Table of Contents

1 Redpine MCU SAPI Introduction 15
1.1 Overview 15
1.2 Features 15
1.3 Architecture 15
1.4 Quick start 16
1.5 Redpine MCU Driver Library 16

2 Quickstart with Redpine MCU SAPIs 18
2.1 Redpine MCU SAPIs Examples in Keil IDE 18
2.1.1 Starting a multi-project view 18
2.1.2 Select project 18
2.2 Redpine MCU SAPIs Examples in IAR IDE 19
2.2.1 Starting a multi-project view 19

3 Building a project 21
3.1 Building a new project in Keil iDE 21
3.1.1 Start new project 21
3.1.2 Add source files 21
3.1.3 Include paths 22
3.1.4 Changing the default optimization level 23
3.2 Building a new project in Keil iDE with Keil SVD Pack 23
3.2.1 Start new project 23
3.2.2 Select Device 23
3.2.3 Selecting and building the library file for the examples 24
3.3 Building a Project in IAR Embedded workbench 24
3.3.1 Start new project 24
3.3.2 Select device 24
3.3.3 Add source files 25
3.3.4 Include paths 25
3.4 Custom Board 26

4 Toolchain and Board Porting Guide 27
4.1 MCU Tool-chain Setup 27
4.1.1 Prerequisites 27
4.1.2 Get Tool-chain Package 27
4.1.3 Create Example Project : 27
4.1.4 Project Compilation 30
4.1.5 6.Steps to Flash and Debug 37

5 Keil Middleware 40

6 Examples 41

7 Brown Out Detection 46
7.1 Overview 46
7.2 Programming Sequence 46
7.3 API Descriptions 47
7.4 RSI_BOD_SoftTriggerGetBatteryStatus 47
7.5 RSI_BOD_Enable 47
7.6 RSI_BOD_SetMode 48
7.7 RSI_BOD_GetThreshold 48
7.8 RSI_BOD_ConfigSlotValue 49
7.9 RSI_BOD_ButtonWakeUpEnable 49
7.10 RSI_BOD_Buttonvalue 50
7.11 RSI_BOD_BlackOutReset 50
7.12 RSI_BOD_BGSampleEnable 51
7.13 RSI_BOD_BGSampleDisable 51
7.14 RSI_BOD_IntrEnable 52
7.15 RSI_BOD_ButtonIntrEnable 52
7.16 RSI_BOD_IntrClr 52
7.17 RSI_BOD_ButtonIntrClr 53
7.18 RSI_BOD_GetIntrStatus 53

8 CMSIS Support 55
8.1 Overview 55

8.2 CMSIS Core 55
8.3 CMSIS Driver 55
8.3.1 GSPI 55
8.3.2 CAN 55
8.3.3 Ethernet 55
8.3.4 I2C 55
8.3.5 MCI 55
8.3.6 SAI 55
8.3.7 SPI 56
8.3.8 USART 56
8.3.9 USB 56
8.4 CMSIS DSP 56
8.4.1 FIM 56

9 IR Decoder 58
9.1 Overview 58
9.2 Programming Sequence 58
9.3 API Descriptions 60
9.3.1 RSI_IR_OffDuration 60
9.3.2 RSI_IR_OnDuration 60
9.3.3 RSI_IR_Framedonethreshold 61
9.3.4 RSI_IR_Detectionthreshold 61
9.3.5 RSI_IR_SetConfiguration 62
9.3.6 RSI_IR_GetMemoryDepth 62
9.3.7 RSI_IR_MemoryReadEnable 63
9.3.8 RSI_IR_ReadData 63
9.3.9 RSI_IR_SoftwareRestart 64
9.3.10 RSI_IR_ClrConfiguration 64
9.3.11 RSI_IR_MemoryReadEnable 65

10 Temperature Sensor 66
10.1 RO Temperature Sensor 66
10.2 Overview 66
10.3 Programming Sequence 66
10.4 API Descriptions 73

11 ULP Subsystem (ULPSS) Clock 80
11.1 Overview 80
11.2 Programming Sequence 80
11.3 API Descriptions 82
11.3.1 RSI_CLK_XtalClkConfig 82
11.3.2 RSI_ULPSS_UlpProcClkConfig 82
11.3.3 RSI_ULPSS_ClockConfig 83
11.3.4 RSI_ULPSS_PeripheralEnable 84
11.3.5 RSI_ULPSS_AuxClkConfig 85
11.3.6 RSI_ULPSS_TimerClkConfig 86
11.3.7 RSI_ULPSS_TimerClkDisable 87
11.3.8 RSI_ULPSS_VadClkConfig 87
11.3.9 RSI_ULPSS_TouchClkConfig 88
11.3.10 RSI_ULPSS_UlpSsiClkConfig 89
11.3.11 RSI_ULPSS_UlpI2sClkConfig 90
11.3.12 RSI_ULPSS_UlpUartClkConfig 91
11.3.13 RSI_ULPSS_SlpSensorClkConfig 92
11.3.14 RSI_ULPSS_RefClkConfig 92
11.3.15 RSI_ULPSS_UlpPeriClkEnable 93
11.3.16 RSI_ULPSS_UlpPeriClkDisable 94
11.3.17 RSI_ULPSS_UlpDynClkEnable 95
11.3.18 RSI_ULPSS_UlpDynClkDisable 96
11.3.19 RSI_ULPSS_PeripheralDisable 98

12 Windowed Watchdog Timer (WWDT) 99
12.1 Overview 99
12.2 Programming Sequence 99
12.3 API Descriptions 100
12.3.1 RSI_WWDT_Init 100
12.3.2 RSI_WWDT_IntrUnMask 101
12.3.3 RSI_WWDT_ConfigIntrTimer 101
12.3.4 RSI_WWDT_ConfigSysRstTimer 102
12.3.5 RSI_WWDT_Start 102
12.3.6 RSI_WWDT_ReStart 103
12.3.7 RSI_WWDT_IntrMask 103
12.3.8 RSI_WWDT_IntrClear 104

12.3.9 RSI_WWDT_GetIntrStatus 104
12.3.10 RSI_WWDT_DeInit 104
12.3.11 RSI_WWDT_Disable 105

13 Real-time Clock (RTC) 106
13.1 Overview 106
13.2 Programming Sequence 106
13.3 API Descriptions 108
13.3.1 RSI_RTC_Init 108
13.3.2 RSI_RTC_Start 108
13.3.3 RSI_RTC_SetDateTime 109
13.3.4 RSI_RTC_SetAlarmDateTime 109
13.3.5 RSI_RTC_AlamEnable 110
13.3.6 RSI_RTC_IntrUnMask 111
13.3.7 RSI_RTC_GetDateTime 111
13.3.8 RSI_RTC_GetAlarmDateTime 112
13.3.9 RSI_RTC_SetDayOfWeek 112
13.3.10 RSI_RTC_GetIntrStatus 113
13.3.11 RSI_RTC_IntrClear 113
13.3.12 RSI_RTC_IntrMask 114
13.3.13 RSI_RTC_Stop 114

14 Clocks (CLK) 116
14.1 Overview 116
14.2 Programming Sequence 116
14.3 API Descriptions 117
14.3.1 RSI_CLK_SocPllClkBypassEnable 117
14.3.2 RSI_CLK_IntfPLLClkBypassEnable 118
14.3.3 RSI_CLK_CanClkConfig 118
14.3.4 RSI_CLK_SocPllClkEnable 119
14.3.5 RSI_CLK_SocPllPdEnable 119
14.3.6 RSI_CLK_SocPllTurnOn 120
14.3.7 RSI_CLK_SocPllTurnOff 121
14.3.8 RSI_CLK_I2sPllTurnOff 121
14.3.9 RSI_CLK_I2sPllClkReset 121
14.3.10 RSI_CLK_I2sPllTurnOn 122
14.3.11 RSI_CLK_I2sPllSetFreqDiv 122
14.3.12 RSI_CLK_I2sPllClkEnable 123
14.3.13 RSI_CLK_IntfPllClkReset 123
14.3.14 RSI_CLK_IntfPllClkEnable 124
14.3.15 RSI_CLK_IntfPLLTurnOn 125
14.3.16 RSI_CLK_IntfPLLTurnOn 125
14.3.17 RSI_CLK_IntfPLLTurnOff 125
14.3.18 RSI_CLK_I2sClkConfig 126
14.3.19 RSI_CLK_PeripheralClkEnable2 126
14.3.20 RSI_CLK_M4SocClkConfig 128
14.3.21 RSI_CLK_SocPllSetFreqDiv 129
14.3.22 RSI_CLK_IntfPllSetFreqDiv 129
14.3.23 RSI_CLK_SlpClkConfig 130
14.3.24 RSI_CLK_EthernetClkConfig 131
14.3.25 RSI_CLK_SsiMstClkConfig 132
14.3.26 RSI_CLK_SdMemClkConfig 132
14.3.27 RSI_CLK_CtClkConfig 133
14.3.28 RSI_CLK_CciClkConfig 134
14.3.29 RSI_CLK_CheckPllLock 135
14.3.30 RSI_CLK_SetSocPllFreq 135
14.3.31 RSI_CLK_SocPllClkSet 136
14.3.32 RSI_CLK_SocPllClkReset 137
14.3.33 RSI_CLK_I2sPllClkBypassEnable 137
14.3.34 RSI_CLK_I2sPllPdEnable 138
14.3.35 RSI_CLK_SetI2sPllFreq 138
14.3.36 RSI_CLK_I2sPllClkSet 139
14.3.37 RSI_CLK_IntfPllPdEnable 139
14.3.38 RSI_CLK_SetIntfPllFreq 140
14.3.39 RSI_CLK_IntfPllClkSet 140
14.3.40 RSI_CLK_PeripheralClkEnable1 141
14.3.41 RSI_CLK_PeripheralClkDisable1 142
14.3.42 RSI_CLK_PeripheralClkDisable2 143
14.3.43 RSI_CLK_PeripheralClkEnable3 144
14.3.44 RSI_CLK_PeripheralClkDisable3 145
14.3.45 RSI_CLK_DynamicClkGateDisable 147
14.3.46 RSI_CLK_DynamicClkGateDisable2 148

14.3.47 RSI_CLK_DynamicClkGateEnable ... 149
14.3.48 RSI_CLK_DynamicClkGateEnable2 ... 150
14.3.49 RSI_ULPSS_EnableRefClks ... 150
14.3.50 RSI_ULPSS_DisableRefClks ... 151
14.3.51 RSI_CLK_M4ssRefClkConfig ... 152
14.3.52 RSI_CLK_QspiClkConfig ... 152
14.3.53 RSI_CLK_UsartClkConfig ... 153
14.3.54 RSI_CLK_McuClkOutConfig ... 154
14.3.55 RSI_CLK_M4SocClkDiv ... 155
14.3.56 RSI_CLK_QspiClkDiv ... 156
14.3.57 RSI_CLK_CtClkDiv ... 156
14.3.58 RSI_CLK_SsiMstClkDiv ... 157
14.3.59 RSI_CLK_CciClkDiv ... 157
14.3.60 RSI_CLK_I2sClkDiv ... 158
14.3.61 RSI_CLK_SdmemClkDiv ... 158
14.3.62 RSI_CLK_UsartClkDiv ... 159
14.3.63 RSI_CLK_SlpClkCalibConfig ... 160
14.3.64 RSI_CLK_GspiClkConfig ... 160
14.3.65 RSI_CLK_I2CClkConfig ... 161
14.3.66 RSI_CLK_XtalClkConfig ... 162
14.3.67 RSI_CLK_USBClkConfig ... 162
14.3.68 RSI_CLK_PeripheralClkEnable ... 163
14.3.69 RSI_CLK_PeripheralClkDisable ... 164

15 Companion Chip Interface (CCI) ... 166
15.1 Overview ... 166
15.2 Programming Sequence ... 166
15.3 API Descriptions ... 167
15.3.1 RSI_CCI_AmsEnable ... 167
15.3.2 RSI_CCI_AMS_Initialise ... 167
15.3.3 RSI_CCI_SetFifoThreshlod ... 168
15.3.4 RSI_CCI_PrefetchEnable ... 168
15.3.5 RSI_CCI_IntClear ... 169

16 Configurable Timers (CT) ... 170
16.1 Overview ... 170
16.2 Programming Sequence ... 170
16.3 API Descriptions ... 172
16.3.1 RSI_CT_Config ... 172
16.3.2 RSI_CT_SetControl ... 173
16.3.3 RSI_CT_StartEventSelect ... 173
16.3.4 RSI_CT_HaltEventSelect ... 175
16.3.5 RSI_CT_ContinueEventSelect ... 176
16.3.6 RSI_CT_InterruptEventSelect ... 178
16.3.7 RSI_CT_InterruptEventConfig ... 179
16.3.8 RSI_CT_OutputEventSelect ... 180
16.3.9 RSI_CT_OutputEventConfig ... 181
16.3.10 RSI_CT_GetInterruptStatus ... 182
16.3.11 RSI_CT_InterruptClear ... 182
16.3.12 RSI_CT_ResumeHaltEvent ... 183
16.3.13 RSI_CT_StartEventConfig ... 183
16.3.14 RSI_CT_ClearControl ... 184
16.3.15 RSI_CT_ContinueEventConfig ... 185
16.3.16 RSI_CT_HaltEventConfig ... 185
16.3.17 RSI_CT_IncrementEventConfig ... 186
16.3.18 RSI_CT_CaptureEventConfig ... 187
16.3.19 RSI_CT_IncrementEventSelect ... 187
16.3.20 RSI_CT_CaptureEventSelect ... 189
16.3.21 RSI_CT_OutputEventADCTrigger ... 190
16.3.22 RSI_CT_SetCount ... 190
16.3.23 RSI_CT_OCUConfigSet ... 191
16.3.24 RSI_CT_OCUConfigReset ... 192
16.3.25 RSI_CT_InterruptEnable ... 192
16.3.26 RSI_CT_InterruptDisable ... 193
16.3.27 RSI_CT_EdgeLevelEventControl ... 194
16.3.28 RSI_CT_SetTimerMuxSelect ... 195
16.3.29 RSI_CT_PeripheralReset ... 196
16.3.30 RSI_CT_StartSoftwareTrig ... 196
16.3.31 RSI_CT_OCUModeSet ... 197
16.3.32 RSI_CT_SetMatchCount ... 197
16.3.33 RSI_CT_CaptureRead ... 198
16.3.34 RSI_CT_GetCounter ... 198

16.3.35 RSI_CT_SetCounerSync ... 199
16.3.36 RSI_CT_OCUHighLowToggleSelect ... 199
16.3.37 RSI_CT_WFGControlConfig ... 200
16.3.38 RSI_CT_OCUControl ... 201
16.3.39 RSI_CT_WFGComapreValueSet ... 202
16.3.40 RSI_CT_StopEventConfig ... 203
16.3.41 RSI_CT_StopEventSelect ... 203

17 Cyclic Redundancy Check (CRC) ... 206
17.1 Overview ... 206
17.2 Programming sequence ... 206
17.3 ... 206
17.4 API Description ... 206
17.4.1 RSI_CRC_SetGenControl ... 206
17.4.2 RSI_CRC_Polynomial ... 207
17.4.3 RSI_CRC_Polynomial_Width ... 207
17.4.4 RSI_CRC_LfsrInit ... 208
17.4.5 RSI_CRC_Use_Swapped_Init ... 209
17.4.6 RSI_CRC_Set_DataWidthType ... 209
17.4.7 RSI_CRC_SetFifoThresholds ... 210
17.4.8 RSI_CRC_WriteData ... 210
17.4.9 RSI_Monitor_CRCcalc ... 211
17.4.10 RSI_CRC_GetGenStatus ... 211
17.4.11 RSI_CRC_LfsrDynamicWrite ... 212
17.4.12 RSI_CRC_GetFifoStatus ... 212
17.4.13 RSI_CRC_ResetFifo ... 213

18 eFUSE ... 214
18.1 Overview ... 214
18.2 Programming sequence ... 214
18.3 API Descriptions ... 216
18.3.1 RSI_EFUSE_WriteBit ... 216
18.3.2 RSI_EFUSE_FsmReadByte ... 217
18.3.3 RSI_EFUSE_Enable ... 217
18.3.4 RSI_EFUSE_Disable ... 218
18.3.5 RSI_EFUSE_ReadData ... 218
18.3.6 RSI_EFUSE_WriteAddr ... 219
18.3.7 RSI_EFUSE_MemMapReadByte ... 219
18.3.8 RSI_EFUSE_MemMapReadWord ... 220

19 Enhanced General Purpose Input Output (EGPIO) ... 221
19.1 Overview ... 221
19.2 Programming sequence ... 221
19.3 API Descriptions ... 222
19.3.1 RSI_EGPIO_SetPinMux ... 222
19.3.2 RSI_EGPIO_SetDir ... 223
19.3.3 RSI_EGPIO_SetPin ... 224
19.3.4 RSI_EGPIO_GetPin ... 224
19.3.5 RSI_EGPIO_GetDir ... 225
19.3.6 RSI_EGPIO_PinIntSel ... 225
19.3.7 RSI_EGPIO_SetIntFallEdgeEnable ... 226
19.3.8 RSI_EGPIO_SetIntFallEdgeDisable ... 227
19.3.9 RSI_EGPIO_SetIntRiseEdgeEnable ... 227
19.3.10 RSI_EGPIO_SetIntRiseEdgeDisable ... 228
19.3.11 RSI_EGPIO_SetIntLowLevelEnable ... 228
19.3.12 RSI_EGPIO_IntMask ... 229
19.3.13 RSI_EGPIO_IntUnMask ... 229
19.3.14 RSI_EGPIO_SetIntLowLevelDisable ... 230
19.3.15 RSI_EGPIO_SetIntHighLevelEnable ... 230
19.3.16 RSI_EGPIO_SetIntHighLevelDisable ... 231
19.3.17 RSI_EGPIO_GetIntStat ... 231
19.3.18 RSI_EGPIO_IntClr ... 232
19.3.19 RSI_EGPIO_UlpSocGpioMode ... 232
19.3.20 RSI_EGPIO_SetPortMask ... 233
19.3.21 RSI_EGPIO_SetPortUnMask ... 234
19.3.22 RSI_EGPIO_PortMaskedLoad ... 234
19.3.23 RSI_EGPIO_SetPort ... 235
19.3.24 RSI_EGPIO_PortLoad ... 235
19.3.25 RSI_EGPIO_WordLoad ... 236
19.3.26 RSI_EGPIO_ClrPort ... 236
19.3.27 RSI_EGPIO_TogglePort ... 237
19.3.28 RSI_EGPIO_GetPort ... 237

19.3.29 RSI_EGPIO_GroupIntOneEnable ... 238
19.3.30 RSI_EGPIO_GroupIntOneDisable ... 239
19.3.31 RSI_EGPIO_GroupIntTwoEnable ... 239
19.3.32 RSI_EGPIO_GroupIntMask ... 240
19.3.33 RSI_EGPIO_GroupIntUnMask ... 240
19.3.34 RSI_EGPIO_GroupIntEnable ... 241
19.3.35 RSI_EGPIO_GroupIntDisable ... 241
19.3.36 RSI_EGPIO_GroupIntLevel ... 242
19.3.37 RSI_EGPIO_GroupIntEdge ... 242
19.3.38 RSI_EGPIO_GroupIntAnd ... 243
19.3.39 RSI_EGPIO_GroupIntOr ... 243
19.3.40 RSI_EGPIO_GroupIntStat ... 244
19.3.41 RSI_EGPIO_GroupIntWkeUpEnable ... 244
19.3.42 RSI_EGPIO_GroupIntWkeUpDisable ... 245
19.3.43 RSI_EGPIO_GroupIntClr ... 245
19.3.44 RSI_EGPIO_GroupIntTwoDisable ... 246
19.3.45 RSI_EGPIO_SetGroupIntOnePol ... 247
19.3.46 RSI_EGPIO_SetGroupIntTwoPol ... 247
19.3.47 RSI_EGPIO_HostPadsGpioModeEnable ... 248
19.3.48 RSI_EGPIO_HostPadsGpioModeDisable ... 248
19.3.49 RSI_EGPIO_PadSelectionEnable ... 249
19.3.50 RSI_EGPIO_PadSelectionDisable ... 249
19.3.51 RSI_EGPIO_PadReceiverEnable ... 250
19.3.52 RSI_EGPIO_PadReceiverDisable ... 250
19.3.53 RSI_EGPIO_PadSdioConnected ... 251
19.3.54 RSI_EGPIO_PadDriverDisableState ... 251
19.3.55 RSI_EGPIO_PadDriverStrengthSelect ... 252
19.3.56 RSI_EGPIO_PadPowerOnStartEnable ... 252
19.3.57 RSI_EGPIO_PadActiveHighSchmittTrigger ... 253
19.3.58 RSI_EGPIO_PadSlewRateControll ... 254
19.3.59 RSI_EGPIO_UlpPadReceiverEnable ... 254
19.3.60 RSI_EGPIO_UlpPadReceiverDisable ... 255
19.3.61 RSI_EGPIO_UlpPadDriverDisableState ... 255
19.3.62 RSI_EGPIO_UlpPadDriverStrengthSelect ... 256
19.3.63 RSI_EGPIO_UlpPadPowerOnStartEnable ... 256
19.3.64 RSI_EGPIO_UlpPadActiveHighSchmittTrigger ... 257
19.3.65 RSI_EGPIO_UlpPadSlewRateControl ... 257

20 Filter Interpolation Matrix Multiplication(FIM) ... 259
20.1 Overview ... 259
20.2 Programming Sequence ... 259
20.3 API Descriptions ... 265
20.3.1 fim_read_data ... 265
20.3.2 rsi_fim_scalar_add_q15 ... 266
20.3.3 fim_scalar_sub_q7 ... 267
20.3.4 rsi_fim_scalar_sub_q15 ... 268
20.3.5 fim_scalar_sub_q31 ... 268
20.3.6 fim_scalar_sub_f32 ... 269
20.3.7 fim_scalar_mul_q15 ... 270
20.3.8 fim_vector_add_q15 ... 271
20.3.9 fim_vector_sub_q15 ... 271
20.3.10 fim_vector_mul_q15 ... 272
20.3.11 fim_absSqr_q7 ... 273
20.3.12 fim_absSqr_q15 ... 273
20.3.13 fim_absSqr_q31 ... 274
20.3.14 fim_absSqr_f32 ... 275
20.3.15 fim_interrupt_handler ... 275
20.3.16 rsi_fim_fir_q15 ... 276
20.3.17 rsi_fim_Iir_init_f32 ... 276
20.3.18 rsi_fim_Iir_f32 ... 277
20.3.19 rsi_fim_Iir_init_q31 ... 278
20.3.20 rsi_fim_Iir_q31 ... 279
20.3.21 rsi_fim_Iir_init_q15 ... 279
20.3.22 rsi_fim_Iir_q15 ... 280
20.3.23 rsi_fim_fir_interpolate_q15 ... 281

21 Overview ... 283

22 Micro Direct Memory Access (µDMA) ... 284
22.1 Overview ... 284
22.2 Programming sequence ... 284
22.3 API Descriptions ... 285
22.3.1 UDMAx_Initialize ... 285

22.3.2 UDMAx_Uninitialize ... 286
22.3.3 UDMAx_ChannelConfigure ... 287
22.3.4 UDMAx_ChannelEnable ... 288
22.3.5 UDMAx_ChannelDisable ... 289
22.3.6 UDMAx_ChannelGetCount ... 289
22.3.7 ... 291

23 Pulse Width Modulator (PWM) ... 292
23.1 Overview ... 292
23.2 Programming sequence ... 292
23.3 API Descriptions ... 293
23.3.1 RSI_MCPWM_BaseTimerSelect ... 293
23.3.2 RSI_MCPWM_SetOutputPolarity ... 294
23.3.3 RSI_MCPWM_InterruptHandler ... 295
23.3.4 RSI_MCPWM_SetOutputMode ... 295
23.3.5 RSI_MCPWM_SetBaseTimerMode ... 296
23.3.6 RSI_MCPWM_SetTimePeriod ... 296
23.3.7 RSI_MCPWM_GetTimePeriod ... 297
23.3.8 RSI_MCPWM_SetDutyCycle ... 298
23.3.9 RSI_MCPWM_InterruptEnable ... 298
23.3.10 RSI_MCPWM_Start ... 299
23.3.11 RSI_MCPWM_OutputOverrideEnable ... 300
23.3.12 RSI_MCPWM_OverrideValueSet ... 301
23.3.13 RSI_MCPWM_OverrideValueReSet ... 301
23.3.14 RSI_MCPWM_OutputOverrideDisable ... 302
23.3.15 RSI_MCPWM_SpecialEventTriggerConfig ... 303
23.3.16 RSI_MCPWM_DeadTimeValueSet ... 304
23.3.17 RSI_MCPWM_ChannelReset ... 305
23.3.18 RSI_MCPWM_CounterReset ... 305
23.3.19 RSI_MCPWM_PeriodControlConfig ... 306
23.3.20 RSI_MCPWM_FaultAValueSet ... 307
23.3.21 RSI_MCPWM_FaultBValueSet ... 308
23.3.22 RSI_MCPWM_ReadCounter ... 308
23.3.23 RSI_MCPWM_GetCounterDir ... 309
23.3.24 RSI_MCPWM_DeadTimeEnable ... 310
23.3.25 RSI_MCPWM_DeadTimeEnable ... 310
23.3.26 RSI_MCPWM_DeadTimeDisable ... 311
23.3.27 RSI_MCPWM_DutyCycleControlSet ... 311
23.3.28 RSI_MCPWM_DutyCycleControlReset ... 312
23.3.29 RSI_MCPWM_OverrideControlSet ... 313
23.3.30 RSI_MCPWM_OverrideControlReSet ... 313
23.3.31 RSI_MCPWM_OverrideControlSet ... 314
23.3.32 RSI_MCPWM_FaultControlSet ... 314
23.3.33 RSI_MCPWM_FaultControlReSet ... 315
23.3.34 RSI_MCPWM_SpecialEventTriggerEnable ... 316
23.3.35 RSI_MCPWM_SpecialEventTriggerDisable ... 316
23.3.36 RSI_MCPWM_DeadTimeControlSet ... 317
23.3.37 RSI_MCPWM_DeadTimeControlReSet ... 317
23.3.38 RSI_PWM_GetInterruptStatus ... 318
23.3.39 RSI_MCPWM_InterruptClear ... 319
23.3.40 RSI_MCPWM_InterruptDisable ... 320
23.3.41 RSI_MCPWM_ExternalTriggerControl ... 321
23.3.42 RSI_MCPWM_Stop ... 322
23.3.43 ... 322

24 Quadrature Encoder Interface (QEI) ... 323
24.1 Overview ... 323
24.2 Programming Sequence ... 323
24.3 API Descriptions ... 325
24.3.1 RSI_QEI_Enable ... 325
24.3.2 void RSI_QEI_SetMode ... 325
24.3.3 RSI_QEI_GetMode ... 326
24.3.4 RSI_QEI_SetConfiguration ... 326
24.3.5 RSI_QEI_ConfigureDeltaTimeAndFreq ... 327
24.3.6 RSI_QEI_StartVelocityCounter ... 328
24.3.7 RSI_QEI_IntrUnMask ... 328
24.3.8 RSI_QEI_GetIntrStatus ... 329
24.3.9 RSI_QEI_ClrIntrStatus ... 329
24.3.10 RSI_QEI_IntrMask ... 330
24.3.11 RSI_QEI_GetVelocity ... 331
24.3.12 RSI_QEI_GetPosition ... 331
24.3.13 RSI_QEI_GetIndex ... 332

24.3.14 RSI_QEI_SetControls ... 332
24.3.15 RSI_QEI_ClrControls ... 333
24.3.16 RSI_QEI_SetMaxPosCnt ... 334
24.3.17 RSI_QEI_GetMaxPosCnt ... 334
24.3.18 RSI_QEI_SetPosMatch ... 335
24.3.19 RSI_QEI_GetPosMatch ... 335
24.3.20 RSI_QEI_SetMaxIndex ... 336
24.3.21 RSI_QEI_GetMaxIndex ... 336
24.3.22 RSI_QEI_GetDirection ... 337
24.3.23 RSI_QEI_GetStatus ... 337
24.3.24 RSI_QEI_StopVelocityCounter ... 338
24.3.25 RSI_QEI_SetDigitalFilterClkDiv ... 338
24.3.26 RSI_QEI_GetDigitalFilterClkDiv ... 339
24.3.27 RSI_QEI_SetModuleFreq ... 339
24.3.28 RSI_QEI_GetModuleFreq ... 340
24.3.29 RSI_QEI_SetDeltaTime ... 340
24.3.30 RSI_QEI_GetDeltaTime ... 341
24.3.31 RSI_QEI_ClrConfiguration ... 341
24.3.32 RSI_QEI_Disable ... 342

25 Quad Serial Peripheral Interface Flash(QSPI) ... 344
25.1 Overview ... 344
25.2 Programming Sequence ... 344
25.3 API Descriptions ... 348
25.3.1 RSI_QSPI_SpiInit ... 348
25.3.2 RSI_QSPI_SpiErase ... 349
25.3.3 RSI_QSPI_SpiWrite ... 349
25.3.4 RSI_QSPI_ManualRead ... 350
25.3.5 RSI_QSPI_WriteToFlash ... 351
25.3.6 RSI_QSPI_SwitchQspi2 ... 352
25.3.7 RSI_QSPI_WaitFlashStatusIdle ... 352
25.3.8 RSI_QSPI_EnableStatusRegWrite ... 353
25.3.9 RSI_QSPI_StatusRegWrite ... 353
25.3.10 RSI_QSPI_FlashRegRead ... 354
25.3.11 RSI_QSPI_FlashRegWrite ... 354
25.3.12 RSI_QSPI_SetFlashMode ... 355
25.3.13 RSI_QSPI_ConfigQflash4Read ... 356
25.3.14 RSI_QSPI_AutoInit ... 356
25.3.15 RSI_QSPI_AutoRead ... 357
25.3.16 RSI_QSPI_FlashInit ... 357
25.3.17 RSI_QSPI_SpiRead ... 358
25.3.18 RSI_QSPI_Usleep ... 359
25.3.19 RSI_QSPI_WriteBlockProtect ... 359
25.3.20 RSI_QSPI_QspiLoadKey ... 360
25.3.21 RSI_QSPI_QspiLoadNonce ... 360
25.3.22 RSI_QSPI_SegSecEn ... 361
25.3.23 RSI_QSPI_StatusControlRegWrite ... 362
25.3.24 RSI_QSPI_FlashProtection ... 362

26 Random Number Generator (RNG) ... 364
26.1 Overview ... 364
26.2 Programming sequence ... 364
26.3 API Descriptions ... 364
26.3.1 RSI_RNG_Start ... 364
26.3.2 RSI_RNG_GetBytes ... 365
26.3.3 RSI_RNG_Stop ... 365

27 General Purpose Direct Memory Access (GPDMA) ... 367
27.1 Overview ... 367
27.2 Programming sequence ... 367
27.3 API Descriptions ... 369
27.3.1 RSI_GPDMA_GetMemSize ... 369
27.3.2 RSI_GPDMA_Init ... 370
27.3.3 RSI_GPDMA_RegisterCallback ... 370
27.3.4 RSI_GPDMA_FIFOConfig ... 371
27.3.5 RSI_GPDMA_SetupChannel ... 372
27.3.6 RSI_GPDMA_BuildDescriptors ... 372
27.3.7 RSI_GPDMA_SetupChannelTransfer ... 373
27.3.8 RSI_GPDMA_InterruptEnable ... 374
27.3.9 RSI_GPDMA_DMAChannelTrigger ... 375
27.3.10 RSI_GPDMA_InterruptHandler ... 375
27.3.11 RSI_RPDMA_InterruptStatus ... 376

27.3.12 RSI_GPDMA_InterruptClear ... 377
27.3.13 RSI_GPDMA_InterruptDisable ... 377
27.3.14 RSI_GPDMA_ErrorStatusClear ... 378
27.3.15 RSI_GPDMA_GetErrorStatus ... 379
27.3.16 RSI_GPDMA_ChannelIsEnabled ... 379
27.3.17 RSI_GPDMA_AbortChannel ... 380
27.3.18 RSI_RPDMA_GetCapabilities ... 381
27.3.19 RSI_GPDMA_GetVersion ... 381
27.3.20 RSI_GPDMA_DeInit ... 382

28 Serial Input Output (SIO) ... 383
28.1 Overview ... 383
28.2 Programming Sequence: ... 383
28.3 API Descriptions ... 386
28.3.1 RSI_SIO_Init ... 386
28.3.2 RSI_SIO_InitSpi ... 386
28.3.3 RSI_SIO_SpiCsAssert ... 387
28.3.4 RSI_SIO_SpiTrasnfer ... 387
28.3.5 RSI_SIO_SpiCsDeAssert ... 388
28.3.6 RSI_SIO_UartInit ... 388
28.3.7 RSI_SIO_UARTSend ... 389
28.3.8 RSI_SIO_UARTReadBlocking ... 390
28.3.9 RSI_SIO_UARTSendBlocking ... 390
28.3.10 RSI_SIO_UARTRead ... 391
28.3.11 RSI_SIO_I2cGenerateStart ... 391
28.3.12 RSI_SIO_I2cTransfer ... 392
28.3.13 RSI_SIO_I2cWrite ... 393
28.3.14 RSI_SIO_I2cRead ... 393
28.3.15 RSI_SIO_I2cGenerateStop ... 394

29 Timer ... 395
29.1 Overview ... 395
29.2 Programming sequence ... 395
29.3 API Descriptions ... 396
29.3.1 RSI_TIMERS_SetTimerMode ... 396
29.3.2 RSI_TIMERS_SetTimerType ... 397
29.3.3 RSI_TIMERS_MicroSecTimerConfig ... 397
29.3.4 RSI_TIMERS_SetMatch ... 398
29.3.5 RSI_TIMERS_InterruptEnable ... 399
29.3.6 RSI_TIMERS_TimerStart ... 399
29.3.7 RSI_TIMERS_ReadTimer ... 400
29.3.8 RSI_TIMERS_GetTimerType ... 400
29.3.9 RSI_TIMERS_InterruptStatus ... 401
29.3.10 RSI_TIMERS_InterruptClear ... 401
29.3.11 RSI_TIMERS_InterruptDisable ... 402
29.3.12 RSI_TIMERS_TimerStop ... 402

30 Power Management Unit(PMU) ... 404
30.1 Overview ... 404
30.2 Programming sequence ... 404
30.3 API Description ... 406
30.3.1 RSI_PS_PowerStateChangePs4toPs2 ... 406
30.3.2 RSI_PS_PowerStateChangePs2toPs4 ... 407
30.3.3 RSI_PS_ClrWkpUpStatus ... 408
30.3.4 RSI_PS_RetentionSleepConfig ... 409
30.3.5 RSI_PS_EnterDeepSleep ... 410
30.3.6 RSI_PS_PowerStateChangePs4toPs3 ... 410
30.3.7 RSI_PS_PowerStateChangePs3toPs4 ... 411
30.3.8 RSI_PS_M4ssPeriPowerDown ... 411
30.3.9 RSI_PS_M4ssPeriPowerUp ... 412
30.3.10 RSI_PS_UlpssPeriPowerDown ... 413
30.3.11 RSI_PS_UlpssPeriPowerUp ... 414
30.3.12 RSI_PS_NpssPeriPowerUp ... 415
30.3.13 RSI_PS_NpssPeriPowerDown ... 416
30.3.14 RSI_PS_M4ssRamBanksPowerDown ... 417
30.3.15 RSI_PS_SetRamRetention ... 417
30.3.16 RSI_PS_ClrRamRetention ... 418
30.3.17 RSI_PS_UlpssRamBanksPowerDown ... 419
30.3.18 RSI_PS_UlpssRamBanksPowerUp ... 419
30.3.19 void RSI_PS_SetRamRetention ... 420
30.3.20 RSI_PS_ClrRamRetention ... 421
30.3.21 RSI_PS_UlpssRamBanksPowerDown ... 421

30.3.22 RSI_PS_UlpssRamBanksPowerUp ... 422
30.3.23 RSI_PS_SetWkpSources ... 423
30.3.24 RSI_PS_ClrWkpSources ... 424
30.3.25 RSI_PS_GetWkpSources ... 425
30.3.26 RSI_PS_EnableFirstBootUp ... 425
30.3.27 RSI_PS_PowerSupplyEnable ... 426
30.3.28 RSI_PS_PowerSupplyDisable ... 426
30.3.29 RSI_PS_FsmHfClkSel ... 427
30.3.30 RSI_PS_FsmLfClkSel ... 428
30.3.31 RSI_PS_PmuGoodTimeDurationConfig ... 428
30.3.32 RSI_PS_XtalGoodTimeDurationConfig ... 429
30.3.33 RSI_PS_Ps2PmuLdoOffDelayConfig ... 429
30.3.34 RSI_PS_Ps4PmuBuckOnDelayConfig ... 430
30.3.35 RSI_PS_GetWkpUpStatus ... 430
30.3.36 RSI_PS_GetComnIntrSts ... 430
30.3.37 RSI_PS_NpssIntrUnMask ... 431
30.3.38 RSI_PS_NpssIntrMask ... 432
30.3.39 RSI_PS_EnableLpSleep ... 433
30.3.40 RSI_PS_SkipXtalWaitTime ... 433
30.3.41 RSI_PS_EnterShipMode ... 434
30.3.42 RSI_PS_UlpToDcDcMode ... 435
30.3.43 RSI_PS_BodPwrGateButtonCalibEnable ... 435
30.3.44 RSI_PS_BodPwrGateButtonCalibDisable ... 435
30.3.45 RSI_PS_AnalogPeriPtatEnable ... 436
30.3.46 RSI_PS_AnalogPeriPtatDisable ... 436
30.3.47 RSI_PS_BodClksPtatEnable ... 436
30.3.48 RSI_PS_BodClksPtatDisable ... 437
30.3.49 RSI_PS_XtalEnable ... 437
30.3.50 RSI_PS_XtalDisable ... 438
30.3.51 RSI_PS_FlashLdoEnable ... 438
30.3.52 RSI_PS_FlashLdoDisable ... 438
30.3.53 RSI_PS_SocPllSpiDisable ... 439
30.3.54 RSI_PS_SocPllVddIsoEnable ... 439
30.3.55 RSI_PS_SocPllVddIsoDiable ... 440
30.4 RSI_PS_SocPllSpiEnable ... 440
30.4.1 RSI_ConfigBuckBoost ... 440
30.4.2 RSI_ConfigPmuShutDown ... 441
30.4.3 RSI_ConfigXtal ... 441
30.4.4 RSI_PS_PmuSetConfig ... 442
30.4.5 RSI_PS_PmuClrConfig ... 443

31 Comparator ... 444
31.1 Overview ... 444
31.2 Programming Sequence ... 444
31.3 API Descriptions ... 445
31.3.1 RSI_COMP_Config ... 445
31.3.2 RSI_COMP_Enable ... 446
31.3.3 RSI_COMP_ResistorBank_Enable ... 447
31.3.4 RSI_COMP_ResistorBank_Thrsh ... 447
31.3.5 RSI_COMP_ReferenceScalarOut ... 448
31.3.6 RSI_COMP_RefenceBufferOut_Enable ... 448

32 PUF ... 450
32.1 Overview ... 450
32.2 Programming Sequence ... 450
32.2.1 rsi_puf_enroll_req ... 451
32.2.2 rsi_puf_enroll_disable_req ... 451
32.2.3 rsi_puf_start_req ... 452
32.2.4 rsi_puf_set_key_req ... 452
32.2.5 rsi_puf_set_key_disable_req ... 453
32.2.6 rsi_puf_get_key_req ... 454
32.2.7 rsi_puf_get_key_disable_req ... 454
32.2.8 rsi_puf_load_key_req ... 455
32.2.9 rsi_puf_intr_key_req ... 455
32.2.10 rsi_puf_aes_encrypt_req ... 456
32.2.11 rsi_puf_aes_decrypt_req ... 457
32.2.12 rsi_puf_aes_mac_req ... 458

33 Crypto ... 460
33.1 AES ... 460
33.1.1 Overview ... 460
33.1.2 Programming Sequence ... 460

33.2 ECDH ... 461
33.2.1 Overview ... 461
33.2.2 Programming Sequence ... 461
33.2.3 Point Addition ... 463
33.2.4 Point Subtraction ... 464
33.2.5 Point Doubling (PD) ... 465
33.2.6 Point Affine ... 466
33.3 Exponentiation ... 466
33.3.1 Overview ... 466
33.3.2 Programming Sequence ... 466
33.4 HMAC-SHA ... 468
33.4.1 Overview ... 468
33.4.2 Programming Sequence ... 468
33.5 SHA ... 469
33.5.1 Overview ... 469
33.5.2 Programming Sequence ... 469
33.5.3 rsi_sha ... 470

34 Analog to Digital Converter ... 472
34.1 Overview ... 472
34.2 Programming Sequence ... 472
34.3 API Descriptions ... 475
34.3.1 RSI_ADC_PingPongMemoryAdrConfig ... 475
34.3.2 RSI_ADC_PingPongReInit ... 476
34.3.3 RSI_ADC_PingpongEnable ... 476
34.3.4 RSI_ADC_PingpongDisable ... 477
34.3.5 RSI_ADC_InternalPerChnlDmaEnable ... 477
34.3.6 RSI_ADC_InternalPerChnlDmaDisable ... 478
34.3.7 RSI_ADC_Config ... 478
34.3.8 RSI_ADC_ChannelConfig ... 479
34.3.9 RSI_ADC_StaticMode ... 480
34.3.10 RSI_ADC_ChannelSamplingRate ... 480
34.3.11 RSI_ADC_FifoMode ... 481
34.3.12 RSI_ADC_Start ... 482
34.3.13 RSI_ADC_Stop ... 482
34.3.14 RSI_ADC_ChnlEnable ... 483
34.3.15 RSI_ADC_ChnlDisable ... 483
34.3.16 RSI_ADC_ReadDataStatic ... 484
34.3.17 RSI_ADC_ReadData ... 484
34.3.18 RSI_ADC_ClkDivfactor ... 485
34.3.19 RSI_ADC_ChnlIntrUnMask ... 486
34.3.20 RSI_ADC_ChnlIntrMask ... 486
34.3.21 RSI_ADC_ChnlIntrClr ... 487
34.3.22 RSI_ADC_ChnlIntrStatus ... 487
34.3.23 RSI_ADC_PowerControl ... 488
34.3.24 RSI_ADC_NoiseAvgMode ... 488
34.3.25 RSI_ADC_ThresholdConfig ... 489
34.3.26 RSI_ADC_InterruptHandler ... 490
34.3.27 RSI_ADC_ThreshInterruptClr ... 490
34.3.28 RSI_ADC_Calibration ... 491
34.3.29 RSI_ADC_AUXBypassLdoEnable ... 491
34.3.30 RSI_ADC_GainOffsetCal ... 492
34.3.31 RSI_ADC_VrefCal ... 493

35 Digital to Analog Converter ... 494
35.1 Overview ... 494
35.2 Programming Sequence ... 494
35.3 API Descriptions ... 496
35.3.1 RSI_DAC_Config ... 496
35.3.2 RSI_DAC_WriteData ... 497
35.3.3 RSI_DAC_Start ... 497
35.3.4 RSI_DAC_Stop ... 498
35.3.5 RSI_DAC_ClkDivFactor ... 498
35.3.6 RSI_DAC_PowerControl ... 499
35.3.7 RSI_DAC_SetFifoThreshold ... 499
35.3.8 RSI_DAC_DynamicModeConfig ... 500
35.3.9 RSI_DAC_DynamicModeWriteData ... 501
35.3.10 RSI_DAC_DynamicModeStart ... 501
35.3.11 RSI_DAC_DynamicModeStop ... 502

36 Building a new project in Keil iDE ... 504
36.1 Start new project ... 504

36.2 Select device 504
36.3 Add source files 505
36.4 Include paths 506
36.5 Creating Examples: 507
36.6 508

37 Wake-Fi Receive 509
37.1 Introduction 509
37.1.1 Example Overview 509
37.2 Configuration and Execution of the Application 509
37.2.1 Configuring the Application 509
37.2.2 Executing the Application 511
37.2.3 API Descriptions 512

38 Voice Activity Detection(VAD) 524
38.1 Overview 524
38.2 Programming Sequence 524
38.3 API Descriptions 527
38.3.1 RSI_VAD_PingPongMemoryAddrConfig 527
38.3.2 RSI_VAD_Config 528
38.3.3 RSI_VAD_Enable 529
38.3.4 RSI_VAD_InterruptClr 529
38.3.5 RSI_VAD_SetAlgorithmThreshold 530
38.3.6 RSI_VAD_Set_Delay 531
38.3.7 RSI_VAD_Input 532
38.3.8 RSI_VAD_FrameEnergyConfig 532
38.3.9 RSI_VAD_Stop 533
38.3.10 RSI_VAD_ProccessDone 533
38.3.11 RSI_VAD_FastClkEnable 534
38.3.12 RSI_VAD_ProcessData 534

39 Capacitive Touch Sensor 536
39.1 Overview 536
39.2 Programming Sequence 536
39.3 API Descriptions 540

40 Deep sleep timer 555
40.1 Overview 555
40.2 Programming sequence 555
40.3 API Description 557
40.3.1 RSI_DST_DurationSet 557
40.3.2 RSI_DST_IntrUnMask 557
40.3.3 RSI_DST_IntrMask 558
40.3.4 RSI_DST_TimerIntrClear 558

41 NPSS GPIOs 559
41.1 Overview 559
41.2 Programming sequence 559
41.3 API Description 561
41.3.1 RSI_NPSSGPIO_SetPinMux 561
41.3.2 RSI_NPSSGPIO_InputBufferEn 562
41.3.3 RSI_NPSSGPIO_SetDir 562
41.3.4 RSI_NPSSGPIO_GetDir 563
41.3.5 RSI_NPSSGPIO_SetPin 563
41.3.6 RSI_NPSSGPIO_GetPin 564
41.3.7 RSI_NPSSGPIO_SetPolarity 564
41.3.8 RSI_NPSSGPIO_SetWkpGpio 565
41.3.9 RSI_NPSSGPIO_ClrWkpGpio 565
41.3.10 RSI_NPSSGPIO_IntrMask 566
41.3.11 RSI_NPSSGPIO_IntrUnMask 566
41.3.12 RSI_NPSSGPIO_SetIntFallEdgeEnable 567
41.3.13 RSI_NPSSGPIO_ClrIntFallEdgeEnable 567
41.3.14 RSI_NPSSGPIO_SetIntRiseEdgeEnable 568
41.3.15 RSI_NPSSGPIO_ClrIntRiseEdgeEnable 568
41.3.16 RSI_NPSSGPIO_SetIntLevelHighEnable 568
41.3.17 RSI_NPSSGPIO_ClrIntLevelHighEnable 569
41.3.18 RSI_NPSSGPIO_SetIntLevelLowEnable 569
41.3.19 RSI_NPSSGPIO_ClrIntLevelLowEnable 570
41.3.20 RSI_NPSSGPIO_ClrIntr 570
41.3.21 RSI_NPSSGPIO_GetIntrStatus 571

42 Redpine MCU SAPI Manual Revision History 572

43 A2DP source application 573

44 Operational Amplifier 574
44.1 Overview 574
44.2 Programming Sequence 574
44.3 API Descriptions 575
44.3.1 Description 583
44.3.2 Description 584
44.3.3 Description 587
44.3.4 Description 588
44.3.5 Description 591
44.3.6 Description 592
44.3.7 Description 595
44.3.8 Description 596
44.3.9 Description 597

45 MCU FreeRTOS Documentation. 599
45.1 1). FreeRTOS Architecture and Porting : 599
45.2 2). FreeRTOS standard reference Guide : 599

About the Document

This document is a preliminary version of Redpine MCU SAPI Manual to customers under a Non-Disclosure Agreement (NDA).

1 Redpine MCU SAPI Introduction

1.1 Overview

The Redpine MCU Simple API (SAPI) is a comprehensive collection of peripheral API and driver code to simplify application development on Redpine MCU products.Users can develop application firmware without having to learn the underlying peripheral register interface and other details.

The SAPI is intended to run on Cortex-M4 core available in WiSeMCU, MCU and other Redpine products. With its uniform API, SAPI enables easy migration between Redpine products. The SAPI provides CMSIS-Driver API for MCU peripherals that are CMSIS-compliant. For other Redpine proprietary peripherals, SAPI provides a proprietary API. This document explains MCU SAPI architecture, configuration and API descriptions.

1.2 Features

- Supports all Redpine MCU peripherals
- CMSIS-Driver compatible for CMSIS-supported peripherals
- Redpine proprietary APIs for Redpine proprietary peripherals
- Works with or without RTOS
- Out-of-box support for FreeRTOS
- Out-of-box support for IAR, Keil and GCC IDEs, portable to other IDEs
- Out-of-box support for Keil MDK middleware (USB host/device stack, file system), FatFs file system, and lwIP.
- Provides example for each peripheral usage

1.3 Architecture

Figure 1 Redpine MCU Architecture

- **Redpine Driver :** Contains all Redpine proprietary driver peripheral list in Redpine MCU.

- **CMSIS Driver :** Contains all CMSIS supported driver peripheral list in Redpine MCU.
- **Middleware components**: Uses CMSIS Driver to provide higher-level services to User Application.

1.4 Quick start

1. Quickstart with Redpine MCU SAPIs : Explain the Redpine MCU Examples with Keil and IAR IDE.
2. Building a project : Explain the building a project of Redpine MCU SAPIs with Keil IDE.
3. Building a project : Explain the building a project of Redpine MCU SAPIs with IAR Embedded workbench.
4. Toolchain and Board Porting Guide : Explain the Porting Guide of Redpine MCU SAPIs with different IDE.

1.5 Redpine MCU Driver Library

The MCU SAPIs are designed to ease the use of the peripherals of the Repine MCU series. You do not need to know about the register and bit structure of the peripherals. You need only to set up a configuration and make use of the API functions, such as initialization of peripheral. For most of the peripherals, several example codes are provided. You can study how to use the peripheral quickly by running the according examples.

Real-time Clock (RTC) :Explains about Calender block

Companion Chip Interface (CCI):Explains about Companion Chip Interface block

Cyclic Redundancy Check (CRC):Explains about Cyclic Redundancy Check block

Configurable Timers (CT):Explains about Configurable Timers block

eFUSE:Explains about EFUSE block

Enhanced General Purpose Input Output (EGPIO):Explains about Enhanced GPIO block

IR Decoder:Explains about IR decoder.

Clocks (CLK):Explains about M4SS CLOCK block

Pulse Width Modulator (PWM):Explains about Pulse Width Modulation block

Quadrature Encoder Interface (QEI) :Explains about Quadrature Encoder block

Quad Serial Peripheral Interface Flash(QSPI):Explains about Quad Serial Peripheral Interface block

Random Number Generator (RNG):Explains about Random Number Generator block

General Purpose Direct Memory Access (GPDMA):Explains about Direct Memory Access block

Serial Input Output (SIO):Explains of Serial GPIO block

Temperature Sensor: Explains about Temperature Sensor

Timer :Explains about TIMER block

Micro Direct Memory Access (µDMA) :Explains about Micro Direct Memory Access block

ULP Subsystem (ULPSS) Clock :Explains about ULPSS CLOCK block

Windowed Watchdog Timer (WWDT) :Explains about Window Watch Dog Timer block

Power Management Unit(PMU): Explains about power save block.

Comparator: Explains about comparator block.

PUF: Explains about PUF block.

Crypto : Explains about hardware security accelerator such as AES,ECDH,Exponentaition,HMAC-SHA and SHA.

Analog to Digital Converter : Explains about analog to digital converter block.

Digital to Analog Converter : Explains about digital to analog converter block.

Wake-Fi Receive : Explains about wake-fi receiver block.

Voice Activity Detection(VAD) : Explains about voice activity detection block.

Capacitive Touch Sensor : Explains about capacitive touch sensor block.

Deep sleep timer : Explains about deep sleep timer block.

NPSS GPIOs : Explains about NPSS GPIOs

2 Quickstart with Redpine MCU SAPIs

2.1 Redpine MCU SAPIs Examples in Keil IDE

2.1.1 Starting a multi-project view

Multiple examples are organized into multi-projects groups for UV5. To select a project group, double-click the appropriate multi-project file in the Keil examples directory . Once a multi-project file is started, all the examples associated with that project will be shown in the Project view. Refer to the image below:

Figure 2 Multi-project View

2.1.2 Select project

For selecting required project example, right click **"Project: EGPIO"** and set as active.

Figure 3 Select Project

Once you select the example, debug the project.

2.2 Redpine MCU SAPIs Examples in IAR IDE

2.2.1 Starting a multi-project view

Multiple examples are organized into multi-projects groups for Embedded Workbench 7.3 . To select a project group, double-click the appropriate multi-project file in the IAR examples directory . Once a multi-project file is started, all the examples associated with that project will be shown in the Project view. Refer to the image below:

Figure 4 Multi-project View

Select project

For selecting required project example, right click **"Project:CCI"** and set as active.

Figure 5 Select Project

Once you select the example, debug the project.

3 Building a project

3.1 Building a new project in Keil iDE

This section explains how to create a new project using Keil uVision5 for Redpine supported MCUs.

3.1.1 Start new project

Create a new project in Keil uVision5 IDE and select target in **"Redpine Signals"**.(E.g In 'RS1xy00 series' select the 'RS1xy00_1MB' for 1MB flash or 'RS1xy00_4MB' for 4MB flash)

Figure 6 Create New Project

3.1.2 Add source files

For adding a source file, right click **"New Group"** and click **"Add New Items to Group 'New Group' ".** Similarly, you can add existing files to the same group by clicking **"Add Existing Files to Group 'New Group' ".** Refer to the image below:

Path for Redpine proprietary peripheral source files :
Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Path for Redpine proprietary peripheral source files :
Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_library\systemlevel\src

Path for CMSIS supported peripheral source files: **Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_library\cmsis**

Path for Redpine MCU SAPIs Examples: **Redpine_MCU_Vx.y.z\Examples\Host_MCU\Peripheral_Examples**

Path for Redpine MCU Startup Files: **Redpine_MCU_Vx.y.z\Host_MCU\common\chip\src**

Figure 7 Add Source files

3.1.3 Include paths

In order to include file, click c/c++ tab. You may add header file path in include path.

Path for Redpine proprietary peripheral include files:
Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc

Path for Redpine proprietary peripheral include files:
Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc

Path for Redpine proprietary peripheral include files:
Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\inc

Path for CMSIS supported peripheral include files: **Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\cmsis**

Path for CMSIS supported peripheral include files: **Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\cmsis\config**

Path for CMSIS supported peripheral include files:
Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\cmsis\config\CMSIS\Driver\Include

Path for CMSIS supported peripheral include files: **Redpine_MCU_Vx.y.z\Host_MCU\Common\cmsis**

Path for CMSIS supported peripheral include files : **Redpine_MCU_Vx.y.z\Host_MCU\Common\chip\inc**

Path for CMSIS supported peripheral include files : **Redpine_MCU_Vx.y.z\Host_MCU\Common\board\inc**

Path for Redpine MCU Startup include files:

Figure 8 Add Include Paths

3.1.4 Changing the default optimization level

All library and application projects for Keil are configured to build at full optimization.

Using the level of optimization (such as level 3 (-O3), typically makes the code harder to debug, but gives the best possible code size and performance. You can change the optimization settings by opening the project options by right-clicking the project in the project browser window and selecting Options. After this, select the C/C++ tab and select the optimization level you want for your build images. Using an optimization level of (Level 0 (-O0)) will give a better debug experience, but a larger image.

Figure 9 Default Optimization Level

Note :

This screenshot has been captured by using Keil uVision5.18.0.0. If version is changed, screenshot may also change.

3.2 Building a new project in Keil iDE with Keil SVD Pack

3.2.1 Start new project

Install "Keil.RedpineMCU_DFP.1.0.0 version" of Keil SVD pack and create a new project.

3.2.2 Select Device

Select "Redpine MCU" device. Refer to the image below:

Figure 10 Select Device

3.2.3 Selecting and building the library file for the examples

All examples depend on a library file. Click Manage Run-Time Environment tab and select the library. Also, select the dependent file as well as the related RSI proprietary and CMSIS driver file. Refer to the image below:

Figure 11 Add Library Files

Note :

This screenshot has been captured by using Keil uVision5.18.0.0. If version is changed, screenshot may also change.

3.3 Building a Project in IAR Embedded workbench

3.3.1 Start new project

Open IAR Embedded Workbench 7.3 and select "Redpine MCU" and the package of the MCU product you use.

3.3.2 Select device

Select Redpine MCU device .

Search for the exact device in the "Target" table and select the same.

Figure 12 Select Device

3.3.3 Add source files

In order to add source files, right click the group. You can add source file ,main.c file or groups to it. Refer to the image below:

Figure 13 Add Source File

3.3.4 Include paths

On left navigation pane click c/c++ tab as shown in the figure below. After this, click Preprocessor tab. Under this tab, you can add include file directories as well as macros.

Figure 14 Add Includes Paths

Note:

This screenshot has been captured by using Embedded Workbench 7.3. If version is changed, screenshot may also change.

3.4 Custom Board

The board.c file enabled to supports Redpine Smart EVK LED and UART prints.If customer designs his own custom board then he need to modify pin associated to LED and UART.

4 Toolchain and Board Porting Guide

The Redpine MCU currently supports Keil, IAR and GCC tool chains. Refer to the Redpine MCU Porting Guide on Redpine's document portal for instructions on running with tool chains other than IAR, Keil and GCC.and porting the Redpine MCU SAPIs on different boards.

4.1 MCU Tool-chain Setup

4.1.1 Prerequisites

- Windows based PC with at least 4Gb of RAM and 5Gb of free space.
- Java 8 Update 171 or later. Download it for free from official Java support page.

> **Note**
>
> Eclipse requires a 64-bit Java for 64-bit machine and 32-bit Java for 32-bit machine. Even if it is perfectly possible to use a 32-bit JVM on a 64-bitmachine.

4.1.2 Get Tool-chain Package

- Download the tool-chain package as per the working environment from https://github.com/redpinesignalsinc/WiSeMCUIDE.

For Windows 32-bit ----------> RS1xy00_Toolchain_v0.5_Win32.zip
For Windows 64-bit ----------> RS1xy00_Toolchain_v0.5_Win64.zip

- Extract the contents of the package 'RS1xy00_Toolchain_v0.5_Win32.zip' to any local drive say, 'D:\'.At the end of the process you will find the folder 'D:\RS1xy00_Toolchain' containing the complete tool-chain.

> **Note**
>
> Following steps are based on 32-bit package.

4.1.3 Create Example Project :

- To create new Project, Right-click in the **'Project Explorer'** window. Select **'New'->'C Project'**.

Create new project

- Enter **'Project name'** and select **'Redpine RS1xy00 C/C++ Project'** in the **"Project Type"** section. Select **'Cross ARM GCC'** in the **"Toolchains"** section and click **'Next >'**.

Select the required Redpine device

- To change default directory Uncheck **'Use default location'.**
- Enter **'Location'**, **'Project name'** and select **'Redpine RS1xy00 C/C++ Project'** in the **"Project Type"** section.
- Select **'Cross ARM GCC'** in the **"Toolchains"** section and click **'Next >'.**

Select the cross compiler

- Select the check-box **'Peripheral Examples'** for Peripheral Projects and click **'Next >'**.

Select the peripheral examples

- Select the check-box **'Wireless Examples'** only when using Wireless Projects. At this time Uncheck **'Peripheral Examples'.**

Select the wireless examples

> **Note**
>
> It is not recommended to select both check-boxes at a time (Peripheral Examples / Wireless Examples). Only one category should be selected at a time.

- Click **'Next >'.**

- Click **'Next >'**

- Update GCC path. GCC is available at **'D:\RS1xy00_Toolchain\ gcc-arm\bin'** and click **'Finish'.**

- Default Project template shows selected category folder under Example folder. A sample **'Blinky'** project is created by default, as shown below,

4.1.4 Project Compilation

This section details steps for configurations to compile and build.

- To reduce compilation time. Right-click on the project folder and select **'Properties'.**

- Select **'C/C++ Build'** in the left pane. Click **'Behavior'** tab, check **'Enable parallel build'** and click **'Use optimal jobs'**. Click **'Apply'** and **'OK'**.

- Update **'Build Tools'** path in the project settings. Right-click on the project folder and select **'Properties'.**

- Select **'C/C++ Build'** ->**'Environment'** in the left pane. Add **'Build Tools'** path (**D:\RS1xy00_Toolchain\Build Tools\2.6-201507152002\bin**) to **'PATH'** variable under **'Environment variables to set'** by selecting **'PATH'** and clicking on **'Edit...'** option.

Note

No need to modify CWD and PWD (Get updated with project workspace).

- Add **'Build Tools'** path in system environment variables.

  1.From the desktop, right click the **Computer** icon.

  2.Choose **Properties** from the context menu.

  3.Click the **Advanced system settings** link.

  4.Click **Environment Variables** In the section **System Variables**, find the PATH environment variable and select it. Click Edit.

  If the PATH environment variable does not exist, click New.

  5.In the **Edit System Variable** (or **New System Variable**) window, specify the value of the PATH environment variable (D:\RS1XY00_Toolchain\Build Tools\2.6-201507152002\bin) as shown below , Click **OK**.

- Select **'C/C++ Build'** ->**'Environment'** in the left pane.
- Click on Add to add **'OPENOCD_PATH'.**

- Enter **'Name'** and **'Value'** fields as shown below, check **'Add to all configurations'** option and click **'OK'.**

  1.**Name: OPENOCD_PATH**

  2.**Value : D:\RS1xy00_Toolchain\openocd-win32\share\openocd\scripts**

- Click **'OK'**

- Click **'Apply'.**
- Select **'C/C++ Build'** -> **'Settings'**. In **'Build Steps'** tab add following command in the **'Post-build steps'** option. Click **'OK'**.
  arm-none-eabi-objcopy -O binary ${ProjName}.elf $(OPENOCD_PATH)/RS1XY00.bin

- Right-click on the project folder and click **'Build project'** to compile the project.

- Following logs will be displayed in the Console after successful compilation.

5.Configuring OpenOCD

OpenOCD is used for Flash Programming and Debugging.

- Click on **Run >Debug Configurations.**

- Double-click on **'GDB OpenOCD Debugging'** to create new launch as shown below.

- Fill **'Name'** under **'Main'** tab. Browse and select the project in the **'Project:'** option. And select corresponding **'.elf'** file in **'C/C++ Application:'**

  using **'Search Project'**. Click **'Apply'**.

- Under **'Debugger'** tab,In OpenOCD setup, check **'Start OpenOCD locally'** option.
- Browse and select openocd executable **'openocd.exe'** available at **'D:\RS1xy00_Toolchain\openocd-win32\bin\openocd.exe'** in the 'Executable:' field.
- Leave **'GDB port:'** and **'Telnet port:'** with default values, as shown below.
- In 'Config options:' field, add **"-f board/rs1xy00.cfg".**
- In GDB Client Setup browse and select gdb executable **'arm-none-eabi-gdb.exe'** path (available at **D:\RS1xy00_Toolchain\gcc-arm\bin\arm-none-eabi-gdb.exe**)in the 'Executable:' field.
- Click **'Apply'.**

- Under **'Startup'** and **'Common'** tabs, select configurations as shown below. Click **'Apply'** and **'OK'**

- Click **'Apply'** and **'Close'.**

4.1.5 6.Steps to Flash and Debug

- Select **'Blinky'** under debug icon as shown below,

- After successful connection with MCU, following prints would be displayed and flash programming would be started. Progress would be displayed at the right bottom of the eclipse IDE.

- Following prints would be displayed in the openocd console, after the flash programming is completed. OpenOCD prints “wrote xxxx bytes from file RS1xy00.bin in xxxx sec”.

- Eclipse prompts while switching to debug perspective. Select **‘Remember my decision’** and click **‘Yes’**.

- Debug session would be launched as shown below, and control points at main().

- Click on 'Resume' icon (or F8) to see LED blinking.

5 Keil Middleware

Overview
This section explains the Keil Middleware Version 7.5.0.

USB
It supports the keil USB Middleware components along with CMSIS USB driver. For more information, refer to the link : http://www.keil.com/pack/doc/mw/USB/html/index.html

File System
It supports the keil File system Middleware components along with CMSIS MCI driver. For more information refer to the link : http://www.keil.com/pack/doc/mw/FileSystem/html/index.html

Network stack
It supports the keil network stack Middleware components along with CMSIS ETHERNET driver. For more information refer to the link : http://www.keil.com/pack/doc/mw/Network/html/index.html

6 Examples

	Example	Description	Path
1	Active_HP	This example demonstrates measuring the power when core is running at higher clock.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
2	Active_ULP	This example demonstrates measuring the power when core is running at 20MHZ clock in PS2	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
3	ADC	This example print the equivalent ADC input voltage from obtained from ADC samples.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
4	ADC 1Ksps sampling rate	In this example core is going to sleep with ADC as wake up source here ADC sampling rate set 1k sample per second.Measured power consumption for ADC 1ksps sampling rate.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
5	ADC 10Ksps sampling rate	In this example core is going to sleep with ADC as wake up source here ADC sampling rate set 10k sample per second.Measured power consumption for ADC 10ksps sampling rate.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
6	ADC 100Ksps sampling rate	In this example core is going to sleep with ADC as wake up source here ADC sampling rate set 100k sample per second.Measured power consumption for ADC 100ksps sampling rate.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
7	ADC 1Msps sampling rate	In this example core is going to sleep with ADC as wake up source here ADC sampling rate set 1M sample per second.Measured power consumption for ADC 1Msps sampling rate.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
8	Battery Status	This example read the battery status of chip.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
9	BJT temperature sensor	This example demonstrates about bjt temperature sensor update the temperature at every second and prints on hyperterminal.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
10	BJT temperature sensor power save	This example demonstrates about temperature sensor in ps4 sleep with ram retention.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
11	Blackout Monitor	It explains how to configure the Blackout Monitor in low power mode (i.e. If the voltage is less than BOD threshold then BOD enables the blackout by which chip reset will be given at low power mode and chip get reset when the voltage is less than the reset voltage)	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
12	Blinky	This example will toggles the GPIO 10times per second.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
13	Brownout Detection	It explains how to configure BOD block, BOD interrupt will be raised when the voltage is less than the BOD threshold	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\

	Example	Description	Path
		level.	
14	Button	It explains how to configure the Button block where we can configure three button ranges (i.e. three voltage ranges) .Button interrupt will serve when the voltage is in the range of three buttons even we can read at which the button range interrupt triggered(i.e at first button range or second button range or third button range of voltage).	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
15	CAN Example	This example demonstrates the data transmission with other CAN enabled device without hardware filter configuration.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
16	CCI_Master	This CCI(Companion Chip Interface)example shows how to use the CCI to access the second chip memory.This application configures CCI master mode in the chip1.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
17	CCI_Slave	This example configures CCI slave mode in the chip2 to access memory from chip1(master) to chip2(slave).	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
18	Comparator	This example demonstrates comparator functionality here comparator compare the external input voltages.Depends on input voltage for comparator interrupt is occurs. Non-inverting input supply > Inverting input supply : comparator interrupt is occurs.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
19	CRC		Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
20	CT Example	This example demonstrates Configurable Timers Counter operations using external events.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
21	CTS	This example demonstrates about touch sensor which generates interrupt whenever touch happens on one of the sensor out of eight sensor.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
22	CTS_Touch	This example demonstrates about touch sensor in ps2 sleep with ram retention.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
23	DAC	Apply the standard sine wave samples to DAC input register by using UDMA , and observe the DAC output on ULP_GPIO4.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
24	Deep Sleep	To measure the deep sleep current (I.e PS0 state)	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
25	Deepsleep_with_dc_dc_on	To measure the deep sleep current (I.e PS0 state) with DC DC is on	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
26	EFUSE Example	Efuse example writes some bit locations in efuse controller and read those locations.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
27	Ethernet	This example sends arp packets through	Redpine_MCU_Vx.y.z\Host_MCU\Examples\

	Example	Description	Path
		ethernet cable.	Peripheral_Examples\
28	FIM	On chip FIM block will perform matrix multiplication operation between two matrix's.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
29	FreeRTOS	This example demonstrates the keil RTOS(RTX) functionality,here create the 3 task and toggle respective TriLED for each task serving form core.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
30	GPDMA Example	This example demonstrates DMA API usage for memory to memory transfer.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
31	GSPI	This example transmits data on MOSI pin and receives same data on MISO pin (loopback tranfer)	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
32	I2C Example	This example demonstrates I2C read and write with EEPROM Slave.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
33	I2S	This example demonstrates I2S master mode configuration and TX RX operations in loopback.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
34	PWM Example	This example demonstrates PWM output signal generation with varying duty cycle for with required frequency.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
35	IR_Decoder	This example demonstrates the decoding IR signal pattern and read the receiving data pattern.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
36	M4_group_interrupt	This example configures two inputs for group interrupt(GROUP INTR1 , GROUP INTR2) generation,it configures these two pins in edge(high polarity) with Logical OR event,interrupt will be generated for every rise edge on two input pins when Logical OR satisfies.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
37	M4_pin_interrupt	This example will configures a pin to contribute for pin interrupt with level low event enable.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
38	OPAMP	This example demonstrate OPAMP unity gain buffer functionality.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
39	PS2_Retention	This example measure the power when memory are retain in PS2.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
40	PS2_Sleep	This example demonstrate PS2 sleep and wakeup where RTC configures as a wake up source for M4.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
41	PS4_Retention	This example measure the power when memory are retain in PS4.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
42	PS4_Sleep	This example demonstrate PS4 sleep and wakeup where RTC configures as a wake up source for M4.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
43	Ps4_sleep_with_retntion_flash	This example demonstrates M4 sleep and wake up functionality with RAM retention and code execute with flash.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\

	Example	Description	Path
44	Ps4_sleep_with_retntion_flash	This example demonstrates M4 sleep and wake up functionality without RAM retention and code execute with flash.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
45	PWM	This example generate the PWM signals on 1L and 2L channels with a step of 1 duty cycle from 0 to 100.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
46	QEI	This example calculate the QEI velocity and get the direction of motor.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
47	RNG	This example generate the random number with true random number generator.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
48	RO_Temp_Sensor	This example calculate and update the temperature at every second and print on hyper terminal.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
49	RTC		Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
50	SDIOH	The SDIO host example shows how to write and read data in SDIO slave using smih interface.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
51	SIO	This example demonstrates data transfer with SIO_SPI loop back, here SIO_SPI Master instance is configured with 16 bit data width and 1Mhz speed and hardware loop back is connected to send 1k data.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
52	soc_pll_freq_measurement	This example measure the configured SOC PLL frequency.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
53	SPI Example	This example demonstrates data transfer with SPI loop back mode, here SPI Master instance is configured with 16 bit data width and 10Mhz speed and hardware loop back is connected to send 1k data.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
54	SPI_SDC	This example demonstrates M4 sleep and wake up in PS2 state with SPI(ULPSS based wake up) as wake up source for M4.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
55	Timer Example	This example demonstrates Timer initialization and blink led with different time delay.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
56	UART	The example shows how to use the UART loopback mode.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
57	UDMA Example	This example demonstrates DMA API usage for memory to memory transfer.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
58	ulp_group_interrupt	This example configures two input pins for group interrupt generation,it configures these two pins in edge(high polarity) with Logical OR event,interrupt will be generated for every rise edge on two input pins when Logical OR satisfies.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
59	ulp_pin_interrupt	This example will configures a pin to contribute for pin interrupt with level low event enable.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\

	Example	Description	Path
60	usart_master	This example is for USART master configuration,it will sends and receives data in fullduplex mode.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
61	usart_slave	This example is for USART slave configuration,it will sends and receives data in full duplex mode(when master generates clock).	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
62	wakeup_interrupt	This example configures these two pins in edge(high polarity) with Logical OR event,interrupt will be generated for every rise edge on two input pins when Logical OR satisfies,it gives an interrupt any one of the input pin matches its status with the configured polarity.Wakeup interrupts can be detected even when clock is not there for egpio module.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\
63	WDT	The WWDT example demonstrates the WDT warning trigger interrupt handler.The watchdog generates a warning WWDT interrupt and then restarts the WWDT on the warning (the LED0 will toggle on each warning interval cycle).After 10 warning interrupts the re-start request is stopped to generates a watchdog reset.	Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\

7 Brown Out Detection

7.1 Overview

This section explains how to configure and use the BOD using Redpine MCU SAPIs.

7.2 Programming Sequence

```
#include "rsi_chip.h"  /* Redpine_MCU_Vx.y.z\Host_MCU\common\chip\inc */
#include "rsi_bod.h"   /* Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\inc
*/
#include "rsi_board.h" /* Redpine_MCU_Vx.y.z\Host_MCU\common\board\inc */

#define  AUTOMATIC
//#define MANUAL
/* Mode selection */
#define PMU_GOOD_TIME                    31    /*Duration in us*/
#define XTAL_GOOD_TIME                   31    /*Duration in us*/
#define  MANUAL_MODE                     0
#define AUTOMATIC_MODE                   1
#define ENABLE                           1
#define DISABLE                          0
#define BOD_THRESHOLD                    3.35
#define SLOT_VALUE                       2
#define NVIC_BOD                         NPSS_TO_MCU_BOD_INTR_IRQn
#define NVIC_BUTTON                      NPSS_TO_MCU_BUTTON_INTR_IRQn
#define  NPSS_TO_MCU_BUTTON_IRQHandler   IRQ024_Handler
#define  NPSS_TO_MCU_BOD_IRQHandler      IRQ023_Handler


volatile float vbatt_status =0;

void NPSS_TO_MCU_BOD_IRQHandler(void)
{
    DEBUGOUT("Your Vbatt status is less than the threshold voltage i.e%fV
\n",RSI_BOD_SoftTriggerGetBatteryStatus());
    RSI_BOD_IntrClr();
}


int main(void)
{
    volatile static uint32_t t_value =0,button_number=0 ;
    /*Configures the system default clock and power configurations*/
     SystemCoreClockUpdate();
    #ifdef DEBUG_UART
    /*Init debug UART*/
     DEBUGINIT();
    #endif

    #ifdef AUTOMATIC
    /*BOD enable and set the threshold value */
    RSI_BOD_Enable(ENABLE,BOD_THRESHOLD) ;

    /* set the slot value */
    RSI_BOD_ConfigSlotValue(SLOT_VALUE) ;

    /*set the BOD to automatic  mode*/
    RSI_BOD_SetMode(AUTOMATIC_MODE) ;
    /*enable bod interrupt */
    RSI_BOD_IntrEnable(ENABLE) ;

    RSI_BOD_BodTestSel(1,3);
    /*Gets the threshold value */

    /* BOD NVIC enable  */
    NVIC_EnableIRQ(NVIC_BOD);
```

```
    t_value = RSI_BOD_GetThreshold();

    #endif

   #ifdef MANUAL
   RSI_BOD_BodTestSel(1,3);
   /*BOD enable and set the threshold value */
   RSI_BOD_Enable(ENABLE,BOD_THRESHOLD) ;
   /*set the BOD to automatic  mode*/
   RSI_BOD_SetMode(MANUAL_MODE) ;
         /*enable bod interrupt */
   RSI_BOD_IntrEnable(ENABLE) ;
     /* BOD NVIC enable  */
   NVIC_EnableIRQ(NVIC_BOD);

  #endif
   while(1)
   {
      __WFI();
   }
}
```

Code Block 1 Brown Out Detection example

7.3 API Descriptions

7.4 RSI_BOD_SoftTriggerGetBatteryStatus

Source File : rsi_bod.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\src\

Prototype

```
float RSI_BOD_SoftTriggerGetBatteryStatus(void)
```

Description

This API is used to to get the battery level status

Return values

Battery level

Example

```
float battery_status;
/*To get the battery level status*/
battery_status = RSI_BOD_SoftTriggerGetBatteryStatus();
```

7.5 RSI_BOD_Enable

Source File : rsi_bod.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\src\

Prototype

```
void RSI_BOD_Enable(uint8_t enable,float vbatt_threshold)
```

Description

This API is used to to enable or disable the BOD and to set threshold value.

Parameters

Parameter	Description
enable	1: enables the BOD. 0: disables the BOD.
vbatt_threshold	Set's the threshold value

Return values

None

Example

```
/*BOD enable and set the threshold value */
RSI_BOD_Enable(ENABLE,BOD_THRESHOLD) ;
```

7.6 RSI_BOD_SetMode

Source File : rsi_bod.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\src\

Prototype

```
error_t RSI_BOD_SetMode(uint8_t mode)
```

Description

This API is used to set the BOD mode

Parameter

Parameter	Description
mode	1: enable automatic mode 0: enable manual mode

Return values

Zero on success and error code on failure

Example

```
#define  AUTOMATIC_MODE  1
/*set the BOD to automatic  mode*/
 RSI_BOD_SetMode(AUTOMATIC_MODE) ;
```

7.7 RSI_BOD_GetThreshold

Source File : rsi_bod.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\src\

Prototype

```
uint32_t RSI_BOD_GetThreshold(void)
```

Description

This API is used to get the threshold value

Return values

threshold value

Example

```
uint32_t t_value;
/*Get the threshold value*/
t_value = RSI_BOD_GetThreshold();
```

7.8 RSI_BOD_ConfigSlotValue

Source File : rsi_bod.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\src\

Prototype

```
error_t RSI_BOD_ConfigSlotValue(uint16_t slot_value)
```

Description

This API is used to configure the slot value for automatic BOD

Parameter

Parameter	Description
slot_value	Slot value after which comparator outputs are sampled in auto mode. The lowest possible values for this parameter 1. The highest possible values for this parameter 32767.

Return values

Zero on success and error code on failure

Example

```
/* set the slot value */
RSI_BOD_ConfigSlotValue(SLOT_VALUE) ;
```

7.9 RSI_BOD_ButtonWakeUpEnable

Source File : rsi_bod.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\src\

Prototype

```
void RSI_BOD_ButtonWakeUpEnable(uint8_t enable)
```

Description

This API is used to enable or disable Button wake up

Parameter

Parameter	Description
enable	1: enables the button. 0: disables the button.

Return values

None

Example

```
/*Enable Button wake up */
RSI_BOD_ButtonWakeUpEnable(ENABLE);
```

7.10 RSI_BOD_Buttonvalue

Source File : rsi_bod.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\src\

Prototype

```
uint32_t RSI_BOD_Buttonvalue(void)
```

Description

This API is used to get the Button value.

Return values

Returns the button value

Example

```
uint32_t button_status_value;
/*Get the  Button value */
button_status_value = RSI_BOD_Buttonvalue();
```

7.11 RSI_BOD_BlackOutReset

Source File : rsi_bod.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\src\

Prototype

```
void RSI_BOD_BlackOutReset(uint16_t enable)
```

Description

This API is used to enable/disable the black out reset of BOD.

Parameter

Parameter	Description
enable	1: enables black out reset . 0: disables black out reset.

Return values

None

Example

```
/*Enable  black out reset  */
RSI_BOD_BlackOutReset(ENABLE);
```

7.12 RSI_BOD_BGSampleEnable

Source File : rsi_bod.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\src\

Prototype

```
void RSI_BOD_BGSampleEnable(void)
```

Description

This API is used to enable the blackout reset in sleep mode.

Return values

None

Example

```
/*enable  blackout reset in sleep mode.. */
RSI_BOD_BGSampleEnable();
```

7.13 RSI_BOD_BGSampleDisable

Source File : rsi_bod.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\src\

Prototype

```
void RSI_BOD_BGSampleDisable(void)
```

Description

This API is used to disable the blackout reset in sleep mode.

Return values

None

Example

```
 /*disable  blackout reset in sleep mode. */
RSI_BOD_BGSampleDisable();
```

7.14 RSI_BOD_IntrEnable

Source File : rsi_bod.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\src\

Prototype

```
void RSI_BOD_IntrEnable(uint16_t enable)
```

Description

This API is used to enable the BOD interrupt.

Return values

None

Example

```
/*enable the BOD interrupt. */
RSI_BOD_IntrEnable(ENABLE);
```

7.15 RSI_BOD_ButtonIntrEnable

Source File : rsi_bod.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\src\

Prototype

```
void RSI_BOD_ButtonIntrEnable(uint16_t enable)
```

Description

This API is used to enable the button interrupt.

Return values

None

Example

```
/*enable's the button interrupt. */
RSI_BOD_ButtonIntrEnable(ENABLE);
```

7.16 RSI_BOD_IntrClr

Source File : rsi_bod.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\src\

Prototype

```
void RSI_BOD_IntrClr(void)
```

Description

This API is used to clear the BOD interrupt.

Return values

None

Example

```
/*Clear the BOD interrupt */
RSI_BOD_IntrClr(ENABLE);
```

7.17 RSI_BOD_ButtonIntrClr

Source File : rsi_bod.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\src\

Prototype

```
void RSI_BOD_ButtonIntrClr(void)
```

Description

This API is used to clear the button interrupt.

Return values

None

Example

```
/* Clear the button interrupt. */
RSI_BOD_ButtonIntrClr(ENABLE);
```

7.18 RSI_BOD_GetIntrStatus

Source File : rsi_bod.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\src\

Prototype

```
uint32_t RSI_BOD_GetIntrStatus(void)
```

Description

This API is used to get status of the BOD interrupt.

Return values

Interrupt status

Example

```
uint32_t interrupt_status;
/*get status of the BOD interrupt.. */
interrupt_status = RSI_BOD_GetIntrStatus();
```

8 CMSIS Support

8.1 Overview

This section explains all the CMSIS supported Repine MCU peripherals such as CAN,Ethernet,I2C,MCI,SAI,SPI,USART,USB and CMSIS DSP .

For more information, refer to the link **: http://www.keil.com/pack/doc/CMSIS/General/html/index.html**

8.2 CMSIS Core

It supports all CMSIS-Core features.

Refer to the link :**http://www.keil.com/pack/doc/CMSIS/Core/html/index.htmlhttp://www.keil.com/pack/doc/CMSIS/Core_A/html/index.html**

8.3 CMSIS Driver

8.3.1 GSPI

It supports all CMSIS SPI driver features.

Refer to the link :**https://www.keil.com/pack/doc/CMSIS/Driver/html/group__spi__interface__gr.htmlhttp://www.keil.com/pack/doc/CMSIS/Driver/html/group__can__interface__gr.html**

8.3.2 CAN

It supports all CMSIS CAN driver features.

Refer to the link : **http//www.keil.com/pack/doc/CMSIS/Driver/html/group__can__interface__gr.html**

8.3.3 Ethernet

It supports all CMSIS Ethernet driver features.

Refer to the link : **http//www.keil.com/pack/doc/CMSIS/Driver/html/group__eth__interface__gr.html**

8.3.4 I2C

It supports almost all the CMSIS I2C driver features except the below features. (Refer to the link : **http//www.keil.com/pack/doc/CMSIS/Driver/html/group__i2c__interface__gr.html**).

1. ARM_I2C_EVENT_BUS_ERROR
2. ARM_I2C_EVENT_BUS_CLEAR
3. ARM_I2C_BUS_SPEED_FAST_PLUS

8.3.5 MCI

It supports all CMSIS MCI driver features.

Refer to the link : **http//www.keil.com/pack/doc/CMSIS/Driver/html/group__mci__interface__gr.html**

8.3.6 SAI

It supports almost all the CMSIS SAI driver features except the below features. (Refer to the link : **http//www.keil.com/pack/doc/CMSIS/Driver/html/group__sai__interface__gr.html**)

1. AC'97 protocol
2. Mono mode

3. Companding

4. MCLK (Master Clock) pin

5. Frame error event (ARM_SAI_EVENT_FRAME_ERROR)

8.3.7 SPI

It supports almost all the CMSIS SAI driver features except the below features. (Refer to the link : **http//www.keil.com/pack/doc/CMSIS/Driver/html/group__spi__interface__gr.html**)

CMSIS SPI driver API is implemented in a way to support master and slave configurations for separate hardware instances. Redpine MCU hardware supports two master instances and one slave instance i.e. SPI1 represents master configuration, SPI2 represents slave configuration and SPI3 represents ULP master cofiguration.

1. SPI1 & SPI3 only support Master mode, if configured as slave returns ARM_SPI_ERROR_MODE
2. SPI2 only supports Slave mode, if configured as slave returns ARM_SPI_ERROR_MODE.
3. SPI does not support Simplex mode in any instance, if configured as slave returns ARM_SPI_ERROR_MODE.
4. BIT order MSB first only supported, if configured as any other returns ARM_SPI_ERROR_BIT_ORDER.

8.3.8 USART

It supports almost all the CMSIS USART driver features except the below features. (Refer to the link : http//www.keil.com/pack/doc/CMSIS/Driver/html/group__usart__interface__gr.html)

1. UART Smart Card mode

2. Smart Card Clock generator

8.3.9 USB

It supports all CMSIS USB driver features.

Refer to the link : **http//www.keil.com/pack/doc/CMSIS/Driver/html/group__usb__interface__gr.html**

8.4 CMSIS DSP

8.4.1 FIM

The features which support Redpine MCU FIM apart from CMSIS DSP feature are outlined below.

1. Scalar Add(real-cmplx, cmplx-real ,cmplx-cmplx)
2. Scalar Multiplication(real-cmplx, cmplx-real ,cmplx-cmplx)
3. Scalar Subtraction(for all types of data)
4. Vector Add(real-cmplx, cmplx-real ,cmplx-cmplx)
5. Vector Subtraction(real-cmplx, cmplx-real ,cmplx-cmplx)
6. Vector Multiplication(real-cmplx)
7. Squaring of real numbers
8. IIR filtering operation

The following features supported by FIM and which are CMSIS compatible.

1. Scalar Addition for real inputs
2. Scalar Multiplication for real inputs
3. Vector Addition for real inputs
4. Vector Subtraction for real inputs
5. Vector Multiplication for real inputs, complex-complex, complex - real inputs
6. Matrix multiplication with real inputs

7. FIR for real-real inputs, real inputs complex coefficients, real coefficients and complex inputs, complex inputs and coefficients
8. FIR Interpolation for real inputs

Output in case of addition, subtraction would be half of the actual output, but incase of filters,multiplication the output would the desired one.

Floating point API's i.e. f32 format are not one-to-one compatible. They require hex inputs instead of float inputs. The input and the obtained output will be of 8.23 (8 bits integral values and 23 decimal values) format.

Example:

int32_t srcAF32[5] = {0xCCCCC, 0x266666, 0xFFC00000, 0x691687, 0xFF828F5D};

int32_t srcBF32[5] = {0x333333, 0xFFF33334, 0x400000, 0x528F5C, 0xFFE51EB9};

Equivalent value for 0.1 in 0xCCCCC. To obtain the value multiply (0.1)*(2^23).

arm_sub_f32_opt(srcA , srcB , pDst ,blockSize**)** ;

Apart from above mentioned list, there are limitations.

Refer to the link : **http//www.keil.com/pack/doc/CMSIS/DSP/html/index.html**

9 IR Decoder

9.1 Overview

IR decoder is used to decode IR signals. This section explains how to configure and use the IR decoder using Redpine MCU SAPIs.

9.2 Programming Sequence

```
#include "rsi_chip.h"    /* Redpine_MCU_Vx.y.z\Host_MCU\common\chip\inc\ */
#include "rsi_ir.h"      /* Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\ */
#define TEST_PIN_IR_INPUT          7U
#define TEST_PIN_IR_OUTPUT         5U

/**
 * @brief  IR decoder interrupt handler, its clear the pending interrupt.
 * @param  None
 * @retval None
 */
void IRQ015_Handler(void)
{
  /*Clear IRQ*/
  NVIC_ClearPendingIRQ(NVIC_IR);
  ir_intr=1;
  flag = 0;
  return ;
}

/**
 * @brief  Systic handler to emulate the IR pattern.
 * @param  None
 * @retval None
 */
void SysTick_Handler(void)
{
    static uint8_t toggle=0;
    /*Emulates IR patterns*/
    if(flag < 10)
    {
        toggle = !toggle;
    RSI_EGPIO_SetPin(EGPIO, 0, SYSTICK_PIN, toggle);
    }
    flag++;
    return;
}

/**
 * @brief  Enable the systic pin for input to IR.
 * @param  None
 * @retval None
 */
void Gpio_Pin_Mux(void)
{
    RSI_EGPIO_PadSelectionEnable(1);
    RSI_EGPIO_SetPinMux(EGPIO , 0, SYSTICK_PIN  , EGPIO_PIN_MUX_MODE0);
    RSI_EGPIO_SetDir(EGPIO , 0, SYSTICK_PIN  ,EGPIO_CONFIG_DIR_OUTPUT);
}

/**
 * @brief  Main program.
 * @param  None
 * @retval None
 */
int main(void)
{
    uint16_t i=0;
```

```
/*Configures the system default clock and power configurations*/
 SystemCoreClockUpdate();

/*Clock enable */
RSI_IPMU_ClockMuxSel(1);

RSI_PS_FsmLfClkSel(KHZ_RC_CLK_SEL);

RSI_PS_FsmHfClkSel(FSM_32MHZ_RC);

RSI_CLK_PeripheralClkEnable1(M4CLK , ULPSS_CLK_ENABLE);

/* Configure the ULP-IR clock */
RSI_ULPSS_UlpPeriClkEnable(ULPCLK,IR_PCLK_EN);

/* Set the pin mux for systic pattern generation pin */
Gpio_Pin_Mux();

/*ULP pad receive enable*/
RSI_EGPIO_PadSelectionEnable(18);

RSI_EGPIO_UlpPadReceiverEnable(TEST_PIN_IR_INPUT);

RSI_EGPIO_UlpPadReceiverEnable(TEST_PIN_IR_OUTPUT);

/*GPIO Muxing enable */
RSI_EGPIO_SetPinMux(EGPIO1 , 0, TEST_PIN_IR_INPUT  , TEST_PIN_IR_INPUT_MODE);

RSI_EGPIO_SetPinMux(EGPIO1 , 0, TEST_PIN_IR_OUTPUT , TEST_PIN_IR_OUTPUT_MODE);

/*configure the IR off time */
RSI_IR_OffDuration(IR , IR_OFF_DURATION);

/*configure the IR on time */
RSI_IR_OnDuration(IR  , IR_ON_DURATION);

/* define the frame done threshold value */
RSI_IR_Framedonethreshold(IR  , IR_FRAME_DONE_THRESHOLD);

/* define the detection  threshold value */
RSI_IR_Detectionthreshold(IR  , IR_DET_THRESHOLD);

/* set the bit of enable clock ir core */
RSI_IR_SetConfiguration(IR , CONFIG_EN_CLK_IR_CORE);

/* set the bit for continuous ir detection mode */
RSI_IR_SetConfiguration(IR , CONFIG_EN_CONT_IR_DET);

/*enable the IRQ of IR */
NVIC_EnableIRQ(NVIC_IR);

/* set the bit for ir detection mode */
RSI_IR_SetConfiguration(IR , CONFIG_EN_IR_DET);

/* Set the systic timer to generate the pattern */
SysTick_Config(SOC_OPER_FREQUENCY/ RSI_BLINK_RATE);

while(1)
{
    while(!ir_intr);
    ir_intr = 0;
    /* Wait for some time to read the data */
    i = 1000;
    while(i--);
    Memlocation_num=RSI_IR_GetMemoryDepth(IR);

   for(i= 0; i < Memlocation_num ; i++)
   {
    ir_data[i] = RSI_IR_ReadData(IR,i);
   }
    /* Restart the IR module */
```

```
        RSI_IR_SoftwareRestart(IR);
    }
   /*Application code never reach here just to satisfy the standard main*/
}
```

Code Block 2 IR Decoder example

9.3 API Descriptions

9.3.1 RSI_IR_OffDuration

Source File : rsi_ir.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE error_t RSI_IR_OffDuration(IR_Type* pIr , uint32_t off_duration)
```

Description

This API is used to configure the off duration of IR decoder.

Parameters

Parameter	Description
pIr	Pointer to the IR register instance
off_duration	IR Sleep duration timer value. Programmable value for OFF duration for power cycling on External IR Sensor.Count to be programmed write to clock ticks of 32KHz clock.Programmed value is (1/32K)*off_duration.

Return values

return zero RSI_OK on success and return error code on failure.

Example

```
/*set off time duration */
RSI_IR_OffDuration(IR , 20);
in the above parameter we get off time of (1/32K)*20 = 0.625ms
```

9.3.2 RSI_IR_OnDuration

Source File : rsi_ir.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE error_t RSI_IR_OnDuration(IR_Type* pIr , uint16_t on_duration)
```

Description

This API is used to configure the on duration of IR decoder.

Parameters

Parameter	Description

Parameter	Description
pIr	Pointer to the IR register instance
on_duration	IR Sleep duration timer value. Programmable value for ON duration for power cycling on External IR Sensor.Count to be programmed write to clock ticks of 32KHz clock.Programmed value is (1/32K)*off_duration.

Return values

return zero RSI_OK on success and return error code on failure.

Example

```
/*set on time duration */
RSI_IR_OnDuration(IR , 20);
in the above parameter we get on time of (1/32K)*20 = 0.625ms
```

9.3.3 RSI_IR_Framedonethreshold

Source File : rsi_ir.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE error_t RSI_IR_Framedonethreshold(IR_Type* pIr,uint16_t frame_threshold)
```

Description

This API is used count with respect to 32KHz clock after not more toggle are expected to a given pattern.

Parameters

Parameter	Description
pIr	Pointer to the IR register instance
frame_threshold	frame done threshold value.

Return values

return zero RSI_OK on success and return error code on failure.

Example

```
/*frame done check */
RSI_IR_Framedonethreshold(IR,20);
```

9.3.4 RSI_IR_Detectionthreshold

Source File : rsi_ir.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE error_t RSI_IR_Detectionthreshold(IR_Type* pIr,uint16_t detection_threshold)
```

Description

This API is used minimum number of edges to detected during on-time failing which IR detection is re-stated.

Parameters

Parameter	Description
pIr	Pointer to the IR register instance
detection_threshold	detection threshold value.

Return values

return zero RSI_OK on success and return error code on failure.

Example

```
/*frame detection check */
RSI_IR_Detectionthreshold(IR,20);
```

9.3.5 RSI_IR_SetConfiguration

Source File : rsi_ir.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_IR_SetConfiguration(IR_Type* pIr , uint32_t flags)
```

Description

This API is used set the configure the IR modes.

Parameters

Parameter	Description
pIr	Pointer to the IR register instance
flags	Ored values of IR configuration flags • CONFIG_SREST_IR_CORE • CONFIG_EN_CONT_IR_DET • CONFIG_EN_CONT_IR_DET • CONFIG_EN_CONT_IR_DET • CONFIG_EN_CONT_IR_DET

Return values

None

Example

```
/*set configuration */
 RSI_IR_SetConfiguration(IR , (CONFIG_SREST_IR_CORE | CONFIG_EN_CONT_IR_DET));
```

9.3.6 RSI_IR_GetMemoryDepth

Source File : rsi_ir.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE uint32_t RSI_IR_GetMemoryDepth(IR_Type* pIr)
```

Description

This API returns the IR data samples depth.

Parameter

Parameter	Description
pIr	Pointer to the IR register instance

Return values

number samples received.

Example

```
/* declaration */
uint32_t memory_depth;
/* memory depth*/
memory_depth = RSI_IR_GetMemoryDepth(IR);
```

9.3.7 RSI_IR_MemoryReadEnable

Source File : rsi_ir.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_IR_MemoryReadEnable(IR_Type* pIr)
```

Description

This API is used enable the memory read option.

Parameter

Parameter	Description
pIr	Pointer to the IR register instance

Return values

return zero RSI_OK on success and return error code on failure.

Example

```
/* memory read enable*/
RSI_IR_MemoryReadEnable(IR);
```

9.3.8 RSI_IR_ReadData

Source File : rsi_ir.c

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
uint16_t RSI_IR_ReadData(IR_Type* pIr,uint16_t memory_address)
```

Description

This API is used read IR address.

Parameter

Parameter	Description
pIr	Pointer to the IR register instance
memory_address	memory address value (0 .. 128).

Return values

16-Bit IR data recieved (BIT[15] in received data will indicate the polarity of pulse) remaining bit will contain the incremented counter value of the pulse.

Example

```
/* declaration */
uint16_t ir_data;
/* read data */
ir_data = RSI_IR_ReadData(IR,0);
```

9.3.9 RSI_IR_SoftwareRestart

Source File : rsi_ir.c

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
void RSI_IR_SoftwareRestart(IR_Type* pIr)
```

Description

This API software restart to IR operation.

Parameter

Parameter	Description
pIr	Pointer to the IR register instance

Return values

None

Example

```
/* Restart the IR module */
RSI_IR_SoftwareRestart(IR);
```

9.3.10 RSI_IR_ClrConfiguration

Source File : rsi_ir.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_IR_ClrConfiguration(IR_Type* pIr , uint32_t flags)
```

Description

This API is used clear configure the IR modes.

