

libjtaghal

Generated by Doxygen 1.8.13

Contents

1	jtaghal	1
2	Module Index	3
2.1	Modules	3
3	Hierarchical Index	5
3.1	Class Hierarchy	5
4	Class Index	9
4.1	Class List	9
5	File Index	13
5.1	File List	13
6	Module Documentation	19
6.1	JTAG interface layer	19
6.1.1	Detailed Description	19
6.2	Stuff not in another group yet	20
6.2.1	Detailed Description	23
6.2.2	Function Documentation	23
6.2.2.1	FlipBitAndEndian32Array()	23
6.2.2.2	FlipBitAndEndianArray()	24
6.2.2.3	FlipBitArray()	24
6.2.2.4	FlipByte()	24
6.2.2.5	FlipByteArray()	25
6.2.2.6	FlipEndian32Array()	25
6.2.2.7	FlipEndianArray()	25
6.2.2.8	GetTime()	25
6.2.2.9	MirrorBitArray()	26
6.2.2.10	PeekBit()	26
6.2.2.11	PokeBit()	26

7	Class Documentation	29
7.1	ARM7TDMISProcessor Class Reference	29
7.1.1	Detailed Description	31
7.1.2	Member Function Documentation	31
7.1.2.1	Erase()	31
7.1.2.2	IsProgrammed()	32
7.1.2.3	LoadFirmwareImage()	32
7.1.2.4	PostInitProbes()	32
7.1.2.5	Program()	33
7.2	ARMAPBDevice Class Reference	33
7.2.1	Detailed Description	34
7.3	ARMCoreSightDevice Class Reference	35
7.3.1	Detailed Description	35
7.4	ARMCortexA57 Class Reference	36
7.4.1	Detailed Description	37
7.5	ARMCortexA9 Class Reference	38
7.5.1	Detailed Description	40
7.6	ARMCortexM4 Class Reference	40
7.6.1	Detailed Description	41
7.7	ARMDebugAccessPort Class Reference	41
7.7.1	Detailed Description	42
7.8	ARMDebugMemAccessPort Class Reference	43
7.8.1	Detailed Description	44
7.9	ARMDebugMemAPControlStatusWord Union Reference	44
7.9.1	Detailed Description	45
7.10	ARMDebugPeripheralIDRegister Union Reference	45
7.10.1	Detailed Description	46
7.11	ARMDebugPeripheralIDRegisterBits Class Reference	46
7.11.1	Detailed Description	47
7.12	ARMDebugPort Class Reference	47

7.12.1 Detailed Description	48
7.13 ARMDebugPortIDRegister Union Reference	49
7.13.1 Detailed Description	49
7.14 ARMDevice Class Reference	50
7.14.1 Detailed Description	51
7.14.2 Constructor & Destructor Documentation	51
7.14.2.1 ARMDevice()	51
7.14.3 Member Function Documentation	51
7.14.3.1 CreateDevice()	52
7.15 ARMFlashPatchBreakpoint Class Reference	52
7.15.1 Detailed Description	54
7.16 ARMJtagDebugPort Class Reference	54
7.16.1 Detailed Description	56
7.16.2 Member Function Documentation	56
7.16.2.1 APRegisterRead()	56
7.16.2.2 APRegisterWrite()	57
7.16.2.3 DPRRegisterRead()	57
7.16.2.4 GetDescription()	57
7.16.2.5 PostInitProbes()	58
7.17 ARMJtagDebugPortStatusRegister Union Reference	58
7.17.1 Detailed Description	59
7.18 ARMv7DebugIDRegister Union Reference	59
7.18.1 Detailed Description	60
7.19 ARMv7DebugStatusControlRegister Union Reference	60
7.19.1 Detailed Description	61
7.20 ARMv7MProcessor Class Reference	62
7.20.1 Detailed Description	63
7.20.2 Member Function Documentation	63
7.20.2.1 DebugHalt()	63
7.21 ARMv7Processor Class Reference	64

7.21.1 Detailed Description	66
7.21.2 Member Function Documentation	66
7.21.2.1 DebugHalt()	66
7.22 ARMv8Processor Class Reference	67
7.22.1 Detailed Description	69
7.23 AttachedMemoryDevice Class Reference	69
7.23.1 Detailed Description	69
7.24 ByteArrayFirmwareImage Class Reference	70
7.24.1 Detailed Description	71
7.25 CPLD Class Reference	72
7.25.1 Detailed Description	73
7.25.2 Member Function Documentation	73
7.25.2.1 ParseJEDFile()	73
7.25.2.2 ReadIntLine()	74
7.26 CPLDBitstream Class Reference	74
7.26.1 Detailed Description	76
7.27 DebuggableDevice Class Reference	76
7.27.1 Detailed Description	77
7.28 DebuggerInterface Class Reference	77
7.28.1 Detailed Description	78
7.29 DigilentJtagInterface Class Reference	79
7.29.1 Detailed Description	81
7.29.2 Constructor & Destructor Documentation	81
7.29.2.1 DigilentJtagInterface()	81
7.29.2.2 ~DigilentJtagInterface()	81
7.29.3 Member Function Documentation	81
7.29.3.1 GetFrequency()	81
7.29.3.2 GetName()	82
7.29.3.3 GetSerial()	82
7.29.3.4 GetUserID()	82

7.29.3.5	SendDummyClocks()	82
7.29.3.6	ShiftData()	83
7.29.3.7	ShiftTMS()	83
7.30	EjtagControlRegister Union Reference	84
7.30.1	Detailed Description	85
7.31	EjtagImplementationCodeRegister Union Reference	85
7.31.1	Detailed Description	85
7.32	FirmwareImage Class Reference	86
7.32.1	Detailed Description	87
7.32.2	Member Data Documentation	87
7.32.2.1	idcode	87
7.33	FPGA Class Reference	87
7.33.1	Detailed Description	88
7.34	FPGABitstream Class Reference	88
7.34.1	Detailed Description	90
7.35	FreescaleDevice Class Reference	91
7.35.1	Detailed Description	92
7.35.2	Constructor & Destructor Documentation	92
7.35.2.1	FreescaleDevice()	93
7.35.3	Member Function Documentation	93
7.35.3.1	CreateDevice()	93
7.36	FreescaleIMXDevice Class Reference	94
7.36.1	Detailed Description	95
7.36.2	Member Enumeration Documentation	95
7.36.2.1	instructions	95
7.36.3	Member Function Documentation	96
7.36.3.1	Erase()	96
7.36.3.2	GetDescription()	96
7.36.3.3	IsProgrammed()	96
7.36.3.4	PostInitProbes()	97

7.36.3.5	Program()	97
7.37	FreescaleIMXSmartDMA Class Reference	97
7.37.1	Detailed Description	99
7.37.2	Member Function Documentation	99
7.37.2.1	GetDescription()	99
7.37.2.2	PostInitProbes()	99
7.38	FreescaleMicrocontroller Class Reference	100
7.38.1	Detailed Description	101
7.39	FTDIJtagInterface Class Reference	101
7.39.1	Detailed Description	104
7.39.2	Constructor & Destructor Documentation	105
7.39.2.1	FTDIJtagInterface()	105
7.39.2.2	~FTDIJtagInterface()	105
7.39.3	Member Function Documentation	105
7.39.3.1	Commit()	105
7.39.3.2	DoReadback()	106
7.39.3.3	GenerateShiftPacket()	106
7.39.3.4	GetAPIVersion()	106
7.39.3.5	GetDefaultFrequency()	107
7.39.3.6	GetDescription()	107
7.39.3.7	GetInterfaceCount()	107
7.39.3.8	GetSerialNumber()	108
7.39.3.9	IsJtagCapable()	108
7.39.3.10	IsSplitScanSupported()	109
7.39.3.11	ReadData()	109
7.39.3.12	SendDummyClocks()	109
7.39.3.13	SendDummyClocksDeferred()	110
7.39.3.14	ShiftData()	110
7.39.3.15	ShiftDataReadOnly()	111
7.39.3.16	ShiftDataWriteOnly()	111

7.39.3.17 ShiftTMS()	112
7.39.3.18 SyncCheck()	112
7.39.3.19 WriteData() [1/2]	113
7.39.3.20 WriteData() [2/2]	113
7.39.3.21 WriteDataRow()	113
7.40 GPIOInterface Class Reference	114
7.40.1 Detailed Description	115
7.41 JtagDevice Class Reference	116
7.41.1 Detailed Description	119
7.41.2 Constructor & Destructor Documentation	119
7.41.2.1 JtagDevice()	119
7.41.3 Member Function Documentation	119
7.41.3.1 Commit()	119
7.41.3.2 CreateDevice()	119
7.41.3.3 EnterShiftDR()	120
7.41.3.4 GetDescription()	120
7.41.3.5 IsSplitScanSupported()	120
7.41.3.6 PostInitProbes()	120
7.41.3.7 ResetToIdle()	121
7.41.3.8 ScanDR()	121
7.41.3.9 ScanDRDeferred()	121
7.41.3.10 ScanDRSplitRead()	121
7.41.3.11 ScanDRSplitWrite()	122
7.41.3.12 SendDummyClocks()	122
7.41.3.13 SendDummyClocksDeferred()	122
7.41.3.14 SetIR() [1/2]	122
7.41.3.15 SetIR() [2/2]	122
7.41.3.16 ShiftData()	123
7.42 JtagDummy Class Reference	123
7.42.1 Detailed Description	124

7.42.2	Constructor & Destructor Documentation	125
7.42.2.1	JtagDummy()	125
7.42.3	Member Function Documentation	125
7.42.3.1	GetDescription()	125
7.42.3.2	PostInitProbes()	125
7.43	JtagException Class Reference	126
7.43.1	Detailed Description	127
7.43.2	Constructor & Destructor Documentation	127
7.43.2.1	JtagException()	127
7.43.3	Member Function Documentation	128
7.43.3.1	GetDescription()	128
7.44	JtagFPGA Class Reference	128
7.44.1	Detailed Description	129
7.44.2	Constructor & Destructor Documentation	129
7.44.2.1	JtagFPGA()	129
7.44.3	Member Function Documentation	129
7.44.3.1	GetUserVIDPID()	130
7.45	JtagInterface Class Reference	130
7.45.1	Detailed Description	133
7.45.2	Constructor & Destructor Documentation	136
7.45.2.1	JtagInterface()	136
7.45.2.2	~JtagInterface()	136
7.45.3	Member Function Documentation	136
7.45.3.1	Commit()	136
7.45.3.2	CreateDummyDevices()	137
7.45.3.3	EnterShiftDR()	137
7.45.3.4	EnterShiftIR()	137
7.45.3.5	GetDataBitCount()	137
7.45.3.6	GetDevice()	138
7.45.3.7	GetDeviceCount()	138

7.45.3.8	GetDummyClockCount()	138
7.45.3.9	GetFrequency()	139
7.45.3.10	GetIDCode()	139
7.45.3.11	GetModeBitCount()	140
7.45.3.12	GetName()	140
7.45.3.13	GetRecoverableErrorCount()	140
7.45.3.14	GetSerial()	141
7.45.3.15	GetShiftOpCount()	141
7.45.3.16	GetShiftTime()	141
7.45.3.17	GetUserID()	142
7.45.3.18	InitializeChain()	142
7.45.3.19	IsSplitScanSupported()	142
7.45.3.20	LeaveExit1DR()	143
7.45.3.21	LeaveExit1IR()	143
7.45.3.22	ResetToIdle()	143
7.45.3.23	ScanDR()	143
7.45.3.24	ScanDRDeferred()	144
7.45.3.25	ScanDRSplitRead()	145
7.45.3.26	ScanDRSplitWrite()	145
7.45.3.27	SendDummyClocks()	146
7.45.3.28	SendDummyClocksDeferred()	146
7.45.3.29	SetIR() [1/2]	147
7.45.3.30	SetIR() [2/2]	147
7.45.3.31	SetIRDeferred()	148
7.45.3.32	ShiftData()	148
7.45.3.33	ShiftDataReadOnly()	148
7.45.3.34	ShiftDataWriteOnly()	149
7.45.3.35	ShiftTMS()	150
7.45.3.36	SwapOutDummy()	150
7.45.3.37	TestLogicReset()	150

7.46 LockableDevice Class Reference	151
7.46.1 Detailed Description	152
7.46.2 Member Function Documentation	152
7.46.2.1 ClearReadLock()	152
7.46.2.2 SetReadLock()	152
7.47 MicrochipDevice Class Reference	153
7.47.1 Detailed Description	154
7.47.2 Constructor & Destructor Documentation	154
7.47.2.1 MicrochipDevice()	155
7.47.3 Member Function Documentation	155
7.47.3.1 CreateDevice()	155
7.48 MicrochipMicrocontroller Class Reference	156
7.48.1 Detailed Description	157
7.49 MicrochipPIC32Device Class Reference	157
7.49.1 Detailed Description	160
7.49.2 Member Enumeration Documentation	160
7.49.2.1 instructions	160
7.49.2.2 mtap_instructions	161
7.49.3 Member Function Documentation	161
7.49.3.1 Erase()	161
7.49.3.2 GetDescription()	161
7.49.3.3 IsProgrammed()	162
7.49.3.4 PostInitProbes()	162
7.49.3.5 Program()	162
7.50 MicrochipPIC32DeviceInfo Struct Reference	163
7.50.1 Detailed Description	163
7.51 MicrochipPIC32DeviceStatusRegister Union Reference	163
7.51.1 Detailed Description	164
7.52 Microcontroller Class Reference	164
7.52.1 Detailed Description	165

7.52.2	Member Function Documentation	165
7.52.2.1	LoadFirmwareImage()	165
7.53	NetworkedJtagInterface Class Reference	166
7.53.1	Detailed Description	168
7.53.2	Member Function Documentation	168
7.53.2.1	Commit()	168
7.53.2.2	Connect()	169
7.53.2.3	EnterShiftDR()	169
7.53.2.4	EnterShiftIR()	169
7.53.2.5	GetDataBitCount()	170
7.53.2.6	GetDummyClockCount()	170
7.53.2.7	GetFrequency()	170
7.53.2.8	GetModeBitCount()	171
7.53.2.9	GetName()	171
7.53.2.10	GetRecoverableErrorCount()	171
7.53.2.11	GetSerial()	172
7.53.2.12	GetShiftOpCount()	172
7.53.2.13	GetUserID()	173
7.53.2.14	IsSplitScanSupported()	173
7.53.2.15	LeaveExit1DR()	173
7.53.2.16	LeaveExit1IR()	173
7.53.2.17	ResetToIdle()	174
7.53.2.18	SendDummyClocks()	174
7.53.2.19	SendDummyClocksDeferred()	174
7.53.2.20	ShiftData()	175
7.53.2.21	ShiftDataReadOnly()	175
7.53.2.22	ShiftDataWriteOnly()	176
7.53.2.23	TestLogicReset()	176
7.54	PipeJtagInterface Class Reference	177
7.54.1	Detailed Description	180

