls voltage reduction based True Temperature Dependency for processors that implement the IntelThermal Velocity Boost (TVB) feature.
- UINT8 IaIccUnlimitedMode

  Support IA Unlimited ICCMAX up to maximum value 512A., ***0: Disable****. 1: Enable.*
- UINT8 GtIccUnlimitedMode

  Support GT Unlimited ICCMAX up to maximum value 512A., ***0: Disable****. 1: Enable.*
- UINT8 VccsaBootVoltageSel

  This service controls VCCSA Boot Voltage by programming the EPOC2 bits (strap).
- UINT8 CpuBandgapRefMode

  This service controls CPU Bandgap bypass or not by programming the CPU EPOC2 bits (strap).
- UINT8 VcciaBootVoltageSel

  This service controls VCCIA boot voltage by programming SOC-North soft straps.
- UINT8 BclkAdaptiveVoltage

  Bclk Adaptive Voltage enable/disable. ***0: Disabled****, 1: Enabled. When enabled, the CPU V/F curves are aware of BCLK frequency when calculated.*
- UINT8 RealtimeMemoryTiming

  Enable/Disable the message sent to the CPU to allow realtime memory timing changes after MRC_DONE. ***0=Disable****, 1=Enable.*
- UINT8 Avx2RatioOffset

  AVX2 Ratio Offset. ***0: No offset****. Range is 0-31. Used to lower the AVX ratio to maximize possible ratio for SSE workload.*
- UINT8 GranularRatioOverride

  Enable or disable OC Granular Ratio Override. ***0: Disable****, 1: enable.*
- UINT8 Avx2VoltageScaleFactor

  Avx2 Voltage Guardband Scale Factor This controls the AVX2 Voltage Guardband Scale factor applied to AVX2 workloads.
- UINT8 TjMaxOffset

  TjMax Offset.
- UINT8 CoreVfConfigScope

  Core VF Configuration Scope.
- UINT16 IaIccMax

  Voltage Regulator Current Limit (Icc Max) for IA, GT and SA.
- UINT8 CorePllVoltageOffset

  Core PLL voltage offset. ***0: No offset****. Range 0-15 in 17.5mv units.*
- UINT8 RingPllVoltageOffset

  Ring PLL voltage offset. ***0: No offset****. Range 0-15 in 17.5mv units.*
- UINT8 IaAtomPllVoltageOffset

  IA Atom PLL voltage offset. ***0: No offset****. Range 0-15 in 17.5mv units.*

- UINT8 CpuDlvrMode

  *Select Core(s) and RING DLVR Mode, **0: Regulation Mode**. 1: Power Gate Mode.*
- UINT8 TscDisableHwFixup

  *Core HW Fixup disable during TSC copy from PMA to APIC, **0: Enable**. 1: Disable.*
- UINT8 SaIccUnlimitedMode

  *Support SA Unlimited ICCMAX up to maximum value 512A., **0: Disable**. 1: Enable.*
- UINT8 UnderVoltProtection

  When UnderVolt Protection is enabled, user will be not be able to program under voltage in OS runtime.
- UINT8 CoreOpPointReportingEn

  *Enables Core Operating point reporting. **0: Disable**, 1: Enable.*
- UINT8 FllOverclockMode

  FLL_OVERCLOCK_MODE: 0x0 = no overclocking, 0x1 = ratio overclocking with nominal (0.5-1x) reference clock frequency, 0x2 = BCLK overclocking with elevated (1-3x) reference clock frequency, 0x3 = BCLK overclocking with extreme elevated (3-5x) reference clock frequency and ratio limited to 63.
- UINT8 CpuD2dRatio

  Set CPU D2D Ratio from Range 15 to 40.
- INT16 PerCoreVoltageOffset [MAX_OC_BIG_CORES]

  Array used to specifies the selected Core Offset Voltage.
- UINT8 PerCoreVoltageMode [MAX_OC_BIG_CORES]

  Array used to specifies the selected Core Voltage Mode.
- UINT16 PerCoreVoltageOverride [MAX_OC_BIG_CORES]

  Array used to specifies the selected Core Voltage Override.
- UINT16 PerCoreAdaptiveVoltage [MAX_OC_BIG_CORES]

  Array used to specifies the selected Core Adaptive Voltage.
- UINT8 PerAtomClusterVoltageMode [MAX_OC_ATOM_CLUSTERS]

  Array used to specifies the selected Atom Core Voltage Mode.
- UINT16 PerAtomClusterVoltageOverride [MAX_OC_ATOM_CLUSTERS]

  Array used to specifies the selected Atom Core Voltage Override.
- UINT16 PerAtomClusterAdaptiveVoltage [MAX_OC_ATOM_CLUSTERS]

  Array used to specifies the selected Atom Core Adaptive Voltage.
- UINT8 AtomClusterRatio [MAX_OC_ATOM_CLUSTERS]

  The Atom cluster max ratio overrides.
- UINT8 ComputeDieSscEnable

  Enable or Disable Compute Die SSC configuration.
- UINT8 SocDieSscEnable

  Enable or Disable Soc Die SSC configuration.
- UINT32 CpuBclkOcFrequency

  *CPU BCLK OC Frequency in 10KHz units increasing. Value 9800 (10KHz) = 98MHz **0 - Auto**. Range is 8000-50000 (10KHz)*
- UINT32 SocBclkOcFrequency

  *SOC BCLK OC Frequency in 10KHz units increasing. Value 9800 (10KHz) = 98MHz **0 - Auto**. Range is 4000-100000 (10KHz)*
- UINT8 PvdRatioThreshold

  The Pvd Ratio Threshold for HUb die is the threshold value for input ratio (P0 to Pn) to select the multiplier so that the output is within the DCO frequency range.
- UINT8 PvdMode

  PVD Mode selects Static PVD ratio for Hub-die (when PVD THRESHOLD value is 0).
- UINT8 GranularityBins [MAX_CDIE][MAX_OC_CCP_MODULES]

  Per Module Granularity Bins Override.
- UINT8 PcoreTvbTempThreshold0

  Temperature Threshold 0 for all Pcores (in degrees C).
- UINT8 PcoreTvbTempThreshold1

Temperature Threshold 1 for all Pcores (in degrees C).

- UINT8 EcoreTvbTempThreshold0

  Temperature Threshold 0 for all Ecores (in degrees C).

- UINT8 EcoreTvbTempThreshold1

  Temperature Threshold 1 for all Ecores (in degrees C).

- UINT64 DisablePerCoreMask

  Core disable mask is a bitwise indication of which core should be disabled, while ActiveCoreCount is to set the number of active cores.

- UINT8 TvbConfigLimitSelect

  Limits Select, 0 - Per CCP (Module), Command Param2 = CCP ID, CCP ID can range from 0-15.

- UINT8 OcTvb

  OC TVB Enable/Disable.

- UINT8 PerPcoreRatioDownBinAboveT0 [MAX_OC_BIG_CORES]

  Down Bins (delta) for Temperature Threshold 0.

- UINT8 PerPcoreRatioDownBinAboveT1 [MAX_OC_BIG_CORES]

  Down Bins (delta) for Temperature Threshold 1.

- UINT8 PerEcoreCcpRatioDownBinAboveT0 [MAX_OC_ATOM_CLUSTERS]

  Down Bins (delta) for Temperature Threshold 0.

- UINT8 PerEcoreCcpRatioDownBinAboveT1 [MAX_OC_ATOM_CLUSTERS]

  Down Bins (delta) for Temperature Threshold 1.

- UINT8 PerPcoreGrRatioDownBinAboveT0 [MAX_OC_BIG_CORE_TRL_GROUPS]

  Down Bins (delta) for Temperature Threshold 0.

- UINT8 PerPcoreGrRatioDownBinAboveT1 [MAX_OC_BIG_CORE_TRL_GROUPS]

  Down Bins (delta) for Temperature Threshold 1.

- UINT8 PerCoreMaxRatio [MAX_OC_BIG_CORES]

  Array used to specifies the selected Core Max Ratio.

- UINT8 PerAtomClusterMaxRatio [MAX_OC_ATOM_CLUSTERS]

  Array used to specifies the selected Atom Cluster Max Ratio.

- UINT8 VoltageMode [MAX_OC_DOMAINS]

  Voltage mode, specifies which voltage mode the processor will be operating.

- UINT8 MaxOcRatio [MAX_OC_DOMAINS]

  Maximum turbo ratio override allows to increase frequency beyond the fused max turbo ratio limit (P0).

- UINT16 VoltageOverride [MAX_OC_DOMAINS]

  The voltage override which is applied to the entire range of cpu core frequencies.

- UINT16 AdaptiveVoltage [MAX_OC_DOMAINS]

  Adaptive voltage target used to define the interpolation voltage point when operating in adaptive mode range.

- INT16 VoltageOffset [MAX_OC_DOMAINS]

  The voltage offset applied on top of all other voltage modes.

- UINT8 VfPointOffsetMode [MAX_OC_DOMAINS]

  Selects Voltage & Frequency Point Offset between Legacy and Selection modes.

- UINT8 VfPointCount [MAX_OC_DOMAINS]

  Number of supported Voltage & Frequency Point.

- UINT8 VfPointRatio [MAX_OC_DOMAINS][MAX_OC_VF_POINTS]

  Array for the each selected VF Point to display the Core Ration. It's only for read command and not for write.

- INT16 VfPointOffset [MAX_OC_DOMAINS][MAX_OC_VF_POINTS]

  Array used to specifies the Voltage Offset applied to the each selected VF Point.

- UINT16 MaxVoltageLimit [MAX_CDIE][MAX_OC_DOMAINS]

  *Array for the each Domain to set OC Max Voltage limits **Default: 0** Range: 0 to 2000 NOTE: Only Core/Ring Domain support multi-die, for the other domains DIE = 0 as they are package scope.*

- UINT8 ProcessVmaxLimit

  Disabling Process Vmax Limit will allow user to set any voltage.

- UINT8 CorePllCurrentRefTuningOffset

  Core PLL Current Reference Tuning Offset. ***0: No offset****. Range 0-15.*
- UINT8 RingPllCurrentRefTuningOffset

  Ring PLL Current Reference Tuning Offset. ***0: No offset****. Range 0-15.*
- UINT8 IaAtomPllCurrentRefTuningOffset

  IaAtom PLL Current Reference Tuning Offset. ***0: No offset****. Range 0-15.*
- UINT8 CoreMinRatio

  Request Core Min Ratio.
- UINT8 NegativeTemperatureReporting

  Negative Temperature Reporting Enable. ***0: Disable****, 1: enable.*
- UINT8 PcorePowerDensityThrottle

  Power Density Throttle control allows user to disable P-core and E-core throttling for power density protection.

12.124.1 Detailed Description

Overclocking Configuration Structure.

Revision 1:

- Initial version.

Definition at line 38 of file OverclockingConfig.h.

12.124.2 Member Data Documentation

12.124.2.1 AdaptiveVoltage

```
UINT16 OVERCLOCKING_PREMEM_CONFIG::AdaptiveVoltage[MAX_OC_DOMAINS]
```

Adaptive voltage target used to define the interpolation voltage point when operating in adaptive mode range.

Used when VoltageMode = Adaptive. **0. no override**. Range 0-2000mV.

Definition at line 313 of file OverclockingConfig.h.

12.124.2.2 AtomClusterRatio

```
UINT8 OVERCLOCKING_PREMEM_CONFIG::AtomClusterRatio[MAX_OC_ATOM_CLUSTERS]
```

The Atom cluster max ratio overrides.

Default: 0 Range: 0 to 120.

Definition at line 164 of file OverclockingConfig.h.

12.124.2.3 Avx2VoltageScaleFactor

`UINT8 OVERCLOCKING_PREMEM_CONFIG::Avx2VoltageScaleFactor`

Avx2 Voltage Guardband Scale Factor This controls the AVX2 Voltage Guardband Scale factor applied to AVX2 workloads.

Valid range is 0-200 in 1/100 units, where a value of 125 would apply a 1.25 scale factor. A value of 0 means no scale factor applied (no change to voltage on AVX commands) A value of 100 applies the default voltage guardband values (1.0 factor). A value $>$ 100 will increase the voltage guardband on AVX2 workloads. A value $<$ 100 will decrease the voltage guardband on AVX2 workloads. **0. No scale factor applied**

Definition at line 102 of file OverclockingConfig.h.

12.124.2.4 ComputeDieSscEnable

`UINT8 OVERCLOCKING_PREMEM_CONFIG::ComputeDieSscEnable`

Enable or Disable Compute Die SSC configuration.

- **0: Disable**
- 1: Enable

Definition at line 172 of file OverclockingConfig.h.

12.124.2.5 CoreMinRatio

`UINT8 OVERCLOCKING_PREMEM_CONFIG::CoreMinRatio`

Request Core Min Ratio.

Limit Core minimum ratio for extreme overclocking. Default 0 indicates no request.

Definition at line 354 of file OverclockingConfig.h.

12.124.2.6 CoreVfConfigScope

`UINT8 OVERCLOCKING_PREMEM_CONFIG::CoreVfConfigScope`

Core VF Configuration Scope.

Allows both all-core VF curve or per-core VF curve configuration. **0: All-core Scope, setting the VF curve for all cores.** 1: Per-core Scope, setting the VF curve per-core.

Definition at line 114 of file OverclockingConfig.h.

12.124.2.7 CpuBandgapRefMode

`UINT8 OVERCLOCKING_PREMEM_CONFIG::CpuBandgapRefMode`

This service controls CPU Bandgap bypass or not by programming the CPU EPOC2 bits (strap).

Default: 0: Normal 1: Bandgap Bypassed

Definition at line 81 of file OverclockingConfig.h.

12.124.2.8 CpuD2dRatio

`UINT8 OVERCLOCKING_PREMEM_CONFIG::CpuD2dRatio`

Set CPU D2D Ratio from Range 15 to 40.

Default 0 indicates no setting.

Definition at line 146 of file OverclockingConfig.h.

12.124.2.9 DisablePerCoreMask

`UINT64 OVERCLOCKING_PREMEM_CONFIG::DisablePerCoreMask`

Core disable mask is a bitwise indication of which core should be disabled, while ActiveCoreCount is to set the number of active cores.

Each bit represents physical core id for both P-core and E-core 1 - Disable 0 - Ignored Default is 0, means no Core disable mask setting.

Definition at line 235 of file OverclockingConfig.h.

12.124.2.10 EcoreTvbTempThreshold0

`UINT8 OVERCLOCKING_PREMEM_CONFIG::EcoreTvbTempThreshold0`

Temperature Threshold 0 for all Ecores (in degrees C).

Running ABOVE this temperature will clip delta Down Bins for T0 from the resolved OC Ratio, when Thermal Turbo is enabled from OCMB 0x18 Bit 2 = 0 (Enabled). **70** . Range 0-100.

Definition at line 220 of file OverclockingConfig.h.

12.124.2.11 EcoreTvbTempThreshold1

```
UINT8 OVERCLOCKING_PREMEM_CONFIG::EcoreTvbTempThreshold1
```

Temperature Threshold 1 for all Ecores (in degrees C).

Running ABOVE this temperature will clip delta Down Bins for T1 from the resolved OC Ratio, when Thermal Turbo is enabled from OCMB 0x18 Bit 2 = 0 (Enabled). **100** . Range 0-100.

Definition at line 226 of file OverclockingConfig.h.

12.124.2.12 GranularityBins

```
UINT8 OVERCLOCKING_PREMEM_CONFIG::GranularityBins[MAX_CDIE][MAX_OC_CCP_MODULES]
```

Per Module Granularity Bins Override.

Array is used to specify Granularity Bins for specific Module on a specific CDie. It's a negative offset to current resolved P0 ratio. Programming Granularity Bins will always reduce the core frequency. value is how many bins of 'Granularity Units' should be subtracted from the otherwise-configured P0 frequency. **value = 0 (default)** Range 0-255.

Definition at line 202 of file OverclockingConfig.h.

12.124.2.13 IaIccMax

```
UINT16 OVERCLOCKING_PREMEM_CONFIG::IaIccMax
```

Voltage Regulator Current Limit (Icc Max) for IA, GT and SA.

This value represents the Maximum instantaneous current allowed at any given time. The value is represented in 1/4 A increments. A value of 400 = 100A. **0 (HW default)**. Range is 4-2047.

Definition at line 120 of file OverclockingConfig.h.

12.124.2.14 MaxOcRatio

```
UINT8 OVERCLOCKING_PREMEM_CONFIG::MaxOcRatio[MAX_OC_DOMAINS]
```

Maximum turbo ratio override allows to increase frequency beyond the fused max turbo ratio limit (P0).

0. no override/HW defaults.. Range non-turbo max.

Definition at line 301 of file OverclockingConfig.h.

12.124.2.15 NegativeItd

```
UINT8 OVERCLOCKING_PREMEM_CONFIG::NegativeItd
```

This service controls voltage reduction based True Temperature Dependency for processors that implement the IntelThermal Velocity Boost (TVB) feature.

Core voltages will be lowered based on Core temperature 0: Disables Negative ITD. **1: Enables Negative ITD.**

Definition at line 69 of file OverclockingConfig.h.

12.124.2.16 OcSupport

```
UINT8 OVERCLOCKING_PREMEM_CONFIG::OcSupport
```

Overclocking support.

This controls whether OC mailbox transactions are sent. If disabled, all policies in this config block besides Oc↩ Support and OcLock will be ignored. **0: Disable**; 1: Enable.

Note

If PcdOverclockEnable is disabled, this should also be disabled.

Definition at line 47 of file OverclockingConfig.h.

12.124.2.17 OcTvb

```
UINT8 OVERCLOCKING_PREMEM_CONFIG::OcTvb
```

OC TVB Enable/Disable.

This control will come into picture only when Thermal Turbo is enabled from OCMB 0x18/0x19 (bit 2). When disabled, BIOS will not send OC TVB parameter commands 0x24/0x25. When OcTvb is enabled, BIOS will allow user to modify and program new parameters for temperature thresholds T0, T1 and delta DownBins for T0 and T1. **0: Disable**, 1: Enable.

Definition at line 251 of file OverclockingConfig.h.

12.124.2.18 PcorePowerDensityThrottle

```
UINT8 OVERCLOCKING_PREMEM_CONFIG::PcorePowerDensityThrottle
```

Power Density Throttle control allows user to disable P-core and E-core throttling for power density protection.

This is a sticky bit, Once set, will remain set until cold or warm reset is performed. MailBox control is to disable the Power Density Throttling (reverse encoding is used). 0: Disable;**1: Enable**;

Definition at line 361 of file OverclockingConfig.h.

12.124.2.19 PcoreTvbTempThreshold0

```
UINT8 OVERCLOCKING_PREMEM_CONFIG::PcoreTvbTempThreshold0
```

Temperature Threshold 0 for all Pcores (in degrees C).

Running ABOVE this temperature will clip delta Down Bins for T0 from the resolved OC Ratio, when Thermal Turbo is enabled from OCMB 0x18 Bit 2 = 0 (Enabled). **70** . Range 0-100.

Definition at line 208 of file OverclockingConfig.h.

12.124.2.20 PcoreTvbTempThreshold1

```
UINT8 OVERCLOCKING_PREMEM_CONFIG::PcoreTvbTempThreshold1
```

Temperature Threshold 1 for all Pcores (in degrees C).

Running ABOVE this temperature will clip delta Down Bins for T1 from the resolved OC Ratio, when Thermal Turbo is enabled from OCMB 0x18 Bit 2 = 0 (Enabled). **100** . Range 0-100.

Definition at line 214 of file OverclockingConfig.h.

12.124.2.21 PerCoreVoltageOffset

```
INT16 OVERCLOCKING_PREMEM_CONFIG::PerCoreVoltageOffset[MAX_OC_BIG_CORES]
```

Array used to specifies the selected Core Offset Voltage.

This voltage is specified in millivolts.

Definition at line 152 of file OverclockingConfig.h.

12.124.2.22 PerEcoreCcpRatioDownBinAboveT0

```
UINT8 OVERCLOCKING_PREMEM_CONFIG::PerEcoreCcpRatioDownBinAboveT0[MAX_OC_ATOM_CLUSTERS]
```

Down Bins (delta) for Temperature Threshold 0.

When running above T0, the ratio of Per Ecore module will be clipped by MAX_RATIO[n]-this value, when Thermal Turbo is enabled from OCMB 0x18 Bit 2 = 0 (Enabled). **0** . Range 0-10.

Definition at line 269 of file OverclockingConfig.h.

12.124.2.23 PerEcoreCcpRatioDownBinAboveT1

```
UINT8 OVERCLOCKING_PREMEM_CONFIG::PerEcoreCcpRatioDownBinAboveT1[MAX_OC_ATOM_CLUSTERS]
```

Down Bins (delta) for Temperature Threshold 1.

When running above T1, the ratio of Per Ecore module will be clipped by MAX_RATIO[n]-this value, when Thermal Turbo is enabled from OCMB 0x18 Bit 2 = 0 (Enabled). **0** . Range 0-10.

Definition at line 275 of file OverclockingConfig.h.

12.124.2.24 PerPcoreGrRatioDownBinAboveT0

```
UINT8 OVERCLOCKING_PREMEM_CONFIG::PerPcoreGrRatioDownBinAboveT0[MAX_OC_BIG_CORE_TRL_GROUPS]
```

Down Bins (delta) for Temperature Threshold 0.

When running above T0, the ratio of Per Pcore group will be clipped by MAX_RATIO[n]-this value, when Thermal Turbo is enabled from OCMB 0x18 Bit 2 = 0 (Enabled). **1** . Range 0-10.

Definition at line 281 of file OverclockingConfig.h.

12.124.2.25 PerPcoreGrRatioDownBinAboveT1

```
UINT8 OVERCLOCKING_PREMEM_CONFIG::PerPcoreGrRatioDownBinAboveT1[MAX_OC_BIG_CORE_TRL_GROUPS]
```

Down Bins (delta) for Temperature Threshold 1.

When running above T1, the ratio of Per Pcore group will be clipped by MAX_RATIO[n]-this value, when Thermal Turbo is enabled from OCMB 0x18 Bit 2 = 0 (Enabled). **2** . Range 0-10.

Definition at line 287 of file OverclockingConfig.h.

12.124.2.26 PerPcoreRatioDownBinAboveT0

```
UINT8 OVERCLOCKING_PREMEM_CONFIG::PerPcoreRatioDownBinAboveT0[MAX_OC_BIG_CORES]
```

Down Bins (delta) for Temperature Threshold 0.

When running above T0, the ratio of Per Pcore module will be clipped by MAX_RATIO[n]-this value, when Thermal Turbo is enabled from OCMB 0x18 Bit 2 = 0 (Enabled). **1** . Range 0-10.

Definition at line 257 of file OverclockingConfig.h.

12.124.2.27 PerPcoreRatioDownBinAboveT1

```
UINT8 OVERCLOCKING_PREMEM_CONFIG::PerPcoreRatioDownBinAboveT1[MAX_OC_BIG_CORES]
```

Down Bins (delta) for Temperature Threshold 1.

When running above T1, the ratio of Per Pcore module will be clipped by MAX_RATIO[n]-this value, when Thermal Turbo is enabled from OCMB 0x18 Bit 2 = 0 (Enabled). **2** . Range 0-10.

Definition at line 263 of file OverclockingConfig.h.

12.124.2.28 ProcessVmaxLimit

```
UINT8 OVERCLOCKING_PREMEM_CONFIG::ProcessVmaxLimit
```

Disabling Process Vmax Limit will allow user to set any voltage.

If this limit is not disabled then, request above factory set limits will be rejected. This is a sticky bit, BIOS can only set this bit, it cannot be cleared by mailbox command. This setting persists over warm reset, and cleared on cold reset. MailBox control is to disable the ProcessVmaxLimit (reverse encoding is used). 0: Disable;**1: Enable**; Note: Disabling the voltage limit checks may cause permanent damage to processor. Note: This bit cannot be set after BIOS_RST_CPL.

Definition at line 347 of file OverclockingConfig.h.

12.124.2.29 PvdMode

```
UINT8 OVERCLOCKING_PREMEM_CONFIG::PvdMode
```

PVD Mode selects Static PVD ratio for Hub-die (when PVD THRESHOLD value is 0).

0x0 = div-1 (VCO = Output clock), 0x1 = div-2 (VCO = 2x Output clock) 0x2 = div-4 (VCO = 4x Output clock), 0x3 = div-8 (VCO = 8x Output clock)

Definition at line 194 of file OverclockingConfig.h.

12.124.2.30 PvdRatioThreshold

```
UINT8 OVERCLOCKING_PREMEM_CONFIG::PvdRatioThreshold
```

The Pvd Ratio Threshold for HUb die is the threshold value for input ratio (P0 to Pn) to select the multiplier so that the output is within the DCO frequency range.

This threshold is applied to SA and MC/CMI PLL for Hub die. Range 0-63. When the threshold is 0, static PVD ratio is selected based on the PVD Mode for Hub. **0: Default**.

Definition at line 188 of file OverclockingConfig.h.

12.124.2.31 RingDownBin

`UINT8 OVERCLOCKING_PREMEM_CONFIG::RingDownBin`

Ring Downbin enable/disable.

When enabled, the CPU will force the ring ratio to be lower than the core ratio. Disabling will allow the ring and core ratios to run at the same frequency. Uses OC Mailbox command 0x19. 0: Disables Ring Downbin feature. **1: Enables Ring downbin feature.**

Definition at line 56 of file OverclockingConfig.h.

12.124.2.32 SocDieSscEnable

`UINT8 OVERCLOCKING_PREMEM_CONFIG::SocDieSscEnable`

Enable or Disable Soc Die SSC configuration.

- **0: Disable**
- 1: Enable

Definition at line 179 of file OverclockingConfig.h.

12.124.2.33 ThermalTurbo

`UINT8 OVERCLOCKING_PREMEM_CONFIG::ThermalTurbo`

This service controls the Intel Thermal Velocity Boost (TVB) feature enabling.

If enabled, High temperatures may result in lower ratios than P0. It is required to be disabled for supporting overclocking at frequencies higher than the default max turbo frequency. 0: Disables Thermal Turbo. **1: Enables Thermal Turbo**.

Definition at line 63 of file OverclockingConfig.h.

12.124.2.34 TjMaxOffset

`UINT8 OVERCLOCKING_PREMEM_CONFIG::TjMaxOffset`

TjMax Offset.

Specified value here is clipped by pCode (125 - TjMax Offset) to support TjMax in the range of 62 to 115 deg Celsius. **Default: 0 Hardware Defaults** Range 10 to 63. 0 = No offset / Keep HW default.

Definition at line 107 of file OverclockingConfig.h.

12.124.2.35 TvbConfigLimitSelect

```
UINT8 OVERCLOCKING_PREMEM_CONFIG::TvbConfigLimitSelect
```

Limits Select, 0 - Per CCP (Module), Command Param2 = CCP ID, CCP ID can range from 0-15.

1 - Per Pcore group number, Command Param2 = Pcore Group Number (0-7), a Pcore active group has number of active cores as present in MSR 0x1AE. **0** .

Definition at line 243 of file OverclockingConfig.h.

12.124.2.36 UnderVoltProtection

```
UINT8 OVERCLOCKING_PREMEM_CONFIG::UnderVoltProtection
```

When UnderVolt Protection is enabled, user will be not be able to program under voltage in OS runtime.

0: protection is disabled, **1:protection is enabled** .

Definition at line 133 of file OverclockingConfig.h.

12.124.2.37 VcciaBootVoltageSel

```
UINT8 OVERCLOCKING_PREMEM_CONFIG::VcciaBootVoltageSel
```

This service controls VCCIA boot voltage by programming SOC-North soft straps.

Default: 0: Nominal 1: High Voltage (support VCCIA boot voltage higher than 1.65v (max 2.01v))

Definition at line 86 of file OverclockingConfig.h.

12.124.2.38 VccsaBootVoltageSel

```
UINT8 OVERCLOCKING_PREMEM_CONFIG::VccsaBootVoltageSel
```

This service controls VCCSA Boot Voltage by programming the EPOC2 bits (strap).

Default: 0: Nominal 1: High Voltage(up to 1.2/1.3V)

Definition at line 76 of file OverclockingConfig.h.

12.124.2.39 VfPointOffset

```
INT16 OVERCLOCKING_PREMEM_CONFIG::VfPointOffset[MAX_OC_DOMAINS][MAX_OC_VF_POINTS]
```

Array used to specifies the Voltage Offset applied to the each selected VF Point.

This voltage is specified in millivolts. **Default: 0** Range: 0 to 1000

Definition at line 332 of file OverclockingConfig.h.

12.124.2.40 VfPointOffsetMode

```
UINT8 OVERCLOCKING_PREMEM_CONFIG::VfPointOffsetMode[MAX_OC_DOMAINS]
```

Selects Voltage & Frequency Point Offset between Legacy and Selection modes.

Need Reset System after enabling OverClocking Feature to initialize the default value. **0: In Legacy Mode, setting a global offset for the entire VF curve.** 1: In Selection modes, setting a selected VF point.

Definition at line 325 of file OverclockingConfig.h.

12.124.2.41 VoltageMode

```
UINT8 OVERCLOCKING_PREMEM_CONFIG::VoltageMode[MAX_OC_DOMAINS]
```

Voltage mode, specifies which voltage mode the processor will be operating.

0: Adaptive Mode allows the voltage curve when beyond fused range; 1: Override, sets one voltage for the entire frequency range.

Definition at line 296 of file OverclockingConfig.h.

12.124.2.42 VoltageOffset

```
INT16 OVERCLOCKING_PREMEM_CONFIG::VoltageOffset[MAX_OC_DOMAINS]
```

The voltage offset applied on top of all other voltage modes.

This offset is applied over the entire frequency range. This is a 2's complement number in mV units. **Default: 0** Range: -1000 to 1000.

Definition at line 318 of file OverclockingConfig.h.

12.124.2.43 VoltageOverride

```
UINT16 OVERCLOCKING_PREMEM_CONFIG::VoltageOverride[MAX_OC_DOMAINS]
```

The voltage override which is applied to the entire range of cpu core frequencies.

Used when VoltageMode = Override. **0. no override**. Range 0-2000 mV.

Definition at line 307 of file OverclockingConfig.h.

The documentation for this struct was generated from the following file:

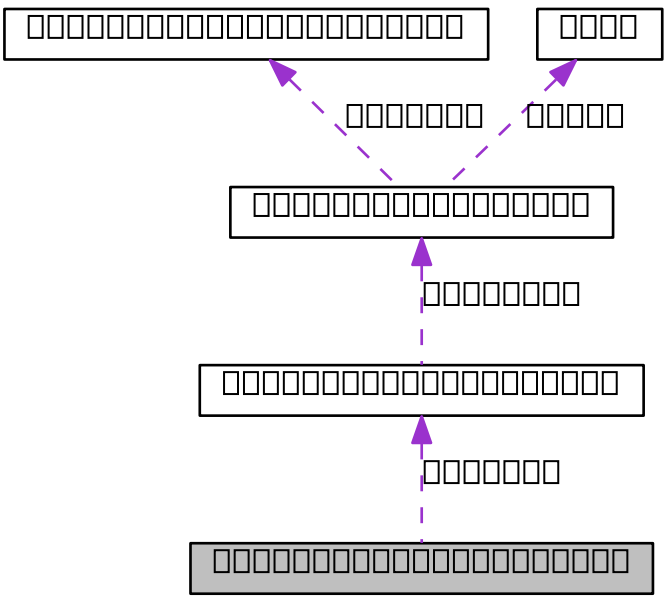
- OverclockingConfig.h

12.125 PCH_DCI_PREMEM_CONFIG Struct Reference

The PCH_DCI_PREMEM_CONFIG block describes policies related to Direct Connection Interface (DCI)

```
#include <DciConfig.h>
```

Collaboration diagram for PCH_DCI_PREMEM_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT8 DciEn

  DCI enable.
- UINT8 DciDbcMode

  USB DbC enable mode.
- UINT8 DciClkEnable

  Determine if to enable or disable DCI clock request in lowest power state.
- UINT8 KeepEarlyTrace

  Early trace is activated by default.

12.125.1 Detailed Description

The PCH_DCI_PREMEM_CONFIG block describes policies related to Direct Connection Interface (DCI)

Revision 1:

- Initial version.

Definition at line 65 of file DciConfig.h.

12.125.2 Member Data Documentation

12.125.2.1 DciClkEnable

```
UINT8 PCH_DCI_PREMEM_CONFIG::DciClkEnable
```

Determine if to enable or disable DCI clock request in lowest power state.

0:Disabled; **1:Enabled**

Definition at line 84 of file DciConfig.h.

12.125.2.2 DciDbcMode

```
UINT8 PCH_DCI_PREMEM_CONFIG::DciDbcMode
```

USB DbC enable mode.

Disabled: Clear both USB2/3DBCEN; USB2: Set USB2DBCEN; USB3: Set USB3DBCEN; Both: Set both U↩ SB2/3DBCEN; No Change: Comply with HW value Refer to definition of DCI_USB_DBC_MODE for supported settings. 0:Disabled; 1:USB2; 2:USB3; 3:Both; **4:No Change**

Definition at line 79 of file DciConfig.h.

12.125.2.3 DciEn

```
UINT8 PCH_DCI_PREMEM_CONFIG::DciEn
```

DCI enable.

Determine if to enable DCI debug from host. **0:Disabled**; 1:Enabled

Definition at line 72 of file DciConfig.h.

12.125.2.4 KeepEarlyTrace

```
UINT8 PCH_DCI_PREMEM_CONFIG::KeepEarlyTrace
```

Early trace is activated by default.

Enable to keep early trace data and keep tracing; disable to stop tracing; When debug probe is connected and granted, force to no change, let host tool to set it. **0: Disable**; 1: Enable early trace; 2:No Change;

Definition at line 91 of file DciConfig.h.

The documentation for this struct was generated from the following file:

- DciConfig.h

12.126 PCH_DEVICE_INTERRUPT_CONFIG Struct Reference

The PCH_DEVICE_INTERRUPT_CONFIG block describes interrupt pin, IRQ and interrupt mode for PCH device.

```
#include <InterruptConfig.h>
```

Public Attributes

- UINT8 Device

  Device number.
- UINT8 Function

  Device function.
- UINT8 IntX

  Interrupt pin: INTA-INTD (see PCH_INT_PIN)
- UINT8 Irq

  IRQ to be set for device.

12.126.1 Detailed Description

The PCH_DEVICE_INTERRUPT_CONFIG block describes interrupt pin, IRQ and interrupt mode for PCH device.

Definition at line 45 of file InterruptConfig.h.

The documentation for this struct was generated from the following file:

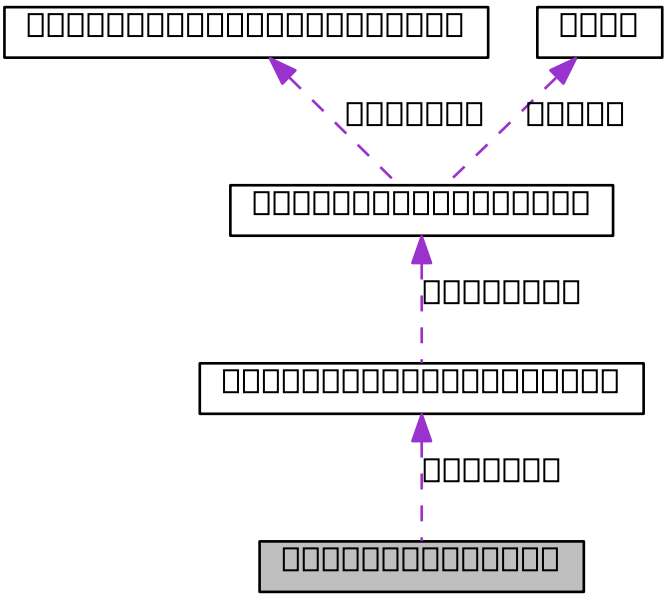
- InterruptConfig.h

12.127 PCH_DMI_CONFIG Struct Reference

The PCH_DMI_CONFIG block describes the expected configuration of the PCH for DMI.

`#include <PchDmiConfig.h>`

Collaboration diagram for PCH_DMI_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header
  Config Block Header.
- UINT32 PwrOptEnable: 1
  ***0: Disable**; 1: Enable DMI Power Optimizer on PCH side.*
- UINT32 DmiAspmCtrl: 8

*ASPM configuration on the PCH side of the DMI/OPI Link. Default is **PchPcieAspmAutoConfig***

- UINT32 CwbEnable: 1

  *0: Disable; **1: Enable** Central Write Buffer feature configurable and enabled by default*

- UINT32 L1RpCtl: 1

  *0: Disable; **1: Enable** Allow DMI enter L1 when all root ports are in L1, L0s or link down. Disabled by default.*

- UINT32 DmiPowerReduction: 1

  When set to TRUE turns on:

- UINT32 ClockGating: 1

  0: Disable; 1: Enable clock gating.

- UINT32 Rsvdbits: 19

  Reserved bits.

12.127.1 Detailed Description

The PCH_DMI_CONFIG block describes the expected configuration of the PCH for DMI.

Revision 1:

- Initial version.

Definition at line 36 of file PchDmiConfig.h.

12.127.2 Member Data Documentation

12.127.2.1 DmiPowerReduction

```
UINT32 PCH_DMI_CONFIG::DmiPowerReduction
```

When set to TRUE turns on:

- L1 State Controller Power Gating
- L1 State PHY Data Lane Power Gating
- PHY Common Lane Power Gating
- Hardware Autonomous Enable
- PMC Request Enable and Sleep Enable

Definition at line 51 of file PchDmiConfig.h.

The documentation for this struct was generated from the following file:

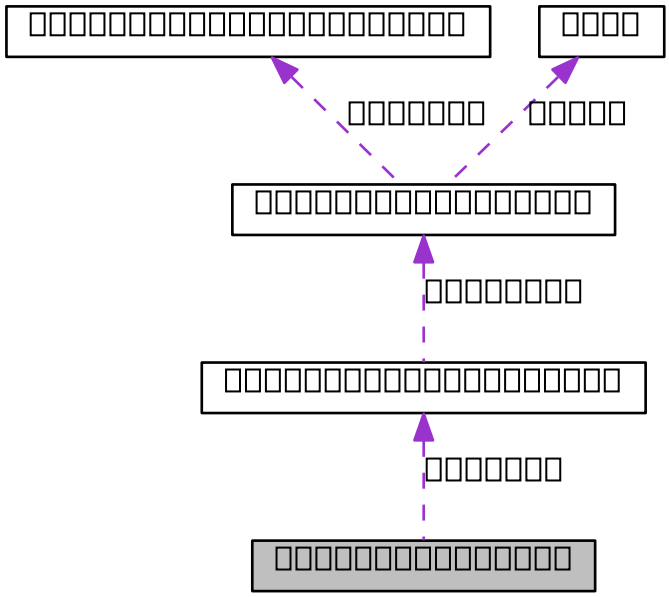
- PchDmiConfig.h

12.128 PCH_ESPI_CONFIG Struct Reference

This structure contains the policies which are related to ESPI.

`#include <EspiConfig.h>`

Collaboration diagram for PCH_ESPI_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT32 LgmrEnable: 1

  LPC (eSPI) Memory Range Decode Enable.
- UINT32 BmeHostDeviceEnabled: 1

  eSPI host and device BME settings.
- UINT32 HostC10ReportEnable: 1

  Host HOST_C10 (Virtual Wire) to device Enable (VWHC10OE) ***0b: Disable HOST_C10 reporting (HOST_C10 indication from PMC is ignored)*** *1b: Enable HOST_C10 reporting to device via eSPI Virtual Wire (upon receiving a HOST_C10 indication from PMC)*
- UINT32 LockLinkConfiguration: 1

  eSPI Link Configuration Lock (SBLCL) If set to TRUE then communication through SET_CONFIG/GET_CONFIG to eSPI device addresses from range 0x0 - 0x7FF ***1: TRUE****, 0: FALSE*
- UINT32 EspiPmHAE: 1

  Hardware Autonomous Enable (HAE) If set to TRUE, then the IP may request a PG whenever it is idle.
- UINT32 HideNonFatalErrors: 1

  Disable/Enable Non-Fatal Error Reporting (PCNFEE) If set to TRUE, then NFE bit of PCERR_SLV0 and PCERR_↩ SLV1 is cleared which disables non-fatal error reporting.

- UINT32 NmiEnableCs1: 1

  eSPI NMI Enable CS1 Set this bit to enable eSPI NMI VW events to be processed by the SOC
- UINT32 RsvdBits: 25

  Reserved bits.

12.128.1 Detailed Description

This structure contains the policies which are related to ESPI.

Revision 1:

- Initial revision **Revision 2**:
- Added LockLinkConfiguration field to config block

Definition at line 37 of file EspiConfig.h.

12.128.2 Member Data Documentation

12.128.2.1 BmeHostDeviceEnabled

```
UINT32 PCH_ESPI_CONFIG::BmeHostDeviceEnabled
```

eSPI host and device BME settings.

When TRUE, then the BME bit enabled in eSPI host and device. 0: FALSE, **1: TRUE**

Definition at line 50 of file EspiConfig.h.

12.128.2.2 LgmrEnable

```
UINT32 PCH_ESPI_CONFIG::LgmrEnable
```

LPC (eSPI) Memory Range Decode Enable.

When TRUE, then the range specified in PCLGMR[31:16] is enabled for decoding to LPC (eSPI). **0: FALSE**, 1: TRUE

Definition at line 44 of file EspiConfig.h.

The documentation for this struct was generated from the following file:

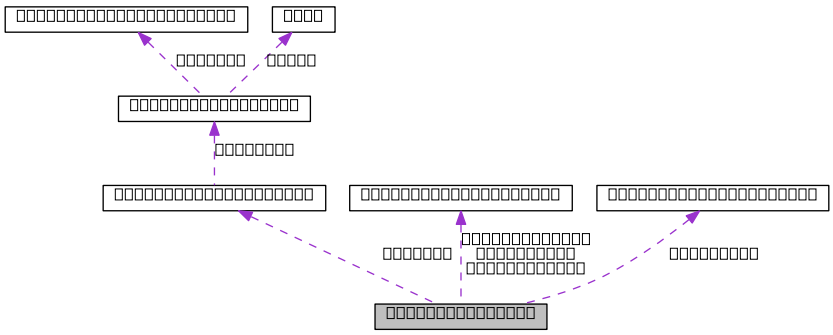
- EspiConfig.h

12.129 PCH_FIVR_CONFIG Struct Reference

The PCH_FIVR_CONFIG block describes FIVR settings.

```
#include <FivrConfig.h>
```

Collaboration diagram for PCH_FIVR_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- FIVR_EXT_RAIL_CONFIG ExtV1p05Rail

  External V1P05 VR rail configuration.
- FIVR_EXT_RAIL_CONFIG ExtVnnRail

  External Vnn VR rail configuration.
- FIVR_EXT_RAIL_CONFIG ExtVnnRailSx

  Additional External Vnn VR rail configuration that will get applied in Sx entry SMI callback.
- FIVR_VCCIN_AUX_CONFIG VccinAux

  VCCIN_AUX voltage rail configuration.
- UINT32 FivrDynPm: 1

  Enable/Disable FIVR Dynamic Power Management Default is ***1*** .

12.129.1 Detailed Description

The PCH_FIVR_CONFIG block describes FIVR settings.

Definition at line 150 of file FivrConfig.h.

12.129.2 Member Data Documentation

12.129.2.1 ExtVnnRailSx

```
FIVR_EXT_RAIL_CONFIG PCH_FIVR_CONFIG::ExtVnnRailSx
```

Additional External Vnn VR rail configuration that will get applied in Sx entry SMI callback.

Required only if External Vnn VR needs different settings for Sx than those specified in ExtVnnRail.

Definition at line 165 of file FivrConfig.h.

The documentation for this struct was generated from the following file:

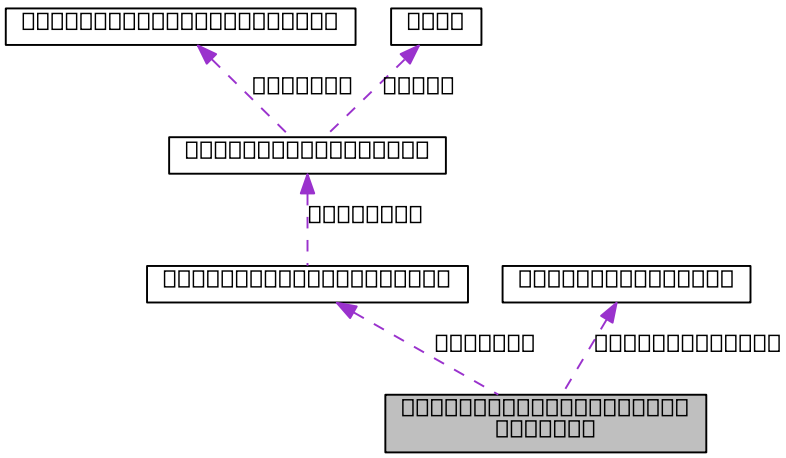
- FivrConfig.h

12.130 PCH_FLASH_PROTECTION_CONFIG Struct Reference

The PCH provides a method for blocking writes and reads to specific ranges in the SPI flash when the Protected Ranges are enabled.

```
#include <FlashProtectionConfig.h>
```

Collaboration diagram for PCH_FLASH_PROTECTION_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- PROTECTED_RANGE ProtectRange [PCH_FLASH_PROTECTED_RANGES]

  Protected Flash Ranges.

12.130.1 Detailed Description

The PCH provides a method for blocking writes and reads to specific ranges in the SPI flash when the Protected Ranges are enabled.

PROTECTED_RANGE is used to specify if flash protection are enabled, the write protection enable bit and the read protection enable bit, and to specify the upper limit and lower base for each register Platform code is responsible to get the range base by PchGetSpiRegionAddresses routine, and set the limit and base accordingly.

Definition at line 62 of file FlashProtectionConfig.h.

The documentation for this struct was generated from the following file:

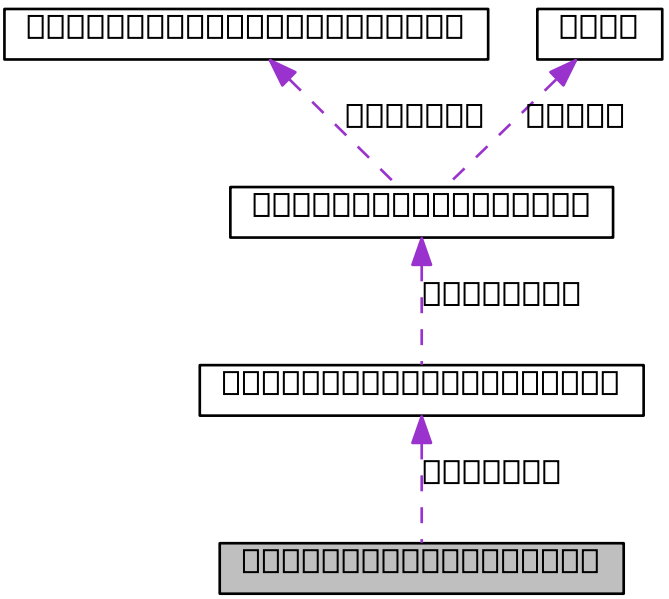
- FlashProtectionConfig.h

12.131 PCH_GENERAL_CONFIG Struct Reference

PCH General Configuration **Revision 2**: - Add VtdEnabled field.

```
#include <PchGeneralConfig.h>
```

Collaboration diagram for PCH_GENERAL_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT32 Crid: 1

  This member describes whether or not the Compatibility Revision ID (CRID) feature of PCH should be enabled.
- UINT32 LegacyIoLowLatency: 1

  Set to enable low latency of legacy IO.
- UINT32 VtdEnabled: 1

  This member describes whether or not the VTD feature should be enabled in PCH.
- UINT32 RsvdBits0: 29

  Reserved bits.

12.131.1 Detailed Description

PCH General Configuration **Revision 2**: - Add VtdEnabled field.

Definition at line 41 of file PchGeneralConfig.h.

12.131.2 Member Data Documentation

12.131.2.1 Crid

```
UINT32 PCH_GENERAL_CONFIG::Crid
```

This member describes whether or not the Compatibility Revision ID (CRID) feature of PCH should be enabled.

0: Disable; 1: Enable

Definition at line 47 of file PchGeneralConfig.h.

12.131.2.2 LegacyIoLowLatency

```
UINT32 PCH_GENERAL_CONFIG::LegacyIoLowLatency
```

Set to enable low latency of legacy IO.

Some systems require lower IO latency irrespective of power. This is a tradeoff between power and IO latency.

Note

: Once this is enabled, DmiAspm, Pcie DmiAspm in SystemAgent and ITSS Clock Gating are forced to disabled. **0: Disable**, 1: Enable

Definition at line 56 of file PchGeneralConfig.h.

12.131.2.3 VtdEnabled

```
UINT32 PCH_GENERAL_CONFIG::VtdEnabled
```

This member describes whether or not the VTD feature should be enabled in PCH.

0: Disable; 1: Enable

Definition at line 61 of file PchGeneralConfig.h.

The documentation for this struct was generated from the following file:

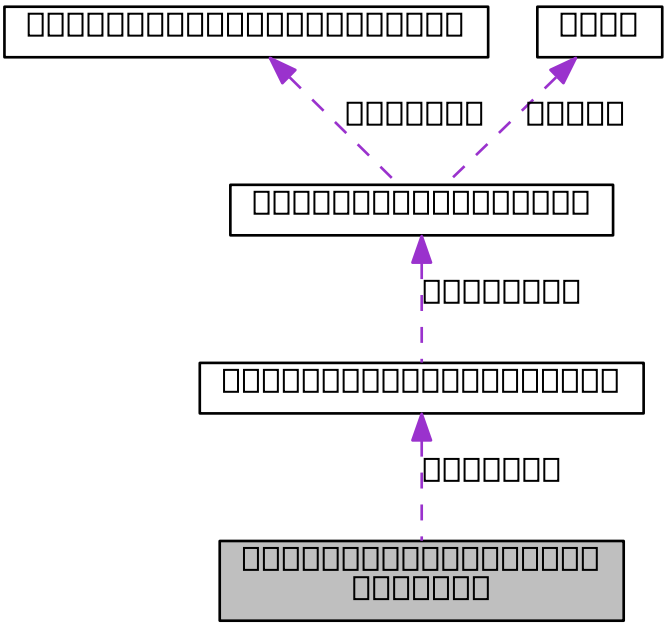
- PchGeneralConfig.h

12.132 PCH_GENERAL_PREMEM_CONFIG Struct Reference

PCH General Pre-Memory Configuration **Revision 1**: - Initial version.

```
#include <PchGeneralConfig.h>
```

Collaboration diagram for PCH_GENERAL_PREMEM_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT32 Port80Route: 1

  *Control where the Port 80h cycles are sent, **0: LPC**; 1: PCI.*
- UINT32 GpioOverride: 3

  *Gpio override Level – **0: Disable**;.*
- UINT32 IoeDebugEn: 1

  Enable/Disable IOE Debug.
- UINT32 PmodeClkEn: 1

  Enable/Disable PMODE clock.
- UINT32 PmcPrivacyConsent: 1

  Set/Clear PMC Privacy Consent.
- UINT32 DisableResets: 2

  Disable/Enable Resets triggered by the system.
- UINT32 RsvdBits0: 23

  Reserved bits.

12.132.1 Detailed Description

PCH General Pre-Memory Configuration **Revision 1**: - Initial version.

Revision 2: - Added GpioOverride. **Revision 3**: - Added IoeDebugEn, PmodeClkEn **Revision 4**: - Added Pmc↩ PrivacyConsent

Definition at line 72 of file PchGeneralConfig.h.

12.132.2 Member Data Documentation

12.132.2.1 DisableResets

```
UINT32 PCH_GENERAL_PREMEM_CONFIG::DisableResets
```

Disable/Enable Resets triggered by the system.

0: Platform POR; 1: Force Enable; 2: Force Disable.

Definition at line 103 of file PchGeneralConfig.h.

12.132.2.2 GpioOverride

```
UINT32 PCH_GENERAL_PREMEM_CONFIG::GpioOverride
```

Gpio override Level – **0: Disable**;.

- 1: Override Level 1 - Skips GPIO configuration in PEI/FSPM/FSPT phase
- 2: Override Level 2 - Reserved for future use

Definition at line 84 of file PchGeneralConfig.h.

12.132.2.3 IoeDebugEn

```
UINT32 PCH_GENERAL_PREMEM_CONFIG::IoeDebugEn
```

Enable/Disable IOE Debug.

When enabled, IOE D2D Dfx link and clock will keep up for debug **0: Disable**; 1: Enable

Definition at line 89 of file PchGeneralConfig.h.

12.132.2.4 PmcPrivacyConsent

```
UINT32 PCH_GENERAL_PREMEM_CONFIG::PmcPrivacyConsent
```

Set/Clear PMC Privacy Consent.

When enabled, PMC will collect unique ID in CrashLog **0: Disable**; 1: Enable

Definition at line 99 of file PchGeneralConfig.h.

12.132.2.5 PmodeClkEn

```
UINT32 PCH_GENERAL_PREMEM_CONFIG::PmodeClkEn
```

Enable/Disable PMODE clock.

When enabled, Pmode clock will toggle for XDP use **0: Disable**; 1: Enable

Definition at line 94 of file PchGeneralConfig.h.

The documentation for this struct was generated from the following file:

- PchGeneralConfig.h

12.133 PCH_HSIO_PCIE_LANE_CONFIG Struct Reference

The PCH_HSIO_PCIE_LANE_CONFIG describes HSIO settings for PCIe lane.

`#include <HsioPcieConfig.h>`

Public Attributes

- UINT32 HsioRxSetCtleEnable: 1

  ***0: Disable**; 1: Enable PCH PCIe Gen 3 Set CTLE Value*
- UINT32 HsioRxSetCtle: 6

  PCH PCIe Gen 3 Set CTLE Value.
- UINT32 HsioTxGen1DownscaleAmpEnable: 1

  ***0: Disable**; 1: Enable PCH PCIe Gen 1 TX Output Downscale Amplitude Adjustment value override*
- UINT32 HsioTxGen1DownscaleAmp: 6

  PCH PCIe Gen 1 TX Output Downscale Amplitude Adjustment value.
- UINT32 HsioTxGen2DownscaleAmpEnable: 1

  ***0: Disable**; 1: Enable PCH PCIe Gen 2 TX Output Downscale Amplitude Adjustment value override*
- UINT32 HsioTxGen2DownscaleAmp: 6

  PCH PCIe Gen 2 TX Output Downscale Amplitude Adjustment value.
- UINT32 HsioTxGen3DownscaleAmpEnable: 1

  ***0: Disable**; 1: Enable PCH PCIe Gen 3 TX Output Downscale Amplitude Adjustment value override*
- UINT32 HsioTxGen3DownscaleAmp: 6

  PCH PCIe Gen 3 TX Output Downscale Amplitude Adjustment value.
- UINT32 RsvdBits0: 4

  Reserved Bits.
- UINT32 HsioTxGen1DeEmphEnable: 1

  ***0: Disable**; 1: Enable PCH PCIe Gen 1 TX Output De-Emphasis Adjustment Setting value override*
- UINT32 HsioTxGen1DeEmph: 6

  PCH PCIe Gen 1 TX Output De-Emphasis Adjustment Setting.
- UINT32 HsioTxGen2DeEmph3p5Enable: 1

  ***0: Disable**; 1: Enable PCH PCIe Gen 2 TX Output -3.5dB Mode De-Emphasis Adjustment Setting value override*
- UINT32 HsioTxGen2DeEmph3p5: 6

  PCH PCIe Gen 2 TX Output -3.5dB Mode De-Emphasis Adjustment Setting.
- UINT32 HsioTxGen2DeEmph6p0Enable: 1

  ***0: Disable**; 1: Enable PCH PCIe Gen 2 TX Output -6.0dB Mode De-Emphasis Adjustment Setting value override*
- UINT32 HsioTxGen2DeEmph6p0: 6

  PCH PCIe Gen 2 TX Output -6.0dB Mode De-Emphasis Adjustment Setting.
- UINT32 RsvdBits1: 11

  Reserved Bits.

12.133.1 Detailed Description

The PCH_HSIO_PCIE_LANE_CONFIG describes HSIO settings for PCIe lane.

Definition at line 34 of file HsioPcieConfig.h.

The documentation for this struct was generated from the following file:

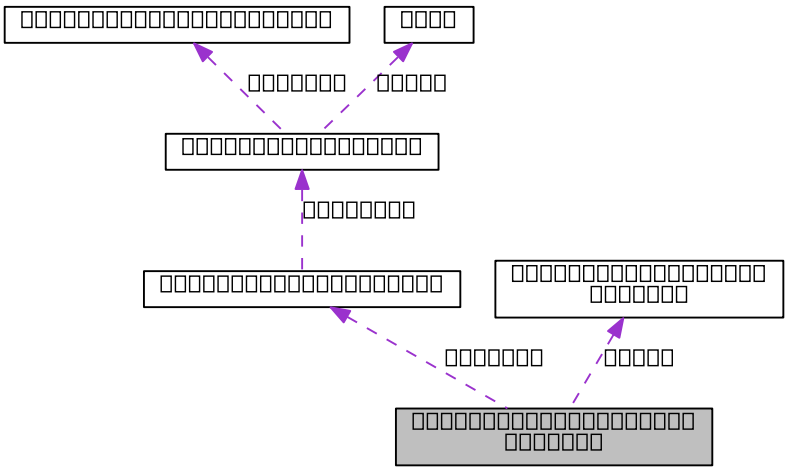
- HsioPcieConfig.h

12.134 PCH_HSIO_PCIE_PREMEM_CONFIG Struct Reference

The PCH_HSIO_PCIE_CONFIG block describes the configuration of the HSIO for PCIe lanes.

`#include <HsioPcieConfig.h>`

Collaboration diagram for PCH_HSIO_PCIE_PREMEM_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header
  Config Block Header.
- PCH_HSIO_PCIE_LANE_CONFIG Lane [PCH_MAX_PCIE_ROOT_PORTS]
  These members describe the configuration of HSIO for PCIe lanes.

12.134.1 Detailed Description

The PCH_HSIO_PCIE_CONFIG block describes the configuration of the HSIO for PCIe lanes.

Definition at line 62 of file HsioPcieConfig.h.

The documentation for this struct was generated from the following file:

- HsioPcieConfig.h

12.135 PCH_HSIO_SATA_PORT_LANE Struct Reference

The PCH_HSIO_SATA_PORT_LANE describes HSIO settings for SATA Port lane.

`#include <HsioSataConfig.h>`

Public Attributes

- UINT32 HsioRxGen1EqBoostMagEnable: 1

  ***0: Disable**; 1: Enable Receiver Equalization Boost Magnitude Adjustment Value override*
- UINT32 HsioRxGen1EqBoostMag: 6

  SATA 1.5 Gb/sReceiver Equalization Boost Magnitude Adjustment value.
- UINT32 HsioRxGen2EqBoostMagEnable: 1

  ***0: Disable**; 1: Enable Receiver Equalization Boost Magnitude Adjustment Value override*
- UINT32 HsioRxGen2EqBoostMag: 6

  SATA 3.0 Gb/sReceiver Equalization Boost Magnitude Adjustment value.
- UINT32 HsioRxGen3EqBoostMagEnable: 1

  ***0: Disable**; 1: Enable Receiver Equalization Boost Magnitude Adjustment Value override*
- UINT32 HsioRxGen3EqBoostMag: 6

  SATA 6.0 Gb/sReceiver Equalization Boost Magnitude Adjustment value.
- UINT32 HsioTxGen1DownscaleAmpEnable: 1

  ***0: Disable**; 1: Enable SATA 1.5 Gb/s TX Output Downscale Amplitude Adjustment value override*
- UINT32 HsioTxGen1DownscaleAmp: 6

  SATA 1.5 Gb/s TX Output Downscale Amplitude Adjustment value.
- UINT32 RsvdBits0: 4

  Reserved bits.
- UINT32 HsioTxGen2DownscaleAmpEnable: 1

  ***0: Disable**; 1: Enable SATA 3.0 Gb/s TX Output Downscale Amplitude Adjustment value override*
- UINT32 HsioTxGen2DownscaleAmp: 6

  SATA 3.0 Gb/s TX Output Downscale Amplitude Adjustment.
- UINT32 HsioTxGen3DownscaleAmpEnable: 1

  ***0: Disable**; 1: Enable SATA 6.0 Gb/s TX Output Downscale Amplitude Adjustment value override*
- UINT32 HsioTxGen3DownscaleAmp: 6

  SATA 6.0 Gb/s TX Output Downscale Amplitude Adjustment.
- UINT32 HsioTxGen1DeEmphEnable: 1

  ***0: Disable**; 1: Enable SATA 1.5 Gb/s TX Output De-Emphasis Adjustment Setting value override*
- UINT32 HsioTxGen1DeEmph: 6

  SATA 1.5 Gb/s TX Output De-Emphasis Adjustment Setting.
- UINT32 HsioTxGen2DeEmphEnable: 1

  ***0: Disable**; 1: Enable SATA 3.0 Gb/s TX Output De-Emphasis Adjustment Setting value override*
- UINT32 HsioTxGen2DeEmph: 6

  SATA 3.0 Gb/s TX Output De-Emphasis Adjustment Setting.
- UINT32 RsvdBits1: 4

  Reserved bits.
- UINT32 HsioTxGen3DeEmphEnable: 1

  ***0: Disable**; 1: Enable SATA 6.0 Gb/s TX Output De-Emphasis Adjustment Setting value override*
- UINT32 HsioTxGen3DeEmph: 6

  SATA 6.0 Gb/s TX Output De-Emphasis Adjustment Setting value override.
- UINT32 RsvdBits2: 25

  Reserved bits.

12.135.1 Detailed Description

The PCH_HSIO_SATA_PORT_LANE describes HSIO settings for SATA Port lane.