Parameters

Parameter	Description
pIr	Pointer to the IR register instance
flags	Ored values of IR configuration flags • CONFIG_SREST_IR_CORE • CONFIG_EN_CONT_IR_DET • CONFIG_EN_CONT_IR_DET • CONFIG_EN_CONT_IR_DET • CONFIG_EN_CONT_IR_DET

Return values

None

Example

```
/*clear configuration */
RSI_IR_ClrConfiguration(IR , (CONFIG_SREST_IR_CORE | CONFIG_EN_CONT_IR_DET));
```

9.3.11 RSI_IR_MemoryReadEnable

Source File : rsi_ir.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_IR_MemoryReadEnable(IR_Type* pIr)
```

Description

This API is used enable the memory read option.

Parameter

Parameter	Description
pIr	Pointer to the IR register instance

Return values

return zero RSI_OK on success and return error code on failure.

Example

```
/* memory read enable*/
RSI_IR_MemoryReadEnable(IR);
```

10 Temperature Sensor

10.1 RO Temperature Sensor

10.2 Overview

This section explains how to configure and use the Temperature Sensor using Redpine MCU SAPIs.

10.3 Programming Sequence

```
#include "rsi_temp_sensor.h" /*
Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\inc */
#include "base_types.h"
#include "rsi_chip.h"        /* Redpine_MCU_Vx.y.z\Host_MCU\common\chip\inc */
#include "rsi_board.h"       /* Redpine_MCU_Vx.y.z\Host_MCU\common\board\inc */
#include<math.h>

#define SIZE         15

#define NPSS_RTC_IRQ_Handler    IRQ029_Handler
#define NVIC_RTC MCU_CAL_RTC_IRQn

#define NOMI_TEMPERATURE     25
#define MSB_BIT              31

RTC_TIME_CONFIG_T rtcConfig;
uint32_t Temp,temp_read,f1,f2,f2_count;
int32_t Temp_negative;

char binary[SIZE + 1], onesComp[SIZE + 1], twosComp[SIZE + 1];
int16_t decimal_val,base=1;

/*two's complement conversion function*/
void temp_conversion(uint32_t temp)
{
    uint8_t i=0,rem,carry=1;
    uint8_t num=0;
    decimal_val=0,base=1;
    while(temp>0)
    {
        binary[i] = temp % 2;
        temp = temp / 2;
        i++;
    }

    for(i=0;i<MSB_BIT;i++)
    {
        if(binary[i]==0)
            onesComp[i]=1;
        else
            onesComp[i]=0;
    }

    for(i=0; i<MSB_BIT; i++)
    {
        if(onesComp[i] == 1&& (carry == 1))
        {
            twosComp[i] = 0;
        }
        else if((onesComp[i] == 0) && (carry == 1))
        {
            twosComp[i] = 1;
            carry = 0;
        }
        else
```

```
        {
            twosComp[i] = onesComp[i];
        }
    }

    for(i=0; i<MSB_BIT; i++)
    {
        num=twosComp[i];
        rem = num % 10;
        decimal_val = decimal_val + rem * base;
        base = base * 2;
    }
}

void NPSS_RTC_IRQ_Handler(void)
{
    volatile uint32_t statusRead = 0;

    statusRead = RSI_RTC_GetIntrStatus();

    if(statusRead & NPSS_TO_MCU_SEC_INTR){
        RSI_RTC_IntrClear (NPSS_TO_MCU_SEC_INTR);
    }
    /*read the temperature*/
    Temp=RSI_TS_ReadTemp(MCU_TEMP);

    rtcConfig.DayOfWeek        = Saturday;
    rtcConfig.MilliSeconds     = 999;
    rtcConfig.Century          = 0;
    rtcConfig.Day              = 18;
    rtcConfig.Hour             = 2;
    rtcConfig.Minute           = 20;
    rtcConfig.Month            = January;
    rtcConfig.Second           = 0 ;
    rtcConfig.Year             = 16;

    RSI_RTC_SetDateTime(RTC ,&rtcConfig);

    if(Temp & BIT(10))
    {
        /*read the f2 count*/
        f2=RSI_TS_GetPtatClkCnt(MCU_TEMP);
        /*substract f2_nominal count from f2_count NOTE:configure the f2_nominal
          temp_read=(f2-f2_nominal)*/
        temp_read=(f2-RSI_IPMU_RO_TsEfuse());
        if(temp_read & BIT(31))
        {
            temp_conversion(temp_read);
            /*calculate the negative temperature*/
            Temp_negative=(25-(decimal_val * 1.3437));
#ifdef DEBUG_UART
            DEBUGOUT("temp_read=%d   ",Temp_negative);
#endif
        }
    }
    else{
#ifdef DEBUG_UART
        DEBUGOUT("temp_read=%d  ",Temp);
#endif
    }
    return ;
}

int main()
{
    uint32_t cntr;
    SystemCoreClockUpdate();
    #ifdef DEBUG_UART
    DEBUGINIT();
#endif
```

```
    /*Enable clock sources*/
    RSI_IPMU_ClockMuxSel(1);
    RSI_PS_FsmLfClkSel(KHZ_RC_CLK_SEL);
    RSI_PS_FsmHfClkSel(FSM_32MHZ_RC);
    /*configure the slope,nominal temperature and f2_nominal*/
    RSI_TS_Config(MCU_TEMP,NOMI_TEMPERATURE);
    /*disable the bjt based temp sensor*/
    RSI_TS_RoBjtEnable(MCU_TEMP,0);
    /*Enable the RO based temp sensor*/
    RSI_TS_Enable(MCU_TEMP,1);
    /*update the temperature periodically*/
    RSI_Periodic_TempUpdate(TIME_PERIOD,1,0);
    /*read the reference clk count*/
    f1=RSI_TS_GetRefClkCnt(MCU_TEMP);
    /*read the ptat clk count*/
    f2=RSI_TS_GetPtatClkCnt(MCU_TEMP);

    /*Init RTC*/
    RSI_RTC_Init(RTC);


    /*RTC configuration with some default time */
    rtcConfig.DayOfWeek        = Saturday;
    rtcConfig.MilliSeconds     = 999;
    rtcConfig.Century          = 0;
    rtcConfig.Day              = 18;
    rtcConfig.Hour             = 2;
    rtcConfig.Minute           = 20;
    rtcConfig.Month            = January;
    rtcConfig.Second           = 0 ;
    rtcConfig.Year             = 16;

    /*start RTC */
    RSI_RTC_Start(RTC);
    /*Set the RTC configuration*/
    RSI_RTC_SetDateTime(RTC ,&rtcConfig);
    /*Enable RTC second based interrupt*/
    RSI_RTC_IntrUnMask( RTC_SEC_INTR );
    /*Start RTC */
    RSI_RTC_Start(RTC);
    /*Enable NVIC for second based interrupts */
    NVIC_EnableIRQ(NVIC_RTC);

    while(1){
        /*Wait for interrupt */
        __WFI();
    }

    return 0;
}
```

Code Block 3 RO Temperature Sensor Example

```
#include "rsi_chip.h"  /* Redpine_MCU_Vx.y.z\Host_MCU\common\chip\inc */
#include "rsi_adc.h"   /* Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc */
#include "rsi_board.h" /* Redpine_MCU_Vx.y.z\Host_MCU\common\board\inc */
#include "rsi_opamp.h" /* Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc */

#include "UDMA.h"
#include "rsi_ipmu.h"

extern RSI_DRIVER_UDMA Driver_UDMA1;
static RSI_DRIVER_UDMA *UDMAdrv1 = &Driver_UDMA1;
extern RSI_UDMA_HANDLE_T udmaHandle1;

#define ADC_CALIB   1
```

```
volatile uint32_t done,ret;

#define  BUFFER_SIZE    3
#define  NPSS_GPIO_PIN   3
uint16_t rx_buffer[BUFFER_SIZE];

#define  UDMA1_IRQHandler  IRQ010_Handler
#define  ADC_IRQ_Handler   IRQ011_Handler

#define NPSS_RTC_IRQ_Handler    IRQ029_Handler
#define NVIC_RTC MCU_CAL_RTC_IRQn

RTC_TIME_CONFIG_T rtcConfig;

uint8_t rtc_done=0;

void NPSS_RTC_IRQ_Handler(void)
{
    volatile uint32_t statusRead = 0;

    statusRead = RSI_RTC_GetIntrStatus();

    if(statusRead & NPSS_TO_MCU_SEC_INTR){
        RSI_RTC_IntrClear (NPSS_TO_MCU_SEC_INTR);
    }

  RSI_RTC_SetDateTime(RTC ,&rtcConfig);

  rtc_done=1;

    return;
}

opamp_config UnityGB =
{
    {
     /*opamp1_dyn_en;*/         0,
     /*opamp1_sel_p_mux;*/      5,
        /*opamp1_sel_n_mux;*/     4,
     /*opamp1_lp_mode;*/        0,
     /*opamp1_r1_sel;*/         1,
     /*opamp1_r2_sel;*/         0,
     /*opamp1_en_res_bank;*/    0,
     /*opamp1_res_mux_sel;*/    0,
     /*opamp1_res_to_out_vdd;*/ 0,
     /*opamp1_out_mux_en;*/     1,
   /*opamp1_out_mux_sel;*/    0,
     /*opamp1_enable;*/         1
    },
    {
        /*   opamp2_dyn_en;*/      0,
        /*      opamp2_sel_p_mux;*/   0,
        /*     opamp2_sel_n_mux;*/   0,
        /*     opamp2_lp_mode;*/     0,
        /*     opamp2_r1_sel;*/      0,
        /*     opamp2_r2_sel;*/      0,
        /*     opamp2_en_res_bank;*/ 0,
        /*     opamp2_res_mux_sel;*/ 0,
        /*     opamp2_res_to_out_vdd;*/ 0,
        /*     opamp2_out_mux_en;*/     0,
        /*     opamp2_enable;*/         0,

    },
    {
        /* opamp3_dyn_en;*/          0,
        /* opamp3_sel_p_mux;*/       0,
        /* opamp3_sel_n_mux;*/       0,
        /* opamp3_lp_mode;*/         0,
        /* opamp3_r1_sel;*/          0,
        /* opamp3_r2_sel;*/          0,
        /* opamp3_en_res_bank;*/     0,
```

```
        /* opamp3_res_mux_sel;*/      0,
        /* opamp3_res_to_out_vdd;*/   0,
        /* opamp3_out_mux_en;*/       0,
        /* opamp3_enable;*/           0,
    }
};

RSI_ADC_CALLBACK_T vsADCCallBack;
uint32_t adc_done=0;


void udmaTransferComplete(uint32_t event, uint8_t ch)
{
  if(event == UDMA_EVENT_XFER_DONE)
  {
    if(ch == 11)
    {
      done = 1;
    }
  }
}

void UDMA_Write()
{
    RSI_UDMA_CHA_CONFIG_DATA_T control;
  RSI_UDMA_CHA_CFG_T config;


    memset(&control, 0, sizeof(RSI_UDMA_CHA_CONFIG_DATA_T));
    memset(&config, 0, sizeof(RSI_UDMA_CHA_CFG_T));

    config.altStruct = 0;
  config.burstReq  = 1;
  config.channelPrioHigh = 0;
  config.periAck = 0;
  config.periphReq = 0;
  config.reqMask = 0;
  config.dmaCh     = 11;

    /* Setup source to desination copy for trigger for memory */
    /* Descriptor configurations */
    control.transferType       = UDMA_MODE_AUTO;
    control.nextBurst                     = 0;
    control.totalNumOfDMATrans = BUFFER_SIZE -1;
    control.rPower             = ARBSIZE_2;
    control.srcProtCtrl              = 0x000;
    control.dstProtCtrl              = 0x000;
    control.srcSize            = SRC_SIZE_16;
    control.srcInc             = SRC_INC_NONE;
    control.dstSize            = DST_SIZE_16;
    control.dstInc             = DST_INC_16;

   /* Initialise dma */
   UDMAdrv1->Initialize();


     /* Configure dma channel */
   UDMAdrv1->ChannelConfigure( 11,(uint32_t )0x24043814, (uint32_t )rx_buffer,
BUFFER_SIZE,
                              control, &config, udmaTransferComplete );
   /* Enable dma channel */
   UDMAdrv1->ChannelEnable(11);
}


void ADC_IRQ_Handler(void)
{
    RSI_ADC_InterruptHandler(AUX_ADC_DAC_COMP,&vsADCCallBack);
  adc_done=1;
}
```

```
uint32_t i=0,adc_off=0,Vbg=0;
int32_t Voffset=0;
float Temp;
uint16_t adc2_out[1]={0},adc_out[]={0};

uint16_t read_data;

void bjt_temp_sensor()
{
    RSI_ADC_ClkDivfactor(AUX_ADC_DAC_COMP,48,64);

    RSI_Opamp1_Config(OPAMP,0,&UnityGB);

    RSI_ADC_Config(AUX_ADC_DAC_COMP,0,1,3,0);

    RSI_ADC_StaticMode(AUX_ADC_DAC_COMP,20,0,0);

    RSI_ADC_ChannelSamplingRate(AUX_ADC_DAC_COMP,0,0,2);

    RSI_ADC_Start(AUX_ADC_DAC_COMP);

    RSI_ADC_FifoMode(AUX_ADC_DAC_COMP,0,20,0,0);

        while(adc_done);

    UDMA_Write();

    UDMAdrv1->DMAEnable();

    RSI_UDMA_ChannelSoftwareTrigger(udmaHandle1,11);

    while(!done);

    done=0;

    adc_out[i]=rx_buffer[i];

    if(adc_out[i] & BIT(11) )
    {
      adc_out[i] = ((adc_out[i] ) & 0xF7FF);
    }else{
      adc_out[i] = ( adc_out[i] | BIT(11) );
    }

    RSI_ADC_Stop(AUX_ADC_DAC_COMP);

    RSI_ADC_Config(AUX_ADC_DAC_COMP,0,1,3,0);

    RSI_ADC_TempSensorEnable(AUX_ADC_DAC_COMP);

    RSI_ADC_StaticMode(AUX_ADC_DAC_COMP,23,0,0);

    RSI_ADC_Start(AUX_ADC_DAC_COMP);

    RSI_ADC_FifoMode(AUX_ADC_DAC_COMP,0,23,0,0);

    while(adc_done);

    UDMA_Write();

    UDMAdrv1->DMAEnable();

    RSI_UDMA_ChannelSoftwareTrigger(udmaHandle1,11);

    while(!done);

    done=0;

    adc2_out[i]=rx_buffer[i];

    if(adc2_out[i] & BIT(11) )
```

```
    {
      adc2_out[i] = ((adc2_out[i] ) & 0xF7FF);
    }else{
      adc2_out[i] = ( adc2_out[i] | BIT(11) );
    }

    RSI_ADC_Stop(AUX_ADC_DAC_COMP);

        adc_off=RSI_IPMU_Auxadcoff_SeEfuse();

        Vbg=RSI_IPMU_Vbg_Tsbjt_Efuse();

        Voffset=(961-RSI_IPMU_Delvbe_Tsbjt_Efuse());

       Temp=-273+(310*((((adc2_out[i]-adc_off)/(float)(adc_out[i]-
adc_off))*(Vbg/(float)1000))+((Voffset)/(float)1000)));
#ifdef DEBUG_UART
        DEBUGOUT("Temp=%f",Temp);
#endif
}

/*set bg sampling voltage*/
void RSI_BG_VlgProg()
{
    *(volatile uint32_t *)(0x2405A50C)   = 0x50E80010;

    *(volatile uint32_t *)(0x2405A4A4) |= BIT(4);

}

int main(void)
{
    /*Configures the system default clock and power configurations*/
    SystemCoreClockUpdate();

  RSI_BG_VlgProg();

#ifdef DEBUG_UART
    DEBUGINIT();
#endif
    RSI_ADC_PowerControl(ADC_POWER_ON);

  RSI_ULPSS_AuxClkConfig(ULPCLK, ENABLE_STATIC_CLK ,ULP_AUX_32MHZ_RC_CLK);

  ULP_SPI_MEM_MAP(BG_SCDC_PROG_REG_1)|=BIT(3);

  *(volatile uint32_t *)(0x24043A10)   = 0x5b;

  RSI_ADC_ClkDivfactor(AUX_ADC_DAC_COMP,0,1);

    #if ADC_CALIB
  RSI_ADC_Calibration();
    #endif
    /*Init RTC*/
    RSI_RTC_Init(RTC);

    /*RTC configuration with some default time */
    rtcConfig.DayOfWeek        = Saturday;
    rtcConfig.MilliSeconds     = 999;
    rtcConfig.Century          = 0;
    rtcConfig.Day              = 18;
    rtcConfig.Hour             = 2;
    rtcConfig.Minute           = 20;
    rtcConfig.Month            = January;
    rtcConfig.Second           = 0 ;
    rtcConfig.Year             = 16;

    /*start RTC */
    RSI_RTC_Start(RTC);
    /*Set the RTC configuration*/
    RSI_RTC_SetDateTime(RTC ,&rtcConfig);
```

```
    /*Enable RTC second based interrupt*/
    RSI_RTC_IntrUnMask( RTC_SEC_INTR );
  /*Start RTC */
    RSI_RTC_Start(RTC);

    NVIC_EnableIRQ(ADC_IRQn);

    NVIC_EnableIRQ(UDMA1_IRQn);

    /*Enable NVIC for second based interrupts */
    NVIC_EnableIRQ(NVIC_RTC);

  while(1)
    {
      bjt_temp_sensor();

    while(!rtc_done);

        rtc_done=0;
    }
}
```

Code Block 4 BJT Temperature sensor example

10.4 API Descriptions

RSI_TS_Enable

Source File : rsi_temp_sensor.c

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\src\

Prototype

```
void RSI_TS_Enable(NPSS_TEMPSENSOR_Type  *pstcTempSens , boolean_t bEn)
```

Description

This API is used to enable / disable the temperature sensor.

Parameters

Parameter	Description
pstcTempSens	Pointer to the temperature sensor register instance
bEn	enable / disable parameter • 0: Disable • 1: Enable

Return values

None

Example

```
/* temperature sensor enable*/
RSI_TS_RefClkSel(MCU_TEMP,1);
```

RSI_TS_Config

Source File : rsi_temp_sensor.c

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\src\

Prototype

```
void RSI_TS_Config(NPSS_TEMPSENSOR_Type  *pstcTempSens,uint32_t u32Nomial)
```

Description

This API is used to set the slope of the temperature sensor.

Parameters

Parameter	Description
pstcTempSens	Pointer to the temperature sensor register instance
u32Nomial	nominal temperature value

Return values

None

Example

```
uint32_t nominal=25;

/* configure TS structure*/
RSI_TS_Config(MCU_TEMP,nominal);
```

RSI_TS_SetCntFreez

Source File : rsi_temp_sensor.c

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\src\

Prototype

```
void RSI_TS_SetCntFreez(NPSS_TEMPSENSOR_Type  *pstcTempSens , uint32_t u32CountFreez)
```

Description

This API is used to set the count of reference clock on which ptat clock counts.

Parameters

Parameter	Description
pstcTempSens	Pointer to the temperature sensor register instance
u32CountFreez	count of reference clock on which ptat clock counts

Return values

None

Example

```
uint32_t count_freez=250;

/* set count of reference clock */
RSI_TS_SetCntFreez(MCU_TEMP,count_freez);
```

RSI_TS_RefClkSel

Source File : rsi_temp_sensor.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\src\

Prototype

```
void RSI_TS_RefClkSel(NPSS_TEMPSENSOR_Type  *pstcTempSens , boolean_t bRefClk)
```

Description

This API is used to select the reference clock to the temperature sensor.

Parameters

Parameter	Description
pstcTempSens	Pointer to the temperature sensor register instance
bRefClk	reference clock selection • 0: reference RO clock from analog • 1: MCU FSM clock

Return values

None

Example

```
/* select the reference clock for RO clock */
RSI_TS_RefClkSel(MCU_TEMP,0);

/* select the reference clock for FSM clock */
RSI_TS_RefClkSel(MCU_TEMP,1);
```

RSI_TS_ReadTemp

Source File : rsi_temp_sensor.c

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\src\

Prototype

```
uint32_t RSI_TS_ReadTemp(NPSS_TEMPSENSOR_Type  *pstcTempSens)
```

Description

This API is used to read the temperature value.

Parameter

Parameter	Description
pstcTempSens	Pointer to the temperature sensor register instance

Return values

returns the temperature value.

Example

```
uint32_t readtemp;
/* read temperature sensor value */
readtemp=RSI_TS_ReadTemp(MCU_TEMP);
```

RSI_TS_RoBjtEnable

Source File : rsi_temp_sensor.c

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\src\

Prototype

```
void RSI_TS_RoBjtEnable(NPSS_TEMPSENSOR_Type  *pstcTempSens , boolean_t  enable)
```

Description

This API is used to enable ROBJT.

Parameters

Parameter	Description
pstcTempSens	Pointer to the temperature sensor register instance
enable	enable or disable bjt based temp sensor • 0: Disable • 1: Enable

Return values

returns the temperature value.

Example

```
/* RO BJT enable */
RSI_TS_RoBjtEnable(MCU_TEMP,1);
/* RO BJT disable */
RSI_TS_RoBjtEnable(MCU_TEMP,0);
```

RSI_TS_LoadBjt

Source File : rsi_temp_sensor.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\src\

Prototype

```
void RSI_TS_LoadBjt(NPSS_TEMPSENSOR_Type *pstcTempSens , uint32_t temp)
```

Description

This API is used to temperature sensor in BJT mode.

Parameters

Parameter	Description
pstcTempSens	Pointer to the temperature sensor register instance
temp	known temprature

Return values

None

Example

```
uint32_t temp=100;
/* TS in BJT mode */
RSI_TS_LoadBjt(MCU_TEMP,temp);
```

RSI_TS_GetRefClkCnt

Source File : rsi_temp_sensor.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\src\

Prototype

```
uint32_t RSI_TS_GetRefClkCnt(NPSS_TEMPSENSOR_Type  *pstcTempSens)
```

Description

This API is used to read the reference clock count.

Parameter

Parameter	Description
pstcTempSens	Pointer to the temperature sensor register instance

Return values

returns the reference clock count

Example

```
uint32_t refcount;
/* get reference count */
refcount=RSI_TS_GetRefClkCnt(MCU_TEMP);
```

RSI_TS_GetPtatClkCnt

Source File : rsi_temp_sensor.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\src\

Prototype

```
uint32_t RSI_TS_GetPtatClkCnt(NPSS_TEMPSENSOR_Type  *pstcTempSens)
```

Description

This API is used to read the ptat clock count.

Parameter

Parameter	Description
pstcTempSens	Pointer to the temperature sensor register instance

Return values

returns the ptat clock count.

Example

```
uint32_t refcount;
/* get reference count */
patcount=RSI_TS_RSI_TS_GetPtatClkCnt(MCU_TEMP);
```

RSI_Periodic_TempUpdate

Source File : rsi_temp_sensor.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\src\

Prototype

```
void RSI_Periodic_TempUpdate(TimePeriodCalib_t *temp,uint8_t enable,uint8_t trigger_time)
```

Description

This API is used to update the temperature periodically.

Parameter

Parameter	Description
temp	Pointer to the timeperiod register instance
enable	enable periodic checking of temp
trigger_time	periodic check time in sec • 0 for 1 sec • 1 for 2sec • 2 for 4 sec • 3 for 5 sec temperature update

Return values

None

Example

```
RSI_Periodic_TempUpdate(TIME_PERIOD,1,0);
```

11 ULP Subsystem (ULPSS) Clock

11.1 Overview

This section explains how to configure and use the ULP Subsystem Clock using Redpine MCU SAPIs.

11.2 Programming Sequence

```
#include "rsi_chip.h" /* Redpine_MCU_Vx.y.z\Host_MCU\common\chip\inc */

/* define specific peripheral MACRO for demonstrate that specific peripheral example */
int main()
{
 #ifdef ULPSSPROCESSOR
    RSI_CLK_XtalClkConfig(2);
    RSI_ULPSS_UlpProcClkConfig(ULPCLK,ULP_PROC_ULP_32KHZ_XTAL_CLK,1,&Delay);
    RSI_ULPSS_UlpProcClkConfig(ULPCLK,ULP_PROC_ULP_32MHZ_RC_CLK,2,&Delay);
    RSI_ULPSS_UlpProcClkConfig(ULPCLK,ULP_PROC_ULP_20MHZ_RO_CLK,2,&Delay);
    RSI_ULPSS_ClockConfig(M4CLK,Enable,0,0);
    RSI_ULPSS_UlpProcClkConfig(ULPCLK,ULP_PROC_SOC_CLK,2,&Delay);
    RSI_ULPSS_UlpProcClkConfig(ULPCLK,ULP_PROC_ULP_DOUBLER_CLK,4,&Delay);
    RSI_ULPSS_UlpProcClkConfig(ULPCLK,ULP_PROC_REF_CLK,2,&Delay);
    RSI_ULPSS_UlpProcClkConfig(ULPCLK,ULP_PROC_ULP_32KHZ_RO_CLK,1,&Delay);
    RSI_ULPSS_UlpProcClkConfig(ULPCLK,ULP_PROC_ULP_32KHZ_RC_CLK,1,&Delay);
#endif

#ifdef PHERIPHERALENABLE
    RSI_ULPSS_PeripheralEnable(ULPCLK, ULP_I2C_CLK,ENABLE_STATIC_CLK);
    RSI_ULPSS_PeripheralEnable(ULPCLK, ULP_EGPIO_CLK,ENABLE_STATIC_CLK);
    RSI_ULPSS_PeripheralEnable(ULPCLK, ULP_AUX_CLK,ENABLE_STATIC_CLK);
    RSI_ULPSS_PeripheralEnable(ULPCLK, ULP_FIM_CLK,ENABLE_STATIC_CLK);
    RSI_ULPSS_PeripheralEnable(ULPCLK, ULP_VAD_CLK,ENABLE_STATIC_CLK);
    RSI_ULPSS_PeripheralEnable(ULPCLK, ULP_TIMER_CLK,ENABLE_STATIC_CLK);
    RSI_ULPSS_PeripheralEnable(ULPCLK, ULP_UDMA_CLK,ENABLE_STATIC_CLK);
    RSI_ULPSS_PeripheralEnable(ULPCLK, ULP_TOUCH_CLK,ENABLE_STATIC_CLK);
    RSI_ULPSS_PeripheralEnable(ULPCLK, ULP_UART_CLK,ENABLE_STATIC_CLK);
    RSI_ULPSS_PeripheralEnable(ULPCLK, ULP_SSI_CLK,ENABLE_STATIC_CLK);
    RSI_ULPSS_PeripheralEnable(ULPCLK, ULP_I2S_CLK,ENABLE_STATIC_CLK);
#endif

#ifdef AUXADC
    RSI_CLK_XtalClkConfig(2);
    RSI_ULPSS_AuxClkConfig(ULPCLK,ENABLE_STATIC_CLK,ULP_AUX_32KHZ_XTAL_CLK);
    RSI_ULPSS_AuxClkConfig(ULPCLK,ENABLE_STATIC_CLK,ULP_AUX_32MHZ_RC_CLK);
    RSI_ULPSS_AuxClkConfig(ULPCLK,ENABLE_STATIC_CLK,ULP_AUX_20MHZ_RO_CLK);
    RSI_ULPSS_ClockConfig(M4CLK,Enable,0,0);
    RSI_ULPSS_AuxClkConfig(ULPCLK,ENABLE_STATIC_CLK,ULP_AUX_ULP_SOC_CLK);
    RSI_ULPSS_AuxClkConfig(ULPCLK,ENABLE_STATIC_CLK,ULP_AUX_ULP_DOUBLER_CLK);
    RSI_ULPSS_AuxClkConfig(ULPCLK,ENABLE_STATIC_CLK,ULP_AUX_REF_CLK);
    RSI_ULPSS_AuxClkConfig(ULPCLK,ENABLE_STATIC_CLK,ULP_AUX_I2S_PLL_CLK);
    RSI_ULPSS_AuxClkConfig(ULPCLK,ENABLE_STATIC_CLK,ULP_AUX_32KHZ_RC_CLK);
    RSI_ULPSS_AuxClkConfig(ULPCLK,ENABLE_STATIC_CLK,ULP_AUX_32KHZ_RO_CLK);
#endif

#ifdef TIMERULP
    RSI_CLK_XtalClkConfig(2);
    RSI_ULPSS_TimerClkConfig(ULPCLK,ENABLE_STATIC_CLK,1,ULP_TIMER_32KHZ_XTAL_CLK,1);
    RSI_ULPSS_TimerClkConfig(ULPCLK,ENABLE_STATIC_CLK,1,ULP_TIMER_32MHZ_RC_CLK,1);
    RSI_ULPSS_TimerClkConfig(ULPCLK,ENABLE_STATIC_CLK,1,ULP_TIMER_20MHZ_RO_CLK,1);
    RSI_ULPSS_ClockConfig(M4CLK,Enable,0,0);
    RSI_ULPSS_TimerClkConfig(ULPCLK,ENABLE_STATIC_CLK,1,ULP_TIMER_ULP_SOC_CLK,1);
    RSI_ULPSS_TimerClkConfig(ULPCLK,ENABLE_STATIC_CLK,1,ULP_TIMER_REF_CLK,1);
    RSI_ULPSS_TimerClkConfig(ULPCLK,ENABLE_STATIC_CLK,1,ULP_TIMER_32KHZ_RO_CLK,1);
    RSI_ULPSS_TimerClkConfig(ULPCLK,ENABLE_STATIC_CLK,1,ULP_TIMER_32KHZ_RC_CLK,1);
#endif

#ifdef VADULPCLK
```

```
    RSI_CLK_XtalClkConfig(2);
    RSI_ULPSS_VadClkConfig(ULPCLK ,ULP_VAD_32KHZ_XTAL_CLK  , ULP_VAD_ULP_PROCESSOR_CLK,2);
    RSI_ULPSS_VadClkConfig(ULPCLK ,ULP_VAD_32KHZ_RC_CLK  , ULP_VAD_REF_CLK,1);
    RSI_ULPSS_VadClkConfig(ULPCLK ,ULP_VAD_32KHZ_RC_CLK  , ULP_VAD_32MHZ_RC_CLK,3);
    RSI_ULPSS_VadClkConfig(ULPCLK ,ULP_VAD_32KHZ_XTAL_CLK  , ULP_VAD_20MHZ_RO_CLK,2);
    RSI_ULPSS_ClockConfig(M4CLK,Enable,0,0);
    RSI_ULPSS_VadClkConfig(ULPCLK ,ULP_VAD_32KHZ_RO_CLK  , ULP_VAD_ULP_SOC_CLK,1);
    RSI_ULPSS_VadClkConfig(ULPCLK ,ULP_VAD_32KHZ_RO_CLK  , ULP_VAD_ULP_PROCESSOR_CLK,1);
#endif

#ifdef TOUCHULPCLK
    RSI_CLK_XtalClkConfig(2);
    RSI_ULPSS_TouchClkConfig(ULPCLK,ULP_TOUCH_32KHZ_XTAL_CLK,2);
    RSI_ULPSS_TouchClkConfig(ULPCLK,ULP_TOUCH_32MHZ_RC_CLK,3);
    RSI_ULPSS_TouchClkConfig(ULPCLK,ULP_TOUCH_20MHZ_RO_CLK,4);
    RSI_ULPSS_ClockConfig(M4CLK,Enable,0,0);
    RSI_ULPSS_TouchClkConfig(ULPCLK,ULP_TOUCH_ULP_SOC_CLK,1);
    RSI_ULPSS_TouchClkConfig(ULPCLK,ULP_TOUCH_REF_CLK,3);
    RSI_ULPSS_TouchClkConfig(ULPCLK,ULP_TOUCH_32KHZ_RO_CLK,1);
    RSI_ULPSS_TouchClkConfig(ULPCLK,ULP_TOUCH_32KHZ_RC_CLK,1);
#endif

#ifdef SSIULPCLK
   RSI_CLK_XtalClkConfig(2);
   RSI_ULPSS_UlpSsiClkConfig(ULPCLK,ENABLE_STATIC_CLK,ULP_SSI_ULP_32KHZ_XTAL_CLK,1);
   RSI_ULPSS_UlpSsiClkConfig(ULPCLK,ENABLE_STATIC_CLK,ULP_SSI_ULP_32MHZ_RC_CLK,2);
   RSI_ULPSS_UlpSsiClkConfig(ULPCLK,ENABLE_STATIC_CLK,ULP_SSI_ULP_20MHZ_RO_CLK,3);
   RSI_ULPSS_ClockConfig(M4CLK,Enable,0,0);
   RSI_ULPSS_UlpSsiClkConfig(ULPCLK,ENABLE_STATIC_CLK,ULP_SSI_SOC_CLK,4);
   RSI_ULPSS_UlpSsiClkConfig(ULPCLK,ENABLE_STATIC_CLK,ULP_SSI_REF_CLK,2);
   RSI_ULPSS_UlpSsiClkConfig(ULPCLK,ENABLE_STATIC_CLK,ULP_SSI_ULP_32KHZ_RO_CLK,1);
   RSI_ULPSS_UlpSsiClkConfig(ULPCLK,ENABLE_STATIC_CLK,ULP_SSI_ULP_32KHZ_RC_CLK,1);
#endif

#ifdef I2SULPCLK
   RSI_CLK_XtalClkConfig(2);
   RSI_ULPSS_UlpI2sClkConfig(ULPCLK  ,ULP_I2S_ULP_32KHZ_XTAL_CLK,4);
   RSI_ULPSS_UlpI2sClkConfig(ULPCLK  ,ULP_I2S_ULP_32MHZ_RC_CLK,3);
   RSI_ULPSS_UlpI2sClkConfig(ULPCLK  ,ULP_I2S_ULP_20MHZ_RO_CLK,4);
   RSI_ULPSS_ClockConfig(M4CLK,Enable,0,0);
   RSI_ULPSS_UlpI2sClkConfig(ULPCLK  ,ULP_I2S_SOC_CLK,3);
   RSI_ULPSS_UlpI2sClkConfig(ULPCLK  ,ULP_I2S_ULP_DOUBLER_CLK,2);
   RSI_ULPSS_UlpI2sClkConfig(ULPCLK  ,ULP_I2S_PLL_CLK,4);
   RSI_ULPSS_UlpI2sClkConfig(ULPCLK  ,ULP_I2S_REF_CLK,2);
   RSI_ULPSS_UlpI2sClkConfig(ULPCLK  ,ULP_I2S_ULP_32KHZ_RO_CLK,1);
   RSI_ULPSS_UlpI2sClkConfig(ULPCLK  ,ULP_I2S_ULP_32KHZ_RC_CLK,1);
#endif

#ifdef UARTULPCLK
    RSI_CLK_XtalClkConfig(2);
    RSI_ULPSS_UlpUartClkConfig(ULPCLK,ENABLE_STATIC_CLK,1,ULP_UART_ULP_32KHZ_XTAL_CLK,2);
    RSI_ULPSS_UlpUartClkConfig(ULPCLK,ENABLE_STATIC_CLK,1,ULP_UART_ULP_32MHZ_RC_CLK,4);
    RSI_ULPSS_UlpUartClkConfig(ULPCLK,ENABLE_STATIC_CLK,1,ULP_UART_ULP_20MHZ_RO_CLK,3);
    RSI_ULPSS_ClockConfig(M4CLK,Enable,0,0);
    RSI_ULPSS_UlpUartClkConfig(ULPCLK,ENABLE_STATIC_CLK,1,ULP_UART_SOC_CLK,2);
    RSI_ULPSS_UlpUartClkConfig(ULPCLK,ENABLE_STATIC_CLK,1,ULP_UART_ULP_DOUBLER_CLK,2);
    RSI_ULPSS_UlpUartClkConfig(ULPCLK,ENABLE_STATIC_CLK,1,ULP_UART_REF_CLK,3);
    RSI_ULPSS_UlpUartClkConfig(ULPCLK,ENABLE_STATIC_CLK,1,ULP_UART_ULP_32KHZ_RO_CLK,1);
    RSI_ULPSS_UlpUartClkConfig(ULPCLK,ENABLE_STATIC_CLK,1,ULP_UART_ULP_32KHZ_RC_CLK,1);
#endif

#ifdef SENSORULP
    RSI_CLK_SlpClkConfig(M4CLK, 3);
    RSI_ULPSS_SlpSensorClkConfig(ULPCLK , 1 ,2);
#endif

}
```

Code Block 5 ULP Subsystem (ULPSS) Clock Example

11.3 API Descriptions

11.3.1 RSI_CLK_XtalClkConfig

Source File : rsi_rom_clks.h

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
error_t RSI_CLK_XtalClkConfig(uint8_t xtalPin)
```

Description

This API is used to configure the Xtal clock

Parameters

Parameter	Description
xtalPin	Pin number of NPSS_GPIO. Possible values are 0,1,2,3,4

Return values

returns 0 on success ,Error code on failure

Example

```
/* To configure the Xtal clock source */
RSI_CLK_XtalClkConfig(2);
```

11.3.2 RSI_ULPSS_UlpProcClkConfig

Source File : rsi_rom_ulpss.h

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_ULPSS_UlpProcClkConfig (ULPClk_Type *pULPCLK,
ULP_PROC_CLK_SELECT_T clkSource, uint16_t divFactor, cdDelay delayFn)
```

Description

This API is used to configure the ULPSS processor clock source.

Parameters

Parameter	Description
pULPCLK	Pointer to the ulp clock register instance
clkSource	Enum values of clock source to select as processor clock Possible values to this Enum are following. • ULP_PROC_REF_CLK • ULP_PROC_ULP_32KHZ_RO_CLK, • ULP_PROC_ULP_32KHZ_RC_CLK • ULP_PROC_ULP_32KHZ_XTAL_CLK • ULP_PROC_ULP_32MHZ_RC_CLK

Parameter	Description
	• ULP_PROC_ULP_20MHZ_RO_CLK • ULP_PROC_SOC_CLK • ULP_PROC_ULP_DOUBLER_CLK
divFactor	To divide the clock, maximum division of clock is 255.

delayFn	Delay call back function, here if XTAL clock use for ULP pro clock then 1.5 sec delay required to turn of XTAL,
	void delay_fn()
	{
	}

Return values

returns 0 on success ,Error code on failure

Precondition

In order to enable the XTAL CLK source need to configure the NPSS_GPIO pins which can be done through RSI_CLK_XtalClkConfig(uint8_t xtalPin) API i.e we need to call that API first In order to enable the soc CLK source need to configure the Ulpss soc Clk from M4 soc clk please refer RSI_ULPSS_ClockConfig(M4Clk_Type *pCLK,boolean_t clkEnable,uint16_t divFactor,boolean_t oddDivFactor);

Example

```
void Delay()
{
}
/* To configure the ULPSS processor clock source */
RSI_ULPSS_UlpProcClkConfig(ULPCLK,ULP_PROC_ULP_32KHZ_RC_CLK,1,&Delay);
```

11.3.3 RSI_ULPSS_ClockConfig

Source File : rsi_rom_ulpss.h

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_ULPSS_ClockConfig (M4Clk_Type *pCLK, boolean_t clkEnable,
uint16_t divFactor, boolean_t oddDivFactor)
```

Description

This API is used to select the ULPSS processor clock source when input is soc clk source which is greater than 100MHz.

Parameters

Parameter	Description
pCLK	Pointer to the pll register instance
clkEnable	is to enable or disable the ulpss_soc clock • Enable 1: Enables the clock • Disable 0: Disables the clock
divFactor	To divide the clock, ensure that oddDivFactor is 0 then divFactor must be even value or else odd value. Maximum division value is 63.
oddDivFactor	Selects the type of divider for m4_soc_clk_2ulpss

Parameter	Description
	• 0 : Clock Divider(even) is selected • 1 : Odd Divider is selected

Return values

returns 0 on success ,Error code on failure

Example

```
#define   ENABLE         1
/* To select the ULPSS processor clock source when input is soc clk source which is
greater than 100MHz */
RSI_ULPSS_ClockConfig(M4CLK,Enable,0,0);
```

11.3.4 RSI_ULPSS_PeripheralEnable

Source File : rsi_rom_ulpss.h

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE RSI_ULPSS_PeripheralEnable (ULPClk_Type *pULPCLK, ULPPERIPHERALS_CLK_T
module, CLK_ENABLE_T clkType)
```

Description

This API is used to enable the particular ULP peripherial Clock.

Parameters

Parameter	Description
pULPCLK	Pointer to the ulp clock register instance
module	To select particular ulp pheripheral. • ULP_I2C_CLK, • ULP_EGPIO_CLK • ULP_AUX_CLK • ULP_FIM_CLK • ULP_VAD_CLK • ULP_TIMER_CLK • ULP_UDMA_CLK • ULP_TOUCH_CLK • ULP_UART_CLK • ULP_SSI_CLK • ULP_I2S_CLK
clkType	To select the clock as dynamic or static clock. See the #CLK_ENABLE_T for more info

Return values

returns 0 on success ,Error code on failure

Example

```
/* enable peripheral clock */
RSI_ULPSS_PeripheralEnable(ULPCLK, ULP_I2C_CLK,ENABLE_STATIC_CLK);  /* Enable i2c clock */
```

11.3.5 RSI_ULPSS_AuxClkConfig

Source File : rsi_rom_ulpss.h

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_ULPSS_AuxClkConfig (ULPClk_Type *pULPCLK, CLK_ENABLE_T clkType,
ULP_AUX_CLK_SELECT_T clkSource)
```

Description

This API is used to configure the AUX clock source.

Parameters

Parameter	Description
pULPCLK	Pointer to the ulp clock register instance
clkType	To select the clock as dynamic or static clock. Possible values for this parameter are: • ENABLE_DYN_CLK • ENABLE_STATIC_CLK
clkSource	Value of clock source to select as ADC clock. Possible values for this parameter are: • ULP_AUX_REF_CLK • ULP_AUX_32KHZ_RO_CLK • ULP_AUX_32KHZ_RC_CLK • ULP_AUX_32KHZ_XTAL_CLK • ULP_AUX_32MHZ_RC_CLK • ULP_AUX_20MHZ_RO_CLK • ULP_AUX_ULP_SOC_CLK • ULP_AUX_ULP_DOUBLER_CLK • ULP_AUX_I2S_PLL_CLK

Return values

returns 0 on success ,Error code on failure

Precondition

There are two XTAL Clk sources one is Internal and external XTAL clk source. In order to enable the external XTAL clk source need to configure the NPSS_GPIO pins which can be done through RSI_CLK_XtalClkConfig(uint8_t xtalPin) API i.e we need to call that API first In order to enable the soc CLK source need to configure the Ulpss soc Clk from M4 soc clk please refer RSI_ULPSS_ClockConfig(M4Clk_Type *pCLK,boolean_t clkEnable,uint16_t divFactor,boolean_t oddDivFactor);

Example

```
/* To configure the AUX clock source */
```

```
RSI_ULPSS_AuxClkConfig(ULPCLK,ENABLE_STATIC_CLK,ULP_AUX_32KHZ_RO_CLK);
```

11.3.6 RSI_ULPSS_TimerClkConfig

Source File : rsi_rom_ulpss.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_ULPSS_TimerClkConfig (ULPClk_Type *pULPCLK, CLK_ENABLE_T
clkType, boolean_t bTmrSync, ULP_TIMER_CLK_SELECT_T clkSource, uint8_t skipSwitchTime)
```

Description

This API is used to configure the timer clock source.

Parameters

Parameter	Description
pULPCLK	Pointer to the ulp clock register instance
clkType	To select the clock as dynamic or static clock. Possible values for this parameter are: • ENABLE_DYN_CLK • ENABLE_STATIC_CLK
bTmrSync	To enable Ulp timer in synchronous mode to ULPSS pclk • 1: Enables • 0: Disables
clkSource	Value of clock source to select as Timer clock. Possible values for this parameter are: • ULP_TIMER_REF_CLK • ULP_TIMER_32KHZ_RO_CLK • ULP_TIMER_32KHZ_RC_CLK • ULP_TIMER_32KHZ_XTAL_CLK • ULP_TIMER_32MHZ_RC_CLK • ULP_TIMER_20MHZ_RO_CLK • ULP_TIMER_ULP_SOC_CLK
skipSwitchTime	To skip the switching of timer clk. • 1 : Wait for switching timer clk • 0 : Skip waiting for switching timer clk

Return values

returns 0 on success ,Error code on failure

Precondition

There are two XTAL Clk sources one is Internal and external XTAL clock source. In order to enable the external XTAL clock source need to configure the NPSS_GPIO pins which can be done through RSI_CLK_XtalClkConfig(uint8_t xtalPin) API i.e we need to call that API first In order to enable the soc clock source need to configure the Ulpss soc clock from M4 soc clock please refer RSI_ULPSS_ClockConfig(M4Clk_Type *pCLK,boolean_t clkEnable,uint16_t divFactor,boolean_t

oddDivFactor);

Example

```
/* To configure the Timer clock source */
RSI_ULPSS_TimerClkConfig(ULPCLK,ENABLE_STATIC_CLK,1,ULP_TIMER_REF_CLK,1);
```

11.3.7 RSI_ULPSS_TimerClkDisable

Source File : rsi_rom_ulpss.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_ULPSS_TimerClkDisable(ULPClk_Type *pULPCLK )
```

Description

This API is used to disable the timer clock source

Parameters

Parameter	Description
pULPCLK	Pointer to the ulp clock register instance

Return values

returns 0 on success ,Error code on failure

Example

```
/* To disable clock source for timer peripheral */
RSI_ULPSS_TimerClkDisable(ULPCLK);
```

11.3.8 RSI_ULPSS_VadClkConfig

Source File : rsi_rom_ulpss.h

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_ULPSS_VadClkConfig (ULPClk_Type *pULPCLK, ULP_VAD_CLK_SELECT_T
clkSource, ULP_VAD_FCLK_SELECT_T FclkSource, uint16_t divFactor)
```

Description

This API is used to configure the VAD clock source.

Parameters

Parameter	Description
pULPCLK	Pointer to the ulp clock register instance
clkSource	Value of clock source to select as vad clock. Possible values for this parameter are: • ULP_VAD_32KHZ_RO_CLK

Parameter	Description
	• ULP_VAD_32KHZ_RC_CLK • ULP_VAD_32KHZ_XTAL_CLK
FclkSource	Ulp vad Fast clock select. Possible values for this parameter are: • ULP_VAD_ULP_PROCESSOR_CLK • ULP_VAD_REF_CLK • ULP_VAD_32MHZ_RC_CLK • ULP_VAD_20MHZ_RO_CLK • ULP_VAD_ULP_SOC_CLK
divFactor	To divide the clock,Maximum clock division factor for VAD is 255.

Return values

returns 0 on success ,Error code on failure

Precondition

There are two XTAL Clk sources one is Internal and external XTAL clk source. In order to enable the external XTAL clk source need to configure the NPSS_GPIO pins which can be done through RSI_CLK_XtalClkConfig(uint8_t xtalPin) API i.e we need to call that API first In order to enable the soc CLK source need to configure the Ulpss soc Clk from M4 soc clk please refer RSI_ULPSS_ClockConfig(M4Clk_Type *pCLK,boolean_t clkEnable,uint16_t divFactor,boolean_t oddDivFactor); In order to enable the ulpss processor clock source need to configure the RSI_ULPSS_UlpProcClkConfig(ULPClk_Type *pULPCLK ,boolean_t clkEnable,uint8_t clkSource,uint16_t divFactor,delayMs cbDelay)

Example

```
/* To configure the VAD clock source */
RSI_ULPSS_VadClkConfig (ULPCLK,ULP_VAD_32KHZ_RO_CLK,ULP_VAD_ULP_PROCESSOR_CLK,1);
/*!< ULP_PROCESSOR_CLK selection*/
```

11.3.9 RSI_ULPSS_TouchClkConfig

Source File : rsi_rom_ulpss.h

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_ULPSS_TouchClkConfig (ULPClk_Type *pULPCLK,
ULP_TOUCH_CLK_SELECT_T clkSource, uint16_t divFactor)
```

Description

This API is used to configure the Touch clock source.

Parameters

Parameter	Description
pULPCLK	Pointer to the ulp clock register instance
clkSource	Value of clock source to select as vad clock. Possible values for this parameter are: • ULP_TOUCH_REF_CLK • ULP_TOUCH_32KHZ_RO_CLK

Parameter	Description
	• ULP_TOUCH_32KHZ_RC_CLK • ULP_TOUCH_32KHZ_XTAL_CLK • ULP_TOUCH_32MHZ_RC_CLK • ULP_TOUCH_20MHZ_RO_CLK • ULP_TOUCH_ULP_SOC_CLK
divFactor	To divide the clock,Maximum clock division factor value for touch sensor is 255.

Return values

returns 0 on success ,Error code on failure

Note

There are two XTAL Clk sources one is Internal and external XTAL clk source. In order to enable the external XTAL clk sourceneed to configure the NPSS_GPIO pins

which can be done through RSI_CLK_XtalClkConfig(uint8_t xtalPin) API i.e we need to call that API first.

In order to enable the soc CLK source need to configure the Ulpss soc Clk from M4 soc clk

please refer RSI_ULPSS_ClockConfig(M4Clk_Type *pCLK,boolean_t clkEnable,uint16_t divFactor,boolean_t oddDivFactor);

Example

```
/* To configure the Touch clock source */
RSI_ULPSS_TouchClkConfig (ULPCLK,ULP_VAD_ULP_PROCESSOR_CLK,1);      /*!< ULP_PROCESSOR_CLK
selection*/
```

11.3.10 RSI_ULPSS_UlpSsiClkConfig

Source File : rsi_rom_ulpss.h

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_ULPSS_UlpSsiClkConfig (ULPClk_Type *pULPCLK, CLK_ENABLE_T
clkType, ULP_SSI_CLK_SELECT_T clkSource, uint16_t divFactor)
```

Description

This API is used to configure the SSI clock source.

Parameters

Parameter	Description
pULPCLK	Pointer to the ulp clock register instance
clkType	To enable or disable the SSI clock • Enable 1: Enables the clock • Disable 0: Disables the clock
clkSource	Value of clock source to select as Ulp SSI clock. Possible values for this parameter are • ULP_SSI_REF_CLK, • ULP_SSI_ULP_32KHZ_RO_CLK,

Parameter	Description
	• ULP_SSI_ULP_32KHZ_RC_CLK, • ULP_SSI_ULP_32KHZ_XTAL_CLK, • ULP_SSI_ULP_32MHZ_RC_CLK, • ULP_SSI_ULP_20MHZ_RO_CLK, • ULP_SSI_SOC_CLK,
divFactor	To divide the clock , Maximum clock division value for SSI clock is 127.

Return values

returns 0 on success ,Error code on failure

Example

```
/* To configure the SSI clock source */
RSI_ULPSS_UlpSsiClkConfig(ULPCLK,ENABLE_STATIC_CLK,ULP_SSI_ULP_32KHZ_RC_CLK,1);
```

11.3.11 RSI_ULPSS_UlpI2sClkConfig

Source File : rsi_rom_ulpss.h

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_ULPSS_UlpI2sClkConfig (ULPClk_Type *pULPCLK,
ULP_I2S_CLK_SELECT_T clkSource, uint16_t divFactor)
```

Description

This API is used to configure the I2S clock source.

Parameters

Parameter	Description
pULPCLK	Pointer to the ulp clock register instance
clkSource	Value of clock source to select as I2S clock. Possible values for this parameter are: • ULP_I2S_REF_CLK • ULP_I2S_ULP_32KHZ_RO_CLK • ULP_I2S_ULP_32KHZ_RC_CLK • ULP_I2S_ULP_32KHZ_XTAL_CLK • ULP_I2S_ULP_32MHZ_RC_CLK • ULP_I2S_ULP_20MHZ_RO_CLK • ULP_I2S_SOC_CLK • ULP_I2S_ULP_DOUBLER_CLK • ULP_I2S_PLL_CLK
divFactor	To divide the clock, Maximum clock division value for ULP_I2S is 255.

Return values

returns 0 on success ,Error code on failure

Example

```
/* To configure the I2S clock source. */
RSI_ULPSS_UlpI2sClkConfig(ULPCLK  ,ULP_I2S_ULP_32KHZ_RC_CLK,1);
```

11.3.12 RSI_ULPSS_UlpUartClkConfig

Source File : rsi_rom_ulpss.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_ULPSS_UlpUartClkConfig (ULPClk_Type *pULPCLK, CLK_ENABLE_T
clkType, boolean_t bFrClkSel, ULP_UART_CLK_SELECT_T clkSource, uint16_t divFactor)
```

Description

This API is used to configure the UART clock source

Parameters

Parameter	Description
pULPCLK	Pointer to the ulp clock register instance
clkType	To select the clock as dynamic or static clock. Possible values for this parameter are: • ENABLE_DYN_CLK • ENABLE_STATIC_CLK
bFrClkSel	To enable or disable ulp uart clk selection • 1: Fractional Divider output is selected • 0: Swallow Divider output is selected
clkSource	Value of clock source to select as processor clock Possible values for this parameter are: • ULP_UART_REF_CLK • ULP_UART_ULP_32KHZ_RO_CLK • ULP_UART_ULP_32KHZ_RC_CLK • ULP_UART_ULP_32KHZ_XTAL_CLK • ULP_UART_ULP_32MHZ_RC_CLK • ULP_UART_ULP_20MHZ_RO_CLK • ULP_UART_SOC_CLK • ULP_UART_ULP_DOUBLER_CLK
divFactor	To divide the clock,Maximum clock division value for ULP_UART is 7.

Return values

returns 0 on success ,Error code on failure

Example

```
/* To configure the UART clock source */
```

```
RSI_ULPSS_UlpUartClkConfig(ULPCLK,ENABLE_STATIC_CLK,1,ULP_UART_ULP_32KHZ_RC_CLK,1);
```

11.3.13 RSI_ULPSS_SlpSensorClkConfig

Source File : rsi_rom_ulpss.h

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_ULPSS_SlpSensorClkConfig (ULPClk_Type *pULPCLK, boolean_t
clkEnable, uint32_t divFactor)
```

Description

This API is used to configure the sleep sensor clock source.

Parameters

Parameter	Description
pULPCLK	Pointer to the ulp clock register instance
clkEnable	To enable or disable the sleep sensor clock • Enable 1: Enables the clock • Disable 0: Disables the clock
divFactor	To divide the clock,Maximum clock division value for sleep sensor is 255.

Return values

returns 0 on success ,Error code on failure

Note

In order to enable the XTAL CLK source need to configure the NPSS_GPIO pins
which can be done through RSI_CLK_XtalClkConfig(uint8_t xtalPin) API i.e we need to call that API first

Example

```
/* To configure the sleep sensor clock source */
RSI_ULPSS_SlpSensorClkConfig (ULPCLK,1,1);
```

11.3.14 RSI_ULPSS_RefClkConfig

Source File : rsi_rom_ulpss.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_ULPSS_RefClkConfig(ULPSS_REF_CLK_SEL_T clkSource)
```

Description

This API is used to select the ULPSS processor ref clk configuration.

Parameter

Parameter	Description
clkSource	Clock source to be selected. Possible values to this API are following. • ULP_PROC_REF_CLK, • ULP_PROC_ULP_32KHZ_RO_CLK, • ULP_PROC_ULP_32KHZ_RC_CLK, • ULP_PROC_ULP_32KHZ_XTAL_CLK, • ULP_PROC_ULP_32MHZ_RC_CLK, • ULP_PROC_ULP_20MHZ_RO_CLK, • ULP_PROC_SOC_CLK, • ULP_PROC_ULP_DOUBLER_CLK

Return values

returns 0 on success ,Error code on failure.

Example

```
/* configure ULPSS clock */
RSI_ULPSS_RefClkConfig(ULP_PROC_REF_CLK);
```

11.3.15 RSI_ULPSS_UlpPeriClkEnable

Source File : rsi_rom_ulpss.h

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE  error_t RSI_ULPSS_UlpPeriClkEnable (ULPClk_Type *pULPCLK, uint32_t
u32Flags)
```

Description

This API is used to enable different pheriheral clocks in ULPSS.

Parameters

Parameter	Description
pULPCLK	Pointer to the ulp clock register instance
u32Flags	Ored value of the clock enable bits of particular peripheral Possible values for this parameter are • ULPSS_TASS_QUASI_SYNC • ULPSS_M4SS_SLV_SEL • AUX_SOC_EXT_TRIG_2_SEL • AUX_SOC_EXT_TRIG_1_SEL • AUX_ULP_EXT_TRIG_2_SEL • AUX_ULP_EXT_TRIG_1_SEL • TIMER_PCLK_EN • EGPIO_PCLK_ENABLE

Parameter	Description
	• EGPIO_PCLK_DYN_CTRL_DISABLE • CLK_ENABLE_ULP_MEMORIES • VAD_CLK_EN • FIM_CLK_EN • REG_ACCESS_SPI_CLK_EN • EGPIO_CLK_EN • CLK_ENABLE_TIMER • VAD_PCLK_ENABLE • FIM_PCLK_ENABLE • SCLK_ENABLE_UART • PCLK_ENABLE_UART • SCLK_ENABLE_SSI_MASTER • PCLK_ENABLE_SSI_MASTER • CLK_ENABLE_I2S • PCLK_ENABLE_I2C • RELEASE_SOFT_RESET • PCM_FSYNC_START • PCM_ENABLE

Return values

returns 0 on success ,Error code on failure

Example

```
/* enable peripheral clock */
RSI_ULPSS_UlpPeriClkEnable(ULPCLK,ULPSS_TASS_QUASI_SYNC|ULPSS_M4SS_SLV_SEL);
```

11.3.16 RSI_ULPSS_UlpPeriClkDisable

Source File : rsi_rom_ulpss.h

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_ULPSS_UlpPeriClkDisable (ULPClk_Type *pULPCLK, uint32_t
u32Flags)
```

Description

This API is used to disable diifferent pheriheral clocks in ULPSS.

Parameters

Parameter	Description
pULPCLK	Pointer to the ulp clock register instance
u32Flags	is to Ored value of the clock enable bits of particular peripheral Possible values for this parameter are

Parameter	Description
	• ULPSS_TASS_QUASI_SYNC • ULPSS_M4SS_SLV_SEL • AUX_SOC_EXT_TRIG_2_SEL • AUX_SOC_EXT_TRIG_1_SEL • AUX_ULP_EXT_TRIG_2_SEL • AUX_ULP_EXT_TRIG_1_SEL • TIMER_PCLK_EN • EGPIO_PCLK_ENABLE • EGPIO_PCLK_DYN_CTRL_DISABLE • CLK_ENABLE_ULP_MEMORIES • VAD_CLK_EN • FIM_CLK_EN • REG_ACCESS_SPI_CLK_EN • EGPIO_CLK_EN • CLK_ENABLE_TIMER • VAD_PCLK_ENABLE • FIM_PCLK_ENABLE • SCLK_ENABLE_UART • PCLK_ENABLE_UART • SCLK_ENABLE_SSI_MASTER • PCLK_ENABLE_SSI_MASTER • CLK_ENABLE_I2S • PCLK_ENABLE_I2C • RELEASE_SOFT_RESET • PCM_FSYNC_START • PCM_ENABLE

Return values

returns 0 on success ,Error code on failure

Example

```
/* disable peripheral clock */
RSI_ULPSS_UlpPeriClkDisable (ULPCLK,ULPSS_TASS_QUASI_SYNC|ULPSS_M4SS_SLV_SEL);
```

11.3.17 RSI_ULPSS_UlpDynClkEnable

Source File : rsi_rom_ulpss.h

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_ULPSS_UlpDynClkEnable (ULPClk_Type *pULPCLK, uint32_t u32Flags)
```

Description

This API is used to enable different pheriheral clocks in ULPSS.

Parameter

Parameter	Description
pULPCLK	Pointer to the ulp clock register instance
u32Flags	Ored value of the clock enable bits of particular peripheral Possible values for this parameter are • ULPSS_TASS_QUASI_SYNC • ULPSS_M4SS_SLV_SEL • AUX_SOC_EXT_TRIG_2_SEL • AUX_SOC_EXT_TRIG_1_SEL • AUX_ULP_EXT_TRIG_2_SEL • AUX_ULP_EXT_TRIG_1_SEL • TIMER_PCLK_EN • EGPIO_PCLK_ENABLE • EGPIO_PCLK_DYN_CTRL_DISABLE • CLK_ENABLE_ULP_MEMORIES • VAD_CLK_EN • FIM_CLK_EN • REG_ACCESS_SPI_CLK_EN • EGPIO_CLK_EN • CLK_ENABLE_TIMER • VAD_PCLK_ENABLE • FIM_PCLK_ENABLE • SCLK_ENABLE_UART • PCLK_ENABLE_UART • SCLK_ENABLE_SSI_MASTER • PCLK_ENABLE_SSI_MASTER • CLK_ENABLE_I2S • PCLK_ENABLE_I2C • RELEASE_SOFT_RESET • PCM_FSYNC_START • PCM_ENABLE

Return values

returns 0 on success ,Error code on failure

Example

```
/* enable dyanamic peripheral clock */
RSI_ULPSS_UlpDynClkEnable (ULPCLK,ULPSS_TASS_QUASI_SYNC|ULPSS_M4SS_SLV_SEL);
```

11.3.18 RSI_ULPSS_UlpDynClkDisable

Source File : rsi_rom_ulpss.h

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_ULPSS_UlpDynClkDisable (ULPClk_Type *pULPCLK, uint32_t u32Flags)
```

Description

This API is used to enable different pheriheral clocks in ULPSS.

Parameters

Parameter	Description
pULPCLK	Pointer to the ulp clock register instance
u32Flags	Ored value of the clock enable bits of particular peripheral Possible values for this parameter are • ULPSS_TASS_QUASI_SYNC • ULPSS_M4SS_SLV_SEL • AUX_SOC_EXT_TRIG_2_SEL • AUX_SOC_EXT_TRIG_1_SEL • AUX_ULP_EXT_TRIG_2_SEL • AUX_ULP_EXT_TRIG_1_SEL • TIMER_PCLK_EN • EGPIO_PCLK_ENABLE • EGPIO_PCLK_DYN_CTRL_DISABLE • CLK_ENABLE_ULP_MEMORIES • VAD_CLK_EN • FIM_CLK_EN • REG_ACCESS_SPI_CLK_EN • EGPIO_CLK_EN • CLK_ENABLE_TIMER • VAD_PCLK_ENABLE • FIM_PCLK_ENABLE • SCLK_ENABLE_UART • PCLK_ENABLE_UART • SCLK_ENABLE_SSI_MASTER • PCLK_ENABLE_SSI_MASTER • CLK_ENABLE_I2S • PCLK_ENABLE_I2C • RELEASE_SOFT_RESET • PCM_FSYNC_START • PCM_ENABLE

Return values

returns 0 on success ,Error code on failure

Example

```
/* disable dyanamic peripheral clock */
RSI_ULPSS_UlpDynClkDisable (ULPCLK,ULPSS_TASS_QUASI_SYNC|ULPSS_M4SS_SLV_SEL);
```

11.3.19 RSI_ULPSS_PeripheralDisable

Source File : rsi_rom_ulpss.h

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE_t RSI_ULPSS_PeripheralDisable (ULPClk_Type *pULPCLK, ULPPERIPHERALS_CLK_T
module)
```

Description

This API is used to Disable the particular ULP peripherial Clock.

Parameters

Parameter	Description
pULPCLK	Pointer to the ulp clock register instance
module	To select particular ulp pheripheral. Possible values for this parameter are: • ULP_I2C_CLK, • ULP_EGPIO_CLK • ULP_AUX_CLK • ULP_FIM_CLK • ULP_VAD_CLK • ULP_TIMER_CLK • ULP_UDMA_CLK • ULP_TOUCH_CLK • ULP_UART_CLK • ULP_SSI_CLK • ULP_I2S_CLK

Return values

returns 0 on success ,Error code on failure

Example

```
/* disable peripheral clock */
RSI_ULPSS_PeripheralDisable(ULPCLK, ULP_I2C_CLK,ENABLE_STATIC_CLK);  /* disable i2c clock
*/
```

12 Windowed Watchdog Timer (WWDT)

12.1 Overview

This section explains how to configure and use the Windowed Watchdog Timer using Redpine MCU SAPIs.

12.2 Programming Sequence

```
#include "rsi_chip.h" /* Redpine_MCU_Vx.y.z\Host_MCU\common\chip\inc */
#include "rsi_board.h" /* Redpine_MCU_Vx.y.z\Host_MCU\board\inc\ */

#define WDT_IRQHandler            IRQ020_Handler            /*!<WDT IRQ Handler       */
#define NVIC_WDT                  NPSS_TO_MCU_WDT_INTR_IRQn /*!<WDT NVIC IRQ Number   */

#define WDT_SYSTEM_RESET_TIMER    18  /*!<Watch dog SOC reset delay timer programming
values. Counter is decremented
                                      with respect to clock tick of FSM clock. Total
duration = 2^WDT_SYSTEM_RESET_TIMER
                                      FSM clocks (32Khz clock)*/
#define WDT_INTERRUPT_TIMER       16 /*!<watchdog window timer Total duration =
2^WDT_INTERRUPT_TIMER FSM clocks(32Khz clocks) */
#define WDT_RESET_GENERATION_CNT  10 /*!<Selects after how many interrupts watch dog RE-
START should not provide               */

/**
 * @brief  WDT interrupt handler
 * @param  None
 * @return None
 */
void WDT_IRQHandler(void)
{
    /*Clear interrupt */
    RSI_WWDT_IntrClear();
    wdtIntrCnt++;
    /*Do not re-start the WDT after 10 periodic interrupts to generate the RESET from
WDT*/
    if(wdtIntrCnt <WDT_RESET_GENERATION_CNT){
        /*Restart the WDT */
        RSI_WWDT_ReStart(WDT);
    }
    #ifdef DEBUG_UART
      /*  Prints on hyper-terminal  */
      DEBUGOUT("WDT handler %d\r\n",wdtIntrCnt);
    #endif
    /*Toggle LED0*/
    RSI_Board_LED_Toggle(0);
    return ;
}

int main(void)
{
    /*Configures the system default clock and power configurations*/
    SystemCoreClockUpdate();
    RSI_Board_Init();

    RSI_Board_LED_Set(1, 1);
    RSI_Board_LED_Set(2, 1);

    #ifdef DEBUG_UART
     DEBUGOUT("Main..!\r\n");
    #endif

    /*Check the system RESET status*/
    if(RSI_PS_GetComnIntrSts() & WATCH_DOG_WINDOW_RESET_INTERRUPT)
    {
     #ifdef DEBUG_UART
        DEBUGOUT("WDT RESET occurred..\r\n");
```

```
      #endif
     }
     else
     {
      #ifdef DEBUG_UART
         DEBUGOUT("Power on RESET occurred..\r\n");
      #endif
     }
     /*FSM clock enable for WDT to be functional*/
     /*Enable clock sources */
     RSI_IPMU_ClockMuxSel(1);
     RSI_PS_FsmLfClkSel(KHZ_RO_CLK_SEL);
     RSI_PS_FsmHfClkSel(FSM_32MHZ_RC);
     /*Un mask WDT interrupt */
     RSI_WWDT_IntrUnMask();
     /* Time Period Programming */
     RSI_TIMEPERIOD_TimerClkSel(TIME_PERIOD , 0x003E7FFF);
     /*configure the WDT reset and interrupt counters */
     RSI_WWDT_ConfigIntrTimer(WDT , WDT_INTERRUPT_TIMER);
     /*Configure the WDT reset value*/
     RSI_WWDT_ConfigSysRstTimer(WDT , WDT_SYSTEM_RESET_TIMER);

     /*NVIC interrupt enables*/
     NVIC_EnableIRQ(NVIC_WDT);
     /*Start WDT */
     RSI_WWDT_Start(WDT);

     /*Wait for interrupt*/
     while(1)
     {
         __ASM("WFI");
     }
 }
```

Code Block 6 Windowed Watchdog Timer example

12.3 API Descriptions

12.3.1 RSI_WWDT_Init

Source File : rsi_wwdt.c

Path : Redpine _MCU_Vx.y.z\Host_MCU\library\driver\inc

Prototype

```
void RSI_WWDT_Init(MCU_WDT_Type *pstcWDT)
```

Description

This API is used to initialize the Watch dog timer.

Parameter

Parameter	Description
pstcWDT	Pointer to the WDT register instance

Return values

None

Example

```
/* Initialize WDT*/
```

```
RSI_WWDT_Init(WDT);
```

12.3.2 RSI_WWDT_IntrUnMask

Source File : rsi_wwdt.h

Path : Redpine _MCU_Vx.y.z\Host_MCU\library\driver\inc

Prototype

```
STATIC INLINE void RSI_WWDT_IntrUnMask(void)
```

Description

This API is used to unmask the Watch dog timer.

Parameter

None

Return values

None

Example

```
/*unmask interrupt*/
RSI_WWDT_IntrUnMask();
```

12.3.3 RSI_WWDT_ConfigIntrTimer

Source File : rsi_wwdt.h

Path : Redpine _MCU_Vx.y.z\Host_MCU\library\driver\inc

Prototype

```
STATIC INLINE void RSI_WWDT_ConfigIntrTimer(MCU_WDT_Type *pstcWDT , uint16_t
u16IntrTimerVal)
```

Description

This API is used to configure the interrupt timer of the watch dog timer.

Parameters

Parameter	Description
pstcWDT	Pointer to the WDT register instance
u16IntrTimerVal	interrupt timer value

Return values

None

Example

```
/* declaration */
#define WDT_INTERRUPT_TIMER       16
/*configure the WDT reset and interrupt counters */
```

```
RSI_WWDT_ConfigIntrTimer(WDT , WDT_INTERRUPT_TIMER);
```

12.3.4 RSI_WWDT_ConfigSysRstTimer

Source File : rsi_wwdt.h

Path : Redpine _MCU_Vx.y.z\Host_MCU\library\driver\inc

Prototype

```
STATIC INLINE void RSI_WWDT_ConfigSysRstTimer(MCU_WDT_Type *pstcWDT , uint16_t
u16SysRstVal)
```

Description

This API is used to the system reset timer of the watch dog timer.

Parameters

Parameter	Description
pstcWDT	Pointer to the WDT register instance
u16SysRstVal	reset value

Return values

None

Example

```
/* declaration */
#define WDT_SYSTEM_RESET_TIMER    18
/*Configure the WDT reset value*/
RSI_WWDT_ConfigSysRstTimer(WDT , WDT_SYSTEM_RESET_TIMER);
```

12.3.5 RSI_WWDT_Start

Source File : rsi_wwdt.c

Path : Redpine _MCU_Vx.y.z\library\systemlevel\src

Prototype

```
void RSI_WWDT_Start(MCU_WDT_Type *pstcWDT)
```

Description

This API is used to start WDT.

Parameter

Parameter	Description
pstcWDT	Pointer to the WDT register instance

Return values

None

Example

```
/*Start WDT */
RSI_WWDT_Start(WDT);
```

12.3.6 RSI_WWDT_ReStart

Source File : rsi_wwdt.h

Path : Redpine _MCU_Vx.y.z\Host_MCU\library\driver\inc

Prototype

```
STATIC INLINE void RSI_WWDT_ReStart(MCU_WDT_Type *pstcWDT)
```

Description

This API is used to restart the Watch dog timer.

Parameter

Parameter	Description
pstcWDT	Pointer to the WDT register instance

Return values

None

Example

```
/*restart WDT*/
RSI_WWDT_ReStart(WDT);
```

12.3.7 RSI_WWDT_IntrMask

Source File : rsi_wwdt.h

Path : Redpine _MCU_Vx.y.z\Host_MCU\library\driver\inc

Prototype

```
STATIC INLINE void RSI_WWDT_IntrMask(void)
```

Description

This API is used to mask the Watch dog timer.

Parameter

None

Return values

None

Example

```
/*mask interrupt*/
RSI_WWDT_IntrMask();
```

12.3.8 RSI_WWDT_IntrClear

Source File : rsi_wwdt.c

Path : Redpine _MCU_Vx.y.z\library\systemlevel\src

Prototype

```
void RSI_WWDT_IntrClear(void)
```

Description

This API is used to clear the interrupt.

Parameter

None

Return values

None

Example

```
/*clear interrupt*/
RSI_WWDT_IntrClear();
```

12.3.9 RSI_WWDT_GetIntrStatus

Source File : rsi_wwdt.c

Path : Redpine _MCU_Vx.y.z\library\systemlevel\src

Prototype

```
uint8_t RSI_WWDT_GetIntrStatus(void)
```

Description

This API is used to getting status of interrupt.

Parameter

None

Return values

return status of interrupt if interrupt is enable return 1 or else return zero.

Example

```
/* declaration */
uint8_t intr_status;
/*get status of interrupt*/
intr_status=RSI_WWDT_GetIntrStatus();
```

12.3.10 RSI_WWDT_DeInit

Source File : rsi_wwdt.c

Path : Redpine _MCU_Vx.y.z\library\systemlevel\src

Prototype

```
void RSI_WWDT_DeInit(MCU_WDT_Type *pstcWDT)
```

Description

This API is used to deinitialize WDT.

Parameter

Parameter	Description
pstcWDT	Pointer to the WDT register instance

Return values

None

Example

```
/*deinitialize WDT */
RSI_WWDT_DeInit(WDT);
```

12.3.11 RSI_WWDT_Disable

Source File : rsi_wwdt.h

Path : Redpine _MCU_Vx.y.z\Host_MCU\library\driver\inc

Prototype

```
STATIC INLINE void RSI_WWDT_Disable(MCU_WDT_Type *pstcWDT)
```

Description

This API is used to Disable the Watch dog timer.

Parameter

Parameter	Description
pstcWDT	Pointer to the WDT register instance

Return values

None

Example

```
/*Disable WDT*/
RSI_WWDT_Disable(WDT);
```

13 Real-time Clock (RTC)

13.1 Overview

This section explains how to configure and use the Real-time Clock using Redpine MCU SAPIs.

13.2 Programming Sequence

```
#include "rsi_chip.h" /* Redpine_MCU_Vx.y.z\Host_MCU\common\chip\inc */
#include "rsi_board.h /*  Redpine_MCU_Vx.y.z\Host_MCU\board\inc */
#define NPSS_ALARM_IRQHandler   IRQ028_Handler
/**
 * @brief  This is Alarm IRQ Handler
 * @param  None
 * @return None
 */
void NPSS_ALARM_IRQHandler(void)
{
    volatile uint32_t statusRead = 0;

    /*Get the interrupt status */
    statusRead = RSI_RTC_GetIntrStatus();
    if(statusRead & NPSS_TO_MCU_ALARM_INTR)
    {
        /*Clear wake up interrupt */
        RSI_RTC_IntrClear(RTC_ALARM_INTR);
        /*Update seconds for next boundary alarm */
        alarmConfig.Second   = (alarmConfig.Second + ALARAM_PERIODIC_TIME);

        /*Update minutes for next boundary alarm */
        if(alarmConfig.Second  >= 59)
        {
            alarmConfig.Second = 0;
            alarmConfig.Minute = (alarmConfig.Minute + 1);
        }
        /*Update minutes for next boundary alarm */
        if(alarmConfig.Minute >= 59)
        {
            alarmConfig.Minute =0;
            alarmConfig.Hour = (alarmConfig.Hour + 1);
        }
        /*Update date for next boundary alarm */
        if(alarmConfig.Hour  >= 23)
        {
            alarmConfig.Hour  =0;
            alarmConfig.Day = (alarmConfig.Day + 1);
        }
        /*Reconfigure the Alarm for next alarm boundary*/
        RSI_RTC_SetAlarmDateTime(RTC , &alarmConfig);
    }
   RSI_RTC_GetDateTime(RTC , &readTime);
   #ifdef DEBUG_UART
    DEBUGOUT("TIME:
%d:%d:%d:%d\n",readTime.Hour,readTime.Minute,readTime.Second,readTime.MilliSeconds);
    DEBUGOUT("DATE: %d/%d/%d\n\n",readTime.Day,readTime.Month,(readTime.Year + 2000));
   #endif // DEBUG_UART
    return ;
}

/**
 * @brief  Main program.
 * @param  None
 * @retval None
 */
int main(void)
{
```

```
    /*Configures the system default clock and power configurations*/
     SystemCoreClockUpdate();
    #ifdef DEBUG_UART
      /*Init debug UART*/
      DEBUGINIT();
    #endif
    /*Enable clock sources */
    RSI_IPMU_ClockMuxSel(1);
    RSI_PS_FsmLfClkSel(KHZ_RC_CLK_SEL);
    RSI_PS_FsmHfClkSel(FSM_32MHZ_RC);

    /*Init RTC*/
    RSI_RTC_Init(RTC);

    /*RTC configuration with some default time */
    rtcConfig.DayOfWeek        = Saturday;
    rtcConfig.MilliSeconds     = 0;
    rtcConfig.Century          = 0;
    rtcConfig.Day              = 27;
    rtcConfig.Hour             = 6;
    rtcConfig.Minute           = 30;
    rtcConfig.Month            = Febuary;
    rtcConfig.Second           = 2 ;
    rtcConfig.Year             = 17;

    /*RTC alarm configuration */
    alarmConfig.DayOfWeek      = Saturday;
    alarmConfig.Second         = (2 + ALARAM_PERIODIC_TIME);
    alarmConfig.MilliSeconds   = 0;
    alarmConfig.Century        = 0;
    alarmConfig.Day            = 27;
    alarmConfig.Hour           = 6;
    alarmConfig.Minute         = 30;
    alarmConfig.Month          = Febuary;
    alarmConfig.Year           = 17;

    /*start RTC */
    RSI_RTC_Start(RTC);
    /*Set the RTC configuration*/
    RSI_RTC_SetDateTime(RTC ,&rtcConfig);
    /*Set Alarm configuration */
    RSI_RTC_SetAlarmDateTime(RTC , &alarmConfig);
    /*Enable Alarm feature*/
    RSI_RTC_AlamEnable(RTC ,ENABLE);
    /*Enable RTC ALARM ,sec and milli second interrupts*/
    RSI_RTC_IntrUnMask(RTC_ALARM_INTR | RTC_SEC_INTR | RTC_MSEC_INTR);
    /*Initilization RTC CALIBRATION*/
    RSI_RTC_CalibInitilization();
    /* To calibrate rc and ro */
    RSI_RTC_ROCLK_Calib(TIME_PERIOD,1,1,3 ,1,1,0);
    /*Start RTC */
    RSI_RTC_Start(RTC);
    /*Enable NVIC for RTC */
    NVIC_EnableIRQ(NVIC_RTC_ALARM);
    /*Enable NVIC for Sec and milli second interrupts */
    NVIC_EnableIRQ(NVIC_RTC);

    while(1)
    {
     /*Wait for interrupt */
      __WFI();
    }
  /*Code will never reach here,Just to satisfy the standard main*/
}
```

Code Block 7 RTC example

13.3 API Descriptions

13.3.1 RSI_RTC_Init

Source File : rsi_rtc .c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\src\

Prototype

```
void RSI_RTC_Init(RTC_Type   *Cal)
```

Description

This API is used to initialization the RTC block.

Parameter

Parameter	Description
Cal	pointer to the RTC register instance

Return values

None

Example

```
/*Initialization of RTC*/
 RSI_RTC_Init(RTC);
```

Note

Please ensure the FSM high frequency and FSM low frequency clocks are enabled before calling this API.

13.3.2 RSI_RTC_Start

Source File : rsi_rtc .c

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\src\

Prototype

```
void RSI_RTC_Start(RTC_Type *Cal)
```

Description

This API is used to start the RTC.

Parameter

Parameter	Description
Cal	pointer to the RTC register instance

Return values

None

Example

```
/*start RTC */
RSI_RTC_Start(RTC);
```

13.3.3 RSI_RTC_SetDateTime

Source File : rsi_rtc .c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\src\

Prototype

```
error_t RSI_RTC_SetDateTime(RTC_Type *Cal , RTC_TIME_CONFIG_T *date)
```

Description

This API is used to set the RTC configuration.

Parameters

Parameter	Description
Cal	pointer to the RTC register instance
date	pointer to the RTC configuration structure

Return values

Zero on success and error code on failure

Example

```
/*Define the structure variable to RTC_TIME_CONFIG_T to configure the RTC module*/
static RTC_TIME_CONFIG_T rtcConfig;

/*RTC configuration with some default time */
rtcConfig.DayOfWeek        = Saturday;
rtcConfig.MilliSeconds    = 0;
rtcConfig.Century          = 0;
rtcConfig.Day              = 27;
rtcConfig.Hour             = 6;
rtcConfig.Minute           = 30;
rtcConfig.Month            = Febuary;
rtcConfig.Second           = 2 ;
rtcConfig.Year             = 16;

/*Set the RTC configuration*/
RSI_RTC_SetDateTime(RTX

,&rtcConfig);
```

13.3.4 RSI_RTC_SetAlarmDateTime

Source File : rsi_rtc .c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\src\

Prototype

```
error_t RSI_RTC_SetAlarmDateTime(RTC_Type *Cal , RTC_TIME_CONFIG_T *alarm)
```

Description

This API is used to Set the alarm for RTC module

Parameters

Parameter	Description
Cal	pointer to the RTC register instance
alarm	pointer to alarm configuration structure

Return values

Zero on success and error code on failure.

Example

```
/*Define the structure variable to RTC_TIME_CONFIG_T to configure the RTC module*/
static RTC_TIME_CONFIG_T alarmConfig;

/*RTC alarm configuration */
alarmConfig.enDayOfWeek      = Saturday;
alarmConfig.u8Second         = 2;
alarmConfig.u16MilliSeconds  = 0;
alarmConfig.u8Century        = 0;
alarmConfig.u8Day            = 27;
alarmConfig.u8Hour           = 6;
alarmConfig.u8Minute         = 30;
alarmConfig.u8Month          = Febuary;
alarmConfig.u8Year           = 16;

/*Set Alarm configuration */
RSI_RTC_SetAlarmDateTime(RTC  , &alarmConfig);
```

13.3.5 RSI_RTC_AlamEnable

Source File : rsi_rtc .c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\src\

Prototype

```
void RSI_RTC_AlamEnable(RTC_Type   *Cal , boolean_t bVal)
```

Description

This API is used to Set the alarm for RTC module.

Parameters

Parameter	Description
Cal	pointer to the RTC register instance
bVal	to enable or disable the alarm 0 : Disable the alarm 1 : Enable the alarm

Return values

None

Example

```
/*Enable Alarm feature*/
RSI_RTC_AlamEnable(RTC,ENABLE);
/*Or*/
/*Disable Alarm feature*/
```

```
RSI_RTC_AlamEnable(RTC,DISABLE);
```

13.3.6 RSI_RTC_IntrUnMask

Source File : rsi_rtc .c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\src\

Prototype

```
void RSI_RTC_IntrUnMask(uint32_t intr)
```

Description

This API is used to enable the RTC alarm interrupts.

Parameter

Parameter	Description
intr	Ored value of interrupt to be enabled/Un-mask • RTC_MSEC_INTR : Use this macro to enable msec interrupt • RTC_SEC_INTR : Use this macro to enable sec interrupt • RTC_ALARM_INTR : Use this macro to enable alarm interrupt

Return values

None

Example

```
/*Enable RTC ALARM interrupt*/
RSI_RTC_IntrUnMask(RTC_ALARM_INTR);
```

13.3.7 RSI_RTC_GetDateTime

Source File : rsi_rtc .c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\src\

Prototype

```
error_t RSI_RTC_GetDateTime(RTC_Type *Cal ,  RTC_TIME_CONFIG_T *date)
```

Description

This API is used to Get the RTC time

Parameters

Parameter	Description
Cal	pointer to the RTC register instance
date	pointer to the rtc structure to hold the current time parameters

Return values

Zero on success and error code on failure

Example

```
/*Define the structure variable to RTC_TIME_CONFIG_T to configure the RTC module*/
static RTC_TIME_CONFIG_T  readTime;

/*Get RTC time */
RSI_RTC_GetDateTime(RTC  , &readTime);
```

13.3.8 RSI_RTC_GetAlarmDateTime

Source File : rsi_rtc .c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\src\

Prototype

```
error_t RSI_RTC_GetDateTime(RTC_Type *Cal ,  RTC_TIME_CONFIG_T *date)
```

Description

This API is used to Get alarm configurations for RTC

Parameters

Parameter	Description
Cal	pointer to the RTC register instance
pstcAlarm	pointer to the RTC alarm configuration structure

Return values

Zero on success and error code on failure.

Example

```
/*Define the structure variable to RTC_TIME_CONFIG_T to configure the RTC module*/
static RTC_TIME_CONFIG_T  readAlarmConfig;

/*Read the configured alaram values*/
RSI_RTC_GetAlarmDateTime(RTC  , &readAlarmConfig);
```

13.3.9 RSI_RTC_SetDayOfWeek

Source File : rsi_rtc .c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\src\

Prototype

```
void RSI_RTC_SetDayOfWeek(RTC_Type *Cal , RTC_DAY_OF_WEEK_T dayInWeek)
```

Description

This API is used to set the Day in a week.

Parameters

Parameter	Description

Parameter	Description
Cal	pointer to the RTC register instance
dayInWeek	enum value of days in a week following are the possible values for this parameter. • Sunday • Monday • Tuesday • Wednesday • Thursday • Friday • Saturday

Return values

None

Example

```
/*set day in week */
RSI_RTC_SetDayOfWeek(RTC,1);
```

13.3.10 RSI_RTC_GetIntrStatus

Source File : rsi_rtc .c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\src\

Prototype

```
uint32_t  RSI_RTC_GetIntrStatus(void)
```

Description

This API is used to get the RTC alarm interrupt status

Parameter

None

Return values

interrupt status.

Example

```
/* interrupt status read variable declaration */
volatile uint32_t statusRead = 0;

/*Get the interrupt status */
statusRead = RSI_RTC_GetIntrStatus();
```

13.3.11 RSI_RTC_IntrClear

Source File : rsi_rtc .c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\src\

Prototype

```
void RSI_RTC_IntrClear(uint32_t intr)
```

Description

This API is used to clear the RTC alarm interrupts

Parameter

Parameter	Description
intr	Ored value of interrupt to be enabled/Un-mask RTC_MSEC_INTR : Use this macro to enable msec interrupt RTC_SEC_INTR : Use this macro to enable sec interrupt RTC_ALARM_INTR : Use this macro to enable alarm interrupt

Return values

None

Example

```
/*Clear wake up interrupt */
RSI_RTC_IntrClear(RTC_ALARM_INTR);
```

13.3.12 RSI_RTC_IntrMask

Source File : rsi_rtc .c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\src\

Prototype

```
void RSI_RTC_IntrMask(uint32_t intr)
```

Description

This API is used to disable the RTC alarm interrupts

Parameter

Parameter	Description
intr	Ored value of interrupt to be enabled/Un-mask RTC_MSEC_INTR : Use this macro to enable msec interrupt RTC_SEC_INTR : Use this macro to enable sec interrupt RTC_ALARM_INTR : Use this macro to enable alarm interrupt

Return values

None

Example

```
/* Disable RTC ALARM interrupt */
RSI_RTC_IntrMask(RTC_ALARM_INTR);
```

13.3.13 RSI_RTC_Stop

Source File : rsi_rtc .c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\src\

Prototype

```
void RSI_RTC_Stop(RTC_Type   *Cal)
```

Description

This API is used to stop the RTC operation.

Parameter

Parameter	Description
Cal	pointer to the RTC register instance

Return values

None

Example

```
/*stop RTC */
RSI_RTC_Stop(RTC);
```

14 Clocks (CLK)

14.1 Overview

This section explains how to configure and use the M4 Subsystem using Redpine MCU SAPIs.

14.2 Programming Sequence

```
#include "rsi_chip.h" /* Redpine_MCU_Vx.y.z\Host_MCU\common\chip\inc */

/* define specific peripheral MACRO for demonstrate that specific peripheral clock example
*/

int main()
{
#ifdef PWMPLL

   RSI_CLK_ByPassClkConfig(M4CLK, 1 , IntfPlClkWithbyp , I2sPllBypClk);
   RSI_CLK_ByPassClkConfig(M4CLK, 1 , SocPllClkWithbyp , I2sPllBypClk);
   RSI_CLK_SocPllClkBypassEnable(1);
   RSI_CLK_IntfPLLClkBypassEnable(1);
   RSI_CLK_CanClkConfig(M4CLK ,1,    2);
   RSI_CLK_CanClkConfig(M4CLK ,1,    4);
   RSI_CLK_SocPllClkEnable(M4CLK);
   RSI_CLK_SocPllPdEnable(Enable);
   RSI_CLK_SocPllTurnOn(M4CLK);
   RSI_CLK_SocPllTurnOff(M4CLK);
   RSI_CLK_I2sPllTurnOff(M4CLK);
   RSI_CLK_I2sPllClkReset();
   RSI_CLK_I2sPllTurnOn(M4CLK);
   RSI_CLK_I2sPllSetFreqDiv(2,3);
   RSI_CLK_I2sPllClkEnable(M4CLK);
   RSI_CLK_I2sPllTurnOff(M4CLK);
   RSI_CLK_IntfPllClkReset();
   RSI_CLK_IntfPllClkEnable(M4CLK);
   RSI_CLK_IntfPLLTurnOn(M4CLK);
   RSI_CLK_IntfPLLTurnOff(M4CLK);
   RSI_CLK_I2sClkConfig(M4CLK,1 ,I2sPllClk ,2);
   RSI_CLK_PeripheralClkEnable2(M4CLK ,(SSI_MST_PCLK_ENABLE | PLL_INTF_CLK_ENABLE |
SSI_MST_SCLK_ENABLE));
#endif

#ifdef M4SOC
   RSI_CLK_M4SocClkConfig(M4CLK,1,M4UlpRefClk ,2);
   RSI_CLK_SocPllSetFreqDiv(M4CLK , 1,1,39,220, 0);
   RSI_CLK_M4SocClkConfig(M4CLK,1,M4SocPllClk ,3);
   RSI_CLK_M4SocClkConfig(M4CLK,1,M4ModemPllClk1 ,6);
   RSI_CLK_IntfPllSetFreqDiv(M4CLK,1,1,39,220,0);
   RSI_CLK_M4SocClkConfig(M4CLK,1,M4IntfPllClk ,4);
   RSI_CLK_SlpClkConfig(M4CLK, 1);
   RSI_CLK_M4SocClkConfig(M4CLK,1,M4SleepClk ,1);
   RSI_CLK_M4SocClkConfig(M4CLK,1,M4SocPllClk ,1);
   RSI_CLK_M4SocClkConfig(M4CLK,1,M4HostClk ,2);
   RSI_PS_PowerStateChangePs4toPs0();

#endif

#ifdef SSI

   RSI_CLK_EthernetClkConfig(M4CLK,1 ,1 ,ETH_SOC_PLL_CLK ,2);
   RSI_CLK_EthernetClkConfig(M4CLK,1 ,1 ,ETH_INTF_PLL_CLK ,4);
   RSI_CLK_EthernetClkConfig(M4CLK,1 ,1 ,ETH_SOC_PLL_CLK ,2);
   RSI_CLK_EthernetClkConfig(M4CLK,1 ,1 ,ETH_SOC_PLL_CLK ,2);
   RSI_CLK_SsiMstClkConfig(M4CLK ,ENABLE_STATIC_CLK,SSI_ULPREFCLK,2);
   RSI_CLK_SsiMstClkConfig(M4CLK ,ENABLE_STATIC_CLK ,SSI_SOCPLLCLK,3);
   RSI_CLK_SsiMstClkConfig(M4CLK ,ENABLE_STATIC_CLK ,SSI_MODEMPLLCLK1,4);
   RSI_CLK_SsiMstClkConfig(M4CLK ,ENABLE_STATIC_CLK ,SSI_INTFPLLCLK,3);
```

```
    RSI_CLK_SsiMstClkConfig(M4CLK,ENABLE_STATIC_CLK ,SSI_MODELPLLCLK2,2);
    RSI_CLK_SsiMstClkEnable(M4CLK,ENABLE_STATIC_CLK);
#endif
#endif

#ifdef SDMEM
    RSI_CLK_SdMemClkConfig(M4CLK,1 ,SDMEM_SOCPLLCLK ,2);
    RSI_CLK_SdMemClkConfig(M4CLK,1,SDMEM_MODEMPLLCLK1 ,3);
    RSI_CLK_SdMemClkConfig(M4CLK,1 ,SDMEM_INTFPLLCLK ,2);

#endif

#ifdef CT
    RSI_CLK_SctClkConfig(M4CLK ,CT_ULPREFCLK ,2,ENABLE_STATIC_CLK);
    RSI_CLK_SctClkConfig(M4CLK ,CT_INTFPLLCLK ,3,ENABLE_STATIC_CLK);
    RSI_CLK_SctClkConfig(M4CLK ,CT_SOCPLLCLK ,4,ENABLE_STATIC_CLK);
    RSI_CLK_SctClkConfig(M4CLK ,M4_SOCCLKFOROTHERCLKSSCT ,2,ENABLE_STATIC_CLK);
#endif
#ifdef CCI

    RSI_CLK_CciClkConfig(M4CLK,CCI_M4_SOC_CLK_FOR_OTHER_CLKS,2);
    RSI_CLK_CciClkConfig(M4CLK,CCI_INTF_PLL_CLK,3);
    RSI_CLK_CciClkConfig(M4CLK,CCI_M4_SOC_CLK_NO_SWL_SYNC_CLK_TREE,4);
#endif
while(1);
}
```

Code Block 8 M4ss clock example

14.3 API Descriptions

14.3.1 RSI_CLK_SocPllClkBypassEnable

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_CLK_SocPllClkBypassEnable(boolean_t clkEnable)
```

Description

This API is used to Enable bypass clock.

Parameter

Parameter	Description
clkEnable	Enum value to enable or disable the clock Enable (1) : Enables bypass clock Disable (0) : Disables bypass clock

Return values

Returns zero on success ,on failure return error code.

Example

```
#define Enable   1
#define Disable   0

/* Enable soc pll bypass clock */
RSI_CLK_SocPllClkBypassEnable(Enable);
```

```
/* Disable soc pll bypass clock */
RSI_CLK_SocPllClkBypassEnable(Disable);
```

14.3.2 RSI_CLK_IntfPLLClkBypassEnable

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t  RSI_CLK_IntfPLLClkBypassEnable(boolean_t clkEnable)
```

Description

This API is used to Enable bypass clock.

Parameter

Parameter	Description
clkEnable	enum value to enable or disable the clock Enable : Enables bypass clock Disable : Disables bypass clock

Return values

Returns zero on success ,on failure return error code.

Example

```
#define   Enable      1
#define   Disable     0
/* enable intfpll bypass clock */
RSI_CLK_IntfPLLClkBypassEnable(Enable);

/* disable intfpll bypass clock */
RSI_CLK_IntfPLLClkBypassEnable(Disable);
```

14.3.3 RSI_CLK_CanClkConfig

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

Description

This API is used to configure the Can clocks

```
STATIC INLINE error_t RSI_CLK_CanClkConfig(M4Clk_Type *pCLK ,    uint32_t
divFactor,CLK_ENABLE_T clkType)
```

Parameters

Parameter	Description
pCLK	Pointer to the pll register instance

Parameter	Description
divFactor	Division value for Can Clock,Maximum clock division factor for CAN is 255.
clkType	Enum value to select static clock or dynamic clock. See #CLK_ENABLE_T for more info possible enum value is below • ENABLE_DYN_CLK • ENABLE_STATIC_CLK

Return values

Returns zero on success ,on failure return error code.

Example

```
/* CAN clock configuration */
RSI_CLK_CanClkConfig(M4CLK ,1,ENABLE_STATIC_CLK);
```

14.3.4 RSI_CLK_SocPllClkEnable

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_CLK_SocPllClkEnable(boolean_t clkEnable)
```

Description

This API is used to Enable the SoC-PLL output clock

Parameter

Parameter	Description
clkEnable	Enum value to enable or disable the clock Enable : Enables clock Disable : Disables clock

Return values

Returns zero on success ,on failure return error code.

Example

```
#define Enable   1
#define Disable   0

/* Enable soc pll clock */
RSI_CLK_SocPllClkEnable(Enable);

/* Disable soc pll clock */
RSI_CLK_SocPllClkEnable(Disable);
```

14.3.5 RSI_CLK_SocPllPdEnable

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_CLK_SocPllPdEnable(boolean_t en)
```

Description

This API is used to Enable the PdEnable(power down).

Parameter

Parameter	Description
en	Enable or disable the PdEnable Enable : Enables power down mode Disable : Disables power down mode

Return values

Returns zero on success ,on failure Return error code.

Example

```
#define Enable   1
#define Disable   0

/* Enable power down mode */
RSI_CLK_SocPllPdEnable(Enable);

/* Disable power down mode */
RSI_CLK_SocPllPdEnable(Disable);
```

14.3.6 RSI_CLK_SocPllTurnOn

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_CLK_SocPllTurnOn()
```

Description

This API is used to TurnOn the SOC_PLL.

Parameter

None

Return values

Returns zero on success ,on failure return error code.

Example

```
/* turn off pll clock */
RSI_CLK_SocPllTurnOn();
```

14.3.7 RSI_CLK_SocPllTurnOff

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_CLK_SocPllTurnOff()
```

Description

This API is used to TurnOff the SOC_PLL.

Parameter

None

Return values

Returns zero on success ,on failure return error code.

Example

```
/* turn off soc pll */
RSI_CLK_SocPllTurnOff();
```

14.3.8 RSI_CLK_I2sPllTurnOff

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_CLK_I2sPllTurnOff()
```

Description

This API is used to TurnOff the I2s_PLL.

Parameter

None

Return values

Returns zero on success ,on failure return error code.

Example

```
/* Turn off i2s pll */
RSI_CLK_I2sPllTurnOff();
```

14.3.9 RSI_CLK_I2sPllClkReset

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

Description

This API is used to reset the I2s_pll_clk.

```
STATIC INLINE error_t  RSI_CLK_I2sPllClkReset()
```

Parameter

None

Return value

Returns zero on success ,on failure return error code.

Example

```
/* pll clock reset */
RSI_CLK_I2sPllClkReset(M4CLK);
```

14.3.10 RSI_CLK_I2sPllTurnOn

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_CLK_I2sPllTurnOn()
```

Description

This API is used to TurnOn the I2s_PLL

Parameter

None

Return values

Returns zero on success ,on failure return error code.

Example

```
/* Turn on i2s */
RSI_CLK_I2sPllTurnOn();
```

14.3.11 RSI_CLK_I2sPllSetFreqDiv

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_CLK_I2sPllSetFreqDiv(M4Clk_Type *pCLK,uint16_t
u16DivFactor1,uint16_t u16DivFactor2,uint16_t nFactor,uint16_t mFactor, uint16_t fcwF)
```

Description

This API is used to divide I2s_PLL Clock.

Parameters

Parameter	Description
pCLK	Pointer to the pll register instance
u16DivFactor	Post Division factor1
u16DivFactor2	Post Division factor2
nFactor	N factor for PLL
mFactor	M factor for PLL
fcwF	Fractional Frequency Control Word

Return values

Returns zero on success ,on failure return error code.

Example

```
RSI_CLK_I2sPllSetFreqDiv(M4CLK,2,3,3,5,10);
```

14.3.12 RSI_CLK_I2sPllClkEnable

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_CLK_I2sPllClkEnable(boolean_t clkEnable)
```

Description

This API is used to Enable the I2s_PLL output clock.

Parameter

Parameter	Description
clkEnable	Enum value to enable or disable the clock Enable : Enables clock for i2s Disable : Disables clock for i2s

Return values

Returns zero on success ,on failure return error code.

Example

```
#define Enable   1
#define Disable   0

/* Enable i2s pll clock */
RSI_CLK_I2sPllClkEnable(Enable);

/* Disable i2s pll clock */
RSI_CLK_I2sPllClkEnable(Disable);
```

14.3.13 RSI_CLK_IntfPllClkReset

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\library\systemlevel\src\

Prototype

```
STATIC INLINE error_t  RSI_CLK_IntfPllClkReset()
```

Description

This API is used to Reset the Intf_pll_clk.

Parameter

None

Return values

Returns zero on success ,on failure return error code.

Example

```
/* reset clock*/
RSI_CLK_IntfPllClkReset();
```

14.3.14 RSI_CLK_IntfPllClkEnable

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_CLK_IntfPllClkEnable(boolean_t clkEnable)
```

Description

This API is used to Enable the Intf_PLL output clock.

Parameter

Parameter	Description
clkEnable	Enum value to enable or disable the clock Enable(1) : Enables clock Disable(0) : Disables clock

Return values

Returns zero on success ,on failure return error code.

Example

```
#define Enable   1
#define Disable   0

/* intf pll clock enble */
RSI_CLK_IntfPllClkEnable(Enable);

/* intf pll clock disable */
RSI_CLK_IntfPllClkEnable(Disable);
```

14.3.15 RSI_CLK_IntfPLLTurnOn

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t  RSI_CLK_IntfPLLTurnOn()
```

Description

This API is used to TurnOn the Intf_PLL.

Parameter

None

Return values

Returns zero on success ,on failure return error code.

Example

```
RSI_CLK_IntfPLLTurnOn();
```

14.3.16 RSI_CLK_IntfPLLTurnOn

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
error_t  RSI_CLK_IntfPLLTurnOn()
```

Description

This API is used to TurnOn the Intf_PLL.

Parameter

None

Return values

Returns zero on success ,on failure return error code.

Example

```
RSI_CLK_IntfPLLTurnOn();
```

14.3.17 RSI_CLK_IntfPLLTurnOff

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t  RSI_CLK_IntfPLLTurnOff()
```

Description

This API is used to TurnOff the Intf_PLL.

Parameter

None

Return values

Returns zero on success ,on failure return error code.

Example

```
/* turn off intf pll clock */
RSI_CLK_IntfPLLTurnOff();
```

14.3.18 RSI_CLK_I2sClkConfig

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

Description

```
STATIC INLINE error_t RSI_CLK_I2sClkConfig(M4Clk_Type *pCLK ,I2S_CLK_SRC_SEL_T clkSource ,
        uint32_t divFactor)
```

This API is used to configure the I2S clocks

Parameters

Parameter	Description
pCLK	Pointer to the pll register instance
clkSource	Enum valuse for I2S clock sources to be selected. see possible I2S clock sources at #I2S_CLK_SRC_SEL • I2S_PLLCLK • I2S_M4SOCCLKFOROTHERS
divFactor	Division value for I2S Clock ,Maximum clock division factor for I2S is 63.

Return values

Returns zero on success ,on failure return error code.

Example

```
/* i2s clock configire */
RSI_CLK_I2sClkConfig(M4CLK,I2S_M4SOCCLKFOROTHERS ,3);
```

14.3.19 RSI_CLK_PeripheralClkEnable2

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t  RSI_CLK_PeripheralClkEnable2(M4Clk_Type *pCLK ,uint32_t flags)
```

Description

This API is used to enable the peripheral cloks for SET2 register

Parameters

Parameter	Description
pCLK	Pointer to the pll register instance
flags	Ored values of peripheral bits to be enable. select flag between below flag list: • GEN_SPI_MST1_HCLK_ENABLE • CAN1_PCLK_ENABLE • CAN1_CLK_ENABLE • UDMA_HCLK_ENABLE • I2C_BUS_CLK_ENABLE • I2C_2_BUS_CLK_ENABLE • SSI_SLV_PCLK_ENABLE • SSI_SLV_SCLK_ENABLE • QSPI_CLK_ENABLE • QSPI_HCLK_ENABLE • I2SM_INTF_SCLK_ENABLE • I2SM_PCLK_ENABLE • QE_PCLK_ENABLE • MCPWM_PCLK_ENABLE • SGPIO_PCLK_ENABLE • EGPIO_PCLK_ENABLE • ARM_CLK_ENABLE • SSI_MST_PCLK_ENABLE • SSI_MST_SCLK_ENABLE • MEM2_CLK_ENABLE • MEM_CLK_ULP_ENABLE • ROM_CLK_ENABLE • PLL_INTF_CLK_ENABLE • SEMAPHORE_CLK_ENABLE • TOT_CLK_ENABLE • RMII_SOFT_RESET

Return values

Returns zero on success ,on failure return error code.

Example

```
RSI_Clk_PeripheralClkEnable2(M4CLK ,(GEN_SPI_MST1_HCLK_ENABLE | SSI_MST_PCLK_ENABLE));
```

14.3.20 RSI_CLK_M4SocClkConfig

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_CLK_M4SocClkConfig(M4Clk_Type *pCLK ,M4_SOC_CLK_SRC_SEL_T
clkSource ,
        uint32_t divFactor)
```

Description

This API is used to configure the m4_soc clocks

Parameters

Parameter	Description
pCLK	Pointer to the pll register instance
clkSource	To select the pheripheral clock or processor clock possible enum value is below • M4_ULPREFCLK • M4_RESERVED • M4_SOCPLLCLK • M4_MODEMPLLCLK1 • M4_INTFPLLCLK • M4_SLEEPCLK=5,
divFactor	division value for M4SOC clock

Return values

Returns zero on success ,on failure return error code.

Example

```
RSI_CLK_M4SocClkConfig(M4CLK,M4_SLEEPCLK,3);
```

Precondition

1.For using UlpRefClk clksource need to configure M4ssRefClk frequency. For that need to call RSI_CLK_M4ssRefClkConfig Api first
2.For using SocPllCLK clksource need to configure SocPll frequency. For that need to call RSI_CLK_SetSocPllFreq Api first
3.For using IntfPllCLK clksource need to configure IntfPll frequency. For that need to call RSI_CLK_SetIntfPllFreq Api first
4.For using Sleep clksource need to configure Sleep Clock. For that need to call RSI_CLK_SlpClkConfig Api first
5.For using Sleep clksource need to configure Sleep Clock. For that need to call RSI_CLK_SlpClkConfig Api first

14.3.21 RSI_CLK_SocPllSetFreqDiv

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_CLK_SocPllSetFreqDiv(M4Clk_Type *pCLK , boolean_t
clk_en,uint16_t
        divFactor,uint16_t nFactor,uint16_t mFactor,uint16_t fCwf,
        uint16_t dcofixsel,uint16_t ldoprog)
```

Description

This API is used to set the Soc PLL clock to particular frequency

Parameters

Parameter	Description
pCLK	Pointer to the pll register instance
clk_en	To enble the soc_pll_clk output enable
divFactor	PLL post division factor
nFactor	N factor of PLL
mFactor	M factor of PLL
fCwf	Fractional Frequency Control Word. For below 200MHz fcwF is 0 and above 200Mhz if the frequency is odd program FCW_F as 8192
dcofixsel	Dco Fixed Ring select
ldoprog	SOCPLL LDO output voltage select

Return values

Returns zero (0) on success ,on failure return error code.

Example

```
/* set frequency division */
RSI_CLK_SocPllSetFreqDiv(M4CLK , 1,1,3,5,220,0,5);
```

Note

For <= 200Mhz ---> ldo_prog =4 and dco_fix_sel=1

For 201-250Mhz ---> ldo_prog =5 and dco_fix_sel=0

For >=251Mhz ---> ldo_prog =5 and dco_fix_sel=2

14.3.22 RSI_CLK_IntfPllSetFreqDiv

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t  RSI_CLK_IntfPllSetFreqDiv(M4Clk_Type *pCLK , boolean_t
clk_en,uint16_t divFactor,uint16_t nFactor,uint16_t mFactor,uint16_t fcwF,uint16_t
```

```
dcoFixSel,uint16_t ldoProg)
```

Description

This API is used to divide the Intf PLL clock frequency.

Parameters

Parameter	Description
pCLK	Pointer to the pll register instance
clk_en	Enble the intf_pll_clk output enable
divFactor	PLL post division factor
nFactor	N factor of PLL
mFactor	M factor of PLL
fcwF	Fractional Frequency Control Word
dcoFixSel	Dco Fixed Ring select
ldoProg	INTFPLL LDO output voltage select

Return values

Returns zero on success ,on failure return error code.

Example

```
/* divide pll frequency */
RSI_CLK_IntfPllSetFreqDiv(M4CLK,1,1,3,5,200,1,4)
```

Note

For <= 200Mhz ---> ldo_prog =4 and dco_fix_sel=1

For 201-250Mhz ---> ldo_prog =5 and dco_fix_sel=0

For >=251Mhz ---> ldo_prog =5 and dco_fix_sel=2

14.3.23 RSI_CLK_SlpClkConfig

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_CLK_SlpClkConfig(M4Clk_Type *pCLK ,  SLEEP_CLK_SRC_SEL_T clkSrc)
```

Description

This API is used to configure the SLEEP Clocks

Parameters

Parameter	Description

Parameter	Description
pCLK	Pointer to the pll register instance
clkSrc	Enum values to select the clock sources for sleep clock. possible enum value is below • SLP_ULP_32KHZ_RC_CLK • SLP_ULP_32KHZ_XTAL_CLK • SLP_CLK_GATED • SLP_ULP_32KHZ_RO_CLK

Return values

Returns zero on success, on failure return error code.

Example

```
/* sleep clock */
RSI_CLK_SlpClkConfig(M4CLK,SLP_ULP_32KHZ_XTAL_CLK );
```

14.3.24 RSI_CLK_EthernetClkConfig

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_CLK_EthernetClkConfig(M4Clk_Type *pCLK  ,boolean_t swalloEn
,ETHERNET_CLK_SRC_SEL_T clkSource,
        uint32_t divFactor)
```

Description

This API is used to configure the Ethernet clocks

Parameters

Parameter	Description
pCLK	Pointer to the pll register instance
swalloEn	Enable or disable the swallo functionality • 1 : Swallo enabled • 0 : Swallo disabled
clkSource	Enum valuse for PLL_Intf clock sources to be selected. possible enum value is below • ETH_SOC_PLL_CLK, • ETH_INTF_PLL_CLK,
divFactor	Ethernet clock division value,Maximum clock division factor for Ethernet is 15.

Return values

Returns zero on success ,on failure return error code.

Example

```
/* ethernet clock cofigure */
RSI_CLK_EthernetClkConfig(M4CLK,1,ETH_INTF_PLL_CLK ,4);
```

14.3.25 RSI_CLK_SsiMstClkConfig

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_CLK_SsiMstClkConfig(M4Clk_Type *pCLK ,CLK_ENABLE_T clkType
,SSI_MST_CLK_SRC_SEL_T clkSource ,
        uint32_t   divFactor)
```

Description

This API is used to configure the SSI clocks

Parameters

Parameter	Description
pCLK	Pointer to the pll register instance
clkType	Enum value to select static clock or dynamic clock possible enum value is below • ENABLE_DYN_CLK • ENABLE_STATIC_CLK
clkSource	Enum values for SSI clock sources to be selected possible enum value is below • SSI_ULPREFCLK • SSI_SOCPLLCLK • SSI_MODEMPLLCLK1 • SSI_INTFPLLCLK • SSI_MODELPLLCLK2 • M4_SOCCLKFOROTHERCLKS
divFactor	The division value for SSI Clock,Maximum clock division factor for SSI is 15.

Return values

Returns zero on success ,on failure return error code.

Example

```
RSI_CLK_SsiMstClkConfig(M4CLK,ENABLE_STATIC_CLK ,SSI_MODELPLLCLK2,2);
```

14.3.26 RSI_CLK_SdMemClkConfig

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_CLK_SdMemClkConfig(M4Clk_Type *pCLK ,boolean_t swalloEn
,SDMEM_CLK_SRC_SEL_T clkSource ,
        uint32_t divFactor)
```

Description

This API is used to configure the SdMem clocks

Parameters

Parameter	Description
pCLK	Pointer to the pll register instance
swalloEn	Enable or disable the swallo functionality • 1 : swalloEn enabled • 0 : swalloEn disabled
clkSource	Enum values for SdMem clock sources to be selected possible enum value is below • SDMEM_SOCPLLCLK • SDMEM_MODEMPLLCLK1 • SDMEM_INTFPLLCLK • M4_SOCCLKFOROTHERCLKSSDMEM
divFactor	Division value for SdMem Clock,Maximum clock division factor for SDMEM is 63.

Return values

Returns zero on success ,on failure return error code.

Example

```
RSI_CLK_SdMemClkConfig(M4CLK,1 ,SDMEM_INTFPLLCLK ,2);
```

14.3.27 RSI_CLK_CtClkConfig

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_CLK_CtClkConfig(M4Clk_Type *pCLK ,CT_CLK_SRC_SEL_T clkSource
,uint32_t divFactor,
        CLK_ENABLE_T clkType)
```

Description

This API is used to configure the CT clocks

Parameters

Parameter	Description

Parameter	Description
pCLK	Pointer to the pll register instance
clkSource	Enum values for CT clock sources to be selected. see possible CT clock sources at CT_CLK_SRC_SEL_T possible enum value is below • CT_ULPREFCLK • CT_INTFPLLCLK • CT_SOCPLLCLK • M4_SOCCLKFOROTHERCLKSCT
divFactor	Division value for CT Clock,Maximum clock division factor for CT is 63.
clkType	Enum value to select static clock or dynamic clock possible enum value is below • ENABLE_DYN_CLK • ENABLE_STATIC_CLK

Return values

Returns zero on success ,on failure return error code.

Example

```
/* CT clock configuration */
RSI_CLK_CtClkConfig(M4CLK,CT_ULPREFCLK,1,ENABLE_STATIC_CLK);
```

14.3.28 RSI_CLK_CciClkConfig

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_CLK_CciClkConfig(M4Clk_Type *pCLK ,CCI_CLK_SRC_SEL_T clkSource
,uint32_t divFactor,
        CLK_ENABLE_T clkType)
```

Description

This API is used to configure the CCI clocks

Parameters

Parameter	Description
pCLK	Pointer to the pll register instance
clkSource	Enum values for CCI clock sources to be selected. see possible CCI clock sources at #CCI_CLK_SRC_SEL_T • CCI_M4_SOC_CLK_FOR_OTHER_CLKS • CCI_INTF_PLL_CLK • CCI_M4_SOC_CLK_NO_SWL_SYNC_CLK_TREE

Parameter	Description
divFactor	Division value for CCI Clock,Maximum clock division factor for CCI is 15.
clkType	Enum value to select static clock or dynamic clock possible enum value is below • ENABLE_DYN_CLK • ENABLE_STATIC_CLK

Return values

Returns zero on success ,on failure return error code.

Example

```
/* CCI clock configuration */
RSI_CLK_CciClkConfig(M4CLK,CCI_M4_SOC_CLK_NO_SWL_SYNC_CLK_TREE,1,ENABLE_STATIC_CLK);
```

14.3.29 RSI_CLK_CheckPllLock

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\\

Prototype

```
STATIC INLINE boolean_t RSI_CLK_CheckPllLock(PLL_TYPE_T  pllType)
```

Description

This API is used to check the lock status of pll

Parameter

Parameter	Description
pllType	To select the soc_pll, intf_pll and i2s_pll. value as parameter: • SOC_PLL : soc_pll clk • INTF_PLL : intf_pll clk • I2S_PLL : i2s_pll clk

Return values

Return 1 then for lock status high(enable) and return 0 then for lock status low(disable).

Example

```
boolean_t status;
/* pll lock status */
RSI_CLK_CheckPllLock(SOC_PLL);
```

14.3.30 RSI_CLK_SetSocPllFreq

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_CLK_SetSocPllFreq(M4Clk_Type *pCLK,uint32_t socPllFreq,uint32_t
pllRefClk)
```

Description

This API is used to set the Soc PLL clock to particular frequency

Parameters

Parameter	Description
pCLK	Pointer to the pll register instance
socPllFreq	Frequency value in Mhz for Soc_Pll_Clk
pllRefClk	Frequency value in Mhz for Reference clk

Return values

Returns zero on success ,on failure return error code.

Example

```
/* set frequency */
RSI_CLK_SetSocPllFreq(M4CLK,128000000,40000000);
```

Note

Only 1Mhz steps applicable to the this API, 0.96Mhz steps are not supported.

14.3.31 RSI_CLK_SocPllClkSet

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_CLK_SocPllClkSet(M4Clk_Type *pCLK
```

Description

This API is used to Enables the SoC-PLL.

Parameter

Parameter	Description
pCLK	Pointer to the pll register instance

Return values

Returns zero on success ,on failure return error code.

Example

```
/* Enable soc-pll */
RSI_CLK_SocPllClkSet(M4CLK);
```

14.3.32 RSI_CLK_SocPllClkReset

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_CLK_SocPllClkReset()
```

Description

This API is used to Reset the Soc_pll_clk.

Parameter

None

Return values

Returns zero on success ,on failure return error code.

Example

```
/* pll clock reset */
RSI_CLK_SocPllClkReset();
```

14.3.33 RSI_CLK_I2sPllClkBypassEnable

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_CLK_I2sPllClkBypassEnable(boolean_t clkEnable)
```

Description

This API is used to Enable bypass clock.

Parameter

Parameter	Description
clkEnable	Enum value to enable or disable the clock Enable : Enables bypass clock for i2s Disable : Disables bypass clock for i2s

Return values

Returns zero on success ,on failure return error code.

Example

```
#define Enable   1
#define Disable   0

/* Enable i2s pll bypass clock */
RSI_CLK_I2sPllClkBypassEnable(Enable);

/* Disable i2s pll bypass clock */
```

```
RSI_CLK_I2sPllClkBypassEnable(Disable);
```

14.3.34 RSI_CLK_I2sPllPdEnable

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_CLK_I2sPllPdEnable(boolean_t en)
```

Description

This API is used to Enable the PdEnable(power down) for I2S.

Parameter

Parameter	Description
en	Enable or disable the PdEnable for i2s Enable : Enables power down mode Disable : Disables power down mode

Return values

Returns zero on success ,on failure return error code.

Example

```
#define Enable   1
#define Disable   0

/* Power down enable for i2s */
RSI_CLK_I2sPllPdEnable(Enable);

/* Power down disable for i2s */
RSI_CLK_I2sPllPdEnable(Disable);
```

14.3.35 RSI_CLK_SetI2sPllFreq

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_CLK_SetI2sPllFreq(M4Clk_Type *pCLK,uint32_t i2sPllFreq, uint32_t
fXtal)
```

Description

This API is used to set the I2s_pll clock to particular frequency.

Parameters

Parameter	Description
pCLK	Pointer to the pll register instance
fXtal	Frequency value in Mhz for I2S_PLL Clk

Return values

Returns zero on success ,on failure return error code.

Example

```
/* set frequency */
RSI_CLK_SetI2sPllFreq(M4CLK,1411200,40000000);
```

14.3.36 RSI_CLK_I2sPllClkSet

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_CLK_I2sPllClkSet(M4Clk_Type *pCLK)
```

Description

This API is used to set the I2s_pll_clk.

Parameters

Parameter	Description
pCLK	Pointer to the pll register instance

Return values

Returns zero on success ,on failure return error code.

Example

```
/* pll clock set */
RSI_CLK_I2sPllClkSet(M4CLK);
```

14.3.37 RSI_CLK_IntfPllPdEnable

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_CLK_IntfPllPdEnable(boolean_t en)
```

Description

This API is used to Enable the PdEnable(power down).

Parameter

Parameter	Description
en	Enable or disable the PdEnable

Return values

Returns zero on success ,on failure return error code.

Example

```
#define Enable   1
#define Disable   0

/* intf pll power down enble */
RSI_CLK_IntfPllPdEnable(Enable);

/* intf pll power down disable */
RSI_CLK_IntfPllPdEnable(Disable);
```

14.3.38 RSI_CLK_SetIntfPllFreq

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_CLK_SetIntfPllFreq(M4Clk_Type *pCLK,uint32_t
intfPllFreq,uint32_t pllRefClk)
```

Description

This API is used to set the INTFPLL clock to particular frequency

Parameters

Parameter	Description
pCLK	Pointer to the pll register instance
intfPllFreq	Frequency value in Mhz for INTFPLL Clk
pllRefClk	Frequency value in Mhz for Reference clk

Return values

Returns zero on success ,on failure return error code.

Example

```
RSI_CLK_SetIntfPllFreq(M4CLK,1,40000000);
```

Note

Only 1Mhz steps applicable to the this API, 0.96Mhz steps are not supported

14.3.39 RSI_CLK_IntfPllClkSet

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

Description

This API is used to Enables the Intf-PLL.

Parameter

Parameter	Description
pCLK	Pointer to the pll register instance

Return values

Returns zero on success ,on failure return error code.

Example

```
/* set clock */
RSI_CLK_IntfPllClkSet(M4CLK);
```

14.3.40 RSI_CLK_PeripheralClkEnable1

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t  RSI_CLK_PeripheralClkEnable1(M4Clk_Type *pCLK ,uint32_t flags)
```

Description

This API is used to Enable the peripheral cloks for SET1 register.

Parameters

Parameter	Description
pCLK	Pointer to the pll register instance
flags	Ored values of peripheral bits to be enbled. select flag between below flag list: • USART1_PCLK_ENABLE • USART1_SCLK_ENABLE • USART2_PCLK_ENABLE • USART2_SCLK_ENABLE • CT_CLK_ENABLE • CT_PCLK_ENABLE • ICACHE_CLK_ENABLE • ICACHE_CLK_2X_ENABLE • RPDMA_HCLK_ENABLE • SOC_PLL_SPI_CLK_ENABLE • IID_CLK_ENABLE • SDIO_SYS_HCLK_ENABLE • CRC_CLK_ENABLE_M4 • M4SS_UM_CLK_STATIC_EN • ETH_HCLK_ENABLE

Parameter	Description
	• HWRNG_PCLK_ENABLE • GNSS_MEM_CLK_ENABLE • CCI_PCLK_ENABLE • CCI_HCLK_ENABLE • CCI_CLK_ENABLE • MASK_HOST_CLK_WAIT_FIX • MASK31_HOST_CLK_CNT • SD_MEM_INTF_CLK_ENABLE • MASK_HOST_CLK_AVAILABLE_FIX

Return values

Returns zero on success ,on failure return error code.

Example

```
RSI_Clk_PeripheralClkEnable1(M4CLK ,(USART1_PCLK_ENABLE | USART1_SCLK_ENABLE ));
```

14.3.41 RSI_CLK_PeripheralClkDisable1

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t  RSI_CLK_PeripheralClkDisable1(M4Clk_Type *pCLK ,uint32_t flags)
```

Description

This API is used to disable the peripheral cloks for CLR1 register

Parameters

Parameter	Description
pCLK	Pointer to the pll register instance
flags	Ored values of peripheral bits to be disable. select flag between below flag list: • USART1_PCLK_ENABLE • USART1_SCLK_ENABLE • USART2_PCLK_ENABLE • USART2_SCLK_ENABLE • CT_CLK_ENABLE • CT_PCLK_ENABLE • ICACHE_CLK_ENABLE • ICACHE_CLK_2X_ENABLE • RPDMA_HCLK_ENABLE • SOC_PLL_SPI_CLK_ENABLE

Parameter	Description
	• IID_CLK_ENABLE • SDIO_SYS_HCLK_ENABLE • CRC_CLK_ENABLE_M4 • M4SS_UM_CLK_STATIC_EN • ETH_HCLK_ENABLE • HWRNG_PCLK_ENABLE • GNSS_MEM_CLK_ENABLE • CCI_PCLK_ENABLE • CCI_HCLK_ENABLE • CCI_CLK_ENABLE • MASK_HOST_CLK_WAIT_FIX • MASK31_HOST_CLK_CNT • SD_MEM_INTF_CLK_ENABLE • MASK_HOST_CLK_AVAILABLE_FIX

Return values

Returns zero on success ,on failure return error code.

Example

```
RSI_Clk_PeripheralClkDisable1(M4CLK ,(USART1_PCLK_ENABLE | USART1_SCLK_ENABLE ));
```

14.3.42 RSI_CLK_PeripheralClkDisable2

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t  RSI_CLK_PeripheralClkDisable2(M4Clk_Type *pCLK ,uint32_t flags)
```

Description

This API is used to disable the peripheral cloks for CLR2 register

Parameters

Parameter	Description
pCLK	Pointer to the pll register instance
flags	Ored values of peripheral bits to be Disable. select flag between below flag list: • GEN_SPI_MST1_HCLK_ENABLE • CAN1_PCLK_ENABLE • CAN1_CLK_ENABLE • UDMA_HCLK_ENABLE • I2C_BUS_CLK_ENABLE

Parameter	Description
	• I2C_2_BUS_CLK_ENABLE • SSI_SLV_PCLK_ENABLE • SSI_SLV_SCLK_ENABLE • QSPI_CLK_ENABLE • QSPI_HCLK_ENABLE • I2SM_INTF_SCLK_ENABLE • I2SM_PCLK_ENABLE • QE_PCLK_ENABLE • MCPWM_PCLK_ENABLE • SGPIO_PCLK_ENABLE • EGPIO_PCLK_ENABLE • ARM_CLK_ENABLE • SSI_MST_PCLK_ENABLE • SSI_MST_SCLK_ENABLE • MEM2_CLK_ENABLE • MEM_CLK_ULP_ENABLE • ROM_CLK_ENABLE • PLL_INTF_CLK_ENABLE • SEMAPHORE_CLK_ENABLE • TOT_CLK_ENABLE • RMII_SOFT_RESET

Return values

Returns zero on success ,on failure return error code.