7.54.2	Member Function Documentation	180
7.54.2.1	Commit()	180
7.54.2.2	EnterShiftDR()	180
7.54.2.3	EnterShiftIR()	180
7.54.2.4	GetDataBitCount()	181
7.54.2.5	GetDummyClockCount()	181
7.54.2.6	GetFrequency()	181
7.54.2.7	GetModeBitCount()	182
7.54.2.8	GetName()	182
7.54.2.9	GetRecoverableErrorCount()	182
7.54.2.10	GetSerial()	183
7.54.2.11	GetShiftOpCount()	183
7.54.2.12	GetUserID()	184
7.54.2.13	IsSplitScanSupported()	184
7.54.2.14	LeaveExit1DR()	184
7.54.2.15	LeaveExit1IR()	184
7.54.2.16	ResetToIdle()	185
7.54.2.17	SendDummyClocks()	185
7.54.2.18	SendDummyClocksDeferred()	185
7.54.2.19	ShiftData()	186
7.54.2.20	ShiftDataReadOnly()	186
7.54.2.21	ShiftDataWriteOnly()	187
7.54.2.22	TestLogicReset()	187
7.55	ProgrammableDevice Class Reference	188
7.55.1	Detailed Description	188
7.55.2	Member Function Documentation	189
7.55.2.1	Erase()	189
7.55.2.2	IsProgrammed()	189
7.55.2.3	LoadFirmwareImage() [1/2]	189
7.55.2.4	LoadFirmwareImage() [2/2]	190

7.55.2.5	Program()	190
7.56	ProgrammableLogicDevice Class Reference	191
7.56.1	Detailed Description	192
7.57	RawBinaryFirmwareImage Class Reference	192
7.57.1	Detailed Description	193
7.58	SerialNumberedDevice Class Reference	194
7.58.1	Detailed Description	194
7.58.2	Member Function Documentation	194
7.58.2.1	GetPrettyPrintedSerialNumber()	195
7.58.2.2	GetSerialNumber()	195
7.58.2.3	GetSerialNumberLength()	195
7.58.2.4	GetSerialNumberLengthBits()	196
7.58.2.5	ReadingSerialRequiresReset()	196
7.59	STM32Device Class Reference	196
7.59.1	Detailed Description	198
7.59.2	Member Function Documentation	199
7.59.2.1	ClearReadLock()	199
7.59.2.2	Erase()	199
7.59.2.3	GetDescription()	199
7.59.2.4	GetPrettyPrintedSerialNumber()	200
7.59.2.5	GetSerialNumber()	200
7.59.2.6	GetSerialNumberLength()	200
7.59.2.7	GetSerialNumberLengthBits()	201
7.59.2.8	IsProgrammed()	201
7.59.2.9	LoadFirmwareImage()	201
7.59.2.10	PostInitProbes()	201
7.59.2.11	Program()	202
7.59.2.12	ReadingSerialRequiresReset()	202
7.59.2.13	SetReadLock()	202
7.60	STMicroDevice Class Reference	203

7.60.1 Detailed Description	203
7.60.2 Member Function Documentation	203
7.60.2.1 CreateDevice()	204
7.61 STMicroMicrocontroller Class Reference	204
7.61.1 Detailed Description	206
7.62 UncertainBoolean Class Reference	206
7.62.1 Detailed Description	206
7.63 Xilinx3DFPGABitstream Class Reference	207
7.63.1 Detailed Description	209
7.64 Xilinx7SeriesDevice Class Reference	209
7.64.1 Detailed Description	212
7.64.2 Member Enumeration Documentation	212
7.64.2.1 instructions	212
7.64.3 Constructor & Destructor Documentation	212
7.64.3.1 Xilinx7SeriesDevice()	213
7.64.4 Member Function Documentation	213
7.64.4.1 Erase()	213
7.64.4.2 GetDescription()	213
7.64.4.3 GetSerialNumber()	214
7.64.4.4 GetSerialNumberLength()	214
7.64.4.5 GetSerialNumberLengthBits()	214
7.64.4.6 IsProgrammed()	215
7.64.4.7 LoadFirmwareImage()	215
7.64.4.8 ParseBitstreamInternals()	215
7.64.4.9 Program()	216
7.64.4.10 ReadWordConfigRegister()	216
7.65 Xilinx7SeriesDeviceConfigurationFrame Union Reference	217
7.65.1 Detailed Description	218
7.65.2 Member Data Documentation	218
7.65.2.1 op	218

7.65.2.2	type	218
7.66	Xilinx7SeriesDeviceStatusRegister Union Reference	218
7.66.1	Detailed Description	219
7.67	XilinxCoolRunnerIIDevice Class Reference	220
7.67.1	Detailed Description	222
7.67.2	Member Enumeration Documentation	222
7.67.2.1	instructions	222
7.67.2.2	packages	223
7.67.3	Member Function Documentation	223
7.67.3.1	Erase()	223
7.67.3.2	GeneratePermutationTable()	224
7.67.3.3	GetDescription()	224
7.67.3.4	GetPaddingSize()	224
7.67.3.5	GetShiftRegisterDepth()	224
7.67.3.6	GetShiftRegisterWidth()	224
7.67.3.7	IsProgrammed()	225
7.67.3.8	LoadFirmwareImage()	225
7.67.3.9	PostInitProbes()	225
7.67.3.10	Program()	226
7.68	XilinxCoolRunnerIIDeviceStatusRegister Union Reference	226
7.68.1	Detailed Description	227
7.69	XilinxCPLD Class Reference	227
7.69.1	Detailed Description	228
7.70	XilinxCPLDBitstream Class Reference	228
7.70.1	Detailed Description	229
7.71	XilinxDevice Class Reference	230
7.71.1	Detailed Description	230
7.71.2	Member Function Documentation	230
7.71.2.1	CreateDevice()	230
7.72	XilinxFPGA Class Reference	231

7.72.1	Detailed Description	232
7.72.2	Constructor & Destructor Documentation	232
7.72.2.1	XilinxFPGA()	232
7.72.3	Member Function Documentation	233
7.72.3.1	ParseBitstreamCore()	233
7.72.3.2	ParseBitstreamInternals()	233
7.72.3.3	PostInitProbes()	234
7.72.3.4	ReadingSerialRequiresReset()	234
7.73	XilinxFPGABitstream Class Reference	235
7.73.1	Detailed Description	237
7.74	XilinxSpartan3ADevice Class Reference	237
7.74.1	Detailed Description	240
7.74.2	Member Enumeration Documentation	240
7.74.2.1	deviceids	240
7.74.2.2	instructions	240
7.74.3	Constructor & Destructor Documentation	241
7.74.3.1	XilinxSpartan3ADevice()	241
7.74.4	Member Function Documentation	241
7.74.4.1	Erase()	241
7.74.4.2	GetDescription()	241
7.74.4.3	GetSerialNumber()	242
7.74.4.4	GetSerialNumberLength()	242
7.74.4.5	GetSerialNumberLengthBits()	243
7.74.4.6	IsProgrammed()	243
7.74.4.7	LoadFirmwareImage()	243
7.74.4.8	ParseBitstreamInternals()	244
7.74.4.9	Program()	244
7.74.4.10	ReadWordConfigRegister()	245
7.75	XilinxSpartan3ADeviceConfigurationFrame Union Reference	245
7.75.1	Detailed Description	246

7.75.2	Member Data Documentation	246
7.75.2.1	count	246
7.75.2.2	op	246
7.75.2.3	type	246
7.76	XilinxSpartan3ADeviceStatusRegister Union Reference	247
7.76.1	Detailed Description	247
7.77	XilinxSpartan6Device Class Reference	248
7.77.1	Detailed Description	251
7.77.2	Member Enumeration Documentation	251
7.77.2.1	deviceids	251
7.77.2.2	instructions	251
7.77.3	Constructor & Destructor Documentation	252
7.77.3.1	XilinxSpartan6Device()	252
7.77.4	Member Function Documentation	252
7.77.4.1	Erase()	252
7.77.4.2	GetDescription()	252
7.77.4.3	GetSerialNumber()	253
7.77.4.4	GetSerialNumberLength()	253
7.77.4.5	GetSerialNumberLengthBits()	254
7.77.4.6	IsProgrammed()	254
7.77.4.7	LoadFirmwareImage()	254
7.77.4.8	ParseBitstreamInternals()	255
7.77.4.9	Program()	255
7.77.4.10	ReadWordConfigRegister()	256
7.77.4.11	ReadWordsConfigRegister()	256
7.78	XilinxSpartan6DeviceConfigurationFrame Union Reference	257
7.78.1	Detailed Description	257
7.78.2	Member Data Documentation	257
7.78.2.1	count	258
7.78.2.2	op	258

7.78.2.3	type	258
7.79	XilinxSpartan6DeviceStatusRegister Union Reference	258
7.79.1	Detailed Description	259
7.80	XilinxUltrascaleDevice Class Reference	260
7.80.1	Detailed Description	263
7.80.2	Member Enumeration Documentation	264
7.80.2.1	cmd_values	264
7.80.2.2	instructions	264
7.80.3	Constructor & Destructor Documentation	264
7.80.3.1	XilinxUltrascaleDevice()	264
7.80.4	Member Function Documentation	265
7.80.4.1	Erase()	265
7.80.4.2	GetDescription()	265
7.80.4.3	GetSerialNumber()	266
7.80.4.4	GetSerialNumberLength()	266
7.80.4.5	GetSerialNumberLengthBits()	266
7.80.4.6	IsProgrammed()	267
7.80.4.7	LoadFirmwareImage()	267
7.80.4.8	ParseBitstreamInternals()	267
7.80.4.9	Program()	268
7.80.4.10	ReadWordConfigRegister()	268
7.81	XilinxUltrascaleDeviceConfigurationFrame Union Reference	269
7.81.1	Detailed Description	270
7.81.2	Member Data Documentation	270
7.81.2.1	op	270
7.81.2.2	type	270
7.82	XilinxUltrascaleDeviceStatusRegister Union Reference	271
7.82.1	Detailed Description	272

8	File Documentation	273
8.1	ARM7TDMISProcessor.cpp File Reference	273
8.1.1	Detailed Description	273
8.2	ARM7TDMISProcessor.h File Reference	273
8.2.1	Detailed Description	274
8.3	ARMAPBDevice.cpp File Reference	274
8.3.1	Detailed Description	274
8.4	ARMAPBDevice.h File Reference	274
8.4.1	Detailed Description	275
8.5	ARMCoreSightDevice.cpp File Reference	275
8.5.1	Detailed Description	275
8.6	ARMCoreSightDevice.h File Reference	275
8.6.1	Detailed Description	276
8.7	ARMCortexA57.cpp File Reference	276
8.7.1	Detailed Description	276
8.8	ARMCortexA57.h File Reference	276
8.8.1	Detailed Description	277
8.9	ARMCortexA9.cpp File Reference	277
8.9.1	Detailed Description	277
8.10	ARMCortexA9.h File Reference	277
8.10.1	Detailed Description	278
8.11	ARMCortexM4.cpp File Reference	278
8.11.1	Detailed Description	278
8.12	ARMCortexM4.h File Reference	278
8.12.1	Detailed Description	279
8.13	ARMDebugAccessPort.cpp File Reference	279
8.13.1	Detailed Description	279
8.14	ARMDebugAccessPort.h File Reference	279
8.14.1	Detailed Description	280
8.14.2	Variable Documentation	280

8.14.2.1	reserved_zero	280
8.14.2.2	revision	281
8.14.2.3	type	281
8.14.2.4	variant	281
8.14.2.5	word	281
8.15	ARMDebugMemAccessPort.cpp File Reference	282
8.15.1	Detailed Description	282
8.16	ARMDebugMemAccessPort.h File Reference	282
8.16.1	Detailed Description	283
8.16.2	Variable Documentation	283
8.16.2.1	mode	284
8.17	ARMDebugPeripheralIDRegister.h File Reference	284
8.17.1	Detailed Description	285
8.18	ARMDebugPort.h File Reference	285
8.18.1	Detailed Description	285
8.19	ARMDevice.cpp File Reference	285
8.19.1	Detailed Description	286
8.20	ARMDevice.h File Reference	286
8.20.1	Detailed Description	286
8.21	ARMFlashPatchBreakpoint.cpp File Reference	286
8.21.1	Detailed Description	287
8.22	ARMFlashPatchBreakpoint.h File Reference	287
8.22.1	Detailed Description	287
8.23	ARMJtagDebugPort.cpp File Reference	287
8.23.1	Detailed Description	287
8.24	ARMJtagDebugPort.h File Reference	288
8.24.1	Detailed Description	289
8.25	ARMv7MProcessor.cpp File Reference	289
8.25.1	Detailed Description	290
8.26	ARMv7MProcessor.h File Reference	290

8.26.1 Detailed Description	290
8.27 ARMv7Processor.cpp File Reference	290
8.27.1 Detailed Description	291
8.28 ARMv7Processor.h File Reference	291
8.28.1 Detailed Description	293
8.28.2 Enumeration Type Documentation	293
8.28.2.1 ARMDebugArchVersion	293
8.28.3 Variable Documentation	294
8.28.3.1 reserved	294
8.29 ARMv8Processor.cpp File Reference	294
8.29.1 Detailed Description	294
8.30 ARMv8Processor.h File Reference	294
8.30.1 Detailed Description	295
8.31 AttachedMemoryDevice.h File Reference	295
8.31.1 Detailed Description	295
8.32 ByteArrayFirmwareImage.cpp File Reference	295
8.32.1 Detailed Description	296
8.33 ByteArrayFirmwareImage.h File Reference	296
8.33.1 Detailed Description	296
8.34 CPLD.cpp File Reference	296
8.34.1 Detailed Description	296
8.35 CPLD.h File Reference	297
8.35.1 Detailed Description	297
8.36 CPLDBitstream.cpp File Reference	297
8.36.1 Detailed Description	297
8.37 CPLDBitstream.h File Reference	297
8.37.1 Detailed Description	298
8.38 DebuggableDevice.cpp File Reference	298
8.38.1 Detailed Description	298
8.39 DebuggableDevice.h File Reference	298

8.39.1 Detailed Description	299
8.40 DebuggerInterface.cpp File Reference	299
8.40.1 Detailed Description	299
8.41 DebuggerInterface.h File Reference	299
8.41.1 Detailed Description	300
8.42 DigilentJtagInterface.cpp File Reference	300
8.42.1 Detailed Description	300
8.43 DigilentJtagInterface.h File Reference	300
8.43.1 Detailed Description	301
8.44 FirmwareImage.cpp File Reference	301
8.44.1 Detailed Description	301
8.45 FirmwareImage.h File Reference	301
8.45.1 Detailed Description	302
8.46 FPGA.cpp File Reference	302
8.46.1 Detailed Description	302
8.47 FPGA.h File Reference	302
8.47.1 Detailed Description	302
8.48 FPGABitstream.cpp File Reference	303
8.48.1 Detailed Description	303
8.49 FPGABitstream.h File Reference	303
8.49.1 Detailed Description	303
8.50 FreescaleDevice.cpp File Reference	303
8.50.1 Detailed Description	304
8.51 FreescaleDevice.h File Reference	304
8.51.1 Detailed Description	304
8.52 FreescaleIMXDevice.cpp File Reference	304
8.52.1 Detailed Description	304
8.53 FreescaleIMXDevice.h File Reference	305
8.53.1 Detailed Description	305
8.54 FreescaleIMXSmartDMA.cpp File Reference	306

8.54.1 Detailed Description	306
8.55 FreescaleIMXSmartDMA.h File Reference	306
8.55.1 Detailed Description	307
8.56 FreescaleMicrocontroller.cpp File Reference	307
8.56.1 Detailed Description	307
8.57 FreescaleMicrocontroller.h File Reference	307
8.57.1 Detailed Description	308
8.58 FTDIJtagInterface.cpp File Reference	308
8.58.1 Detailed Description	308
8.59 FTDIJtagInterface.h File Reference	309
8.59.1 Detailed Description	309
8.60 GPIOInterface.cpp File Reference	309
8.60.1 Detailed Description	310
8.61 GPIOInterface.h File Reference	310
8.61.1 Detailed Description	310
8.62 JtagDevice.cpp File Reference	310
8.62.1 Detailed Description	311
8.63 JtagDevice.h File Reference	311
8.63.1 Detailed Description	311
8.64 JtagDummy.cpp File Reference	311
8.64.1 Detailed Description	312
8.65 JtagDummy.h File Reference	312
8.65.1 Detailed Description	312
8.66 JtagException.cpp File Reference	312
8.66.1 Detailed Description	312
8.67 JtagException.h File Reference	313
8.67.1 Detailed Description	313
8.67.2 Macro Definition Documentation	313
8.67.2.1 JtagExceptionWrapper	313
8.68 JtagFPGA.cpp File Reference	314