Definition at line 32 of file HsioSataConfig.h.

The documentation for this struct was generated from the following file:

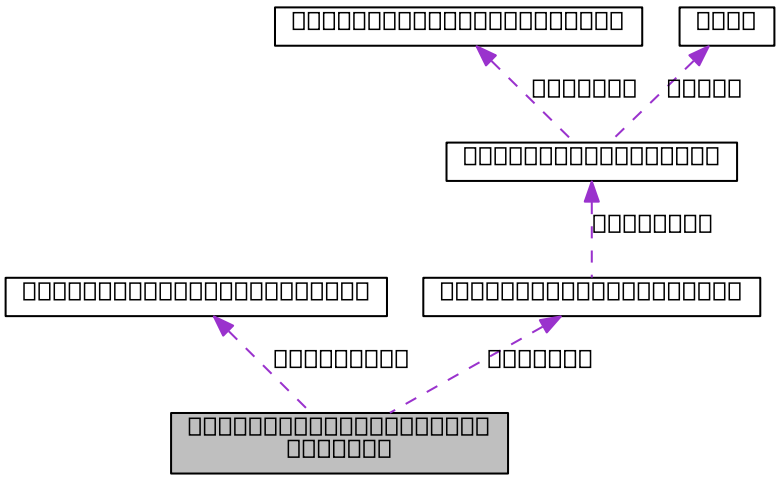
- HsioSataConfig.h

12.136 PCH_HSIO_SATA_PREMEM_CONFIG Struct Reference

The PCH_HSIO_SATA_CONFIG block describes the HSIO configuration of the SATA controller.

`#include <HsioSataConfig.h>`

Collaboration diagram for PCH_HSIO_SATA_PREMEM_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- PCH_HSIO_SATA_PORT_LANE PortLane [PCH_MAX_SATA_PORTS]

  These members describe the configuration of HSIO for SATA lanes.

12.136.1 Detailed Description

The PCH_HSIO_SATA_CONFIG block describes the HSIO configuration of the SATA controller.

Definition at line 68 of file HsioSataConfig.h.

The documentation for this struct was generated from the following file:

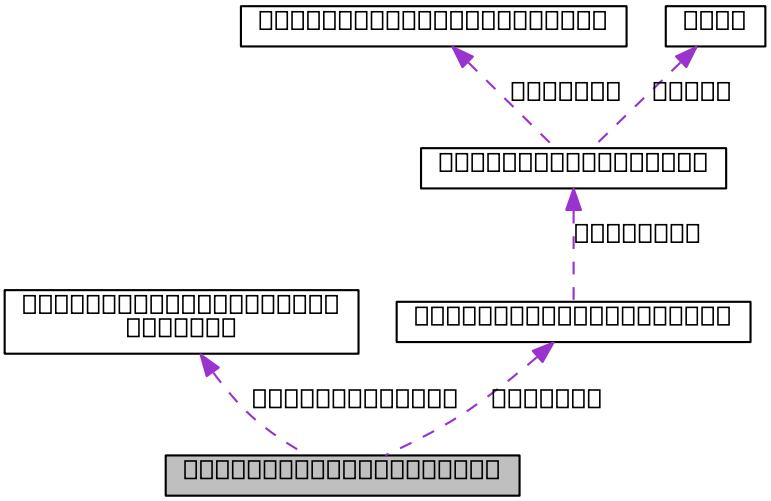
- HsioSataConfig.h

12.137 PCH_INTERRUPT_CONFIG Struct Reference

The PCH_INTERRUPT_CONFIG block describes interrupt settings for PCH.

```
#include <InterruptConfig.h>
```

Collaboration diagram for PCH_INTERRUPT_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT8 NumOfDevIntConfig

  Number of entries in DevIntConfig table.
- UINT8 Rsvd0 [3]

  Reserved bytes, align to multiple 4.
- PCH_DEVICE_INTERRUPT_CONFIG DevIntConfig [PCH_MAX_DEVICE_INTERRUPT_CONFIG]

  Array which stores PCH devices interrupts settings.
- UINT8 GpioIrqRoute

  *Interrupt routing for GPIO. Default is **14**.*
- UINT8 SciIrqSelect

  *Interrupt select for SCI. Default is **9**.*
- UINT8 TcoIrqSelect

  *Interrupt select for TCO. Default is **9**.*
- UINT8 TcoIrqEnable

  *Enable IRQ generation for TCO. **0: Disable**; 1: Enable.*

12.137.1 Detailed Description

The PCH_INTERRUPT_CONFIG block describes interrupt settings for PCH.

Definition at line 61 of file InterruptConfig.h.

The documentation for this struct was generated from the following file:

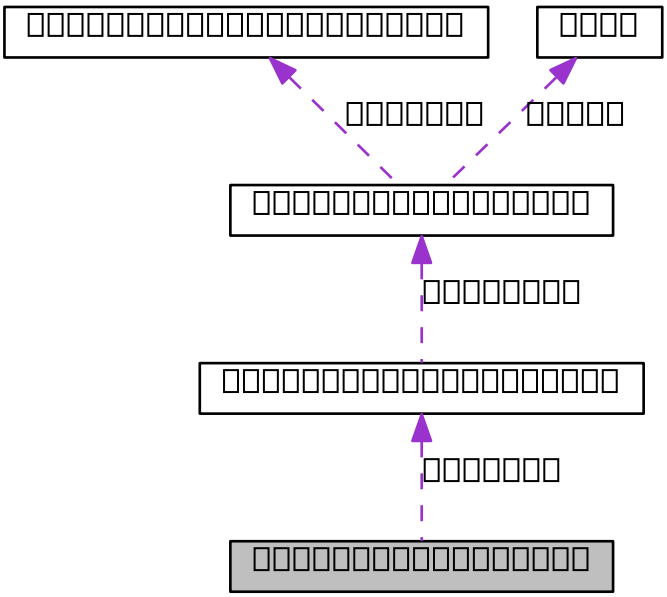
- InterruptConfig.h

12.138 PCH_IOAPIC_CONFIG Struct Reference

The PCH_IOAPIC_CONFIG block describes the expected configuration of the PCH IO APIC, it's optional and PCH code would ignore it if the BdfValid bit is not TRUE.

```
#include <IoApicConfig.h>
```

Collaboration diagram for PCH_IOAPIC_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT32 IoApicEntry24_119: 1

  *0: Disable; **1: Enable** IOAPIC Entry 24-119*
- UINT32 Enable8254ClockGating: 1

  Enable 8254 Static Clock Gating during early POST time.
- UINT32 Enable8254ClockGatingOnS3: 1

  Enable 8254 Static Clock Gating on S3 resume path.
- UINT32 RsvdBits1: 29

  Reserved bits.
- UINT8 IoApicId

  *This member determines IOAPIC ID. Default is **0x02**.*
- UINT8 Rsvd0 [3]

  Reserved bytes.

12.138.1 Detailed Description

The PCH_IOAPIC_CONFIG block describes the expected configuration of the PCH IO APIC, it's optional and PCH code would ignore it if the BdfValid bit is not TRUE.

Bus:device:function fields will be programmed to the register P2SB IBDF(P2SB PCI offset R6Ch-6Dh), it's using for the following purpose: As the Requester ID when initiating Interrupt Messages to the processor. As the Completer ID when responding to the reads targeting the IOxAPI's Memory-Mapped I/O registers. This field defaults to Bus 0: Device 31: Function 0 after reset. BIOS can program this field to provide a unique Bus:Device:Function number for the internal IOxAPIC. The address resource range of IOAPIC must be reserved in E820 and ACPI as system resource.

Definition at line 43 of file IoApicConfig.h.

12.138.2 Member Data Documentation

12.138.2.1 Enable8254ClockGating

```
UINT32 PCH_IOAPIC_CONFIG::Enable8254ClockGating
```

Enable 8254 Static Clock Gating during early POST time.

0: Disable, **1: Enable** Setting 8254CGE is required to support SLP_S0. Enable this if 8254 timer is not used. However, set 8254CGE=1 in POST time might fail to boot legacy OS using 8254 timer. Make sure it is disabled to support legacy OS using 8254 timer.

Note

: For some OS environment that it needs to set 8254CGE in late state it should set this policy to FALSE and use ItssSet8254ClockGateState (TRUE) in SMM later. This is also required during S3 resume. To avoid SMI requirement in S3 reusme path, it can enable the Enable8254ClockGatingOnS3 and RC will do 8254 CGE programming in PEI during S3 resume with BOOT_SAI.

Definition at line 59 of file IoApicConfig.h.

12.138.2.2 Enable8254ClockGatingOnS3

```
UINT32 PCH_IOAPIC_CONFIG::Enable8254ClockGatingOnS3
```

Enable 8254 Static Clock Gating on S3 resume path.

0: Disable, **1: Enable** This is only applicable when Enable8254ClockGating is disabled. If Enable8254ClockGating is enabled, RC will do the 8254 CGE programming on S3 resume path as well.

Definition at line 66 of file IoApicConfig.h.

The documentation for this struct was generated from the following file:

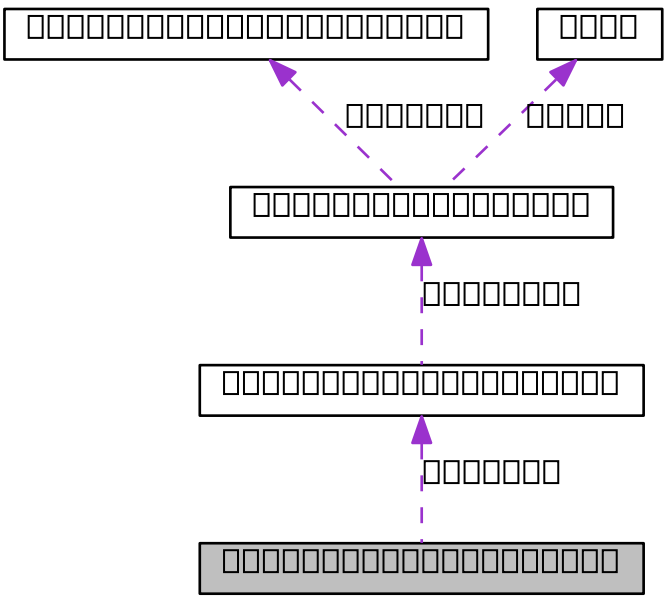
- IoApicConfig.h

12.139 PCH_LOCK_DOWN_CONFIG Struct Reference

The PCH_LOCK_DOWN_CONFIG block describes the expected configuration of the PCH for security requirement.

```
#include <LockDownConfig.h>
```

Collaboration diagram for PCH_LOCK_DOWN_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT32 GlobalSmi: 1

  ***(Test)** Enable SMI_LOCK bit to prevent writes to the Global SMI Enable bit.*
- UINT32 BiosInterface: 1

  ***(Test)** Enable BIOS Interface Lock Down bit to prevent writes to the Backup Control Register Top Swap bit and the General Control and Status Registers Boot BIOS Straps.*
- UINT32 BiosLock: 1

  Enable the BIOS Lock Enable (BLE) feature and set EISS bit (D31:F5:RegDCh[5]) for the BIOS region protection.
- UINT32 SpiEiss: 1

  Separate EISS policy to be handled separately from BiosLock.
- UINT32 UnlockGpioPads: 1

  ***(Test)** This test option when set will force all GPIO pads to be unlocked before BIOS transitions to POSTBOOT_SAI.*
- UINT32 RsvdBits0: 27

  Reserved bits.

12.139.1 Detailed Description

The PCH_LOCK_DOWN_CONFIG block describes the expected configuration of the PCH for security requirement.

Definition at line 35 of file LockDownConfig.h.

12.139.2 Member Data Documentation

12.139.2.1 BiosInterface

```
UINT32 PCH_LOCK_DOWN_CONFIG::BiosInterface
```

(Test) Enable BIOS Interface Lock Down bit to prevent writes to the Backup Control Register Top Swap bit and the General Control and Status Registers Boot BIOS Straps.

Intel strongly recommends that BIOS sets the BIOS Interface Lock Down bit. Enabling this bit will mitigate malicious software attempts to replace the system BIOS with its own code. 0: Disable; **1: Enable**.

Definition at line 48 of file LockDownConfig.h.

12.139.2.2 BiosLock

```
UINT32 PCH_LOCK_DOWN_CONFIG::BiosLock
```

Enable the BIOS Lock Enable (BLE) feature and set EISS bit (D31:F5:RegDCh[5]) for the BIOS region protection.

When it is enabled, the BIOS Region can only be modified from SMM. If this EISS bit is set, then WPD must be a '1' and InSMM.STS must be '1' also in order to write to BIOS regions of SPI Flash. If this EISS bit is clear, then the InSMM.STS is a don't care. The BIOS must set the EISS bit while BIOS Guard support is enabled. In recovery path, platform can temporary disable EISS for SPI programming in PEI phase or early DXE phase. When PcdSmm↩ VariableEnable is FALSE, to support BIOS regions update outside of SMM, the BiosLock must be set to Disabled by platform. 0: Disable; **1: Enable.**

Definition at line 63 of file LockDownConfig.h.

12.139.2.3 GlobalSmi

```
UINT32 PCH_LOCK_DOWN_CONFIG::GlobalSmi
```

(Test) Enable SMI_LOCK bit to prevent writes to the Global SMI Enable bit.

0: Disable; **1: Enable**.

Definition at line 40 of file LockDownConfig.h.

12.139.2.4 UnlockGpioPads

```
UINT32 PCH_LOCK_DOWN_CONFIG::UnlockGpioPads
```

(Test) This test option when set will force all GPIO pads to be unlocked before BIOS transitions to POSTBOOT_SAI.

This option should not be enabled in production configuration and used only for debug purpose when free runtime reconfiguration of GPIO pads is needed. **0: Disable**; 1: Enable.

Definition at line 72 of file LockDownConfig.h.

The documentation for this struct was generated from the following file:

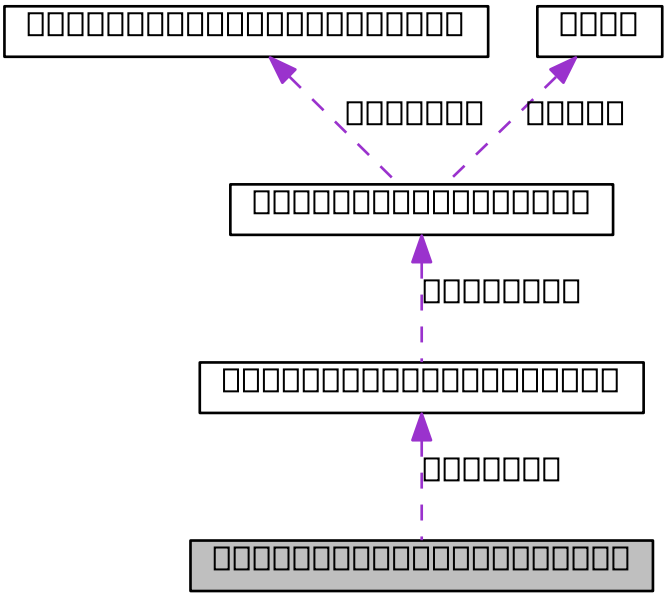
- LockDownConfig.h

12.140 PCH_LPC_PREMEM_CONFIG Struct Reference

This structure contains the policies which are related to LPC.

`#include <LpcConfig.h>`

Collaboration diagram for PCH_LPC_PREMEM_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header
  Config Block Header.
- UINT32 EnhancePort8xhDecoding: 1
  Enhance the port 8xh decoding.
- UINT32 LpcPmHAE: 1
  Hardware Autonomous Enable.
- UINT32 RsvdBits: 30
  Reserved bits.

12.140.1 Detailed Description

This structure contains the policies which are related to LPC.

Definition at line 32 of file LpcConfig.h.

12.140.2 Member Data Documentation

12.140.2.1 EnhancePort8xhDecoding

```
UINT32 PCH_LPC_PREMEM_CONFIG::EnhancePort8xhDecoding
```

Enhance the port 8xh decoding.

Original LPC only decodes one byte of port 80h, with this enhancement LPC can decode word or dword of port 80h-83h.

Note

: this will occupy one LPC generic IO range register. While this is enabled, read from port 80h always return 0x00. 0: Disable, **1: Enable**

Definition at line 40 of file LpcConfig.h.

12.140.2.2 LpcPmHAE

```
UINT32 PCH_LPC_PREMEM_CONFIG::LpcPmHAE
```

Hardware Autonomous Enable.

When enabled, LPC will automatically engage power gating when it has reached its idle condition. 0: Disable, **1: Enable**

Definition at line 46 of file LpcConfig.h.

The documentation for this struct was generated from the following file:

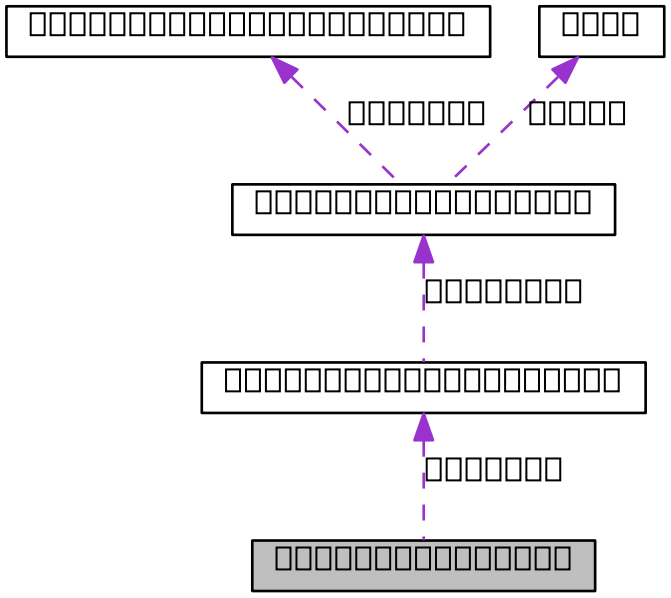
- LpcConfig.h

12.141 PCH_P2SB_CONFIG Struct Reference

This structure contains the policies which are related to P2SB device.

`#include <P2sbConfig.h>`

Collaboration diagram for PCH_P2SB_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT32 SbAccessUnlock: 1

  (Test) *The sideband MMIO register access to specific ports will be locked before 3rd party code execution.*
- UINT32 Rsvdbits: 31

  Reserved bits.

12.141.1 Detailed Description

This structure contains the policies which are related to P2SB device.

Definition at line 32 of file P2sbConfig.h.

12.141.2 Member Data Documentation

12.141.2.1 SbAccessUnlock

```
UINT32 PCH_P2SB_CONFIG::SbAccessUnlock
```

(Test) The sideband MMIO register access to specific ports will be locked before 3rd party code execution.

Currently it disables PSFx access. This policy unlocks the sideband MMIO space for those IPs. **0: Lock sideband access** ; 1: Unlock sideband access. NOTE: Do not set this policy "SbAccessUnlock" unless its necessary.

Definition at line 42 of file P2sbConfig.h.

The documentation for this struct was generated from the following file:

- P2sbConfig.h

12.142 PCH_PCIE_CLOCK Struct Reference

PCH_PCIE_CLOCK describes PCIe source clock generated by PCH.

```
#include <PchPcieRpConfig.h>
```

Public Attributes

- UINT8 Usage

  Purpose of given clock (see PCH_PCIE_CLOCK_USAGE). Default: Unused, 0xFF.
- UINT8 ClkReq

  ClkSrc - ClkReq mapping. Default: 1:1 mapping with Clock numbers.
- UINT8 RsvdBytes [2]

  Reserved byte.

12.142.1 Detailed Description

PCH_PCIE_CLOCK describes PCIe source clock generated by PCH.

Definition at line 173 of file PchPcieRpConfig.h.

The documentation for this struct was generated from the following file:

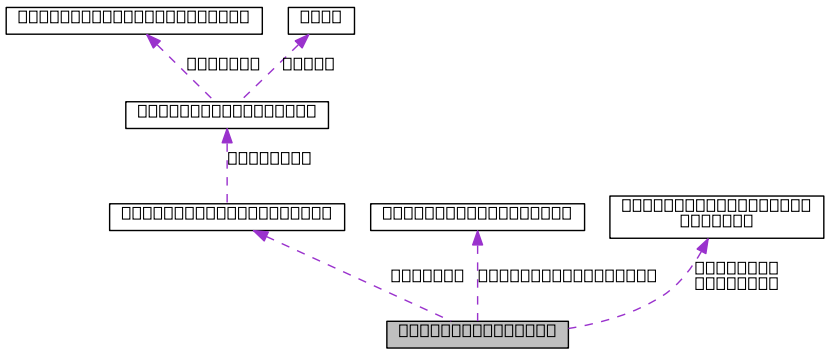
- PchPcieRpConfig.h

12.143 PCH_PCIE_CONFIG Struct Reference

The PCH_PCIE_CONFIG block describes the expected configuration of the PCH PCI Express controllers **Revision 1**:

```
#include <PchPcieRpConfig.h>
```

Collaboration diagram for PCH_PCIE_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- PCIE_COMMON_CONFIG PcieCommonConfig

  These members describe the configuration of each PCH PCIe root port.
- UINT8 PchPciePort8xhDecodePortIndex

  ***(Test)** The Index of PCIe Port that is selected for Port8xh Decode (0 Based)*
- UINT8 PcieTbtPerfBoost

  Bitmap of TBT performance boost enabled PCIe controllers to which discrete TBT controllers connect.
- UINT32 PcieClockGating

  This member describes whether the PCI Express Clock Gating for each root port is enabled by platform modules.

12.143.1 Detailed Description

The PCH_PCIE_CONFIG block describes the expected configuration of the PCH PCI Express controllers **Revision 1**:

- Initial version. **Revision 2**:
- Moved EnablePort8xhDecode policy to PCIE_COMMON_CONFIG **Revision 3**:
- Added GlobalPcieAer

Definition at line 253 of file PchPcieRpConfig.h.

12.143.2 Member Data Documentation

12.143.2.1 PcieClockGating

`UINT32 PCH_PCIE_CONFIG::PcieClockGating`

This member describes whether the PCI Express Clock Gating for each root port is enabled by platform modules.

O: POR; 1: Enable; 2: Disable

Definition at line 280 of file PchPcieRpConfig.h.

12.143.2.2 PcieTbtPerfBoost

`UINT8 PCH_PCIE_CONFIG::PcieTbtPerfBoost`

Bitmap of TBT performance boost enabled PCIe controllers to which discrete TBT controllers connect.

Each bit represents a specific controller: BIT0 - BIT7: PCIe controllers bitmap, where bit i corresponds to PCIe controller(i), e.g. BIT0 - PXPA. The max supported PCIe controller number depends on silicon.

Definition at line 274 of file PchPcieRpConfig.h.

The documentation for this struct was generated from the following file:

- PchPcieRpConfig.h

12.144 PCH_PCIE_ROOT_PORT_CONFIG Struct Reference

The PCH_PCI_EXPRESS_ROOT_PORT_CONFIG describe the feature and capability of each PCH PCIe root port.

`#include <PchPcieRpConfig.h>`

Public Attributes

- UINT8 ExtSync

  an instance of Pcie Common Config
- UINT8 SystemErrorEnable

  *Indicate whether the System Error is enabled. **0: Disable**; 1: Enable.*
- UINT8 MvcEnabled

  The Multiple VC (MVC) supports hardware to avoid HoQ block for latency sensitive TC.
- UINT8 VppPort

  Virtual Pin Port is industry standard introduced to PCIe Hot Plug support in systems when GPIO pins expansion is needed.
- UINT8 VppAddress

  PCIe Hot Plug VPP SMBus Address. Default is zero.

12.144.1 Detailed Description

The PCH_PCI_EXPRESS_ROOT_PORT_CONFIG describe the feature and capability of each PCH PCIe root port.

Definition at line 214 of file PchPcieRpConfig.h.

12.144.2 Member Data Documentation

12.144.2.1 ExtSync

`UINT8 PCH_PCIE_ROOT_PORT_CONFIG::ExtSync`

an instance of Pcie Common Config

Indicate whether the extended synch is enabled. **0: Disable**; 1: Enable.

Definition at line 216 of file PchPcieRpConfig.h.

12.144.2.2 MvcEnabled

`UINT8 PCH_PCIE_ROOT_PORT_CONFIG::MvcEnabled`

The Multiple VC (MVC) supports hardware to avoid HoQ block for latency sensitive TC.

Currently it is only applicable to Root Ports with 2pX4 port configuration with 2 VCs,or DMI port configuration with 3 VCs. For Root Ports 2pX4 configuration, two RPs (RP0, RP2) shall support two PCIe VCs (VC0 & VC1) and the other RPs (RP1, RP3) shall be disabled. **0: Disable**; 1: Enable

Definition at line 229 of file PchPcieRpConfig.h.

12.144.2.3 VppPort

`UINT8 PCH_PCIE_ROOT_PORT_CONFIG::VppPort`

Virtual Pin Port is industry standard introduced to PCIe Hot Plug support in systems when GPIO pins expansion is needed.

It is server specific feature. **0x00: Default**; 0xFF: Disabled

Definition at line 235 of file PchPcieRpConfig.h.

The documentation for this struct was generated from the following file:

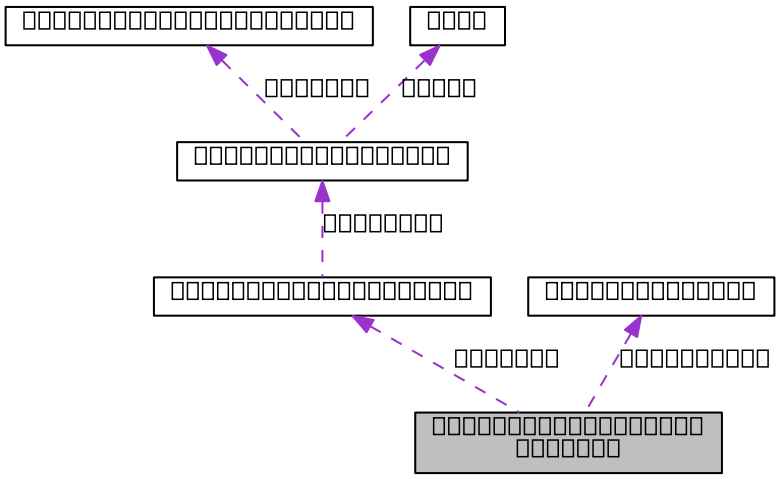
- PchPcieRpConfig.h

12.145 PCH_PCIE_RP_PREMEM_CONFIG Struct Reference

The PCH_PCIE_RP_PREMEM_CONFIG block describes early configuration of the PCH PCI Express controllers **Revision 1**:

```
#include <PchPcieRpConfig.h>
```

Collaboration diagram for PCH_PCIE_RP_PREMEM_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT32 RpEnabledMask

  Root Port enabling mask.
- PCH_PCIE_CLOCK PcieClock [PCH_MAX_PCIE_CLOCKS]

  Configuration of PCIe source clocks.
- UINT8 Bifurcation [PCH_MAX_PCIE_CONTROLLERS]

  *Per Controller Bifurcation Configuration **0: Disabled**; 1: 4x1; 2: 1x2_2x1; 3: 2x2; 4: 1x4; 5: 4x2; 6: 1x4_2x2; 7: 2x2_1x4; 8: 2x4; 9: 1x8 (see: PCIE_BIFURCATION_CONFIG)*
- UINT8 SkipExtGfxScan

  ***(Test)** :1=Skip External Gfx Device Scan; **0=Scan for external graphics devices**. Set this policy to skip External Graphics card scanning if the platform uses Internal Graphics only.*

12.145.1 Detailed Description

The PCH_PCIE_RP_PREMEM_CONFIG block describes early configuration of the PCH PCI Express controllers **Revision 1**:

- Initial version.

Definition at line 289 of file PchPcieRpConfig.h.

12.145.2 Member Data Documentation

12.145.2.1 RpEnabledMask

```
UINT32 PCH_PCIE_RP_PREMEM_CONFIG::RpEnabledMask
```

Root Port enabling mask.

Bit0 presents RP1, Bit1 presents RP2, and so on. 0: Disable; **1: Enable**.

Definition at line 296 of file PchPcieRpConfig.h.

The documentation for this struct was generated from the following file:

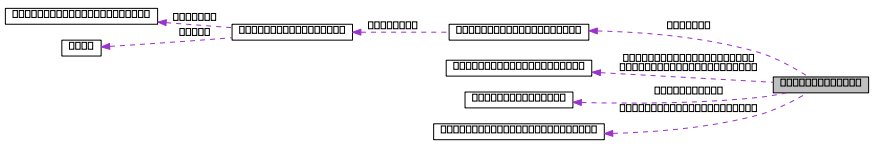
- PchPcieRpConfig.h

12.146 PCH_PM_CONFIG Struct Reference

The PCH_PM_CONFIG block describes expected miscellaneous power management settings.

```
#include <PmConfig.h>
```

Collaboration diagram for PCH_PM_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- PCH_WAKE_CONFIG WakeConfig

  Specify Wake Policy.
- UINT32 PchDeepSxPol: 4

  Deep Sx Policy. Refer to PCH_DEEP_SX_CONFIG for each value. Default is ***PchDeepSxPolDisable****.*
- UINT32 PchSlpS3MinAssert: 4

  SLP_S3 Minimum Assertion Width Policy. Refer to PCH_SLP_S3_MIN_ASSERT for each value. Default is ***Pch↩SlpS350ms****.*
- UINT32 PchSlpS4MinAssert: 4

  SLP_S4 Minimum Assertion Width Policy. Refer to PCH_SLP_S4_MIN_ASSERT for each value. Default is ***Pch↩SlpS44s****.*
- UINT32 PchSlpSusMinAssert: 4

*SLP_SUS Minimum Assertion Width Policy. Refer to PCH_SLP_SUS_MIN_ASSERT for each value. Default is **PchSlpSus4s**.*

- UINT32 PchSlpAMinAssert: 4

  *SLP_A Minimum Assertion Width Policy. Refer to PCH_SLP_A_MIN_ASSERT for each value. Default is **PchSlpA2s**.*
- UINT32 SlpStrchSusUp: 1

  This member describes whether or not the LPC ClockRun feature of PCH should be enabled.
- UINT32 SlpLanLowDc: 1

  Enable/Disable SLP_LAN# Low on DC Power.
- UINT32 PwrBtnOverridePeriod: 3

  PCH power button override period.
- UINT32 DisableEnergyReport: 1

  ***(Test)** Disable/Enable PCH to CPU enery report feature.*
- UINT32 DisableDsxAcPresentPulldown: 1

  When set to Disable, PCH will internal pull down AC_PRESENT in deep SX and during G3 exit.
- UINT32 DisableNativePowerButton: 1

  Power button native mode disable.
- UINT32 MeWakeSts: 1

  *Clear the ME_WAKE_STS bit in the Power and Reset Status (PRSTS) register. 0: Disable; **1: Enable**.*
- UINT32 WolOvrWkSts: 1

  *Clear the WOL_OVR_WK_STS bit in the Power and Reset Status (PRSTS) register. 0: Disable; **1: Enable**.*
- UINT32 PsOnEnable: 1

  Decide if PS_ON is to be enabled.
- UINT32 CppmFaEn: 1

  Decide if CPPM Force Alignment is to be enabled.
- UINT32 CpuC10GatePinEnable: 1

  Enable/Disable platform support for CPU_C10_GATE# pin to control gating of CPU VccIO and VccSTG rails instead of SLP_S0# pin.
- UINT32 ModPhySusPgEnable: 1

  Enable/Disable ModPHY SUS Power Domain Dynamic Gating.
- UINT32 Usb2PhySusPgEnable: 1

  ***(Test)** This policy option enables USB2 PHY SUS Well Power Gating functionality.*
- UINT32 OsIdleEnable: 1

  Enable Os Idle Mode.
- UINT32 V1p05PhyExtFetControlEn: 1

  *Enable control using EXT_PWR_GATE# pin of external FET to power gate v1p05-PHY **0: Disable**; 1: Enable.*
- UINT32 V1p05IsExtFetControlEn: 1

  *Enable control using EXT_PWR_GATE2# pin of external FET to power gate v1p05-IS supply **0: Disable**; 1: Enable.*
- UINT32 S0ixAutoDemotion: 1

  Enable/Disable the Low Power Mode Host S0ix Auto-Demotion feature.
- UINT32 LatchEventsC10Exit: 1

  Enable/Disable Latch Events C10 Exit.
- UINT32 PmcWdtTimerEn: 1

  Enable/Disable PMC Watch Dog Timer.
- UINT8 PchPwrCycDur

  Reset Power Cycle Duration could be customized in the unit of second.
- UINT8 PciePllSsc

  Specifies the Pcie Pll Spread Spectrum Percentage The value of this policy is in 1/10th percent units.
- UINT8 C10DynamicThresholdAdjustment

  Tells BIOS to enable C10 dynamic threshold adjustment mode.
- UINT8 Rsvd0 [1]

  Reserved bytes.

- PMC_LPM_S0IX_SUB_STATE_EN LpmS0ixSubStateEnable

  (Test) *Low Power Mode Enable/Disable config.*
- UINT8 GlobalResetMasksOverride

  Set true to enable override of Global Reset Event/Trigger masks.
- UINT8 Rsvd1 [2]

  Reserved bytes.
- UINT32 ThermTimerDelay: 10

  PMC Therm Timer Delay Register (THERM_TIMER_DELAY).
- UINT32 PmErDebugMode: 1

  Energy Reporting Debug Mode.

12.146.1 Detailed Description

The PCH_PM_CONFIG block describes expected miscellaneous power management settings.

The PowerResetStatusClear field would clear the Power/Reset status bits, please set the bits if you want PCH Init driver to clear it, if you want to check the status later then clear the bits.

Revision 1:

- Initial version.

Definition at line 211 of file PmConfig.h.

12.146.2 Member Data Documentation

12.146.2.1 C10DynamicThresholdAdjustment

```
UINT8 PCH_PM_CONFIG::C10DynamicThresholdAdjustment
```

Tells BIOS to enable C10 dynamic threshold adjustment mode.

BIOS will only attemt to enable it on PCH SKUs which support it.

Definition at line 398 of file PmConfig.h.

12.146.2.2 CppmFaEn

```
UINT32 PCH_PM_CONFIG::CppmFaEn
```

Decide if CPPM Force Alignment is to be enabled.

When enabled, PMC allows stalling of the backbone or blocking the DMI transmit arbiter. 0: Disable; **1: Enable**

Definition at line 296 of file PmConfig.h.

12.146.2.3 CpuC10GatePinEnable

`UINT32 PCH_PM_CONFIG::CpuC10GatePinEnable`

Enable/Disable platform support for CPU_C10_GATE# pin to control gating of CPU VccIO and VccSTG rails instead of SLP_S0# pin.

This policy needs to be set if board design includes support for CPU_C10_GATE# pin. 0: Disable; **1: Enable**

Definition at line 303 of file PmConfig.h.

12.146.2.4 DisableDsxAcPresentPulldown

`UINT32 PCH_PM_CONFIG::DisableDsxAcPresentPulldown`

When set to Disable, PCH will internal pull down AC_PRESENT in deep SX and during G3 exit.

When set to Enable, PCH will not pull down AC_PRESENT. This setting is ignored when DeepSx is not supported. Default is **0:Disable**

Definition at line 258 of file PmConfig.h.

12.146.2.5 DisableEnergyReport

`UINT32 PCH_PM_CONFIG::DisableEnergyReport`

(Test) Disable/Enable PCH to CPU enery report feature.

0: Disable; 1: Enable. Enery Report is must have feature. Wihtout Energy Report, the performance report by workloads/benchmarks will be unrealistic because PCH's energy is not being accounted in power/performance management algorithm. If for some reason PCH energy report is too high, which forces CPU to try to reduce its power by throttling, then it could try to disable Energy Report to do first debug. This might be due to energy scaling factors are not correct or the LPM settings are not kicking in.

Definition at line 251 of file PmConfig.h.

12.146.2.6 DisableNativePowerButton

`UINT32 PCH_PM_CONFIG::DisableNativePowerButton`

Power button native mode disable.

While FALSE, the PMC's power button logic will act upon the input value from the GPIO unit, as normal. While TRUE, this will result in the PMC logic constantly seeing the power button as de-asserted. **Default is FALSE.**

Definition at line 265 of file PmConfig.h.

12.146.2.7 GlobalResetMasksOverride

```
UINT8 PCH_PM_CONFIG::GlobalResetMasksOverride
```

Set true to enable override of Global Reset Event/Trigger masks.

Values from GlobalResetTriggerMask and GlobalResetEventMask will be used as override value. **0: Disable**, 1: Enable

Definition at line 437 of file PmConfig.h.

12.146.2.8 LatchEventsC10Exit

```
UINT32 PCH_PM_CONFIG::LatchEventsC10Exit
```

Enable/Disable Latch Events C10 Exit.

When this bit is set to 1, SLP_S0# entry events in SLP_S0_DEBUG_REGx registers are captured on C10 exit (instead of C10 entry which is default) **0: Disable**; 1: Enable.

Definition at line 352 of file PmConfig.h.

12.146.2.9 LpmS0ixSubStateEnable

```
PMC_LPM_S0IX_SUB_STATE_EN PCH_PM_CONFIG::LpmS0ixSubStateEnable
```

(Test) Low Power Mode Enable/Disable config.

Configure if respective S0i2/3 sub-states are to be supported by the platform. By default all sub-states are enabled but for test purpose respective states can be disabled. **Default is 0xFF**

Definition at line 408 of file PmConfig.h.

12.146.2.10 ModPhySusPgEnable

```
UINT32 PCH_PM_CONFIG::ModPhySusPgEnable
```

Enable/Disable ModPHY SUS Power Domain Dynamic Gating.

EXT_PWR_GATE# signal (if supported on platform) can be used to control external FET for power gating ModPHY

Note

: This setting is not supported and ignored on PCH-H 0: Disable; **1: Enable**.

Definition at line 312 of file PmConfig.h.

12.146.2.11 OsIdleEnable

`UINT32 PCH_PM_CONFIG::OsIdleEnable`

Enable Os Idle Mode.

0: Disable; **1: Enable**.

Definition at line 325 of file PmConfig.h.

12.146.2.12 PchPwrCycDur

`UINT8 PCH_PM_CONFIG::PchPwrCycDur`

Reset Power Cycle Duration could be customized in the unit of second.

Please refer to EDS for all support settings. PCH HW default is 4 seconds, and range is 1∼4 seconds, where **0 is default**, 1 is 1 second, 2 is 2 seconds, ... 4 is 4 seconds. And make sure the setting correct, which never less than the following register.

- GEN_PMCON_B.SLP_S3_MIN_ASST_WDTH
- GEN_PMCON_B.SLP_S4_MIN_ASST_WDTH
- PWRM_CFG.SLP_A_MIN_ASST_WDTH
- PWRM_CFG.SLP_LAN_MIN_ASST_WDTH

Definition at line 385 of file PmConfig.h.

12.146.2.13 PciePllSsc

`UINT8 PCH_PM_CONFIG::PciePllSsc`

Specifies the Pcie Pll Spread Spectrum Percentage The value of this policy is in 1/10th percent units.

Valid spread range is 0-20. A value of 0xFF is reserved for AUTO. A value of 0 is SSC of 0.0%. A value of 20 is SSC of 2.0% The default is **0xFF: AUTO - No BIOS override**.

Definition at line 393 of file PmConfig.h.

12.146.2.14 PmcWdtTimerEn

`UINT32 PCH_PM_CONFIG::PmcWdtTimerEn`

Enable/Disable PMC Watch Dog Timer.

When this bit is set to 1, timers are enabled that can trigger a system reset. 0: Disable; **1: Enable**.

Definition at line 358 of file PmConfig.h.

12.146.2.15 PsOnEnable

`UINT32 PCH_PM_CONFIG::PsOnEnable`

Decide if PS_ON is to be enabled.

This is available on desktop only. PS_ON is a new C10 state from the CPU on desktop SKUs that enables a lower power target that will be required by the California Energy Commission (CEC). When FALSE, PS_ON is to be disabled.} **0: Disable**; 1: Enable.

Definition at line 290 of file PmConfig.h.

12.146.2.16 PwrBtnOverridePeriod

`UINT32 PCH_PM_CONFIG::PwrBtnOverridePeriod`

PCH power button override period.

000b-4s, 001b-6s, 010b-8s, 011b-10s, 100b-12s, 101b-14s **Default is 0: 4s**

Definition at line 239 of file PmConfig.h.

12.146.2.17 S0ixAutoDemotion

`UINT32 PCH_PM_CONFIG::S0ixAutoDemotion`

Enable/Disable the Low Power Mode Host S0ix Auto-Demotion feature.

This feature enables the PMC to autonomously manage the deepest allowed S0ix substate to combat thrashing between power management states. 0: Disable; **1: Enable**.

Definition at line 345 of file PmConfig.h.

12.146.2.18 SlpLanLowDc

`UINT32 PCH_PM_CONFIG::SlpLanLowDc`

Enable/Disable SLP_LAN# Low on DC Power.

0: Disable; **1: Enable**. Configure On DC PHY Power Diable according to policy SlpLanLowDc. When this is enabled, SLP_LAN# will be driven low when ACPRESENT is low. This indicates that LAN PHY should be powered off on battery mode. This will override the DC_PP_DIS setting by WolEnableOverride.

Definition at line 233 of file PmConfig.h.

12.146.2.19 SlpStrchSusUp

```
UINT32 PCH_PM_CONFIG::SlpStrchSusUp
```

This member describes whether or not the LPC ClockRun feature of PCH should be enabled.

0: Disable; 1: Enable **0: Disable**; 1: Enable SLP_X Stretching After SUS Well Power Up

Definition at line 225 of file PmConfig.h.

12.146.2.20 ThermTimerDelay

```
UINT32 PCH_PM_CONFIG::ThermTimerDelay
```

PMC Therm Timer Delay Register (THERM_TIMER_DELAY).

Unit in us(microseconds). Adding delay time between CPU thermal trip propagating through PCH and PCH generating a Global Reset **Default is 500**

Definition at line 453 of file PmConfig.h.

12.146.2.21 Usb2PhySusPgEnable

```
UINT32 PCH_PM_CONFIG::Usb2PhySusPgEnable
```

(Test) This policy option enables USB2 PHY SUS Well Power Gating functionality.

Note

: This setting is not supported and ignored on PCH-H 0: disable USB2 PHY SUS Well Power Gating **1: enable USB2 PHY SUS Well Power Gating**

Definition at line 320 of file PmConfig.h.

The documentation for this struct was generated from the following file:

- PmConfig.h

12.147 PCH_SATA_PORT_CONFIG Struct Reference

This structure configures the features, property, and capability for each SATA port.

```
#include <SataConfig.h>
```

Public Attributes

- UINT32 Enable: 1

  Enable SATA port.
- UINT32 HotPlug: 1

  ***0: Disable**; 1: Enable*
- UINT32 InterlockSw: 1

  ***0: Disable**; 1: Enable*
- UINT32 External: 1

  ***0: Disable**; 1: Enable*
- UINT32 SpinUp: 1

  ***0: Disable**; 1: Enable the COMRESET initialization Sequence to the device*
- UINT32 SolidStateDrive: 1

  ***0: HDD**; 1: SSD*
- UINT32 DevSlp: 1

  ***0: Disable**; 1: Enable DEVSLP on the port*
- UINT32 EnableDitoConfig: 1

  ***0: Disable**; 1: Enable DEVSLP Idle Timeout settings (DmVal, DitoVal)*
- UINT32 DmVal: 4

  *DITO multiplier. Default is **15**.*
- UINT32 DitoVal: 10

  *DEVSLP Idle Timeout (DITO), Default is **625**.*
- UINT32 ZpOdd: 1

  *Support zero power ODD **0: Disable**, 1: Enable.*
- UINT32 DevSlpResetConfig: 4

  *0: Hardware default; **0x01: GpioResumeReset**; 0x03: GpioHostDeepReset; 0x05: GpioPlatformReset; 0x07↩ : GpioDswReset*
- UINT32 RxPolarity: 1

  ***0: Disable**; 1: Enable; Rx Polarity*
- UINT32 RsvdBits0: 4

  Reserved fields for future expansion w/o protocol change.

12.147.1 Detailed Description

This structure configures the features, property, and capability for each SATA port.

Definition at line 60 of file SataConfig.h.

12.147.2 Member Data Documentation

12.147.2.1 Enable

```
UINT32 PCH_SATA_PORT_CONFIG::Enable
```

Enable SATA port.

It is highly recommended to disable unused ports for power savings 0: Disable; **1: Enable**

Definition at line 65 of file SataConfig.h.

12.147.2.2 ZpOdd

```
UINT32 PCH_SATA_PORT_CONFIG::ZpOdd
```

Support zero power ODD **0: Disable**, 1: Enable.

This is also used to disable ModPHY dynamic power gate.

Definition at line 79 of file SataConfig.h.

The documentation for this struct was generated from the following file:

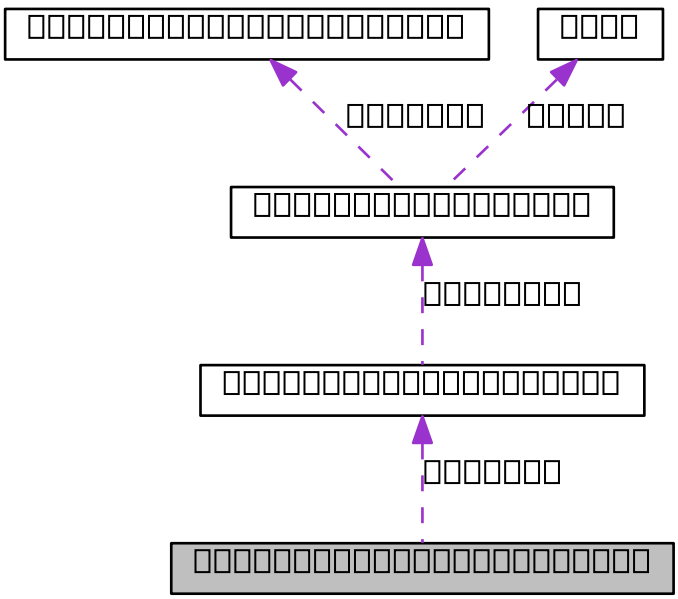
- SataConfig.h

12.148 PCH_SMBUS_PREMEM_CONFIG Struct Reference

The SMBUS_CONFIG block lists the reserved addresses for non-ARP capable devices in the platform.

```
#include <SmbusConfig.h>
```

Collaboration diagram for PCH_SMBUS_PREMEM_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Revision 1: Init version.
- UINT32 Enable: 1

  This member describes whether or not the SMBus controller of PCH should be enabled.
- UINT32 ArpEnable: 1

  *Enable SMBus ARP support, **0: Disable**; 1: Enable.*
- UINT32 DynamicPowerGating: 1

  ***(Test) Disable** or Enable Smbus dynamic power gating.*
- UINT32 SpdWriteDisable: 1

  (Test)** SPD Write Disable, 0: leave SPD Write Disable bit; **1: set SPD Write Disable bit.
- UINT32 SmbAlertEnable: 1

  *Enable SMBus Alert pin (SMBALERT#). 0: **Disabled, 1: Enabled.***
- UINT32 RsvdBits0: 27

  Reserved bits.
- UINT16 SmbusIoBase

  *SMBUS Base Address (IO space). Default is **0xEFA0**.*
- UINT8 Rsvd0

  Reserved bytes.
- UINT8 NumRsvdSmbusAddresses

  The number of elements in the RsvdSmbusAddressTable.
- UINT8 RsvdSmbusAddressTable [PCH_MAX_SMBUS_RESERVED_ADDRESS]

  Array of addresses reserved for non-ARP-capable SMBus devices.

12.148.1 Detailed Description

The SMBUS_CONFIG block lists the reserved addresses for non-ARP capable devices in the platform.

Definition at line 36 of file SmbusConfig.h.

12.148.2 Member Data Documentation

12.148.2.1 Enable

`UINT32 PCH_SMBUS_PREMEM_CONFIG::Enable`

This member describes whether or not the SMBus controller of PCH should be enabled.

0: Disable; **1: Enable**.

Definition at line 45 of file SmbusConfig.h.

12.148.2.2 Header

```
CONFIG_BLOCK_HEADER PCH_SMBUS_PREMEM_CONFIG::Header
```

Revision 1: Init version.

Config Block Header

Definition at line 40 of file SmbusConfig.h.

12.148.2.3 SpdWriteDisable

```
UINT32 PCH_SMBUS_PREMEM_CONFIG::SpdWriteDisable
```

(Test) SPD Write Disable, 0: leave SPD Write Disable bit; **1: set SPD Write Disable bit.**

For security recommendations, SPD write disable bit must be set.

Definition at line 52 of file SmbusConfig.h.

The documentation for this struct was generated from the following file:

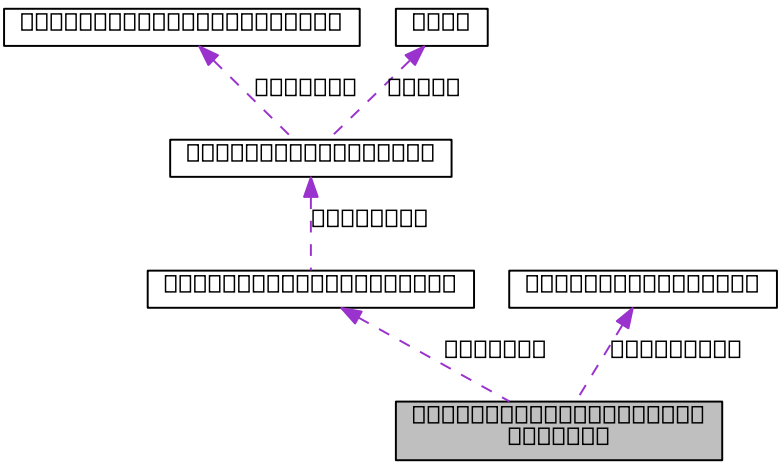
- SmbusConfig.h

12.149 PCH_TRACE_HUB_PREMEM_CONFIG Struct Reference

PCH Trace Hub PreMem Configuration Contains Trace Hub settings for PCH side tracing **Revision 1**: - Initial version.

```
#include <TraceHubConfig.h>
```

Collaboration diagram for PCH_TRACE_HUB_PREMEM_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- TRACE_HUB_CONFIG TraceHub

  Trace Hub Config.

12.149.1 Detailed Description

PCH Trace Hub PreMem Configuration Contains Trace Hub settings for PCH side tracing **Revision 1**: - Initial version.

Definition at line 106 of file TraceHubConfig.h.

The documentation for this struct was generated from the following file:

- TraceHubConfig.h

12.150 PCH_WAKE_CONFIG Struct Reference

This structure allows to customize PCH wake up capability from S5 or DeepSx by WOL, LAN, PCIE wake events.

`#include <PmConfig.h>`

Public Attributes

- UINT32 PmeB0S5Dis: 1

  Corresponds to the PME_B0_S5_DIS bit in the General PM Configuration B (GEN_PMCON_B) register.
- UINT32 WolEnableOverride: 1

  *Corresponds to the "WOL Enable Override" bit in the General PM Configuration B (GEN_PMCON_B) register. 0: Disable; **1: Enable**.*
- UINT32 PcieWakeFromDeepSx: 1

  *Determine if enable PCIe to wake from deep Sx. **0: Disable**; 1: Enable.*
- UINT32 WoWlanEnable: 1

  *Determine if WLAN wake from Sx, corresponds to the "HOST_WLAN_PP_EN" bit in the PWRM_CFG3 register. **0: Disable**; 1: Enable.*
- UINT32 WoWlanDeepSxEnable: 1

  *Determine if WLAN wake from DeepSx, corresponds to the "DSX_WLAN_PP_EN" bit in the PWRM_CFG3 register. **0: Disable**; 1: Enable.*
- UINT32 LanWakeFromDeepSx: 1

  *Determine if enable LAN to wake from deep Sx. 0: Disable; **1: Enable**.*

12.150.1 Detailed Description

This structure allows to customize PCH wake up capability from S5 or DeepSx by WOL, LAN, PCIE wake events.

Definition at line 37 of file PmConfig.h.

12.150.2 Member Data Documentation

12.150.2.1 PmeB0S5Dis

```
UINT32 PCH_WAKE_CONFIG::PmeB0S5Dis
```

Corresponds to the PME_B0_S5_DIS bit in the General PM Configuration B (GEN_PMCON_B) register.

When set to 1, this bit blocks wake events from PME_B0_STS in S5, regardless of the state of PME_B0_EN. When cleared (default), wake events from PME_B0_STS are allowed in S5 if PME_B0_EN = 1. **0: Disable**; 1: Enable.

Definition at line 43 of file PmConfig.h.

The documentation for this struct was generated from the following file:

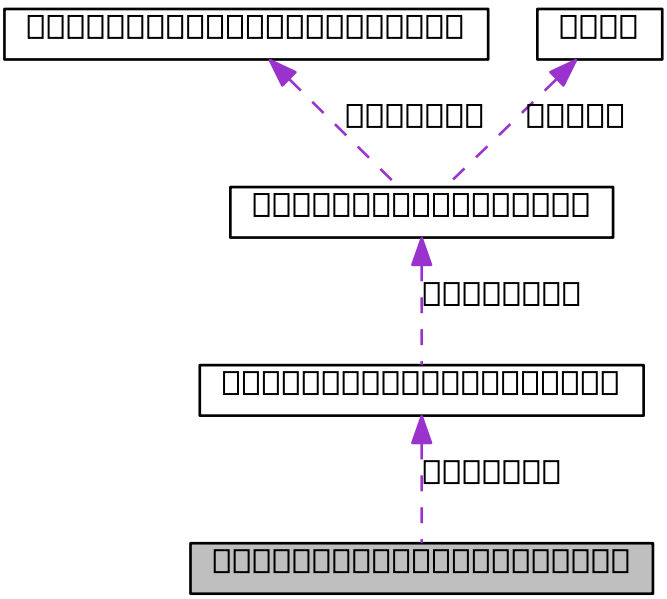
- PmConfig.h

12.151 PCH_WDT_PREMEM_CONFIG Struct Reference

This policy clears status bits and disable watchdog, then lock the WDT registers.

```
#include <WatchDogConfig.h>
```

Collaboration diagram for PCH_WDT_PREMEM_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT32 DisableAndLock: 1

  ***(Test)** Set 1 to clear WDT status, then disable and lock WDT registers. **0: Disable**; 1: Enable.*

12.151.1 Detailed Description

This policy clears status bits and disable watchdog, then lock the WDT registers.

while WDT is designed to be disabled and locked by Policy, bios should not enable WDT by WDT PPI. In such case, bios shows the warning message but not disable and lock WDT register to make sure WDT event trigger correctly.

Definition at line 37 of file WatchDogConfig.h.

The documentation for this struct was generated from the following file:

- WatchDogConfig.h

12.152 PCIE_COMMON_CONFIG Struct Reference

PCIe Common Config.

```
#include <PcieConfig.h>
```

Public Attributes

- UINT32 RpFunctionSwap: 1

  RpFunctionSwap allows BIOS to use root port function number swapping when root port of function 0 is disabled.
- UINT32 ComplianceTestMode: 1

  Compliance Test Mode shall be enabled when using Compliance Load Board.
- UINT32 RsvdBits0: 30

  Reserved bits.
- UINT8 EnablePort8xhDecode

  ***(Test)** This member describes whether PCIE root port Port 8xh Decode is enabled.*

12.152.1 Detailed Description

PCIe Common Config.

Note

This structure will be expanded to hold all common PCIe policies between SA and PCH

Definition at line 299 of file PcieConfig.h.

12.152.2 Member Data Documentation

12.152.2.1 ComplianceTestMode

`UINT32 PCIE_COMMON_CONFIG::ComplianceTestMode`

Compliance Test Mode shall be enabled when using Compliance Load Board.

0: Disable, 1: Enable

Definition at line 320 of file PcieConfig.h.

12.152.2.2 EnablePort8xhDecode

`UINT8 PCIE_COMMON_CONFIG::EnablePort8xhDecode`

(Test) This member describes whether PCIE root port Port 8xh Decode is enabled.

0: Disable; 1: Enable.

Definition at line 325 of file PcieConfig.h.

12.152.2.3 RpFunctionSwap

`UINT32 PCIE_COMMON_CONFIG::RpFunctionSwap`

RpFunctionSwap allows BIOS to use root port function number swapping when root port of function 0 is disabled.

A PCIE device can have higher functions only when Function0 exists. To satisfy this requirement, BIOS will always enable Function0 of a device that contains more than 0 enabled root ports.

- **Enabled: One of enabled root ports get assigned to Function0.** This offers no guarantee that any particular root port will be available at a specific DevNr:FuncNr location
- Disabled: Root port that corresponds to Function0 will be kept visible even though it might be not used. That way rootport - to - DevNr:FuncNr assignment is constant. This option will impact ports 1, 9, 17. NOTE: This option will not work if ports 1, 9, 17 are fused or configured for RST PCIe storage or disabled through policy In other words, it only affects ports that would become hidden because they have no device connected. NOTE: Disabling function swap may have adverse impact on power management. This option should ONLY be used when each one of root ports 1, 9, 17:
  - is configured as PCIe and has correctly configured ClkReq signal, or
  - does not own any mPhy lanes (they are configured as SATA or USB)

Definition at line 315 of file PcieConfig.h.

The documentation for this struct was generated from the following file:

- PcieConfig.h

12.153 PCIE_DEVICE_OVERRIDE Struct Reference

PCIe device table entry entry.

`#include <PcieConfig.h>`

Public Attributes

- UINT16 VendorId

  The vendor Id of Pci Express card ASPM setting override, 0xFFFF means any Vendor ID.
- UINT16 DeviceId

  The Device Id of Pci Express card ASPM setting override, 0xFFFF means any Device ID.
- UINT8 RevId

  The Rev Id of Pci Express card ASPM setting override, 0xFF means all steppings.
- UINT8 BaseClassCode

  The Base Class Code of Pci Express card ASPM setting override, 0xFF means all base class.
- UINT8 SubClassCode

  The Sub Class Code of Pci Express card ASPM setting override, 0xFF means all sub class.
- UINT8 EndPointAspm

  Override device ASPM (see: PCH_PCIE_ASPM_CONTROL) Bit 1 must be set in OverrideConfig for this field to take effect.
- UINT16 OverrideConfig

  The override config bitmap (see: PCH_PCIE_OVERRIDE_CONFIG).
- UINT16 L1SubstatesCapOffset

  The L1Substates Capability Offset Override.
- UINT8 L1SubstatesCapMask

  L1 Substate Capability Mask.
- UINT8 L1sCommonModeRestoreTime

  L1 Substate Port Common Mode Restore Time Override.
- UINT8 L1sTpowerOnScale

  L1 Substate Port Tpower_on Scale Override.
- UINT8 L1sTpowerOnValue

  L1 Substate Port Tpower_on Value Override.
- UINT16 SnoopLatency

  SnoopLatency bit definition Note: All Reserved bits must be set to 0.
- UINT16 NonSnoopLatency

  NonSnoopLatency bit definition Note: All Reserved bits must be set to 0.
- UINT8 ForceLtrOverride

  Forces LTR override to be permanent The default way LTR override works is: rootport uses LTR override values provided by BIOS until connected device sends an LTR message, then it will use values from the message This settings allows force override of LTR mechanism.

12.153.1 Detailed Description

PCIe device table entry entry.

The PCIe device table is being used to override PCIe device ASPM settings. To take effect table consisting of such entries must be instelled as PPI on gPchPcieDeviceTablePpiGuid. Last entry VendorId must be 0.

Definition at line 340 of file PcieConfig.h.

12.153.2 Member Data Documentation

12.153.2.1 ForceLtrOverride

```
UINT8 PCIE_DEVICE_OVERRIDE::ForceLtrOverride
```

Forces LTR override to be permanent The default way LTR override works is: rootport uses LTR override values provided by BIOS until connected device sends an LTR message, then it will use values from the message This settings allows force override of LTR mechanism.

If it's enabled, then: rootport will use LTR override values provided by BIOS forever; LTR messages sent from connected device will be ignored

Definition at line 434 of file PcieConfig.h.

12.153.2.2 L1sCommonModeRestoreTime

```
UINT8 PCIE_DEVICE_OVERRIDE::L1sCommonModeRestoreTime
```

L1 Substate Port Common Mode Restore Time Override.

(applicable if bit 2 is set in OverrideConfig) L1sCommonModeRestoreTime and L1sTpowerOnScale can have a valid value of 0, but not the L1sTpowerOnValue. If L1sTpowerOnValue is zero, all L1sCommonModeRestoreTime, L1sTpowerOnScale, and L1sTpowerOnValue are ignored, and only L1SubstatesCapOffset is override.

Definition at line 366 of file PcieConfig.h.

12.153.2.3 L1sTpowerOnScale

```
UINT8 PCIE_DEVICE_OVERRIDE::L1sTpowerOnScale
```

L1 Substate Port Tpower_on Scale Override.

(applicable if bit 2 is set in OverrideConfig) L1sCommonModeRestoreTime and L1sTpowerOnScale can have a valid value of 0, but not the L1sTpowerOnValue. If L1sTpowerOnValue is zero, all L1sCommonModeRestoreTime, L1sTpowerOnScale, and L1sTpowerOnValue are ignored, and only L1SubstatesCapOffset is override.

Definition at line 373 of file PcieConfig.h.

12.153.2.4 L1sTpowerOnValue

`UINT8 PCIE_DEVICE_OVERRIDE::L1sTpowerOnValue`

L1 Substate Port Tpower_on Value Override.

(applicable if bit 2 is set in OverrideConfig) L1sCommonModeRestoreTime and L1sTpowerOnScale can have a valid value of 0, but not the L1sTpowerOnValue. If L1sTpowerOnValue is zero, all L1sCommonModeRestoreTime, L1sTpowerOnScale, and L1sTpowerOnValue are ignored, and only L1SubstatesCapOffset is override.

Definition at line 380 of file PcieConfig.h.