Example

```
RSI_Clk_PeripheralClkDisable2(M4CLK ,(GEN_SPI_MST1_HCLK_ENABLE | SSI_MST_PCLK_ENABLE));
```

14.3.43 RSI_CLK_PeripheralClkEnable3

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t  RSI_CLK_PeripheralClkEnable3(M4Clk_Type *pCLK ,uint32_t flags)
```

Description

This API is used to enable the peripheral cloks for SET3 register

Parameters

Parameter	Description
pCLK	Pointer to the pll register instance
flags	Ored values of peripheral bits to be enable. select flag between below flag list: • BUS_CLK_ENABLE • M4_CORE_CLK_ENABLE • CM_BUS_CLK_ENABLE • MISC_CONFIG_PCLK_ENABLE • EFUSE_CLK_ENABLE • ICM_CLK_ENABLE • MEM1_CLK_ENABLE • MEM3_CLK_ENABLE • USB_PHY_CLK_IN_ENABLE • QSPI_CLK_ONEHOT_ENABLE • QSPI_M4_SOC_SYNC • EGPIO_CLK_ENABLE • I2C_CLK_ENABLE • I2C_2_CLK_ENABLE • EFUSE_PCLK_ENABLE • SGPIO_PCLK_ENABLE • TASS_M4SS_64K_SWITCH_CLK_ENABLE • TASS_M4SS_128K_SWITCH_CLK_ENABLE • TASS_M4SS_SDIO_SWITCH_CLK_ENABLE • TASS_M4SS_USB_SWITCH_CLK_ENABLE • ROM_MISC_STATIC_ENABLE • M4_SOC_CLK_FOR_OTHER_ENABLE • ICACHE_ENABLE

Return values

Returns zero on success ,on failure return error code.

Example

```
RSI_Clk_PeripheralClkEnable3(M4CLK ,(M4_SOC_CLK_FOR_OTHER_ENABLE |
ROM_MISC_STATIC_ENABLE));
```

14.3.44 RSI_CLK_PeripheralClkDisable3

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t  RSI_CLK_PeripheralClkDisable3(M4Clk_Type *pCLK ,uint32_t flags)
```

Description

This API is used to disable the peripheral cloks for CLR3 register.

Parameters

Parameter	Description
pCLK	Pointer to the pll register instance
flags	Ored values of peripheral bits to be Disable. select flag between below flag list: • BUS_CLK_ENABLE • M4_CORE_CLK_ENABLE • CM_BUS_CLK_ENABLE • MISC_CONFIG_PCLK_ENABLE • EFUSE_CLK_ENABLE • ICM_CLK_ENABLE • MEM1_CLK_ENABLE • MEM3_CLK_ENABLE • USB_PHY_CLK_IN_ENABLE • QSPI_CLK_ONEHOT_ENABLE • QSPI_M4_SOC_SYNC • EGPIO_CLK_ENABLE • I2C_CLK_ENABLE • I2C_2_CLK_ENABLE • EFUSE_PCLK_ENABLE • SGPIO_PCLK_ENABLE • TASS_M4SS_64K_SWITCH_CLK_ENABLE • TASS_M4SS_128K_SWITCH_CLK_ENABLE • TASS_M4SS_SDIO_SWITCH_CLK_ENABLE • TASS_M4SS_USB_SWITCH_CLK_ENABLE • ROM_MISC_STATIC_ENABLE • M4_SOC_CLK_FOR_OTHER_ENABLE • ICACHE_ENABLE

Return values

Returns zero on success ,on failure return error code.

Example

```
RSI_Clk_PeripheralClkDisable3(M4CLK ,(M4_SOC_CLK_FOR_OTHER_ENABLE |
```

```
ROM_MISC_STATIC_ENABLE));
```

14.3.45 RSI_CLK_DynamicClkGateDisable

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

Description

This API is used to disable the dynamic clock gate for peripherals.

Parameters

Parameter	Description
pCLK	Pointer to the pll register instance
flags	Ored value of the register bits: • SDIO_SYS_HCLK_DYN_CTRL_DISABLE • BUS_CLK_DYN_CTRL_DISABLE • GPDMA_HCLK_DYN_CTRL_DISABLE • EGPIO_PCLK_DYN_CTRL_DISABLE • SGPIO_PCLK_DYN_CTRL_DISABLE • TOT_CLK_DYN_CTRL_DISABLE • HWRNG_PCLK_DYN_CTRL_DISABLE • USART1_SCLK_DYN_CTRL_DISABLE • USART1_PCLK_DYN_CTRL_DISABLE • USART2_SCLK_DYN_CTRL_DISABLE • USART2_PCLK_DYN_CTRL_DISABLE • SSI_SLV_SCLK_DYN_CTRL_DISABLE • SSI_SLV_PCLK_DYN_CTRL_DISABLE • I2SM_INTF_SCLK_DYN_CTRL_DISABLE • SEMAPHORE_CLK_DYN_CTRL_DISABLE • ARM_CLK_DYN_CTRL_DISABLE • SSI_MST_SCLK_DYN_CTRL_DISABLE • MEM1_CLK_DYN_CTRL_DISABLE • MEM2_CLK_DYN_CTRL_DISABLE • MEM_CLK_ULP_DYN_CTRL_DISABLE • MEM_CLK_ULP_DYN_CTRL_DISABLE • SSI_MST_PCLK_DYN_CTRL_DISABLE • ICACHE_DYN_GATING_DISABLE • CCI_PCLK_DYN_CTRL_DISABLE

Parameter	Description
	• MISC_CONFIG_PCLK_DYN_CTRL_DISABLE

Return values

Returns zero on success ,on failure return error code.

Example

```
RSI_Clk_DynamicClkGateDisable(M4CLK , (SDIO_SYS_HCLK_DYN_CTRL_DISABLE |
BUS_CLK_DYN_CTRL_DISABLE));
```

14.3.46 RSI_CLK_DynamicClkGateDisable2

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t  RSI_CLK_DynamicClkGateDisable2(M4Clk_Type *pCLK ,uint32_t  flags)
```

Description

This API is used to disable the dynamic clock gate for peripherals.

Parameters

Parameter	Description
pCLK	Pointer to the pll register instance
flags	Ored value of the register bits • SOC_PLL_SPI_CLK_DYN_CTRL_DISABLE • I2C_BUS_DYN_CTRL_DISABLE • I2C_2_BUS_CLK_DYN_CTRL_DISABLE • CT_PCLK_DYN_CTRL_DISABLE • CAN1_PCLK_DYN_CTRL_DISABLE • EFUSE_CLK_DYN_CTRL_DISABLE • EFUSE_PCLK_DYN_CTRL_DISABLE • PWR_CTRL_CLK_DYN_CTRL_DISABLE

Return values

Returns zero on success ,on failure return error code.

Example

```
 RSI_CLK_DynamicClkGateDisable2(M4CLK , (EFUSE_CLK_DYN_CTRL_DISABLE |
EFUSE_PCLK_DYN_CTRL_DISABLE));
```

14.3.47 RSI_CLK_DynamicClkGateEnable

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t  RSI_CLK_DynamicClkGateEnable(M4Clk_Type *pCLK ,uint32_t  flags)
```

Description

This API is used to enable the dynamic clock gate for peripherals

Parameters

Parameter	Description
pCLK	Pointer to the pll register instance
flags	Ored value of the register bits • SDIO_SYS_HCLK_DYN_CTRL_DISABLE • BUS_CLK_DYN_CTRL_DISABLE • GPDMA_HCLK_DYN_CTRL_DISABLE • EGPIO_PCLK_DYN_CTRL_DISABLE • SGPIO_PCLK_DYN_CTRL_DISABLE • TOT_CLK_DYN_CTRL_DISABLE • HWRNG_PCLK_DYN_CTRL_DISABLE • USART1_SCLK_DYN_CTRL_DISABLE • USART1_PCLK_DYN_CTRL_DISABLE • USART2_SCLK_DYN_CTRL_DISABLE • USART2_PCLK_DYN_CTRL_DISABLE • SSI_SLV_SCLK_DYN_CTRL_DISABLE • SSI_SLV_PCLK_DYN_CTRL_DISABLE • I2SM_INTF_SCLK_DYN_CTRL_DISABLE • SEMAPHORE_CLK_DYN_CTRL_DISABLE • ARM_CLK_DYN_CTRL_DISABLE • SSI_MST_SCLK_DYN_CTRL_DISABLE • MEM1_CLK_DYN_CTRL_DISABLE • MEM2_CLK_DYN_CTRL_DISABLE • MEM_CLK_ULP_DYN_CTRL_DISABLE • MEM_CLK_ULP_DYN_CTRL_DISABLE • SSI_MST_PCLK_DYN_CTRL_DISABLE • ICACHE_DYN_GATING_DISABLE • CCI_PCLK_DYN_CTRL_DISABLE • MISC_CONFIG_PCLK_DYN_CTRL_DISABLE

Return values

Returns zero on success ,on failure return error code.

Example

```
RSI_Clk_DynamicClkGateEnable(M4CLK , (SDIO_SYS_HCLK_DYN_CTRL_DISABLE |
BUS_CLK_DYN_CTRL_DISABLE));
```

14.3.48 RSI_CLK_DynamicClkGateEnable2

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t  RSI_CLK_DynamicClkGateEnable2(M4Clk_Type *pCLK ,uint32_t  flags)
```

Description

This API is used to enable the dynamic clock gate for peripherals

Parameters

Parameter	Description
pCLK	Pointer to the pll register instance
flags	Ored value of the register bits • SOC_PLL_SPI_CLK_DYN_CTRL_DISABLE • I2C_BUS_DYN_CTRL_DISABLE • I2C_2_BUS_CLK_DYN_CTRL_DISABLE • CT_PCLK_DYN_CTRL_DISABLE • CAN1_PCLK_DYN_CTRL_DISABLE • EFUSE_CLK_DYN_CTRL_DISABLE • EFUSE_PCLK_DYN_CTRL_DISABLE • PWR_CTRL_CLK_DYN_CTRL_DISABLE

Return values

Returns zero on success ,on failure return error code.

Example

```
RSI_Clk_DynamicClkGateEnable2(M4CLK , (EFUSE_CLK_DYN_CTRL_DISABLE |
EFUSE_PCLK_DYN_CTRL_DISABLE));
```

14.3.49 RSI_ULPSS_EnableRefClks

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_ULPSS_EnableRefClks(REF_CLK_ENABLE_T enable,  SRC_TYPE_T
srcType,cdDelay   delayFn)
```

Description

This API is used to enable the ULP reference clocks and provide delay for clock starting

Parameters

Parameter	Description
enable	To enable the particular reference clock possible enum value is below • MCU_ULP_40MHZ_CLK_EN • MCU_ULP_DOUBLER_CLK_EN • MCU_ULP_20MHZ_RING_OSC_CLK_EN • MCU_ULP_32MHZ_RC_CLK_EN • MCU_ULP_32KHZ_XTAL_CLK_EN • MCU_ULP_32KHZ_RO_CLK_EN • MCU_ULP_32KHZ_RC_CLK_EN
srcType	To select the pheripheral clock or processor clock
delayFn	Call back fuction used to create delay by using loops or timers in application code

Return values

Returns zero on success ,on failure return error code.

Example

```
RSI_ULPSS_EnableRefClks(MCU_ULP_32MHZ_RC_CLK_EN,ULP_PERIPHERAL_CLK,&delay);  /* delay is
call back function */
```

14.3.50 RSI_ULPSS_DisableRefClks

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_ULPSS_DisableRefClks(REF_CLK_ENABLE_T clk_type)
```

Description

This API is used to enable the ULP reference clocks and provide delay for clock starting

Parameters

Parameter	Description
enable	To enable the particular reference clock possible enum value is below • MCU_ULP_40MHZ_CLK_EN • MCU_ULP_DOUBLER_CLK_EN • MCU_ULP_20MHZ_RING_OSC_CLK_EN • MCU_ULP_32MHZ_RC_CLK_EN • MCU_ULP_32KHZ_XTAL_CLK_EN • MCU_ULP_32KHZ_RO_CLK_EN

Parameter	Description
	• MCU_ULP_32KHZ_RC_CLK_EN
srcType	To select the pheripheral clock or processor clock
delayFn	Call back fuction used to create delay by using loops or timers in application code

Return values

Returns zero on success ,on failure return error code.

Example

```
RSI_ULPSS_DisableRefClks(MCU_ULP_32MHZ_RC_CLK_EN);  /* delay is call back function */
```

14.3.51 RSI_CLK_M4ssRefClkConfig

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t  RSI_CLK_M4ssRefClkConfig(M4Clk_Type *pCLK ,M4SS_REF_CLK_SEL_T
clkSource)
```

Description

This API is used to configure the m4ss_ref clocks

Parameters

Parameter	Description
pCLK	Pointer to the pll register instance
clkSource	To select the pheripheral clock or processor clock possible enum value is below • ULP_32MHZ_RC_BYP_CLK • ULP_32MHZ_RC_CLK • RF_REF_CLK • MEMS_REF_CLK • ULP_20MHZ_RINGOSC_CLK • ULP_DOUBLER_CLK=6

Return values

Returns zero on success ,on failure return error code.

Example

```
RSI_CLK_M4ssRefClkConfig(M4CLK,ULP_DOUBLER_CLK);
```

14.3.52 RSI_CLK_QspiClkConfig

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_CLK_QspiClkConfig(M4Clk_Type *pCLK ,QSPI_CLK_SRC_SEL_T
clkSource,boolean_t swalloEn,
        boolean_t OddDivEn,uint32_t divFactor)
```

Description

This API is used to configure the Qspi clocks.

Parameters

Parameter	Description
pCLK	Pointer to the pll register instance
clkSource	Enum valuse for Qspi clock sources to be selected. see possible enum value for qspi clock sources • QSPI_ULPREFCLK • QSPI_MODELPLLCLK2 • QSPI_INTFPLLCLK=2 • QSPI_SOCPLLCLK=3 • M4_SOCCLKNOSWLSYNCCLKTREEGATED
swalloEn	To enable or disable the swallo functionality. See user manual for more info • 1 : swalloEn enabled • 0 : swalloEn disabled
OddDivEn	To enable or disable the odd div functionality. See user manual for more info • 1 : OddDivEn enabled • 0 : OddDivEn disabled
divFactor	Division value for Qspi Clock. Maximum value of qspi division factor is 63.

Return values

Returns zero on success ,on failure return error code.

Example

```
/* QSPI clock configure */
RSI_CLK_QspiClkConfig(M4CLK,QSPI_ULPREFCLK,1,1,6);
```

Precondition

1.For using UlpRefClk clksource need to configure M4ssRefClk frequency.

2.For that need to call RSI_CLK_M4ssRefClkConfig Api first

14.3.53 RSI_CLK_UsartClkConfig

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_CLK_UsartClkConfig(M4Clk_Type *pCLK ,CLK_ENABLE_T
clkType,boolean_t FracDivEn,
        EN_USART_T enUsart,USART_CLK_SRC_SEL_T clkSource,uint32_t divFactor)
```

Description

This API is used to configure the Usart clocks

Parameters

Parameter	Description
pCLK	Pointer to the pll register instance
clkType	Boolean value to enable or disable clock mode • 1 Enable : Enables the Usart clock • 0 Disable: Disables the Usart clock
FracDivEn	To enable or disable Fractional Division functionality • 1 : FracDivEn enabled • 0 : FracDivEn disabled
enUsart	Enum values for different Usart instances see possible bypass clock sources possible enum value is below • USART1=0 • USART2=1
clkSource	Enum values for Usart clock sources to be selected possible enum value is below • USART_ULPREFCLK • USART_SOCPLLCLK • USART_MODELPLLCLK2 • USART_INTFPLLCLK • M4_SOCCLKFOROTHERCLOCKS
divFactor	the division value for Usart Clock,Maximum value of usart clock division factor is 15.

Return values

Returns zero on success ,on failure return error code.

Example

```
/* USART clock configuration */
ROM_CLK_UsartClkConfig(M4CLK ,ENABLE_STATIC_CLK,1 ,USART2,M4_SOCCLKFOROTHERCLOCKS,2);
```

14.3.54 RSI_CLK_McuClkOutConfig

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_CLK_McuClkOutConfig(M4Clk_Type *pCLK ,MCU_CLKOUT_SRC_SEL_T
clkSource  ,
        uint32_t  divFactor)
```

Description

This API is used to configure the McuClkOut clocks

Parameters

Parameter	Description
pCLK	Pointer to the pll register instance
clkSource	Enum values of MCU_CLKOUT. possible enum value is below • MCUCLKOUT_ULP_32MHZ_RC_CLK • MCUCLKOUT_RF_REF_CLK • MCUCLKOUT_MEMS_REF_CLK • MCUCLKOUT_ULP_20MHZ_RINGOSC_CLK • MCUCLKOUT_ULP_DOUBLER_CLK • MCUCLKOUT_ULP_32KHZ_RC_CLK • MCUCLKOUT_ULP_32KHZ_XTAL_CLK • MCUCLKOUT_ULP_32KHZ_RO_CLK • MCUCLKOUT_INTF_PLL_CLK • MCUCLKOUT_MODEM_PLL_CLK1 • MCUCLKOUT_MODEM_PLL_CLK2 • MCUCLKOUT_SOC_PLL_CLK • MCUCLKOUT_I2S_PLL_CLK • MCUCLKOUT_USB_PLL_CLK,
divFactor	Division value for McuClkOut Clock,Maximum clock division factor for mcu clock out is 63.

Return values

Returns zero on success ,on failure return error code.

Example

```
RSI_CLK_McuClkOutConfig(M4CLK,MCUCLKOUT_USB_PLL_CLK,2);
```

14.3.55 RSI_CLK_M4SocClkDiv

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_CLK_M4SocClkDiv(M4Clk_Type *pCLK , uint32_t divFactor)
```

Description

This API is used to divide the M4soc clock

Parameters

Parameter	Description
pCLK	Pointer to the pll register instance
divFactor	M4Soc clock division value,Maximum clock division factor for M4 SOC clock is 63.

Return values

Returns zero on success ,on failure return error code.

Example

```
RSI_CLK_M4SocClkDiv(M4CLK,1);
```

14.3.56 RSI_CLK_QspiClkDiv

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_CLK_QspiClkDiv(M4Clk_Type *pCLK , boolean_t u8SwallowEn ,
boolean_t u8OddDivEn , uint32_t divFactor)
```

Description

This API is used to divide the QSPI clock

Parameters

Parameter	Description
pCLK	Pointer to the pll register instance
u8SwallowEn	To enable or disable the swallo functionality • 1 : Swallo enabled • 0 : Swallo disabled
u8OddDivEn	To enable or disable the odd division functionality • 1 : Odd division enabled • 0 : Odd division disabled
divFactor	QSPI clock division value,Maximum clock division factor for QSPI clock is 63.

Return values

Returns zero on success ,on failure return error code.

Example

```
/* QSPI clock division */
RSI_CLK_QspiClkDiv(M4CLK,1,0,2);
```

14.3.57 RSI_CLK_CtClkDiv

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_CLK_CtClkDiv(M4Clk_Type *pCLK , uint32_t divFactor)
```

Description

This API is used to divide the QSPI clock

Parameters

Parameter	Description
pCLK	Pointer to the pll register instance
divFactor	CT clock division value,Maximum clock division factor for CT clock is 63.

Return values

Returns zero on success ,on failure return error code.

Example

```
RSI_CLK_CtClkDiv(M4CLK,2);
```

14.3.58 RSI_CLK_SsiMstClkDiv

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_CLK_SsiMstClkDiv(M4Clk_Type *pCLK ,  uint32_t divFactor )
```

Description

This API is used to divide the SSI clock

Parameters

Parameter	Description
pCLK	Pointer to the pll register instance
divFactor	SSI clock division value,Maximum clock division factor for SSI master clock is 15.

Return values

Returns zero on success ,on failure return error code.

Example

```
/* ssi master clock division */
RSI_CLK_SsiMstClkDiv(M4CLK,2);
```

14.3.59 RSI_CLK_CciClkDiv

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_CLK_CciClkDiv(M4Clk_Type *pCLK ,  uint32_t divFactor)
```

Description

This API is used to divide the CCI clock

Parameters

Parameter	Description
pCLK	Pointer to the pll register instance
divFactor	CCI clock division value,Maximum clock division factor for CCI clock is 15.

Return values

Returns zero on success ,on failure return error code.

Example

```
/* CCI clock division */
RSI_CLK_CciClkDiv(M4CLK,2);
```

14.3.60 RSI_CLK_I2sClkDiv

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_CLK_I2sClkDiv(M4Clk_Type *pCLK ,  uint32_t divFactor)
```

Description

This API is used to divide the I2S clock

Parameters

Parameter	Description
pCLK	Pointer to the pll register instance
divFactor	I2S clock division value,Maximum clock division factor for I2s clock is 63.

Return values

Returns zero on success ,on failure return error code.

Example

```
/* i2s division clock */
RSI_CLK_I2sClkDiv(M4CLK,1);
```

14.3.61 RSI_CLK_SdmemClkDiv

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_CLK_SdmemClkDiv(M4Clk_Type *pCLK , boolean_t u8SwallowEn ,
uint32_t divFactor)
```

Description

This API is used to divide the SDMEM clock

Parameters

Parameter	Description
pCLK	Pointer to the pll register instance
u8SwallowEn	Enable or disable clock sawllowing.
divFactor	SDMEM clock division value,Maximum clock division factor for SDMEM clock is 63.

Return values

Returns zero on success, on failure return error code.

Example

```
/* divide SDMEM clock */
RSI_CLK_SdmemClkDiv(M4CLK,1,1);
```

14.3.62 RSI_CLK_UsartClkDiv

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_CLK_UsartClkDiv(M4Clk_Type *pCLK , EN_USART_T EN_USART_T ,
uint8_t u8FracDivEn, uint32_t divFactor)
```

Description

This API is used to divide the USART/UART clock

Parameters

Parameter	Description
pCLK	Pointer to the pll register instance
EN_USART_T	Enum of uart numbers.See#EN_USART_T for more information enum value is below • USART1 • USART2
u8FracDivEn	To enable or disable fractional division feature • 1 : Clock Swallo type divider is selected

Parameter	Description
	• 0 : Fractional Divider typed is selected
divFactor	USART/UART clock division value,Maximum clock division factor for USART clock is 15.

Return values

Returns zero on success, on failure return error code.

Example

```
/* UASRT clock division */
RSI_CLK_UsartClkDiv(M4CLK,USART1,1,1);
```

14.3.63 RSI_CLK_SlpClkCalibConfig

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE uint32_t RSI_CLK_SlpClkCalibConfig(M4Clk_Type *pCLK ,  uint8_t clkCycles)
```

Description

This API is used to calibrate the sleep clock

Parameters

Parameter	Description
pCLK	Pointer to the pll register instance
clkCycles	clkCycles : These bits are used to program the number of clock cycles over which clock calibration is to be done. • 00 => 1 Cycle • 01 => 2 Cycles • 10 => 3 Cycles • 11 => 4 Cycles

Return values

Returns the calibration duration.

Example

```
uint32_t calib_sleep;

/* calibrate sleep clock value */
calib_sleep=RSI_CLK_SlpClkCalibConfig(M4CLK,0);
```

14.3.64 RSI_CLK_GspiClkConfig

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_CLK_GspiClkConfig(M4Clk_Type *pCLK ,  GSPI_CLK_SRC_SEL_T clkSel
)
```

Description

This API is used to configure the GSPI Clocks

Parameters

Parameter	Description
pCLK	Pointer to the pll register instance
clkSel	Enum values to select the clock sources. possible values of GSPI_CLK_SRC_SEL_T below • GSPI_M4_SOC_CLK_FOR_OTHER_CLKS • GSPI_ULP_REF_CLK • GSPI_SOC_PLL_CLK • GSPI_MODEM_PLL_CLK2 • GSPI_INTF_PLL_CLK

Return values

Returns zero on success, on failure return error code.

Example

```
/* gspi clock configure */
RSI_CLK_GspiClkConfig(M4CLK,GSPI_INTF_PLL_CLK);
```

14.3.65 RSI_CLK_I2CClkConfig

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_CLK_I2CClkConfig(M4Clk_Type *pCLK , boolean_t clkEnable,EN_I2C_T
enI2C)
```

Description

This API is used to configure the I2C clock.

Parameters

Parameter	Description
pCLK	Pointer to the pll register instance
clkEnable	Boolean value to enable or disable clock mode • clkEnable : Enables the I2C clock • Disable : Disables the I2C clock
enI2C	Possible enum values is below • I2C1_INSTAN=0

Parameter	Description
	• I2C2_INSTAN=1

Return values

Returns zero on success, on failure return error code.

Example

```
#define    Enable        1
RSI_CLK_I2CClkConfig(M4CLK,Enable,I2C1_INSTAN);
```

14.3.66 RSI_CLK_XtalClkConfig

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_CLK_XtalClkConfig(uint8_t xtalPin)
```

Description

This API is used to configure the Xtal clock

Parameter

Parameter	Description
xtalPin	Pin number of NPSS_GPIO. Possible values are 0,1,2,3,4

Return values

Returns zero on success, on failure return error code.

Example

```
/* configure Xtal clock */
RSI_CLK_XtalClkConfig(1);
```

14.3.67 RSI_CLK_USBClkConfig

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_CLK_USBClkConfig(M4Clk_Type *pCLK ,USB_CLK_SRC_SEL_T clkSource
,uint16_t divFactor)
```

Description

This API is used to configure the USB clock

Parameters

Parameter	Description

Parameter	Description
0	Pointer to the pll register instance
clkSource	Different clock sources for USB_PHY_CLK. possible clock enum value is below • USB_MEMS_REF_CLK • USB_REFERENCE_CLK • USB_PLL_CLK
divFactor	USB clock division value

Return values

Returns zero on success, on failure return error code.

Example

```
/* USB clock configuration */
RSI_CLK_USBClkConfig(M4CLK, USB_REFERENCE_CLK,3);
```

14.3.68 RSI_CLK_PeripheralClkEnable

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_CLK_PeripheralClkEnable(M4Clk_Type *pCLK ,PERIPHERALS_CLK_T
module,CLK_ENABLE_T clkType)
```

Description

This API is used to enable the particular clock

Parameters

Parameter	Description
pCLK	Pin number of NPSS_GPIO. Possible values are 0,1,2,3,4
module	To select particular pheripheral. Possible enum value is below • USART1_CLK • USART2_CLK • SSIMST_CLK, • SSISLAVE_CLK • CT_CLK • SD_MEM_CLK • CCI_CLK • QSPI_CLK • RPDMA_CLK • UDMA_CLK • PWM_CLK

Parameter	Description
	• CAN_CLK • GSPI_CLK • EGPIO_CLK • ETHERNET_CLK • MCUCLKOUT_CLK • HWRNG_CLK • I2SM_CLK
clkType	To select the clock as dynamic or static clock. possible enum value below • ENABLE_DYN_CLK, • ENABLE_STATIC_CLK

Return values

Returns zero on success, on failure return error code.

Example

```
/* enable peripheral clock */
RSI_CLK_PeripheralClkEnable(M4CLK,USART1_CLK,ENABLE_STATIC_CLK);
```

14.3.69 RSI_CLK_PeripheralClkDisable

Source File : rsi_rom_clks.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_CLK_PeripheralClkDisable(M4Clk_Type *pCLK ,PERIPHERALS_CLK_T
module)
```

Description

This API is used to disable the particular clock

Parameters

Parameter	Description
pCLK	Pin number of NPSS_GPIO. Possible values are 0,1,2,3,4
module	To select particular pheripheral. Possible enum value is below • USART1_CLK • USART2_CLK • SSIMST_CLK, • SSISLAVE_CLK • CT_CLK

Parameter	Description
	• SD_MEM_CLK • CCI_CLK • QSPI_CLK • RPDMA_CLK • UDMA_CLK • PWM_CLK • CAN_CLK • GSPI_CLK • EGPIO_CLK • ETHERNET_CLK • MCUCLKOUT_CLK • HWRNG_CLK • I2SM_CLK

Return values

Returns zero on success, on failure return error code.

Example

```
/* disable peripheral clock */
RSI_CLK_PeripheralClkDisable(M4CLK,USART1_CLK);            //USART clock disable
```

15 Companion Chip Interface (CCI)

15.1 Overview

This section explains how to configure and use the Companion Chip Interface using Redpine MCU SAPIs.

15.2 Programming Sequence

```
#include "rsi_chip.h" /* Redpine_MCU_Vx.y.z\Host_MCU\common\chip\inc */

void RSI_CCI_InitPinMux()
 {
    RSI_EGPIO_SetPinMux(EGPIO, 0, 12,  9);
    RSI_EGPIO_SetPinMux(EGPIO, 0, 39,  9);
    RSI_EGPIO_SetPinMux(EGPIO, 0, 14,  10);
    RSI_EGPIO_SetPinMux(EGPIO, 0, 15,  10);

    RSI_EGPIO_SetPinMux(EGPIO, 0, 10,  9);
    RSI_EGPIO_SetPinMux(EGPIO, 0, 22,  9);
    RSI_EGPIO_SetPinMux(EGPIO, 0, 23,  9);
    RSI_EGPIO_SetPinMux(EGPIO, 0, 24,  9);
}
void Write_Data(uint32_t Offset,uint32_t Data)
{
    *( volatile uint32_t *)(CCI_AHB_SLAVE_ADDR+Offset)=Data;
}
uint32_t  Read_Data(uint32_t Offset)
{
    return *( volatile uint32_t *)(CCI_AHB_SLAVE_ADDR+Offset);
}
int main()
{

    RSI_CCI_Init_t cci_config;
    cci_config.mode                                = SDR_MODE;             //!SDR mode
selected
    cci_config.slave_enable                        = 0x1;                  //!slave 1
enable
    cci_config.early_bus_termination               = 0;                    //!early bus
termination if enable ,1
    cci_config.address_width_config                = _40BIT_ADDRESS_WIDTH; //!address
width config
    cci_config.interface_width                     = CCI_QUAD_MODE;        //!cci
interface width
    cci_config.translation_enable                  = 1;                    //!enable
translation bit
    cci_config.slave_lsb_address[0]                = SLAVE_LSB_ADDRESS;    //!slave lsb
address to be load
    cci_config.slave_msb_address[0]                = SLAVE_MSB_ADDRESS;    //!slave msb
address to be load
    cci_config.translation_address                 = TRANSLATION_ADDRESS;  //!cci
translation address
    cci_config.slave_timeout                       = SLAVE_TIMEOUT;        //!slave
timeout value to be programmable
    cci_config.cci_cntrl_en                        = 1;                    //!enable
this bit for cci controller enable

    //!AMS enable(only for master)
    RSI_CCI_AmsEnable();

    //!enable cci clock
    clk_cci_clk_config(M4CLK ,CCI_M4_SOC_CLK_FOR_OTHER_CLKS ,1,ENABLE_STATIC_CLK);

    //!enable gpio muxing
    RSI_CCI_InitPinMux();

    //!Initialise cci master
```

```
    RSI_CCI_AMS_Initialise(&cci_config);

    //!write data to slave memory
    Write_Data(0x6000,0x12345678);

    //!read data from slave memory
    read=Read_Data(0x6000);

    while(1)
    {
        /*Sleep till interrupt occurs*/
        DEBUGOUT("Core is Sleeping...\n");
        __WFI();
    }
    /*Statement will never reach here , just to satisfy the standard main*/
    return 0;
}
```

Code Block 9 Companion Chip Interface Example

15.3 API Descriptions

15.3.1 RSI_CCI_AmsEnable

Source File : rsi_cci.c

Path : Redpine _MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
void RSI_CCI_AmsEnable (void)
```

Description

This API enables the CCI in the master mode.

Example

```
/* AMS enable */
RSI_CCI_AmsEnable();
```

15.3.2 RSI_CCI_AMS_Initialise

Source File : rsi_cci.c

Path : Redpine _MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
error_t RSI_CCI_AMS_Initialise (RSI_CCI_Init_t *p_cci_config)
```

Description

This API configures the CCI peripheral.

Parameter

Parameter	Description
p_cci_config	CCI configuration structure pointer • RSI_CCI_Init_t

Return value

returns 0 RSI_OK on success ,Error code on failure.

Example

```
RSI_CCI_Init_t cci_config;
/* configure structure parameter */
cci_config.mode                                  = SDR_MODE;
cci_config.slave_enable                          = 0x1;
cci_config.early_bus_termination                 = 0;
cci_config.address_width_config                  = _40BIT_ADDRESS_WIDTH;
cci_config.interface_width                       = CCI_QUAD_MODE;
cci_config.translation_enable                    = 1;
cci_config.slave_lsb_address[0]                  = SLAVE_LSB_ADDRESS;
cci_config.slave_msb_address[0]                  = SLAVE_MSB_ADDRESS;
cci_config.translation_address                   = TRANSLATION_ADDRESS;
cci_config.slave_timeout                         = SLAVE_TIMEOUT;
cci_config.cci_cntrl_en                          = 1;

/* enables the CCI in the master mode. */
RSI_CCI_AMS_Initialise(&cci_config);
```

15.3.3 RSI_CCI_SetFifoThreshlod

Source File : rsi_cci.c

Path : Redpine _MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
error_t RSI_CCI_SetFifoThreshlod (volatile CCI_Type *pstcCCI, uint8_t val)
```

Description

This API sets the CCI threshold fifo value.

Parameter

Parameter	Description
pstcCCI	Pointer to the CCI register instance.
val	Threshold value

Return value

RSI_OK if success full or else error code

Example

```
/* set threshold value */
RSI_CCI_SetFifoThreshlod(CCI,8);
```

15.3.4 RSI_CCI_PrefetchEnable

Source File : rsi_cci.c

Path : Redpine _MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
error_t RSI_CCI_PrefetchEnable (volatile CCI_Type *pstcCCI)
```

Description

This API is used for prefetct enable.

Parameter

Parameter	Description
pstcCCI	Pointer to the CCI register instance

Return value

RSI_OK if success full or else error code

Example

```
/* prefetch enable */
RSI_CCI_PrefetchEnable(CCI);
```

15.3.5 RSI_CCI_IntClear

Source File : rsi_cci.c

Path : Redpine _MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
void RSI_CCI_IntClear (volatile CCI_Type *pstcCCI, uint8_t interrupt)
```

Description

This API is used to clear the interrupts.

Parameters

Parameter	Description
pstcCCI	Pointer to the CCI register instance
interrupt	0-2 for interrupt from peer chips 3 for message interrupt from slave

Return value

RSI_OK if success full or else error code

Example

```
/* clear interrupt */
RSI_CCI_IntClear(CCI,2);
```

16 Configurable Timers (CT)

16.1 Overview

This section explains how to configure and use the Configurable Timers using Redpine MCU SAPIs.

16.2 Programming Sequence

```
#include "rsi_chip.h"  /* Redpine_MCU_Vx.y.z\Host_MCU\Common\chip\inc */
#include "rsi_board.h" /* Redpine_MCU_Vx.y.z\Host_MCU\Common\board\inc */


/* Private typedef -----------------------------------------------------------*/

/* Private macro -------------------------------------------------------------*/
#define    CT_RATE       10000
#define    TICKS         1000        /* 1 ms Tick rate */
/* Private define ------------------------------------------------------------*/
#define    CT_IRQHandler         IRQ034_Handler
/* Private variables ---------------------------------------------------------*/
uint8_t isr_check = 0;
OCU_PARAMS_T vsOCUparams;
RSI_CT_CALLBACK_T vsCB;

void SysTick_Handler(void)
{

}
/**
 * @brief     CT pin muxing
 * @return    Nothing
 */


void RSI_CT_PinMux()
{
    RSI_EGPIO_PadReceiverEnable(50);
    RSI_EGPIO_PadReceiverEnable(7);
    RSI_EGPIO_PadSelectionEnable(12);
    /* CT_OUT0 GPIO */
    RSI_EGPIO_SetPinMux(EGPIO ,3 , GPIO2 ,EGPIO_PIN_MUX_MODE6);
    RSI_EGPIO_SetPinMux(EGPIO ,0 , GPIO7 ,EGPIO_PIN_MUX_MODE6);
}

/* CT OCU pulse width con-fig */
uint32_t CT_SetPeriod(uint32_t freq)
{
   uint32_t rate = 0;
   rate = SystemCoreClock/freq;
   /* set the Peak Match value for CT0 coutner0 & counter 1 */
   RSI_CT_SetMatchCount(CT0, rate,0,0);
   RSI_CT_SetMatchCount(CT0, rate,0,1);
   return rate;
}


/* Converts duty cycle percentage to system ticks */
uint32_t CT_PercentageToTicks(uint8_t percent,uint32_t freq)
{
   uint16_t ticks = 0;
   uint32_t rate = 0;

   rate = SystemCoreClock/freq;
   ticks = (rate * percent) / 100;
   return ticks;
}
```

```
/* Main program. */
int main()
{
    int loop = 1;
    uint32_t delay = 0;
    int pwm_out_0, pwm_out_1, duty_p = 0, incr = 1, Led = 0;
    volatile uint32_t time_period = 0;
    uint32_t ticks =0;

    OCU_PARAMS_T vsOCUparams;
    /*Configures the system default clock and power configurations*/
     SystemCoreClockUpdate();
    /*    Enable the DEBUG UART port for debug prints and  Set up and initialize all
required
        blocks and functions  related to the board hardware.  */
     RSI_Board_Init();

    /* ULPGPIO_0 and ULPGPIO_2 are using*/
    /*  Sets the state of a board '2'number LED to ON. */
    RSI_Board_LED_Set(2, 1);

    /* enables pad selection bits,enables receive_enable for the CT pins and
    configures the pins for CT.(enable gpio muxing )   */

    RSI_CT_PinMux();

    /*    Initialization and Configure the CT clocks */
    RSI_CLK_CtClkConfig(M4CLK, M4_SOCCLKFOROTHERCLKSCT ,1, ENABLE_STATIC_CLK );

      /* set the 32bit coutner*/
    RSI_CT_Config(CT0,0);

    /* Sets control for counter0 and counter 1 */
    RSI_CT_SetControl( CT0, PERIODIC_EN_COUNTER_0_FRM_REG|
                            PERIODIC_EN_COUNTER_1_FRM_REG );
    /* CT OCU pulse width config */
    ticks = CT_SetPeriod(CT_RATE);

    /* Set Duty cycle value for channel 0 and channel 1*/
    RSI_MCPWM_SetDutyCycle(MCPWM, 0 ,PWM_CHNL_0);       /* LED */
    RSI_MCPWM_SetDutyCycle(MCPWM, ticks, PWM_CHNL_1);   /* OUT */

    vsOCUparams.CompareVal1_0 = 0x0;
    vsOCUparams.CompareVal1_1 = ticks/2;

    /* OCU configuration on Counter0 & 1 of CT0 instance */
    RSI_CT_OCUConfigSet(CT0,OUTPUT_OCU_0 | OUTPUT_OCU_1 |
                            MAKE_OUTPUT_0_HIGH_SEL_0 |
                            MAKE_OUTPUT_0_LOW_SEL_0     |
                            MAKE_OUTPUT_1_HIGH_SEL_1 |
                            MAKE_OUTPUT_1_LOW_SEL_1 |
                            OCU_SYNC_WITH_0 |OCU_SYNC_WITH_1);

    /* OCU counter 0 compare values and compare next values */
    RSI_CT_OCUControl(CT0,COUNTER_0,0, &vsOCUparams, &vsCB);

    /* OCU counter 1 compare values and compare next values */
    RSI_CT_OCUControl(CT0,COUNTER_1,0, &vsOCUparams, &vsCB);

    /*  Programs General control register bit fields */
    RSI_CT_SetControl(CT0,COUNTER_0_UP | COUNTER_1_UP|
                        COUNTER0_SYNC_TRIG |COUNTER1_SYNC_TRIG|
                        COUNTER0_TRIG|COUNTER1_TRIG);
    /* Enable SysTick Timer */
    SysTick_Config(SystemCoreClock / TICKS);

    while (loop)
    {
      delay++;
```

```
        if(delay >= 20)
        {
          duty_p = duty_p + incr;
          if (duty_p < 0)
          {
              duty_p = 0;
              incr = 1;
          }
          if (duty_p > 100)
          {
              duty_p = 100;
              incr = -1;
              Led = 1 - Led;
          }
          if (Led)
          {
                  pwm_out_0 = duty_p;
                  pwm_out_1 = 100;
          }
          else
          {
                 pwm_out_0 = 100;
                 pwm_out_1 = duty_p;
          }
          /* Converts duty cycle percentage to system ticks */
          vsOCUparams.CompareVal1_0 = CT_PercentageToTicks(pwm_out_0, CT_RATE);
          vsOCUparams.CompareVal1_1 = CT_PercentageToTicks(pwm_out_1, CT_RATE);

          /* Set duty cycle */
          RSI_CT_OCUControl(CT0,COUNTER_0,0, &vsOCUparams, &vsCB);  /* LED */
          RSI_CT_OCUControl(CT0,COUNTER_1,0, &vsOCUparams, &vsCB);  /* LED */
          delay = 0;
          }
          __WFI();
    }
  }
```

Code Block 10 Configurable Timers Example

16.3 API Descriptions

16.3.1 RSI_CT_Config

Source File : rsi_ct.h

Path : Redpine_MCU_V1.x.x\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_CT_Config (RSI_CT_T *pCT, boolean_t cfg)
```

Description

This API is used to set the 32bit/ 16bit coutners.

Parameters

Parameter	Description
pCT	Pointer to the CT instance register area
cfg	• cfg = 0 32bit Counter, • cfg = 1 16bit counter

Return values

none

Example

```
/*  used to set the 32bit/ 16bit coutners */
RSI_CT_Config(CT0,0);
```

16.3.2 RSI_CT_SetControl

Source File : rsi_ct.h

Path : Redpine_MCU_V1.x.x\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_CT_SetControl (RSI_CT_T *pCT, uint32_t value)
```

Description

Programs General control register bit fields.

Parameters

Parameter	Description
pCT	Pointer to the CT instance register area
value	This parameter can be the logical OR of the below parameters • COUNTER32_BITMODE : sets 32bit mode • SOFTRESET_COUNTER_0 : Resets coutner 0 • PERIODIC_ENCOUNTER_0 : sets periodic mode • COUNTER0_TRIG : starts counter 0 • COUNTER0_UP_DOWN : Coutner 0 direction (0,1,2,3) • COUNTER0_SYNC_TRIG : This enables the counter 0 to run/active when sync is found. • BUF_REG0EN : Buffer will be enabled and in path for coutner 0 • SOFTRESET_COUNTER_1 : Resets coutner 1 • PERIODIC_ENCOUNTER_1 : sets periodic mode • COUNTER1_TRIG : starts counter 1 • COUNTER1_UP_DOWN : Coutner 1 direction (0,1,2,3) • COUNTER1_SYNC_TRIG : This enables the counter 1 to run/active when sync is found. • BUF_REG1EN : Buffer will be enabled and in path for counter 1.

Return values

none

Example

```
/* Sets control for counter0 and counter 1 */
RSI_CT_SetControl( CT0, PERIODIC_EN_COUNTER_0_FRM_REG|PERIODIC_EN_COUNTER_1_FRM_REG);
```

16.3.3 RSI_CT_StartEventSelect

Source File : rsi_ct.h

Path: Redpine_MCU_V1.x.x\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_CT_StartEventSelect (RSI_CT_T *pCT, uint32_t value)
```

Description

This API is used to select the input event to start any counter

Parameters

Parameter	Description
pCT	Pointer to the CT instance register area
value	This parameter can be select CT, possible value is as below. • CT_EVT_0 • CT_EVT_1 • CT_EVT_2 • CT_EVT_3 • CT_EVT_4 • CT_EVT_5 • CT_EVT_6 • CT_EVT_7 • CT_EVT_8 • CT_EVT_9 • CT_EVT_10 • CT_EVT_11 • CT_EVT_12 • CT_EVT_13 • CT_EVT_14 • CT_EVT_15 • CT_EVT_16 • CT_EVT_17 • CT_EVT_18 • CT_EVT_19 • CT_EVT_20 • CT_EVT_21 • CT_EVT_22 • CT_EVT_23 • CT_EVT_24 • CT_EVT_25 • CT_EVT_26 • CT_EVT_27 • CT_EVT_28 • CT_EVT_29

Parameter	Description
	• CT_EVT_30 • CT_EVT_31 • CT_EVT_32 • CT_EVT_33 • CT_EVT_34 • CT_EVT_35 • CT_EVT_36 • CT_EVT_37

Return values

none

Example

```
/* select event */
RSI_CT_StartEventSelect(CT0,CT_EVT_0);
```

16.3.4 RSI_CT_HaltEventSelect

Source File : rsi_ct.h

Path : Redpine_MCU_V1.x.x\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_CT_HaltEventSelect (RSI_CT_T *pCT, uint32_t value)
```

Description

Configures event for HALT operation of Counter.

Parameters

Parameter	Description
pCT	Pointer to the CT instance register area
Value	This parameter can be select CT, possible value is as below. • CT_EVT_0 • CT_EVT_1 • CT_EVT_2 • CT_EVT_3 • CT_EVT_4 • CT_EVT_5 • CT_EVT_6 • CT_EVT_7 • CT_EVT_8 • CT_EVT_9 • CT_EVT_10 • CT_EVT_11

Parameter	Description
	• CT_EVT_12 • CT_EVT_13 • CT_EVT_14 • CT_EVT_15 • CT_EVT_16 • CT_EVT_17 • CT_EVT_18 • CT_EVT_19 • CT_EVT_20 • CT_EVT_21 • CT_EVT_22 • CT_EVT_23 • CT_EVT_24 • CT_EVT_25 • CT_EVT_26 • CT_EVT_27 • CT_EVT_28 • CT_EVT_29 • CT_EVT_30 • CT_EVT_31 • CT_EVT_32 • CT_EVT_33 • CT_EVT_34 • CT_EVT_35 • CT_EVT_36 • CT_EVT_37

Return values

none

Example

```
/*Event 2 rise edge for continue counter operation*/
RSI_CT_HaltEventSelect(CT0, CT_EVT_2);
```

16.3.5 RSI_CT_ContinueEventSelect

Source File : rsi_ct.h

Path: Redpine_MCU_V1.x.x\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_CT_ContinueEventSelect (RSI_CT_T *pCT, uint32_t value)
```

Description

Configures event for Continue operation of Counter.

Parameters

Parameter	Description
pCT	Pointer to the CT instance register area
Value	This parameter can be select CT, possible value is as below. • CT_EVT_0 • CT_EVT_1 • CT_EVT_2 • CT_EVT_3 • CT_EVT_4 • CT_EVT_5 • CT_EVT_6 • CT_EVT_7 • CT_EVT_8 • CT_EVT_9 • CT_EVT_10 • CT_EVT_11 • CT_EVT_12 • CT_EVT_13 • CT_EVT_14 • CT_EVT_15 • CT_EVT_16 • CT_EVT_17 • CT_EVT_18 • CT_EVT_19 • CT_EVT_20 • CT_EVT_21 • CT_EVT_22 • CT_EVT_23 • CT_EVT_24 • CT_EVT_25 • CT_EVT_26 • CT_EVT_27 • CT_EVT_28 • CT_EVT_29 • CT_EVT_30 • CT_EVT_31 • CT_EVT_32 • CT_EVT_33 • CT_EVT_34

Parameter	Description
	• CT_EVT_35 • CT_EVT_36 • CT_EVT_37

Return values

none

Example

```
/*Event 2 rise edge for continue counter operation*/
RSI_CT_ContinueEventSelect(CT0, CT_EVT_2);
```

16.3.6 RSI_CT_InterruptEventSelect

Source File : rsi_ct.h

Path : Redpine_MCU_V1.x.x\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_CT_InterruptEventSelect (RSI_CT_T *pCT, uint32_t value)
```

Description

This API is used to select the input event for interrupt.

Parameters

Parameter	Description
pCT	Pointer to the CT instance register area
Value	This parameter can be select CT event, possible value is as below. • CT_EVT_0 • CT_EVT_1 • CT_EVT_2 • CT_EVT_3 • CT_EVT_4 • CT_EVT_5 • CT_EVT_6 • CT_EVT_7 • CT_EVT_8 • CT_EVT_9 • CT_EVT_10 • CT_EVT_11 • CT_EVT_12 • CT_EVT_13 • CT_EVT_14 • CT_EVT_15 • CT_EVT_16

Parameter	Description
	• CT_EVT_17 • CT_EVT_18 • CT_EVT_19 • CT_EVT_20 • CT_EVT_21 • CT_EVT_22 • CT_EVT_23 • CT_EVT_24 • CT_EVT_25 • CT_EVT_26 • CT_EVT_27 • CT_EVT_28 • CT_EVT_29 • CT_EVT_30 • CT_EVT_31 • CT_EVT_32 • CT_EVT_33 • CT_EVT_34 • CT_EVT_35 • CT_EVT_36 • CT_EVT_37

Return values

none

Example

```
/* event for counter interrupt operation */
RSI_CT_InterruptEventSelect(CT0, CT_EVT_21);
```

16.3.7 RSI_CT_InterruptEventConfig

Source File : rsi_ct.h

Path : Redpine_MCU_V1.x.x\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_CT_InterruptEventConfig (RSI_CT_T *pCT, uint32_t andValue, uint32_t
orValue)
```

Description

This API is used to Configure AND/OR event for interrupt operation.

Parameters

Parameter	Description

Parameter	Description
pCT	Pointer to the CT instance register area
andValue	• AND_COUNTER0(valid,event) : pass the required valid and events to this macro for counter0 • AND_COUNTER1(valid,event) : pass the required valid and events to this macro for counter1
orValue	• OR_COUNTER0(valid,event) :pass the required valid and events to this macro for counter0 • OR_COUNTER1(valid,event) :pass the required valid and events to this macro for counter1

Return values

none

Example

```
/* AND event for counter interrupt operation */
RSI_CT_InterruptEventConfig (CT0,AND_COUNTER0(0x2,0x3),OR_COUNTER0(0x3,0xF));
```

16.3.8 RSI_CT_OutputEventSelect

Source File : rsi_ct.h

Path : Redpine_MCU_V1.x.x\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_CT_OutputEventSelect (RSI_CT_T *pCT, uint32_t value)
```

Description

This API is used to select the input event to output counter value.

Parameters

Parameter	Description
pCT	Pointer to the CT instance register area
Value	This parameter can be select CT event, possible value is as below. • CT_EVT_0 • CT_EVT_1 • CT_EVT_2 • CT_EVT_3 • CT_EVT_4 • CT_EVT_5 • CT_EVT_6 • CT_EVT_7 • CT_EVT_8 • CT_EVT_9 • CT_EVT_10 • CT_EVT_11 • CT_EVT_12 • CT_EVT_13

Parameter	Description
	• CT_EVT_14 • CT_EVT_15 • CT_EVT_16 • CT_EVT_17 • CT_EVT_18 • CT_EVT_19 • CT_EVT_20 • CT_EVT_21 • CT_EVT_22 • CT_EVT_23 • CT_EVT_24 • CT_EVT_25 • CT_EVT_26 • CT_EVT_27 • CT_EVT_28 • CT_EVT_29 • CT_EVT_30 • CT_EVT_31 • CT_EVT_32 • CT_EVT_33 • CT_EVT_34 • CT_EVT_35 • CT_EVT_36 • CT_EVT_37

Return values

none

Example

```
/* output CT event select */
RSI_CT_OutputEventSelect (CT0, CT_EVT_9);
```

16.3.9 RSI_CT_OutputEventConfig

Source File : rsi_ct.h

Path : Redpine_MCU_V1.x.x\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_CT_OutputEventConfig (RSI_CT_T *pCT, uint32_t andValue, uint32_t
orValue)
```

Description

This API is used to Configure AND/OR event for output operation.

Parameters

Parameter	Description
pCT	Pointer to the CT instance register area
andValue	• AND_COUNTER0(valid,event) : pass the required valid and events to this macro for counter0 • AND_COUNTER1(valid,event) : pass the required valid and events to this macro for counter1
orValue	• OR_COUNTER0(valid,event) :pass the required valid and events to this macro for counter0 • OR_COUNTER1(valid,event) :pass the required valid and events to this macro for counter1

Return values

none

Example

```
/*OR event for counter Output operation*/
RSI_CT_OutputEventSelect(CT0, CT_EVT_22);
RSI_CT_OutputEventConfig(CT0, 0, OR_COUNTER0(0x3,0xF);
```

16.3.10 RSI_CT_GetInterruptStatus

Source File : rsi_ct.h

Path : Redpine_MCU_V1.x.x\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE uint32_t RSI_CT_GetInterruptStatus (RSI_CT_T *pCT)
```

Description

This API is use to get the interrupt status.

Parameter

Parameter	Description
pCT	Pointer to the CT instance register area

Return values

Return interrupt status

Example

```
uint32_t  status_1 =0
/*Get interrupt status*/
status_1 = RSI_CT_GetInterruptStatus(CT1);
```

16.3.11 RSI_CT_InterruptClear

Source File : rsi_ct.h

Path : Redpine_MCU_V1.x.x\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_CT_InterruptClear (RSI_CT_T *pCT, uint32_t clrFlags)
```

Description

Clear the specified interrupt flag in State Configurable Timer.

Parameters

Parameter	Description
pCT	Pointer to the CT instance register area
clrFlags	CT Interrupt Ack register value,this parameter can be the logical OR of the • INTR_0_l : interrupt event flag for counter 0 • COUNTER_0_IS_ZERO_L : counter hit zero for counter 0 • COUNTER_0_IS_PEAK_L : counter hit peak for counter 0 • INTR_1_L : interrupt event flag for counter 1 • COUNTER_0_IS_ZERO_L : counter hit zero for counter 1 • COUNTER_0_IS_PEAK_L : counter hit peak for counter 1

Return values

none

Example

```
/* interrupt clear */
RSI_CT_InterruptClear(CT0,RSI_CT_EVENT_INTR_0_l);
```

16.3.12 RSI_CT_ResumeHaltEvent

Source File : rsi_ct.h

Path : Redpine_MCU_V1.x.x\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_CT_ResumeHaltEvent (RSI_CT_T *pCT, boolean_t counterNum)
```

Description

This API is used to Reset any counter.

Parameters

Parameter	Description
pCT	Pointer to the CT instance register area
counterNum	Coutner 0/1

Return values

none

Example

```
#define   COUNTER_0   0
/*resume counter from halt state*/
RSI_CT_ResumeHaltEvent(CT0,COUNTER_0);
```

16.3.13 RSI_CT_StartEventConfig

Source File : rsi_ct.h

Path: Redpine_MCU_V1.x.x\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_CT_StartEventConfig (RSI_CT_T *pCT, uint32_t andValue, uint32_t
orValue)
```

Description

This API is used to Configure AND/OR event for start counter operation.

Parameters

Parameter	Description
pCT	Pointer to the CT instance register area
andValue	• AND_COUNTER0(valid,event) : pass the required valid and events to this macro for counter0 • AND_COUNTER1(valid,event) : pass the required valid and events to this macro for counter1
orValue	• OR_COUNTER0(valid,event) :pass the required valid and events to this macro for counter0 • OR_COUNTER1(valid,event) :pass the required valid and events to this macro for counter1

Return values

none

Example

```
RSI_CT_StartEventConfig(CT0,AND_COUNTER0(0x3,0xF),OR_COUNTER0(0x3,0xF));
```

16.3.14 RSI_CT_ClearControl

Source File : rsi_ct.h

Path : Redpine_MCU_V1.x.x\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_CT_ClearControl (RSI_CT_T *pCT, uint32_t value)
```

Description

This API is used to Reset the required control bits in general control set register.

Parameters

Parameter	Description
pCT	Pointer to the CT instance register area
value	This parameter can be the logical OR of the required bit ,fields in CT Reset Control register as below. • COUNTER32_BITMODE : Sets 16bit mode • PERIODIC_ENCOUNTER_0 : Sets Counter_0 will be in single count mode. • COUNTER0_UP_DOWN : counter 0 to run in up/down/up-down/down-up directions (0,1,2,3) • BUF_REG0EN : Buffer will be disabled and in path and in path for coutner 0 • PERIODIC_ENCOUNTER_1 : sets coutner 1 will be in single count mode

Parameter	Description
	• COUNTER1_UP_DOWN : Coutner 1 to run in up/down/up-down/down-up directions(0,1,2,3) • BUF_REG1EN : Buffer will be disabled and in path for counter 1.

Return values

none

Example

```
/* Reset control for counter0 and counter 1 */
 RSI_CT_ClearControl( CT0, PERIODIC_EN_COUNTER_0_FRM_REG|PERIODIC_EN_COUNTER_1_FRM_REG);
```

16.3.15 RSI_CT_ContinueEventConfig

Source File : rsi_ct.h

Path : Redpine_MCU_V1.x.x\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_CT_ContinueEventConfig (RSI_CT_T *pCT, uint32_t andValue, uint32_t
orValue)
```

Description

This API is used to Configure AND/OR event for Continue operation.

Parameters

Parameter	Description
pCT	Pointer to the CT instance register area
andValue	• AND_COUNTER0(valid,event) : pass the required valid and events to this macro for counter0 • AND_COUNTER1(valid,event) : pass the required valid and events to this macro for counter1
orValue	• OR_COUNTER0(valid,event) :pass the required valid and events to this macro for counter0 • OR_COUNTER1(valid,event) :pass the required valid and events to this macro for counter1

Return values

none

Example

```
RSI_CT_ContinueEventConfig(CT0,AND_COUNTER0(0x1,0xF),OR_COUNTER0(0x3,0xF));
```

16.3.16 RSI_CT_HaltEventConfig

Source File : rsi_ct.h

Path: Redpine_MCU_V1.x.x\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_CT_HaltEventConfig (RSI_CT_T *pCT, uint32_t andValue, uint32_t
orValue)
```

Description

This API is used to Configure AND/OR event for Halt operation.

Parameters

Parameter	Description
pCT	Pointer to the CT instance register area
andValue	• AND_COUNTER0(valid,event) : pass the required valid and events to this macro for counter0 • AND_COUNTER1(valid,event) : pass the required valid and events to this macro for counter1
orValue	• OR_COUNTER0(valid,event) :pass the required valid and events to this macro for counter0 • OR_COUNTER1(valid,event) :pass the required valid and events to this macro for counter1

Return values

none

Example

```
RSI_CT_HaltEventConfig (CT0,AND_COUNTER0(0x3,0xF),OR_COUNTER0(0x3,0xF));
```

16.3.17 RSI_CT_IncrementEventConfig

Source File : rsi_ct.h

Path: Redpine_MCU_V1.x.x\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_CT_IncrementEventConfig (RSI_CT_T *pCT, uint32_t andValue, uint32_t
orValue)
```

Description

This API is used to Configure AND/OR event for increment operation.

Parameters

Parameter	Description
pCT	Pointer to the CT instance register area
andValue	• AND_COUNTER0(valid,event) : pass the required valid and events to this macro for counter0 • AND_COUNTER1(valid,event) : pass the required valid and events to this macro for counter1
orValue	• OR_COUNTER0(valid,event) :pass the required valid and events to this macro for counter0 • OR_COUNTER1(valid,event) :pass the required valid and events to this macro for counter1

Return values

none

Example

```
RSI_CT_IncrementEventConfig (CT0,AND_COUNTER0(0x2,0x3),OR_COUNTER0(0x3,0xF));
```

16.3.18 RSI_CT_CaptureEventConfig

Source File : rsi_ct.h

Path: Redpine_MCU_V1.x.x\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_CT_CaptureEventConfig (RSI_CT_T *pCT, uint32_t andValue, uint32_t
orValue)
```

Description

This API is used to Configure AND/OR event for capture operation.

Parameters

Parameter	Description
pCT	Pointer to the CT instance register area
andValue	• AND_COUNTER0(valid,event) : pass the required valid and events to this macro for counter0 • AND_COUNTER1(valid,event) : pass the required valid and events to this macro for counter1
orValue	• OR_COUNTER0(valid,event) :pass the required valid and events to this macro for counter0 • OR_COUNTER1(valid,event) :pass the required valid and events to this macro for counter1

Return values

none

Example

```
RSI_CT_CaptureEventConfig (CT0,AND_COUNTER0(0x2,0x3),OR_COUNTER0(0x3,0xF));
```

16.3.19 RSI_CT_IncrementEventSelect

Source File : rsi_ct.h

Path : Redpine_MCU_V1.x.x\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_CT_IncrementEventSelect (RSI_CT_T *pCT, uint32_t value)
```

Description

Configures event for Increment operation of Counter.

Parameters

Parameter	Description
pCT	Pointer to the CT instance register area
Value	This parameter can be select CT, possible value is as below. • CT_EVT_0 • CT_EVT_1 • CT_EVT_2 • CT_EVT_3

Parameter	Description
	• CT_EVT_4 • CT_EVT_5 • CT_EVT_6 • CT_EVT_7 • CT_EVT_8 • CT_EVT_9 • CT_EVT_10 • CT_EVT_11 • CT_EVT_12 • CT_EVT_13 • CT_EVT_14 • CT_EVT_15 • CT_EVT_16 • CT_EVT_17 • CT_EVT_18 • CT_EVT_19 • CT_EVT_20 • CT_EVT_21 • CT_EVT_22 • CT_EVT_23 • CT_EVT_24 • CT_EVT_25 • CT_EVT_26 • CT_EVT_27 • CT_EVT_28 • CT_EVT_29 • CT_EVT_30 • CT_EVT_31 • CT_EVT_32 • CT_EVT_33 • CT_EVT_34 • CT_EVT_35 • CT_EVT_36 • CT_EVT_37

Return values

none

Example

```
/* increment event operation */
RSI_CT_IncrementEventSelect (CT0, CT_EVT_2);
```

16.3.20 RSI_CT_CaptureEventSelect

Source File : rsi_ct.h

Path : Redpine_MCU_V1.x.x\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_CT_CaptureEventSelect(RSI_CT_T *pCT, uint32_t value)
```

Description

This API is used to select the input event to capture counter value.

Parameters

Parameter	Description
pCT	Pointer to the CT instance register area
Value	This parameter can be select CT event, possible value is as below. • CT_EVT_0 • CT_EVT_1 • CT_EVT_2 • CT_EVT_3 • CT_EVT_4 • CT_EVT_5 • CT_EVT_6 • CT_EVT_7 • CT_EVT_8 • CT_EVT_9 • CT_EVT_10 • CT_EVT_11 • CT_EVT_12 • CT_EVT_13 • CT_EVT_14 • CT_EVT_15 • CT_EVT_16 • CT_EVT_17 • CT_EVT_18 • CT_EVT_19 • CT_EVT_20 • CT_EVT_21 • CT_EVT_22 • CT_EVT_23 • CT_EVT_24 • CT_EVT_25 • CT_EVT_26 • CT_EVT_27

Parameter	Description
	• CT_EVT_28 • CT_EVT_29 • CT_EVT_30 • CT_EVT_31 • CT_EVT_32 • CT_EVT_33 • CT_EVT_34 • CT_EVT_35 • CT_EVT_36 • CT_EVT_37

Return values

none

Example

```
/* capture CT event */
RSI_CT_CaptureEventSelect(CT0, CT_EVT_2);
```

16.3.21 RSI_CT_OutputEventADCTrigger

Source File : rsi_ct.h

Path : Redpine_MCU_V1.x.x\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_CT_OutputEventADCTrigger (RSI_CT_MUX_REG_T *pCTmux, uint8_t
output1, uint8_t output2)
```

Description

This API is used to select one of the ADC trigger output.

Parameters

Parameter	Description
pCT	Pointer to the CT instance register area
Value	output event for ADC trigger (0 to 31)
output2	output event for ADC trigger (0 to 31)

Return values

none

Example

```
RSI_CT_OutputEventADCTrigger(CT0,2,2)
```

16.3.22 RSI_CT_SetCount

Source File : rsi_ct.h

Path : Redpine_MCU_V1.x.x\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_CT_SetCount (RSI_CT_T *pCT, uint32_t count)
```

Description

Sets the Counter Initial value.

Parameters

Parameter	Description
pCT	Pointer to the CT instance register area
count	32 bit Coutner initial value,this parameter can be the logical OR of the • COUNTER_0 : Counter 0 load value (0x0 to 0xFFFF) • COUNTER_1 : Counter 1 load value (0x0 to 0xFFFF) • possible values are 0x0 to 0xFFFFFFFF

Return values

none

Example

```
/* set counter initial value */
 RSI_CT_SetCount(CT0,0xF);
```

16.3.23 RSI_CT_OCUConfigSet

Source File : rsi_ct.h

Path : Redpine_MCU_V1.x.x\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_CT_OCUConfigSet (RSI_CT_T *pCT, uint32_t value)
```

Description

This API is used to set OCU control parameters.

Parameters

Parameter	Description
pCT	Pointer to the CT instance register area
value	OCU control register value,this parameter can be the logical OR of the below parameters • OUTPUT_IS_OCU_0 : sets coutner 0 output in OCU mode • SYNC_WITH_0 : Indicates whether the other channel is in sync with this counter(0,1,2,3) • OCU_DMA_MODE_0 : OCU DMA mode is active or not for counter-0 • OCU_8_16_MODE_0 : 16 bits or only 8-bits of the counter-0 are used in OCU mode • OUTPUT_IS_OCU_1 : sets coutner 1 output in OCU mode • SYNC_WITH_1 : Indicates whether the other channel is in sync with this counter(0,1,2,3)

Parameter	Description
	• OCU_DMA_MODE_1 : OCU DMA mode is active or not for counter-1 • OCU_8_16_MODE_1 : 16 bits or only 8-bits of the counter-1 are used in OCU mode

Return values

none

Example

```
/* OCU configuration on Counter0 & 1 of CT0 instance */
RSI_CT_OCUConfigSet(CT0,    OUTPUT_OCU_0 | OUTPUT_OCU_1);
```

16.3.24 RSI_CT_OCUConfigReset

Source File : rsi_ct.h

Path : Redpine_MCU_V1.x.x\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_CT_OCUConfigReset (RSI_CT_T *pCT, uint32_t value)
```

Description

This API is used to set OCU control parameters.

Parameters

Parameter	Description
pCT	Pointer to the CT instance register area
value	OCU control register value,this parameter can be the logical OR required bit. • OUTPUT_IS_OCU_0 : sets coutner 0 output in OCU mode • SYNC_WITH_0 : Indicates whether the other channel is in sync with this counter(0,1,2,3) • OCU_DMA_MODE_0 : OCU DMA mode is active or not for counter-0 • OCU_8_16_MODE_0 : 16 bits or only 8-bits of the counter-0 are used in OCU mode • OUTPUT_IS_OCU_1 : sets coutner 1 output in OCU mode • SYNC_WITH_1 : Indicates whether the other channel is in sync with this counter(0,1,2,3) • OCU_DMA_MODE_1 : OCU DMA mode is active or not for counter-1 • OCU_8_16_MODE_1 : 16 bits or only 8-bits of the counter-1 are used in OCU mode

Return values

none

Example

```
/* reset OCU configuration *?
RSI_CT_OCUConfigReset (CT0, OUTPUT_OCU_0 | OUTPUT_OCU_1);
```

16.3.25 RSI_CT_InterruptEnable

Source File : rsi_ct.h

Path : Redpine_MCU_V1.x.x\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_CT_InterruptEnable (RSI_CT_T *pCT, uint32_t unmaskFlags)
```

Description

Enable the interrupts in State Configurable Timer.

Parameters

Parameter	Description
pCT	Pointer to the CT instance register area
unmaskFlags	CT Interrupt unmask register value,this parameter can be the logical OR of the • INTR_0_l : interrupt event flag for counter 0 • COUNTER_0_IS_ZERO_L : counter hit zero for counter 0 • COUNTER_0_IS_PEAK_L : counter hit peak for counter 0 • INTR_1_L : interrupt event flag for counter 1 • COUNTER_0_IS_ZERO_L : counter hit zero for counter 1 • COUNTER_0_IS_PEAK_L : counter hit peak for counter 1

Return values

none

Example

```
/* Enable the interrupts in State Configurable Timer. */
RSI_CT_InterruptEnable(CT3, RSI_CT_EVENT_COUNTER_1_IS_PEAK_l);
```

16.3.26 RSI_CT_InterruptDisable

Source File : rsi_ct.h

Path : Redpine_MCU_V1.x.x\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_CT_InterruptDisable (RSI_CT_T *pCT, uint32_t maskFlags)
```

Description

Disable the interrupts in State Configurable Timer.

Parameters

Parameter	Description
pCT	Pointer to the CT instance register area
maskFlags	CT Interrupt mask register value,this parameter can be the logical OR of the • INTR_0_l : interrupt event flag for counter 0 • COUNTER_0_IS_ZERO_L : counter hit zero for counter 0 • COUNTER_0_IS_PEAK_L : counter hit peak for counter 0 • INTR_1_L : interrupt event flag for counter 1

Parameter	Description
	• COUNTER_0_IS_ZERO_L : counter hit zero for counter 1 • COUNTER_0_IS_PEAK_L : counter hit peak for counter 1

Return values

none

Example

```
/* Disable the interrupts in State Configurable Timer. */
RSI_CT_InterruptDisable (CT3, RSI_CT_EVENT_COUNTER_1_IS_PEAK_l);
```

16.3.27 RSI_CT_EdgeLevelEventControl

Source File : rsi_ct.h

Path : Redpine_MCU_V1.x.x\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_CT_EdgeLevelEventControl (RSI_CT_T *pCT, uint32_t value)
```

Description

This API is used to control the input event generation to CT.

Parameters

Parameter	Description
pCT	Pointer to the CT instance register area
value	This parameter can be select CT event, possible value is as below. • CT_EVT_0 • CT_EVT_1 • CT_EVT_2 • CT_EVT_3 • CT_EVT_4 • CT_EVT_5 • CT_EVT_6 • CT_EVT_7 • CT_EVT_8 • CT_EVT_9 • CT_EVT_10 • CT_EVT_11 • CT_EVT_12 • CT_EVT_13 • CT_EVT_14 • CT_EVT_15 • CT_EVT_16 • CT_EVT_17

Parameter	Description
	• CT_EVT_18 • CT_EVT_19 • CT_EVT_20 • CT_EVT_21 • CT_EVT_22 • CT_EVT_23 • CT_EVT_24 • CT_EVT_25 • CT_EVT_26 • CT_EVT_27 • CT_EVT_28 • CT_EVT_29 • CT_EVT_30 • CT_EVT_31 • CT_EVT_32 • CT_EVT_33 • CT_EVT_34 • CT_EVT_35 • CT_EVT_36 • CT_EVT_37

Return values

none

Example

```
RSI_CT_EdgeLevelEventControl(CT0,CT_EVT_2);
```

16.3.28 RSI_CT_SetTimerMuxSelect

Source File : rsi_ct.h

Path : Redpine_MCU_V1.x.x\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_CT_SetTimerMuxSelect (RSI_CT_MUX_REG_T *pCTMux, uint8_t timerIns)
```

Description

This API is used to Select Timer using mux.

Parameters

Parameter	Description
pCTMux	Pointer to the CT Mux instance register area
timerIns	If 0 - instance,if 1 - instance

Return values

none

Example

```
RSI_CT_SetTimerMuxSelect(CT0,1);
```

16.3.29 RSI_CT_PeripheralReset

Source File : rsi_ct.h

Path : Redpine_MCU_V1.x.x\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_CT_PeripheralReset (RSI_CT_T *pCT, boolean_t counterNum)
```

Description

This API is use to Resume the HALT operation of counter with I/O events.

Parameters

Parameter	Description
pCT	Pointer to the CT instance register area
counterNum	Coutner 0/1

Return values

none

Example

```
#define   COUNTER_0   0
RSI_CT_PeripheralReset(CT0,COUNTER_0);
```

16.3.30 RSI_CT_StartSoftwareTrig

Source File : rsi_ct.h

Path : Redpine_MCU_V1.x.x\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_CT_StartSoftwareTrig (RSI_CT_T *pCT, boolean_t counterNum)
```

Description

This API is use to starts the Counter form software register.

Parameters

Parameter	Description
pCT	Pointer to the CT instance register area
counterNum	Coutner 0/1

Return values

none

Example

```
#define   COUNTER_0   0
/*Start counters from s/w*/
RSI_CT_StartSoftwareTrig(CT1, COUNTER_0);
```

16.3.31 RSI_CT_OCUModeSet

Source File : rsi_ct.h

Path : Redpine_MCU_V1.x.x\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_CT_OCUModeSet (RSI_CT_T *pCT, boolean_t counterNum)
```

Description

This API is use to OCU mode configuration setting.

Parameters

Parameter	Description
pCT	Pointer to the CT instance register area
counterNum	Coutner 0/1

Return values

none

Example

```
#define   COUNTER_0   0
/* OCU mode set */
RSI_CT_OCUModeSet(CT1, COUNTER_0);
```

16.3.32 RSI_CT_SetMatchCount

Source File : rsi_ct.h

Path : Redpine_MCU_V1.x.x\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_CT_SetMatchCount (RSI_CT_T *pCT, uint32_t value, boolean_t
counterMode, boolean_t counterNum)
```

Description

This API is use to sets the match load value for counter 0 and coutner 1.

Parameters

Parameter	Description
pCT	Pointer to the CT instance register area
value	Match register or match buffer register value,this parameter can be the logical OR of the

Parameter	Description
	• COUNTER_0_MATCH : Counter 0 match value (0x0 to 0xFFFF) • COUNTER_1_MATCH : Counter 1 match value (0x0 to 0xFFFF)
counterMode	counterMode 0/1
counterNum	Coutner 0/1

Return values

none

Example

```
/* set the Peak Match value for CT0 coutner0 & counter 1 */
RSI_CT_SetMatchCount(CT0, rate,0,0);
```

16.3.33 RSI_CT_CaptureRead

Source File : rsi_ct.h

Path : Redpine_MCU_V1.x.x\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE uint16_t RSI_CT_CaptureRead (RSI_CT_T *pCT, boolean_t counterNum)
```

Description

This API is use to gets the captured counter value.

Parameters

Parameter	Description
pCT	Pointer to the CT instance register area
counterNum	Coutner 0/1

Return values

Return coutner value at the time of capture event occurs

Example

```
#define   COUNTER_0   0
/* cpture read */
RSI_CT_CaptureRead(CT0,COUNTER_0);
```

16.3.34 RSI_CT_GetCounter

Source File : rsi_ct.h

Path : Redpine_MCU_V1.x.x\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE uint32_t RSI_CT_GetCounter (RSI_CT_T *pCT, boolean_t counterNum, boolean_t
mode)
```

Description

This API is use to gets the captured counter value.

Parameters

Parameter	Description
pCT	Pointer to the CT instance register area
counterNum	Coutner 0/1
mode	mode 0/1

Return values

Return the counter value

Example

```
uint32_t Stop_Counter = 0;
/* gets the captured counter value. */
Stop_Counter=    RSI_CT_GetCounter(CT0,0,_32_BIT);
```

16.3.35 RSI_CT_SetCounerSync

Source File : rsi_ct.h

Path : Redpine_MCU_V1.x.x\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_CT_SetCounerSync (RSI_CT_T *pCT, uint8_t syncCounter, boolean_t
counterNum)
```

Description

This API is use to sets the captured counter value.

Parameters

Parameter	Description
pCT	Pointer to the CT instance register area
syncCounter	set the counter number to be in sync
counterNum	Coutner 0/1

Return values

none

Example

```
/* set syn counter */
RSI_CT_SetCounerSync(CT0,1,1);
```

16.3.36 RSI_CT_OCUHighLowToggleSelect

Source File : rsi_rom_ct.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void  RSI_CT_OCUHighLowToggleSelect(RSI_CT_T *pCT,boolean_t lowHigh,
```

```
                                                    boolean_t counterNum,uint8_t outputSel)
```

Description

This API is used to toggle output state of OCU.

Parameters

Parameter	Description
pCT	Pointer to the CT instance register area
lowHigh	Output low/high state
counterNum	Coutner 0/1
outputSel	OCU possibilities(0 to 7)

Return values

none

Example

```
/* OCU toggle */
RSI_CT_OCUHighLowToggleSelect(CT0,1,1,0);
```

16.3.37 RSI_CT_WFGControlConfig

Source File : rsi_rom_ct.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_CT_WFGControlConfig(RSI_CT_T *pCT, WFG_PARAMS_T ctrlReg )
```

Description

This API is used to control output state of WFG.

Parameters

Parameter	Description
pCT	Pointer to the CT instance register area
ctrlReg	Variable to a structure of type WFG_PARAMS_T

Return values

ERROR_CT_INVALID_ARG : If WFG_PARAMS_T structure parameters are invalid

RSI_OK : If process is done successfully

Example

```
WFG_PARAMS_T  wfg_config;
/* configuration value */
wfg_config.output0_tgl0_sel=7;
wfg_config.output0_tgl1_sel=7;
wfg_config.tgl_cnt0_peak=0xFF;
wfg_config.output1_tgl0_sel=7;
wfg_config.output1_tgl1_sel=7;
```

```
wfg_config.tgl_cnt1_peak=0xFF;

/*WFG control configuration */
RSI_CT_WFGControlConfig(CT0,wfg_config);
```

16.3.38 RSI_CT_OCUControl

Source File : rsi_rom_ct.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_CT_OCUControl( RSI_CT_T *pCT,boolean_t counterNum,boolean_t
dmaEn,
                                         OCU_PARAMS_T *pOCUparams,RSI_CT_CALLBACK_T *pCB)
```

Description

This API is used to controls OCU operation.

Parameters

Parameter	Description
pCT	Pointer to the CT instance register area
counterNum	Counter 0/1
dmaEn	DMA enable flag • 0 : not enable DMA. • 1 : DMA enable.
pOCUparams	Pointer to the OCU parameters structure and its members are as below OCU_PARAMS_T • CompareVal1_0 : Threshold one of Counter 0,possible values (0x0 to 0xFFFF) • CompareVal2_0 : Threshold two of Counter 0,possible values (0x0 to 0xFFFF) • CompareVal1_1 : Threshold two of Counter 1,possible values (0x0 to 0xFFFF) • CompareVal2_1 : Threshold two of Counter 1,possible values (0x0 to 0xFFFF) • CompareVal1_0 : Threshold two of Counter 0,possible values (0x0 to 0xFFFF) • CompareNextVal2_0 : next compare value of counter 0,possible values (0x0 to 0xFFFF) • CompareNextVal1_1 : next compare value of counter 1,possible values (0x0 to 0xFFFF) • CompareNextVal2_1 : next compare value of counter 1,possible values (0x0 to 0xFFFF) • SyncValue : Holds the starting point of counters for synchronization purpose This is logical OR of sync values for counter 0& 1, possible values for two counters (0x0 to 0xFFFF).
pCB	Pointer to CT callback structure

Return values

ERROR_CT_INVALID_ARG : If passed parameters are invalid

RSI_OK : If process is done successfully

Example

```
OCU_PARAMS_T vsOCUparams;
RSI_CT_CALLBACK_T vsCB;
/* configure OCU control */
```

```
vsOCUparams .CompareVal1_0      = 0x800;
vsOCUparams .CompareVal2_0      = 0x1000;
vsOCUparams .CompareVal1_1      = 0x800;
vsOCUparams .CompareVal2_1      = 0x1000;
vsOCUparams.SyncWith0           = 0x50;
vsOCUparams.SyncWith1           = 0x50;
/* OCU control */
RSI_CT_OCUControl(CT0,1,1,&vsOCUparams,&vsCB);
```

16.3.39 RSI_CT_WFGComapreValueSet

Source File : rsi_rom_ct.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_CT_WFGComapreValueSet( RSI_CT_T *pCT,boolean_t counterNum,
                                                 OCU_PARAMS_T *pOCUparams)
```

Description

This API is used to controls WFG operation.

Parameters

Parameter	Description
pCT	Pointer to the CT instance register area
counterNum	Counter 0/1
pOCUparams	Pointer to the OCU parameters structure and its members are as below OCU_PARAMS_T • CompareVal1_0 : Threshold one of Counter 0,possible values (0x0 to 0xFFFF) • CompareVal2_0 : Threshold two of Counter 0,possible values (0x0 to 0xFFFF) • CompareVal1_1 : Threshold one of Counter 1,possible values (0x0 to 0xFFFF) • CompareVal2_1 : Threshold one of Counter 1,possible values (0x0 to 0xFFFF) • SyncValue : Holds the starting point of counters for synchronization purpose This is logical OR of sync values for counter 0& 1 possible values for two counters (0x0 to 0xFFFF)

Return values

ERROR_CT_INVALID_ARG : If passed parameters are invalid

RSI_OK : If process is done successfully

Example

```
OCU_PARAMS_T vsOCUparams;
RSI_CT_CALLBACK_T vsCB;
/* configure OCU control */
vsOCUparams .CompareVal1_0      = 0x800;
vsOCUparams .CompareVal2_0      = 0x1000;
vsOCUparams .CompareVal1_1      = 0x800;
vsOCUparams .CompareVal2_1      = 0x1000;
vsOCUparams.SyncWith0           = 0x50;
vsOCUparams.SyncWith1           = 0x50;
/* compare value set */
RSI_CT_WFGComapreValueSet(CT0,1,&vsOCUparams);
```

16.3.40 RSI_CT_StopEventConfig

Source File : rsi_ct.h

Path: Redpine_MCU_V1.x.x\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_CT_StopEventConfig (RSI_CT_T *pCT, uint32_t andValue, uint32_t
orValue)
```

Description

This API is used to Configure AND/OR event for stop counter operation.

Parameters

Parameter	Description
pCT	Pointer to the CT instance register area
andValue	• AND_COUNTER0(valid,event) : pass the required valid and events to this macro for counter0 • AND_COUNTER1(valid,event) : pass the required valid and events to this macro for counter1
orValue	• OR_COUNTER0(valid,event) :pass the required valid and events to this macro for counter0 • OR_COUNTER1(valid,event) :pass the required valid and events to this macro for counter1

Return values

none

Example

```
RSI_CT_StartEventConfig(CT0,AND_COUNTER0(0x3,0xF),OR_COUNTER0(0x3,0xF));
```

16.3.41 RSI_CT_StopEventSelect

Source File : rsi_ct.h

Path: Redpine_MCU_V1.x.x\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_CT_StopEventSelect (RSI_CT_T *pCT, uint32_t value)
```

Description

This API is used to select the input event to stop counter.

Parameters

Parameter	Description
pCT	Pointer to the CT instance register area
value	This parameter can be select CT, possible value is as below. • CT_EVT_0 • CT_EVT_1 • CT_EVT_2 • CT_EVT_3

	• CT_EVT_4 • CT_EVT_5 • CT_EVT_6 • CT_EVT_7 • CT_EVT_8 • CT_EVT_9 • CT_EVT_10 • CT_EVT_11 • CT_EVT_12 • CT_EVT_13 • CT_EVT_14 • CT_EVT_15 • CT_EVT_16 • CT_EVT_17 • CT_EVT_18 • CT_EVT_19 • CT_EVT_20 • CT_EVT_21 • CT_EVT_22 • CT_EVT_23 • CT_EVT_24 • CT_EVT_25 • CT_EVT_26 • CT_EVT_27 • CT_EVT_28 • CT_EVT_29 • CT_EVT_30 • CT_EVT_31 • CT_EVT_32 • CT_EVT_33 • CT_EVT_34 • CT_EVT_35 • CT_EVT_36 • CT_EVT_37

Return values

none

Example

```
/* Event 3 rise edge for stop counter operation */
RSI_CT_StopEventSelect(CT0, CT_EVT_3)
```

17 Cyclic Redundancy Check (CRC)

17.1 Overview

This section explains how to configure and use the Cyclic Redundancy Check using Redpine MCU SAPIs.

17.2 Programming sequence

```
#include "rsi_chip.h" /* Redpine_MCU_Vx.y.z\Host_MCU\common\chip\inc */
#include "rsi_rom_crc.h /* Redpine_MCU_Vx.y.z\Host_MCU\library\rom_driver\inc */

RSI_CRC_PARAMS_T crc_params;

int main()
{
 crc_params.polynomial=0xd95eaae5;
 crc_params.polyWidth =32;
 crc_params.lfsrVal=0x0000;
 crc_params.widthType=0;
 crc_params.dinWidth=32;
 crc_params.numBytes =4;
 crc_params.aempty=6 ;
 crc_params.afull =2;
 crc_params.InputData =0x6e95c16f;
 crc_params.swapDin =1;
 crc_params.useUdma =0;
 crc_params.swapLfsr =1;
 crc_params.inputLocation=0;
 /*to clears the FIFO and settles all the state machines to their IDLE */
 RSI_CRC_SetGenControl(CRC);
 /* Enable Clock FOr CRC Engine*/
 RSI_CLK_PeripheralClkEnable1(M4CLK , CRC_CLK_ENABLE_M4);
 RSI_CRC_Polynomial(CRC, &crc_params );
 RSI_CRC_Polynomial_Width(CRC, &crc_params );
 RSI_CRC_LfsrInit(CRC, &crc_params );
 RSI_CRC_UseSwapped_Init(CRC,&crc_params );
 RSI_CRC_Set_DataWidthType(CRC,&crc_params  );
 RSI_CRC_SetFifoThresholds(CRC,&crc_params);
 RSI_CRC_WriteData(CRC, &crc_params);
 crc_result=RSI_Monitor_CRCcalc(CRC,&crc_params);
 status=RSI_CRC_GetGenStatus(CRC);
 while(1);
}
```

Code Block 11 Cyclic Redundancy Check Example

17.3

17.4 API Description

17.4.1 RSI_CRC_SetGenControl

Source File : rsi_rom_crc.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void  RSI_CRC_SetGenControl( CRC_Type *pCRC)
```

Description

This API is used to clears the FIFO and settles all the state machines to their IDLE.

Parameter

Parameter	Description
pCRC	Pointer to the CRC Register set instance area.

Return values

None

Example

```
/* clear fifo and reset machines */
 RSI_CRC_SetGenControl(CRC);
```

17.4.2 RSI_CRC_Polynomial

Source File : rsi_rom_crc.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void  RSI_CRC_Polynomial( CRC_Type *pCRC, RSI_CRC_PARAMS_T *pCRCParams )
```

Description

This API is used to load the polynomial value into the polynomial reg.

Parameter

Parameter	Description
pCRC	Pointer to the CRC Register set instance area.
pCRCParams	Pointer to the CRC Parameters structure. The possible variable is polynomial

Return values

None

Example

```
/* load the polynomial value into the polynomial reg */
RSI_CRC_Polynomial(CRC, &crc_params );
```

17.4.3 RSI_CRC_Polynomial_Width

Source File : rsi_rom_crc.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE uint32_t  RSI_CRC_Polynomial_Width( CRC_Type *pCRC, RSI_CRC_PARAMS_T
*pCRCParams )
```

Description

This API is used to load the width of the polynomial .Number of bits/width of the polynomial has to be written here for the computation of final CRC. If a new width has to be configured, clear the existing length first by writing 0x1f in polynomial_ctrl_reset register.

Parameters

Parameter	Description
pCRC	Pointer to the CRC Register set instance area.
pCRCParams	Pointer to the CRC Parameters structure. The possible variable is polynomial.

Return values

None

Example

```
/* declaration */
RSI_CRC_PARAMS_T crc_params;
/*   used to load the width of the polynomail */
crc_params.polynomial=0xd95eaae5;
crc_params.polyWidth =32;
crc_params.lfsrVal=0x0000;
crc_params.widthType=0;
crc_params.dinWidth=32;
crc_params.numBytes =4;
crc_params.aempty=6 ;
crc_params.afull =2;
crc_params.InputData =0x6e95c16f;
crc_params.swapDin =1;
crc_params.useUdma =0;
crc_params.swapLfsr =1;
crc_params.inputLocation=0;

 RSI_CRC_Polynomial_Width(CRC, &crc_params );
```

17.4.4 RSI_CRC_LfsrInit

Source File : rsi_rom_crc.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void  RSI_CRC_LfsrInit( CRC_Type *pCRC, RSI_CRC_PARAMS_T *pCRCParams )
```

Description

This API is used to initialize the LFSR Value. When ever the LFSR need to be Initialized this has to be updated with the Init value.

Parameters

Parameter	Description
pCRC	Pointer to the CRC Register set instance area.

Parameter	Description
pCRCParams	Pointer to the CRC Parameters structure. The possible variable is lfsr_val.

Return values

None

Example

```
/*  initialize the LFSR Value */
RSI_CRC_LfsrInit(CRC, &crc_params );
```

17.4.5 RSI_CRC_Use_Swapped_Init

Source File : rsi_rom_crc.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE uint32_t  RSI_CRC_Use_Swapped_Init( CRC_Type *pCRC ,RSI_CRC_PARAMS_T
*pCRCParams)
```

Description

This API is used for swapped init value. If this is set bit swapped version of LFSR init value will be loaded / initialized to LFSR state.

Parameters

Parameter	Description
pCRC	Pointer to the CRC Register set instance area
pCRCParams	Pointer to the CRC Parameters structure

Return values

None

Example

```
/*    used for swapped init value */
RSI_CRC_Use_Swapped_Init(CRC,&crc_params );
```

17.4.6 RSI_CRC_Set_DataWidthType

Source File : rsi_rom_crc.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE uint32_t  RSI_CRC_Set_DataWidthType( CRC_Type *pCRC, RSI_CRC_PARAMS_T
*pCRCParams )
```

Description

This API is used to Set and control the data width types.

Parameters

Parameter	Description
pCRC	Pointer to the CRC Register set instance area.
pCRCParams	Pointer to the CRC Parameters structure. The possible parameter is num_bytes. When only the data width type is DIN_WIDTH_FROM_CNT then only uses this variable.

Return values

return RSI_OK on success and failure return error code.

Example

```
/*  used to Set and control the data width types. */
RSI_CRC_Set_DataWidthType(CRC,&crc_params );
```

17.4.7 RSI_CRC_SetFifoThresholds

Source File : rsi_rom_crc.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE uint32_t RSI_CRC_SetFifoThresholds( CRC_Type *pCRC, RSI_CRC_PARAMS_T
*pCRCParams)
```

Description

This API is used to Set the FIFO Threshold Levels.

Parameters

Parameter	Description
pCRC	Pointer to the CRC Register set instance area.
pCRCParams	Pointer to the CRC Parameters structure. The possible parameters are aempty and afull. Max value is 15 for both the threshold levels

Return values

return RSI_OK on success and failure return error code

Example

```
/*  used to Set the FIFO Threshold Levels. */
RSI_CRC_SetFifoThresholds(CRC,&crc_params);
```

17.4.8 RSI_CRC_WriteData

Source File : rsi_rom_crc.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE uint32_t  RSI_CRC_WriteData( CRC_Type *pCRC, RSI_CRC_PARAMS_T *pCRCParams,
uint32_t data)
```

Description

This API is used to Set the Input data Information for CRC Calculation

Parameters

Parameter	Description
pCRC	Pointer to the CRC Register set instance area.
pCRCParams	Pointer to the CRC Parameters structure. The possible parameters are aempty and afull.

Return values

return RSI_OK on success and failure return error code.

Example

```
/*  used to Set the Input data Information for CRC Calculation */
RSI_CRC_WriteData(CRC, &crc_params);
```

17.4.9 RSI_Monitor_CRCcalc

Source File : rsi_rom_crc.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE uint32_t  RSI_Monitor_CRCcalc( CRC_Type *pCRC, RSI_CRC_PARAMS_T *pCRCParams)
```

Description

This API is used to monitor the CRC Caluclation status and the returns the CRC Value.

Parameters

Parameter	Description
pCRC	Pointer to the CRC Register set instance area.
pCRCParams	Pointer to the CRC Parameters structure. The possible parameter is width_type. This Params always 2 when we want Max value is 15 for both the threshold levels.

Return values

CRC Value to be returned.

Example

```
  uint32_t crc = 0;
/*  used to Set the Input data Information for CRC Calculation */
 crc =RSI_Monitor_CRCcalc(RegInst,&crc_params);
```

17.4.10 RSI_CRC_GetGenStatus

Source File : rsi_rom_crc.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE uint32_t  RSI_CRC_GetGenStatus( CRC_Type *pCRC)
```

Description

This API is used to get the general status of the crc signals.

Parameter

Parameter	Description
pCRC	Pointer to the CRC Register set instance area.

Return values

Returns Crc status signals.

Example

```
/* variable for general status */
uint32_t status;
/*  used to get the general status of the crc signals */
 status=RSI_CRC_GetGenStatus(CRC)
```

17.4.11 RSI_CRC_LfsrDynamicWrite

Source File : rsi_rom_crc.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void  RSI_CRC_LfsrDynamicWrite(CRC_Type *pCRC, RSI_CRC_PARAMS_T *pCRCParams
)
```

Description

This API can be used for writing the LFSR state directly. .

Parameters

Parameter	Description
pCRC	Pointer to the CRC Register set instance area.
pCRCParams	Pointer to the CRC Parameters structure. The possible parameter is width_type. This Params always 2 when we want Max value is 15 for both the threshold levels.

Return values

None.

Example

```
  uint32_t crc = 0;
/*  used to Set the Input data Information for CRC Calculation */
 crc =RSI_Monitor_CRCcalc(RegInst,&crc_params);
```

17.4.12 RSI_CRC_GetFifoStatus

Source File : rsi_rom_crc.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE uint32_t  RSI_CRC_GetFifoStatus(CRC_Type *pCRC)
```

Description

This API is used to get the fifo status of the crc occupancy.

Parameters

Parameter	Description
pCRC	Pointer to the CRC Register set instance area.

Return values

Returns Crc Fifo status signals

Example

```
uint32_t status=0;
/*  used to get the fifo status of the crc occupancy */
status=RSI_CRC_GetFifoStatus(CRC);
```

17.4.13 RSI_CRC_ResetFifo

Source File : rsi_rom_crc.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void  RSI_CRC_ResetFifo(CRC_Type *pCRC)
```

Description

This API is used to Reset fifo pointer. This clears the FIFO.When this is set.

Parameters

Parameter	Description
pCRC	Pointer to the CRC Register set instance area.

Return values

None.

Example

```
/* Reset fifo pointer */
RSI_CRC_ResetFifo (CRC);
```

18 eFUSE

18.1 Overview

This section explains how to configure and use the eFuse using Redpine MCU SAPIs.

18.2 Programming sequence

```
#include "rsi_chip.h" /* Redpine_MCU_Vx.y.z\Host_MCU\common\chip\inc */
#include "rsi_board.h" /* Redpine_MCU_Vx.y.z\Host_MCU\common\board\inc */
uint8_t i=1;
int main()
{
     uint8_t u8DataZero,u8DataOne ,u8DataTwo;
     /*Configures the system default clock and power configurations*/
     SystemCoreClockUpdate();
     while(i)
     {
        /*NOTE : in eFUSE programming, programming only onces is possible by default all
locations are zeros.
           Each write operation sets the bits in particular location it is only one time
programmable and those
         particular locations will be corrupted ,If you wants to run these example do
manually "i" value to zero */
     }
     if(CLOCK > 120000000)
     {
        /*Configure the prefetch and registering when SOC clock is more than 120Mhz*/
        /*Configure the SOC PLL to 220MHz*/
        ICACHE2_ADDR_TRANSLATE_1_REG  = BIT(21);  //Icache registering when clk freq more
than 120
        /*When set, enables registering in M4-TA AHB2AHB. This will have performance
penalty. This has to be set above 100MHz*/
        MISC_CFG_SRAM_REDUNDANCY_CTRL = BIT(4);
        MISC_CONFIG_MISC_CTRL1 |= BIT(4); //Enable Register RSI as clock frequency is 200
Mhz

        MISC_QUASI_SYNC_MODE |= BIT(6);
        /*Enable Inter subsystem memory Registering as m4_soc_clk clock is going to TAss.
above 120Mhz we have to enable this
         .Also enabling prefetch as when registering by default prefetch is expected to be
enabled to save the cycles which
         are lost in registering*/
        MISC_QUASI_SYNC_MODE |= (BIT(6) | BIT(7));
     }
    /* Switch M4 SOC clock to Reference clock */
    /* Default keep M4 in reference clock     */

    /* Configure the m4_soc clocks */
    RSI_CLK_M4SocClkConfig(M4CLK,M4_ULPREFCLK,0);

    /* Set the Soc PLL clock to particular frequency */
    RSI_CLK_SetSocPllFreq(M4CLK,CLOCK,40000000);

    /* Configure the m4_soc clocks */
    RSI_CLK_M4SocClkConfig(M4CLK,M4_SOCPLLCLK,0);

    /*  Enable the peripheral clocks for SET3 register */
    RSI_CLK_PeripheralClkEnable3(M4CLK,(BIT(5)|BIT(19)));

    /*Writing 0XAF Data at address 0x00000*/
    /*Addredd :0x00000 -> Data : 1010 1111  */

    /* Write WRITE_ADD_0 , bit location 0*/
    RSI_EFUSE_WriteBit(EFUSE ,WRITE_ADD_0 , 0,HOLD);
```

```
/* Write WRITE_ADD_0 , bit location 1*/
RSI_EFUSE_WriteBit(EFUSE ,WRITE_ADD_0 , 1,HOLD);

/* Write WRITE_ADD_0 , bit location 2*/
RSI_EFUSE_WriteBit(EFUSE ,WRITE_ADD_0 , 2,HOLD);

/* Write WRITE_ADD_0 , bit location 3*/
RSI_EFUSE_WriteBit(EFUSE ,WRITE_ADD_0 , 3,HOLD);

/* Write WRITE_ADD_0 , bit location 5*/
RSI_EFUSE_WriteBit(EFUSE ,WRITE_ADD_0 , 4,HOLD);

/* Write WRITE_ADD_0 , bit location 7*/
RSI_EFUSE_WriteBit(EFUSE ,WRITE_ADD_0 , 5,HOLD);


/*Writing 0X2A Data at address 0x00001*/
/*Addredd :0x00001 -> Data : 0010 1010  */

/* Write WRITE_ADD_1 , bit location 1*/
RSI_EFUSE_WriteBit(EFUSE ,WRITE_ADD_1 , 1,HOLD);

/* Write WRITE_ADD_1 , bit location 3*/
RSI_EFUSE_WriteBit(EFUSE ,WRITE_ADD_1 , 3,HOLD);

/* Write WRITE_ADD_1 , bit location 5*/
RSI_EFUSE_WriteBit(EFUSE ,WRITE_ADD_1 , 5,HOLD);


/* Writing 0X55 Data at address 0x00001*/
/* Addredd :0x00002 -> Data : 0101 0101  */
/* Write WRITE_ADD_2 , bit location 0*/
RSI_EFUSE_WriteBit(EFUSE ,WRITE_ADD_2 , 0,HOLD);

/* Write WRITE_ADD_2 , bit location 2*/
RSI_EFUSE_WriteBit(EFUSE ,WRITE_ADD_2 , 2,HOLD);

/* Write WRITE_ADD_2 , bit location 4*/
RSI_EFUSE_WriteBit(EFUSE ,WRITE_ADD_2 , 4,HOLD);

/* Write WRITE_ADD_2 , bit location 6*/
RSI_EFUSE_WriteBit(EFUSE ,WRITE_ADD_2 , 6,HOLD);

/*eFUSE read Operation*/

/* Read byte fRSI address 0 in FSM read mode */
RSI_EFUSE_FsmReadByte(EFUSE ,WRITE_ADD_0, &u8DataZero,CLOCK);
if(u8DataZero==0xaf)
{
    /*  Prints on hyper-terminal  */
    DEBUGOUT("MATCH\n");
}
else
{
    /*  Prints on hyper-terminal  */
    DEBUGOUT("UNMATCH\n");
}

/* Read byte fRSI address 1 in FSM read mode */
RSI_EFUSE_FsmReadByte(EFUSE ,WRITE_ADD_1, &u8DataOne,CLOCK);

if(u8DataOne==0x2a)
{
    /*  Prints on hyper-terminal  */
    DEBUGOUT("MATCH\n");
}
else
{
    /*  Prints on hyper-terminal  */
    DEBUGOUT("UNMATCH\n");
```

```
    }
    /* Read byte fRSI address 2 in FSM read mode */
    RSI_EFUSE_FsmReadByte(EFUSE ,WRITE_ADD_2, &u8DataTwo,CLOCK);
    if(u8DataTwo==0x55)
    {
        /*  Prints on hyper-terminal  */
        DEBUGOUT("MATCH\n");
    }
    else
    {
        /*  Prints on hyper-terminal  */
        DEBUGOUT("UNMATCH\n");
    }

    while(1);
}
```

Code Block 12 EFUSE Example

18.3 API Descriptions

18.3.1 RSI_EFUSE_WriteBit

Source File : rsi_rom_efuse.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc

Prototype

```
STATIC INLINE error_t RSI_EFUSE_WriteBit(EFUSE_Type *pstcEfuse , uint16_t u16Addr ,
uint8_t u8BitPos,uint32_t hold_time)
```

Description

Any Bit in this macro can be programmed in any order by raising STROBE high for a period specified by TPGM with a proper address selected.There is only one programming scheme,which is single bit programming.

Parameters

Parameter	Description
pstcEfuse	Pointer to the EFUSE register instance
u16Addr	Address of EFUSE to the data
u8BitPos	Position of bit to be written in 8 bit location in address
hold_time	Count value depends on clock frequency of EFUSE controller Hold time will be computed like below note:Strobe signal Clear count value in direct access mode. Value depends on APB clock frequency of EFUSE controller. ex:clock =32mhz 1/32=0.03125 2/0.03125 =0x40 for 32MHZ clock ,hold time is 0x40

Return values

Nonzero : If fails

0 : If success

Example

```
#define WRITE_ADD_0  0x0000
#define BIT_POS      0
#define HOLD_TIME    0x140 //!(for 160mhz clock)
RSI_EFUSE_WriteBit(EFUSE,WRITE_ADD_0,BIT_POS,HOLD_TIME);
```

18.3.2 RSI_EFUSE_FsmReadByte

Source File : rsi_rom_efuse.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc

Prototype

```
STATIC INLINE error_t RSI_EFUSE_FsmReadByte(EFUSE_Type *pstcEfuse , uint16_t u16Addr ,
uint8_t *pu8Byte ,uint32_t SocClk)
```

Description

This API is used to read the data from 32x8 byte eFUSE memory(OTP) in fsm mode

Parameters

Parameter	Description
pstcEfuse	Pointer to the EFUSE register instance
u16Addr	Byte address to be read
pu8Byte	Pointer to hold the read byte
SocClk	Clock frequency of EFUSE controller

Return values

Non zero : If fails

0 : If success

Example

```
#define WRITE_ADD_0  0x0000
#define SOC_CLOCK    160 //160mhz
uint8_t u8DataZero;
RSI_EFUSE_FsmReadByte(EFUSE,WRITE_ADD_0, &u8DataZero,SOC_CLOCK);
```

18.3.3 RSI_EFUSE_Enable

Source File : rsi_rom_efuse.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc

Prototype

```
STATIC INLINE void  RSI_EFUSE_Enable(EFUSE_Type* pstcEfuse)
```

Description

This API Is Used To Enable The EFUSE.

Parameter

Parameter	Description
pstcEfuse	Pointer to the EFUSE register instance

Return values

None

Example

```
/* enable EFUSE */
RSI_EFUSE_Enable(EFUSE);
```

18.3.4 RSI_EFUSE_Disable

Source File : rsi_rom_efuse.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc

Prototype

```
STATIC INLINE void  RSI_EFUSE_Disable(EFUSE_Type* pstcEfuse)
```

Description

This API is used to Disable the EFUSE.

Parameter

Parameter	Description
pstcEfuse	Pointer to the EFUSE register instance

Return values

None

Example

```
/* Disable EFUSE */
RSI_EFUSE_Disable(EFUSE);
```

18.3.5 RSI_EFUSE_ReadData

Source File : rsi_rom_efuse.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc

Prototype

```
STATIC INLINE uint8_t  RSI_EFUSE_ReadData(EFUSE_Type* pstcEfuse)
```

Description

This API is used to read the eFUSE data.

Parameter

Parameter	Description
pstcEfuse	Pointer to the EFUSE register instance

Return values

eFUSE data

Example

```
uint8_t Data=0;
/*  read the eFUSE data */
Data=RSI_EFUSE_ReadData(EFUSE);
```

18.3.6 RSI_EFUSE_WriteAddr

Source File : rsi_rom_efuse.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc

Prototype

```
STATIC INLINE void  RSI_EFUSE_WriteAddr(EFUSE_Type* pstcEfuse , uint16_t u16Addr)
```

Description

This API is used to set the eFUSE address for read and write operations.

Parameters

Parameter	Description
pstcEfuse	Pointer to the EFUSE register instance
u16Addr	Address of EFUSE to the data

Return values
None

Example

```
#define WRITE_ADD_0  0x0000
#define BIT_POS      0
#define HOLD_TIME    0x140 //!(for 160mhz clock)
RSI_EFUSE_WriteAddr(EFUSE,WRITE_ADD_0);
```

18.3.7 RSI_EFUSE_MemMapReadByte

Source File : rsi_rom_efuse.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc

Prototype

```
STATIC INLINE error_t RSI_EFUSE_MemMapReadByte(EFUSE_Type *pstcEfuse , uint16_t u16Addr ,
uint8_t *pu8Byte ,uint32_t SocClk)
```

Description

This API is used to read the data from 32x8 byte eFUSE memory(OTP) in memory mapped mode

Parameters

Parameter	Description

Parameter	Description
pstcEfuse	Pointer to the EFUSE register instance
u16Addr	Byte address to be read
pu8Byte	Pointer to hold the read byte
SocClk	Clock frequency of EFUSE controller

Return values

Non zero : If fails

0 : If success

Example

```
#define WRITE_ADD_0  0x0000
#define SOC_CLOCK    160 //160mhz
uint8_t u8DataZero;
RSI_EFUSE_MemMapReadByte(EFUSE,WRITE_ADD_0, &u8DataZero,SOC_CLOCK);
```

18.3.8 RSI_EFUSE_MemMapReadWord

Source File : rsi_rom_efuse.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc

Prototype

```
STATIC INLINE error_t RSI_EFUSE_MemMapReadWord(EFUSE_Type *pstcEfuse , uint16_t u16Addr ,
uint16_t *pu16Word ,uint32_t SocClk)
```

Description

This API is used to Read the 1 word(16 bits) of data to EFUSE macro

Parameters

Parameter	Description
pstcEfuse	Pointer to the EFUSE register instance
u16Addr	Byte address to be read
pu16Word	Pointer to the buffter to hold the read byte
SocClk	Clock frequency of EFUSE controller

Return values

Non zero : If fails

0 : If success

Example

```
#define SOC_CLOCK     160 //!160MHZ
uint16_t u16DataZero;
RSI_EFUSE_MemMapReadWord(EFUSE,WRITE_ADD_0, &u16DataZero,SOC_CLOCK);
```

19 Enhanced General Purpose Input Output (EGPIO)

19.1 Overview

This section explains how to configure and use the Enhanced General Purpose Input Output using Redpine MCU SAPIs.

19.2 Programming sequence

```
#include "rsi_chip.h" /* Redpine_MCU_Vx.y.z\Host_MCU\common\chip\inc */

#define PININT_IRQ_HANDLER         IRQ059_Handler                   /* GPIO interrupt IRQ
function name             */
#define PININT_NVIC_NAME           EGPIO_PIN_7_IRQn                 /* GPIO interrupt NVIC
interrupt name          */
#define M4_GPIO_PORT               0                                /* GPIO port number
*/
#define M4_GPIO_PIN                6                                /* GPIO pin number
*/
#define PIN_INT                    7                                /* Pin  interrupt
number(0 to 7)               */

/**
 * @brief    Interrupt handler
 * @return    Nothing
 */
void PININT_IRQ_HANDLER(void)
{
    uint32_t gintStatus;

    /*get interrupt status*/
    gintStatus=RSI_EGPIO_GetIntStat(EGPIO,PIN_INT);

    if((gintStatus &EGPIO_PIN_INT_CLR_RISING ) || (gintStatus &EGPIO_PIN_INT_CLR_FALLING
))
    {
        /*clear interrupt*/
        RSI_EGPIO_IntClr(EGPIO, PIN_INT ,INTERRUPT_STATUS_CLR);
    }
    else
    {
        RSI_EGPIO_IntMask(EGPIO , PIN_INT);
    }
    return ;
}

/**
 * @brief  Main program.
 * @param  None
 * @retval None
 */
int main(void)
{
    int forever = 1;
    /*Configures the system default clock and power configurations*/
     SystemCoreClockUpdate();

    /*Enable clock for EGPIO module*/
    RSI_CLK_PeripheralClkEnable(M4CLK,EGPIO_CLK,ENABLE_STATIC_CLK);

    /*PAD selection*/
    RSI_EGPIO_PadSelectionEnable(1);

    /*REN enable */
    RSI_EGPIO_PadReceiverEnable(M4_GPIO_PIN);
```

```
    /*Configure default GPIO mode(0) */
    RSI_EGPIO_SetPinMux(EGPIO,M4_GPIO_PORT ,M4_GPIO_PIN,EGPIO_PIN_MUX_MODE0);

    /*Selects the pin interrupt for the GPIO*/
    RSI_EGPIO_PinIntSel(EGPIO, PIN_INT , M4_GPIO_PORT, M4_GPIO_PIN);

    /*Configures the edge /level interrupt*/
    RSI_EGPIO_SetIntLowLevelEnable(EGPIO,PIN_INT);

    /*Unmask the  interrupt*/
    RSI_EGPIO_IntUnMask(EGPIO , PIN_INT);

    /*NVIC enable */
    NVIC_EnableIRQ(PININT_NVIC_NAME);

    while (forever)
    {
        RSI_EGPIO_IntUnMask(EGPIO , PIN_INT);
    }
    /*Statement will never reach here , just to satisfy the standard main*/
    return 0;
}
```

Code Block 13 M4SS pin interrupt Example

19.3 API Descriptions

19.3.1 RSI_EGPIO_SetPinMux

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_SetPinMux(EGPIO_Type *pEGPIO ,uint8_t port , uint8_t pin ,
uint8_t mux)
```

Description

This API is used to used to set pin multiplexing(GPIO Pin Mode. Ranges 000 -> Mode 0 to 111 -> Mode 7 Used for GPIO Pin Muxing)

Parameters

Parameter	Description
pEGPIO	Pointer to the EGPIO register instance
port	GPIO port number
pin	GPIO pin number
mux	Pin function value Possible values for this parameter are the following • EGPIO_PIN_MUX_MODE0 Select pin mode0 • EGPIO_PIN_MUX_MODE1 : Select pin mode1 • EGPIO_PIN_MUX_MODE2 : Select pin mode 2 • EGPIO_PIN_MUX_MODE3 : Select pin mode 3 • EGPIO_PIN_MUX_MODE4 : Select pin mode 4

Parameter	Description
	• EGPIO_PIN_MUX_MODE5 : Select pin mode 5 • EGPIO_PIN_MUX_MODE6 : Select pin mode 6 • EGPIO_PIN_MUX_MODE7 : Select pin mode 7 • EGPIO_PIN_MUX_MODE8 : Select pin mode 8 • EGPIO_PIN_MUX_MODE9 : Select pin mode 9 • EGPIO_PIN_MUX_MODE10 : Select pin mode • EGPIO_PIN_MUX_MODE11 : Select pin mode 11 • EGPIO_PIN_MUX_MODE12 : Select pin mode 12 • EGPIO_PIN_MUX_MODE13 : Select pin mode 13 • EGPIO_PIN_MUX_MODE14 : Select pin mode 14 • EGPIO_PIN_MUX_MODE15 : Select pin mode 15

Return values

None

Example

```
#define     M4_GPIO_PORT                0
#define     M4_GPIO_PIN                 6
/*configure GPIO mode(0) */
RSI_EGPIO_SetPinMux(EGPIO,M4_GPIO_PORT ,M4_GPIO_PIN,EGPIO_PIN_MUX_MODE0);
```

19.3.2 RSI_EGPIO_SetDir

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_SetDir(EGPIO_Type *pEGPIO ,uint8_t port,uint8_t pin,
boolean_t dir)
```

Description

This API is used to set the EGPIO direction (Direction of the GPIO pin. '1' for INPUT, '0' for OUTPUT)

Parameters

Parameter	Description
pEGPIO	Pointer to the EGPIO register instance
port	GPIO port number
pin	GPIO pin number
dir	boolean type pin direction 0: Output 1: Input

Return values

None

Example

```
#define     M4_GPIO_PORT                0
#define     M4_GPIO_PIN                 6
#define     EGPIO_CONFIG_DIR_OUTPUT     0
/* set pin direction */
RSI_EGPIO_SetDir(EGPIO,M4_GPIO_PORT ,M4_GPIO_PIN, EGPIO_CONFIG_DIR_OUTPUT);
```

19.3.3 RSI_EGPIO_SetPin

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_SetPin(EGPIO_Type *pEGPIO ,uint8_t port,uint8_t pin , uint8_t
val)
```

Description

This API is used to set the GPIO pin value.It Loads 0th bit on to the pin on write &reads the value on pin on read into 0th bit

Parameters

Parameter	Description
pEGPIO	Pointer to the EGPIO register instance
port	GPIO port number
pin	GPIO pin number
val	value to be set for the pin '0' : Logic on Pin '1' : Logic on Pin

Return values

None

Example

```
#define     M4_GPIO_PORT                0
#define     M4_GPIO_PIN                 6
/* set pin as 1 */
RSI_EGPIO_SetPin(EGPIO, M4_GPIO_PORT, M4_GPIO_PIN, 1);

/* set pin as 0 */
RSI_EGPIO_SetPin(EGPIO, M4_GPIO_PORT, M4_GPIO_PIN, 0);
```

19.3.4 RSI_EGPIO_GetPin

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE boolean_t RSI_EGPIO_GetPin(EGPIO_Type *pEGPIO ,uint8_t port,uint8_t pin)
```

Description

This API is used get the GPIO pin status.

Parameters

Parameter	Description
pEGPIO	Pointer to the EGPIO register instance
port	GPIO port number
pin	GPIO pin number

Return values

Pin status

Example

```
#define     M4_GPIO_PORT                0
#define     M4_GPIO_PIN                 6
boolean_t   status;
/* get pin status */
status=RSI_EGPIO_GetPin(EGPIO,M4_GPIO_PORT, M4_GPIO_PIN);
```

19.3.5 RSI_EGPIO_GetDir

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE boolean_t RSI_EGPIO_GetDir(EGPIO_Type *pEGPIO,uint8_t port ,uint8_t pin)
```

Description

This API is used to Get the Direction GPIO(Direction of the GPIO pin. '1' for INPUT,and '0'for OUTPUT)

Parameters

Parameter	Description
pEGPIO	Pointer to the EGPIO register instance
port	GPIO port number
pin	GPIO pin number

Return values

GPIO direction value

Example

```
#define     M4_GPIO_PORT                0
#define     M4_GPIO_PIN                 6
boolean_t  direction;
/* get gpio direction value */
 direction=RSI_EGPIO_GetDir(EGPIO,M4_GPIO_PORT, M4_GPIO_PIN);
```

19.3.6 RSI_EGPIO_PinIntSel

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_PinIntSel(EGPIO_Type *pEGPIO ,uint8_t intCh ,uint8_t port ,
uint8_t pin)
```

Description

This API is used to select the pin for interrupt generation

Parameters

Parameter	Description
pEGPIO	Pointer to the EGPIO register instance
intCh	GPIO pin interrupt channel number (0 to 7)
port	GPIO port number
pin	GPIO pin number

Return values

None

Example

```
#define M4_GPIO_PORT                0
#define M4_GPIO_PIN                 6
#define PIN_INT                     7
/*selects the pin interrupt for the GPIO*/
RSI_EGPIO_PinIntSel(EGPIO, PIN_INT , M4_GPIO_PORT, M4_GPIO_PIN);
```

19.3.7 RSI_EGPIO_SetIntFallEdgeEnable

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_SetIntFallEdgeEnable(EGPIO_Type *pEGPIO ,uint8_t intCh)
```

Description

This API is used to set the pin interrupt mode configuration enables interrupt generation when falling edge is detected on pin '1' for interrupt enabled and '0' for disabled

Parameters

Parameter	Description
pEGPIO	Pointer to the EGPIO register instance
intCh	GPIO pin interrupt channel number (0 to 7)

Return values

None

Example

```
#define PIN_INT                     7
/* set fall edge interrupt */
RSI_EGPIO_SetIntFallEdgeEnable(EGPIO, PIN_INT);
```

19.3.8 RSI_EGPIO_SetIntFallEdgeDisable

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_SetIntFallEdgeDisable(EGPIO_Type *pEGPIO ,uint8_t intCh)
```

Description

This API is used to set the pin interrupt mode configuration disable interrupt generation when falling edge is detected on pin '1' for interrupt enabled and '0' for disabled

Parameters

Parameter	Description
pEGPIO	Pointer to the EGPIO register instance
intCh	GPIO pin interrupt channel number (0 to 7)

Return values

None

Example

```
#define PIN_INT                    7
/* disable fall edge interrupt */
RSI_EGPIO_SetIntFallEdgeEnable(EGPIO, PIN_INT);
```

19.3.9 RSI_EGPIO_SetIntRiseEdgeEnable

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_SetIntRiseEdgeEnable(EGPIO_Type *pEGPIO ,uint8_t intCh)
```

Description

This API to used to set the pin interrupt mode configuration enables interrupt generation when rising edge is detected on pin '1' for interrupt enabled and '0' for disabled

Parameters

Parameter	Description
pEGPIO	Pointer to the EGPIO register instance
intCh	GPIO pin interrupt channel number (0 to 7)

Return values

None

Example

```
#define PIN_INT                    7
/* set rise edge interrupt */
```

```
RSI_EGPIO_SetIntRiseEdgeEnable(EGPIO, PIN_INT);
```

19.3.10 RSI_EGPIO_SetIntRiseEdgeDisable

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_SetIntRiseEdgeDisable(EGPIO_Type *pEGPIO ,uint8_t intCh)
```

Description

This API to used to set the pin interrupt mode configuration disable rise edge interrupt.

Parameters

Parameter	Description
pEGPIO	Pointer to the EGPIO register instance
intCh	GPIO pin interrupt channel number (0 to 7)

Return values

None

Example

```
#define PIN_INT                    7
/* disable rise edge interrupt */
RSI_EGPIO_SetIntRiseEdgeDisable(EGPIO, PIN_INT);
```

19.3.11 RSI_EGPIO_SetIntLowLevelEnable

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_SetIntLowLevelEnable(EGPIO_Type *pEGPIO ,uint8_t intCh)
```

Description

This API to used to set low level interrupt enable mode for pin.