8.68.1 Detailed Description	314
8.69 JtagFPGA.h File Reference	314
8.69.1 Detailed Description	314
8.70 jtaghal.cpp File Reference	314
8.70.1 Detailed Description	315
8.70.2 Function Documentation	315
8.70.2.1 GetBigEndianUInt16FromByteArray()	315
8.70.2.2 GetBigEndianUInt32FromByteArray()	316
8.71 jtaghal.h File Reference	316
8.71.1 Detailed Description	318
8.71.2 Function Documentation	318
8.71.2.1 GetBigEndianUInt16FromByteArray()	318
8.71.2.2 GetBigEndianUInt32FromByteArray()	318
8.72 JtagInterface.cpp File Reference	319
8.72.1 Detailed Description	319
8.73 JtagInterface.h File Reference	319
8.73.1 Detailed Description	319
8.74 LockableDevice.cpp File Reference	319
8.74.1 Detailed Description	320
8.75 LockableDevice.h File Reference	320
8.75.1 Detailed Description	320
8.76 MicrochipDevice.cpp File Reference	320
8.76.1 Detailed Description	321
8.77 MicrochipDevice.h File Reference	321
8.77.1 Detailed Description	321
8.78 MicrochipMicrocontroller.cpp File Reference	321
8.78.1 Detailed Description	321
8.79 MicrochipMicrocontroller.h File Reference	322
8.79.1 Detailed Description	322
8.80 MicrochipPIC32Device.cpp File Reference	322

8.80.1 Detailed Description	322
8.81 MicrochipPIC32Device.h File Reference	323
8.81.1 Detailed Description	324
8.82 Microcontroller.cpp File Reference	325
8.82.1 Detailed Description	325
8.83 Microcontroller.h File Reference	325
8.83.1 Detailed Description	325
8.84 NetworkedJtagInterface.cpp File Reference	325
8.84.1 Detailed Description	326
8.85 NetworkedJtagInterface.h File Reference	326
8.85.1 Detailed Description	326
8.86 PipeJtagInterface.cpp File Reference	326
8.86.1 Detailed Description	326
8.87 PipeJtagInterface.h File Reference	327
8.87.1 Detailed Description	327
8.88 ProgrammableDevice.cpp File Reference	327
8.88.1 Detailed Description	327
8.89 ProgrammableDevice.h File Reference	327
8.89.1 Detailed Description	328
8.90 ProgrammableLogicDevice.cpp File Reference	328
8.90.1 Detailed Description	328
8.91 ProgrammableLogicDevice.h File Reference	328
8.91.1 Detailed Description	329
8.92 RawBinaryFirmwareImage.cpp File Reference	329
8.92.1 Detailed Description	329
8.93 RawBinaryFirmwareImage.h File Reference	329
8.93.1 Detailed Description	330
8.94 SerialNumberedDevice.cpp File Reference	330
8.94.1 Detailed Description	330
8.95 SerialNumberedDevice.h File Reference	330

8.95.1 Detailed Description	331
8.96 STM32Device.cpp File Reference	331
8.96.1 Detailed Description	331
8.97 STM32Device.h File Reference	331
8.97.1 Detailed Description	332
8.98 STMicroDevice.cpp File Reference	332
8.98.1 Detailed Description	332
8.99 STMicroDevice.h File Reference	332
8.99.1 Detailed Description	333
8.100STMicroMicrocontroller.cpp File Reference	333
8.100.1 Detailed Description	333
8.101STMicroMicrocontroller.h File Reference	333
8.101.1 Detailed Description	333
8.102Xilinx3DFPGABitstream.cpp File Reference	334
8.102.1 Detailed Description	334
8.103Xilinx3DFPGABitstream.h File Reference	334
8.103.1 Detailed Description	334
8.104Xilinx7SeriesDevice.cpp File Reference	334
8.104.1 Detailed Description	335
8.105Xilinx7SeriesDevice.h File Reference	335
8.105.1 Detailed Description	337
8.105.2 Variable Documentation	337
8.105.2.1 count	337
8.105.2.2 done	337
8.105.2.3 gts_cfg_b	337
8.105.2.4 gwe	337
8.105.2.5 init_b	338
8.105.2.6 mmcm_lock	338
8.105.2.7 op	338
8.105.2.8 type	339

8.105.2.9 word	339
8.106XilinxCoolRunnerIIDevice.cpp File Reference	339
8.106.1 Detailed Description	339
8.107XilinxCoolRunnerIIDevice.h File Reference	340
8.107.1 Detailed Description	341
8.108XilinxCPLD.cpp File Reference	341
8.108.1 Detailed Description	341
8.109XilinxCPLD.h File Reference	342
8.109.1 Detailed Description	342
8.110XilinxCPLDBitstream.cpp File Reference	342
8.110.1 Detailed Description	342
8.111XilinxCPLDBitstream.h File Reference	343
8.111.1 Detailed Description	343
8.112XilinxDevice.cpp File Reference	343
8.112.1 Detailed Description	344
8.113XilinxDevice.h File Reference	344
8.113.1 Detailed Description	344
8.114XilinxFPGA.cpp File Reference	344
8.114.1 Detailed Description	344
8.115XilinxFPGA.h File Reference	345
8.115.1 Detailed Description	345
8.116XilinxFPGABitstream.cpp File Reference	345
8.116.1 Detailed Description	345
8.117XilinxFPGABitstream.h File Reference	345
8.117.1 Detailed Description	346
8.118XilinxSpartan3ADevice.cpp File Reference	346
8.118.1 Detailed Description	346
8.119XilinxSpartan3ADevice.h File Reference	346
8.119.1 Detailed Description	348
8.119.2 Variable Documentation	348

8.119.2.1 count	349
8.119.2.2 op	349
8.119.2.3 type	349
8.119.2.4 word	349
8.120 XilinxSpartan6Device.cpp File Reference	350
8.120.1 Detailed Description	350
8.121 XilinxSpartan6Device.h File Reference	350
8.121.1 Detailed Description	351
8.121.2 Variable Documentation	352
8.121.2.1 count	352
8.121.2.2 op	352
8.121.2.3 type	352
8.121.2.4 word	353
8.122 XilinxUltrascaleDevice.cpp File Reference	353
8.122.1 Detailed Description	353
8.123 XilinxUltrascaleDevice.h File Reference	353
8.123.1 Detailed Description	355
8.123.2 Variable Documentation	355
8.123.2.1 op	355
8.123.2.2 type	356
8.123.2.3 word	356
Index	357

Chapter 1

jtaghal

JTAG Hardware Abstraction Library

Provides an object-oriented interface to JTAG-accessible devices, JTAG adapters, and so on. Intended to be easily extensible so that applications can add support for custom in-system debug features on FPGAs, etc.

This repo contains the library only, and no client application code or standalone build system.

Most users should probably use [azonenberg/jtaghal-cmake](#) and not this repo.

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

JTAG interface layer	19
Stuff not in another group yet	20

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

ARMAPBDevice	33
ARMCoreSightDevice	35
ARMFlashPatchBreakpoint	52
ARMv7MProcessor	62
ARMCortexM4	40
ARMv7Processor	64
ARMCortexA9	38
ARMv8Processor	67
ARMCortexA57	36
ARMDebugAccessPort	41
ARMDebugMemAccessPort	43
ARMDebugMemAPControlStatusWord	44
ARMDebugPeripheralIDRegister	45
ARMDebugPeripheralIDRegisterBits	46
ARMDebugPortIDRegister	49
ARMJtagDebugPortStatusRegister	58
ARMv7DebugIDRegister	59
ARMv7DebugStatusControlRegister	60
AttachedMemoryDevice	69
DebuggableDevice	76
ARM7TDMISProcessor	29
ARMv7MProcessor	62
ARMv7Processor	64
ARMv8Processor	67
DebuggerInterface	77
ARMDebugPort	47
ARMJtagDebugPort	54
EjtagControlRegister	84
EjtagImplementationCodeRegister	85
FirmwareImage	86
ByteArrayFirmwareImage	70
FPGABitstream	88
XilinxFPGABitstream	235

Xilinx3DFPGABitstream	207
RawBinaryFirmwareImage	192
CPLDBitstream	74
XilinxCPLDBitstream	228
GPIOInterface	114
FTDIJtagInterface	101
NetworkedJtagInterface	166
JtagDevice	116
ARMDevice	50
ARM7TDMISProcessor	29
ARMJtagDebugPort	54
FreescaleDevice	91
FreescaleIMXSmartDMA	97
FreescaleMicrocontroller	100
FreescaleIMXDevice	94
JtagDummy	123
JtagFPGA	128
XilinxFPGA	231
Xilinx7SeriesDevice	209
XilinxSpartan3ADevice	237
XilinxSpartan6Device	248
XilinxUltrascaleDevice	260
MicrochipDevice	153
MicrochipMicrocontroller	156
MicrochipPIC32Device	157
STM32Device	196
XilinxCPLD	227
XilinxCoolRunnerIIDevice	220
JtagException	126
JtagInterface	130
DiligentJtagInterface	79
FTDIJtagInterface	101
NetworkedJtagInterface	166
PipeJtagInterface	177
LockableDevice	151
STM32Device	196
MicrochipPIC32DeviceInfo	163
MicrochipPIC32DeviceStatusRegister	163
ProgrammableDevice	188
ARM7TDMISProcessor	29
Microcontroller	164
FreescaleMicrocontroller	100
MicrochipMicrocontroller	156
STMicroMicrocontroller	204
STM32Device	196
ProgrammableLogicDevice	191
CPLD	72
XilinxCPLD	227
FPGA	87
JtagFPGA	128
SerialNumberedDevice	194
STM32Device	196
XilinxFPGA	231
STMicroDevice	203
STMicroMicrocontroller	204

UncertainBoolean	206
Xilinx7SeriesDeviceConfigurationFrame	217
Xilinx7SeriesDeviceStatusRegister	218
XilinxCoolRunnerIIDeviceStatusRegister	226
XilinxDevice	230
XilinxCPLD	227
XilinxFPGA	231
XilinxSpartan3ADeviceConfigurationFrame	245
XilinxSpartan3ADeviceStatusRegister	247
XilinxSpartan6DeviceConfigurationFrame	257
XilinxSpartan6DeviceStatusRegister	258
XilinxUltrascaleDeviceConfigurationFrame	269
XilinxUltrascaleDeviceStatusRegister	271

Chapter 4

Class Index

4.1 Class List

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

ARM7TDMISProcessor	An ARM7TDMI-S CPU core supporting the ARMv4 architecture, as seen over JTAG (no Core↔Sight support)	29
ARMAPBDevice	A device attached to an ARM APB bus (may be a debug core or something else)	33
ARMCoreSightDevice	Base class for ARM CoreSight components (other than CPU cores) on a debug APB bus	35
ARMCortexA57	An ARM Cortex-A57 CPU core, as seen over a CoreSight APB bus	36
ARMCortexA9	An ARM Cortex-A9 CPU core, as seen over a CoreSight APB bus	38
ARMCortexM4	An ARM Cortex-M4 CPU core, as seen over a CoreSight APB bus	40
ARMDebugAccessPort	Base class for all access ports within an ARMDebugPort	41
ARMDebugMemAccessPort	A bridge from an ARMDebugPort to an ARM memory bus	43
ARMDebugMemAPControlStatusWord	Contents of the CSW register in a MEM-AP (see ADIV5 Architecture Specification 7.6.4)	44
ARMDebugPeripheralIDRegister	ADI component ID register	45
ARMDebugPeripheralIDRegisterBits	ADI component ID register bitfield	46
ARMDebugPort	Base class for ARM debug ports (JTAG-DP, SWJ-DP, etc)	47
ARMDebugPortIDRegister	ARM debug port identification register (see ADIV5 Architecture Specification figure 6-3)	49
ARMDevice	Abstract base class for all ARM Ltd JTAG devices (ADIV5 DAP or legacy CPUs with their own JTAG TAPs)	50
ARMFlashPatchBreakpoint	Cortex-M Flash Patch/Breakpoint Unit (see ARMv7-M architecture ref C1.11)	52
ARMJtagDebugPort	An ARM JTAG-DP (contains one or more APs and a DP)	54
ARMJtagDebugPortStatusRegister	ARM debug port status register (see ADIV5 Architecture Specification figure 6-3)	58

ARMv7DebugIDRegister	ARM debug ID register (see ARMv7 Architecture Reference Manual, C11.11.15)	59
ARMv7DebugStatusControlRegister	ARM debug status/control register (see ARMv7 Architecture Reference Manual, C11.11.20)	60
ARMv7MProcessor	An ARMv7 Cortex-M CPU core, as seen over a CoreSight APB bus	62
ARMv7Processor	An ARMv7 Cortex-A CPU core, as seen over a CoreSight APB bus	64
ARMv8Processor	An ARMv8 Cortex-A CPU core, as seen over a CoreSight APB bus	67
AttachedMemoryDevice	Base classes for devices which can connect to external memory devices	69
ByteArrayFirmwareImage	Generic base class for all firmware images consisting of an array of bytes	70
CPLD	Generic base class for all complex programmable logic devices	72
CPLDBitstream	Abstract base class for CPLD configuration bitstreams	74
DebuggableDevice	Generic base class for all debuggable devices (CPU cores etc)	76
DebuggerInterface	Generic base class for all debugger interfaces (may connect to multiple DebuggableDevice 's in a SoC)	77
DigilentJtagInterface	A JTAG adapter exposed through the Digilent Adept SDK	79
EjtagControlRegister	PIC32 EJTAG control register	84
EjtagImplementationCodeRegister	MIPS EJTAG implementation register	85
FirmwareImage	Generic base class for all firmware images for any kind of programmable device	86
FPGA	Generic base class for all field-programmable gate array devices	87
FPGABitstream	Abstract base class for FPGA configuration bitstreams	88
FreescaleDevice	Abstract base class for all Freescale devices (typically MCUs or parts thereof)	91
FreescaleIMXDevice	A Freescale i.mx applications processor	94
FreescaleIMXSmartDMA	The SDMA in a Freescale i.mx SoC (Chapter 55 of i.mx6 reference manual)	97
FreescaleMicrocontroller	Generic base class for all Freescale MCUs	100
FTDIJtagInterface	A JTAG adapter using the FTDI chipset, accessed through libftd2xx (proprietary driver from F↔TDI)	101
GPIOInterface	A GPIO bitbang interface. Many JTAG adapters have uncommitted GPIOs which may be used for test purposes	114
JtagDevice	A single TAP in the JTAG chain. May not correspond 1:1 with physical silicon dies	116
JtagDummy	An unknown device (IDCODE not recognized, or no IDCODE present) on a JTAG chain	123
JtagException	Base class for all exceptions thrown by libjtaghal	126
JtagFPGA	Abstract base class for all JTAG-programmed FPGAs	128