12.153.2.5 L1SubstatesCapMask

`UINT8 PCIE_DEVICE_OVERRIDE::L1SubstatesCapMask`

L1 Substate Capability Mask.

(applicable if bit 2 is set in OverrideConfig) Set to zero then the L1 Substate Capability [3:0] is ignored, and only L1s values are override. Only bit [3:0] are applicable. Other bits are ignored.

Definition at line 359 of file PcieConfig.h.

12.153.2.6 L1SubstatesCapOffset

`UINT16 PCIE_DEVICE_OVERRIDE::L1SubstatesCapOffset`

The L1Substates Capability Offset Override.

(applicable if bit 2 is set in OverrideConfig) This field can be zero if only the L1 Substate value is going to be override.

Definition at line 353 of file PcieConfig.h.

12.153.2.7 NonSnoopLatency

`UINT16 PCIE_DEVICE_OVERRIDE::NonSnoopLatency`

NonSnoopLatency bit definition Note: All Reserved bits must be set to 0.

BIT[15] - When set to 1b, indicates that the values in bits 9:0 are valid When clear values in bits 9:0 will be ignored BITS[14:13] - Reserved BITS[12:10] - Value in bits 9:0 will be multiplied with the scale in these bits 000b - 1 ns 001b - 32 ns 010b - 1024 ns 011b - 32,768 ns 100b - 1,048,576 ns 101b - 33,554,432 ns 110b - Reserved 111b - Reserved BITS[9:0] - Non Snoop Latency Value. The value in these bits will be multiplied with the scale in bits 12:10

This field takes effect only if bit 3 is set in OverrideConfig.

Definition at line 425 of file PcieConfig.h.

12.153.2.8 SnoopLatency

```
UINT16 PCIE_DEVICE_OVERRIDE::SnoopLatency
```

SnoopLatency bit definition Note: All Reserved bits must be set to 0.

BIT[15] - When set to 1b, indicates that the values in bits 9:0 are valid When clear values in bits 9:0 will be ignored BITS[14:13] - Reserved BITS[12:10] - Value in bits 9:0 will be multiplied with the scale in these bits 000b - 1 ns 001b - 32 ns 010b - 1024 ns 011b - 32,768 ns 100b - 1,048,576 ns 101b - 33,554,432 ns 110b - Reserved 111b - Reserved BITS[9:0] - Snoop Latency Value. The value in these bits will be multiplied with the scale in bits 12:10

This field takes effect only if bit 3 is set in OverrideConfig.

Definition at line 403 of file PcieConfig.h.

The documentation for this struct was generated from the following file:

- PcieConfig.h

12.154 PCIE_EQ_PARAM Struct Reference

Represent lane specific PCIe Gen3 equalization parameters.

```
#include <PcieConfig.h>
```

Public Attributes

- UINT8 Cm
  Coefficient C-1.
- UINT8 Cp
  Coefficient C+1.
- UINT8 Rsvd0 [2]
  Reserved bytes.

12.154.1 Detailed Description

Represent lane specific PCIe Gen3 equalization parameters.

Definition at line 54 of file PcieConfig.h.

The documentation for this struct was generated from the following file:

- PcieConfig.h

12.155 PCIE_LINK_EQ_PLATFORM_SETTINGS Struct Reference

PCIe Link EQ Platform Settings.

```
#include <PcieConfig.h>
```

Public Attributes

- UINT8 PcieLinkEqMethod

  Tells BIOS which link EQ method should be used for this port. Please refer to PCIE_LINK_EQ_METHOD for details of supported methods. Default: PcieLinkHardwareEq.
- UINT8 PcieLinkEqMode

  Tells BIOS which mode should be used for PCIe link EQ. Please refer to PCIE_LINK_EQ_MODE for details of supported modes. Default: depends on SoC.
- UINT8 LocalTxOverrideEn

  Specifies if BIOS should perform local transmitter override during phase 2 of EQ process.
- UINT8 DefaultPresetEnable [PCIE_LINK_EQ_PRESETS_MAX]

  Specifies if default settings should remain in use.
- UINT8 HapcsSearchEnable

  Specifies setting for Hardware Autonomous Preset/Coefficient Search mechanism.
- UINT32 Ph2LocalTxOverridePreset

  Specifies the preset that should be used during local transmitter override during phase 2 of EQ process.
- UINT32 PCETTimer

  PCET Timer value for single PCIe speed.
- UINT8 RemotePresetCoeffoverride

  Remote Transmitter Preset Coefficient Override for single PCIe speed.
- UINT8 EqPh3Bypass

  PCIe Equalization Phase 3 Enable Control.
- UINT8 EqPh23Bypass

  PCIe Equalization Phase 2-3 Enable Control.
- UINT8 TsLockTimer

  8.0GT/s Training Sequence Wait Latency For Presets / Coefficients Evaluation - Gen3 TS Lock Timer
- UINT8 Ph3NoOfPresetOrCoeff

  Tells BIOS how many presets/coefficients should be used during link EQ.
- PCIE_LINK_EQ_COEFFICIENTS Ph3CoefficientsList [PCIE_LINK_EQ_COEFFICIENTS_MAX]

  List of the PCIe coefficients to be used during equalization process. Only valid if PcieLinkEqMode is PcieLinkEq↩ CoefficientMode.
- UINT32 Ph3PresetList [PCIE_LINK_EQ_PRESETS_MAX]

  List of the PCIe preset values to be used during equalization process. Only valid if PcieLinkEqMode is PcieLinkEq↩ PresetMode.
- UINT32 EndPointPresetList [PCIE_LINK_EQ_PRESETS_MAX]

  List of the PCIe preset values to be used during equalization process. Only valid if PcieLinkEqMode is PcieLinkEq↩ PresetMode.
- UINT32 HintList [PCIE_LINK_EQ_PRESETS_MAX]

  List of the Hint values to be used during equalization process for preset calculation.
- UINT32 Ph1DpTxPreset

  Specifies the value of the downstream port transmitter preset to be used during phase 1 of the equalization process. Will be applied to all lanes.
- UINT32 Ph1UpTxPreset

  Specifies the value of the upstream port transmitter preset to be used during phase 1 of the equalization process. Will be applied to all lanes.

12.155.1 Detailed Description

PCIe Link EQ Platform Settings.

Definition at line 111 of file PcieConfig.h.

12.155.2 Member Data Documentation

12.155.2.1 EqPh23Bypass

`UINT8 PCIE_LINK_EQ_PLATFORM_SETTINGS::EqPh23Bypass`

PCIe Equalization Phase 2-3 Enable Control.

- **Disabled** (0x0) : Disable Phase 2 - Phase 3 (Default)
- Enabled (0x1) : Enable Phase 2 - Phase 3

Definition at line 142 of file PcieConfig.h.

12.155.2.2 EqPh3Bypass

`UINT8 PCIE_LINK_EQ_PLATFORM_SETTINGS::EqPh3Bypass`

PCIe Equalization Phase 3 Enable Control.

- **Disabled** (0x0) : Disable phase 3 (Default)
  - Enabled (0x1) : Enable phase 3

Definition at line 136 of file PcieConfig.h.

12.155.2.3 LocalTxOverrideEn

`UINT8 PCIE_LINK_EQ_PLATFORM_SETTINGS::LocalTxOverrideEn`

Specifies if BIOS should perform local transmitter override during phase 2 of EQ process.

If enabled value in Ph2LocalTxOverridePreset must be valid. **0: Disabled**; 1: Enabled

Definition at line 119 of file PcieConfig.h.

12.155.2.4 Ph2LocalTxOverridePreset

```
UINT32 PCIE_LINK_EQ_PLATFORM_SETTINGS::Ph2LocalTxOverridePreset
```

Specifies the preset that should be used during local transmitter override during phase 2 of EQ process.

Used only if LocalTxOverrideEn is TRUE. Will be applied to all PCIe lanes of the root port. Valid up to the PCIE_↩ LINK_EQ_PRESET_MAX value. **Default: 0**< >

Definition at line 128 of file PcieConfig.h.

12.155.2.5 Ph3NoOfPresetOrCoeff

```
UINT8 PCIE_LINK_EQ_PLATFORM_SETTINGS::Ph3NoOfPresetOrCoeff
```

Tells BIOS how many presets/coefficients should be used during link EQ.

Entries in the Ph3CoefficientsList or Ph3PresetList(depending on chosen mode) need to be valid up to the number specified in this field.

Definition at line 149 of file PcieConfig.h.

The documentation for this struct was generated from the following file:

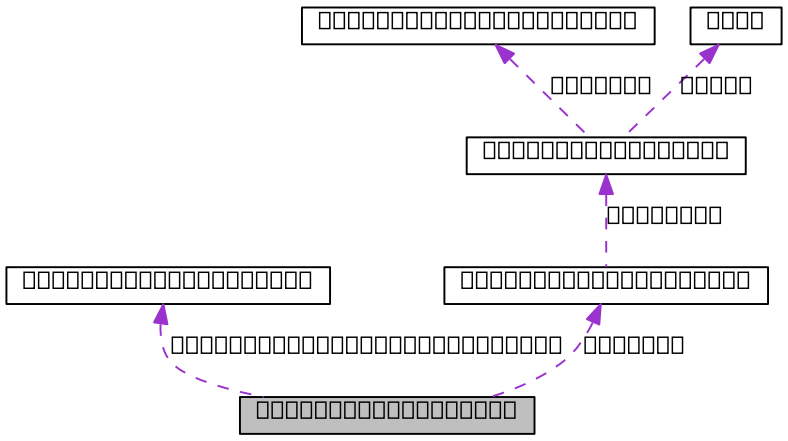
- PcieConfig.h

12.156 PCIE_RP_DXE_CONFIG Struct Reference

The PCIE_RP_DXE_CONFIG block describes the expected configuration of the PCH PCI Express controllers in DXE phase.

```
#include <PchPcieRpConfig.h>
```

Collaboration diagram for PCIE_RP_DXE_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT8 RsvdBytes323 [4]

  Reserved.
- PCIE_DEVICE_OVERRIDE * PcieDeviceOverrideTablePtr

  PCIe device override table The PCIe device table is being used to override PCIe device ASPM settings.

12.156.1 Detailed Description

The PCIE_RP_DXE_CONFIG block describes the expected configuration of the PCH PCI Express controllers in DXE phase.

Revision 1:

- Init version

Definition at line 319 of file PchPcieRpConfig.h.

12.156.2 Member Data Documentation

12.156.2.1 PcieDeviceOverrideTablePtr

```
PCIE_DEVICE_OVERRIDE* PCIE_RP_DXE_CONFIG::PcieDeviceOverrideTablePtr
```

PCIe device override table The PCIe device table is being used to override PCIe device ASPM settings.

And it's only used in DXE phase. Please refer to PCIE_DEVICE_OVERRIDE structure for the table. Last entry VendorId must be 0.

Definition at line 329 of file PchPcieRpConfig.h.

The documentation for this struct was generated from the following file:

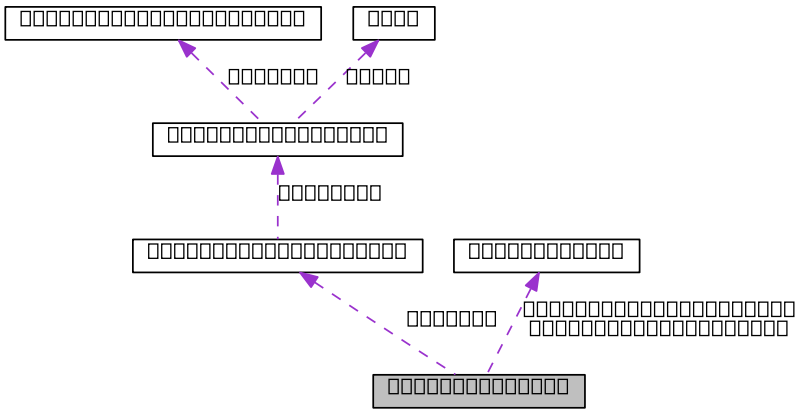
- PchPcieRpConfig.h

12.157 PEI_DMI_CONFIG Struct Reference

The PEI_DMI_CONFIG block describes the expected configuration of the DMI.

`#include <PeiDmiConfig.h>`

Collaboration diagram for PEI_DMI_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT8 DmiMaxLinkSpeed
- UINT8 DmiGen3EqPh2Enable

  ***(Test)** DMI Equalization Phase 2 Enable Control*
- UINT8 DmiGen3EqPh3Method

  ***(Test)** Selects the method for performing Phase3 of Gen3 Equalization on DMI*
- UINT8 DmiGen3ProgramStaticEq

  ***(Test)** Program DMI Gen3 EQ Phase1 Static Presets*
- UINT8 DmiDeEmphasis

  DeEmphasis control for DMI (-6 dB and -3.5 dB are the options)
- UINT8 DmiAspmCtrl

  *ASPM configuration on the DP side of the DMI/OPI Link. Default is **DmiAspmAutoConfig***
- UINT8 DmiAspmL1ExitLatency

  *ASPM configuration on the DP side of the DMI/OPI Link. Default is **DmiAspmAutoConfig***
- UINT8 DmiGen3RootPortPreset [SA_DMI_MAX_LANE]

  Used for programming DMI Gen3 preset values per lane. Range: 0-9, 8 is default for each lane.
- UINT8 DmiGen3EndPointPreset [SA_DMI_MAX_LANE]

  Used for programming DMI Gen3 preset values per lane. Range: 0-9, 7 is default for each lane.
- UINT8 DmiGen3EndPointHint [SA_DMI_MAX_LANE]

  Hint value per lane for the DMI Gen3 End Point. Range: 0-6, 2 is default for each lane.
- UINT8 DmiGen3RxCtlePeaking [SA_DMI_MAX_BUNDLE]

  DMI Gen3 RxCTLEp per-Bundle control.
- UINT8 DmiGen3DsPortRxPreset [SA_DMI_MAX_LANE]

Used for programming DMI Gen3 preset values per lane. Range: 0-9, 8 is default for each lane.

- UINT8 DmiGen3DsPortTxPreset [SA_DMI_MAX_LANE]

  Used for programming DMI Gen3 preset values per lane. Range: 0-9, 8 is default for each lane.
- UINT8 DmiGen3UsPortRxPreset [SA_DMI_MAX_LANE]

  Used for programming DMI Gen3 preset values per lane. Range: 0-9, 8 is default for each lane.
- UINT8 DmiGen3UsPortTxPreset [SA_DMI_MAX_LANE]

  Used for programming DMI Gen3 preset values per lane. Range: 0-9, 8 is default for each lane.
- UINT8 DmiGen3Ltcpre [SA_DMI_MAX_LANE]

  Used for programming DMI Gen3 preset values per lane. Range: 0-9, 8 is default for each lane.
- UINT8 DmiGen3Ltcpo [SA_DMI_MAX_LANE]

  Used for programming DMI Gen3 preset values per lane. Range: 0-9, 8 is default for each lane.
- UINT8 DmiGen3RtcoCpre [SA_DMI_MAX_LANE]

  Used for programming DMI Gen3 preset values per lane. Range: 0-9, 8 is default for each lane.
- UINT8 DmiGen3RtcoCpo [SA_DMI_MAX_LANE]

  Used for programming DMI Gen3 preset values per lane. Range: 0-9, 8 is default for each lane.
- UINT8 DmiGen4Ltcpre [SA_DMI_MAX_LANE]

  Used for programming DMI Gen4 preset values per lane. Range: 0-9, 8 is default for each lane.
- UINT8 DmiGen4Ltcpo [SA_DMI_MAX_LANE]

  Used for programming DMI Gen4 preset values per lane. Range: 0-9, 8 is default for each lane.
- UINT8 DmiGen4DsPortTxPreset [SA_DMI_MAX_LANE]

  Used for programming DMI Gen4 preset values per lane. Range: 0-9, 8 is default for each lane.
- UINT8 DmiGen4UsPortTxPreset [SA_DMI_MAX_LANE]

  Used for programming DMI Gen4 preset values per lane. Range: 0-9, 8 is default for each lane.
- UINT8 DmiGen4RtcoCpre [SA_DMI_MAX_LANE]

  Used for programming DMI Gen4 preset values per lane. Range: 0-9, 8 is default for each lane.
- UINT8 DmiGen4RtcoCpo [SA_DMI_MAX_LANE]

  Used for programming DMI Gen4 preset values per lane. Range: 0-9, 8 is default for each lane.
- UINT8 DmiNewFom

  Used for programming DMI New FOM.
- UINT8 DmiCdrRelock

  Used for programming DMI CDR Relock.

12.157.1 Detailed Description

The PEI_DMI_CONFIG block describes the expected configuration of the DMI.

Revision 1:

- Initial version.

Definition at line 58 of file PeiDmiConfig.h.

12.157.2 Member Data Documentation

12.157.2.1 DmiGen3EqPh2Enable

```
UINT8 PEI_DMI_CONFIG::DmiGen3EqPh2Enable
```

(Test) DMI Equalization Phase 2 Enable Control

- Disabled (0x0) : Disable phase 2
- Enabled (0x1) : Enable phase 2
- **Auto** (0x2) : Use the current default method (Default)

Definition at line 74 of file PeiDmiConfig.h.

12.157.2.2 DmiGen3EqPh3Method

```
UINT8 PEI_DMI_CONFIG::DmiGen3EqPh3Method
```

(Test) Selects the method for performing Phase3 of Gen3 Equalization on DMI

- **Auto** (0x0) : Use the current default method (Default)
- HwEq (0x1) : Use Adaptive Hardware Equalization
- SwEq (0x2) : Use Adaptive Software Equalization (Implemented in BIOS Reference Code)
- Static (0x3) : Use the Static EQs provided in DmiGen3EndPointPreset array for Phase1 AND Phase3 (Instead of just Phase1)
- Disabled (0x4) : Bypass Equalization Phase 3

Definition at line 83 of file PeiDmiConfig.h.

12.157.2.3 DmiGen3ProgramStaticEq

```
UINT8 PEI_DMI_CONFIG::DmiGen3ProgramStaticEq
```

(Test) Program DMI Gen3 EQ Phase1 Static Presets

- Disabled (0x0) : Disable EQ Phase1 Static Presets Programming
- **Enabled** (0x1) : Enable EQ Phase1 Static Presets Programming (Default)

Definition at line 89 of file PeiDmiConfig.h.

12.157.2.4 DmiGen3RxCtlePeaking

```
UINT8 PEI_DMI_CONFIG::DmiGen3RxCtlePeaking[SA_DMI_MAX_BUNDLE]
```

DMI Gen3 RxCTLEp per-Bundle control.

The range of the setting is (0-15). This setting has to be specified based upon platform design and must follow the guideline. Default is 12.

Definition at line 103 of file PeiDmiConfig.h.

12.157.2.5 DmiMaxLinkSpeed

```
UINT8 PEI_DMI_CONFIG::DmiMaxLinkSpeed
```

- **Auto** (0x0) : Maximum possible link speed (Default)
- Gen1 (0x1) : Limit Link to Gen1 Speed
- Gen2 (0x2) : Limit Link to Gen2 Speed
- Gen3 (0x3) : Limit Link to Gen3 Speed

Definition at line 67 of file PeiDmiConfig.h.

The documentation for this struct was generated from the following file:

- PeiDmiConfig.h

12.158 PMC_GLOBAL_RESET_MASK Union Reference

Description of Global Reset Trigger/Event Mask register.

```
#include <PmConfig.h>
```

12.158.1 Detailed Description

Description of Global Reset Trigger/Event Mask register.

Definition at line 138 of file PmConfig.h.

The documentation for this union was generated from the following file:

- PmConfig.h

12.159 PMC_INTERFACE_CONFIG Struct Reference

The PMC_INTERFACE_CONFIG block describes interaction between BIOS and PMC.

`#include <TcssPeiConfig.h>`

Public Attributes

- UINT8 PmcPdEnable

  PMC PD Solution Enable.

12.159.1 Detailed Description

The PMC_INTERFACE_CONFIG block describes interaction between BIOS and PMC.

Definition at line 43 of file TcssPeiConfig.h.

The documentation for this struct was generated from the following file:

- TcssPeiConfig.h

12.160 PMC_LPM_S0IX_SUB_STATE_EN Union Reference

Low Power Mode Enable config.

`#include <PmConfig.h>`

12.160.1 Detailed Description

Low Power Mode Enable config.

Used to configure if respective S0i2/3 sub-states are to be supported by the platform. Each bit corresponds to one LPM state - LPMx->BITx. Some sub-states will require external FETs controlled by EXT_PWR_GATE#/EXT_P↩WR_GATE2# pins to gate v1p05-PHY or v1p05-IS supplies

Definition at line 105 of file PmConfig.h.

12.160.2 Member Data Documentation

12.160.2.1 S0i2p2En

```
UINT32 PMC_LPM_S0IX_SUB_STATE_EN::S0i2p2En
```

LPM2 - S0i2.2 Enable.

Requires EXT_PWR_GATE# controlled FET to gate v1p05 PHY. Refer to V1p05PhyExtFetControlEn.

Definition at line 114 of file PmConfig.h.

12.160.2.2 S0i3p3En

```
UINT32 PMC_LPM_S0IX_SUB_STATE_EN::S0i3p3En
```

LPM5 - S0i3.3 Enable.

Requires EXT_PWR_GATE# controlled FET to gate v1p05 PHY. Refer to V1p05PhyExtFetControlEn.

Definition at line 123 of file PmConfig.h.

12.160.2.3 S0i3p4En

```
UINT32 PMC_LPM_S0IX_SUB_STATE_EN::S0i3p4En
```

LPM7 - S0i3.4 Enable.

Requires EXT_PWR_GATE2# controlled FET to gate v1p05-SRAM/ISCLK. Refer to V1p05IsExtFetControlEn.

Definition at line 129 of file PmConfig.h.

The documentation for this union was generated from the following file:

- PmConfig.h

12.161 PPM_CUSTOM_RATIO_TABLE Struct Reference

This structure is used to describe the custom processor ratio table desired by the platform.

```
#include <CpuPowerMgmtCustomConfig.h>
```

Public Attributes

- UINT8 MaxRatio

  The maximum ratio of the custom ratio table.
- UINT8 NumberOfEntries

  The number of custom ratio state entries, ranges from 2 to 40 for a valid custom ratio table.
- UINT8 Rsvd0 [2]

  Reserved for DWORD alignment.
- UINT32 Cpuid

  The CPU ID for which this custom ratio table applies.
- UINT8 StateRatio [MAX_CUSTOM_RATIO_TABLE_ENTRIES]

  The processor ratios in the custom ratio table.
- UINT8 StateRatioMax16 [MAX_16_CUSTOM_RATIO_TABLE_ENTRIES]

  If there are more than 16 total entries in the StateRatio table, then use these 16 entries to fill max 16 table.

12.161.1 Detailed Description

This structure is used to describe the custom processor ratio table desired by the platform.

Definition at line 39 of file CpuPowerMgmtCustomConfig.h.

12.161.2 Member Data Documentation

12.161.2.1 StateRatioMax16

```
UINT8 PPM_CUSTOM_RATIO_TABLE::StateRatioMax16[MAX_16_CUSTOM_RATIO_TABLE_ENTRIES]
```

If there are more than 16 total entries in the StateRatio table, then use these 16 entries to fill max 16 table.

Note

If NumberOfEntries is 16 or less, or the first entry of this table is 0, then this table is ignored, and up to the top 16 values from the StateRatio table is used instead.

Definition at line 50 of file CpuPowerMgmtCustomConfig.h.

The documentation for this struct was generated from the following file:

- CpuPowerMgmtCustomConfig.h

12.162 PROTECTED_RANGE Struct Reference

Protected Flash Range.

```
#include <FlashProtectionConfig.h>
```

Public Attributes

- UINT32 WriteProtectionEnable: 1

  Write or erase is blocked by hardware. ***0: Disable****; 1: Enable.*
- UINT32 ReadProtectionEnable: 1

  Read is blocked by hardware. ***0: Disable****; 1: Enable.*
- UINT32 RsvdBits: 30

  Reserved.
- UINT16 ProtectedRangeLimit

  The address of the upper limit of protection This is a left shifted address by 12 bits with address bits 11:0 are assumed to be FFFh for limit comparison.
- UINT16 ProtectedRangeBase

  The address of the upper limit of protection This is a left shifted address by 12 bits with address bits 11:0 are assumed to be 0.

12.162.1 Detailed Description

Protected Flash Range.

Definition at line 37 of file FlashProtectionConfig.h.

The documentation for this struct was generated from the following file:

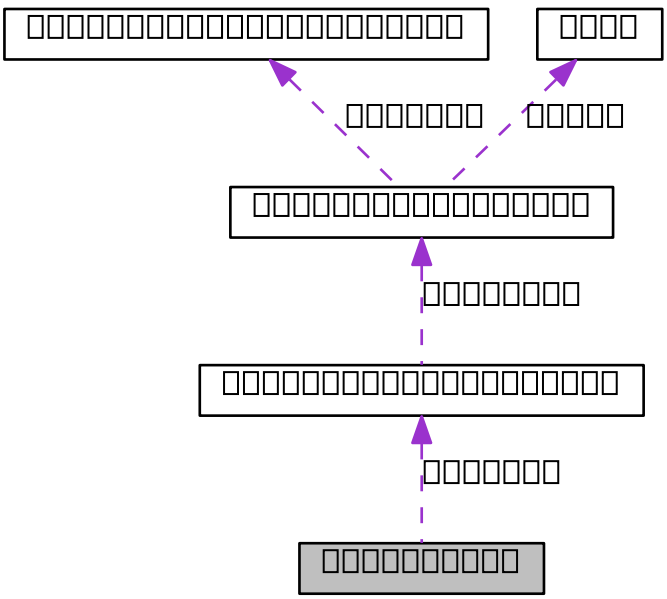
- FlashProtectionConfig.h

12.163 PSF_CONFIG Struct Reference

PSF Configuration **Revision 1**: - Initial version.

```
#include <PsfConfig.h>
```

Collaboration diagram for PSF_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT32 DwbConfigDwbFlushThreshold: 3

  Flush Threshold.
- UINT32 DwbConfigNonxHCIEn: 1

  *Non-xHCI Enable **0: Disable**; 1: Enable*
- UINT32 DwbConfigOBFFEn: 1

  *OBFF Enable **0: Disable**; 1: Enable*
- UINT32 DwbConfigDWBEn: 1

  *Deferred Write Buffer Enable **0: Disable**; 1: Enable*

12.163.1 Detailed Description

PSF Configuration **Revision 1**: - Initial version.

Definition at line 34 of file PsfConfig.h.

12.163.2 Member Data Documentation

12.163.2.1 DwbConfigDwbFlushThreshold

```
UINT32 PSF_CONFIG::DwbConfigDwbFlushThreshold
```

Flush Threshold.

- 0x000: Reserved
- 0x001: 1/8
- 0x010: 2/8
- 0x011: 3/8
- 0x100: 4/8
- 0x101: 5/8
- 0x110: 6/8 – **0x111: 7/8**;

Definition at line 48 of file PsfConfig.h.

The documentation for this struct was generated from the following file:

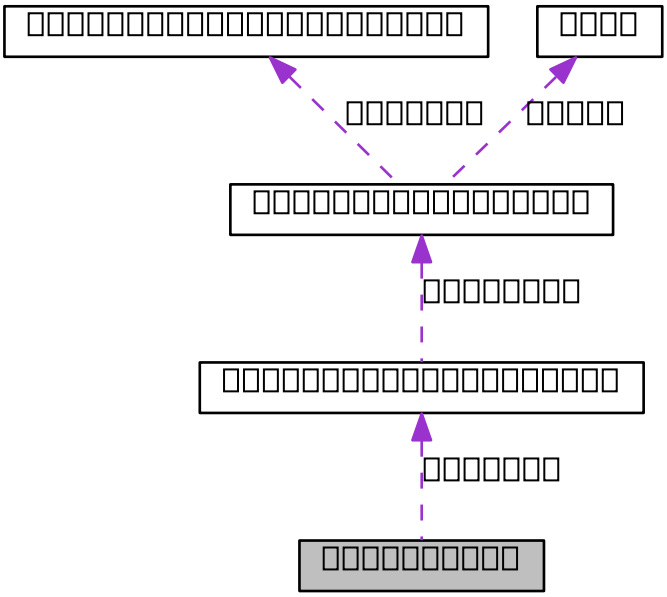
- PsfConfig.h

12.164 RTC_CONFIG Struct Reference

The RTC_CONFIG block describes the expected configuration of RTC configuration.

`#include <RtcConfig.h>`

Collaboration diagram for RTC_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT32 BiosInterfaceLock: 1

  When set, prevents RTC TS (BUC.TS) from being changed.
- UINT32 MemoryLock: 1

  When set, bytes 38h-3Fh in the upper 128bytes bank of RTC RAM are locked and cannot be accessed.

12.164.1 Detailed Description

The RTC_CONFIG block describes the expected configuration of RTC configuration.

Definition at line 34 of file RtcConfig.h.

12.164.2 Member Data Documentation

12.164.2.1 BiosInterfaceLock

```
UINT32 RTC_CONFIG::BiosInterfaceLock
```

When set, prevents RTC TS (BUC.TS) from being changed.

This BILD bit has different function compared to LPC/eSPI, SPI. 0: Disabled; **1: Enabled**

Definition at line 41 of file RtcConfig.h.

12.164.2.2 MemoryLock

```
UINT32 RTC_CONFIG::MemoryLock
```

When set, bytes 38h-3Fh in the upper 128bytes bank of RTC RAM are locked and cannot be accessed.

Writes will be droipped and reads will not return any guaranteed data. 0: Disabled; **1: Enabled**

Definition at line 48 of file RtcConfig.h.

The documentation for this struct was generated from the following file:

- RtcConfig.h

12.165 SA_ADDRESS_DECODE Struct Reference

SA memory address decode.

```
#include <MemoryConfig.h>
```

Public Attributes

- UINT8 Controller
  Offset 0 Zero based Controller number.
- UINT8 Channel
  Offset 1 Zero based Channel number.
- UINT8 Dimm
  Offset 2 Zero based DIMM number.
- UINT8 Rank
  Offset 3 Zero based Rank number.
- UINT8 BankGroup
  Offset 4 Zero based Bank Group number.
- UINT8 Bank
  Offset 5 Zero based Bank number.
- UINT16 Cas
  Offset 6 Zero based CAS number.
- UINT32 Ras
  Offset 8 Zero based RAS number.

12.165.1 Detailed Description

SA memory address decode.

Definition at line 121 of file MemoryConfig.h.

The documentation for this struct was generated from the following file:

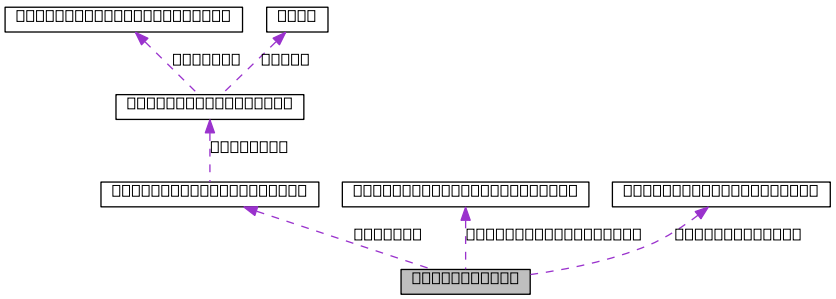
- MemoryConfig.h

12.166 SATA_CONFIG Struct Reference

The SATA_CONFIG block describes the expected configuration of the SATA controllers.

`#include <SataConfig.h>`

Collaboration diagram for SATA_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT8 Enable

  This member describes whether or not the SATA controllers should be enabled.
- UINT8 TestMode

  ***(Test) 0: Disable**; 1: Allow entrance to the PCH SATA test modes*
- UINT8 SalpSupport

  *0: Disable; **1: Enable** Aggressive Link Power Management*
- UINT8 PwrOptEnable

  *0: Disable; **1: Enable** SATA Power Optimizer on PCH side.*
- UINT8 EsataSpeedLimit

  EsataSpeedLimit When enabled, BIOS will configure the PxSCTL.SPD to 2 to limit the eSATA port speed.
- UINT8 LedEnable

  *SATA LED indicates SATA controller activity. 0: Disable; **1: Enable** SATA LED.*
- UINT8 RaidDeviceId

  This option allows to configure SATA controller device ID while in RAID mode.
- UINT8 SataRstInterrupt

Controlls which interrupts will be linked to SATA controller CAP list This option will take effect only if SATA controller is in RAID mode Default: ***PchSataMsix***

- UINT8 SataMode

  Determines the system will be configured to which SATA mode.

- UINT8 SpeedLimit

  Indicates the maximum speed the SATA controller can support.

- UINT8 EnclosureSupport

  Enclosure Management Support. 0: Disable; 1: Enable.

- UINT8 SgpioSupport

  Controlls whenever Serial GPIO support is enabled for controller ***0: Disable****; 1: Enable.*

- PCH_SATA_PORT_CONFIG PortSettings [PCH_MAX_SATA_PORTS]

  This member configures the features, property, and capability for each SATA port.

- SATA_THERMAL_THROTTLING ThermalThrottling

  This field decides the settings of Sata thermal throttling.

- UINT8 LtrEnable

  Latency Tolerance Reporting Mechanism , 0: Platform POR; 1: Enable; 2: Disable.

12.166.1 Detailed Description

The SATA_CONFIG block describes the expected configuration of the SATA controllers.

Revision 1:

- Initial version. **Revision 2**:
- Added DevSlpGpioPinMux.

Definition at line 118 of file SataConfig.h.

12.166.2 Member Data Documentation

12.166.2.1 Enable

```
UINT8 SATA_CONFIG::Enable
```

This member describes whether or not the SATA controllers should be enabled.

0: Disable; **1: Enable**.

Definition at line 123 of file SataConfig.h.

12.166.2.2 EsataSpeedLimit

```
UINT8 SATA_CONFIG::EsataSpeedLimit
```

EsataSpeedLimit When enabled, BIOS will configure the PxSCTL.SPD to 2 to limit the eSATA port speed.

Please be noted, this setting could be cleared by HBA reset, which might be issued by EFI AHCI driver when POST time, or by SATA inbox driver/RST driver after POST. To support the Speed Limitation when POST, the EFI AHCI driver should preserve the setting before and after initialization. For support it after POST, it's dependent on driver's behavior. **0: Disable**; 1: Enable

Definition at line 137 of file SataConfig.h.

12.166.2.3 RaidDeviceId

```
UINT8 SATA_CONFIG::RaidDeviceId
```

This option allows to configure SATA controller device ID while in RAID mode.

Refer to SATA_RAID_DEV_ID enumeration for supported options. Choosing Client will allow RST driver loading, RSTe driver will not be able to load Choosing Alternate will not allow RST inbox driver loading in Windows Choosing Server will allow RSTe driver loading, RST driver will not load **0: Client**; 1: Alternate; 2: Server

Definition at line 147 of file SataConfig.h.

12.166.2.4 SataMode

```
UINT8 SATA_CONFIG::SataMode
```

Determines the system will be configured to which SATA mode.

Refer to SATA_MODE enumeration for supported options. Default is **SataModeAhci**.

Definition at line 159 of file SataConfig.h.

12.166.2.5 SpeedLimit

```
UINT8 SATA_CONFIG::SpeedLimit
```

Indicates the maximum speed the SATA controller can support.

Refer to SATA_SPEED enumeration for supported options. **0h: SataSpeedDefault**; 1h: 1.5 Gb/s (Gen 1); 2h: 3 Gb/s(Gen 2); 3h: 6 Gb/s (Gen 1)

Definition at line 165 of file SataConfig.h.

12.166.2.6 ThermalThrottling

```
SATA_THERMAL_THROTTLING SATA_CONFIG::ThermalThrottling
```

This field decides the settings of Sata thermal throttling.

When the Suggested Setting is enabled, PCH RC will use the suggested representative values.

Definition at line 180 of file SataConfig.h.

The documentation for this struct was generated from the following file:

- SataConfig.h

12.167 SATA_THERMAL_THROTTLING Struct Reference

This structure lists PCH supported SATA thermal throttling register setting for customization.

```
#include <SataConfig.h>
```

Public Attributes

- UINT32 P0T1M: 2
  Port 0 T1 Multipler.
- UINT32 P0T2M: 2
  Port 0 T2 Multipler.
- UINT32 P0T3M: 2
  Port 0 T3 Multipler.
- UINT32 P0TDisp: 2
  Port 0 Tdispatch.
- UINT32 P1T1M: 2
  Port 1 T1 Multipler.
- UINT32 P1T2M: 2
  Port 1 T2 Multipler.
- UINT32 P1T3M: 2
  Port 1 T3 Multipler.
- UINT32 P1TDisp: 2
  Port 1 Tdispatch.
- UINT32 P0Tinact: 2
  Port 0 Tinactive.
- UINT32 P0TDispFinit: 1
  Port 0 Alternate Fast Init Tdispatch.
- UINT32 P1Tinact: 2
  Port 1 Tinactive.
- UINT32 P1TDispFinit: 1
  Port 1 Alternate Fast Init Tdispatch.
- UINT32 SuggestedSetting: 1
  *0: Disable; **1: Enable** suggested representative values*
- UINT32 RsvdBits0: 9
  Reserved bits.

12.167.1 Detailed Description

This structure lists PCH supported SATA thermal throttling register setting for customization.

The settings is programmed through SATA Index/Data registers. When the SuggestedSetting is enabled, the customized values are ignored.

Definition at line 91 of file SataConfig.h.

The documentation for this struct was generated from the following file:

- SataConfig.h

12.168 SCS_SD_DLL Struct Reference

This structre holds the DLL configuration register values that will be programmed by RC if EnableCustomDlls field is set to TRUE.

```
#include <ScsConfig.h>
```

12.168.1 Detailed Description

This structre holds the DLL configuration register values that will be programmed by RC if EnableCustomDlls field is set to TRUE.

Those policies should be used by platform if default values provided by RC are not sufficient to provide stable operation at all supported spped modes. RC will blindly set the DLL values as provided in this structre.

For help with obtaining valid DLL values for your platform please contact enabling support.

Definition at line 52 of file ScsConfig.h.

The documentation for this struct was generated from the following file:

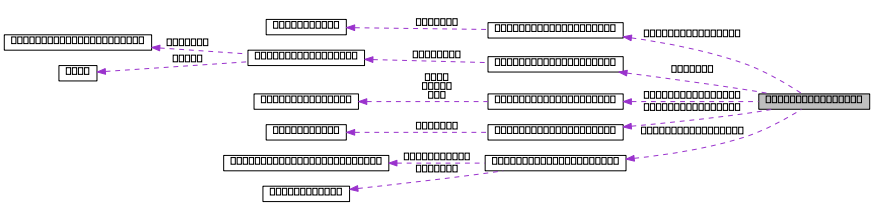
- ScsConfig.h

12.169 SERIAL_IO_CONFIG Struct Reference

The SERIAL_IO_CONFIG block provides the configurations to set the Serial IO controllers.

```
#include <SerialIoConfig.h>
```

Collaboration diagram for SERIAL_IO_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header
  Config Block Header.
- SERIAL_IO_SPI_CONFIG SpiDeviceConfig [PCH_MAX_SERIALIO_SPI_CONTROLLERS]
  SPI Configuration.
- SERIAL_IO_I2C_CONFIG I2cDeviceConfig [PCH_MAX_SERIALIO_I2C_CONTROLLERS]
  I2C Configuration.
- SERIAL_IO_I3C_CONFIG I3cDeviceConfig [PCH_MAX_SERIALIO_I3C_CONTROLLERS]
  I3C Configuration.
- SERIAL_IO_UART_CONFIG UartDeviceConfig [PCH_MAX_SERIALIO_UART_CONTROLLERS]
  UART Configuration.

12.169.1 Detailed Description

The SERIAL_IO_CONFIG block provides the configurations to set the Serial IO controllers.

Revision 1:

- Inital version. **Revision 2:**
- Modified SPI Config to support pinmux functionality. **Revision 3:**
- Modified I3C Config to support more controllers.

Definition at line 41 of file SerialIoConfig.h.

The documentation for this struct was generated from the following file:

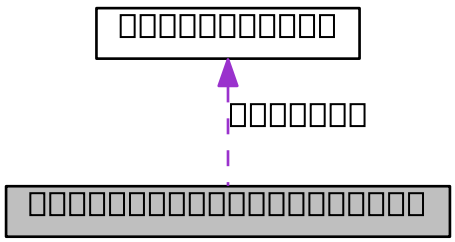
- SerialIoConfig.h

12.170 SERIAL_IO_I2C_CONFIG Struct Reference

Serial IO I2C Controller Configuration.

```
#include <SerialIoDevices.h>
```

Collaboration diagram for SERIAL_IO_I2C_CONFIG:



Public Attributes

- UINT8 PadTermination

  SerialIoI2cPci see SERIAL_IO_I2C_MODE
- I2C_PIN_MUX PinMux

  I2C pin mux configuration.

12.170.1 Detailed Description

Serial IO I2C Controller Configuration.

Definition at line 206 of file SerialIoDevices.h.

12.170.2 Member Data Documentation

12.170.2.1 PadTermination

```
UINT8 SERIAL_IO_I2C_CONFIG::PadTermination
```

SerialIoI2cPci see SERIAL_IO_I2C_MODE

I2C Pads Internal Termination. For more information please see Platform Design Guide. Supported values (check GPIO_ELECTRICAL_CONFIG for reference): **GpioTermNone: No termination**, GpioTermWpu1K: 1kOhm weak pull-up, GpioTermWpu5K: 5kOhm weak pull-up, GpioTermWpu20K: 20kOhm weak pull-up

Definition at line 217 of file SerialIoDevices.h.

The documentation for this struct was generated from the following file:

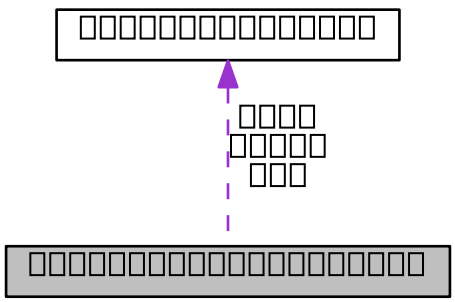
- SerialIoDevices.h

12.171 SERIAL_IO_I3C_CONFIG Struct Reference

The SERIAL_IO_I3C_CONFIG provides the configurations for Serial IO I3C controller.

```
#include <SerialIoDevices.h>
```

Collaboration diagram for SERIAL_IO_I3C_CONFIG:



Public Attributes

- UINT8 Mode
  <b<SerialIoI3cPci ***see SERIAL_IO_I3C_MODE***
- UINT8 Reserved [3]
  Reserved bytes.

12.171.1 Detailed Description

The SERIAL_IO_I3C_CONFIG provides the configurations for Serial IO I3C controller.

Definition at line 255 of file SerialIoDevices.h.

The documentation for this struct was generated from the following file:

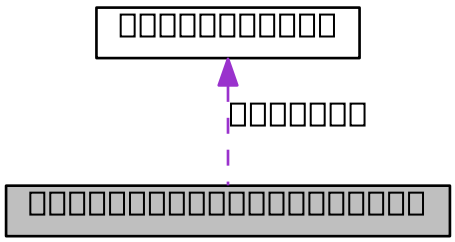
- SerialIoDevices.h

12.172 SERIAL_IO_SPI_CONFIG Struct Reference

The SERIAL_IO_SPI_CONFIG provides the configurations to set the Serial IO SPI controller.

`#include <SerialIoDevices.h>`

Collaboration diagram for SERIAL_IO_SPI_CONFIG:



Public Attributes

- UINT8 Mode
  SerialIoSpiPci *see SERIAL_IO_SPI_MODE*
- UINT8 DefaultCsOutput
  0 = CS0 *CS1, CS2, CS3. Default CS used by the SPI HC*
- UINT8 CsPolarity [PCH_MAX_SERIALIO_SPI_CHIP_SELECTS]
  Selects SPI ChipSelect signal polarity, 0 = low ***1 = High***
- UINT8 CsEnable [PCH_MAX_SERIALIO_SPI_CHIP_SELECTS]
  0 = Enable *1 = Disable. Based on this setting GPIO for given SPIx CSx will be configured in Native mode*
- UINT8 CsMode
  0 = HW Control *1 = SW Control. Sets Chip Select Control mode Hardware or Software.*
- UINT8 CsState
  0 = CS is set to low *1 = CS is set to high*
- SPI_PIN_MUX PinMux
  SPI Pinmux configuration.

12.172.1 Detailed Description

The SERIAL_IO_SPI_CONFIG provides the configurations to set the Serial IO SPI controller.

Definition at line 72 of file SerialIoDevices.h.

The documentation for this struct was generated from the following file:

- SerialIoDevices.h

12.173 SERIAL_IO_UART_ATTRIBUTES Struct Reference

UART Settings.

```
#include <SerialIoDevices.h>
```

Public Attributes

- UINT32 BaudRate

  115200 *Max 6000000 MdePkg.dec PcdUartDefaultBaudRate*
- UINT8 Parity

  1 - No Parity *see EFI_PARITY_TYPE MdePkg.dec PcdUartDefaultParity*
- UINT8 DataBits

  8 *MdePkg.dec PcdUartDefaultDataBits*
- UINT8 StopBits

  1 - One Stop Bit *see EFI_STOP_BITS_TYPE MdePkg.dec PcdUartDefaultStopBits*
- UINT8 AutoFlow

  FALSE *IntelFrameworkModulePkg.dsc PcdIsaBusSerialUseHalfHandshake*

12.173.1 Detailed Description

UART Settings.

Definition at line 129 of file SerialIoDevices.h.

The documentation for this struct was generated from the following file:

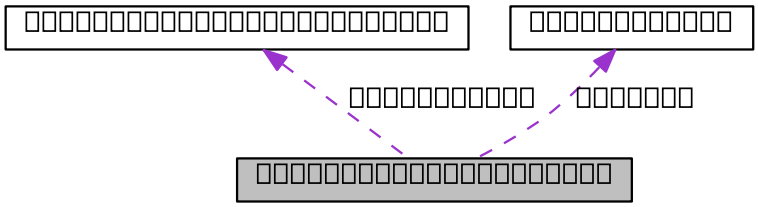
- SerialIoDevices.h

12.174 SERIAL_IO_UART_CONFIG Struct Reference

Serial IO UART Controller Configuration.

`#include <SerialIoDevices.h>`

Collaboration diagram for SERIAL_IO_UART_CONFIG:



Public Attributes

- SERIAL_IO_UART_ATTRIBUTES Attributes

  see SERIAL_IO_UART_ATTRIBUTES
- UART_PIN_MUX PinMux

  UART pin mux configuration.
- UINT8 Mode

  SerialIoUartPci *see SERIAL_IO_UART_MODE*
- UINT8 DBG2

  FALSE *If TRUE adds UART to DBG2 table and overrides UartPg to SerialIoUartPgDisabled*
- UINT8 PowerGating

  SerialIoUartPgAuto *Applies to Hidden/COM/SkipInit see SERIAL_IO_UART_PG*
- UINT8 DmaEnable

  TRUE *Applies to SerialIoUartPci only. Informs OS driver to use DMA, if false it will run in PIO mode*

12.174.1 Detailed Description

Serial IO UART Controller Configuration.

Definition at line 152 of file SerialIoDevices.h.

The documentation for this struct was generated from the following file:

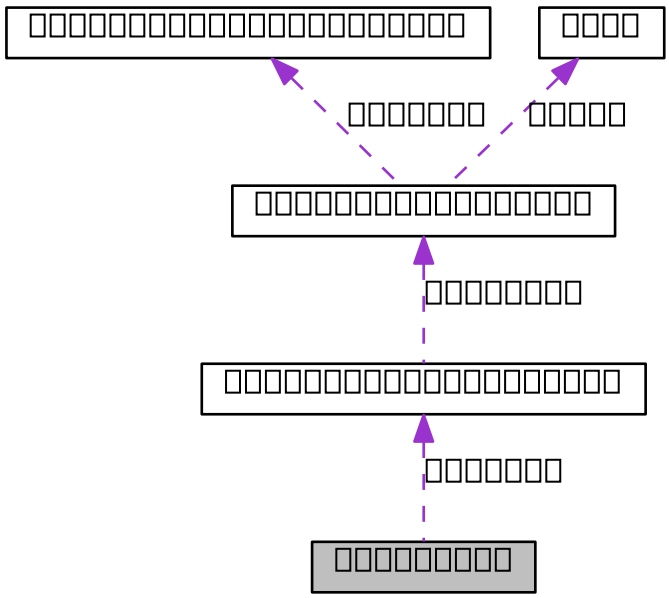
- SerialIoDevices.h

12.175 SI_CONFIG Struct Reference

The Silicon Policy allows the platform code to publish a set of configuration information that the RC drivers will use to configure the silicon hardware.

```
#include <SiConfig.h>
```

Collaboration diagram for SI_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Offset 0 - 27 Config Block Header.
- UINT8 SkipSsidProgramming

  This is used to skip the SSID programming in silicon code.
- UINT8 RsvdBytes0 [3]

  offset 29 - 31
- UINT16 CustomizedSvid

  When SkipSsidProgramming is FALSE, silicon code will use this as default value to program the SVID for all internal devices.
- UINT16 CustomizedSsid

  When SkipSsidProgramming is FALSE, silicon code will use this as default value to program the Sid for all internal devices.
- UINT8 RsvdBytes62 [4]

  Reserved.
- UINT32 * SsidTablePtr

  SsidTablePtr contains the SVID_SID_INIT_ENTRY table.

- UINT16 NumberOfSsidTableEntry

  Number of valid enties in SsidTablePtr.
- UINT8 RsvdBytes1 [2]

  offset 42 - 43
- UINT8 SkipBiosDoneWhenFwUpdate

  This is used to skip setting BIOS_DONE MSR during firmware update boot mode.
- UINT8 RsvdBytes110 [3]

  Offset 45 - 47.

12.175.1 Detailed Description

The Silicon Policy allows the platform code to publish a set of configuration information that the RC drivers will use to configure the silicon hardware.

Revision 1: - Initial version.

Definition at line 37 of file SiConfig.h.

12.175.2 Member Data Documentation

12.175.2.1 CustomizedSsid

```
UINT16 SI_CONFIG::CustomizedSsid
```

When SkipSsidProgramming is FALSE, silicon code will use this as default value to program the Sid for all internal devices.

0: use silicon default SSID 0x7270 , Non-zero: use customized SSID. offset 34 - 35

Definition at line 61 of file SiConfig.h.

12.175.2.2 CustomizedSvid

```
UINT16 SI_CONFIG::CustomizedSvid
```

When SkipSsidProgramming is FALSE, silicon code will use this as default value to program the SVID for all internal devices.

0: use silicon default SVID 0x8086 , Non-zero: use customized SVID. offset 32 - 33

Definition at line 55 of file SiConfig.h.

12.175.2.3 NumberOfSsidTableEntry

`UINT16 SI_CONFIG::NumberOfSsidTableEntry`

Number of valid enties in SsidTablePtr.

This is valid when SkipSsidProgramming is FALSE; **Default is 0.** offset 40 - 41

Definition at line 101 of file SiConfig.h.

12.175.2.4 SkipBiosDoneWhenFwUpdate

`UINT8 SI_CONFIG::SkipBiosDoneWhenFwUpdate`

This is used to skip setting BIOS_DONE MSR during firmware update boot mode.

When set to TRUE and boot mode is BOOT_ON_FLASH_UPDATE, skip setting BIOS_DONE MSR at EndofPei. **0: FALSE**, 1: TRUE Offset 44

Definition at line 109 of file SiConfig.h.

12.175.2.5 SkipSsidProgramming

`UINT8 SI_CONFIG::SkipSsidProgramming`

This is used to skip the SSID programming in silicon code.

When set to TRUE, silicon code will not do any SSID programming and platform code needs to handle that by itself properly. **0: FALSE**, 1: TRUE offset 28

Definition at line 48 of file SiConfig.h.

12.175.2.6 SsidTablePtr

`UINT32* SI_CONFIG::SsidTablePtr`

SsidTablePtr contains the SVID_SID_INIT_ENTRY table.

This is valid when SkipSsidProgramming is FALSE; It doesn't need to contain entries for all Intel internal devices. It can only contains the SVID_SID_INIT_ENTRY entries for those Dev# Func# which needs to be overridden. In the enties, only Dev, Function, SubSystemVendorId, and SubSystemId are required. **Default is NULL.**

E.g. Platform only needs to override BDF 0:31:5 to AAAA:BBBB and BDF 0:31:3 to CCCC:DDDD, it can be done in platform like this: STATIC SVID_SID_INIT_ENTRY mSsidTablePtr[SI_MAX_DEVICE_COUNT] = {0};

VOID SiPolicyUpdate () { UINT32 EntryCount = 0; SiPolicy->SkipSsidProgramming = FALSE; SiPolicy->Ssid↩ TablePtr = mSsidTablePtr;

mSsidTablePtr[EntryCount].Address.Bits.Device = SpiDeviceNumber (); mSsidTablePtr[EntryCount].Address.↩ Bits.Function = SpiFunctionNumber (); mSsidTablePtr[EntryCount].SvidSidValue.SubSystemVendorId = 0xAA↩ AA; mSsidTablePtr[EntryCount].SvidSidValue.SubSystemId = 0xBBBB; EntryCount ++; mSsidTablePtr[Entry↩ Count].Address.Bits.Device = HdaDevNumber (); mSsidTablePtr[EntryCount].Address.Bits.Function = HdaFunc↩ Number (); mSsidTablePtr[EntryCount].SvidSidValue.SubSystemVendorId = 0xCCCC; mSsidTablePtr[Entry↩ Count].SvidSidValue.SubSystemId = 0xDDDD; EntryCount ++; ASSERT (EntryCount < SI_MAX_DEVICE_CO↩ UNT); SiPolicy->NumberOfSsidTableEntry = EntryCount; } offset 36 - 39

Definition at line 95 of file SiConfig.h.

The documentation for this struct was generated from the following file:

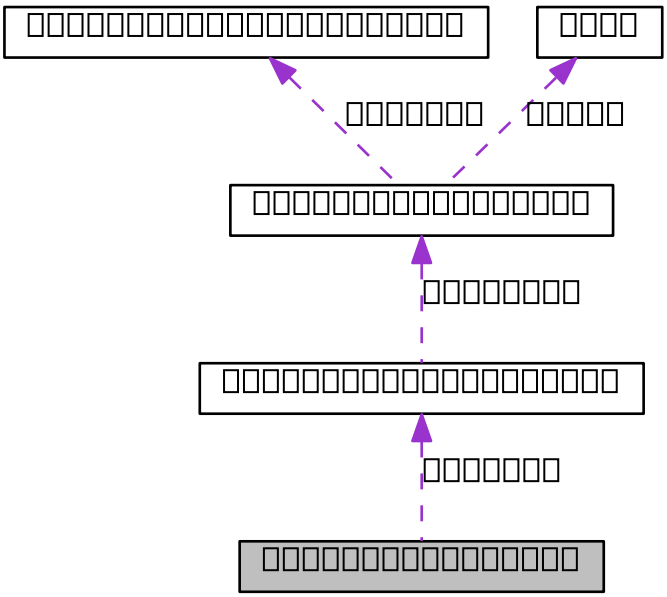
- SiConfig.h

12.176 SI_PREMEM_CONFIG Struct Reference

The Silicon PreMem Policy allows the platform code to publish a set of configuration information that the RC drivers will use to configure the silicon hardware.

`#include <SiPreMemConfig.h>`

Collaboration diagram for SI_PREMEM_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header
  Offset 0 - 27 Config Block Header.
- UINT32 PlatformDebugOption: 4
  Platform Debug Option As a main switch to enable platform debug capability and relevant settings.
- UINT32 RsvdBits: 28
  offset 28 - 31
- UINT8 SkipOverrideBootModeWhenFwUpdate
  offset 32 - 33
- UINT8 RsvdBytes
  offset 34

12.176.1 Detailed Description

The Silicon PreMem Policy allows the platform code to publish a set of configuration information that the RC drivers will use to configure the silicon hardware.

Revision 1:

- Initial version.

Definition at line 44 of file SiPreMemConfig.h.

12.176.2 Member Data Documentation

12.176.2.1 PlatformDebugOption

```
UINT32 SI_PREMEM_CONFIG::PlatformDebugOption
```

Platform Debug Option As a main switch to enable platform debug capability and relevant settings.

Manual: Do not use Platform Debug Option to override other debug-relevant policies, but the user must set each debug option manually, aimed at advanced users.

PlatformDebugOption related policies please refer to PLATFORM_DEBUG_OPTION **0:Disabled**; 2:Enabled TraceHub active; 4:Enabled TraceHub ready; 6:Enabled TraceHub power off; 7:Manual

Definition at line 54 of file SiPreMemConfig.h.

12.176.2.2 SkipOverrideBootModeWhenFwUpdate

```
UINT8 SI_PREMEM_CONFIG::SkipOverrideBootModeWhenFwUpdate
```

offset 32 - 33

This is used to skip override boot mode during firmware update boot mode. When set to TRUE and boot mode is BOOT_ON_FLASH_UPDATE, skip setting boot mode to BOOT_WITH_FULL_CONFIGURATION in PEI memory init. **0: FALSE**, 1: TRUE

Definition at line 65 of file SiPreMemConfig.h.

The documentation for this struct was generated from the following file:

- SiPreMemConfig.h

12.177 SPD_DATA_BUFFER Struct Reference

SPD Data Buffer.

```
#include <MemoryConfig.h>
```

Public Attributes

- UINT8 SpdData [MEM_CFG_MAX_CONTROLLERS][MEM_CFG_MAX_CHANNELS][MEM_CFG_MAX_↩ DIMMS][MEM_CFG_MAX_SPD_SIZE]

  SpdData.

12.177.1 Detailed Description

SPD Data Buffer.

Definition at line 85 of file MemoryConfig.h.

The documentation for this struct was generated from the following file:

- MemoryConfig.h

12.178 SPD_OFFSET_TABLE Struct Reference

SPD Offset Table.

```
#include <MemoryConfig.h>
```

Public Attributes

- UINT16 Start
  Offset 0.
- UINT16 End
  Offset 2.
- UINT8 BootMode
  Offset 4.
- UINT8 Reserved3 [3]
  Offset 5 Reserved for future use.

12.178.1 Detailed Description

SPD Offset Table.

Definition at line 111 of file MemoryConfig.h.

The documentation for this struct was generated from the following file:

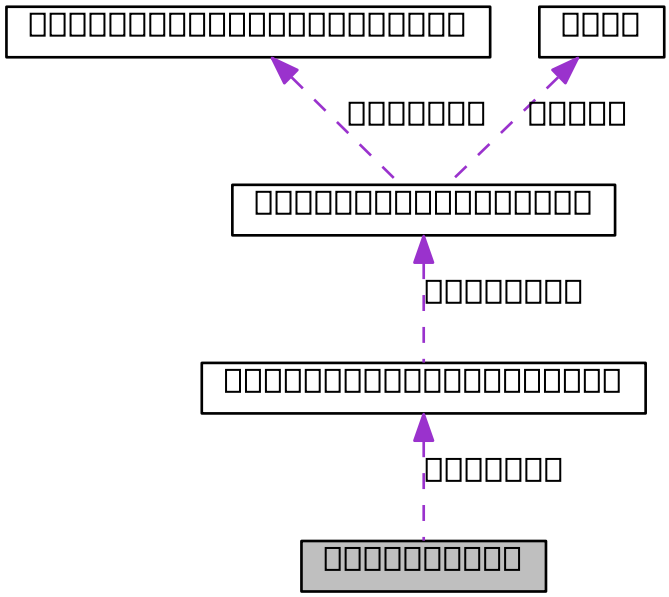
- MemoryConfig.h

12.179 SPI_CONFIG Struct Reference

Basic configuration for option features of PCH SPI Flash controller.

`#include <SpiConfig.h>`

Collaboration diagram for SPI_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT32 ExtendedBiosDecodeRangeEnable: 1

  *Enable extended BIOS Direct Read Region feature Enabling this will make all memory accesses in a decode range to be translated to BIOS region reads from SPI flash **0: Disabled**, 1: Enabled.*
- UINT32 RsvdBits0: 31

  Reserved bits.
- UINT32 ExtendedBiosDecodeRangeBase

  Base address that will be used for Extended Decode Range.
- UINT32 ExtendedBiosDecodeRangeLimit

  Limit address that will be used for Extended Decode Range.

12.179.1 Detailed Description

Basic configuration for option features of PCH SPI Flash controller.

Definition at line 35 of file SpiConfig.h.

12.179.2 Member Data Documentation

12.179.2.1 ExtendedBiosDecodeRangeBase

```
UINT32 SPI_CONFIG::ExtendedBiosDecodeRangeBase
```

Base address that will be used for Extended Decode Range.

This will be ignored when ExtendedBiosDecodeRangeEnable is set to 0.

Definition at line 49 of file SpiConfig.h.

12.179.2.2 ExtendedBiosDecodeRangeLimit

```
UINT32 SPI_CONFIG::ExtendedBiosDecodeRangeLimit
```

Limit address that will be used for Extended Decode Range.

This will be ignored when ExtendedBiosDecodeRangeEnable is set to 0.

Definition at line 54 of file SpiConfig.h.

The documentation for this struct was generated from the following file:

- SpiConfig.h

12.180 SPI_PIN_MUX Struct Reference

SPI signals pin muxing settings.

```
#include <SerialIoDevices.h>
```

Public Attributes

- UINT32 Cs [PCH_MAX_SERIALIO_SPI_CHIP_SELECTS]

  CS Pin mux configuration. Refer to GPIO_*_MUXING_SERIALIO_SPIx_CS_*.
- UINT32 Clk

  CLK Pin mux configuration. Refer to GPIO_*_MUXING_SERIALIO_SPIx_CLK_*.
- UINT32 Miso

  MISO Pin mux configuration. Refer to GPIO_*_MUXING_SERIALIO_SPIx_MISO_*.
- UINT32 Mosi

  MOSI Pin mux configuration. Refer to GPIO_*_MUXING_SERIALIO_SPIx_MOSI_*.