Parameters

Parameter	Description
pEGPIO	Pointer to the EGPIO register instance
intCh	GPIO pin interrupt channel number (0 to 7)

Return values

None

Example

```
#define PIN_INT                    7
/* set low level interrupt */
RSI_EGPIO_SetIntLowLevelEnable(EGPIO, PIN_INT);
```

19.3.12 RSI_EGPIO_IntMask

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_IntMask(EGPIO_Type *pEGPIO ,uint8_t intCh)
```

Description

This API is used to mask the pin interrupt.

Parameters

Parameter	Description
pEGPIO	Pointer to the EGPIO register instance
intCh	GPIO pin interrupt channel number (0 to 7)

Return values

None

Example

```
#define PIN_INT              7
/* disable interrupt */
RSI_EGPIO_IntMask(EGPIO, PIN_INT);
```

19.3.13 RSI_EGPIO_IntUnMask

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_IntUnMask(EGPIO_Type *pEGPIO ,uint8_t intCh)
```

Description

This API is used to used to set the pin interrupt mode configuration(Masks the interrupt. Interrupt will still be seen in status register when enabled '1' for intr masked, '0' for intr unmasked).

Parameters

Parameter	Description
pEGPIO	Pointer to the EGPIO register instance
intCh	GPIO pin interrupt channel number (0 to 7)

Return values

None

Example

```
#define PIN_INT              7
/* enable interrupt */
RSI_EGPIO_IntUnMask(EGPIO, PIN_INT);
```

19.3.14 RSI_EGPIO_SetIntLowLevelDisable

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_SetIntLowLevelDisable(EGPIO_Type *pEGPIO ,uint8_t intCh)
```

Description

This API is used to disable the low pin interrupt mode configuration(enables interrupt generation when pin level is 0 .'1' for intr enabled, '0' for disabled).

Parameters

Parameter	Description
pEGPIO	Pointer to the EGPIO register instance
intCh	GPIO pin interrupt channel number (0 to 7)

Return values

None

Example

```
#define PIN_INT              7
/* disable low level interrupt */
RSI_EGPIO_SetIntLowLevelDisable(EGPIO, PIN_INT);
```

19.3.15 RSI_EGPIO_SetIntHighLevelEnable

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_SetIntHighLevelEnable(EGPIO_Type *pEGPIO ,uint8_t intCh)
```

Description

This API used to set the pin interrupt mode configuration(enables interrupt generation when pin level is 1, '1' for intr enabled '0' for disabled)

Parameters

Parameter	Description
pEGPIO	Pointer to the EGPIO register instance
intCh	GPIO pin interrupt channel number (0 to 7)

Return values

None

Example

```
#define PIN_INT              7
/* enable high level interrupt */
RSI_EGPIO_SetIntHighLevelEnable(EGPIO, PIN_INT);
```

19.3.16 RSI_EGPIO_SetIntHighLevelDisable

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_SetIntHighLevelDisable(EGPIO_Type *pEGPIO ,uint8_t intCh)
```

Description

This API is used to used to set the pin interrupt mode configuration(enables interrupt generation when pin level is 1 ,'1' for intr enabled '0' for disabled).

Parameters

Parameter	Description
pEGPIO	Pointer to the EGPIO register instance
intCh	GPIO pin interrupt channel number (0 to 7)

Return values

None

Example

```
#define PIN_INT              7
/* disable high level interrupt */
RSI_EGPIO_SetIntHighLevelDisable(EGPIO, PIN_INT);
```

19.3.17 RSI_EGPIO_GetIntStat

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE uint8_t RSI_EGPIO_GetIntStat(EGPIO_Type *pEGPIO ,uint8_t intCh)
```

Description

This API is used to used to set the pin interrupt mode configuration(enables interrupt generation when pin level is 1 ,'1' for intr enabled '0' for disabled).

Parameters

Parameter	Description
pEGPIO	Pointer to the EGPIO register instance

Parameter	Description
intCh	GPIO pin interrupt channel number (0 to 7)

Return values

returns the interrupt status register

Example

```
#define PIN_INT              7
uint32_t gintStatus;
/*get interrupt status*/
gintStatus=RSI_EGPIO_GetIntStat(EGPIO,PIN_INT);
```

19.3.18 RSI_EGPIO_IntClr

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_IntClr(EGPIO_Type *pEGPIO ,uint8_t intCh , uint8_t flags)
```

Description

This API is used to used to clear the pin interrupt in status register

Parameters

Parameter	Description
pEGPIO	Pointer to the EGPIO register instance
intCh	GPIO pin interrupt channel number (0 to 7)
flags	GPIO pin interrupt channel number (0 to 7) refer: EGPIO_PIN_INT_CLR_RISING EGPIO_PIN_INT_CLR_FALLING INTERRUPT_STATUS_CLR

Return values

None

Example

```
#define PIN_INT              7
/*clear interrupt*/
RSI_EGPIO_IntClr(EGPIO, PIN_INT ,EGPIO_PIN_INT_CLR_RISING);
```

19.3.19 RSI_EGPIO_UlpSocGpioMode

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_UlpSocGpioMode(ULPClk_Type *pULPCLK,uint8_t gpio,uint8_t
mode)
```

Description

This API is used set ulp soc gpio mode (Gpio pin mode,ranges 000 -> Mode 0 to 111 -> Mode 7 Used for GPIO Pin Muxing).

Parameters

Parameter	Description
pULPCLK	Pointer to the ULP register instance
gpio	Gpio number
mode	GPIO mode possible values for this parameter are the following • EGPIO_PIN_MUX_MODE0 : Select pin mode 0 • EGPIO_PIN_MUX_MODE1 : Select pin mode 1 • EGPIO_PIN_MUX_MODE2 : Select pin mode 2 • EGPIO_PIN_MUX_MODE3 : Select pin mode 3 • EGPIO_PIN_MUX_MODE4 : Select pin mode 4 • EGPIO_PIN_MUX_MODE5 : Select pin mode 5 • EGPIO_PIN_MUX_MODE6 : Select pin mode 6 • EGPIO_PIN_MUX_MODE7 : Select pin mode 7

Return values

None

Example

```
#define     M4_GPIO_PIN                 6
/* gpio in ulp soc mode */
 RSI_EGPIO_UlpSocGpioMode(ULPCLK,M4_GPIO_PIN,EGPIO_PIN_MUX_MODE0);
```

19.3.20 RSI_EGPIO_SetPortMask

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_SetPortMask(EGPIO_Type *pEGPIO ,uint8_t port,uint8_t pin)
```

Description

This API is used to set the EGPIO port mask. When set, pin is masked when written/read through PORT MASK REG.

Parameters

Parameter	Description
pEGPIO	Pointer to the EGPIO register instance
port	GPIO port number
pin	GPIO pin number

Return values

None

Example

```
#define     M4_GPIO_PORT                0
#define     M4_GPIO_PIN                 6
/* set port mask */
 RSI_EGPIO_SetPortMask(EGPIO,M4_GPIO_PORT ,M4_GPIO_PIN);
```

19.3.21 RSI_EGPIO_SetPortUnMask

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void  RSI_EGPIO_SetPortUnMask(EGPIO_Type *pEGPIO ,uint8_t port,uint8_t pin)
```

Description

This API is used to set the EGPIO port unmask. When set, pin is masked when written/read through PORT MASK REG.

Parameters

Parameter	Description
pEGPIO	Pointer to the EGPIO register instance
port	GPIO port number
pin	GPIO pin number

Return values

None

Example

```
#define     M4_GPIO_PORT                0
#define     M4_GPIO_PIN                 6
/* set port mask */
RSI_EGPIO_SetPortUnMask(EGPIO,M4_GPIO_PORT ,M4_GPIO_PIN);
```

19.3.22 RSI_EGPIO_PortMaskedLoad

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_PortMaskedLoad(EGPIO_Type *pEGPIO ,uint8_t port, uint16_t
val)
```

Description

This API is used to set the EGPIO port mask load. When set, pin is masked when written/read through PORT MASK REG.

Parameters

Parameter	Description

Parameter	Description
pEGPIO	Pointer to the EGPIO register instance
port	GPIO port number
val	Port value to be set

Return values

None

Example

```
#define     M4_GPIO_PORT                0
/* loading port masked value */
RSI_EGPIO_PortMaskedLoad(EGPIO,M4_GPIO_PORT,1);
```

19.3.23 RSI_EGPIO_SetPort

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_SetPort(EGPIO_Type *pEGPIO ,uint8_t port , uint16_t val)
```

Description

This API is used to set the port value. Sets the pin when corresponding bit is high. Writing zero has no effect.

Parameters

Parameter	Description
pEGPIO	Pointer to the EGPIO register instance
port	GPIO port number
val	Port value to be set

Return values

None

Example

```
#define     M4_GPIO_PORT                0
/* set port */
RSI_EGPIO_SetPort(EGPIO,M4_GPIO_PORT,2);
```

19.3.24 RSI_EGPIO_PortLoad

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_PortLoad(EGPIO_Type *pEGPIO ,uint8_t port , uint16_t val)
```

Description

This API is used to set the port value.Sets the pin when corresponding bit is high. Writing zero has no effect.

Parameters

Parameter	Description
pEGPIO	Pointer to the EGPIO register instance
port	GPIO port number
val	Port value to be set

Return values

None

Example

```
#define     M4_GPIO_PORT                0
/* load port value */
RSI_EGPIO_PortLoad(EGPIO,M4_GPIO_PORT,2);
```

19.3.25 RSI_EGPIO_WordLoad

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_WordLoad(EGPIO_Type *pEGPIO ,uint8_t pin , uint16_t val)
```

Description

This API is used to set the port value. Sets the pin when corresponding bit is high. Writing zero has no effect.

Paramters

Parameter	Description
pEGPIO	Pointer to the EGPIO register instance
port	GPIO port number
val	Port value to be set

Return values

None

Example

```
#define     M4_GPIO_PORT                0
/* word load */
RSI_EGPIO_WordLoad(EGPIO,M4_GPIO_PORT,1);
```

19.3.26 RSI_EGPIO_ClrPort

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_ClrPort(EGPIO_Type *pEGPIO ,uint8_t port , uint16_t val)
```

Description

This API is used to clear the port value. Clears the pin when corresponding bit is high. Writing zero has no effect.

Parameters

Parameter	Description
pEGPIO	Pointer to the EGPIO register instance
port	GPIO port number
val	Port value to be set

Return values

None

Example

```
#define     M4_GPIO_PORT                0
/* clear port */
RSI_EGPIO_ClrPort(EGPIO,M4_GPIO_PORT,1);
```

19.3.27 RSI_EGPIO_TogglePort

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_TogglePort(EGPIO_Type *pEGPIO ,uint8_t port , uint16_t val)
```

Description

This API is used to toggle the port.Toggles the pin when corresponding bit is high. Writing zero has not effect.

Parameters

Parameter	Description
pEGPIO	Pointer to the EGPIO register instance
port	GPIO port number
val	Port value to be set

Return values

None

Example

```
#define     M4_GPIO_PORT                0
/* toggle port */
RSI_EGPIO_TogglePort(EGPIO,M4_GPIO_PORT,0XFFFF);  /* All 16 GPIO toggle in port */
```

19.3.28 RSI_EGPIO_GetPort

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
uint16_t RSI_EGPIO_GetPort(EGPIO_Type *pEGPIO ,uint8_t port)
```

Description

This API is used to used to get the EGPIO port value. Reads the value on GPIO pins irrespective of the pin mode.

Parameters

Parameter	Description
pEGPIO	Pointer to the EGPIO register instance
port	GPIO port number
val	Port value to be set

Return values

port value

Example

```
uint16_t portval;
/* get port value */
portval=RSI_EGPIO_GetPort(EGPIO,M4_GPIO_PORT
```

19.3.29 RSI_EGPIO_GroupIntOneEnable

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_GroupIntOneEnable(EGPIO_Type *pEGPIO,uint8_t port,uint8_t
pin)
```

Description

This API is used to enable the group interrupt one , When set, the corresponding GPIO pin is selected for group interrupt 1 generation.

Parameters

Parameter	Description
pEGPIO	Pointer to the EGPIO register instance
port	GPIO port number
pin	GPIO pin number

Return values

None

Example

```
#define     M4_GPIO_PORT                0
#define     M4_GPIO_PIN                 6
/*enables group interrupt on pin*/
```

```
RSI_EGPIO_GroupIntOneEnable(EGPIO,M4_GPIO_PORT,M4_GPIO_PIN);
```

19.3.30 RSI_EGPIO_GroupIntOneDisable

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_GroupIntOneDisable(EGPIO_Type *pEGPIO,uint8_t port,uint8_t
pin)
```

Description

This API is used to used to disable the group interrupt one.

Parameters

Parameter	Description
pEGPIO	Pointer to the EGPIO register instance
port	GPIO port number
pin	GPIO pin number

Return values

None

Example

```
#define     M4_GPIO_PORT                0
#define     M4_GPIO_PIN                 6
/*disable group interrupt on pin*/
RSI_EGPIO_GroupIntOneEnable(EGPIO,M4_GPIO_PORT,M4_GPIO_PIN);
```

19.3.31 RSI_EGPIO_GroupIntTwoEnable

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_GroupIntTwoEnable(EGPIO_Type *pEGPIO,uint8_t port,uint8_t
pin)
```

Description

This API is used to enable the group interrupt Two , When set,the corresponding GPIO pin is selected for group interrupt 2 generation.

Parameters

Parameter	Description
pEGPIO	Pointer to the EGPIO register instance
port	GPIO port number

Parameter	Description
pin	GPIO pin number

Return values

None

Example

```
#define     M4_GPIO_PORT                0
#define     M4_GPIO_PIN                 6
/* interrupt 2 enable */
RSI_EGPIO_GroupIntTwoEnable(EGPIO,M4_GPIO_PORT,M4_GPIO_PIN);
```

19.3.32 RSI_EGPIO_GroupIntMask

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_GroupIntMask(EGPIO_Type *pEGPIO ,uint8_t grpInt)
```

Description

This API is used to configure the group interrupts(1-mask,0-unmask).

Parameters

Parameter	Description
pEGPIO	Pointer to the EGPIO register instance
grpInt	Group interrupt number

Return values

None

Example

```
#define GROUPINT                 0
/* mask group interrupt number */
RSI_EGPIO_GroupIntMask(EGPIO,GROUPINT);
```

19.3.33 RSI_EGPIO_GroupIntUnMask

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_GroupIntUnMask(EGPIO_Type *pEGPIO ,uint8_t grpInt)
```

Description

This API is used to configure the group interrupts(1-mask,0-unmask)

Parameters

Parameter	Description
pEGPIO	Pointer to the EGPIO register instance
grpInt	Group interrupt number

Return values

None

Example

```
#define GROUPINT                 0
/* unmask group interrupt number */
RSI_EGPIO_GroupIntUnMask(EGPIO,GROUPINT);
```

19.3.34 RSI_EGPIO_GroupIntEnable

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_GroupIntEnable(EGPIO_Type *pEGPIO ,uint8_t grpInt)
```

Description

This API is used to configure the group interrupts(1-enable, 0-disable)

Parameters

Parameter	Description
pEGPIO	Pointer to the EGPIO register instance
grpInt	Group interrupt number

Return values

None

Example

```
#define GROUPINT                 0
/* enable group interrupt */
RSI_EGPIO_GroupIntEnable(EGPIO,GROUPINT);
```

19.3.35 RSI_EGPIO_GroupIntDisable

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_GroupIntDisable(EGPIO_Type *pEGPIO ,uint8_t grpInt)
```

Description

This API is used to configure the group interrupts(1-enable, 0-disable).

Parameters

Parameter	Description
pEGPIO	Pointer to the EGPIO register instance
grpInt	Group interrupt number

Return values

None

Example

```
#define GROUPINT                 0
/* disable group interrupt */
RSI_EGPIO_GroupIntDisable(EGPIO,GROUPINT);
```

19.3.36 RSI_EGPIO_GroupIntLevel

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_GroupIntLevel(EGPIO_Type *pEGPIO ,uint8_t grpInt)
```

Description

This API is used to configure the group interrupts(0-level,1-edge)

Parameters

Parameter	Description
pEGPIO	Pointer to the EGPIO register instance
grpInt	Group interrupt number

Return values

None

Example

```
#define GROUPINT                 0
/* configure group level interrupt */
RSI_EGPIO_GroupIntLevel(EGPIO,GROUPINT);
```

19.3.37 RSI_EGPIO_GroupIntEdge

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_GroupIntEdge(EGPIO_Type *pEGPIO ,uint8_t grpInt)
```

Description

This API is used to configure the group interrupts(0-level,1-edge).

Parameters

Parameter	Description
pEGPIO	Pointer to the EGPIO register instance
grpInt	Group interrupt number

Return values

None

Example

```
#define GROUPINT                 0
/* configure group edge interrupt */
RSI_EGPIO_GroupIntLevel(EGPIO,GROUPINT);
```

19.3.38 RSI_EGPIO_GroupIntAnd

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_GroupIntAnd(EGPIO_Type *pEGPIO ,uint8_t grpInt)
```

Description

This API is used to configure the group interrupts(0-AND ,1-Or).

Parameters

Parameter	Description
pEGPIO	Pointer to the EGPIO register instance
grpInt	Group interrupt number

Return values

None

Example

```
#define GROUPINT                 0
/* configure group AND interrupt */
RSI_EGPIO_GroupIntLevel(EGPIO,GROUPINT);
```

19.3.39 RSI_EGPIO_GroupIntOr

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_GroupIntOr(EGPIO_Type *pEGPIO ,uint8_t grpInt)
```

Description

This API is used to configure the group interrupts(0- AND , 1-Or).

Parameters

Parameter	Description
pEGPIO	Pointer to the EGPIO register instance
grpInt	Group interrupt number

Return values

None

Example

```
#define GROUPINT                 0
/* configure group AND interrupt */
RSI_EGPIO_GroupIntOr(EGPIO,GROUPINT);
```

19.3.40 RSI_EGPIO_GroupIntStat

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
uint32_t RSI_EGPIO_GroupIntStat(EGPIO_Type *pEGPIO ,uint8_t grpInt)
```

Description

This API to used to get the group interrupt status.

Parameters

Parameter	Description
pEGPIO	Pointer to the EGPIO register instance
grpInt	Group interrupt number

Return values

returns the group interrupt status register.

Example

```
#define GROUPINT                 0
uint32_t gintStatus;
/* get interrupt status */
gintStatus=RSI_EGPIO_GroupIntStat(EGPIO,GROUPINT);
```

19.3.41 RSI_EGPIO_GroupIntWkeUpEnable

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_GroupIntWkeUpEnable(EGPIO_Type *pEGPIO ,uint8_t grpInt)
```

Description

This API to used to Enable the group interrupt wakeup interrupt.

Parameters

Parameter	Description
pEGPIO	Pointer to the EGPIO register instance
grpInt	Group interrupt number

Return values

None

Example

```
#define GROUPINT                 0
/* group interrupt wakeup enable */
RSI_EGPIO_GroupIntStat(EGPIO,GROUPINT);
```

19.3.42 RSI_EGPIO_GroupIntWkeUpDisable

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_GroupIntWkeUpDisable(EGPIO_Type *pEGPIO ,uint8_t grpInt)
```

Description

This API to used to Disable the group interrupt wakeup interrupt.

Parameters

Parameter	Description
pEGPIO	Pointer to the EGPIO register instance
grpInt	Group interrupt number

Return values

None

Example

```
#define GROUPINT                 0
/* group interrupt wakeup disable */
RSI_EGPIO_GroupIntWkeUpEnable(EGPIO,GROUPINT);
```

19.3.43 RSI_EGPIO_GroupIntClr

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_GroupIntClr(EGPIO_Type *pEGPIO ,uint8_t grpInt , uint8_t
flags)
```

Description

This API is used to used to clear the group interrupt status.

Parameters

Parameter	Description
pEGPIO	Pointer to the EGPIO register instance
grpInt	Group interrupt number
flags	clear flags, possible flag value is below: • EGPIO_PIN_INT_CLR_FALLING • EGPIO_PIN_INT_CLR_RISING • INTERRUPT_STATUS_CLR • WAKEUP_INTERRUPT

Return values

None

Example

```
#define GROUPINT                 0
/*clears group interrupt*/
RSI_EGPIO_GroupIntClr(ULPSSEGPIO, GROUPINT ,INTERRUPT_STATUS_CLR);            /* ULP
subsystem pin */
```

19.3.44 RSI_EGPIO_GroupIntTwoDisable

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_GroupIntTwoDisable(EGPIO_Type *pEGPIO ,uint8_t port ,uint8_t
pin)
```

Description

This API is used to used to clear the group interrupt status.

Parameters

Parameter	Description
pEGPIO	Pointer to the EGPIO register instance
port	PORT number
pin	PIN number

Return values

None

Example

```
#define     M4_GPIO_PORT                 0
#define     M4_GPIO_PIN                  6
/* disable group interrupt2 */
```

```
RSI_EGPIO_GroupIntTwoDisable(EGPIO,M4_GPIO_PORT,M4_GPIO_PIN);
```

19.3.45 RSI_EGPIO_SetGroupIntOnePol

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_SetGroupIntOnePol(EGPIO_Type *pEGPIO ,uint8_t port , uint8_t
pin , uint8_t pol)
```

Description

This API is used to set the group polarity of interrupt one.Decides the active value of the pin to be considered for group interrupt 1 generation when enabled.

Parameters

Parameter	Description
pEGPIO	Pointer to the EGPIO register instance
port	PORT number
pin	PIN number
pol	Polarity of interrupt '0' : group interrupt gets generated when GPIO input pin status is '0'. '1' : group interrupt gets generated when GPIO input pin status is '1'

Return values

None

Example

```
#define     M4_GPIO_PORT                0
#define     M4_GPIO_PIN                 6
#define     POL_HIGH                    1
/*configures group interrupt polarity high */
RSI_EGPIO_SetGroupIntOnePol(EGPIO,M4_GPIO_PORT,M4_GPIO_PIN,POL_HIGH);
```

19.3.46 RSI_EGPIO_SetGroupIntTwoPol

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_SetGroupIntTwoPol(EGPIO_Type *pEGPIO ,uint8_t port,uint8_t
pin , uint8_t pol)
```

Description

This API is used to set the group polarity of interrupt two. Decides the active value of the pin to be considered for group interrupt 2 generation when enabled.

Parameters

Parameter	Description
pEGPIO	Pointer to the EGPIO register instance
port	PORT number
pin	PIN number
pol	Polarity of interrupt '0' : group interrupt gets generated when GPIO input pin status is '0'. '1' : group interrupt gets generated when GPIO input pin status is '1'

Return values

None

Example

```
#define     M4_GPIO_PORT                0
#define     M4_GPIO_PIN                 6
#define     POL_HIGH                    1
/* configure group interrupt2 */
RSI_EGPIO_SetGroupIntTwoPol(EGPIO,M4_GPIO_PORT,M4_GPIO_PIN,POL_HIGH);
```

19.3.47 RSI_EGPIO_HostPadsGpioModeEnable

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_HostPadsGpioModeEnable(uint8_t u8GpioNum)
```

Description

This API is used to select the host pad gpios(26 to 30).

Parameter

Parameter	Description
u8GpioNum	Gpio number to be use.

Return values

None

Example

```
#define     M4_GPIO_PIN                 6
/* enable host pads gpio mode */
RSI_EGPIO_HostPadsGpioModeEnable(M4_GPIO_PIN);
```

19.3.48 RSI_EGPIO_HostPadsGpioModeDisable

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_HostPadsGpioModeDisable(uint8_t u8GpioNum)
```

Description

This API is used to deselect the host pad gpios(26 to 30)

Parameter

Parameter	Description
u8GpioNum	Gpio number to be use.

Return values

None

Example

```
#define     M4_GPIO_PIN                 6
/* disable host pads gpio mode */
RSI_EGPIO_HostPadsGpioModeDisable(M4_GPIO_PIN);
```

19.3.49 RSI_EGPIO_PadSelectionEnable

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_PadSelectionEnable(uint8_t padNum)
```

Description

This API is used to select the pad(0 to 21).

Parameter

Parameter	Description
padNum	PAD number to be use

Return values

None

Example

```
uint8_t pad=1;
/* enable pad */
RSI_EGPIO_PadSelectionEnable(pad);
```

19.3.50 RSI_EGPIO_PadSelectionDisable

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_PadSelectionDisable(uint8_t padNum)
```

Description

This API is used to deselect the pad(0 to 21).

Parameter

Parameter	Description
padNum	PAD number to be use

Return values

None

Example

```
uint8_t pad=1;
/* disable pad selection */
RSI_EGPIO_PadSelectionDisable(pad);
```

19.3.51 RSI_EGPIO_PadReceiverEnable

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_PadReceiverEnable(uint8_t u8GpioNum)
```

Description

This API is used to enable the receiver enable bit(REN).

Parameter

Parameter	Description
u8GpioNum	GPIO num to be use

Return values

None

Example

```
#define     M4_GPIO_PIN               6
/* pad receive enable */
RSI_EGPIO_PadReceiverEnable(M4_GPIO_PIN);
```

19.3.52 RSI_EGPIO_PadReceiverDisable

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_PadReceiverDisable(uint8_t u8GpioNum)
```

Description

This API is used to Disable the receiver enable bit(REN).

Parameter

Parameter	Description
u8GpioNum	GPIO num to be use

Return values

None

Example

```
#define     M4_GPIO_PIN              8
/* disble pad receiver */
RSI_EGPIO_PadReceiverDisable(M4_GPIO_PIN);
```

19.3.53 RSI_EGPIO_PadSdioConnected

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_PadSdioConnected(STATIC INLINE void)
```

Description

This API is used to use the SDIO pins(25 to 30) in M4 or TA (0 for M4SS and 1 for TASS)

Parameter

None

Return values

None

Example

```
/* sdio pin connect */
RSI_EGPIO_PadSdioConnected(STATIC INLINE void);
```

19.3.54 RSI_EGPIO_PadDriverDisableState

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_PadDriverDisableState(uint8_t u8GpioNum , en_driver_state_t
endstate)
```

Description

This API is used to control the Driver disabled state control.

Parameter	Description
u8GpioNum	GPIO num to be use
endstate	the value to be passed

Parameter	Description
	possible enum values are • HiZ • Pullup • Pulldown • Repeater

Return values

None

Example

```
#define     M4_GPIO_PIN               8
RSI_EGPIO_PadDriverDisableState(M4_GPIO_PIN,HiZ);
```

19.3.55 RSI_EGPIO_PadDriverStrengthSelect

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_PadDriverStrengthSelect(uint8_t u8GpioNum ,
en_driver_strength_select_t strength)
```

Description

This API is used to select Drive strength.

Parameters

Parameter	Description
u8GpioNum	GPIO num to be use
strength	Drive strength selector(E1,E2) possible enum values are • two_milli_amps • four_milli_amps • eight_milli_amps • twelve_milli_amps

Return values

None

Example

```
#define     M4_GPIO_PIN            8
RSI_EGPIO_PadDriverStrengthSelect(M4_GPIO_PIN,four_milli_amps);
```

19.3.56 RSI_EGPIO_PadPowerOnStartEnable

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_PadPowerOnStartEnable(uint8_t u8GpioNum ,uint8_t val)
```

Description

This API is used to select Power on Start enable.

Parameters

Parameter	Description
u8GpioNum	GPIO num to be use
val	• POS = 1 : Enables active pull down for invalid power; • POS = 0 : Active pull down capability disabled . When one of the power supplies is invalid and active high POS is set to 1, PAD is pulled to weak 0. When POS is set to 0, PAD remains in a high-Z state. : Default 0

Return values

None

Example

```
#define     M4_GPIO_PIN            8
RSI_EGPIO_PadPowerOnStartEnable(M4_GPIO_PIN,1);
```

19.3.57 RSI_EGPIO_PadActiveHighSchmittTrigger

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_PadActiveHighSchmittTrigger(uint8_t u8GpioNum ,uint8_t val)
```

Description

Active high Schmitt trigger (Hysteresis) select.

Parameters

Parameter	Description
u8GpioNum	GPIO num to be use
val	SMT=0 : No hysteresis; Default value for reset is 1'b1 and others is 1'b0

Return values

None

Example

```
#define     M4_GPIO_PIN            8
RSI_EGPIO_PadActiveHighSchmittTrigger(M4_GPIO_PIN,0);
```

19.3.58 RSI_EGPIO_PadSlewRateControll

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_PadSlewRateControll(uint8_t u8GpioNum ,uint8_t val)
```

Description

This API is used to control the slew rate.

Parameters

Parameter	Description
u8GpioNum	GPIO num to be use
val	slew rate • SR = 0 : Slow (half frequency) • SR = 1 : Fast ,Default 1

Return values

None

Example

```
#define     M4_GPIO_PIN            8
RSI_EGPIO_PadActiveHighSchmittTrigger(M4_GPIO_PIN,0);
```

19.3.59 RSI_EGPIO_UlpPadReceiverEnable

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_UlpPadReceiverEnable(uint8_t u8GpioNum)
```

Description

This API is used to enable the REN for ULP.

Parameter

Parameter	Description
u8GpioNum	GPIO num to be used

Return values

None

Example

```
#define     ULP_GPIO_PIN            8
/* ulp pad receive enable */
RSI_EGPIO_UlpPadReceiverEnable(ULP_GPIO_PIN);
```

19.3.60 RSI_EGPIO_UlpPadReceiverDisable

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_UlpPadReceiverDisable(uint8_t u8GpioNum)
```

Description

This API is used to disable the REN for ULP.

Parameter

Parameter	Description
u8GpioNum	GPIO num to be used

Return values

None

Example

```
#define     ULP_GPIO_PIN            8
/* ulp pad receiver disable */
RSI_EGPIO_UlpPadReceiverDisable(ULP_GPIO_PIN);
```

19.3.61 RSI_EGPIO_UlpPadDriverDisableState

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_UlpPadDriverDisableState(uint8_t u8GpioNum ,
en_ulp_driver_disable_state_t  disablestate)
```

Description

This API is used to control the Driver disabled state control.

Parameters

Parameter	Description
u8GpioNum	GPIO num to be used
disablestate	The value to be passed possible enum values are • ulp_HiZ • ulp_Pullup • ulp_Pulldown • ulp_Repeater

Return values

None

Example

```
#define     ULP_GPIO_PIN                8
/* disable driver control state */
RSI_EGPIO_UlpPadDriverDisableState(ULP_GPIO_PIN,ulp_Pulldown);
```

19.3.62 RSI_EGPIO_UlpPadDriverStrengthSelect

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_UlpPadDriverStrengthSelect(uint8_t u8GpioNum ,
en_ulp_driver_strength_select_t strength)
```

Description

this API is used to select Drive strength.

Parameters

Parameter	Description
u8GpioNum	GPIO num to be used
strength	Drive strength selector(E1,E2) possible enum values are • ulp_two_milli_amps • ulp_four_milli_amps • ulp_eight_milli_amps • ulp_twelve_milli_amps

Return values

None

Example

```
#define     ULP_GPIO_PIN                8
/* driver strenght select */
RSI_EGPIO_UlpPadDriverStrengthSelect(ULP_GPIO_PIN,ulp_two_milli_amps );
```

19.3.63 RSI_EGPIO_UlpPadPowerOnStartEnable

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_UlpPadPowerOnStartEnable(uint8_t u8GpioNum ,uint8_t val)
```

Description

ULP Power-on-Start enable.

Parameters

Parameter	Description
u8GpioNum	GPIO num to be used
val	possible values are POS = 1 : Enables active pull down for invalid power POS = 0 : Active pull down capability disabled . When one of the power supplies is invalid and active high POS is set to 1,PAD is pulled to weak 0. When POS is set to 0, PAD remains in a high Z state. : Default 0

Return values

None

Example

```
#define     ULP_GPIO_PIN                8
RSI_EGPIO_UlpPadPowerOnStartEnable(ULP_GPIO_PIN,1);
```

19.3.64 RSI_EGPIO_UlpPadActiveHighSchmittTrigger

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_UlpPadActiveHighSchmittTrigger(uint8_t u8GpioNum ,uint8_t
val)
```

Description

Active high Schmitt trigger (Hysteresis) select.

Parameters

Parameter	Description
u8GpioNum	GPIO num to be used
val	possible values are SMT=0 : No hysteresis; Default value for reset is 1'b1 and others is 1'b0

Return values

None

Example

```
#define     ULP_GPIO_PIN                8
RSI_EGPIO_UlpPadActiveHighSchmittTrigger(ULP_GPIO_PIN,0);
```

19.3.65 RSI_EGPIO_UlpPadSlewRateControl

Source File : rsi_rom_egpio.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_EGPIO_UlpPadSlewRateControll(uint8_t u8GpioNum ,uint8_t val)
```

Description

Slew Rate Control.

Parameters

Parameter	Description
u8GpioNum	GPIO num to be used
val	slew rate SR = 0 : Slow (half frequency); SR = 1 for Fast , Default 1

Return values

None

Example

```
#define     ULP_GPIO_PIN               8
/* controls slew rate */
 RSI_EGPIO_UlpPadSlewRateControll(ULP_GPIO_PIN,1);
```

20 Filter Interpolation Matrix Multiplication(FIM)

20.1 Overview

This section explains how to configure and use the Filter Interpolation Matrix Multiplication using Redpine MCU SAPIs.

20.2 Programming Sequence

```
#include "rsi_chip.h  /* Redpine_MCU_Vx.y.z\Host_MCU\common\chip\inc */

STATIC INLINE void FIM_IRQHandler()
{
    fim_interrupt_handler(M4_FIM);
    stat = 1;
}

/* For starting this example enable ADD_SUB_MUL this MACRO */
#ifdef  ADD_SUB_MUL
int main()
{
    uint32_t matrix_len;
    uint32_t mode  ;

    RSI_PS_UlpssPeriPowerDown(ULPSS_PWRGATE_ULP_FIM);
    RSI_PS_UlpssPeriPowerUp(ULPSS_PWRGATE_ULP_FIM);

    /*Fim clock enable */
    ROM_ULPSS_PeripheralEnable(ULPCLK,ULP_FIM_CLK,ENABLE_STATIC_CLK);


    mode = ADD_SCALAR;  /* assign required value to the mode. */

    /* To use a particular feature pass corresponding enums in swith case */
switch(mode)
{
 case ADD_SCALAR:
    NVIC_EnableIRQ(FIM_IRQn);
    /*API to add real vector to a real scalar of format F32 */
    arm_offset_f32_opt(srcAF32,sCALARF32,pDstF32,blockSize);
    while(stat != 1);
    /*Read the destination */
    stat = 0;
    fim_read_data(BANK1,blockSize,pDstF32,FORMAT_F32,ULP_FIM_COP_DATA_REAL_REAL);
    /* API to add real vector to a real scalar of
    8-bit fractional data type in 1.7 format   */
    arm_offset_q7_opt(srcA1,sCALAR1,pDst1,blockSize);
    while(stat != 1);
    /*Read the destination */
    stat = 0;
    fim_read_data(BANK1,blockSize,pDst1,FORMAT_Q7,ULP_FIM_COP_DATA_REAL_REAL);  // may not
be required.
    /* API to add real vector to a real scalar of   */
    /* 16-bit fractional data type in 1.15 format  */
    arm_offset_q15_opt(srcA2,sCALAR2,pDst2,blockSize);
    while(stat != 1);
    /*Read the destination */
    stat = 0;
    fim_read_data(BANK1,blockSize,pDst2,FORMAT_Q15,ULP_FIM_COP_DATA_REAL_REAL);
    /* API to add real vector to a real scalar of   */
    /* 32-bit fractional data type in 1.31 format  */
    arm_offset_q31_opt(srcA3,sCALAR3,pDst3,blockSize);
    while(stat != 1);
    /*Read the destination */
    stat = 0;
    fim_read_data(BANK1,blockSize,pDst3,FORMAT_Q31,ULP_FIM_COP_DATA_REAL_REAL);
    /*API to add complex data */
    /* Format q15 */
```

```
rsi_fim_scalar_add_q15(incmplx,scalar_cmplx,blockSize*2,ULP_FIM_COP_DATA_CPLX_REAL,0x00,BA
NK1);
    while(stat != 1);
    /*Read the destination */
    stat = 0;
    fim_read_data(BANK1,blockSize*2,pDst2,FORMAT_Q15,ULP_FIM_COP_DATA_CPLX_REAL);
    break;

 case SUB_SCALAR:
    NVIC_EnableIRQ(FIM_IRQn);
    /*API to subtract a scalar */
    /* Format q7 only for real values */
    fim_scalar_sub_q7(srcA1,sCALAR1,pDst1,blockSize);
    while(stat != 1);
    /*Read the destination */
    stat = 0;
    fim_read_data(BANK1,blockSize,pDst1,FORMAT_Q7,ULP_FIM_COP_DATA_REAL_REAL);
    /* Format q15*/

rsi_fim_scalar_sub_q15(incmplx,scalar_cmplx/*0x3920*/,NULL,blockSize*2,ULP_FIM_COP_DATA_CP
LX_CPLX,0x00,BANK1);
    while(stat != 1);
    /*Read the destination */
    stat = 0;
    fim_read_data(BANK1,blockSize*2,pDst4,FORMAT_Q15,ULP_FIM_COP_DATA_REAL_CPLX);
    /* Format q31*/
    fim_scalar_sub_q31(srcA3,sCALAR3,pDst3/*0x39ff2000*/,blockSize);
    while(stat != 1);
    /*Read the destination */
    stat = 0;
    fim_read_data(BANK1,blockSize,pDst3,FORMAT_Q31,ULP_FIM_COP_DATA_REAL_REAL);
    /* Format f32 */
    fim_scalar_sub_f32(srcAF32,sCALARF32/*0x039F0200*/,pDstF32,blockSize);
    while(stat != 1);
    /*Read the destination */
    stat = 0;
    fim_read_data(BANK1,blockSize,pDstF32,FORMAT_F32,ULP_FIM_COP_DATA_REAL_REAL);
    break;

 case MUL_SCALAR:
    NVIC_EnableIRQ(FIM_IRQn);

    /*API to multiply real vector with a real scalar */
    arm_scale_f32_opt(srcAF32,sCALARF32,pDstF32,blockSize);
    while(stat != 1);
    stat = 0;
    /*Read the destination */
    fim_read_data(BANK1,blockSize,pDstF32,FORMAT_F32,ULP_FIM_COP_DATA_REAL_REAL);
    /*API to multiply real vector with a real scalar */
    arm_scale_q7_opt(srcA1,scaleFractq7,0,pDst1,blockSize);
    while(stat != 1);
    stat = 0;
    /*Read the destination */
    fim_read_data(BANK1,blockSize,pDst1,FORMAT_Q7,ULP_FIM_COP_DATA_REAL_REAL);
    /*API to multiply real vector with a real scalar */
    arm_scale_q15_opt(srcA2,scaleFractq15,0,pDst2,blockSize);
    while(stat != 1);
    stat = 0;
    /*Read the destination */
    fim_read_data(BANK1,blockSize,pDst2,FORMAT_Q15,ULP_FIM_COP_DATA_REAL_REAL);
    /*API to multiply real vector with a real scalar */
    arm_scale_q31_opt(srcA3,scaleFractq31,0 /*shift*/,pDst3,blockSize);
    while(stat != 1);
    stat = 0;
    /*Read the destination */
    fim_read_data(BANK1,blockSize,pDst3,FORMAT_Q31,ULP_FIM_COP_DATA_REAL_REAL);
    /*API to multiply complex data */
    fim_scalar_mul_q15(incmplx,scalar_cmplx,blockSize*2,ULP_FIM_COP_DATA_CPLX_CPLX);
    while(stat != 1);
    stat = 0;
```

```
    fim_read_data(BANK1,blockSize*2,pDst2,FORMAT_Q15,ULP_FIM_COP_DATA_CPLX_CPLX);
    break;

 case ADD_VECTOR:
    NVIC_EnableIRQ(FIM_IRQn);
    /*API to add real vectors */
    arm_add_f32_opt( srcAF32,srcBF32,blockSize);
    while(stat != 1);
    stat = 0;
    /*Read the destination */
    fim_read_data(BANK2,blockSize,pDstF32,FORMAT_F32,ULP_FIM_COP_DATA_REAL_REAL);
    /*API to add real vectors */
    arm_add_q7_opt(srcA1, srcB1, pDst1, blockSize);
    while(stat != 1);
    stat = 0;
    /*Read the destination */
    fim_read_data(BANK2,blockSize,pDst1,FORMAT_Q7,ULP_FIM_COP_DATA_REAL_REAL);
    arm_add_q15_opt(srcA2, srcB2, pDst2, blockSize);
    while(stat != 1);
    stat = 0;
    /*Read the destination */
    fim_read_data(BANK2,blockSize,pDst2,FORMAT_Q15,ULP_FIM_COP_DATA_REAL_REAL);
    arm_add_q31_opt(srcA3, srcB3, pDst3, blockSize);
    while(stat != 1);
    stat = 0;
    /*Read the destination */
    fim_read_data(BANK2,blockSize,pDst3,FORMAT_Q31,ULP_FIM_COP_DATA_REAL_REAL);
    /*API to add complex data */
    fim_vector_add_q15(incmplx,inReal,pDst2,blockSize*2, ULP_FIM_COP_DATA_CPLX_REAL);
    while(stat != 1);
    stat = 0;
    /*Read the destination */
    fim_read_data(BANK2,blockSize*2,pDst2,FORMAT_Q15,ULP_FIM_COP_DATA_CPLX_CPLX);

 case SUB_VECTOR:
    NVIC_EnableIRQ(FIM_IRQn);
    /*API to subtract a real vector */
    arm_sub_f32_opt(srcAF32,srcBF32,pDstF32,blockSize);
    while(stat != 1);
    stat = 0;
    /*Read the destination */
    fim_read_data(BANK2,blockSize,pDstF32,FORMAT_F32,ULP_FIM_COP_DATA_REAL_REAL);
    /*API to subtract a real vector */
    arm_sub_q7_opt(srcA1,srcB1,pDst1,blockSize);
    while(stat != 1);
    stat = 0;
    /*Read the destination */
    fim_read_data(BANK2,blockSize,pDst1,FORMAT_Q7,ULP_FIM_COP_DATA_REAL_REAL);
    arm_sub_q15_opt(srcA2,srcB2,pDst2,blockSize);
    while(stat != 1);
    stat = 0;
    /*Read the destination */
    fim_read_data(BANK2,blockSize,pDst2,FORMAT_Q15,ULP_FIM_COP_DATA_REAL_REAL);

    arm_sub_q31_opt(srcA3,srcB3,pDst3,blockSize);
    while(stat != 1);
    stat = 0;
    /*Read the destination */
    fim_read_data(BANK2,blockSize,pDst3,FORMAT_Q31,ULP_FIM_COP_DATA_REAL_REAL);
    /*API to subtract complex data */

fim_vector_sub_q15(inReal,inCmplx_vector,pDst2,blockSize*2,ULP_FIM_COP_DATA_REAL_CPLX);
    while(stat != 1);
    stat = 0;
    fim_read_data(BANK2,blockSize*2,pDst2,FORMAT_Q15,ULP_FIM_COP_DATA_CPLX_REAL);
    break;

 case MUL_VECTOR:
    NVIC_EnableIRQ(FIM_IRQn);
    /*API to multiply real vectors */
    arm_mult_f32_opt(srcAF32,srcBF32,0x9,blockSize);
```

```
    while(stat != 1);
    stat = 0;
    /*Read the destination */
    fim_read_data(BANK2,blockSize,pDstF32,FORMAT_F32,ULP_FIM_COP_DATA_REAL_REAL);
    arm_mult_q7_opt(srcA1,srcB1,pDst1,blockSize);
    while(stat != 1);
    stat = 0;
    fim_read_data(BANK2,blockSize,pDst1,FORMAT_Q7,ULP_FIM_COP_DATA_REAL_REAL);

    arm_mult_q15_opt(srcA2,srcB2,pDst2,blockSize);
    while(stat != 1);
    stat = 0;
    fim_read_data(BANK2,blockSize,pDst2,FORMAT_Q15,ULP_FIM_COP_DATA_REAL_REAL);

    arm_mult_q31_opt(srcA3,srcB3,pDst3,blockSize);
    while(stat != 1);
    stat = 0;
    fim_read_data(BANK2,blockSize,pDst3,FORMAT_Q31,ULP_FIM_COP_DATA_REAL_REAL);

    fim_vector_mul_q15(inReal,incmplx,pDst2,blockSize*2);
    while(stat != 1);
    stat = 0;
    fim_read_data(BANK2,blockSize*2,pDst2,FORMAT_Q15,ULP_FIM_COP_DATA_REAL_CPLX);

    /*API to multiply real vector with a complex vector */
    arm_cmplx_mult_real_q15_opt(incmplx,inReal,pDst2,blockSize*2);
    while(stat != 1);
    stat = 0;
    fim_read_data(BANK2,blockSize*2,pDst2,FORMAT_Q15,ULP_FIM_COP_DATA_REAL_CPLX);

    /* API to multiply complex data */
    arm_cmplx_mult_cmplx_q15_opt(input1,input2,pDst2,blockSize*2);
    while(stat != 1);
    stat = 0;
    fim_read_data(BANK2,blockSize,pDst2,FORMAT_Q15,ULP_FIM_COP_DATA_REAL_CPLX);
    break;

  case NORM_SQUARE:
    NVIC_EnableIRQ(FIM_IRQn);
    /*API to square cmplex number */
    arm_cmplx_mag_squared_q15_opt(sqNUM,pDst2,blockSize*2);
    while(stat != 1);
    stat = 0;
    fim_read_data(BANK1,blockSize,pDst2,FORMAT_Q15,ULP_FIM_COP_DATA_REAL_REAL);
    /*API to square realnum */
    fim_absSqr_q7(srcA1,blockSize);
    while(stat != 1);
    stat = 0;
    fim_read_data(BANK1,blockSize,pDst1,FORMAT_Q7,ULP_FIM_COP_DATA_REAL_REAL);
    fim_absSqr_q15(srcA2,blockSize);
    while(stat != 1);
    stat = 0;
    fim_read_data(BANK1,blockSize,pDst2,FORMAT_Q15,ULP_FIM_COP_DATA_REAL_REAL);
    fim_absSqr_q31(srcA3,blockSize);
    while(stat != 1);
    stat = 0;
    fim_read_data(BANK1,blockSize,pDst3,FORMAT_Q31,ULP_FIM_COP_DATA_REAL_REAL);
    fim_absSqr_f32(srcAF32,blockSize);
    while(stat != 1);
    stat = 0;
    fim_read_data(BANK1,blockSize,pDstF32,FORMAT_F32,ULP_FIM_COP_DATA_REAL_REAL);
    break;

  case MUL_MAT :
    NVIC_EnableIRQ(FIM_IRQn);
    ret = arm_mat_mult_f32_opt(&x_y,&y,&z);
    if(ret == RSI_OK)
    {
    /*No error in API*/
    /*Wait for the multiplication done*/
    while(stat != 1);
```

```
    stat = 0;
    /*Read the destination */
    matrix_len = (NUM_ROW1*NUM_COL2);
    fim_read_data(BANK2,matrix_len,pDst,FORMAT_F32,ULP_FIM_COP_DATA_REAL_REAL);
    }
    else
    {
      /*check the input parameters*/
      while(1);
    }
    ret = arm_mat_mult_q31_opt(&x31,&y31,&z31);
    if(ret == RSI_OK)
    {
    /*No error in API*/
    /*Wait for the multiplication done*/
    while(stat != 1);
    stat = 0;
    /*Read the destination */
    matrix_len = (NUM_ROW1*NUM_COL2);
    fim_read_data(BANK2,matrix_len,c2,FORMAT_Q31,ULP_FIM_COP_DATA_REAL_REAL);
    }
    else
    {
      /*check the input parameters*/
      while(1);
    }
    ret = arm_mat_mult_q15_opt(&x15,&y15,&z15,NULL);
    if(ret == RSI_OK)
    {  /*No error in API*/
       /*Wait for the multiplication done*/
       while(stat != 1);
       stat = 0;
       /*Read the destination */
       matrix_len = (NUM_ROW1*NUM_COL2);
       fim_read_data(BANK2,matrix_len,c1,FORMAT_Q15,ULP_FIM_COP_DATA_REAL_REAL);
     }
    else
    {
      /*check the input parameters*/
      while(1);
    }
}
#endif

/* For starting this example enable FIR_IIR MACRO */

#ifdef  FIR_IIR
int main()
{
         NVIC_EnableIRQ(FIM_IRQn);
         arm_fir_init_f32_opt(&SF32, NUM_TAPS , (int32_t *)&filterCoeffsF32[0],
&filerStateF32[0],SAMPLES_LENGTH);
         arm_fir_f32_opt(&SF32,   testInputF32,  pDstF32 , SAMPLES_LENGTH);
         while(stat != 1);
         stat = 0;
         fim_read_data(BANK2,((2 + SAMPLES_LENGTH) - 1),pDstF32,FORMAT_F32
,ULP_FIM_COP_DATA_REAL_REAL);

         arm_fir_init_q31_opt(&S31, NUM_TAPS , (q31_t *)&filterCoeffsQ31[0],
&filerStateQ31[0], SAMPLES_LENGTH);
         arm_fir_q31_opt(&S31, testInputQ31,pDst3,SAMPLES_LENGTH);
         while(stat != 1);
         stat = 0;
         fim_read_data(BANK2,((NUM_TAPS + SAMPLES_LENGTH) - 1),pDst3,FORMAT_Q31
,ULP_FIM_COP_DATA_REAL_REAL);
         floatToHex(firInput , 2048, testInputQ31_1024,1);
         floatToHex(filcoeff , 2048, filterCoeffsQ31_1024,1);
         arm_fir_init_q31_opt(&S31, 2 , (q31_t *)&filterCoeffsQ31_1024[0],
&filerStateQ31[0],1021);
         arm_fir_q31_opt(&S31, testInputQ31_1024,pDst3,1021);
         while(stat != 1);
```

```
        stat = 0;
        fim_read_data(BANK4,((1021) +1),pDst3,FORMAT_Q31 ,ULP_FIM_COP_DATA_REAL_REAL);
        arm_fir_init_q15_opt(&S15, NUM_TAPS , (q15_t *)&filterCoeffsQ15[0],
&filerStateQ15[0], SAMPLES_LENGTH);
        arm_fir_q15_opt(&S15, testInputQ15,pDst2,SAMPLES_LENGTH);
        while(stat != 1);
        stat = 0;
        fim_read_data(BANK2,((2 + SAMPLES_LENGTH) -
1),pDst4,FORMAT_Q15,ULP_FIM_COP_DATA_REAL_REAL );
        arm_fir_init_q7_opt(&S7, NUM_TAPS , (q7_t *)&filterCoeffsQ7[0], &filerStateQ7[0],
SAMPLES_LENGTH);
        arm_fir_q7_opt(&S7, testInputQ7,pDst1,SAMPLES_LENGTH);
        while(stat != 1);
        stat = 0;
        fim_read_data(BANK2,((2 + SAMPLES_LENGTH) -
1),pDst1,FORMAT_Q7,ULP_FIM_COP_DATA_REAL_REAL );
        arm_fir_init_q15_opt(&S15, NUM_TAPS , (q15_t *)&filterCoeffsQ15_reaL,
&filerStateQ15[0], SAMPLES_LENGTH);
        rsi_fim_fir_q15(&S15, ComplexInputQ31
,pDst2,SAMPLES_LENGTH,ULP_FIM_COP_DATA_REAL_CPLX,0x00,BANK1,BANK2);
        while(stat != 1);
        stat = 0;
        fim_read_data(BANK2,((NUM_TAPS + SAMPLES_LENGTH) -
1)*2,pDst2,FORMAT_Q15,ULP_FIM_COP_DATA_REAL_CPLX );
        arm_fir_init_q15_opt(&S15, NUM_TAPS , (q15_t *)&ComplexCoeffQ15_1,
&filerStateQ15[0], SAMPLES_LENGTH);
        rsi_fim_fir_q15(&S15, RealInputQ15
,pDst2,SAMPLES_LENGTH,ULP_FIM_COP_DATA_CPLX_REAL,0x00,BANK1,BANK2);
        while(stat != 1);
        stat = 0;
        fim_read_data(BANK2,((NUM_TAPS + SAMPLES_LENGTH) -
1)*2,pDst2,FORMAT_Q15,ULP_FIM_COP_DATA_CPLX_REAL );
        arm_fir_init_q15_opt(&S15, NUM_TAPS , (q15_t *)&ComplexCoeffQ15_1,
&filerStateQ15[0], SAMPLES_LENGTH);
        rsi_fim_fir_q15(&S15, ComplexInputQ31
,pDst2,SAMPLES_LENGTH,ULP_FIM_COP_DATA_CPLX_CPLX,0x00,BANK1,BANK2);
        while(stat != 1);
        stat = 0;
        fim_read_data(BANK2,((NUM_TAPS + SAMPLES_LENGTH) -
1)*2,pDst2,FORMAT_Q15,ULP_FIM_COP_DATA_CPLX_CPLX )

 }
/*stop main here */
    while(1);
}
#endif

/* For starting this example enable INTERPOLATE MACRO */
#ifdef INTERPOLATE

int main(STATIC INLINE void)
{

    RSI_PS_UlpssPeriPowerDown(ULPSS_PWRGATE_ULP_FIM);
    RSI_PS_UlpssPeriPowerUp(ULPSS_PWRGATE_ULP_FIM);
    /*Fim clock enable */
    ROM_ULPSS_PeripheralEnable(ULPCLK, ULP_FIM_CLK,ENABLE_STATIC_CLK);

    NVIC_EnableIRQ(FIM_IRQn);

    rsi_fim_Iir_init_f32( &If32,NUM_TAPS, (int32_t *)&filterCoeffsF32[0],pvCoeffs,NULL);
    rsi_fim_Iir_f32(&If32,testInputF32,pDst,SAMPLES_LENGTH,0x00,BANK1,BANK2);
    while(stat != 1);
    stat = 0;
    fim_read_data(BANK2,((2 + SAMPLES_LENGTH) -
1),pDst,FORMAT_F32,ULP_FIM_COP_DATA_REAL_REAL);
//////////////////////////////Q31//////////////////////////////////////////
    rsi_fim_Iir_init_q31( &Iq31,NUM_TAPS, (int32_t *)&filterCoeffsQ31[0], pvCoeffs,NULL);
    rsi_fim_Iir_q31(&Iq31,testInputQ31,pDst3,SAMPLES_LENGTH,0x00,BANK1,BANK2);
    while(stat != 1);
    stat = 0;
```

```
    fim_read_data(BANK2,((2+SAMPLES_LENGTH) -
1),pDst3,FORMAT_Q31,ULP_FIM_COP_DATA_REAL_REAL);
//////////////////////////////////////////Q15//////////////////////////////////////////
    rsi_fim_Iir_init_q15( &Iq15,NUM_TAPS, (int16_t *)&filterCoeffsQ15[0],pvCoffsq15,NULL);
    rsi_fim_Iir_q15(&Iq15,testInputQ15,pDst2,SAMPLES_LENGTH,
ULP_FIM_COP_DATA_REAL_REAL,0x00,BANK1,BANK2);
    while(stat != 1);
    stat = 0;
    fim_read_data(BANK2,((2+SAMPLES_LENGTH) -
1),pDst2,FORMAT_Q15,ULP_FIM_COP_DATA_REAL_REAL);

    while(1);
}
#endif

/* For starting this example enable IIR MACRO */
#ifdef IIR
  /*
  Input Data (Real numbers) = {0.99, 0.309, -0.809,-0.809,0.309,0.99};
  Coefficients (Complex numbers = {0.821 + 0.5i , 0.645 + 0.4i};
 */
int main()
{
    RSI_PS_UlpssPeriPowerDown(ULPSS_PWRGATE_ULP_FIM);
    RSI_PS_UlpssPeriPowerUp(ULPSS_PWRGATE_ULP_FIM);
    /*Fim clock enable */
    RSI_ULPSS_PeripheralEnable(ULPCLK,ULP_FIM_CLK,ENABLE_STATIC_CLK);

    NVIC_EnableIRQ(FIM_IRQn);

    rsi_fim_Iir_init_q15( &Iq15,NUM_TAPS, (int16_t
*)&filterCoeffsQ15_reaL[0],pvCoffsq15,NULL);
    rsi_fim_Iir_q15(&Iq15,ComplexInputQ31,pDst2,SAMPLES_LENGTH,
ULP_FIM_COP_DATA_REAL_CPLX,0x00,BANK1,BANK2);

    while(stat != 1);
    stat = 0;

    fim_read_data(BANK2,((2*SAMPLES_LENGTH) -
1)*2,pDst2,FORMAT_Q15,ULP_FIM_COP_DATA_REAL_CPLX);
  while(1);
}
#endif
```

Code Block 14 FIM Example

Note:

FIM having CMSIS supporting SAPIs as well as Redpine MCU SAPIs ,so above example some CMSIS support SAPIs call. for CMSIS SAPIs refer CMSIS DSP section or refer this link http://www.keil.com/pack/doc/CMSIS/DSP/html/index.html.

20.3 API Descriptions

20.3.1 fim_read_data

Source File : rsi_rom_fim.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void fim_read_data( uint32_t bank,uint32_t length,volatile STATIC INLINE
void *pDst,uint8_t data_type,typ_data_t type_data)
```

Description

This API is used to set the FIM to read the output.

Parameters

Parameter	Description
bank	This is the output bank address
length	This is the size of the input array
pDst	This is the pointer to required output array
data_type	This specifies q7,q15 , q31 formats
typ_data	This is the enum value of type of data, following are the enum values : • ULP_FIM_COP_DATA_REAL_REAL • ULP_FIM_COP_DATA_CPLX_REAL • ULP_FIM_COP_DATA_REAL_CPLX • ULP_FIM_COP_DATA_CPLX_CPLX

Return values

None

Example

```
/* declaration */
here q15_t is signed short int.
#define BANK2                        (0x1000 >> 2)
#define SAMPLES_LENGTH              6
q15_t pDst2[30];
#define  FORMAT_Q15    3

/*read data */
fim_read_data(BANK2,((2+SAMPLES_LENGTH) -
1)*2,pDst2,FORMAT_Q15,ULP_FIM_COP_DATA_CPLX_CPLX);
```

20.3.2 rsi_fim_scalar_add_q15

Source File : rsi_fim_filterScalar.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
STATIC INLINE void rsi_fim_scalar_add_q15 (q15_t *pSrc, q15_t *scale, uint32_t blockSize,
typ_data_t typ_data, uint16_t inBank, uint16_t outBank)
```

Description

This API is used to set the FIM Scalar substraction for real data.

Parameters

Parameter	Description
pSrc	This is the pointer to an input vector
scale	This is the constant value that need to be subtracted from each elements of vector array
blockSize	This is the size of the input array

Parameter	Description
typ_data	This is to specify real-complex, complex-real or complex-complex data The possible values for this parameter are: • ULP_FIM_COP_DATA_REAL_REAL • ULP_FIM_COP_DATA_CPLX_REAL, • ULP_FIM_COP_DATA_REAL_CPLX, • ULP_FIM_COP_DATA_CPLX_CPLX
inBank	This is to store input data samples
outBank	This stores output data samples

Return values

None

Example

```
uint32_t blockSize = 5;
#define BANK1        (0x0800 >> 2)
q15_t incmplx[10] =
{0x3333,0xC000,0xF334,0x3333,0x4000,0x6916,0x528F,0x8290,0xE51F,0xCCC};
q15_t scalar_cmplx[2] = {0x2000,0x4000};
rsi_fim_scalar_add_q15(incmplx,scalar_cmplx,blockSize*2,ULP_FIM_COP_DATA_CPLX_REAL,0x00,BA
NK1);
```

20.3.3 fim_scalar_sub_q7

Source File : rsi_rom_fim.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void fim_scalar_sub_q7(q7_t *pSrc,q7_t scale,q7_t *pDst, uint32_t blockSize)
```

Description

Th API used to set the FIM Scalar Subtraction.

Parameters

Parameter	Description
pSrc	This is the pointer to input vector
scale	This is the constant value that need to be subtracted from each elements of vector array
pDst	This is the pointer to output vector
blockSize	This is the size of the input array

Return values

None

Example

```
/* declaration */
here  q7_t is signed char.
q7_t srcA1[5] = {0xC,0x26,0xC0,0x69,0x83};
```

```
q7_t sCALAR1 = 0x39;
q7_t pDst1[10];
uint32_t blockSize = 5;

/* Format q7 only for real values */
 fim_scalar_sub_q7(srcA1,sCALAR1,pDst1,blockSize);
```

20.3.4 rsi_fim_scalar_sub_q15

Source File : rsi_rom_fim.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void rsi_fim_scalar_sub_q15(q15_t *pSrc,q15_t *scale, q15_t *pDst,uint32_t
blockSize, typ_data_t typ_data,uint16_t inBank, uint16_t outBank)
```

Description

Th API used to set the FIM Scalar Subtraction.

Parameters

Parameter	Description
pSrc	This is the pointer to input vector
scale	This is the constant value that need to be subtracted from each elements of vector array
pDst	This is the pointer to output vector
blockSize	This is the size of the input array
typ_data	This is to specify real-complex , complex-real or complex-complex data
inBank	This is to store input data samples
outBank	This is to store output data samples

Return values

None

Example

```
/* declaration */
#define BANK1      (0x0800 >> 2)
here  q7_t is signed Short integer .
q15_t incmplx[10] =
{0x3333,0xC000,0xF334,0x3333,0x4000,0x6916,0x528F,0x8290,0xE51F,0xCCC};
q15_t scalar_cmplx[2] = {0x2000,0x4000};
uint32_t blockSize = 5;

/* Format q15*/
rsi_fim_scalar_sub_q15(incmplx,scalar_cmplx/*0x3920*/,NULL,blockSize*2,ULP_FIM_COP_DATA_CP
LX_CPLX,0x00,BANK1);
```

20.3.5 fim_scalar_sub_q31

Source File : rsi_rom_fim.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void fim_scalar_sub_q31(q31_t *pSrc,q31_t scale, q31_t *pDst,uint32_t
blockSize)
```

Description

Th API used to set the FIM Scalar Subtraction.

Parameters

Parameter	Description
pSrc	This is the pointer to an input vector
scale	This is the constant value that need to be subtracted from each elements of vector array
pDst	This is the pointer to output vector
blockSize	This is the size of the input array

Return values

None

Example

```
/* declaration */
here q31_t is signed int.
q31_t srcA3[6] = {0xCCCCCCC, 0x26666666, 0xC0000000, 0x6916872B, 0x828F5c29};
q31_t sCALAR3 = 0x39FF2E48 ;
q31_t pDst3[1025];
uint32_t blockSize = 5;
/* Format q31*/
fim_scalar_sub_q31(srcA3,sCALAR3,pDst3/*0x39ff2000*/,blockSize);
```

20.3.6 fim_scalar_sub_f32

Source File : rsi_rom_fim.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void fim_scalar_sub_f32( int32_t *pSrc,int32_t scale,int32_t *pDst, uint32_t
length)
```

Description

This API is used to set the FIM Scalar Subtraction

Parameters

Parameter	Description
pSrc	This is the pointer to an input vector
scale	This is the constant value that need to be subtracted from each elements of vector array
pDst	This is the pointer to output vector
length	This is the size of the input array

Return values

None

Example

```
/* declaration */
int32_t is signed int.
int32_t srcAF32[5] = {0xCCCCC, 0x266666, 0xFFC00000, 0x691687, 0xFF828F5D};
int32_t sCALARF32 = 0x39FF2E;
int32_t pDstF32[10];
uint32_t blockSize = 5;
/* Format f32 */
fim_scalar_sub_f32(srcAF32,sCALARF32/*0x039F0200*/,pDstF32,blockSize);
```

20.3.7 fim_scalar_mul_q15

Source File : rsi_rom_fim.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void fim_scalar_mul_q15(q15_t *pSrc,q15_t *scale,uint32_t blockSize,
typ_data_t typ_data )
```

Description

This API is used to set the FIM Scalar Multiplication

Parameters

Parameter	Description
pSrc	This is the pointer to an input vector
scale	This is the constant value that need to be subtracted from each elements of vector array
blockSize	This is the size of the input array
typ_data	This is the enum value of type of data, following are the enum value • ULP_FIM_COP_DATA_REAL_REAL • ULP_FIM_COP_DATA_CPLX_REAL • ULP_FIM_COP_DATA_REAL_CPLX • ULP_FIM_COP_DATA_CPLX_CPLX

Return values

None

Example

```
/* declaration */
here q15_t is signed short int.
q15_t incmplx[10] =
{0x3333,0xC000,0xF334,0x3333,0x4000,0x6916,0x528F,0x8290,0xE51F,0xCCC};
q15_t scalar_cmplx[2] = {0x2000,0x4000};
uint32_t blockSize = 5;

/*API to multiply complex data */
 fim_scalar_mul_q15(incmplx,scalar_cmplx,blockSize*2,ULP_FIM_COP_DATA_CPLX_CPLX);
```

20.3.8 fim_vector_add_q15

Source File : rsi_rom_fim.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void fim_vector_add_q15(q15_t *pIn1, q15_t *pIn2,q15_t *pDst,uint32_t
blockSize, typ_data_t typ_data)
```

Description

This API is used to set the FIM Vector Addition

Parameters

Parameter	Description
pIn1	This is the pointer to an input vector A
pIn2	This is the pointer to an input vector B
blockSize	This is the size of the input array
typ_data	This is the enum value of type of data, following are enum values: • ULP_FIM_COP_DATA_REAL_REAL • ULP_FIM_COP_DATA_CPLX_REAL • ULP_FIM_COP_DATA_REAL_CPLX • ULP_FIM_COP_DATA_CPLX_CPLX

Return values

None

Example

```
/* declaration */
here q15_t is signed short int.
q15_t incmplx[10] =
{0x3333,0xC000,0xF334,0x3333,0x4000,0x6916,0x528F,0x8290,0xE51F,0xCCC};
q15_t inReal [5] = {0x3333,0xF334,0x4000,0x528F,0xE51F};
uint32_t blockSize = 5;

/*API to add complex data */
fim_vector_add_q15(incmplx,inReal,pDst2,blockSize*2, ULP_FIM_COP_DATA_CPLX_REAL);
```

20.3.9 fim_vector_sub_q15

Source File : rsi_rom_fim.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void fim_vector_sub_q15( q15_t *pIn1, q15_t *pIn2,q15_t *pDst,uint32_t
blockSize,typ_data_t typ_data)
```

Description

This API is used to set the FIM Vector Subtraction.

Parameters

Parameter	Description
pIn1	This is the pointer to an input vector A
pIn2	This is the pointer to an input vector B
blockSize	This is the size of the input array
typ_data	This is the enum value of type of data, following are the enum values: • ULP_FIM_COP_DATA_REAL_REAL • ULP_FIM_COP_DATA_CPLX_REAL • ULP_FIM_COP_DATA_REAL_CPLX • ULP_FIM_COP_DATA_CPLX_CPLX

Return values

None

Example

```
/* declaration */
here q15_t is signed short int.
q15_t inputCmplx[10] = {0x3333,0xD99A, 0xF334,0x3333, 0x4000,0x8000, 0x528F,0x1999,
0xE51F,0x4000};
q15_t inCmplx_vector[10] =
{0xCCC,0x3333,0x2666,0xF334,0xC000,0x4000,0x6916,0x528F,0x8290,0xE51F};
q15_t pDst2[30];
uint32_t blockSize = 5;

fim_vector_sub_q15(inputCmplx,inCmplx_vector,pDst2,blockSize*2,ULP_FIM_COP_DATA_CPLX_CPLX)
;
```

20.3.10 fim_vector_mul_q15

Source File : rsi_rom_fim.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void fim_vector_mul_q15( q15_t *pIn1, q15_t *pIn2,q15_t *pDst,uint32_t
blockSize)
```

Description

This API is used for Vector Multiplication for complex-real data.

Parameters

Parameter	Description
pIn1	This is the pointer to an input vector A
pIn2	This is the pointer to an input vector B
blockSize	This is the size of the input array
typ_data	This is the enum value of type of data, following are the enum values: • ULP_FIM_COP_DATA_REAL_REAL • ULP_FIM_COP_DATA_CPLX_REAL • ULP_FIM_COP_DATA_REAL_CPLX

Parameter	Description
	• ULP_FIM_COP_DATA_CPLX_CPLX

Return values

None

Example

```
/* declaration */
here q15_t is signed short int.
q15_t inReal [5] = {0x3333,0xF334,0x4000,0x528F,0xE51F};
q15_t incmplx[10] =
{0x3333,0xC000,0xF334,0x3333,0x4000,0x6916,0x528F,0x8290,0xE51F,0xCCC};
q15_t pDst2[30];
uint32_t blockSize = 5;

fim_vector_mul_q15(inReal,incmplx,pDst2,blockSize*2);
```

20.3.11 fim_absSqr_q7

Source File : rsi_rom_fim.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void fim_absSqr_q7( q7_t *pSrc,uint32_t length)
```

Description

This API is used to set the FIM Absolute Squaring for real number.

Parameters

Parameter	Description
pSrc	This is the pointer of input for squaring a number
length	This is the size of the input array

Return values

None

Example

```
/* declaration */
here q7_t is signed char.
q7_t srcA1[5] = {0xC,0x26,0xC0,0x69,0x83};
uint32_t blockSize = 5;

/*API to square realnum */
fim_absSqr_q7(srcA1,blockSize);
```

20.3.12 fim_absSqr_q15

Source File : rsi_rom_fim.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void fim_absSqr_q15 (q15_t *pSrc, uint32_t length)
```

Description

This API is used to set the FIM Absolute Squaring for real number.

Parameters

Parameter	Description
pSrc	This is the pointer of input for squaring a number
length	This is the size of the input array

Return values

None

Example

```
/* declaration */
here q15_t is signed char.
q15_t srcA2[5] = {0xCCC, 0x2666, 0xC000, 0x6916, 0x8290};
uint32_t blockSize = 5;

/*API to square realnum */
fim_absSqr_q15(srcA2,blockSize);
```

20.3.13 fim_absSqr_q31

Source File : rsi_rom_fim.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void fim_absSqr_q31 (q31_t *pSrc, uint32_t length)
```

Description

This API is used to set the FIM Absolute Squaring for real number.

Parameters

Parameter	Description
pSrc	This is the pointer of input for squaring a number
length	This is the size of the input array

Return values

None

Example

```
/* declaration */
here q31_t is signed char.
q31_t srcA3[6] = {0xCCCCCCC, 0x26666666, 0xC0000000, 0x6916872B, 0x828F5c29};
uint32_t blockSize = 6;

/*API to square realnum */
fim_absSqr_q31(srcA3,blockSize);
```

20.3.14 fim_absSqr_f32

Source File : rsi_rom_fim.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void fim_absSqr_f32 (int32_t *pSrc, uint32_t length)
```

Description

This API is used to set the FIM Absolute Squaring for real number.

Parameters

Parameter	Description
pSrc	This is the pointer of input for squaring a number
length	This is the size of the input array

Return values

None

Example

```
/* declaration */
int32_t srcAF32[5] = {0xCCCCC, 0x266666, 0xFFC00000, 0x691687, 0xFF828F5D}
uint32_t blockSize = 5;
/*API to square realnum */
fim_absSqr_f32(srcF32,blockSize);
```

20.3.15 fim_interrupt_handler

Source File : rsi_rom_fim.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void fim_interrupt_handler( volatile M4_FIM_TYPE *ptFim)
```

Description

This API Clears interrupt status of fim

Parameters

Parameter	Description
ptFim	This is the pointer to the FIM register instance

Return values

none

Example

```
fim_interrupt_handler(M4_FIM);
```

20.3.16 rsi_fim_fir_q15

Source File : rsi_fim_filterScalar.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
STATIC INLINE void rsi_fim_fir_q15 (arm_fir_instance_q15_opt *S, q15_t *pSrc, q15_t *pDst,
uint32_t blockSize, typ_data_t typ_data, uint16_t inBank1, uint16_t inBank2, uint16_t
outBank)
```

Description

This API is used to initialization function for the floating-point IIR lattice filter.

Parameters

Parameter	Description
S	This is the pointer to an instance of the Q15 FIR interpolator structure.
pSrc	This is the pointer to the block of input data.
pDst	This points to the block of output data.
blockSize	These are the number of samples to process per call.
typ_data	This is to specify real-complex , complex-real or complex-complex data The possible values for this parameter are: • ULP_FIM_COP_DATA_REAL_REAL • ULP_FIM_COP_DATA_CPLX_REAL, • ULP_FIM_COP_DATA_REAL_CPLX, • ULP_FIM_COP_DATA_CPLX_CPLX
inBank1	This is to store input data samples
inBank2	This is to store input data samples
outBank	This stores output data samples

Return values

None

Example

```
/* declaration */
arm_fir_instance_q15_opt S15
typedef int16_t q15_t;
q15_t ComplexInputQ31[12] = {0x0CCC,0x1999,0xE667,0x2666,0x1999,0xD99A,
0x1999,0xD99A,0xE667,0x2666,0x0CCC,0x1999};
q15_t pDst2[30];
#define SAMPLES_LENGTH              6
#define BANK1                        (0x0800 >> 2)
#define BANK2                        (0x1000 >> 2)

 rsi_fim_fir_q15(&S15, ComplexInputQ31
,pDst2,SAMPLES_LENGTH,ULP_FIM_COP_DATA_REAL_CPLX,0x00,BANK1,BANK2)
```

20.3.17 rsi_fim_Iir_init_f32

Source File : rsi_fim_filterScalar.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
void rsi_fim_Iir_init_f32 (fim_iir_instance_f32 *S, uint16_t numStages, int32_t *pCoeffs,
int32_t *pvCoeffs, int32_t *pState)
```

Description

This API is used to initialization function for the floating-point IIR lattice filter.

Parameters

Parameter	Description
S	This is the pointer to an instance of the floating-point IIR lattice structure.
numStages	These are the number of stages in the filter.
pCoeffs	This points to the reflection coefficient buffer. The array is of length numStages.
pvCoeffs	This points to the ladder coefficient buffer. The array is of length numStages+1.
pState	This points to the state buffer. The array is of length numStages+blockS

Return values

None

Example

```
/* declaration */
fim_iir_instance_f32 If32;
#define NUM_TAPS       2
int32_t filterCoeffsF32[NUM_TAPS] = {0x5A8240,0x5A8240};
int32_t pvCoeffs[2] =  {0,0};
#define NULL 0

 rsi_fim_Iir_init_f32( &If32,NUM_TAPS, (int32_t *)&filterCoeffsF32[0],pvCoeffs,NULL);
```

20.3.18 rsi_fim_Iir_f32

Source File : rsi_fim_filterScalar.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
STATIC INLINE void rsi_fim_Iir_f32(fim_iir_instance_f32 *S,int32_t *pSrc,int32_t
*pDst,uint32_t blockSize,uint16_t inBank1, uint16_t inBank2, uint16_t outBank)
```

Description

This API is used to processing function for the floating-point IIR lattice filter

Parameters

Parameter	Description
S	This is the pointer to an instance of the Q15 FIR interpolator structure.
pSrc	This is the pointer to the block of input data.
pDst	This points to the block of output data.
blockSize	These are the number of samples to process per call.

Parameter	Description
typ_data	This is to specify real-complex , complex-real or complex-complex data The possible values for this parameter are: • ULP_FIM_COP_DATA_REAL_REAL • ULP_FIM_COP_DATA_CPLX_REAL, • ULP_FIM_COP_DATA_REAL_CPLX, • ULP_FIM_COP_DATA_CPLX_CPLX
inBank1	This is to store input data samples
inBank2	This is to store input data samples
outBank	This stores output data samples

Return values

None

Example

```
/* declaration */
#define BANK1                        (0x0800 >> 2)
#define BANK2                        (0x1000 >> 2)
#define SAMPLES_LENGTH              6
fim_iir_instance_f32 If32;
typedef int16_t q15_t;
int32_t testInputF32[SAMPLES_LENGTH] = {0x7EB851,0x278D47,0xFF9872B1,0xFF9872B1,
0x278D47,0x7EB851};
uint32_t pDst[255] ;


rsi_fim_Iir_f32(&If32,testInputF32,pDst,SAMPLES_LENGTH,0x00,BANK1,BANK2);
```

20.3.19 rsi_fim_Iir_init_q31

Source File :rsi_fim_filterScalar.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
STATIC INLINE void rsi_fim_Iir_init_q31 (fim_iir_instance_q31 *S, uint16_t numStages,
q31_t *pCoeffs, q31_t *pvCoeffs, uint32_t *pState)
```

Description

This API is used to initialization function for the Q31 IIR lattice filter.

Parameters

Parameter	Description
S	This is the pointer to an instance of the Q31 IIR lattice structure
numStages	These are the number of stages in the filter.
pCoeffs	This points to the reflection coefficient buffer. The array is of length numStages.
pvCoeffs	This points to the ladder coefficient buffer. The array is of length numStages+1.
pState	This points to the state buffer. The array is of length numStages+blockS

Return values

None

Example

```
/* declaration */
#define NUM_TAPS                 2
q31_t  filterCoeffsQ31[NUM_TAPS] = {0x5A8240B7, 0x5A8240B7};
int32_t pvCoeffs[2] =  {0,0};
#define NULL 0
fim_iir_instance_q31 Iq31;

rsi_fim_Iir_init_q31( &Iq31,NUM_TAPS, (int32_t *)&filterCoeffsQ31[0], pvCoeffs,NULL);
```

20.3.20 rsi_fim_Iir_q31

Source File : rsi_fim_filterScalar.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
STATIC INLINE void rsi_fim_Iir_q31 (fim_iir_instance_q31 *S, int32_t *pSrc, q31_t *pDst,
uint32_t blockSize, uint16_t inBank1, uint16_t inBank2, uint16_t outBank)
```

Description

This API is used to processing function for the floating-point IIR lattice filter.

Parameters

Parameter	Description
S	This is the pointer to an instance of the Q31 IIR lattice structure
pSrc	This is the pointer to the block of input data.
pDst	This points to the block of output data.
blockSize	These are the number of samples to process per call.
inBank1	This is to store input data samples
inBank2	This is to store input data samples
outBank	This stores output data samples

Return values

None

Example

```
/* declaration */
#define BANK2                        (0x1000 >> 2)
#define BANK1                        (0x0800 >> 2)
#define SAMPLES_LENGTH              6
q31_t pDst3[13];
q31_t testInputQ31[SAMPLES_LENGTH] = {0x7EB851EB,0x278D4FDF,0x9872B021,
0x9872B021,0x278D4FDF,0x7EB851EB};
fim_iir_instance_q31 Iq31;

rsi_fim_Iir_q31(&Iq31,testInputQ31,pDst3,SAMPLES_LENGTH,0x00,BANK1,BANK2);
```

20.3.21 rsi_fim_Iir_init_q15

Source File : rsi_fim_filterScalar.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
STATIC INLINE void rsi_fim_Iir_init_q15 (fim_iir_instance_q15 *S, uint16_t numStages,
q15_t *pCoeffs, q15_t *pvCoeffs, q15_t *pState)
```

Description

This API is used initialization function for the Q15 IIR lattice filter. .

Parameters

Parameter	Description
S	This is to an instance of the Q15 IIR lattice structure.
numStages	These are the number of stages in the filter.
pCoeffs	This points to the reflection coefficient buffer. The array is of length numStages.
pvCoeffs	This points to the ladder coefficient buffer. The array is of length numStages+1.
pState	This points to the state buffer. The array is of length numStages+blockS

Return values

None

Example

```
/* declaration */
fim_iir_instance_q15 Iq15;
#define NUM_TAPS                    2
typedef int16_t q15_t;
q15_t  filterCoeffsQ15[NUM_TAPS] = {0x5A82, 0x5A82};
q15_t pvCoffsq15[2] = {0};
#define NULL 0

rsi_fim_Iir_init_q15( &Iq15,NUM_TAPS, (int16_t *)&filterCoeffsQ15[0],pvCoffsq15,NULL);
```

20.3.22 rsi_fim_Iir_q15

Source File : rsi_fim_filterScalar.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
STATIC INLINE void rsi_fim_Iir_q15(fim_iir_instance_q15 *S,q15_t *pSrc,q15_t
*pDst,uint32_t blockSize, typ_data_t typ_data,uint16_t inBank1, uint16_t inBank2, uint16_t
outBank)
```

Description

This API is used to processing function for the Q15 IIR lattice filter

Parameters

Parameter	Description
S	This is the pointer to an instance of the Q15 FIR interpolator structure.
pSrc	This is the pointer to the block of input data.

Parameter	Description
pDst	This points to the block of output data.
blockSize	These are the number of samples to process per call.
typ_data	This is to specify real-complex , complex-real or complex-complex data The possible values for this parameter are: • ULP_FIM_COP_DATA_REAL_REAL • ULP_FIM_COP_DATA_CPLX_REAL, • ULP_FIM_COP_DATA_REAL_CPLX, • ULP_FIM_COP_DATA_CPLX_CPLX
inBank1	This is to store input data samples
inBank2	This is to store input data samples
outBank	This stores output data samples

Return values

None

Example

```
/* declaration */
fim_iir_instance_q15 Iq15;
#define BANK1                        (0x0800 >> 2)
#define BANK2                        (0x1000 >> 2)
#define SAMPLES_LENGTH                6
q15_t pDst2[5];
q15_t ComplexInputQ31[12] = {0x0CCC,0x1999,0xE667,0x2666,0x1999,0xD99A,
0x1999,0xD99A,0xE667,0x2666,0x0CCC,0x1999};


rsi_fim_Iir_q15(&Iq15,testInputQ15,pDst2,SAMPLES_LENGTH,
ULP_FIM_COP_DATA_REAL_REAL,0x00,BANK1,BANK2);
```

20.3.23 rsi_fim_fir_interpolate_q15

Source File : rsi_fim_filterScalar.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
void rsi_fim_fir_interpolate_q15( arm_fir_interpolate_instance_q15_opt *S,q15_t *
pSrc,q15_t *pDst,
uint32_t blockSize,typ_data_t typ_data,uint16_t inBank1, uint16_t inBank2, uint16_t
outBank)
```

Description

This API is used to processing function for the floating-point IIR lattice filter

Parameters

Parameter	Description
S	This is the pointer to an instance of the Q15 FIR interpolator structure.
pSrc	This is the pointer to the block of input data.

Parameter	Description
pDst	This points to the block of output data.
blockSize	These are number of samples to process per call.
typ_data	This is to specify real-complex , complex-real or complex-complex data The possible values for this parameter • ULP_FIM_COP_DATA_REAL_REAL • ULP_FIM_COP_DATA_CPLX_REAL, • ULP_FIM_COP_DATA_REAL_CPLX, • ULP_FIM_COP_DATA_CPLX_CPLX
inBank1	This is to store input data samples
inBank2	This is to store input data samples
outBank	This stores output data samples

Return values

None

Example

```
/* declaration */
#define BANK1                        (0x0800 >> 2)
#define BANK2                        (0x1000 >> 2)
#define SAMPLES_LENGTH                6
arm_fir_interpolate_instance_q15_opt vsInterpolateQ15;
q15_t RealInputQ15[SAMPLES_LENGTH]   = {0x7EB8,0x278D,0x9872, 0x9872,0x278D,0x7EB8};

rsi_fim_fir_interpolate_q15(&vsInterpolateQ15,RealInputQ15,NULL,SAMPLES_LENGTH,ULP_FIM_COP
_DATA_REAL_CPLX,0x00,BANK1,BANK2);
```

21 Overview

This section explains how to configure and use the Generic Serial Peripheral Interface using Redpine MCU SAPIs.

Please refer CMSIS Support Page for information

22 Micro Direct Memory Access (µDMA)

22.1 Overview

This section explains how to configure and use the Micro Direct Memory Access using Redpine MCU SAPIs.

22.2 Programming sequence

```
#include "rsi_chip.h" /* Redpine_MCU_Vx.y.z\Host_MCU\common\chip\inc */

/* Private typedef -----------------------------------------------------------*/

/* Private macro -------------------------------------------------------------*/
#define  SIZE_BUFFERS            (1024)
#define  DMA_DESC_LEN            (1024)
#define  CHNL0                    0
/* Private define ------------------------------------------------------------*/

/* Private variables ---------------------------------------------------------*/
extern RSI_DRIVER_UDMA Driver_UDMA0;
static RSI_DRIVER_UDMA *UDMAdrv0 = &Driver_UDMA0;
extern RSI_UDMA_HANDLE_T udmaHandle0;

uint32_t src0[SIZE_BUFFERS];
uint32_t dst0[SIZE_BUFFERS];

volatile uint32_t done,ret;
/* Private function prototypes -----------------------------------------------*/

/* Private functions ---------------------------------------------------------*/

/**
 * @brief      UDMA  controller transfer descriptor chain complete callback
 * @param[in]  event dma transfer events
 * @param[in]  ch    dma channel number
 * @return     None
 */
void udmaTransferComplete(uint32_t event, uint8_t ch)
{
  if(event == UDMA_EVENT_XFER_DONE)
  {
    if(ch == 0)
    {
      done = 1;
    }
  }
}

/**
 * @brief  Main program.
 * @param  None
 * @retval None
 */
int main()
{
   int loop =0;

   RSI_UDMA_CHA_CONFIG_DATA_T control;
   RSI_UDMA_CHA_CFG_T config;
     /*Configures the system default clock and power configurations*/
   SystemCoreClockUpdate();
       /* Initialize UART for debug prints */
   DEBUGINIT();
   memset(&control, 0, sizeof(RSI_UDMA_CHA_CONFIG_DATA_T));
   memset(&config, 0, sizeof(RSI_UDMA_CHA_CFG_T));

   config.altStruct = 0;
```

```
    config.burstReq  = 1;
    config.channelPrioHigh = 0;
    config.dmaCh = CHNL0;
    config.periAck = 0;
    config.periphReq = 0;
    config.reqMask = 0;

    control.transferType       = UDMA_MODE_AUTO;
    control.nextBurst                    = 0;
    control.totalNumOfDMATrans = (DMA_DESC_LEN-1);
    control.rPower             = ARBSIZE_16;
    control.srcProtCtrl               = 0x000;
    control.dstProtCtrl               = 0x000;
    control.srcSize            = SRC_SIZE_32;
    control.srcInc             = SRC_INC_32;
    control.dstSize            = DST_SIZE_32;
    control.dstInc             = DST_INC_32;

    for (loop = 0; loop < SIZE_BUFFERS; loop++)
    {
      src0[loop] = loop + 1 ;
      dst0[loop] = 0;
    }
    /*  Prints on hyper-terminal  */
    DEBUGOUT("UDMA Memory to Memory data transfer example\r\n");
    /* Initialize dma */
    UDMAdrv0->Initialize();

    /* Configure dma channel */
    UDMAdrv0->ChannelConfigure( CHNL0,(uint32_t)src0, (uint32_t)dst0, SIZE_BUFFERS,
                                control, &config, udmaTransferComplete );
    /* Enable dma channel */
    UDMAdrv0->ChannelEnable(CHNL0);

    /* Enable dma controller */
    UDMAdrv0->DMAEnable();

    /* Wait till dma done */
    while(!done);

    /* Compare data buffers */
    ret = memcmp(dst0, src0, sizeof(src0));
    if(ret)
    {
          /*  Prints on hyper-terminal  */
      DEBUGOUT("Data comparison fail\r\n");
    }
    else
    {
          /*  Prints on hyper-terminal  */
      DEBUGOUT("Data comparison success\r\n");
    }
    /* Uninitialize dma */
    UDMAdrv0->Uninitialize();

    while(1);
  }
```

Code Block 15 Micro Direct Memory Access Example

22.3 API Descriptions

22.3.1 UDMAx_Initialize

Source File : UDMA.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\cmsis_driver\

Prototype

```
static int32_t UDMAx_Initialize (UDMA_RESOURCES *udma)
```

Description

Initialize UDMA peripheral.**Parameter**

Parameter	Description
udma	uDAM resource structure variable #UDMA_RESOURCES Structure members and description • UDMA0_Type *reg; /* UDMA register interface */ • IRQn_Type udma_irq_num; /* UDMA Event IRQ Number */ • RSI_UDMA_DESC_T *desc; /* Run-Time control information */

Return values

Return 0: function succeeded

Return -1: function failed

Example

```
extern RSI_DRIVER_UDMA Driver_UDMA0;
static RSI_DRIVER_UDMA *UDMAdrv0 = &Driver_UDMA0;
extern RSI_UDMA_HANDLE_T udmaHandle0;

 /* Initialize dma */
UDMAdrv0->Initialize();
```

22.3.2 UDMAx_Uninitialize

Source File : UDMA.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\cmsis_driver\

Prototype

```
static int32_t UDMAx_Uninitialize (UDMA_RESOURCES *udma)
```

Description

De-initialize GPDMA peripheral

Parameter

Parameter	Description
udma	uDAM resource structure variable #UDMA_RESOURCES Structure members and description • UDMA0_Type *reg; /* UDMA register interface */ • IRQn_Type udma_irq_num; /* UDMA Event IRQ Number */ • RSI_UDMA_DESC_T *desc; /* Run-Time control information */

Return values

Return 0: function succeeded

Return -1: function failed

Example

```
extern RSI_DRIVER_UDMA Driver_UDMA0;
static RSI_DRIVER_UDMA *UDMAdrv0 = &Driver_UDMA0;
extern RSI_UDMA_HANDLE_T udmaHandle0;

/* Uninitialize dma */
UDMAdrv0->Uninitialize();
```

22.3.3 UDMAx_ChannelConfigure

Source File : UDMA.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\cmsis_driver\

Prototype

```
static int32_t UDMAx_ChannelConfigure (UDMA_RESOURCES *udma,
                                       uint8_t              ch,
                                       uint32_t             src_addr,
                                       uint32_t             dest_addr,
                                       uint32_t             size,
                                       RSI_UDMA_CHA_CONFIG_DATA_T control,
                                       RSI_UDMA_CHA_CFG_T   *config,
                                       UDMA_SignalEvent_t   cb_event)
```

Description

Configure GPDMA channel for next transfer.

Parameter

Parameter	Description
udma	uDAM resource structure variable #UDMA_RESOURCES Structure members and description • UDMA0_Type *reg; /* UDMA register interface */ • IRQn_Type udma_irq_num; /* UDMA Event IRQ Number */ • RSI_UDMA_DESC_T *desc; /* Run-Time control information */
ch	Channel number (0..7)
src_addr	Source address
dest_addr	Destination address
size	Amount of data to transfer
control	Channel control
config	Channel configuration structure variable.
cb_event	Channel callback pointer

Return values

Return 0: function succeeded

Return -1: function failed

Example

```
/* call back API description */
/**
 * @brief      UDMA  controller transfer descriptor chain complete callback
 * @param[in]  event dma transfer events
 * @param[in]  ch    dma channel number
 * @return     None
 */
void udmaTransferComplete(uint32_t event, uint8_t ch)
{
  if(event == UDMA_EVENT_XFER_DONE)
  {
    if(ch == 0)
    {
      done = 1;
    }
  }
}

   uint32_t src0[SIZE_BUFFERS];
   uint32_t dst0[SIZE_BUFFERS];


   RSI_UDMA_CHA_CONFIG_DATA_T control;
   RSI_UDMA_CHA_CFG_T config;  config.altStruct = 0;

   config.burstReq  = 1;
   config.channelPrioHigh = 0;
   config.dmaCh = CHNL0;
   config.periAck = 0;
   config.periphReq = 0;
   config.reqMask = 0;

   control.transferType       = UDMA_MODE_AUTO;
   control.nextBurst                     = 0;
   control.totalNumOfDMATrans = (DMA_DESC_LEN-1);
   control.rPower             = ARBSIZE_16;
   control.srcProtCtrl        = 0x000;
   control.dstProtCtrl        = 0x000;
   control.srcSize            = SRC_SIZE_32;
   control.srcInc             = SRC_INC_32;
   control.dstSize            = DST_SIZE_32;
   control.dstInc             = DST_INC_32;

   /* Configure dma channel */
   UDMAdrv0->ChannelConfigure( CHNL0,(uint32_t)src0, (uint32_t)dst0, SIZE_BUFFERS,
                               control, &config, udmaTransferComplete );
```

22.3.4 UDMAx_ChannelEnable

Source File : UDMA.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\cmsis_driver\

Prototype

```
static int32_t UDMAx_ChannelEnable (uint8_t ch,UDMA_RESOURCES *udma)
```

Description

Enable uDMA channel.

Parameter

Parameter	Description
ch	Channel number (0..7)

Parameter	Description
udma	uDAM resource structure variable #UDMA_RESOURCES Structure members and description • UDMA0_Type *reg; /* UDMA register interface */ • IRQn_Type udma_irq_num; /* UDMA Event IRQ Number */ • RSI_UDMA_DESC_T *desc; /* Run-Time control information */

Return values

Return 0: function succeeded

Return -1: function failed

Example

```
/* Enable dma channel */
UDMAdrv0->ChannelEnable(CHNL0);
```

22.3.5 UDMAx_ChannelDisable

Source File : UDMA.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\cmsis_driver\

Prototype

```
static int32_t UDMAx_ChannelDisable (uint8_t ch, UDMA_RESOURCES *udma)
```

Description

Disable uDMA channel

Parameter

Parameter	Description
ch	Channel number (0..7)
udma	uDAM resource structure variable #UDMA_RESOURCES Structure members and description • UDMA0_Type *reg; /* UDMA register interface */ • IRQn_Type udma_irq_num; /* UDMA Event IRQ Number */ • RSI_UDMA_DESC_T *desc; /* Run-Time control information */

Return values

Return 0: function succeeded

Return -1: function failed

Example

```
/* Disable dma channel */
UDMAdrv0->ChannelDisable(CHNL0);
```

22.3.6 UDMAx_ChannelGetCount

Source File : UDMA.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\cmsis_driver\

Prototype

```
static uint32_t UDMAx_ChannelGetCount ( uint8_t ch, RSI_UDMA_CHA_CONFIG_DATA_T control,
                                        RSI_UDMA_CHA_CFG_T  config,
                                        UDMA_RESOURCES *udma )
```

Description

Gets number of remaining transfers of a descriptor

Parameter

Parameter	Description
ch	Channel number (0..7)
control	Control data structure pointer , structure variable as below • transferType : The operating mode of the DMA cycle • nextBurst :Used to force the channel to only respond to burst requests at the tail end of a scatter-gather transfer • totalNumOfDMATrans : total number of DMA transfers that the DMA cycle contains • rPower : Number of DMA transfers can occur before the controller rearbitrates • srcProtCtrl : Performs control operation when the controller reads the source data • dstProtCtrl : Performs control operation when the controller writes the destination data • srcSize : Source data size • srcInc : Source address increment • dstSize : Destination data size • dstInc : Destination address increment
config	uDAM resource structure variable #UDMA_RESOURCES Structure members and description • UDMA0_Type *reg; /* UDMA register interface */ • IRQn_Type udma_irq_num; /* UDMA Event IRQ Number */ • RSI_UDMA_DESC_T *desc; /* Run-Time control information */

Return values

Return remaining data length

Example

```
#define CHANNEL_NUMBER      0
uint32_t src0[SIZE_BUFFERS];
uint32_t dst0[SIZE_BUFFERS];
uint32_t reaming_transfer;

RSI_UDMA_CHA_CONFIG_DATA_T control;
RSI_UDMA_CHA_CFG_T config;  config.altStruct = 0;

config.burstReq  = 1;
config.channelPrioHigh = 0;
config.dmaCh = CHNL0;
config.periAck = 0;
config.periphReq = 0;
config.reqMask = 0;
```

```
control.transferType        = UDMA_MODE_AUTO;
control.nextBurst                     = 0;
control.totalNumOfDMATrans = (DMA_DESC_LEN-1);
control.rPower              = ARBSIZE_16;
control.srcProtCtrl         = 0x000;
control.dstProtCtrl         = 0x000;
control.srcSize             = SRC_SIZE_32;
control.srcInc              = SRC_INC_32;
control.dstSize             = DST_SIZE_32;
control.dstInc              = DST_INC_32;

reaming_transfer = UDMAdrv0->ChannelGetCount(CHANNEL_NUMBER,control);
```

Note

```
before calling this API trigger the uDMA transmission by using "UDMAdrv0->DMAEnable();"
API.
```

22.3.7

23 Pulse Width Modulator (PWM)

23.1 Overview

This section explains how to configure and use the Pulse Width Modulator using Redpine MCU SAPIs.

23.2 Programming sequence

```
#include "rsi_chip.h" /* Redpine_MCU_Vx.y.z\Host_MCU\common\chip\inc */
#include "rsi_rom_crc.h /* Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc
*/

#define  MCPWM_RATE    10000
#define  TICKS         1000
void RSI_MCPWM_PinMux()
{
     RSI_EGPIO_SetPinMux(&EGPIO ,EGPIO_PORT0 , GPIO6 ,EGPIO_PIN_MUX_MODE8);
     RSI_EGPIO_SetPinMux(&EGPIO ,EGPIO_PORT0 , GPIO7 ,EGPIO_PIN_MUX_MODE8);
     RSI_EGPIO_SetPinMux(&EGPIO ,EGPIO_PORT0 , GPIO15 ,EGPIO_PIN_MUX_MODE8);
     RSI_EGPIO_SetPinMux(&EGPIO ,EGPIO_PORT0 , GPIO10 ,EGPIO_PIN_MUX_MODE8);
     RSI_EGPIO_SetPinMux(&EGPIO ,EGPIO_PORT0 , GPIO11 ,EGPIO_PIN_MUX_MODE8);
     RSI_EGPIO_SetPinMux(&EGPIO ,EGPIO_PORT0 , GPIO12 ,EGPIO_PIN_MUX_MODE8);
     RSI_EGPIO_SetPinMux(&EGPIO ,EGPIO_PORT1 , GPIO15 ,EGPIO_PIN_MUX_MODE8);
     RSI_EGPIO_SetPinMux(&EGPIO ,EGPIO_PORT2 , GPIO0 ,EGPIO_PIN_MUX_MODE8);
}

int main()
{
    int loop = 1;
    uint32_t rate = 0;
    uint32_t delay = 0;
    int pwm_out_0, pwm_out_1, duty_p = 0, incr = 1, Led = 0;
    uint16_t time_period = 0;
    uint32_t ticks =0;
    uint32_t temp0=0, temp1=0;

    /* Sets pin mux */
    RSI_MCPWM_PinMux();

    /* MCPWM clock enable */
    RSI_CLK_PeripheralClkEnable( M4CLK, PWM_CLK, ENABLE_STATIC_CLK );

    RSI_MCPWM_BaseTimerSelect(MCPWM, ONE_TIMER_FOR_EACH_CHNL);
    RSI_MCPWM_SetOutputPolarity(MCPWM,1 ,1);

    /* Set PWM output mode */
    RSI_MCPWM_SetBaseTimerMode(MCPWM, TMR_FREE_RUN_MODE, PWM_CHNL_0);
    RSI_MCPWM_SetBaseTimerMode(MCPWM, TMR_FREE_RUN_MODE, PWM_CHNL_1);

    /* Set time period */
    rate = SystemCoreClock/MCPWM_RATE;
    RSI_MCPWM_SetTimePeriod( MCPWM,PWM_CHNL_0 ,rate ,COUNTER_INIT_VAL);
    RSI_MCPWM_SetTimePeriod( MCPWM,PWM_CHNL_1 ,rate ,COUNTER_INIT_VAL);

    RSI_MCPWM_GetTimePeriod(MCPWM,PWM_CHNL_1,&time_period);
    ticks = time_period/2;

    /* Set Duty cycle value for channel 0 and channel 1*/
    RSI_MCPWM_SetDutyCycle(MCPWM, 0 ,PWM_CHNL_0);       /* LED */
    RSI_MCPWM_SetDutyCycle(MCPWM, ticks, PWM_CHNL_1);   /* OUT */

    /* Start PWM */
    RSI_MCPWM_Start( MCPWM , PWM_CHNL_0);
    RSI_MCPWM_Start( MCPWM , PWM_CHNL_1);

    /* Enable SysTick Timer */
    SysTick_Config(SystemCoreClock / TICKS);
```

```
    while (loop)
     {
        delay++;
        if (delay >= 20)
        {
            duty_p = duty_p + incr;
            if (duty_p < 0)
        {
                duty_p = 0;
                incr = 1;
            }
            if (duty_p > 100)
        {
                duty_p = 100;
                incr = -1;
                Led = 1 - Led;
         }
            if (Led)
        {
                pwm_out_0 = duty_p;
                pwm_out_1 = 100;
        }
            else
        {
                pwm_out_0 = 100;
                pwm_out_1 = duty_p;
        }

       if(pwm_out_0 == 100)
            {
                RSI_MCPWM_OutputOverrideEnable(MCPWM, PWM_OUTPUT_L0);
                RSI_MCPWM_OverrideValueSet(MCPWM, PWM_OUTPUT_L0, 1);
            }
          else
            {
                RSI_MCPWM_OutputOverrideDisable(MCPWM, PWM_OUTPUT_L0);
                /* Set duty cycle */
                (MCPWM, MCPWM_PercentageToTicks(pwm_out_0,PWM_CHNL_0),PWM_CHNL_0);   /* LED
*/
            }

          if(pwm_out_1 == 100)
            {
                RSI_MCPWM_OutputOverrideEnable(MCPWM, PWM_OUTPUT_L1);
                RSI_MCPWM_OverrideValueSet(MCPWM, PWM_OUTPUT_L1, 1);
            }
           else
            {
                RSI_MCPWM_OutputOverrideDisable(MCPWM, PWM_OUTPUT_L1);
                /* Set duty cycle */
                (MCPWM, MCPWM_PercentageToTicks(pwm_out_1,PWM_CHNL_1),PWM_CHNL_1);   /* LED
*/
            }
              delay = 0;
            }
              __WFI();
  }
}
```

Code Block 16 Pulse Width Modulator Example

23.3 API Descriptions

23.3.1 RSI_MCPWM_BaseTimerSelect

Source File : rsi_pwm.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_MCPWM_BaseTimerSelect( RSI_MCPWM_T *pMCPWM, uint8_t baseTime)
```

Description

This API is used to select number of base timers as four base timers for four channels or one base timer for all channels of MCPWM.