JtagInterface	Abstract representation of a JTAG adapter	130
LockableDevice	Generic base class for all devices which have some kind of read/write protection	151
MicrochipDevice	Abstract base class for all Microchip devices (typically MCUs)	153
MicrochipMicrocontroller	Generic base class for all Microchip MCUs	156
MicrochipPIC32Device	A Microchip PIC32 microcontroller (MX, MZ, MM, etc)	157
MicrochipPIC32DeviceInfo	Internal data structure storing properties of a single SKU in the PIC32 family	163
MicrochipPIC32DeviceStatusRegister	Status register for a Microchip PIC32 device	163
Microcontroller	Generic base class for all microcontrollers	164
NetworkedJtagInterface	Thin wrapper around TCP sockets for talking to a jtagd instance	166
PipeJtagInterface	Thin wrapper around pipes for talking to an openfpga JtagPipeBridge	177
ProgrammableDevice	Generic base class for all programmable devices (PLD, MCU, flash, etc)	188
ProgrammableLogicDevice	Generic base class for all programmable logic devices (FPGA and CPLD)	191
RawBinaryFirmwareImage	Raw binary firmware image loaded from a file	192
SerialNumberedDevice	Abstract base class for all devices that have a unique die serial number	194
STM32Device	A STM32 microcontroller	196
STMicroDevice	Abstract base class for all STMicro devices	203
STMicroMicrocontroller	Generic base class for all STMicro MCUs	204
UncertainBoolean	A boolean value with an attached level of uncertainty	206
Xilinx3DFPGABitstream	A bitstream for Xilinx 3D FPGAs (multiple dies on a passive interposer, each with their own bitstream)	207
Xilinx7SeriesDevice	A Xilinx 7-series FPGA device	209
Xilinx7SeriesDeviceConfigurationFrame	7-series configuration frame (see UG470 page 87)	217
Xilinx7SeriesDeviceStatusRegister	7-series status register (see UG470 table 5-28)	218
XilinxCoolRunnerIIDevice	A Xilinx CoolRunner-II device	220
XilinxCoolRunnerIIDeviceStatusRegister	Status register for a Xilinx CoolRunner-II device	226
XilinxCPLD	Generic base class for all Xilinx CPLD devices	227
XilinxCPLDBitstream	A bitstream for Xilinx CPLDs	228
XilinxDevice	Abstract base class for all Xilinx devices (FPGA, CPLD, flash, etc)	230
XilinxFPGA	Abstract base class for all Xilinx FPGAs	231

XilinxFPGABitstream	
A bitstream for Xilinx FPGAs	235
XilinxSpartan3ADevice	
A Xilinx Spartan-3A FPGA device	237
XilinxSpartan3ADeviceConfigurationFrame	
Spartan-3A configuration frame header (see UG332 page 323)	245
XilinxSpartan3ADeviceStatusRegister	
Spartan-3A status register (see UG332 table 17-13, pages 327-328)	247
XilinxSpartan6Device	
A Xilinx Spartan-6 FPGA device	248
XilinxSpartan6DeviceConfigurationFrame	
Spartan-6 configuration frame (see UG380 page 91)	257
XilinxSpartan6DeviceStatusRegister	
Spartan-6 status register (see UG380 table 5-35)	258
XilinxUltrascaleDevice	
A Xilinx Ultrascale or Ultrascale+ FPGA device	260
XilinxUltrascaleDeviceConfigurationFrame	
UltraScale configuration frame (see UG570 page 158)	269
XilinxUltrascaleDeviceStatusRegister	
UltraScale status register (see UG570 table 9-25)	271

Chapter 5

File Index

5.1 File List

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

ARM7TDMISProcessor.cpp	
Implementation of ARM7TDMISProcessor	273
ARM7TDMISProcessor.h	
Declaration of ARM7TDMISProcessor	273
ARMAPBDevice.cpp	
Implementation of ARMAPBDevice	274
ARMAPBDevice.h	
Declaration of ARMAPBDevice	274
ARMCoreSightDevice.cpp	
Base class for ARM CoreSight components on a debug APB bus	275
ARMCoreSightDevice.h	
Declaration of ARMCoreSightDevice	275
ARMCortexA57.cpp	
Implementation of ARMCortexA57	276
ARMCortexA57.h	
Declaration of ARMCortexA57	276
ARMCortexA9.cpp	
Implementation of ARMCortexA9	277
ARMCortexA9.h	
Declaration of ARMCortexA9	277
ARMCortexM4.cpp	
Implementation of ARMCortexM4	278
ARMCortexM4.h	
Declaration of ARMCortexM4	278
ARMDebugAccessPort.cpp	
Implementation of ARMDebugAccessPort	279
ARMDebugAccessPort.h	
Declaration of ARMDebugAccessPort	279
ARMDebugMemAccessPort.cpp	
Implementation of ARMDebugMemAccessPort	282
ARMDebugMemAccessPort.h	
Declaration of ARMDebugMemAccessPort	282
ARMDebugPeripheralIDRegister.h	
Declaration of ARMDebugPeripheralIDRegister	284
ARMDebugPort.h	
Declaration of ARMDebugPort	285

ARMDevice.cpp	Implementation of ARMDevice	285
ARMDevice.h	Declaration of ARMDevice	286
ARMFlashPatchBreakpoint.cpp	ARM Cortex-M Flash Patch/Breakpoint	286
ARMFlashPatchBreakpoint.h	Declaration of ARMFlashPatchBreakpoint	287
ARMJtagDebugPort.cpp	Implementation of ARMJtagDebugPort	287
ARMJtagDebugPort.h	Declaration of ARMJtagDebugPort	288
ARMv7MPProcessor.cpp	Implementation of ARMv7MPProcessor	289
ARMv7MPProcessor.h	Declaration of ARMv7MPProcessor	290
ARMv7Processor.cpp	Implementation of ARMv7Processor	290
ARMv7Processor.h	Declaration of ARMv7Processor	291
ARMv8Processor.cpp	Implementation of ARMv8Processor	294
ARMv8Processor.h	Declaration of ARMv8Processor	294
AttachedMemoryDevice.h	Declaration of AttachedMemoryDevice	295
ByteArrayFirmwareImage.cpp	Implementation of ByteArrayFirmwareImage	295
ByteArrayFirmwareImage.h	Declaration of ByteArrayFirmwareImage	296
CPLD.cpp	Implementation of CPLD	296
CPLD.h	Declaration of CPLD	297
CPLDBitstream.cpp	Implementation of CPLDBitstream	297
CPLDBitstream.h	Declaration of CPLDBitstream	297
DebuggableDevice.cpp	Implementation of DebuggableDevice	298
DebuggableDevice.h	Declaration of DebuggableDevice	298
DebuggerInterface.cpp	Implementation of DebuggerInterface	299
DebuggerInterface.h	Declaration of DebuggerInterface	299
DiligentJtagInterface.cpp	Implementation of DiligentJtagInterface	300
DiligentJtagInterface.h	Declaration of DiligentJtagInterface	300
FirmwareImage.cpp	Implementation of FirmwareImage	301
FirmwareImage.h	Declaration of FirmwareImage	301
FPGA.cpp	Implementation of FPGA	302
FPGA.h	Declaration of FPGA	302

FPGABitstream.cpp	Implementation of FPGABitstream	303
FPGABitstream.h	Declaration of FPGABitstream	303
FreescaleDevice.cpp	Implementation of FreescaleDevice	303
FreescaleDevice.h	Declaration of FreescaleDevice	304
FreescaleIMXDevice.cpp	Implementation of FreescaleIMXDevice	304
FreescaleIMXDevice.h	Declaration of FreescaleIMXDevice	305
FreescaleIMXSmartDMA.cpp	Implementation of FreescaleIMXSmartDMA	306
FreescaleIMXSmartDMA.h	Declaration of FreescaleIMXSmartDMA	306
FreescaleMicrocontroller.cpp	Implementation of FreescaleMicrocontroller	307
FreescaleMicrocontroller.h	Declaration of FreescaleMicrocontroller	307
FTDIJtagInterface.cpp	Implementation of FTDIJtagInterface	308
FTDIJtagInterface.h	Declaration of FTDIJtagInterface	309
GPIOInterface.cpp	Implementation of GPIOInterface	309
GPIOInterface.h	Declaration of GPIOInterface	310
JEDECVendorID_enum.h		??
jtagd_opcodes_enum.h		??
JtagDevice.cpp	Implementation of JtagDevice	310
JtagDevice.h	Declaration of JtagDevice	311
JtagDummy.cpp	Implementation of JtagDummy	311
JtagDummy.h	Declaration of JtagDummy	312
JtagException.cpp	Implementation of JtagException	312
JtagException.h	Declaration of JtagException	313
JtagFPGA.cpp	Implementation of JtagFPGA	314
JtagFPGA.h	Declaration of JtagFPGA	314
jtaghal.cpp	Implementation of global functions	314
jtaghal.h	Main library include file	316
JtagInterface.cpp	Implementation of JtagInterface	319
JtagInterface.h	Declaration of JtagInterface	319
LockableDevice.cpp	Implementation of LockableDevice	319
LockableDevice.h	Declaration of LockableDevice	320

MicrochipDevice.cpp	Implementation of MicrochipDevice	320
MicrochipDevice.h	Declaration of MicrochipDevice	321
MicrochipMicrocontroller.cpp	Implementation of MicrochipMicrocontroller	321
MicrochipMicrocontroller.h	Declaration of MicrochipMicrocontroller	322
MicrochipPIC32Device.cpp	Implementation of MicrochipPIC32Device	322
MicrochipPIC32Device.h	Declaration of MicrochipPIC32Device	323
Microcontroller.cpp	Implementation of Microcontroller	325
Microcontroller.h	Declaration of Microcontroller	325
NetworkedJtagInterface.cpp	Implementation of NetworkedJtagInterface	325
NetworkedJtagInterface.h	Declaration of NetworkedJtagInterface	326
PipeJtagInterface.cpp	Implementation of PipeJtagInterface	326
PipeJtagInterface.h	Declaration of PipeJtagInterface	327
ProgrammableDevice.cpp	Implementation of ProgrammableDevice	327
ProgrammableDevice.h	Declaration of ProgrammableDevice	327
ProgrammableLogicDevice.cpp	Implementation of ProgrammableLogicDevice	328
ProgrammableLogicDevice.h	Declaration of ProgrammableLogicDevice	328
RawBinaryFirmwareImage.cpp	Implementation of RawBinaryFirmwareImage	329
RawBinaryFirmwareImage.h	Declaration of RawBinaryFirmwareImage	329
SerialNumberedDevice.cpp	Implementation of SerialNumberedDevice	330
SerialNumberedDevice.h	Declaration of SerialNumberedDevice	330
STM32Device.cpp	Implementation of STM32Device	331
STM32Device.h	Declaration of STM32Device	331
STMicroDevice.cpp	Implementation of STMicroDevice	332
STMicroDevice.h	Declaration of STMicroDevice	332
STMicroDeviceID_enum.h	??
STMicroMicrocontroller.cpp	Implementation of STMicroMicrocontroller	333
STMicroMicrocontroller.h	Declaration of STMicroMicrocontroller	333
UserPID_enum.h	??
UserVID_enum.h	??
Xilinx3DFPGABitstream.cpp	Implementation of Xilinx3DFPGABitstream	334

Xilinx3DFPGABitstream.h	
Declaration of Xilinx3DFPGABitstream	334
Xilinx7SeriesDevice.cpp	
Implementation of Xilinx7SeriesDevice	334
Xilinx7SeriesDevice.h	
Declaration of Xilinx7SeriesDevice	335
XilinxCoolRunnerIIDevice.cpp	
Implementation of XilinxCoolRunnerIIDevice	339
XilinxCoolRunnerIIDevice.h	
Declaration of XilinxCoolRunnerIIDevice	340
XilinxCPLD.cpp	
Implementation of XilinxCPLD	341
XilinxCPLD.h	
Declaration of XilinxCPLD	342
XilinxCPLDBitstream.cpp	
Implementation of XilinxCPLDBitstream	342
XilinxCPLDBitstream.h	
Declaration of XilinxCPLDBitstream	343
XilinxDevice.cpp	
Implementation of XilinxDevice	343
XilinxDevice.h	
Declaration of XilinxDevice	344
XilinxDeviceID_enum.h	??
XilinxFPGA.cpp	
Implementation of XilinxFPGA	344
XilinxFPGA.h	
Declaration of XilinxFPGA	345
XilinxFPGABitstream.cpp	
Implementation of XilinxFPGABitstream	345
XilinxFPGABitstream.h	
Declaration of XilinxFPGABitstream	345
XilinxSpartan3ADevice.cpp	
Implementation of XilinxSpartan3ADevice	346
XilinxSpartan3ADevice.h	
Declaration of XilinxSpartan3ADevice	346
XilinxSpartan6Device.cpp	
Implementation of XilinxSpartan6Device	350
XilinxSpartan6Device.h	
Declaration of XilinxSpartan6Device	350
XilinxUltrascaleDevice.cpp	
Implementation of XilinxUltrascaleDevice	353
XilinxUltrascaleDevice.h	
Declaration of XilinxUltrascaleDevice	353

Chapter 6

Module Documentation

6.1 JTAG interface layer

Classes

- class [DigilentJtagInterface](#)
A JTAG adapter exposed through the Digilent Adept SDK.
- class [FTDIJtagInterface](#)
A JTAG adapter using the FTDI chipset, accessed through libftd2xx (proprietary driver from FTDI)
- class [JtagInterface](#)
Abstract representation of a JTAG adapter.
- class [NetworkedJtagInterface](#)
Thin wrapper around TCP sockets for talking to a jtagd instance.
- class [PipeJtagInterface](#)
Thin wrapper around pipes for talking to an openfpga JtagPipeBridge.

6.1.1 Detailed Description

The JTAG interface layer exposes many different JTAG hardware devices as a simple C++ API.

6.2 Stuff not in another group yet

Classes

- class [ARM7TDMISProcessor](#)
An ARM7TDMI-S CPU core supporting the ARMv4 architecture, as seen over JTAG (no CoreSight support)
- class [ARMAPBDevice](#)
A device attached to an ARM APB bus (may be a debug core or something else)
- class [ARMCoreSightDevice](#)
Base class for ARM CoreSight components (other than CPU cores) on a debug APB bus.
- class [ARMCortexA57](#)
An ARM Cortex-A57 CPU core, as seen over a CoreSight APB bus.
- class [ARMCortexA9](#)
An ARM Cortex-A9 CPU core, as seen over a CoreSight APB bus.
- class [ARMCortexM4](#)
An ARM Cortex-M4 CPU core, as seen over a CoreSight APB bus.
- class [ARMDebugAccessPort](#)
Base class for all access ports within an [ARMDebugPort](#).
- class [ARMDebugMemAccessPort](#)
A bridge from an [ARMDebugPort](#) to an ARM memory bus.
- class [ARMDebugPort](#)
Base class for ARM debug ports (JTAG-DP, SWJ-DP, etc)
- class [ARMDevice](#)
Abstract base class for all ARM Ltd JTAG devices (ADiv5 DAP or legacy CPUs with their own JTAG TAPs)
- class [ARMFlashPatchBreakpoint](#)
Cortex-M Flash Patch/Breakpoint Unit (see ARMv7-M architecture ref C1.11)
- class [ARMJtagDebugPort](#)
An ARM JTAG-DP (contains one or more APs and a DP)
- class [ARMv7MProcessor](#)
An ARMv7 Cortex-M CPU core, as seen over a CoreSight APB bus.
- class [ARMv7Processor](#)
An ARMv7 Cortex-A CPU core, as seen over a CoreSight APB bus.
- class [ARMv8Processor](#)
An ARMv8 Cortex-A CPU core, as seen over a CoreSight APB bus.
- class [AttachedMemoryDevice](#)
Base classes for devices which can connect to external memory devices.
- class [ByteArrayFirmwareImage](#)
Generic base class for all firmware images consisting of an array of bytes.
- class [CPLD](#)
Generic base class for all complex programmable logic devices.
- class [CPLDBitstream](#)
Abstract base class for [CPLD](#) configuration bitstreams.
- class [DebuggableDevice](#)
Generic base class for all debuggable devices (CPU cores etc)
- class [DebuggerInterface](#)
Generic base class for all debugger interfaces (may connect to multiple [DebuggableDevice](#)'s in a SoC)
- class [FirmwareImage](#)
Generic base class for all firmware images for any kind of programmable device.
- class [FPGA](#)
Generic base class for all field-programmable gate array devices.