12.180.1 Detailed Description

SPI signals pin muxing settings.

If signal can be enable only on a single pin then this parameter is ignored by RC. Refer to GPIO_∗_MUXING_SE↩ RIALIO_SPIx_∗ in GpioPins∗.h for supported settings on a given platform

Definition at line 62 of file SerialIoDevices.h.

The documentation for this struct was generated from the following file:

- SerialIoDevices.h

12.181 SVID_SID_VALUE Struct Reference

Subsystem Vendor ID / Subsystem ID.

```
#include <SiConfig.h>
```

12.181.1 Detailed Description

Subsystem Vendor ID / Subsystem ID.

Definition at line 122 of file SiConfig.h.

The documentation for this struct was generated from the following file:

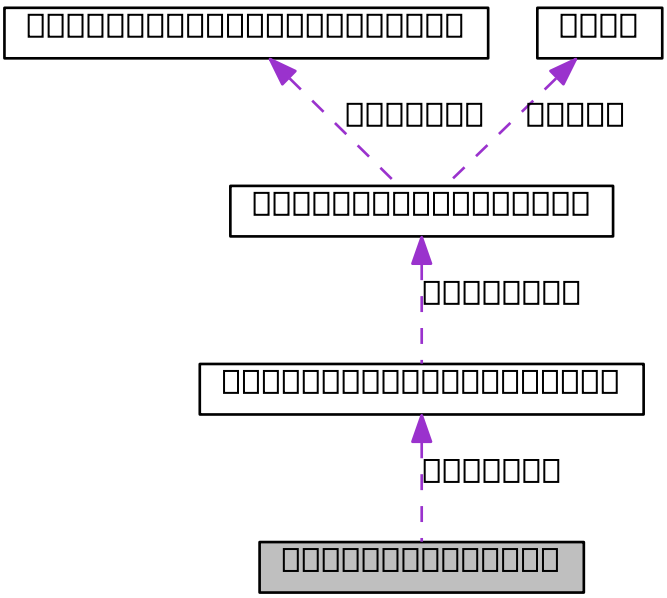
- SiConfig.h

12.182 SWIZZLE_CONFIG Struct Reference

This config block describes how to configure Interrupt Swizzling functionality in EAH.

```
#include <SwizzleConfig.h>
```

Collaboration diagram for SWIZZLE_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.

12.182.1 Detailed Description

This config block describes how to configure Interrupt Swizzling functionality in EAH.

INTx swizzling works on offset basis. If swizzle by 0 then INTA becomes INTA If swizzle by 1 then INTA becomes INTB If swizzle by 2 then INTA becomes INTC If swizzle by 3 then INTA becomes INTD

Definition at line 51 of file SwizzleConfig.h.

The documentation for this struct was generated from the following file:

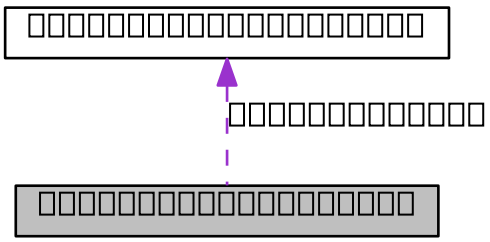
- SwizzleConfig.h

12.183 TCSS_IOM_PEI_CONFIG Struct Reference

The TCSS_IOM_PEI_CONFIG block describes IOM settings for TCSS.

```
#include <TcssPeiConfig.h>
```

Collaboration diagram for TCSS_IOM_PEI_CONFIG:



Public Attributes

- IOM_INTERFACE_CONFIG IomInterface

  Config settings are BIOS <-> IOM interface.
- PMC_INTERFACE_CONFIG PmcInterface

  Config settings for BIOS <-> PMC interface.
- UINT8 TcStateLimit

  Tcss C-State deep stage.
- UINT8 TcNotifyIgd

  Wake iGfx from D3 on HPD_ASSERT or HPD_IRQ assocaited with the DP_ALT and DP tunneling connection. 0: Disable; 1: Enable.
- UINT8 Reserved [2]

  Reserved bytes.

12.183.1 Detailed Description

The TCSS_IOM_PEI_CONFIG block describes IOM settings for TCSS.

Definition at line 117 of file TcssPeiConfig.h.

The documentation for this struct was generated from the following file:

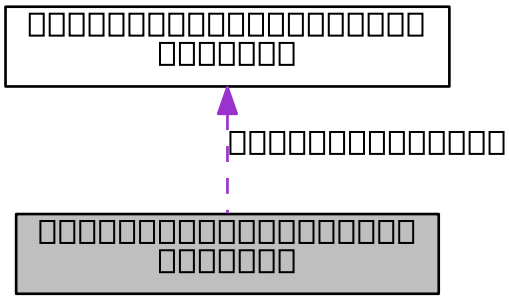
- TcssPeiConfig.h

12.184 TCSS_IOM_PEI_PERMEM_CONFIG Struct Reference

The TCSS_IOM_PEI_PERMEM_CONFIG block describes IOM settings for TCSS.

`#include <TcssPeiPreMemConfig.h>`

Collaboration diagram for TCSS_IOM_PEI_PERMEM_CONFIG:



Public Attributes

- IOM_AUX_ORI_GPIO_PAD_CONFIG IomAuxPortPad [MAX_IOM_AUX_BIAS_COUNT]
  The RECEPTACLE_SBUx_BIAS_CTL port config setting.

12.184.1 Detailed Description

The TCSS_IOM_PEI_PERMEM_CONFIG block describes IOM settings for TCSS.

Definition at line 93 of file TcssPeiPreMemConfig.h.

The documentation for this struct was generated from the following file:

- TcssPeiPreMemConfig.h

12.185 TCSS_MISC_PEI_CONFIG Struct Reference

The TCSS_MISC_PEI_CONFIG block describes MISC settings for TCSS.

`#include <TcssPeiConfig.h>`

Public Attributes

- TCSS_TYPEC_CONV_USBA TcssConvUsbA [MAX_TCSS_USB3_PORTS]

  TCSS Port Convert to TypeA.
- TBT_LSX_OE_PAD_CONFIG LsxOePortPad [MAX_TCSS_USB3_PORTS]

  The Lsx Oe port config setting.
- UINT8 LsxOe

  Bits defining value for LSx OE Orientation.
- UINT8 Reserved [3]

  Reserved bytes.

12.185.1 Detailed Description

The TCSS_MISC_PEI_CONFIG block describes MISC settings for TCSS.

Definition at line 134 of file TcssPeiConfig.h.

The documentation for this struct was generated from the following file:

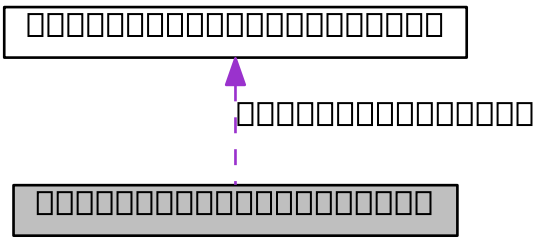
- TcssPeiConfig.h

12.186 TCSS_PCIE_PEI_POLICY Struct Reference

TCSS_PCIE_PEI_POLICY describes PCIe port settings for TCSS.

```
#include <TcssPeiConfig.h>
```

Collaboration diagram for TCSS_PCIE_PEI_POLICY:



Public Attributes

- UINT8 TcssTbtPerfBoost

  Bitmap of TBT performance boost enabled TCSS PCIe root ports.
- UINT8 Reserved [3]

  Reserved bytes.

12.186.1 Detailed Description

TCSS_PCIE_PEI_POLICY describes PCIe port settings for TCSS.

Definition at line 101 of file TcssPeiConfig.h.

12.186.2 Member Data Documentation

12.186.2.1 TcssTbtPerfBoost

```
UINT8 TCSS_PCIE_PEI_POLICY::TcssTbtPerfBoost
```

Bitmap of TBT performance boost enabled TCSS PCIe root ports.

Each bit represents a specific port: BIT0 - BIT7: TCSS PCIe root ports bitmap, where bit i corresponds to TCSS PCIe root port(i), e.g. BIT0 - TCSS PCIe root port0. The max supported TCSS PCIe root port number depends on silicon.

Definition at line 110 of file TcssPeiConfig.h.

The documentation for this struct was generated from the following file:

- TcssPeiConfig.h

12.187 TCSS_PCIE_PORT_POLICY Struct Reference

The TCSS_PCIE_PORT_POLICY block describes PCIe settings for TCSS.

```
#include <TcssPeiConfig.h>
```

Public Attributes

- UINT8 AcsEnabled

  Indicate whether the ACS is enabled. 0: Disable; ***1: Enable****.*
- UINT8 LtrEnable

  Latency Tolerance Reporting Mechanism. ***0: Disable****; 1: Enable.*
- UINT8 PtmEnabled

  Enables PTM capability.
- UINT8 Aspm

  The ASPM configuration of the root port (see: PCH_PCIE_ASPM_CONTROL). Default is
- UINT8 SlotNumber

  Indicates the slot number for the root port. Default is the value as root port index.
- UINT8 SlotPowerLimitScale

  (Test) *Specifies scale used for slot power limit value. Leave as 0 to set to default. Default is* ***zero****.*
- UINT16 SlotPowerLimitValue

  (Test) *Specifies upper limit on power supplies by slot. Leave as 0 to set to default. Default is* ***zero****.*

- UINT8 AdvancedErrorReporting

  *Indicate whether the Advanced Error Reporting is enabled. **0: Disable**; 1: Enable.*
- UINT8 UnsupportedRequestReport

  *Indicate whether the Unsupported Request Report is enabled. **0: Disable**; 1: Enable.*
- UINT8 FatalErrorReport

  *Indicate whether the Fatal Error Report is enabled. **0: Disable**; 1: Enable.*
- UINT8 NoFatalErrorReport

  *Indicate whether the No Fatal Error Report is enabled. **0: Disable**; 1: Enable.*
- UINT8 CorrectableErrorReport

  *Indicate whether the Correctable Error Report is enabled. **0: Disable**; 1: Enable.*
- UINT8 SystemErrorOnFatalError

  *Indicate whether the System Error on Fatal Error is enabled. **0: Disable**; 1: Enable.*
- UINT8 SystemErrorOnNonFatalError

  *Indicate whether the System Error on Non Fatal Error is enabled. **0: Disable**; 1: Enable.*
- UINT8 SystemErrorOnCorrectableError

  *Indicate whether the System Error on Correctable Error is enabled. **0: Disable**; 1: Enable.*
- UINT16 LtrMaxSnoopLatency

  Latency Tolerance Reporting, Max Snoop Latency.
- UINT16 LtrMaxNoSnoopLatency

  Latency Tolerance Reporting, Max Non-Snoop Latency.
- UINT8 SnoopLatencyOverrideMode

  Latency Tolerance Reporting, Snoop Latency Override Mode.
- UINT8 SnoopLatencyOverrideMultiplier

  Latency Tolerance Reporting, Snoop Latency Override Multiplier.
- UINT16 SnoopLatencyOverrideValue

  Latency Tolerance Reporting, Snoop Latency Override Value.
- UINT8 NonSnoopLatencyOverrideMode

  Latency Tolerance Reporting, Non-Snoop Latency Override Mode.
- UINT8 NonSnoopLatencyOverrideMultiplier

  Latency Tolerance Reporting, Non-Snoop Latency Override Multiplier.
- UINT16 NonSnoopLatencyOverrideValue

  Latency Tolerance Reporting, Non-Snoop Latency Override Value.
- UINT8 ForceLtrOverride

  ***0: Disable**; 1: Enable.*
- UINT8 LtrConfigLock

  ***0: Disable**; 1: Enable.*

12.187.1 Detailed Description

The TCSS_PCIE_PORT_POLICY block describes PCIe settings for TCSS.

Definition at line 66 of file TcssPeiConfig.h.

The documentation for this struct was generated from the following file:

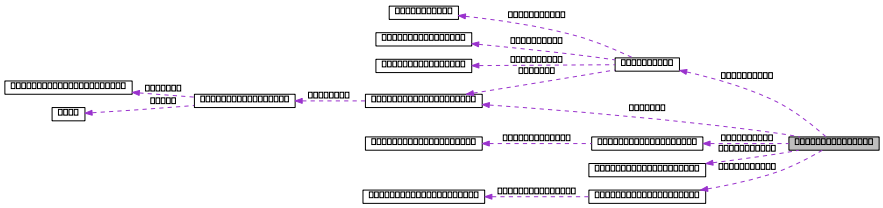
- TcssPeiConfig.h

12.188 TCSS_PEI_CONFIG Struct Reference

The TCSS_PEI_CONFIG block describes TCSS settings for SA.

`#include <TcssPeiConfig.h>`

Collaboration diagram for TCSS_PEI_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Offset 0-27 Config Block Header.
- TCSS_PCIE_PEI_POLICY PciePolicy

  The PCIe Config.
- USB_CONFIG UsbConfig

  USB config is shared between PCH and SA.
- TCSS_IOM_PEI_CONFIG IomConfig

  The Iom Config.
- TCSS_MISC_PEI_CONFIG MiscConfig

  The MISC Config.

12.188.1 Detailed Description

The TCSS_PEI_CONFIG block describes TCSS settings for SA.

Definition at line 144 of file TcssPeiConfig.h.

The documentation for this struct was generated from the following file:

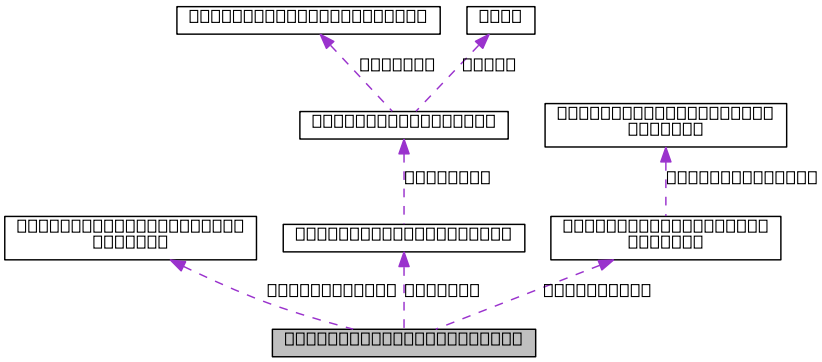
- TcssPeiConfig.h

12.189 TCSS_PEI_PREMEM_CONFIG Struct Reference

This configuration block describes TCSS settings.

`#include <TcssPeiPreMemConfig.h>`

Collaboration diagram for TCSS_PEI_PREMEM_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header
  Offset 0-27 Config Block Header.
- TCSS_USBTC_PEI_PERMEM_CONFIG UsbTcConfig
  USB Type C Port Configuration.
- TCSS_IOM_PEI_PERMEM_CONFIG IomConfig
  The Iom PreMem Config.
- UINT8 TcssEnable
  TCSS is enabled or not.
- UINT8 TcssPlatConf
  Platform configuration (retimer)
- UINT8 Rsvd [2]
  Reserved for future use.

12.189.1 Detailed Description

This configuration block describes TCSS settings.

Revision 1:

- Initial version. **Revision 2**:
- Added TCSS_IOM_PEI_PERMEM_CONFIG **Revision 3**:
- Added TcssPlatConf

Definition at line 108 of file TcssPeiPreMemConfig.h.

The documentation for this struct was generated from the following file:

- TcssPeiPreMemConfig.h

12.190 TCSS_USBTC_PEI_PERMEM_CONFIG Union Reference

The TCSS_USBTC_PEI_PERMEM_CONFIG block describes IOM settings for TCSS.

`#include <TcssPeiPreMemConfig.h>`

12.190.1 Detailed Description

The TCSS_USBTC_PEI_PERMEM_CONFIG block describes IOM settings for TCSS.

Definition at line 67 of file TcssPeiPreMemConfig.h.

The documentation for this union was generated from the following file:

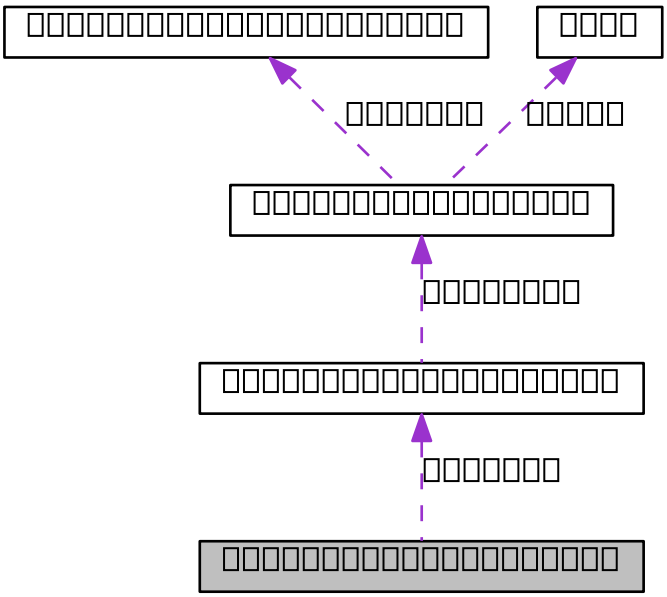
- TcssPeiPreMemConfig.h

12.191 TELEMETRY_PEI_CONFIG Struct Reference

This configuration block describes Telemetry settings in PostMem.

`#include <TelemetryPeiConfig.h>`

Collaboration diagram for TELEMETRY_PEI_CONFIG:



Public Attributes

- UINT32 CpuCrashLogEnable
  Config Block Header.

12.191.1 Detailed Description

This configuration block describes Telemetry settings in PostMem.

Revision 1:

- Initial version.

Definition at line 66 of file TelemetryPeiConfig.h.

The documentation for this struct was generated from the following file:

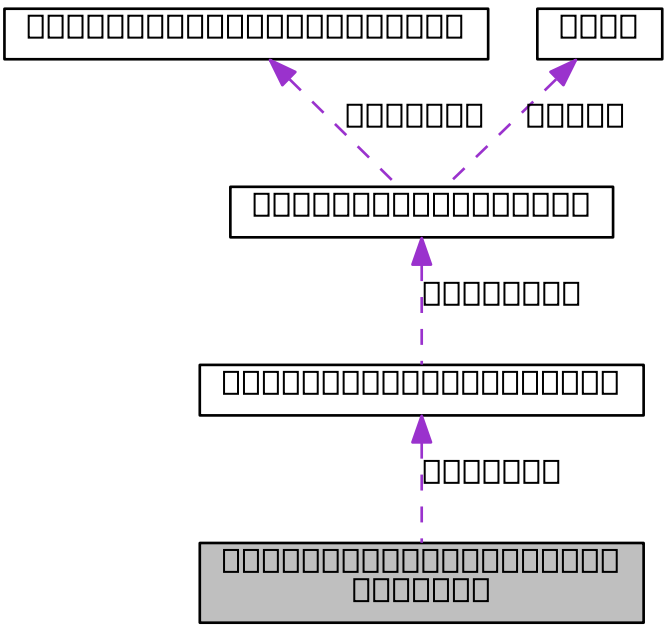
- TelemetryPeiConfig.h

12.192 TELEMETRY_PEI_PREMEM_CONFIG Struct Reference

This configuration block describes Telemetry settings in PreMem.

`#include <TelemetryPeiConfig.h>`

Collaboration diagram for TELEMETRY_PEI_PREMEM_CONFIG:



Public Attributes

- UINT32 CpuCrashLogDevice

  Config Block Header.
- UINT32 StreamTracerSize

  Enable or Disable CrashLog/Telemetry Device 0- Disable, ***1- Enable***
- UINT8 RsvdBytes53 [4]

  StreamTracer size in KB, Range 64KB - 512MB.
- UINT64 StreamTracerBase

  Reserved.
- UINT32 StreamTracerMode

  StreamTracer physical address.
- UINT8 RsvdBytes56 [4]

  0 - Disable StreamTracer, 3 - Allow user to enter size, 8192 - Auto mode with size 8MB, 524288 - Advanced trace mode with size 512MB

12.192.1 Detailed Description

This configuration block describes Telemetry settings in PreMem.

Revision 1:

- Initial version.

Definition at line 51 of file TelemetryPeiConfig.h.

The documentation for this struct was generated from the following file:

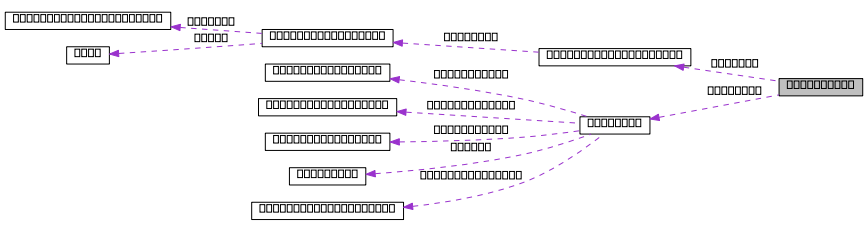
- TelemetryPeiConfig.h

12.193 THC_CONFIG Struct Reference

THC_CONFIG block provides the configurations forTouch Host Controllers.

`#include <ThcConfig.h>`

Collaboration diagram for THC_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header
  Config Block Header.
- THC_PORT ThcPort [2]
  Port Configuration.

12.193.1 Detailed Description

THC_CONFIG block provides the configurations forTouch Host Controllers.

Assignment field in each THC port controlls the THC behavior.

Available scenarios: 1: Single Port 0 used by THC0

- THC0 Enabled
- Port0 assigned to THC0
- Port1 unassigned
- THC1 will be automatically Disabled. 2: Both ports used by THC0
- THC0 Enabled
- Port0 assigned to THC0
- Port1 assigned to THC0
- THC1 will be automatically Disabled. 3: Port 0 used by THC0 and Port 1 used by THC1
- THC0 Enabled
- Port0 assigned to THC0
- THC1 Enabled
- Port1 assigned to THC1. **4: Both Ports unassigned.** Both THC Controllers will be disabled in that case.

Note

Invalid scenario that will cause ASSERT.

1. Same port Number assigned to THC0 or THC1.
2. Two Ports assigned to THC1.

Revision 1:

- Inital version. **Revision 2:**
- Add HidOverSpi THC_MODE **Revision 3:**
- Expand HidOverSpi support **Revision 4:**
- Add ResetSequencingDelay in THC_HID_OVER_SPI **Revision 5:**
- Moved out reset data from THC_HID_OVER_SPI
- Add THC_HID_OVER_I2C **Revision 6:**
- Added THC_WOT_PIN_CONFIG to enable configuration of ACPI resource of Wake Pin **Revision 7:**
- Added THC_PORT_SUBSYTEM_ID to program customized SubSytem ID **Revision 8:**
- expand THC_PORT structure with addition of FRAME_SIZE_AND_INT_DELAY with 4 more variables for Max Frame Size and Interrupt delay support

Definition at line 225 of file ThcConfig.h.

The documentation for this struct was generated from the following file:

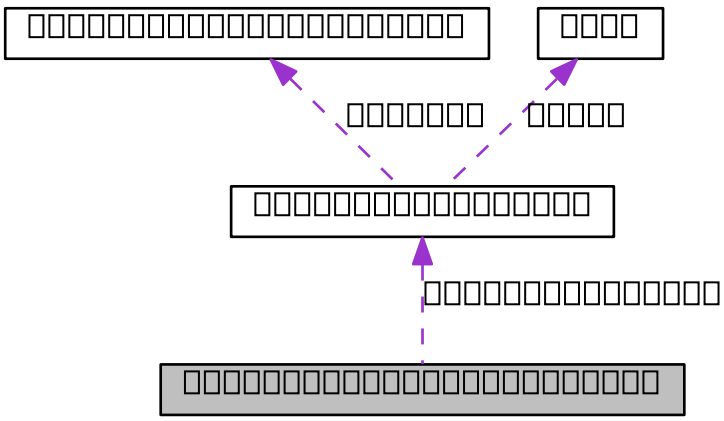
- ThcConfig.h

12.194 THC_DATA_SSID_CONFIG_HOB Struct Reference

THC SSID Data Hob.

`#include <ThcConfig.h>`

Collaboration diagram for THC_DATA_SSID_CONFIG_HOB:



Public Attributes

- EFI_HOB_GUID_TYPE EfiHobGuidType
  GUID.
- SVID_SID_SKIP_TABLE * ThcSsidTable
  THC SSID table.
- UINT16 NumberOfThcSsidTable
  number of THC SSID table
- UINT8 Reserved [2]
  Reserved for Hob size allignment.

12.194.1 Detailed Description

THC SSID Data Hob.

Definition at line 233 of file ThcConfig.h.

The documentation for this struct was generated from the following file:

- ThcConfig.h

12.195 THC_HID_OVER_I2C Struct Reference

THC Hid Over I2C mode related settings.

`#include <ThcConfig.h>`

Public Attributes

- UINT32 DeviceAddress

  Device Address.
- UINT32 ConnectionSpeed

  Connection Speed - Frequency [Hz].
- UINT32 AddressingMode

  Addressing Mode 0x1: The connection uses 10-bit addressing.
- UINT32 DeviceDescriptorAddress

  Device Descriptor Address.
- UINT32 StandardModeSerialClockLineHighPeriod

  Serial Clock Line High Period.
- UINT32 StandardModeSerialClockLineLowPeriod

  Standard Mode Serial Clock Line Low Period.
- UINT32 StandardModeSerialDataLineTransmitHoldPeriod

  Standard Mode Serial Data Line Transmit Hold Period.
- UINT32 StandardModeSerialDataLineReceiveHoldPeriod

  Standard Mode Serial Data Line Receive Hold Period.
- UINT32 FastModeSerialClockLineHighPeriod

  Fast Mode Serial Clock Line High Period.
- UINT32 FastModeSerialClockLineLowPeriod

  Fast Mode Serial Clock Line Low Period.
- UINT32 FastModeSerialDataLineTransmitHoldPeriod

  Fast Mode Serial Data Line Transmit Hold Period.
- UINT32 FastModeSerialDataLineReceiveHoldPeriod

  Fast Mode Serial Data Line Receive Hold Period.
- UINT32 MaxSuppressedSpikesSMFMFMP

  Maximum Length Of Suppressed Spikes In Std Mode Fast Mode And Fast Mode Plus.
- UINT32 FastModePlusSerialClockLineHighPeriod

  Fast Mode Plus Serial Clock Line High Period.
- UINT32 FastModePlusSerialClockLineLowPeriod

  Fast Mode Plus Serial Clock Line Low Period.
- UINT32 FastModePlusSerialDataLineTransmitHoldPeriod

  Fast Mode Plus Serial Data Line Transmit Hold Period.
- UINT32 FastModePlusSerialDataLineReceiveHoldPeriod

  Fast Mode Plus Serial Data Line Receive Hold Period.
- UINT32 HighSpeedModePlusSerialClockLineHighPeriod

  High Speed Mode Plus Serial Clock Line High Period.
- UINT32 HighSpeedModePlusSerialClockLineLowPeriod

  High Speed Mode Plus Serial Clock Line Low Period.
- UINT32 HighSpeedModePlusSerialDataLineTransmitHoldPeriod

  High Speed Mode Plus Serial Data Line Transmit Hold Period.
- UINT32 HighSpeedModePlusSerialDataLineReceiveHoldPeriod

  High Speed Mode Plus Serial Data Line Receive Hold Period.
- UINT32 MaximumLengthOfSuppressedSpikesInHighSpeedMode

  Maximum Length Of Suppressed Spikes In High Speed Mode.

12.195.1 Detailed Description

THC Hid Over I2C mode related settings.

Only applicable when HidOverI2c is enbled.

Definition at line 89 of file ThcConfig.h.

12.195.2 Member Data Documentation

12.195.2.1 AddressingMode

```
UINT32 THC_HID_OVER_I2C::AddressingMode
```

Addressing Mode 0x1: The connection uses 10-bit addressing.

0x0: The connection uses 7-bit addressing.

Definition at line 97 of file ThcConfig.h.

The documentation for this struct was generated from the following file:

- ThcConfig.h

12.196 THC_HID_OVER_SPI Struct Reference

THC Hid Over SPI mode related settings.

```
#include <ThcConfig.h>
```

Public Attributes

- UINT32 Frequency
  Connection Speed - SPI Frequency in Hz.
- UINT32 InputReportHeaderAddress
  Input Report Header Address.
- UINT32 InputReportBodyAddress
  Input Report Body Address.
- UINT32 OutputReportAddress
  Output Report Address.
- UINT32 ReadOpcode
  Read Opcode.
- UINT32 WriteOpcode
  Write Opcode.
- UINT32 Flags
  Flags.
- UINT32 LimitPacketSize
  Limit Packet Size.

12.196.1 Detailed Description

THC Hid Over SPI mode related settings.

Only applicable when HidOverSpi is enbled.

Definition at line 66 of file ThcConfig.h.

The documentation for this struct was generated from the following file:

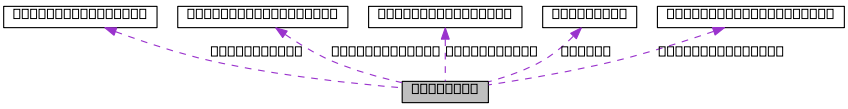
- ThcConfig.h

12.197 THC_PORT Struct Reference

Port Configuration structure required for each Port that THC might use.

`#include <ThcConfig.h>`

Collaboration diagram for THC_PORT:



Public Attributes

- UINT32 Assignment

  Sets THCx assignment see THC_PORT_ASSIGNMENT.
- UINT32 InterruptPinMuxing

  Each GPIO PORTx/SPIx INTB Pin has different muxing options refer to GPIO_∗_MUXING_THC_SPIx_∗.
- UINT8 WakeOnTouch

  *1 = Enable **0 = Disable**. Based on this setting vGPIO for given THC will be in native mode, and additional _CRS for wake will be exposed in ACPI*
- UINT8 ThcDsyncPadEnable

  *1 = Enable **0 = Disable**. Based on this settings enables display sync GPIO for given THC*
- UINT8 Mode

  *2 = Hid over I2C **1 = Hid over SPI 0 = THC. Switch between Intel THC protocol and Industry standard HID Over SPI protocol.***
- UINT8 TimestampTimerMode

  *Timestamp timer behavior in D0i2 **1 = Timer resets to 0 when entering D0i2** 0 = Timer is paused instead of reset to 0 when entering D0i2.*
- THC_WOT_PIN_CONFIG WotPinConfig

  Wake On Touch pin configuration.
- THC_HID_OVER_SPI HidOverSpi

  Hid Over Spi mode settings.
- UINT32 ActiveLtr

  ***0xFFFFFFFF = Driver Default** Active Ltr*

- UINT32 IdleLtr

  0xFFFFFFFF = Driver Default *Idle Ltr*
- UINT32 PerformanceLimitation

  Performance Limitation.
- UINT32 DisplayFrameSyncPeriod

  Period of the emulated display frame sync [ms] The minimum period is 2ms, maximum period is 100ms Default: ***10****ms.*
- THC_RESET Reset

  Reset Line settings.
- THC_HID_OVER_I2C HidOverI2c

  Hid Over I2c mode settings.
- UINT32 Clock

  For available options see THC_PORT_CLOCK.
- THC_PORT_SUBSYTEM_ID ThcSubSystemId

  Customized Subsystem ID.
- FRAME_SIZE_AND_INT_DELAY FrameSizeAndIntDelay

  Frame Size And Int Delay structure.

12.197.1 Detailed Description

Port Configuration structure required for each Port that THC might use.

Definition at line 148 of file ThcConfig.h.

The documentation for this struct was generated from the following file:

- ThcConfig.h

12.198 THC_PORT_SUBSYTEM_ID Struct Reference

Structure to hold Sub System ID for port.

`#include <ThcConfig.h>`

Public Attributes

- UINT16 CustomizedSsid

  Sets Customized SubSytem ID for Port.
- UINT16 CustomizedSvid

  Sets Customized SubSytem Vendor ID for Port.

12.198.1 Detailed Description

Structure to hold Sub System ID for port.

Definition at line 129 of file ThcConfig.h.

The documentation for this struct was generated from the following file:

- ThcConfig.h

12.199 THC_RESET Struct Reference

THC Reset Pad related settings.

```
#include <ThcConfig.h>
```

Public Attributes

- UINT32 ResetPad

  *Reset Pad - **0x0 = THC HW default** BIOS will pass Pad infromation to ACPI, this Pad is used in _RST function.*
- UINT32 ResetPadTrigger

  Reset Pad Trigger.
- UINT32 ResetSequencingDelay

  Policy control for reset sequencing delay (_INI, _RST)

12.199.1 Detailed Description

THC Reset Pad related settings.

Definition at line 80 of file ThcConfig.h.

The documentation for this struct was generated from the following file:

- ThcConfig.h

12.200 THC_WOT_PIN_CONFIG Struct Reference

Structure to hold Wake On Touch pin configuration for GPIO resource in ACPI.

```
#include <ThcConfig.h>
```

12.200.1 Detailed Description

Structure to hold Wake On Touch pin configuration for GPIO resource in ACPI.

Definition at line 137 of file ThcConfig.h.

The documentation for this struct was generated from the following file:

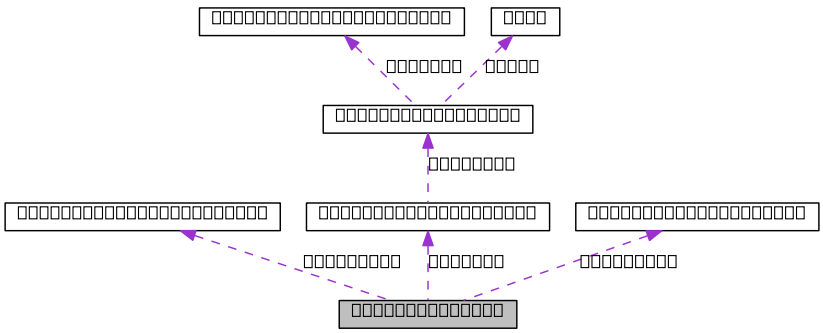
- ThcConfig.h

12.201 THERMAL_CONFIG Struct Reference

The THERMAL_CONFIG block describes the expected configuration of the Thermal IP block.

`#include <ThermalConfig.h>`

Collaboration diagram for THERMAL_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT32 PchHotEnable: 1

  Enable PCHHOT# pin assertion when temperature is higher than PchHotLevel. 0: ***Disabled, 1: Enabled.***
- THERMAL_THROTTLE_LEVELS TTLevels

  This field decides the settings of Thermal throttling.
- DMI_HW_WIDTH_CONTROL DmiHaAWC

  This field decides the settings of DMI throttling.
- UINT16 PchHotLevel

  The recommendation is the same as Cat Trip point.

12.201.1 Detailed Description

The THERMAL_CONFIG block describes the expected configuration of the Thermal IP block.

Definition at line 83 of file ThermalConfig.h.

12.201.2 Member Data Documentation

12.201.2.1 DmiHaAWC

`DMI_HW_WIDTH_CONTROL THERMAL_CONFIG::DmiHaAWC`

This field decides the settings of DMI throttling.

When the Suggested Setting is enabled, PCH RC will use the suggested representative values.

Definition at line 96 of file ThermalConfig.h.

12.201.2.2 PchHotLevel

`UINT16 THERMAL_CONFIG::PchHotLevel`

The recommendation is the same as Cat Trip point.

This field decides the temperature, default is **120**. Temperature value used for PCHHOT# pin assertion based on 2s complement format

- 0x001 positive 1'C
- 0x000 0'C
- 0x1FF negative 1'C
- 0x1D8 negative 40'C
- and so on

Definition at line 107 of file ThermalConfig.h.

12.201.2.3 TTLevels

`THERMAL_THROTTLE_LEVELS THERMAL_CONFIG::TTLevels`

This field decides the settings of Thermal throttling.

When the Suggested Setting is enabled, PCH RC will use the suggested representative values.

Definition at line 91 of file ThermalConfig.h.

The documentation for this struct was generated from the following file:

- ThermalConfig.h

12.202 THERMAL_THROTTLE_LEVELS Struct Reference

This structure lists PCH supported throttling register setting for custimization.

`#include <ThermalConfig.h>`

Public Attributes

- UINT32 T0Level: 9

  Custimized T0Level value. If SuggestedSetting is used, this setting is ignored.
- UINT32 T1Level: 9

  Custimized T1Level value. If SuggestedSetting is used, this setting is ignored.
- UINT32 T2Level: 9

  Custimized T2Level value. If SuggestedSetting is used, this setting is ignored.
- UINT32 TTEnable: 1

  Enable the thermal throttle function. If SuggestedSetting is used, this settings is ignored.
- UINT32 TTState13Enable: 1

  When set to 1 and the programmed GPIO pin is a 1, then PMSync state 13 will force at least T2 state.
- UINT32 TTLock: 1

  When set to 1, this entire register (TL) is locked and remains locked until the next platform reset.
- UINT32 SuggestedSetting: 1

  0: Disable; ***1: Enable*** *suggested representative values.*
- UINT32 RsvdBits0: 1

  Reserved bits.
- UINT32 Rsvd0

  Reserved bytes.

12.202.1 Detailed Description

This structure lists PCH supported throttling register setting for custimization.

When the SuggestedSetting is enabled, the customized values are ignored.

Definition at line 33 of file ThermalConfig.h.

12.202.2 Member Data Documentation

12.202.2.1 TTLock

```
UINT32 THERMAL_THROTTLE_LEVELS::TTLock
```

When set to 1, this entire register (TL) is locked and remains locked until the next platform reset.

If SuggestedSetting is used, this setting is ignored.

Definition at line 47 of file ThermalConfig.h.

12.202.2.2 TTState13Enable

```
UINT32 THERMAL_THROTTLE_LEVELS::TTState13Enable
```

When set to 1 and the programmed GPIO pin is a 1, then PMSync state 13 will force at least T2 state.

If SuggestedSetting is used, this setting is ignored.

Definition at line 42 of file ThermalConfig.h.

The documentation for this struct was generated from the following file:

- ThermalConfig.h

12.203 TRACE_HUB_CONFIG Struct Reference

TRACE_HUB_CONFIG block describes TraceHub settings.

```
#include <TraceHubConfig.h>
```

Public Attributes

- UINT64 MemReg0Size

  Trace hub memory buffer region size value.
- UINT8 EnableMode

  Trace hub mode.
- UINT8 AetEnabled

  AET is one of FW trace agents which is exclusively rounted to one TraceHub.
- UINT8 BiosTraceSink

  *Indicate if BIOS trace is sent to this TraceHub instance **0 = Disable**; 1 = Enabled.*
- UINT8 Reserved [5]

  Filling reserved for future use & Config block DW alignment.

12.203.1 Detailed Description

TRACE_HUB_CONFIG block describes TraceHub settings.

Definition at line 70 of file TraceHubConfig.h.

12.203.2 Member Data Documentation

12.203.2.1 AetEnabled

`UINT8 TRACE_HUB_CONFIG::AetEnabled`

AET is one of FW trace agents which is exclusively rounted to one TraceHub.

i.e. when one TraceHub device enableds it, the other one must disable it. **0 = Disable**; 1 = Enabled

Definition at line 89 of file TraceHubConfig.h.

12.203.2.2 EnableMode

`UINT8 TRACE_HUB_CONFIG::EnableMode`

Trace hub mode.

Default is disabled. **0 = Disable**; 1 = Enabled Refer to TRACE_HUB_ENABLE_MODE

Definition at line 83 of file TraceHubConfig.h.

12.203.2.3 MemReg0Size

`UINT64 TRACE_HUB_CONFIG::MemReg0Size`

Trace hub memory buffer region size value.

Supported size refer to TRACE_BUFFER_SIZE

Definition at line 76 of file TraceHubConfig.h.

The documentation for this struct was generated from the following file:

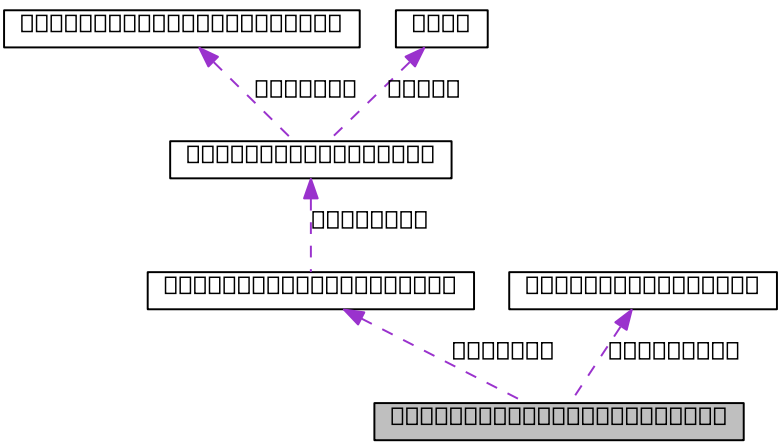
- TraceHubConfig.h

12.204 TRACE_HUB_PREMEM_CONFIG Struct Reference

Trace Hub PreMem Configuration Contains Trace Hub settings **Revision 1**: - Initial version.

`#include <TraceHubConfig.h>`

Collaboration diagram for TRACE_HUB_PREMEM_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header
  Config Block Header.
- TRACE_HUB_CONFIG TraceHub [MaxTraceHub]
  Trace Hub Config.

12.204.1 Detailed Description

Trace Hub PreMem Configuration Contains Trace Hub settings **Revision 1**: - Initial version.

Definition at line 117 of file TraceHubConfig.h.

The documentation for this struct was generated from the following file:

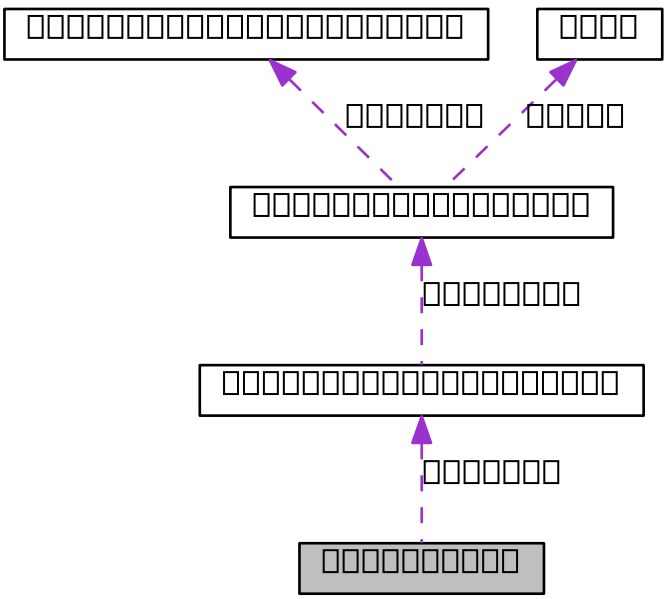
- TraceHubConfig.h

12.205 TSN_CONFIG Struct Reference

TSN Config settings.

`#include <TsnConfig.h>`

Collaboration diagram for TSN_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT8 Enable [PCH_MAX_TSN_PORT]

  Determines if enable PCH internal TSN, 0: Disable; ***1: Enable****.*
- TSN_MAC_ADDR Port [PCH_MAX_TSN_PORT]

  MAC Address Data Structure.

12.205.1 Detailed Description

TSN Config settings.

Definition at line 57 of file TsnConfig.h.

The documentation for this struct was generated from the following file:

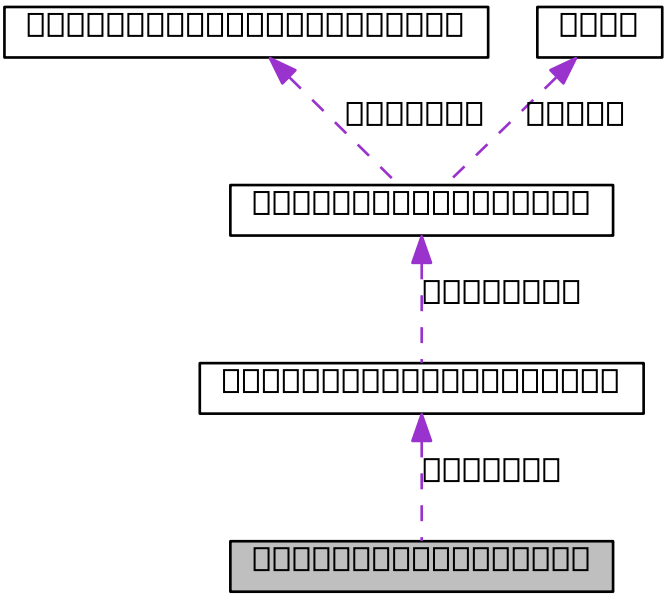
- TsnConfig.h

12.206 TXT_PREMEM_CONFIG Struct Reference

CPU TXT PreMemory Configuration Structure.

`#include <TxtConfig.h>`

Collaboration diagram for TXT_PREMEM_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT32 SinitMemorySize

  Size of SINIT module if installed in flash part. Zero otherwise.
- UINT32 TxtHeapMemorySize

  Size of memory reserved for TXT Heap. This memory is used by MLE.
- UINT32 TxtDprMemorySize

  Size of DPR protected memory reserved for Intel TXT component.
- EFI_PHYSICAL_ADDRESS TxtDprMemoryBase

  Base address of DPR protected memory reserved for Intel TXT component.
- UINT64 TxtLcpPdSize

  Size of Platform Default Launch Control Policy data if installed in flash part. Zero otherwise.
- EFI_PHYSICAL_ADDRESS TxtLcpPdBase

  Base address of Platform Default Launch Control Policy data if installed in flash part. Zero otherwise.
- EFI_PHYSICAL_ADDRESS ApStartupBase

  Base address of TXT AP Startup code.
- EFI_PHYSICAL_ADDRESS BiosAcmBase

  Base address of BIOS ACM in flash part.
- UINT32 BiosAcmSize

  Size of ACM Binary.
- UINT32 TgaSize

  Size of Trusted Graphics Aperture if supported by chipset.
- UINT8 TxtImplemented

  This field currently is used to tell MRC if it should run after TXT initializatoin completed: ***0=Run without waiting for TXT****, 1=Run after TXT initialization by callback.*
- UINT8 CmosTxtOffset

  CMOS offset for TXT policy data.
- UINT8 Reserved [2]

  Reserved bytes for alignment.

12.206.1 Detailed Description

CPU TXT PreMemory Configuration Structure.

Note

Optional. These policies will be ignored if CPU_SECURITY_PREMEM_CONFIG -> Txt is disabled, or PeiTxtLibNull is used.

Revision 1:

- Initial version.

Definition at line 38 of file TxtConfig.h.

The documentation for this struct was generated from the following file:

- TxtConfig.h

12.207 UART_PIN_MUX Struct Reference

UART signals pin muxing settings.

`#include <SerialIoDevices.h>`

Public Attributes

- UINT32 Rx

  RXD Pin mux configuration. Refer to GPIO_∗_MUXING_SERIALIO_UARTx_RXD_∗.
- UINT32 Tx

  TXD Pin mux configuration. Refer to GPIO_∗_MUXING_SERIALIO_UARTx_TXD_∗.
- UINT32 Rts

  RTS Pin mux configuration. Refer to GPIO_∗_MUXING_SERIALIO_UARTx_RTS_∗.
- UINT32 Cts

  CTS Pin mux configuration. Refer to GPIO_∗_MUXING_SERIALIO_UARTx_CTS_∗.

12.207.1 Detailed Description

UART signals pin muxing settings.

If signal can be enable only on a single pin then this parameter is ignored by RC. Refer to GPIO_∗_MUXING_SE↩ RIALIO_UARTx_∗ in GpioPins∗.h for supported settings on a given platform

Definition at line 142 of file SerialIoDevices.h.

The documentation for this struct was generated from the following file:

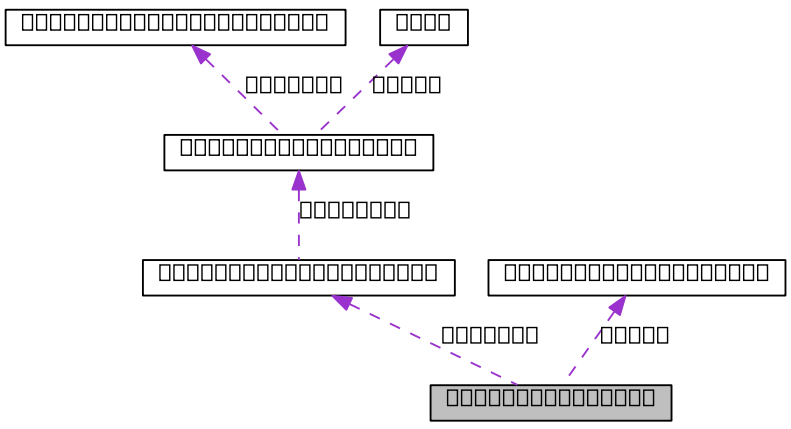
- SerialIoDevices.h

12.208 USB2_PHY_CONFIG Struct Reference

This structure holds info on how to tune electrical parameters of USB2 ports based on board layout.

`#include <Usb2PhyConfig.h>`

Collaboration diagram for USB2_PHY_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- USB2_PHY_PARAMETERS Port [MAX_USB2_PORTS]

  This structure configures per USB2 port physical settings.

12.208.1 Detailed Description

This structure holds info on how to tune electrical parameters of USB2 ports based on board layout.

Revision 1:

- Initial version.

Definition at line 88 of file Usb2PhyConfig.h.

12.208.2 Member Data Documentation

12.208.2.1 Port

```
USB2_PHY_PARAMETERS USB2_PHY_CONFIG::Port[MAX_USB2_PORTS]
```

This structure configures per USB2 port physical settings.

It allows to setup the port location and port length, and configures the port strength accordingly. Changing this policy values from default ones may require disabling USB2 PHY Sus Well Power Gating through Usb2PhySusPgEnable on PCH-LP

Definition at line 96 of file Usb2PhyConfig.h.

The documentation for this struct was generated from the following file:

- Usb2PhyConfig.h

12.209 USB2_PHY_PARAMETERS Struct Reference

This structure configures per USB2 AFE settings.

```
#include <Usb2PhyConfig.h>
```

Public Attributes

- UINT8 Petxiset

  Per Port HS Preemphasis Bias (PERPORTPETXISET) 000b - 0mV 001b - 11.25mV 010b - 16.9mV 011b - 28.15mV 100b - 28.15mV 101b - 39.35mV 110b - 45mV 111b - 56.3mV.
- UINT8 Txiset

  Per Port HS Transmitter Bias (PERPORTTXISET) 000b - 0mV 001b - 11.25mV 010b - 16.9mV 011b - 28.15mV 100b - 28.15mV 101b - 39.35mV 110b - 45mV 111b - 56.3mV.
- UINT8 Predeemp

  Per Port HS Transmitter Emphasis (IUSBTXEMPHASISEN) 00b - Emphasis OFF 01b - De-emphasis ON 10b - Pre-emphasis ON 11b - Pre-emphasis & De-emphasis ON.
- UINT8 Pehalfbit

  Per Port Half Bit Pre-emphasis (PERPORTTXPEHALF) 1b - half-bit pre-emphasis 0b - full-bit pre-emphasis.
- UINT8 PredeempSemiflexEn

  Semi Flexi Pre/DE-emphasis (reg_predeemp_semiflexi_en): 1: Choose the predeemp setting (increase the DC amplitude) 0: Choose the predeemp setting (decrease the DC amplitude)

12.209.1 Detailed Description

This structure configures per USB2 AFE settings.

It allows to setup the port electrical parameters.

Definition at line 35 of file Usb2PhyConfig.h.

The documentation for this struct was generated from the following file:

- Usb2PhyConfig.h

12.210 USB2_PORT_CONFIG Struct Reference

This structure configures per USB2.0 port settings like enabling and overcurrent protection.

`#include <UsbConfig.h>`

Public Attributes

- UINT32 OverCurrentPin: 8

  These members describe the specific over current pin number of USB 2.0 Port N.
- UINT32 Enable: 1

  *0: Disable; **1: Enable**.*
- UINT32 PortResetMessageEnable: 1

  0: Disable USB2 Port Reset Message; 1: Enable USB2 Port Reset Message
- UINT32 SwDeviceModeEnable: 1

  0: Disable USB2 Port Device Mode; 1: Force USB2 Port Device Mode
- UINT32 Usb3CompatiblePort: 3

  0: Not PCH USB3 port; 1: PCH USB3 (eSS) port
- UINT32 Usb3CompatiblePortNum: 5

  0: port1; 1: port2; 2: port3. Usb3CompatiblePort with 1 is the premise.
- UINT32 RsvdBits0: 13

  Reserved bits.

12.210.1 Detailed Description

This structure configures per USB2.0 port settings like enabling and overcurrent protection.

Definition at line 40 of file UsbConfig.h.

12.210.2 Member Data Documentation

12.210.2.1 OverCurrentPin

```
UINT32 USB2_PORT_CONFIG::OverCurrentPin
```

These members describe the specific over current pin number of USB 2.0 Port N.

It is SW's responsibility to ensure that a given port's bit map is set only for one OC pin Description. USB2 and USB3 on the same combo Port must use the same OC pin.

Definition at line 46 of file UsbConfig.h.

The documentation for this struct was generated from the following file:

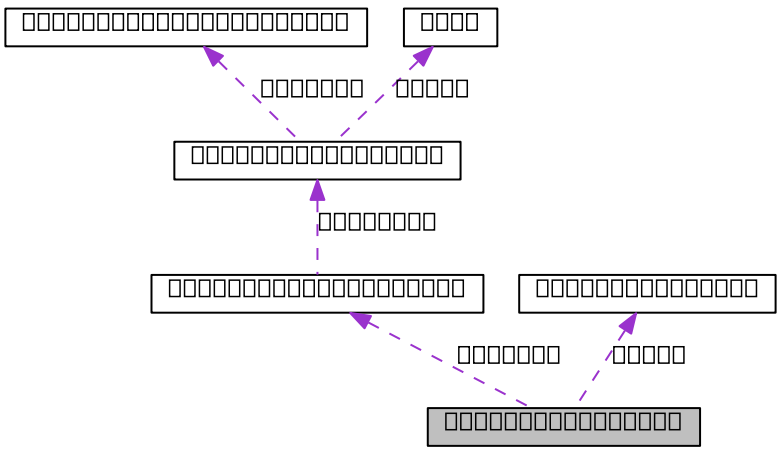
- UsbConfig.h

12.211 USB3_HSIO_CONFIG Struct Reference

Structure for holding USB3 tuning parameters.

```
#include <Usb3HsioConfig.h>
```

Collaboration diagram for USB3_HSIO_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- HSIO_PARAMETERS Port [MAX_USB3_PORTS]

  These members describe whether the USB3 Port N of PCH is enabled by platform modules.

12.211.1 Detailed Description

Structure for holding USB3 tuning parameters.

Revision 1:

- Initial version. **Revision 2**:
- USB 3.0 TX Output Unique Transition Bit Scale policies added

Definition at line 141 of file Usb3HsioConfig.h.

The documentation for this struct was generated from the following file:

- Usb3HsioConfig.h

12.212 USB3_PORT_CONFIG Struct Reference

This structure configures per USB3.x port settings like enabling and overcurrent protection.

`#include <UsbConfig.h>`

Public Attributes

- UINT32 OverCurrentPin: 8

  These members describe the specific over current pin number of USB 3.x Port N.
- UINT32 Enable: 1

  *0: Disable; **1: Enable**.*
- UINT32 RsvdBits0: 23

  Reserved bits.

12.212.1 Detailed Description

This structure configures per USB3.x port settings like enabling and overcurrent protection.

Definition at line 58 of file UsbConfig.h.

12.212.2 Member Data Documentation

12.212.2.1 OverCurrentPin

```
UINT32 USB3_PORT_CONFIG::OverCurrentPin
```

These members describe the specific over current pin number of USB 3.x Port N.

It is SW's responsibility to ensure that a given port's bit map is set only for one OC pin Description. USB2 and USB3 on the same combo Port must use the same OC pin.

Definition at line 64 of file UsbConfig.h.

The documentation for this struct was generated from the following file:

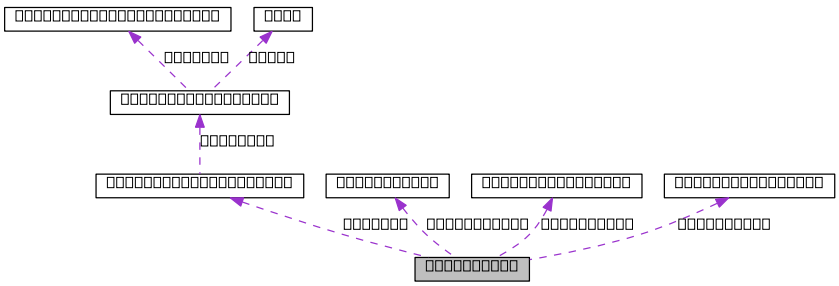
- UsbConfig.h

12.213 USB_CONFIG Struct Reference

This member describes the expected configuration of the USB controller, Platform modules may need to refer Setup options, schematic, BIOS specification to update this field.

```
#include <UsbConfig.h>
```

Collaboration diagram for USB_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT32 PdoProgramming: 1

  This policy option when set will make BIOS program Port Disable Override register during PEI phase.
- UINT32 OverCurrentEnable: 1

  This option allows for control whether USB should program the Overcurrent Pins mapping into xHCI.
- UINT32 XhciOcLock: 1

  ***(Test)** If this policy option is enabled then BIOS will program OCCFDONE bit in xHCI meaning that OC mapping data will be consumed by xHCI and OC mapping registers will be locked.*
- UINT32 LtrOverrideEnable: 1

  Enabling this feature will allow for overriding LTR values for xHCI controller.
- UINT32 UaolEnable: 1

Enable/disable option for USB Audio Offload feture.

- UINT32 HsiiEnable: 1

  *Enable HS Interrupt IN Alarm 0: disable - Disable HSII **1: enable - Enable HSII***
- UINT32 ControllerDisable: 1

  Disable Xhci Controller 0: Disable; 1: Enable.
- UINT32 DwbEnable: 1

  *Enable DWB 0: disable **1: enable***
- UINT32 RsvdBits0: 24

  Reserved bits.
- UINT32 LtrHighIdleTimeOverride

  High Idle Time Control override value This setting is used only if LtrOverrideEnable is enabled.
- UINT32 LtrMediumIdleTimeOverride

  Medium Idle Time Control override value This setting is used only if LtrOverrideEnable is enabled.
- UINT32 LtrLowIdleTimeOverride

  Low Idle Time Control override value This setting is used only if LtrOverrideEnable is enabled.
- USB2_PORT_CONFIG PortUsb20 [MAX_USB2_PORTS]

  These members describe whether the USB2 Port N of PCH is enabled by platform modules.
- USB3_PORT_CONFIG PortUsb30 [MAX_USB3_PORTS]

  These members describe whether the USB3 Port N of PCH is enabled by platform modules.
- XDCI_CONFIG XdciConfig

  This member describes whether or not the xDCI controller should be enabled.
- UINT32 Usb31Speed

  USB3.1 Speeed selection.

12.213.1 Detailed Description

This member describes the expected configuration of the USB controller, Platform modules may need to refer Setup options, schematic, BIOS specification to update this field.

The Usb20OverCurrentPins and Usb30OverCurrentPins field must be updated by referring the schematic.

Revision 1:

- Initial version. **Revision 2**:
- Added UaolEnable to control USB Audio Offload Capability. **Revision 3**:
- Add HsiiEnable enable option to control HSII feature **Revision 4**:
- Add Usb31Speed to select gen1 or gen2 speed
- Add SwDeviceModeEnable to control device mode of usb2 port **Revision 5**:
- Add DwbEnable to control the Deferred Write Buffer feature

Definition at line 99 of file UsbConfig.h.

12.213.2 Member Data Documentation

12.213.2.1 LtrOverrideEnable

```
UINT32 USB_CONFIG::LtrOverrideEnable
```

Enabling this feature will allow for overriding LTR values for xHCI controller.

Values used for programming will be taken from this config block and BIOS will disregard recommended ones. **0: disable - do not override recommended LTR values** 1: enable - override recommended LTR values

Definition at line 134 of file UsbConfig.h.

12.213.2.2 OverCurrentEnable

```
UINT32 USB_CONFIG::OverCurrentEnable
```

This option allows for control whether USB should program the Overcurrent Pins mapping into xHCI.

Disabling this feature will disable overcurrent detection functionality. Overcurrent Pin mapping data is contained in respective port structures (i.e. USB30_PORT_CONFIG) in OverCurrentPin field. By default this Overcurrent functionality should be enabled and disabled only for OBS debug usage. **1: Will program USB OC pin mapping in respective xHCI controller registers** 0: Will clear OC pin mapping allow for OBS usage of OC pins

Definition at line 117 of file UsbConfig.h.

12.213.2.3 PdoProgramming

```
UINT32 USB_CONFIG::PdoProgramming
```

This policy option when set will make BIOS program Port Disable Override register during PEI phase.

When disabled BIOS will not program the PDO during PEI phase and leave PDO register unlocked for later programming. If this is disabled, platform code MUST set it before booting into OS. **1: Enable** 0: Disable

Definition at line 108 of file UsbConfig.h.

12.213.2.4 UaolEnable

```
UINT32 USB_CONFIG::UaolEnable
```

Enable/disable option for USB Audio Offload feture.

Disabling this will disable UAOL capability in xHCI and UAOL ACPI definitions will be hidden. 0: disable UAOL **1: enable UAOL**

Definition at line 141 of file UsbConfig.h.

12.213.2.5 Usb31Speed

```
UINT32 USB_CONFIG::Usb31Speed
```

USB3.1 Speeed selection.

Each bit corresponds to a port.

Definition at line 190 of file UsbConfig.h.

12.213.2.6 XhciOcLock

```
UINT32 USB_CONFIG::XhciOcLock
```

(Test) If this policy option is enabled then BIOS will program OCCFDONE bit in xHCI meaning that OC mapping data will be consumed by xHCI and OC mapping registers will be locked.