Parameter

Parameter	Description
pMCPWM	This is the pointer to the MCPWM instance register area
baseTime	if 0, one base timer for each channel If 1, only one base timer for all channels

Return values

None

Example

```
/* base timer select */
RSI_MCPWM_BaseTimerSelect( MCPWM, 0);
```

23.3.2 RSI_MCPWM_SetOutputPolarity

Source File : rsi_rom_pwm.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void  RSI_MCPWM_SetOutputPolarity( RSI_MCPWM_T *pMCPWM,boolean_t
polL,boolean_t polH )
```

Description

This API is used to set output mode for the MCPWM.

Parameter

Parameter	Description
pMCPWM	This is the pointer to the MCPWM instance register area
polL	This is the ouput polarity for low side (L3, L2, L1, L0)
polH	This is the ouput polarity for high side (H3, H2, H1, H0)

Return values

None

Example

```
/* Set PWM output polarity*/
RSI_MCPWM_SetOutputPolarity(MCPWM,1 ,1);
```

23.3.3 RSI_MCPWM_InterruptHandler

Source File : rsi_rom_pwm.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void  RSI_MCPWM_InterruptHandler( RSI_MCPWM_T *pMCPWM,RSI_MCPWM_CALLBACK_T
*pCallBack )
```

Description

Handles all interrupt flags of MCPWM..

Parameter

Parameter	Description
pMCPWM	This is the pointer to the MCPWM instance register area
pCallBack	Pointer to the structure of type RSI_CALLBACK

Return values

None

Example

```
void callback()
{
  /*call back api */
}
/* Handle the PWM interrupt */
RSI_MCPWM_InterruptHandler(MCPWM,&callback);
```

23.3.4 RSI_MCPWM_SetOutputMode

Source File : rsi_rom_pwm.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_MCPWM_SetOutputMode( RSI_MCPWM_T *pMCPWM,boolean_t mode,uint8_t
chnlNum )
```

Description

This API is used to set output mode for the MCPWM.

Parameter

Parameter	Description
pMCPWM	This is the pointer to the MCPWM instance register area
mode	If one then complementary output mode and if zero then independent output mode.

Parameter	Description
chnlNum	This is the channel number (0 to 3)

Return values

ERROR_PWM_INVALID_ARG : If selecetd dead time counter is invalid

RSI_OK : If process is done successfully

Example

```
/* Set PWM output mode for channel 0 */
RSI_MCPWM_SetOutputMode(MCPWM, 1,0);
```

23.3.5 RSI_MCPWM_SetBaseTimerMode

Source File : rsi_rom_pwm.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_MCPWM_SetBaseTimerMode( RSI_MCPWM_T *pMCPWM,uint8_t mode,uint8_t
chnlNum )
```

Description

This API is used to set the mode of base timer for required channel.

Parameter

Parameter	Description
pMCPWM	This is the pointer to the MCPWM instance register area
pwmOutput	This is the base timer operating mode as mentioned below : • 000 : free running mode • 001 : single event mode • 010 : down count mode • 100 : up/down mode • 101 : up/dowm mode with interrupts for double PWM updates
chnlNum	This is the channel number (0 to 3)

Return values

ERROR_PWM_INVALID_ARG : If selecetd dead time counter is invalid

RSI_OK : If process is done successfully

Example

```
/* Set PWM output mode for channel 0 */
RSI_MCPWM_SetBaseTimerMode(MCPWM, TMR_FREE_RUN_MODE, PWM_CHNL_0);
```

23.3.6 RSI_MCPWM_SetTimePeriod

Source File : rsi_rom_pwm.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_MCPWM_SetOutputMode( RSI_MCPWM_T *pMCPWM, boolean_t mode,uint8_t
chnlNum )
```

Description

This API is used to set time period and counter initial,value for the required MCPWM channel.

Parameter

Parameter	Description
pMCPWM	This is the pointer to the MCPWM instance register area
chnlNum	This is the channel number(0 to 3)
period	This is the time period value.
initVal	This is to update the base time counter initial value

Return values

ERROR_PWM_INVALID_CHNLNUM : If channel number is invalid

RSI_OK : If process is done successfully

Example

```
/* counter initial value */
#define COUNTER_INIT_VAL 0
/* define frequency */
#define  freq    10000
/* Get period value by using SystemCoreClock and freq */
rate = SystemCoreClock/freq;
/* Set time period */
RSI_MCPWM_SetTimePeriod( MCPWM,PWM_CHNL_0 ,rate ,COUNTER_INIT_VAL);
```

23.3.7 RSI_MCPWM_GetTimePeriod

Source File : rsi_rom_pwm.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE uint16_t RSI_MCPWM_GetTimePeriod( RSI_MCPWM_T *pMCPWM,uint8_t
chnlNum,uint16_t *timePeriod)
```

Description

Get time period for required channel.

Parameter

Parameter	Description
pMCPWM	This is the pointer to the MCPWM instance register area
chnlNum	This is the channel number(0-3).
timePeriod	This is the pointer to read time period.

Return values

ERROR_PWM_INVALID_CHNLNUM : If channel number is invalid

RSI_OK : If process is done successfully

Example

```
/* declare channel number and time period variable */
uint8_t chnl_num=0;
uint16_t time_period = 0;
/* Get time period */
RSI_MCPWM_GetTimePeriod(MCPWM,chnl_num,&time_period);
```

23.3.8 RSI_MCPWM_SetDutyCycle

Source File : rsi_pwm.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE error_t RSI_MCPWM_SetDutyCycle( RSI_MCPWM_T *pMCPWM, uint16_t
dutyCycle,uint8_t chnlNum )
```

Description

This API is used to set duty cycle for the required MCPWM channel.

Parameter

Parameter	Description
pMCPWM	This is the pointer to the MCPWM instance register area
dutyCycle	This is the duty cycle value
chnlNum	This is the channel number (0 to 3)

Return values

None

Example

```
/* define duty cycle */
#define    TMR0_DUTYCYCLE  0xA80
/* set duty cycle */
RSI_MCPWM_SetDutyCycle( MCPWM, TMR0_DUTYCYCLE ,0);
```

23.3.9 RSI_MCPWM_InterruptEnable

Source File : rsi_pwm.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_MCPWM_InterruptEnable( RSI_MCPWM_T *pMCPWM, uint16_t flag )
```

Description

This API is used to enable the interrupts of MCPWM.

Parameter

Parameter	Description

Parameter	Description
pMCPWM	This is the pointer to the MCPWM instance register area
clrFlag	This can be logical OR of the below parameters • RISE_PWM_TIME_PERIOD_MATCH_CH0 : Time base interrupt for 0th channel without considering postscaler if bit is 1 then pwm time period match interrupt for 0th channel will be cleared,if zero then no effect. • PWM_TIME_PRD_MATCH_INTR_CH0 : Time base interrupt for 0th channel which considering postscaler if bit is 1 then pwm time period match interrupt for 0th channel will be cleared if zero then no effect. • FLT_A_INTR : Fault A pin interrupt if bit is one then pwm faultA interrupt will be cleared if zero then no effect. • FLT_B_INTR : Fault B pin interrupt if bit is one pwm faultB interrupt will be cleared if zero then no effect. • RISE_PWM_TIME_PERIOD_MATCH_CH1 : Time base interrupt for 1st channel without considering postscaler if bit is one pwm time period match interrupt for 1st channel will be cleared if bit is zero no effect • PWM_TIME_PRD_MATCH_INTR_CH1 : Time base interrupt for 1st channel which considering postscaler if bit is 1 then pwm time period match interrupt for 1st channel will be cleared if zero then no effect. • RISE_PWM_TIME_PERIOD_MATCH_CH2 : Time base interrupt for 2nd channel without considering postscaler if bit is one pwm time period match interrupt for 2nd channel will be cleared if bit is zero no effect • PWM_TIME_PRD_MATCH_INTR_CH2 : Time base interrupt for 2nd channel which considering postscaler if bit is one pwm time period match interrupt for 2nd channel will be cleared if bit is zero no effect • RISE_PWM_TIME_PERIOD_MATCH_CH3 : Time base interrupt for 3rd channel without considering postscaler if bit is one pwm time period match interrupt for 3rd channel will be cleared if bit is zero no effect • PWM_TIME_PRD_MATCH_INTR_CH3 : Time base interrupt for 3rd channel which considering postscaler if bit is one pwm time period match interrupt for 3rd channel will be cleared if bit is zero no effect

Return values

None

Example

```
/* enable interrupt */
RSI_MCPWM_InterruptEnable( MCPWM, RSI_MCPWM_EVENT_TIME_PERIOD_MATCH_CH0 );
```

23.3.10 RSI_MCPWM_Start

Source File : rsi_rom_pwm.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_MCPWM_Start( RSI_MCPWM_T *pMCPWM, uint8_t chnlNum )
```

Description

This API is used to start the MCPWM operation for required channel.

Parameter

Parameter	Description
pMCPWM	This is the pointer to the MCPWM instance register area
chnlNum	This is the channel number (0 to 3)

Return values

ERROR_PWM_INVALID_CHNLNUM : If channel number is invalid

RSI_OK : If process is done successfully

Example

```
/* Start MCPWM  */
RSI_MCPWM_Start(MCPWM,0);
```

23.3.11 RSI_MCPWM_OutputOverrideEnable

Source File : rsi_pwm.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE error_t RSI_MCPWM_OutputOverrideEnable( RSI_MCPWM_T *pMCPWM,uint8_t
pwmOutput )
```

Description

This API is used to enable the output override operation of MCPWM.

Parameter

Parameter	Description
pMCPWM	Pointer to the MCPWM instance register area
pwmOutput	Pwm output over ride,possible values are as below • 0 : L0 • 1 : L1 • 2 : L2 • 3 : L3 • 4 : H0 • 5 : H1 • 6 : H2 • 7 : H3
chnlNum	Channel number(0 to 3)

Return values

ERROR_PWM_INVALID_CHNLNUM : If channel is invalid.

RSI_OK : If process is done successfully.

Example

```
/* over ride enable */
RSI_MCPWM_OutputOverrideEnable( MCPWM, PWM_OUTPUT_L0);
```

23.3.12 RSI_MCPWM_OverrideValueSet

Source File : rsi_pwm.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
 STATIC INLINE error_t RSI_MCPWM_OverrideValueSet( RSI_MCPWM_T *pMCPWM,uint8_t
pwmOutput,uint8_t value )
```

Description

This API is used to disable the output override operation of MCPWM.

Parameter

Parameter	Description
pMCPWM	This is the pointer to the MCPWM instance register area
pwmOutput	This is the Pwm output over ride, the possible values are as below: • 0 : L0 • 1 : L1 • 2 : L2 • 3 : L3 • 4 : H0 • 5 : H1 • 6 : H2 • 7 : H3
value	This override value 1 or 0

Return values

ERROR_PWM_INVALID_CHNLNUM : If channel is invalid.

RSI_OK : If process is done successfully.

Example

```
/* over ride value set */
RSI_MCPWM_OverrideValueSet( MCPWM, PWM_OUTPUT_L0,1);
```

23.3.13 RSI_MCPWM_OverrideValueReSet

Source File : rsi_pwm.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
 STATIC INLINE error_t RSI_MCPWM_OverrideValueReSet( RSI_MCPWM_T *pMCPWM,uint8_t
pwmOutput,uint8_t value )
```

Description

This API is used to reset override value for the required output of MCPWM.

Parameter

Parameter	Description
pMCPWM	This is the pointer to the MCPWM instance register area
pwmOutput	This is the Pwm output over ride, the possible values are as below: • 0 : L0 • 1 : L1 • 2 : L2 • 3 : L3 • 4 : H0 • 5 : H1 • 6 : H2 • 7 : H3
value	This override value 1 or 0

Return values

ERROR_PWM_INVALID_CHNLNUM : If channel is invalid.

RSI_OK : If process is done successfully.

Example

```
/* over ride value set */
RSI_MCPWM_OverrideValueReSet( MCPWM, PWM_OUTPUT_L0,1);
```

23.3.14 RSI_MCPWM_OutputOverrideDisable

Source File : rsi_pwm.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE error_t RSI_MCPWM_OutputOverrideDisable( RSI_MCPWM_T *pMCPWM,uint8_t
pwmOutput )
```

Description

This API is used to disable the output override operation of MCPWM.

Parameter

Parameter	Description
pMCPWM	This is the pointer to the MCPWM instance register area
pwmOutput	This is the pwm output over ride, the possible values are as below:

Parameter	Description
	• 0 : L0 • 1 : L1 • 2 : L2 • 3 : L3 • 4 : H0 • 5 : H1 • 6 : H2 • 7 : H3
chnlNum	This is the channel number (0 to 3)

Return values

ERROR_PWM_INVALID_CHNLNUM : If channel is invalid.

RSI_OK : If process is done successfully.

Example

```
/* over ride enable */
RSI_MCPWM_OutputOverrideDisable( MCPWM, PWM_OUTPUT_L0);
```

23.3.15 RSI_MCPWM_SpecialEventTriggerConfig

Source File : rsi_rom_pwm.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void  RSI_MCPWM_SpecialEventTriggerConfig( RSI_MCPWM_T *pMCPWM,boolean_t
svtDir,
                                                         RSI_MCPWM_SVT_CONFIG_T
*pMCPWMSVTConfig)
```

Description

This API is used to configure special event trigger generation for required MCPWM channel which allows the A/D converter to be synchronized to the PWM time base.

Parameter

Parameter	Description
pMCPWM	This is the pointer to the MCPWM instance register area
svtDir	This is the special event trigger for time base direction • 1 : A special event trigger will occur when PWM time base is counting down • 0 : A special event trigger will occur when PWM time base is counting up
pMCPWMSVTConfig	This is the pointer to the structure of type RSI_MCPWM_SVT_CONFIG_T • svtPostscalar : PWM special event trigger output postscale value 0000 means 1:1 post scale to 1111 means 1:16 post scale • svtCompareVal : Compare value to generate special event trigger.

Parameter	Description
	• svtChannel : Out of four channels, we can generate special event for one channel

Return values

None

Example

```
/* declaration  SVT configuration structure variable */
RSI_MCPWM_SVT_CONFIG_T vsMCPWMSVTConfig;

/* Initialization of SVT structure */
vsMCPWMSVTConfig.svtPostscalar = TIME_PERIOD_POSTSCALE_1_1;
vsMCPWMSVTConfig.svtCompareVal = 0x100;
vsMCPWMSVTConfig.svtChannel = PWM_CHNL_0;

/* special trigger */
RSI_MCPWM_SpecialEventTriggerConfig(MCPWM, COUNTUP, &vsMCPWMSVTConfig);
```

23.3.16 RSI_MCPWM_DeadTimeValueSet

Source File : rsi_rom_pwm.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_MCPWM_DeadTimeValueSet( RSI_MCPWM_T *pMCPWM
,RSI_MCPWM_DT_CONFIG_T *pMCPWMDeadTimeConfig,uint8_t chnlNum)
```

Description

This API is used to set dead time value to be inserted at rise edge or fall edge for required MCPWM channel.

Parameter

Parameter	Description
pMCPWM	This is the pointer to the MCPWM instance register area

pMCPWMDeadTimeConfig	This is the pointer to the structure of type RSI_MCPWM_DT_CONFIG_T
	• counterSelect : Selects coutner A/B for deadtime insertion.
	• preScaleA : Dead time prescale selection bits for unit A is below like
	  ○ 00 : clock period for dead time unit A is 1x input clock period
	  ○ 01 : clock period for dead time unit A is 2x input clock period
	  ○ 10 : clock period for dead time unit A is 4x input clock period
	  ○ 11 : clock period for dead time unit A is 8x input clock period
	• preScaleB : Dead time prescale selection bits for unit B is below like
	  ○ 00 : clock period for dead time unit B is 1x input clock period
	  ○ 01 : clock period for dead time unit B is 2x input clock period
	  ○ 10 : clock period for dead time unit B is 4x input clock period
	  ○ 11 : clock period for dead time unit B is 8x input clock period
	• DeadTimeA : Dead time A value to load into deadtime counter A
	• DeadTimeB : Dead time B value to load into deadtime counter B

Parameter	Description
chnlNum	This is the channel number (0 to 3)

Return values

ERROR_PWM_INVALID_ARG : If selecetd dead time counter is invalid

RSI_OK : If process is done successfully

Example

```
/* declare dead time structure */
RSI_MCPWM_DT_CONFIG_T vsMCPWMDeadTimeConfig;

/* Initialization of dead time structure */
vsMCPWMDeadTimeConfig.deadTimeA = 0x10;
vsMCPWMDeadTimeConfig.preScaleB = 0x00;
vsMCPWMDeadTimeConfig.deadTimeB = 0x20;
vsMCPWMDeadTimeConfig.preScaleA = 0x00;

/* Sets Deadtime value to be inserted in anactive state */
RSI_MCPWM_DeadTimeValueSet( MCPWM , &vsMCPWMDeadTimeConfig,PWM_CHNL_0);
```

23.3.17 RSI_MCPWM_ChannelReset

Source File : rsi_rom_pwm.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_MCPWM_ChannelReset( RSI_MCPWM_T *pMCPWM,uint8_t chnlNum )
```

Description

This API is used to reset the required channel of MCPWM.

Parameter

Parameter	Description
pMCPWM	This is the pointer to the MCPWM instance register area
chnlNum	This is the channel number(0 to 3)

Return values

ERROR_PWM_INVALID_ARG : If selecetd dead time counter is invalid

RSI_OK : If process is done successfully

Example

```
/* reset channel */
RSI_MCPWM_ChannelReset(MCPWM,0);
```

23.3.18 RSI_MCPWM_CounterReset

Source File : rsi_rom_pwm.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_MCPWM_CounterReset( RSI_MCPWM_T *pMCPWM,uint8_t chnlNum )
```

Description

This API is used to reset the counter from required channel of MCPWM.

Parameter

Parameter	Description
pMCPWM	This is the pointer to the MCPWM instance register area
chnlNum	This is the channel number(0 to 3)

Return values

ERROR_PWM_INVALID_ARG : If selecetd dead time counter is invalid

RSI_OK : If process is done successfully

Example

```
/* reset counter */
RSI_MCPWM_CounterReset(MCPWM,0);
```

23.3.19 RSI_MCPWM_PeriodControlConfig

Source File : rsi_rom_pwm.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_MCPWM_PeriodControlConfig( RSI_MCPWM_T *pMCPWM,uint32_t
postScale,uint32_t preScale,uint8_t chnlNum )
```

Description

This API is used to set base time period control for the required MCPWM channel.

Parameter

Parameter	Description
pMCPWM	This is the pointer to the MCPWM instance register area
postScale	This is the time base output postscale bits ,possible values are as below • 0000 means 1:1 • 0001 means 1:2 To • 1111 means 1:16
preScale	This is the base timer input clock prescale select value,possible values are as below • 000 : 1x input clock period • 010 : 4x input clock period • 011 : 8x input clock period • 100 : 16x input clock period • 101 : 32x input clock period

Parameter	Description
	• 110 : 64x input clock period
chnlNum	This is the channel number (0 to 3)

Return values

ERROR_PWM_INVALID_ARG : If selecetd dead time counter is invalid

RSI_OK : If process is done successfully

Example

```
/* Set the prescale and postscale values for channel 1 */
RSI_MCPWM_PeriodControlConfig( MCPWM, 0,0,2);
```

23.3.20 RSI_MCPWM_FaultAValueSet

Source File : rsi_rom_pwm.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_MCPWM_FaultAValueSet( RSI_MCPWM_T *pMCPWM,uint8_t pwmOutput,
                                                uint8_t value)
```

Description

This API is used to set fault A pin output value to be overridden when fault condition occurs..

Parameter

Parameter	Description
pMCPWM	Pointer to the MCPWM instance register area
pwmOutput	PWM outputs are as below • 0 : L0 • 1 : L1 • 2 : L2 • 3 : L3 • 4 : H0 • 5 : H1 • 6 : H2 • 7 : H3
value	Fault input A PWM output override value below • 1 : PWM output pin is driven active on an external fault input A event • 0 : PWM output pin is driven inactive on an external fault input A event

Return values

ERROR_PWM_INVALID_ARG : If selecetd dead time counter is invalid

RSI_OK : If process is done successfully

Example

```
/* Set fault A value */
RSI_MCPWM_FaultAValueSet(MCPWM, PWM_OUTPUT_L0,1)
```

23.3.21 RSI_MCPWM_FaultBValueSet

Source File : rsi_rom_pwm.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_MCPWM_FaultBValueSet( RSI_MCPWM_T *pMCPWM,uint8_t
pwmOutput,uint8_t value)
```

Description

This API is used to set fault B pin output value to be overridden when fault condition occurs.

Parameter

Parameter	Description
pMCPWM	This is the pointer to the MCPWM instance register area
pwmOutput	The PWM outputs are as below • 0 : L0 • 1 : L1 • 2 : L2 • 3 : L3 • 4 : H0 • 5 : H1 • 6 : H2 • 7 : H3
value	The fault input B PWM output override the value as below • 1 : PWM output pin is driven active on an external fault input B event • 0 : PWM output pin is driven inactive on an external fault input B event

Return values

ERROR_PWM_INVALID_ARG : If selecetd dead time counter is invalid

RSI_OK : If process is done successfully

Example

```
/* Set fault B value */
RSI_MCPWM_FaultAValueSet(MCPWM, PWM_OUTPUT_L0,1)
```

23.3.22 RSI_MCPWM_ReadCounter

Source File : rsi_pwm.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE error_t RSI_MCPWM_ReadCounter( RSI_MCPWM_T *pMCPWM,uint16_t
*counterVal,uint8_t chnlNum)
```

Description

This API is used to read the counter current value.

Parameter

Parameter	Description
pMCPWM	This is the pointer to the MCPWM instance register area
counterVal	This is the counter value
chnlNum	This is the channel number(0-3).

Return values

ERROR_PWM_INVALID_CHNLNUM : If channel number is invalid

RSI_OK : If process is done successfully

Example

```
/* declare counter read variable*/
uint16_t CounterRead;
/* read counter value */
RSI_MCPWM_ReadCounter( MCPWM, &CounterRead , PWM_CHNL_0);
```

23.3.23 RSI_MCPWM_GetCounterDir

Source File : rsi_pwm.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE error_t RSI_MCPWM_GetCounterDir( RSI_MCPWM_T *pMCPWM,uint8_t
*counterDir,uint8_t chnlNum )
```

Description

This API is used to get time period counter direction status of required MCPWM channel.

Parameter

Parameter	Description
pMCPWM	This is the pointer to the MCPWM instance register area
counterDir	This is the counter direction as up/down counter.
chnlNum	This is the channel number (0-3).

Return values

ERROR_PWM_INVALID_CHNLNUM : If channel number is invalid

RSI_OK : If process is done successfully

Example

```
/* declare counter direction variable*/
uint8_t Dir;
```

```
/* get counter direction */
RSI_MCPWM_GetCounterDir( MCPWM, &Dir, PWM_CHNL_0);
```

23.3.24 RSI_MCPWM_DeadTimeEnable

Source File : rsi_pwm.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_MCPWM_DeadTimeEnable( RSI_MCPWM_T *pMCPWM,uint32_t flag )
```

Description

This enables dead time insertion at rise edge or fall edge of any four channels.

Parameter

Parameter	Description
pMCPWM	This is the pointer to the MCPWM instance register area
flag	This can be ORing of below values : • DT_EN_CH0 • DT_EN_CH1 • DT_EN_CH2 • DT_EN_CH3

Return values

None

Example

```
/* Deadtime insertion enable */
RSI_MCPWM_DeadTimeEnable( MCPWM ,PWM_CHNL_2);
```

23.3.25 RSI_MCPWM_DeadTimeEnable

Source File : rsi_pwm.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_MCPWM_DeadTimeEnable( RSI_MCPWM_T *pMCPWM,uint32_t flag )
```

Description

This enables dead time insertion at rise edge or fall edge of any four channels.

Parameter

Parameter	Description
pMCPWM	This is the pointer to the MCPWM instance register area
flag	This can be ORing of below values :

Parameter	Description
	• DT_EN_CH0 • DT_EN_CH1 • DT_EN_CH2 • DT_EN_CH3

Return values

None

Example

```
/* Deadtime insertion enable */
RSI_MCPWM_DeadTimeEnable( MCPWM ,DT_EN_CH0);
```

23.3.26 RSI_MCPWM_DeadTimeDisable

Source File : rsi_pwm.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_MCPWM_DeadTimeDisable( RSI_MCPWM_T *pMCPWM,uint32_t flag )
```

Description

This API is used to disable the dead time mode for the specified MCPWM generator.

Parameter

Parameter	Description
pMCPWM	This is the pointer to the MCPWM instance register area
flag	This can be ORing of below values : • DT_EN_CH0 • DT_EN_CH1 • DT_EN_CH2 • DT_EN_CH3

Return values

None

Example

```
/* Deadtime insertion enable */
RSI_MCPWM_DeadTimeDisable( MCPWM ,DT_EN_CH0);
```

23.3.27 RSI_MCPWM_DutyCycleControlSet

Source File : rsi_pwm.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE error_t RSI_MCPWM_DutyCycleControlSet( RSI_MCPWM_T *pMCPWM,uint32_t
value,uint8_t chnlNum )
```

Description

This API is used to set duty cycle control parameters for the required MCPWM channel.

Parameter

Parameter	Description
pMCPWM	This is the pointer to the MCPWM instance register area
value	This can be logical OR of below parameters • : Enable to update the duty cycle immediately • : Duty cycle register updation disable
chnlNum	This is the channel number (0 to 3)

Return values

ERROR_PWM_INVALID_CHNLNUM : If channel is invalid.

RSI_OK : If process is done successfully.

Example

```
/* Control parameter for setting a duty cycle */
RSI_MCPWM_DutyCycleControlSet( MCPWM, DUTYCYCLE_UPDATE_DIS,PWM_CHNL_2);
```

23.3.28 RSI_MCPWM_DutyCycleControlReset

Source File : rsi_pwm.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE error_t RSI_MCPWM_DutyCycleControlReset( RSI_MCPWM_T *pMCPWM,uint32_t
value,uint8_t chnlNum )
```

Description

This API is used to reset the duty cycle control parameters of required MCPWM channel.

Parameter

Parameter	Description
pMCPWM	This is the pointer to the MCPWM instance register area
value	This can be logical OR of below parameters • : Enable to update the duty cycle immediately • : Duty cycle register updation disable
chnlNum	This is the channel number(0 to 3)

Return values

ERROR_PWM_INVALID_CHNLNUM : If channel is invalid.

RSI_OK : If process is done successfully.

Example

```
/* reset control parameter in duty cycle */
RSI_MCPWM_DutyCycleControlReset(
MCPWM,DUTYCYCLE_UPDATE_EN0|DUTYCYCLE_UPDATE_EN1,PWM_CHNL_0);
```

23.3.29 RSI_MCPWM_OverrideControlSet

Source File : rsi_pwm.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_MCPWM_OverrideControlSet( RSI_MCPWM_T *pMCPWM, uint32_t value )
```

Description

This API is used to set override value for the required output of MCPWM.

Parameter

Parameter	Description
pMCPWM	This is the pointer to the MCPWM instance register area
value	if value is 1, the Output override is synced with pwm time period depending on operating mode, if 0 then no effect

Return values

None

Example

```
/* Output override is synced with pwm time period */
RSI_MCPWM_OverrideControlSet( MCPWM, OVERRIDE_SYNC_EN);
```

23.3.30 RSI_MCPWM_OverrideControlReSet

Source File : rsi_pwm.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_MCPWM_OverrideControlReSet( RSI_MCPWM_T *pMCPWM, uint32_t value )
```

Description

This API is used to reset override value for the required output of MCPWM.

Parameter

Parameter	Description
pMCPWM	This is the pointer to the MCPWM instance register area
value	if value is 1 then Output override is synced with pwm time period depending on operating mode, if 0 then no effect

Return values

None

Example

```
/* Output override is synced with pwm time period */
RSI_MCPWM_OverrideControlReSet( MCPWM, OVERRIDE_SYNC_EN);
```

23.3.31 RSI_MCPWM_OverrideControlSet

Source File : rsi_pwm.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_MCPWM_OverrideControlSet( RSI_MCPWM_T *pMCPWM, uint32_t value )
```

Description

This API is used to set override value for the required output of MCPWM.

Parameter

Parameter	Description
pMCPWM	This is the pointer to the MCPWM instance register area
value	if value is 1, Output override is synced with pwm time period depending on operating mode, if 0 then no effect

Return values

None

Example

```
/* Output override is synced with pwm time period */
RSI_MCPWM_OverrideControlSet( MCPWM, OVERRIDE_SYNC_EN);
```

23.3.32 RSI_MCPWM_FaultControlSet

Source File : rsi_pwm.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_MCPWM_FaultControlSet( RSI_MCPWM_T *pMCPWM, uint32_t value )
```

Description

This API is used to set output fault override control parameters for required PWM output

Parameter

Parameter	Description
pMCPWM	This is the pointer to the MCPWM instance register area
value	This can be logical OR of below parameters • FLT_A_MODE : if bit one then cycle by cycle by mode and zero then latched mode

Parameter	Description
	• FLT_B_MODE : if bit one then cycle by cycle by mode and zero then latched mode • OP_POLARITY_H Ouput polarity for high (H3, H2, H1, H0) side signals. if bit 0 then in active low mode and 1 then active high mode. • OP_POLARITY_L Ouput polarity for low (L3, L2, L1, L0) side signals. if bit 0 then in active low mode and 1 then active high mode. • FLT_A_ENABLE : enable fault A • FLT_B_ENABLE : enable fault B • COMPLEMENT_MODE : PWM I/O pair mode if bit is 1 then PWM I/O pin pair is in the complementary output mode if bit is 0 then PWM I/O pin pair is in the independent output mode

Return values

None

Example

```
/* Set fault override control parameter */
RSI_MCPWM_FaultControlSet(MCPWM,  FLTB_ENABLE |FLTB_MODE);
```

23.3.33 RSI_MCPWM_FaultControlReSet

Source File : rsi_pwm.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_MCPWM_FaultControlReSet( RSI_MCPWM_T *pMCPWM,uint32_t value )
```

Description

This API is used to reset output fault override control parameters for required PWM output.

Parameter

Parameter	Description
pMCPWM	This is the pointer to the MCPWM instance register area
value	This can be logical OR of below parameters : • FLT_A_MODE : if bit one then cycle by cycle by mode and zero then latched mode • FLT_B_MODE : if bit one then cycle by cycle by mode and zero then latched mode • OP_POLARITY_H Ouput polarity for high (H3, H2, H1, H0) side signals. if bit 0 then in active low mode and 1 then active high mode. • OP_POLARITY_L Ouput polarity for low (L3, L2, L1, L0) side signals. if bit 0 then in active low mode and 1 then active high mode. • FLT_A_ENABLE : enable fault A • FLT_B_ENABLE : enable fault B • COMPLEMENT_MODE : PWM I/O pair mode if bit is 1 then PWM I/O pin pair is in the complementary output mode if bit is 0 then PWM I/O pin pair is in the independent output mode

Return values

None

Example

```
/* reset fault override control parameter */
RSI_MCPWM_FaultControlReSet(MCPWM, FLTA_ENABLE)
```

23.3.34 RSI_MCPWM_SpecialEventTriggerEnable

Source File : rsi_pwm.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_MCPWM_SpecialEventTriggerEnable(RSI_MCPWM_T *pMCPWM )
```

Description

This API is used to enable generation of special event trigger for required channel of MCPWM .

Parameter

Parameter	Description
pMCPWM	This is the pointer to the MCPWM instance register area

Return values

None

Example

```
/* enable SVT */
RSI_MCPWM_SpecialEventTriggerEnable(MCPWM);
```

23.3.35 RSI_MCPWM_SpecialEventTriggerDisable

Source File : rsi_pwm.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_MCPWM_SpecialEventTriggerDisable(RSI_MCPWM_T *pMCPWM )
```

Description

This API is used to disable generation of special event trigger for required channel of MCPWM .

Parameter

Parameter	Description
pMCPWM	This is the pointer to the MCPWM instance register area

Return values

None

Example

```
/* disable SVT */
```

```
RSI_MCPWM_SpecialEventTriggerDisable(MCPWM);
```

23.3.36 RSI_MCPWM_DeadTimeControlSet

Source File : rsi_pwm.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_MCPWM_DeadTimeControlSet( RSI_MCPWM_T *pMCPWM, uint32_t value )
```

Description

This API is used to set dead time control parameters for the reqired channel.

Parameter

Parameter	Description
pMCPWM	This is the pointer to the MCPWM instance register area
value	This can be logical OR of below parameters • DEADTIME_SELECT_ACTIVE: Deadtime select bits for PWM going active Possible values are as below if bit zero then use counter A , if one then use counter B • DEADTIME_SELECT_INACTIVE: Deadtime select bits for PWM going inactive Possible values are as below if bit zero then use counter A , if one then use counter B

Return values

None

Example

```
/* disable SVT */
RSI_MCPWM_DeadTimeControlSet(MCPWM,DEADTIME_SELECT_ACTIVE);
```

23.3.37 RSI_MCPWM_DeadTimeControlReSet

Source File : rsi_pwm.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_MCPWM_DeadTimeControlReSet( RSI_MCPWM_T *pMCPWM, uint32_t value )
```

Description

This API is used to reset dead time control for the MCPWM.

Parameter

Parameter	Description
pMCPWM	This is the pointer to the MCPWM instance register area
value	This can be logical OR of below parameters

Parameter	Description
	• DEADTIME_SELECT_ACTIVE: Deadtime select bits for PWM going active Possible values are as below if bit zero then use counter A , if one then use counter B • DEADTIME_SELECT_INACTIVE: Deadtime select bits for PWM going inactive Possible values are as below DEADTIME_SELECT_INACTIVE if bit zero then use counter A , if one then use counter B

Return values

None

Example

```
/* disable SVT */
RSI_MCPWM_DeadTimeControlReSet(MCPWM,DEADTIME_SELECT_ACTIVE);
```

23.3.38 RSI_PWM_GetInterruptStatus

Source File : rsi_pwm.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE uint16_t RSI_PWM_GetInterruptStatus( RSI_MCPWM_T *pMCPWM, uint16_t flag )
```

Description

This API is used to get the interrupt status of interrupt flags of MCPWM.

Parameter

Parameter	Description
pMCPWM	This is the pointer to the MCPWM instance register area
flag	This can be logical OR of the below parameters • RISE_PWM_TIME_PERIOD_MATCH_CH0 : Time base interrupt for 0th channel without considering postscaler if bit is 1 then pwm time period match interrupt for 0th channel will be cleared,if zero then no effect. • PWM_TIME_PRD_MATCH_INTR_CH0 : Time base interrupt for 0th channel which considering postscaler if bit is 1 then pwm time period match interrupt for 0th channel will be cleared if zero then no effect. • FLT_A_INTR : Fault A pin interrupt if bit is one then pwm faultA interrupt will be cleared if zero then no effect. • FLT_B_INTR : Fault B pin interrupt if bit is one pwm faultB interrupt will be cleared if zero then no effect. • RISE_PWM_TIME_PERIOD_MATCH_CH1 : Time base interrupt for 1st channel without considering postscaler if bit is one pwm time period match interrupt for 1st channel will be cleared if bit is zero no effect • PWM_TIME_PRD_MATCH_INTR_CH1 : Time base interrupt for 1st channel which considering postscaler if bit is 1 then pwm time period match interrupt for 1st channel will be cleared if zero then no effect.

Parameter	Description
	• RISE_PWM_TIME_PERIOD_MATCH_CH2 : Time base interrupt for 2nd channel without considering postscaler if bit is one pwm time period match interrupt for 2nd channel will be cleared if bit is zero no effect • PWM_TIME_PRD_MATCH_INTR_CH2 : Time base interrupt for 2nd channel which considering postscaler if bit is one pwm time period match interrupt for 2nd channel will be cleared if bit is zero no effect • RISE_PWM_TIME_PERIOD_MATCH_CH3 : Time base interrupt for 3rd channel without considering postscaler if bit is one pwm time period match interrupt for 3rd channel will be cleared if bit is zero no effect • PWM_TIME_PRD_MATCH_INTR_CH3 : Time base interrupt for 3rd channel which considering postscaler if bit is one pwm time period match interrupt for 3rd channel will be cleared if bit is zero no effect

Return values

None

Example

```
/* get interrupt status*/
RSI_PWM_GetInterruptStatus( MCPWM , RISE_PWM_TIME_PERIOD_MATCH_CH0);
```

23.3.39 RSI_MCPWM_InterruptClear

Source File : rsi_pwm.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_MCPWM_InterruptClear( RSI_MCPWM_T *pMCPWM, uint32_t clrFlag )
```

Description

This API is used to clear the interrupts of MCPWM.

Parameter

Parameter	Description
pMCPWM	This is the pointer to the MCPWM instance register area
clrFlag	This can be logical OR of the below parameters • RISE_PWM_TIME_PERIOD_MATCH_CH0_ACK : Time base interrupt for 0th channel without considering postscaler if bit is one then pwm time period match interrupt for 0th channel will be cleared if bit zero then no effect. • PWM_TIME_PRD_MATCH_INTR_CH0_ACK : Time base interrupt for 0th channel which considering postscaler if bit is one then pwm time period match interrupt for 0th channel will be cleared if bit zero then no effect. • FLT_A_INTR_ACK : Fault A pin interrupt if bit is set pwm faultA interrupt will be cleared,if zero then ,no effect. • FLT_A_INTR_ACK : Fault B pin interrupt

Parameter	Description
	if bit is set pwm faultB interrupt will be cleared,if zero then ,no effect. • RISE_PWM_TIME_PERIOD_MATCH_CH1_ACK : Time base interrupt for 1th channel without considering postscaler if bit is one then pwm time period match interrupt for 1th channel will be cleared if bit zero then no effect. • PWM_TIME_PRD_MATCH_INTR_CH1_ACK : Time base interrupt for 1th channel which considering postscaler if bit is one then pwm time period match interrupt for 1th channel will be cleared if bit zero then no effect. • RISE_PWM_TIME_PERIOD_MATCH_CH2_ACK : Time base interrupt for 2nd channel without considering postscaler if bit is one then pwm time period match interrupt for 1th channel will be cleared if bit zero then no effect. • PWM_TIME_PRD_MATCH_INTR_CH2_ACK : Time base interrupt for 2nd channel which considering postscaler if bit is one then pwm time period match interrupt for 2nd channel will be cleared if bit zero then no effect. • RISE_PWM_TIME_PERIOD_MATCH_CH3_ACK : Time base interrupt for 3rd channel without considering postscaler if bit is one then pwm time period match interrupt for 3rd channel will be cleared if bit zero then no effect. • : Time base interrupt for 3rd channel which considering postscaler if bit is one then pwm time period match interrupt for 3rd channel will be cleared if bit zero then no effect.

Return values

None

Example

```
/* clear interrupt */
RSI_MCPWM_InterruptClear( MCPWM ,RSI_MCPWM_EVENT_TIME_PERIOD_MATCH_CH0);
```

23.3.40 RSI_MCPWM_InterruptDisable

Source File : rsi_pwm.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_MCPWM_InterruptDisable( RSI_MCPWM_T *pMCPWM, uint16_t flag )
```

Description

This API is used to disable the interrupts of MCPWM.

Parameter

Parameter	Description
pMCPWM	This is the pointer to the MCPWM instance register area
clrFlag	This can be logical OR of the below parameters • RISE_PWM_TIME_PERIOD_MATCH_CH0 : Time base interrupt for 0th channel without considering postscaler if bit is 1 then pwm time period match interrupt for 0th channel will be cleared,if

Parameter	Description
	zero then no effect. • PWM_TIME_PRD_MATCH_INTR_CH0 : Time base interrupt for 0th channel which considering postscaler if bit is 1 then pwm time period match interrupt for 0th channel will be cleared if zero then no effect. • FLT_A_INTR : Fault A pin interrupt if bit is one then pwm faultA interrupt will be cleared if zero then no effect. • FLT_B_INTR : Fault B pin interrupt if bit is one pwm faultB interrupt will be cleared if zero then no effect. • RISE_PWM_TIME_PERIOD_MATCH_CH1 : Time base interrupt for 1st channel without considering postscaler if bit is one pwm time period match interrupt for 1st channel will be cleared if bit is zero no effect • PWM_TIME_PRD_MATCH_INTR_CH1 : Time base interrupt for 1st channel which considering postscaler if bit is 1 then pwm time period match interrupt for 1st channel will be cleared if zero then no effect. • RISE_PWM_TIME_PERIOD_MATCH_CH2 : Time base interrupt for 2nd channel without considering postscaler if bit is one pwm time period match interrupt for 2nd channel will be cleared if bit is zero no effect • PWM_TIME_PRD_MATCH_INTR_CH2 : Time base interrupt for 2nd channel which considering postscaler if bit is one pwm time period match interrupt for 2nd channel will be cleared if bit is zero no effect • RISE_PWM_TIME_PERIOD_MATCH_CH3 : Time base interrupt for 3rd channel without considering postscaler if bit is one pwm time period match interrupt for 3rd channel will be cleared if bit is zero no effect • PWM_TIME_PRD_MATCH_INTR_CH3 : Time base interrupt for 3rd channel which considering postscaler if bit is one pwm time period match interrupt for 3rd channel will be cleared if bit is zero no effect

Return values

None

Example

```
/* enable interrupt */
RSI_MCPWM_InterruptDisable( MCPWM, RSI_MCPWM_EVENT_TIME_PERIOD_MATCH_CH0 );
```

23.3.41 RSI_MCPWM_ExternalTriggerControl

Source File : rsi_pwm.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_MCPWM_ExternalTriggerControl( RSI_MCPWM_T *pMCPWM, boolean_t enable
)
```

Description

This API is used to enable to use external trigger for base time counter increment or decrement of MCPWM

Parameter

Parameter	Description
pMCPWM	This is the pointer to the MCPWM instance register area
enable	If 0 then disable external trigger If 1 then enable external trigger.

Return values

None

Example

```
/* external event trigger */
RSI_MCPWM_ExternalTriggerControl(MCPWM,1);
```

23.3.42 RSI_MCPWM_Stop

Source File : rsi_rom_pwm.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_MCPWM_Stop( RSI_MCPWM_T *pMCPWM,uint8_t chnlNum )
```

Description

This API is used to stops the MCPWM operation for required channel.

Parameter

Parameter	Description
pMCPWM	This is the pointer to the MCPWM instance register area
chnlNum	This is the channel number (0 to 3)

Return values

ERROR_PWM_INVALID_CHNLNUM : If channel number is invalid

RSI_OK : If process is done successfully

Example

```
/* Stop MCPWM  */
RSI_MCPWM_Stop(MCPWM,0);
```

23.3.43

24 Quadrature Encoder Interface (QEI)

24.1 Overview

This section explains how to configure and use the Quadrature Encoder Interface using Redpine MCU SAPIs.

24.2 Programming Sequence

```
#include "rsi_chip.h" /* Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\common\chip\inc */
#include "rsi_rom_crc.h /* Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc
*/

#define RSI_BLINK_RATE        1000                /*<! 1000 ticks per second     */
#define SOC_OPER_FREQUENCY    30000000          /*<! 30Mhz is SOC frequency   */
#define QEI_IDX_PIN           6
#define QEI_PHB_PIN           36
#define QEI_PHA_PIN           8
#define QEI_DIR_PIN           9
#define QEI_IDX_PORT          0
#define QEI_PHB_PORT          0
#define QEI_PHA_PORT          0
#define QEI_DIR_PORT          0
#define QEI_IDX_REVOLUTIONS   10
#define QEI_STIMULUS_PHA      18
#define QEI_STIMULUS_PHB      19
#define QEI_STIMULUS_IDX      20
#define QEI_STIMULUS_PORT     0
#define QEI_DELTA_TIME_IN_US  100000

static void RSI_QEI_StimulusPinMuxInit(void)
{
    /*Pad selection enable */
    RSI_EGPIO_PadSelectionEnable(1);
    /*PAD receive enable */
    RSI_EGPIO_PadReceiverEnable(QEI_STIMULUS_PHA);
    RSI_EGPIO_PadReceiverEnable(QEI_STIMULUS_PHB);
    RSI_EGPIO_PadReceiverEnable(QEI_STIMULUS_IDX);
    /*QEI stimulus pin MUX configuration*/
    RSI_EGPIO_SetPinMux(EGPIO , QEI_STIMULUS_PORT , QEI_STIMULUS_PHA ,
EGPIO_PIN_MUX_MODE0);
    RSI_EGPIO_SetPinMux(EGPIO , QEI_STIMULUS_PORT , QEI_STIMULUS_PHB ,
EGPIO_PIN_MUX_MODE0);
    RSI_EGPIO_SetPinMux(EGPIO , QEI_STIMULUS_PORT , QEI_STIMULUS_IDX ,
EGPIO_PIN_MUX_MODE0);
    /*Configure the direction*/
    RSI_EGPIO_SetDir(EGPIO , QEI_STIMULUS_PORT , QEI_STIMULUS_PHA ,
EGPIO_CONFIG_DIR_OUTPUT);
    RSI_EGPIO_SetDir(EGPIO , QEI_STIMULUS_PORT , QEI_STIMULUS_PHB ,
EGPIO_CONFIG_DIR_OUTPUT);
    RSI_EGPIO_SetDir(EGPIO , QEI_STIMULUS_PORT , QEI_STIMULUS_IDX ,
EGPIO_CONFIG_DIR_OUTPUT);
    /*Configure the pin value*/
    RSI_EGPIO_SetPin(EGPIO , QEI_STIMULUS_PORT , QEI_STIMULUS_PHA , 0);
    RSI_EGPIO_SetPin(EGPIO , QEI_STIMULUS_PORT , QEI_STIMULUS_PHB , 1);
    RSI_EGPIO_SetPin(EGPIO , QEI_STIMULUS_PORT , QEI_STIMULUS_IDX , 0);
    return;
}
void QEI_IRQHandler(void)
{
    volatile uint16_t intStatus =0;

    /*Get interrupt status*/
    intStatus = RSI_QEI_GetIntrStatus(QEI);

    if(intStatus & QEI_POS_CNT_RST_INTR_LVL)
    {
        /*Clear interrupt */
```

```
        RSI_QEI_ClrIntrStatus(QEI,QEI_POS_CNT_RST_INTR_LVL);
    }
    if(intStatus & QEI_IDX_CNT_MAT_INT_LVL)
    {
        /*Clear interrupt */
        RSI_QEI_ClrIntrStatus(QEI,QEI_IDX_CNT_MAT_INT_LVL);
    }
    if(intStatus & QEI_POS_CNT_ERR_INTR_LVL)
    {
        /*Clear interrupt */
        RSI_QEI_ClrIntrStatus(QEI,QEI_POS_CNT_ERR_INTR_LVL);
    }
    if(intStatus & QEI_VELO_LESS_THAN_INTR_LVL)
    {
        /*Clear interrupt */
        RSI_QEI_ClrIntrStatus(QEI,QEI_VELO_LESS_THAN_INTR_LVL);
    }
    if(intStatus & QEI_POS_CNT_MAT_INTR_LVL)
    {
        /*Clear interrupt */
        RSI_QEI_ClrIntrStatus(QEI,QEI_POS_CNT_MAT_INTR_LVL);
    }
    /*Velocity measurement*/
    if(intStatus & VELOCITY_COMPUTATION_OVER_INTR_LVL)
    {
        qeiVelocityComputeDone = 1;

        RSI_QEI_IntrMask(QEI,QEI_VEL_COMPUTATION_OVER_INTR_MASK);
        /*Clear interrupt */
        RSI_QEI_ClrIntrStatus(QEI,VELOCITY_COMPUTATION_OVER_INTR_LVL);
        /*Get QEI Velocity */
        qeiVelocity=RSI_QEI_GetVelocity(QEI);
        /*Velocity value returns the number of positions in given delta time*/
        /*RPM is number of positions /revolution*/
        qeiMotorRpm = (qeiVelocity / QEI_IDX_REVOLUTIONS);
        /*Start qeiVelocity counter */
        RSI_QEI_StartVelocityCounter(QEI);
    }
    return;
}

int main(void)
{
    uint32_t qeiPosition=0,qeiIdx =0;
    uint8_t qeiDirection;
    volatile int forever=0;

    /*Pin MUX INIT */
    RSI_QEI_StimulusPinMuxInit();
    RSI_QEI_PinMuxInit();
    /*Clock enable for QEI*/
    RSI_QEI_Enable(QEI);
    /*Configure the encoding mode*/
    RSI_QEI_SetMode(QEI,QEI_ENCODING_MODE_1X);
    /*Configure the mode of QEI operation */
    /*Swap Phase A and Phase B if required*/
    RSI_QEI_SetConfiguration(QEI,QEI_SWAP_PHASE_AB_B);
    /*Configure the QEI base clock and delta time to compute the velocity*/
    RSI_QEI_ConfigureDeltaTimeAndFreq(QEI,SOC_OPER_FREQUENCY,QEI_DELTA_TIME_IN_US);
    /*Start the velocity counter*/
    RSI_QEI_StartVelocityCounter(QEI);
    /*Interrupt un mask*/
    RSI_QEI_IntrUnMask(QEI,QEI_VEL_COMPUTATION_OVER_INTR_MASK);
    /*Interrupt mask*/
    /*NVIC enable for QEI*/
    NVIC_EnableIRQ(QEI_NVIC_NAME);
    /*Configure the system tick timer to generate the QEI signals*/
    SysTick_Config(SOC_OPER_FREQUENCY/ RSI_BLINK_RATE);
    /*Enable NVIC for QEI*/
    forever=1;
    while(forever)
```

```
    {
        /*Get the position count*/
        qeiPosition  = RSI_QEI_GetPosition(QEI);
        /*Get QEI index count*/
        qeiIdx= RSI_QEI_GetIndex(QEI);
        /*Get QEI direction*/
        qeiDirection = RSI_QEI_GetDirection(QEI);
        /**/
        if(qeiVelocityComputeDone){
            qeiVelocityComputeDone=0;
            /*Un mask the interrupt for velocity computation*/
            RSI_QEI_IntrUnMask(QEI,QEI_VEL_COMPUTATION_OVER_INTR_MASK);
        }
    }
    return 0;
}
```

Code Block 17 Quadrature Encoder Interface Example

24.3 API Descriptions

24.3.1 RSI_QEI_Enable

Source File : rsi_qei.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
void RSI_QEI_Enable(volatile QEI_Type  *pstcQei)
```

Description

Enables the clock to the QEI module.

Parameter

Parameter	Description
pstcQei	This is the pointer to the QEI register instance

Return values

None

Example

```
/* enable for QEI*/
RSI_QEI_Enable(QEI);
```

24.3.2 void RSI_QEI_SetMode

Source File : rsi_qei.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_QEI_SetMode (volatile QEI_Type *pstcQei, uint8_t encMode)
```

Description

Sets the QEI encoding modes (eg:1X , 2X , 4X)

Parameters

Parameter	Description
pstcQei	This is the pointer to the QEI register instance
encMode	This is the encoding mode The enum values are as below : • QEI_ENCODING_MODE_1X • QEI_ENCODING_MODE_2X • QEI_ENCODING_MODE_4X

Return values

None

Example

```
/* Set mode */
RSI_QEI_SetMode(QEI,QEI_ENCODING_MODE_1X);
```

24.3.3 RSI_QEI_GetMode

Source File : rsi_qei.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE uint32_t RSI_QEI_GetMode (volatile QEI_Type *pstcQei)
```

Description

Get the QEI encoding modes (eg:1X , 2X , 4X)

Parameter

Parameter	Description
pstcQei	This is the pointer to the QEI register instance

Return values

Returns the Encoding mode

Example

```
uint32_t QEI_Mode;
/* get mode */
QEI_Mode=RSI_QEI_GetMode(QEI);
```

24.3.4 RSI_QEI_SetConfiguration

Source File : rsi_qei.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_QEI_SetConfiguration(volatile QEI_Type  *pstcQei, uint32_t
configParms)
```

Description

Sets the QEI module configurations.

Parameters

Parameter	Description
pstcQei	This is the pointer to the QEI register instance
configParms	These are the OR'ed values configurations needs to be passed to this variable The possible values are as below : • QEI_SFT_RST • QEI_SWAP_PHASE_AB_B • QEI_POS_CNT_RST_WITH_IDX_EN • QEI_POS_CNT_DIR_CTRL • QEI_POS_CNT_DIR_FRM_REG • QEI_IDX_CNT_RST_EN • QEI_DIGITAL_FILTER_BYPASS • QEI_TIMER_MODE • QEI_START_VELOCITY_CNTR • QEI_STOP_IN_IDLE • QEI_POS_CNT_16_BIT_MDE • QEI_POS_CNT_RST • QEI_IDX_CNT_RST

Return values

None

Example

```
/*Configure the mode of QEI operation */
/*Swap Phase A and Phase B if required*/
RSI_QEI_SetConfiguration(QEI,QEI_SWAP_PHASE_AB_B);
```

24.3.5 RSI_QEI_ConfigureDeltaTimeAndFreq

Source File : rsi_qei.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_QEI_ConfigureDeltaTimeAndFreq (volatile QEI_Type *pstcQei, uint32_t
freq, uint32_t PeriodInUs)
```

Description

Configure the delta time and QEI module input frequency for velocity measurements.

Parameters

Parameter	Description
pstcQei	This is the pointer to the QEI register instance
freq	This is the QEI module input frequency (typically soc clock) in hertz.
PeriodInUs	This is the velocity measurement time in micro seconds (ex: for sec program 1000000)

Return values

None

Example

```
#define QEI_DELTA_TIME_IN_US  1000000
/*Configure the QEI base clock and delta time to compute the velocity*/
RSI_QEI_ConfigureDeltaTimeAndFreq(QEI,SOC_OPER_FREQUENCY,QEI_DELTA_TIME_IN_US);
```

24.3.6 RSI_QEI_StartVelocityCounter

Source File : rsi_qei.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_QEI_StartVelocityCounter (volatile QEI_Type *pstcQei)
```

Description

Starts the velocity counter.

Parameter

Parameter	Description
pstcQei	This is the pointer to the QEI register instance

Return values

none

Example

```
/*Start the velocity counter */
RSI_QEI_StartVelocityCounter(QEI);
```

24.3.7 RSI_QEI_IntrUnMask

Source File : rsi_qei.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_QEI_IntrUnMask (volatile QEI_Type *pstcQei, uint32_t intrMask)
```

Description

Un masks the QEI interrupts.

Parameters

Parameter	Description
pstcQei	This is the pointer to the QEI register instance
intrMask	These are the OR'ed values of interrupt mask bits to be un masked The possible value are as below: • QEI_POS_CNT_RESET_INTR_UNMASK • QEI_IDX_CNT_MATCH_INTR_UNMASK • QEI_VEL_LESS_THAN_INTR_UNMASK • QEI_IDX_POS_MATCH_INTR_UNMASK • QEI_VEL_COMPUTATION_OVER_INTR_MASK • QEI_POS_CNTR_ERR_INTR_UNMASK

Return values

None

Example

```
/*Interrupt un mask*/
RSI_QEI_IntrUnMask(QEI,QEI_VEL_COMPUTATION_OVER_INTR_MASK);
```

24.3.8 RSI_QEI_GetIntrStatus

Source File : rsi_qei.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE uint32_t RSI_QEI_GetIntrStatus (volatile QEI_Type *pstcQei)
```

Description

Get the interrupt status of QEI module.

Parameter

Parameter	Description
pstcQei	Pointer to the QEI register instance

Return values

returns the interrupt status bits

Example

```
volatile uint16_t intStatus =0;
/*Get interrupt status*/
intStatus = RSI_QEI_GetIntrStatus(QEI);
```

24.3.9 RSI_QEI_ClrIntrStatus

Source File : rsi_qei.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_QEI_ClrIntrStatus (volatile QEI_Type *pstcQei, uint32_t intrMask)
```

Description

Clears the QEI interrupt status.

Parameters

Parameter	Description
pstcQei	Pointer to the QEI register instance
intrMask	OR'ed values of interrupt mask bits to be cleared • QEI_POS_CNT_RST_INTR_LVL • QEI_IDX_CNT_MAT_INT_LVL • QEI_POS_CNT_ERR_INTR_LVL • QEI_VELO_LESS_THAN_INTR_LVL • QEI_POS_CNT_MAT_INTR_LVL • VELOCITY_COMPUTATION_OVER_INTR_LVL

Return values

None

Example

```
/*Clear interrupt status*/
RSI_QEI_ClrIntrStatus(QEI,QEI_POS_CNT_RST_INTR_LVL);
```

24.3.10 RSI_QEI_IntrMask

Source File : rsi_qei.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_QEI_IntrMask (volatile QEI_Type *pstcQei, uint32_t intrMask)
```

Description

Masks the QEI interrupts.

Parameters

Parameter	Description
pstcQei	This is the pointer to the QEI register instance
intrMask	These are the OR'ed values of interrupt mask bits to be masked The possible values are as below : • QEI_POS_CNT_RESET_INTR_UNMASK • QEI_IDX_CNT_MATCH_INTR_UNMASK

Parameter	Description
	• QEI_VEL_LESS_THAN_INTR_UNMASK • QEI_IDX_POS_MATCH_INTR_UNMASK • QEI_VEL_COMPUTATION_OVER_INTR_MASK • QEI_POS_CNTR_ERR_INTR_UNMASK

Return values

None

Example

```
/*Masks the QEI interrupts*/
RSI_QEI_IntrMask(QEI,QEI_VEL_COMPUTATION_OVER_INTR_MASK);
```

24.3.11 RSI_QEI_GetVelocity

Source File : rsi_qei.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE uint32_t RSI_QEI_GetVelocity (volatile QEI_Type *pstcQei)
```

Description

Get the velocity of QEI module for programmed delta time.

Parameter

Parameter	Description
pstcQei	This is the pointer to the QEI register instance

Return values

The number of position counts for configured delta time from start of velocity trigger.

Example

```
static uint32_t qeiVelocity=0
/*Get QEI Velocity */
qeiVelocity=RSI_QEI_GetVelocity(QEI);
```

24.3.12 RSI_QEI_GetPosition

Source File : rsi_qei.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE uint32_t RSI_QEI_GetPosition (volatile QEI_Type *pstcQei)
```

Description

Gets the current position increment counter value to know the position of the motor.

Parameter

Parameter	Description
pstcQei	This is the pointer to the QEI register instance

Return values

position current position counter value

QEI_POSITION_CNT_REG

Example

```
uint32_t qeiPosition=0
/*Get the position count*/
qeiPosition  = RSI_QEI_GetPosition(QEI);
```

24.3.13 RSI_QEI_GetIndex

Source File : rsi_qei.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE uint32_t RSI_QEI_GetIndex (volatile QEI_Type *pstcQei)
```

Description

Gets the current index increment counter value to know the index count of the motor.

Parameter

Parameter	Description
pstcQei	This is the pointer to the QEI register instance

Return values

index current counter value

QEI_INDEX_CNT_REG

Example

```
uint32_t qeiIdx =0
/*Get QEI index count*/
qeiIdx= RSI_QEI_GetIndex(QEI);
```

24.3.14 RSI_QEI_SetControls

Source File : rsi_qei.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_QEI_SetControls (volatile QEI_Type *pstcQei, uint32_t cntrlParms)
```

Description

Sets the QEI control configuration.

Parameters

Parameter	Description
pstcQei	This is the pointer to the QEI register instance
cntrlParms	These are the OR'ed values control field bits The possible values are as below : • QEI_INDEX_CNT_INIT • QEI_UNIDIRECTIONAL_INDEX • QEI_UNIDIRECTIONAL_VELOCITY

Return values

None

Example

```
/* set control parameter */
RSI_QEI_SetControls(QEI,QEI_UNIDIRECTIONAL_INDEX);
```

24.3.15 RSI_QEI_ClrControls

Source File : rsi_qei.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_QEI_ClrControls (volatile QEI_Type *pstcQei, uint32_t cntrlParms)
```

Description

Clears the QEI control configuration.

Parameters

Parameter	Description
pstcQei	This is the pointer to the QEI register instance
configParms	These are the OR'ed values control field bits The possible values are as below : • QEI_INDEX_CNT_INIT • QEI_UNIDIRECTIONAL_INDEX • QEI_UNIDIRECTIONAL_VELOCITY

Return values

None

Example

```
/* clear control parameter */
RSI_QEI_ClrControls(QEI,QEI_UNIDIRECTIONAL_INDEX);
```

24.3.16 RSI_QEI_SetMaxPosCnt

Source File : rsi_qei.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_QEI_SetMaxPosCnt (volatile QEI_Type *pstcQei, uint32_t maxPosition)
```

Description

Sets the MAX value for position counter. This is a maximum count value that is allowed to increment in the position counter. If position counter reaches this value in positive direction, will get set to zero and if reaches zero in negative direction, will get set to this max count.Pos_cnt_rst_with_index_en, qei_ctrl_reg[2] should be reset (0) for setting the position counter when reaches max/zero value. QEI_POSITION_MAX_CNT_LSW_REG.

Parameters

Parameter	Description
pstcQei	This is the pointer to the QEI register instance
maxPosition	This is the max position value

Return values

None

Example

```
/* set position count */
RSI_QEI_SetMaxPosCnt(QEI,1000);
```

24.3.17 RSI_QEI_GetMaxPosCnt

Source File : rsi_qei.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE uint32_t RSI_QEI_GetMaxPosCnt (volatile QEI_Type *pstcQei)
```

Description

Gets the maximum position count.

Parameter

Parameter	Description
pstcQei	This is the pointer to the QEI register instance

Return values

Returns the maximum position count.

Example

```
uint32_t MaxPosCnt;
/* get maximum position count value /
Maxposcnt=RSI_QEI_GetMaxPosCnt(QEI);
```

24.3.18 RSI_QEI_SetPosMatch

Source File : rsi_qei.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_QEI_SetPosMatch (volatile QEI_Type *pstcQei, uint32_t
matchPosition)
```

Description

Sets the position match value to compare the position counter. When it is matched with position counter, interrupt is raised.

Parameters

Parameter	Description
pstcQei	This is the pointer to the QEI register instance
matchPosition	This is the position match value. QEI_POSITION_MATCH_REG

Return values

Returns the maximum position count

Example

```
/* set position match value */
RSI_QEI_SetPosMatch(QEI,500);
```

24.3.19 RSI_QEI_GetPosMatch

Source File : rsi_qei.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE uint32_t RSI_QEI_GetPosMatch (volatile QEI_Type *pstcQei)
```

Description

Gets the position match value

Parameter

Parameter	Description

Parameter	Description
pstcQei	This is the pointer to the QEI register instance

Return values

Returns the position match value

Example

```
uint32_t MaxPosMatch;
MaxPosMatch=RSI_QEI_GetPosMatch(QEI);
```

24.3.20 RSI_QEI_SetMaxIndex

Source File : rsi_qei.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_QEI_SetMaxIndex (volatile QEI_Type *pstcQei, uint16_t maxIdxCntVal)
```

Description

Configures the QEI index maximum count.This is a maximum count value that is allowed to increment in the index counter.

Parameters

Parameter	Description
pstcQei	This is the pointer to the QEI register instance
maxIdxCntVal	This is the maximum allowed index value to be programmed.

Return values

None

Example

```
/* set max index count value */
RSI_QEI_SetMaxIndex(QEI,20);
```

24.3.21 RSI_QEI_GetMaxIndex

Source File : rsi_qei.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE uint16_t RSI_QEI_GetMaxIndex(volatile QEI_Type  *pstcQei)
```

Description

Gets the maximum index value.

Parameters

Parameter	Description
pstcQei	This is the pointer to the QEI register instance
maxIdxCntVal	This is the maximum allowed index value to be programmed.

Return values

None

Example

```
uint32_t get_index_count;
/* get max index count value */
get_index_count = RSI_QEI_GetMaxIndex(QEI);
```

24.3.22 RSI_QEI_GetDirection

Source File : rsi_qei.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE boolean_t RSI_QEI_GetDirection (volatile QEI_Type *pstcQei)
```

Description

Gets the direction of the QEI interfaced motor.

Parameter

Parameter	Description
pstcQei	This is the pointer to the QEI register instance

Return values

Returns 1: (+) Positive direction QEI_STATUS_REG_b 0: (-) Negative direction QEI_STATUS_REG_b

Example

```
uint8_t qeiDirection;
/*Get QEI direction*/
qeiDirection = RSI_QEI_GetDirection(QEI);
```

24.3.23 RSI_QEI_GetStatus

Source File : rsi_qei.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE int32_t RSI_QEI_GetStatus (volatile QEI_Type *pstcQei)
```

Description

Gets the status of the QEI module.

Parameter

Parameter	Description
pstcQei	This is the pointer to the QEI register instance

Return values

Returns the status of the QEI. (QEI_STATUS_REG)

Example

```
int32_t Get_Status;
/* get QEI module status */
Get_Status=RSI_QEI_GetStatus(QEI);
```

24.3.24 RSI_QEI_StopVelocityCounter

Source File : rsi_qei.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_QEI_StopVelocityCounter (volatile QEI_Type *pstcQei)
```

Description

Stops the velocity counter.

Parameter

Parameter	Description
pstcQei	this is the pointer to the QEI register instance

Return values

None

Example

```
/* stop velocity counter */
RSI_QEI_StopVelocityCounter(QEI);
```

24.3.25 RSI_QEI_SetDigitalFilterClkDiv

Source File : rsi_qei.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_QEI_SetDigitalFilterClkDiv (volatile QEI_Type *pstcQei, uint32_t
div)
```

Description

Configures the digital filter clock division selects.

Parameters

Parameter	Description
pstcQei	This is the pointer to the QEI register instance
div	This is the digital filter clock division selects

Return values

None

Example

```
RSI_QEI_SetDigitalFilterClkDiv(QEI,100);
```

24.3.26 RSI_QEI_GetDigitalFilterClkDiv

Source File : rsi_qei.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE uint32_t RSI_QEI_GetDigitalFilterClkDiv(volatile QEI_Type  *pstcQei)
```

Description

Configures the digital filter clock division selects.

Parameters

Parameter	Description
pstcQei	This is the pointer to the QEI register instance
div	This is the digital filter clock division selects

Return values

None

Example

```
uint32_t filter_div_val;
/* Get the filter division value */
filter_div_val = RSI_QEI_GetDigitalFilterClkDiv(QEI);
```

24.3.27 RSI_QEI_SetModuleFreq

Source File : rsi_qei.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_QEI_SetModuleFreq (volatile QEI_Type *pstcQei, uint32_t freq)
```

Description

Sets the QEI module frequency for velocity measurements computations.

Parameters

Parameter	Description
pstcQei	This is the pointer to the QEI register instance
freq	This is the QEI module input frequency (typically soc clock) in hertz (QEI_CLK_FREQ_REG)

Return values

None

Example

```
/*set module frequency */
RSI_QEI_SetModuleFreq(QEI,10000);
```

24.3.28 RSI_QEI_GetModuleFreq

Source File : rsi_qei.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE uint32_t RSI_QEI_GetModuleFreq (volatile QEI_Type *pstcQei)
```

Description

Gets the QEI module frequency for velocity measurements computations.

Parameter

Parameter	Description
pstcQei	This is the pointer to the QEI register instance

Return values

Returns the module frequency in hertz (QEI_CLK_FREQ_REG)

Example

```
uint32_t  Get_Freq;
/* get madule frequency */
Get_Freq=RSI_QEI_GetModuleFreq(QEI);
```

24.3.29 RSI_QEI_SetDeltaTime

Source File : rsi_qei.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_QEI_SetDeltaTime (volatile QEI_Type *pstcQei, uint32_t PeriodInUs)
```

Description

Sets the QEI module delta time for velocity computation.

Parameters

Parameter	Description
pstcQei	This is the pointer to the QEI register instance
PeriodInUs	This is the velocity measurement time in micro seconds (ex: for sec program 1000000)

Return values

Returns the module frequency in hertz QEI_CLK_FREQ_REG

Example

```
#define QEI_DELTA_TIME_IN_US  1000000
/* set delta time */
RSI_QEI_SetDeltaTime(QEI,QEI_DELTA_TIME_IN_US );
```

24.3.30 RSI_QEI_GetDeltaTime

Source File : rsi_qei.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE uint32_t RSI_QEI_GetDeltaTime (volatile QEI_Type *pstcQei)
```

Description

Gets the QEI module delta time configured for velocity computation.

Parameter

Parameter	Description
pstcQei	This is the pointer to the QEI register instance

Return values

Returns the configured delta time.

Example

```
uint32_t  delta_time;
/* get delta time */
delta_time=RSI_QEI_GetDeltaTime(QEI);
```

24.3.31 RSI_QEI_ClrConfiguration

Source File : rsi_qei.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_QEI_ClrConfiguration (volatile QEI_Type *pstcQei, uint32_t
configParms)
```

Description

Clears the QEI module configurations.

Parameters

Parameter	Description
pstcQei	This is the pointer to the QEI register instance
configParms	These are the OR'ed values configurations need to be passed to this variable The possible values are as below: • QEI_SFT_RST • QEI_SWAP_PHASE_AB_B • QEI_POS_CNT_RST_WITH_IDX_EN • QEI_POS_CNT_DIR_CTRL • QEI_POS_CNT_DIR_FRM_REG • QEI_IDX_CNT_RST_EN • QEI_DIGITAL_FILTER_BYPASS • QEI_TIMER_MODE • QEI_START_VELOCITY_CNTR • QEI_STOP_IN_IDLE • QEI_POS_CNT_16_BIT_MDE • QEI_POS_CNT_RST • QEI_IDX_CNT_RST

Return values

None

Example

```
/*clear QEI configuration */
RSI_QEI_ClrConfiguration(QEI,QEI_SWAP_PHASE_AB_B);
```

24.3.32 RSI_QEI_Disable

Source File : rsi_qei.c

Path : Redpine_MCU_Vx.y.z\library\driver\src

Prototype

```
void RSI_QEI_Disable (volatile QEI_Type *pstcQei)
```

Description

Disables the clock to the QEI module.

Parameter

Parameter	Description

Parameter	Description
pstcQei	This is the pointer to the QEI register instance

Return values

None

Example

```
/* disable QEI */
RSI_QEI_Disable(QEI);
```

25 Quad Serial Peripheral Interface Flash(QSPI)

25.1 Overview

This section explains how to configure and use the Quad Serial Peripheral Interface using Redpine MCU SAPIs.

25.2 Programming Sequence

```
/* Includes ------------------------------------------------------------------*/
#include "rsi_qspi.h"
#include "rsi_egpio.h"
#include "rsi_qspi_proto.h"
#include "rsi_chip.h"
#include "UDMA.h"
#include "rsi_board.h"

/* Private typedef -----------------------------------------------------------*/

/* Private define ------------------------------------------------------------*/
//#define    DMA_READ
#define      IO_READ

#define    PadSelectionEnable 15
#define    QSPI_MODE                    2

/*M4 QSPI  pin set */
#define    M4SS_QSPI_CLK    58
#define    M4SS_QSPI_CSN0   59
#define    M4SS_QSPI_D0     60
#define    M4SS_QSPI_D1     61
#define    M4SS_QSPI_D2     62
#define    M4SS_QSPI_D3     63

/* Private macro -------------------------------------------------------------*/
#define    SINGLE_MODE_ENABLE  1
#define    ADDRESS             0x8012020
#define    SIZE                256
#define    CHNL0               0

/* Private variables ---------------------------------------------------------*/
extern RSI_DRIVER_UDMA Driver_UDMA0;
static RSI_DRIVER_UDMA *UDMAdrv = &Driver_UDMA0;
extern RSI_UDMA_HANDLE_T udmaHandle0;
volatile uint32_t done;
uint32_t  read_data[SIZE],data_write[SIZE] ;

/* Private functions ---------------------------------------------------------*/
void Genrate_sample_data(void)
{
    int i ;
    for(i=0;i<= SIZE ; i++)
    {
        data_write[i] = i;
    }
}

/**
 * @brief      UDMA  controller transfer descriptor chain complete callback
 * @param[in]  event dma transfer events
 * @param[in]  ch    dma channel number
 * @return     None
 */
void udmaTransferComplete(uint32_t event, uint8_t ch)
{
    if(event == UDMA_EVENT_XFER_DONE)
    {
        if(ch == 0)
```

```
        {
            done = 1;
        }
    }
}

/* This function configures DMA for read operation */
void UDMA_Read(void)
{
    RSI_UDMA_CHA_CONFIG_DATA_T control;
    RSI_UDMA_CHA_CFG_T config;

    memset(&control, 0, sizeof(RSI_UDMA_CHA_CONFIG_DATA_T));
    memset(&config, 0, sizeof(RSI_UDMA_CHA_CFG_T));

    config.altStruct = 0;
    config.burstReq  = 1;
    config.channelPrioHigh = 0;
    config.dmaCh = CHNL0;
    config.periAck = 0;
    config.periphReq = 0;
    config.reqMask = 0;

    control.transferType       = UDMA_MODE_BASIC;
    control.nextBurst                     = 0;
    control.totalNumOfDMATrans = 255 ;
    control.rPower             = ARBSIZE_256 ;
    control.srcProtCtrl              = 0x000;
    control.dstProtCtrl              = 0x000;
    control.srcSize            = SRC_SIZE_32;
    control.srcInc             = SRC_INC_32;
    control.dstSize            = DST_SIZE_32;
    control.dstInc             = DST_INC_32;

    /* Initialize dma */
    UDMAdrv->Initialize();

    UDMAdrv->ChannelConfigure( CHNL0,   ADDRESS, (uint32_t)read_data,SIZE,
                                control, &config,  udmaTransferComplete );

    /* Enable dma channel */
    UDMAdrv->ChannelEnable(CHNL0);

    /* Enable dma controller */
    UDMAdrv->DMAEnable();

    RSI_UDMA_ChannelSoftwareTrigger(udmaHandle0,CHNL0);
}


/* This function fills QSPI configuration structures */
void GetQspiConfig(spi_config_t *spi_config)
{
    memset(spi_config, 0, sizeof(spi_config_t));

    spi_config->spi_config_1.inst_mode         = SINGLE_MODE;
    spi_config->spi_config_1.addr_mode         = SINGLE_MODE;
    spi_config->spi_config_1.data_mode         = SINGLE_MODE;
    spi_config->spi_config_1.dummy_mode        = SINGLE_MODE;
    spi_config->spi_config_1.extra_byte_mode   = SINGLE_MODE;
    spi_config->spi_config_1.read_cmd          = READ;
    spi_config->spi_config_1.no_of_dummy_bytes = 0;
    spi_config->spi_config_1.prefetch_en       = DIS_PREFETCH;
    spi_config->spi_config_1.dummy_W_or_R      = DUMMY_READS;
    spi_config->spi_config_1.extra_byte_en     = 0;
    spi_config->spi_config_1.d3d2_data         = 3;
    spi_config->spi_config_1.continuous        = CONTINUOUS;
    spi_config->spi_config_1.flash_type        = SST_SPI_FLASH;


    spi_config->spi_config_2.auto_mode         = EN_AUTO_MODE;
```

```
    spi_config->spi_config_2.cs_no              = CHIP_ZERO;
    spi_config->spi_config_2.neg_edge_sampling = NEG_EDGE_SAMPLING;
    spi_config->spi_config_2.qspi_clk_en        = QSPI_FULL_TIME_CLK;
    spi_config->spi_config_2.protection         = DNT_REM_WR_PROT;
    spi_config->spi_config_2.dma_mode           = NO_DMA;
    spi_config->spi_config_2.swap_en            = SWAP;
    spi_config->spi_config_2.full_duplex        = IGNORE_FULL_DUPLEX;
    spi_config->spi_config_2.wrap_len_in_bytes = NO_WRAP;
    spi_config->spi_config_2.addr_width_valid  = 0;
    spi_config->spi_config_2.addr_width         = _24BIT_ADDR;
    spi_config->spi_config_2.pinset_valid       = 0;
    spi_config->spi_config_2.flash_pinset       = GPIO_0_TO_5;
    spi_config->spi_config_2.dummy_cycles_for_controller = 0;

    spi_config->spi_config_3.wr_cmd                   = 0x2;
    spi_config->spi_config_3.wr_inst_mode             = SINGLE_MODE;
    spi_config->spi_config_3.wr_addr_mode             = SINGLE_MODE;
    spi_config->spi_config_3.wr_data_mode             = SINGLE_MODE;
    spi_config->spi_config_3.xip_mode                 = 0;
    spi_config->spi_config_3._16bit_cmd_valid         = 0;
    spi_config->spi_config_3._16bit_rd_cmd_msb        = 0;
    spi_config->spi_config_3.ddr_mode_en              = 0;
    spi_config->spi_config_3.dummys_4_jump            = 1;

    spi_config->spi_config_4._16bit_wr_cmd_msb        = 0;
    spi_config->spi_config_4.qspi_ddr_clk_en          = 0;
    spi_config->spi_config_4.qspi_manual_ddr_phasse   = 0;
    spi_config->spi_config_4.ddr_data_mode            = 0;
    spi_config->spi_config_4.ddr_addr_mode            = 0;
    spi_config->spi_config_4.ddr_inst_mode            = 0;
    spi_config->spi_config_4.ddr_dummy_mode           = 0;
    spi_config->spi_config_4.ddr_extra_byte           = 0;
    spi_config->spi_config_4.dual_flash_mode          = 0;
    spi_config->spi_config_4.secondary_csn            = 1;
    spi_config->spi_config_4.polarity_mode            = 0;
    spi_config->spi_config_4.valid_prot_bits          = 4;
    spi_config->spi_config_4.no_of_ms_dummy_bytes     = 0;
    spi_config->spi_config_4.ddr_dll_en               = 0;
    spi_config->spi_config_4.continue_fetch_en        = 0;

    spi_config->spi_config_5.block_erase_cmd          = BLOCK_ERASE;
    spi_config->spi_config_5.busy_bit_pos             = 0;
    spi_config->spi_config_5.d7_d4_data               = 0xf;
    spi_config->spi_config_5.dummy_bytes_for_rdsr     = 0x0;
    spi_config->spi_config_5.reset_type               = 0x0;

    spi_config->spi_config_6.chip_erase_cmd           = CHIP_ERASE;
    spi_config->spi_config_6.sector_erase_cmd         = SECTOR_ERASE;

    spi_config->spi_config_7.status_reg_write_cmd     = 0x1;
    spi_config->spi_config_7.status_reg_read_cmd      = 0x5;
}

/**
 * @brief    Configures the pin MUX for QSPI interface
 * @return    Nothing
 */
void RSI_QSPI_PinMuxInit(void)
{
    /*Pad selection enable */
    RSI_EGPIO_PadSelectionEnable(15);
    /*Receive enable for QSPI GPIO*/
    RSI_EGPIO_PadReceiverEnable(M4SS_QSPI_CLK);
    RSI_EGPIO_PadReceiverEnable(M4SS_QSPI_CSN0);
    RSI_EGPIO_PadReceiverEnable(M4SS_QSPI_D0);
    RSI_EGPIO_PadReceiverEnable(M4SS_QSPI_D1);
    RSI_EGPIO_PadReceiverEnable(M4SS_QSPI_D2);
    RSI_EGPIO_PadReceiverEnable(M4SS_QSPI_D3);

    /*Set GPIO pin MUX for QSPI*/
    RSI_EGPIO_SetPinMux(EGPIO,0, M4SS_QSPI_CLK  , QSPI_MODE);
```

```
    RSI_EGPIO_SetPinMux(EGPIO,0, M4SS_QSPI_CSN0 , QSPI_MODE);
    RSI_EGPIO_SetPinMux(EGPIO,0, M4SS_QSPI_D0      , QSPI_MODE);
    RSI_EGPIO_SetPinMux(EGPIO,0, M4SS_QSPI_D1     , QSPI_MODE);
    RSI_EGPIO_SetPinMux(EGPIO,0, M4SS_QSPI_D2     , QSPI_MODE);
    RSI_EGPIO_SetPinMux(EGPIO,0, M4SS_QSPI_D3     , QSPI_MODE);
}


/**
 * @brief  Main program.
 * @param  None
 * @retval None
 */
int main()
{
    int i=0;

    /*Init the QSPI configurations structure */
    spi_config_t spi_configs_init;

    /*Configures the system default clock and power configurations*/
    SystemCoreClockUpdate();

    /* Initialize UART for debug prints */
    DEBUGINIT();

    /* Genrates data buffer */
    Genrate_sample_data();

    /*Get QSPI  structures configuration     */
    GetQspiConfig(&spi_configs_init);

    /* Configures the pin MUX for QSPI  pins*/
    RSI_QSPI_PinMuxInit();

    /* initializes QSPI  */
    RSI_QSPI_SpiInit((qspi_reg_t*)QSPI_BASE,&spi_configs_init,0,0,0) ;

    /* Erases the SECTOR   */
    RSI_QSPI_SpiErase((qspi_reg_t *)QSPI_BASE,&spi_configs_init, SECTOR_ERASE, ADDRESS, 1,
0);

    /* writes the data to required address using qspi */
    RSI_QSPI_SpiWrite((qspi_reg_t *)QSPI_BASE,  &spi_configs_init,0x2 ,
                      ADDRESS, (uint8_t *) &data_write[0],
                      sizeof( data_write), 256, _1BYTE, 0,0,0,0,0,0 );
/* DMA Read configs */
#ifdef DMA_READ
    /* Read the data by using UDMA */
    UDMA_Read();
    /* Wait till dma done */
    while(!done);
#endif

/* IO Read config */
#ifdef IO_READ
    /* Reads from the address in manual mode */
    RSI_QSPI_ManualRead((qspi_reg_t *)(QSPI_BASE),&spi_configs_init,
            ADDRESS, (uint8_t *) read_data, 0/*_32BIT*/, sizeof( data_write),0,0,0) ;
#endif

    /* compares both read and write data */
    for(i=0;i<(sizeof( data_write )/sizeof( data_write[0]));i++)
    {
        if(read_data[i]==data_write[i])
        {
            continue;
        }
        else
        {
            break;
```

```
        }
    }

    if(i== (sizeof( data_write )/sizeof( data_write[0]))    )
    {
        /*  Prints on hyper-terminal  */
        DEBUGOUT("Data comparison success\r\n");
    }
    else
    {
        /*  Prints on hyper-terminal  */
        DEBUGOUT("Data comparison fail\r\n");
    }

    while(1);
}
```

Code Block 18 Quad Serial Peripheral Interface Example

25.3 API Descriptions

25.3.1 RSI_QSPI_SpiInit

Source File : rsi_rom_qspi.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_QSPI_SpiInit(qspi_reg_t *qspi_reg,spi_config_t *spi_config,uint32_t
flash_init_req,uint32_t wr_reg_delay_ms, uint8_t fifo_thrsld)
```

Description

This function initializes GPIO, QSPI and flash

Parameters

Parameter	Description
qspi_reg	This is the pointer to qspi_reg_t structure. It contains all qspi regsisters (qspi_reg_t)
spi_config	This is the pointer to spi_config_t structure, spi configuration spi_config_t
flash_init_req	If set only then the qspi_flash_init is called, allows to skip qspi_flash_init in case of ulp_wakeup
wr_reg_delay_ms	This is the delay in ms provided after a register, write operation is performed on flash
fifo_thrsld	This is the FIFO threshold level value

Return values

None

Example

```
/* QSPI initialization */
RSI_QSPI_SpiInit(qspi_reg_s,spi_config,1,0,1);
```

Note

It is expected that gpio init for qspi is already done by the caller

25.3.2 RSI_QSPI_SpiErase

Source File : rsi_rom_qspi.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_QSPI_SpiErase(qspi_reg_t *qspi_reg, spi_config_t *spi_config,
uint32_t erase_cmd, uint32_t blk_sec_addr, uint32_t dis_hw_ctrl, uint32_t wr_reg_delay_ms)
```

Description

This function erases the flash

Parameters

Parameter	Description
qspi_reg	This is the pointer to qspi_reg_t structure, contains all qspi regsisters (qspi_reg_t)
spi_config	This is the pointer to spi_config_t structure,spi configuration spi_config_t
erase_cmd	This is the erase command to be used
blk_sec_addr	This is the Block or Sector Address, in case of block/sector erase
dis_hw_ctrl	This is to disable hw ctrl, while waiting for flash to go idle
wr_reg_delay_ms	This is the delay in ms provided after a register ,write operation is performed on flash

Return values

None

Example

```
/* address in flash */
volatile uint32_t addr = 0x1E100;
/* Erase flash */
RSI_QSPI_SpiErase(qspi_reg_s,  spi_config, SECTOR_ERASE /*SECTOR_ERASE*/, addr, 1, 0);
```

25.3.3 RSI_QSPI_SpiWrite

Source File : rsi_rom_qspi.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
uint32_t RSI_QSPI_SpiWrite(qspi_reg_t *qspi_reg, spi_config_t *spi_config, uint32_t
write_cmd, uint32_t addr, uint8_t  *data,  uint32_t len_in_bytes, uint16_t page_size,
uint32_t hsize, uint32_t dis_hw_ctrl, uint32_t wr_reg_delay_ms,uint32_t check_en,uint32_t
udma_enable,STATIC INLINE void *udmaHandle,STATIC INLINE void *rpdmaHandle)
```

Description

This function writes to flash using qspi

Parameters

Parameter	Description
qspi_reg	This is the pointer to qspi_reg_t structure, contains all qspi regsisters (qspi_reg_t)
spi_config	This is the pointer to spi_config_t structure, spi configuration spi_config_t
write_cmd	This is the write command to be used
addr	This is the address in the flash memory
data	This is the pointer to read data buffer
len_in_bytes	These are the no. of bytes to be read
page_size	The max.burst size allowed by flash.
hsize	8 bits, 16 bits or 32 bits xfer on AHB
dis_hw_ctrl	This is to disable hw ctrl, while waiting for flash to go idle
wr_reg_delay_ms	This is the delay in ms provided after a register, write operation is performed on flash
check_en	If check enable is 1 then data read again and compare with transmit data
udma_enable	1 : uses UDMA 0 - uses RPDMA
udmaHandle	This is the udma context handler
rpdmaHandle	This is the rpdma context handler

Return values

Return the status is return is zero then successfully data write if non-zero then fail.

Example

```
/* write buffer initialization */
uint8_t wr_data3[8192];
/* address in flash */
volatile uint32_t addr = 0x1E100;
/* QSPI Write into flash */
RSI_QSPI_SpiWrite(qspi_reg_s,  spi_config, 0x12ED /* write cmd*/, (addr), wr_data3, 512,
256, _1BYTE, 0, 0,1,1,udmaHandle,rpdmaHandle);
```

25.3.4 RSI_QSPI_ManualRead

Source File : rsi_rom_qspi.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_QSPI_ManualRead(qspi_reg_t *qspi_reg,spi_config_t
*spi_config,uint32_t addr,uint8_t *data,
 *uint32_t hsize,uint32_t len_in_bytes, uint32_t manual_udma_read,
 STATIC INLINE void *udmaHandle, STATIC INLINE void *rpdmaHandle)
```

Description

This function reads from the flash in manual mode,for that it configures the qspi and flash into the required read mode. Data from the qspi fifo can be read using dma mode or using the i/o reads.

Parameters

Parameter	Description
qspi_reg	This is the pointer to qspi_reg_t structure, contains all qspi regsisters (qspi_reg_t)
spi_config	This is the pointer to spi_config_t structure, spi configuration (spi_config_t)
addr	This is the address in flash memory
data	This is the pointer to read data buffer
hsize	8 bits, 16 bits or 32 bits xfer on AHB
len_in_bytes	These are the no. of bytes to be read
manual_udma_read	1 : uses UDMA 0 - uses RPDMA
udmaHandle	This is the uDMA context handler
rpdmaHandle	This is the RPDMA context handler

Return values

None

Example

```
/* read buffer initialization */
uint8_t rd_data[1024];
/* address in flash */
volatile uint32_t addr = 0x1E100;
/* Read data in manual mode */
RSI_QSPI_ManualRead(qspi_reg_s,  spi_config, addr , ((uint8_t *)rd_data), 0/*_32BIT*/,
4096,0,0,rpdmaHandle);
```

25.3.5 RSI_QSPI_WriteToFlash

Source File : rsi_rom_qspi.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_QSPI_WriteToFlash(qspi_reg_t *qspi_reg, uint32_t len_in_bits,
uint32_t cmd_addr_data,uint32_t cs_no)
```

Description

This function writes to flash (8-bit) command or address or data

Parameter

Parameter	Description
qspi_reg	This is the pointer to qspi_reg_t structure, contains all qspi regsisters (qspi_reg_t)
len_in_bits	These are the no. of valid bits
cmd_addr_data	Cmd/Addr/Data
cs_no	This is the chip select no.

Return values

None

Example

```
/* write data or command or address into flash */
RSI_QSPI_WriteToFlash(qspi_reg_s,1024,0x3/*read*/,0);
```

25.3.6 RSI_QSPI_SwitchQspi2

Source File : rsi_rom_qspi.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_QSPI_SwitchQspi2(qspi_reg_t *qspi_reg,uint32_t mode,uint32_t cs_no)
```

Description

This function changes the mode of qspi to SPI/DUAL/QUAD for the required chip select

Parameter

Parameter	Description
qspi_reg	Pointer to qspi_reg_t structure contains all qspi regsisters, qspi_reg_t
mode	The target bus mode for QSPI i.e. SPI/DUAL/QUAD/OCTA
cs_no	Chip select no.

Return values

None

Example

```
/* change mode to SPI */
RSI_QSPI_SwitchQspi2(qspi_reg_s,DUAL_FLASH_MODE,0);
```

25.3.7 RSI_QSPI_WaitFlashStatusIdle

Source File : rsi_rom_qspi.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE uint32_t RSI_QSPI_WaitFlashStatusIdle(qspi_reg_t *qspi_reg,    spi_config_t
*spi_config,    uint32_t wr_reg_delay_ms)
```

Description

This function waits for flash status to go idle

Parameters

Parameter	Description
qspi_reg	This is the pointer to qspi_reg_t structure, contains all qspi regsisters, (qspi_reg_t)
spi_config	This is the pointer to spi_config_t structure, spi configuration (spi_config_t)
wr_reg_delay_ms	This is the delay in ms provided after a register, write operation is performed on flash

Return values

return flash status value

Example

```
uint32_t status;
/* wait till flash status will be ideal */
status=RSI_QSPI_WaitFlashStatusIdle(qspi_reg_s,  spi_config,0);
```

25.3.8 RSI_QSPI_EnableStatusRegWrite

Source File : rsi_rom_qspi.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_QSPI_EnableStatusRegWrite(qspi_reg_t *qspi_reg,uint32_t flash_type,
spi_config_t *spi_config,    uint32_t cs_no)
```

Description

This function enables status register write

Parameters

Parameter	Description
qspi_reg	This is the pointer to qspi_reg_t structure, contains all qspi regsisters, (qspi_reg_t)
flash_type	This is the type of the flash
spi_config	This is the pointer to spi_config_t structure, spi configuration (spi_config_t)
cs_no	This is the chip select no.

Return values

None

Example

```
/* check status for transmission */
RSI_QSPI_EnableStatusRegWrite(qspi_reg_s,0x5,spi_config,0);
```

25.3.9 RSI_QSPI_StatusRegWrite

Source File : rsi_rom_qspi.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_QSPI_StatusRegWrite(qspi_reg_t *qspi_reg,    uint16_t write_value,
spi_config_t *spi_config,uint32_t wr_reg_delay_ms)
```

Description

This function writes to status register ,the protect word and waits till the write is in progress

Parameters

Parameter	Description
qspi_reg	This is the pointer to qspi_reg_t structure, contains all qspi regsisters (qspi_reg_t)
write_value	This is the write value in status register.
spi_config	This is the pointer to spi_config_t structure, spi configuration (spi_config_t)
wr_reg_delay_ms	This is the delay in ms provided after a register, write operation is performed on flash

Return values

None

Example

```
/* write in status register */
RSI_QSPI_StatusRegWrite(qspi_reg_s,  spi_config,0)
```

25.3.10 RSI_QSPI_FlashRegRead

Source File : rsi_rom_qspi.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE uint32_t RSI_QSPI_FlashRegRead(qspi_reg_t *qspi_reg, uint8_t reg_read_cmd,
uint32_t cs_no,  spi_config_t *spi_config)
```

Description

This function reads a register from the flash

Parameters

Parameter	Description
qspi_reg	Pointer to qspi_reg_t structure contains all qspi regsisters, qspi_reg_t
reg_read_cmd	Read command value.
cs_no	Chip select no.
spi_config	Pointer to spi_config_t structure,spi configuration spi_config_t

Return values

value read from config register

Example

```
uint32_t  flash_read;
/* QSPI flash read */
flash_read=RSI_QSPI_FlashRegRead(qspi_reg_s,0x3,0,spi_config);
```

25.3.11 RSI_QSPI_FlashRegWrite

Source File : rsi_rom_qspi.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_QSPI_FlashRegWrite(qspi_reg_t *qspi_reg,uint32_t reg_write_cmd,
uint32_t reg_write_value,    uint32_t cs_no,uint32_t wr_reg_delay_ms)
```

Description

This function writes a register in the flash

Parameters

Parameter	Description
qspi_reg	This is the pointer to qspi_reg_t structure contains all qspi regsisters, (qspi_reg_t)
reg_write_cmd	Register write command to be used
reg_write_value	Value to be written to the register
cs_no	Chip select no.
wr_reg_delay_ms	Delay in ms provided after a register ,write operation is performed on flash

Return values

None

Example

```
/* write value in flash*/
RSI_QSPI_FlashRegWrite(qspi_reg_s,0x6,100,0,0);
```

25.3.12 RSI_QSPI_SetFlashMode

Source File : rsi_rom_qspi.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_QSPI_SetFlashMode(qspi_reg_t *qspi_reg,uint32_t data_mode, uint32_t
cs_no,uint32_t ddr_en,uint32_t flash_type)
```

Description

This function sets the mode for SST_QUAD_FLASH & QSPI.

Parameters

Parameter	Description
qspi_reg	This is the pointer to qspi_reg_t structure, contains all qspi regsisters (qspi_reg_t)
data_mode	This is the operational mode for QSPI and flash
cs_no	This is the chip select no.
ddr_en	This is to enable dual data rate
flash_type	This is the type of the flash variant

Return values

None

Example

```
/* set flash mode */
RSI_QSPI_SetFlashMode(qspi_reg_s,DUAL_FLASH_MODE,0,1,FLASH);
```

25.3.13 RSI_QSPI_ConfigQflash4Read

Source File : rsi_rom_qspi.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_QSPI_ConfigQflash4Read(qspi_reg_t *qspi_reg,spi_config_t
*spi_config,uint32_t addr)
```

Description

This function configures the flash to the desired ,mode specified by spi_config i.e. the instruction, address and any other stages before data phase are executed by this function.

Parameters

Parameter	Description
qspi_reg	Pointer to qspi_reg_t structure contains all qspi regsisters, qspi_reg_t
spi_config	Pointer to spi_config_t structure,spi configuration spi_config_t
addr	Address in flash memory

Return values

None

Example

```
/* address in flash */
volatile uint32_t addr = 0x1E100;
/* configure flash */
RSI_QSPI_ConfigQflash4Read(qspi_reg_s,  spi_config,addr);
```

25.3.14 RSI_QSPI_AutoInit

Source File : rsi_rom_qspi.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_QSPI_AutoInit(qspi_reg_t *qspi_reg,spi_config_t *spi_config)
```

Description

This function initializes auto mode

Parameters

Parameter	Description
qspi_reg	This is the pointer to qspi_reg_t structure, contains all qspi regsisters (qspi_reg_t)
spi_config	This is the pointer to spi_config_t structure, spi configuration

Parameter	Description
	spi_config_t

Return values

None

Example

```
/* QSPI Auto initialization */
RSI_QSPI_AutoInit(qspi_reg_s,  spi_config);
```

25.3.15 RSI_QSPI_AutoRead

Source File : rsi_rom_qspi.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_QSPI_AutoRead(uint32_t cs_no,uint32_t addr,uint8_t *data,uint32_t
hsize,uint32_t len_in_bytes,spi_config_t *spi_config,
uint32_t dma_flags)
```

Description

This function reads from the flash in auto mode

Parameters

Parameter	Description
cs_no	This is the chip select no.(0-3)
addr	This is the address in flash memory
data	This is the pointer to read data buffer
hsize	8 bits, 16 bits or 32 bits xfer on AHB
len_in_bytes	These are the no. of bytes
spi_config	This is the pointer to spi_config_t structure, spi configuration spi_config_t
dma_flags	This is the DMA flag type

Return values

None

Example

```
/* address in flash */
volatile uint32_t addr = 0x1E100;
uint32_t   *data;       /* data pointer */
/* read data */
RSI_QSPI_AutoRead(1,addr,data,_16BIT,1024,spi_config,1);     /* udma flag */
```

25.3.16 RSI_QSPI_FlashInit

Source File : rsi_rom_qspi.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_QSPI_FlashInit(qspi_reg_t *qspi_reg,spi_config_t
*spi_config,uint32_t wr_reg_delay_ms)
```

Description

This function initializes QSPI_flash

Parameters

Parameter	Description
qspi_reg	This is the pointer to qspi_reg_t structure, contains all qspi regsisters (qspi_reg_t)
spi_config	This is the pointer to spi_config_t structure, spi configuration spi_config_t
wr_reg_delay_ms	This is the delay in ms provided after a register ,write operation is performed on flash

Return values

None

Example

```
/* QSPI flash initialization*/
RSI_QSPI_FlashInit(qspi_reg_s,  spi_config,0);
```

25.3.17 RSI_QSPI_SpiRead

Source File : rsi_rom_qspi.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_QSPI_SpiRead(qspi_reg_t *qspi_reg,  spi_config_t *spi_config,
 uint32_t addr,uint8_t  *data,   uint32_t  hsize,   uint32_t len_in_bytes, uint32_t
manual_udma_read, STATIC INLINE void *udmaHandle,  STATIC INLINE void *rpdmaHandle)
```

Description

This function is a mother function to RSI_QSPI_AutoRead & RSI_QSPI_ManualRead

Parameters

Parameter	Description
qspi_reg	Pointer to qspi_reg_t structure contains all qspi regsisters, qspi_reg_t
spi_config	Pointer to spi_config_t structure,spi configuration spi_config_t
addr	Address in flash memory
data	Pointer to read data buffer
hsize	8 bits, 16 bits or 32 bits xfer on AHB
len_in_bytes	No. of bytes to be read
manual_udma_read	1 : uses UDMA, 0 - uses RPDMA
udmaHandle	This is the uDMA context handler
rpdmaHandle	This is the RPDMA context handler

Return values

None

Example

```
/* read buffer initialization */
uint8_t rd_data[1024];
/* address in flash */
volatile uint32_t addr = 0x1E100;
/* QSPI read */
RSI_QSPI_SpiRead(qspi_reg_s,  spi_config,(addr),((uint8_t
*)rd_data),0,4096,0,0,rpdmaHandle);
```

25.3.18 RSI_QSPI_Usleep

Source File : rsi_rom_qspi.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_QSPI_Usleep(uint32_t delay_us)
```

Description

This function is used gives delay.

Parameters

Parameter	Description
delay	This is the uint32_t number of delay

Return values

None

Example

```
/* Delay for 50 us */
RSI_QSPI_Usleep(50);
```

25.3.19 RSI_QSPI_WriteBlockProtect

Source File : rsi_rom_qspi.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_QSPI_WriteBlockProtect(qspi_reg_t *qspi_reg,    uint32_t protect,
uint32_t cs_no,    uint32_t num_prot_bytes,    uint32_t wr_reg_delay_ms)
```

Description

This function writes to block protection reg of SST_QUAD_FLASH.

Parameters

Parameter	Description
qspi_reg	This is the pointer to qspi_reg_t structure, contains all qspi regsisters (qspi_reg_t)

Parameter	Description
protect	This is the block protection register value.
cs_no	This is the chip select no.
num_prot_bytes	This is the protection register length either 6 (SST26VF016) or 10 bytes (SST26VF032)
wr_reg_delay_ms	This is the delay in ms provided after a register write operation is performed on flash

Return values

None

Example

```
/* block protection value */
RSI_QSPI_WriteBlockProtect(qspi_reg_s,ADEST_PROTECT_CMD,0,1000,100)
/* ADESTO flash related */
```

25.3.20 RSI_QSPI_QspiLoadKey

Source File : rsi_rom_qspi.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_QSPI_QspiLoadKey(qspi_reg_t *qspi_reg,    uint32_t *key, uint32_t
kh_enable)
```

Description

This function is used to load the AES key to QSPI Controller.

Parameters

Parameter	Description
qspi_reg	This is the pointer to qspi_reg_t structure, contains all qspi regsisters qspi_reg_t
key	This is the pointer to the key.
kh_enable	This is the kh_enable enable bit.

Return values

None

Example

```
uint32_t *key;          /* here gives specific pointer key value for secure purpose*/
RSI_QSPI_QspiLoadKey(qspi_reg_s,key,1);
```

25.3.21 RSI_QSPI_QspiLoadNonce

Source File : rsi_rom_qspi.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_QSPI_QspiLoadNonce(qspi_reg_t *qspi_reg, uint32_t *nonce)
```

Description

This function is used to load the Nonce to QSPI Controller. M4 QSPI controller is a secure controller, it can decrypt the data inline while fetching from flash

Nonce is a 12 byte random data that is appended with flash offset address in generating cipher

Parameters

Parameter	Description
qspi_reg	Pointer to qspi_reg_t structure,contains all qspi regsisters qspi_reg_t
nonce	Pointer to the nonce

Return values

None

Example

```
uint32_t *nonce;          /* here gives specific pointer value for secure purpose*/

RSI_QSPI_QspiLoadNonce(qspi_reg_s,nonce);
```

25.3.22 RSI_QSPI_SegSecEn

Source File : rsi_rom_qspi.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_QSPI_SegSecEn(qspi_reg_t *qspi_reg,    uint32_t seg_no,    uint32_t
start_addr,    uint32_t end_addr)
```

Description

This function is used to program secure segments of flash to QSPI controller.

Parameters

Parameter	Description
qspi_reg	Pointer to qspi_reg_t structure,contains all qspi regsisters qspi_reg_t
seg_no	Segment number
start_addr	Start address of segment in flash
end_addr	End address of segment in flash

Return values

None

Example

```
uint32_t  *startuint32_t  *end
```

```
RSI_QSPI_SegSecEn(qspi_reg_s,12,(start),(end));
```

25.3.23 RSI_QSPI_StatusControlRegWrite

Source File : rsi_rom_qspi.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_QSPI_StatusControlRegWrite(spi_config_t * spi_config,    qspi_reg_t
*qspi_reg,    uint16_t write_command,    uint32_t addr,    uint16_t write_value, uint32_t
cs_no,uint32_t wr_reg_delay_ms)
```

Description

This function is used to write data to flash.

Parameters

Parameter	Description
spi_config	Pointer to spi_config_t structure,spi configuration spi_config_t
qspi_reg	Pointer to qspi_reg_t structure,contains all qspi regsisters qspi_reg_t
write_command	Command for write data in flash.
addr	Address in flash memory
write_value	Write value which write in flash.
cs_no	Chip select no.
wr_reg_delay_ms	Delay in ms provided after a register write ,operation is performed on flash.

Return values

None

Example

```
RSI_QSPI_StatusControlRegWrite(spi_config,qspi_reg_s,0x12ED /* write cmd*/,(addr),100,0,0)
// remaining
```

25.3.24 RSI_QSPI_FlashProtection

Source File : rsi_rom_qspi.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_QSPI_FlashProtection(spi_config_t *spi_config,qspi_reg_t *qspi_reg,
uint32_t protection, uint32_t wr_reg_delay_ms)
```

Description

This function is used to program secure data of flash to QSPI controller.

Parameters

number of value which is we want protect

Parameter	Description
spi_config	This is the pointer to spi_config_t structure,spi configuration spi_config_t
qspi_reg	This is the pointer to qspi_reg_t structure,contains all qspi regsisters qspi_reg_t
protection	This are the number of values which need protection
wr_reg_delay_ms	This is the delay in ms provided after a register write, operation is performed on flash.

Return values

None

Example

```
/* number of value in protection */
RSI_QSPI_FlashProtection(qspi_reg_s,  spi_config,4092,0);
```

26 Random Number Generator (RNG)

26.1 Overview

This section explains how to configure and use the Random Number Generator using Redpine MCU SAPIs.

26.2 Programming sequence

```
#include "rsi_chip.h" /* Redpine_MCU_Vx.y.z\Host_MCU\common\chip\inc */
#include "rsi_rom_rng.h" /* Redpine_MCU_Vx.y.z\Host_MCU\library\rom_driver\inc */


#define RNG_TRUE_RANDOM    0
#define RNG_PSEUDO_RANDOM  1

int main()
{
  int i;
  uint32_t random_bytes[20];
  //! Enable PSEUDO radom number generator
  RSI_RNG_Start(HWRNG, RNG_TRUE_RANDOM);
  /* delay */
  for( i=0;i<=10000;i++);
  //! Get random bytes
  RSI_RNG_GetBytes(HWRNG, random_bytes, 20);
  RSI_RNG_Stop(HWRNG);
  while(1);
}
```

Code Block 19 Random Number Generator Example

26.3 API Descriptions

26.3.1 RSI_RNG_Start

Source File : rsi_rom_rng.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc

Prototype

```
STATIC INLINE uint32_t  RSI_RNG_Start(HWRNG_Type *pRNG, uint8_t rngMode)
```

Description

This API is used to start the Random Number Generation.

Parameters

Parameters	Description
pRNG	This is the random number Generator handler
rngMode	This is the mode of the Random Number Generator - \ref RNG_TRUE_RANDOM - For True RNG - \ref RNG_PSEUDO_RANDOM - For Psudo RNG

Return values

Returns 0 \ref RSI_OK on success, non zero on failure.

Example

```
/* define true random number generator macro */
#define RNG_TRUE_RANDOM    0
/* start RNG */
RSI_RNG_Start(HWRNG, RNG_TRUE_RANDOM);
```

26.3.2 RSI_RNG_GetBytes

Source File : rsi_rom_rng.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc

Prototype

```
STATIC INLINE void  RSI_RNG_GetBytes(HWRNG_Type *pRNG, uint32_t *randomBytes, uint32_t
numberOfBytes)
```

Description

This API is used to get the random number bytes.

Parameters

Parameters	Description
pRNG	This is the random number Generator handler
randomBytes	This is the variable or array to store generated random bytes
numberOfBytes	These are the number of bytes to generate

Return values

None

Example

```
/* Initialize output buffer*/
uint32_t random_bytes[20];
/*Get random bytes*/
RSI_RNG_GetBytes(HWRNG, random_bytes, 19);
```

26.3.3 RSI_RNG_Stop

Source File : rsi_rom_rng.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc

Prototype

```
STATIC INLINE void RSI_RNG_Stop(HWRNG_Type *pRNG)
```

Description

This API is used to stop the Random Number Generation.

Parameters

Parameters	Description

Parameters	Description
pRNG	This is the random number Generator handler

Return values

None

Example

```
/* stop RNG */
RSI_RNG_Stop(HWRNG);
```

27 General Purpose Direct Memory Access (GPDMA)

27.1 Overview

This section explains how to configure and use the General Purpose Direct Memory Access using Redpine MCU SAPIs.