- class [FPGABitstream](#)
Abstract base class for [FPGA](#) configuration bitstreams.
- class [FreescaleDevice](#)
Abstract base class for all Freescale devices (typically MCUs or parts thereof)
- class [FreescaleIMXDevice](#)
A Freescale i.mx applications processor.
- class [FreescaleIMXSmartDMA](#)
The SDMA in a Freescale i.mx SoC (Chapter 55 of i.mx6 reference manual)
- class [FreescaleMicrocontroller](#)
Generic base class for all Freescale MCUs.
- class [JtagDevice](#)
A single TAP in the JTAG chain. May not correspond 1:1 with physical silicon dies.
- class [JtagDummy](#)
An unknown device (IDCODE not recognized, or no IDCODE present) on a JTAG chain.
- class [JtagException](#)
Base class for all exceptions thrown by libjtaghal.
- class [JtagFPGA](#)
Abstract base class for all JTAG-programmed FPGAs.
- class [LockableDevice](#)
Generic base class for all devices which have some kind of read/write protection.
- class [MicrochipDevice](#)
Abstract base class for all Microchip devices (typically MCUs)
- class [MicrochipMicrocontroller](#)
Generic base class for all Microchip MCUs.
- union [MicrochipPIC32DeviceStatusRegister](#)
Status register for a Microchip PIC32 device.
- union [EjtagImplementationCodeRegister](#)
MIPS EJTAG implementation register.
- class [MicrochipPIC32Device](#)
A Microchip PIC32 microcontroller (MX, MZ, MM, etc)
- class [Microcontroller](#)
Generic base class for all microcontrollers.
- class [ProgrammableDevice](#)
Generic base class for all programmable devices (PLD, MCU, flash, etc)
- class [ProgrammableLogicDevice](#)
Generic base class for all programmable logic devices ([FPGA](#) and [CPLD](#))
- class [RawBinaryFirmwareImage](#)
Raw binary firmware image loaded from a file.
- class [SerialNumberedDevice](#)
Abstract base class for all devices that have a unique die serial number.
- class [STM32Device](#)
A STM32 microcontroller.
- class [STMicroDevice](#)
Abstract base class for all STMicro devices.
- class [STMicroMicrocontroller](#)
Generic base class for all STMicro MCUs.
- class [Xilinx3DFPGABitstream](#)
A bitstream for Xilinx 3D FPGAs (multiple dies on a passive interposer, each with their own bitstream)
- union [Xilinx7SeriesDeviceConfigurationFrame](#)
7-series configuration frame (see UG470 page 87)
- union [Xilinx7SeriesDeviceStatusRegister](#)

- 7-series status register (see UG470 table 5-28)
- class [Xilinx7SeriesDevice](#)
 - A Xilinx 7-series *FPGA* device.
- union [XilinxCoolRunnerIIDeviceStatusRegister](#)
 - Status register for a Xilinx CoolRunner-II device.
- class [XilinxCoolRunnerIIDevice](#)
 - A Xilinx CoolRunner-II device.
- class [XilinxCPLD](#)
 - Generic base class for all Xilinx *CPLD* devices.
- class [XilinxCPLDBitstream](#)
 - A bitstream for Xilinx *CPLDs*.
- class [XilinxDevice](#)
 - Abstract base class for all Xilinx devices (*FPGA*, *CPLD*, flash, etc)
- class [XilinxFPGA](#)
 - Abstract base class for all Xilinx *FPGAs*.
- class [XilinxFPGABitstream](#)
 - A bitstream for Xilinx *FPGAs*.
- union [XilinxSpartan3ADeviceConfigurationFrame](#)
 - Spartan-3A configuration frame header (see UG332 page 323)
- union [XilinxSpartan3ADeviceStatusRegister](#)
 - Spartan-3A status register (see UG332 table 17-13, pages 327-328)
- class [XilinxSpartan3ADevice](#)
 - A Xilinx Spartan-3A *FPGA* device.
- union [XilinxSpartan6DeviceConfigurationFrame](#)
 - Spartan-6 configuration frame (see UG380 page 91)
- union [XilinxSpartan6DeviceStatusRegister](#)
 - Spartan-6 status register (see UG380 table 5-35)
- class [XilinxSpartan6Device](#)
 - A Xilinx Spartan-6 *FPGA* device.
- union [XilinxUltrascaleDeviceConfigurationFrame](#)
 - UltraScale configuration frame (see UG570 page 158)
- union [XilinxUltrascaleDeviceStatusRegister](#)
 - UltraScale status register (see UG570 table 9-25)
- class [XilinxUltrascaleDevice](#)
 - A Xilinx Ultrascale or Ultrascale+ *FPGA* device.

Functions

- bool [PeekBit](#) (const unsigned char *data, int nbit)
 - Extracts a bit from a bit string.
- void [PokeBit](#) (unsigned char *data, int nbit, bool val)
 - Writes a bit to a bit string.
- unsigned char [FlipByte](#) (unsigned char c)
 - Flips the bits in a byte.
- void [FlipByteArray](#) (unsigned char *data, int len)
 - Reverses an array of bytes in place without changing bit ordering.
- void [FlipBitArray](#) (unsigned char *data, int len)
 - Reverses the bit ordering in an array of bytes, but does not change byte ordering.
- void [MirrorBitArray](#) (unsigned char *data, int bitlen)
 - Reverses the bit ordering in an array of bits (need not be integer byte size)

- void [FlipEndianArray](#) (unsigned char *data, int len)
Swaps endianness in an array of 16-bit values.
- void [FlipEndian32Array](#) (unsigned char *data, int len)
Swaps endianness in an array of 32-bit values.
- void [FlipBitAndEndianArray](#) (unsigned char *data, int len)
Reverses the bit ordering in an array of bytes, as well as 16-bit endianness.
- void [FlipBitAndEndian32Array](#) (unsigned char *data, int len)
Reverses the bit ordering in an array of bytes, as well as 32-bit endianness.
- double [GetTime](#) ()
Returns a timestamp suitable for performance measurement.
- union [MicrochipPIC32DeviceStatusRegister](#) `__attribute__((packed))`

Variables

- class [ARMDebugAccessPort](#) `__attribute__((packed))`
- [ARMDebugMemAccessPort](#) `__attribute__((packed))`
- [ARMDebugPort](#) `__attribute__((packed))`
- [ARMJtagDebugPort](#) `__attribute__((packed))`
- [ARMv7Processor](#) `__attribute__((packed))`
- [Xilinx7SeriesDevice](#) `__attribute__((packed))`
- [XilinxCoolRunnerIIDevice](#) `__attribute__((packed))`
- [XilinxSpartan3ADevice](#) `__attribute__((packed))`
- [XilinxSpartan6Device](#) `__attribute__((packed))`
- [XilinxUltrascaleDevice](#) `__attribute__((packed))`

6.2.1 Detailed Description

6.2.2 Function Documentation

6.2.2.1 FlipBitAndEndian32Array()

```
void FlipBitAndEndian32Array (
    unsigned char * data,
    int len )
```

Reverses the bit ordering in an array of bytes, as well as 32-bit endianness.

Parameters

<i>data</i>	The buffer to manipulate
<i>len</i>	Length, in bytes, of the buffer

6.2.2.2 FlipBitAndEndianArray()

```
void FlipBitAndEndianArray (
    unsigned char * data,
    int len )
```

Reverses the bit ordering in an array of bytes, as well as 16-bit endianness.

Parameters

<i>data</i>	The buffer to manipulate
<i>len</i>	Length, in bytes, of the buffer

6.2.2.3 FlipBitArray()

```
void FlipBitArray (
    unsigned char * data,
    int len )
```

Reverses the bit ordering in an array of bytes, but does not change byte ordering.

Parameters

<i>data</i>	The buffer to manipulate
<i>len</i>	Length, in bytes, of the buffer

6.2.2.4 FlipByte()

```
unsigned char FlipByte (
    unsigned char c )
```

Flips the bits in a byte.

Parameters

<i>c</i>	Input byte
----------	------------

Returns

Output byte

6.2.2.5 FlipByteArray()

```
void FlipByteArray (
    unsigned char * data,
    int len )
```

Reverses an array of bytes in place without changing bit ordering.

Parameters

<i>data</i>	The buffer to manipulate
<i>len</i>	Length, in bytes, of the buffer

6.2.2.6 FlipEndian32Array()

```
void FlipEndian32Array (
    unsigned char * data,
    int len )
```

Swaps endianness in an array of 32-bit values.

Parameters

<i>data</i>	The buffer to manipulate
<i>len</i>	Length, in bytes, of the buffer (must be a multiple of 4)

6.2.2.7 FlipEndianArray()

```
void FlipEndianArray (
    unsigned char * data,
    int len )
```

Swaps endianness in an array of 16-bit values.

Parameters

<i>data</i>	The buffer to manipulate
<i>len</i>	Length, in bytes, of the buffer (must be even)

6.2.2.8 GetTime()

```
double GetTime ( )
```

Returns a timestamp suitable for performance measurement.

The base unit is seconds.

Returns

The timestamp.

6.2.2.9 MirrorBitArray()

```
void MirrorBitArray (
    unsigned char * data,
    int bitlen )
```

Reverses the bit ordering in an array of bits (need not be integer byte size)

Parameters

<i>data</i>	The buffer to manipulate
<i>bitlen</i>	Length, in bits, of the buffer

6.2.2.10 PeekBit()

```
bool PeekBit (
    const unsigned char * data,
    int nbit )
```

Extracts a bit from a bit string.

(data[0] & 1) is considered to be the LSB.

Parameters

<i>data</i>	The bit string
<i>nbit</i>	Index (zero based) of the bit to extract

Returns

Value of the bit

6.2.2.11 PokeBit()

```
void PokeBit (
    unsigned char * data,
```



```
int nbit,  
bool val )
```

Writes a bit to a bit string.

(data[0] & 1) is considered to be the LSB.

Parameters

<i>data</i>	The bit string
<i>nbit</i>	Index (zero based) of the bit to write
<i>val</i>	The value to write at that bit

Chapter 7

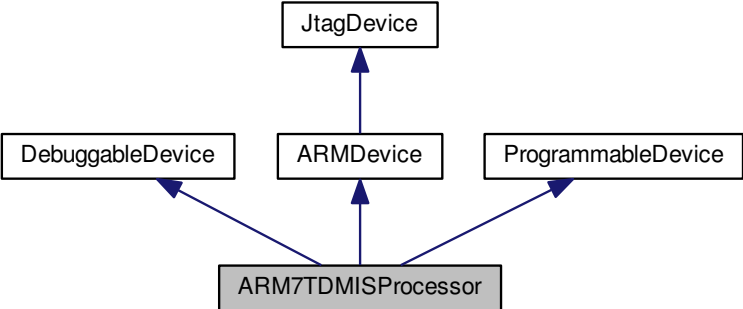
Class Documentation

7.1 ARM7TDMISProcessor Class Reference

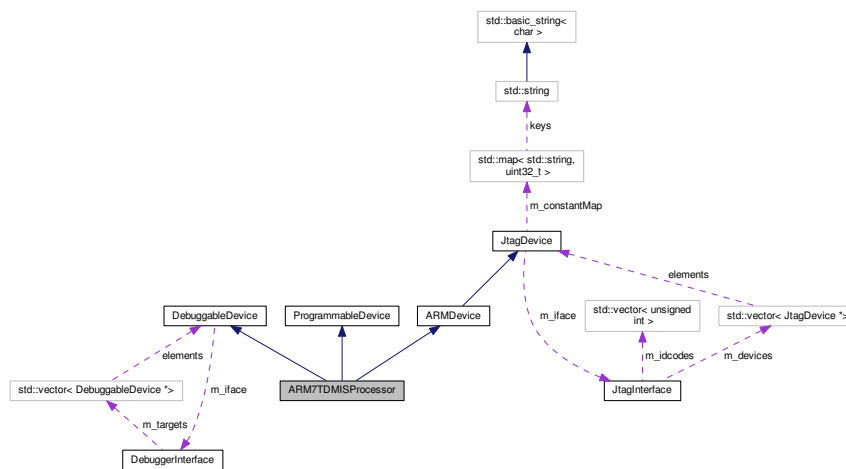
An ARM7TDMI-S CPU core supporting the ARMv4 architecture, as seen over JTAG (no CoreSight support)

```
#include <ARM7TDMISProcessor.h>
```

Inheritance diagram for ARM7TDMISProcessor:



Collaboration diagram for ARM7TDMISProcessor:



Public Types

- enum **JTAG_INSTRUCTIONS** {
SCAN_N = 0x2, **RESTART** = 0x4, **INTEST** = 0xc, **IDCODE** = 0xe,
BYPASS = 0xf }

Public Member Functions

- **ARM7TDMISProcessor** (unsigned int [partnum](#), unsigned int rev, unsigned int idcode, [JtagInterface](#) *iface, size_t pos)
- virtual void [PostInitProbes](#) (bool quiet)
 - Does a post-initialization probe of the device to read debug ROMs etc.*
- virtual std::string [GetDescription](#) ()
- virtual void [PrintInfo](#) ()
- virtual bool [IsProgrammed](#) ()
 - Determines if this device is programmed or blank.*
- virtual [FirmwareImage](#) * [LoadFirmwareImage](#) (const unsigned char *data, size_t len)
 - Parses an in-memory image of a firmware image into a format suitable for loading into the device.*
- virtual void [Erase](#) ()
 - Erases the device configuration and restores the device to a blank state.*
- virtual void [Program](#) ([FirmwareImage](#) *image)
 - Loads a new firmware image onto the device.*
- virtual void [DebugHalt](#) ()
 - Halts the CPU and enters debug state.*
- virtual void [DebugResume](#) ()
- virtual void [PrintRegisters](#) ()
- virtual uint32_t [ReadMemory](#) (uint32_t addr)
 - Checks if the CPU is halted due to a fatal error.*
- virtual void [WriteMemory](#) (uint32_t addr, uint32_t value)

Public Attributes

- unsigned int **m_rev**
- unsigned int **m_selectedChain**

Protected Types

- enum **IceRegisters** {
DEBUG_CTRL = 0x00, **DEBUG_STAT** = 0x01, **DCC_CTRL** = 0x04, **DCC_DATA** = 0x05,
WATCH0_ADDR = 0x08, **WATCH0_AMASK** = 0x09, **WATCH0_DATA** = 0x0a, **WATCH0_DMASK** = 0x0b,
WATCH0_CTRL = 0x0c, **WATCH0_CMASK** = 0x0d, **WATCH1_ADDR** = 0x10, **WATCH1_AMASK** = 0x11,
WATCH1_DATA = 0x12, **WATCH1_DMASK** = 0x13, **WATCH1_CTRL** = 0x14, **WATCH1_CMASK** = 0x15 }

Protected Member Functions

- void **SelectScanChain** (uint8_t n)
- void **SelectDebugChain** ()
- void **SelectIceRTChain** ()
- void **WriteIceRegister** (uint8_t reg, uint32_t value)
- uint32_t **ReadIceRegister** (uint8_t reg)

Additional Inherited Members

7.1.1 Detailed Description

An ARM7TDMI-S CPU core supporting the ARMv4 architecture, as seen over JTAG (no CoreSight support)

7.1.2 Member Function Documentation

7.1.2.1 Erase()

```
void ARM7TDMISProcessor::Erase ( ) [virtual]
```

Erases the device configuration and restores the device to a blank state.