OverCurrent mapping data is taken from respective port data structure from OverCurrentPin field. If Enable↩ OverCurrent policy is enabled this also should be enabled, otherwise xHCI won't consume OC mapping data. **1: Program OCCFDONE bit and make xHCI consume OverCurrent mapping data** 0: Do not program OCCFDONE bit making it possible to use OBS debug on OC pins.

Definition at line 127 of file UsbConfig.h.

The documentation for this struct was generated from the following file:

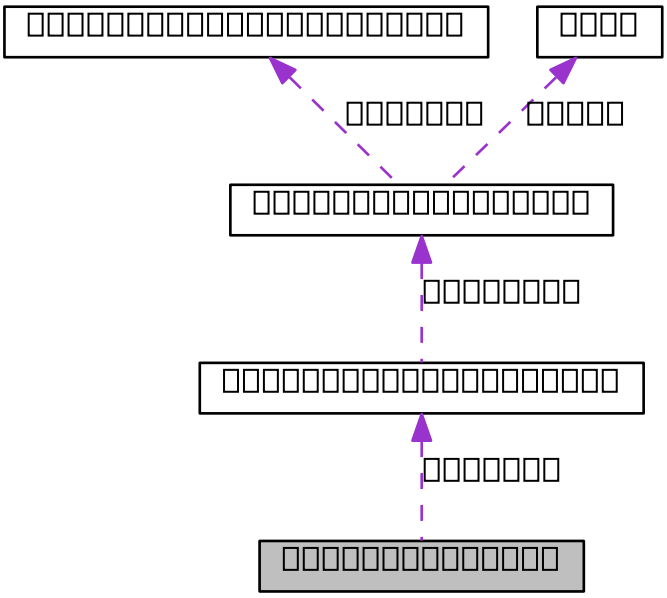
- UsbConfig.h

12.214 VMD_PEI_CONFIG Struct Reference

This configuration block is to configure VMD related variables used in PostMem PEI.

```
#include <VmdPeiConfig.h>
```

Collaboration diagram for VMD_PEI_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Offset 0-27 Config Block Header.
- UINT8 VmdEnable

  *Offset 28 This field used to enable VMD controller **1=Enable(default)** 0=Disable.*
- UINT8 VmdGlobalMapping

  *Offset 29 This field used to enable Global Mapping 1=Enable **0=Disable(default)***
- UINT8 RsvdBytes51 [2]

  Offset 30-39 Reserved.
- RP_BDF_DATA VmdPortEnable [VMD_MAX_DEVICES]

  *Offset 40 to 163 This field used to to store b/d/f for each root port along with enable Support 1=Enable **0=Disable (default)***
- UINT32 VmdCfgBarBase

  *Temp Address VMD CFG BAR Default is **0xA0000000***
- VOID * VmdVariablePtr

  This config block will be updated as per the EFI variable.
- UINT32 VmdMemBar1Base

  *Temp Address VMD CFG BAR Default is **0xA2000000***
- UINT32 VmdMemBar2Base

  *Temp Address VMD CFG BAR Default is **0xA4000000***

12.214.1 Detailed Description

This configuration block is to configure VMD related variables used in PostMem PEI.

If VMD Device is not supported, all policies can be ignored. **Revision 1**:

- Initial version.

Definition at line 47 of file VmdPeiConfig.h.

The documentation for this struct was generated from the following file:

- VmdPeiConfig.h

12.215 XDCI_CONFIG Struct Reference

The XDCI_CONFIG block describes the configurations of the xDCI Usb Device controller.

`#include <UsbConfig.h>`

Public Attributes

- UINT32 Enable: 1
  This member describes whether or not the xDCI controller should be enabled.
- UINT32 RsvdBits0: 31
  Reserved bits.

12.215.1 Detailed Description

The XDCI_CONFIG block describes the configurations of the xDCI Usb Device controller.

Definition at line 73 of file UsbConfig.h.

12.215.2 Member Data Documentation

12.215.2.1 Enable

`UINT32 XDCI_CONFIG::Enable`

This member describes whether or not the xDCI controller should be enabled.

0: Disable; **1: Enable**.

Definition at line 78 of file UsbConfig.h.

The documentation for this struct was generated from the following file:

- UsbConfig.h

Chapter 13

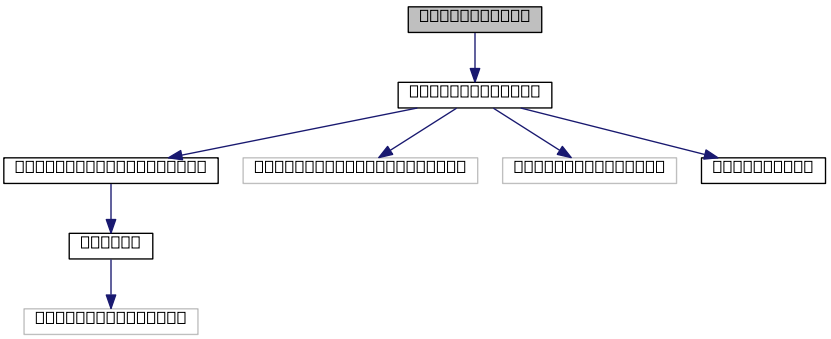
File Documentation

13.1 AdrConfig.h File Reference

ADR policy.

```
#include <ConfigBlock.h>
```

Include dependency graph for AdrConfig.h:



Classes

- union ADR_GLOBAL_RESET_ENABLE

  ADR Source Enable.
- struct ADR_CONFIG

  ADR Configuration ***Revision 1****: - Initial version.*

13.1.1 Detailed Description

ADR policy.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2019 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

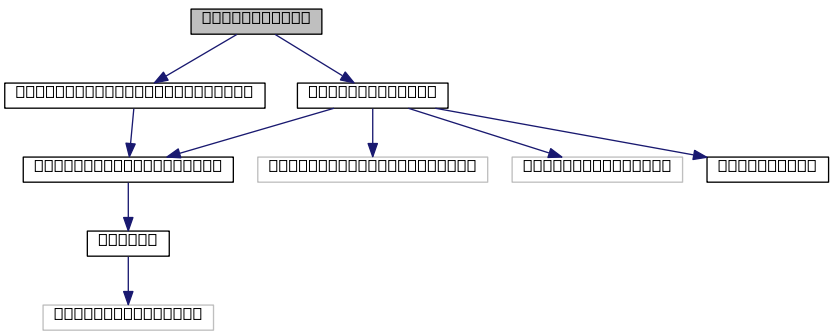
Specification Reference:

13.2 AmtConfig.h File Reference

AMT Config Block for PEI/DXE phase.

```
#include <Library/ConfigBlockLib.h>
#include <ConfigBlock.h>
```

Include dependency graph for AmtConfig.h:



Classes

- struct AMT_PEI_CONFIG

  AMT Pei Configuration Structure.
- struct AMT_DXE_CONFIG

  AMT Dxe Configuration Structure.

13.2.1 Detailed Description

AMT Config Block for PEI/DXE phase.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2015 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

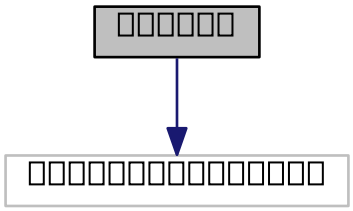
Specification Reference:

13.3 Base.h File Reference

Root include file for Mde Package Base type modules.

```
#include <ProcessorBind.h>
```

Include dependency graph for Base.h:



This graph shows which files directly or indirectly include this file:

Classes

- struct GUID

  128 bit buffer containing a unique identifier value.
- struct IPv4_ADDRESS

  4-byte buffer.
- struct IPv6_ADDRESS

  16-byte buffer.
- struct _LIST_ENTRY

  _LIST_ENTRY structure definition.

Macros

- #define GLOBAL_REMOVE_IF_UNREFERENCED

  Remove the global variable from the linked image if there are no references to it after all compiler and linker optimizations have been performed.
- #define UNREACHABLE()

  Signal compilers and analyzers that this call is not reachable.
- #define NORETURN

  Signal compilers and analyzers that the function cannot return.
- #define ANALYZER_UNREACHABLE()

  Signal the analyzer that this call is not reachable.
- #define ANALYZER_NORETURN

  Signal the analyzer that the function cannot return.
- #define RETURNS_TWICE

  Tell the code optimizer that the function will return twice.
- #define _CONCATENATE(a, b) __CONCATENATE(a, b)

  Private worker functions for ASM_PFX()
- #define ASM_PFX(name) _CONCATENATE (__USER_LABEL_PREFIX__, name)

  *The **USER_LABEL_PREFIX** macro predefined by GNUC represents the prefix on symbols in assembly language.*
- #define CONST const

  Datum is read-only.
- #define STATIC static

  Datum is scoped to the current file or function.
- #define VOID void

  Undeclared type.
- #define IN

  Datum is passed to the function.
- #define OUT

  Datum is returned from the function.
- #define OPTIONAL

  Passing the datum to the function is optional, and a NULL is passed if the value is not supplied.
- #define TRUE ((BOOLEAN)(1==1))

  Boolean true value.
- #define FALSE ((BOOLEAN)(0==1))

  Boolean false value.
- #define NULL ((VOID *) 0)

  *NULL pointer (VOID *)*
- #define MAX_INT8 ((INT8)0x7F)

  Maximum values for common UEFI Data Types.

- #define MIN_INT8 (((INT8) -127) - 1)

  Minimum values for the signed UEFI Data Types.
- #define _INT_SIZE_OF(n) ((sizeof (n) + sizeof (UINTN) - 1) &∼(sizeof (UINTN) - 1))

  Return the size of argument that has been aligned to sizeof (UINTN).
- #define VA_START(Marker, Parameter) (Marker = (VA_LIST) ((UINTN) & (Parameter) + _INT_SIZE_OF (Parameter)))

  Retrieves a pointer to the beginning of a variable argument list, based on the name of the parameter that immediately precedes the variable argument list.
- #define VA_ARG(Marker, TYPE) (∗(TYPE ∗) ((Marker += _INT_SIZE_OF (TYPE)) - _INT_SIZE_OF (TYPE)))

  Returns an argument of a specified type from a variable argument list and updates the pointer to the variable argument list to point to the next argument.
- #define VA_END(Marker) (Marker = (VA_LIST) 0)

  Terminates the use of a variable argument list.
- #define VA_COPY(Dest, Start) ((void)((Dest) = (Start)))

  Initializes a VA_LIST as a copy of an existing VA_LIST.
- #define _BASE_INT_SIZE_OF(TYPE) ((sizeof (TYPE) + sizeof (UINTN) - 1) / sizeof (UINTN))

  Returns the size of a data type in sizeof(UINTN) units rounded up to the nearest UINTN boundary.
- #define BASE_ARG(Marker, TYPE) (∗(TYPE ∗) ((Marker += _BASE_INT_SIZE_OF (TYPE)) - _BASE_INT_SIZE_OF (TYPE)))

  Returns an argument of a specified type from a variable argument list and updates the pointer to the variable argument list to point to the next argument.
- #define OFFSET_OF(TYPE, Field) ((UINTN) &(((TYPE ∗)0)->Field))

  The macro that returns the byte offset of a field in a data structure.
- #define ALIGNOF(TYPE) OFFSET_OF (struct { CHAR8 C; TYPE A; }, A)

  Returns the alignment requirement of a type.
- #define BASE_CR(Record, TYPE, Field) ((TYPE ∗) (VOID ∗) ((CHAR8 ∗) (Record) - OFFSET_OF (TYPE, Field)))

  Macro that returns a pointer to the data structure that contains a specified field of that data structure.
- #define IS_POW2(Value) ((Value) != 0U && ((Value) & ((Value) - 1U)) == 0U)

  Checks whether a value is a power of two.
- #define IS_ALIGNED(Value, Alignment) (((Value) & ((Alignment) - 1U)) == 0U)

  Checks whether a value is aligned by a specified alignment.
- #define ADDRESS_IS_ALIGNED(Address, Alignment) IS_ALIGNED ((UINTN) (Address), Alignment)

  Checks whether a pointer or address is aligned by a specified alignment.
- #define ALIGN_VALUE_ADDEND(Value, Alignment) (((Alignment) - (Value)) & ((Alignment) - 1U))

  Determines the addend to add to a value to round it up to the next boundary of a specified alignment.
- #define ALIGN_VALUE(Value, Alignment) ((Value) + ALIGN_VALUE_ADDEND (Value, Alignment))

  Rounds a value up to the next boundary using a specified alignment.
- #define ALIGN_POINTER(Pointer, Alignment) ((VOID ∗) (ALIGN_VALUE ((UINTN)(Pointer), (Alignment))))

  Adjust a pointer by adding the minimum offset required for it to be aligned on a specified alignment boundary.
- #define ALIGN_VARIABLE(Value) ALIGN_VALUE ((Value), sizeof (UINTN))

  Rounds a value up to the next natural boundary for the current CPU.
- #define MAX(a, b) (((a) > (b)) ? (a) : (b))

  Return the maximum of two operands.
- #define MIN(a, b) (((a) < (b)) ? (a) : (b))

  Return the minimum of two operands.
- #define ABS(a) (((a) < 0) ? (-(a)) : (a))

  Return the absolute value of a signed operand.
- #define ENCODE_ERROR(StatusCode) ((RETURN_STATUS)(MAX_BIT | (StatusCode)))

  Produces a RETURN_STATUS code with the highest bit set.
- #define ENCODE_WARNING(StatusCode) ((RETURN_STATUS)(StatusCode))

  Produces a RETURN_STATUS code with the highest bit clear.

- #define RETURN_ERROR(StatusCode) (((RETURN_STATUS)(StatusCode)) >= MAX_BIT)

  Returns TRUE if a specified RETURN_STATUS code is an error code.
- #define RETURN_SUCCESS (RETURN_STATUS)(0)

  The operation completed successfully.
- #define RETURN_LOAD_ERROR ENCODE_ERROR (1)

  The image failed to load.
- #define RETURN_INVALID_PARAMETER ENCODE_ERROR (2)

  The parameter was incorrect.
- #define RETURN_UNSUPPORTED ENCODE_ERROR (3)

  The operation is not supported.
- #define RETURN_BAD_BUFFER_SIZE ENCODE_ERROR (4)

  The buffer was not the proper size for the request.
- #define RETURN_BUFFER_TOO_SMALL ENCODE_ERROR (5)

  The buffer was not large enough to hold the requested data.
- #define RETURN_NOT_READY ENCODE_ERROR (6)

  There is no data pending upon return.
- #define RETURN_DEVICE_ERROR ENCODE_ERROR (7)

  The physical device reported an error while attempting the operation.
- #define RETURN_WRITE_PROTECTED ENCODE_ERROR (8)

  The device can not be written to.
- #define RETURN_OUT_OF_RESOURCES ENCODE_ERROR (9)

  The resource has run out.
- #define RETURN_VOLUME_CORRUPTED ENCODE_ERROR (10)

  An inconsistency was detected on the file system causing the operation to fail.
- #define RETURN_VOLUME_FULL ENCODE_ERROR (11)

  There is no more space on the file system.
- #define RETURN_NO_MEDIA ENCODE_ERROR (12)

  The device does not contain any medium to perform the operation.
- #define RETURN_MEDIA_CHANGED ENCODE_ERROR (13)

  The medium in the device has changed since the last access.
- #define RETURN_NOT_FOUND ENCODE_ERROR (14)

  The item was not found.
- #define RETURN_ACCESS_DENIED ENCODE_ERROR (15)

  Access was denied.
- #define RETURN_NO_RESPONSE ENCODE_ERROR (16)

  The server was not found or did not respond to the request.
- #define RETURN_NO_MAPPING ENCODE_ERROR (17)

  A mapping to the device does not exist.
- #define RETURN_TIMEOUT ENCODE_ERROR (18)

  A timeout time expired.
- #define RETURN_NOT_STARTED ENCODE_ERROR (19)

  The protocol has not been started.
- #define RETURN_ALREADY_STARTED ENCODE_ERROR (20)

  The protocol has already been started.
- #define RETURN_ABORTED ENCODE_ERROR (21)

  The operation was aborted.
- #define RETURN_ICMP_ERROR ENCODE_ERROR (22)

  An ICMP error occurred during the network operation.
- #define RETURN_TFTP_ERROR ENCODE_ERROR (23)

  A TFTP error occurred during the network operation.
- #define RETURN_PROTOCOL_ERROR ENCODE_ERROR (24)

A protocol error occurred during the network operation.

- #define RETURN_INCOMPATIBLE_VERSION ENCODE_ERROR (25)

  A function encountered an internal version that was incompatible with a version requested by the caller.
- #define RETURN_SECURITY_VIOLATION ENCODE_ERROR (26)

  The function was not performed due to a security violation.
- #define RETURN_CRC_ERROR ENCODE_ERROR (27)

  A CRC error was detected.
- #define RETURN_END_OF_MEDIA ENCODE_ERROR (28)

  The beginning or end of media was reached.
- #define RETURN_END_OF_FILE ENCODE_ERROR (31)

  The end of the file was reached.
- #define RETURN_INVALID_LANGUAGE ENCODE_ERROR (32)

  The language specified was invalid.
- #define RETURN_COMPROMISED_DATA ENCODE_ERROR (33)

  The security status of the data is unknown or compromised and the data must be updated or replaced to restore a valid security status.
- #define RETURN_IP_ADDRESS_CONFLICT ENCODE_ERROR (34)

  There is an address conflict address allocation.
- #define RETURN_HTTP_ERROR ENCODE_ERROR (35)

  A HTTP error occurred during the network operation.
- #define RETURN_WARN_UNKNOWN_GLYPH ENCODE_WARNING (1)

  The string contained one or more characters that the device could not render and were skipped.
- #define RETURN_WARN_DELETE_FAILURE ENCODE_WARNING (2)

  The handle was closed, but the file was not deleted.
- #define RETURN_WARN_WRITE_FAILURE ENCODE_WARNING (3)

  The handle was closed, but the data to the file was not flushed properly.
- #define RETURN_WARN_BUFFER_TOO_SMALL ENCODE_WARNING (4)

  The resulting buffer was too small, and the data was truncated to the buffer size.
- #define RETURN_WARN_STALE_DATA ENCODE_WARNING (5)

  The data has not been updated within the timeframe set by local policy for this type of data.
- #define RETURN_WARN_FILE_SYSTEM ENCODE_WARNING (6)

  The resulting buffer contains UEFI-compliant file system.
- #define RETURN_WARN_RESET_REQUIRED ENCODE_WARNING (7)

  The operation will be processed across a system reset.
- #define SIGNATURE_16(A, B) ((A) | (B << 8))

  Returns a 16-bit signature built from 2 ASCII characters.
- #define SIGNATURE_32(A, B, C, D) (SIGNATURE_16 (A, B) | (SIGNATURE_16 (C, D) << 16))

  Returns a 32-bit signature built from 4 ASCII characters.
- #define SIGNATURE_64(A, B, C, D, E, F, G, H) (SIGNATURE_32 (A, B, C, D) | ((UINT64) (SIGNATURE_32 (E, F, G, H)) << 32))

  Returns a 64-bit signature built from 8 ASCII characters.
- #define RETURN_ADDRESS(L) ((VOID *) 0)

  Get the return address of the calling function.
- #define ARRAY_SIZE(Array) (sizeof (Array) / sizeof ((Array)[0]))

  Return the number of elements in an array.

Typedefs

- typedef struct _LIST_ENTRY LIST_ENTRY

  LIST_ENTRY structure definition.
- typedef CHAR8 * VA_LIST

  Variable used to traverse the list of arguments.
- typedef UINTN * BASE_LIST

  Pointer to the start of a variable argument list stored in a memory buffer.

Functions

- STATIC_ASSERT (sizeof(BOOLEAN)==1, "sizeof (BOOLEAN) does not meet UEFI Specification Data Type requirements")

  Portable definition for compile time assertions.

13.3.1 Detailed Description

Root include file for Mde Package Base type modules.

This is the include file for any module of type base. Base modules only use types defined via this include file and can be ported easily to any environment. There are a set of base libraries in the Mde Package that can be used to implement base modules.

Copyright (c) 2006 - 2021, Intel Corporation. All rights reserved.
Portions copyright (c) 2008 - 2009, Apple Inc. All rights reserved.
SPDX-License-Identifier: BSD-2-Clause-Patent

13.3.2 Macro Definition Documentation

13.3.2.1 _BASE_INT_SIZE_OF

```
#define _BASE_INT_SIZE_OF(
              TYPE ) ((sizeof (TYPE) + sizeof (UINTN) - 1) / sizeof (UINTN))
```

Returns the size of a data type in sizeof(UINTN) units rounded up to the nearest UINTN boundary.

Parameters

TYPE	The date type to determine the size of.

Returns

The size of TYPE in sizeof (UINTN) units rounded up to the nearest UINTN boundary.

Definition at line 720 of file Base.h.

13.3.2.2 _INT_SIZE_OF

```
#define _INT_SIZE_OF(
              n ) ((sizeof (n) + sizeof (UINTN) - 1) &~(sizeof (UINTN) - 1))
```

Return the size of argument that has been aligned to sizeof (UINTN).

Parameters

n	The parameter size to be aligned.

Returns

The aligned size.

Definition at line 579 of file Base.h.

13.3.2.3 ABS

```
#define ABS(
              a )  (((a) < 0) ?  (-(a)) :  (a))
```

Return the absolute value of a signed operand.

This macro returns the absolute value of the signed operand specified by a.

Parameters

a	The signed operand.

Returns

The absolute value of the signed operand.

Definition at line 1020 of file Base.h.

13.3.2.4 ADDRESS_IS_ALIGNED

```
#define ADDRESS_IS_ALIGNED(
              Address,
              Alignment ) IS_ALIGNED ((UINTN) (Address), Alignment)
```

Checks whether a pointer or address is aligned by a specified alignment.

Parameters

Address	The pointer or address to check.
Alignment	The alignment boundary used to check against.

Return values

TRUE	Address is aligned by Alignment.

Return values

FALSE	Address is not aligned by Alignment.

Definition at line 923 of file Base.h.

13.3.2.5 ALIGN_POINTER

```
#define ALIGN_POINTER(
              Pointer,
              Alignment ) ((VOID *) (ALIGN_VALUE ((UINTN)(Pointer), (Alignment))))
```

Adjust a pointer by adding the minimum offset required for it to be aligned on a specified alignment boundary.

This function rounds the pointer specified by Pointer to the next alignment boundary specified by Alignment. The pointer to the aligned address is returned.

Parameters

Pointer	The pointer to round up.
Alignment	The alignment boundary to use to return an aligned pointer.

Returns

Pointer to the aligned address.

Definition at line 963 of file Base.h.

13.3.2.6 ALIGN_VALUE

```
#define ALIGN_VALUE(
              Value,
              Alignment ) ((Value) + ALIGN_VALUE_ADDEND (Value, Alignment))
```

Rounds a value up to the next boundary using a specified alignment.

This function rounds Value up to the next boundary using the specified Alignment. This aligned value is returned.

Parameters

Value	The value to round up.
Alignment	The alignment boundary used to return the aligned value.

Returns

A value up to the next boundary.

Definition at line 948 of file Base.h.

13.3.2.7 ALIGN_VALUE_ADDEND

```
#define ALIGN_VALUE_ADDEND(
              Value,
              Alignment ) (((Alignment) - (Value)) & ((Alignment) - 1U))
```

Determines the addend to add to a value to round it up to the next boundary of a specified alignment.

Parameters

Value	The value to round up.
Alignment	The alignment boundary used to return the addend.

Returns

Addend to round Value up to alignment boundary Alignment.

Definition at line 934 of file Base.h.

13.3.2.8 ALIGN_VARIABLE

```
#define ALIGN_VARIABLE(
              Value ) ALIGN_VALUE ((Value), sizeof (UINTN))
```

Rounds a value up to the next natural boundary for the current CPU.

This is 4-bytes for 32-bit CPUs and 8-bytes for 64-bit CPUs.

This function rounds the value specified by Value up to the next natural boundary for the current CPU. This rounded value is returned.

Parameters

Value	The value to round up.

Returns

Rounded value specified by Value.

Definition at line 977 of file Base.h.

13.3.2.9 ALIGNOF

```
#define ALIGNOF(
              TYPE ) OFFSET_OF (struct { CHAR8 C; TYPE A; }, A)
```

Returns the alignment requirement of a type.

Parameters

TYPE	The name of the type to retrieve the alignment requirement of.

Returns

Alignment requirement, in Bytes, of TYPE.

Definition at line 792 of file Base.h.

13.3.2.10 ANALYZER_NORETURN

```
#define ANALYZER_NORETURN
```

Signal the analyzer that the function cannot return.

This excludes compilers.

Definition at line 157 of file Base.h.

13.3.2.11 ANALYZER_UNREACHABLE

```
#define ANALYZER_UNREACHABLE( )
```

Signal the analyzer that this call is not reachable.

This excludes compilers.

Definition at line 131 of file Base.h.

13.3.2.12 ARRAY_SIZE

```
#define ARRAY_SIZE(
              Array ) (sizeof (Array) / sizeof ((Array)[0]))
```

Return the number of elements in an array.

Parameters

Array	An object of array type. Array is only used as an argument to the sizeof operator, therefore Array is never evaluated. The caller is responsible for ensuring that Array's type is not incomplete; that is, Array must have known constant size.

Returns

The number of elements in Array. The result has type UINTN.

Definition at line 1393 of file Base.h.

13.3.2.13 BASE_ARG

```
#define BASE_ARG(
              Marker,
              TYPE ) (*(TYPE *) ((Marker += _BASE_INT_SIZE_OF (TYPE)) - _BASE_INT_SIZE_OF (TY↩
PE)))
```

Returns an argument of a specified type from a variable argument list and updates the pointer to the variable argument list to point to the next argument.

This function returns an argument of the type specified by TYPE from the beginning of the variable argument list specified by Marker. Marker is then updated to point to the next argument in the variable argument list. The method for computing the pointer to the next argument in the argument list is CPU specific following the EFIAPI ABI.

Parameters

Marker	The pointer to the beginning of a variable argument list.
TYPE	The type of argument to retrieve from the beginning of the variable argument list.

Returns

An argument of the type specified by TYPE.

Definition at line 738 of file Base.h.

13.3.2.14 BASE_CR

```
#define BASE_CR(
              Record,
              TYPE,
              Field ) ((TYPE *) (VOID *) ((CHAR8 *) (Record) - OFFSET_OF (TYPE, Field)))
```

Macro that returns a pointer to the data structure that contains a specified field of that data structure.

This is a lightweight method to hide information by placing a public data structure inside a larger private data structure and using a pointer to the public data structure to retrieve a pointer to the private data structure.

This function computes the offset, in bytes, of field specified by Field from the beginning of the data structure specified by TYPE. This offset is subtracted from Record, and is used to return a pointer to a data structure of the type specified by TYPE. If the data type specified by TYPE does not contain the field specified by Field, then the module will not compile.

Parameters

Record	Pointer to the field specified by Field within a data structure of type TYPE.
TYPE	The name of the data structure type to return. This data structure must contain the field specified by Field.
Field	The name of the field in the data structure specified by TYPE to which Record points.

Returns

A pointer to the structure from one of it's elements.

Definition at line 891 of file Base.h.

13.3.2.15 ENCODE_ERROR

```
#define ENCODE_ERROR(
              StatusCode ) ((RETURN_STATUS)(MAX_BIT | (StatusCode)))
```

Produces a RETURN_STATUS code with the highest bit set.

Parameters

StatusCode	The status code value to convert into a warning code. StatusCode must be in the range 0x00000000..0x7FFFFFFF.

Returns

The value specified by StatusCode with the highest bit set.

Definition at line 1037 of file Base.h.

13.3.2.16 ENCODE_WARNING

```
#define ENCODE_WARNING(
              StatusCode ) ((RETURN_STATUS)(StatusCode))
```

Produces a RETURN_STATUS code with the highest bit clear.

Parameters

StatusCode	The status code value to convert into a warning code. StatusCode must be in the range 0x00000000..0x7FFFFFFF.

Returns

The value specified by StatusCode with the highest bit clear.

Definition at line 1048 of file Base.h.

13.3.2.17 FALSE

```
#define FALSE ((BOOLEAN)(0==1))
```

Boolean false value.

UEFI Specification defines this value to be 0, but this form is more portable.

Definition at line 307 of file Base.h.

13.3.2.18 IS_ALIGNED

```
#define IS_ALIGNED(
              Value,
              Alignment ) (((Value) & ((Alignment) - 1U)) == 0U)
```

Checks whether a value is aligned by a specified alignment.

Parameters

Value	The value to check.
Alignment	The alignment boundary used to check against.

Return values

TRUE	Value is aligned by Alignment.
FALSE	Value is not aligned by Alignment.

Definition at line 912 of file Base.h.

13.3.2.19 IS_POW2

```
#define IS_POW2(
              Value ) ((Value) != 0U && ((Value) & ((Value) - 1U)) == 0U)
```

Checks whether a value is a power of two.

Parameters

Value	The value to check.

Return values

TRUE	Value is a power of two.
FALSE	Value is not a power of two.

Definition at line 901 of file Base.h.

13.3.2.20 MAX

```
#define MAX(
              a,
              b )  (((a) > (b)) ?  (a) :  (b))
```

Return the maximum of two operands.

This macro returns the maximum of two operand specified by a and b. Both a and b must be the same numerical types, signed or unsigned.

Parameters

a	The first operand with any numerical type.
b	The second operand. Can be any numerical type as long as is the same type as a.

Returns

Maximum of two operands.

Definition at line 992 of file Base.h.

13.3.2.21 MIN

```
#define MIN(
              a,
              b )  (((a) < (b)) ?  (a) :  (b))
```

Return the minimum of two operands.

This macro returns the minimal of two operand specified by a and b. Both a and b must be the same numerical types, signed or unsigned.

Parameters

a	The first operand with any numerical type.
b	The second operand. It should be the same any numerical type with a.

Returns

Minimum of two operands.

Definition at line 1007 of file Base.h.

13.3.2.22 NORETURN

```
#define NORETURN
```

Signal compilers and analyzers that the function cannot return.

It is up to the compiler to remove any code past a call to functions flagged with this attribute.

Definition at line 107 of file Base.h.

13.3.2.23 OFFSET_OF

```
#define OFFSET_OF(
              TYPE,
              Field ) ((UINTN) &(((TYPE *)0)->Field))
```

The macro that returns the byte offset of a field in a data structure.

This function returns the offset, in bytes, of field specified by Field from the beginning of the data structure specified by TYPE. If TYPE does not contain Field, the module will not compile.

Parameters

TYPE	The name of the data structure that contains the field specified by Field.
Field	The name of the field in the data structure.

Returns

Offset, in bytes, of field.

Definition at line 758 of file Base.h.

13.3.2.24 RETURN_ADDRESS

```
#define RETURN_ADDRESS(
              L ) ((VOID *) 0)
```

Get the return address of the calling function.

Parameters

L	Return Level.

Returns

0 as compilers don't support this feature.

Definition at line 1379 of file Base.h.

13.3.2.25 RETURN_BUFFER_TOO_SMALL

```
#define RETURN_BUFFER_TOO_SMALL ENCODE_ERROR (5)
```

The buffer was not large enough to hold the requested data.

The required buffer size is returned in the appropriate parameter when this error occurs.

Definition at line 1093 of file Base.h.

13.3.2.26 RETURN_ERROR

```
#define RETURN_ERROR(
              StatusCode ) (((RETURN_STATUS)(StatusCode)) >= MAX_BIT)
```

Returns TRUE if a specified RETURN_STATUS code is an error code.

This function returns TRUE if StatusCode has the high bit set. Otherwise, FALSE is returned.

Parameters

StatusCode	The status code value to evaluate.

Return values

TRUE	The high bit of StatusCode is set.
FALSE	The high bit of StatusCode is clear.

Definition at line 1061 of file Base.h.

13.3.2.27 RETURNS_TWICE

```
#define RETURNS_TWICE
```

Tell the code optimizer that the function will return twice.

This prevents wrong optimizations which can cause bugs. Tell the code optimizer that the function will return twice. This prevents wrong optimizations which can cause bugs.

Definition at line 177 of file Base.h.

13.3.2.28 SIGNATURE_16

```
#define SIGNATURE_16(
              A,
              B ) ((A) | (B << 8))
```

Returns a 16-bit signature built from 2 ASCII characters.

This macro returns a 16-bit value built from the two ASCII characters specified by A and B.

Parameters

A	The first ASCII character.
B	The second ASCII character.

Returns

A 16-bit value built from the two ASCII characters specified by A and B.

Definition at line 1293 of file Base.h.

13.3.2.29 SIGNATURE_32

```
#define SIGNATURE_32(
              A,
              B,
              C,
              D ) (SIGNATURE_16 (A, B) | (SIGNATURE_16 (C, D) << 16))
```

Returns a 32-bit signature built from 4 ASCII characters.

This macro returns a 32-bit value built from the four ASCII characters specified by A, B, C, and D.

Parameters

A	The first ASCII character.
B	The second ASCII character.
C	The third ASCII character.
D	The fourth ASCII character.

Returns

A 32-bit value built from the two ASCII characters specified by A, B, C and D.

Definition at line 1310 of file Base.h.

13.3.2.30 SIGNATURE_64

```
#define SIGNATURE_64(
              A,
              B,
              C,
              D,
              E,
              F,
              G,
              H )  (SIGNATURE_32 (A, B, C, D) | ((UINT64) (SIGNATURE_32 (E, F, G, H)) << 32))
```

Returns a 64-bit signature built from 8 ASCII characters.

This macro returns a 64-bit value built from the eight ASCII characters specified by A, B, C, D, E, F, G,and H.

Parameters

A	The first ASCII character.
B	The second ASCII character.
C	The third ASCII character.
D	The fourth ASCII character.
E	The fifth ASCII character.
F	The sixth ASCII character.
G	The seventh ASCII character.
H	The eighth ASCII character.

Returns

A 64-bit value built from the two ASCII characters specified by A, B, C, D, E, F, G and H.

Definition at line 1331 of file Base.h.

13.3.2.31 TRUE

```
#define TRUE ((BOOLEAN)(1==1))
```

Boolean true value.

UEFI Specification defines this value to be 1, but this form is more portable.

Definition at line 301 of file Base.h.

13.3.2.32 UNREACHABLE

```
#define UNREACHABLE( )
```

Signal compilers and analyzers that this call is not reachable.

It is up to the compiler to remove any code past that point.

Definition at line 77 of file Base.h.

13.3.2.33 VA_ARG

```
#define VA_ARG(
              Marker,
              TYPE ) (*(TYPE *) ((Marker += _INT_SIZE_OF (TYPE)) - _INT_SIZE_OF (TYPE)))
```

Returns an argument of a specified type from a variable argument list and updates the pointer to the variable argument list to point to the next argument.

This function returns an argument of the type specified by TYPE from the beginning of the variable argument list specified by Marker. Marker is then updated to point to the next argument in the variable argument list. The method for computing the pointer to the next argument in the argument list is CPU-specific following the EFIAPI ABI.

Parameters

Marker	VA_LIST used to traverse the list of arguments.
TYPE	The type of argument to retrieve from the beginning of the variable argument list.

Returns

An argument of the type specified by TYPE.

Definition at line 679 of file Base.h.

13.3.2.34 VA_COPY

```
#define VA_COPY(
              Dest,
              Start ) ((void)((Dest) = (Start)))
```

Initializes a VA_LIST as a copy of an existing VA_LIST.

This macro initializes Dest as a copy of Start, as if the VA_START macro had been applied to Dest followed by the same sequence of uses of the VA_ARG macro as had previously been used to reach the present state of Start.

Parameters

Dest	VA_LIST used to traverse the list of arguments.
Start	VA_LIST used to traverse the list of arguments.

Definition at line 704 of file Base.h.

13.3.2.35 VA_END

```
#define VA_END(
              Marker ) (Marker = (VA_LIST) 0)
```

Terminates the use of a variable argument list.

This function initializes Marker so it can no longer be used with VA_ARG(). After this macro is used, the only way to access the variable argument list is by using VA_START() again.

Parameters

Marker	VA_LIST used to traverse the list of arguments.

Definition at line 691 of file Base.h.

13.3.2.36 VA_START

```
#define VA_START(
              Marker,
              Parameter ) (Marker = (VA_LIST) ((UINTN) & (Parameter) + _INT_SIZE_OF (Parameter)))
```

Retrieves a pointer to the beginning of a variable argument list, based on the name of the parameter that immediately precedes the variable argument list.

This function initializes Marker to point to the beginning of the variable argument list that immediately follows Parameter. The method for computing the pointer to the next argument in the argument list is CPU-specific following the EFIAPI ABI.

Parameters

Marker	The VA_LIST used to traverse the list of arguments.
Parameter	The name of the parameter that immediately precedes the variable argument list.

Returns

A pointer to the beginning of a variable argument list.

Definition at line 661 of file Base.h.

13.3.3 Typedef Documentation

13.3.3.1 BASE_LIST

```
typedef UINTN* BASE_LIST
```

Pointer to the start of a variable argument list stored in a memory buffer.

Same as UINT8 *.

Definition at line 711 of file Base.h.

13.3.3.2 VA_LIST

```
typedef CHAR8* VA_LIST
```

Variable used to traverse the list of arguments.

This type can vary by implementation and could be an array or structure.

Definition at line 643 of file Base.h.

13.3.4 Function Documentation

13.3.4.1 STATIC_ASSERT()

```
STATIC_ASSERT (
             sizeof(BOOLEAN)  == 1,
             "sizeof (BOOLEAN) does not meet UEFI Specification Data Type requirements"  )
```

Portable definition for compile time assertions.

Equivalent to C11 static_assert macro from assert.h.

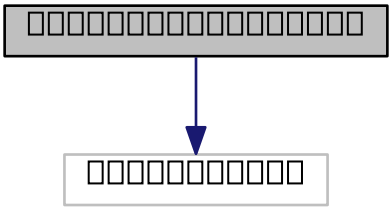
Parameters

Expression	Boolean expression.
Message	Raised compiler diagnostic message when expression is false.

13.4 BiosGuardConfig.h File Reference

CPU BIOS Guard Config Block.

`#include <BiosGuard.h>`
Include dependency graph for BiosGuardConfig.h:



Classes

- struct BIOS_GUARD_CONFIG

  BIOS Guard Configuration Structure.

Macros

- #define BIOS_GUARD_CONFIG_REVISION 1

  BIOS Guard 2.2 Configuration Revision.
- #define BIOSGUARD_TOTAL_HASH_MSRS 6

  Total HW HASH MSRS supported for BIOS Guard 2.2.
- #define BIOSGUARD_MAX_HASH_SIZE 48

  Maximum size for SHA384 HASH for BIOS Guard 2.2.

Typedefs

- typedef EFI_STATUS(∗ EC_PROVISION_EAV) (IN UINT32 Eav, OUT UINT8 ∗ReturnValue)

  Security function for BIOS guard: EC_C_PFAT_PROVISION_EAV.
- typedef EFI_STATUS(∗ EC_CMD_LOCK) (OUT UINT8 ∗ReturnValue)

  Security function for BIOS guard: EC_C_PFAT_CMD_LOCK.

Variables

- EFI_GUID gBiosGuardConfigGuid
  BIOS Guard Configuration GUID.

13.4.1 Detailed Description

CPU BIOS Guard Config Block.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2022 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.4.2 Typedef Documentation

13.4.2.1 EC_CMD_LOCK

```
typedef EFI_STATUS( * EC_CMD_LOCK) (OUT UINT8 *ReturnValue)
```

Security function for BIOS guard: EC_C_PFAT_CMD_LOCK.

Parameters

out	*ReturnValue*	The return value from EC FW.

Return values

EFI_SUCCESS	Command success
EFI_DEVICE_ERROR	Command error

Definition at line 80 of file BiosGuardConfig.h.

13.4.2.2 EC_PROVISION_EAV

```
typedef EFI_STATUS( * EC_PROVISION_EAV) (IN UINT32 Eav, OUT UINT8 *ReturnValue)
```

Security function for BIOS guard: EC_C_PFAT_PROVISION_EAV.

Parameters

out	*ReturnValue*	The return value from EC FW for Ephemeral Auth Value.

Return values

EFI_SUCCESS	Command success
EFI_DEVICE_ERROR	Command error

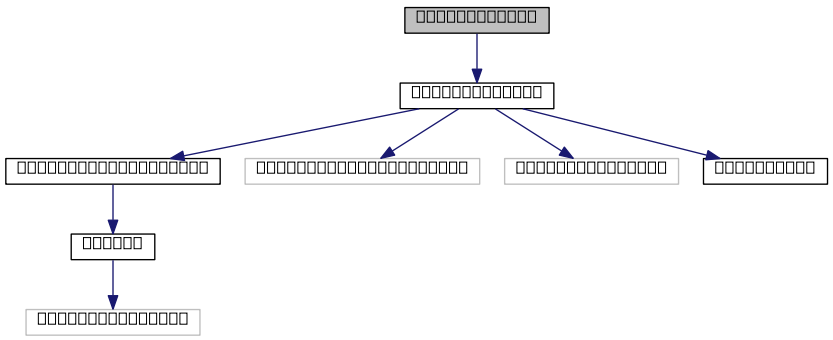
Definition at line 64 of file BiosGuardConfig.h.

13.5 CnviConfig.h File Reference

CNVi policy.

```
#include <ConfigBlock.h>
```

Include dependency graph for CnviConfig.h:



Classes

- struct CNVI_PIN_MUX

  CNVi signals pin muxing settings.
- struct CNVI_CONFIG

  The CNVI_CONFIG block describes the expected configuration of the CNVi IP.
- struct CNVI_PREMEM_CONFIG

  The CNVI_PREMEM_CONFIG block describes the expected configuration of the CNVi IP.

Enumerations

- enum CNVI_MODE

  CNVi Mode options.
- enum CNVI_BT_INTERFACE

  CNVi Bluetooth interface options.
- enum CNVI_BT_OFFLOAD_INTERFACE

  CNVi Bluetooth Audio Offload Interface.

13.5.1 Detailed Description

CNVi policy.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2016 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

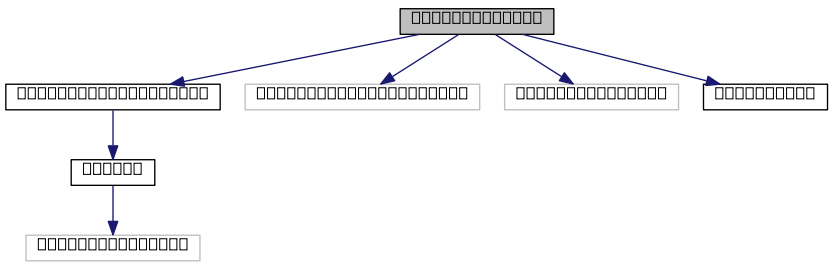
Specification Reference:

13.6 ConfigBlock.h File Reference

Header file for Config Block Lib implementation.

```
#include <Uefi/UefiBaseType.h>
#include <Uefi/UefiMultiPhase.h>
#include <Pi/PiBootMode.h>
#include <Pi/PiHob.h>
```

Include dependency graph for ConfigBlock.h:



This graph shows which files directly or indirectly include this file:

Classes

- struct _CONFIG_BLOCK_HEADER

  Config Block Header.
- struct _CONFIG_BLOCK

  Config Block.
- struct _CONFIG_BLOCK_TABLE_STRUCT

  Config Block Table Header.

Typedefs

- typedef struct _CONFIG_BLOCK_HEADER CONFIG_BLOCK_HEADER

  Config Block Header.
- typedef struct _CONFIG_BLOCK CONFIG_BLOCK

  Config Block.
- typedef struct _CONFIG_BLOCK_TABLE_STRUCT CONFIG_BLOCK_TABLE_HEADER

  Config Block Table Header.

13.6.1 Detailed Description

Header file for Config Block Lib implementation.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2019 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

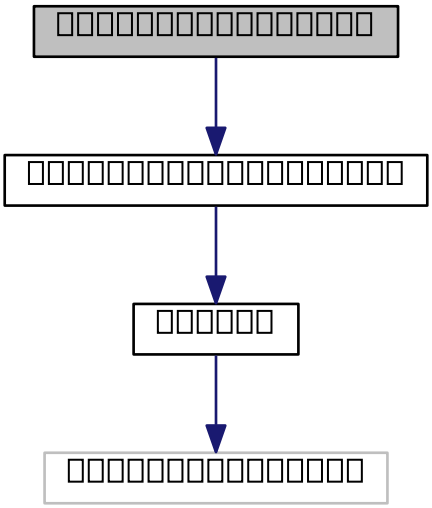
Specification

13.7 ConfigBlockLib.h File Reference

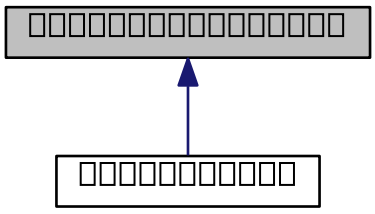
Header file for Config Block Lib implementation.

```
#include <Uefi/UefiBaseType.h>
```

Include dependency graph for ConfigBlockLib.h:



This graph shows which files directly or indirectly include this file:



Functions

- EFI_STATUS CreateConfigBlockTable (IN UINT16 TotalSize, OUT VOID **ConfigBlockTableAddress)

  Create config block table.
- EFI_STATUS AddConfigBlock (IN VOID *ConfigBlockTableAddress, OUT VOID **ConfigBlockAddress)

  Add config block into config block table structure.
- EFI_STATUS GetConfigBlock (IN VOID *ConfigBlockTableAddress, IN EFI_GUID *ConfigBlockGuid, OUT VOID **ConfigBlockAddress)

  Retrieve a specific Config Block data by GUID.

13.7.1 Detailed Description

Header file for Config Block Lib implementation.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2019 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification

13.7.2 Function Documentation

13.7.2.1 AddConfigBlock()

```
EFI_STATUS AddConfigBlock (
             IN VOID * ConfigBlockTableAddress,
             OUT VOID ** ConfigBlockAddress )
```

Add config block into config block table structure.

Parameters

in	*ConfigBlockTableAddress*	- A pointer to the beginning of Config Block Table Address
out	*ConfigBlockAddress*	- On return, points to a pointer to the beginning of Config Block Address

Return values

EFI_OUT_OF_RESOURCES	- Config Block Table is full and cannot add new Config Block or Config Block Offset Table is full and cannot add new Config Block.
EFI_SUCCESS	- Successfully added Config Block

13.7.2.2 CreateConfigBlockTable()

```
EFI_STATUS CreateConfigBlockTable (
```

```
            IN UINT16 TotalSize,
            OUT VOID ** ConfigBlockTableAddress )
```

Create config block table.

Parameters

in	*TotalSize*	- Max size to be allocated for the Config Block Table
out	*ConfigBlockTableAddress*	- On return, points to a pointer to the beginning of Config Block Table Address

Return values

EFI_INVALID_PARAMETER	- Invalid Parameter
EFI_OUT_OF_RESOURCES	- Out of resources
EFI_SUCCESS	- Successfully created Config Block Table at ConfigBlockTableAddress

13.7.2.3 GetConfigBlock()

```
EFI_STATUS GetConfigBlock (
            IN VOID * ConfigBlockTableAddress,
            IN EFI_GUID * ConfigBlockGuid,
            OUT VOID ** ConfigBlockAddress )
```

Retrieve a specific Config Block data by GUID.

Parameters

in	*ConfigBlockTableAddress*	- A pointer to the beginning of Config Block Table Address
in	*ConfigBlockGuid*	- A pointer to the GUID uses to search specific Config Block
out	*ConfigBlockAddress*	- On return, points to a pointer to the beginning of Config Block Address

Return values

EFI_NOT_FOUND	- Could not find the Config Block
EFI_SUCCESS	- Config Block found and return

13.8 CpuInitConfig.h File Reference

CPU Initialization Config Block.

Macros

- #define CPU_INIT_CONFIG_REVISION 2

  CPU Initialization Configuration Structure.

13.8.1 Detailed Description

CPU Initialization Config Block.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2015 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.8.2 Macro Definition Documentation

13.8.2.1 CPU_INIT_CONFIG_REVISION

```
#define CPU_INIT_CONFIG_REVISION 2
```

CPU Initialization Configuration Structure.

Revision 1:

- Initial version. **Revision 2**:
- Added new configuration options for Intel TCC Feature (AcSplitLock)

Definition at line 37 of file CpuInitConfig.h.

13.9 CpuInitPreMemConfig.h File Reference

CPU Initialization PreMemory Config Block.

Classes

- struct CPU_INIT_PREMEM_CONFIG

  CPU Config Library PreMemory Configuration Structure.

13.9.1 Detailed Description

CPU Initialization PreMemory Config Block.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2015 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

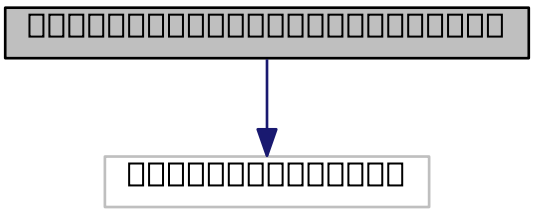
This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.10 CpuPowerDeliveryConfig.h File Reference

CPU Power Delivery Config Block.

`#include <CpuPowerMgmt.h>`
Include dependency graph for CpuPowerDeliveryConfig.h:



Classes

- struct CPU_POWER_DELIVERY_CONFIG
  CPU Power Delivery Configuration Structure.

Macros

- #define MAX_CUSTOM_CTDP_ENTRIES 3
  Defines the maximum number of custom ConfigTdp entries supported.

13.10.1 Detailed Description

CPU Power Delivery Config Block.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2022 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.10.2 Macro Definition Documentation

13.10.2.1 MAX_CUSTOM_CTDP_ENTRIES

```
#define MAX_CUSTOM_CTDP_ENTRIES 3
```

Defines the maximum number of custom ConfigTdp entries supported.

Warning

: Changing this define would cause DWORD alignment issues in policy structures.

Definition at line 34 of file CpuPowerDeliveryConfig.h.

13.11 CpuPowerMgmtBasicConfig.h File Reference

CPU Power Management Basic Config Block.

Macros

- #define CPU_POWER_MGMT_BASIC_CONFIG_REVISION 1

  CPU Power Management Basic Configuration Structure.

13.11.1 Detailed Description

CPU Power Management Basic Config Block.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2015 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.11.2 Macro Definition Documentation

13.11.2.1 CPU_POWER_MGMT_BASIC_CONFIG_REVISION

```
#define CPU_POWER_MGMT_BASIC_CONFIG_REVISION 1
```

CPU Power Management Basic Configuration Structure.

Revision 1:

- Initial version.

Definition at line 39 of file CpuPowerMgmtBasicConfig.h.

13.12 CpuPowerMgmtCustomConfig.h File Reference

CPU Power Managment Custom Config Block.

Classes

- struct PPM_CUSTOM_RATIO_TABLE

  This structure is used to describe the custom processor ratio table desired by the platform.
- struct CPU_POWER_MGMT_CUSTOM_CONFIG

  CPU Power Management Custom Configuration Structure.

Macros

- #define MAX_CUSTOM_RATIO_TABLE_ENTRIES 40

  Defines the maximum number of custom ratio states supported.

13.12.1 Detailed Description

CPU Power Managment Custom Config Block.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2015 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.13 CpuPowerMgmtTestConfig.h File Reference

CPU Power Management Test Config Block.

Macros

- #define CPU_POWER_MGMT_TEST_CONFIG_REVISION 1

  CPU Power Management Test Configuration Structure.

Enumerations

- enum MAX_PKG_C_STATE

  PPM Package C State Limit.
- enum CUSTOM_POWER_UNIT

  Custom Power Units.
- enum PPM_IRM_SETTING

  PPM Interrupt Redirection Mode Selection.

13.13.1 Detailed Description

CPU Power Management Test Config Block.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2015 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.13.2 Macro Definition Documentation

13.13.2.1 CPU_POWER_MGMT_TEST_CONFIG_REVISION

```
#define CPU_POWER_MGMT_TEST_CONFIG_REVISION 1
```

CPU Power Management Test Configuration Structure.

Revision 1:

- Initial version.

Definition at line 76 of file CpuPowerMgmtTestConfig.h.

13.13.3 Enumeration Type Documentation

13.13.3.1 CUSTOM_POWER_UNIT

```
enum CUSTOM_POWER_UNIT
```

Custom Power Units.

User can choose to enter in watts or 125 milliwatt increments.

Enumerator

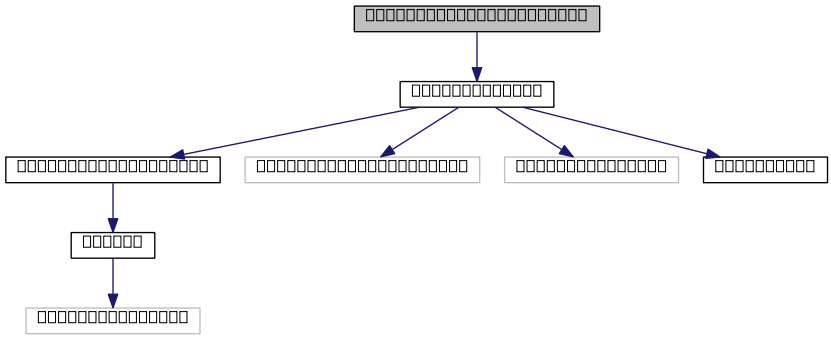
PowerUnitWatts	in Watts.
PowerUnit125MilliWatts	in 125 milliwatt increments. Example: 90 power units times 125 mW equals 11.250 W.

Definition at line 49 of file CpuPowerMgmtTestConfig.h.

13.14 CpuPowerMgmtVrConfig.h File Reference

CPU Power Management VR Config Block.

`#include <ConfigBlock.h>`
Include dependency graph for CpuPowerMgmtVrConfig.h:



Macros

- #define MAX_NUM_VRS 6

  Defines the maximum number of VR domains supported.
- #define CPU_POWER_MGMT_VR_CONFIG_REVISION 2

  CPU Power Management VR Configuration Structure.

13.14.1 Detailed Description

CPU Power Management VR Config Block.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2015 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.14.2 Macro Definition Documentation

13.14.2.1 CPU_POWER_MGMT_VR_CONFIG_REVISION

```
#define CPU_POWER_MGMT_VR_CONFIG_REVISION 2
```

CPU Power Management VR Configuration Structure.

Revision 1:

- Initial version. **Revision 2**:
- Added VccsaShutdown.

Definition at line 43 of file CpuPowerMgmtVrConfig.h.

13.14.2.2 MAX_NUM_VRS

```
#define MAX_NUM_VRS 6
```

Defines the maximum number of VR domains supported.

Warning

: Changing this define would cause DWORD alignment issues in policy structures.

Definition at line 32 of file CpuPowerMgmtVrConfig.h.

13.15 CpuSecurityPreMemConfig.h File Reference

CPU Security PreMemory Config Block.

Classes

- struct CPU_SECURITY_PREMEM_CONFIG

  CPU Security PreMemory Configuration Structure.

13.15.1 Detailed Description

CPU Security PreMemory Config Block.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2023 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.16 CpuTestConfig.h File Reference

CPU Test Config Block.

Classes

- struct CPU_TEST_CONFIG
  CPU Test Configuration Structure.

13.16.1 Detailed Description

CPU Test Config Block.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2015 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

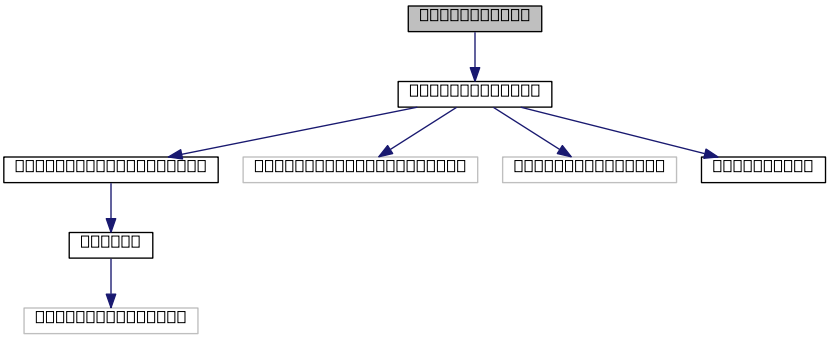
Specification Reference:

13.17 DciConfig.h File Reference

Dci policy.

```
#include <ConfigBlock.h>
```

Include dependency graph for DciConfig.h:



Classes

- struct PCH_DCI_PREMEM_CONFIG

  The PCH_DCI_PREMEM_CONFIG block describes policies related to Direct Connection Interface (DCI)

Enumerations

- enum DCI_MAILBOX_REG

  ExI mailbox handler configuration registers Register default value: 0x00000000.

13.17.1 Detailed Description

Dci policy.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2015 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

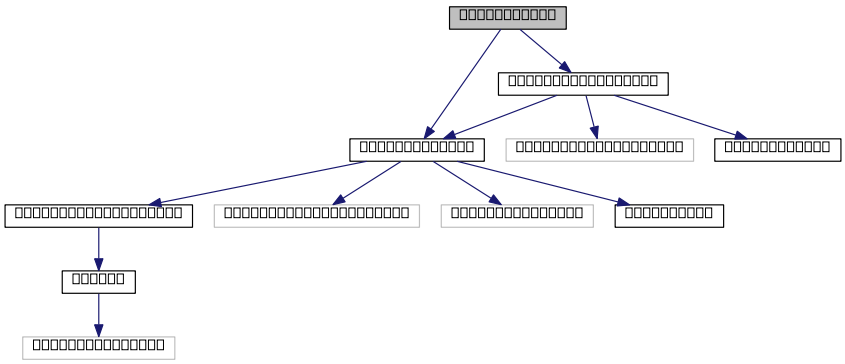
Specification Reference:

13.18 DmiConfig.h File Reference

Config block definition for PCH DMI.

```
#include <ConfigBlock.h>
#include <PchPcieRpConfig.h>
```

Include dependency graph for DmiConfig.h:



13.18.1 Detailed Description

Config block definition for PCH DMI.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2022 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

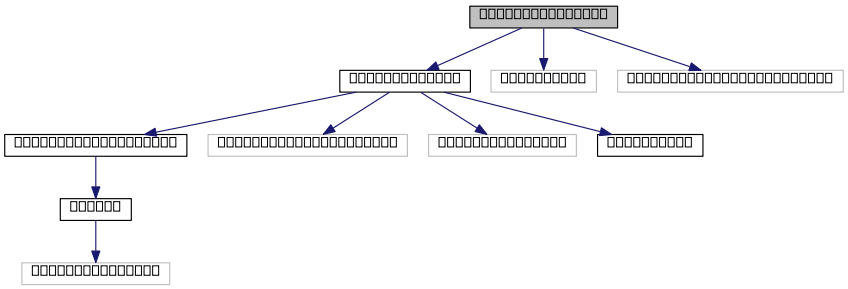
13.19 DxeITbtConfig.h File Reference

Headr file for iTBT DXE Policy.

```
#include <ConfigBlock.h>
#include <TcssInfo.h>
```

```
#include <DxeTbtCommonDefinition.h>
```

Include dependency graph for DxeITbtConfig.h:



Classes

- struct _DXE_ITBT_GENERIC_CONFIG

  ITBT Generic Data Structure to be used cross to RP and controller in DXE phase.
- struct _DXE_ITBT_CONFIG

  ITBT DXE configuration
  ***Revision 1**:*

Typedefs

- typedef struct _DXE_ITBT_GENERIC_CONFIG DXE_ITBT_GENERIC_CONFIG

  ITBT Generic Data Structure to be used cross to RP and controller in DXE phase.
- typedef struct _DXE_ITBT_CONFIG DXE_ITBT_CONFIG

  ITBT DXE configuration
  ***Revision 1**:*

13.19.1 Detailed Description

Headr file for iTBT DXE Policy.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2017 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.19.2 Typedef Documentation

13.19.2.1 DXE_ITBT_CONFIG

```
typedef struct _DXE_ITBT_CONFIG DXE_ITBT_CONFIG
```

ITBT DXE configuration
Revision 1:

- Initial version. **Revision 2**:
- Add ITbtVproDockSupport in DXE_ITBT_GENERIC_CONFIG structure.

13.20 EspiConfig.h File Reference

Espi policy.

Classes

- struct PCH_ESPI_CONFIG

  This structure contains the policies which are related to ESPI.

13.20.1 Detailed Description

Espi policy.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2015 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

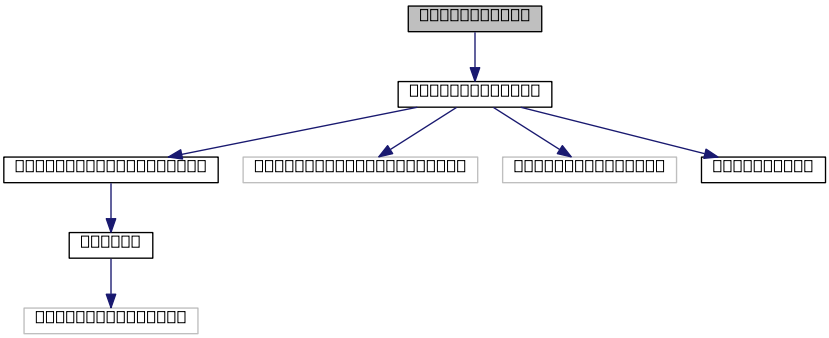
Specification Reference:

13.21 FiaConfig.h File Reference

PCH FIA MUX config block containing user configuration options.

```
#include <ConfigBlock.h>
```

Include dependency graph for FiaConfig.h:



13.21.1 Detailed Description

PCH FIA MUX config block containing user configuration options.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2020 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.22 FivrConfig.h File Reference

PCH FIVR policy.

Classes

- struct FIVR_EXT_RAIL_CONFIG

  Structure for V1p05/Vnn VR rail configuration.
- struct FIVR_VCCIN_AUX_CONFIG

  Structure for VCCIN_AUX voltage rail configuration.
- struct PCH_FIVR_CONFIG

  The PCH_FIVR_CONFIG block describes FIVR settings.