27.2 Programming sequence

```
#include "rsi_rpdma.h   /* Redpine_MCU_Vx.y.z\library\rom_driver\inc */
#define  RPDMA_IRQHandler         IRQ031_Handle

/**
 * @brief  GPDMA Interrupt Handler
 * @param  None
 * @return None
 */
void GPDMA_IRQHandler(void)
{
    RSI_GPDMA_InterruptHandler(GPDMAHandle);
}

void SetupChannelDesc()
{
    uint32_t j;
    RSI_GPDMA_DESC_T XferCfg;
    RSI_GPDMA_DESC_T *pPrevDesc;
    RSI_GPDMA_CHA_CFG_T chaCfg;

   chaCfg.channelPrio = 3;
   chaCfg.descFetchDoneIntr = 0;
   chaCfg.hrespErr = 1;
   chaCfg.gpdmacErr = 1;
   chaCfg.xferDoneIntr = 1;
   chaCfg.dmaCh = 0;

   /* Setup channel 0 for trigger operation and M2M transfer */
   if (RSI_GPDMA_SetupChannel(GPDMAHandle, (RSI_GPDMA_CHA_CFG_T *) &chaCfg) != RSI_OK)
   {
      DEBUGOUT("Error setting up channel\r\n");
   }

   /* Channel Control Config */
   XferCfg.chnlCtrlConfig.transSize = XFERSIZE/NUMGPDMADESC;
   XferCfg.chnlCtrlConfig.transType = MEMORY_MEMORY;
   XferCfg.chnlCtrlConfig.dmaFlwCtrl = DMA_FLW_CTRL;
   XferCfg.chnlCtrlConfig.mastrIfFetchSel = 1;
   XferCfg.chnlCtrlConfig.mastrIfSendSel = 0;
   XferCfg.chnlCtrlConfig.destDataWidth = DST_32_DATA_WIDTH;
   XferCfg.chnlCtrlConfig.srcDataWidth = SRC_32_DATA_WIDTH;
   XferCfg.chnlCtrlConfig.srcAlign = 0;
   XferCfg.chnlCtrlConfig.linkListOn = 1;
   XferCfg.chnlCtrlConfig.linkListMstrSel = 0;
   XferCfg.chnlCtrlConfig.srcAddContiguous = 1;
   XferCfg.chnlCtrlConfig.dstAddContiguous = 1;
   XferCfg.chnlCtrlConfig.retryOnErr = 0;
   XferCfg.chnlCtrlConfig.linkInterrupt = 1;
   XferCfg.chnlCtrlConfig.srcFifoMode = 0;
   XferCfg.chnlCtrlConfig.dstFifoMode = 0;

   /* Misc Channel Config */
   XferCfg.miscChnlCtrlConfig.ahbBurstSize = AHBBURST_SIZE_16;
   XferCfg.miscChnlCtrlConfig.destDataBurst = DST_BURST_SIZE_16;
   XferCfg.miscChnlCtrlConfig.srcDataBurst = DST_BURST_SIZE_16;
   XferCfg.miscChnlCtrlConfig.destChannelId = 0;
   XferCfg.miscChnlCtrlConfig.srcChannelId = 0;
   XferCfg.miscChnlCtrlConfig.dmaProt = 0;
   XferCfg.miscChnlCtrlConfig.memoryFillEn = 0;
```

```
   XferCfg.miscChnlCtrlConfig.memoryOneFill = 0;

   pPrevDesc = NULL;
   for(j = 0; j < NUMGPDMADESC; j++)
   {
    XferCfg.src  = &src[(XFERSIZE/NUMGPDMADESC) * j];
    XferCfg.dest = &dst[(XFERSIZE/NUMGPDMADESC) * j];

    if(RSI_GPDMA_BuildDescriptors(GPDMAHandle,&XferCfg,&GPDMADesc[j],pPrevDesc)!= RSI_OK)
    {
      DEBUGOUT("Error building descriptor chain (single link)\r\n");
    }
    pPrevDesc = &GPDMADesc[j];
   }
    RSI_GPDMA_SetupChannelTransfer( GPDMAHandle,GPDMA_CHNL0, GPDMADesc);
}

int main()
{
    uint32_t memSize, *devMem;
    uint32_t i;
    RSI_GPDMA_INIT_T GPDMAInit;
    volatile bool done;
    /*Configures the system default clock and power configurations*/
    SystemCoreClockUpdate();

    dmaDone = false;

    /* Initialize UART for debug prints*/
    DEBUGINIT();

    /*  Prints on hyper-terminal  */
    DEBUGOUT("GPDMA Memory to Memory data transfer example\r\n");

    /* clear stack structures before use */
    memset(&GPDMAInit, 0, sizeof(RSI_GPDMA_INIT_T));

    /* Get needed size for driver context memory */
    memSize = RSI_GPDMA_GetMemSize();
    if (memSize > sizeof(memBuff))
    {
        /*  Prints on hyper-terminal  */
        DEBUGOUT("Can't allocate memory for driver context\r\n");
    }
    devMem = memBuff;    /* Or just use malloc(memSize) */

    /* Initialize driver */
    GPDMAInit.pUserData = (void *) &done;
    GPDMAInit.baseC = (uint32_t) GPDMA_C;
    GPDMAInit.baseG = (uint32_t) GPDMA_G;
    GPDMAInit.sramBase = (uint32_t) &GPDMADesc;

    /* Initialize driver context parameters*/
    GPDMAHandle = RSI_GPDMA_Init(devMem, &GPDMAInit);
    if (GPDMAHandle == NULL)
    {
        /*  Prints on hyper-terminal  */
        DEBUGOUT("Error initializing GPDMA\r\n");
    }
    /* GPDMA clock enable */
    RSI_CLK_PeripheralClkEnable(M4CLK,RPDMA_CLK,ENABLE_STATIC_CLK);

    /* Enable the interrupt for the DMA controller */
    NVIC_EnableIRQ(GPDMA_IRQn);

    /* Register error, descriptor completion, and descriptor chain completion callbacks
for channel 0 */
    RSI_GPDMA_RegisterCallback(GPDMAHandle, RSI_GPDMA_XFERCOMPLETE_CB, (void *)
GPDMATransferComplete);
    RSI_GPDMA_RegisterCallback(GPDMAHandle, RSI_GPDMA_XFERDESCFETCHCOMPLETE_CB, (void *)
GPDMATransferDescFetchComplete);
```

```
    RSI_GPDMA_RegisterCallback(GPDMAHandle, RSI_GPDMA_XFERHRESPERROR_CB, (void *)
GPDMATransferError);
    RSI_GPDMA_RegisterCallback(GPDMAHandle, RSI_GPDMA_XFERGPDMACERROR_CB, (void *)
GPDMATransferError);

    /* Populate some data to copy */
    for (i = 0; i < TRNSFER_LEN; i++)
    {
        src[i] = i + 1;
    }
    /* Setup descriptor */
    SetupChannelDesc();

    /* Trigger channel */
    RSI_GPDMA_DMAChannelTrigger( GPDMAHandle,GPDMA_CHNL0);

    /* Wait for DMA to fire indicating completion */
    __WFI();

    /* Compare source and destination buffers */
    ret = memcmp(src, dst, XFERSIZE);

    if(ret)
    {
        /*  Prints on hyper-terminal  */
      DEBUGOUT("Data comparison fail\r\n");
    }
    else
    {
        /*  Prints on hyper-terminal  */
      DEBUGOUT("Data comparison success\r\n");
    }
    while(1);
}
```

Code Block 20 Redpine Direct Memory Access Example

27.3 API Descriptions

27.3.1 RSI_GPDMA_GetMemSize

Source File : rsi_rom_gpdma.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc

Prototype

```
STATIC INLINE uint32_t  RSI_GPDMA_GetMemSize(void)
```

Description

Get memory size in bytes needed for GPDMA controller driver context.

Parameter

None

Return values

Size in bytes.

Example

```
uint32_t size;
/* Get memory size */
size=RSI_GPDMA_GetMemSize();
```

27.3.2 RSI_GPDMA_Init

Source File : rsi_rom_gpdma.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc

Prototype

```
STATIC INLINE RSI_GPDMA_HANDLE_T RSI_GPDMA_Init(void *mem, const RSI_GPDMA_INIT_T *pInit)
```

Description

This API is used to initialize driver context parameters.

Parameters

Parameter	Description
mem	This is the pointer to the memory area used to driver context
pInit	This is the pointer to DMA controller init data

Return values

NULL on error, or a pointer to the device context handle

Example

```
#define  NUMGPDMADESC        4
static RSI_GPDMA_DESC_T GPDMADesc[NUMGPDMADESC];
uint32_t  *devMem-0;

/* Initialization structure variable */
RSI_GPDMA_INIT_T GPDMAInit;

/* Initialize driver */
GPDMAInit.pUserData = (void *) &done;
GPDMAInit.baseC = (uint32_t) GPDMA_C;
GPDMAInit.baseG = (uint32_t) GPDMA_G;
GPDMAInit.sramBase = (uint32_t) &GPDMADesc;

/* Initialize driver context parameters*/
GPDMAHandle = RSI_GPDMA_Init(devMem, &GPDMAInit);
```

27.3.3 RSI_GPDMA_RegisterCallback

Source File : rsi_rom_gpdma.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc

Prototype

```
STATIC INLINE void  RSI_GPDMA_RegisterCallback( RSI_GPDMA_HANDLE_T pHandle,uint32_t
cbIndex,void *pCB )
```

Description

This API is used to register callbacks.

Parameters

Parameter	Description
pHandle	This is the pointer to memory area used to driver context
cbIndex	This is an index for various callback pointers
pCB	This is the pointer to callback function

Return values

None

Example

```
/* declaration */
RSI_GPDMA_HANDLE_T GPDMAHandle;

/**
 * @brief   GPDMA controller transfer descriptor chain complete callback
 * @param[in]   GPDMAHandle structure variable to driver context handle
 * @param[in]   pTranDesc   Pointer to transfer descriptor
 * @return  None
 */
void GPDMATransferComplete(RSI_GPDMA_HANDLE_T GPDMAHandle, RSI_GPDMA_DESC_T *pTranDesc)
{}

/* Register error, descriptor completion, and descriptor chain completion callbacks for
channel 0 */
RSI_GPDMA_RegisterCallback(GPDMAHandle, RSI_GPDMA_XFERCOMPLETE_CB, (void *)
GPDMATransferComplete);
```

27.3.4 RSI_GPDMA_FIFOConfig

Source File : rsi_rpdma.h

Path : Redpine_MCU_V1.x.x\Host_MCU\Peripheral_Library\driver\inc

Prototype

```
STATIC INLINE void RSI_GPDMA_FIFOConfig( RSI_GPDMA_HANDLE_T pHandle, uint8_t dmaCh,
uint32_t startAdr,uint32_t size )
```

Description

Set fifo configuration for data transmission.

Parameters

Parameter	Description
pHandle	This is the pointer to driver context handle
dmaCh	This is the DMA channel number(0-7)
startAdr	This is the starting address for data transfer.
size	This is the size of data transfer.

Return values

None

Example

```
/*declaration*/
```

```
RSI_GPDMA_HANDLE_T GPDMAHandle;
/* fifo configuration */
RSI_RPDMA_FIFOConfig( GPDMAHandle,3,48,16);
```

27.3.5 RSI_GPDMA_SetupChannel

Source File : rsi_rom_gpdma.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc

Prototype

```
STATIC INLINE error_t RSI_GPDMA_SetupChannel(RSI_GPDMA_HANDLE_T pHandle,
RSI_GPDMA_CHA_CFG_T *pCfg)
```

Description

This API is used to configures required parameters for a channel.

Parameters

Parameter	Description
pHandle	This is the pointer to memory area used to driver context
pCfg	RSI_GPDMA_CHA_CFG_T Pointer to DMA channel configuration structure

Return values

ERROR_GPDMA_CHNL_BUSY : If selected channel is busy

ERROR_GPDMA_INVALID_ARG : If channel number is invalid

RSI_OK : If process is done successfully

Example

```
/* declaration */
RSI_GPDMA_HANDLE_T GPDMAHandle;
RSI_GPDMA_CHA_CFG_T chaCfg;
/*Initialize channel configuration structure */
chaCfg.channelPrio = 3;
chaCfg.descFetchDoneIntr = 0;
chaCfg.hrespErr = 1;
chaCfg.gpdmacErr = 1;
chaCfg.xferDoneIntr = 1;
chaCfg.dmaCh = 0;
/* Setup channel 0 for trigger operation and M2M transfer */
RSI_GPDMA_SetupChannel(GPDMAHandle, (RSI_RPDMA_CHA_CFG_T *) &chaCfg);
```

27.3.6 RSI_GPDMA_BuildDescriptors

Source File : rsi_rom_gpdma.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc

Prototype

```
STATIC INLINE error_t RSI_GPDMA_BuildDescriptors(RSI_GPDMA_HANDLE_T
pHandle,RSI_GPDMA_DESC_T   *pXferCfg,RSI_GPDMA_DESC_T   *pDesc,
                                                RSI_GPDMA_DESC_T   *pDescPrev )
```

Description

Sets and build the descriptor for data transfer..

Parameters

Parameter	Description
pHandle	This is the pointer to memory area used to driver context
pXferCfg	In this structure define channel configuration structure pointer,define all structure member
pDesc	This is the GPDMA descriptor structure pointer
pDescPrev	This is the previous GPDMA descriptor.

Return values

ERROR_GPDMA_INVALID_ARG : If any descriptor parameter is invalid

RSI_OK : If process is done successfully

Example

```
/* declaration of structure */
RSI_GPDMA_HANDLE_T GPDMAHandle;
RSI_GPDMA_DESC_T XferCfg;
RSI_GPDMA_DESC_T *pPrevDesc;
static RSI_GPDMA_DESC_T GPDMADesc[4];

/* Channel Control Config */
XferCfg.chnlCtrlConfig.transSize = XFERSIZE/NUMRPDMADESC;
XferCfg.chnlCtrlConfig.transType = MEMORY_MEMORY;
XferCfg.chnlCtrlConfig.dmaFlwCtrl = DMA_FLW_CTRL;
XferCfg.chnlCtrlConfig.mastrIfFetchSel = 1;
XferCfg.chnlCtrlConfig.mastrIfSendSel = 0;
XferCfg.chnlCtrlConfig.destDataWidth = DST_32_DATA_WIDTH;
XferCfg.chnlCtrlConfig.srcDataWidth = SRC_32_DATA_WIDTH;
XferCfg.chnlCtrlConfig.srcAlign = 0;
XferCfg.chnlCtrlConfig.linkListOn = 1;
XferCfg.chnlCtrlConfig.linkListMstrSel = 0;
XferCfg.chnlCtrlConfig.srcAddContiguous = 1;
XferCfg.chnlCtrlConfig.dstAddContiguous = 1;
XferCfg.chnlCtrlConfig.retryOnErr = 0;
XferCfg.chnlCtrlConfig.linkInterrupt = 1;
XferCfg.chnlCtrlConfig.srcFifoMode = 0;
XferCfg.chnlCtrlConfig.dstFifoMode = 0;

/* Misc Channel Config */
XferCfg.miscChnlCtrlConfig.ahbBurstSize = AHBBURST_SIZE_16;
XferCfg.miscChnlCtrlConfig.destDataBurst = DST_BURST_SIZE_16;
XferCfg.miscChnlCtrlConfig.srcDataBurst = DST_BURST_SIZE_16;
XferCfg.miscChnlCtrlConfig.destChannelId = 0;
XferCfg.miscChnlCtrlConfig.srcChannelId = 0;
XferCfg.miscChnlCtrlConfig.dmaProt = 0;
XferCfg.miscChnlCtrlConfig.memoryFillEn = 0;
XferCfg.miscChnlCtrlConfig.memoryOneFill = 0;

/* Build Descriptor of channel for M2M transfer */
RSI_RPDMA_BuildDescriptors(GPDMAHandle,&XferCfg,&GPDMADesc[j],pPrevDesc);
```

27.3.7 RSI_GPDMA_SetupChannelTransfer

Source File : rsi_rom_gpdma.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc

Prototype

```
STATIC INLINE error_t RSI_GPDMA_SetupChannelTransfer( RSI_GPDMA_HANDLE_T pHandle,uint8_t
dmaCh,RSI_GPDMA_DESC_T *pDesc )
```

Description

Sets the control parameters for a GPDMA channel control structure.

Parameters

Parameter	Description
pHandle	This is the pointer to the memory area used to driver context
dmaCh	This is the DMA channel number(0-7)
pDesc	This is the GPDMA descriptor structure pointer

Return values

ERROR_GPDMA_INVALID_TRANS_LEN : If transSize parameter of descriptor is invalid

ERROR_GPDMA_INVALID_XFERMODE : If transType parameter of descriptor is invalid

ERROR_GPDMA_FLW_CTRL : If dmaFlwCtrl parameter of descriptor is invalid

ERROR_GPDMA_INVALIDCHNLNUM : If DMA channel number is invalid

ERROR_GPDMA_INVALID_ARG : If any other descriptor parameter is invalid

RSI_OK : If process is done successfully

Example

```
/* declaration of structure */
#define  NUMGPDMADESC        4
static RSI_GPDMA_DESC_T GPDMADesc[NUMGPDMADESC];
RSI_GPDMA_HANDLE_T GPDMAHandle;
/* channel control structure of channel 0 for M2M transfer */
RSI_RPDMA_SetupChannelTransfer( GPDMAHandle,0, GPDMADesc);
```

27.3.8 RSI_GPDMA_InterruptEnable

Source File : rsi_rom_gpdma.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc

Prototype

```
STATIC INLINE error_t RSI_GPDMA_InterruptEnable( RSI_GPDMA_HANDLE_T
pHandle,RSI_GPDMA_CHA_CFG_T *pCfg )
```

Description

This API is used to enable interrupt flags for required GPDMAchannel.

Parameters

Parameter	Description
pHandle	This is the pointer to memory area used to driver context

Parameter	Description
pCfg	This is the pointer to DMA channel configuration structure

Return values

ERROR_GPDMA_INVALIDCHNLNUM : If DMA channel number is invalid

RSI_OK : If process is done successfully

Example

```
/* declaration */
RSI_GPDMA_HANDLE_T GPDMAHandle;
RSI_GPDMA_CHA_CFG_T chaCfg;
/*Initialize channel configuration structure */
chaCfg.channelPrio = 3;
chaCfg.descFetchDoneIntr = 0;
chaCfg.hrespErr = 1;
chaCfg.gpdmacErr = 1;
chaCfg.xferDoneIntr = 1;
chaCfg.dmaCh = 0;
/* Enable interrupt */
RSI_GPDMA_InterruptEnable(GPDMAHandle,&chaCfg);
```

27.3.9 RSI_GPDMA_DMAChannelTrigger

Source File : rsi_rom_gpdma.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc

Prototype

```
STATIC INLINE error_t RSI_GPDMA_DMAChannelTrigger( RSI_GPDMA_HANDLE_T pHandle ,uint8_t
dmaCh)
```

Description

This API is used to enable the required channel of RPDMA.

Parameters

Parameter	Description
pHandle	This is the pointer to memory area used to driver context
dmaCh	This is the channel number (0 to 7)

Return values

ERROR_GPDMA_INVALIDCHNLNUM : If DMA channel number is invalid

RSI_OK : If process is done successfully

Example

```
/* declaration */
RSI_GPDMA_HANDLE_T GPDMAHandle;
/* Software trigger of DMA channel 0*/
RSI_RPDMA_DMAChannelTrigger( GPDMAHandle,0);
```

27.3.10 RSI_GPDMA_InterruptHandler

Source File : rsi_rom_gpdma.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc

Prototype

```
STATIC INLINE void  RSI_GPDMA_InterruptHandler(RSI_GPDMA_HANDLE_T pHandle)
```

Description

This API is used to handle DMA interrupts for all channels.

Parameters

Parameter	Description
pHandle	This is the pointer to memory area used to driver context

Return values

None

Example

```
/* declaration */
RSI_GPDMA_HANDLE_T GPDMAHandle;
/* handle the interrupt */
RSI_GPDMA_InterruptHandler(GPDMAHandle);
```

27.3.11 RSI_RPDMA_InterruptStatus

Source File : rsi_rom_gpdma.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc

Prototype

```
STATIC INLINE uint32_t  RSI_GPDMA_InterruptStatus( RSI_GPDMA_HANDLE_T
pHandle,RSI_GPDMA_CHA_CFG_T *pCfg )
```

Description

This API is used to get the interrupt status of required GPDMA channel.

Parameters

Parameter	Description
pHandle	This is the pointer to memory area used to driver context
pCfg	This is the pointer to DMA channel configuration structure

Return values

Returns the interrupt status of any one of the below flags

LINK_LIST_DONE : Next link list pointer points to zero

PHRL_END_OF_TFR : Indicates last burst or single transfer

ERROR_GPDMA_INVALIDCHNLNUM : If DMA channel number is invalid

RSI_OK : If process is done successfully

Example

```
/* declaration */
RSI_GPDMA_HANDLE_T GPDMAHandle;
RSI_GPDMA_CHA_CFG_T chaCfg;
/*Initialize channel configuration structure */
```

```
chaCfg.channelPrio = 3;
chaCfg.descFetchDoneIntr = 0;
chaCfg.hrespErr = 1;
chaCfg.gpdmacErr = 1;
chaCfg.xferDoneIntr = 1;
chaCfg.dmaCh = 0;
/*Get interrupt Status*/
RSI_GPDMA_InterruptStatus(GPDMAHandle,&chaCfg );
```

27.3.12 RSI_GPDMA_InterruptClear

Source File : rsi_rom_gpdma.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc

Prototype

```
STATIC INLINE error_t RSI_GPDMA_InterruptClear( RSI_GPDMA_HANDLE_T
pHandle,RSI_GPDMA_CHA_CFG_T *pCfg )
```

Description

This API is used to clear the interrupts of required GPDMA channel.

Parameters

Parameter	Description
pHandle	This is the pointer to memory area used to driver context
pCfg	This is the pointer to DMA channel configuration structure

Return values

ERROR_GPDMA_INVALIDCHNLNUM : If DMA channel number is invalid

RSI_OK : If process is done successfully

Example

```
/* declaration */
RSI_GPDMA_HANDLE_T GPDMAHandle;
RSI_GPDMA_CHA_CFG_T chaCfg;
/*Initialize channel configuration structure */
chaCfg.channelPrio = 3;
chaCfg.descFetchDoneIntr = 0;
chaCfg.hrespErr = 1;
chaCfg.gpdmacErr = 1;
chaCfg.xferDoneIntr = 1;
chaCfg.dmaCh = 0;
/*Clear interrupt */
RSI_GPDMA_InterruptClear(GPDMAHandle,&chaCfg);
```

27.3.13 RSI_GPDMA_InterruptDisable

Source File : rsi_rom_gpdma.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc

Prototype

```
STATIC INLINE error_t RSI_GPDMA_InterruptDisable( RSI_GPDMA_HANDLE_T
pHandle,RSI_GPDMA_CHA_CFG_T *pCfg )
```

Description

This API is used to disable the interrupt for required GPDMA channel.

Parameters

Parameter	Description
pHandle	This is the pointer to memory area used to driver context
pCfg	This is the pointer to DMA channel configuration structure #RSI_GPDMA_CHA_CFG_T

Return values

ERROR_GPDMA_INVALIDCHNLNUM : If DMA channel number is invalid

RSI_OK : If process is done successfully

Example

```
/* declaration */
RSI_GPDMA_HANDLE_T GPDMAHandle;
RSI_GPDMA_CHA_CFG_T chaCfg;
/*Initialize channel configuration structure */
chaCfg.channelPrio = 3;
chaCfg.descFetchDoneIntr = 0;
chaCfg.hrespErr = 1;
chaCfg.gpdmacErr = 1;
chaCfg.xferDoneIntr = 1;
chaCfg.dmaCh = 0;
/*Disable interrupt */
RSI_GPDMA_InterruptDisable(rpdmaHandle,&chaCfg );
```

27.3.14 RSI_GPDMA_ErrorStatusClear

Source File : rsi_rom_gpdma.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc

Prototype

```
STATIC INLINE error_t RSI_GPDMA_ErrorStatusClear( RSI_GPDMA_HANDLE_T
pHandle,RSI_GPDMA_CHA_CFG_T *pCfg )
```

Description

This API is used to clear the errors for the required GPDMA channel.

Parameters

Parameter	Description
pHandle	This is the pointer to memory area used to driver context
pCfg	This is the pointer to DMA channel configuration structure

Return values

ERROR_GPDMA_INVALIDCHNLNUM : If DMA channel number is invalid

RSI_OK : If process is done successfully

Example

```
/* declaration */
RSI_GPDMA_HANDLE_T GPDMAHandle;
RSI_GPDMA_CHA_CFG_T chaCfg;
/*Initialize channel configuration structure */
chaCfg.channelPrio = 3;
chaCfg.descFetchDoneIntr = 0;
chaCfg.hrespErr = 1;
chaCfg.gpdmacErr = 1;
chaCfg.xferDoneIntr = 1;
chaCfg.dmaCh = 0;
/*Clear error status in DMA channel */
RSI_GPDMA_ErrorStatusClear(GPDMAHandle,&chaCfg);
```

27.3.15 RSI_GPDMA_GetErrorStatus

Source File : rsi_rom_gpdma.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc

Prototype

```
STATIC INLINE uint32_t  RSI_GPDMA_GetErrorStatus( RSI_GPDMA_HANDLE_T
pHandle,RSI_GPDMA_CHA_CFG_T *pCfg )
```

Description

This API is used to get the error status of required GPDMA channel.

Parameters

Parameter	Description
pHandle	This is the pointer to memory area used to driver context
pCfg	This is the pointer to DMA channel configuration structure

Return values

Return error status of either HRESP_ERR(0) or GPDMAC_ERR(1)

Example

```
/* declaration */
uint32_t status;
RSI_GPDMA_HANDLE_T GPDMAHandle;
RSI_GPDMA_CHA_CFG_T chaCfg;
/*Initialize channel configuration structure */
chaCfg.channelPrio = 3;
chaCfg.descFetchDoneIntr = 0;
chaCfg.hrespErr = 1;
chaCfg.gpdmacErr = 1;
chaCfg.xferDoneIntr = 1;
chaCfg.dmaCh = 0;
/*Get error status */
status = RSI_GPDMA_GetErrorStatus(GPDMAHandle,&chaCfg);
```

27.3.16 RSI_GPDMA_ChannelIsEnabled

Source File : rsi_rom_gpdma.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc

Prototype

```
STATIC INLINE uint32_t  RSI_GPDMA_ChannelIsEnabled( RSI_GPDMA_HANDLE_T pHandle ,uint8_t
dmaCh )
```

Description

This API is used to check the required GPDMAchannel is enabled or not.

Parameters

Parameter	Description
pHandle	This is the pointer to memory area used to driver context
dmaCh	This is the channel number(0 to 7)

Return values

ERROR_GPDMA_INVALIDCHNLNUM : If DMA channel number is invalid

RSI_OK : If process is done successfully

Example

```
/* declaration */
RSI_GPDMA_HANDLE_T GPDMAHandle;
/*channel enable or not? */
RSI_GPDMA_ChannelIsEnabled(GPDMAHandle,0);
```

27.3.17 RSI_GPDMA_AbortChannel

Source File : rsi_rom_gpdma.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc

Prototype

```
STATIC INLINE error_t RSI_GPDMA_AbortChannel( RSI_GPDMA_HANDLE_T pHandle,uint8_t dmaCh )
```

Description

This API is used to Abort the channel transfer.

Parameters

Parameter	Description
pHandle	This is the pointer to memory area used to driver context
dmaCh	This is the DMA channel number (0-7)

Return values

ReturnS zero on success ,on failure return error code

Example

```
/* declaration */
RSI_GPDMA_HANDLE_T GPDMAHandle;
/* Abort channel 0  */
RSI_RPDMA_AbortChannel(GPDMAHandle,0);
```

27.3.18 RSI_RPDMA_GetCapabilities

Source File : rsi_rpdma.c

Path : Redpine_MCU_V1.x.x\Host_MCU\Peripheral_Library\driver\src

Prototype

```
RSI_GPDMA_CAPABILITIES_T RSI_GPDMA_GetCapabilities(void)
```

Description

This API get the capabilities of the RPDMA

Parameter

None

Return values

GPDMA capabilities structure

Example

```
/*declaration*/
RSI_GPDMA_CAPABILITIES_T vsDriverCapabilities;
/* get driver capability */
vsDriverCapabilities=RSI_GPDMA_GetCapabilities();
```

27.3.19 RSI_GPDMA_GetVersion

Source File : rsi_rpdma.c

Path : Redpine_MCU_V1.x.x\Host_MCU\Peripheral_Library\driver\scr

Prototype

```
RSI_DRIVER_VERSION RSI_GPDMA_GetVersion(void)
```

Description

Set fifo configuration for data transmission.

Parameter

None

Return values

structure of type RSI_DRIVER_VERSION and its members are as below

- RSI_GPDMA_API_VERSION : Version of the CMSIS-Driver specification used to implement this driver.
- RSI_GPDMA_DRV_VERSION : GPDMA peripheral source code version of the actual driver implementation.

Example

```
/*declaration*/
RSI_DRIVER_VERSION vsDriverVersion;
/* get driver version */
vsDriverVersion=RSI_GPDMA_GetVersion();
```

27.3.20 RSI_GPDMA_DeInit

Source File : rsi_rom_gpdma.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc

Prototype

```
STATIC INLINE void  RSI_GPDMA_DeInit( RSI_GPDMA_HANDLE_T pHandle,RSI_GPDMA_CHA_CFG_T *pCfg
)
```

Description

This API is used to Uninitialize driver context parameters.

Parameters

Parameter	Description
pHandle	This is the pointer to memory area used to driver context
pCfg	This is the pointer to DMA channel configuration structure

Return values

None

Example

```
/* declaration */
uint32_t status;
RSI_GPDMA_HANDLE_T GPDMAHandle;
RSI_GPDMA_CHA_CFG_T chaCfg;
/*Initialize channel configuration structure */
chaCfg.channelPrio = 3;
chaCfg.descFetchDoneIntr = 0;
chaCfg.hrespErr = 1;
chaCfg.gpdmacErr = 1;
chaCfg.xferDoneIntr = 1;
chaCfg.dmaCh = 0;
/* De-initialization channel configuration */
RSI_GPDMA_DeInit( GPDMAHandle,&chaCfg);
```

28 Serial Input Output (SIO)

28.1 Overview

This section explains how to configure and use the Serial Input Output using Redpine MCU SAPIs.

28.2 Programming Sequence:

```
#include "rsi_chip.h" /* Redpine_MCU_Vx.y.z\Host_MCU\common\chip\inc */

#define SPI_BUFFSENDSIZE 255
/*Global buffers to hold the TX and RX data */
uint16_t u16SpiTxBuf[SPI_BUFFSENDSIZE] , u16SpiRxBuf[SPI_BUFFSENDSIZE];
uint32_t spi_tx_dne , spi_rx_dne ;

/*Configuration and Transfer structure*/
stc_sio_spi_cfg_t  pstcSpiConfig ={0};
stc_sio_spi_xfer_t pstcSpiXfer ={0};

/* Required Pin MUX */
static void RSI_SIO_PinInit(void)
{
    RSI_EGPIO_PadSelectionEnable(1);
    RSI_EGPIO_PadReceiverEnable(6);
    RSI_EGPIO_SetPinMux(EGPIO ,0 , 6 ,  EGPIO_PIN_MUX_MODE1);  // SIO 0
    RSI_EGPIO_PadSelectionEnable(1);
    RSI_EGPIO_PadReceiverEnable(7);
    RSI_EGPIO_SetPinMux(EGPIO ,0 , 7 , EGPIO_PIN_MUX_MODE1);  // SIO 1
    RSI_EGPIO_PadSelectionEnable(1);
    RSI_EGPIO_PadReceiverEnable(8);
    RSI_EGPIO_SetPinMux(EGPIO ,0 , 8 , EGPIO_PIN_MUX_MODE1);  // SIO 2
    RSI_EGPIO_PadSelectionEnable(1);
    RSI_EGPIO_PadReceiverEnable(9);
    RSI_EGPIO_SetPinMux(EGPIO ,0 , 9 ,  EGPIO_PIN_MUX_MODE1);  // SIO 3

    RSI_EGPIO_PadSelectionEnable(21);
    RSI_EGPIO_PadReceiverEnable(76);
    RSI_EGPIO_SetPinMux(EGPIO ,0 , 76 , EGPIO_PIN_MUX_MODE1);  // SIO 4
    RSI_EGPIO_UlpPadReceiverEnable(12);
    RSI_EGPIO_SetPinMux(EGPIO1 ,0 , 12 , EGPIO_PIN_MUX_MODE6);  // SIO 4

    RSI_EGPIO_PadSelectionEnable(21);
    RSI_EGPIO_PadReceiverEnable(77);
    RSI_EGPIO_SetPinMux(EGPIO ,0 , 77 , EGPIO_PIN_MUX_MODE1);  // SIO 5
    RSI_EGPIO_UlpPadReceiverEnable(13);
    RSI_EGPIO_SetPinMux(EGPIO1 ,0 , 13 , EGPIO_PIN_MUX_MODE6);  // SIO 5

    RSI_EGPIO_PadSelectionEnable(21);
    RSI_EGPIO_PadReceiverEnable(78);
    RSI_EGPIO_SetPinMux(EGPIO ,0 , 78 , EGPIO_PIN_MUX_MODE1);  // SIO 6
    RSI_EGPIO_UlpPadReceiverEnable(14);
    RSI_EGPIO_SetPinMux(EGPIO1 ,0 , 14 , EGPIO_PIN_MUX_MODE6);  // SIO 6

    RSI_EGPIO_PadSelectionEnable(21);
    RSI_EGPIO_PadReceiverEnable(79);
    RSI_EGPIO_SetPinMux(EGPIO ,0 , 79 , EGPIO_PIN_MUX_MODE1);  // SIO 7
    RSI_EGPIO_UlpPadReceiverEnable(15);
    RSI_EGPIO_SetPinMux(EGPIO1 ,0 , 15 , EGPIO_PIN_MUX_MODE6);  // SIO 7
    return ;
}
int main(void)
{
   int  cnt = 0;
#ifdef SIO_SPI_CASE
    /*SPI configuration*/
    pstcSpiConfig.u32SpiClockFrq = SIO_SPI_CLK_FREQUENCY;
```

```
    /*Default MSB First */
    pstcSpiConfig.u8BitOrder      = 1;
    pstcSpiConfig.u8SpiClkCh      = SIO_SPI_CLK_CH ;
    pstcSpiConfig.u8SpiMosiCh     = SIO_SPI_MOSI_CH;
    pstcSpiConfig.u8SpiMisoCh     = SIO_SPI_MISO_CH;
    pstcSpiConfig.u8SpiCsCh       = SIO_SPI_CS_CH;
    pstcSpiConfig.u8BitLen        = SIO_SPI_BIT_LEN;
    pstcSpiConfig.u8Mode          = SIO_SPI_MODE;

    /* Enable the SIO pin muxing*/
    RSI_SIO_PinInit();

    /* Enable the SIO*/
    RSI_SIO_Init(SIO);

    /*Initialise the SIO-SPI */
    RSI_SIO_InitSpi(SIO , &pstcSpiConfig);

    /*Enable the SIO NIV Interrupt */
    NVIC_EnableIRQ(SIO_IRQn);

    while(1)
    {
        /* Populate some TX data and clear RX data */
        for (cnt = 0; cnt < SPI_BUFFSENDSIZE; cnt++) {
            u16SpiTxBuf[cnt] = cnt + 1 + ((cnt + 1) << 8);
            u16SpiRxBuf[cnt] =  0;
        }

        /*SPI Transfer configuration */
        pstcSpiXfer.rxBuff            = u16SpiRxBuf;
        pstcSpiXfer.rxCount           = SPI_BUFFSENDSIZE ;
        pstcSpiXfer.sselNum           = 0;
        pstcSpiXfer.txBuff            = u16SpiTxBuf;
        pstcSpiXfer.txCount           = SPI_BUFFSENDSIZE ;
        pstcSpiXfer.u8BitLen          = SIO_SPI_BIT_LEN;
        pstcSpiXfer.u8Status          = SioSpiIdle;
        pstcSpiXfer.pfnCb             = SioSpiCbFn;


        RSI_SIO_SpiCsAssert(SIO,pstcSpiConfig.u8SpiCsCh);

        RSI_SIO_SpiTrasnfer(SIO , &pstcSpiXfer);

        /* Status will change from BUSY once transfer is complete */
    while (pstcSpiXfer.u8Status == SioSpiBusy) {}

        RSI_SIO_SpiCsDeAssert(SIO,pstcSpiConfig.u8SpiCsCh);

        cnt++;
    }

#endif

#ifdef SIO_UART_CASE

    /*local variables */
    stc_sio_uart_config_t UartInitstc = {0};
    uint8_t i = 0;
    uint8_t data=0 , rxdata =0 ;
    uint8_t u8TxDat[100] ={0};
    uint8_t u8RxDat[100] ={0};

    /*Uart configuration structure */
    UartInitstc.u32BaudRate          = 115200;
    UartInitstc.u8Bitlen             = 8 ;
    UartInitstc.u8Parity             = 0 ;
    UartInitstc.u8SioUartRxChannel   = 2;
    UartInitstc.u8SioUartTxChannel   = 3;
    UartInitstc.u8StopBits           = 1;
    UartInitstc.pfn                  = SioUartCbFn;
```

```
    /* Enable the SIO pin MUX*/
    RSI_SIO_PinInit();

    /* Enable the SIO*/
    RSI_SIO_Init(SIO);

    /*Enable the SIO NIV Interrupt */
    NVIC_EnableIRQ(SIO_IRQn);

    /*Enable SIO in UART mode */
    RSI_SIO_UartInit(SIO ,  &UartInitstc);

    /*UART TX Data*/

    /*populate some data into tx buf */
    strcpy(u8TxDat , "HELLO SIO UART WORKING AWESOME");

    while(1)
    {
        RSI_SIO_UARTSend(SIO ,  u8TxDat , strlen(u8TxDat));

        RSI_SIO_UARTReadBlocking(SIO ,u8RxDat , strlen(u8TxDat) );
    }
#endif
#ifdef SIO_I2C_CASE

    uint8_t   u8TxDat[17] = {0};
    uint8_t   u8RxDat[17] = {0} , u8val = 0 ;

    stc_sio_i2c_config_t i2cConfig = {0};

    RSI_SIO_PinInit();
    /* Enable the SIO*/
    RSI_SIO_Init(SIO);

    i2cConfig.u32SampleRate = 400000 ;
    i2cConfig.u8SioI2cOen   = 2 ;
    i2cConfig.u8SioI2cScl   = 6 ;
    i2cConfig.u8SioI2cSda   = 7 ;

    /*Prepare the I2C Write buffer */
    for(cnt = 1 ; cnt < 15 ; cnt++)
    {
        u8TxDat[cnt] = u8val++;
    }
    while(1)
    {
        /*Write memory with 10 Bytes*/
        RSI_SIO_I2cGenerateStart(SIO);
        /*Delay for read data from EEPROM */
        for(cnt = 0 ; cnt < 16 ; cnt++)
        {
        }
        RSI_SIO_I2cWrite(SIO , &i2cConfig , 0x50 , u8TxDat , 1);

        /*Initiate the EEPROM Read */
        RSI_SIO_I2cRead(SIO , &i2cConfig , 0x50 , u8RxDat , 15);
        RSI_SIO_I2cGenerateStop(SIO);

        while(1);
    }
#endif
    return 0;
}
```

Code Block 21 Serial input output Example

28.3 API Descriptions

28.3.1 RSI_SIO_Init

Source File : rsi_sio.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_SIO_Init(volatile SIO_Type *pstcSio)
```

Description

This API is used to initialize the SIO.

Parameters

Parameter	Description
pstcSio	This is the pointer to the register instance of SIO module .

Return values

RSI_OK on success and error code on failure

Example

```
/* Initialise the SIO */
RSI_SIO_Init(SIO);
```

28.3.2 RSI_SIO_InitSpi

Source File : rsi_sio.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_SIO_InitSpi (volatile SIO_Type *pstcSio, stc_sio_spi_cfg_t *pstcSpiConfig)
```

Description

This API is used to configure the SPI in SIO.

Parameters

Parameter	Description
pstcSio	This is the pointer to the register instance of SIO module .
pstcSpiConfig	This is the pointer to SIO spi configuration structure

Return values

None

Example

```
/*Configuration and Transfer structure*/
stc_sio_spi_cfg_t  pstcSpiConfig ={0};

/*SPI configuration*/
pstcSpiConfig.u32SpiClockFrq = SIO_SPI_CLK_FREQUENCY;
```

```
/*Default MSB First */
pstcSpiConfig.u8BitOrder      = 1;
pstcSpiConfig.u8SpiClkCh      = SIO_SPI_CLK_CH ;
pstcSpiConfig.u8SpiMosiCh     = SIO_SPI_MOSI_CH;
pstcSpiConfig.u8SpiMisoCh     = SIO_SPI_MISO_CH;
pstcSpiConfig.u8SpiCsCh       = SIO_SPI_CS_CH;
pstcSpiConfig.u8BitLen        = SIO_SPI_BIT_LEN;
pstcSpiConfig.u8Mode          = SIO_SPI_MODE;
/* Initialise the SIO-SPI */
RSI_SIO_InitSpi(SIO , &pstcSpiConfig);
```

28.3.3 RSI_SIO_SpiCsAssert

Source File : rsi_sio.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
void RSI_SIO_SpiCsAssert (volatile SIO_Type *pstcSio, uint8_t u8CsNo)
```

Description

This API is used to assert the SPI master chip select.

Parameters

Parameter	Description
pstcSio	This is the pointer to the register instance of SIO module .
u8CsNo	This is the chip select number (0..7)

Return values

None

Example

```
/*  used to assert the SPI master chip select.  */
RSI_SIO_SpiCsAssert(SIO,pstcSpiConfig.u8SpiCsCh);
```

28.3.4 RSI_SIO_SpiTrasnfer

Source File : rsi_sio.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_SIO_SpiTrasnfer (volatile SIO_Type *pstcSio, stc_sio_spi_xfer_t *pstcSpiXfer)
```

Description

This API is used to make the SIO-SPI Transfer in non blocking mode.

Parameters

Parameter	Description
pstcSio	This is the pointer to the register instance of SIO module

Parameter	Description
pstcSpiXfer	This is the pointer to the SIO-Transfer structure.

Return values

0: on success , Error code on failure case

Example

```
/*Configuration and Transfer structure*/
stc_sio_spi_xfer_t pstcSpiXfer ={0};
/*SPI Transfer configuration */
pstcSpiXfer.rxBuff            = u16SpiRxBuf;
pstcSpiXfer.rxCount           = SPI_BUFFSENDSIZE ;
pstcSpiXfer.sselNum           = 0;
pstcSpiXfer.txBuff            = u16SpiTxBuf;
pstcSpiXfer.txCount           = SPI_BUFFSENDSIZE ;
pstcSpiXfer.u8BitLen          = SIO_SPI_BIT_LEN;
pstcSpiXfer.u8Status          = SioSpiIdle;
pstcSpiXfer.pfnCb             = SioSpiCbFn;
/*  used to make the SIO-SPI Transfer in non blocking mode  */
RSI_SIO_SpiTrasnfer(SIO , &pstcSpiXfer);
```

28.3.5 RSI_SIO_SpiCsDeAssert

Source File : rsi_sio.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
void RSI_SIO_SpiCsDeAssert (volatile SIO_Type *pstcSio, uint8_t u8CsNo)
```

Description

This API is used to deassert the SPI master chip select.

Parameters

Parameter	Description
pstcSio	This is the pointer to the register instance of SIO module .
u8CsNo	This chip select number (0..7)

Return values

None

Example

```
/*  used to deassert the SPI master chip select.  */
RSI_SIO_SpiCsDeAssert(SIO,pstcSpiConfig.u8SpiCsCh)
```

28.3.6 RSI_SIO_UartInit

Source File : rsi_sio.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
uint8_t RSI_SIO_UartInit (SIO_Type *pstcSio, stc_sio_uart_config_t *pstcConfig)
```

Description

This API is used to Initialization of uart in sio.

Parameters

Parameter	Description
pstcSio	This is the pointer to the register instance of SIO module
pstcConfig	This is the pointer to the uart config in SIO module

Return values

Return 0 on succesful execuation.

Example

```
stc_sio_uart_config_t UartInitstc = {0};
/*Uart configuration structure */
UartInitstc.u32BaudRate          = 115200;
UartInitstc.u8Bitlen             = 8 ;
UartInitstc.u8Parity             = 0 ;
UartInitstc.u8SioUartRxChannel   = 2;
UartInitstc.u8SioUartTxChannel   = 3;
UartInitstc.u8StopBits           = 1;
UartInitstc.pfn                  = SioUartCbFn;
/*Enable SIO in UART mode */
RSI_SIO_UartInit(SIO ,  &UartInitstc);
```

28.3.7 RSI_SIO_UARTSend

Source File : rsi_sio.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
uint16_t RSI_SIO_UARTSend (SIO_Type *pstcSio, const void *u16ptr, uint16_t u16Len)
```

Description

This API is used to send the data over UART. .

Parameters

Parameter	Description
pstcSio	This is the pointer to the register instance of SIO module
u16ptr	This is the data pointer to send
u16Len	This is the data length

Return values

Zero on success

Example

```
/* declaration */
uint8_t u8TxDat[100] ={0};
/*populate some data into tx buf */
strcpy(u8TxDat , "HELLO SIO UART WORKING AWESOME");
```

```
/* used to send the data over UART.  */
RSI_SIO_UARTSend(SIO ,  u8TxDat , strlen(u8TxDat));
```

28.3.8 RSI_SIO_UARTReadBlocking

Source File : rsi_sio.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_SIO_UARTReadBlocking (volatile SIO_Type *pstcSio, void *data, int numBytes)
```

Description

This API is used to read data in UART blocking mode

Parameters

Parameter	Description
pstcSio	This is the pointer to the register instance of SIO module
data	This is the data pointer
numBytes	This is the number of bytes to send

Return values

Return zero on success.

Example

```
/* declaration */
uint8_t u8RxDat[100] ={0};
/*  used to read data in UART blocking mode */
RSI_SIO_UARTReadBlocking(SIO ,u8RxDat , strlen(u8RxDat) );
```

28.3.9 RSI_SIO_UARTSendBlocking

Source File : rsi_sio.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
int RSI_SIO_UARTSendBlocking (SIO_Type *pstcSio, const void *data, int numBytes
```

Description

This API is used to write data in UART mode.

Parameters

Parameter	Description
pstcSio	This is the pointer to the register instance of SIO module
data	This is the data pointer
numBytes	These are the number of bytes to be sent

Return values

Number of byte send.

Example

```
/* declaration */
uint8_t u8TxDat[100] ={0};
/*populate some data into tx buf */
strcpy(u8TxDat , "HELLO SIO UART WORKING AWESOME");
/* This API is used to write data in UART mode */
RSI_SIO_UARTSendBlocking(SIO ,  u8TxDat , strlen(u8TxDat))
```

28.3.10 RSI_SIO_UARTRead

Source File : rsi_sio.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_SIO_UARTRead (volatile SIO_Type *pstcSio, void *data, int numBytes)
```

Description

This API is used to read data in UART mode.

Parameters

Parameter	Description
pstcSio	This is the pointer to the register instance of SIO module
data	This is the data pointer
numBytes	These are the number of bytes to be read

Return values

Return zero on success.

Example

```
void   *data;
/* UART  data read */
RSI_SIO_UARTRead(SIO ,data,10);                /* read 1024 byte data */
```

28.3.11 RSI_SIO_I2cGenerateStart

Source File : rsi_sio.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
void RSI_SIO_I2cGenerateStart (volatile SIO_Type *pstcSio)
```

Description

This API is used to I2C generate start in sio.

Parameter

Parameter	Description

Parameter	Description
pstcSio	This is the pointer to the register instance of SIO module

Return values

None

Example

```
/* used to I2C generate start in sio */
RSI_SIO_I2cGenerateStart(SIO);
```

28.3.12 RSI_SIO_I2cTransfer

Source File : rsi_sio.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_SIO_I2cTransfer (volatile SIO_Type *pstcSio, stc_sio_i2c_config_t *pstcConfig,
uint8_t u8SlaveAddr, uint8_t *u8PtrTxDat, uint16_t u16TxLen, uint8_t *u8PtrRxDat, uint16_t
u16RxLen)
```

Description

This API is used to data transfer using i2c in sio.

Parameters

Parameter	Description
pstcSio	This is the pointer to the register instance of SIO module
pstcConfig	This is the pointer to the I2C config in SIO module
u8SlaveAddr	This is the slave address
u8PtrTxDat	This is the pointer to the data TX buffer
u16TxLen	This is the TX data length
u8PtrRxDat	This is the pointer to the data RX buffe
u16RxLen	This is the RX data length

Return values

error codes

Example

```
uint8_t   u8TxDat[17] = {0};
uint8_t   u8RxDat[17] = {0} , u8val = 0 ;
stc_sio_i2c_config_t i2cConfig = {0};

i2cConfig.u32SampleRate = 400000 ;
i2cConfig.u8SioI2cOen   = 2 ;
i2cConfig.u8SioI2cScl   = 6 ;
i2cConfig.u8SioI2cSda   = 7 ;
/* used to data transfer using i2c in sio */
RSI_SIO_I2cTransfer(SIO , &i2cConfig , 0x50 , u8TxDat , 15 , 0 , 0);
```

28.3.13 RSI_SIO_I2cWrite

Source File : rsi_sio.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_SIO_I2cWrite (volatile SIO_Type *pstcSio, stc_sio_i2c_config_t *pstcConfig,
uint8_t u8SlaveAddr, uint8_t *u8Data, uint16_t u16Len)
```

Description

This API is used to write data using i2c in sio.

Parameters

Parameter	Description
pstcSio	This is the pointer to the register instance of SIO module
pstcConfig	This is the pointer to the I2C config in SIO module
u8SlaveAddr	This is the slave address
u8Data	This is the pointer to the data
u16Len	This is the data length

Return values

error codes

Example

```
uint8_t   u8TxDat[17] = {0};
uint8_t   u8RxDat[17] = {0} , u8val = 0 ;
stc_sio_i2c_config_t i2cConfig = {0};

i2cConfig.u32SampleRate = 400000 ;
i2cConfig.u8SioI2cOen   = 2 ;
i2cConfig.u8SioI2cScl   = 6 ;
i2cConfig.u8SioI2cSda   = 7 ;

/* used to write data using i2c in sio. */
RSI_SIO_I2cWrite(SIO , &i2cConfig , 0x50 , u8TxDat , 15);
```

28.3.14 RSI_SIO_I2cRead

Source File : rsi_sio.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_SIO_I2cRead (volatile SIO_Type *pstcSio, stc_sio_i2c_config_t *pstcConfig,
uint8_t u8SlaveAddr, uint8_t *u8Data, uint16_t u16Len)
```

Description

This API is used to read data using i2c in sio.

Parameters

Parameter	Description

Parameter	Description
pstcSio	This is the pointer to the register instance of SIO module
pstcConfig	This is the pointer to the I2C config in SIO module
u8SlaveAddr	This is the slave address
u8Data	This is the pointer to the data
u16Len	This is the data length

Return values

error codes

Example

```
uint8_t   u8TxDat[17] = {0};
uint8_t   u8RxDat[17] = {0} , u8val = 0 ;
stc_sio_i2c_config_t i2cConfig = {0};

i2cConfig.u32SampleRate = 400000 ;
i2cConfig.u8SioI2cOen   = 2 ;
i2cConfig.u8SioI2cScl   = 6 ;
i2cConfig.u8SioI2cSda   = 7 ;

/*Initiate the EEPROM Read */
RSI_SIO_I2cRead(SIO , &i2cConfig , 0x50 , u8RxDat , 15);
```

28.3.15 RSI_SIO_I2cGenerateStop

Source File : rsi_sio.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
void RSI_SIO_I2cGenerateStop (volatile SIO_Type *pstcSio)
```

Description

This API is used to I2C generate stop in sio.

Parameter

Parameter	Description
pstcSio	This is the pointer to the register instance of SIO module .

Return values

None

Example

```
/* used to I2C generate stop in sio */
RSI_SIO_I2cGenerateStop(SIO);
```

29 Timer

29.1 Overview

This section explains how to configure and use the Timers using Redpine MCU SAPIs.

29.2 Programming sequence

```
#include "rsi_chip.h"       /* Redpine_MCU_Vx.y.z\Host_MCU\common\chip\inc */

/**
 * @brief  TIMER0 Interrupt Handler
 * @param  None
 * @return None
 */
void TIMER0_IRQHandler()
{
    /* Clears interrupt */
    RSI_TIMERS_InterruptClear(TIMERS, TIMER_0);
  RSI_Board_LED_Toggle(1);
}

/**
 * @brief  TIMER1 Interrupt Handler
 * @param  None
 * @return None
 */
void TIMER1_IRQHandler()
{
    /* Clears interrupt */
    RSI_TIMERS_InterruptClear(TIMERS, TIMER_1);
  RSI_Board_LED_Toggle(2);
}

/**
 * @brief  TIMER0 Configurations
 * @param  None
 * @return None
 */
void Timer0Config()
{
  /* Sets periodic mode */
  RSI_TIMERS_SetTimerMode(TIMERS, PERIODIC_TIMER, TIMER_0);

  /* Sets timer in 1 Micro second mode */
  RSI_TIMERS_SetTimerType(TIMERS, MICRO_SEC_MODE, TIMER_0);

  /* 1 Micro second timer configuration */
  RSI_TIMERS_MicroSecTimerConfig(TIMERS, TIMER_0, 32, 0 ,MICRO_SEC_MODE);

  RSI_TIMERS_SetMatch(TIMERS, TIMER_0,1000000);

  /* Enables timer interrupt */
  RSI_TIMERS_InterruptEnable(TIMERS , TIMER_0);
}

/**
 * @brief  TIMER1 Configurations
 * @param  None
 * @return None
 */
void Timer1Config()
{
  /* Sets periodic mode */
  RSI_TIMERS_SetTimerMode(TIMERS, PERIODIC_TIMER, TIMER_1);

  /* Sets timer in 1 Micro second mode */
```

```
  RSI_TIMERS_SetTimerType(TIMERS, MICRO_SEC_MODE, TIMER_1);

  /* 1 Micro second timer configuration */
    RSI_TIMERS_MicroSecTimerConfig(TIMERS, TIMER_1, 32, 0 ,MICRO_SEC_MODE);

  RSI_TIMERS_SetMatch(TIMERS, TIMER_1,5000000);

  /* Enables timer interrupt */
  RSI_TIMERS_InterruptEnable(TIMERS , TIMER_1);
}

int main()
{
  SystemCoreClockUpdate();
  /* Timer clock config */
  RSI_ULPSS_TimerClkConfig( ULPCLK ,ENABLE_STATIC_CLK,0,ULP_TIMER_32MHZ_RC_CLK,0);

  /* Timer 0 Configuration */
  Timer0Config();
  /* Timer 1 Configuration */
  Timer1Config();

  /* GPIO Configuration */
  RSI_Board_Init();

  /*  Sets the state of a board '0'number LED to ON. */
  RSI_Board_LED_Set(0, 1);

  /* Interrupt map to ARM */
  NVIC_EnableIRQ(TIMER0_IRQn);
  NVIC_EnableIRQ(TIMER1_IRQn);

  /* Timers start */
  RSI_TIMERS_TimerStart(TIMERS, TIMER_0);
  RSI_TIMERS_TimerStart(TIMERS, TIMER_1);

  /* Code never reaches here. Only used to satisfy standard main() */
  while(1);
}
```

Code Block 22 Timer Example

29.3 API Descriptions

29.3.1 RSI_TIMERS_SetTimerMode

Source File : rsi_timers.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc

Prototype

```
STATIC INLINE error_t RSI_TIMERS_SetTimerMode (RSI_TIMERS_T *pTIMER, boolean_t mode,
uint8_t timerNum))
```

Description

This API is used to set the timer mode.

Parameters

Parameter	Description
pTIMER	This is the pointer to the TIMERS instance register area
mode	This is the mode in which mode timer run

Parameter	Description
	• PERIODIC_TIMER • ONESHOT_TIMER
timerNum	This is the timer number (0 to 3)

Return values

Return the timer error code.

Example

```
#define    TIMER_0      0
/* Sets periodic mode */
RSI_TIMERS_SetTimerMode(TIMERS, PERIODIC_TIMER, TIMER_0);
```

29.3.2 RSI_TIMERS_SetTimerType

Source File : rsi_timers.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc

Prototype

```
STATIC INLINE error_t RSI_TIMERS_SetTimerType (RSI_TIMERS_T *pTIMER, uint8_t timerType,
uint8_t timerNum)
```

Description

This API is used to set the timer type.

Parameters

Parameter	Description
pTIMER	This is the pointer to the TIMERS instance register area
timerType	This is the timer type • MICRO_SEC_MODE • _256_MICRO_SEC_MODE • COUNTER_DOWN_MODE
timerNum	This is the timer number (0 to 3)

Return values

Return the timer error code.

Example

```
#define    TIMER_0      0
/* Sets timer in 1 Micro second mode */
RSI_TIMERS_SetTimerType(TIMERS, MICRO_SEC_MODE, TIMER_0);
```

29.3.3 RSI_TIMERS_MicroSecTimerConfig

Source File : rsi_rom_timer.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc

Prototype

```
STATIC INLINE error_t RSI_TIMERS_MicroSecTimerConfig(RSI_TIMERS_T *pTIMER, uint8_t
timerNum,uint16_t integer,uint8_t fractional,
                                                    uint8_t mode)
```

Description

This API is used to configure timer mode as a 1 micro second or 256 micro second mode.

Parameters

Parameter	Description
pTIMER	This is the pointer to the TIMERS instance register area
timerNum	This is the timer number (0 to 3)
integer	This is an integer part of number of clock cycles per microsecond of the system clock being used
fractional	This is the fractional part of clock cycles per microsecond of the system clock being used
mode	This is the timer mode • MICRO_SEC_MODE • _256_MICRO_SEC_MODE • COUNTER_DOWN_MODE

Return values

- TIMERS_INVAL_TIMER_MODE : If timer mode is invalid

- TIMERS_INVAL_TIMER_NUM_ERROR : If timer number is invalid
- RSI_OK : If process is done

Example

```
#define    TIMER_0      0
/* Micro second timer configuration */
RSI_TIMERS_MicroSecTimerConfig(TIMERS, TIMER_0,32, 0,MICRO_SEC_MODE);
```

29.3.4 RSI_TIMERS_SetMatch

Source File : rsi_timers.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc

Prototype

```
STATIC INLINE error_t RSI_TIMERS_SetMatch (RSI_TIMERS_T *pTIMER, uint8_t timerNum,
uint32_t match)
```

Description

This API is used to set the timer match value.

Parameters

Parameter	Description
pTIMER	This is the pointer to the TIMERS instance register area

Parameter	Description
timerNum	This is the Timer number (0 to 3)
match	This is the delay time

Return values

eturn the timer error code

Example

```
#define    TIMER_0      0
/* generate delay */
RSI_TIMERS_SetMatch(TIMERS, TIMER_0,1000);
```

29.3.5 RSI_TIMERS_InterruptEnable

Source File : rsi_timers.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc

Prototype

```
STATIC INLINE error_t RSI_TIMERS_InterruptEnable (RSI_TIMERS_T *pTIMER, uint8_t timerNum)
```

Description

This API is used to enable the timer interrupt.

Parameters

Parameter	Description
pTIMER	Pointer to the TIMERS instance register area
timerNum	Timer number(0 to 3)

Return values

return the timer error code

Example

```
#define    TIMER_0      0
/* Enables timer interrupt */
RSI_TIMERS_InterruptEnable(TIMERS , TIMER_0);
```

29.3.6 RSI_TIMERS_TimerStart

Source File : rsi_timers.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc

Prototype

```
STATIC INLINE error_t RSI_TIMERS_TimerStart (RSI_TIMERS_T *pTIMER, uint8_t timerNum)
```

Description

This API is used to start the timer.

Parameters

Parameter	Description
pTIMER	This is the pointer to the TIMERS instance register area
timerNum	This is the Timer number (0 to 3)

Return values

Return the timer error code.

Example

```
#define    TIMER_0      0
/* start Timer 0 */
RSI_TIMERS_TimerStart(TIMERS, TIMER_0);
```

29.3.7 RSI_TIMERS_ReadTimer

Source File : rsi_rom_timers.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc

Prototype

```
STATIC INLINE uint32_t  RSI_TIMERS_ReadTimer(RSI_TIMERS_T *pTIMER, uint8_t
timerNum,boolean_t countDir)
```

Description

This API is used to get the count of the required timer.

Parameters

Parameter	Description
pTIMER	This is the pointer to the TIMERS instance register area
timerNum	This is the Timer number (0 to 3)
countDir	This is for reading/tracking counter in up counting this bit has to be set.

Return values

Timer count value.

Example

```
uint32_t read_count;
#define    TIMER_0      0
/* read timer */
read_count = RSI_TIMERS_ReadTimer(TIMERS, TIMER_0,1);
```

29.3.8 RSI_TIMERS_GetTimerType

Source File : rsi_timers.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc

Prototype

```
STATIC INLINE uint32_t RSI_TIMERS_GetTimerType (RSI_TIMERS_T *pTIMER, uint8_t timerNum)
```

Description

This API is used to get the type of timer.

Parameter

Parameter	Description
pTIMER	This is the pointer to the TIMERS instance register area
timerNum	This is the timer number(0 to 3)

Return values

return the type of timer if valid timer else error code

Example

```
uint32_t timer_type;
#define    TIMER_0      0
/* get timer type*/
timer_type=RSI_TIMERS_GetTimerType(TIMERS,TIMER_0);
```

29.3.9 RSI_TIMERS_InterruptStatus

Source File : rsi_timers.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc

Prototype

```
STATIC INLINE uint8_t RSI_TIMERS_InterruptStatus (RSI_TIMERS_T *pTIMER, uint8_t timerNum)
```

Description

This API is used to get the timer interrupt status.

Parameters

Parameter	Description
pTIMER	This is the pointer to the TIMERS instance register area
timerNum	This is the timer number (0 to 3)

Return values

ReturnS the timer interrupt status if valid timer else 0.

Example

```
#define    TIMER_0      0
uint16_t status;
/* used to get the timer interrupt status.  */
status = RSI_TIMERS_InterruptStatus(TIMERS, TIMER_0)
```

29.3.10 RSI_TIMERS_InterruptClear

Source File : rsi_timers.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc

Prototype

```
STATIC INLINE error_t RSI_TIMERS_InterruptClear (RSI_TIMERS_T *pTIMER, uint8_t timerNum)
```

Description

This API is used to clear the timer interrupt.

Parameters

Parameter	Description
pTIMER	tThis is the pointer to the TIMERS instance register area
timerNum	This is the timer number(0 to 3)

Return values

return the timer error code

Example

```
#define    TIMER_0      0
/* Clears inerrupt */
RSI_TIMERS_InterruptClear(TIMERS, TIMER_0);
```

29.3.11 RSI_TIMERS_InterruptDisable

Source File : rsi_timers.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc

Prototype

```
STATIC INLINE error_t RSI_TIMERS_InterruptDisable (RSI_TIMERS_T *pTIMER, uint8_t timerNum)
```

Description

This API is used to disable the timer interrupt.

Parameters

Parameter	Description
pTIMER	This is the pointer to the TIMERS instance register area
timerNum	This is the timer number (0 to 3)

Return values

ReturnS the timer error code

Example

```
#define    TIMER_0      0
/* disable interrupt */
RSI_TIMERS_InterruptDisable(TIMERS , TIMER_0);
```

29.3.12 RSI_TIMERS_TimerStop

Source File : rsi_timers.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc

Prototype

```
STATIC INLINE error_t RSI_TIMERS_TimerStop (RSI_TIMERS_T *pTIMER, uint8_t timerNum)
```

Description

This API is used to stop the timer.

Parameters

Parameter	Description
pTIMER	This is the pointer to the TIMERS instance register area
timerNum	This is the timer number (0 to 3)

Return values

Return the timer error code.

Example

```
#define    TIMER_0      0
/* stop timer0 */
RSI_TIMERS_TimerStop(TIMERS,TIMER_0);
```

30 Power Management Unit(PMU)

30.1 Overview

This section explains how to configure and use the Power Management Unit Redpine MCU SAPIs.

30.2 Programming sequence

```
/*Example to use Deep sleep timer as a wake up source in PS2 state */
int main()
{

    /*Configures the system default clock and power configurations*/
    SystemCoreClockUpdate();

    /*Configure the NPSS-GPIO for sleep wake up indication*/
    RSI_NPSSGPIO_InputBufferEn(NPSS_GPIO_PIN , 1U);
    RSI_NPSSGPIO_SetPinMux(NPSS_GPIO_PIN , 0);
    RSI_NPSSGPIO_SetDir(NPSS_GPIO_PIN , NPSS_GPIO_DIR_OUTPUT);

    /*Change the TASS reference clock to 32MHz RC clock
   NOTE: This should not be used in WiSeMCU mode , should be used in MCU only mode
  */
  RSI_ChangeTassRefClock();

    RSI_CLK_PeripheralClkDisable3(M4CLK , M4_SOC_CLK_FOR_OTHER_ENABLE); /* Disable
OTHER_CLK which is enabled at Start-up */
  RSI_ULPSS_TimerClkDisable( ULPCLK ); /* Disable Timer clock which is enabled in
Bootloader */

    RSI_ULPSS_DisableRefClks( MCU_ULP_40MHZ_CLK_EN );             /* Disabling 40MHz
Clocks */
    RSI_ULPSS_DisableRefClks( MCU_ULP_32KHZ_RC_CLK_EN );        /* Disabling LF_RC Clocks
*/

    RSI_PS_BodPwrGateButtonCalibDisable();    /* Power-Down Button Calibration */
    RSI_PS_AnalogPeriPtatDisable();                    /* Disable PTAT for Analog
Peripherals */
    RSI_PS_BodClksPtatDisable();                        /* Disable PTAT for Brown-Out
Detection Clocks */

    /* Power-Down Analog Peripherals */
    RSI_IPMU_PowerGateClr(
                AUXDAC_PG_ENB
            |    AUXADC_PG_ENB
            |    WURX_CORR_PG_ENB
            |    WURX_PG_ENB
            | ULP_ANG_CLKS_PG_ENB
            |    CMP_NPSS_PG_ENB
    );

    /* Power-Down Domains in NPSS */
    RSI_PS_NpssPeriPowerDown(
                SLPSS_PWRGATE_ULP_MCUWDT
            |    SLPSS_PWRGATE_ULP_MCUPS
            |    SLPSS_PWRGATE_ULP_MCUTS
            |    SLPSS_PWRGATE_ULP_MCUSTORE2
            |    SLPSS_PWRGATE_ULP_MCUSTORE3
    );

    RSI_PS_PowerSupplyDisable(
              POWER_ENABLE_TIMESTAPING);

    RSI_PS_SocPllSpiDisable();            /* Power-Down High-Frequency PLL Domain */
    RSI_PS_QspiDllDomainDisable();    /* Power-Down QSPI-DLL Domain */
    RSI_PS_XtalDisable();                        /* Power-Down XTAL-OSC Domain */
```

```
    RSI_PS_WirelessShutdown();             /* Shutdown Wireless since Wireless Processor
is not present */


   /* TO trim RCMhz clock to 20Mhz*/
   RSI_IPMU_M20rcOsc_TrimEfuse();
/* Change ULPSS-REF and M4SS-REF and NPSS-HF_FSM clocks to 20MHz RO to be used in PS2
state */
    RSI_ULPSS_EnableRefClks( MCU_ULP_32MHZ_RC_CLK_EN, ULP_PERIPHERAL_CLK, 0 );
/* Enable HF-RO Clock */
    RSI_ULPSS_RefClkConfig( ULPSS_ULP_32MHZ_RC_CLK ); /* Configuring ULPSS Reference Clock
to HF-RO */
    RSI_CLK_M4ssRefClkConfig( M4CLK, ULP_32MHZ_RC_CLK );   /* Configuring ULPSS Reference
Clock to HF-RO */

    /* Switching from PS4 to PS2 state */
     RSI_PS_PowerStateChangePs4toPs2(ULP_MCU_MODE ,
            PWR_MUX_SEL_ULPSSRAM_SCDC_0_9        ,
            PWR_MUX_SEL_M4_ULP_RAM_SCDC_0_9      ,
            PWR_MUX_SEL_M4_ULP_RAM16K_SCDC_0_9   ,
            PWR_MUX_SEL_M4ULP_SCDC_0_6           ,
            PWR_MUX_SEL_ULPSS_SCDC_0_9           ,
            DISABLE_BG_SAMPLE_ENABLE             ,
            DISABLE_DC_DC_ENABLE                 ,
            DISABLE_SOCLDO_ENABLE                ,
            DISABLE_STANDBYDC
    );

    RSI_PS_EnableFirstBootUp(1);    /* Enable first boot up */
    RSI_PS_SkipXtalWaitTime(1);        /* XTAL wait time is skipped since RC_32MHZ Clock
is used for Processor on wakeup */

    RSI_PS_SetRamRetention(    M4ULP_RAM16K_RETENTION_MODE_EN ); /* Enable SRAM Retention
of 16KB during Sleep */

    RSI_TIMEPERIOD_TimerClkSel(TIME_PERIOD , 0x003E7FFF);
    /*Deep sleep duration count 7500us*/
    RSI_DST_DurationSet(DS_TIMER_DURATION);
    /* Interrupt unmask */
    RSI_DST_IntrUnMask();
    /*Set wake source as the Alarm interrupt from NPSS */
    RSI_PS_SetWkpSources(DST_BASED_WAKEUP);
    NVIC_EnableIRQ(NVIC_DS_TIMER);
    while(1)
    {

        /*Make GPIO-Low */
      RSI_NPSSGPIO_SetPin(NPSS_GPIO_PIN,0U);

       RSI_PS_FsmHfClkSel(FSM_32MHZ_RC);        /* Configuring HF-RC Clock for HF-FSM */
        /*Configure the wake up paramters for boot loader */
        RSI_PS_RetentionSleepConfig(0 , (uint32_t)RSI_PS_RestoreCpuContext, 0 ,
RSI_WAKEUP_WITH_RETENTION_WO_ULPSS_RAM);
      /*Goto Sleep with retention */
        RSI_PS_EnterDeepSleep(SLEEP_WITH_RETENTION,DISABLE_LF_MODE);
        /*Up on wake up execution resumes from here*/
        RSI_PS_FsmHfClkSel(FSM_20MHZ_RO);        /* Configuring HF-RC Clock for HF-FSM */

        /*Make GPIO-High */
      RSI_NPSSGPIO_SetPin(NPSS_GPIO_PIN,1U);

    }
}
```

Code Block 23 Power save Example

30.3 API Description

30.3.1 RSI_PS_PowerStateChangePs4toPs2

Source File : rsi_rom_power_save.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_PS_PowerStateChangePs4toPs2(ULP_MODE_T enCtxSel, uint8_t
PwrMuxSelUlpssRam, uint8_t pwrMuxSelM4UlpRam, uint8_t pwrMuxSelM4UlpRam16K, uint8_t
pwrMuxSelM4Ulp,uint8_t pwrMuxSelUlpss, uint8_t bgSampleEnable, uint8_t dcDcEnable, uint8_t
socLdoEnable, uint8_t standByDc)
```

Description

This API is used to change the power transition state from Power save state 4/3 to power save state 2.

Parameters

Parameter	Description
enCtxSel	This is to select enum for the context top ULP mode enum value is below • 01 - ULP-MCU Mode • 11 - UULP-MCU Mode (1st 16K of M4 RAM is dedicated to IM, 2nd 16K of M4 RAM is dedicated to DM)
PwrMuxSelUlpssRam	This is to Select value for ULPSS (Peripheral) RAN Power MUX • 3 SOC LDO • 1 SCDCDC 0.9 • 0 SCDCDC 0.6
pwrMuxSelM4UlpRam	This is to select value for M4 ULP RAM Power MUX • 3 SOC LDO • 1 SCDCDC 0.9 • 0 SCDCDC 0.6
pwrMuxSelM4UlpRam16K	This is to select value for M4 ULP RAM 16K Power MUX • 3 SOC LDO • 1 SCDCDC 0.9 • 0 SCDCDC 0.6
pwrMuxSelM4Ulp	This is to select value for M4 ULP (Peripherals + CORTEX Core)Power MUX • 3 SOC LDO • 1 SCDCDC 0.9 • 0 SCDCDC 0.6
pwrMuxSelUlpss	This is to select value for ULPSS(Peripherals) Power MUX • 1 SOC LDO • 0 SCDCDC 0.9
bgSampleEnable	This enable bg sample possible value are below

Parameter	Description
	• Enable: 1 • Disable: 0
dcDcEnable	This is Dc Dc enable possible value is below • Enable: 1 • Disable: 0
socLdoEnable	This is to enable Soc Ldo • Enable: 1 • Disable: 0
standByDc	This is to enable stand by Dc enable • Enable: 1 • Disable: 0

Return values

Returns 0 on success, return error code on error

Example

```
/* required macro value */
#define PWR_MUX_SEL_ULPSSRAM_SCDC_0_9       1
#define PWR_MUX_SEL_ULP_SCDC_0_9            1
#define PWR_MUX_SEL_M4ULP_SCDC_0_9          1
#define PWR_MUX_SEL_ULPSS_SCDC_0_9          1
#define DISABLE_BG_SAMPLE_ENABLE            0
#define DISABLE_DC_DC_ENABLE                0
#define DISABLE_SOCLDO_ENABLE               0
#define DISABLE_STANDBYDC                   0

/* power state change ps4 to 2 */
RSI_PS_PowerStateChangePs4toPs2(ULP_MCU_MODE , PWR_MUX_SEL_ULPSSRAM_SCDC_0_9      ,
          PWR_MUX_SEL_M4_ULP_RAM_SCDC_0_9    ,
          PWR_MUX_SEL_M4_ULP_RAM16K_SCDC_0_9 ,
          PWR_MUX_SEL_M4ULP_SCDC_0_6         ,
          PWR_MUX_SEL_ULPSS_SCDC_0_9         ,
          DISABLE_BG_SAMPLE_ENABLE           ,
          DISABLE_DC_DC_ENABLE               ,
          DISABLE_SOCLDO_ENABLE              ,
          DISABLE_STANDBYDC
          );
```

30.3.2 RSI_PS_PowerStateChangePs2toPs4

Source File : rsi_rom_power_save.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE error_t RSI_PS_PowerStateChangePs2toPs4 (uint32_t PmuBuckTurnOnWaitTime,
uint32_t SocLdoTurnOnWaitTime)
```

Description

This API is used to change the power state from PS2 to PS4.

Parameters

Parameter	Description
PmuBuckTurnOnWaitTime	This is the PMU buck time
PwrMuxSelUlpssRam	This is soc ldo turn on time

Return values

Returns 0 on success,return error code on error

Example

```
/* power save change */
RSI_PS_PowerStateChangePs2toPs4(100,100);
```

30.3.3 RSI_PS_ClrWkpUpStatus

Source File : rsi_rom_power_save.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_PS_ClrWkpUpStatus(uint32_t wakeUpIntrClear)
```

Description

This API is used clear the NPSS/wake up interrupts.

Parameter

Parameter	Description
wakeUpIntrClear	These are the OR'ed values of register bits of NPSS interrupt register value is below • NPSS_TO_MCU_WDT_INTR • NPSS_TO_MCU_GPIO_INTR_0 • NPSS_TO_MCU_GPIO_INTR_1 • NPSS_TO_MCU_GPIO_INTR_2 • NPSS_TO_MCU_GPIO_INTR_3 • NPSS_TO_MCU_GPIO_INTR_4 • NPSS_TO_MCU_CMP_INTR_1 • NPSS_TO_MCU_CMP_INTR_2 • NPSS_TO_MCU_CMP_INTR_3 • NPSS_TO_MCU_CMP_INTR_4 • NPSS_TO_MCU_RFWAKEUP_INTR • NPSS_TO_MCU_BOD_INTR • NPSS_TO_MCU_BUTTON_INTR • NPSS_TO_MCU_SDC_INTR • NPSS_TO_MCU_WIRELESS_INTR • NPSS_TO_MCU_WAKEUP_INTR • NPSS_TO_MCU_ALARM_INTR • NPSS_TO_MCU_SEC_INTR • NPSS_TO_MCU_MSEC_INTR

Parameter	Description
	• NPSS_TO_MCU_PROCESSOR_INTR • NPSS_TO_MCU_HOST_INTR • NPSS_TO_MCU_DST_INTR

Return values

None

Example

```
/* clear interrupt value */
RSI_PS_ClrWkpUpStatus(NPSS_TO_MCU_WDT_INTR);
```

30.3.4 RSI_PS_RetentionSleepConfig

Source File : rsi_rom_power_save.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\rom_driver\inc\

Prototype

```
STATIC INLINE void RSI_PS_RetentionSleepConfig(uint32_t stack_address, uint32_t
jump_cb_address, uint32_t vector_offset,uint32_t mode)
```

Description

This API is used clear the NPSS/wake up interrupts.

Parameter

Parameter	Description
stack_address	stack pointer address to be used by bootloader
jump_cb_address	Control block memory address or function address to be branched up on wake up
vector_offset	IVT offset to be programmed by boot-loader up on wake up.
mode	Following are the possible sleep modes or possible values to the parameter : • RSI_WAKEUP_FROM_FLASH_MODE : Wakes from flash with retention. Upon wake up control jumps to wake up handler in flash.In this mode ULPSS RAMs are used to store the stack pointer and wake up handler address. • RSI_WAKEUP_WITH_OUT_RETENTION : Without retention sleep common for both FLASH/RAM based execution.In this mode ULPSS RAMs are used to store the stack pointer and control block address. if stack_addr and jump_cb_addr are not valid then 0x2404_0C00 and 0x2404_0000 are used for stack and control block address respectively. • RSI_WAKEUP_WITH_RETENTION : With retention branches to wake up handler in RAM.In this mode ULPSS RAMs are used to store the wake up handler address. • RSI_WAKEUP_WITH_RETENTION_WO_ULPSS_RAM : In this mode ULPSS RAMs are not used by boot-loader instead it uses the NPSS battery flip flops. • RSI_WAKEUP_WO_RETENTION_WO_ULPSS_RAM : In this mode ULPSS RAMs are not used by boot-loader.Instead it uses the NPSS battery flip flops to store the stack and derives the control block address by adding 0XC00 to the stack address stored in battery flops.

Return values

None

Example

```
/*Configure the wake up paramters for boot loader */
RSI_PS_RetentionSleepConfig(0 , (uint32_t)RSI_PS_RestoreCpuContext, 0 ,
RSI_WAKEUP_WITH_RETENTION_WO_ULPSS_RAM);
```

30.3.5 RSI_PS_EnterDeepSleep

Source File : system_RS1xxxx.c

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Common\chip\src

Prototype

```
error_t RSI_PS_EnterDeepSleep(SLEEP_TYPE_T sleepType , uint8_t lf_clk_mode)
```

Description

This API is used to keep the system in sleep state. from all possible active states.

Parameters

Parameter	Description
sleepType	Selects the retention or non retention mode of processor. refer 'SLEEP_TYPE_T'. • SLEEP_WITH_RETENTION : When this is used, user must configure the which RAMs to be retained during sleep by using the 'RSI_PS_SetRamRetention()' function.
lf_clk_mode	This parameter is used to switch the processor clock from high frequency clock to low-frequency clock. This is used in some critical power save cases. • 0: 'DISABLE_LF_MODE' Normal mode of operation , recommended in most of the applications. • 1: 'LF_32_KHZ_RC' Processor clock is configured to low-frequency RC clock • 2: 'LF_32_KHZ_XTAL' Processor clock is configured to low-frequency XTAL clock • 3: 'EXTERNAL_CAP_MODE' Switches the supply to internal cap mode 0.95V.

Return value

Returns the execution status.

Example

```
/*Goto Sleep with retention */
RSI_PS_EnterDeepSleep(SLEEP_WITH_RETENTION,DISABLE_LF_MODE);
```

30.3.6 RSI_PS_PowerStateChangePs4toPs3

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_PS_PowerStateChangePs4toPs3 (void)
```

Description

This API is used to Change the power state from PS4 to PS3

Parameter

None

Return value

None

Example

```
/* change state ps4 to ps3 */
RSI_PS_PowerStateChangePs4toPs3();
```

30.3.7 RSI_PS_PowerStateChangePs3toPs4

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_PS_PowerStateChangePs3toPs4 (void)
```

Description

This API is used to Change the power state from PS3 to PS4.

Parameter

None

Return value

None

Example

```
/* change state ps3 to ps4 */
RSI_PS_PowerStateChangePs3toPs4();
```

30.3.8 RSI_PS_M4ssPeriPowerDown

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_PS_M4ssPeriPowerDown (uint32_t mask)
```

Description

This API is used to power gate the M4SS peripherals.

Parameter

Parameter	Description
mask	OR'ed value of the power gates • M4SS_PWRGATE_ULP_M4_DEBUG • M4SS_PWRGATE_ULP_M4_FPU • M4SS_PWRGATE_ULP_ICACHE • M4SS_PWRGATE_ULP_QSPI • M4SS_PWRGATE_ULP_ETHERNET • M4SS_PWRGATE_ULP_SDIO_SPI • M4SS_PWRGATE_ULP_USB • M4SS_PWRGATE_ULP_RPDMA • M4SS_PWRGATE_ULP_PERI1 • M4SS_PWRGATE_ULP_PERI2 • M4SS_PWRGATE_ULP_PERI3 • M4SS_PWRGATE_ULP_CCI • M4SS_PWRGATE_ULP_EFUSE • M4SS_PWRGATE_ULP_SD_MEM

Return values

None

Example

```
RSI_PS_M4ssPeriPowerDown(M4SS_PWRGATE_ULP_M4_FPU|M4SS_PWRGATE_ULP_QSPI);
```

30.3.9 RSI_PS_M4ssPeriPowerUp

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_PS_M4ssPeriPowerUp (uint32_t mask)
```

Description

This API is used to power gate the M4SS peripherals.

Parameter

Parameter	Description
mask	These are the OR'ed values of the power gates M4SS_PWRCTRL_SET_REG • M4SS_PWRGATE_ULP_M4_DEBUG • M4SS_PWRGATE_ULP_M4_FPU • M4SS_PWRGATE_ULP_ICACHE • M4SS_PWRGATE_ULP_QSPI

Parameter	Description
	• M4SS_PWRGATE_ULP_ETHERNET • M4SS_PWRGATE_ULP_SDIO_SPI • M4SS_PWRGATE_ULP_USB • M4SS_PWRGATE_ULP_RPDMA • M4SS_PWRGATE_ULP_PERI1 • M4SS_PWRGATE_ULP_PERI2 • M4SS_PWRGATE_ULP_PERI3 • M4SS_PWRGATE_ULP_CCI • M4SS_PWRGATE_ULP_EFUSE • M4SS_PWRGATE_ULP_SD_MEM

Return values

None

Example

```
/*Power Up the power gates */
RSI_PS_M4ssPeriPowerUp(
        M4SS_PWRGATE_ULP_M4_DEBUG |
        M4SS_PWRGATE_ULP_M4_FPU   |
        M4SS_PWRGATE_ULP_IID      |
        M4SS_PWRGATE_ULP_ETHERNET |
        M4SS_PWRGATE_ULP_SDIO_SPI |
        M4SS_PWRGATE_ULP_USB      |
        M4SS_PWRGATE_ULP_RPDMA    |
        M4SS_PWRGATE_ULP_PERI1  |
        M4SS_PWRGATE_ULP_PERI2  |
        M4SS_PWRGATE_ULP_PERI3  |
        M4SS_PWRGATE_ULP_CCI      |
        M4SS_PWRGATE_ULP_SD_MEM);
```

30.3.10 RSI_PS_UlpssPeriPowerDown

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_PS_UlpssPeriPowerDown(uint32_t mask)
```

Description

This API is used to power gate the ULPSS peripherals.

Parameter

Parameter	Description
mask	These are the OR'ed values of the power gates ULPSS_PWRCTRL_CLEAR_REG • ULPTASS_PWRGATE_ULP_MISC • ULPTASS_PWRGATE_ULP_CAP

Parameter	Description
	• ULPTASS_PWRGATE_ULP_VAD • ULPTASS_PWRGATE_ULP_UART • ULPTASS_PWRGATE_ULP_SSI • ULPTASS_PWRGATE_ULP_I2S • ULPTASS_PWRGATE_ULP_I2C • ULPTASS_PWRGATE_ULP_AUX • ULPTASS_PWRGATE_ULP_IR • ULPTASS_PWRGATE_ULP_UDMA • ULPTASS_PWRGATE_ULP_FIM

Return values

None

Example

```
RSI_PS_UlpssPeriPowerDown(RSI_PS_UlpssPeriPowerDown|ULPSS_PWRGATE_ULP_VAD);
```

30.3.11 RSI_PS_UlpssPeriPowerUp

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_PS_UlpssPeriPowerUp (uint32_t mask)
```

Description

This API is used to un power gate the ULPSS peripherals.

Parameter

Parameter	Description
mask	OR'ed value of the power gates ULPSS_PWRCTRL_CLEAR_REG • ULPTASS_PWRGATE_ULP_MISC • ULPTASS_PWRGATE_ULP_CAP • ULPTASS_PWRGATE_ULP_VAD • ULPTASS_PWRGATE_ULP_UART • ULPTASS_PWRGATE_ULP_SSI • ULPTASS_PWRGATE_ULP_I2S • ULPTASS_PWRGATE_ULP_I2C • ULPTASS_PWRGATE_ULP_AUX • ULPTASS_PWRGATE_ULP_IR • ULPTASS_PWRGATE_ULP_UDMA

Parameter	Description
	• ULPTASS_PWRGATE_ULP_FIM

Return values

None

Example

```
/*ULPSS Peripheral power gate */
RSI_PS_UlpssPeriPowerUp(
         ULPTASS_PWRGATE_ULP_MISC      |
         ULPTASS_PWRGATE_ULP_CAP       |
         ULPTASS_PWRGATE_ULP_VAD       |
         ULPTASS_PWRGATE_ULP_UART      |
         ULPTASS_PWRGATE_ULP_SSI       |
         ULPTASS_PWRGATE_ULP_I2S       |
         ULPTASS_PWRGATE_ULP_I2C       |
         ULPTASS_PWRGATE_ULP_AUX       |
         ULPTASS_PWRGATE_ULP_IR         |
         ULPTASS_PWRGATE_ULP_UDMA      |
         ULPTASS_PWRGATE_ULP_FIM     );
```

30.3.12 RSI_PS_NpssPeriPowerUp

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_PS_NpssPeriPowerUp (uint32_t mask)
```

Description

This API is used to un power gate the NPSS peripherals.

Parameter

Parameter	Description
mask	These are the OR'ed values of the power domains MCUAON_NPSS_PWRCTRL_SET_REG register value is below • SLPSS_PWRGATE_ULP_NWPAPB_MCU_CTRL • SLPSS_PWRGATE_ULP_TIMEPERIOD • SLPSS_PWRGATE_ULP_MCUSTORE3 • SLPSS_PWRGATE_ULP_MCUSTORE2 • SLPSS_PWRGATE_ULP_MCUSTORE1 • SLPSS_PWRGATE_ULP_MCUTS • SLPSS_PWRGATE_ULP_MCUPS • SLPSS_PWRGATE_ULP_MCUWDT • SLPSS_PWRGATE_ULP_MCURTC • SLPSS_PWRGATE_ULP_MCUFSM • SLPSS_PWRGATE_ULP_MCUBFFS

Return values

None

Example

```
/*Power up NPSS peripherals*/
 RSI_PS_NpssPeriPowerUp (
      SLPSS_PWRGATE_ULP_MCUSTORE3 |
      SLPSS_PWRGATE_ULP_MCUSTORE2 |
      SLPSS_PWRGATE_ULP_MCUTS     |
      SLPSS_PWRGATE_ULP_MCUPS     |
      SLPSS_PWRGATE_ULP_MCURTC
 );
```

30.3.13 RSI_PS_NpssPeriPowerDown

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_PS_NpssPeriPowerDown (uint32_t mask)
```

Description

This API is used to power gate the NPSS peripherals.

Parameter

Parameter	Description
mask	These are the OR'ed values of the power domains MCUAON_NPSS_PWRCTRL_CLEAR_REG register below value • SLPSS_PWRGATE_ULP_NWPAPB_MCU_CTRL • SLPSS_PWRGATE_ULP_TIMEPERIOD • SLPSS_PWRGATE_ULP_MCUSTORE3 • SLPSS_PWRGATE_ULP_MCUSTORE2 • SLPSS_PWRGATE_ULP_MCUSTORE1 • SLPSS_PWRGATE_ULP_MCUTS • SLPSS_PWRGATE_ULP_MCUPS • SLPSS_PWRGATE_ULP_MCUWDT • SLPSS_PWRGATE_ULP_MCURTC • SLPSS_PWRGATE_ULP_MCUFSM • SLPSS_PWRGATE_ULP_MCUBFFS

Return values

None

Example

```
/*Power down NPSS peripherals*/
RSI_PS_NpssPeriPowerDown(
      SLPSS_PWRGATE_ULP_MCUSTORE3 |
      SLPSS_PWRGATE_ULP_MCUSTORE2 |
      SLPSS_PWRGATE_ULP_MCUTS     |
      SLPSS_PWRGATE_ULP_MCUPS     |
      SLPSS_PWRGATE_ULP_MCURTC
```

```
);
```

30.3.14 RSI_PS_M4ssRamBanksPowerDown

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_PS_M4ssRamBanksPowerDown (uint32_t mask)
```

Description

This API is used to power gate the M4SS RAM Banks.

Parameter

Parameter	Description
mask	These are the OR'ed values of the RAM power gates M4_SRAM_PWRCTRL_CLEAR_REG1 • RAM_BANK_8 • RAM_BANK_9 • RAM_BANK_10 • RAM_BANK_11 • RAM_BANK_12 • RAM_BANK_13 • RAM_BANK_0 • RAM_BANK_1 • RAM_BANK_2

Return values

None

Example

```
/*Power gate the sub RAM sections of M4SS*/
RSI_PS_M4ssRamBanksPowerDown(
         RAM_BANK_8 | /* M4ULP     */
         RAM_BANK_9 | /* M4ULP     */
         RAM_BANK_10| /* TASS RAM  */
         RAM_BANK_11| /* TASS RAM  */
         RAM_BANK_12| /* TASS RAM  */
         RAM_BANK_13| /* TASS RAM  */
         RAM_BANK_0 | /* M4SS      */
         RAM_BANK_1 | /* M4SS      */
         RAM_BANK_2   /* M4SS      */
 );
```

30.3.15 RSI_PS_SetRamRetention

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_PS_SetRamRetention (uint32_t ramRetention)
```

Description

This API is used to set the RAM retention enable for the RAM during sleep.

Parameter

Parameter	Description
ramRetention	These are the OR'ed values of the RAM retention bits MCU_FSM_SLEEP_CTRLS_AND_WAKEUP_MODE register • M4ULP_RAM16K_RETENTION_MODE_EN • ULPSS_RAM_RETENTION_MODE_EN • TA_RAM_RETENTION_MODE_EN • M4ULP_RAM_RETENTION_MODE_EN • M4SS_RAM_RETENTION_MODE_EN

Return values

None

Example

```
/*select which banks needs to be powered on during sleep*/
RSI_PS_SetRamRetention(M4ULP_RAM_RETENTION_MODE_EN | M4ULP_RAM16K_RETENTION_MODE_EN
|ULPSS_RAM_RETENTION_MODE_EN |M4SS_RAM_RETENTION_MODE_EN);
```

30.3.16 RSI_PS_ClrRamRetention

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_PS_ClrRamRetention (uint32_t ramRetention)
```

Description

This API is used to clear the RAM retention enable for the RAM during sleep.

Parameter

Parameter	Description
ramRetention	These are the OR'ed values of the RAM retention bits MCU_FSM_SLEEP_CTRLS_AND_WAKEUP_MODE register • M4ULP_RAM16K_RETENTION_MODE_EN • ULPSS_RAM_RETENTION_MODE_EN • TA_RAM_RETENTION_MODE_EN • M4ULP_RAM_RETENTION_MODE_EN • M4SS_RAM_RETENTION_MODE_EN

Return values

None

Example

```
RSI_PS_ClrRamRetention(M4ULP_RAM_RETENTION_MODE_EN | M4ULP_RAM16K_RETENTION_MODE_EN
|ULPSS_RAM_RETENTION_MODE_EN |M4SS_RAM_RETENTION_MODE_EN);
```

30.3.17 RSI_PS_UlpssRamBanksPowerDown

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_PS_UlpssRamBanksPowerDown (uint32_t mask)
```

Description

This API is used to clear the RAM retention enable for the RAM during sleep.

Parameter

Parameter	Description
mask	These are the OR'ed valueS of the RAM power gates ULPSS_RAM_PWRCTRL_CLEAR_REG1 register value is below • ULPSS_2K_BANK_0 • ULPSS_2K_BANK_1 • ULPSS_2K_BANK_2 • ULPSS_2K_BANK_3 • ULPSS_2K_BANK_4 • ULPSS_2K_BANK_5 • ULPSS_2K_BANK_6 • ULPSS_2K_BANK_7

Return values

None

Example

```
/*ULPSS RAM clear*/
RSI_PS_UlpssRamBanksPowerDown(
      ULPSS_2K_BANK_3 |
      ULPSS_2K_BANK_4 |
      ULPSS_2K_BANK_5 |
      ULPSS_2K_BANK_6 |
      ULPSS_2K_BANK_7
 );
```

30.3.18 RSI_PS_UlpssRamBanksPowerUp

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_PS_UlpssRamBanksPowerUp (uint32_t mask)
```

Description

This API is used to un power gate the ULPSS RAM Banks.

Parameter

Parameter	Description
mask	These are the OR'ed values of the RAM power gates ULPSS_RAM_PWRCTRL_SET_REG1 register value is below • ULPSS_2K_BANK_0 • ULPSS_2K_BANK_1 • ULPSS_2K_BANK_2 • ULPSS_2K_BANK_3 • ULPSS_2K_BANK_4 • ULPSS_2K_BANK_5 • ULPSS_2K_BANK_6 • ULPSS_2K_BANK_7

Return values

None

Example

```
RSI_PS_UlpssRamBanksPowerUp(        ULPSS_2K_BANK_3 |
       ULPSS_2K_BANK_4 |
       ULPSS_2K_BANK_5 |
       ULPSS_2K_BANK_6 |
       ULPSS_2K_BANK_7);
```

30.3.19 void RSI_PS_SetRamRetention

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_PS_SetRamRetention (uint32_t ramRetention)
```

Description

This API is used to set the RAM retention enable for the RAM during sleep.

Parameter

Parameter	Description
ramRetention	These are the OR'ed values of the RAM retention bits MCU_FSM_SLEEP_CTRLS_AND_WAKEUP_MODE register value is below • M4ULP_RAM16K_RETENTION_MODE_EN • ULPSS_RAM_RETENTION_MODE_EN

Parameter	Description
	• TA_RAM_RETENTION_MODE_EN • M4ULP_RAM_RETENTION_MODE_EN • M4SS_RAM_RETENTION_MODE_EN

Return values

None

Example

```
/*select which banks needs to be powered on during sleep*/
RSI_PS_SetRamRetention(
      M4ULP_RAM_RETENTION_MODE_EN    |
      M4ULP_RAM16K_RETENTION_MODE_EN | /*From BANK3 to BANK 6*/
      ULPSS_RAM_RETENTION_MODE_EN    | /*For S port RAM access*/
      M4SS_RAM_RETENTION_MODE_EN);
```

30.3.20 RSI_PS_ClrRamRetention

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_PS_ClrRamRetention(uint32_t ramRetention)
```

Description

This API is used to clear the RAM retention enable for the RAM during sleep.

Parameter

Parameter	Description
mask	These are the OR'ed values of the RAM retention bits MCU_FSM_SLEEP_CTRLS_AND_WAKEUP_MODE register value is below • M4ULP_RAM16K_RETENTION_MODE_EN • ULPSS_RAM_RETENTION_MODE_EN • TA_RAM_RETENTION_MODE_EN • M4ULP_RAM_RETENTION_MODE_EN • M4SS_RAM_RETENTION_MODE_EN

Return values

none

Example

```
RSI_PS_ClrRamRetention(  M4ULP_RAM_RETENTION_MODE_EN    |
      M4ULP_RAM16K_RETENTION_MODE_EN | /*From BANK3 to BANK 6*/
      ULPSS_RAM_RETENTION_MODE_EN    | /*For S port RAM access*/
      M4SS_RAM_RETENTION_MODE_EN);
```

30.3.21 RSI_PS_UlpssRamBanksPowerDown

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_PS_UlpssRamBanksPowerDown (uint32_t mask)
```

Description

This API is used to power gate the ULPSS RAM Banks.

Parameter

Parameter	Description
mask	These are the OR'ed values of the RAM power gates ULPSS_RAM_PWRCTRL_CLEAR_REG1 register value is below • ULPSS_2K_BANK_0 • ULPSS_2K_BANK_1 • ULPSS_2K_BANK_2 • ULPSS_2K_BANK_3 • ULPSS_2K_BANK_4 • ULPSS_2K_BANK_5 • ULPSS_2K_BANK_6 • ULPSS_2K_BANK_7

Return values

None

Example

```
/*ULPSS RAM clear*/
RSI_PS_UlpssRamBanksPowerDown(
      ULPSS_2K_BANK_3 |
      ULPSS_2K_BANK_4 |
      ULPSS_2K_BANK_5 |
      ULPSS_2K_BANK_6 |
      ULPSS_2K_BANK_7
);
```

30.3.22 RSI_PS_UlpssRamBanksPowerUp

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_PS_UlpssRamBanksPowerUp (uint32_t mask)
```

Description

This API is used to un power gate the ULPSS RAM Banks.

Parameter

Parameter	Description

Parameter	Description
mask	These are the OR'ed values of the RAM power gates ULPSS_RAM_PWRCTRL_CLEAR_REG1 register value is below • ULPSS_2K_BANK_0 • ULPSS_2K_BANK_1 • ULPSS_2K_BANK_2 • ULPSS_2K_BANK_3 • ULPSS_2K_BANK_4 • ULPSS_2K_BANK_5 • ULPSS_2K_BANK_6 • ULPSS_2K_BANK_7

Return values

None

Example

```
/*Power up the ULPSS RAM bank for ADC samples*/
RSI_PS_UlpssRamBanksPowerUp(ULPSS_2K_BANK_5);
```

30.3.23 RSI_PS_SetWkpSources

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_PS_SetWkpSources (uint32_t wakeUpsrcMask)
```

Description

This API is used to set the wake up source to wake up from deep sleep. .

Parameter

Parameter	Description
wakeUpsrcMask	These are the OR'ed values of the wake up sources MCU_FSM_SLEEP_CTRLS_AND_WAKEUP_MODE register value is below • WDT_INTR_BASED_WAKEUP • MSEC_BASED_WAKEUP • SEC_BASED_WAKEUP • ALARM_BASED_WAKEUP • SDCSS_BASED_WAKEUP • ULPSS_BASED_WAKEUP • WAKEIF_BASED_WAKEUP • COMPR_BASED_WAKEUP • GPIO_BASED_WAKEUP • M4_PROCS_BASED_WAKEUP

Parameter	Description
	• WIRELESS_BASED_WAKEUP • HOST_BASED_WAKEUP • DST_BASED_WAKEUP • WIC_BASED_WAKEUP

Return values

None

Example

```
/*Enable ULPSS based sleep*/
RSI_PS_SetWkpSources(ULPSS_BASED_WAKEUP);
```

30.3.24 RSI_PS_ClrWkpSources

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_PS_ClrWkpSources (uint32_t wakeUpsrcMask)
```

Description

This API is used to set the wake up source to wake up from deep sleep. .

Parameter

Parameter	Description
wakeUpsrcMask	These are the OR'ed values of the wake up sources MCU_FSM_SLEEP_CTRLS_AND_WAKEUP_MODE register value is below • WDT_INTR_BASED_WAKEUP • MSEC_BASED_WAKEUP • SEC_BASED_WAKEUP • ALARM_BASED_WAKEUP • SDCSS_BASED_WAKEUP • ULPSS_BASED_WAKEUP • WAKEIF_BASED_WAKEUP • COMPR_BASED_WAKEUP • GPIO_BASED_WAKEUP • M4_PROCS_BASED_WAKEUP • WIRELESS_BASED_WAKEUP • HOST_BASED_WAKEUP • DST_BASED_WAKEUP • WIC_BASED_WAKEUP

Return values

None

Example

```
RSI_PS_ClrWkpSources(WDT_INTR_BASED_WAKEUP);
```

30.3.25 RSI_PS_GetWkpSources

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE uint32_t RSI_PS_GetWkpSources (void)
```

Description

This API is used to get the wake up source.

Parameter

None

Return values

register bits of wake up sources

Example

```
uint32_t wkpsource;

wkpsource=RSI_PS_GetWkpSources();
```

30.3.26 RSI_PS_EnableFirstBootUp

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_PS_EnableFirstBootUp (boolean_t enable)
```

Description

This API is used to SET and CLEAR the First boot up bit.

Parameter

Parameter	Description
enable	MCU_FSM_CLK_ENS_AND_FIRST_BOOTUP_b 0: disable the first boot , MCU_FSM_CLK_ENS_AND_FIRST_BOOTUP_b 1: enable the first boot up

Return values

None

Example

```
RSI_PS_EnableFirstBootUp(1);
```

30.3.27 RSI_PS_PowerSupplyEnable

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_PS_PowerSupplyEnable (uint32_t mask)
```

Description

This API is used to enable the supply to some NPSS peripherals.

Parameter

Parameter	Description
mask	These are the OR'ed values of the Power supply enable • ENABLE_WDT_IN_SLEEP • ENABLE_WURX_DETECTION • RESET_MCU_BBF_DM_EN • DISABLE_TURNOFF_SRAM_PERI • ENABLE_SRAM_DS_CTRL • POWER_ENABLE_FSM_PERI • POWER_ENABLE_TIMESTAPING • POWER_ENABLE_DEEPSLEEP_TIMER • POWER_ENABLE_RETENTION_DM

Return values

none

Example

```
#define       Enable    1
#define       disable    0
/* power supply enable */
RSI_PS_PowerSupplyEnable(POWER_ENABLE_TIMESTAPING |POWER_ENABLE_DEEPSLEEP_TIMER);
```

30.3.28 RSI_PS_PowerSupplyDisable

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_PS_PowerSupplyDisable (uint32_t mask)
```

Description

This API is used to disable the supply to some NPSS peripherals.

Parameter

Parameter	Description
mask	These are the OR'ed values of the Power supply disable

Parameter	Description
	• ENABLE_WDT_IN_SLEEP • ENABLE_WURX_DETECTION • RESET_MCU_BBF_DM_EN • DISABLE_TURNOFF_SRAM_PERI • ENABLE_SRAM_DS_CTRL • POWER_ENABLE_FSM_PERI • POWER_ENABLE_TIMESTAPING • POWER_ENABLE_DEEPSLEEP_TIMER • POWER_ENABLE_RETENTION_DM

Return values

none

Example

```
/* power supply disable */
RSI_PS_PowerSupplyDisable (POWER_ENABLE_TIMESTAPING |POWER_ENABLE_DEEPSLEEP_TIMER);
```

30.3.29 RSI_PS_FsmHfClkSel

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_PS_FsmHfClkSel (FSM_CLK_T fsmHfClk)
```

Description

This API is used to configure the FSM high frequency clock.

Parameter

Parameter	Description
fsmHfClk	enum value of the high frequency clock sources MCU_FSM_CLKS_REG_b enum value is below • FSM_NO_CLOCK • FSM_20MHZ_RO • FSM_32MHZ_RC • FSM_40MHZ_XTAL

Return values

None

Example

```
RSI_PS_FsmHfClkSel(FSM_32MHZ_RC);
```

30.3.30 RSI_PS_FsmLfClkSel

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_PS_FsmLfClkSel (AON_CLK_T fsmLfClk)
```

Description

This API is used to configure the FSM low frequency clock.

Parameter

Parameter	Description
fsmLfClk	enum value of the low frequency clock sources MCUAON_KHZ_CLK_SEL_POR_RESET_STATUS_b enum value is below • KHZ_RO_CLK_SEL • KHZ_RC_CLK_SEL • KHZ_XTAL_CLK_SEL

Return values

none

Example

```
RSI_PS_FsmLfClkSel(KHZ_RO_CLK_SEL);
```

30.3.31 RSI_PS_PmuGoodTimeDurationConfig

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_PS_PmuGoodTimeDurationConfig (uint8_t pmuDuration)
```

Description

This API is used to configure the PMU good time.

Parameter

Parameter	Description
pmuDuration	(0 to 31) are possible value is applied in power of 2.

Return values

none

Example

```
#define PMU_GOOD_TIME        31    /*Duration in us*/
/*Configure the PMU turn on time */
```

```
RSI_PS_PmuGoodTimeDurationConfig(PMU_GOOD_TIME);
```

30.3.32 RSI_PS_XtalGoodTimeDurationConfig

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_PS_XtalGoodTimeDurationConfig (uint8_t xtalDuration)
```

Description

This API is used to configure the XTAL good time.

Parameter

Parameter	Description
xtalDuration	(0 to 31) are possible value is applied in power of 2. MCU_FSM_XTAL_AND_PMU_GOOD_COUNT_REG_b

Return values

none

Example

```
#define XTAL_GOOD_TIME       31    /*Duration in us*/
/*Configure the PMU turn on time */
RSI_PS_XtalGoodTimeDurationConfig (XTAL_GOOD_TIME);
```

30.3.33 RSI_PS_Ps2PmuLdoOffDelayConfig

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_PS_Ps2PmuLdoOffDelayConfig (uint8_t ldoOffDelay)
```

Description

This API is used to configure LDO off delay.

Parameter

Parameter	Description
ldoOffDelay	(0 to 31) are possible value is applied in power of 2

Return values

None

Example

```
RSI_PS_Ps2PmuLdoOffDelayConfig(16);
```

30.3.34 RSI_PS_Ps4PmuBuckOnDelayConfig

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_PS_Ps4PmuBuckOnDelayConfig (uint8_t pmuBuckOnDelay)
```

Description

This API is used to configure buck on delay.

Parameter

Parameter	Description
pmuBuckOnDelay	(0 to 31) are possible value is applied in power of 2. MCU_FSM_POWER_CTRL_AND_DELAY_b

Return values

None

Example

```
RSI_PS_Ps4PmuBuckOnDelayConfig(31);
```

30.3.35 RSI_PS_GetWkpUpStatus

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE uint32_t RSI_PS_GetWkpUpStatus (void)
```

Description

This API is used to get the wake up/ NPSS interrupt status NPSS_INTR_STATUS_REG

Parameter

None

Return values

register bits of NPSS interrupt status register

Example

```
volatile   uint16_t wakeUpSrc=0;
wakeUpSrc = RSI_PS_GetWkpUpStatus();
```

30.3.36 RSI_PS_GetComnIntrSts

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE uint32_t RSI_PS_GetComnIntrSts (void)
```

Description

This API is used to get the wake up / NPSS common interrupt status.

Return values

register bits of NPSS interrupt status register MCU_FSM_WAKEUP_STATUS_REG .

Example

```
uint32_t ReadCommIntrSts
ReadCommIntrSts = RSI_PS_GetComnIntrSts();
```

30.3.37 RSI_PS_NpssIntrUnMask

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void  RSI_PS_NpssIntrUnMask (uint32_t mask)
```

Description

This API is used to un mask the NPSS interrupts.

Parameter

Parameter	Description
mask	OR'ed value of the NPSS interrupt bits NPSS_INTR_MASK_CLR_REG register value is below • NPSS_TO_MCU_WDT_INTR • NPSS_TO_MCU_GPIO_INTR_0 • NPSS_TO_MCU_GPIO_INTR_1 • NPSS_TO_MCU_GPIO_INTR_2 • NPSS_TO_MCU_GPIO_INTR_3 • NPSS_TO_MCU_GPIO_INTR_4 • NPSS_TO_MCU_CMP_INTR_1 • NPSS_TO_MCU_CMP_INTR_2 • NPSS_TO_MCU_CMP_INTR_3 • NPSS_TO_MCU_CMP_INTR_4 • NPSS_TO_MCU_RFWAKEUP_INTR • NPSS_TO_MCU_BOD_INTR • NPSS_TO_MCU_BUTTON_INTR • NPSS_TO_MCU_SDC_INTR • NPSS_TO_MCU_WIRELESS_INTR • NPSS_TO_MCU_WAKEUP_INTR

Parameter	Description
	• NPSS_TO_MCU_ALARM_INTR • NPSS_TO_MCU_SEC_INTR • NPSS_TO_MCU_MSEC_INTR • NPSS_TO_MCU_PROCESSOR_INTR • NPSS_TO_MCU_HOST_INTR • NPSS_TO_MCU_DST_INTR

Return values

None

Example

```
RSI_PS_NpssIntrUnMask (NPSS_TO_MCU_RFWAKEUP_INTR);
```

30.3.38 RSI_PS_NpssIntrMask

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_PS_NpssIntrMask (uint32_t mask)
```

Description

This API is used to mask the NPSS interrupts.

Parameter

Parameter	Description
mask	OR'ed value of the NPSS interrupt bits NPSS_INTR_MASK_SET_REG register value is below • NPSS_TO_MCU_WDT_INTR • NPSS_TO_MCU_GPIO_INTR_0 • NPSS_TO_MCU_GPIO_INTR_1 • NPSS_TO_MCU_GPIO_INTR_2 • NPSS_TO_MCU_GPIO_INTR_3 • NPSS_TO_MCU_GPIO_INTR_4 • NPSS_TO_MCU_CMP_INTR_1 • NPSS_TO_MCU_CMP_INTR_2 • NPSS_TO_MCU_CMP_INTR_3 • NPSS_TO_MCU_CMP_INTR_4 • NPSS_TO_MCU_RFWAKEUP_INTR • NPSS_TO_MCU_BOD_INTR • NPSS_TO_MCU_BUTTON_INTR • NPSS_TO_MCU_SDC_INTR

Parameter	Description
	• NPSS_TO_MCU_WIRELESS_INTR • NPSS_TO_MCU_WAKEUP_INTR • NPSS_TO_MCU_ALARM_INTR • NPSS_TO_MCU_SEC_INTR • NPSS_TO_MCU_MSEC_INTR • NPSS_TO_MCU_PROCESSOR_INTR • NPSS_TO_MCU_HOST_INTR • NPSS_TO_MCU_DST_INTR

Return values

None

Example

```
RSI_PS_NpssIntrMask (NPSS_TO_MCU_RFWAKEUP_INTR);
```

30.3.39 RSI_PS_EnableLpSleep

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_PS_EnableLpSleep (boolean_t lpSleep)
```

Description

This API is used to enable/disable the lp sleep mode.

Parameter

Parameter	Description
lpSleep	1:enable lp sleep , 0 : disable lp sleep MCU_FSM_SLEEP_CTRLS_AND_WAKEUP_MODE_b

Return values

None

Example

```
#define      Enable    1
/* enable lp sleep */
RSI_PS_EnableLpSleep(Enable);
```

30.3.40 RSI_PS_SkipXtalWaitTime

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_PS_SkipXtalWaitTime (boolean_t val)
```

Description

This API is used to skip the XTAL wait time.

Parameter

Parameter	Description
val	1: skip XTAL wait time 0 Do not skip XTAL wait time MCU_FSM_SLEEP_CTRLS_AND_WAKEUP_MODE_b

Return values

none

Example

```
/*Enable the skip XTAL wait time*/
RSI_PS_SkipXtalWaitTime(1);
```

30.3.41 RSI_PS_EnterShipMode

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
void RSI_PS_EnterShipMode (uint8_t wakeUpDelay, boolean_t gpioBased, SHUT_DOWN_WKP_MODE_T
gpioSel)
```

Description

This API is used to keep MCU in shelf mode.