After this function is called, regardless of success or failure, all existing connections to on-chip code become invalid.

Exceptions

<i>JtagException</i>	if the erase operation fails
----------------------	------------------------------

Implements [ProgrammableDevice](#).

7.1.2.2 IsProgrammed()

```
bool ARM7TDMISProcessor::IsProgrammed ( ) [virtual]
```

Determines if this device is programmed or blank.

Returns

true if programmed, false if blank

Implements [ProgrammableDevice](#).

7.1.2.3 LoadFirmwareImage()

```
FirmwareImage * ARM7TDMISProcessor::LoadFirmwareImage (
    const unsigned char * data,
    size_t len ) [virtual]
```

Parses an in-memory image of a firmware image into a format suitable for loading into the device.

Exceptions

JtagException	if the image is malformed
-------------------------------	---------------------------

Parameters

<i>data</i>	Pointer to the start of the firmware image, including headers
<i>len</i>	Length of the firmware image

Returns

Pointer to an [FirmwareImage](#) object suitable for passing to [Configure\(\)](#).

Implements [ProgrammableDevice](#).

7.1.2.4 PostInitProbes()

```
void ARM7TDMISProcessor::PostInitProbes (
    bool quiet ) [virtual]
```

Does a post-initialization probe of the device to read debug ROMs etc.

Parameters

<i>quiet</i>	Do minimal probing to avoid triggering security lockdowns
--------------	---

Implements [JtagDevice](#).

7.1.2.5 Program()

```
void ARM7TDMISProcessor::Program (
    FirmwareImage * image ) [virtual]
```

Loads a new firmware image onto the device.

After this function is called, regardless of success or failure, all existing connections to on-chip code become invalid.

Exceptions

JtagException	if the erase operation fails
-------------------------------	------------------------------

Parameters

<i>image</i>	The parsed image to load
--------------	--------------------------

Implements [ProgrammableDevice](#).

The documentation for this class was generated from the following files:

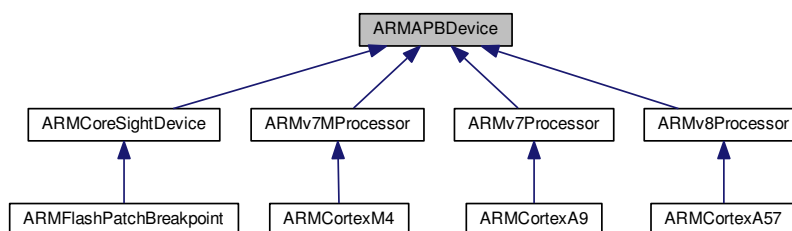
- [ARM7TDMISProcessor.h](#)
- [ARM7TDMISProcessor.cpp](#)

7.2 ARMAPBDevice Class Reference

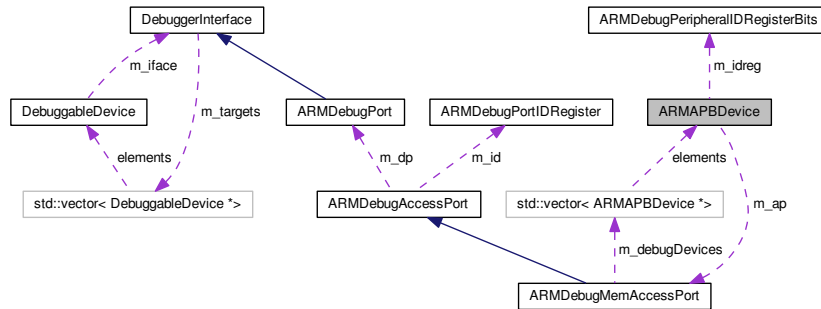
A device attached to an ARM APB bus (may be a debug core or something else)

```
#include <ARMAPBDevice.h>
```

Inheritance diagram for ARMAPBDevice:



Collaboration diagram for ARMAPBDevice:



Public Member Functions

- **ARMAPBDevice** ([ARMDebugMemAccessPort](#) *ap, uint32_t address, [ARMDebugPeripheralIDRegisterBits](#) idreg)
- virtual std::string **GetDescription** ()=0
- virtual void **PrintInfo** ()=0

Protected Member Functions

- uint32_t **ReadRegisterByOffset** (uint32_t offset)
reads a register given the offset from our base address
- uint32_t **ReadRegisterByIndex** (uint32_t index)
reads a register given the index into a 32-bit register space
- void **WriteRegisterByIndex** (uint32_t index, uint32_t value)
writes a register given the index into a 32-bit register space
- void **WriteRegisterByOffset** (uint32_t offset, uint32_t value)
writes a register given the offset from our base address

Protected Attributes

- [ARMDebugMemAccessPort](#) * m_ap
The Mem-AP.
- [ARMDebugPeripheralIDRegisterBits](#) m_idreg
The peripheral ID register.
- uint32_t m_address
Base address of the device.

7.2.1 Detailed Description

A device attached to an ARM APB bus (may be a debug core or something else)

The documentation for this class was generated from the following files:

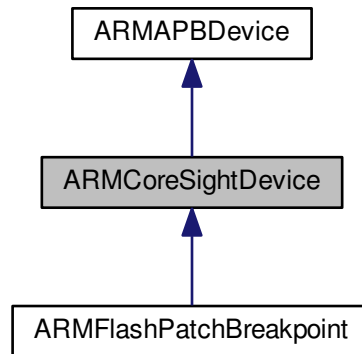
- [ARMAPBDevice.h](#)
- [ARMAPBDevice.cpp](#)

7.3 ARMCoresightDevice Class Reference

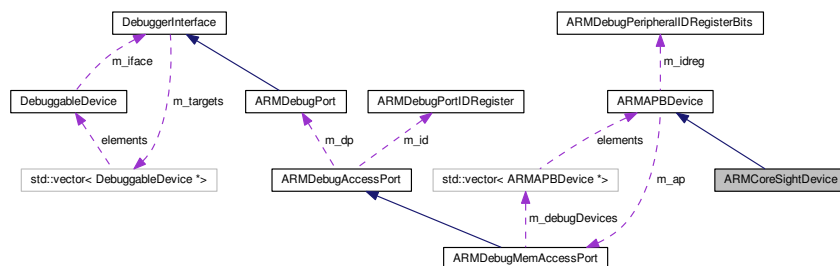
Base class for ARM CoreSight components (other than CPU cores) on a debug APB bus.

```
#include <ARMCoresightDevice.h>
```

Inheritance diagram for ARMCoresightDevice:



Collaboration diagram for ARMCoresightDevice:



Public Member Functions

- **ARMCoresightDevice** ([ARMDebugMemAccessPort](#) *ap, uint32_t address, [ARMDebugPeripheralIDRegisterBits](#) idreg)
- virtual std::string **GetDescription** ()
- virtual void **PrintInfo** ()

Additional Inherited Members

7.3.1 Detailed Description

Base class for ARM CoreSight components (other than CPU cores) on a debug APB bus.

The documentation for this class was generated from the following files:

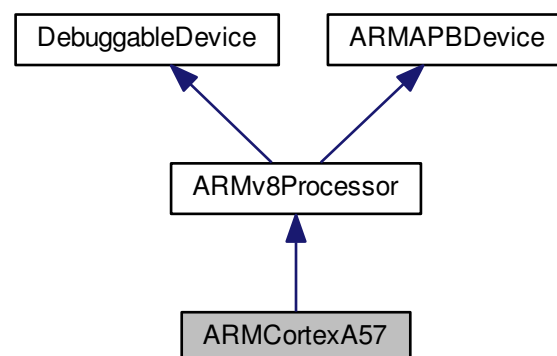
- [ARMCoresightDevice.h](#)
- [ARMCoresightDevice.cpp](#)

7.4 ARMCortexA57 Class Reference

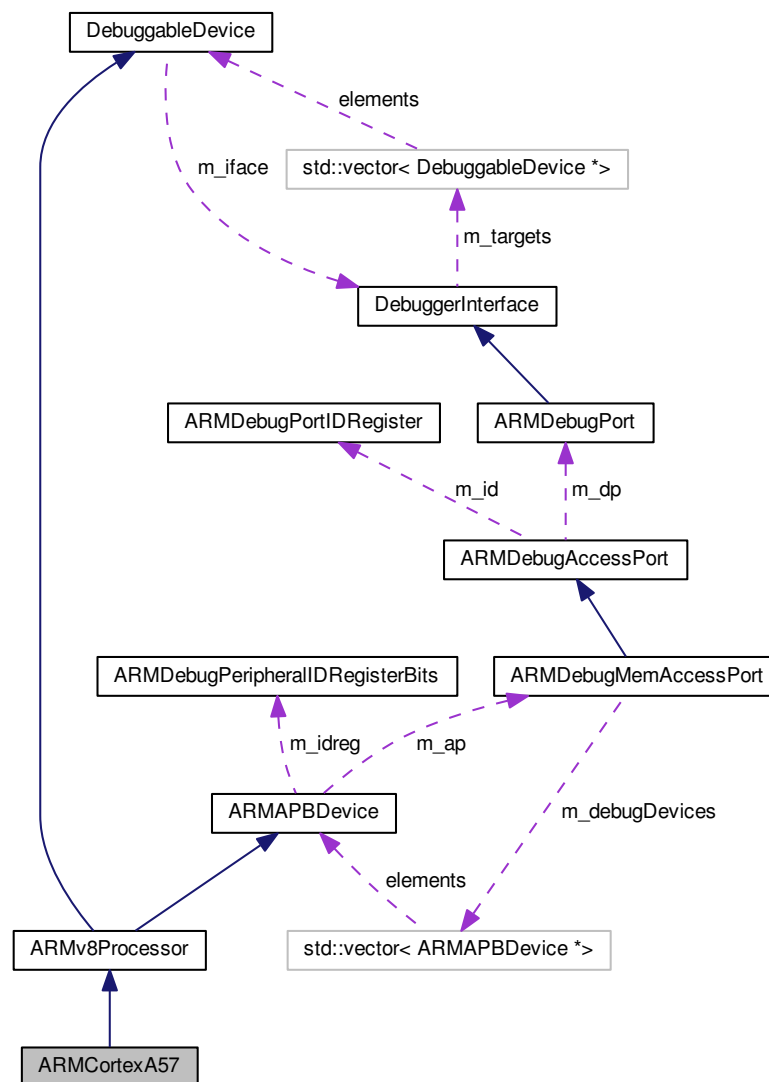
An ARM Cortex-A57 CPU core, as seen over a CoreSight APB bus.

```
#include <ARMCortexA57.h>
```

Inheritance diagram for ARMCortexA57:



Collaboration diagram for ARMCortexA57:



Public Member Functions

- **ARMCortexA57** ([DebuggerInterface](#) *iface, [ARMDebugMemAccessPort](#) *ap, uint32_t address, [ARMDebugPeripheralIDRegisterBits](#) idreg)
- virtual std::string **GetDescription** ()
- virtual void **PrintInfo** ()

Additional Inherited Members

7.4.1 Detailed Description

An ARM Cortex-A57 CPU core, as seen over a CoreSight APB bus.

The documentation for this class was generated from the following files:

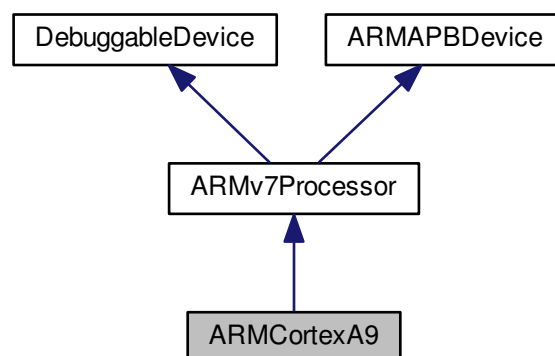
- [ARMCortexA57.h](#)
- [ARMCortexA57.cpp](#)

7.5 ARMCortexA9 Class Reference

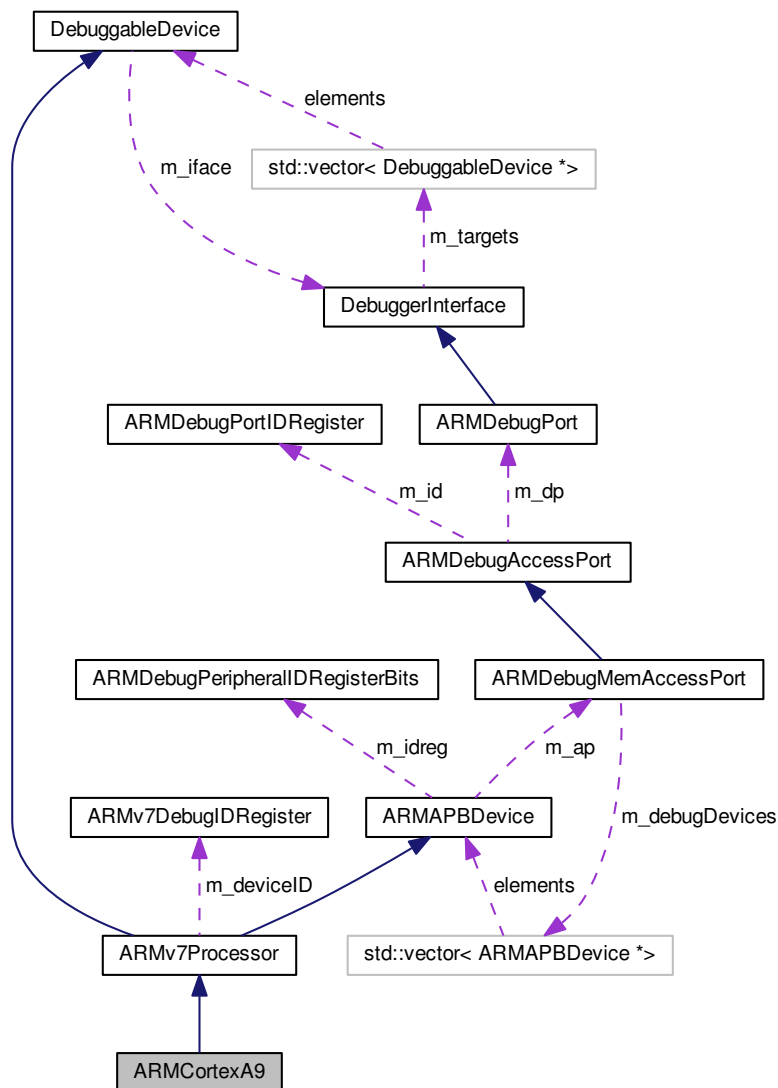
An ARM Cortex-A9 CPU core, as seen over a CoreSight APB bus.

```
#include <ARMCortexA9.h>
```

Inheritance diagram for ARMCortexA9:



Collaboration diagram for ARMCortexA9:



Public Member Functions

- **ARMCortexA9** ([DebuggerInterface](#) *iface, [ARMDDebugMemAccessPort](#) *ap, uint32_t address, [ARMDDebugPeripheralIDRegisterBits](#) idreg)
- virtual std::string **GetDescription** ()
- virtual void **PrintInfo** ()
- uint32_t **SampleProgramCounter** ()
Sample program counter (for sample-based profiling)

Additional Inherited Members

7.5.1 Detailed Description

An ARM Cortex-A9 CPU core, as seen over a CoreSight APB bus.