Enumerations

- enum FIVR_RAIL_SX_STATE

  Rail support in S0ix and Sx Settings other than FivrRailDisabled can be OR'ed.

13.22.1 Detailed Description

PCH FIVR policy.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2017 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.23 FlashProtectionConfig.h File Reference

FlashProtection policy.

Classes

- struct PROTECTED_RANGE

  Protected Flash Range.
- struct PCH_FLASH_PROTECTION_CONFIG

  The PCH provides a method for blocking writes and reads to specific ranges in the SPI flash when the Protected Ranges are enabled.

13.23.1 Detailed Description

FlashProtection policy.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2015 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

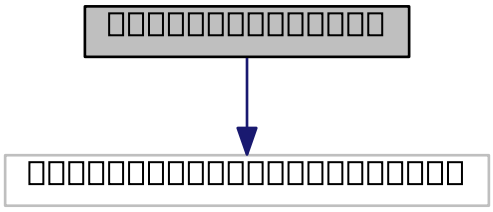
This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.24 FspErrorInfo.h File Reference

FSP Error Information to describe errors inside FSP that bootloader may take some actions to handle those error scenarios.

`#include <Guid/FspErrorInfoHob.h>`
Include dependency graph for FspErrorInfo.h:



Classes

- struct FSP_ERROR_INFO
  FSP Error Information Block.

Macros

- #define STATUS_CODE_DATA_TYPE_FSP_ERROR_GUID
  GUID value indicating the FSP error information.

13.24.1 Detailed Description

FSP Error Information to describe errors inside FSP that bootloader may take some actions to handle those error scenarios.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2019 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.25 FspFixedPcds.h File Reference

This file lists all FixedAtBuild PCDs referenced in FSP integration guide.

Macros

- #define PcdFspAreaBaseAddress 0xFFE76000

  FspAreaBaseAddress.
- #define PcdFspImageIdString $PTLFSP$

  FspImageIdString.
- #define PcdGlobalDataPointerAddress 0xFED00148

  GlobalDataPointerAddress.
- #define PcdTemporaryRamBase 0xFEF00000

  TemporaryRamBase.
- #define PcdTemporaryRamSize 0x000FF000

  TemporaryRamSize.
- #define PcdFspReservedBufferSize 0x100

  FspReservedBufferSize.

13.25.1 Detailed Description

This file lists all FixedAtBuild PCDs referenced in FSP integration guide.

Those value may vary in different FSP revision to meet different requirements.

13.26 FspmArchConfigPpi.h File Reference

Header file for FSP-M Arch Config PPI.

Classes

- struct FSPM_ARCH_CONFIG_PPI
  This PPI provides FSP-M Arch Config PPI.

Macros

- #define FSPM_ARCH_CONFIG_GUID
  Global ID for the FSPM_ARCH_CONFIG_PPI.

13.26.1 Detailed Description

Header file for FSP-M Arch Config PPI.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2018 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification

13.27 FspPerformance.h File Reference

Library for Fsp Performance Lib Config file.

13.27.1 Detailed Description

Library for Fsp Performance Lib Config file.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2024 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

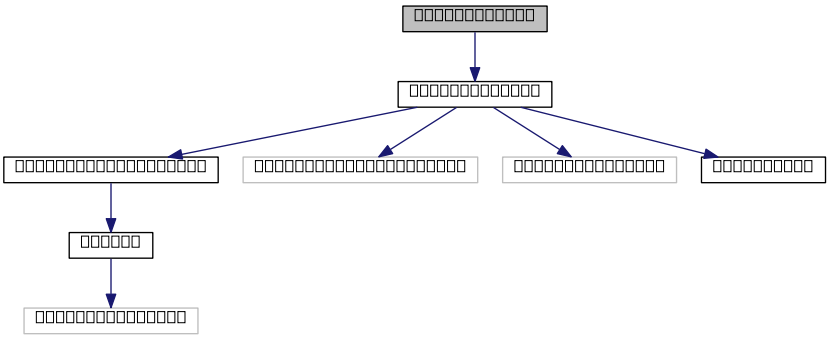
Specification Reference:

13.28 FspVConfig.h File Reference

FspV Config block.

```
#include <ConfigBlock.h>
```

Include dependency graph for FspVConfig.h:



Classes

- struct FSPV_PREMEM_CONFIG

  The FSPV_PREMEM_CONFIG block describes policies related to FSP Validation.
- struct FSPV_POSTMEM_CONFIG

  The FSPV_POSTMEM_CONFIG block describes policies related to FSP Validation.

Enumerations

- enum FSPM_TEST_PARAMETER

  Add FSPM phase specific test case IDs.
- enum FSPS_TEST_PARAMETER

  Add FSPS phase specific test case IDs.

13.28.1 Detailed Description

FspV Config block.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2023 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.29 FusaConfig.h File Reference

Functional Safety policy.

Classes

- struct FUSA_CONFIG

  The FUSA_CONFIG block describes the expected configuration of Functional Safety.

Macros

- #define FUSA_CONFIG_REVISION 5

  Making any setup structure change after code frozen will need to maintain backward compatibility, bump up structure revision and update below history table
  ***Revision 1**: - Initial version.*

Enumerations

- enum LOCKSTEP_MODE

  LOCKSTEP ENABLE.

13.29.1 Detailed Description

Functional Safety policy.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2019 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.29.2 Macro Definition Documentation

13.29.2.1 FUSA_CONFIG_REVISION

```
#define FUSA_CONFIG_REVISION 5
```

Making any setup structure change after code frozen will need to maintain backward compatibility, bump up structure revision and update below history table
Revision 1: - Initial version.

Revision 2: - Add FusaConfigEnable switch, BIST scan and locksteps for module. **Revision 3**: - Add LpcLockstep **Revision 4**: - Add CFI Parity variables **Revision 5**: - Add IsFusaDiagnosticMode bit

Definition at line 36 of file FusaConfig.h.

13.30 GbeConfig.h File Reference

Gigabit Ethernet policy.

Classes

- struct GBE_CONFIG

  PCH intergrated GBE controller configuration settings.

13.30.1 Detailed Description

Gigabit Ethernet policy.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2015 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.31 GpioDevConfig.h File Reference

GPIO device policy.

Classes

- struct GPIO_DXE_CONFIG

  This structure contains the DXE policies which are related to GPIO device.

13.31.1 Detailed Description

GPIO device policy.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2017 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

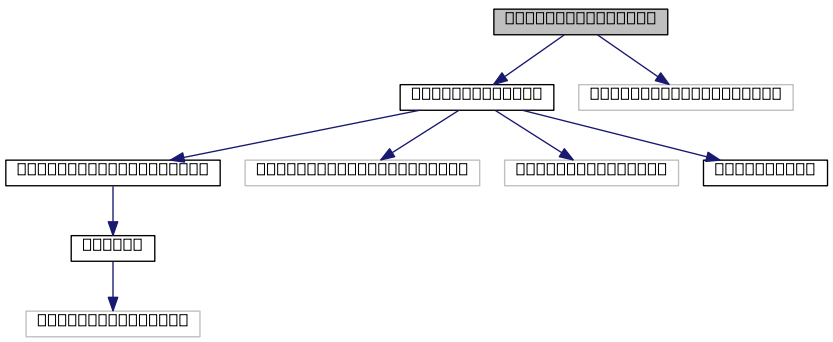
Specification Reference:

13.32 HdAudioConfig.h File Reference

HDAUDIO policy.

```
#include <ConfigBlock.h>
#include <Library/PchLimits.h>
```

Include dependency graph for HdAudioConfig.h:



Classes

- struct HDA_VERB_TABLE_HEADER

  Azalia verb table header Every verb table should contain this defined header and followed by azalia verb commands.
- struct HDA_MIC_PRIVACY

  HD Audio Link Policies.
- struct HDA_LINK_HDA

  HD Audio Link Policies.
- struct HDA_LINK_DMIC

  HD Audio DMIC Interface Policies.
- struct HDA_LINK_SSP

  HD Audio SSP Interface Policies.
- struct HDA_LINK_SNDW

  HD Audio SNDW Interface Policies.
- struct HDA_SNDW_MULTILANE

  HD Audio SNDW Multilane Policies.
- struct HDAUDIO_CONFIG

  This structure contains the policies which are related to HD Audio device (cAVS).
- struct HDAUDIO_PREMEM_CONFIG

  This structure contains the premem policies which are related to HD Audio device (cAVS).
- struct HDAUDIO_DXE_CONFIG

  This structure contains the DXE policies which are related to HD Audio device (cAVS).

Macros

- #define HDAUDIO_VERB_TABLE_VIDDID(Vid, Did) (UINT32)((UINT16)Vid | ((UINT16)Did << 16))

  The PCH_HDAUDIO_CONFIG block describes the expected configuration of the Intel HD Audio feature.
- #define HDAUDIO_VERB_TABLE_INIT(Vid, Did, Rid, Sdi, ...)

  Use this macro to create HDAUDIO_VERB_TABLE and populate size automatically.

13.32.1 Detailed Description

HDAUDIO policy.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2015 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

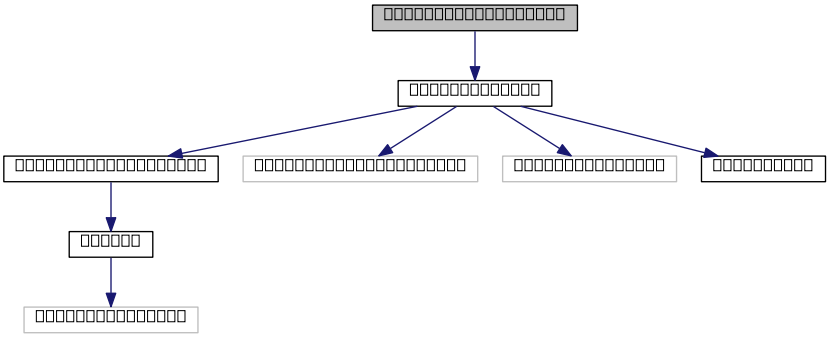
This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.33 HostBridgeConfig.h File Reference

Configurations for HostBridge.

`#include <ConfigBlock.h>`
Include dependency graph for HostBridgeConfig.h:



Classes

- struct HOST_BRIDGE_PREMEM_CONFIG

  This configuration block describes HostBridge settings in PreMem.
- struct HOST_BRIDGE_PEI_CONFIG

  This configuration block describes HostBridge settings in Post-Mem.

13.33.1 Detailed Description

Configurations for HostBridge.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2021 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.34 HsioConfig.h File Reference

HSIO policy.

Classes

- struct HSIO_CONFIG

  The HSIO_CONFIG block provides HSIO message related settings.

13.34.1 Detailed Description

HSIO policy.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2015 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

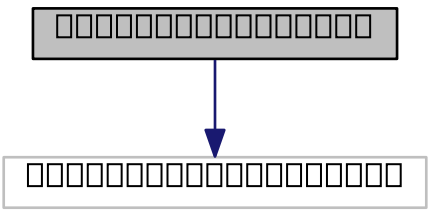
Specification Reference:

13.35 HsioPcieConfig.h File Reference

HSIO pcie policy.

```
#include <Library/PchLimits.h>
```

Include dependency graph for HsioPcieConfig.h:



Classes

- struct PCH_HSIO_PCIE_LANE_CONFIG

  The PCH_HSIO_PCIE_LANE_CONFIG describes HSIO settings for PCIe lane.
- struct PCH_HSIO_PCIE_PREMEM_CONFIG

  The PCH_HSIO_PCIE_CONFIG block describes the configuration of the HSIO for PCIe lanes.

13.35.1 Detailed Description

HSIO pcie policy.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2015 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.36 HsioSataConfig.h File Reference

Hsio Sata policy.

Classes

- struct PCH_HSIO_SATA_PORT_LANE

  The PCH_HSIO_SATA_PORT_LANE describes HSIO settings for SATA Port lane.
- struct PCH_HSIO_SATA_PREMEM_CONFIG

  The PCH_HSIO_SATA_CONFIG block describes the HSIO configuration of the SATA controller.

13.36.1 Detailed Description

Hsio Sata policy.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2015 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.37 HybridGraphicsConfig.h File Reference

Hybrid Graphics policy definitions.

Classes

- struct HYBRID_GRAPHICS_CONFIG

  This Configuration block configures CPU PCI Express 0/1/2 RTD3 GPIOs & Root Port.

13.37.1 Detailed Description

Hybrid Graphics policy definitions.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2014 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

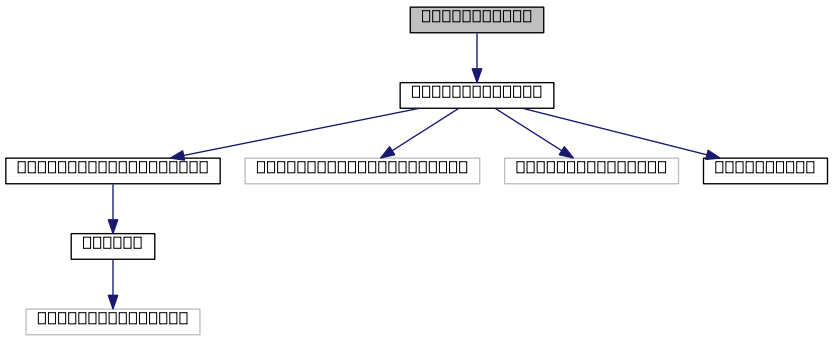
Specification Reference:

13.38 IaxConfig.h File Reference

IAX policy definitions.

```
#include <ConfigBlock.h>
```

Include dependency graph for IaxConfig.h:



Classes

- struct IAX_PEI_CONFIG

  IAX PEI configuration ***Revision 1****:*

13.38.1 Detailed Description

IAX policy definitions.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2021 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

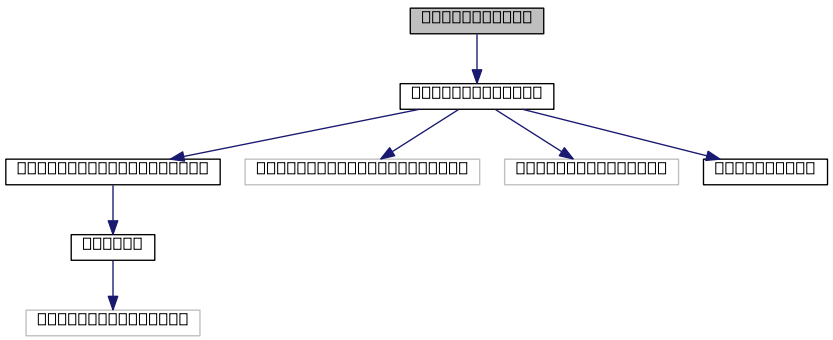
Specification Reference:

13.39 IehConfig.h File Reference

Integrated Error Handler policy.

```
#include <ConfigBlock.h>
```

Include dependency graph for IehConfig.h:



Classes

- struct IEH_CONFIG

  The IEH_CONFIG block describes the expected configuration of the PCH Integrated Error Handler.

13.39.1 Detailed Description

Integrated Error Handler policy.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2018 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

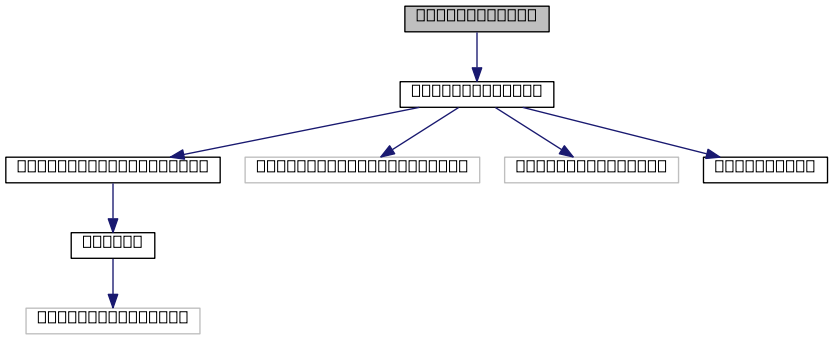
Specification Reference:

13.40 IGpuConfig.h File Reference

Policy definition for Internal Graphics Config Block.

```
#include <ConfigBlock.h>
```

Include dependency graph for IGpuConfig.h:



Classes

- struct DDI_CONFIGURATION

  This structure configures the Native GPIOs for DDI port per VBT settings.
- struct MRC_ERROR_KEY_VALUE_ENTRY

  MRC Error Key Value Entry structure maps an MRC error code to an error message string.
- struct IGPU_PEI_PREMEM_CONFIG

  This Configuration block is to configure GT related PreMem data/variables.
- struct IGPU_PEI_CONFIG

  This configuration block is to configure IGD related variables used in PostMem PEI.
- struct IGPU_DXE_CONFIG

  This configuration block is to configure IGD related variables used in DXE.

13.40.1 Detailed Description

Policy definition for Internal Graphics Config Block.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2021 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

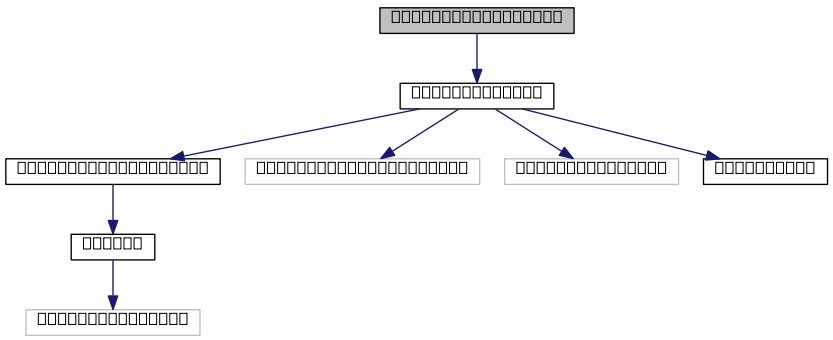
Specification Reference:

13.41 InterruptConfig.h File Reference

Interrupt policy.

```
#include <ConfigBlock.h>
```

Include dependency graph for InterruptConfig.h:



Classes

- struct PCH_DEVICE_INTERRUPT_CONFIG

  The PCH_DEVICE_INTERRUPT_CONFIG block describes interrupt pin, IRQ and interrupt mode for PCH device.
- struct PCH_INTERRUPT_CONFIG

  The PCH_INTERRUPT_CONFIG block describes interrupt settings for PCH.

Macros

- #define PCH_MAX_DEVICE_INTERRUPT_CONFIG 128

  Number of all PCH devices.
- #define PCH_MAX_PXRC_CONFIG 8

  Number of PXRC registers in ITSS.
- #define PCH_MAX_ITSS_IPC_REGS 4

  Number of IPC registers in ITSS.
- #define PCH_MAX_ITSS_IRQ_NUM 120

  Maximum number of IRQs.

Enumerations

- enum PCH_INT_PIN

13.41.1 Detailed Description

Interrupt policy.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2015 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.41.2 Enumeration Type Documentation

13.41.2.1 PCH_INT_PIN

```
enum PCH_INT_PIN
```

Enumerator

PchNoInt	No Interrupt Pin.

Definition at line 34 of file InterruptConfig.h.

13.42 IoApicConfig.h File Reference

IoApic policy.

Classes

- struct PCH_IOAPIC_CONFIG

  The PCH_IOAPIC_CONFIG block describes the expected configuration of the PCH IO APIC, it's optional and PCH code would ignore it if the BdfValid bit is not TRUE.

13.42.1 Detailed Description

IoApic policy.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2015 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

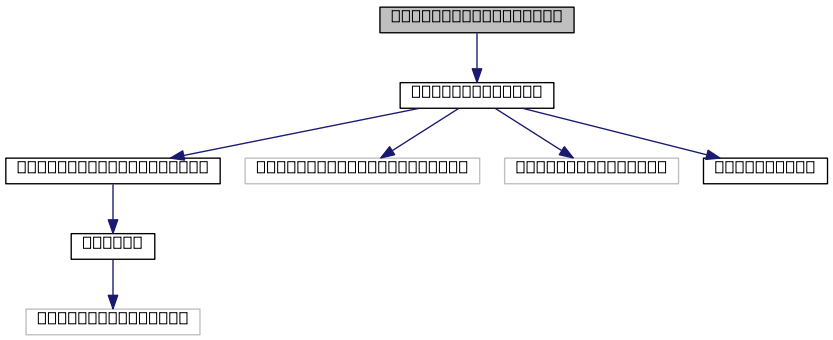
Specification Reference:

13.43 IpuPreMemConfig.h File Reference

IPU policy definitions.

```
#include <ConfigBlock.h>
```

Include dependency graph for IpuPreMemConfig.h:



Classes

- struct IPU_PREMEM_CONFIG

  IPU PreMem configuration
  ***Revision 1**:*

13.43.1 Detailed Description

IPU policy definitions.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2017 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.44 IshConfig.h File Reference

ISH policy.

Classes

- struct ISH_GPIO_CONFIG

  ISH GPIO settings.
- struct ISH_SPI_PIN_CONFIG

  SPI signals settings.
- struct ISH_UART_PIN_CONFIG

  UART signals settings.
- struct ISH_I2C_PIN_CONFIG

  I2C signals settings.
- struct ISH_SPI

  Struct contains GPIO pins assigned and signal settings of SPI.
- struct ISH_UART

  Struct contains GPIO pins assigned and signal settings of UART.
- struct ISH_I2C

  Struct contains GPIO pins assigned and signal settings of I2C.
- struct ISH_GP

  Struct contains GPIO pins assigned and signal settings of GP.
- struct ISH_CONFIG

  The ISH_CONFIG block describes Integrated Sensor Hub device.
- struct ISH_PREMEM_CONFIG

  Premem Policy for Integrated Sensor Hub device.

13.44.1 Detailed Description

ISH policy.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2015 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

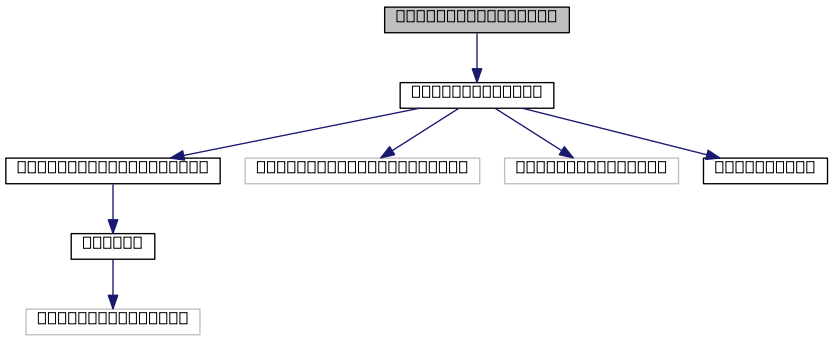
Specification Reference:

13.45 LockDownConfig.h File Reference

Lock down policy.

```
#include <ConfigBlock.h>
```

Include dependency graph for LockDownConfig.h:



Classes

- struct PCH_LOCK_DOWN_CONFIG

  The PCH_LOCK_DOWN_CONFIG block describes the expected configuration of the PCH for security requirement.

13.45.1 Detailed Description

Lock down policy.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2015 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.46 LpcConfig.h File Reference

Lpc policy.

Classes

- struct PCH_LPC_PREMEM_CONFIG

  This structure contains the policies which are related to LPC.

13.46.1 Detailed Description

Lpc policy.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2015 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

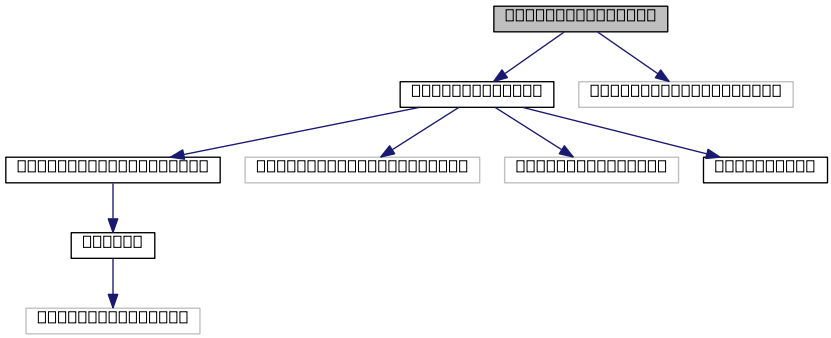
Specification Reference:

13.47 LpssI2cConfig.h File Reference

I2C config defines.

```
#include <ConfigBlock.h>
#include <Library/PchLimits.h>
```

Include dependency graph for LpssI2cConfig.h:



Classes

- struct LPSS_I2C_PIN_MUX

  I2C signals pin muxing settings.
- struct LPSS_I2C_CONTROLLER_CONFIG

  Serial IO I2C Controller Configuration.
- struct LPSS_I2C_CONFIG

  The LPSS_I2C_CONFIG block provides the configurations to set the Serial IO controllers.

Enumerations

- enum LPSS_I2C_I2C_MODE

  Available working modes for Lpss I2C Host Controller.

13.47.1 Detailed Description

I2C config defines.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2023 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.47.2 Enumeration Type Documentation

13.47.2.1 LPSS_I2C_I2C_MODE

```
enum LPSS_I2C_I2C_MODE
```

Available working modes for Lpss I2C Host Controller.

```
 0: LpssI2cDisabled;
    - Device is placed in D3
    - Gpio configuration is skipped
    - PSF:
    !important! If given device is Function 0 and other higher functions on given device
                are enabled, PSF disabling is skipped. PSF default will remain and device PCI CFG Space will s
                This is needed to allow PCI enumerator access functions above 0 in a multifunction device.
```

1: LpssI2cPci;

- Gpio pin configuration in native mode for each assigned pin
- Device will be enabled in PSF
- Only BAR0 will be enabled

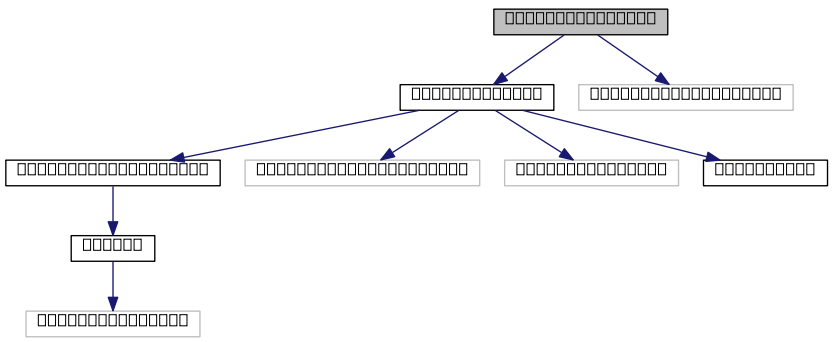
Definition at line 47 of file LpssI2cConfig.h.

13.48 LpssI3cConfig.h File Reference

I3C config defines.

```
#include <ConfigBlock.h>
#include <Library/PchLimits.h>
```

Include dependency graph for LpssI3cConfig.h:



Classes

- struct I3C_MUX_GPIO_CONFIG

  I3C GPIO settings.
- struct LPSS_I3C_DEVICE_CONFIG

  The LPSS_I3C_DEVICE_CONFIG provides the configurations for Serial IO I3C controller.
- struct LPSS_I3C_CONFIG

  The LPSS_I3C_CONFIG block provides the configurations to set the Serial IO controllers.

Enumerations

- enum I3C_MODE

  Available working modes for LPSS I3C Host Controller.

13.48.1 Detailed Description

I3C config defines.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2023 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.48.2 Enumeration Type Documentation

13.48.2.1 I3C_MODE

```
enum I3C_MODE
```

Available working modes for LPSS I3C Host Controller.

```
 0: I3cDisabled;
    - Device is placed in D3
    - Gpio configuration is skipped
    - Device will be disabled in PSF
```

1: I3cPci;

- Gpio pin configuration in native mode for each assigned pin SDA and SCL
- Device will be enabled in PSF
- Only Bar 0 will be enabled 2: I3cPhantom;
- Dedicated mode for I3C1, controls if GPIOs are enabled

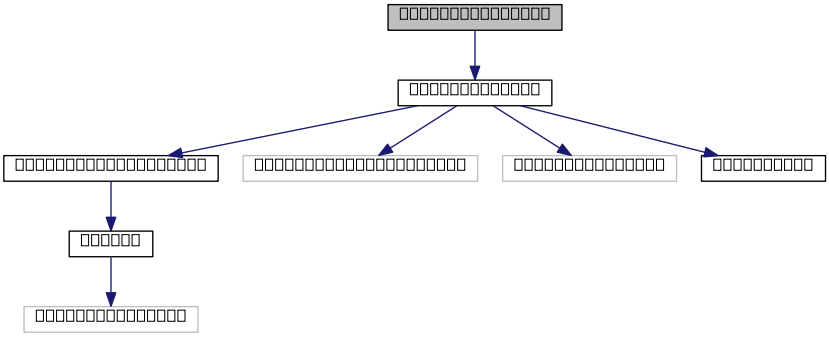
Definition at line 47 of file LpssI3cConfig.h.

13.49 LpssSpiConfig.h File Reference

LPSS SPI config defines.

```
#include <ConfigBlock.h>
```

Include dependency graph for LpssSpiConfig.h:



Classes

- struct LPSS_SPI_PIN_MUX

  SPI signals pin muxing settings.
- struct LPSS_SPI_DEVICE_CONFIG

  The LPSS_SPI_CONFIG provides the configurations to set the Serial IO SPI controller.
- struct LPSS_SPI_CONFIG

  The LPSS_SPI_CONFIG block provides the configurations to set the LPSS SPI controllers ***Revision 1:***

Enumerations

- enum LPSS_SPI_MODE

  Available working modes for LPSS SPI Host Controller.
- enum LPSS_CS_POLARITY

  Used to set Inactive/Idle polarity of Spi Chip Select.

13.49.1 Detailed Description

LPSS SPI config defines.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2023 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.49.2 Enumeration Type Documentation

13.49.2.1 LPSS_SPI_MODE

```
enum LPSS_SPI_MODE
```

Available working modes for LPSS SPI Host Controller.

```
 0: LpssSpiDisabled;
    - Device is placed in D3
    - Gpio configuration is skipped
    - PSF:
    !important! If given device is Function 0 and other higher functions on given device
                are enabled, PSF disabling is skipped. PSF default will remain and device PCI CFG Space will s
                This is needed to allow PCI enumerator access functions above 0 in a multifunction device.
```

1: LpssSpiPci;

- Gpio pin configuration in native mode for each assigned pin
- Device will be enabled in PSF
- Only BAR0 will be enabled

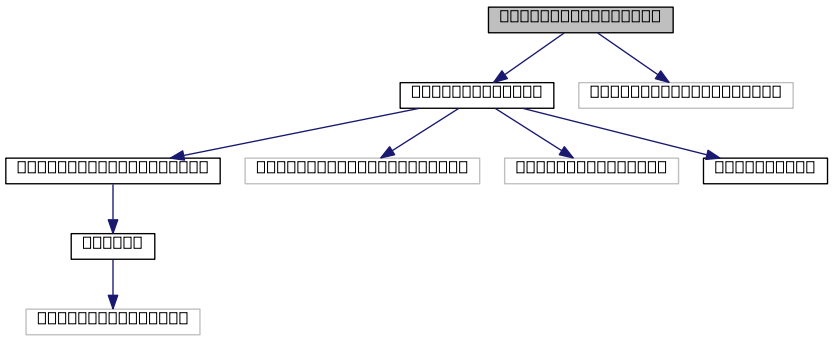
Definition at line 50 of file LpssSpiConfig.h.

13.50 LpssUartConfig.h File Reference

LPSS UART config defines.

```
#include <ConfigBlock.h>
#include <Library/PchLimits.h>
```

Include dependency graph for LpssUartConfig.h:



Classes

- struct LPSS_UART_ATTRIBUTES

  UART Settings.
- struct LPSS_UART_PIN_MUX

  UART signals pin muxing settings.
- struct LPSS_UART_DEVICE_CONFIG

  Lpss UART Controller Configuration.
- struct LPSS_UART_CONFIG

  The LPSS_UART_CONFIG block provides the configurations to set the LPSS UART controllers ***Revision 1:***

Enumerations

- enum LPSS_UART_MODE

  Available working modes for Lpss UART Host Controller.
- enum LPSS_UART_PG

13.50.1 Detailed Description

LPSS UART config defines.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2023 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.50.2 Enumeration Type Documentation

13.50.2.1 LPSS_UART_MODE

```
enum LPSS_UART_MODE
```

Available working modes for Lpss UART Host Controller.

```
 0: LpssUartDisabled;
    - Device is placed in D3
    - Gpio configuration is skipped
    - PSF:
    !important! If given device is Function 0 and other higher functions on given device
                are enabled, PSF disabling is skipped. PSF default will remain and device PCI CFG Space will s
                This is needed to allow PCI enumerator access functions above 0 in a multifunction device.
```

1: LpssUartPci;

- Designated for Serial IO UART OS driver usage
- Gpio pin configuration in native mode for each assigned pin
- Device will be enabled in PSF
- Only BAR0 will be enabled 2: LpssUartHidden;
- Designated for BIOS and/or DBG2 OS kernel debug
- Gpio pin configuration in native mode for each assigned pin
- Device disabled in the PSF
- Both BARs are enabled, BAR1 becomes devices Pci Config Space
- BAR0 assigned from the global PCH reserved memory range, reported as motherboard resource by SIRC

  Note

  If this controller is located at function 0 and it's mode is set to hidden it will not be visible in the PCI space. 3: LpssUartCom;
- Designated for 16550/PNP0501 compatible COM device
- Gpio pin configuration in native mode for each assigned pin
- Device disabled in the PSF
- Both BARs are enabled, BAR1 becomes devices Pci Config Space
- BAR0 assigned from the global PCH reserved memory range, reported as motherboard resource by SIRC 4: LpssUartSkipInit;
- Gpio configuration is skipped
- PSF configuration is skipped
- BAR assignemnt is skipped
- D-satate setting is skipped

Definition at line 71 of file LpssUartConfig.h.

13.50.2.2 LPSS_UART_PG

```
enum LPSS_UART_PG
```

Enumerator

LpssUartPgDisabled	No _PS0/_PS3 support, device left in D0, after initialization.
LpssUartPgEnabled	In mode: LpssUartCom; _PS0/_PS3 that supports getting device out of reset In mode: LpssUartPci Keeps UART in D0 and assigns Fixed MMIO for SEC/PEI usage only.
LpssUartPgAuto	_PS0 and _PS3, detection through ACPI if device was initialized prior to first PG. If it was used (DBG2) PG is disabled,

Definition at line 79 of file LpssUartConfig.h.

13.51 MeDxeConfig.h File Reference

ME config block for DXE phase.

Classes

- struct ME_DXE_CONFIG

  ME Dxe Configuration Structure.

13.51.1 Detailed Description

ME config block for DXE phase.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2015 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

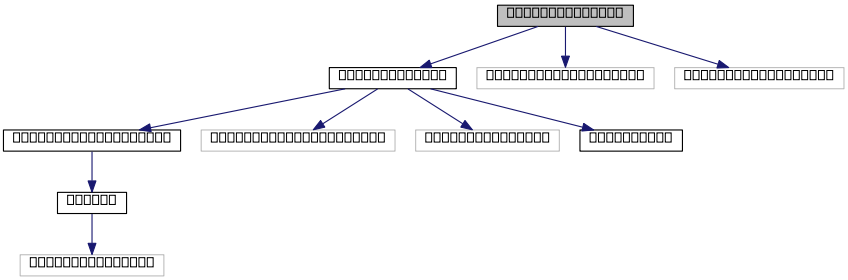
Specification Reference:

13.52 MemoryConfig.h File Reference

Policy definition of Memory Config Block Used by projects: PantherLake.

```
#include <ConfigBlock.h>
#include "Ptl/CMrcExtInputs.h"
#include "Ptl/CMrcExtTypes.h"
```

Include dependency graph for MemoryConfig.h:



Classes

- struct SPD_DATA_BUFFER

  SPD Data Buffer.
- struct MEMORY_DQDQS_MAPPING

  DqDqs Mapping.
- struct MEMORY_RCOMP

  Rcomp Policies.
- struct SPD_OFFSET_TABLE

  SPD Offset Table.
- struct SA_ADDRESS_DECODE

  SA memory address decode.
- struct MEMORY_CONFIGURATION

  Memory Configuration The contents of this structure are CRC'd by the MRC for option change detection.
- struct MEMORY_CONFIG_NO_CRC

  Memory Configuration The contents of this structure are not CRC'd by the MRC for option change detection.

Macros

- #define MEM_CFG_MAX_CONTROLLERS MAX_CONTROLLER

  Memory SubSystem Definitions.
- #define PEI_MR_SMRAM_ABSEG_MASK 0x01

  SMRAM Memory Range.

Enumerations

- enum SA_SPD

  SA SPD profile selections.
- enum SPD_BOOT_MODE

  Define the boot modes used by the SPD read function.

13.52.1 Detailed Description

Policy definition of Memory Config Block Used by projects: PantherLake.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2023 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.52.2 Enumeration Type Documentation

13.52.2.1 SA_SPD

`enum SA_SPD`

SA SPD profile selections.

Enumerator

Default	0, Default SPD
UserDefined	1, User Defined profile
XMPProfile1	2, XMP Profile 1
XMPProfile2	3, XMP Profile 2
XMPProfileMax	Ensures SA_SPD is UINT8.

Definition at line 63 of file MemoryConfig.h.

13.52.2.2 SPD_BOOT_MODE

`enum SPD_BOOT_MODE`

Define the boot modes used by the SPD read function.

Enumerator

SpdCold	Cold boot.
SpdWarm	Warm boot.
SpdS3	S3 resume.
SpdFast	Fast boot.
SpdBootModeMax	Delimiter.

Definition at line 74 of file MemoryConfig.h.

13.53 MemorySubSystemConfig.h File Reference

Policy definition for Memory SubSystem Config Block.

Classes

- struct MEMORY_SUB_SYSTEM_CONFIG

  This Configuration block configures Memory SubSystem ***Revision 1****:*

13.53.1 Detailed Description

Policy definition for Memory SubSystem Config Block.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2023 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

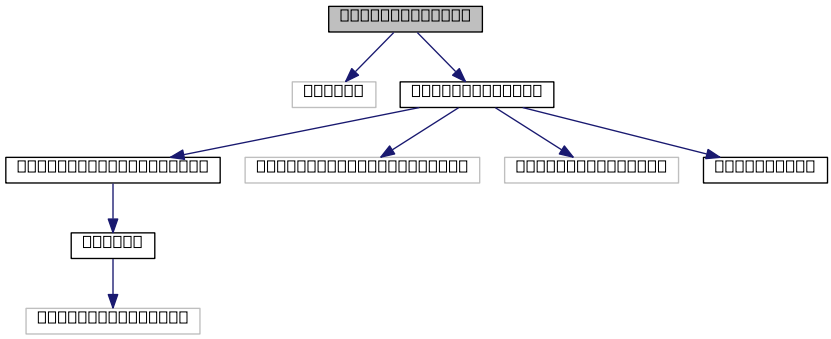
Specification Reference:

13.54 MePeiConfig.h File Reference

ME config block for PEI phase.

```
#include <Uefi.h>
#include <ConfigBlock.h>
```

Include dependency graph for MePeiConfig.h:



Classes

- struct ME_PEI_PREMEM_CONFIG

  ME Pei Pre-Memory Configuration Structure.
- struct ME_PEI_CONFIG

  ME Pei Post-Memory Configuration Structure.

13.54.1 Detailed Description

ME config block for PEI phase.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2015 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

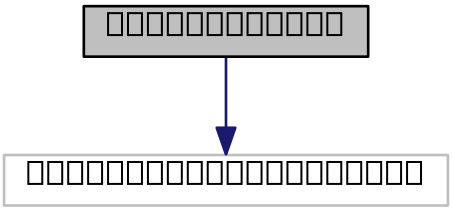
Specification Reference:

13.55 MpServices.h File Reference

This file declares UEFI PI Multi-processor PPI.

```
#include <Protocol/MpService.h>
```

Include dependency graph for MpServices.h:



Classes

- struct _EFI_PEI_MP_SERVICES_PPI

  This PPI is installed by some platform or chipset-specific PEIM that abstracts handling multiprocessor support.

Typedefs

- typedef EFI_STATUS(∗ EFI_PEI_MP_SERVICES_GET_NUMBER_OF_PROCESSORS) (IN CONST E↩ FI_PEI_SERVICES ∗∗PeiServices, IN EFI_PEI_MP_SERVICES_PPI ∗This, OUT UINTN ∗NumberOf↩ Processors, OUT UINTN ∗NumberOfEnabledProcessors)

  Get the number of CPU's.
- typedef EFI_STATUS(∗ EFI_PEI_MP_SERVICES_GET_PROCESSOR_INFO) (IN CONST EFI_PEI_SER↩ VICES ∗∗PeiServices, IN EFI_PEI_MP_SERVICES_PPI ∗This, IN UINTN ProcessorNumber, OUT EFI_P↩ ROCESSOR_INFORMATION ∗ProcessorInfoBuffer)

  Get information on a specific CPU.
- typedef EFI_STATUS(∗ EFI_PEI_MP_SERVICES_STARTUP_ALL_APS) (IN CONST EFI_PEI_SERVICES ∗∗PeiServices, IN EFI_PEI_MP_SERVICES_PPI ∗This, IN EFI_AP_PROCEDURE Procedure, IN BOOLEAN SingleThread, IN UINTN TimeoutInMicroSeconds, IN VOID ∗ProcedureArgument OPTIONAL)

  Activate all of the application processors.
- typedef EFI_STATUS(∗ EFI_PEI_MP_SERVICES_STARTUP_THIS_AP) (IN CONST EFI_PEI_SERVIC↩ ES ∗∗PeiServices, IN EFI_PEI_MP_SERVICES_PPI ∗This, IN EFI_AP_PROCEDURE Procedure, IN UINTN ProcessorNumber, IN UINTN TimeoutInMicroseconds, IN VOID ∗ProcedureArgument OPTIONAL)

  Activate a specific application processor.
- typedef EFI_STATUS(∗ EFI_PEI_MP_SERVICES_SWITCH_BSP) (IN CONST EFI_PEI_SERVICES ∗∗PeiServices, IN EFI_PEI_MP_SERVICES_PPI ∗This, IN UINTN ProcessorNumber, IN BOOLEAN EnableOldBSP)

  Switch the boot strap processor.
- typedef EFI_STATUS(∗ EFI_PEI_MP_SERVICES_ENABLEDISABLEAP) (IN CONST EFI_PEI_SERVIC↩ ES ∗∗PeiServices, IN EFI_PEI_MP_SERVICES_PPI ∗This, IN UINTN ProcessorNumber, IN BOOLEAN EnableAP, IN UINT32 ∗HealthFlag OPTIONAL)

  Enable or disable an application processor.
- typedef EFI_STATUS(∗ EFI_PEI_MP_SERVICES_WHOAMI) (IN CONST EFI_PEI_SERVICES ∗∗Pei↩ Services, IN EFI_PEI_MP_SERVICES_PPI ∗This, OUT UINTN ∗ProcessorNumber)

  Identify the currently executing processor.

13.55.1 Detailed Description

This file declares UEFI PI Multi-processor PPI.

This PPI is installed by some platform or chipset-specific PEIM that abstracts handling multiprocessor support.

Copyright (c) 2015 - 2017, Intel Corporation. All rights reserved.
SPDX-License-Identifier: BSD-2-Clause-Patent

Revision Reference:

This PPI is introduced in PI Version 1.4.

13.55.2 Typedef Documentation

13.55.2.1 EFI_PEI_MP_SERVICES_ENABLEDISABLEAP

```
typedef EFI_STATUS( * EFI_PEI_MP_SERVICES_ENABLEDISABLEAP) (IN CONST EFI_PEI_SERVICES **Pei↩
Services, IN EFI_PEI_MP_SERVICES_PPI *This, IN UINTN ProcessorNumber, IN BOOLEAN EnableAP, IN
UINT32 *HealthFlag OPTIONAL)
```

Enable or disable an application processor.

Parameters

in	*PeiServices*	An indirect pointer to the PEI Services Table published by the PEI Foundation.
in	*This*	A pointer to the EFI_PEI_MP_SERVICES_PPI instance.
in	*ProcessorNumber*	The handle number of the AP. The range is from 0 to the total number of logical processors minus 1. The total number of logical processors can be retrieved by EFI_PEI_MP_SERVICES_PPI.GetNumberOfProcessors().
in	*EnableAP*	Specifies the new state for the processor for enabled, FALSE for disabled.
in	*HealthFlag*	If not NULL, a pointer to a value that specifies the new health status of the AP. This flag corresponds to StatusFlag defined in EFI_PEI_MP_SERVICES_PPI.GetProcessorInfo(). Only the PROCESSOR_HEALTH_STATUS_BIT is used. All other bits are ignored. If it is NULL, this parameter is ignored.

Return values

EFI_SUCCESS	The specified AP was enabled or disabled successfully.
EFI_UNSUPPORTED	Enabling or disabling an AP cannot be completed prior to this service returning.
EFI_UNSUPPORTED	Enabling or disabling an AP is not supported.
EFI_DEVICE_ERROR	The calling processor is an AP.
EFI_NOT_FOUND	Processor with the handle specified by ProcessorNumber does not exist.
EFI_INVALID_PARAMETER	ProcessorNumber specifies the BSP.

Definition at line 229 of file MpServices.h.

13.55.2.2 EFI_PEI_MP_SERVICES_GET_NUMBER_OF_PROCESSORS

```
typedef EFI_STATUS( * EFI_PEI_MP_SERVICES_GET_NUMBER_OF_PROCESSORS) (IN CONST EFI_PEI_SERVI↩
CES **PeiServices, IN EFI_PEI_MP_SERVICES_PPI *This, OUT UINTN *NumberOfProcessors, OUT UINTN
*NumberOfEnabledProcessors)
```

Get the number of CPU's.

Parameters

in	*PeiServices*	An indirect pointer to the PEI Services Table published by the PEI Foundation.
in	*This*	Pointer to this instance of the PPI.
out	*NumberOfProcessors*	Pointer to the total number of logical processors in the system, including the BSP and disabled APs.
out	*NumberOfEnabledProcessors*	Number of processors in the system that are enabled.

Return values

EFI_SUCCESS	The number of logical processors and enabled logical processors was retrieved.
EFI_DEVICE_ERROR	The calling processor is an AP.
EFI_INVALID_PARAMETER	NumberOfProcessors is NULL. NumberOfEnabledProcessors is NULL.

Definition at line 44 of file MpServices.h.

13.55.2.3 EFI_PEI_MP_SERVICES_GET_PROCESSOR_INFO

```
typedef EFI_STATUS( * EFI_PEI_MP_SERVICES_GET_PROCESSOR_INFO) (IN CONST EFI_PEI_SERVICES **Pei↩
Services, IN EFI_PEI_MP_SERVICES_PPI *This, IN UINTN ProcessorNumber, OUT EFI_PROCESSOR_INFOR↩
MATION *ProcessorInfoBuffer)
```

Get information on a specific CPU.

Parameters

in	*PeiServices*	An indirect pointer to the PEI Services Table published by the PEI Foundation.
in	*This*	Pointer to this instance of the PPI.
in	*ProcessorNumber*	Pointer to the total number of logical processors in the system, including the BSP and disabled APs.
out	*ProcessorInfoBuffer*	Number of processors in the system that are enabled.

Return values

EFI_SUCCESS	Processor information was returned.
EFI_DEVICE_ERROR	The calling processor is an AP.

Return values

EFI_INVALID_PARAMETER	ProcessorInfoBuffer is NULL.
EFI_NOT_FOUND	The processor with the handle specified by ProcessorNumber does not exist in the platform.

Definition at line 69 of file MpServices.h.

13.55.2.4 EFI_PEI_MP_SERVICES_STARTUP_ALL_APS

```
typedef EFI_STATUS( * EFI_PEI_MP_SERVICES_STARTUP_ALL_APS) (IN CONST EFI_PEI_SERVICES **Pei↩
Services, IN EFI_PEI_MP_SERVICES_PPI *This, IN EFI_AP_PROCEDURE Procedure, IN BOOLEAN Single↩
Thread, IN UINTN TimeoutInMicroSeconds, IN VOID *ProcedureArgument OPTIONAL)
```

Activate all of the application processors.

Parameters

in	*PeiServices*	An indirect pointer to the PEI Services Table published by the PEI Foundation.
in	*This*	A pointer to the EFI_PEI_MP_SERVICES_PPI instance.
in	*Procedure*	A pointer to the function to be run on enabled APs of the system.
in	*SingleThread*	If TRUE, then all the enabled APs execute the function specified by Procedure one by one, in ascending order of processor handle number. If FALSE, then all the enabled APs execute the function specified by Procedure simultaneously.
in	*TimeoutInMicroSeconds*	Indicates the time limit in microseconds for APs to return from Procedure, for blocking mode only. Zero means infinity. If the timeout expires before all APs return from Procedure, then Procedure on the failed APs is terminated. All enabled APs are available for next function assigned by EFI_PEI_MP_SERVICES_PPI.StartupAllAPs() or EFI_PEI_MP_SERVICES_PPI.StartupThisAP(). If the timeout expires in blocking mode, BSP returns EFI_TIMEOUT.
in	*ProcedureArgument*	The parameter passed into Procedure for all APs.

Return values

EFI_SUCCESS	In blocking mode, all APs have finished before the timeout expired.
EFI_DEVICE_ERROR	Caller processor is AP.
EFI_NOT_STARTED	No enabled APs exist in the system.
EFI_NOT_READY	Any enabled APs are busy.
EFI_TIMEOUT	In blocking mode, the timeout expired before all enabled APs have finished.
EFI_INVALID_PARAMETER	Procedure is NULL.

Definition at line 112 of file MpServices.h.

13.55.2.5 EFI_PEI_MP_SERVICES_STARTUP_THIS_AP

```
typedef EFI_STATUS( * EFI_PEI_MP_SERVICES_STARTUP_THIS_AP) (IN CONST EFI_PEI_SERVICES **Pei↩
Services, IN EFI_PEI_MP_SERVICES_PPI *This, IN EFI_AP_PROCEDURE Procedure, IN UINTN Processor↩
Number, IN UINTN TimeoutInMicroseconds, IN VOID *ProcedureArgument OPTIONAL)
```

Activate a specific application processor.

Parameters

in	*PeiServices*	An indirect pointer to the PEI Services Table published by the PEI Foundation.
in	*This*	A pointer to the EFI_PEI_MP_SERVICES_PPI instance.
in	*Procedure*	A pointer to the function to be run on enabled APs of the system.
in	*ProcessorNumber*	The handle number of the AP. The range is from 0 to the total number of logical processors minus 1. The total number of logical processors can be retrieved by EFI_PEI_MP_SERVICES_PPI.GetNumberOfProcessors().
in	*TimeoutInMicroSeconds*	Indicates the time limit in microseconds for APs to return from Procedure, for blocking mode only. Zero means infinity. If the timeout expires before all APs return from Procedure, then Procedure on the failed APs is terminated. All enabled APs are available for next function assigned by EFI_PEI_MP_SERVICES_PPI.StartupAllAPs() or EFI_PEI_MP_SERVICES_PPI.StartupThisAP(). If the timeout expires in blocking mode, BSP returns EFI_TIMEOUT.
in	*ProcedureArgument*	The parameter passed into Procedure for all APs.

Return values

EFI_SUCCESS	In blocking mode, specified AP finished before the timeout expires.
EFI_DEVICE_ERROR	The calling processor is an AP.
EFI_TIMEOUT	In blocking mode, the timeout expired before the specified AP has finished.
EFI_NOT_FOUND	The processor with the handle specified by ProcessorNumber does not exist.
EFI_INVALID_PARAMETER	ProcessorNumber specifies the BSP or disabled AP.
EFI_INVALID_PARAMETER	Procedure is NULL.

Definition at line 157 of file MpServices.h.

13.55.2.6 EFI_PEI_MP_SERVICES_SWITCH_BSP

```
typedef EFI_STATUS( * EFI_PEI_MP_SERVICES_SWITCH_BSP) (IN CONST EFI_PEI_SERVICES **PeiServices,
IN EFI_PEI_MP_SERVICES_PPI *This, IN UINTN ProcessorNumber, IN BOOLEAN EnableOldBSP)
```

Switch the boot strap processor.

Parameters

in	*PeiServices*	An indirect pointer to the PEI Services Table published by the PEI Foundation.
in	*This*	A pointer to the EFI_PEI_MP_SERVICES_PPI instance.
in	*ProcessorNumber*	The handle number of the AP. The range is from 0 to the total number of logical processors minus 1. The total number of logical processors can be retrieved by
		EFI_PEI_MP_SERVICES_PPI.GetNumberOfProcessors().
in	*EnableOldBSP*	If TRUE, then the old BSP will be listed as an enabled AP. Otherwise, it will be disabled.

Return values

EFI_SUCCESS	BSP successfully switched.
EFI_UNSUPPORTED	Switching the BSP cannot be completed prior to this service returning.
EFI_UNSUPPORTED	Switching the BSP is not supported.
EFI_DEVICE_ERROR	The calling processor is an AP.
EFI_NOT_FOUND	The processor with the handle specified by ProcessorNumber does not exist.
EFI_INVALID_PARAMETER	ProcessorNumber specifies the current BSP or a disabled AP.
EFI_NOT_READY	The specified AP is busy.

Definition at line 192 of file MpServices.h.

13.55.2.7 EFI_PEI_MP_SERVICES_WHOAMI

```
typedef EFI_STATUS( * EFI_PEI_MP_SERVICES_WHOAMI) (IN CONST EFI_PEI_SERVICES **PeiServices, IN
EFI_PEI_MP_SERVICES_PPI *This, OUT UINTN *ProcessorNumber)
```

Identify the currently executing processor.

Parameters

in	*PeiServices*	An indirect pointer to the PEI Services Table published by the PEI Foundation.
in	*This*	A pointer to the EFI_PEI_MP_SERVICES_PPI instance.
out	*ProcessorNumber*	The handle number of the AP. The range is from 0 to the total number of logical processors minus 1. The total number of logical processors can be retrieved by EFI_PEI_MP_SERVICES_PPI.GetNumberOfProcessors().

Return values

EFI_SUCCESS	The current processor handle number was returned in ProcessorNumber.
EFI_INVALID_PARAMETER	ProcessorNumber is NULL.

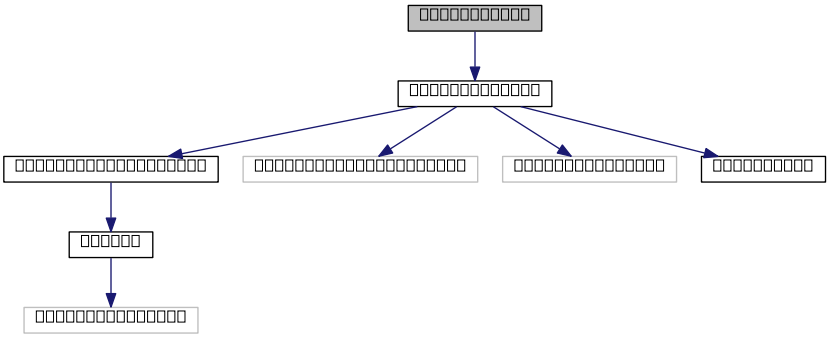
Definition at line 254 of file MpServices.h.

13.56 NpuConfig.h File Reference

NPU policy definitions.

```
#include <ConfigBlock.h>
```

Include dependency graph for NpuConfig.h:



Classes

- struct NPU_PEI_CONFIG

  NPU PEI configuration
  ***Revision 1**:*

13.56.1 Detailed Description

NPU policy definitions.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2020 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

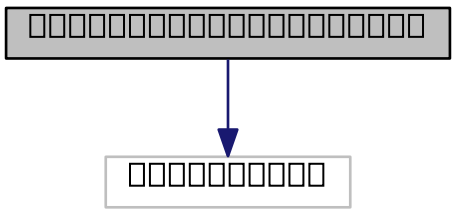
This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.57 OverclockingConfig.h File Reference

Overclocking Config Block.

`#include <OcLimits.h>`
Include dependency graph for OverclockingConfig.h:



Classes

- struct OVERCLOCKING_PREMEM_CONFIG

  Overclocking Configuration Structure.

13.57.1 Detailed Description

Overclocking Config Block.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2021 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.58 P2sbConfig.h File Reference

P2sb policy.

Classes

- struct PCH_P2SB_CONFIG

  This structure contains the policies which are related to P2SB device.

13.58.1 Detailed Description

P2sb policy.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2015 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.59 PchDmiConfig.h File Reference

DMI policy.

Classes

- struct PCH_DMI_CONFIG

  The PCH_DMI_CONFIG block describes the expected configuration of the PCH for DMI.

13.59.1 Detailed Description

DMI policy.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2015 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.60 PchGeneralConfig.h File Reference

PCH General policy.

Classes

- struct PCH_GENERAL_CONFIG

  PCH General Configuration ***Revision 2****: - Add VtdEnabled field.*
- struct PCH_GENERAL_PREMEM_CONFIG

  PCH General Pre-Memory Configuration ***Revision 1****: - Initial version.*

Enumerations

- enum PCH_RESERVED_PAGE_ROUTE

13.60.1 Detailed Description

PCH General policy.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2015 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.60.2 Enumeration Type Documentation

13.60.2.1 PCH_RESERVED_PAGE_ROUTE

`enum PCH_RESERVED_PAGE_ROUTE`

Enumerator

PchReservedPageToLpc	Port 80h cycles are sent to LPC.
PchReservedPageToPcie	Port 80h cycles are sent to PCIe.

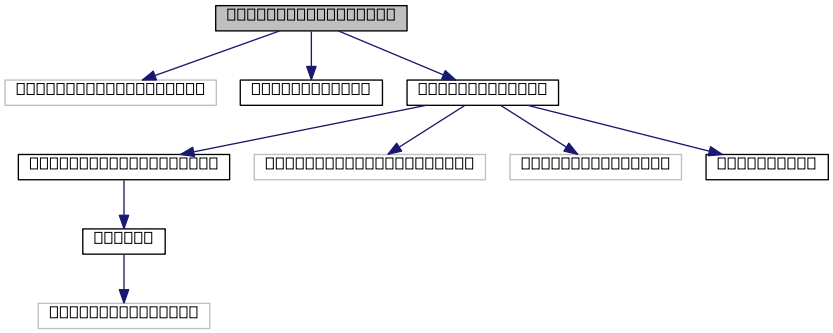
Definition at line 32 of file PchGeneralConfig.h.

13.61 PchPcieRpConfig.h File Reference

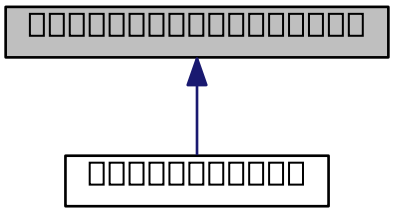
PCH Pcie root port policy.

```
#include <Library/PchLimits.h>
#include <PcieConfig.h>
#include <ConfigBlock.h>
```

Include dependency graph for PchPcieRpConfig.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct PCH_PCIE_CLOCK

  PCH_PCIE_CLOCK describes PCIe source clock generated by PCH.
- struct DMI_LANE_CONFIG

  The PCH_DMI_LANE_CONFIG block describes specific lane configuration.
- struct PCH_PCIE_ROOT_PORT_CONFIG

  The PCH_PCI_EXPRESS_ROOT_PORT_CONFIG describe the feature and capability of each PCH PCIe root port.
- struct PCH_PCIE_CONFIG

*The PCH_PCIE_CONFIG block describes the expected configuration of the PCH PCI Express controllers **Revision 1**:*

- struct PCH_PCIE_RP_PREMEM_CONFIG

  *The PCH_PCIE_RP_PREMEM_CONFIG block describes early configuration of the PCH PCI Express controllers **Revision 1**:*
- struct PCIE_RP_DXE_CONFIG

  The PCIE_RP_DXE_CONFIG block describes the expected configuration of the PCH PCI Express controllers in DXE phase.

Enumerations

- enum PCH_PCIE_ASPM_CONTROL

  The values before AutoConfig match the setting of PCI Express Base Specification 1.1, please be careful for adding new feature.
- enum PCH_PCIE_L1SUBSTATES_CONTROL

  Refer to PCH EDS for the PCH implementation values corresponding to below PCI-E spec defined ranges.
- enum PCH_PCIE_CLOCK_USAGE

13.61.1 Detailed Description

PCH Pcie root port policy.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2015 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.61.2 Enumeration Type Documentation

13.61.2.1 PCH_PCIE_CLOCK_USAGE

```
enum PCH_PCIE_CLOCK_USAGE
```

Enumerator

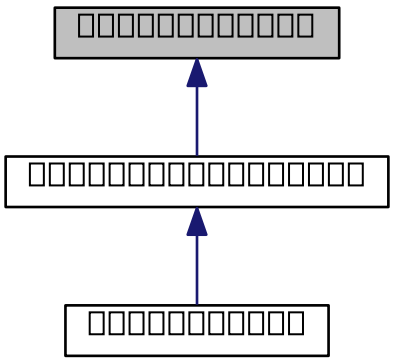
PchClockUsageCpuPcie0	Quantity of PCH and CPU PCIe ports, as well as their encoding in this enum, may change between silicon generations and series. Do not assume that PCH port 0 will be always encoded by 0. Instead, it is recommended to use (PchClockUsagePchPcie0 + PchPortIndex) style to be forward-compatible
PchClockUsageUnspecified	In use for a purpose not listed above.

Definition at line 130 of file PchPcieRpConfig.h.

13.62 PcieConfig.h File Reference

PCIe Config Block.

This graph shows which files directly or indirectly include this file:



Classes

- struct PCIE_EQ_PARAM

  Represent lane specific PCIe Gen3 equalization parameters.
- struct PCIE_LINK_EQ_PLATFORM_SETTINGS

  PCIe Link EQ Platform Settings.
- struct PCIE_COMMON_CONFIG

  PCIe Common Config.
- struct PCIE_DEVICE_OVERRIDE

  PCIe device table entry entry.