Parameters

Parameter	Description
wakeUpDelay	wake up delay (Possible values : 0, 1, 2, ... 7 ,Delay will be 2^8 = 256 clock with 32Khz RC clock.)
gpioBased	GPIO based Shelf mode Entering. If set '1' by processor, On Falling edge of NPSS GPIO
gpioSel	Post Division factor2. See user manual for more info. • NPSS_GPIO_2_BASED • NPSS_GPIO_1_BASED • NPSS_GPIO_2_AND_3_BASED • NPSS_GPIO_2_OR_3_BASED

Return values

None

Example

```
RSI_PS_EnterShipMode(1,1,NPSS_GPIO_2_BASED );
```

30.3.42 RSI_PS_UlpToDcDcMode

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_PS_UlpToDcDcMode(void)
```

Description

This API is configures SC-DCDC from LDO to DCDC Mode

Return values

none

Example

```
/*Configures SC-DCDC from LDO to DCDC Mode */
RSI_PS_UlpToDcDcMode();
```

30.3.43 RSI_PS_BodPwrGateButtonCalibEnable

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_PS_BodPwrGateButtonCalibEnable(void)
```

Description

This API is used to enable the power-gate enable signal for button calib and vbatt status checking block

Return values

none

Example

```
/*Enable the power-gate enable signal for button calib and vbatt status checking block */
RSI_PS_BodPwrGateButtonCalibEnable();
```

30.3.44 RSI_PS_BodPwrGateButtonCalibDisable

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_PS_BodPwrGateButtonCalibDisable(void)
```

Description

This API is used to disable the power-gate enable signal for button calib and vbatt status checking block

Return values

none

Example

```
/*Disable the power-gate enable signal for button calib and vbatt status checking block */
RSI_PS_BodPwrGateButtonCalibDisable();
```

30.3.45 RSI_PS_AnalogPeriPtatEnable

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_PS_AnalogPeriPtatEnable(void)
```

Description

This API is used to enable the ptat currents to analog peripherals

Return values

none

Example

```
/*Enable the ptat currents to analog peripherals */
RSI_PS_AnalogPeriPtatEnable();
```

30.3.46 RSI_PS_AnalogPeriPtatDisable

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_PS_AnalogPeriPtatDisable(void)
```

Description

This API is used to disable the ptat currents to analog peripherals

Return values

none

Example

```
/*Disable the ptat currents to analog peripherals */
RSI_PS_AnalogPeriPtatDisable();
```

30.3.47 RSI_PS_BodClksPtatEnable

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_PS_BodClksPtatEnable(void)
```

Description

This API is used to enable the ptat currents to clocks and bod(cmp_npss)

Return values

none

Example

```
/*Enable the ptat currents to clocks and bod(cmp_npss) */
RSI_PS_BodClksPtatEnable();
```

30.3.48 RSI_PS_BodClksPtatDisable

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_PS_BodClksPtatDisable(void)
```

Description

This API is used to disable the ptat currents to clocks and bod(cmp_npss)

Return values

none

Example

```
/*Disable the ptat currents to clocks and bod(cmp_npss) */
RSI_PS_BodClksPtatDisable();
```

30.3.49 RSI_PS_XtalEnable

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_PS_XtalEnable(void)
```

Description

This API is used to enable the XTAL

Return values

none

Example

```
/*Enable the XTAL */
```

```
RSI_PS_XtalEnable();
```

30.3.50 RSI_PS_XtalDisable

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_PS_XtalDisable(void)
```

Description

This API is used to disable the XTAL

Return values

none

Example

```
/*Disable the XTAL */
RSI_PS_XtalDisable();
```

30.3.51 RSI_PS_FlashLdoEnable

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_PS_FlashLdoEnable(void)
```

Description

This API is used to enable flash LDO

Return values

none

Example

```
/*Enable flash LDO  */
RSI_PS_FlashLdoEnable();
```

30.3.52 RSI_PS_FlashLdoDisable

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_PS_FlashLdoDisable(void)
```

Description

This API is used to disable flash LDO.

Return values

none

Example

```
/*Disable flash LDO.*/
RSI_PS_FlashLdoDisable();
```

30.3.53 RSI_PS_SocPllSpiDisable

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_PS_SocPllSpiDisable(void)
```

Description

This API is used to disable the Soc-PLL SPI PG

Return values

none

Example

```
/*Disable the Soc-PLL SPI PG */
 RSI_PS_SocPllSpiDisable();
```

30.3.54 RSI_PS_SocPllVddIsoEnable

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void  RSI_PS_SocPllVddIsoEnable(void)
```

Description

This API is used to enable the Soc-PLL ISO VDDG

Return values

none

Example

```
/*Enable the Soc-PLL SPI PG */
 RSI_PS_SocPllVddIsoEnable();
```

30.3.55 RSI_PS_SocPllVddIsoDiable

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_PS_SocPllVddIsoDiable(void)
```

Description

This API is used to disable the Soc-PLL ISO VDD

Return values

none

Example

```
/*Disable the Soc-PLL ISO VDD */
RSI_PS_SocPllVddIsoDiable();
```

30.4 RSI_PS_SocPllSpiEnable

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_PS_SocPllSpiEnable(void)
```

Description

This API is used to enable the Soc-PLL SPI PG

Return values

none

Example

```
/*Enable the Soc-PLL SPI PG */
RSI_PS_SocPllSpiEnable();
```

30.4.1 RSI_ConfigBuckBoost

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_ConfigBuckBoost(uint8_t cntrl , uint8_t enable)
```

Parameters

Parameter	Description
cntrl	0: Software controlled 1: Hardware controlled.
enable	0: Disabled if controlled by software(cntrl = 0) 1: Enabled if controlled by software(cntrl = 1)

Description

This API is used to to control the buck boost.

Return values

none

Example

```
/*  Disable by software */
RSI_ConfigBuckBoost(1,0);
```

30.4.2 RSI_ConfigPmuShutDown

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_ConfigPmuShutDown(uint8_t cntrl , uint8_t enable)
```

Parameters

Parameter	Description
cntrl	0: Software controlled 1: Hardware controlled.
enable	0: Disabled if controlled by software(cntrl = 0) 1: Enabled if controlled by software(cntrl = 1)

Description

This API is used to control the pmu shut down mode

Return values

none

Example

```
/* Disable by software  */
RSI_ConfigPmuShutDown(1,0);
```

30.4.3 RSI_ConfigXtal

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_ConfigXtal(uint8_t cntrl , uint8_t enable)
```

Parameters

Parameter	Description
cntrl	0: Software controlled 1: Hardware controlled.
enable	0: Disabled if controlled by software(cntrl = 0) 1: Enabled if controlled by software(cntrl = 1)

Description

This API is used to control the Xtal

Return values

none

Example

```
/* Disable by software  */
RSI_ConfigXtal(1,0);
```

30.4.4 RSI_PS_PmuSetConfig

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_PS_PmuSetConfig(uint32_t  mask)
```

Parameters

Parameter	Description
mask	Ored values of PMU status bits Following are the possible parameters for this parameter • PMU_STS_DCDC_ON • PMU_STS_FLASH_LDO_ON • PMU_STS_SOC_LDO_ON

Description

This API is used to enable/set the PMU status

Return values

none

Example

```
/*Enable PMU STS DCDC ON and PMU STS FLASH LDO ON */
RSI_PS_PmuSetConfig(PMU_STS_DCDC_ON |PMU_STS_FLASH_LDO_ON);
```

30.4.5 RSI_PS_PmuClrConfig

Source File : rsi_power_save.h

Path : Redpine_MCU_Vx.y.z\l\Host_MCU\Peripheral_Library\systemlevel\inc

Prototype

```
STATIC INLINE void RSI_PS_PmuClrConfig(uint32_t  mask)
```

Parameters

Parameter	Description
mask	Ored values of PMU status bits Following are the possible parameters for this parameter • PMU_STS_DCDC_ON • PMU_STS_FLASH_LDO_ON • PMU_STS_SOC_LDO_ON

Description

This API is used to disable/clear the PMU status

Return values

none

Example

```
/*Disable PMU STS DCDC ON and PMU STS FLASH LDO ON */
RSI_PS_PmuClrConfig(PMU_STS_DCDC_ON |PMU_STS_FLASH_LDO_ON);
```

31 Comparator

31.1 Overview

This section explains how to configure and use the Comparator using Redpine MCU SAPIs.

31.2 Programming Sequence

```
#include "rsi_chip.h"  /* Redpine_MCU_Vx.y.z\Host_MCU\common\chip\inc */

void IRQ008_Handler(void)
{
   NVIC_DisableIRQ(COMP1_IRQn);
}
/**
 * @brief  This function define pin mux for comparator.
 * @param  mode    :  Type of mode (Differential or single mode)
 * @retval None
 */
void comparator_pinmux(void)
{
    RSI_EGPIO_PadReceiverDisable(AGPIO_PIN0);
    RSI_EGPIO_PadReceiverDisable(AGPIO_PIN1);
    RSI_EGPIO_PadReceiverDisable(AGPIO_PIN9);

    RSI_EGPIO_SetPinMux(EGPIO1,AGPIO_PORT ,AGPIO_PIN0,AGPIO_MODE);       /* non-inverting
external input pin */
    RSI_EGPIO_SetPinMux(EGPIO1,AGPIO_PORT ,AGPIO_PIN1,AGPIO_MODE);    /*  inverting
external input pin */

    RSI_EGPIO_SetPinMux(EGPIO1,COMP_OUTPUT_PORT,COMP_OUTPUT_PIN,COMP_OUTPUT_MODE); /*
comparator1 output pin */
}

int main()
{
    /*Configures the system default clock and power configurations*/
     SystemCoreClockUpdate();
    /* Configure the AUX clock source */
    RSI_ULPSS_AuxClkConfig(ULPCLK, ENABLE_STATIC_CLK ,ULP_AUX_32MHZ_RC_CLK);

    /*  Address of a target register ULP  */
    ULP_SPI_MEM_MAP(BG_SCDC_PROG_REG1)|=BIT(3);

    /* Bypass ldo disable */
    RSI_ADC_AUXLdoConfig(AUX_ADC_DAC_COMP,0,0xb);

    /* Disable the receiver enable bit(REN) and configures the pin to Comparator */
    comparator_pinmux();

    /* Configure the comparator parameters. */
    RSI_COMP_Config(AUX_ADC_DAC_COMP,COMP1,COMP1_POSITIVE_INPUT,COMP1_NEGATIVE_INPUT,
                    HYST_ENABLE,FILTER_ENABLE);

    /* enable  the comparator */
    RSI_COMP_Enable(AUX_ADC_DAC_COMP,COMP1,Enable);

    /* enables a device-specific interrupt in the NVIC interrupt controller of comparator
. */
    NVIC_EnableIRQ(COMP1_IRQn);

    while(1)
    {
        /* enables a device-specific interrupt in the NVIC interrupt controller of
comparator . */
       NVIC_EnableIRQ(COMP1_IRQn);
    }
```

```
}
```

31.3 API Descriptions

31.3.1 RSI_COMP_Config

Source File : rsi_comparator.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_COMP_Config(AUX_ADC_DAC_COMP_Type* comp      , uint8_t comp_number , uint8_t
sel_p_mux, uint8_t    sel_n_mux , uint8_t hyst_en     , uint8_t filter_en)
```

Description

This API is used to configure comparator.

Parameters

Parameter	Description
comp	Pointer to the comparator register instance.
comp_number	comparator no to be enabled or disabled • 1:for comparator 1 • 2:for comparator 2
sel_p_mux	select positive input for comparator • 0: compx_p0 x is 1 or 2 i.e comp1 and comp2 • 1: compx_p1 • 2: DAC • 3: reference buffer out • 4: reference scalor out • 5: resistor bank out • 6: opamp1_out • 7: opamp2_out • 8:opamp3_out
sel_n_mux	select negative input for comparator • 0: compx_p0 x is 1 or 2 i.e comp1 and comp2 • 1: compx_p1 • 2: DAC • 3: reference buffer out • 4: reference scalor out • 5: resistor bank out • 6: opamp1_out

Parameter	Description
	• 7: opamp2_out • 8:opamp3_out
hyst_en	control the hysteresis of the comparator(0 to 4)
filter_en	To enable filter for comparator • 1:enable • 0:disable

Return values

RSI_OK If valid comp else value return error code.

Example

```
#define    COMP1      1
#define    COMP2      2
#define    Enable     1
#define    Disable    0

/* configure comparator */
 RSI_COMP_Config(AUX_ADC_DAC_COMP,COMP1,0,0,1,Enable);
```

31.3.2 RSI_COMP_Enable

Source File : rsi_comparator.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_COMP_Enable(AUX_ADC_DAC_COMP_Type* comp,uint8_t comp_number, uint8_t enable)
```

Description

This API is used to enable and disable the comparator.

Parameters

Parameter	Description
comp	Pointer to the comparator register instance.
comp_number	comparator no to be enabled or disabled • 1:for comparator 1 • 2:for comparator 2
enable	enable and disable comp • 1:enable • 0:disable

Return values

RSI_OK If valid comparator else non_zero value

Example

```
#define    COMP1      1
#define    COMP2      2
```

```
#define    Enable     1
#define    Disable    0

/* enable comparator */
RSI_COMP_Enable(COMP1,Enable);

/* disable comparator */
RSI_COMP_Enable(AUX_ADC_DAC_COMP,Disable);
```

31.3.3 RSI_COMP_ResistorBank_Enable

Source File : rsi_comparator.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
void RSI_COMP_ResistorBank_Enable(AUX_ADC_DAC_COMP_Type* comp,uint8_t enable)
```

Description

This API is used to enable the Register bank output.

Parameters

Parameter	Description
comp	Pointer to the comparator register instance
enable	Enable the register bank. • 1:enable

Return values

None

Example

```
#define    Enable     1
#define    Disable    0

/* enable register bank output */
RSI_COMP_ResistorBank_Enable(AUX_ADC_DAC_COMP,Enable);
```

31.3.4 RSI_COMP_ResistorBank_Thrsh

Source File : rsi_comparator.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_COMP_ResistorBank_Thrsh(AUX_ADC_DAC_COMP_Type* comp,uint16_t value_thrsh)
```

Description

This API is used to set register bank threshold value.

Parameters

Parameter	Description
comp	Pointer to the comparator register instance
value_thrsh	Value of register threshold value. This value the output of BOD , and that output is one of input for comparator.

Return values

Return zero on success , return error code on failure.

Example

```
#define    value_thrsh    4
/* Set register bank threshold value */
RSI_COMP_ResistorBank_Thrsh(AUX_ADC_DAC_COMP,value_thrsh);
```

31.3.5 RSI_COMP_ReferenceScalarOut

Source File : rsi_comparator.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_COMP_ReferenceScalarOut(AUX_ADC_DAC_COMP_Type* comp,uint16_t
scalar_factor_value)
```

Description

This API is used to set register bank threshold value.

Parameters

Parameter	Description
comp	Pointer to the comparator register instance
scalar_factor_value	Set reference scalar factor value.

Return values

Return zero on success , return error code on failure.

Example

```
#define    scalar_factor_value  10
/* Set register bank threshold value */
RSI_COMP_ReferenceScalarOut(AUX_ADC_DAC_COMP,scalar_factor_value);
```

31.3.6 RSI_COMP_RefenceBufferOut_Enable

Source File : rsi_comparator.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
void RSI_COMP_RefenceBufferOut_Enable(AUX_ADC_DAC_COMP_Type* comp,uint8_t enable)
```

Description

This API is used to enable reference buffer output.

Parameters

Parameter	Description
comp	Pointer to the comparator register instance
enable	Set reference scalar factor value. • 1 : Enable

Return values

Return zero on success , return error code on failure.

Example

```
#define    enable  1
/* Set register bank threshold value */
RSI_COMP_RefenceBufferOut_Enable(AUX_ADC_DAC_COMP,enable);
```

32 PUF

32.1 Overview

This section explains how to configure and use the PUF using Redpine MCU SAPIs.

32.2 Programming Sequence

```
int main()
{
    int32_t status;

    status = rsi_puf_enroll_req();
    if(status != RSI_SUCCESS)
    {
#ifdef RSI_ENABLE_DEBUG_PRINT
        printf("\n PUF Start Enroll Fail/Blocked Status %d \n", status);
#endif
        return status;
    }
#ifdef RSI_ENABLE_DEBUG_PRINT
    printf("\n PUF Enroll Success \n");
#endif

#if RSI_PUF_ENROLL_DIS
    //! Disable Enroll
    status = rsi_puf_enroll_disable_req();
    if(status != RSI_SUCCESS)
    {
#ifdef RSI_ENABLE_DEBUG_PRINT
        printf("\n Disable Enroll Fail Status %d \n", status);
#endif
        return status;
    }
#ifdef RSI_ENABLE_DEBUG_PRINT
    printf("\n PUF Further Enroll Disabled \n");
#endif
#endif

#if RSI_PUF_SET_KEY_DIS
    //! Disable set key
    status = rsi_puf_set_key_disable_req();
    if(status != RSI_SUCCESS)
    {
#ifdef RSI_ENABLE_DEBUG_PRINT
        printf("\n Disable Set Key Fail Status %d \n", status);
#endif
        return status;
    }
#ifdef RSI_ENABLE_DEBUG_PRINT
    printf("\n PUF Set Key Disabled \n");
#endif
#endif

#if RSI_PUF_GET_KEY_DIS
    //! Disable Get Key
    status = rsi_puf_get_key_disable_req();
    if(status != RSI_SUCCESS)
    {
#ifdef RSI_ENABLE_DEBUG_PRINT
        printf("\n Disable Get Key Fail Status %d \n", status);
#endif
        return status;
    }
#ifdef RSI_ENABLE_DEBUG_PRINT
    printf("\n PUF Get Key Disabled \n");
#endif
```

```
#endif

    return 0;
}
```

Code Block 24 PUF Example

32.2.1 rsi_puf_enroll_req

Prototype

```
int32_t rsi_puf_enroll_req(void)
```

Description

This API enrolls PUF. The API upon success will save activation code on flash. The stored activation code shall be used for every further start operation on PUF.

Parameters

None

Return Values

Value	Description
0	Successful execution of the command
Non Zero	**If return value is lesser than 0** -3 : Command given in wrong state -4 : Buffer not available to serve the command **If return value is greater than 0** 0xCC2F, 0xCC33, 0xCC34, 0xCC35

Example

```
status = rsi_puf_enroll_req();
```

32.2.2 rsi_puf_enroll_disable_req

Prototype

```
int32_t rsi_puf_enroll_disable_req(void)
```

Description

This API blocks the further enrollment of PUF.

Parameters

None

Return Values

Value	Description
0	Successful execution of the command
Non Zero	**If return value is lesser than 0**

Value	Description
	-3 : Command given in wrong state -4 : Buffer not available to serve the command **If return value is greater than 0** 0xCC32, 0xCC33, 0xCC34

Example

```
status = rsi_puf_enroll_disable_req();
```

32.2.3 rsi_puf_start_req

Prototype

```
int32_t rsi_puf_start_req(void)
```

Description

This API starts the PUF if valid activation code is available in the flash. If activation code on flash is valid, enrollmemnt operation returns success or else fails to start PUF. Start operation is must for any further operation with PUF.

Parameters

None

Return Values

Value	Description
0	Successful execution of the command
Non Zero	**If return value is lesser than 0** -3 : Command given in wrong state -4 : Buffer not available to serve the command **If return value is greater than 0** 0xCC31, 0xCC32, 0xCC33, 0xCC35

Example

```
status = rsi_puf_start_req();
```

32.2.4 rsi_puf_set_key_req

Prototype

```
int32_t rsi_puf_set_key_req(uint8_t key_index,uint16_t key_size,uint8_t *key_ptr,uint8_t
*set_key_resp,uint16_t length)
```

Description

This API is used to request for a set of key operation for the given key, a Key code is generated by PUF which is returned if operation is success. This API will return failure if there is a failure in system or if this feature is blocked prior.

Parameters

Parameter	Description

Parameter	Description
Key_index	Key Index of Key to be generated (0-15) To increase the randomness
key_size	Key Size in bytes, 0 : 128bit key 1 : 256bit key
key_ptr	This is the pointer to key provided by application
set_key_resp	This is the keycode to the key provided. This is an output parameter.
Length	This is the length of the result buffer in bytes to hold keycode.

Return Values

Value	Description
0	Successful execution of the command
Non Zero	**If return value is lesser than 0** -2 : Invalid parameters -3 : Command given in wrong state -4 : Buffer not available to serve the command **If return value is greater than 0** 0xCC2F, 0xCC32, 0xCC33, 0xCC34

Example

```
status = rsi_puf_set_key_req(1, PUF_KEY_SIZE_128 , RSI_AES_KEY, keycode,
KEY_CODE_SIZE_BYTES );
```

32.2.5 rsi_puf_set_key_disable_req

Prototype

```
int32_t rsi_puf_set_key_disable_req(void)
```

Description

This API blocks the set key for further operations on PUF.

Parameters

None

Return Values

Value	Description
0	Successful execution of the command
Non Zero	**If return value is lesser than 0** -3 : Command given in wrong state -4 : Buffer not available to serve the command **If return value is greater than 0** 0xCC32, 0xCC33, 0xCC34

Example

```
status = rsi_puf_set_key_disable_req();
```

32.2.6 rsi_puf_get_key_req

Prototype

```
int32_t rsi_puf_get_key_req(uint8_t *keycode_ptr,uint8_t *get_key_resp,uint16_t length)
```

Description

This API regenerates the key for the given key code using PUF. If operation is success, key is returned or else error is returned. This API will return failure if there is a failure in system or if this feature is blocked prior.

Parameters

Parameter	Description
keycode_ptr	This is the pointer to KeyCode
get_key_resp	This is the pointer to key, this is an output parameter
length	This is the length of the result buffer in bytes to hold key.

Return Values

Value	Description
0	Successful execution of the command
Non Zero	**If return value is lesser than 0** -2 : Invalid Parameters -3 : Command given in wrong state -4 : Buffer not available to serve the command **If return value is greater than 0** 0xCC2F, 0xCC32, 0xCC33, 0xCC34

Example

```
status = rsi_puf_get_key_req(keycode, get_key, KEY_CODE_SIZE_BYTES );
```

32.2.7 rsi_puf_get_key_disable_req

Prototype

```
int32_t rsi_puf_get_key_disable_req(void)
```

Description

This API blocks the further get key operations on PUF.

Parameters

None

Return Values

Value	Description
0	Successful execution of the command
Non Zero	**If return value is lesser than 0** -3 : Command given in wrong state -4 : Buffer not available to serve the command **If return value is greater than 0** 0xCC32, 0xCC33, 0xCC34

Example

```
status = rsi_puf_get_key_disable_req();
```

32.2.8 rsi_puf_load_key_req

Prototype

```
int32_t rsi_puf_load_key_req(uint8_t *keycode_ptr)
```

Description

This API regenerates the key for the given key code using PUF, and loads it into AES engine. If operation is success, key is loaded into AES or else error is returned. This API will return failure if there is a failure in system or if this feature is blocked prior.

Parameters

Parameter	Description
key_ptr	This is the pointer to keycode provided by an application

Return Values

Value	Description
0	Successful execution of the command
Non Zero	**If return value is lesser than 0** -2 : Invalid parameters -3 : Command given in wrong state -4 : Buffer not available to serve the command **If return value is greater than 0** 0xCC2F, 0xCC32, 0xCC33, 0xCC34

Example

```
status = rsi_puf_load_key_req(keycode);
```

32.2.9 rsi_puf_intr_key_req

Prototype

```
int32_t rsi_puf_set_key_req(uint8_t key_index,uint8_t key_size,uint8_t
*intr_key_resp,uint16_t length)
```

Description

This API requests for intrinsic key operation for the given keysize , a Key code is generated by PUF which is returned if operation is success. This API will return failure if there is a failure in system or if this feature is blocked prior.

Parameters

Parameter	Description
Key_index	This is the key index of the Key to be generated (0-15) To increase the randomness
key_size	This is the key size in bytes, 0 : 128bit key 1 : 256bit key
set_key_resp	This is the keycode to the key provided. This is an output parameter.
Length	This is the length of the result buffer in bytes to hold keycode.

Return Values

Value	Description
0	Successful execution of the command
Non Zero	**If return value is lesser than 0** -2 : Invalid parameters -3 : Command given in wrong state -4 : Buffer not available to serve the command **If return value is greater than 0** 0xCC2F, 0xCC32, 0xCC33, 0xCC34

Example

```
status = rsi_puf_intr_key_req(2, PUF_KEY_SIZE_128 , keycodeI, KEY_CODE_SIZE_BYTES );
```

32.2.10 rsi_puf_aes_encrypt_req

Prototype

```
int32_t rsi_puf_aes_encrypt_req(uint8_t mode,uint8_t key_source,uint16_t key_size,uint8_t
*key_ptr,uint16_t data_size,uint8_t *data_ptr,uint16_t iv_size,uint8_t *iv_ptr,uint8_t
*aes_encry_resp,uint16_t length)
```

Description

This API encrypts data inputted with Key provided or with key which is already loaded into AES by PUF. This API provides provision for encryption with AES engine into modes (ECB, CBC). Parameters should be provided to API depending on mode of usage. API will return failure if there is an error in input.

Parameters

Parameter	Description

Parameter	Description
mode	This is AES encryption mode 0 : ECB 1 : CBC
Key_source	This is the encyption key source, 1 : AES engine, provided by application as key_ptr 0 : PUF
key_size	This is the Key Size in bytes, 0 : 128bit key 1 : 256bit key
key_ptr	This is the pointer to the key provided by application
data_size	This is the size of data in bytes
data_ptr	This is the pointer to data to be encrypted
iv_size	This is the intialization vector size(if CBC mode) 0 : 128bit key
iv_ptr	This is the pointer to IV (if CBC mode)
Aes_encry_resp	This is the pointer to the encrypted data. This is an output parameter.
length	This is the length of the resultant buffer in bytes to hold encrypted data.

Return Values

Value	Description
0	Successful execution of the command
Non Zero	**If return value is lesser than 0** -2 : Invalid parameters -3 : Command given in wrong state -4 : Buffer not available to serve the command **If return value is greater than 0** 0xCC32

Example

```
status = rsi_puf_aes_encrypt_req(AES_ECB_MODE, AES_AS_KEY_SOURCE, PUF_KEY_SIZE_128,
RSI_AES_KEY , 16, RSI_AES_PLAIN_TXT , 0, NULL, aes_encry_data , 16);
```

32.2.11 rsi_puf_aes_decrypt_req

Prototype

```
int32_t rsi_puf_aes_decrypt_req (uint8_t mode,uint8_t key_source,uint16_t key_size,uint8_t
*key_ptr,uint16_t data_size,uint8_t *data_ptr,uint16_t iv_size,uint8_t *iv_ptr,uint8_t
*aes_decry_resp,uint16_t length)
```

Description

This API decrypts data inputted with Key provided or with key which is already loaded into AES by PUF. This API provides provision for decryption with AES engine in two modes (ECB, CBC). Parameters should be provided to API depending on mode of usage. API will return failure if there is any error in input.

Parameters

Parameter	Description
mode	This is the AES encryption mode 0 : ECB 1 : CBC
Key_source	This is the decryption key source, 1 : AES engine, provided by application as key_ptr 0 : PUF
key_size	This is the Key Size in bytes, 0 : 128bit key 1 : 256bit key
key_ptr	This is the pointer to key provided by application
data_size	This is the size of data in bytes
data_ptr	This is the pointer to data to be decrypted
iv_size	This is the intialization vector size (if CBC mode) 0 : 128bit key
iv_ptr	This is the pointer to IV (if CBC mode)
aes_decry_resp	This is the pointer to the decrypted data. This is an output parameter.
length	This is the length of the resultant buffer in bytes to hold decrypted data.

Return Values

Value	Description
0	Successful execution of the command
Non Zero	**If return value is lesser than 0** -2 : Invalid parameters -3 : Command given in wrong state -4 : Buffer not available to serve the command **If return value is greater than 0** 0xCC32

Example

```
status = rsi_puf_aes_decrypt_req(AES_ECB_MODE, AES_AS_KEY_SOURCE, PUF_KEY_SIZE_128,
RSI_AES_KEY , 16, aes_encry_data , 0, NULL, aes_decry_data , 16 );
```

32.2.12 rsi_puf_aes_mac_req

Prototype

```
int32_t rsi_puf_aes_mac_req (uint8_t key_source, uint16_t key_size,uint8_t
```

```
*key_ptr,uint16_t data_size,uint8_t *data_ptr,uint16_t iv_size,uint8_t *iv_ptr,uint8_t
*aes_mac_resp,uint16_t length)
```

Description

This API generates Message authentication check (MAC) for the data inputted with provided key as well as Initialization Vector (IV). Parameters should be provided to API depending on mode of usage. API will return failure if there is any error in input.

Parameters

Parameter	Description
Key_source	This is the key source to generate MAC, 1 : provided by application as key_ptr 0 : PUF
key_size	This is the Key Size in bytes, 0 : 128bit key 1 : 256bit key
key_ptr	This is the pointer to the key provided by application
data_size	This is the size of data in bytes
data_ptr	This is the pointer to Data
iv_size	This is the intialization of vector size (if CBC mode) 0 : 128bit key
iv_ptr	This is the pointer to IV(if CBC mode)
aes_mac_resp	This is the pointer to MAC data, this is an output parameter.
length	This is the length of the result buffer in bytes to hold MAC data.

Return Values

Value	Description
0	Successful execution of the command
Non Zero	**If return value is lesser than 0** -2 : Invalid parameters -3 : Command given in wrong state -4 : Buffer not available to serve the command **If return value is greater than 0** 0xcc32

Example

```
status = rsi_puf_aes_mac_req(AES_AS_KEY_SOURCE, PUF_KEY_SIZE_128, RSI_AES_KEY , (16),
RSI_AES_PLAIN_TXT, PUF_IV_SIZE_128, RSI_AES_CBC_IV , aes_mac , 16);
```

33 Crypto

This section explains hardware security accelerator such as AES,ECDH,Exponentaition,HMAC-SHA and SHA.

33.1 AES

33.1.1 Overview

This section explains how to configure and use the AES using Redpine MCU SAPIs.

33.1.2 Programming Sequence

```
int main()
{
    int32_t status;
#if 1
  memset(aes_encry_data, 0, 256); //memset response buffers
  status = rsi_aes(CTR_MODE, AES_ENCRYPTION, msg, sizeof(msg), key, sizeof(key), iv,
aes_encry_data);
  if(status != RSI_SUCCESS)
  {
#ifdef RSI_ENABLE_DEBUG_PRINT
    printf("\n AES Encryption Fail Status %d \n", status);
#endif
    return status;
  }
#ifdef RSI_ENABLE_DEBUG_PRINT
  printf("\n Encryption Done with AES ECB 32bytes key, Data Size 256 Bytes \n");
#endif

  //!Decrypt using AES-CTR mode, 128bit key
  memset(aes_decry_data, 0, 256); //memset response buffers
  status = rsi_aes(CTR_MODE, AES_DECRYPTION, aes_encry_data, sizeof(msg) , key,
sizeof(key), iv, aes_decry_data);
  //status = rsi_aes(CTR_MODE, AES_DECRYPTION, aes_encry_data, strlen(RSI_AES_PLAIN_TXT) ,
RSI_AES_KEY, AES_KEY_SIZE_256, NULL, aes_decry_data);
  if(status != RSI_SUCCESS)
  {
#ifdef RSI_ENABLE_DEBUG_PRINT
    printf("\n AES Decryption Fail Status %d \n", status);
#endif
    return status;
  }
#ifdef RSI_ENABLE_DEBUG_PRINT
  printf("\n Decryption Done with AES ECB 32bytes key, Data Size 256 \n ");
#endif
#endif

  return 0;
}
```

33.1.2.1 rsi_aes

Prototype

```
int32_t rsi_aes(uint8_t aes_mode, uint8_t enc_dec, uint8_t *msg,uint32_t msg_length,
uint8_t *key, uint32_t key_length, uint8_t *output)
```

Description

This API is used to encryption and decryption .

Parameters

Parameter	Description
aes_mode	AES mode 1 – For AES CBC mode 2 – For AES ECB mode 3 – For AES CTR mode
enc_dec	1 – For AES Encryption 2 – For AES Decryption
msg	Pointer to message
msg_length	Total message length in bytes
key	Pointer to AES key
key_length	AES key length in bytes 16 – For AES 128 mode 24 – For AES 192 mode 32 – For AES 256 mode
output	Output parameter to hold AES encrypted/decrypted data

Return values

return zero on success non zero on failure.

Example

```
  status = rsi_aes(ECB_MODE, AES_DECRYPTION, aes_encry_data, sizeof(msg) , key2,
AES_KEY_SIZE_256, NULL, aes_decry_data);
```

33.2 ECDH

33.2.1 Overview

This section explains how to configure and use the ECDH using Redpine MCU SAPIs.

33.2.2 Programming Sequence

```
int main()
{
    int32_t status;
    status = rsi_crypto_ecdh_app();
   return status;
 }
int32_t rsi_crypto_ecdh_app()
{
    int32_t status = RSI_SUCCESS;

    //! Buffers to store responses
    uint8_t rx[32] = {0};
    uint8_t ry[32] = {0};
    uint8_t rz[32] = {0};

    //! WC initialization
    status = rsi_wireless_init(0, 0);
```

```
    if(status != RSI_SUCCESS)
    {
        return status;
    }

#if 0
    status = rsi_ecdh_point_multiplication(ECDH_224, d, sx, sy, sz, rx, ry, rz);
    if(status != RSI_SUCCESS)
    {
#ifdef RSI_ENABLE_DEBUG_PRINT
        printf("\n ECDH multiplication Fail Status %d \n", status);
#endif
        return status;
    }
#ifdef RSI_ENABLE_DEBUG_PRINT
    printf("\n ECDH multiplication Done \n");
#endif
#endif

#if 0
    status = rsi_ecdh_point_addition(ECDH_224, sx, sy, sz, tx, ty, tz, rx, ry, rz);
    if(status != RSI_SUCCESS)
    {
#ifdef RSI_ENABLE_DEBUG_PRINT
        printf("\n ECDH addition Fail Status %d \n", status);
#endif
        return status;
    }
#ifdef RSI_ENABLE_DEBUG_PRINT
    printf("\n ECDH addition Done \n");
#endif
#endif

#if 0
    status = rsi_ecdh_point_subtraction(ECDH_192, sx, sy, sz, tx, ty, tz, rx, ry, rz);
    if(status != RSI_SUCCESS)
    {
#ifdef RSI_ENABLE_DEBUG_PRINT
        printf("\n ECDH subtraction Fail Status %d \n", status);
#endif
        return status;
    }
#ifdef RSI_ENABLE_DEBUG_PRINT
    printf("\n ECDH subtraction Done \n");
#endif
#endif

#if 0
    status = rsi_ecdh_point_double(ECDH_224, sx, sy, sz, rx, ry, rz);
    if(status != RSI_SUCCESS)
    {
#ifdef RSI_ENABLE_DEBUG_PRINT
        printf("\n ECDH point double Fail Status %d \n", status);
#endif
        return status;
    }
#ifdef RSI_ENABLE_DEBUG_PRINT
    printf("\n ECDH point double Done \n");
#endif
#endif

#if 1
  status = rsi_ecdh_point_affine(ECDH_192, sx, sy, sz, rx, ry, rz);
    //status = rsi_ecdh_point_affine(ECDH_192, rx, ry, rz, rx, ry, rz);
    if(status != RSI_SUCCESS)
    {
#ifdef RSI_ENABLE_DEBUG_PRINT
        printf("\n ECDH affinity Fail Status %d \n", status);
#endif
        return status;
    }
```

```
#ifdef RSI_ENABLE_DEBUG_PRINT
    printf("\n ECDH affinity Done \n");
#endif
#endif

    return 0;
}
```

33.2.2.1 rsi_ecdh_point_multiplication

Prototype

```
int32_t rsi_ecdh_point_multiplication(uint8_t ecdh_mode, uint8_t *d, uint8_t *sx, uint8_t
*sy, uint8_t *sz,
uint8_t *rx, uint8_t *ry, uint8_t *rz)
```

Description

This API is used to ecdh point multiplication.

Parameters

Parameter	Description
ecdh_mode	• 1 – For ECDH 192 • 2 – For ECDH 224 • 3 – For ECDH 256
d	Pointer to scalar value that needs to be multiplied
sx	Pointers to x, y, z coordinates of the point to be multiplied with scalar 'd'
sy	Pointers to x, y, z coordinates of the result point

Return values

return zero on success non zero on failure.

Example

```
status = rsi_ecdh_point_multiplication(ECDH_224, d, sx, sy, sz, rx, ry, rz);
```

33.2.3 Point Addition

33.2.3.1 rsi_ecdh_point_addition

Prototype

```
int32_t rsi_ecdh_point_addition(uint8_t ecdh_mode,uint8_t *sx, uint8_t *sy,
uint8_t*sz,uint8_t *tx, uint8_t *ty, uint8_t *tz,uint8_t *rx, uint8_t *ry, uint8_t *rz)
```

Description

This API is used to ecdh point addition.

Parameters

Parameter	Description

Parameter	Description
ecdh_mode	• 1 – For ECDH 192 • 2 – For ECDH 224 • 3 – For ECDH 256
sx	Pointers to x, y, z coordinates of the point1 that needs to be added.
sy	Pointers to x, y, z coordinates of the point1 that needs to be added
sz	Pointers to x, y, z coordinates of the point1 that needs to be added
tx	Pointers to x, y, z coordinates of the point2 that needs to be added
ty	Pointers to x, y, z coordinates of the point2 that needs to be added.
tz	Pointers to x, y, z coordinates of the point2 that needs to be added
rx	Pointers to x, y, z coordinates of the result point
ry	Pointers to x, y, z coordinates of the result point
rz	Pointers to x, y, z coordinates of the result point

Return values

return zero on success non zero on failure.

Example

```
status = rsi_ecdh_point_addition(ECDH_224, d, sx, sy, sz, rx, ry, rz);
```

33.2.4 Point Subtraction

33.2.4.1 rsi_ecdh_point_substraction

Prototype

```
int32_t rsi_ecdh_point_substraction(uint8_t ecdh_mode,uint8_t *sx, uint8_t *sy,
uint8_t*sz,uint8_t *tx, uint8_t *ty, uint8_t *tz,uint8_t *rx, uint8_t *ry, uint8_t *rz)
```

Description

This API is used to ecdh Point Subtraction.

Parameters

Parameter	Description
ecdh_mode	• 1 – For ECDH 192 • 2 – For ECDH 224 • 3 – For ECDH 256
sx	Pointers to x, y, z coordinates of the point1 that needs to be subtraction.
sy	Pointers to x, y, z coordinates of the point1 that needs to be subtraction.
sz	Pointers to x, y, z coordinates of the point1 that needs to be subtraction.

Parameter	Description
tx	Pointers to x, y, z coordinates of the point2 that needs to be subtraction.
ty	Pointers to x, y, z coordinates of the point2 that needs to be subtraction.
tz	Pointers to x, y, z coordinates of the point2 that needs to be subtraction.
rx	Pointers to x, y, z coordinates of the result point.
ry	Pointers to x, y, z coordinates of the result point.
rz	Pointers to x, y, z coordinates of the result point.

Return values

return zero on success non zero on failure.

Example

```
status = rsi_ecdh_point_substraction(ECDH_224, d, sx, sy, sz, rx, ry, rz);
```

33.2.5 Point Doubling (PD)

33.2.5.1 rsi_point_double

Prototype

```
int32_t rsi_point_double(uint8_t ecdh_mode,uint8_t *sx, uint8_t *sy, uint8_t *sz,uint8_t
*rx, uint8_t *ry, uint8_t *rz)
```

Description

This API is used to ecdh point Doubling

Parameters

Parameter	Description
ecdh_mode	• 1 – For ECDH 192 • 2 – For ECDH 224 • 3 – For ECDH 256
sx	Pointers to x, y, z coordinates of the point1 that needs to be doubled .
sy	Pointers to x, y, z coordinates of the point1 that needs to be doubled
sz	Pointers to x, y, z coordinates of the point1 that needs to be doubled
rx	Pointers to x, y, z coordinates of the result point
ry	Pointers to x, y, z coordinates of the result point
rz	Pointers to x, y, z coordinates of the result point

Return values

return zero on success non zero on failure.

Example

```
status = rsi_ecdh_point_double(ECDH_224, d, sx, sy, sz, rx, ry, rz);
```

33.2.6 Point Affine

33.2.6.1 rsi_ecdh_point_affine

Prototype

```
int32_t rsi_ecdh_point_affine(uint8_t ecdh_mode,uint8_t *sx, uint8_t *sy, uint8_t
*sz,uint8_t *rx, uint8_t *ry, uint8_t *rz)
```

Description

This API is used to ecdh point affine

Parameters

Parameter	Description
ecdh_mode	• 1 – For ECDH 192 • 2 – For ECDH 224 • 3 – For ECDH 256
sx	Pointers to x, y, z coordinates of the point1 that needs to be affined .
sy	Pointers to x, y, z coordinates of the point1 that needs to be affined
sz	Pointers to x, y, z coordinates of the point1 that needs to be affined
rx	Pointers to x, y, z coordinates of the result point
ry	Pointers to x, y, z coordinates of the result point
rz	Pointers to x, y, z coordinates of the result point

Return values

return zero on success non zero on failure.

Example

```
status = rsi_ecdh_point_affine(ECDH_224, d, sx, sy, sz, rx, ry, rz);
```

33.3 Exponentiation

33.3.1 Overview

This section explains how to configure and use the Exponentiation using Redpine MCU SAPIs.

33.3.2 Programming Sequence

```
int main()
{
    int32_t status;
```

```
    status = rsi_crypto_exp_app();

    return status;

}

int32_t rsi_crypto_exp_app()
{
    int32_t status = RSI_SUCCESS;

    //Buffers to store responses
  uint8_t exp_result[1000];

    //! WC initialization
    status = rsi_wireless_init(0, 0);
    if(status != RSI_SUCCESS)
    {
        return status;
    }

    //! Memset before filling
    memset(exp_result, 0, 1000); //memset response buffers

    //!Encrypt using AES CBC mode 128bit Key
    status = rsi_exponentiation(prime, sizeof(prime), base, sizeof(base), exp,
sizeof(exp), exp_result );
    if(status != RSI_SUCCESS)
    {
#ifdef RSI_ENABLE_DEBUG_PRINT
        printf("\n Exponentiation Fail Status %d \n", status);
#endif
        return status;
    }
#ifdef RSI_ENABLE_DEBUG_PRINT
    printf("\n Exponentiation Done \n");
#endif

    return 0;
}
```

33.3.2.1 rsi_exponentiation

Prototype

```
int32_t rsi_exponentiation(uint8_t *prime, uint32_t prime_length,uint8_t *base, uint32_t
base_length,uint8_t *exponent,uint32_ exponent_length,uint8_t *exp_result);
```

Description

This API is used to exponentiation.

Parameters

Parameter	Description
prime	Pointer to prime value
prime_length	Length of the prime in bytes
base	Pointer to base value
base_length	Length of the base in bytes

Parameter	Description
exponent	Pointer to exponent value
exponent_length	Length of the exponent in bytes
exp_result	Computed exponentiation result

Return values

return zero on success non zero on failure.

Example

```
int32_t status;

status=rsi_exponentiation(prime, sizeof(prime), base, sizeof(base), exp, sizeof(exp),
exp_result );
```

33.4 HMAC-SHA

33.4.1 Overview

This section explains how to configure and use the HMAC-SHA using Redpine MCU SAPIs.

33.4.2 Programming Sequence

```
int main()
{
    int32_t status;

    status = rsi_crypto_hmac_sha_app();

    return status;

}

int32_t rsi_crypto_hmac_sha_app()
{
    int32_t status = RSI_SUCCESS;

    //Buffers to store responses
    uint8_t digest[64];

    //! WC initialization
    status = rsi_wireless_init(0, 0);
    if(status != RSI_SUCCESS)
    {
        return status;
    }

    //! Memset before filling
    memset(digest, 0, 64);

    status = rsi_hmac_sha(SHA_512, msg, strlen(msg), key, strlen(key), digest);
    if(status != RSI_SUCCESS)
    {
#ifdef RSI_ENABLE_DEBUG_PRINT
        printf("\n HMAC SHA 256 Failed %d \n", status);
#endif
        return status;
    }
```

```
#ifdef RSI_ENABLE_DEBUG_PRINT
    printf("\n HMAC SHA 256 success\n");
#endif
    return 0;
}
```

33.4.2.1 rsi_hmac_sha

Prototype

```
int32_t rsi_hmac_sha(uint8_t hmac_sha_mode, uint8_t *msg, uint32_t msg_length, uint8_t
*key, uint32_t key_length, uint8_t *digest);
```

Description

This API is used to hmac sha.

Parameters

Parameter	Description
hmac_sha_mode	Mode of the HMAC-SHA 1 – For HMAC-SHA1 2 – For HMAC-SHA256 3 – For HMAC-SHA384 4 – For HMAC-SHA512
msg	Pointer to message
msg_length	Total message length
key	Pointer to HMAC key
key_length	HMAC key length in bytes
digest	Output parameter to hold computed digest from HMAC-SHA

Return values

return zero on success non zero on failure.

Example

```
int32_t status;

status = rsi_hmac_sha(SHA_512, msg, strlen(msg), key, strlen(key), digest);
```

33.5 SHA

33.5.1 Overview

This section explains how to configure and use the SHA using Redpine MCU SAPIs.

33.5.2 Programming Sequence

```
int main()
{
    int32_t status;
```

```
 status = rsi_crypto_sha_app();

return status;

}

int32_t rsi_crypto_sha_app()
{
    int32_t status = RSI_SUCCESS;

    //Buffers to store responses
    uint8_t digest[50];

    //! WC initialization
    status = rsi_wireless_init(0, 0);
    if(status != RSI_SUCCESS)
    {
        return status;
    }

    //! Memset before filling
    memset(digest, 0, 50);

    status = rsi_sha(SHA_256,SHA,sizeof(SHA),digest);
    if(status != RSI_SUCCESS)
    {
#ifdef RSI_ENABLE_DEBUG_PRINT
        printf("\n SHA Failed %d \n", status);
#endif
        return status;
    }
#ifdef RSI_ENABLE_DEBUG_PRINT
    printf("\n SHA success\n");
    printf("\n\n\nDone\n\n\n");
#endif
    return 0;
}
```

33.5.3 rsi_sha

Prototype

```
int32_t rsi_sha(uint8_t sha_mode, uint8_t *msg, uint32_t msg_length, uint8_t *digest);
```

Description

This API is used to SHA.

Parameters

Parameter	Description
sha_mode	Mode of the SHA 1 – For SHA1 2 – For SHA256 3 – For SHA384 4 – For SHA512
msg	Pointer to message
msg_length	Total message length
digest	Output parameter to hold computed digest from SHA

Return values

return zero on success non zero on failure.

Example

```
status = rsi_sha(SHA_256,SHA,sizeof(SHA),digest);
```

34 Analog to Digital Converter

34.1 Overview

This section explains how to configure and use the ADC using Redpine MCU SAPIs.

34.2 Programming Sequence

```
/* Includes ------------------------------------------------------------------*/
#include "rsi_chip.h"    /* Redpine_MCU_Vx.y.z\Host_MCU\common\chip\inc */
#include "rsi_board.h"   /* Redpine_MCU_Vx.y.z\Host_MCU\common\board\inc */
#include "rsi_bod.h"     /* Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\inc
*/
/* Private typedef -----------------------------------------------------------*/

/* Private macro -------------------------------------------------------------*/
#define  ADC_IRQ_Handler   IRQ011_Handler
/* Private define ------------------------------------------------------------*/
#define  PING_ADDRESS              0x24060000
#define  PONG_ADDRESS              0x240603FF

#define  PING_LENGTH               1023
#define  PONG_LENGHT               1023

#define  PING_ENABLE               1
#define  PONG_ENABLE               1
#define  CHANNEL_NO                1
#define  FIFO_THR                  3

#define  ANALOG_MODE               7
#define  POSITIVE_INP_SEL          2
#define  NEGATIVE_INP_SEL          2
#define  INPUT_MODE                1           /* If INPUT_MODE is 0,single ended mode
will be configured
                                                  If INPUT_MODE is 1,differential mode
will be configured */
#define  CLOCK_DIV_FAC             16
#define  SAMPLING_RATE_FAC         10

#define  INPUT_POS_PIN             4           /* Here ULP_GPIO_4 is using as pos input to
ADC,Can program ULP_GPIO pins from 0 to 15 */
#define  INPUT_NEG_PIN             5           /* Here ULP_GPIO_5 is using as neg input to
ADC,Can program ULP_GPIO pins from 0 to 15 */

#define  BUFFER_SIZE               1023

#define  GAIN_OFFSET_CAL_ENABLE    1           /* If this macro is 1 then gain and offset
calculation will be done on ADC samples */

float  max_ip_volt_scdc = 2.4;

/* Private variables ---------------------------------------------------------*/

int16_t ping_buf[BUFFER_SIZE];
int16_t pong_buf[BUFFER_SIZE];

volatile uint16_t a,i;
RSI_ADC_CALLBACK_T vsADCCallBack;
volatile uint16_t dout;
volatile float vout,vref_v;
float battery_status;
/* Private functions ---------------------------------------------------------*/
/**
 * @brief  This function define pin mux.
 * @param  mode    :  Type of mode (Differential or single mode)
 * @retval None
 */
```

```
void adc_input_pin_mux(uint8_t mode)
{
  if(mode)
  {
    /* Differential mode pin config */
    /* both pos and neg input pins are ocnfigured for Differential mode */
    RSI_EGPIO_SetPinMux(EGPIO1,0,INPUT_POS_PIN,ANALOG_MODE);
    RSI_EGPIO_SetPinMux(EGPIO1,0,INPUT_NEG_PIN,ANALOG_MODE);
  }
  else
  {
    /* Either pos or neg input can be selected for Single ended mode */
    RSI_EGPIO_SetPinMux(EGPIO1,0,INPUT_POS_PIN,ANALOG_MODE);
  }
}

/**
 * @brief  This function is ADC IRQ to clear the ADC interrupt.
 * @param  None
 * @retval None
 */
void ADC_IRQ_Handler(void)
{
    RSI_ADC_InterruptHandler(AUX_ADC_DAC_COMP,&vsADCCallBack);
}

/**
 * @brief  This function is callback function to reconfigure ping and pong address.
 * @param  channel_no  :  Channel number
 * @param  event       :  Event type
 * @retval None
 */
void call_register( uint16_t channel_no,uint8_t event)
{
    if(event & MULTI_CHANNEL_EVENT)
    {
        if(a)
        {
          /* Reconfigure the ping and pong address */
          RSI_ADC_PingPongReInit(AUX_ADC_DAC_COMP,channel_no,0,PONG_ENABLE);

          /* Read the pong address ADC data */

RSI_ADC_ReadData(AUX_ADC_DAC_COMP,pong_buf,0,channel_no,GAIN_OFFSET_CAL_ENABLE,INPUT_MODE)
;

          /* print the equivalent ADC input voltage from obtain ADC samples */
          PrintADCInputVoltage(pong_buf,INPUT_MODE);

          a=0;
        }
        else
        {
          /* Reconfigure the ping and pong address */
          RSI_ADC_PingPongReInit(AUX_ADC_DAC_COMP,channel_no,PING_ENABLE,0);

          /* Read the ping address ADC data */

RSI_ADC_ReadData(AUX_ADC_DAC_COMP,ping_buf,1,channel_no,GAIN_OFFSET_CAL_ENABLE,INPUT_MODE)
;

         a=1;
        }
    }
}

/**
 * @brief  Main program.
 * @param  None
 * @retval None
 */
```

```
int main()
{
    /* At this stage the MICROCONTROLLER clock setting is already configured,
     * this is done through SystemInit() function which is called from startup
     * file (startup_rs1xxxx.s) before to branch to application main.
     * To reconfigure the default setting of SystemInit() function, refer to
     * system_rs1xxxx.c file
     */
    SystemCoreClockUpdate();

    vsADCCallBack.adccallbacFunc = call_register;

    /* Read the input suuply for chip */
    battery_status = RSI_BOD_SoftTriggerGetBatteryStatus();

    if(battery_status < max_ip_volt_scdc)
    {
      RSI_IPMU_ProgramConfigData(hp_ldo_voltsel);
    }

    RSI_Board_Init();

    /* Powerup of ADC block */
    RSI_ADC_PowerControl(ADC_POWER_ON);

    /* Clock configuration of ADC */
    RSI_ULPSS_AuxClkConfig(ULPCLK, ENABLE_STATIC_CLK ,ULP_AUX_32MHZ_RC_CLK);

    /* Bypass ldo disable */
    RSI_ADC_AUXLdoConfig(AUX_ADC_DAC_COMP,0,0xb);

    /* Clock division factor , here divide clock to 1mhz*/
    RSI_ADC_ClkDivfactor(AUX_ADC_DAC_COMP,0,CLOCK_DIV_FAC);

    /* Calibrate the ADC */
    RSI_ADC_Calibration();

    /* configures ULP GPIO 4 in analog mode */
    adc_input_pin_mux(INPUT_MODE);

    /* Configure the ADC in multichannel mode with internal DMA */
    RSI_ADC_Config(AUX_ADC_DAC_COMP,1,0,FIFO_THR,1);

    /* Configure the required channel of ADC for conversion */
    RSI_ADC_ChannelConfig( AUX_ADC_DAC_COMP, CHANNEL_NO, POSITIVE_INP_SEL,
NEGATIVE_INP_SEL, INPUT_MODE );

     /* Configure the ping and pong memory address */
    RSI_ADC_PingPongMemoryAdrConfig(
AUX_ADC_DAC_COMP,CHANNEL_NO,PING_ADDRESS,PONG_ADDRESS,PING_LENGTH,
                                    PONG_LENGHT, PING_ENABLE, PONG_ENABLE);

    /*Configure the sampling rate of ADC */
    RSI_ADC_ChannelSamplingRate(AUX_ADC_DAC_COMP,CHANNEL_NO,0,SAMPLING_RATE_FAC);

    /* Enable the ADC interrupt */
    RSI_ADC_ChnlIntrUnMask(AUX_ADC_DAC_COMP,CHANNEL_NO);

    NVIC_EnableIRQ(ADC_IRQn);

    /* Ping pong enable */
    RSI_ADC_PingpongEnable(AUX_ADC_DAC_COMP,CHANNEL_NO);

    /* DMA enable */
    RSI_ADC_InternalPerChnlDmaEnable(AUX_ADC_DAC_COMP,CHANNEL_NO);

    /* ADC channel enable */
    RSI_ADC_ChnlEnable(AUX_ADC_DAC_COMP,CHANNEL_NO);

    /* Starts ADC conversion */
    RSI_ADC_Start(AUX_ADC_DAC_COMP);
```

```
    while(1);
}
```

Code Block 25 Analog to Digital converter example

34.3 API Descriptions

34.3.1 RSI_ADC_PingPongMemoryAdrConfig

Source File : rsi_adc.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_ADC_PingPongMemoryAdrConfig(AUX_ADC_DAC_COMP_Type *pstcADC,
                                        uint32_t channel ,
                                        uint32_t ping_addr,
                                        uint32_t pong_addr,
                                        uint16_t ping_length,
                                        uint16_t pong_length,
                                        uint8_t ping_enable,
                                        uint8_t pong_enable )
```

Description

This API is used to configure Ping-pong memory location along with the length of the samples.

Parameters

Parameter	Description
pstcADC	Pointer to the AUX_ADC_DAC_COMP_Type
channel	ADC channel to be configured as 0,1,2 ...15
ping_addr	ping address of in ULPSS memory
pong_addr	ping address of in ULPSS memory (e.g pong address = ping address + ping length)
ping_length	ping length
pong_length	pong length
ping_enable	This parameter set when the ping address required to configure .
pong_enable	This parameter set when the pong address required to configure .

Return values

Return zero on success.

Example

```
#define ADC_PING_MEM      0x24060000
#define ADC_PONG_MEM      0x240603FF
#define ADC_PING_LENG     1023
#define ADC_PONG_LENG     1023
```

```
RSI_ADC_PingPongMemoryAdrConfig(AUX_ADC_DAC_COMP,ADC_CHNNNO,ADC_PING_MEM,
                                    ADC_PING_MEM,ADC_PING_LENG,ADC_PONG_LENG,1,1);
```

34.3.2 RSI_ADC_PingPongReInit

Source File : rsi_adc.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_ADC_PingPongReInit(AUX_ADC_DAC_COMP_Type *pstcADC, uint8_t channel,uint8_t
ping_enable,uint8_t pong_enable )
```

Description

This API is used to re configure Ping-pong memory location along with the length

Parameters

Parameter	Description
pstcADC	Pointer to the AUX_ADC_DAC_COMP_Type
channel	ADC channel to be configured as 0,1,2 ...15
ping_enable	Enable ping address reconfiguration
pong_enable	Enable pong address reconfiguration

Return values

Return zero on success.

Example

```
#define  ADC_CHNNO     0
#define  PING_ENABLE   1
#define  PONG_ENABLE   0
RSI_ADC_PingPongReInit(AUX_ADC_DAC_COMP,ADC_CHNNNO,PING_ENABLE,PONG_ENABLE);
```

34.3.3 RSI_ADC_PingpongEnable

Source File : rsi_adc.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_ADC_PingpongEnable(AUX_ADC_DAC_COMP_Type *pstcADC, uint32_t channel)
```

Description

This API is used to Enable ping pong for corresponding ADC channels.

Parameters

Parameter	Description
pstcADC	Pointer to the

Parameter	Description
	AUX_ADC_DAC_COMP_Type
channel	ADC channel to be configured as 0,1,2 ...15

Return values

Return zero on success.

Example

```
#define   ADC_CHNNO 0

RSI_ADC_PingpongEnable(AUX_ADC_DAC_COMP,ADC_CHNNNO);
```

34.3.4 RSI_ADC_PingpongDisable

Source File : rsi_adc.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_ADC_PingpongDisable(AUX_ADC_DAC_COMP_Type *pstcADC, uint32_t channel)
```

Description

This API is used to disable the ping pong for corresponding ADC channels.

Parameters

Parameter	Description
pstcADC	Pointer to the AUX_ADC_DAC_COMP_Type
channel	ADC channel to be configured as 0,1,2 ...15

Return values

Return zero on success.

Example

```
#define   ADC_CHNNO 0

RSI_ADC_PingpongDisable(AUX_ADC_DAC_COMP,ADC_CHNNNO);
```

34.3.5 RSI_ADC_InternalPerChnlDmaEnable

Source File : rsi_adc.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_ADC_InternalPerChnlDmaEnable(AUX_ADC_DAC_TYPE *pstcADC, uint32_t channel)
```

Description

This API is used to internal dma channel enable.

Parameters

Parameter	Description
pstcADC	Pointer to the AUX_ADC_DAC_TYPE
channel	ADC channel to be configured as 0,1,2 ...15

Return values

Return zero on success.

Example

```
#define   ADC_CHNNO 0

RSI_ADC_InternalPerChnlDmaEnable(AUX_ADC_DAC_COMP,ADC_CHNNNO);
```

34.3.6 RSI_ADC_InternalPerChnlDmaDisable

Source File : rsi_adc.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_ADC_InternalPerChnlDmaDisable(AUX_ADC_DAC_COMP_Type *pstcADC, uint32_t
channel)
```

Description

This API is used to disable the internal dma channel for corresponding ADC channels.

Parameters

Parameter	Description
pstcADC	Pointer to the AUX_ADC_DAC_COMP_Type
channel	ADC channel to be configured as 0,1,2 ...15

Return values

Return zero on success.

Example

```
#define   ADC_CHNNO 0

RSI_ADC_InternalPerChnlDmaDisable(AUX_ADC_DAC_COMP,ADC_CHNNNO);
```

34.3.7 RSI_ADC_Config

Source File : rsi_adc.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_ADC_Config(AUX_ADC_DAC_COMP_Type *pstcADC,uint8_t multi_channel_en,
                       uint8_t static_fifo_mode, uint8_t fifo_threshold,
```

```
                        uint8_t internal_dma_en )
```

Description

This API is used to configure the ADC.

Parameters

Parameter	Description
pstcADC	Pointer to the AUX_ADC_DAC_COMP_Type
multi_channel_en	This parameter define the multichannel enable or disable
static_fifo_mode	Static mode enable parameter static_fifo_mode =1 static mode static_fifo_mode =0 fifo mode
fifo_threshold	fifo threshold value for ADC operation
internal_dma_en	Internal DMA enable parameter

Return values

Return zero on success.

Example

```
#define FIFO_THR       3

RSI_ADC_Config(AUX_ADC_DAC_COMP,1,0,FIFO_THR,1);
```

34.3.8 RSI_ADC_ChannelConfig

Source File : rsi_adc.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_ADC_ChannelConfig( AUX_ADC_DAC_COMP_Type *pstcADC, uint32_t channel ,
                               uint8_t an_perif_adc_ip_sel,uint8_t an_perif_adc_in_sel,
                               uint8_t an_perif_adc_diffmode )
```

Description

This API is used to configure the ADC channels.

Parameters

Parameter	Description
pstcADC	Pointer to the AUX_ADC_DAC_COMP_Type
channel	ADC channel to be configured as 0,1,2 ...15
an_perif_adc_ip_sel	ADC positive input select in multi channel mode
an_perif_adc_in_sel	ADC negative input select in multi channel mode

Parameter	Description
an_perif_adc_diffmode	ADC differential mode selection in multi channel mode

Return values

Return zero on success.

Example

```
#define   ADC_CHNNO        0
#define   POSITIVE_IN_SEL  2
#define   NEGATIVE_IN_SEL  2

RSI_ADC_ChannelConfig(AUX_ADC_DAC_COMP,ADC_CHNNNO,POSITIVE_IN_SEL,NEGATIVE_IN_SEL,1);
```

34.3.9 RSI_ADC_StaticMode

Source File : rsi_adc.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_ADC_StaticMode(AUX_ADC_DAC_COMP_Type *pstcADC,uint16_t an_perif_adc_ip_sel,
uint16_t an_perif_adc_in_sel, uint8_t an_perif_adc_diffmode)
```

Description

This API is used to configure the ADC in Static Mode.

Parameters

Parameter	Description
pstcADC	Pointer to the AUX_ADC_DAC_COMP_Type
an_perif_adc_ip_sel	ADC positive input select in static mode
an_perif_adc_in_sel	ADC negative input select in static mode
an_perif_adc_diffmode	ADC differential mode selection in static mode

Return values

Return zero on success.

Example

```
#define   POSITIVE_IN_SEL  2
#define   NEGATIVE_IN_SEL  2

RSI_ADC_StaticMode(AUX_ADC_DAC_COMP,POSITIVE_IN_SEL,NEGATIVE_IN_SEL,1);
```

34.3.10 RSI_ADC_ChannelSamplingRate

Source File : rsi_adc.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_ADC_ChannelSamplingRate(AUX_ADC_DAC_COMP_Type *pstcADC, uint32_t channel ,
```

```
uint16_t adc_ch_offset, uint16_t adc_ch_freq_val)
```

Description

This API is used to configure the sampling rate for ADC .

Parameters

Parameter	Description
pstcADC	Pointer to the AUX_ADC_DAC_COMP_Type
channel	ADC channel to be configured as 0,1,2 ...15
adc_ch_offset	channel offset for each channel
adc_ch_freq_val	channel frequency for each channel to set sampling rate .

Return values

Return zero on success.

Example

```
#define   ADC_CHNNNO           0
#define   ADC_SAMPLING_VALUE  11
#define   ADC_OFFSET_VALUE     0
RSI_ADC_ChannelSamplingRate(AUX_ADC_DAC_COMP,ADC_CHNNNO,ADC_OFFSET_VALUE,ADC_SAMPLING_VALU
E);
```

34.3.11 RSI_ADC_FifoMode

Source File : rsi_adc.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_ADC_FifoMode(AUX_ADC_DAC_COMP_Type *pstcADC,uint16_t channel_no,uint16_t
an_perif_adc_ip_sel, uint16_t an_perif_adc_in_sel, uint8_t an_perif_adc_diffmode)
```

Description

This API is used to configure the ADC in FIFO Mode.

Parameters

Parameter	Description
pstcADC	Pointer to the AUX_ADC_DAC_COMP_Type
channel_no	Pointer to the ADC channel configuration structure
an_perif_adc_ip_sel	ADC positive input select in Fifo mode.
an_perif_adc_in_sel	ADC negative input select in Fifo mode.
an_perif_adc_diffmode	ADC differential mode selection in Fifo mode.

Return values

Return zero on success.

Example

```
#define   ADC_CHNNO        0
#define   POSITIVE_IN_SEL  2  /* ULP_GPIO 4 */
#define   NEGATIVE_IN_SEL  2  /* ULP_GPIO 5 */

RSI_ADC_FifoMode(AUX_ADC_DAC_COMP,ADC_CHNNO,POSITIVE_IN_SEL,NEGATIVE_IN_SEL,1);
```

34.3.12 RSI_ADC_Start

Source File : rsi_adc.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_ADC_Start( AUX_ADC_DAC_COMP_Type *pstcADC )
```

Description

This API is used to start the ADC.

Parameters

Parameter	Description
pstcADC	Pointer to the AUX_ADC_DAC_COMP_Type

Return values

Return zero on success.

Example

```
RSI_ADC_Start(AUX_ADC_DAC_COMP);
```

34.3.13 RSI_ADC_Stop

Source File : rsi_adc.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_ADC_Stop( AUX_ADC_DAC_COMP_Type *pstcADC )
```

Description

This API is used to stop the ADC.

Parameters

Parameter	Description
pstcADC	Pointer to the AUX_ADC_DAC_COMP_Type

Return values

Return zero on success .

Example

```
RSI_ADC_Stop(AUX_ADC_DAC_COMP);
```

34.3.14 RSI_ADC_ChnlEnable

Source File : rsi_adc.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_ADC_ChnlEnable(AUX_ADC_DAC_COMP_Type *pstcADC, uint32_t channel)
```

Description

This API is used to Enable the ADC.

Parameters

Parameter	Description
pstcADC	Pointer to the AUX_ADC_DAC_COMP_Type
channel	ADC channel to be configured as 0,1,2 ...15

Return values

Return zero on success, error code on failure.

Example

```
#define   ADC_CHNNO      0

RSI_ADC_ChnlEnable(AUX_ADC_DAC_COMP,ADC_CHNNNO);
```

34.3.15 RSI_ADC_ChnlDisable

Source File : rsi_adc.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_ADC_ChnlDisable(AUX_ADC_DAC_COMP_Type *pstcADC, uint32_t channel
```

Description

This API is used to Disable the ADC.

Parameters

Parameter	Description
pstcADC	Pointer to the AUX_ADC_DAC_COMP_Type
channel	ADC channel to be configured as 0,1,2 ...15

Return values

Return zero on success.

Example

```
#define   ADC_CHNNO      0

RSI_ADC_ChnlDisable(AUX_ADC_DAC_COMP,ADC_CHNNNO);
```

34.3.16 RSI_ADC_ReadDataStatic

Source File : rsi_adc.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
int16_t RSI_ADC_ReadDataStatic(AUX_ADC_DAC_COMP_Type *pstcADC,uint8_t
data_process_en,uint8_t diff_en)
```

Description

This API is used to Read the ADC samples when static mode is enabled.

Parameters

Parameter	Description
pstcADC	Pointer to the AUX_ADC_DAC_COMP_Type
data_process_en	This parameter define for the output ADC samples gain and calculation enable or disable. data_process_en - 1 (Gain and offset value calculation will be done output ADC samples) data_process_en - 0 (disable the gain and calculation on output samples)
diff_en	diff_en : ADC mode of operation single ended or differential ended diff_en - 1 differential ended mode enable. diff_en - 0 single ended mode enable.

Return values

Return ADC output.

Example

```
RSI_ADC_ReadDataStatic(AUX_ADC_DAC_COMP,ADC_CHNNNO,0);
```

34.3.17 RSI_ADC_ReadData

Source File : rsi_adc.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_ADC_ReadData(AUX_ADC_DAC_COMP_Type *pstcADC, int16_t *data,
                         uint8_t ping_pong,uint16_t channel)
```

Description

This API is used to Read the ADC samples when ulp memories are used.

Parameters

Parameter	Description
pstcADC	Pointer to the AUX_ADC_DAC_COMP_Type
data	Pointer to read buffer
ping_pong	• 1 - ping memory must be read • 0 - pong memory must be read
data_process_en	This parameter define for the output ADC samples gain and calculation enable or disable. data_process_en - 1 (Gain and offset value calculation will be done output ADC samples) data_process_en - 0 (disable the gain and calculation on output samples)
diff_en	diff_en : ADC mode of operation single ended or differential ended diff_en - 1 differential ended mode enable. diff_en - 0 single ended mode enable.

Return values

Return zero on success.

Example

```
int32_t data[1023];
RSI_ADC_ReadData(AUX_ADC_DAC_COMP,data,1,0,0);
```

34.3.18 RSI_ADC_ClkDivfactor

Source File : rsi_adc.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_ADC_ClkDivfactor(AUX_ADC_DAC_COMP_Type *pstcADC, uint16_t adc_on_time ,
                             uint16_t adc_total_duration)
```

Description

This API is used to set clock with configurable on time.

Parameters

Parameter	Description

Parameter	Description
pstcADC	Pointer to the AUX_ADC_DAC_COMP_Type
adc_on_time	ON duration of the clock
adc_total_duration	Total ON and OFF duration of the clock

Return values

Return zero on success.

Example

```
RSI_ADC_ClkDivfactor(AUX_ADC_DAC_COMP,0,4);
On time parameter use to control the duty cycle of ADC clock.In the case of OPAMP
operation on time of ADC clock is minimum 10us So in this case use the "ON Time" and
"Total time" parameter.
on time = number of adc clocks required for on duration
total time = on time + off time (off time =  number of adc clk required as off time)
clock obtained after on time and total time configuration is duty cycled clock.
sampling value = no of clocks of duty cycled clock required to achieve required sampling
rate.
```

34.3.19 RSI_ADC_ChnlIntrUnMask

Source File : rsi_adc.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_ADC_ChnlIntrUnMask(AUX_ADC_DAC_COMP_Type *pstcADC, uint32_t channel)
```

Description

This API is used to Unmask interrupt for ADC channel.

Parameters

Parameter	Description
pstcADC	Pointer to the AUX_ADC_DAC_COMP_Type
channel	ADC channel to be configured as 0,1,2 ...15

Return values

Return zero on success.

Example

```
#define  ADC_CHANNEL_NUMBER     0
RSI_ADC_ChnlIntrUnMask(AUX_ADC_DAC_COMP,ADC_CHANNEL_NUMBER);
```

34.3.20 RSI_ADC_ChnlIntrMask

Source File : rsi_adc.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_ADC_ChnlIntrMask(AUX_ADC_DAC_COMP_Type *pstcADC, uint32_t channel)
```

Description

This API is used to Mask interrupt for ADC channel.

Parameters

Parameter	Description
pstcADC	Pointer to the AUX_ADC_DAC_COMP_Type
channel	ADC channel to be configured as 0,1,2 ...15

Return values

Return zero on success.

Example

```
#define  ADC_CHANNEL_NUMBER     0
RSI_ADC_ChnlIntrMask(AUX_ADC_DAC_COMP,ADC_CHANNEL_NUMBER);
```

34.3.21 RSI_ADC_ChnlIntrClr

Source File : rsi_adc.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_ADC_ChnlIntrClr(AUX_ADC_DAC_COMP_Type *pstcADC, uint32_t channel)
```

Description

This API is used to clear interrupt of ADC channel.

Parameters

Parameter	Description
pstcADC	Pointer to the AUX_ADC_DAC_COMP_Type
channel	ADC channel to be configured as 0,1,2 ...15

Return values

Return zero on success.

Example

```
#define  ADC_CHANNEL_NUMBER     0
RSI_ADC_ChnlIntrClr(AUX_ADC_DAC_COMP,ADC_CHANNEL_NUMBER);
```

34.3.22 RSI_ADC_ChnlIntrStatus

Source File : rsi_adc.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
uint32_t RSI_ADC_ChnlIntrStatus(AUX_ADC_DAC_COMP_Type *pstcADC)
```

Description

This API is used to status the ADC channel.

Parameters

Parameter	Description
pstcADC	Pointer to the AUX_ADC_DAC_COMP_Type
channel	ADC channel to be configured as 0,1,2 ...15

Return values

Return status of interrupt.

Example

```
uint8_t status;

status = RSI_ADC_ChnlIntrStatus(AUX_ADC_DAC_COMP);
```

34.3.23 RSI_ADC_PowerControl

Source File : rsi_adc.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
void RSI_ADC_PowerControl(POWER_STATE state)
```

Description

This API is used to Power On and off for ADC.

Parameters

Parameter	Description
state	• ADC_POWER_ON - To powerup adc powergates • ADC_POWER_OFF - To powerdown adc powergates

Return values

None

Example

```
RSI_ADC_PowerControl(ADC_POWER_ON);
```

34.3.24 RSI_ADC_NoiseAvgMode

Source File : rsi_adc.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_ADC_NoiseAvgMode(AUX_ADC_DAC_COMP_Type *pstcADC, bool en_disable)
```

Description

This API is used to Enable or Disable Noise averaging mode.

Parameters

Parameter	Description
pstcADC	Pointer to the AUX_ADC_DAC_COMP_Type
en_disable	• 1 - To enable noise averaging mode • 0 - To disable noise averaging mode

Return values

None

Example

```
#define ENABLE    1
RSI_ADC_NoiseAvgMode(AUX_ADC_DAC_COMP,ENABLE)
```

34.3.25 RSI_ADC_ThresholdConfig

Source File : rsi_adc.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_ADC_ThresholdConfig(AUX_ADC_DAC_COMP_Type *pstcADC , uint32_t
threshold1,uint32_t threshold2, uint8_t range)
```

Description

This API is used to compare threshold value with adc data.

Parameters

Parameter	Description
pstcADC	Pointer to the AUX_ADC_DAC_COMP_Type
threshold1	Threshold value to be programmed
threshold2	Threshold value to be programmed when range is 1
range	When adc data compared lies in certain range of threshold values set this bit

Return values

Return zero on success.

Example

```
RSI_ADC_ThresholdConfig(AUX_ADC_DAC_COMP,0x990,0x997,1);
```

34.3.26 RSI_ADC_InterruptHandler

Source File : rsi_adc.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_ADC_InterruptHandler( AUX_ADC_DAC_COMP_Type *pstcADC,RSI_ADC_CALLBACK_T
*pADCCallBack)
```

Description

This API is used for handle the interrupt and clear adc interrupt.

Parameters

Parameter	Description
pstcADC	Pointer to the AUX_ADC_DAC_COMP_Type
pADCCallBack	Callback pointer of following structure RSI_ADC_CALLBACK_T

Return values

None

Example

```
RSI_ADC_CALLBACK_T vsADCCallBack;

RSI_ADC_InterruptHandler(AUX_ADC_DAC_COMP,&vsADCCallBack);
```

34.3.27 RSI_ADC_ThreshInterruptClr

Source File : rsi_adc.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
void RSI_ADC_ThreshInterruptClr(AUX_ADC_DAC_COMP_Type *pstcADC)
```

Description

This API is used clear the threshold interrupt.

Parameters

Parameter	Description
pstcADC	Pointer to the AUX_ADC_DAC_COMP_Type

Return values

Return zero on success.

Example

```
RSI_ADC_ThreshInterruptClr(AUX_ADC_DAC_COMP);
```

34.3.28 RSI_ADC_Calibration

Source File : rsi_adc.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
void  RSI_ADC_Calibration(void)
```

Description

This API is used for calibration.

Parameters

None

Return values

None

Example

```
RSI_ADC_Calibration();
```

34.3.29 RSI_ADC_AUXBypassLdoEnable

Source File : rsi_adc.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
void RSI_ADC_AUXLdoConfig(AUX_ADC_DAC_COMP_Type *pstcADC,uint16_t bypass_en,uint16_t
value)
```

Description

This API is use for configure the AUX LDO.

Parameters

Parameter	Description
pstcADC	Pointer to the AUX_ADC_DAC_COMP_Type
bypass_en	LDO bypass enable or disable • 1 - Bypass the LDO mode • 0 - In LDO mode.
value	Word to set the output voltage

Parameter	Description
	• 0 - 1.6 v • 1 - 1.68 v • 2 - 1.76v • 3 -1.84v • 4 -1.92v • 5 -2v • 6 -2.08v • 7 -2.16v • 8 -2.24v • 9 -2.32v • 10 -2.4v • 11 -2.48v • 12 -2.64v • 13 - 2.64v • 14 - 2.72v • 15 - 2.8v

Return values

None

Example

```
RSI_ADC_AUXLdoConfig(AUX_ADC_DAC_COMP,1,0xb);
```

34.3.30 RSI_ADC_GainOffsetCal

Source File : rsi_adc.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
int16_t  RSI_ADC_GainOffsetCal(int16_t data,uint8_t diff_en)
```

Description

This API is used to calculate gain and offset value.

Parameters

Parameter	Description
data	Output data sample.
diff_en	This parameter define differential mode is enable or disable.

Return values

Updated sample value.

Example

```
RSI_ADC_GainOffsetCa(data,1);
```

34.3.31 RSI_ADC_VrefCal

Source File : rsi_adc.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
float RSI_ADC_VrefCal(void)
```

Description

This API is used to calculate vref value for application.

Parameters

None

Return values

Return the verf LDO voltage value.

Example

```
RSI_ADC_VrefCal();
```

35 Digital to Analog Converter

35.1 Overview

This section explains how to configure and use the DAC using Redpine MCU SAPIs.

35.2 Programming Sequence

```
#include "rsi_chip.h"   /* Redpine_MCU_Vx.y.z\Host_MCU\common\chip\inc */
#include "rsi_board.h"  /* Redpine_MCU_Vx.y.z\Host_MCU\common\board\inc */

/* Private macro ------------------------------------------------------------*/
#define  UDMA1_IRQHandler  IRQ010_Handler
#define  OUTPUT_PIN_NUMBER  4
#define  OUTPUT_MUX_ENABLE  1
#define  OUTPUT_MUX_SEL     0   /* Select DAC output on ULP_GPIO4 */
#define  DAC_FIFO_THR       4   /* Set FIFO threshold as 4 */
#define  DAC_FIFO_MODE      0
#define  BUFFER_SIZE        1024

uint16_t dac_100k_n0dbfs[1024] ={/* Standard the sine wave samples */}
/**
 * @brief      This API configure the pin for DAC output.
 * @param[in]  Pin number to use play DAC output on GPIO , (ULP_GPIO4 or ULP_GPIO15)
 * @return     None
 */
void dac_output_pin_mux(uint16_t pin_number)
{
    RSI_EGPIO_SetPinMux(EGPIO1,0,pin_number,EGPIO_PIN_MUX_MODE7);
}

/**
 * @brief      UDMA  controller transfer descriptor chain complete callback
 * @param[in]  event dma transfer events
 * @param[in]  ch    dma channel number
 * @return     None
 */
void udmaTransferComplete(uint32_t event, uint8_t ch)
{
  if(event == UDMA_EVENT_XFER_DONE)
  {
    if(ch == 10)
    {
      done = 1;
    }
  }
}

/**
 * @brief      UDMA  configuration and enable function.
 * @param[in]  event : dma transfer events
 * @param[in]  ch    : dma channel number
 * @return     None
 */
void UDMA_Write(void)
{
    memset(&control, 0, sizeof(RSI_UDMA_CHA_CONFIG_DATA_T));
    memset(&config, 0, sizeof(RSI_UDMA_CHA_CFG_T));

    config.altStruct = 0;
    config.burstReq  = 1;
    config.channelPrioHigh = 0;
    config.periAck = 0;
    config.periphReq = 0;
    config.reqMask = 0;
    config.dmaCh     = 10;
```

```
    /* Setup source to desination copy for trigger for memory */
    /* Descriptor configurations */
    control.transferType       = UDMA_MODE_BASIC;
    control.nextBurst                     = 0;
    control.totalNumOfDMATrans = BUFFER_SIZE -1;
    control.rPower             = ARBSIZE_4;
    control.srcProtCtrl              = 0x000;
    control.dstProtCtrl              = 0x000;
    control.srcSize            = SRC_SIZE_16;
    control.srcInc             = SRC_INC_16;
    control.dstSize            = DST_SIZE_16;
    control.dstInc             = DST_INC_NONE;

   /* Initialise dma */
   UDMAdrv1->Initialize();

     /* Configure dma channel */
   UDMAdrv1->ChannelConfigure( 10,(uint32_t )tx_buffer, (uint32_t )0x24043810,
BUFFER_SIZE,
                               control, &config, udmaTransferComplete );
   /* Enable dma channel */
   UDMAdrv1->ChannelEnable(10);
}

/**
 * @brief  Main program.
 * @param  None
 * @retval None
 */
int main(void)
{
    SystemCoreClockUpdate();

    /* Power gate enable for DAC */
    RSI_DAC_PowerControl(DAC_POWER_ON);

    /* Enble the 32mrc clock to DAC */
    RSI_ULPSS_AuxClkConfig(ULPCLK, ENABLE_STATIC_CLK ,ULP_AUX_32MHZ_RC_CLK);

    /* Configure the refernce LDO voltage as 2.4v */
    RSI_ADC_AUXBypassLdoEnable(AUX_ADC_DAC_COMP,0,0xb);

    /* Write the 1024 digital sample to ULPSS memory */
    for(i = 0; i < 1024; i=i+2)
    {
     *(volatile uint32_t *)(0x24061000 + (j * 4)) = (dac_100k_n0dbfs[i+1] << 16) |
(dac_100k_n0dbfs[i]);
     j = j + 1;
    }

    /* Configure the ULP_GPIO4 as analog mode for DAC output */
    dac_output_pin_mux(OUTPUT_PIN_NUMBER);

    /* Divide DAC clock */
    RSI_DAC_ClkDivFactor(AUX_ADC_DAC_COMP, 16);

    /* Configure the DAC in FIFO mode */
    RSI_DAC_Config(AUX_ADC_DAC_COMP,DAC_FIFO_MODE,OUTPUT_MUX_ENABLE,OUTPUT_MUX_SEL);

    /* Set the FIFO threshold value */
    RSI_DAC_SetFifoThreshold(AUX_ADC_DAC_COMP,DAC_FIFO_THR);

    /* Start the DAC */
    RSI_DAC_Start(AUX_ADC_DAC_COMP,1);

    /* Enable the UDMA interrupt */
    NVIC_EnableIRQ(UDMA1_IRQn);

    while(1)
    {
        /* write data to dac data reg through udma*/
```

```
        UDMA_Write();
        /* Enable dma controller */
        UDMAdrv1->DMAEnable();
        /* Wait till dma done here external dma fill the DAC input buffer*/
        while(!done);
        done =0;
    }
}
```

Code Block 26 Digital to Analog converter example

35.3 API Descriptions

35.3.1 RSI_DAC_Config

Source File : rsi_dac.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_DAC_Config(AUX_ADC_DAC_COMP_Type *pstcDAC,uint8_t static_fifo_mode,uint16_t
aux_dac_out_mux_en,uint16_t aux_dac_out_mux_sel)
```

Description

This API is used to configure DAC in static or FIFO mode.

Parameters

Parameter	Description
pstcADC	Pointer to the AUX_ADC_DAC_COMP_Type
static_fifo_mode	This parameter define DAC in FIFO mode or static mode • static_fifo_mode = 0 , for Fifo mode • static_fifo_mode = 1 , for Static mode
aux_dac_out_mux_en	This parameter define DAC output play on AGPIO or not. • aux_dac_out_mux_en = 0 , DAC output not play on any AGPIO. • aux_dac_out_mux_en = 1 , DAC output on GPIO.
aux_dac_out_mux_sel	This parameter define DAC output play on which AGPIO • aux_dac_out_mux_sel = 0 , DAC output play on ULP_GPIO4 • aux_dac_out_mux_sel = 1 , DAC output play on ULP_GPIO15

Return values

Return zero on success.

Example

```
#define  OUTPUT_MUX_ENABLE  1
#define  OUTPUT_MUX_SEL     0   /* Select DAC output on ULP_GPIO4 */
#define  DAC_FIFO_MODE      0
/* Configure the DAC in FIFO mode */
RSI_DAC_Config(AUX_ADC_DAC_COMP,DAC_FIFO_MODE,OUTPUT_MUX_ENABLE,OUTPUT_MUX_SEL);
```

35.3.2 RSI_DAC_WriteData

Source File : rsi_dac.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_DAC_WriteData(AUX_ADC_DAC_COMP_Type *pstcDAC,uint16_t *data,
                          uint8_t static_fifo_mode,uint16_t len)
```

Description

This API is used to write input data to DAC in static or in FIFO mode.

Parameters

Parameter	Description
pstcADC	Pointer to the AUX_ADC_DAC_COMP_Type
data	This parameter define input data to DAC.
static_fifo_mode	This parameter define write data to DAC in FIFO mode or in static mode • static_fifo_mode = 1 , Write data to DAC in static mode • static_fifo_mode = 0 , Write data to DAC in fifo mode
len	Number of sample write to DAC input buffer.

Return values

Return zero on success , error code on failure.

Example

```
int32_t data[512]={/*Digital input samples */ };
RSI_DAC_WriteData(AUX_ADC_DAC_COMP,data,0,512);
```

35.3.3 RSI_DAC_Start

Source File : rsi_dac.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_DAC_Start( AUX_ADC_DAC_COMP_Type *pstcDAC ,uint16_t aux_dac_en)
```

Description

This API is used to start the DAC operation in Static or FIFO mode.

Parameters

Parameter	Description
pstcADC	Pointer to the AUX_ADC_DAC_COMP_Type
aux_dac_en	• aux_dac_en : 1 -> Enable DAC in static mode operation or FIFO mode operation. • aux_dac_en : 0 ->Enable DAC in static mode operation or FIFO mode operation.

Return values

Return zero on success , error code on failure.

Example

```
RSI_DAC_Start(AUX_ADC_DAC_COMP,1);
```

35.3.4 RSI_DAC_Stop

Source File : rsi_dac.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_DAC_Stop( AUX_ADC_DAC_COMP_Type *pstcDAC,uint32_t channel,
                      uint16_t dac_dyn_en,uint16_t fifo_mode)
```

Description

This API is used to stop the DAC operation in FIFO or Static mode.

Parameters

Parameter	Description
pstcADC	Pointer to the AUX_ADC_DAC_COMP_Type

Return values

Return zero on success , error code on failure.

Example

```
RSI_DAC_Stop(AUX_ADC_DAC_COMP);
```

35.3.5 RSI_DAC_ClkDivFactor

Source File : rsi_dac.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_DAC_ClkDivFactor(AUX_ADC_DAC_COMP_Type *pstcDAC, uint16_t div_factor)
```

Description

This API is used to set clock with configurable on time.

Parameters

Parameter	Description
pstcADC	Pointer to the AUX_ADC_DAC_COMP_Type
div_factor	Clock division factor to be programmed

Return values

Return zero on success , error code on failure.

Example

```
RSI_DAC_ClkDivFactor(AUX_ADC_DAC_COMP,4);
```

35.3.6 RSI_DAC_PowerControl

Source File : rsi_dac.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
void RSI_DAC_PowerControl(POWER_STATE_DAC state)
```

Description

This API is used to Power On and off for DAC.

Parameters

Parameter	Description
state	Pointer to the AUX_ADC_DAC_COMP_Type

Return values

Return zero on success , error code on failure.

Example

```
RSI_DAC_PowerControl(DAC_POWER_ON);
```

35.3.7 RSI_DAC_SetFifoThreshold

Source File : rsi_dac.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
void RSI_DAC_SetFifoThreshold(AUX_ADC_DAC_COMP_Type *pstcDAC,uint32_t fifo_threshold)
```

Description

This API is used to set fifo threshold value for DAC

Parameters

Parameter	Description
pstcDAC	Pointer to the AUX_ADC_DAC_COMP_Type
fifo_threshold	Fifo threshold value for DAC

Return values

Return zero on success , error code on failure.

Example

```
#define FIFO_THR     4
/* Select DAC output on ULP_GPIO4 */
RSI_DAC_SetFifoThreshold(AUX_ADC_DAC_COMP,FIFO_THR);
```

35.3.8 RSI_DAC_DynamicModeConfig

Source File : rsi_dac.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_DAC_DynamicModeConfig(AUX_ADC_DAC_COMP_Type *pstcDAC,uint16_t
channel_no,uint16_t aux_dac_out_mux_en,uint16_t aux_dac_out_mux_sel)
```

Description

This API is used to configure DAC in dynamic mode(DAC control through ADC module)

Parameters

Parameter	Description
pstcDAC	Pointer to the AUX_ADC_DAC_COMP_Type
channel_no	DAC channel to be configured as 0,1,2 ...15 when ADC multi channel enable is present
aux_dac_out_mux_en	This parameter define DAC output play on AGPIO or not. • aux_dac_out_mux_en = 0 , DAC output not play on any AGPIO. • aux_dac_out_mux_en = 1 , DAC output on GPIO.
aux_dac_out_mux_sel	This parameter define DAC output play on which AGPIO • aux_dac_out_mux_sel = 0 , DAC output play on ULP_GPIO4 • aux_dac_out_mux_sel = 1 , DAC output play on ULP_GPIO15

Return values

Return zero on success , error code on failure.

Example

```
#define CHANNEL_NUMBER    0
#define  OUTPUT_MUX_ENABLE  1
#define  OUTPUT_MUX_SEL     0
```

```
RSI_DAC_DynamicModeConfig(AUX_ADC_DAC_COMP,CHANNEL_NUMBER,OUTPUT_MUX_ENABLE,OUTPUT_MUX_SEL
);
```

35.3.9 RSI_DAC_DynamicModeWriteData

Source File : rsi_dac.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_DAC_DynamicModeWriteData(AUX_ADC_DAC_COMP_Type *pstcDAC,uint16_t channel_no,
                                     uint16_t *data,uint32_t len)
```

Description

This API is used to write input data DAC in dynamic mode(DAC control through ADC module)

Parameters

Parameter	Description
pstcDAC	Pointer to the AUX_ADC_DAC_COMP_Type
channel_no	DAC channel to be configured as 0,1,2 ...15 when ADC multi channel enable is present
data	Input buffer pointer
len	length of digital samples which play in DAC .

Return values

Return zero on success , error code on failure.

Example

```
#define CHANNEL_NUMBER    0
int32_t data[512]={/*Digital input samples */ };
RSI_DAC_DynamicModeWriteData(AUX_ADC_DAC_COMP,CHANNEL_NUMBER,data,512);
```

35.3.10 RSI_DAC_DynamicModeStart

Source File : rsi_dac.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_DAC_DynamicModeStart( AUX_ADC_DAC_COMP_Type *pstcDAC ,uint32_t channel,
                                  uint16_t aux_dac_en)
```

Description

This API is used to start the DAC in dynamic mode.(DAC control through ADC module)

Parameters

Parameter	Description

Parameter	Description
pstcDAC	Pointer to the AUX_ADC_DAC_COMP_Type
channel_no	DAC channel to be configured as 0,1,2 ...15 when ADC multi channel enable is present
aux_dac_en	This parameter define DAC enable or disable. • aux_dac_en = 0 , Disable DAC for dynamic mode operation. • aux_dac_en = 1 , Enable DAC for dynamic mode operation.

Return values

Return zero on success , error code on failure.

Example

```
#define CHANNEL_NUMBER    0
RSI_DAC_DynamicModeStart(AUX_ADC_DAC_COMP,CHANNEL_NUMBER,1);
```

35.3.11 RSI_DAC_DynamicModeStop

Source File : rsi_dac.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_DAC_DynamicModeStop( AUX_ADC_DAC_COMP_Type *pstcDAC,uint32_t channel)
```

Description

This API is used to stop the DAC in dynamic mode..(DAC control through ADC module)

Parameters

Parameter	Description
pstcDAC	Pointer to the AUX_ADC_DAC_COMP_Type
channel_no	DAC channel to be configured as 0,1,2 ...15 when ADC multi channel enable is present

Return values

Return zero on success , error code on failure.

Example

```
#define CHANNEL_NUMBER    0
RSI_DAC_DynamicModeStop(AUX_ADC_DAC_COMP,CHANNEL_NUMBER);
```

36 Building a new project in Keil iDE

This section explains how to create a new project using Keil uVision5 for ARM Cotex M4.

36.1 Start new project

Create a new project in Keil uVision5 IDE and select target as **"ARMCM4_FP"**.

Figure 15 Create New Project

36.2 Select device

Select device as **"ARMCM4_FP"** under ARM series.Press **"OK"** on Manage Run-Time Environment.

Figure 16 Select device

36.3 Add source files

For adding a source file, right click **"New Group"** and click **"Add New Items to Group 'New Group' ".** Similarly, you can add existing files to the same group by clicking **"Add Existing Files to Group 'New Group' ".** Refer to the image below:

Path for Redpine proprietary board.c source files : **MCU\board\src**

Path for Redpine proprietary peripheral source files : **MCU\library\driver\src**

Path for Redpine proprietary peripheral source files : **MCU\library\systemlevel\src**

Path for CMSIS supported peripheral source files : **MCU\library\cmsis**

Path for Redpine MCU Startup Files : **MCU\common\chip\src**

Path for Redpine MCU Startup File(.s file) : Add **"startup_RS1xxxx.s"** file which is located in sample example project in that startup folder .**MCU\example_projects\Keil\blinky\startup**

Figure 17 Add Source files

36.4 Include paths

In order to include file, click c/c++ tab. You may add header file path in include path.

Path for Redpine proprietary peripheral include files : **MCU\library\rom_driver\inc**

Path for Redpine proprietary peripheral include files : **MCU\library\driver\inc**

Path for Redpine proprietary peripheral include files : **MCU\library\systemlevel\inc**

Path for Redpine MCU Startup include files : **MCU\common\chip\inc**

Path for Redpine proprietary board.h include files : **MCU\board\inc**

Path for CMSIS supported peripheral include files : **MCU\library\cmsis**

Path for CMSIS supported peripheral config include files : **MCU\library\cmsis_driver\config**

Path for CMSIS supported peripheral driver include files : **library\cmsis_driver\CMSIS\Driver\Include**

Path for Common cmsis include file : **common\cmsis**

Figure 18 Add Include Paths

36.5 Creating Examples:

For adding a new example file, right click **"New Group"** and click **"Add New Items to Group 'New Group' "**. Refer to the image below , create example add in project group.

Figure 19 Add example file

Note :

This screenshot has been captured by using Keil uVision5.18.0.0. If version is changed, screenshot may also change.

36.6

37 Wake-Fi Receive

37.1 Introduction

This example demonstrates Wake-Fi pattern receive application. The feature(s) used in this example may or may not be available in your part. Refer to the product datasheet to verify the features available on your part.

37.1.1 Example Overview

37.1.1.1 Overview

This application set up the wake Rx configuration to detect the transmitter pattern. Set the specific match value for wurx to detect the pattern.

37.1.1.2 Sequence of Events

This example explains user how to:

- Set the match value for pattern detection purpose
- Configure the channel frequency for calibration purpose which is same to transmitter
- Set the pattern type which want to receive (e.g L1L2,L2 etc)
- Set the pattern length(e.g 64bits,32bits etc)
- Set the threshold value

37.1.1.3 MCU based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of MCU
- Redpine module

Figure 20 Setup diagram

37.2 Configuration and Execution of the Application

37.2.1 Configuring the Application

For executing the demo application of wurx, following parameter are required to change and verify the proper Wake-Fi application,

Open **'wurx_demo_example.c'** (Redpine_MCU_Vx.y.z\Host_MCU\Examples\Peripheral_Examples\wurx_demo_example) application file

in the project and change the required parameter as shown below:

1. **'L1_BYPASS_ENABLE'**
   This macro defines the L1 frequency pattern enabling or disabling. If L1L2 pattern is selected then set the macro value as 0.
   If L2 pattern is selected then set the macro value as 1.

```
#define L1_BYPASS_ENABLE                                                    1
```

2. **'L1_RECEIVE_FREQ_FAC'**
   This macro is used for selection of L1 frequency. Please note this macro is not used for L2 pattern.
   Following are the specified frequency values for L1 pattern:

   0 for 0.125 khz
   1 for 0.250 Khz
   2 for 0.5 khz
   3 for 1 khz

```
#define L1_RECEIVE_FREQ_FAC                                                 0
```

3. **'L2_RECEIVE_FREQ_FAC'**
   This macro is used for the selection of L2 frequency. It is also defined as receive frequency of wurx.
   Following are the specified frequency values for L2 pattern:

   0 for 4 khz
   1 for 8 Khz
   2 for 16 khz
   3 for 32 khz

```
#define L2_RECEIVE_FREQ_FAC                                                 3
```

4. **'IMPU_CALIB_CHANL_FREQ'**
   This macro defines the calibration of channel frequency "vco" and it is calculated by using following formula:
   Formula = (channel frequency / 2*40) * 2^6;

   Example: Consider the value of channel frequency as **2475**, then its calculated output will be ((2475/40*2)*2^6) = 1980 or 0x7bc

```
#define IMPU_CALIB_CHANL_FREQ                                               0x7BC
```

5. **'L1_PATTERN_LENGTH_SELECT'**
   This macro is used for selecting L1 frequency pattern length.
   Following are the specified frequency values for L1 pattern:

   0 for 2 bits
   1 for 4 bits
   2 for 8 bits
   3 for 16 bits

```
#define L1_PATTERN_LENGTH_SELECT                                            0
```

   Note:
   In the L2 pattern only mode this parameter not used anywhere.

6. **'L2_PATTERN_LENGTH_SELECT'**
   This macro is used for selecting L2 frequency pattern length.
   Following are the specified frequency values for L2 pattern:

1 for 1 bits
2 for 2 bits
3 for 4 bits
4 for 8 bits
5 for 16 bits
6 for 32 bits
0 and 7 for 64 bits

```
#define L2_PATTERN_LENGTH_SELECT                                    7
```

7. **'PATTERN_LENGTH'**
   This macro is used for selecting pattern length for various patterns such as 64 bits,32 bits and so on.

```
#define PATTERN_LENGTH                                              64
```

8. **'THRESHOLD_PERCENTAGE'**
   This parameter is useful for setting the pattern threshold value.
   The pattern length will be shown in percentage format such as 75%, 90% and so on.

```
#define THRESHOLD_PERCENTAGE                                        75
```

9. For the wurx detection purpose set the pattern match value.
   The match value compares with the received pattern and if the match value and received pattern are same then interrupt gets generated.

```
uint32_t Match_value[3]={0xC78E1,0x30319D,0x2783C7};
```

In above array,
Match_value[0] defines MSB of pattern
Match_value[1] defines MID of pattern
Match_value[2] defines LSB of pattern

Here, MSB is 20 bits, MID is 22 bit and LSB is 22 bits. Please refer to the below note for better understanding.

Note:

If we consider pattern length as 64 and pattern is as following {msb:11000111100011100001,MID:1100000011000110011101,LSB:1001111000001111000111}For above pattern divide three field as below:1. LSB = For lsb field take LSB 22 bits and calculate the hex value e.g 1001111000001111000111 = 0x2783C72. MID = For MID register take middle 22 bits and calculate the hex value for that binary values 1100000011000110011101 = 0x30319D,3. MSB = For MSB register MSB 20 bits and calculate the hex value for that binary values 11000111100011100001= 0xC78E1.

37.2.2 Executing the Application

1. Connect Redpine device to the PC open IDE.
2. Configure the wurx parameter by using above apis .
3. Build and launch the application.
4. After launch, navigate to M4 sleep wait for wurx pattern detection.
5. After required pattern is detected, the wurx interrupt occurs and M4 module wakes up. On every successful detection of pattern, tri-LED blinks.

37.2.3 API Descriptions

37.2.3.1 RSI_WURX_Init

Source File : rsi_wurx.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
void RSI_WURX_Init(uint16_t bypass_l1_enable,uint16_t l1_freq_div,uint16_t l2_freq_div)
```

Description

This API is used to initialization of l1 and l2 frequency as well as enable wurx.

Parameters

Parameter	Description
bypass_l1_enable	Enable or disable the bypass functionality of level1
l1_freq_div	Set the level1 frequency by using division factor ,This parameter define frequency. • 0 : 0.125 khz, • 1: 0.250 Khz, • 2: 0.5 khz , • 3: 1 khz, • default : 2 khz
l2_freq_div	Set the level2 frequency by using division factor ,This parameter define frequency. • 0 : 4 khz, • 1: 8 Khz, • 2: 16 khz , • 3: 32 khz

Return values

None

Example

```
#define L1_BYPASS_ENABLE              0     /* L1 : 0 l1 bypass disable and L1 : 1 l1
bypass enable  */
#define L1_RECEIVE_FREQ_FAC           4     /* l1 receive frequecy selection value  as 2
khz*/
#define L2_RECEIVE_FREQ_FAC           3     /* l2 receive frequecy selection value  as
32khz*/
/* Set the l1 and l2 receive frequcy value */
RSI_WURX_Init(L1_BYPASS_ENABLE,L1_RECEIVE_FREQ_FAC,L2_RECEIVE_FREQ_FAC);
```

37.2.3.2 RSI_IPMU_40MhzClkCalib

Source File : rsi_wurx.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
void RSI_IPMU_40MhzClkCalib(uint16_t clk_enable,uint32_t channel_selection_value)
```

Description

This API is used to calculate the 40MHZ VCO calibration.

Parameters

Parameter	Description
clk_enable	Clock enable
channel_selection_value	This parameter define on which channel frequency use for transmission . E.g. If channel 11 use for transmission the channel_11 = 2475. channel_selection_value = (((channel_11)/(cal_clock*2)) * 2^6) Here cal_clock is 40Mhz So [channel_selection_value =(((2475)/(40*2)) * 2^6) Note : Refer RSI_WURX_CalVCOCalFreq() API ,its return calculated value.

Return values

None

Example

```
#define IMPU_CALIB_CHANL_FREQ         0x7BC /* channel_11 select for calibration
(((2475)/80)*64) */
/* Start the VCO calibration */
RSI_IPMU_40MhzClkCalib(1,IMPU_CALIB_CHANL_FREQ);
```

37.2.3.3 RSI_IPMU_DCCalib

Source File : rsi_wurx.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
void RSI_IPMU_DCCalib()
```

Description

This API is used to calculate the manual DC calibration as well as enable periodic detection enable.

Parameters

None

Return values

None

Note

```
Before the calling DC calibration API,Enable the correlation clock by using
'RSI_WURX_CorrEnable(1)' API.
```

Example

```
/* Enable the DC calibration for wurx */
RSI_IPMU_DCCalib();
```

37.2.3.4 RSI_WURX_CorrEnable

Source File : rsi_wurx.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
void RSI_WURX_CorrEnable(uint16_t wurx_enable)
```

Description

This API is used enable wurx correlation.

Parameters

Parameter	Description
wurx_enable	Enable wurx correlation

Return values

None

Example

```
/* Enable the correlation clock */
RSI_WURX_CorrEnable(1);
```

37.2.3.5 RSI_WURX_SetWakeUpThreshold

Source File : rsi_wurx.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
void RSI_WURX_SetWakeUpThreshold(uint16_t threshold_1 ,uint16_t threshold_2)
```

Description

This API is used set up threshold value for operation.

Parameters

Parameter	Description
threshold_1	Threshold value for pattern1.
threshold_2	Threshold value for pattern2.

Return values

None

Example

```
volatile uint16_t threshold1;
/* Calculate the detection threshold */
threshold1=RSI_WURX_CalThershValue(PATTERN_LENGTH,THRESHOLD_PERCENTAGE);

/* Set the the Threshold value */
RSI_WURX_SetWakeUpThreshold(threshold1,threshold1);
```

37.2.3.6 RSI_WURX_Pattern2DetectionEnable

Source File : rsi_wurx.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
void RSI_WURX_Pattern2DetectionEnable(uint16_t enable)
```

Description

This API is used enable pattern2.

Parameters

Parameter	Description
enable	Enable the pattern2 detection bit

Return values

None

Example

```
RSI_WURX_Pattern2DetectionEnable(ENABLE);
```

37.2.3.7 RSI_WURX_TailDataDecodeEnable

Source File : rsi_wurx.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
void RSI_WURX_TailDataDecodeEnable(uint16_t enable,uint16_t data_len)
```

Description

This API is used to enable the Tail data decode.

Parameters

Parameter	Description
enable	Enable the tail data decode bit.
data_len	Set the detection bit length. • 0 for 64 bit • 1 for 128 bit • 2 for 192 bit • 3 for 256 bit

Return values

None

Example

```
RSI_WURX_TailDataDecodeEnable(ENABLE,0);
```

37.2.3.8 RSI_WURX_GetTailData

Source File : rsi_wurx.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_WURX_GetTailData(uint32_t *tail_data,uint16_t tail_data_len)
```

Description

This API is used get the tail data

Parameters

Parameter	Description
tail_data	Pointer to store the tail data.
tail_data_len	This parameter define number of bit read in tail data. • 0 for 64 bit • 1 for 128 bit • 2 for 192 bit • 3 for 256 bit

Return values

return status this api ,its pass or fail.

Example

```
#define NUMBER_TAIL_DATA_BITS  0
uint32_t data[2];
RSI_WURX_GetTailData(data,NUMBER_TAIL_DATA_BITS);
```

37.2.3.9 RSI_WURX_CalThershValue

Source File : rsi_wurx.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
uint32_t RSI_WURX_CalThershValue(uint32_t bit_length,uint32_t percentage)
```

Description

This API is used to calculate the threshold value.

Parameters

Parameter	Description
bit_length	Bit length 64 or 32 bit.

Parameter	Description
percentage	Percentage the calculate the threshold value.

Return values

return the threshold value.

Example

```
volatile uint16_t threshold1;
/* Calculate the detection threshold */
threshold1=RSI_WURX_CalThershValue(PATTERN_LENGTH,THRESHOLD_PERCENTAGE);
```

37.2.3.10 RSI_WURX_Pattern1MatchValue

Source File : rsi_wurx.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
void RSI_WURX_Pattern1MatchValue(uint32_t *match_value)
```

Description

This API is used set the match value for detection pattern1 purpose.

Parameters

Parameter	Description
match_value	Pointer of match value array. Match value array contain 3 element of uint32_t types. E.g • Match_value[0] = MSB of pattern • Match_value[1] = MID of pattern • Match_value[2] = LSB of pattern

Return values

None

Example

```
/* MSB_MID_LSB */
uint32_t Match_value[3]={0xa6a56,0x1A5595,0x1965a9};
RSI_WURX_Pattern1MatchValue(Match_value);
```

37.2.3.11 Note

```
If we consider pattern length as 64 and pattern is as following
{msb:11000111100011100001,MID:1100000011000110011101,LSB:1001111000001111000111}
For above pattern divide three field as below:
1. LSB  = For lsb field take LSB 22 bits and calculate the hex value e.g
1001111000001111000111 =  0x2783C7
2.MID   = For MID register take middle 22 bits and calculate the hex value for that binary
values  1100000011000110011101 = 0x30319D,
3. MSB = For MSB register MSB  20 bits and calculate the hex value for that binary values
11000111100011100001= 0xC78E1.
```

37.2.3.12 RSI_WURX_Pattern2MatchValue

Source File : rsi_wurx.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
void RSI_WURX_Pattern2MatchValue(uint32_t *match_value)
```

Description

This API is used set the match value for detection pattern2 purpose.

Parameters

Parameter	Description
match_value	Pointer of match value array. Match value array contain 3 element of uint32_t types. E.g • Match_value[0] = MSB of pattern • Match_value[1] = MID of pattern • Match_value[2] = LSB of pattern

Return values

None

Example

```
/* MSB_MID_LSB */
uint32_t Match_value[3]={0xa6a56,0x1A5595,0x1965a9};
RSI_WURX_Pattern2MatchValue(Match_value);
```

37.2.3.13 Note

```
If we consider pattern length as 64 and pattern is as following
{msb:11000111100011100001,MID:1100000011000110011101,LSB:1001111000001111000111}
For above pattern divide three field as below:
1. LSB  = For lsb field take LSB 22 bits and calculate the hex value e.g
1001111000001111000111 =  0x2783C7
2.MID   = For MID register take middle 22 bits and calculate the hex value for that binary
values  1100000011000110011101 = 0x30319D,
3. MSB = For MSB register MSB  20 bits and calculate the hex value for that binary values
11000111100011100001= 0xC78E1.
```

37.2.3.14 RSI_WURX_SetPatternLength

Source File : rsi_wurx.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
void RSI_WURX_SetPatternLength(uint16_t enable,uint16_t l1_len,uint16_t l2_len)
```

Description

This API is used set pattern length for wurx.

Parameters

Parameter	Description
enable	Enable the set pattern length API
l1_len	value to decide the l1 pattern length. Ranges • 0: 2 bits , • 1: 4 bits , • 2:8 bits • 3: 16 bits
l2_len	value to decide the l2 pattern length . Ranges • 1 : 1 bits, • 2: 2 bits, • 3: 4 bits • 4 : 8 bits, • 5: 16 bits, • 6: 32 bits

Return values

None

Example

```
#define L1_PATTERN_LENGTH_SELECT      1     /* l1 pattern length as 4bit */
#define L2_PATTERN_LENGTH_SELECT      7     /* l2 pattern length as 64bit*/
/* Set the pattern length for l1 and l2 frequency */
RSI_WURX_SetPatternLength(1,L1_PATTERN_LENGTH_SELECT,L2_PATTERN_LENGTH_SELECT);
```

37.2.3.15 RSI_WURX_ReadPatternLength

Source File : rsi_wurx.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
uint16_t RSI_WURX_ReadPatternLength()
```

Description

This API is used read pattern length

Parameters

None

Return values

Return configured pattern length.

Example

```
uint16_t pattern_length;
pattern_length = RSI_WURX_ReadPatternLength();
```

37.2.3.16 RSI_WURX_AnalogOff

Source File : rsi_wurx.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
void RSI_WURX_AnalogOff()
```

Description

This API is used to off the wurx analog block.

Parameters

None

Return values

None

Example

```
RSI_WURX_AnalogOff();
```

37.2.3.17 RSI_WURX_DigitalOff

Source File : rsi_wurx.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
void RSI_WURX_DigitalOff()
```

Description

This API is used to off the wurx digital block.

Parameters

None

Return values

None

Example

```
RSI_WURX_DigitalOff();
```

37.2.3.18 RSI_WURX_TaildataPresent

Source File : rsi_wurx.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
uint16_t RSI_WURX_TaildataPresent()
```

Description

This API is used to verify the tail data detection is present or not.

Parameters

None

Return values

Return the number of tail data bits, Function return following possible values

- 0 : 64 bits tail data
- 1 : 128 bits tail data.
- 2 : 192 bits tail data
- 3 : 256 bits tail data.

Example

```
uint16_t tail_data;
tail_data=RSI_WURX_TaildataPresent();
```

37.2.3.19 RSI_WURX_SoftwareRestart

Source File : rsi_wurx.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
void RSI_WURX_SoftwareRestart(void)
```

Description

This API is used to do software restart of wurx.

Parameters

None

Return values

None

Example

```
RSI_WURX_SoftwareRestart();
```

37.2.3.20 RSI_WURX_GetPatternType

Source File : rsi_wurx.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
int32_t RSI_WURX_GetPatternType(void)
```

Description

This API is use to get pattern type . e.g pattern1 or pattern2 or false wake up.

Parameters

None

Return values

Pattern type .if its return 1 then pattern1 and 2 then pattern2 , if -1 then false wake up.

Note

This API should be call before clearing wurx interrupt .

Example

```
volatile uint16_t pattern_type;
pattern_type =  RSI_WURX_GetPatternType(); /* Get the pattern type */
```

37.2.3.21 RSI_WURX_CalVCOCalFreq

Source File : rsi_wurx.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
uint32_t RSI_WURX_CalVCOCalFreq(uint32_t frequncy_value)
```

Description

This API is use VCO calibration frequency value.

Parameters

Parameter	Description
frequncy_value	Transmission channel frequency value.

Return values

Return the VCO calibration frequency value.

Note

This API should be call before 'RSI_IPMU_40MhzClkCalib()' to calculate the IPMU calibration frequency.