The documentation for this class was generated from the following files:

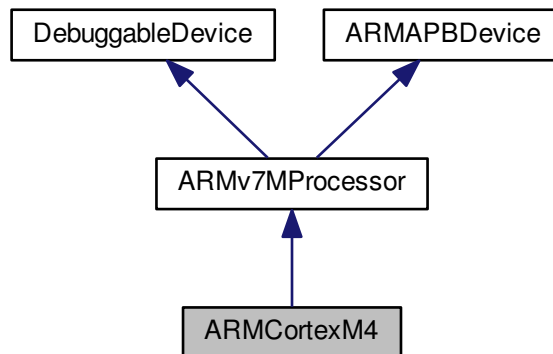
- [ARMCortexA9.h](#)
- [ARMCortexA9.cpp](#)

7.6 ARMCortexM4 Class Reference

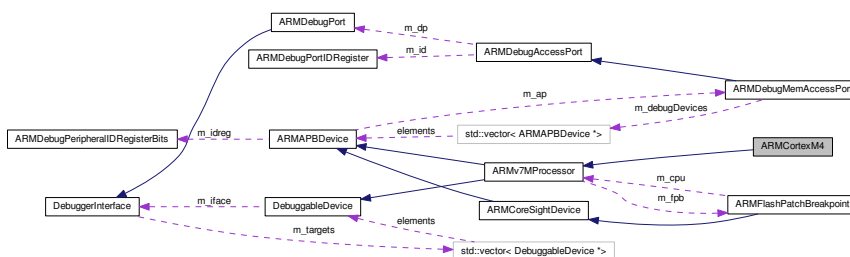
An ARM Cortex-M4 CPU core, as seen over a CoreSight APB bus.

```
#include <ARMCortexM4.h>
```

Inheritance diagram for ARMCortexM4:



Collaboration diagram for ARMCortexM4:



Public Member Functions

- **ARMCortexM4** ([DebuggerInterface](#) *iface, [ARMDebugMemAccessPort](#) *ap, uint32_t address, [ARMDebugPeripheralIDRegisterBits](#) idreg)
- virtual std::string **GetDescription** ()
- virtual void **PrintInfo** ()

Additional Inherited Members

7.6.1 Detailed Description

An ARM Cortex-M4 CPU core, as seen over a CoreSight APB bus.

The documentation for this class was generated from the following files:

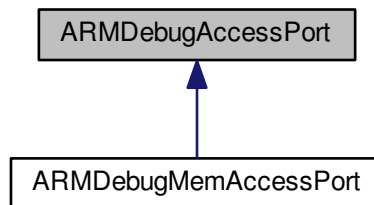
- [ARMCortexM4.h](#)
- [ARMCortexM4.cpp](#)

7.7 ARMDebugAccessPort Class Reference

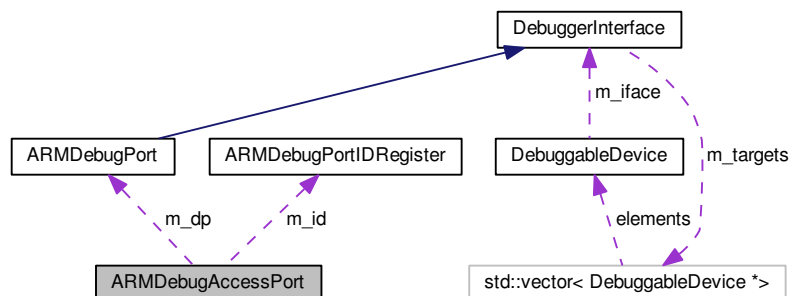
Base class for all access ports within an [ARMDebugPort](#).

```
#include <ARMDebugAccessPort.h>
```

Inheritance diagram for ARMDebugAccessPort:



Collaboration diagram for ARMDebugAccessPort:



Public Types

- enum **dap_type** {
 DAP_JTAG = 0, **DAP_AHB** = 1, **DAP_APB** = 2, **DAP_AXI** = 4,
 DAP_INVALID }

Public Member Functions

- **ARMDebugAccessPort** ([ARMDebugPort](#) *dp, uint8_t apnum, [ARMDebugPortIDRegister](#) id)
- virtual void **Initialize** ()=0
- dap_type **GetBusType** ()
- unsigned int **GetVersion** ()
- virtual void **PrintStatusRegister** ()=0
- virtual bool **IsEnabled** ()=0
- virtual std::string **GetDescription** ()=0
- [ARMDebugPort](#) * **GetDebugPort** ()
- uint8_t **GetAPNumber** ()

Public Attributes

- enum [ARMDebugAccessPort::dap_type](#) **__attribute__**

Protected Attributes

- [ARMDebugPort](#) * **m_dp**
- uint8_t **m_apnum**
- [ARMDebugPortIDRegister](#) **m_id**
- dap_type **m_daptype**

7.7.1 Detailed Description

Base class for all access ports within an [ARMDebugPort](#).

The documentation for this class was generated from the following files:

- [ARMDebugAccessPort.h](#)
- [ARMDebugAccessPort.cpp](#)

Public Member Functions

- **ARMDebugMemAccessPort** ([ARMDebugPort](#) *dp, uint8_t apnum, [ARMDebugPortIDRegister](#) id)
- virtual void **Initialize** ()
- uint32_t **ReadWord** (uint32_t addr)
- void **WriteWord** (uint32_t addr, uint32_t value)
- virtual void **PrintStatusRegister** ()
- virtual std::string **GetDescription** ()
- [ARMDebugMemAPControlStatusWord](#) **GetStatusRegister** ()
- virtual bool **IsEnabled** ()
- uint32_t **GetDebugBaseAddress** ()
- size_t **GetDeviceCount** ()
- [ARMAPBDevice](#) * **GetDevice** (size_t i)

Public Attributes

- enum [ARMDebugMemAccessPort::AccessSize](#) **__attribute__**

Protected Member Functions

- void **FindRootRomTable** ()
- void **LoadROMTable** (uint32_t baseAddress)
- void **ProcessDebugBlock** (uint32_t base_address, uint32_t id_base, [ARMDebugPeripheralIDRegister](#) reg)
Reads the ROM table for a debug block to figure out what's going on.

Protected Attributes

- bool **m_debugBusIsDedicated**
- bool **m_hasDebugRom**
- uint32_t **m_debugBaseAddress**
- std::vector< [ARMAPBDevice](#) * > **m_debugDevices**
The list of devices found on the AP.

7.8.1 Detailed Description

A bridge from an [ARMDebugPort](#) to an ARM memory bus.

The documentation for this class was generated from the following files:

- [ARMDebugMemAccessPort.h](#)
- [ARMDebugMemAccessPort.cpp](#)

7.9 ARMDebugMemAPControlStatusWord Union Reference

Contents of the CSW register in a MEM-AP (see ADIv5 Architecture Specification 7.6.4)

```
#include <ARMDebugMemAccessPort.h>
```

Public Member Functions

- ```

struct {
 unsigned int size:3
 Size of the access to perform.
 unsigned int reserved_zero_1:1
 Reserved, should be zero.
 unsigned int auto_increment:2
 Address increment/pack mode.
 unsigned int enable:1
 Debug port enable (RO)
 unsigned int busy:1
 Transfer in progress.
 unsigned int mode:4
 Operating mode (write as zero, read undefined)
 unsigned int reserved_zero_2:11
 Reserved, should be zero.
 unsigned int secure_priv_debug:1
 Secure privileged debug flag (not sure what this is)
 unsigned int bus_protect:6
 Bus access protection (implementation defined)
 unsigned int nonsecure_transfer:1
 Secure transfer (high=nonsecure)
 unsigned int reserved_zero_3: 1
} __attribute__((packed)) bits

```

## Public Attributes

- `uint32_t word`  
*The raw status register value.*

### 7.9.1 Detailed Description

Contents of the CSW register in a MEM-AP (see ADIv5 Architecture Specification 7.6.4)

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

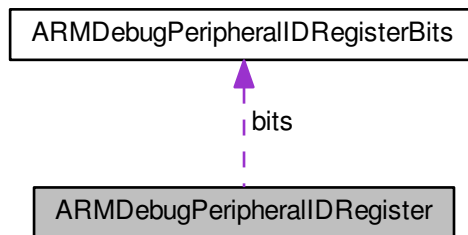
- [ARMDebugMemAccessPort.h](#)

## 7.10 ARMDebugPeripheralIDRegister Union Reference

ADI component ID register.

```
#include <ARMDebugPeripheralIDRegister.h>
```

Collaboration diagram for ARMDebugPeripheralIDRegister:



### Public Attributes

- [ARMDebugPeripheralIDRegisterBits bits](#)  
*The bitfield.*
- `uint64_t word`  
*The raw status register value.*

#### 7.10.1 Detailed Description

ADI component ID register.

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

- [ARMDebugPeripheralIDRegister.h](#)

## 7.11 ARMDebugPeripheralIDRegisterBits Class Reference

ADI component ID register bitfield.

```
#include <ARMDebugPeripheralIDRegister.h>
```

### Public Attributes

- unsigned int `partnum:12`  
*Part number (TODO)*
- unsigned int `jep106_id:7`  
*JEP106 identity code.*
- unsigned int `jep106_used:1`  
*Indicates if JEP106 code is valid.*
- unsigned int `revnum:4`  
*Peripheral revision number.*

- unsigned int [cust\\_mod](#):4  
*Customer modification ID.*
- unsigned int [revand](#):4  
*Manufacturer rev number (stepping)*
- unsigned int [jep106\\_cont](#):4  
*JEP106 continuation code.*
- unsigned int [log\\_4k\\_blocks](#):4  
*Log2(#4K address space blocks)*
- unsigned int [reserved\\_zero](#):24  
*Unmapped.*

### 7.11.1 Detailed Description

ADI component ID register bitfield.

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

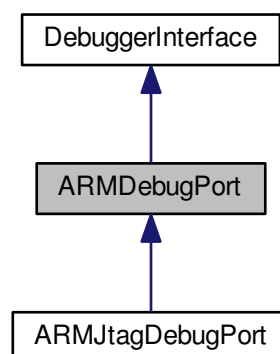
- [ARMDebugPeripheralIDRegister.h](#)

## 7.12 ARMDebugPort Class Reference

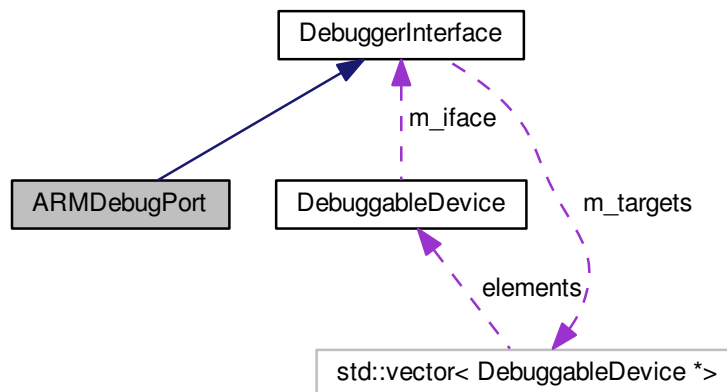
Base class for ARM debug ports (JTAG-DP, SWJ-DP, etc)

```
#include <ARMDebugPort.h>
```

Inheritance diagram for ARMDebugPort:



Collaboration diagram for ARMDebugPort:



## Public Types

- enum **ApReg** {  
**REG\_MEM\_CSW** = 0x00, **REG\_MEM\_TAR** = 0x04, **REG\_MEM\_DRW** = 0x0C, **REG\_MEM\_BASE** = 0xF8,  
**REG\_IDR** = 0xFC }

## Public Member Functions

- virtual void **PrintStatusRegister** ()=0
- virtual uint32\_t **ReadDebugRegister** (uint32\_t address)=0
- virtual void **WriteDebugRegister** (uint32\_t address, uint32\_t value)=0

## Protected Member Functions

- virtual uint32\_t **APRegisterRead** (uint8\_t ap, ApReg addr)=0
- virtual void **APRegisterWrite** (uint8\_t ap, ApReg addr, uint32\_t wdata)=0

## Friends

- class **ARMDebugMemAccessPort**

## Additional Inherited Members

### 7.12.1 Detailed Description

Base class for ARM debug ports (JTAG-DP, SWJ-DP, etc)

The documentation for this class was generated from the following files:

- [ARMDebugPort.h](#)
- [ARMDebugPort.cpp](#)

## 7.13 ARMDebugPortIDRegister Union Reference

ARM debug port identification register (see ADIV5 Architecture Specification figure 6-3)

```
#include <ARMDebugAccessPort.h>
```

### Public Member Functions

- struct {  
    unsigned int [type](#):4  
        *Type of AP.*  
    unsigned int [variant](#):4  
        *Variant of AP.*  
    unsigned int [reserved\\_zero](#):8  
        *Reserved, SBZ.*  
    unsigned int [is\\_mem\\_ap](#):1  
        *Class (1 = mem-AP, 0=not mem-AP)*  
    unsigned int [identity](#):7  
        *Identity code (must be 0x3B)*  
    unsigned int [continuation](#):4  
        *Continuation code (must be 0x4)*  
    unsigned int [revision](#): 4  
        *Revision of the AP design.*  
} **\_\_attribute\_\_((packed))** [bits](#)

### Public Attributes

- [uint32\\_t word](#)  
*The raw status register value.*

#### 7.13.1 Detailed Description

ARM debug port identification register (see ADIV5 Architecture Specification figure 6-3)

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

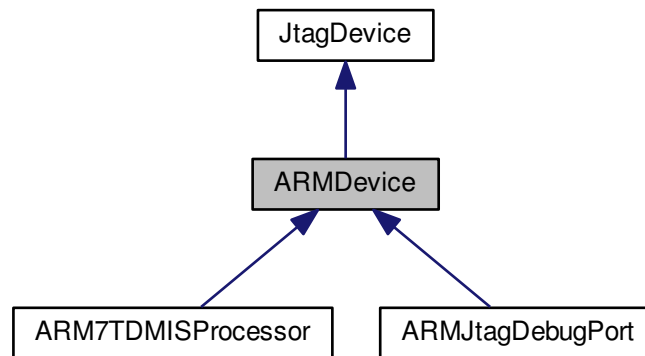
- [ARMDebugAccessPort.h](#)

## 7.14 ARMDevice Class Reference

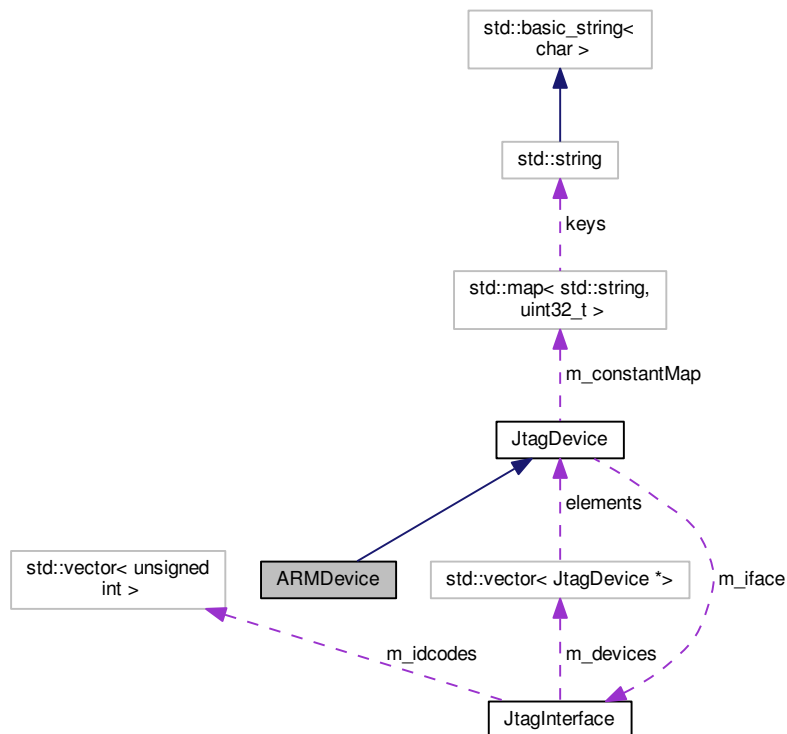
Abstract base class for all ARM Ltd JTAG devices (ADIV5 DAP or legacy CPUs with their own JTAG TAPs)

```
#include <ARMDevice.h>
```

Inheritance diagram for ARMDevice:



Collaboration diagram for ARMDevice:





## Public Member Functions

- [ARMDevice](#) (unsigned int idcode, [JtagInterface](#) \*iface, size\_t pos, size\_t irlength)  
*Initializes this device.*
- virtual [~ARMDevice](#) ()  
*Default virtual destructor.*

## Static Public Member Functions

- static [JtagDevice](#) \* [CreateDevice](#) (unsigned int idcode, [JtagInterface](#) \*iface, size\_t pos)  
*Creates a [ARMDevice](#) given an ID code.*

## Additional Inherited Members

### 7.14.1 Detailed Description

Abstract base class for all ARM Ltd JTAG devices (ADiv5 DAP or legacy CPUs with their own JTAG TAPs)

### 7.14.2 Constructor & Destructor Documentation

#### 7.14.2.1 ARMDevice()

```
ARMDevice::ARMDevice (
 unsigned int idcode,
 JtagInterface * iface,
 size_t pos,
 size_t irlength)
```

Initializes this device.