Enumerations

- enum PCIE_FORM_FACTOR

  Specifies the form factor that the slot implements.
- enum PCIE_LINK_EQ_METHOD
- enum PCIE_LINK_EQ_MODE

13.62.1 Detailed Description

PCIe Config Block.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2018 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.62.2 Enumeration Type Documentation

13.62.2.1 PCIE_FORM_FACTOR

`enum PCIE_FORM_FACTOR`

Specifies the form factor that the slot implements.

For custom form factors that do not require any special handling please set PcieFormFactorOther.

Definition at line 82 of file PcieConfig.h.

13.62.2.2 PCIE_LINK_EQ_METHOD

`enum PCIE_LINK_EQ_METHOD`

Enumerator

PcieLinkHardwareEq	Hardware is responsible for performing coefficient/preset search.
PcieLinkFixedEq	No coefficient/preset search is performed. Fixed values are used.

Definition at line 93 of file PcieConfig.h.

13.62.2.3 PCIE_LINK_EQ_MODE

```
enum PCIE_LINK_EQ_MODE
```

Enumerator

PcieLinkEqPresetMode	Use presets during PCIe link equalization.
PcieLinkEqCoefficientMode	Use coefficients during PCIe link equalization.

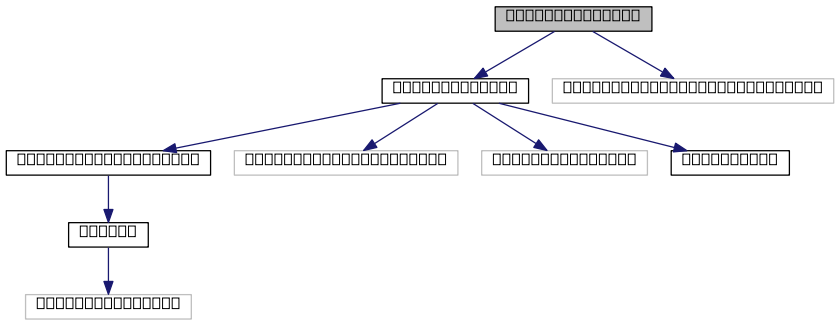
Definition at line 98 of file PcieConfig.h.

13.63 PeiDmiConfig.h File Reference

Config block defination for DMI.

```
#include <ConfigBlock.h>
#include <Defines/HostBridgeDefines.h>
```

Include dependency graph for PeiDmiConfig.h:



Classes

- struct DMI_EQ_PARAM

  Represent lane specific Dmi Gen3 equalization parameters.
- struct PEI_DMI_CONFIG

  The PEI_DMI_CONFIG block describes the expected configuration of the DMI.

Enumerations

- enum DMI_ASPM

  The values before AutoConfig match the setting of PCI Express Base Specification 1.1, please be careful for adding new feature.

13.63.1 Detailed Description

Config block defination for DMI.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2021 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

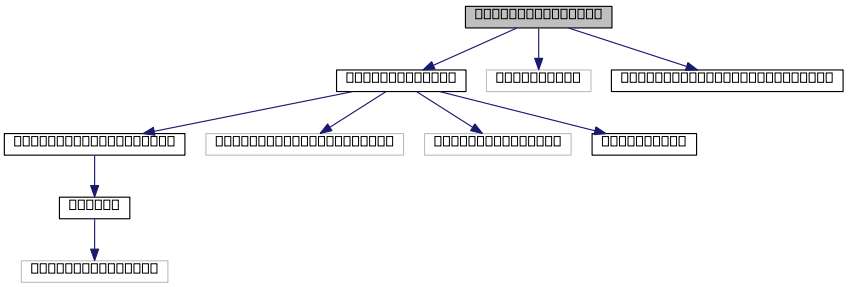
Specification Reference:

13.64 PeilTbtConfig.h File Reference

Header file for TBT PEI Policy.

```
#include <ConfigBlock.h>
#include <TcssInfo.h>
#include <PeiITbtGenericStructure.h>
```

Include dependency graph for PeilTbtConfig.h:



Classes

- struct _PEI_ITBT_CONFIG

  ITBT PEI configuration
  ***Revision 1**:*

Typedefs

- typedef struct _PEI_ITBT_CONFIG PEI_ITBT_CONFIG

  ITBT PEI configuration
  ***Revision 1**:*

13.64.1 Detailed Description

Header file for TBT PEI Policy.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2017 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.64.2 Typedef Documentation

13.64.2.1 PEI_ITBT_CONFIG

```
typedef struct _PEI_ITBT_CONFIG PEI_ITBT_CONFIG
```

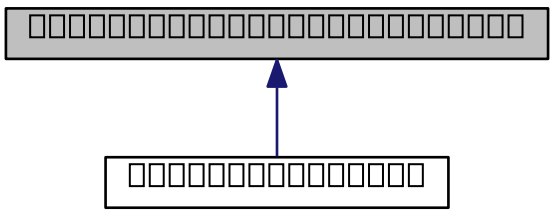
ITBT PEI configuration
Revision 1:

- Initial version. **Revision 2**:
- Add ITbtPcieTunnelingForUsb4, deprecated ITbtSecurityLevel.

13.65 PeiITbtGenericStructure.h File Reference

ITBT Policy definition to be referred in both PEI and DXE phase.

This graph shows which files directly or indirectly include this file:



Classes

- struct _ITBT_ROOTPORT_CONFIG

  iTBT RootPort Data Structure
- struct _ITBT_GENERIC_CONFIG

  ITBT Controller Data Structure to be used cross to RP and controller to be shared by CONFIG_BLOCK and HOB.

Typedefs

- typedef struct _ITBT_ROOTPORT_CONFIG ITBT_ROOTPORT_CONFIG

  iTBT RootPort Data Structure
- typedef struct _ITBT_GENERIC_CONFIG ITBT_GENERIC_CONFIG

  ITBT Controller Data Structure to be used cross to RP and controller to be shared by CONFIG_BLOCK and HOB.

13.65.1 Detailed Description

ITBT Policy definition to be referred in both PEI and DXE phase.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2017 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.66 PeiPreMemSiDefaultPolicy.h File Reference

This file defines the function to initialize default silicon policy PPI.

Classes

- struct _PEI_PREMEM_SI_DEFAULT_POLICY_INIT_PPI

  This PPI provides function to install default silicon policy.

Typedefs

- typedef EFI_STATUS(∗ PEI_PREMEM_POLICY_INIT) (VOID)

  Initialize and install default silicon policy PPI.

13.66.1 Detailed Description

This file defines the function to initialize default silicon policy PPI.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2019 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.67 PeiSiDefaultPolicy.h File Reference

This file defines the function to initialize default silicon policy PPI.

Classes

- struct _PEI_SI_DEFAULT_POLICY_INIT_PPI
  This PPI provides function to install default silicon policy.

Typedefs

- typedef EFI_STATUS(∗ PEI_POLICY_INIT) (VOID)
  Initialize and install default silicon policy PPI.

13.67.1 Detailed Description

This file defines the function to initialize default silicon policy PPI.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2019 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.68 PiHob.h File Reference

HOB related definitions in PI.

This graph shows which files directly or indirectly include this file:

Classes

- struct EFI_HOB_GENERIC_HEADER

  Describes the format and size of the data inside the HOB.
- struct EFI_HOB_HANDOFF_INFO_TABLE

  Contains general state information used by the HOB producer phase.
- struct EFI_HOB_MEMORY_ALLOCATION_HEADER

  EFI_HOB_MEMORY_ALLOCATION_HEADER describes the various attributes of the logical memory allocation.
- struct EFI_HOB_MEMORY_ALLOCATION

  Describes all memory ranges used during the HOB producer phase that exist outside the HOB list.
- struct EFI_HOB_MEMORY_ALLOCATION_STACK

  Describes the memory stack that is produced by the HOB producer phase and upon which all post-memory-installed executable content in the HOB producer phase is executing.
- struct EFI_HOB_MEMORY_ALLOCATION_BSP_STORE

  Defines the location of the boot-strap processor (BSP) BSPStore ("Backing Store Pointer Store").
- struct EFI_HOB_MEMORY_ALLOCATION_MODULE

  Defines the location and entry point of the HOB consumer phase.
- struct EFI_HOB_RESOURCE_DESCRIPTOR

  Describes the resource properties of all fixed, nonrelocatable resource ranges found on the processor host bus during the HOB producer phase.
- struct EFI_HOB_GUID_TYPE

  Allows writers of executable content in the HOB producer phase to maintain and manage HOBs with specific GUID.
- struct EFI_HOB_FIRMWARE_VOLUME

  Details the location of firmware volumes that contain firmware files.
- struct EFI_HOB_FIRMWARE_VOLUME2

  Details the location of a firmware volume that was extracted from a file within another firmware volume.
- struct EFI_HOB_FIRMWARE_VOLUME3

  Details the location of a firmware volume that was extracted from a file within another firmware volume.
- struct EFI_HOB_CPU

  Describes processor information, such as address space and I/O space capabilities.
- struct EFI_HOB_MEMORY_POOL

  Describes pool memory allocations.
- struct EFI_HOB_UEFI_CAPSULE

  Each UEFI capsule HOB details the location of a UEFI capsule.
- union EFI_PEI_HOB_POINTERS

  Union of all the possible HOB Types.

Macros

- #define EFI_HOB_HANDOFF_TABLE_VERSION 0x0009

  Value of version in EFI_HOB_HANDOFF_INFO_TABLE.

Typedefs

- typedef UINT32 EFI_RESOURCE_TYPE

  The resource type.
- typedef UINT32 EFI_RESOURCE_ATTRIBUTE_TYPE

  A type of recount attribute type.

13.68.1 Detailed Description

HOB related definitions in PI.

Copyright (c) 2006 - 2017, Intel Corporation. All rights reserved.
SPDX-License-Identifier: BSD-2-Clause-Patent

Revision Reference:

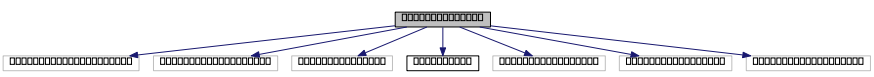
PI Version 1.9

13.69 PiMultiPhase.h File Reference

Include file matches things in PI for multiple module types.

```
#include <Pi/PiFirmwareVolume.h>
#include <Pi/PiFirmwareFile.h>
#include <Pi/PiBootMode.h>
#include <Pi/PiHob.h>
#include <Pi/PiDependency.h>
#include <Pi/PiStatusCode.h>
#include <Pi/PiS3BootScript.h>
```

Include dependency graph for PiMultiPhase.h:



Classes

- struct EFI_MMRAM_DESCRIPTOR

  Structure describing a MMRAM region and its accessibility attributes.
- struct _EFI_MM_RESERVED_MMRAM_REGION

  Structure describing a MMRAM region which cannot be used for the MMRAM heap.

Macros

- #define DXE_ERROR(StatusCode) (MAX_BIT | (MAX_BIT >> 2) | StatusCode)

  Produces an error code in the range reserved for use by the Platform Initialization Architecture Specification.
- #define EFI_REQUEST_UNLOAD_IMAGE DXE_ERROR (1)

  If this value is returned by an EFI image, then the image should be unloaded.
- #define EFI_NOT_AVAILABLE_YET DXE_ERROR (2)

  If this value is returned by an API, it means the capability is not yet installed/available/ready to use.
- #define PI_ENCODE_WARNING(a) ((MAX_BIT >> 2) | (a))

  Success and warning codes reserved for use by PI.
- #define PI_ENCODE_ERROR(a) (MAX_BIT | (MAX_BIT >> 2) | (a))

  Error codes reserved for use by PI.
- #define EFI_INTERRUPT_PENDING PI_ENCODE_ERROR (0)

  Return status codes defined in SMM CIS.
- #define EFI_MMRAM_OPEN 0x00000001

  MMRAM states and capabilities.

- #define EFI_AUTH_STATUS_PLATFORM_OVERRIDE 0x01

  Bitmask of values for Authentication Status.

Typedefs

- typedef struct _EFI_MM_RESERVED_MMRAM_REGION EFI_MM_RESERVED_MMRAM_REGION

  Structure describing a MMRAM region which cannot be used for the MMRAM heap.
- typedef VOID(∗ EFI_AP_PROCEDURE) (IN OUT VOID ∗Buffer)

  The function prototype for invoking a function on an Application Processor.
- typedef EFI_STATUS(∗ EFI_AP_PROCEDURE2) (IN VOID ∗ProcedureArgument)

  The function prototype for invoking a function on an Application Processor.

13.69.1 Detailed Description

Include file matches things in PI for multiple module types.

Copyright (c) 2006 - 2018, Intel Corporation. All rights reserved.
SPDX-License-Identifier: BSD-2-Clause-Patent

Revision Reference:

These elements are defined in UEFI Platform Initialization Specification 1.8.A

13.69.2 Macro Definition Documentation

13.69.2.1 DXE_ERROR

```
#define DXE_ERROR(
              StatusCode ) (MAX_BIT | (MAX_BIT >> 2) | StatusCode)
```

Produces an error code in the range reserved for use by the Platform Initialization Architecture Specification.

The supported 32-bit range is 0xA0000000-0xBFFFFFFF The supported 64-bit range is 0xA000000000000000-0xBFFFFFFFFFFFFFFF

Parameters

StatusCode	The status code value to convert into a warning code. StatusCode must be in the range 0x00000000..0x1FFFFFFF.

Returns

The value specified by StatusCode in the PI reserved range.

Definition at line 43 of file PiMultiPhase.h.

13.69.2.2 EFI_AUTH_STATUS_PLATFORM_OVERRIDE

```
#define EFI_AUTH_STATUS_PLATFORM_OVERRIDE 0x01
```

Bitmask of values for Authentication Status.

Authentication Status is returned from EFI_GUIDED_SECTION_EXTRACTION_PROTOCOL and the EFI_PEI_↩ GUIDED_SECTION_EXTRACTION_PPI

xx00 Image was not signed. xxx1 Platform security policy override. Assumes the same meaning as 0010 (the image was signed, the signature was tested, and the signature passed authentication test). 0010 Image was signed, the signature was tested, and the signature passed authentication test. 0110 Image was signed and the signature was not tested. 1010 Image was signed, the signature was tested, and the signature failed the authentication test.

Definition at line 91 of file PiMultiPhase.h.

13.69.2.3 PI_ENCODE_ERROR

```
#define PI_ENCODE_ERROR(
              a ) (MAX_BIT | (MAX_BIT >> 2) | (a))
```

Error codes reserved for use by PI.

Supported 32-bit range is 0xa0000000-0xbfffffff. Supported 64-bit range is 0xa000000000000000-0xbfffffffffffffff.

Definition at line 68 of file PiMultiPhase.h.

13.69.2.4 PI_ENCODE_WARNING

```
#define PI_ENCODE_WARNING(
              a ) ((MAX_BIT >> 2) | (a))
```

Success and warning codes reserved for use by PI.

Supported 32-bit range is 0x20000000-0x3fffffff. Supported 64-bit range is 0x2000000000000000-0x3fffffffffffffff.

Definition at line 61 of file PiMultiPhase.h.

13.69.3 Typedef Documentation

13.69.3.1 EFI_AP_PROCEDURE

```
typedef VOID( * EFI_AP_PROCEDURE) (IN OUT VOID *Buffer)
```

The function prototype for invoking a function on an Application Processor.

This definition is used by the UEFI MP Serices Protocol, and the PI SMM System Table.

Parameters

in,out	*Buffer*	The pointer to private data buffer.

Definition at line 197 of file PiMultiPhase.h.

13.69.3.2 EFI_AP_PROCEDURE2

```
typedef EFI_STATUS( * EFI_AP_PROCEDURE2) (IN VOID *ProcedureArgument)
```

The function prototype for invoking a function on an Application Processor.

This definition is used by the UEFI MM MP Serices Protocol.

Parameters

in	*ProcedureArgument*	The pointer to private data buffer.

Return values

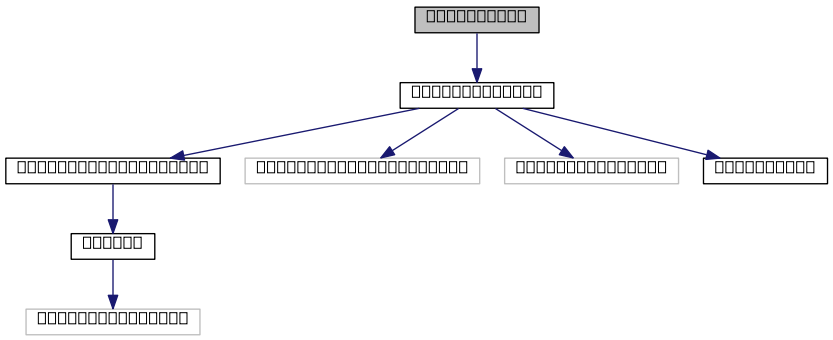
EFI_SUCCESS	Excutive the procedure successfully

Definition at line 213 of file PiMultiPhase.h.

13.70 PmConfig.h File Reference

Power Management policy.

```
#include <ConfigBlock.h>
```
Include dependency graph for PmConfig.h:



Classes

- struct PCH_WAKE_CONFIG

  This structure allows to customize PCH wake up capability from S5 or DeepSx by WOL, LAN, PCIE wake events.
- union PMC_LPM_S0IX_SUB_STATE_EN

  Low Power Mode Enable config.
- union PMC_GLOBAL_RESET_MASK

  Description of Global Reset Trigger/Event Mask register.
- struct PCH_PM_CONFIG

  The PCH_PM_CONFIG block describes expected miscellaneous power management settings.
- struct BCLK_CONFIG

  BCLK configuration.

Enumerations

- enum PCH_SLP_S4_MIN_ASSERT

13.70.1 Detailed Description

Power Management policy.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2015 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.70.2 Enumeration Type Documentation

13.70.2.1 PCH_SLP_S4_MIN_ASSERT

`enum PCH_SLP_S4_MIN_ASSERT`

Enumerator

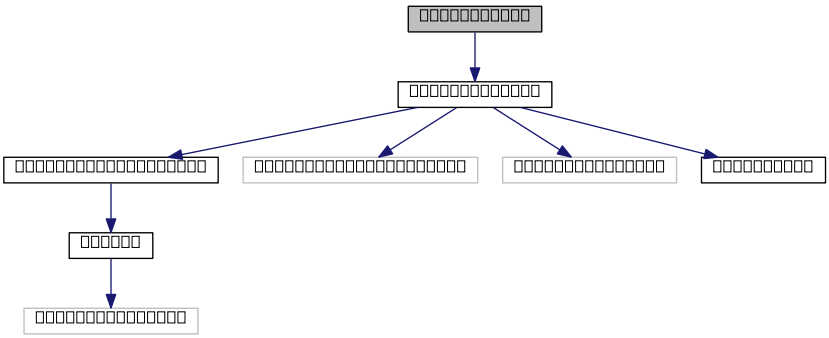
PchSlpS4PchTime	The time defined in PCH EDS Power Sequencing and Reset Signal Timings table.

Definition at line 69 of file PmConfig.h.

13.71 PsfConfig.h File Reference

```
#include <ConfigBlock.h>
```

Include dependency graph for PsfConfig.h:



Classes

- struct PSF_CONFIG

  PSF Configuration ***Revision 1****: - Initial version.*

13.71.1 Detailed Description

Copyright

INTEL CONFIDENTIAL Copyright (C) 2022 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

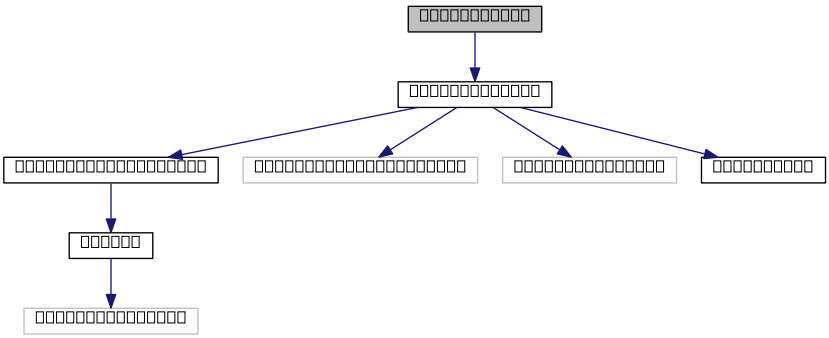
Specification Reference:

13.72 RtcConfig.h File Reference

RTC policy.

```
#include <ConfigBlock.h>
```

Include dependency graph for RtcConfig.h:



Classes

- struct RTC_CONFIG

  The RTC_CONFIG block describes the expected configuration of RTC configuration.

13.72.1 Detailed Description

RTC policy.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2018 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

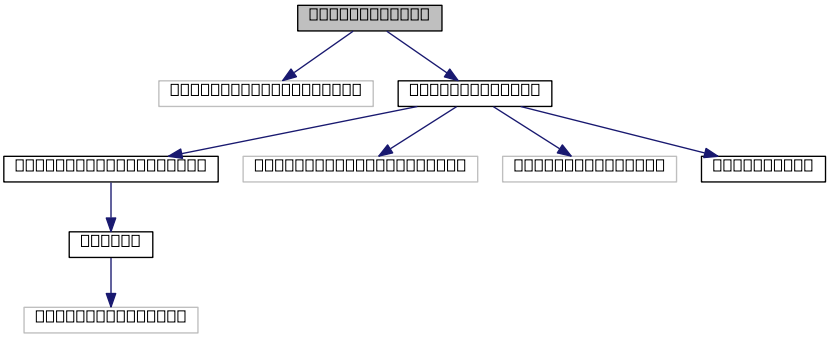
Specification Reference:

13.73 SataConfig.h File Reference

Sata policy.

```
#include <Library/PchLimits.h>
#include <ConfigBlock.h>
```

Include dependency graph for SataConfig.h:



Classes

- struct PCH_SATA_PORT_CONFIG

  This structure configures the features, property, and capability for each SATA port.
- struct SATA_THERMAL_THROTTLING

  This structure lists PCH supported SATA thermal throttling register setting for customization.
- struct SATA_CONFIG

  The SATA_CONFIG block describes the expected configuration of the SATA controllers.

13.73.1 Detailed Description

Sata policy.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2015 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

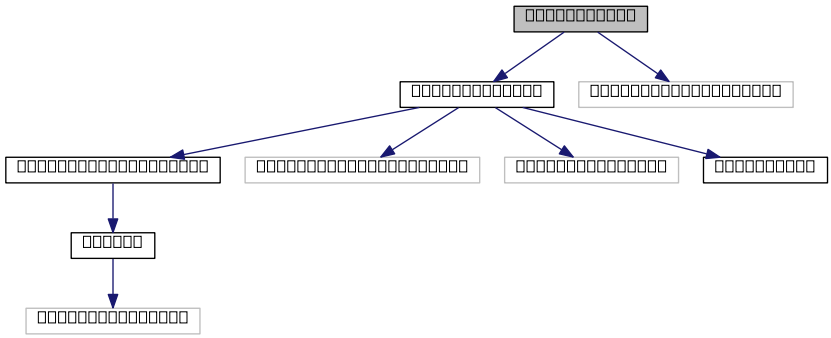
Specification Reference:

13.74 ScsConfig.h File Reference

Scs policy.

```
#include <ConfigBlock.h>
#include <Library/PchLimits.h>
```

Include dependency graph for ScsConfig.h:



Classes

- struct SCS_SD_DLL

  This structre holds the DLL configuration register values that will be programmed by RC if EnableCustomDlls field is set to TRUE.
- struct MUX_GPIO_PARAM

  SD GPIO settings.

13.74.1 Detailed Description

Scs policy.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2015 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

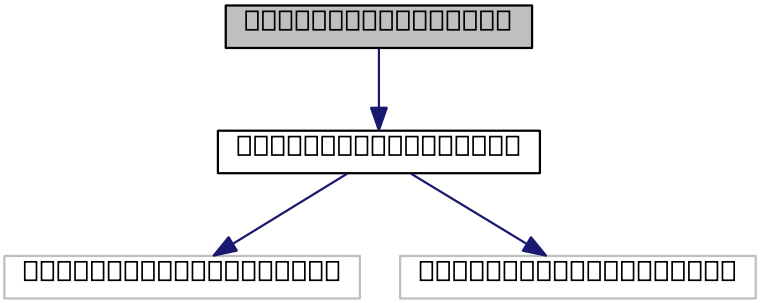
This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.75 SerialIoConfig.h File Reference

Serial IO policy.

```
#include <SerialIoDevices.h>
```
Include dependency graph for SerialIoConfig.h:



Classes

- struct SERIAL_IO_CONFIG

  The SERIAL_IO_CONFIG block provides the configurations to set the Serial IO controllers.

13.75.1 Detailed Description

Serial IO policy.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2016 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

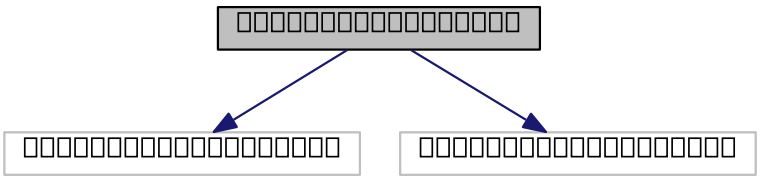
Specification Reference:

13.76 SerialIoDevices.h File Reference

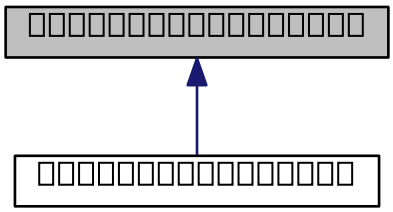
Serial IO policy.

```
#include <Protocol/SerialIo.h>
#include <Library/PchLimits.h>
```

Include dependency graph for SerialIoDevices.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct SPI_PIN_MUX

  SPI signals pin muxing settings.
- struct SERIAL_IO_SPI_CONFIG

  The SERIAL_IO_SPI_CONFIG provides the configurations to set the Serial IO SPI controller.
- struct SERIAL_IO_UART_ATTRIBUTES

  UART Settings.
- struct UART_PIN_MUX

  UART signals pin muxing settings.
- struct SERIAL_IO_UART_CONFIG

  Serial IO UART Controller Configuration.
- struct I2C_PIN_MUX

  I2C signals pin muxing settings.
- struct SERIAL_IO_I2C_CONFIG

Serial IO I2C Controller Configuration.

- struct MUX_GPIO_CONFIG

  I3C GPIO settings.

- struct SERIAL_IO_I3C_CONFIG

  The SERIAL_IO_I3C_CONFIG provides the configurations for Serial IO I3C controller.

Enumerations

- enum SERIAL_IO_SPI_MODE

  Available working modes for SerialIo SPI Host Controller.

- enum SERIAL_IO_CS_POLARITY

  Used to set Inactive/Idle polarity of Spi Chip Select.

- enum SERIAL_IO_UART_MODE

  Available working modes for SerialIo UART Host Controller.

- enum SERIAL_IO_UART_PG
- enum SERIAL_IO_I2C_MODE

  Available working modes for SerialIo I2C Host Controller.

- enum SERIAL_IO_I3C_MODE

  Available working modes for SerialIo I3C Host Controller.

13.76.1 Detailed Description

Serial IO policy.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2016 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.76.2 Enumeration Type Documentation

13.76.2.1 SERIAL_IO_I2C_MODE

```
enum SERIAL_IO_I2C_MODE
```

Available working modes for SerialIo I2C Host Controller.

```
 0: SerialIoI2cDisabled;
    - Device is placed in D3
    - Gpio configuration is skipped
    - PSF:
    !important! If given device is Function 0 and other higher functions on given device
                are enabled, PSF disabling is skipped. PSF default will remain and device PCI CFG Space will s
                This is needed to allow PCI enumerator access functions above 0 in a multifunction device.
```

1: SerialIoI2cPci;

- Gpio pin configuration in native mode for each assigned pin
- Device will be enabled in PSF
- Only BAR0 will be enabled

Definition at line 188 of file SerialIoDevices.h.

13.76.2.2 SERIAL_IO_I3C_MODE

```
enum SERIAL_IO_I3C_MODE
```

Available working modes for SerialIo I3C Host Controller.

```
 0: SerialIoI3cDisabled;
    - Device is placed in D3
    - Gpio configuration is skipped
    - Device will be disabled in PSF
```

1: SerialIoI3cPci;

- Gpio pin configuration in native mode for each assigned pin SDA and SCL
- Device will be enabled in PSF
- Only Bar 0 will be enabled 2: SerialIoI3cPhantom;
- Dedicated mode for I3C1, controls if GPIOs are enabled

Definition at line 238 of file SerialIoDevices.h.

13.76.2.3 SERIAL_IO_SPI_MODE

```
enum SERIAL_IO_SPI_MODE
```

Available working modes for SerialIo SPI Host Controller.

```
 0: SerialIoSpiDisabled;
    - Device is placed in D3
    - Gpio configuration is skipped
    - PSF:
    !important! If given device is Function 0 and other higher functions on given device
                are enabled, PSF disabling is skipped. PSF default will remain and device PCI CFG Space will s
                This is needed to allow PCI enumerator access functions above 0 in a multifunction device.
```

1: SerialIoSpiPci;

- Gpio pin configuration in native mode for each assigned pin
- Device will be enabled in PSF
- Only BAR0 will be enabled

Definition at line 44 of file SerialIoDevices.h.

13.76.2.4 SERIAL_IO_UART_MODE

```
enum SERIAL_IO_UART_MODE
```

Available working modes for SerialIo UART Host Controller.

```
 0: SerialIoUartDisabled;
    - Device is placed in D3
    - Gpio configuration is skipped
    - PSF:
    !important! If given device is Function 0 and other higher functions on given device
                are enabled, PSF disabling is skipped. PSF default will remain and device PCI CFG Space will s
                This is needed to allow PCI enumerator access functions above 0 in a multifunction device.
```

1: SerialIoUartPci;

- Designated for Serial IO UART OS driver usage
- Gpio pin configuration in native mode for each assigned pin
- Device will be enabled in PSF
- Only BAR0 will be enabled 2: SerialIoUartHidden;
- Designated for BIOS and/or DBG2 OS kernel debug
- Gpio pin configuration in native mode for each assigned pin
- Device disabled in the PSF
- Both BARs are enabled, BAR1 becomes devices Pci Config Space
- BAR0 assigned from the global PCH reserved memory range, reported as motherboard resource by SIRC

Note

If this controller is located at function 0 and it's mode is set to hidden it will not be visible in the PCI space. 3: SerialIoUartCom;

- Designated for 16550/PNP0501 compatible COM device
- Gpio pin configuration in native mode for each assigned pin
- Device disabled in the PSF
- Both BARs are enabled, BAR1 becomes devices Pci Config Space
- BAR0 assigned from the global PCH reserved memory range, reported as motherboard resource by SIRC 4: SerialIoUartSkipInit;
- Gpio configuration is skipped
- PSF configuration is skipped
- BAR assignemnt is skipped
- D-satate setting is skipped

Definition at line 118 of file SerialIoDevices.h.

13.76.2.5 SERIAL_IO_UART_PG

`enum SERIAL_IO_UART_PG`

Enumerator

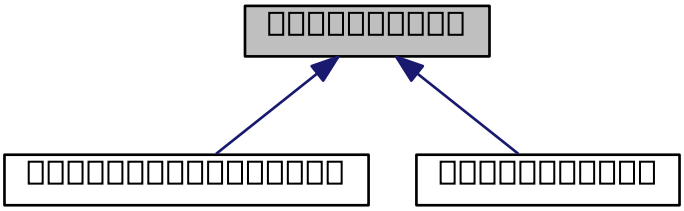
SerialIoUartPgDisabled	No _PS0/_PS3 support, device left in D0, after initialization.
SerialIoUartPgEnabled	In mode: SerialIoUartCom; _PS0/_PS3 that supports getting device out of reset In mode: SerialIoUartPci Keeps UART in D0 and assigns Fixed MMIO for SEC/PEI usage only.
SerialIoUartPgAuto	_PS0 and _PS3, detection through ACPI if device was initialized prior to first PG. If it was used (DBG2) PG is disabled,

Definition at line 161 of file SerialIoDevices.h.

13.77 SiConfig.h File Reference

Si Config Block.

This graph shows which files directly or indirectly include this file:



Classes

- struct SI_CONFIG

  The Silicon Policy allows the platform code to publish a set of configuration information that the RC drivers will use to configure the silicon hardware.

- struct SVID_SID_VALUE

  Subsystem Vendor ID / Subsystem ID.

13.77.1 Detailed Description

Si Config Block.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2016 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

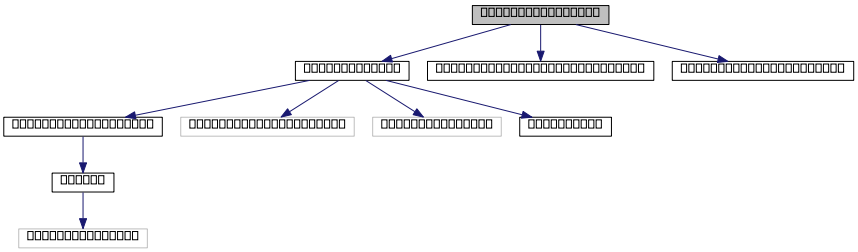
Specification Reference:

13.78 SiPolicyStruct.h File Reference

Intel reference code configuration policies.

```
#include <ConfigBlock.h>
#include <ConfigBlock/SiPreMemConfig.h>
#include <ConfigBlock/SiConfig.h>
```

Include dependency graph for SiPolicyStruct.h:



Classes

- struct _SI_PREMEM_POLICY_STRUCT

  SI Policy PPI in Pre-Mem
  All SI config block change history will be listed here

  .

- struct _SI_POLICY_STRUCT

  SI Policy PPI
  All SI config block change history will be listed here

  .

Macros

- #define SI_POLICY_REVISION 1

  Silicon Policy revision number Any change to this structure will result in an update in the revision number.

- #define SI_PREMEM_POLICY_REVISION 1

  Silicon pre-memory Policy revision number Any change to this structure will result in an update in the revision number.

13.78.1 Detailed Description

Intel reference code configuration policies.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2014 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.78.2 Macro Definition Documentation

13.78.2.1 SI_POLICY_REVISION

```
#define SI_POLICY_REVISION 1
```

Silicon Policy revision number Any change to this structure will result in an update in the revision number.

This member specifies the revision of the Silicon Policy. This field is used to indicate change to the policy structure.

Revision 1:

- Initial version.

Definition at line 38 of file SiPolicyStruct.h.

13.78.2.2 SI_PREMEM_POLICY_REVISION

```
#define SI_PREMEM_POLICY_REVISION 1
```

Silicon pre-memory Policy revision number Any change to this structure will result in an update in the revision number.

Revision 1:

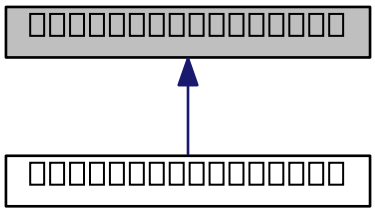
- Initial version.

Definition at line 47 of file SiPolicyStruct.h.

13.79 SiPreMemConfig.h File Reference

Si Config Block PreMem.

This graph shows which files directly or indirectly include this file:



Classes

- struct SI_PREMEM_CONFIG

  The Silicon PreMem Policy allows the platform code to publish a set of configuration information that the RC drivers will use to configure the silicon hardware.

13.79.1 Detailed Description

Si Config Block PreMem.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2017 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

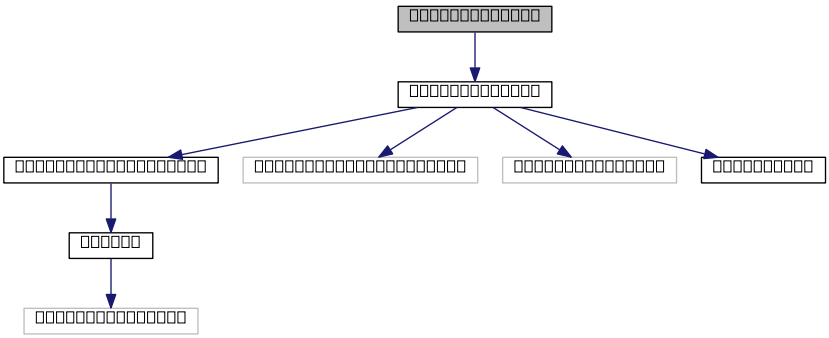
Specification Reference:

13.80 SmbusConfig.h File Reference

Smbus policy.

```
#include <ConfigBlock.h>
```

Include dependency graph for SmbusConfig.h:



Classes

- struct PCH_SMBUS_PREMEM_CONFIG

  The SMBUS_CONFIG block lists the reserved addresses for non-ARP capable devices in the platform.

13.80.1 Detailed Description

Smbus policy.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2015 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

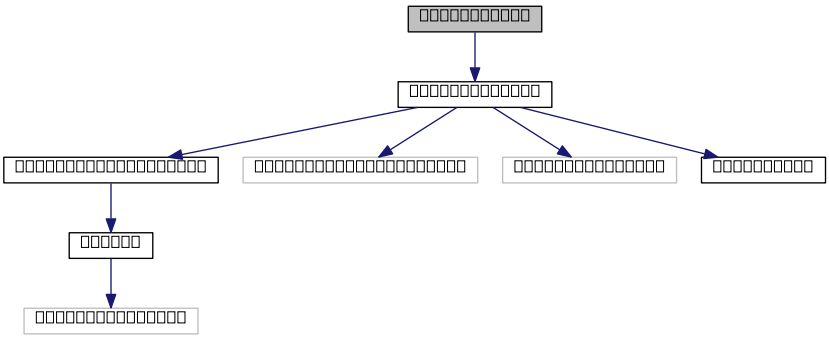
Specification Reference:

13.81 SpiConfig.h File Reference

PCH SPI Flash Controller config block.

```
#include <ConfigBlock.h>
```

Include dependency graph for SpiConfig.h:



Classes

- struct SPI_CONFIG

  Basic configuration for option features of PCH SPI Flash controller.

13.81.1 Detailed Description

PCH SPI Flash Controller config block.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2020 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

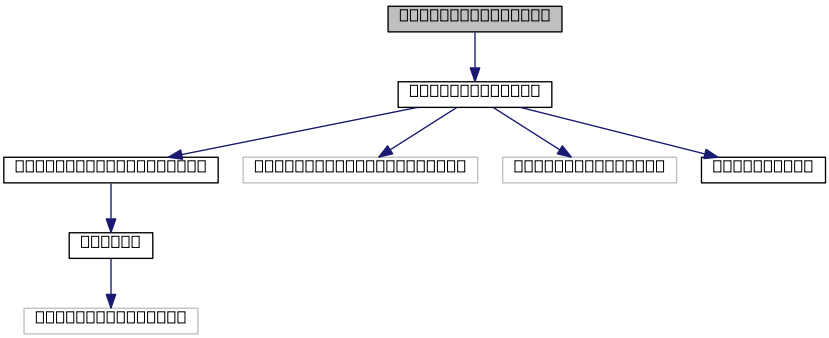
Specification Reference:

13.82 SwizzleConfig.h File Reference

Config block for interrupt swizzle functionality configuration.

```
#include <ConfigBlock.h>
```

Include dependency graph for SwizzleConfig.h:



Classes

- struct SWIZZLE_CONFIG

  This config block describes how to configure Interrupt Swizzling functionality in EAH.

13.82.1 Detailed Description

Config block for interrupt swizzle functionality configuration.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2022 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

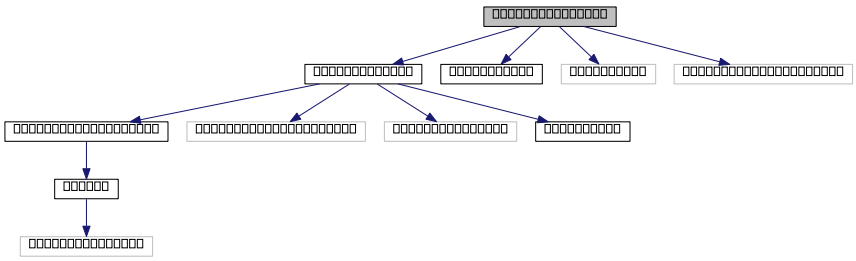
Specification

13.83 TcssPeiConfig.h File Reference

TCSS PEI policy.

```
#include <ConfigBlock.h>
#include <UsbConfig.h>
#include <TcssInfo.h>
#include <TcssPeiPreMemConfig.h>
```

Include dependency graph for TcssPeiConfig.h:



Classes

- struct PMC_INTERFACE_CONFIG

  The PMC_INTERFACE_CONFIG block describes interaction between BIOS and PMC.
- struct TCSS_PCIE_PORT_POLICY

  The TCSS_PCIE_PORT_POLICY block describes PCIe settings for TCSS.
- struct TCSS_PCIE_PEI_POLICY

  TCSS_PCIE_PEI_POLICY describes PCIe port settings for TCSS.
- struct TCSS_IOM_PEI_CONFIG

  The TCSS_IOM_PEI_CONFIG block describes IOM settings for TCSS.
- struct TCSS_MISC_PEI_CONFIG

  The TCSS_MISC_PEI_CONFIG block describes MISC settings for TCSS.
- struct TCSS_PEI_CONFIG

  The TCSS_PEI_CONFIG block describes TCSS settings for SA.

Functions

- VOID TcssConfigDump (IN TCSS_PEI_CONFIG *TcssPeiConfig)

  Print TCSS_PEI_CONFIG and serial out.

13.83.1 Detailed Description

TCSS PEI policy.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2016 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.83.2 Function Documentation

13.83.2.1 TcssConfigDump()

```
VOID TcssConfigDump (
             IN TCSS_PEI_CONFIG * TcssPeiConfig )
```

Print TCSS_PEI_CONFIG and serial out.

Parameters

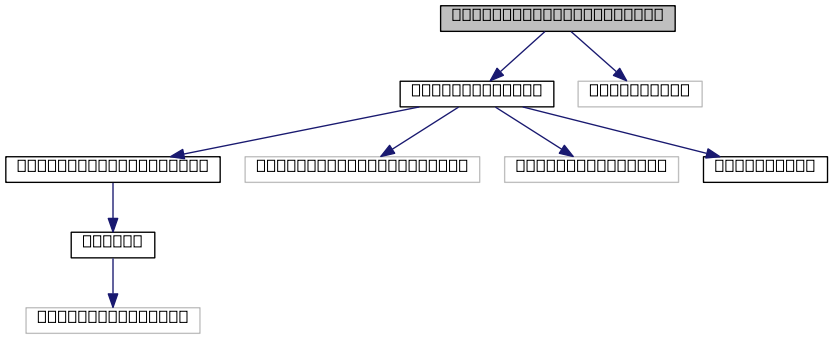
in	*TcssPeiConfig*	Pointer to a TCSS_PEI_CONFIG

13.84 TcssPeiPreMemConfig.h File Reference

TCSS PEI PreMem policy.

```
#include <ConfigBlock.h>
#include <TcssInfo.h>
```

Include dependency graph for TcssPeiPreMemConfig.h:



Classes

- union TCSS_USBTC_PEI_PERMEM_CONFIG

  The TCSS_USBTC_PEI_PERMEM_CONFIG block describes IOM settings for TCSS.
- struct IOM_AUX_ORI_GPIO_PAD_CONFIG

  The IOM_AUX_ORI_GPIO_PAD_CONFIG describes IOM TypeC port map GPIO pin.
- struct TCSS_IOM_PEI_PERMEM_CONFIG

  The TCSS_IOM_PEI_PERMEM_CONFIG block describes IOM settings for TCSS.
- struct TCSS_PEI_PREMEM_CONFIG

  This configuration block describes TCSS settings.

Macros

- #define _TCSS_PORT_CAP

  TCSS USBC SI Policy capability.

Enumerations

- enum TCSS_PORT_CAP

Functions

- VOID TcssPreMemConfigDump (IN TCSS_PEI_PREMEM_CONFIG *TcssPeiPreMemConfig, IN UINT8 MaxTcssPortNumber)

  Print TCSS_PEI_PREMEM_CONFIG and serial out.

13.84.1 Detailed Description

TCSS PEI PreMem policy.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2024 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.84.2 Enumeration Type Documentation

13.84.2.1 TCSS_PORT_CAP

```
enum TCSS_PORT_CAP
```

Enumerator

UsbCDisable	The USBC lane function disabled.
DpOnly	The USBC lane supports Display and Display tunneling.
NoThunderbolt	The USBC lane supports Display, native USB3.
NoPcie	The USBC lane supports Display, native USB3, Display tunneling and USB3 tunneling.
FullFunction	The USBC lane supports Display, native USB3, Display tunneling, USB3 tunneling and PCIe tunneling.

Definition at line 39 of file TcssPeiPreMemConfig.h.

13.84.3 Function Documentation

13.84.3.1 TcssPreMemConfigDump()

```
VOID TcssPreMemConfigDump (
             IN TCSS_PEI_PREMEM_CONFIG * TcssPeiPreMemConfig,
             IN UINT8 MaxTcssPortNumber )
```

Print TCSS_PEI_PREMEM_CONFIG and serial out.

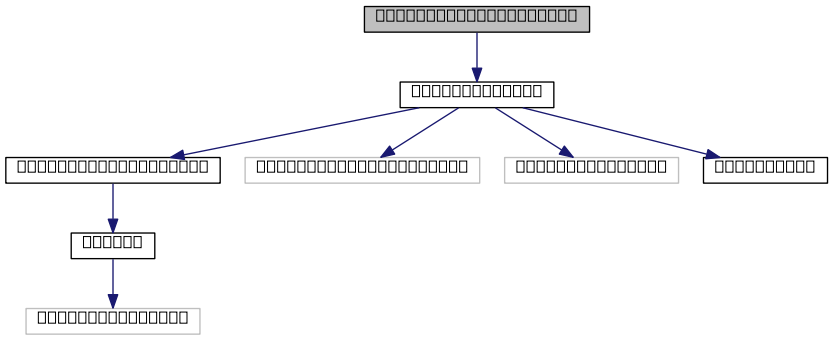
Parameters

in	*TcssPeiPreMemConfig*	Pointer to a TCSS_PEI_PREMEM_CONFIG
in	*MaxTcssPortNumber*	Max number of TCSS Root Ports

13.85 TelemetryPeiConfig.h File Reference

Configurations for Telemetry.

`#include <ConfigBlock.h>`
Include dependency graph for TelemetryPeiConfig.h:



Classes

- struct TELEMETRY_PEI_PREMEM_CONFIG

  This configuration block describes Telemetry settings in PreMem.
- struct TELEMETRY_PEI_CONFIG

  This configuration block describes Telemetry settings in PostMem.

13.85.1 Detailed Description

Configurations for Telemetry.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2019 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

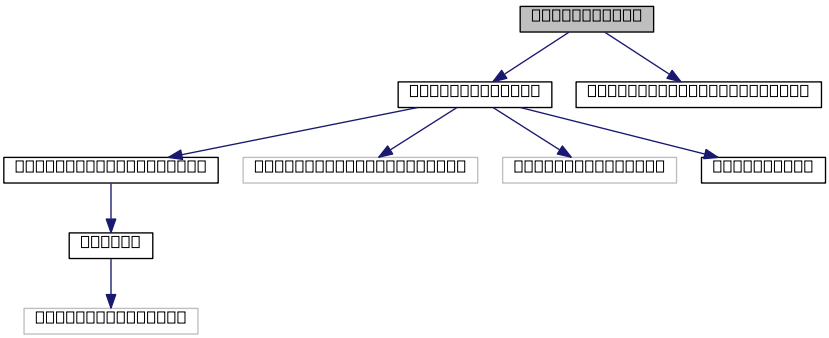
Specification Reference:

13.86 ThcConfig.h File Reference

Touch Host Controller policy.

```
#include <ConfigBlock.h>
#include <ConfigBlock/SiConfig.h>
```

Include dependency graph for ThcConfig.h:



Classes

- struct THC_HID_OVER_SPI

  THC Hid Over SPI mode related settings.
- struct THC_RESET

  THC Reset Pad related settings.
- struct THC_HID_OVER_I2C

  THC Hid Over I2C mode related settings.
- struct THC_PORT_SUBSYTEM_ID

  Structure to hold Sub System ID for port.
- struct THC_WOT_PIN_CONFIG

  Structure to hold Wake On Touch pin configuration for GPIO resource in ACPI.
- struct THC_PORT

  Port Configuration structure required for each Port that THC might use.
- struct THC_CONFIG

  THC_CONFIG block provides the configurations forTouch Host Controllers.
- struct THC_DATA_SSID_CONFIG_HOB

  THC SSID Data Hob.

Enumerations

- enum THC_PORT_ASSIGNMENT

  Available Port Assignments.
- enum THC_MODE

  Available modes.
- enum THC_PORT_CLOCK

  Available Port Clock configurations.

13.86.1 Detailed Description

Touch Host Controller policy.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2017 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.86.2 Enumeration Type Documentation

13.86.2.1 THC_MODE

`enum THC_MODE`

Available modes.

Enumerator

Thc	Intel THC protocol.
HidOverSpi	Hid Over SPI protocol.
HidOverI2c	Hid Over I2C protocol.

Definition at line 47 of file ThcConfig.h.

13.86.2.2 THC_PORT_ASSIGNMENT

`enum THC_PORT_ASSIGNMENT`

Available Port Assignments.

Enumerator

ThcAssignmentNone	None of the avaialbe controllers assigned.
ThcAssignmentThc0	Port assigned to THC0.
ThcAssignmentThc1	Port assigned to THC1.

Definition at line 38 of file ThcConfig.h.

13.87 ThermalConfig.h File Reference

Thermal policy.

Classes

- struct THERMAL_THROTTLE_LEVELS

  This structure lists PCH supported throttling register setting for custimization.
- struct DMI_HW_WIDTH_CONTROL

  This structure allows to customize DMI HW Autonomous Width Control for Thermal and Mechanical spec design.
- struct THERMAL_CONFIG

  The THERMAL_CONFIG block describes the expected configuration of the Thermal IP block.

13.87.1 Detailed Description

Thermal policy.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2015 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

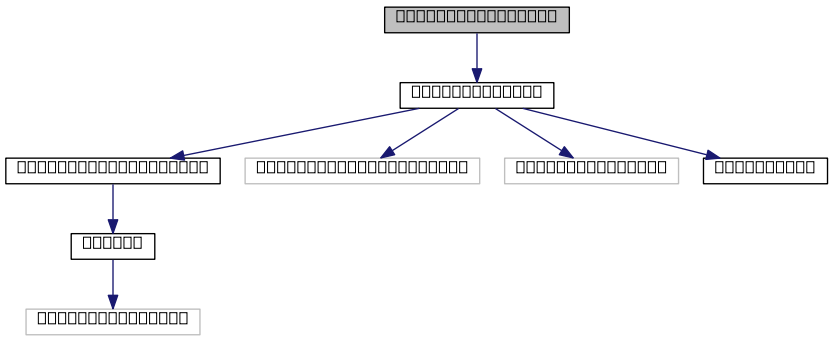
Specification Reference:

13.88 TraceHubConfig.h File Reference

Configurations for TraceHub.

```
#include <ConfigBlock.h>
```

Include dependency graph for TraceHubConfig.h:



Classes

- struct TRACE_HUB_CONFIG

  TRACE_HUB_CONFIG block describes TraceHub settings.
- struct PCH_TRACE_HUB_PREMEM_CONFIG

  PCH Trace Hub PreMem Configuration Contains Trace Hub settings for PCH side tracing ***Revision 1****: - Initial version.*
- struct TRACE_HUB_PREMEM_CONFIG

  Trace Hub PreMem Configuration Contains Trace Hub settings ***Revision 1****: - Initial version.*

Enumerations

- enum TRACE_HUB_ENABLE_MODE

  The TRACE_HUB_ENABLE_MODE describes TraceHub mode of operation.
- enum TRACE_BUFFER_SIZE

  The TRACE_BUFFER_SIZE describes the desired TraceHub buffer size.

13.88.1 Detailed Description

Configurations for TraceHub.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2018 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

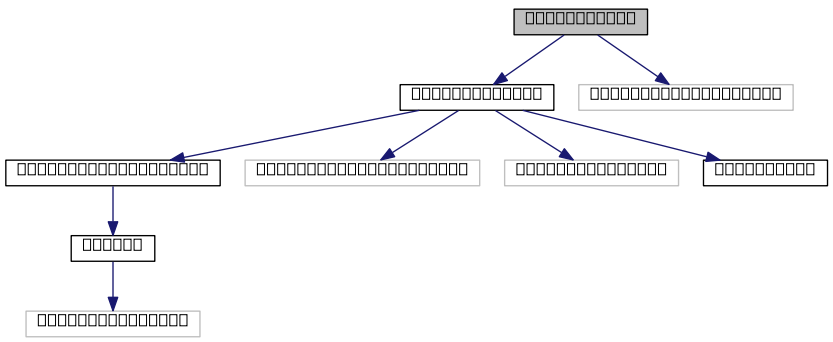
Specification Reference:

13.89 TsnConfig.h File Reference

TSN Config policy.

```
#include <ConfigBlock.h>
#include <Library/PchLimits.h>
```

Include dependency graph for TsnConfig.h:



Classes

- struct TSN_CONFIG

  TSN Config settings.

Macros

- #define TSN_CONFIG_REVISION 4

  *The TSN_CONFIG block describes policies related to Time Sensitive Networking(TSN) **Revision 1**:*

13.89.1 Detailed Description

TSN Config policy.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2019 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.89.2 Macro Definition Documentation

13.89.2.1 TSN_CONFIG_REVISION

```
#define TSN_CONFIG_REVISION 4
```

The TSN_CONFIG block describes policies related to Time Sensitive Networking(TSN) **Revision 1**:

- Initial version. **Revision 2**:
- Added MultiVcEnable **Revision 3**:
- Added Config **Revision 4**:
- Created MAC Addr Array for each port
- Created Enable Array for each port

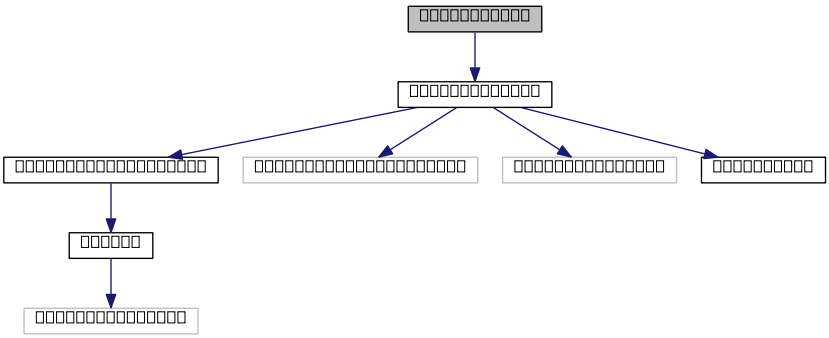
Definition at line 42 of file TsnConfig.h.

13.90 TxtConfig.h File Reference

TXT PreMemory Config Block.

```
#include <ConfigBlock.h>
```

Include dependency graph for TxtConfig.h:



Classes

- struct TXT_PREMEM_CONFIG

  CPU TXT PreMemory Configuration Structure.

13.90.1 Detailed Description

TXT PreMemory Config Block.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2021 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

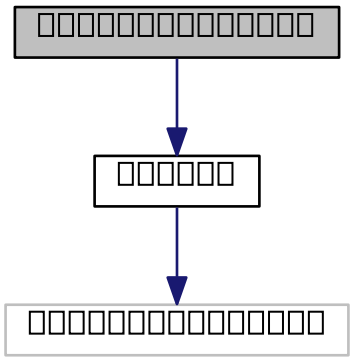
Specification Reference:

13.91 UefiBaseType.h File Reference

Defines data types and constants introduced in UEFI.

```
#include <Base.h>
```

Include dependency graph for UefiBaseType.h:



This graph shows which files directly or indirectly include this file:

Classes

- struct EFI_TIME

  EFI Time Abstraction: Year: 1900 - 9999 Month: 1 - 12 Day: 1 - 31 Hour: 0 - 23 Minute: 0 - 59 Second: 0 - 59 Nanosecond: 0 - 999,999,999 TimeZone: -1440 to 1440 or 2047.
- struct EFI_MAC_ADDRESS

  32-byte buffer containing a network Media Access Control address.
- union EFI_IP_ADDRESS

  16-byte buffer aligned on a 4-byte boundary.

Macros

- #define EFIERR(_a) ENCODE_ERROR(_a)

  Define macro to encode the status code.
- #define EFI_SIZE_TO_PAGES(Size) (((Size) >> EFI_PAGE_SHIFT) + (((Size) & EFI_PAGE_MASK) ? 1 : 0))

  Macro that converts a size, in bytes, to a number of EFI_PAGESs.
- #define EFI_PAGES_TO_SIZE(Pages) ((Pages) << EFI_PAGE_SHIFT)

  Macro that converts a number of EFI_PAGEs to a size in bytes.
- #define EFI_IMAGE_MACHINE_IA32 0x014C

PE32+ Machine type for IA32 UEFI images.

- #define EFI_IMAGE_MACHINE_IA64 0x0200

  PE32+ Machine type for IA64 UEFI images.
- #define EFI_IMAGE_MACHINE_EBC 0x0EBC

  PE32+ Machine type for EBC UEFI images.
- #define EFI_IMAGE_MACHINE_X64 0x8664

  PE32+ Machine type for X64 UEFI images.
- #define EFI_IMAGE_MACHINE_ARMTHUMB_MIXED 0x01C2

  PE32+ Machine type for ARM mixed ARM and Thumb/Thumb2 images.
- #define EFI_IMAGE_MACHINE_AARCH64 0xAA64

  PE32+ Machine type for AARCH64 A64 images.
- #define EFI_IMAGE_MACHINE_RISCV32 0x5032

  PE32+ Machine type for RISC-V 32/64/128.
- #define EFI_IMAGE_MACHINE_LOONGARCH32 0x6232

  PE32+ Machine type for LoongArch 32/64 images.

- #define EFI_SUCCESS RETURN_SUCCESS

  Enumeration of EFI_STATUS.

- #define EFI_NETWORK_UNREACHABLE EFIERR(100)

  ICMP error definitions.

- #define EFI_CONNECTION_FIN EFIERR(104)

  Tcp connection status definitions.

Typedefs

- typedef GUID EFI_GUID

  128-bit buffer containing a unique identifier value.
- typedef RETURN_STATUS EFI_STATUS

  Function return status for EFI API.
- typedef VOID * EFI_HANDLE

  A collection of related interfaces.
- typedef VOID * EFI_EVENT

  Handle to an event structure.
- typedef UINTN EFI_TPL

  Task priority level.
- typedef UINT64 EFI_LBA

  Logical block address.
- typedef UINT64 EFI_PHYSICAL_ADDRESS

  64-bit physical memory address.
- typedef UINT64 EFI_VIRTUAL_ADDRESS

  64-bit virtual memory address.
- typedef IPv4_ADDRESS EFI_IPv4_ADDRESS

  4-byte buffer.
- typedef IPv6_ADDRESS EFI_IPv6_ADDRESS

  16-byte buffer.

13.91.1 Detailed Description

Defines data types and constants introduced in UEFI.

Copyright (c) 2006 - 2021, Intel Corporation. All rights reserved.
Portions copyright (c) 2011 - 2016, ARM Ltd. All rights reserved.
Copyright (c) 2020, Hewlett Packard Enterprise Development LP. All rights reserved.
Copyright (c) 2022, Loongson Technology Corporation Limited. All rights reserved.

SPDX-License-Identifier: BSD-2-Clause-Patent

13.91.2 Macro Definition Documentation

13.91.2.1 EFI_PAGES_TO_SIZE

```
#define EFI_PAGES_TO_SIZE(
              Pages ) ((Pages) << EFI_PAGE_SHIFT)
```

Macro that converts a number of EFI_PAGEs to a size in bytes.

Parameters

Pages	The number of EFI_PAGES. This parameter is assumed to be type UINTN. Passing in a parameter that is larger than UINTN may produce unexpected results.

Returns

The number of bytes associated with the number of EFI_PAGEs specified by Pages.

Definition at line 213 of file UefiBaseType.h.

13.91.2.2 EFI_SIZE_TO_PAGES

```
#define EFI_SIZE_TO_PAGES(
              Size ) (((Size) >> EFI_PAGE_SHIFT) + (((Size) & EFI_PAGE_MASK) ? 1 :  0))
```

Macro that converts a size, in bytes, to a number of EFI_PAGESs.

Parameters

Size	A size in bytes. This parameter is assumed to be type UINTN. Passing in a parameter that is larger than UINTN may produce unexpected results.

Returns

The number of EFI_PAGESs associated with the number of bytes specified by Size.

Definition at line 200 of file UefiBaseType.h.

13.91.3 Typedef Documentation

13.91.3.1 EFI_IPv4_ADDRESS

```
typedef IPv4_ADDRESS EFI_IPv4_ADDRESS
```

4-byte buffer.

An IPv4 internet protocol address.

Definition at line 85 of file UefiBaseType.h.

13.91.3.2 EFI_IPv6_ADDRESS

```
typedef IPv6_ADDRESS EFI_IPv6_ADDRESS
```

16-byte buffer.

An IPv6 internet protocol address.

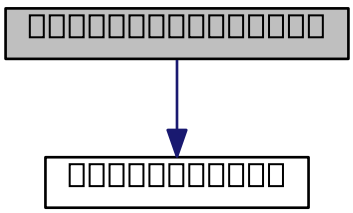
Definition at line 90 of file UefiBaseType.h.

13.92 Usb2PhyConfig.h File Reference

USB2 PHY configuration policy.

```
#include <UsbConfig.h>
```

Include dependency graph for Usb2PhyConfig.h:



Classes

- struct USB2_PHY_PARAMETERS

  This structure configures per USB2 AFE settings.
- struct USB2_PHY_CONFIG

  This structure holds info on how to tune electrical parameters of USB2 ports based on board layout.