Example

```
#define CHANNEL_NUMBER_FREQUENCY   2575
volatile uint16_t freq_val;

/* Calculate the VCO calibration value */
freq_val = RSI_WURX_CalVCOCalFreq(CHANNEL_NUMBER_FREQUENCY);
```

37.2.3.22 RSI_WURX_BGSamplingEnable

Source File : rsi_wurx.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
void RSI_WURX_BGSamplingEnable()
```

Description

This API is use BG sampling enable when m4 in power save mode.

Parameters

None

Return values

None.

Example

```
RSI_WURX_BGSamplingEnable();
```

38 Voice Activity Detection(VAD)

38.1 Overview

This section explains how to configure and use the VAD using Redpine MCU SAPIs.

38.2 Programming Sequence

```
#include "rsi_chip.h"     /* Redpine_MCU_Vx.y.z\Host_MCU\common\chip\inc */
#include "rsi_board.h"    /* Redpine_MCU_Vx.y.z\Host_MCU\common\board\inc */
#include "rsi_opamp.h"     /* Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc */
/* Private typedef -----------------------------------------------------------*/

/* Private macro -------------------------------------------------------------*/
#define PING_IRQHandler            IRQ000_Handler
#define ADC_IRQHandler             IRQ011_Handler
/* Private define ------------------------------------------------------------*/
#define ADC_CHNNNO                 0
#define VAD_PING_ADDR              0x1000
#define ADC_PING_MEM               0x24060000
#define ADC_PONG_MEM               0x240603FF
#define ADC_PING_LENG              1023
#define ADC_PONG_LENG              1023
#define VAD_ENERGY_THRSH           0x28
#define VAD_ACF_THRSH              0x2ff
#define VAD_ZCR_THRSH              0x190
#define VAD_WACF_THRSH             0
#define DATA_SOURCE_SELECT         0
#define SAMPL_PER_ADDR             1
#define SAMPL_PER_FRAME            0x3ff
#define VAD_ALGO_SELECT            0x5
#define START_DELAY                0x5
#define END_DELAY                  0x50
#define VAD_DIG_SCALE              2
#define MAXIMUM_ADC_SAMPLE         1023
#define NPSS_GPIO_PIN              3
#define ADC_FIFO_THR               3
#define INPUT_POS_PIN_NUM          6
#define INPUT_NEG_PIN_NUM          7
/* Private variables ---------------------------------------------------------*/
int16_t ping_buffer[1024];
int16_t pong_buffer[1024];
volatile int32_t dc_est =0,dc = 0;
volatile uint32_t adc_var;
volatile uint8_t energy_status=0,flag=0;
/* OPAMP structure configuration ---------------------------------------------------------
*/
opamp_config opamp_struct =
{
{
     /*opamp1_dyn_en;*/         1,
     /*opamp1_sel_p_mux;*/      4,
     /*opamp1_sel_n_mux;*/      3,
     /*opamp1_lp_mode;*/        1,
     /*opamp1_r1_sel;*/         1,
     /*opamp1_r2_sel;*/         1,
     /*opamp1_en_res_bank;*/    1,
     /*opamp1_res_mux_sel;*/    1,
     /*opamp1_res_to_out_vdd;*/ 0,
     /*opamp1_out_mux_en;*/     0,
   /*opamp1_out_mux_sel;*/    0,
     /*opamp1_enable;*/         1
},
    {
        /*   opamp2_dyn_en;*/      0,
        /*      opamp2_sel_p_mux;*/   0,
        /*     opamp2_sel_n_mux;*/   0,
```

```
        /*      opamp2_lp_mode;*/      0,
        /*      opamp2_r1_sel;*/       0,
        /*      opamp2_r2_sel;*/       0,
        /*      opamp2_en_res_bank;*/ 0,
        /*      opamp2_res_mux_sel;*/ 0,
        /*      opamp2_res_to_out_vdd;*/ 0,
        /*      opamp2_out_mux_en;*/      0,
        /*      opamp2_enable;*/          0,

    },
    {
        /* opamp3_dyn_en;*/           0,
        /* opamp3_sel_p_mux;*/        0,
        /* opamp3_sel_n_mux;*/        0,
        /* opamp3_lp_mode;*/          0,
        /* opamp3_r1_sel;*/           0,
        /* opamp3_r2_sel;*/           0,
        /* opamp3_en_res_bank;*/      0,
        /* opamp3_res_mux_sel;*/      0,
        /* opamp3_res_to_out_vdd;*/   0,
        /* opamp3_out_mux_en;*/       0,
        /* opamp3_enable;*/           0,
    }
};

/**
 * @brief  VAD IRQ handler , clear the VAD interrupt.
 * @param  None
 * @retval None
 */
void PING_IRQHandler()
{
    RSI_VAD_InterruptClr(VAD,1);
  RSI_NPSSGPIO_SetPin(NPSS_GPIO_PIN,1U);
}

/**
 * @brief  ADC IRQ handler , clear the ADC interrupt and process the ADC data to VAD
engine.
 * @param  None
 * @retval None
 */
void ADC_IRQHandler()
{
  RSI_ADC_ChnlIntrClr(AUX_ADC_DAC_COMP,ADC_CHNNNO);
    if(adc_var)
    {
    RSI_ADC_PingPongReInit(AUX_ADC_DAC_COMP,ADC_CHNNNO,0,1);
        dc_est = dc_est >>10;
        RSI_ADC_ReadData(AUX_ADC_DAC_COMP,pong_buffer,0,ADC_CHNNNO,0,0);
        dc_est =
RSI_VAD_ProcessData(VAD,VAD_PING_ADDR,pong_buffer,dc_est,VAD_DIG_SCALE,32);
        dc_est = RSI_VAD_ProcessData(VAD,VAD_PING_ADDR,pong_buffer,dc_est,VAD_DIG_SCALE,
                                     MAXIMUM_ADC_SAMPLE);
      adc_var=0;
    }
    else
    {
        RSI_ADC_PingPongReInit(AUX_ADC_DAC_COMP,ADC_CHNNNO,1,0);
        dc_est = dc_est >>10;
        RSI_ADC_ReadData(AUX_ADC_DAC_COMP,ping_buffer,1,ADC_CHNNNO,0,0);
        dc_est =
RSI_VAD_ProcessData(VAD,VAD_PING_ADDR,ping_buffer,dc_est,VAD_DIG_SCALE,32);
        dc_est = RSI_VAD_ProcessData(VAD,VAD_PING_ADDR,ping_buffer,dc_est,VAD_DIG_SCALE,
                                     MAXIMUM_ADC_SAMPLE);
      adc_var=1;
    }
  RSI_NPSSGPIO_SetPin(NPSS_GPIO_PIN,0U);
    flag=1;
}
```

```
void input_pin_mux(uint16_t pos_input,uint16_t neg_input)
{
  /* MIC output pin mux */
  RSI_EGPIO_SetPinMux(EGPIO1,0,pos_input,EGPIO_PIN_MUX_MODE7);
  /* Constant voltage output

    pin mux */
  RSI_EGPIO_SetPinMux(EGPIO1,0,neg_input,EGPIO_PIN_MUX_MODE7);
}
/**
 * @brief  This function configure ADC parameter.
 * @param  None
 * @retval None
 */
void RSI_ADC_VAD()
{
    /* Enable power to ADC */
    RSI_ADC_PowerControl(ADC_POWER_ON);
    /* Configure the 32MRC clock to ADC */
    RSI_ULPSS_AuxClkConfig(ULPCLK, ENABLE_STATIC_CLK ,ULP_AUX_32MHZ_RC_CLK);
    /* Bypass the LDO mode */
    RSI_ADC_AUXBypassLdoEnable(AUX_ADC_DAC_COMP,1,0xb);
    /* divide clock to 16 and get 1MHZ for ADC calibration */
    RSI_ADC_ClkDivfactor(AUX_ADC_DAC_COMP,0,16);
    /* ADC calibration */
    RSI_ADC_Calibration();
    /* Configure duty cycle */
    RSI_ADC_ClkDivfactor(AUX_ADC_DAC_COMP,320,340);
    /* Configure FIFO threshold value */
    RSI_ADC_Config(AUX_ADC_DAC_COMP,1,0,ADC_FIFO_THR,1);
    /* Configure channel configuration value */
    RSI_ADC_ChannelConfig(AUX_ADC_DAC_COMP,ADC_CHNNNO,20,5,1);
  /* Configure OPAMP */
    RSI_Opamp1_Config(OPAMP,0, & opamp_struct);
    /* Configure channel sampling rate */
    RSI_ADC_ChannelSamplingRate(AUX_ADC_DAC_COMP,ADC_CHNNNO,0,11);
    /* Configure ping and pong enable */
    RSI_ADC_PingpongEnable(AUX_ADC_DAC_COMP,ADC_CHNNNO);
    /* Configure internal DMA enable */
    RSI_ADC_InternalPerChnlDmaEnable(AUX_ADC_DAC_COMP,ADC_CHNNNO);
    /* ADC channel enable */
    RSI_ADC_ChnlEnable(AUX_ADC_DAC_COMP,ADC_CHNNNO);
    /* Configure the ping and pong memory address */
  RSI_ADC_PingPongMemoryAdrConfig(AUX_ADC_DAC_COMP,ADC_CHNNNO,ADC_PING_MEM,
                                  ADC_PONG_MEM,ADC_PING_LENG,ADC_PONG_LENG,1,1);
  /* Unmask the channel interrupt */
    RSI_ADC_ChnlIntrUnMask(AUX_ADC_DAC_COMP,ADC_CHNNNO);
    /* Start ADC */
  RSI_ADC_Start(AUX_ADC_DAC_COMP);
}

/**
 * @brief  Main program.
 * @param  None
 * @retval None
 */
int main(void)
{
    /* At this stage the MICROCONTROLLER clock setting is already configured,
     * this is done through SystemInit() function which is called from startup
     * file (startup_rs1xxxx.s) before to branch to application main.
     * To reconfigure the default setting of SystemInit() function, refer to
     * system_rs1xxxx.c file
     */
    SystemCoreClockUpdate();

    RSI_NPSSGPIO_InputBufferEn(NPSS_GPIO_PIN , 1U);

    RSI_NPSSGPIO_SetPinMux(NPSS_GPIO_PIN , 0);

    RSI_NPSSGPIO_SetDir(NPSS_GPIO_PIN , NPSS_GPIO_DIR_OUTPUT);
```

```
    /* Configure the fast and slow clock for VAD */
    RSI_ULPSS_VadClkConfig(ULPCLK,ULP_VAD_32KHZ_RC_CLK,ULP_VAD_32MHZ_RC_CLK,4);

    /* Configure the sample per frame and sample per address for VAD */
    RSI_VAD_Config(VAD,SAMPL_PER_FRAME,SAMPL_PER_ADDR,1,DATA_SOURCE_SELECT) ;

    /* Select the algorithm and algorithm threshold for VAD */

RSI_VAD_SetAlgorithmThreshold(VAD,VAD_ALGO_SELECT,VAD_ZCR_THRSH,VAD_ACF_THRSH,VAD_WACF_THR
SH,0);

    /* Set the start delay and end delay for ACF algorithm*/
    RSI_VAD_Set_Delay(VAD,START_DELAY,END_DELAY);

    /* Set energy threshold value */
    RSI_VAD_FrameEnergyConfig(VAD,VAD_ENERGY_THRSH,1,1) ;

    /* Configure the ping and pong address configuration*/
    RSI_VAD_PingPongMemoryAddrConfig(VAD,VAD_PING_ADDR,0,1);

    input_pin_mux(INPUT_POS_PIN_NUM,INPUT_NEG_PIN_NUM);

    NVIC_EnableIRQ(VAD_INTR_PING_IRQn);

    NVIC_EnableIRQ(VAD_INTR_PONG_IRQn);

    NVIC_EnableIRQ(ADC_IRQn);

    /* Set the ADC and OPAMP configuration */
    RSI_ADC_VAD();

  while(1)
    {
     /* waiting for VAD interrupt */
     while(!flag);
   flag = 0;
     energy_status = RSI_VAD_ProccessDone(VAD);
     if(energy_status)
     {
       RSI_VAD_FastClkEnable(ULP_VAD_32MHZ_RC_CLK,0);
     }
    }
    return 0;
}
```

Code Block 27 Voice activity detection example

38.3 API Descriptions

38.3.1 RSI_VAD_PingPongMemoryAddrConfig

Source File : rsi_vad.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
void RSI_VAD_PingPongMemoryAddrConfig(RSI_VAD_T *pVAD,uint32_t ping_addr,
                                      uint32_t pong_addr,uint16_t ping_enable)
```

Description

This API is used to configure ping and pong address for VAD.

Parameters

Parameter	Description
pVAD	Pointer to the VAD_Type structure.
ping_addr	13 bit ulp mem address offset for ping buffer
pong_addr	13 bit ulp mem address offset for pong buffer
ping_enable	This parameter enable the ping address configuration. ping_enable - 1 (ping address enable) ping_enable - 0 (pong address enable)

Return values

None

Example

```
#define VAD_PING_ADDR              0x1000
#define PING_ENABLE                1

RSI_VAD_PingPongMemoryAddrConfig(VAD,VAD_PING_ADDR,0,PING_ENABLE);
```

38.3.2 RSI_VAD_Config

Source File : rsi_vad.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_VAD_Config( RSI_VAD_T *pVAD,uint16_t samples_per_frame,
                        uint16_t samples_per_address,bool fullwidth,uint8_t
datasourceselect)
```

Description

This API is used to configure the VAD parameter

Parameters

Parameter	Description
pVAD	Pointer to the VAD_Type structure.
samples_per_frame	Number of samples in one processing frame, maximum value is 1023 and default is 512
samples_per_addres	Number of samples for address • 0 - 4 samples per address • 1 - 2 Samples per address • 2 - 1 Sample per address
fullwidth	• 0 - 12/24 when VAD_REG1_ADDR[21:20] • 1 - 2 Samples per address
datasourceselect	Source of Data for VAD processing • 00/10: Internal Register

Parameter	Description
	• 11: ADC as source • 01: Reserved

Return values

Return zero on success , error code on failure of API.

Example

```
#define SAMPL_PER_FRAME            0x3ff
#define SAMPL_PER_ADDR             1
#define DATA_SOURCE_SELECT         0

RSI_VAD_Config(VAD,SAMPL_PER_FRAME,SAMPL_PER_ADDR,1,DATA_SOURCE_SELECT) ;
```

38.3.3 RSI_VAD_Enable

Source File : rsi_vad.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
void RSI_VAD_Enableprocess(void)
```

Description

This API is used to Enable Processing of VAD.

Parameters

Parameter	Description
pVAD	Pointer to the VAD_Type structure.

Return values

None

Example

```
RSI_VAD_Enable(VAD);
```

38.3.4 RSI_VAD_InterruptClr

Source File : rsi_vad.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
void RSI_VAD_InterruptClr(RSI_VAD_T *pVAD,uint16_t ping_interrupt)
```

Description

This API is used to clear the interrupt of VAD

Parameters

Parameter	Description

Parameter	Description
pVAD	Pointer to the VAD_Type structure.
ping_interrupt	This parameter define which interrupt want to be clear. • ping_interrupt = 1 , To clear the VAD ping interrupt. • ping_interrupt = 0 , To clear the VAD pong interrupt.

Return values

None

Example

```
RSI_VAD_InterruptClr(VAD,1);
```

38.3.5 RSI_VAD_SetAlgorithmThreshold

Source File : rsi_vad.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_VAD_SetAlgorithmThreshold( RSI_VAD_T *pVAD,uint16_t algorithm_type,
                                       uint32_t zcr_threshold,
                                       uint32_t acf_threshold,uint32_t wacf_threshold,
                                       VAD_AMDF_THRESHOLD_T *config )
```

Description

This API is used to set algorithm and threshold value for that algorithm.

Parameters

Parameter	Description
pVAD	Pointer to the VAD_Type structure.
algorithm_type	Select the algorithm type here refer this #VAD_ALGORITHM_SELECT_T enum. pass the specific value for selection of algorithm • 0 - ZCR • 1 - ACF • 2 - AMDF • 3 - WACF • 4 - ZCR_ACF_AMDF_WACF • 5 - ZCR_ACF • 6 - ZCR_AMDF • 7 - ZCR_WACF • 8 - ZCR_WACF
zcr_threshold	This parameter define threshold value for zcr algorithm maximum value is 1023 and default value is 50.
acf_threshold	This parameter define threshold value for acf algorithm maximum value is 4095 and default value is 1024.

Parameter	Description
admf_threshold	This parameter define threshold value for acf algorithm maximum value is 4095 default value is 1024.
wacf_threshold	This parameter define threshold value for wacf_threshold algorithm maximum value is 4095 and default value is 51.
config	VAD_AMDF_THRESHOLD_T structure veriable, configure this structure for AMDF algorithm delay.

Return values

RSI_OK on success of this api execution.

Example

```
#define VAD_ALGO_SELECT            0x5
#define VAD_ACF_THRSH              0x2ff
#define VAD_ZCR_THRSH              0x190
#define VAD_WACF_THRSH             0
/* Select the algorithm and algorithm threshold for VAD */
RSI_VAD_SetAlgorithmThreshold(VAD,VAD_ALGO_SELECT,VAD_ZCR_THRSH,VAD_ACF_THRSH,VAD_WACF_THR
SH,0);
```

38.3.6 RSI_VAD_Set_Delay

Source File : rsi_vad.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_VAD_Set_Delay(RSI_VAD_T *pVAD,uint16_t startdelayval,uint16_t enddelayval)
```

Description

This API is used to set algorithm and threshold value for that algorithm.

Parameters

Parameter	Description
pVAD	Pointer to the VAD_Type structure.
startdelayval	This parameter define the start delay value for ACF,WACF,AMDF algorithm. maximum value is 1023 and default value is 2
enddelayval	This parameter define the end delay value for ACF,WACF,AMDF algorithm. maximum value is 1023 and default value is 16

Return values

RSI_OK on success full execution.

Example

```
#define START_DELAY                0x5
#define END_DELAY                  0x50

/* Set the start delay and end delay for ACF algorithm*/
RSI_VAD_Set_Delay(VAD,START_DELAY,END_DELAY);
```

38.3.7 RSI_VAD_Input

Source File : rsi_vad.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_VAD_Input(RSI_VAD_T *pVAD, int16_t data)
```

Description

This API is used to give the input for VAD.

Parameters

Parameter	Description
pVAD	Pointer to the VAD_Type structure.
data	This parameter input for VAD block input is 1023 and default value is 16

Return values

RSI_OK on success full execution.

Example

```
int16_t data ;     /* ADC output data */
RSI_VAD_Input(VAD,data);
```

38.3.8 RSI_VAD_FrameEnergyConfig

Source File : rsi_vad.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
error_t RSI_VAD_FrameEnergyConfig(RSI_VAD_T *pVAD,uint32_t threshold_frame_energy,
                                  uint32_t threshold_smpl_collect,uint32_t
prog_smpls_for_energy_check)
```

Description

This API is used to configure the frame energy.

Parameters

Parameter	Description
pVAD	Pointer to the VAD_Type structure.
threshold_frame_energy	This parameter give the threshold frame energy, maximum value is 1023 and default value is 0.
threshold_smpl_collect	This parameter give the number of threshold sample collect, maximum value is 1023 and default value is 1.
prog_smpls_for_energy_check	This parameter define the sample for energy check, maximum value is 3 and default value is 1.

Return values

RSI_OK on success full execution.

Example

```
#define VAD_ENERGY_THRSH           0x28
/* Set energy threshold value */
 RSI_VAD_FrameEnergyConfig(VAD,VAD_ENERGY_THRSH,1,1) ;
```

38.3.9 RSI_VAD_Stop

Source File : rsi_vad.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
void RSI_VAD_Stop(RSI_VAD_T *pVAD)
```

Description

This API is used to disable VAD functionality.

Parameters

Parameter	Description
pVAD	Pointer to the VAD_Type structure.

Return values

None

Example

```
RSI_VAD_Stop(VAD);
```

38.3.10 RSI_VAD_ProccessDone

Source File : rsi_vad.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
uint16_t RSI_VAD_ProccessDone(RSI_VAD_T *pVAD)
```

Description

This API is used show the VAD energy detect.

Parameters

Parameter	Description
pVAD	Pointer to the VAD_Type structure.

Return values

Return the VAD energy detection status

- If the return value 1 - Energy detect
- If the return value 0 - No energy detect

Example

```
volatile uint8_t energy_status=0;
energy_status = RSI_VAD_ProccessDone(VAD);
```

38.3.11 RSI_VAD_FastClkEnable

Source File : rsi_vad.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
void RSI_VAD_FastClkEnable(uint16_t fast_clk_sel,uint16_t clk_div_factor)
```

Description

This API is used enable fast clock for VAD peripheral.

Parameters

Parameter	Description
fast_clk_sel	fast clock select for VAD peripheral
clk_div_factor	Select the clock division factor for VAD peripheral

Return values

None

Example

```
RSI_VAD_FastClkEnable(ULP_VAD_32MHZ_RC_CLK,0);
```

38.3.12 RSI_VAD_ProcessData

Source File : rsi_vad.c

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src

Prototype

```
int32_t RSI_VAD_ProcessData(RSI_VAD_T *pVAD,uint32_t vad_addr,
                            int16_t *data_in,uint16_t dc_est,
                            uint32_t dig_scale,uint32_t sample_len)
```

Description

This API is used process data for input to VAD.

Parameters

Parameter	Description
pVAD	Pointer to the VAD_Type structure.
vad_addr	VAD ULPSS memory address
data_in	VAD input samples
dc_est	dc estimation value.

Parameter	Description
dig_scale	Scaling the ADC output
sample_len	Number of samples to process in VAD engine.

Return values

return dc estimation value.

Example

```
int16_t ping_buffer[1023];  /* ADC output buffer */
#define VAD_DIG_SCALE              2
#define MAXIMUM_ADC_SAMPLE         1023
#define VAD_PING_ADDR              0x1000

dc_est = RSI_VAD_ProcessData(VAD,VAD_PING_ADDR,ping_buffer,dc_est,DIG_SCALE,
                                   32);
dc_est = RSI_VAD_ProcessData(VAD,VAD_PING_ADDR,ping_buffer,dc_est,DIG_SCALE,
                                   MAXIMUM_ADC_SAMPLE);
```

Note

```
After ADC interrupt this API should call and pass the ADC sample to this API,before
processing all ADC samples process fast 32 ADC samples for sample per frame detection
purpose so this API must be call 2 time as sample length 32 and sample length maximum ADC
samples.
```

39 Capacitive Touch Sensor

39.1 Overview

This section explains how to configure and use the Temperature Sensor using Redpine MCU SAPIs.

39.2 Programming Sequence

```
/* Includes ------------------------------------------------------------------*/
#include "rsi_chip.h"   /* Redpine_MCU_Vx.y.z\Host_MCU\common\chip\inc */
#include "rsi_board.h"  /* Redpine_MCU_Vx.y.z\Host_MCU\common\board\inc */
#include "rsi_cts.h"    /* Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc */
#include "rsi_opamp.h"  /* Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc */
#include "rsi_ds_timer.h"
#include "UDMA.h"


/* Private macro -------------------------------------------------------------*/
#define NVIC_DST                  NPSS_MCU_INTR_IRQn
#define PMU_GOOD_TIME        31    /*Duration in us*/
#define XTAL_GOOD_TIME       31
#define NPSS_GPIO_PIN   3

#define  UDMA1_IRQHandler  IRQ010_Handler

/* Private variables ---------------------------------------------------------*/

extern RSI_DRIVER_UDMA Driver_UDMA1;
static RSI_DRIVER_UDMA *UDMAdrv1 = &Driver_UDMA1;
extern RSI_UDMA_HANDLE_T udmaHandle1;

volatile uint32_t done,ret;

uint16_t Samples[NO_OF_SLOW_SAMPLES]  __attribute__((at(0x24063800)));
uint16_t Avg_Samples[NO_OF_SLOW_AVG_SAMPLES]  __attribute__((at(0x240638C8)));
uint8_t sensor_count[8] __attribute__((at(0x24063FD0)));
uint16_t rx_buffer[BUFFER_SIZE]  __attribute__((at(0x24063000)));

extern uint32_t Threshold;
extern uint32_t sample_count;
extern uint32_t avg_count;
extern uint32_t default_cal;
extern uint32_t calib;
extern uint32_t Ref_voltage;

uint32_t sample_sum=0,sample_avg;

uint16_t read_data_fifo[64]={0},status=0;
uint32_t i=0;
volatile uint32_t cts_done=0;

/* Private functions ---------------------------------------------------------*/
/**
 * @brief      UDMA  controller transfer descriptor chain complete callback
 * @param[in]  event dma transfer events
 * @param[in]  ch    dma channel number
 * @return     None
 */
void udmaTransferComplete(uint32_t event, uint8_t ch)
{
    if(event == UDMA_EVENT_XFER_DONE)
    {
        if(ch == 9)
        {
            done = 1;
        }
    }
```

```
}

/*udma data read from fifo*/
void UDMA_Read()
{
    RSI_UDMA_CHA_CONFIG_DATA_T control;
    RSI_UDMA_CHA_CFG_T config;

    memset(&control, 0, sizeof(RSI_UDMA_CHA_CONFIG_DATA_T));
    memset(&config, 0, sizeof(RSI_UDMA_CHA_CFG_T));

    config.altStruct = 0;
    config.burstReq  = 1;
    config.channelPrioHigh = 0;
    config.periAck = 0;
    config.periphReq = 0;
    config.reqMask = 0;
    config.dmaCh     = 9;

    /* Setup source to desination copy for trigger for memory */
    /* Descriptor configurations */
    control.transferType       = UDMA_MODE_AUTO;
    control.nextBurst                     = 0;
    control.totalNumOfDMATrans = BUFFER_SIZE -1;
    control.rPower             = ARBSIZE_2;
    control.srcProtCtrl              = 0x000;
    control.dstProtCtrl              = 0x000;
    control.srcSize            = SRC_SIZE_16;
    control.srcInc             = SRC_INC_NONE;
    control.dstSize            = DST_SIZE_16;
    control.dstInc             = DST_INC_16;

    /* Initialise dma */
    UDMAdrv1->Initialize();

    /* Configure dma channel */
    UDMAdrv1->ChannelConfigure( 9,(uint32_t )0x24042c04, (uint32_t )rx_buffer,
BUFFER_SIZE,
            control, &config, udmaTransferComplete );
    /* Enable dma channel */
    UDMAdrv1->ChannelEnable(9);
}


/*CAP SENSOR IRQhandler*/
void ULPSS_CAP_SENSOR_IRQHandler(void)
{
    /*Clear interrupt */
    RSI_CTS_IntrClear(CTS);

    RSI_NPSSGPIO_SetPin(2,1U);

    RSI_NPSSGPIO_SetPin(2,0U);

    NVIC_DisableIRQ(CAP_SENSOR_IRQn);
    /*5ms delay*/
    for(i=25000;i>0;i--);

    cts_done=1;

    return ;
}

/*CTS application function*/
void RSI_CTS()
{
    while(1)
    {
        /*enable the clock sources*/
        RSI_ULPSS_UlpPeriClkEnable(ULPCLK,TOUCH_SENSOR_PCLK_ENABLE);
```

```
RSI_ULPSS_TouchClkConfig(ULPCLK,ULP_TOUCH_32MHZ_RC_CLK,1);
/*enable static clk*/
RSI_CTS_StaticClkEnable(CTS,ENABLE);
/*clk selection for cts operation*/
RSI_CTS_ClkSelection(CTS,1,1,10,0);
/*set the fifo aempty threshold*/
RSI_CTS_ThresholdProgam(CTS,FIFO_THRESHOLD);
/*select cts mode of operation*/
RSI_CTS_ModeSelect(CTS,CONTINUOUS_MODE);
/*enables sampling mode(averaging)*/
RSI_CTS_ConfigSampleMode(CTS,ENABLE);
/*set the buffer delay*/
RSI_CTS_BufferDelay(CTS,8);
/*configure the polynomial length,seed value and polynomial value for generator*/
RSI_CTS_ConfigPolynomial(CTS,ENABLE,32,SEED_VALUE,POLYNOMIAL);
/*bypass the random no generator output*/
RSI_CTS_BypassPRS(CTS,ENABLE);
/*config on and off duration of pwm pulse*/
RSI_CTS_ConfigOnOffTime(CTS,ON_TIME,OFF_TIME);
/*configures the inter sensor delay and no of repetitions of sample*/
RSI_CTS_ConfigSampling(CTS,4,1);
/*Configure sampling pattern and valid sensor*/
RSI_CTS_ConfigSamplingPattern(CTS,SAMPLING_PATTERN,NO_OF_SENSORS);
/*config ref voltage*/
RSI_CTS_ConfigRefVoltage(CTS,Ref_voltage,ENABLE,Samples,Avg_Samples);
/*if average is enabled write 1 to wakeup register*/
RSI_CTS_ConfigWakeUp(CTS,1,Threshold);
/*if wake interrupt enabled enable the fifo afull interrupt*/
RSI_CTS_FifoInterruptEnable(CTS,ENABLE);

/*configure the cts pins*/
RSI_CTS_PinMuxing();
/*enable the cts*/
RSI_CTS_Enable(CTS,ENABLE);

/*enable NVIC for cap sensor*/
NVIC_EnableIRQ(CAP_SENSOR_IRQn);


if(calib)  //FAST CALIBRATION IF calib=1 else slow calib
{
    while(RSI_CTS_GetStatus(CTS) & FIFO_EMPTY_STATUS);

    UDMA_Read();

    UDMAdrv1->DMAEnable();

    RSI_UDMA_ChannelSoftwareTrigger(udmaHandle1,9);

    while(!done);

    done=0;

    for(i=0;i<BUFFER_SIZE;i++)
    {
        sample_sum +=(rx_buffer[i]&0x0fff);
    }

    sample_avg= (sample_sum /BUFFER_SIZE);

    RSI_CTS_Calibration(sample_avg,Samples,Avg_Samples);

    sample_sum=0;

    memset(rx_buffer,0,sizeof(rx_buffer));

}

if(calib)
{
    continue;
```

```
        }else{
            break;
        }
    }

}


/*slow calibration function*/
void RSI_SlowCalibration()
{
    uint32_t read=0;

    while(RSI_CTS_GetStatus(CTS) & FIFO_EMPTY_STATUS);

    read=RSI_CTS_ReadFifo(CTS);

    while(RSI_CTS_GetStatus(CTS) & FIFO_EMPTY_STATUS);

    for(i=0;i<BUFFER_SIZE;i++)
    {
        rx_buffer[i]=RSI_CTS_ReadFifo(CTS);
    }

    RSI_CTS_TouchDetection(rx_buffer,sensor_count);

    for(i=0;i<BUFFER_SIZE;i++)
    {
        sample_sum +=(rx_buffer[i]&0x0fff);
    }

    sample_avg= (sample_sum /BUFFER_SIZE);

    RSI_CTS_SlowCalibration(sample_avg,Samples,Avg_Samples);

    sample_sum=0;

    memset(rx_buffer,0,sizeof(rx_buffer));
}

/**
 * @brief  Main program.
 * @param  None
 * @retval None
 */
int main()
{
    /*Configures the system default clock and power configurations*/
    SystemCoreClockUpdate();
#ifdef DEBUG_UART
    DEBUGINIT();
#endif

    /*config ure the npss gpio*/
    RSI_NPSSGPIO_InputBufferEn(NPSS_GPIO_PIN , 1U);
    RSI_NPSSGPIO_SetPinMux(NPSS_GPIO_PIN , 0);
    RSI_NPSSGPIO_SetDir(NPSS_GPIO_PIN , NPSS_GPIO_DIR_OUTPUT);

    RSI_NPSSGPIO_InputBufferEn(2 , 1U);
    RSI_NPSSGPIO_SetPinMux(2 , 0);
    RSI_NPSSGPIO_SetDir(2 , NPSS_GPIO_DIR_OUTPUT);

    sample_count=0,avg_count=0,Threshold=0;
    /*set the sample and avg_samples buffer with 0*/
    memset(Samples,0,sizeof(Samples));

    memset(Avg_Samples,0,sizeof(Avg_Samples));

    memset(rx_buffer,0,sizeof(rx_buffer));
    /*Assign all the varibles with deafult value*/
    default_cal=1,Ref_voltage=4,calib=1;
```

```
    /*call the hardware setup function*/
    hardware_setup();
    /*set the duration for dst*/
    RSI_DST_IntrUnMask();
    /*enable the nvic for dst timer*/
    NVIC_EnableIRQ(NVIC_DST);
    /*Make GPIO-Low */

    while(1)
    {
        /*set integration cap to 0*/
        RSI_EGPIO_SetPinMux(EGPIO1,0,AGPIO_PIN4,0);
        RSI_EGPIO_SetDir(EGPIO1,0,AGPIO_PIN4,0);
        RSI_EGPIO_SetPin(EGPIO1,0,AGPIO_PIN4,0);

        memset(sensor_count,0,sizeof(sensor_count));
        /*Make GPIO-Low */
        RSI_NPSSGPIO_SetPin(NPSS_GPIO_PIN,1U);
        /*AUX_VDD POWER ON*/
        Aux_VddPowerOn();
        /*set the dst based wakeup source*/
        RSI_PS_SetWkpSources(ULPSS_BASED_WAKEUP);
        /*calls the cap sensor application function*/
        RSI_CTS();
        /*set the duration for dst*/
        RSI_DST_DurationSet(CTS_ON_TIME);
        /*set the dst based wakeup source*/
        RSI_PS_SetWkpSources(DST_BASED_WAKEUP);

        RSI_NPSSGPIO_SetPin(NPSS_GPIO_PIN,0U);
        /*Configure the wake up paramters for boot loader */
        RSI_PS_RetentionSleepConfig(0 , (uint32_t)RSI_PS_RestoreCpuContext, 0 , 4);
        /*Goto Sleep with retention */
        RSI_PS_EnterDeepSleep(SLEEP_WITH_RETENTION,DISABLE_LF_MODE);

        RSI_NPSSGPIO_SetPin(NPSS_GPIO_PIN,1U);
        /*slow calibration*/
        RSI_SlowCalibration();

        RSI_CTS_Enable(CTS,DISABLE);
        /*set the duration for dst*/
        RSI_DST_DurationSet(CTS_OFF_TIME);
        /*clear the ulpss based Interrupt*/
        RSI_PS_ClrWkpSources(ULPSS_BASED_WAKEUP);
        /*set the dst based wakeup*/
        RSI_PS_SetWkpSources(DST_BASED_WAKEUP);

        RSI_NPSSGPIO_SetPin(NPSS_GPIO_PIN,0U);

        RSI_PS_RetentionSleepConfig(0 , (uint32_t)RSI_PS_RestoreCpuContext, 0 , 4);
     /*Pad on control clear*/
        ULP_SPI_MEM_MAP(0x141) &= ~(BIT(11));
        /*Goto Sleep with retention */
       RSI_PS_EnterDeepSleep(SLEEP_WITH_RETENTION,DISABLE_LF_MODE);
        /*Pad on control set*/
        ULP_SPI_MEM_MAP(0x141) |= BIT(11);

    }
}
```

Code Block 28 Capacitive Touch Sensor Example

39.3 API Descriptions

RSI_CTS_ClkSelection

Source File : rsi_cts.c

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
void RSI_CTS_ClkSelection(CTS_Type *cts,uint8_t clk_sel_1,uint8_t clk_sel_2,uint8_t
clk_divider_1,uint8_t clk_divider_2)
```

Description

This API is used for clock selection.

Parameters

Parameter	Description
cts	Pointer to the Capacitive touch sensor register instance
clk_sel_1	mux select value for clk select 1
clk_sel_2	mux select value for clk select 2
clk_divider_1	division factor for clk_1
clk_divider_2	division factor for clk_2

Return values

None

Example

```
RSI_CTS_ClkSelection(CTS,1,1,5,0);
```

Note

```
clock_divider_1 value should based on cts opearating frequency
```

RSI_CTS_GetStatus

Source File : rsi_cts.c

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
uint32_t RSI_CTS_GetStatus(CTS_Type *cts)
```

Description

This API is used to get the CTS status.

Parameters

Parameter	Description
cts	Pointer to the Capacitive touch sensor register instance

Return values

Return the status of an CTS

Example

```
uint32_t status;
/* get the CTS status */
status = RSI_CTS_GetStatus(CTS);
```

RSI_CTS_ConfigPolynomial

Source File : rsi_cts.c

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
void RSI_CTS_ConfigPolynomial(CTS_Type *cts,boolean_t enable,uint8_t poly_length,uint32_t
seed,uint32_t polynomial)
```

Description

This API is used to config polynomial parameters.

Parameters

Parameter	Description
cts	Pointer to the Capacitive touch sensor register instance
enable	enable/disable seed value load • 1 -Enable seed value load • 0 -Disable seed value load
poly_length	polynomial length
seed	seed value to be loaded
polynomial	polynomial for PRS

Return values

None

Example

```
#define SEED_VALUE          0x10241024
#define POLYNOMIAL          0x68110000
#define POLYNOMIAL_LENGTH   32
RSI_CTS_ConfigPolynomial(CTS,1,POLYNOMIAL_LENGTH,SEED_VALUE,POLYNOMIAL)
```

RSI_CTS_ConfigOnOffTime

Source File : rsi_cts.c

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
void RSI_CTS_ConfigOnOffTime(CTS_Type *cts,uint16_t on_time ,uint16_t off_time)
```

Description

This API is used set on and off time for sensor.

Parameters

Parameter	Description
cts	Pointer to the Capacitive touch sensor register instance
on_time	Required on time
off_time	Required off time

Return values

None

Example

```
#define ON_TIME            256
#define OFF_TIME           4
RSI_CTS_ConfigOnOffTime(CTS,ON_TIME,OFF_TIME)
```

RSI_CTS_ConfigSampling

Source File : rsi_cts.c

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
void RSI_CTS_ConfigSampling(CTS_Type *cts,uint16_t delay,uint16_t repetitions)
```

Description

This API is used for configure the sampling delay and no of repetitions of sampling

Parameters

Parameter	Description
cts	Pointer to the Capacitive touch sensor register instance
delay	Inter sensor delay time
repetitions	No of times scan

Return values

None

Example

```
/*configures the inter sensor delay and no of repetitions of sample*/
RSI_CTS_ConfigSampling(CTS,4,1);
```

RSI_CTS_ConfigSamplingPattern

Source File : rsi_cts.c

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
void RSI_CTS_ConfigSamplingPattern(CTS_Type *cts,uint32_t pattern,uint32_t valid_sensor)
```

Description

This API is used for configure the scanning pattern and valid sensor

Parameters

Parameter	Description
cts	Pointer to the Capacitive touch sensor register instance
pattern	Sensor pattern to scan(0x12345678)
valid_sensor	No of valid sensors(1 to 8)

Return values

None

Example

```
#define SAMPLING_PATTERN  0x12345678
/*Configure sampling pattern and valid sensor*/
RSI_CTS_ConfigSamplingPattern(CTS,SAMPLING_PATTERN,8);
```

RSI_CTS_ConfigRefVoltage

Source File : rsi_cts.c

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
void RSI_CTS_ConfigRefVoltage(CTS_Type *cts,uint16_t ref_voltage,boolean_t enable,
                              uint16_t *samples, uint16_t *avg_samples)
```

Description

This API is used to configure reference voltage

Parameters

Parameter	Description
cts	Pointer to the Capacitive touch sensor register instance
ref_voltage	Reference voltage to be set • 0 -0.5V

Parameter	Description
	• 1 -0.6V • 2 -0.7V • 3 -0.8V • 4 -0.9V • 5 -1.0V • 6 -1.1V • 7 -1.2V
enable	Enable Ref voltage select
samples	pointer to array to store samples
avg_samples	pointer to array to store avg_samples

Return values

None

Example

```
uint16_t Samples[NO_OF_SLOW_SAMPLES];
uint16_t Avg_Samples[NO_OF_SLOW_AVG_SAMPLES];

RSI_CTS_ConfigRefVoltage(CTS,4,1,Samples,Avg_Samples)  /* Reference voltage set as 0.9v */
```

RSI_CTS_ConfigWakeUp

Source File : rsi_cts.c

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
void RSI_CTS_ConfigWakeUp(CTS_Type *cts,uint8_t mode,uint16_t threshold)
```

Description

This API is used to configure the wakeup mode

Parameters

Parameter	Description
cts	Pointer to the Capacitive touch sensor register instance
mode	• 1 -wakeup if count is greater than threshold • 0 -wakeup if count is less than threshold
threshold	Wakeup Threshold

Return values

None

Example

```
#define WAKEUP_THRESHOLD   150
/*configure the wakeup threshold*/
RSI_CTS_ConfigWakeUp(CTS,1,WAKEUP_THRESHOLD);
```

RSI_CTS_ReadRandomData

Source File : rsi_cts.c

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
uint32_t RSI_CTS_ReadRandomData(CTS_Type *cts)
```

Description

This API is used read the random data

Parameters

Parameter	Description
cts	Pointer to the Capacitive touch sensor register instance

Return values

Return the current status of psuedo random data

Example

```
uint32_t cts_random_data;
cts_random_data = RSI_CTS_ReadRandomData(CTS);
```

RSI_CTS_IntrClear

Source File : rsi_cts.c

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
void RSI_CTS_IntrClear(CTS_Type *cts)
```

Description

This API is used clear the interrupt for cts

Parameters

Parameter	Description
cts	Pointer to the Capacitive touch sensor register instance

Return values

None

Example

```
RSI_CTS_IntrClear(CTS)
```

RSI_CTS_BypassPRS

Source File : rsi_cts.h

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_CTS_BypassPRS(CTS_Type *cts,boolean_t enable)
```

Description

This API is used Bypass the Random number generator output to the Non-overlapping stream generator

Parameters

Parameter	Description
cts	Pointer to the Capacitive touch sensor register instance
enable	• enable = 1:bypass the random no generator • enable = 0:use random no generator output

Return values

None

Example

```
RSI_CTS_BypassPRS(CTS,1);
```

RSI_CTS_ThresholdProgam

Source File : rsi_cts.h

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_CTS_ThresholdProgam(CTS_Type *cts,uint8_t threshold)
```

Description

This API is used for threshold programming

Parameters

Parameter	Description
cts	Pointer to the Capacitive touch sensor register instance

Parameter	Description
threshold	Threshold for fifo aempty

Return values

None

Example

```
/*set the fifo aempty threshold*/
RSI_CTS_ThresholdProgam(CTS,4);
```

RSI_CTS_StaticClkEnable

Source File : rsi_cts.h

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_CTS_StaticClkEnable(CTS_Type *cts,boolean_t enable)
```

Description

This API is used for static clock gating

Parameters

Parameter	Description
cts	Pointer to the Capacitive touch sensor register instance
enable	• enable = 1:Static clock enable • enable = 0:Gated static clock

Return values

None

Example

```
RSI_CTS_StaticClkEnable(CTS,0);
```

RSI_CTS_ConfigSampleMode

Source File : rsi_cts.h

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_CTS_ConfigSampleMode(CTS_Type *cts,boolean_t avg_enable)
```

Description

This API is used for select averaging mode to apply samples

Parameters

Parameter	Description
cts	Pointer to the Capacitive touch sensor register instance
avg_enable	• 1 - for selecting averaging of samples • 0 - for directly apply samples

Return values

None

Example

```
RSI_CTS_ConfigSampleMode(CTS,1)
```

RSI_CTS_ModeSelect

Source File : rsi_cts.h

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_CTS_ModeSelect(CTS_Type *cts,boolean_t mode)
```

Description

This API is used for scanning mode selection

Parameters

Parameter	Description
cts	Pointer to the Capacitive touch sensor register instance
mode	• 1 - for continuous mode selection • 0 - for one hot mode

Return values

None

Example

```
RSI_CTS_ModeSelect(CTS,1)
```

RSI_CTS_Enable

Source File : rsi_cts.h

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_CTS_Enable(CTS_Type *cts,boolean_t enable)
```

Description

This API is used enable/disable the cts

Parameters

Parameter	Description
cts	Pointer to the Capacitive touch sensor register instance
enable	• 1 - enable the CTS • 0 - disable the CTS

Return values

None

Example

```
RSI_CTS_Enable(CTS,1)
```

RSI_CTS_ReadFifo

Source File : rsi_cts.c

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
uint32_t RSI_CTS_ReadFifo(CTS_Type *cts)
```

Description

This API is used read the data from fifo.

Parameters

Parameter	Description
cts	Pointer to cts config structure

Return values

Sensor count value

Example

```
uint16_t sensor_count;
sensor_count=RSI_CTS_ReadFifo();
```

RSI_CTS_InputSelection

Source File : rsi_cts.c

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
void RSI_CTS_InputSelection(CTS_Type *cts,uint32_t cap_input,uint32_t res_input)
```

Description

This API is used to selection of an capacitance and resistance input.

Parameters

Parameter	Description
cts	Pointer to cts config structure
cap_input	Select the Int cap input 1 -AGPIO8 0 -AGPIO4(Default)
res_input	Select the resistance input select 1 -AGPIO10 0 -AGPIO5(Default)

Return values

None

Example

```
RSI_CTS_InputSelection(CTS,0,0);
```

RSI_CTS_BufferDelay

Source File : rsi_cts.h

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_CTS_BufferDelay(CTS_Type *cts,uint8_t delay)
```

Description

This API is used set the buffer delay

Parameters

Parameter	Description
cts	Pointer to the Capacitive touch sensor register instance
delay	delay time(max value is 31)

Return values

None

Example

```
RSI_CTS_BufferDelay(CTS,1)
```

RSI_CTS_FifoInterruptEnable

Source File : rsi_cts.h

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\inc\

Prototype

```
STATIC INLINE void RSI_CTS_FifoInterruptEnable(CTS_Type *cts,boolean_t enable)
```

Description

This API is used to mask the fifo interrupt

Parameters

Parameter	Description
cts	Pointer to the Capacitive touch sensor register instance
enable	• 1 -fifo afull interrupt will mask • 0 -unmask

Return values

None

Example

```
/* Enable the CTS fifo interrupt */
RSI_CTS_FifoInterruptEnable(CTS,0)
```

RSI_CTS_Calibration

Source File : rsi_cts.c

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
void RSI_CTS_Calibration(uint16_t avg,uint16_t *samples,uint16_t *avg_samples)
```

Description

This API is used to do CTS fast calibration

Parameters

Parameter	Description
avg	Average of the samples
samples	Pointer to array to store samples

avg_samples	Pointer to array to store avg_samples

Return values

None

Example

```
uint16_t Samples[NO_OF_SLOW_SAMPLES];
uint16_t Avg_Samples[NO_OF_SLOW_AVG_SAMPLES];
uint32_t sample_Avg;
RSI_CTS_Calibration(sample_Avg,Samples,Avg_Samples);
```

RSI_CTS_SlowCalibration

Source File : rsi_cts.c

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
void RSI_CTS_SlowCalibration(uint16_t avg, uint16_t *samples, uint16_t *avg_samples)
```

Description

This API is used to do CTS calibration

Parameters

Parameter	Description
avg	Average of the samples
samples	Pointer to array to store samples
avg_samples	Pointer to array to store avg_samples

Return values

None

Example

```
uint16_t Samples[NO_OF_SLOW_SAMPLES];
uint16_t Avg_Samples[NO_OF_SLOW_AVG_SAMPLES];
uint32_t sample_Avg;
RSI_CTS_SlowCalibration(sample_Avg,Samples,Avg_Samples);
```

RSI_CTS_TouchDetection

Source File : rsi_cts.c

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
void RSI_CTS_TouchDetection(uint16_t *fifo_read,uint8_t *sensor_count)
```

Description

This API is used to check which sensor is pressed

Parameters

Parameters	Description
fifo_read	Pointer to Sensor count buffer
sensor_count	Pointer to store pressed sensor no

Return values

None

Example

```
uint16_t rx_buffer[8];    /*Arrray to store sensor count value*/
uint8_t sensor_count[8];   /*Array to store individual sensor count value*/

RSI_CTS_TouchDetection(rx_buffer,sensor_count);
```

40 Deep sleep timer

40.1 Overview

This section explains how to configure and use the Temperature Sensor using Redpine MCU SAPIs.

40.2 Programming sequence

```
#include "rsi_chip.h"  /* Redpine_MCU_Vx.y.z\Host_MCU\common\chip\inc */

/*Example to use Deep sleep timer as a wake up source in PS2 state */
int main()
{

    /*Configures the system default clock and power configurations*/
    SystemCoreClockUpdate();

    /*Configure the NPSS-GPIO for sleep wake up indication*/
    RSI_NPSSGPIO_InputBufferEn(NPSS_GPIO_PIN , 1U);
    RSI_NPSSGPIO_SetPinMux(NPSS_GPIO_PIN , 0);
    RSI_NPSSGPIO_SetDir(NPSS_GPIO_PIN , NPSS_GPIO_DIR_OUTPUT);

    /*Change the TASS reference clock to 32MHz RC clock
     NOTE: This should not be used in WiSeMCU mode , should be used in MCU only mode
    */
    RSI_ChangeTassRefClock();

    RSI_CLK_PeripheralClkDisable3(M4CLK , M4_SOC_CLK_FOR_OTHER_ENABLE); /* Disable
OTHER_CLK which is enabled at Start-up */
    RSI_ULPSS_TimerClkDisable( ULPCLK ); /* Disable Timer clock which is enabled in
Bootloader */

    RSI_ULPSS_DisableRefClks( MCU_ULP_40MHZ_CLK_EN );             /* Disabling 40MHz
Clocks */
    RSI_ULPSS_DisableRefClks( MCU_ULP_32KHZ_RC_CLK_EN );        /* Disabling LF_RC Clocks
*/

    RSI_PS_BodPwrGateButtonCalibDisable();    /* Power-Down Button Calibration */
    RSI_PS_AnalogPeriPtatDisable();                    /* Disable PTAT for Analog
Peripherals */
    RSI_PS_BodClksPtatDisable();                        /* Disable PTAT for Brown-Out
Detection Clocks */

    /* Power-Down Analog Peripherals */
    RSI_IPMU_PowerGateClr(
                AUXDAC_PG_ENB
            |    AUXADC_PG_ENB
            |    WURX_CORR_PG_ENB
            |    WURX_PG_ENB
            | ULP_ANG_CLKS_PG_ENB
            |    CMP_NPSS_PG_ENB
    );

    /* Power-Down Domains in NPSS */
    RSI_PS_NpssPeriPowerDown(
                SLPSS_PWRGATE_ULP_MCUWDT
            |    SLPSS_PWRGATE_ULP_MCUPS
            |    SLPSS_PWRGATE_ULP_MCUTS
            |    SLPSS_PWRGATE_ULP_MCUSTORE2
            |    SLPSS_PWRGATE_ULP_MCUSTORE3
    );

    RSI_PS_PowerSupplyDisable(
              POWER_ENABLE_TIMESTAPING);

    RSI_PS_SocPllSpiDisable();            /* Power-Down High-Frequency PLL Domain */
    RSI_PS_QspiDllDomainDisable();    /* Power-Down QSPI-DLL Domain */
```

```
    RSI_PS_XtalDisable();                        /* Power-Down XTAL-OSC Domain */

    RSI_PS_WirelessShutdown();             /* Shutdown Wireless since Wireless Processor
is not present */


    /* TO trim RCMhz clock to 20Mhz*/
    RSI_IPMU_M20rcOsc_TrimEfuse();
    /* Change ULPSS-REF and M4SS-REF and NPSS-HF_FSM clocks to 20MHz RO to be used in PS2
state */
    RSI_ULPSS_EnableRefClks( MCU_ULP_32MHZ_RC_CLK_EN, ULP_PERIPHERAL_CLK, 0 );
/* Enable HF-RO Clock */
    RSI_ULPSS_RefClkConfig( ULPSS_ULP_32MHZ_RC_CLK ); /* Configuring ULPSS Reference Clock
to HF-RO */
    RSI_CLK_M4ssRefClkConfig( M4CLK, ULP_32MHZ_RC_CLK );   /* Configuring ULPSS Reference
Clock to HF-RO */

    /* Switching from PS4 to PS2 state */
     RSI_PS_PowerStateChangePs4toPs2(ULP_MCU_MODE ,
            PWR_MUX_SEL_ULPSSRAM_SCDC_0_9        ,
            PWR_MUX_SEL_M4_ULP_RAM_SCDC_0_9      ,
            PWR_MUX_SEL_M4_ULP_RAM16K_SCDC_0_9   ,
            PWR_MUX_SEL_M4ULP_SCDC_0_6           ,
            PWR_MUX_SEL_ULPSS_SCDC_0_9           ,
            DISABLE_BG_SAMPLE_ENABLE             ,
            DISABLE_DC_DC_ENABLE                 ,
            DISABLE_SOCLDO_ENABLE                ,
            DISABLE_STANDBYDC
    );

    RSI_PS_EnableFirstBootUp(1);    /* Enable first boot up */
    RSI_PS_SkipXtalWaitTime(1);        /* XTAL wait time is skipped since RC_32MHZ Clock
is used for Processor on wakeup */

    RSI_PS_SetRamRetention(    M4ULP_RAM16K_RETENTION_MODE_EN ); /* Enable SRAM Retention
of 16KB during Sleep */

    RSI_TIMEPERIOD_TimerClkSel(TIME_PERIOD , 0x003E7FFF);
    /*Deep sleep duration count 7500us*/
    RSI_DST_DurationSet(DS_TIMER_DURATION);
    /* Interrupt unmask */
    RSI_DST_IntrUnMask();
    /*Set wake source as the Alarm interrupt from NPSS */
    RSI_PS_SetWkpSources(DST_BASED_WAKEUP);
    NVIC_EnableIRQ(NVIC_DS_TIMER);
    while(1)
    {
      /*Make GPIO-Low */
      RSI_NPSSGPIO_SetPin(NPSS_GPIO_PIN,0U);

      RSI_PS_FsmHfClkSel(FSM_32MHZ_RC);        /* Configuring HF-RC Clock for HF-FSM */
      /*Configure the wake up paramters for boot loader */
      RSI_PS_RetentionSleepConfig(0 , (uint32_t)RSI_PS_RestoreCpuContext, 0 , 4);
      /*Goto Sleep with retention */
      RSI_PS_EnterDeepSleep(SLEEP_WITH_RETENTION,DISABLE_LF_MODE);
      /*Up on wake up execution resumes from here*/
      RSI_PS_FsmHfClkSel(FSM_20MHZ_RO);        /* Configuring HF-RC Clock for HF-FSM */

      /*Make GPIO-High */
      RSI_NPSSGPIO_SetPin(NPSS_GPIO_PIN,1U);
    }
}
```

Code Block 29 Power save Example

40.3 API Description

40.3.1 RSI_DST_DurationSet

Source File : rsi_ds_timer.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\inc\

Prototype

```
STATIC INLINE void RSI_DST_DurationSet(uint32_t dsDuration)
```

Description

This API is used to configure the deep sleep duration of sleep timer.

Parameters

Parameters	Description
dsDuration	sleep duration count in micro seconds

Return values

None

Example

```
#define DS_TIMER_DURATION_TIME_US    1000000  /* sleep time set as 1 second, Every 1
second M4 is wake up by DS timer interrupt */
/* start RNG */
RSI_DST_DurationSet(DS_TIMER_DURATION_TIME_US);
```

40.3.2 RSI_DST_IntrUnMask

Source File : rsi_ds_timer.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\inc\

Prototype

```
STATIC INLINE void RSI_DST_IntrUnMask(void)
```

Description

This API is used to enable deep sleep timer interrupt .

Parameters

None

Return values

None

Example

```
/* Unmask the DS timer interrupt */
RSI_DST_IntrUnMask();
```

40.3.3 RSI_DST_IntrMask

Source File : rsi_ds_timer.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\inc\

Prototype

```
STATIC INLINE void RSI_DST_IntrMask(void)
```

Description

This API is used to disable deep sleep timer interrupt .

Parameters

None

Return values

None

Example

```
/* Unmask the DS timer interrupt */
RSI_DST_IntrMask();
```

40.3.4 RSI_DST_TimerIntrClear

Source File : rsi_ds_timer.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\inc\

Prototype

```
STATIC INLINE void RSI_DST_TimerIntrClear(void)
```

Description

This API is used to clear deep sleep timer interrupt .

Parameters

None

Return values

None

Example

```
/* Unmask the DS timer interrupt */
RSI_DST_TimerIntrClear();
```

41 NPSS GPIOs

41.1 Overview

This section explains how to configure and use the NPSS gpios using Redpine MCU SAPIs.

41.2 Programming sequence

```
#include "rsi_chip.h"  /* Redpine_MCU_Vx.y.z\Host_MCU\common\chip\inc */

#define NPSS_GPIO_IRQ_Handler   IRQ021_Handler      /*<! NPSS GPIO IRQ Handler */
#define NPSS_GPIO_PIN       3
#define PMU_GOOD_TIME           31    /*Duration in us*/
#define XTAL_GOOD_TIME          31    /*Duration in us*/
#define MHZ_CLOCK               32

void NPSS_GPIO_IRQ_Handler(void)
{
    volatile    uint16_t wakeUpSrc=0;
    wakeUpSrc = RSI_PS_GetWkpUpStatus();
    if(wakeUpSrc & NPSS_TO_MCU_GPIO_INTR_2){
    RSI_PS_ClrWkpUpStatus (NPSS_TO_MCU_GPIO_INTR_2);
    }
    return ;
}


/*Application starts here*/
int main()
{
    /*Configures the system default clock and power configurations*/
    SystemCoreClockUpdate();

    /*Configure the NPSS GPIO for sleep wake up indication*/
    RSI_NPSSGPIO_InputBufferEn(NPSS_GPIO_PIN , 1U);
    RSI_NPSSGPIO_SetPinMux(NPSS_GPIO_PIN , 0);
    RSI_NPSSGPIO_SetDir(NPSS_GPIO_PIN , NPSS_GPIO_DIR_OUTPUT);

    /*Change the TASS reference clock to 32MHz RC clock
    NOTE: This should not be used in WiSeMCU mode , should be used in MCU only mode
    */
    RSI_ChangeTassRefClock();

    RSI_IPMU_ClockMuxSel(1);                        /* Configuring RO-32KHz Clock for
BG_PMU */
    RSI_PS_FsmLfClkSel(KHZ_RC_CLK_SEL);    /* Configuring RO-32KHz Clock for LF-FSM */
    RSI_PS_FsmHfClkSel(FSM_32MHZ_RC);        /* Configuring RC-32MHz Clock for HF-FSM */

    RSI_CLK_PeripheralClkDisable3(M4CLK , M4_SOC_CLK_FOR_OTHER_ENABLE); /* Disable
OTHER_CLK which is enabled at Start-up */
    RSI_ULPSS_TimerClkDisable( ULPCLK ); /* Disable Timer clock which is enabled in
Bootloader */

    RSI_ULPSS_DisableRefClks( MCU_ULP_40MHZ_CLK_EN );             /* Disabling 40MHz
Clocks */
    RSI_ULPSS_DisableRefClks( MCU_ULP_32KHZ_RC_CLK_EN );        /* Disabling LF_RC Clocks
*/

    RSI_PS_BodPwrGateButtonCalibDisable();    /* Power-Down Button Calibration */
    RSI_PS_AnalogPeriPtatDisable();                    /* Disable PTAT for Analog
Peripherals */
    RSI_PS_BodClksPtatDisable();                        /* Disable PTAT for Brown-Out
Detection Clocks */

    /* Power-Down Analog Peripherals */
    RSI_IPMU_PowerGateClr(
                AUXDAC_PG_ENB
```

```
            |    AUXADC_PG_ENB
            |    WURX_CORR_PG_ENB
            |    WURX_PG_ENB
            |    ULP_ANG_PWRSUPPLY_PG_ENB
            | ULP_ANG_CLKS_PG_ENB
            |    CMP_NPSS_PG_ENB
    );

    /* Power-Down Domains in NPSS VBAT */
    RSI_PS_NpssPeriPowerDown(
                SLPSS_PWRGATE_ULP_MCUWDT
            |    SLPSS_PWRGATE_ULP_MCUPS
            |    SLPSS_PWRGATE_ULP_MCUTS
            |    SLPSS_PWRGATE_ULP_MCUSTORE2
            |    SLPSS_PWRGATE_ULP_MCUSTORE3
    );



    /*M4SS peripheral power gates */
    RSI_PS_M4ssPeriPowerDown(
            M4SS_PWRGATE_ULP_M4_FPU   |
            M4SS_PWRGATE_ULP_ETHERNET |
            M4SS_PWRGATE_ULP_SDIO_SPI |
            M4SS_PWRGATE_ULP_USB      |
            M4SS_PWRGATE_ULP_RPDMA    |
            M4SS_PWRGATE_ULP_PERI1    |
            M4SS_PWRGATE_ULP_PERI2    |
            M4SS_PWRGATE_ULP_PERI3    |
            M4SS_PWRGATE_ULP_CCI      |
            M4SS_PWRGATE_ULP_SD_MEM);


    RSI_PS_PowerSupplyDisable(
              POWER_ENABLE_TIMESTAPING
            | POWER_ENABLE_DEEPSLEEP_TIMER
    );

    RSI_PS_SocPllSpiDisable();            /* Power-Down High-Frequency PLL Domain */
    RSI_PS_QspiDllDomainDisable();    /* Power-Down QSPI-DLL Domain */
    RSI_PS_XtalDisable();                        /* Power-Down XTAL-OSC Domain */

    RSI_PS_WirelessShutdown();             /* Shutdown Wireless since Wireless Processor
is not present */


    RSI_ULPSS_RefClkConfig( ULPSS_ULP_32MHZ_RC_CLK ); /* Configuring ULPSS Reference Clock
to HF-RO */
    RSI_CLK_M4ssRefClkConfig( M4CLK, ULP_32MHZ_RC_CLK ); /* Configuring ULPSS Reference
Clock to HF-RO */

    RSI_ULPSS_DisableRefClks(    MCU_ULP_32MHZ_RC_CLK_EN );        /* Disabling HF_RC
Clock */

    RSI_ConfigBuckBoost(0,1); /* BuckBoost is configured to ON state bypassing the H/W
Control */

    RSI_PS_EnableFirstBootUp(1);    /* Enable first boot up */
    RSI_PS_SkipXtalWaitTime(1);        /* XTAL wait time is skipped since RC_32MHZ Clock
is used for Processor on wakeup */

    RSI_PS_SetRamRetention(    M4ULP_RAM16K_RETENTION_MODE_EN ); /* Enable SRAM Retention
of 16KB during Sleep */

    /* Configure NPSS VBAT-HF_FSM Clock to 32MHz for Sleep/Wakeup Routine */
    RSI_ULPSS_EnableRefClks( MCU_ULP_32MHZ_RC_CLK_EN, ULP_PERIPHERAL_CLK, 0 );
/* Enable HF-RC Clock */
    RSI_PS_FsmHfClkSel(FSM_32MHZ_RC);        /* Configuring HF-RC Clock for HF-FSM */
    RSI_PS_FsmHfFreqConfig(MHZ_CLOCK);                    /* Configuring HF-FSM Clock
Frequency as 32MHz */
    RSI_TIMEPERIOD_TimerClkSel(TIME_PERIOD , 0x003E7FFF);
```

```
        /*GPIO based wake up */
    /*Configure the NPSS GPIO mode to wake up  */
    RSI_NPSSGPIO_SetPinMux(NPSS_GPIO_2,NPSSGPIO_PIN_MUX_MODE2);

    /*Configure the NPSS GPIO direction to input */
    RSI_NPSSGPIO_SetDir(NPSS_GPIO_2 , NPSS_GPIO_DIR_INPUT);

    /*Configure the NPSS GPIO interrupt polarity */
    RSI_NPSSGPIO_SetPolarity(NPSS_GPIO_2 ,  NPSS_GPIO_INTR_HIGH);

    /*Enable the REN*/
    RSI_NPSSGPIO_InputBufferEn(NPSS_GPIO_2 ,  1);

    RSI_NPSSGPIO_SetWkpGpio(NPSS_GPIO_2_INTR);

    RSI_NPSSGPIO_IntrUnMask(NPSS_GPIO_2_INTR);

    /*Select wake up sources */
    RSI_PS_SetWkpSources(GPIO_BASED_WAKEUP);

    /*Enable the NPSS GPIO interrupt slot*/
    NVIC_EnableIRQ(NPSS_TO_MCU_GPIO_INTR_IRQn);

    while(1)
    {
        /*Make GPIO-Low */
        RSI_NPSSGPIO_SetPin(NPSS_GPIO_PIN,0U);

        /*Configure the wake up paramters for boot loader */
        RSI_PS_RetentionSleepConfig(0 , (uint32_t)RSI_PS_RestoreCpuContext, 0 , 4);
        /*Goto Sleep with retention */
        RSI_PS_EnterDeepSleep(SLEEP_WITH_RETENTION,DISABLE_LF_MODE);

        /*Up on wake up execution resumes from here*/
        RSI_NPSSGPIO_SetPin(NPSS_GPIO_PIN,1U);
 }
}
```

Code Block 30 UULP VBAT GPIO Usage

41.3 API Description

41.3.1 RSI_NPSSGPIO_SetPinMux

Source File : rsi_retention.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\inc\

Prototype

```
STATIC INLINE void RSI_NPSSGPIO_SetPinMux(uint8_t pin, uint8_t mux)
```

Description

This API is used to set the NPSS GPIO pin MUX (mode)

Parameters

Parameters	Description
pin	NPSS GPIO pin number (0...4)
mux	NPSS GPIO MUX value.

Return values

None

Example

```
#define NPSS_GPIO_PIN       3
/* NPSS_GPIO_PIN3 will configure */
RSI_NPSSGPIO_SetPinMux(NPSS_GPIO_PIN , 0);
```

41.3.2 RSI_NPSSGPIO_InputBufferEn

Source File : rsi_retention.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\inc\

Prototype

```
STATIC INLINE void RSI_NPSSGPIO_InputBufferEn(uint8_t pin , boolean_t enable)
```

Description

This API is used to enable/disable NPSS GPIO input buffer.

Parameters

Parameters	Description
pin	NPSS GPIO pin number (0...4)
enable	Enable / Disable NPSS GPIO input buffer • 1 : Enable • 0 : Disable

Return values

None

Example

```
#define  NPSS_GPIO_PIN       3
/* Input buffer enable for NPSS_GPIO_PIN3 */
RSI_NPSSGPIO_InputBufferEn(NPSS_GPIO_PIN , 1U);
```

41.3.3 RSI_NPSSGPIO_SetDir

Source File : rsi_retention.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\inc\

Prototype

```
STATIC INLINE void RSI_NPSSGPIO_SetDir(uint8_t pin, boolean_t dir)
```

Description

This API is used to set the direction of the NPSS GPIO.

Parameters

Parameters	Description
pin	NPSS GPIO pin number (0...4)

Parameters	Description
dir	Direction value (Input / Output) • 1 : Input Direction • 0 : Output Direction

Return values

None

Example

```
#define NPSS_GPIO_PIN       3
/* Set output direction for NPSS_GPIO_PIN3 */
RSI_NPSSGPIO_SetDir(NPSS_GPIO_PIN , NPSS_GPIO_DIR_OUTPUT);
```

41.3.4 RSI_NPSSGPIO_GetDir

Source File : rsi_retention.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\inc\

Prototype

```
STATIC INLINE boolean_t RSI_NPSSGPIO_GetDir(uint8_t pin)
```

Description

This API is used to Get the direction of the NPSS GPIO.

Parameters

Parameters	Description
pin	NPSS GPIO pin number (0...4)

Return values

Returns the GPIO pin direction

Example

```
#define NPSS_GPIO_PIN       3
boolean_t direction;
/* Get direction of NPSS_GPIO_PIN3 */
direction = RSI_NPSSGPIO_GetDir(NPSS_GPIO_PIN);
```

41.3.5 RSI_NPSSGPIO_SetPin

Source File : rsi_retention.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\inc\

Prototype

```
STATIC INLINE void RSI_NPSSGPIO_SetPin(uint8_t pin, boolean_t val)
```

Description

This API is used to set the NPSS GPIO pin value.

Parameters

Parameters	Description
pin	NPSS GPIO pin number (0...4)
val	NPSS GPIO pin value 0 or 1

Return values

None

Example

```
#define NPSS_GPIO_PIN       3
/* Set pin status of 1 for NPSS GPIO 3 */
RSI_NPSSGPIO_SetPin(NPSS_GPIO_PIN,1U);
```

41.3.6 RSI_NPSSGPIO_GetPin

Source File : rsi_retention.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\inc\

Prototype

```
STATIC INLINE boolean_t RSI_NPSSGPIO_GetPin(uint8_t pin)
```

Description

This API is used to Get the NPSS GPIO pin value

Parameters

Parameters	Description
pin	NPSS GPIO pin number (0...4)
val	NPSS GPIO pin value 0 or 1

Return values

None

Example

```
#define NPSS_GPIO_PIN       3
boolean_t_t direction;
/* Set pin status of 1 for NPSS_GPIO_PIN3 */
direction = RSI_NPSSGPIO_GetPin(NPSS_GPIO_PIN);
```

41.3.7 RSI_NPSSGPIO_SetPolarity

Source File : rsi_retention.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\inc\

Prototype

```
STATIC INLINE void RSI_NPSSGPIO_SetPolarity(uint8_t pin , boolean_t level)
```

Description

This API is used to select polarity NPSS GPIO wake up detection when in sleep

Parameters

Parameters	Description
pin	NPSS GPIO pin number (0...4)
level	Set the level for wake up • 1 : high level • 0 : low level

Return values

None

Example

```
#define NPSS_GPIO_PIN       3
RSI_NPSSGPIO_SetPolarity(NPSS_GPIO_PIN,1U);
```

41.3.8 RSI_NPSSGPIO_SetWkpGpio

Source File : rsi_retention.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\inc\

Prototype

```
STATIC INLINE void RSI_NPSSGPIO_SetWkpGpio(uint8_t npssGpioPinIntr)
```

Description

This API is used to set the NPSS GPIO to wake from deep sleep

Parameters

Parameters	Description
npssGpioPinIntr	OR'ed values of the NPSS GPIO interrupts

Return values

None

Example

```
RSI_NPSSGPIO_SetWkpGpio(NPSS_GPIO_2_INTR);
```

41.3.9 RSI_NPSSGPIO_ClrWkpGpio

Source File : rsi_retention.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\inc\

Prototype

```
STATIC INLINE void RSI_NPSSGPIO_ClrWkpGpio(uint8_t npssGpioPinIntr)
```

Description

This API is used to clear the NPSS GPIO to wake from deep sleep

Parameters

Parameters	Description
npssGpioPinIntr	OR'ed values of the NPSS GPIO interrupts

Return values

None

Example

```
RSI_NPSSGPIO_ClrWkpGpio(NPSS_GPIO_2_INTR);
```

41.3.10 RSI_NPSSGPIO_IntrMask

Source File : rsi_retention.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\inc\

Prototype

```
STATIC INLINE void RSI_NPSSGPIO_IntrMask(uint8_t npssGpioPinIntr)
```

Description

This API is used to mask the NPSS GPIO interrupt.

Parameters

Parameters	Description
npssGpioPinIntr	OR'ed values of the NPSS GPIO interrupts

Return values

None

Example

```
RSI_NPSSGPIO_IntrMask(NPSS_GPIO_2_INTR);
```

41.3.11 RSI_NPSSGPIO_IntrUnMask

Source File : rsi_retention.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\inc\

Prototype

```
STATIC INLINE void RSI_NPSSGPIO_IntrUnMask(uint8_t npssGpioPinIntr)
```

Description

This API is used to un mask the NPSS GPIO interrupt

Parameters

Parameters	Description
npssGpioPinIntr	OR'ed values of the NPSS GPIO interrupts

Return values

None

Example

```
RSI_NPSSGPIO_IntrUnMask(NPSS_GPIO_2_INTR);
```

41.3.12 RSI_NPSSGPIO_SetIntFallEdgeEnable

Source File : rsi_retention.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\inc\

Prototype

```
STATIC INLINE void RSI_NPSSGPIO_SetIntFallEdgeEnable(uint8_t npssGpioPinIntr)
```

Description

This API is used to Set the fall edge interrupt detection for NPSS GPIO

Parameters

Parameters	Description
npssGpioPinIntr	OR'ed values of the NPSS GPIO interrupts

Return values

None

Example

```
RSI_NPSSGPIO_SetIntFallEdgeEnable(NPSS_GPIO_2_INTR);
```

41.3.13 RSI_NPSSGPIO_ClrIntFallEdgeEnable

Source File : rsi_retention.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\inc\

Prototype

```
STATIC INLINE void RSI_NPSSGPIO_ClrIntFallEdgeEnable(uint8_t npssGpioPinIntr)
```

Description

This API is used to clear the fall edge interrupt detection for NPSS GPIO

Parameters

Parameters	Description
npssGpioPinIntr	OR'ed values of the NPSS GPIO interrupts

Return values

None

Example

```
RSI_NPSSGPIO_ClrIntFallEdgeEnable(NPSS_GPIO_2_INTR);
```

41.3.14 RSI_NPSSGPIO_SetIntRiseEdgeEnable

Source File : rsi_retention.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\inc\

Prototype

```
STATIC INLINE void RSI_NPSSGPIO_SetIntRiseEdgeEnable(uint8_t npssGpioPinIntr)
```

Description

This API is used to Set the rise edge interrupt detection for NPSS GPIO.

Parameters

Parameters	Description
npssGpioPinIntr	OR'ed values of the NPSS GPIO interrupts

Return values

None

Example

```
RSI_NPSSGPIO_SetIntRiseEdgeEnable(NPSS_GPIO_2);
```

41.3.15 RSI_NPSSGPIO_ClrIntRiseEdgeEnable

Source File : rsi_retention.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\inc\

Prototype

```
STATIC INLINE void RSI_NPSSGPIO_ClrIntRiseEdgeEnable(uint8_t npssGpioPinIntr)
```

Description

This API is used to clear rise edge interrupt detection for NPSS GPIO

Parameters

Parameters	Description
npssGpioPinIntr	OR'ed values of the NPSS GPIO interrupts

Return values

None

Example

```
RSI_NPSSGPIO_ClrIntRiseEdgeEnable(NPSS_GPIO_2);
```

41.3.16 RSI_NPSSGPIO_SetIntLevelHighEnable

Source File : rsi_retention.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\inc\

Prototype

```
STATIC INLINE void RSI_NPSSGPIO_SetIntLevelHighEnable(uint8_t npssGpioPinIntr)
```

Description

This API is used to enable level high detection interrupt.

Parameters

Parameters	Description
npssGpioPinIntr	OR'ed values of the NPSS GPIO interrupts

Return values

None

Example

```
RSI_NPSSGPIO_SetIntLevelHighEnable(NPSS_GPIO_2);
```

41.3.17 RSI_NPSSGPIO_ClrIntLevelHighEnable

Source File : rsi_retention.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\inc\

Prototype

```
STATIC INLINE void RSI_NPSSGPIO_ClrIntLevelHighEnable(uint8_t npssGpioPinIntr)
```

Description

This API is used to clear the level high interrupt detection for NPSS GPIO.

Parameters

Parameters	Description
npssGpioPinIntr	OR'ed values of the NPSS GPIO interrupts

Return values

None

Example

```
RSI_NPSSGPIO_ClrIntLevelHighEnable(NPSS_GPIO_2);
```

41.3.18 RSI_NPSSGPIO_SetIntLevelLowEnable

Source File : rsi_retention.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\inc\

Prototype

```
STATIC INLINE void RSI_NPSSGPIO_SetIntLevelLowEnable(uint8_t npssGpioPinIntr)
```

Description

This API is used to set the level low interrupt detection for NPSS GPIO.

Parameters

Parameters	Description
npssGpioPinIntr	OR'ed values of the NPSS GPIO interrupts

Return values

None

Example

```
RSI_NPSSGPIO_SetIntLevelLowEnable(NPSS_GPIO_2);
```

41.3.19 RSI_NPSSGPIO_ClrIntLevelLowEnable

Source File : rsi_retention.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\inc\

Prototype

```
STATIC INLINE void RSI_NPSSGPIO_ClrIntLevelLowEnable(uint8_t npssGpioPinIntr)
```

Description

This API is used to clear the level low interrupt detection for NPSS GPIO.

Parameters

Parameters	Description
npssGpioPinIntr	OR'ed values of the NPSS GPIO interrupts

Return values

None

Example

```
RSI_NPSSGPIO_ClrIntLevelLowEnable(NPSS_GPIO_2);
```

41.3.20 RSI_NPSSGPIO_ClrIntr

Source File : rsi_retention.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\inc\

Prototype

```
STATIC INLINE void RSI_NPSSGPIO_ClrIntr(uint8_t npssGpioPinIntr)
```

Description

This API is used to clear NPSS GPIO interrupt.

Parameters

Parameters	Description
npssGpioPinIntr	OR'ed values of the NPSS GPIO interrupts

Return values

None

Example

```
RSI_NPSSGPIO_ClrIntr(NPSS_GPIO_2);
```

41.3.21 RSI_NPSSGPIO_GetIntrStatus

Source File : rsi_retention.h

Path : Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\systemlevel\inc\

Prototype

```
STATIC INLINE uint8_t RSI_NPSSGPIO_GetIntrStatus(void)
```

Description

This API is used to get the NPSS GPIO interrupt status.

Parameters

None

Return values

Returns the GPIO status

Example

```
uint8_t status;
status=RSI_NPSSGPIO_GetIntrStatus();
```

42 Redpine MCU SAPI Manual Revision History

Revision No.	Version No.	Date	Changes
1	v1.0	November 2017	Advance Version
2	v1.1	August 2018	Edits
3	v1.2	September 2018	1. Added maximum division vale for ULP Subsystem (ULPSS) Clock 2. Added new programming sequence for NPSS GPIOs
4	v1.3	September 2018	Added new examples
5	v1.4	October 2018	Added new system variable
6.	v1.5	December 2018	Added new examples

43 A2DP source application

44 Operational Amplifier

44.1 Overview

This section explains how to configure and use the Operational amplifier using Redpine MCU SAPIs.

44.2 Programming Sequence

```
/* Includes ------------------------------------------------------------------*/
#include "rsi_chip.h"


/* Private macro -------------------------------------------------------------*/

/*DAC */
#define CLK_DIV                        16
#define SAMPLE_LENGTH                     1
#define STATIC_MODE                    1
#define OUTPUT_MUX_ENABLE              1
#define OUTPUT_MUX_SEL                 0
#define PRBS_SEL                       0          /* Switch off PRBS for DAC */
#define DAC_ENABLE                     0

#define OPAMP1_PINMUX                  1
#define OPAMP2_PINMUX                  0
#define OPAMP3_PINMUX                  0
#define OPAMP_OUT_MUX_SEL              0
#define OPAMP_DYN_MODE_EN              0    /*OPAMP Dynamic mode*/
#define CHANNEL_NO                     0    /*adc channel no in dynamic mode*/
#define OPAMP_LP_MODE                  0    /*OPAMP low power enable*/
#define OPAMP_GPIO                     1

/* Private variables ---------------------------------------------------------*/
/* Standard sin wave samples */
uint16_t input_sample_data[SAMPLE_LENGTH] = {0x200};



/* Private functions ---------------------------------------------------------*/
/**
 * @brief      Configures the opamp related pins
 * @return     None
 */
void OPAMP_Pinmuxing()
{

#if OPAMP1_PINMUX
    RSI_EGPIO_SetPinMux(EGPIO,0,OPAMP1_N_SEL_PIN,AGPIO_MODE);
    RSI_EGPIO_SetPinMux(EGPIO1,0,OPAMP1_P_SEL_PIN,AGPIO_MODE);
    RSI_EGPIO_SetPinMux(EGPIO1,0,OPAMP1_OUT,AGPIO_MODE);
#endif

#if    OPAMP2_PINMUX
    RSI_EGPIO_SetPinMux(EGPIO1,0,OPAMP2_P_SEL_PIN,AGPIO_MODE);
    RSI_EGPIO_SetPinMux(EGPIO1,0,OPAMP2_N_SEL_PIN,AGPIO_MODE);
    RSI_EGPIO_SetPinMux(EGPIO1,0,OPAMP2_OUT,AGPIO_MODE);
#endif

#if OPAMP3_PINMUX
    /* Pad receive Disable*/
    RSI_EGPIO_PadReceiverDisable(OPAMP3_OUT);
    RSI_EGPIO_HostPadsGpioModeEnable(OPAMP3_OUT);
    egpio_pad_sdio_connected();
    RSI_EGPIO_SetPinMux(EGPIO1,0,OPAMP3_P_SEL_PIN,AGPIO_MODE);
    RSI_EGPIO_SetPinMux(EGPIO1,0,OPAMP3_OUT,TGPIO_MODE);
#endif
}
```

```
/**
 * @brief  Main program.
 * @param  None
 * @retval None
 */
int main()
{
    /*Configures the system default clock and power configurations*/
    SystemCoreClockUpdate();

    /* Configure an AUX clock source */
    RSI_ULPSS_AuxClkConfig(ULPCLK, ENABLE_STATIC_CLK ,ULP_AUX_32MHZ_RC_CLK);

    ULP_SPI_MEM_MAP(BG_SCDC_PROG_REG_1)|=BIT(3);
    /*  Configure the reference LDO voltage as 2.48v */
    RSI_ADC_AUXLdoConfig(AUX_ADC_DAC_COMP,0,0xb);

#if DAC_ENABLE
    /* Divide DAC clock */
    RSI_DAC_ClkDivFactor(AUX_ADC_DAC_COMP, CLK_DIV);

    /* Configure the DAC in static mode */

RSI_DAC_Config(AUX_ADC_DAC_COMP,STATIC_MODE,OUTPUT_MUX_ENABLE,OUTPUT_MUX_SEL,PRBS_SEL);

    RSI_DAC_Start(AUX_ADC_DAC_COMP,ENABLE);

    /* Write data in DAC Input register */
    RSI_DAC_WriteData(AUX_ADC_DAC_COMP,input_sample_data,1,1);
#endif

    OPAMP_Pinmuxing();


RSI_OPAMP1_UGB(OPAMP_GPIO,ENABLE,OPAMP_LP_MODE,OUTPUT_MUX_ENABLE,OPAMP_OUT_MUX_SEL,OPAMP_D
YN_MODE_EN,CHANNEL_NO);

    while(1);
}
```

Code Block 31 OPAMP example

44.3 API Descriptions

RSI_OPAMP1_UGB

Source File :rsi_opamp.c

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
void RSI_OPAMP1_UGB(uint8_t vin_p_sel,uint8_t enable,uint8_t lp_mode,uint8_t
out_mux_en,uint8_t out_mux_sel,
        uint8_t dyn_en,uint8_t channel)
```

Description

This API is used to configure the opamp1 in Unity gain buffer mode.

Parameters

Parameter	Description
vin_p_sel	input gpio selection 0 - ULP_GPIO_12 1 - ULP_GPIO_7 2 - ULP_GPIO_0 3 -ULP_GPIO_2 4 -ULP_GPIO_6 5 -ULP_GPIO_8 6 - AUX_DAC_OUT
enable	opamp1 enable 1 -enable 0 -disable
lp_mode	opamp1 low power mode enable 1 -enable 0 -disable
out_mux_en	opamp1 output mux enable 1 - enable 0 - disable
out_mux_sel	opamp1 output mux selection 0 - ULP_GPIO_4 1 - ULP_GPIO_15
dyn_en	opamp1 dynamic mode enable 1 - enable 0 - disable
channel	channel no to set in dynamic mode

Return values

None

Example

```
RSI_OPAMP1_UGB(1,1,0,1,0,0,0);
```

RSI_OPAMP2_UGB

Source File :rsi_opamp.c

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
void RSI_OPAMP2_UGB(uint8_t vin_p_sel,uint8_t enable,uint8_t lp_mode,uint8_t
out_mux_en,uint8_t dyn_en,uint8_t channel)
```

Description

This API is used to configure the opamp2 in Unity gain buffer mode.

Parameters

Parameter	Description
vin_p_sel	input selection 0 - ULP_GPIO_11 1 - ULP_GPIO_5 3 - AUX_DAC_OUT 6 - OPAMP1_OUT)
enable	opamp2 enable 1 -enable 0 -disable
lp_mode	opamp2 low power mode 1 - enable ,0 - disable
out_mux_en	opamp2 output mux enable(ULP_GPIO_9)
dyn_en	opamp2 dynamic mode enable 1 - enable 0 - disable
channel	channel no to set in dynamic mode

Return values

None

Example

```
RSI_OPAMP2_UGB(1,1,0,1,0,0);
```

RSI_OPAMP3_UGB

Source File :rsi_opamp.c

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
void RSI_OPAMP3_UGB(uint8_t vin_p_sel,uint8_t enable,uint8_t lp_mode,uint8_t
out_mux_en,uint8_t dyn_en,uint8_t channel)
```

Description

This API is used to configure the opamp3 in Unity gain buffer mode.

Parameters

Parameter	Description

vin_p_sel	input selection 0 - ULP_GPIO_10 1 - ULP_GPIO_14 2 -AUX_DAC_OUT 5 -OPAMP2_OUT
enable	opamp3 enable 1 -enable 0 -disable
lp_mode	opamp3 low power mode 1 - enable ,0 - disable
out_mux_en	opamp3 output mux enable(GPIO_27)
dyn_en	opamp3 dynamic mode enable 1 - enable 0 - disable
channel	channel no to set in dynamic mode

Return values

None

Example

```
RSI_OPAMP3_UGB(0,1,0,1,0,0);
```

RSI_OPAMP1_TIA

Source File :rsi_opamp.c

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
void RSI_OPAMP1_TIA(uint8_t vref_sel,uint8_t iin_sel,uint8_t enable,uint8_t lp_mode,
                    uint8_t r2_sel, uint8_t out_mux_en,uint8_t out_mux_sel,uint8_t
dyn_en,uint8_t channel)
```

Description

This API is used to configure the opamp1 in Trans Impedance Amplifier.

Parameters

Parameter	**Description**
vref_sel	ref voltage gpio selection 0 - ULP_GPIO_12 1 - ULP_GPIO_7 2 - ULP_GPIO_0 3 -ULP_GPIO_2

	4 -ULP_GPIO_6 5 -ULP_GPIO_8 6 - AUX_DAC_OUT
iin_sel	input current gpio selection 0 - ULP_GPIO_12 1 - ULP_GPIO_7 2 - ULP_GPIO_0 3 -ULP_GPIO_2 4 -ULP_GPIO_6 5 -ULP_GPIO_8
enable	opamp1 enable
lp_mode	opamp1 low power mode enable
r2_Sel	selection of resistor r2 0 - 20kohm 1 - 30kohm 2 - 40kohm 3- 60kohm 4 - 120kohm 5 - 250kohm 6 - 500kohm 7 - 1000kohm
out_mux_en	opamp1 output mux enable
out_mux_sel	opamp1 output mux selection
dyn_en	opamp1 dynamic mode enable
channel	channel no to set in dynamic mode

Return values

None

Example

```
RSI_OPAMP1_TIA(1,2,1,0,1,1,0,0,0);
```

RSI_OPAMP2_TIA

Source File :rsi_opamp.c

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
void RSI_OPAMP2_TIA(uint8_t vref_sel,uint8_t iin_sel,uint8_t enable,uint8_t lp_mode,
        uint8_t r2_sel,uint8_t out_mux_en,uint8_t dyn_en,uint8_t channel)
```

Description

This API is used to configure the opamp2 in Trans Impedance Amplifier.

Parameters

Parameter	Description
vref_sel	ref voltage gpio selection 0 - ULP_GPIO_11 1 - ULP_GPIO_5 3 - AUX_DAC_OUT 6 - OPAMP1_OUT
iin_sel	input current gpio selection 0 - ULP_GPIO_11 1 - ULP_GPIO_5
enable	opamp2 enable
lp_mode	opamp2 low power mode enable
r2_Sel	selection of resistor r2 0 - 20kohm 1 - 30kohm 2 - 40kohm 3- 60kohm 4 - 120kohm 5 - 250kohm 6 - 500kohm 7 - 1000kohm
out_mux_en	opamp2 output mux enable
dyn_en	opamp2 dynamic mode enable
channel	channel no to set in dynamic mode

Return values

None

Example

```
RSI_OPAMP2_TIA(0,1,1,0,1,1,0,0);
```

RSI_OPAMP3_TIA

Source File :rsi_opamp.c

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
void RSI_OPAMP3_TIA(uint8_t vref_sel,uint8_t iin_sel,uint8_t enable,uint8_t lp_mode,
        uint8_t r2_sel,uint8_t out_mux_en,uint8_t dyn_en,uint8_t channel)
```

Description

This API is used to configure the opamp3 in Trans Impedance Amplifier.

Parameters

Parameter	Description
vref_sel	ref voltage gpio selection 0 - ULP_GPIO_10 1 - ULP_GPIO_14 2 - AUX_DAC_OUT 5 - OPAMP2_OUT
iin_sel	input current gpio selection 0 - ULP_GPIO_10 1 - ULP_GPIO_14
enable	opamp3 enable
lp_mode	opamp3 low power mode enable
r2_Sel	selection of resistor r2 0 - 20kohm 1 - 30kohm 2 - 40kohm 3- 60kohm 4 - 120kohm 5 - 250kohm 6 - 500kohm 7 - 1000kohm
out_mux_en	opamp3 output mux enable
dyn_en	opamp3 dynamic mode enable
channel	channel no to set in dynamic mode

Return values

None

Example

```
RSI_OPAMP3_TIA(5,0,1,0,1,1,0,0);
```

RSI_OPAMP1_InvPGA

Source File :rsi_opamp.c

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
void RSI_OPAMP1_InvPGA(uint8_t vref_sel,uint8_t vin_sel,uint8_t enable,uint8_t lp_mode,
        uint8_t r1_sel,uint8_t r2_sel,uint8_t out_mux_en,uint8_t out_mux_sel,uint8_t
dyn_en,uint8_t channel)
```

Description

This API is used to configure the opamp1 in Inverting PGA.

Parameters

Parameter	Description
vref_sel	ref voltage gpio selection 0 - ULP_GPIO_12 1 - ULP_GPIO_7 2 - ULP_GPIO_0 3 -ULP_GPIO_2 4 -ULP_GPIO_6 5 -ULP_GPIO_8 6 - AUX_DAC_OUT
vin_sel	input voltage gpio selection 0 - ULP_GPIO_12 1 - ULP_GPIO_7 2 - ULP_GPIO_0 3 -ULP_GPIO_2 4 -ULP_GPIO_6 5 -ULP_GPIO_8 6 - AUX_ADC_DAC 7- GND
enable	opamp1 enable
lp_mode	opamp1 low power mode enable
r1_sel	selection of resistor r1 0 - 0kohm 1 - 20kohm 2 - 60kohm 3- 140kohm
r2_Sel	selection of resistor r2 0 - 20kohm 1 - 30kohm 2 - 40kohm 3- 60kohm 4 - 120kohm 5 - 250kohm 6 - 500kohm

	7 - 1000kohm
out_mux_en	opamp1 output mux enable
out_mux_sel	opamp1 output mux selection
dyn_en	opamp1 dynamic mode enable
channel	channel no to set in dynamic mode

Return values

None

Example

```
RSI_OPAMP1_InvPGA(4,1,1,0,1,2,1,0,0,0);
```

RSI_OPAMP2_InvPGA

Source File :rsi_opamp.c

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
void RSI_OPAMP2_InvPGA(uint8_t vref_sel,uint8_t vin_sel,uint8_t enable,uint8_t lp_mode,
        uint8_t r1_sel,uint8_t r2_sel,uint8_t out_mux_en,uint8_t dyn_en,uint8_t channel)
```

44.3.1 Description

This API is used to configure the opamp2 in Inverting PGA.

Parameters

Parameter	Description
vref_sel	ref voltage gpio selection 0 - ULP_GPIO_11 1 - ULP_GPIO_5 3 - AUX_DAC_OUT 6 - OPAMP1_OUT
vin_sel	input voltage gpio selection 0 - ULP_GPIO_11 1 - ULP_GPIO_5 2 - AUX_DAC_OUT 4 - GND 5 - OPAMP1_OUT
enable	opamp2 enable
lp_mode	opamp2 low power mode enable
r1_sel	selection of resistor r1

	0 - 0kohm 1 - 20kohm 2 - 60kohm 3- 140kohm
r2_Sel	selection of resistor r2 0 - 20kohm 1 - 30kohm 2 - 40kohm 3- 60kohm 4 - 120kohm 5 - 250kohm 6 - 500kohm 7 - 1000kohm
out_mux_en	opamp2 output mux enable
dyn_en	opamp2 dynamic mode enable
channel	channel no to set in dynamic mode

Return values

None

Example

```
RSI_OPAMP2_InvPGA(0,1,1,0,1,1,1,0,0);
```

RSI_OPAMP3_InvPGA

Source File :rsi_opamp.c

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
void RSI_OPAMP3_InvPGA(uint8_t vref_sel,uint8_t vin_sel,uint8_t enable,uint8_t lp_mode,
                       uint8_t r1_sel,uint8_t r2_sel,uint8_t out_mux_en,uint8_t
dyn_en,uint8_t channel)
```

44.3.2 Description

This API is used to configure the opamp3 in Inverting PGA.

Parameters

Parameter	Description
vref_sel	ref voltage gpio selection 0 - ULP_GPIO_10 1 - ULP_GPIO_14 2 -AUX_DAC_OUT

	5 -OPAMP2_OUT
vin_sel	input voltage gpio selection 0 - ULP_GPIO_10 1 - ULP_GPIO_14 2 - AUX_DAC_OUT 3 - GND 4 - OPAMP2_OUT
enable	opamp3 enable
lp_mode	opamp3 low power mode enable
r1_sel	selection of resistor r1 0 - 0kohm 1 - 20kohm 2 - 60kohm 3- 140kohm
r2_Sel	selection of resistor r2 0 - 20kohm 1 - 30kohm 2 - 40kohm 3- 60kohm 4 - 120kohm 5 - 250kohm 6 - 500kohm 7 - 1000kohm
out_mux_en	opamp3 output mux enable
dyn_en	opamp3 dynamic mode enable
channel	channel no to set in dynamic mode

Return values

None

Example

```
RSI_OPAMP3_InvPGA(0,4,1,0,1,1,1,0,0);
```

RSI_OPAMP1_NonInvPGA

Source File :rsi_opamp.c

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
void RSI_OPAMP1_NonInvPGA(uint8_t vin_sel,uint8_t vref_sel,uint8_t enable,uint8_t lp_mode,
        uint8_t r1_sel,uint8_t r2_sel,uint8_t out_mux_en,uint8_t out_mux_sel,uint8_t
```

```
dyn_en,uint8_t channel)
```

Description

This API is used to configure the opamp1 in NonInverting PGA.

Parameters

Parameter	Description
vin_sel	input voltage gpio selection 0 - ULP_GPIO_12 1 - ULP_GPIO_7 2 - ULP_GPIO_0 3 -ULP_GPIO_2 4 -ULP_GPIO_6 5 -ULP_GPIO_8 6 - AUX_ADC_DAC
vref_sel	ref voltage gpio selection 0 - ULP_GPIO_12 1 - ULP_GPIO_7 2 - ULP_GPIO_0 3 -ULP_GPIO_2 4 -ULP_GPIO_6 5 -ULP_GPIO_8 6 - AUX_ADC_DAC 7- GND
enable	opamp1 enable
lp_mode	opamp1 low power mode enable
r1_sel	selection of resistor r1 0 - 0kohm 1 - 20kohm 2 - 60kohm 3- 140kohm
r2_Sel	selection of resistor r2 0 - 20kohm 1 - 30kohm 2 - 40kohm 3- 60kohm 4 - 120kohm 5 - 250kohm 6 - 500kohm 7 - 1000kohm
out_mux_en	opamp1 output mux enable

out_mux_sel	opamp1 output mux selection
dyn_en	opamp1 dynamic mode enable
channel	channel no to set in dynamic mode

Return values

None

Example

```
RSI_OPAMP1_NonInvPGA(2,1,1,0,1,2,1,0,0,0);
```

RSI_OPAMP2_NonInvPGA

Source File :rsi_opamp.c

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
void RSI_OPAMP2_NonInvPGA(uint8_t vin_sel,uint8_t vref_sel,uint8_t enable,uint8_t lp_mode,
        uint8_t r1_sel,uint8_t r2_sel,uint8_t out_mux_en,uint8_t dyn_en,uint8_t channel)
```

44.3.3 Description

This API is used to configure the opamp2 in NonIverting PGA

Parameters

Parameter	**Description**
vin_sel	input voltage gpio selection 0 - ULP_GPIO_11 1 - ULP_GPIO_5 3 - AUX_DAC_OUT 6 - OPAMP1_OUT
vref_sel	ref voltage gpio selection 0 - ULP_GPIO_11 1 - ULP_GPIO_5 3 - AUX_DAC_OUT 4 - GND,5 - OPAMP1_OUT
enable	opamp2 enable
lp_mode	opamp2 low power mode enable
r1_sel	selection of resistor r1 0 - 0kohm 1 - 20kohm 2 - 60kohm 3- 140kohm

r2_Sel	selection of resistor r2 0 - 20kohm 1 - 30kohm 2 - 40kohm 3- 60kohm 4 - 120kohm 5 - 250kohm 6 - 500kohm 7 - 1000kohm
out_mux_en	opamp2 output mux enable
dyn_en	opamp2 dynamic mode enable
channel	channel no to set in dynamic mode

Return values

None

Example

```
RSI_OPAMP2_NonInvPGA(0,1,1,0,1,1,1,0,0);
```

RSI_OPAMP3_NonInvPGA

Source File :rsi_opamp.c

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
void RSI_OPAMP3_NonInvPGA(uint8_t vin_sel,uint8_t vref_sel,uint8_t enable,uint8_t lp_mode,
        uint8_t r1_sel,uint8_t r2_sel,uint8_t out_mux_en,uint8_t dyn_en,uint8_t channel)
```

44.3.4 Description

This API is used to configure the opamp3 in Inverting PGA.

Parameters

Parameter	**Description**
vin_sel	Input voltage gpio selection 0 - ULP_GPIO_10 1 - ULP_GPIO_14 2 -AUX_DAC_OUT 5 -OPAMP2_OUT
vref_sel	Ref voltage gpio selection 0 - ULP_GPIO_10 1 - ULP_GPIO_14 2 - AUX_DAC_OUT

	3 - GND 4 - OPAMP2_OUT
enable	Opamp3 enable
lp_mode	Opamp3 low power mode enable
r1_sel	selection of resistor r1 0 - 0kohm 1 - 20kohm 2 - 60kohm 3- 140kohm
r2_Sel	selection of resistor r2 0 - 20kohm 1 - 30kohm 2 - 40kohm 3- 60kohm 4 - 120kohm 5 - 250kohm 6 - 500kohm 7 - 1000kohm
out_mux_en	Opamp3 output mux enable
dyn_en	Opamp3 dynamic mode enable
channel	Channel no to set in dynamic mode

Return values

None

Example

```
RSI_OPAMP3_NonInvPGA(0,4,1,0,1,1,1,0,0);
```

RSI_OPAMP1_InvCMP

Source File :rsi_opamp.c

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
void RSI_OPAMP1_InvCMP(uint8_t vin_sel,uint8_t vref_sel,uint8_t enable,uint8_t lp_mode,
        uint8_t r1_sel,uint8_t r2_sel,uint8_t out_mux_en,uint8_t out_mux_sel,uint8_t
dyn_en,uint8_t channel)
```

Description

This API is used to configure the opamp1 as inverting comparator with programmable hysteresis.

Parameters

Parameter	Description
vin_sel	Input voltage gpio selection 0 - ULP_GPIO_12 1 - ULP_GPIO_7 2 -AUX_DAC_OUT
vref_sel	Ref voltage gpio selection 0 - ULP_GPIO_12 1 - ULP_GPIO_7 2 - ULP_GPIO_0 3 -ULP_GPIO_2 4 -ULP_GPIO_6 5 -ULP_GPIO_8 6 - AUX_DAC_OUT 7 - GND
enable	opamp1 enable
lp_mode	opamp1 low power mode enable
r1_sel	selection of resistor r1 0 - 0kohm 1 - 20kohm 2 - 60kohm 3- 140kohm
r2_Sel	selection of resistor r2 0 - 20kohm 1 - 30kohm 2 - 40kohm 3- 60kohm 4 - 120kohm 5 - 250kohm 6 - 500kohm 7 - 1000kohm
out_mux_en	opamp1 output mux enable
out_mux_sel	opamp1 output mux selection
dyn_en	opamp1 dynamic mode enable
channel	channel no to set in dynamic mode

Return values

None

Example

```
RSI_OPAMP1_InvCMP(1,2,1,0,1,1,1,0,0,0);
```

RSI_OPAMP2_InvCMP

Source File :rsi_opamp.c

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
void RSI_OPAMP2_InvCMP(uint8_t vin_sel,uint8_t vref_sel,uint8_t enable,uint8_t lp_mode,
        uint8_t r1_sel,uint8_t r2_sel,uint8_t out_mux_en,uint8_t dyn_en,uint8_t channel)
```

44.3.5 Description

This API is used to configure the opamp2 as inverting comparator with programmable hysteresis.

Parameters

Parameter	Description
vin_sel	Input voltage gpio selection 0 - ULP_GPIO_11 1 - AUX_DAC_OUT
vref_sel	Ref voltage gpio selection 0 - ULP_GPIO_11 1 - ULP_GPIO_5 3 - AUX_DAC_OUT 4 - GND 5 - OPAMP1_OUT
enable	opamp2 enable
lp_mode	opamp2 low power mode enable
r1_sel	selection of resistor r1 0 - 0kohm 1 - 20kohm 2 - 60kohm 3- 140kohm
r2_Sel	selection of resistor r2 0 - 20kohm 1 - 30kohm 2 - 40kohm 3- 60kohm 4 - 120kohm 5 - 250kohm 6 - 500kohm

	7 - 1000kohm
out_mux_en	opamp2 output mux enable
dyn_en	opamp2 dynamic mode enable
channel	channel no to set in dynamic mode

Return values

None

Example

```
RSI_OPAMP2_InvCMP(0,1,1,0,1,1,1,0,0);
```

RSI_OPAMP3_InvCMP

Source File :rsi_opamp.c

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
void RSI_OPAMP3_InvCMP(uint8_t vin_sel,uint8_t vref_sel,uint8_t enable,uint8_t lp_mode,
        uint8_t r1_sel,uint8_t r2_sel,uint8_t out_mux_en,uint8_t dyn_en,uint8_t channel)
```

44.3.6 Description

This API is used to configure the opamp3 as inverting comparator with programmable hysteresis.

Parameters

Parameter	**Description**
vin_sel	Input voltage gpio selection 0 - ULP_GPIO_10 1 - AUX_DAC_OUT
vref_sel	Ref voltage gpio selection 0 - ULP_GPIO_10 1 - ULP_GPIO_14 2 -AUX_DAC_OUT 3 - GND 4 -OPAMP2_OUT
enable	opamp3 enable
lp_mode	opamp3 low power mode enable
r1_sel	selection of resistor r1 0 - 0kohm 1 - 20kohm 2 - 60kohm 3- 140kohm

r2_Sel	selection of resistor r2 0 - 20kohm 1 - 30kohm 2 - 40kohm 3- 60kohm 4 - 120kohm 5 - 250kohm 6 - 500kohm 7 - 1000kohm
out_mux_en	opamp3 output mux enable
dyn_en	opamp3 dynamic mode enable
channel	channel no to set in dynamic mode

Return values

None

Example

```
RSI_OPAMP3_InvCMP(0,4,1,0,1,1,1,0,0);
```

RSI_OPAMP1_NonInvCMP

Source File :rsi_opamp.c

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
void RSI_OPAMP1_NonInvCMP(uint8_t vin_sel,uint8_t vref_sel,uint8_t enable,uint8_t lp_mode,
        uint8_t r1_sel,uint8_t r2_sel,uint8_t out_mux_en,uint8_t out_mux_sel,uint8_t
dyn_en,uint8_t channel)
```

Description

This API is used to configure the opamp1 as NonInverting comparator with programmable hysteresis.

Parameters

Parameter	Description
vin_sel	Input voltage gpio selection 0 - ULP_GPIO_12 1 - ULP_GPIO_7 2 - ULP_GPIO_0 3 - ULP_GPIO_2 4 -ULP_GPIO_6 5 -ULP_GPIO_8

	6 - AUX_DAC_OUT
vref_sel	Ref voltage gpio selection 0 - ULP_GPIO_12 1 - ULP_GPIO_7 2 - AUX_DAC_OUT
enable	opamp1 enable
lp_mode	opamp1 low power mode enable
r1_sel	selection of resistor r1 0 - 0kohm 1 - 20kohm 2 - 60kohm 3- 140kohm
r2_Sel	selection of resistor r2 0 - 20kohm 1 - 30kohm 2 - 40kohm 3- 60kohm 4 - 120kohm 5 - 250kohm 6 - 500kohm 7 - 1000kohm
out_mux_en	opamp1 output mux enable
out_mux_sel	opamp1 output mux selection
dyn_en	opamp1 dynamic mode enable
channel	channel no to set in dynamic mode

Return values

None

Example

```
RSI_OPAMP1_NonInvCMP(2,1,1,0,1,1,1,0,0,0);
```

RSI_OPAMP2_NonInvCMP

Source File :rsi_opamp.c

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
void RSI_OPAMP2_NonInvCMP(uint8_t vin_sel,uint8_t vref_sel,uint8_t enable,uint8_t lp_mode,
        uint8_t r1_sel,uint8_t r2_sel,uint8_t out_mux_en,uint8_t dyn_en,uint8_t channel)
```

44.3.7 Description

This API is used to configure the opamp2 as NonInverting comparator with programmable hysteresis.

Parameters

Parameter	Description
vin_sel	Input voltage gpio selection 0 - ULP_GPIO_11 1 - ULP_GPIO_5 3 - AUX_DAC_OUT 5 - OPAMP1_OUT
vref_sel	Ref voltage gpio selection 0 - ULP_GPIO_11 1 - AUX_DAC_OUT
enable	opamp2 enable
lp_mode	opamp2 low power mode enable
r1_sel	selection of resistor r1 0 - 0kohm 1 - 20kohm 2 - 60kohm 3- 140kohm
r2_Sel	selection of resistor r2 0 - 20kohm 1 - 30kohm 2 - 40kohm 3- 60kohm 4 - 120kohm 5 - 250kohm 6 - 500kohm 7 - 1000kohm
out_mux_en	opamp2 output mux enable
dyn_en	opamp2 dynamic mode enable
channel	channel no to set in dynamic mode

Return values

None

Example

```
RSI_OPAMP2_NonInvCMP(1,0,1,0,1,1,1,0,0);
```

RSI_OPAMP3_NonInvCMP

Source File :rsi_opamp.c

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
void RSI_OPAMP3_NonInvCMP(uint8_t vin_sel,uint8_t vref_sel,uint8_t enable,uint8_t lp_mode,
        uint8_t r1_sel,uint8_t r2_sel,uint8_t out_mux_en,uint8_t dyn_en,uint8_t channel)
```

44.3.8 Description

This API is used to configure the opamp3 as NonInverting comparator with programmable hysteresis.

Parameters

Parameter	Description
vin_sel	Input voltage gpio selection 0 - ULP_GPIO_10 1 - ULP_GPIO_14 2 - AUX_DAC_OUT 4 - OPAMP2_OUT
vref_sel	Ref voltage gpio selection 0 - ULP_GPIO_10 1 - AUX_DAC_OUT
enable	opamp3 enable
lp_mode	opamp3 low power mode enable
r1_sel	selection of resistor r1 0 - 0kohm 1 - 20kohm 2 - 60kohm 3- 140kohm
r2_Sel	selection of resistor r2 0 - 20kohm 1 - 30kohm 2 - 40kohm 3- 60kohm 4 - 120kohm 5 - 250kohm 6 - 500kohm 7 - 1000kohm
out_mux_en	opamp3 output mux enable
dyn_en	opamp3 dynamic mode enable
channel	channel no to set in dynamic mode

Return values

None

Example

```
RSI_OPAMP3_NonInvCMP(4,0,1,0,1,1,1,0,0);
```

RSI_OPAMP_InstrAMP

Source File :rsi_opamp.c

Path: Redpine_MCU_Vx.y.z\Host_MCU\Peripheral_Library\driver\src\

Prototype

```
void RSI_OPAMP_InstrAMP(uint8_t vin_p_sel,uint8_t vin_n_sel,uint8_t enable,uint8_t
lp_mode,
        uint8_t r1_sel,uint8_t r2_sel,uint8_t out_mux_en,uint8_t dyn_en,uint8_t channel)
```

44.3.9 Description

This API is used to configuration of opamp's as instrumentation amplifier.

Parameters

Parameter	Description
vin_p_sel	Input voltage gpio selection 0 - ULP_GPIO_12 1 - ULP_GPIO_7 2 - ULP_GPIO_0 3 -ULP_GPIO_2 4 -ULP_GPIO_6 5 -ULP_GPIO_8
vin_n_sel	Ref voltage gpio selection 0 - ULP_GPIO_11 1 - ULP_GPIO_5
enable	opamp enable
lp_mode	opamp low power mode enable
r1_sel	selection of resistor r1 0 - 0kohm 1 - 20kohm 2 - 60kohm 3- 140kohm
r2_Sel	selection of resistor r2 0 - 20kohm 1 - 30kohm 2 - 40kohm 3- 60kohm

	4 - 120kohm 5 - 250kohm 6 - 500kohm 7 - 1000kohm
out_mux_en	opamp output mux enable
dyn_en	opamp dynamic mode enable
channel	channel no to set in dynamic mode

Return values

None

Example

```
RSI_OPAMP_InstrAMP(1,0,1,0,1,2,1,0,0);
```

45 MCU FreeRTOS Documentation.

45.1 1). FreeRTOS Architecture and Porting :

FreeRTOS.pdf

45.2 2). FreeRTOS standard reference Guide :

USINGTHEFREERT...TIMEKERNEL.pdf

Disclaimer

The information in this document pertains to information related to Redpine Signals, Inc. products. This information is provided as a service to our customers, and may be used for information purposes only. Redpine assumes no liabilities or responsibilities for errors or omissions in this document. This document may be changed at any time at Redpine's sole discretion without any prior notice to anyone. Redpine is not committed to updating this document in the future.
Copyright © 2019 Redpine Signals, Inc. All rights reserved.

Redpine Signals, Inc.
2107 North First Street, Suite #540, San Jose, California 95131,
United States of America.
Phone: +1-408-748-3385, Fax: +1-408-705-2019
Email: sales@redpinesignals.com
Website: www.redpinesignals.com