13.92.1 Detailed Description

USB2 PHY configuration policy.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2018 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

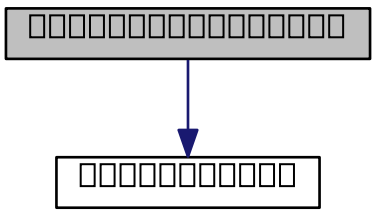
Specification Reference:

13.93 Usb3HsioConfig.h File Reference

USB3 Mod PHY configuration policy.

```
#include <UsbConfig.h>
```

Include dependency graph for Usb3HsioConfig.h:



Classes

- struct HSIO_PARAMETERS

  This structure describes USB3 Port N configuration parameters.
- struct USB3_HSIO_CONFIG

  Structure for holding USB3 tuning parameters.

13.93.1 Detailed Description

USB3 Mod PHY configuration policy.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2018 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

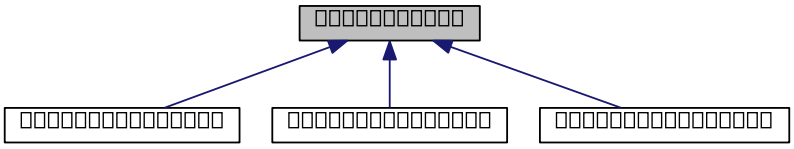
This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.94 UsbConfig.h File Reference

Common USB policy shared between PCH and CPU Contains general features settings for xHCI and xDCI.

This graph shows which files directly or indirectly include this file:



Classes

- struct USB2_PORT_CONFIG

  This structure configures per USB2.0 port settings like enabling and overcurrent protection.
- struct USB3_PORT_CONFIG

  This structure configures per USB3.x port settings like enabling and overcurrent protection.
- struct XDCI_CONFIG

  The XDCI_CONFIG block describes the configurations of the xDCI Usb Device controller.
- struct USB_CONFIG

  This member describes the expected configuration of the USB controller, Platform modules may need to refer Setup options, schematic, BIOS specification to update this field.

Macros

- #define USB_OC_MAX_PINS 16

  Total OC pins number (both physical and virtual)

13.94.1 Detailed Description

Common USB policy shared between PCH and CPU Contains general features settings for xHCI and xDCI.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2017 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

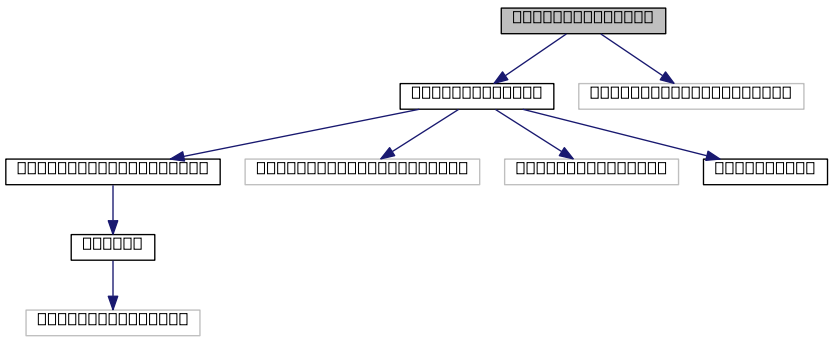
Specification Reference:

13.95 VmdPeiConfig.h File Reference

VMD PEI policy VMD Config Block.

```
#include <ConfigBlock.h>
#include <Library/VmdInfoLib.h>
```

Include dependency graph for VmdPeiConfig.h:



Classes

- struct VMD_PEI_CONFIG

  This configuration block is to configure VMD related variables used in PostMem PEI.

13.95.1 Detailed Description

VMD PEI policy VMD Config Block.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2017 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

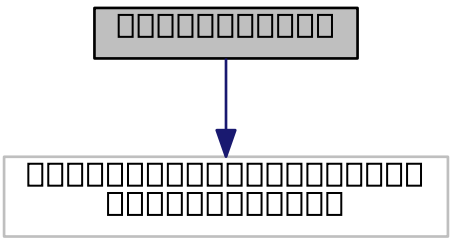
Specification Reference:

13.96 VtdConfig.h File Reference

VT-d policy definitions.

```
#include <ConfigBlock/Vtd/Ver1/VtdConfig.h>
```

Include dependency graph for VtdConfig.h:



13.96.1 Detailed Description

VT-d policy definitions.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2014 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

13.97 WatchDogConfig.h File Reference

WatchDog policy.

Classes

- struct PCH_WDT_PREMEM_CONFIG

  This policy clears status bits and disable watchdog, then lock the WDT registers.

13.97.1 Detailed Description

WatchDog policy.

Copyright

INTEL CONFIDENTIAL Copyright (C) 2015 Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you ("License"). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Specification Reference:

Index

_BASE_INT_SIZE_OF
    Base.h, 364
_CONFIG_BLOCK, 67
_CONFIG_BLOCK_HEADER, 68
_CONFIG_BLOCK_TABLE_STRUCT, 69
_DXE_ITBT_CONFIG, 70
_DXE_ITBT_GENERIC_CONFIG, 71
_EFI_MM_RESERVED_MMRAM_REGION, 71
    MmramReservedSize, 72
    MmramReservedStart, 72
_EFI_PEI_MP_SERVICES_PPI, 73
_INT_SIZE_OF
    Base.h, 364
_ITBT_GENERIC_CONFIG, 73
    ITbtForcePowerOnTimeoutInMs, 74
_ITBT_ROOTPORT_CONFIG, 74
_LIST_ENTRY, 75
_PEI_ITBT_CONFIG, 75
_PEI_PREMEM_SI_DEFAULT_POLICY_INIT_PPI, 76
_PEI_SI_DEFAULT_POLICY_INIT_PPI, 77
_SI_POLICY_STRUCT, 77
_SI_PREMEM_POLICY_STRUCT, 79

ABS
    Base.h, 365
ActiveCoreCount
    CPU_INIT_PREMEM_CONFIG, 95
ActiveLpAtomCoreCount
    CPU_INIT_PREMEM_CONFIG, 95
ActiveSmallCoreCount
    CPU_INIT_PREMEM_CONFIG, 95
AdaptiveVoltage
    OVERCLOCKING_PREMEM_CONFIG, 223
AddConfigBlock
    ConfigBlockLib.h, 387
ADDRESS_IS_ALIGNED
    Base.h, 365
AddressingMode
    THC_HID_OVER_I2C, 334
ADR_CONFIG, 80
ADR_GLOBAL_RESET_ENABLE, 81
AdrConfig.h, 357
AetEnabled
    TRACE_HUB_CONFIG, 341
ALIGN_POINTER
    Base.h, 366
ALIGN_VALUE
    Base.h, 366
ALIGN_VALUE_ADDEND
    Base.h, 367
ALIGN_VARIABLE
    Base.h, 367
ALIGNOF
    Base.h, 367
AMT_DXE_CONFIG, 82
AMT_PEI_CONFIG, 83
    AmtEnabled, 84
    WatchDogEnabled, 84
    WatchDogTimerBios, 85
    WatchDogTimerOs, 85
AmtConfig.h, 358
AmtEnabled
    AMT_PEI_CONFIG, 84
ANALYZER_NORETURN
    Base.h, 368
ANALYZER_UNREACHABLE
    Base.h, 368
ARRAY_SIZE
    Base.h, 368
AtomClusterRatio
    OVERCLOCKING_PREMEM_CONFIG, 223
AudioLinkDmic
    HDAUDIO_PREMEM_CONFIG, 162
AudioLinkHda
    HDAUDIO_PREMEM_CONFIG, 162
AudioLinkSndw
    HDAUDIO_PREMEM_CONFIG, 162
AudioLinkSsp
    HDAUDIO_PREMEM_CONFIG, 163
Avx2VoltageScaleFactor
    OVERCLOCKING_PREMEM_CONFIG, 223

Base.h, 359
    _BASE_INT_SIZE_OF, 364
    _INT_SIZE_OF, 364
    ABS, 365
    ADDRESS_IS_ALIGNED, 365
    ALIGN_POINTER, 366
    ALIGN_VALUE, 366
    ALIGN_VALUE_ADDEND, 367
    ALIGN_VARIABLE, 367
    ALIGNOF, 367
    ANALYZER_NORETURN, 368
    ANALYZER_UNREACHABLE, 368
    ARRAY_SIZE, 368
    BASE_ARG, 369
    BASE_CR, 369
    BASE_LIST, 380
    ENCODE_ERROR, 370
    ENCODE_WARNING, 370

FALSE, 371
IS_ALIGNED, 371
IS_POW2, 371
MAX, 372
MIN, 372
NORETURN, 374
OFFSET_OF, 374
RETURN_ADDRESS, 374
RETURN_BUFFER_TOO_SMALL, 375
RETURN_ERROR, 375
RETURNS_TWICE, 376
SIGNATURE_16, 376
SIGNATURE_32, 376
SIGNATURE_64, 377
STATIC_ASSERT, 380
TRUE, 377
UNREACHABLE, 378
VA_ARG, 378
VA_COPY, 378
VA_END, 379
VA_LIST, 380
VA_START, 379
BASE_ARG
Base.h, 369
BASE_CR
Base.h, 369
BASE_LIST
Base.h, 380
BaseAddress
EFI_HOB_UEFI_CAPSULE, 134
BCLK_CONFIG, 86
CpuBclkPllOn, 87
SocBclkPllOn, 87
BIOS_GUARD_CONFIG, 87
BiosGuard
CPU_SECURITY_PREMEM_CONFIG, 109
BiosGuardConfig.h, 381
EC_CMD_LOCK, 382
EC_PROVISION_EAV, 382
BiosInterface
PCH_LOCK_DOWN_CONFIG, 256
BiosInterfaceLock
RTC_CONFIG, 299
BiosLock
PCH_LOCK_DOWN_CONFIG, 256
BmeHostDeviceEnabled
PCH_ESPI_CONFIG, 240
BootMaxFrequency
CPU_INIT_PREMEM_CONFIG, 96
BtAudioOffload
CNVI_CONFIG, 90
BtInterface
CNVI_CONFIG, 90

C10DynamicThresholdAdjustment
PCH_PM_CONFIG, 268
ChannelToCkdQckMapping
MEMORY_CONFIG_NO_CRC, 210
CkdAddressTable
MEMORY_CONFIG_NO_CRC, 210
CNVI_CONFIG, 89
BtAudioOffload, 90
BtInterface, 90
Mode, 91
CNVI_PIN_MUX, 91
CNVI_PREMEM_CONFIG, 92
CnviConfig.h, 383
ComplianceTestMode
PCIE_COMMON_CONFIG, 281
ComputeDieSscEnable
OVERCLOCKING_PREMEM_CONFIG, 224
ConfigBlock.h, 384
ConfigBlockLib.h, 386
AddConfigBlock, 387
CreateConfigBlockTable, 387
GetConfigBlock, 388
CoreMinRatio
OVERCLOCKING_PREMEM_CONFIG, 224
CoreVfConfigScope
OVERCLOCKING_PREMEM_CONFIG, 224
CppmFaEn
PCH_PM_CONFIG, 268
CPU_INIT_CONFIG_REVISION
CpuInitConfig.h, 389
CPU_INIT_PREMEM_CONFIG, 93
ActiveCoreCount, 95
ActiveLpAtomCoreCount, 95
ActiveSmallCoreCount, 95
BootMaxFrequency, 96
CpuRatio, 96
CrashLogEnable, 96
CrashLogGprs, 96
DfdEnable, 97
FClkFrequency, 97
OcLock, 97
PeciSxReset, 98
PrmrrSize, 98
ReduceXecores, 98
SmmRelocationEnable, 99
TmeBypassCapability, 99
TmeEnable, 99
TsegSize, 100
VmxEnable, 100
CPU_POWER_DELIVERY_CONFIG, 101
DualTauBoost, 104
PowerLimit1, 104
PowerLimit1Time, 104
PowerLimit2Power, 104
PowerLimit3, 105
PowerLimit4, 105
PowerLimit4Boost, 105
SkinTempControl, 105
TccActivationOffset, 106
ThETAlbattEnable, 106
CPU_POWER_MGMT_BASIC_CONFIG_REVISION
CpuPowerMgmtBasicConfig.h, 392
CPU_POWER_MGMT_CUSTOM_CONFIG, 107

CPU_POWER_MGMT_TEST_CONFIG_REVISION
CpuPowerMgmtTestConfig.h, 394
CPU_POWER_MGMT_VR_CONFIG_REVISION
CpuPowerMgmtVrConfig.h, 396
CPU_SECURITY_PREMEM_CONFIG, 108
BiosGuard, 109
GenerateNewTmeKey, 109
TdxEnable, 110
TseEnable, 110
Txt, 110
CPU_TEST_CONFIG, 111
ProcessorTraceMemSize, 112
CpuBandgapRefMode
OVERCLOCKING_PREMEM_CONFIG, 224
CpuBclkPllOn
BCLK_CONFIG, 87
CpuC10GatePinEnable
PCH_PM_CONFIG, 268
CpuD2dRatio
OVERCLOCKING_PREMEM_CONFIG, 225
CpuInitConfig.h, 388
CPU_INIT_CONFIG_REVISION, 389
CpuInitPreMemConfig.h, 389
CpuPowerDeliveryConfig.h, 390
MAX_CUSTOM_CTDP_ENTRIES, 391
CpuPowerMgmtBasicConfig.h, 391
CPU_POWER_MGMT_BASIC_CONFIG_REVISION, 392
CpuPowerMgmtCustomConfig.h, 392
CpuPowerMgmtTestConfig.h, 393
CPU_POWER_MGMT_TEST_CONFIG_REVISION, 394
CUSTOM_POWER_UNIT, 394
PowerUnit125MilliWatts, 395
PowerUnitWatts, 395
CpuPowerMgmtVrConfig.h, 395
CPU_POWER_MGMT_VR_CONFIG_REVISION, 396
MAX_NUM_VRS, 396
CpuRatio
CPU_INIT_PREMEM_CONFIG, 96
CpuSecurityPreMemConfig.h, 396
CpuStart
EFI_MMRAM_DESCRIPTOR, 136
CpuTestConfig.h, 397
CrashLogEnable
CPU_INIT_PREMEM_CONFIG, 96
CrashLogGprs
CPU_INIT_PREMEM_CONFIG, 96
CreateConfigBlockTable
ConfigBlockLib.h, 387
Crid
PCH_GENERAL_CONFIG, 244
CseDataResilience
ME_PEI_CONFIG, 205
CUSTOM_POWER_UNIT
CpuPowerMgmtTestConfig.h, 394
CustomizedSsid
SI_CONFIG, 312
CustomizedSvid
SI_CONFIG, 312

DciClkEnable
PCH_DCI_PREMEM_CONFIG, 235
DciConfig.h, 398
DciDbcMode
PCH_DCI_PREMEM_CONFIG, 235
DciEn
PCH_DCI_PREMEM_CONFIG, 235
DDI_CONFIGURATION, 113
Default
MemoryConfig.h, 433
DfdEnable
CPU_INIT_PREMEM_CONFIG, 97
DisableDsxAcPresentPulldown
PCH_PM_CONFIG, 269
DisableEnergyReport
PCH_PM_CONFIG, 269
DisableNativePowerButton
PCH_PM_CONFIG, 269
DisablePerCoreMask
OVERCLOCKING_PREMEM_CONFIG, 225
DisableResets
PCH_GENERAL_PREMEM_CONFIG, 246
DisplayFusaConfigEnable
FUSA_CONFIG, 148
DMI_EQ_PARAM, 114
DMI_HW_WIDTH_CONTROL, 115
DMI_LANE_CONFIG, 116
DmiConfig.h, 399
DmiGen3EqPh2Enable
PEI_DMI_CONFIG, 291
DmiGen3EqPh3Method
PEI_DMI_CONFIG, 292
DmiGen3ProgramStaticEq
PEI_DMI_CONFIG, 292
DmiGen3RxCtlePeaking
PEI_DMI_CONFIG, 292
DmiHaAWC
THERMAL_CONFIG, 338
DmiMaxLinkSpeed
PEI_DMI_CONFIG, 293
DmiPowerReduction
PCH_DMI_CONFIG, 238
DpOnly
TcssPeiPreMemConfig.h, 482
DualTauBoost
CPU_POWER_DELIVERY_CONFIG, 104
DwbConfigDwbFlushThreshold
PSF_CONFIG, 298
DXE_ERROR
PiMultiPhase.h, 457
DXE_ITBT_CONFIG
DxeITbtConfig.h, 401
DxeITbtConfig.h, 399
DXE_ITBT_CONFIG, 401

EC_CMD_LOCK
BiosGuardConfig.h, 382
EC_PROVISION_EAV
BiosGuardConfig.h, 382
EcoreTvbTempThreshold0
OVERCLOCKING_PREMEM_CONFIG, 225
EcoreTvbTempThreshold1
OVERCLOCKING_PREMEM_CONFIG, 225
EFI_AP_PROCEDURE
PiMultiPhase.h, 459
EFI_AP_PROCEDURE2
PiMultiPhase.h, 460
EFI_AUTH_STATUS_PLATFORM_OVERRIDE
PiMultiPhase.h, 458
EFI_HOB_CPU, 116
Header, 117
EFI_HOB_FIRMWARE_VOLUME, 117
Header, 118
EFI_HOB_FIRMWARE_VOLUME2, 118
Header, 119
EFI_HOB_FIRMWARE_VOLUME3, 120
ExtractedFv, 120
FileName, 121
FvName, 121
Header, 121
EFI_HOB_GENERIC_HEADER, 122
EFI_HOB_GUID_TYPE, 122
Header, 123
EFI_HOB_HANDOFF_INFO_TABLE, 123
EfiMemoryTop, 124
Header, 124
Version, 124
EFI_HOB_MEMORY_ALLOCATION, 125
Header, 126
EFI_HOB_MEMORY_ALLOCATION_BSP_STORE, 126
Header, 127
EFI_HOB_MEMORY_ALLOCATION_HEADER, 127
MemoryBaseAddress, 128
MemoryType, 128
Name, 128
EFI_HOB_MEMORY_ALLOCATION_MODULE, 129
Header, 130
EFI_HOB_MEMORY_ALLOCATION_STACK, 130
Header, 131
EFI_HOB_MEMORY_POOL, 131
Header, 132
EFI_HOB_RESOURCE_DESCRIPTOR, 132
Header, 133
Owner, 133
EFI_HOB_UEFI_CAPSULE, 134
BaseAddress, 134
EFI_IP_ADDRESS, 135
EFI_IPv4_ADDRESS
UefiBaseType.h, 494
EFI_IPv6_ADDRESS
UefiBaseType.h, 494
EFI_MAC_ADDRESS, 136
EFI_MMRAM_DESCRIPTOR, 136
CpuStart, 136
PhysicalStart, 137
RegionState, 137
EFI_PAGES_TO_SIZE
UefiBaseType.h, 493
EFI_PEI_HOB_POINTERS, 138
EFI_PEI_MP_SERVICES_ENABLEDISABLEAP
MpServices.h, 437
EFI_PEI_MP_SERVICES_GET_NUMBER_OF_PROCESSORS
MpServices.h, 438
EFI_PEI_MP_SERVICES_GET_PROCESSOR_INFO
MpServices.h, 438
EFI_PEI_MP_SERVICES_STARTUP_ALL_APS
MpServices.h, 439
EFI_PEI_MP_SERVICES_STARTUP_THIS_AP
MpServices.h, 439
EFI_PEI_MP_SERVICES_SWITCH_BSP
MpServices.h, 440
EFI_PEI_MP_SERVICES_WHOAMI
MpServices.h, 441
EFI_SIZE_TO_PAGES
UefiBaseType.h, 493
EFI_TIME, 138
EfiMemoryTop
EFI_HOB_HANDOFF_INFO_TABLE, 124
Enable
GBE_CONFIG, 150
ISH_PREMEM_CONFIG, 188
PCH_SATA_PORT_CONFIG, 274
PCH_SMBUS_PREMEM_CONFIG, 276
SATA_CONFIG, 302
XDCI_CONFIG, 356
Enable8254ClockGating
PCH_IOAPIC_CONFIG, 254
Enable8254ClockGatingOnS3
PCH_IOAPIC_CONFIG, 254
EnableMode
TRACE_HUB_CONFIG, 342
EnablePort8xhDecode
PCIE_COMMON_CONFIG, 281
ENCODE_ERROR
Base.h, 370
ENCODE_WARNING
Base.h, 370
EnhancePort8xhDecoding
PCH_LPC_PREMEM_CONFIG, 259
EqPh23Bypass
PCIE_LINK_EQ_PLATFORM_SETTINGS, 287
EqPh3Bypass
PCIE_LINK_EQ_PLATFORM_SETTINGS, 287
ErrorType
FSP_ERROR_INFO, 143
EsataSpeedLimit
SATA_CONFIG, 302
EspiConfig.h, 401
ExtendedBiosDecodeRangeBase
SPI_CONFIG, 318
ExtendedBiosDecodeRangeLimit

SPI_CONFIG, 318
ExtractedFv
EFI_HOB_FIRMWARE_VOLUME3, 120
ExtSync
PCH_PCIE_ROOT_PORT_CONFIG, 264
ExtVnnRailSx
PCH_FIVR_CONFIG, 241

FALSE
Base.h, 371
FClkFrequency
CPU_INIT_PREMEM_CONFIG, 97
FiaConfig.h, 402
FileName
EFI_HOB_FIRMWARE_VOLUME3, 121
FIVR_EXT_RAIL_CONFIG, 139
SupportedVoltageStates, 139
Voltage, 140
FIVR_VCCIN_AUX_CONFIG, 140
LowToHighCurModeVolTranTime, 141
OffToHighCurModeVolTranTime, 141
RetToHighCurModeVolTranTime, 141
RetToLowCurModeVolTranTime, 141
FivrConfig.h, 402
FlashProtectionConfig.h, 403
ForceLtrOverride
PCIE_DEVICE_OVERRIDE, 283
FSP_ERROR_INFO, 142
ErrorType, 143
FspErrorInfo.h, 404
FspFixedPcds.h, 405
FSPM_ARCH_CONFIG_PPI, 143
FspmArchConfigPpi.h, 406
FspPerformance.h, 406
FSPV_POSTMEM_CONFIG, 144
FSPV_PREMEM_CONFIG, 145
FspVConfig.h, 407
FullFunction
TcssPeiPreMemConfig.h, 482
FUSA_CONFIG, 146
DisplayFusaConfigEnable, 148
FUSA_CONFIG_REVISION
FusaConfig.h, 408
FusaConfig.h, 408
FUSA_CONFIG_REVISION, 408
FvName
EFI_HOB_FIRMWARE_VOLUME3, 121

GBE_CONFIG, 149
Enable, 150
GbeConfig.h, 409
GenerateNewTmeKey
CPU_SECURITY_PREMEM_CONFIG, 109
GetConfigBlock
ConfigBlockLib.h, 388
GlobalResetMasksOverride
PCH_PM_CONFIG, 269
GlobalSmi
PCH_LOCK_DOWN_CONFIG, 257
GPIO_DXE_CONFIG, 151
HideGpioAcpiDevice, 152
HideHardGpioAcpiDevice, 152
HideVirtGpioAcpiDevice, 152
GpioDevConfig.h, 409
GpioOverride
PCH_GENERAL_PREMEM_CONFIG, 246
GranularityBins
OVERCLOCKING_PREMEM_CONFIG, 226
GUID, 152

HDA_LINK_DMIC, 153
HDA_LINK_HDA, 153
HDA_LINK_SNDW, 154
HDA_LINK_SSP, 154
HDA_MIC_PRIVACY, 155
HDA_SNDW_MULTILANE, 156
HDA_VERB_TABLE_HEADER, 156
HDAUDIO_CONFIG, 157
VerbTableEntryNum, 158
VerbTablePtr, 158
HDAUDIO_DXE_CONFIG, 159
HDAUDIO_PREMEM_CONFIG, 160
AudioLinkDmic, 162
AudioLinkHda, 162
AudioLinkSndw, 162
AudioLinkSsp, 163
SndwMultilane, 163
HdAudioConfig.h, 410
Header
EFI_HOB_CPU, 117
EFI_HOB_FIRMWARE_VOLUME, 118
EFI_HOB_FIRMWARE_VOLUME2, 119
EFI_HOB_FIRMWARE_VOLUME3, 121
EFI_HOB_GUID_TYPE, 123
EFI_HOB_HANDOFF_INFO_TABLE, 124
EFI_HOB_MEMORY_ALLOCATION, 126
EFI_HOB_MEMORY_ALLOCATION_BSP_STORE, 127
EFI_HOB_MEMORY_ALLOCATION_MODULE, 130
EFI_HOB_MEMORY_ALLOCATION_STACK, 131
EFI_HOB_MEMORY_POOL, 132
EFI_HOB_RESOURCE_DESCRIPTOR, 133
PCH_SMBUS_PREMEM_CONFIG, 276
HideGpioAcpiDevice
GPIO_DXE_CONFIG, 152
HideHardGpioAcpiDevice
GPIO_DXE_CONFIG, 152
HideVirtGpioAcpiDevice
GPIO_DXE_CONFIG, 152
HidOverI2c
ThcConfig.h, 486
HidOverSpi
ThcConfig.h, 486
HOST_BRIDGE_PEI_CONFIG, 164
SkipPamLock, 165
HOST_BRIDGE_PREMEM_CONFIG, 165
HostBridgeConfig.h, 412

HSIO_CONFIG, 167
HSIO_PARAMETERS, 168
    HsioCtrlAdaptOffsetCfg, 170
HsioConfig.h, 413
HsioCtrlAdaptOffsetCfg
    HSIO_PARAMETERS, 170
HsioPcieConfig.h, 413
HsioSataConfig.h, 414
HYBRID_GRAPHICS_CONFIG, 170
HybridGraphicsConfig.h, 415

I2C_PIN_MUX, 171
I3C_MODE
    LpssI3cConfig.h, 426
I3C_MUX_GPIO_CONFIG, 172
IaIccMax
    OVERCLOCKING_PREMEM_CONFIG, 226
IAX_PEI_CONFIG, 172
    IaxEnable, 173
IaxConfig.h, 415
IaxEnable
    IAX_PEI_CONFIG, 173
IEH_CONFIG, 174
IehConfig.h, 416
IgdDvmt50PreAlloc
    IGPU_PEI_PREMEM_CONFIG, 179
IGPU_DXE_CONFIG, 175
IGPU_PEI_CONFIG, 176
IGPU_PEI_PREMEM_CONFIG, 177
    IgdDvmt50PreAlloc, 179
    InternalGraphics, 179
    VgaInitControl, 180
IGpuConfig.h, 417
ImguClkOutEn
    IPU_PREMEM_CONFIG, 182
InternalGraphics
    IGPU_PEI_PREMEM_CONFIG, 179
InterruptConfig.h, 418
    PCH_INT_PIN, 419
    PchNoInt, 419
IoApicConfig.h, 420
IoeDebugEn
    PCH_GENERAL_PREMEM_CONFIG, 247
IOM_AUX_ORI_GPIO_PAD_CONFIG, 180
IPU_PREMEM_CONFIG, 181
    ImguClkOutEn, 182
    IpuEnable, 182
IpuEnable
    IPU_PREMEM_CONFIG, 182
IpuPreMemConfig.h, 420
IPv4_ADDRESS, 182
IPv6_ADDRESS, 183
IS_ALIGNED
    Base.h, 371
IS_POW2
    Base.h, 371
ISH_CONFIG, 183
ISH_GP, 184
ISH_GPIO_CONFIG, 185
    PadTermination, 185
    PinMux, 185
ISH_I2C, 186
ISH_I2C_PIN_CONFIG, 187
ISH_PREMEM_CONFIG, 188
    Enable, 188
ISH_SPI, 189
ISH_SPI_PIN_CONFIG, 190
ISH_UART, 191
ISH_UART_PIN_CONFIG, 192
IshConfig.h, 421
ITbtForcePowerOnTimeoutInMs
    _ITBT_GENERIC_CONFIG, 74

KeepEarlyTrace
    PCH_DCI_PREMEM_CONFIG, 236

L1sCommonModeRestoreTime
    PCIE_DEVICE_OVERRIDE, 283
L1sTpowerOnScale
    PCIE_DEVICE_OVERRIDE, 283
L1sTpowerOnValue
    PCIE_DEVICE_OVERRIDE, 283
L1SubstatesCapMask
    PCIE_DEVICE_OVERRIDE, 284
L1SubstatesCapOffset
    PCIE_DEVICE_OVERRIDE, 284
LatchEventsC10Exit
    PCH_PM_CONFIG, 270
LegacyIoLowLatency
    PCH_GENERAL_CONFIG, 244
LgmrEnable
    PCH_ESPI_CONFIG, 240
LocalTxOverrideEn
    PCIE_LINK_EQ_PLATFORM_SETTINGS, 287
LockDownConfig.h, 422
LowToHighCurModeVolTranTime
    FIVR_VCCIN_AUX_CONFIG, 141
LpcConfig.h, 423
LpcPmHAE
    PCH_LPC_PREMEM_CONFIG, 259
LpmS0ixSubStateEnable
    PCH_PM_CONFIG, 270
LPSS_I2C_CONFIG, 193
LPSS_I2C_CONTROLLER_CONFIG, 194
    PadTermination, 194
LPSS_I2C_I2C_MODE
    LpssI2cConfig.h, 425
LPSS_I2C_PIN_MUX, 195
LPSS_I3C_CONFIG, 195
LPSS_I3C_DEVICE_CONFIG, 196
LPSS_SPI_CONFIG, 197
LPSS_SPI_DEVICE_CONFIG, 198
LPSS_SPI_MODE
    LpssSpiConfig.h, 428
LPSS_SPI_PIN_MUX, 199
LPSS_UART_ATTRIBUTES, 199
LPSS_UART_CONFIG, 200
LPSS_UART_DEVICE_CONFIG, 201

LPSS_UART_MODE
LpssUartConfig.h, 429
LPSS_UART_PG
LpssUartConfig.h, 430
LPSS_UART_PIN_MUX, 202
LpssI2cConfig.h, 424
LPSS_I2C_I2C_MODE, 425
LpssI3cConfig.h, 425
I3C_MODE, 426
LpssSpiConfig.h, 427
LPSS_SPI_MODE, 428
LpssUartConfig.h, 428
LPSS_UART_MODE, 429
LPSS_UART_PG, 430
LpssUartPgAuto, 431
LpssUartPgDisabled, 431
LpssUartPgEnabled, 431
LpssUartPgAuto
LpssUartConfig.h, 431
LpssUartPgDisabled
LpssUartConfig.h, 431
LpssUartPgEnabled
LpssUartConfig.h, 431
LtrOverrideEnable
USB_CONFIG, 352

MAX
Base.h, 372
MAX_CUSTOM_CTDP_ENTRIES
CpuPowerDeliveryConfig.h, 391
MAX_NUM_VRS
CpuPowerMgmtVrConfig.h, 396
MaxOcRatio
OVERCLOCKING_PREMEM_CONFIG, 226
ME_DXE_CONFIG, 202
ME_PEI_CONFIG, 204
CseDataResilience, 205
MeUnconfigOnRtcClear, 205
ME_PEI_PREMEM_CONFIG, 206
SkipMbpHob, 207
MeDxeConfig.h, 431
MEMORY_CONFIG_NO_CRC, 208
ChannelToCkdQckMapping, 210
CkdAddressTable, 210
PhyClockToCkdDimm, 210
SerialDebugLevel, 211
SpdAddressTable, 211
MEMORY_CONFIGURATION, 212
MEMORY_DQDQS_MAPPING, 213
MEMORY_RCOMP, 213
MEMORY_SUB_SYSTEM_CONFIG, 214
MemoryBaseAddress
EFI_HOB_MEMORY_ALLOCATION_HEADER, 128
MemoryConfig.h, 432
Default, 433
SA_SPD, 433
SPD_BOOT_MODE, 433
SpdBootModeMax, 434
SpdCold, 434
SpdFast, 434
SpdS3, 434
SpdWarm, 434
UserDefined, 433
XMPProfile1, 433
XMPProfile2, 433
XMPProfileMax, 433
MemoryLock
RTC_CONFIG, 300
MemorySubSystemConfig.h, 434
MemoryType
EFI_HOB_MEMORY_ALLOCATION_HEADER, 128
MemReg0Size
TRACE_HUB_CONFIG, 342
MePeiConfig.h, 435
MeUnconfigOnRtcClear
ME_PEI_CONFIG, 205
MIN
Base.h, 372
MmramReservedSize
_EFI_MM_RESERVED_MMRAM_REGION, 72
MmramReservedStart
_EFI_MM_RESERVED_MMRAM_REGION, 72
Mode
CNVI_CONFIG, 91
ModPhySusPgEnable
PCH_PM_CONFIG, 270
MpServices.h, 436
EFI_PEI_MP_SERVICES_ENABLEDISABLEAP, 437
EFI_PEI_MP_SERVICES_GET_NUMBER_OF_PROCESSORS, 438
EFI_PEI_MP_SERVICES_GET_PROCESSOR_INFO, 438
EFI_PEI_MP_SERVICES_STARTUP_ALL_APS, 439
EFI_PEI_MP_SERVICES_STARTUP_THIS_AP, 439
EFI_PEI_MP_SERVICES_SWITCH_BSP, 440
EFI_PEI_MP_SERVICES_WHOAMI, 441
MRC_ERROR_KEY_VALUE_ENTRY, 215
MUX_GPIO_CONFIG, 216
MUX_GPIO_PARAM, 216
PadTermination, 217
PinMux, 217
MvcEnabled
PCH_PCIE_ROOT_PORT_CONFIG, 264

Name
EFI_HOB_MEMORY_ALLOCATION_HEADER, 128
NegativeItd
OVERCLOCKING_PREMEM_CONFIG, 226
NonSnoopLatency
PCIE_DEVICE_OVERRIDE, 284
NoPcie
TcssPeiPreMemConfig.h, 482

NORETURN
Base.h, 374
NoThunderbolt
TcssPeiPreMemConfig.h, 482
NPU_PEI_CONFIG, 218
NpuEnable, 219
NpuConfig.h, 441
NpuEnable
NPU_PEI_CONFIG, 219
NumberOfSsidTableEntry
SI_CONFIG, 312

OcLock
CPU_INIT_PREMEM_CONFIG, 97
OcSupport
OVERCLOCKING_PREMEM_CONFIG, 227
OcTvb
OVERCLOCKING_PREMEM_CONFIG, 227
OFFSET_OF
Base.h, 374
OffToHighCurModeVolTranTime
FIVR_VCCIN_AUX_CONFIG, 141
OsIdleEnable
PCH_PM_CONFIG, 270
OVERCLOCKING_PREMEM_CONFIG, 219
AdaptiveVoltage, 223
AtomClusterRatio, 223
Avx2VoltageScaleFactor, 223
ComputeDieSscEnable, 224
CoreMinRatio, 224
CoreVfConfigScope, 224
CpuBandgapRefMode, 224
CpuD2dRatio, 225
DisablePerCoreMask, 225
EcoreTvbTempThreshold0, 225
EcoreTvbTempThreshold1, 225
GranularityBins, 226
IaIccMax, 226
MaxOcRatio, 226
NegativeItd, 226
OcSupport, 227
OcTvb, 227
PcorePowerDensityThrottle, 227
PcoreTvbTempThreshold0, 227
PcoreTvbTempThreshold1, 228
PerCoreVoltageOffset, 228
PerEcoreCcpRatioDownBinAboveT0, 228
PerEcoreCcpRatioDownBinAboveT1, 228
PerPcoreGrRatioDownBinAboveT0, 229
PerPcoreGrRatioDownBinAboveT1, 229
PerPcoreRatioDownBinAboveT0, 229
PerPcoreRatioDownBinAboveT1, 229
ProcessVmaxLimit, 230
PvdMode, 230
PvdRatioThreshold, 230
RingDownBin, 230
SocDieSscEnable, 231
ThermalTurbo, 231
TjMaxOffset, 231
TvbConfigLimitSelect, 231
UnderVoltProtection, 232
VcciaBootVoltageSel, 232
VccsaBootVoltageSel, 232
VfPointOffset, 232
VfPointOffsetMode, 233
VoltageMode, 233
VoltageOffset, 233
VoltageOverride, 233
OverclockingConfig.h, 443
OverCurrentEnable
USB_CONFIG, 353
OverCurrentPin
USB2_PORT_CONFIG, 349
USB3_PORT_CONFIG, 350
Owner
EFI_HOB_RESOURCE_DESCRIPTOR, 133

P2sbConfig.h, 443
PadTermination
ISH_GPIO_CONFIG, 185
LPSS_I2C_CONTROLLER_CONFIG, 194
MUX_GPIO_PARAM, 217
SERIAL_IO_I2C_CONFIG, 307
PCH_DCI_PREMEM_CONFIG, 234
DciClkEnable, 235
DciDbcMode, 235
DciEn, 235
KeepEarlyTrace, 236
PCH_DEVICE_INTERRUPT_CONFIG, 236
PCH_DMI_CONFIG, 237
DmiPowerReduction, 238
PCH_ESPI_CONFIG, 239
BmeHostDeviceEnabled, 240
LgmrEnable, 240
PCH_FIVR_CONFIG, 241
ExtVnnRailSx, 241
PCH_FLASH_PROTECTION_CONFIG, 242
PCH_GENERAL_CONFIG, 243
Crid, 244
LegacyIoLowLatency, 244
VtdEnabled, 244
PCH_GENERAL_PREMEM_CONFIG, 245
DisableResets, 246
GpioOverride, 246
IoeDebugEn, 247
PmcPrivacyConsent, 247
PmodeClkEn, 247
PCH_HSIO_PCIE_LANE_CONFIG, 248
PCH_HSIO_PCIE_PREMEM_CONFIG, 249
PCH_HSIO_SATA_PORT_LANE, 249
PCH_HSIO_SATA_PREMEM_CONFIG, 251
PCH_INT_PIN
InterruptConfig.h, 419
PCH_INTERRUPT_CONFIG, 252
PCH_IOAPIC_CONFIG, 253
Enable8254ClockGating, 254
Enable8254ClockGatingOnS3, 254
PCH_LOCK_DOWN_CONFIG, 255

BiosInterface, 256
BiosLock, 256
GlobalSmi, 257
UnlockGpioPads, 257
PCH_LPC_PREMEM_CONFIG, 258
EnhancePort8xhDecoding, 259
LpcPmHAE, 259
PCH_P2SB_CONFIG, 260
SbAccessUnlock, 260
PCH_PCIE_CLOCK, 261
PCH_PCIE_CLOCK_USAGE
PchPcieRpConfig.h, 447
PCH_PCIE_CONFIG, 262
PcieClockGating, 263
PcieTbtPerfBoost, 263
PCH_PCIE_ROOT_PORT_CONFIG, 263
ExtSync, 264
MvcEnabled, 264
VppPort, 264
PCH_PCIE_RP_PREMEM_CONFIG, 265
RpEnabledMask, 266
PCH_PM_CONFIG, 266
C10DynamicThresholdAdjustment, 268
CppmFaEn, 268
CpuC10GatePinEnable, 268
DisableDsxAcPresentPulldown, 269
DisableEnergyReport, 269
DisableNativePowerButton, 269
GlobalResetMasksOverride, 269
LatchEventsC10Exit, 270
LpmS0ixSubStateEnable, 270
ModPhySusPgEnable, 270
OsIdleEnable, 270
PchPwrCycDur, 271
PciePllSsc, 271
PmcWdtTimerEn, 271
PsOnEnable, 271
PwrBtnOverridePeriod, 272
S0ixAutoDemotion, 272
SlpLanLowDc, 272
SlpStrchSusUp, 272
ThermTimerDelay, 273
Usb2PhySusPgEnable, 273
PCH_RESERVED_PAGE_ROUTE
PchGeneralConfig.h, 445
PCH_SATA_PORT_CONFIG, 273
Enable, 274
ZpOdd, 274
PCH_SLP_S4_MIN_ASSERT
PmConfig.h, 461
PCH_SMBUS_PREMEM_CONFIG, 275
Enable, 276
Header, 276
SpdWriteDisable, 277
PCH_TRACE_HUB_PREMEM_CONFIG, 277
PCH_WAKE_CONFIG, 278
PmeB0S5Dis, 279
PCH_WDT_PREMEM_CONFIG, 279
PchClockUsageCpuPcie0
PchPcieRpConfig.h, 448
PchClockUsageUnspecified
PchPcieRpConfig.h, 448
PchDmiConfig.h, 444
PchGeneralConfig.h, 445
PCH_RESERVED_PAGE_ROUTE, 445
PchReservedPageToLpc, 445
PchReservedPageToPcie, 445
PchHotLevel
THERMAL_CONFIG, 339
PchNoInt
InterruptConfig.h, 419
PchPcieRpConfig.h, 446
PCH_PCIE_CLOCK_USAGE, 447
PchClockUsageCpuPcie0, 448
PchClockUsageUnspecified, 448
PchPwrCycDur
PCH_PM_CONFIG, 271
PchReservedPageToLpc
PchGeneralConfig.h, 445
PchReservedPageToPcie
PchGeneralConfig.h, 445
PchSlpS4PchTime
PmConfig.h, 461
PCIE_COMMON_CONFIG, 280
ComplianceTestMode, 281
EnablePort8xhDecode, 281
RpFunctionSwap, 281
PCIE_DEVICE_OVERRIDE, 282
ForceLtrOverride, 283
L1sCommonModeRestoreTime, 283
L1sTpowerOnScale, 283
L1sTpowerOnValue, 283
L1SubstatesCapMask, 284
L1SubstatesCapOffset, 284
NonSnoopLatency, 284
SnoopLatency, 284
PCIE_EQ_PARAM, 285
PCIE_FORM_FACTOR
PcieConfig.h, 449
PCIE_LINK_EQ_METHOD
PcieConfig.h, 449
PCIE_LINK_EQ_MODE
PcieConfig.h, 449
PCIE_LINK_EQ_PLATFORM_SETTINGS, 285
EqPh23Bypass, 287
EqPh3Bypass, 287
LocalTxOverrideEn, 287
Ph2LocalTxOverridePreset, 287
Ph3NoOfPresetOrCoeff, 288
PCIE_RP_DXE_CONFIG, 288
PcieDeviceOverrideTablePtr, 289
PcieClockGating
PCH_PCIE_CONFIG, 263
PcieConfig.h, 448
PCIE_FORM_FACTOR, 449
PCIE_LINK_EQ_METHOD, 449

PCIE_LINK_EQ_MODE, 449
PcieLinkEqCoefficientMode, 450
PcieLinkEqPresetMode, 450
PcieLinkFixedEq, 449
PcieLinkHardwareEq, 449
PcieDeviceOverrideTablePtr
PCIE_RP_DXE_CONFIG, 289
PcieLinkEqCoefficientMode
PcieConfig.h, 450
PcieLinkEqPresetMode
PcieConfig.h, 450
PcieLinkFixedEq
PcieConfig.h, 449
PcieLinkHardwareEq
PcieConfig.h, 449
PciePllSsc
PCH_PM_CONFIG, 271
PcieTbtPerfBoost
PCH_PCIE_CONFIG, 263
PcorePowerDensityThrottle
OVERCLOCKING_PREMEM_CONFIG, 227
PcoreTvbTempThreshold0
OVERCLOCKING_PREMEM_CONFIG, 227
PcoreTvbTempThreshold1
OVERCLOCKING_PREMEM_CONFIG, 228
PdoProgramming
USB_CONFIG, 353
PeciSxReset
CPU_INIT_PREMEM_CONFIG, 98
PEI_DMI_CONFIG, 290
DmiGen3EqPh2Enable, 291
DmiGen3EqPh3Method, 292
DmiGen3ProgramStaticEq, 292
DmiGen3RxCtlePeaking, 292
DmiMaxLinkSpeed, 293
PEI_ITBT_CONFIG
PeiITbtConfig.h, 452
PeiDmiConfig.h, 450
PeiITbtConfig.h, 451
PEI_ITBT_CONFIG, 452
PeiITbtGenericStructure.h, 452
PeiPreMemSiDefaultPolicy.h, 453
PeiSiDefaultPolicy.h, 454
PerCoreVoltageOffset
OVERCLOCKING_PREMEM_CONFIG, 228
PerEcoreCcpRatioDownBinAboveT0
OVERCLOCKING_PREMEM_CONFIG, 228
PerEcoreCcpRatioDownBinAboveT1
OVERCLOCKING_PREMEM_CONFIG, 228
PerPcoreGrRatioDownBinAboveT0
OVERCLOCKING_PREMEM_CONFIG, 229
PerPcoreGrRatioDownBinAboveT1
OVERCLOCKING_PREMEM_CONFIG, 229
PerPcoreRatioDownBinAboveT0
OVERCLOCKING_PREMEM_CONFIG, 229
PerPcoreRatioDownBinAboveT1
OVERCLOCKING_PREMEM_CONFIG, 229
Ph2LocalTxOverridePreset
PCIE_LINK_EQ_PLATFORM_SETTINGS, 287
Ph3NoOfPresetOrCoeff
PCIE_LINK_EQ_PLATFORM_SETTINGS, 288
PhyClockToCkdDimm
MEMORY_CONFIG_NO_CRC, 210
PhysicalStart
EFI_MMRAM_DESCRIPTOR, 137
PI_ENCODE_ERROR
PiMultiPhase.h, 458
PI_ENCODE_WARNING
PiMultiPhase.h, 458
PiHob.h, 455
PiMultiPhase.h, 456
DXE_ERROR, 457
EFI_AP_PROCEDURE, 459
EFI_AP_PROCEDURE2, 460
EFI_AUTH_STATUS_PLATFORM_OVERRIDE, 458
PI_ENCODE_ERROR, 458
PI_ENCODE_WARNING, 458
PinMux
ISH_GPIO_CONFIG, 185
MUX_GPIO_PARAM, 217
PlatformDebugOption
SI_PREMEM_CONFIG, 315
PMC_GLOBAL_RESET_MASK, 293
PMC_INTERFACE_CONFIG, 294
PMC_LPM_S0IX_SUB_STATE_EN, 294
S0i2p2En, 294
S0i3p3En, 295
S0i3p4En, 295
PmConfig.h, 460
PCH_SLP_S4_MIN_ASSERT, 461
PchSlpS4PchTime, 461
PmcPrivacyConsent
PCH_GENERAL_PREMEM_CONFIG, 247
PmcWdtTimerEn
PCH_PM_CONFIG, 271
PmeB0S5Dis
PCH_WAKE_CONFIG, 279
PmodeClkEn
PCH_GENERAL_PREMEM_CONFIG, 247
Port
USB2_PHY_CONFIG, 347
PowerLimit1
CPU_POWER_DELIVERY_CONFIG, 104
PowerLimit1Time
CPU_POWER_DELIVERY_CONFIG, 104
PowerLimit2Power
CPU_POWER_DELIVERY_CONFIG, 104
PowerLimit3
CPU_POWER_DELIVERY_CONFIG, 105
PowerLimit4
CPU_POWER_DELIVERY_CONFIG, 105
PowerLimit4Boost
CPU_POWER_DELIVERY_CONFIG, 105
PowerUnit125MilliWatts
CpuPowerMgmtTestConfig.h, 395

PowerUnitWatts
CpuPowerMgmtTestConfig.h, 395
PPM_CUSTOM_RATIO_TABLE, 295
StateRatioMax16, 296
PrmrrSize
CPU_INIT_PREMEM_CONFIG, 98
ProcessorTraceMemSize
CPU_TEST_CONFIG, 112
ProcessVmaxLimit
OVERCLOCKING_PREMEM_CONFIG, 230
PROTECTED_RANGE, 296
PSF_CONFIG, 297
DwbConfigDwbFlushThreshold, 298
PsfConfig.h, 462
PsOnEnable
PCH_PM_CONFIG, 271
PvdMode
OVERCLOCKING_PREMEM_CONFIG, 230
PvdRatioThreshold
OVERCLOCKING_PREMEM_CONFIG, 230
PwrBtnOverridePeriod
PCH_PM_CONFIG, 272

RaidDeviceId
SATA_CONFIG, 303
ReduceXecores
CPU_INIT_PREMEM_CONFIG, 98
RegionState
EFI_MMRAM_DESCRIPTOR, 137
RetToHighCurModeVolTranTime
FIVR_VCCIN_AUX_CONFIG, 141
RetToLowCurModeVolTranTime
FIVR_VCCIN_AUX_CONFIG, 141
RETURN_ADDRESS
Base.h, 374
RETURN_BUFFER_TOO_SMALL
Base.h, 375
RETURN_ERROR
Base.h, 375
RETURNS_TWICE
Base.h, 376
RingDownBin
OVERCLOCKING_PREMEM_CONFIG, 230
RpEnabledMask
PCH_PCIE_RP_PREMEM_CONFIG, 266
RpFunctionSwap
PCIE_COMMON_CONFIG, 281
RTC_CONFIG, 299
BiosInterfaceLock, 299
MemoryLock, 300
RtcConfig.h, 463

S0i2p2En
PMC_LPM_S0IX_SUB_STATE_EN, 294
S0i3p3En
PMC_LPM_S0IX_SUB_STATE_EN, 295
S0i3p4En
PMC_LPM_S0IX_SUB_STATE_EN, 295
S0ixAutoDemotion
PCH_PM_CONFIG, 272
SA_ADDRESS_DECODE, 300
SA_SPD
MemoryConfig.h, 433
SATA_CONFIG, 301
Enable, 302
EsataSpeedLimit, 302
RaidDeviceId, 303
SataMode, 303
SpeedLimit, 303
ThermalThrottling, 303
SATA_THERMAL_THROTTLING, 304
SataConfig.h, 464
SataMode
SATA_CONFIG, 303
SbAccessUnlock
PCH_P2SB_CONFIG, 260
SCS_SD_DLL, 305
ScsConfig.h, 465
SERIAL_IO_CONFIG, 305
SERIAL_IO_I2C_CONFIG, 306
PadTermination, 307
SERIAL_IO_I2C_MODE
SerialIoDevices.h, 468
SERIAL_IO_I3C_CONFIG, 307
SERIAL_IO_I3C_MODE
SerialIoDevices.h, 469
SERIAL_IO_SPI_CONFIG, 308
SERIAL_IO_SPI_MODE
SerialIoDevices.h, 469
SERIAL_IO_UART_ATTRIBUTES, 309
SERIAL_IO_UART_CONFIG, 310
SERIAL_IO_UART_MODE
SerialIoDevices.h, 470
SERIAL_IO_UART_PG
SerialIoDevices.h, 471
SerialDebugLevel
MEMORY_CONFIG_NO_CRC, 211
SerialIoConfig.h, 466
SerialIoDevices.h, 467
SERIAL_IO_I2C_MODE, 468
SERIAL_IO_I3C_MODE, 469
SERIAL_IO_SPI_MODE, 469
SERIAL_IO_UART_MODE, 470
SERIAL_IO_UART_PG, 471
SerialIoUartPgAuto, 471
SerialIoUartPgDisabled, 471
SerialIoUartPgEnabled, 471
SerialIoUartPgAuto
SerialIoDevices.h, 471
SerialIoUartPgDisabled
SerialIoDevices.h, 471
SerialIoUartPgEnabled
SerialIoDevices.h, 471
SI_CONFIG, 311
CustomizedSsid, 312
CustomizedSvid, 312
NumberOfSsidTableEntry, 312

SkipBiosDoneWhenFwUpdate, 313
SkipSsidProgramming, 313
SsidTablePtr, 313
SI_POLICY_REVISION
SiPolicyStruct.h, 474
SI_PREMEM_CONFIG, 314
PlatformDebugOption, 315
SkipOverrideBootModeWhenFwUpdate, 315
SI_PREMEM_POLICY_REVISION
SiPolicyStruct.h, 474
SiConfig.h, 471
SIGNATURE_16
Base.h, 376
SIGNATURE_32
Base.h, 376
SIGNATURE_64
Base.h, 377
SiPolicyStruct.h, 473
SI_POLICY_REVISION, 474
SI_PREMEM_POLICY_REVISION, 474
SiPreMemConfig.h, 475
SkinTempControl
CPU_POWER_DELIVERY_CONFIG, 105
SkipBiosDoneWhenFwUpdate
SI_CONFIG, 313
SkipMbpHob
ME_PEI_PREMEM_CONFIG, 207
SkipOverrideBootModeWhenFwUpdate
SI_PREMEM_CONFIG, 315
SkipPamLock
HOST_BRIDGE_PEI_CONFIG, 165
SkipSsidProgramming
SI_CONFIG, 313
SlpLanLowDc
PCH_PM_CONFIG, 272
SlpStrchSusUp
PCH_PM_CONFIG, 272
SmbusConfig.h, 476
SmmRelocationEnable
CPU_INIT_PREMEM_CONFIG, 99
SndwMultilane
HDAUDIO_PREMEM_CONFIG, 163
SnoopLatency
PCIE_DEVICE_OVERRIDE, 284
SocBclkPllOn
BCLK_CONFIG, 87
SocDieSscEnable
OVERCLOCKING_PREMEM_CONFIG, 231
SPD_BOOT_MODE
MemoryConfig.h, 433
SPD_DATA_BUFFER, 315
SPD_OFFSET_TABLE, 316
SpdAddressTable
MEMORY_CONFIG_NO_CRC, 211
SpdBootModeMax
MemoryConfig.h, 434
SpdCold
MemoryConfig.h, 434
SpdFast
MemoryConfig.h, 434
SpdS3
MemoryConfig.h, 434
SpdWarm
MemoryConfig.h, 434
SpdWriteDisable
PCH_SMBUS_PREMEM_CONFIG, 277
SpeedLimit
SATA_CONFIG, 303
SPI_CONFIG, 317
ExtendedBiosDecodeRangeBase, 318
ExtendedBiosDecodeRangeLimit, 318
SPI_PIN_MUX, 318
SpiConfig.h, 477
SsidTablePtr
SI_CONFIG, 313
StateRatioMax16
PPM_CUSTOM_RATIO_TABLE, 296
STATIC_ASSERT
Base.h, 380
SupportedVoltageStates
FIVR_EXT_RAIL_CONFIG, 139
SVID_SID_VALUE, 319
SWIZZLE_CONFIG, 319
SwizzleConfig.h, 478

TccActivationOffset
CPU_POWER_DELIVERY_CONFIG, 106
TCSS_IOM_PEI_CONFIG, 321
TCSS_IOM_PEI_PERMEM_CONFIG, 322
TCSS_MISC_PEI_CONFIG, 322
TCSS_PCIE_PEI_POLICY, 323
TcssTbtPerfBoost, 324
TCSS_PCIE_PORT_POLICY, 324
TCSS_PEI_CONFIG, 326
TCSS_PEI_PREMEM_CONFIG, 327
TCSS_PORT_CAP
TcssPeiPreMemConfig.h, 482
TCSS_USBTC_PEI_PERMEM_CONFIG, 328
TcssConfigDump
TcssPeiConfig.h, 480
TcssPeiConfig.h, 479
TcssConfigDump, 480
TcssPeiPreMemConfig.h, 480
DpOnly, 482
FullFunction, 482
NoPcie, 482
NoThunderbolt, 482
TCSS_PORT_CAP, 482
TcssPreMemConfigDump, 482
UsbCDisable, 482
TcssPreMemConfigDump
TcssPeiPreMemConfig.h, 482
TcssTbtPerfBoost
TCSS_PCIE_PEI_POLICY, 324
TdxEnable
CPU_SECURITY_PREMEM_CONFIG, 110
TELEMETRY_PEI_CONFIG, 328

TELEMETRY_PEI_PREMEM_CONFIG, 329
TelemetryPeiConfig.h, 484
Thc
    ThcConfig.h, 486
THC_CONFIG, 330
THC_DATA_SSID_CONFIG_HOB, 332
THC_HID_OVER_I2C, 333
    AddressingMode, 334
THC_HID_OVER_SPI, 334
THC_MODE
    ThcConfig.h, 486
THC_PORT, 335
THC_PORT_ASSIGNMENT
    ThcConfig.h, 486
THC_PORT_SUBSYTEM_ID, 336
THC_RESET, 337
THC_WOT_PIN_CONFIG, 337
ThcAssignmentNone
    ThcConfig.h, 486
ThcAssignmentThc0
    ThcConfig.h, 486
ThcAssignmentThc1
    ThcConfig.h, 486
ThcConfig.h, 485
    HidOverI2c, 486
    HidOverSpi, 486
    Thc, 486
    THC_MODE, 486
    THC_PORT_ASSIGNMENT, 486
    ThcAssignmentNone, 486
    ThcAssignmentThc0, 486
    ThcAssignmentThc1, 486
THERMAL_CONFIG, 338
    DmiHaAWC, 338
    PchHotLevel, 339
    TTLevels, 339
THERMAL_THROTTLE_LEVELS, 339
    TTLock, 340
    TTState13Enable, 340
ThermalConfig.h, 487
ThermalThrottling
    SATA_CONFIG, 303
ThermalTurbo
    OVERCLOCKING_PREMEM_CONFIG, 231
ThermTimerDelay
    PCH_PM_CONFIG, 273
ThETAIbattEnable
    CPU_POWER_DELIVERY_CONFIG, 106
TjMaxOffset
    OVERCLOCKING_PREMEM_CONFIG, 231
TmeBypassCapability
    CPU_INIT_PREMEM_CONFIG, 99
TmeEnable
    CPU_INIT_PREMEM_CONFIG, 99
TRACE_HUB_CONFIG, 341
    AetEnabled, 341
    EnableMode, 342
    MemReg0Size, 342
TRACE_HUB_PREMEM_CONFIG, 342
TraceHubConfig.h, 487
TRUE
    Base.h, 377
TseEnable
    CPU_SECURITY_PREMEM_CONFIG, 110
TsegSize
    CPU_INIT_PREMEM_CONFIG, 100
TSN_CONFIG, 343
TSN_CONFIG_REVISION
    TsnConfig.h, 489
TsnConfig.h, 488
    TSN_CONFIG_REVISION, 489
TTLevels
    THERMAL_CONFIG, 339
TTLock
    THERMAL_THROTTLE_LEVELS, 340
TTState13Enable
    THERMAL_THROTTLE_LEVELS, 340
TvbConfigLimitSelect
    OVERCLOCKING_PREMEM_CONFIG, 231
Txt
    CPU_SECURITY_PREMEM_CONFIG, 110
TXT_PREMEM_CONFIG, 344
TxtConfig.h, 490

UaolEnable
    USB_CONFIG, 353
UART_PIN_MUX, 346
UefiBaseType.h, 491
    EFI_IPv4_ADDRESS, 494
    EFI_IPv6_ADDRESS, 494
    EFI_PAGES_TO_SIZE, 493
    EFI_SIZE_TO_PAGES, 493
UnderVoltProtection
    OVERCLOCKING_PREMEM_CONFIG, 232
UnlockGpioPads
    PCH_LOCK_DOWN_CONFIG, 257
UNREACHABLE
    Base.h, 378
USB2_PHY_CONFIG, 346
    Port, 347
USB2_PHY_PARAMETERS, 347
USB2_PORT_CONFIG, 348
    OverCurrentPin, 349
Usb2PhyConfig.h, 494
Usb2PhySusPgEnable
    PCH_PM_CONFIG, 273
Usb31Speed
    USB_CONFIG, 353
USB3_HSIO_CONFIG, 349
USB3_PORT_CONFIG, 350
    OverCurrentPin, 350
Usb3HsioConfig.h, 495
USB_CONFIG, 351
    LtrOverrideEnable, 352
    OverCurrentEnable, 353
    PdoProgramming, 353
    UaolEnable, 353

Usb31Speed, 353
XhciOcLock, 354
UsbCDisable
TcssPeiPreMemConfig.h, 482
UsbConfig.h, 496
UserDefined
MemoryConfig.h, 433

VA_ARG
Base.h, 378
VA_COPY
Base.h, 378
VA_END
Base.h, 379
VA_LIST
Base.h, 380
VA_START
Base.h, 379
VcciaBootVoltageSel
OVERCLOCKING_PREMEM_CONFIG, 232
VccsaBootVoltageSel
OVERCLOCKING_PREMEM_CONFIG, 232
VerbTableEntryNum
HDAUDIO_CONFIG, 158
VerbTablePtr
HDAUDIO_CONFIG, 158
Version
EFI_HOB_HANDOFF_INFO_TABLE, 124
VfPointOffset
OVERCLOCKING_PREMEM_CONFIG, 232
VfPointOffsetMode
OVERCLOCKING_PREMEM_CONFIG, 233
VgaInitControl
IGPU_PEI_PREMEM_CONFIG, 180
VMD_PEI_CONFIG, 354
VmdPeiConfig.h, 497
VmxEnable
CPU_INIT_PREMEM_CONFIG, 100
Voltage
FIVR_EXT_RAIL_CONFIG, 140
VoltageMode
OVERCLOCKING_PREMEM_CONFIG, 233
VoltageOffset
OVERCLOCKING_PREMEM_CONFIG, 233
VoltageOverride
OVERCLOCKING_PREMEM_CONFIG, 233
VppPort
PCH_PCIE_ROOT_PORT_CONFIG, 264
VtdConfig.h, 498
VtdEnabled
PCH_GENERAL_CONFIG, 244

WatchDogConfig.h, 499
WatchDogEnabled
AMT_PEI_CONFIG, 84
WatchDogTimerBios
AMT_PEI_CONFIG, 85
WatchDogTimerOs
AMT_PEI_CONFIG, 85

XDCI_CONFIG, 356
Enable, 356
XhciOcLock
USB_CONFIG, 354
XMPProfile1
MemoryConfig.h, 433
XMPProfile2
MemoryConfig.h, 433
XMPProfileMax
MemoryConfig.h, 433

ZpOdd
PCH_SATA_PORT_CONFIG, 274