#### Parameters

|                 |                                                       |
|-----------------|-------------------------------------------------------|
| <i>idcode</i>   | The ID code of this device                            |
| <i>iface</i>    | The JTAG adapter this device was discovered on        |
| <i>pos</i>      | Position in the chain that this device was discovered |
| <i>irlength</i> | Length of the JTAG instruction register               |

### 7.14.3 Member Function Documentation

### 7.14.3.1 CreateDevice()

```
JtagDevice * ARMDevice::CreateDevice (
 unsigned int idcode,
 JtagInterface * iface,
 size_t pos) [static]
```

Creates a [ARMDevice](#) given an ID code.

#### Exceptions

|                               |                                                                                       |
|-------------------------------|---------------------------------------------------------------------------------------|
| <a href="#">JtagException</a> | if the ID code supplied is not a valid Microchip device, or not a known family number |
|-------------------------------|---------------------------------------------------------------------------------------|

#### Parameters

|               |                                                       |
|---------------|-------------------------------------------------------|
| <i>idcode</i> | The ID code of this device                            |
| <i>iface</i>  | The JTAG adapter this device was discovered on        |
| <i>pos</i>    | Position in the chain that this device was discovered |

#### Returns

A valid [JtagDevice](#) object, or NULL if the vendor ID was not recognized.

The documentation for this class was generated from the following files:

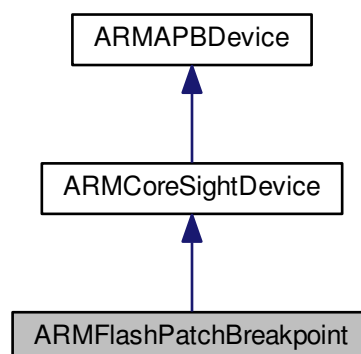
- [ARMDevice.h](#)
- [ARMDevice.cpp](#)

## 7.15 ARMFlashPatchBreakpoint Class Reference

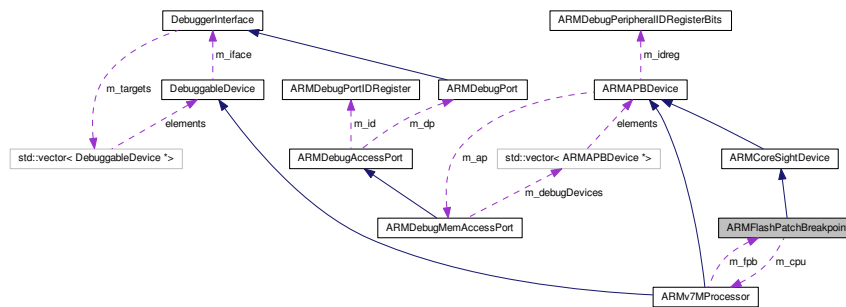
Cortex-M Flash Patch/Breakpoint Unit (see ARMv7-M architecture ref C1.11)

```
#include <ARMFlashPatchBreakpoint.h>
```

Inheritance diagram for ARMFlashPatchBreakpoint:



Collaboration diagram for ARMFlashPatchBreakpoint:



## Public Types

- enum **FpbRegisters** { **FP\_CTRL** = 0, **FP\_REMAP** = 1, **FP\_COMPO** = 2 }

## Public Member Functions

- **ARMFlashPatchBreakpoint** ([ARMv7MProcessor](#) \*cpu, [ARMDebugMemAccessPort](#) \*ap, uint32\_t address, [ARMDebugPeripheralIDRegisterBits](#) idreg)
- virtual std::string **GetDescription** ()
- virtual void **PrintInfo** ()
- uint32\_t **GetCodeComparatorIndex** (uint32\_t i)
- uint32\_t **GetLiteralComparatorIndex** (uint32\_t i)
- uint32\_t **GetCodeComparatorCount** ()
- uint32\_t **GetLiteralComparatorCount** ()
- void **Enable** ()
- void **Disable** ()
- void **SetRemapTableBase** (uint32\_t base)
- void **RemapFlashWord** (uint32\_t slot, uint32\_t flashAddress, uint32\_t newValue)

## Protected Member Functions

- void **ProbeStatusRegisters** ()

## Protected Attributes

- [ARMv7MProcessor](#) \* **m\_cpu**
- uint32\_t **m\_codeComparators**
- uint32\_t **m\_literalComparators**
- bool **m\_enabled**
- bool **m\_canRemap**
- uint32\_t **m\_tableBase**
- uint32\_t **m\_sramBase**

### 7.15.1 Detailed Description

Cortex-M Flash Patch/Breakpoint Unit (see ARMv7-M architecture ref C1.11)

The documentation for this class was generated from the following files:

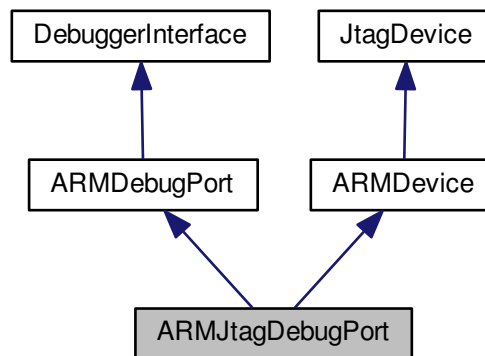
- [ARMFlashPatchBreakpoint.h](#)
- [ARMFlashPatchBreakpoint.cpp](#)

### 7.16 ARMJtagDebugPort Class Reference

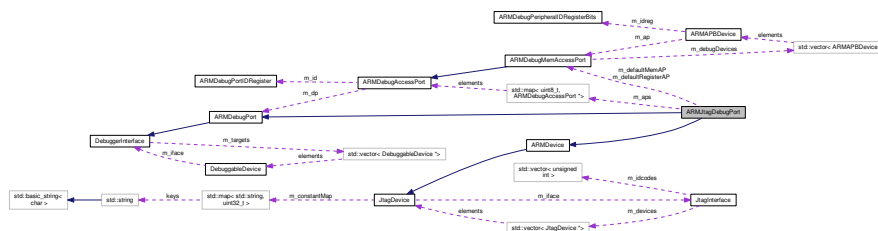
An ARM JTAG-DP (contains one or more APs and a DP)

```
#include <ARMJtagDebugPort.h>
```

Inheritance diagram for ARMJtagDebugPort:



Collaboration diagram for ARMJtagDebugPort:



#### Public Types

- enum **instructions** { **INST\_IDCODE** = 0x0e, **INST\_ABORT** = 0x08, **INST\_DPACC** = 0x0a, **INST\_APACC** = 0x0b }
- enum **DapResult** { **OK\_OR\_FAULT** = 2, **WAIT** = 1 }
- enum **RWFlag** { **OP\_WRITE** = 0, **OP\_READ** = 1 }
- enum **DpReg** { **REG\_CTRL\_STAT** = 1, **REG\_AP\_SELECT** = 2, **REG\_RDBUFF** = 3 }

## Public Member Functions

- **ARMJtagDebugPort** (unsigned int [partnum](#), unsigned int rev, unsigned int idcode, [JtagInterface](#) \*iface, size\_t pos)
- virtual void [PostInitProbes](#) (bool quiet)
 

*Does a post-initialization probe of the device to read debug ROMs etc.*
- virtual std::string [GetDescription](#) ()
 

*Gets a human-readable description of this device.*
- virtual void **PrintInfo** ()
- virtual uint32\_t [ReadMemory](#) (uint32\_t address)
 

*Read a single 32-bit word of memory (TODO support smaller sizes)*
- virtual void [WriteMemory](#) (uint32\_t address, uint32\_t value)
 

*Writes a single 32-bit word of memory (TODO support smaller sizes)*
- virtual uint32\_t **ReadDebugRegister** (uint32\_t address)
- virtual void [WriteDebugRegister](#) (uint32\_t address, uint32\_t value)
 

*Writes a single 32-bit word of memory.*
- [ARMJtagDebugPortStatusRegister](#) [GetStatusRegister](#) ()
 

*Gets the status register.*
- void **PrintStatusRegister** ([ARMJtagDebugPortStatusRegister](#) reg, bool children=true)
- virtual void **PrintStatusRegister** ()

## Static Public Member Functions

- static [JtagDevice](#) \* **CreateDevice** (unsigned int [partnum](#), unsigned int rev, unsigned int idcode, [JtagInterface](#) \*iface, size\_t pos)

## Public Attributes

- enum ARMJtagDebugPort::instructions **\_\_attribute\_\_**

## Protected Member Functions

- void **ClearStatusRegisterErrors** ()
- uint32\_t [DPRegisterRead](#) (DpReg addr)
 

*Reads from a DP register.*
- void **DPRegisterWrite** (DpReg addr, uint32\_t wdata)
- uint32\_t [APRegisterRead](#) (uint8\_t ap, ApReg addr)
 

*Reads from an AP register.*
- void [APRegisterWrite](#) (uint8\_t ap, ApReg addr, uint32\_t wdata)
 

*Writes to an AP register.*
- void **EnableDebugging** ()
- void [DebugAbort](#) ()
 

*Aborts the current AP transaction.*
- void **SetIR** (unsigned char irval)
- void **SetIRDeferred** (unsigned char irval)

## Protected Attributes

- unsigned int `m_rev`  
*Stepping number.*
- unsigned int `m_partnum`  
*Part number (normally IDCODE\_ARM\_DAP\_JTAG)*
- `std::map< uint8_t, ARMDebugAccessPort * >` `m_aps`  
*Access ports.*
- `ARMDebugMemAccessPort * m_defaultMemAP`  
*The default Mem-AP used for memory access.*
- `ARMDebugMemAccessPort * m_defaultRegisterAP`

## Friends

- class `ARMDebugMemAccessPort`

### 7.16.1 Detailed Description

An ARM JTAG-DP (contains one or more APs and a DP)

### 7.16.2 Member Function Documentation

#### 7.16.2.1 APRegisterRead()

```
uint32_t ARMJtagDebugPort::APRegisterRead (
 uint8_t ap,
 ApReg addr) [protected], [virtual]
```

Reads from an AP register.

#### Parameters

|                   |                                   |
|-------------------|-----------------------------------|
| <code>ap</code>   | The number of the AP to access    |
| <code>addr</code> | The ID of the AP register to read |

#### Returns

The value read

Implements `ARMDebugPort`.

### 7.16.2.2 APRegisterWrite()

```
void ARMJtagDebugPort::APRegisterWrite (
 uint8_t ap,
 ApReg addr,
 uint32_t wdata) [protected], [virtual]
```

Writes to an AP register.

#### Parameters

|              |                                   |
|--------------|-----------------------------------|
| <i>ap</i>    | The number of the AP to access    |
| <i>addr</i>  | The ID of the AP register to read |
| <i>wdata</i> | The value to write                |

Implements [ARMDebugPort](#).

### 7.16.2.3 DPRegisterRead()

```
uint32_t ARMJtagDebugPort::DPRegisterRead (
 DpReg addr) [protected]
```

Reads from a DP register.

#### Parameters

|             |                                   |
|-------------|-----------------------------------|
| <i>addr</i> | The ID of the DP register to read |
|-------------|-----------------------------------|

#### Returns

The value read

### 7.16.2.4 GetDescription()

```
std::string ARMJtagDebugPort::GetDescription () [virtual]
```

Gets a human-readable description of this device.

Example: "Xilinx XC6SLX45 stepping 3"

#### Returns

Device description

Implements [JtagDevice](#).

### 7.16.2.5 PostInitProbes()

```
void ARMJtagDebugPort::PostInitProbes (
 bool quiet) [virtual]
```

Does a post-initialization probe of the device to read debug ROMs etc.

#### Parameters

|              |                                                           |
|--------------|-----------------------------------------------------------|
| <i>quiet</i> | Do minimal probing to avoid triggering security lockdowns |
|--------------|-----------------------------------------------------------|

Implements [JtagDevice](#).

The documentation for this class was generated from the following files:

- [ARMJtagDebugPort.h](#)
- [ARMJtagDebugPort.cpp](#)

## 7.17 ARMJtagDebugPortStatusRegister Union Reference

ARM debug port status register (see ADIv5 Architecture Specification figure 6-3)

```
#include <ARMJtagDebugPort.h>
```

### Public Member Functions

- ```
struct {
    unsigned int sticky_overnen:1
        Set to 1 to enable overrun detection.
    unsigned int sticky_overnen:1
        Sticky buffer overrun (if enabled)
    unsigned int transfer_mode:2
        Transfer mode.
    unsigned int sticky_compare:1
        Sticky compare bit.
    unsigned int sticky_err:1
        Sticky error bit.
    unsigned int read_ok:1
        Read status flag.
    unsigned int wr_data_err:1
        Write data error flag.
    unsigned int mask_lane:4
        Byte mask.
    unsigned int trans_count:12
        Transaction counter.
    unsigned int reserved_zero:2
        Reserved, should be zero.
    unsigned int debug_reset_req:1
        Debug reset request.
    unsigned int debug_reset_ack:1
```



```

    Debug reset acknowledgement.
    unsigned int debug\_pwrup\_req:1
    Powerup request.
    unsigned int debug\_pwrup\_ack:1
    Powerup acknowledgement.
    unsigned int sys\_pwrup\_req:1
    Powerup request.
    unsigned int sys\_pwrup\_ack:1
    Powerup acknowledgement.
} __attribute__ ((packed)) bits

```

Public Attributes

- `uint32_t word`
The raw status register value.

7.17.1 Detailed Description

ARM debug port status register (see ADIV5 Architecture Specification figure 6-3)

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

- [ARMJtagDebugPort.h](#)

7.18 ARMv7DebugIDRegister Union Reference

ARM debug ID register (see ARMv7 Architecture Reference Manual, C11.11.15)

```
#include <ARMv7Processor.h>
```

Public Member Functions

- ```

struct {
 unsigned int revision:4
 Implementation defined CPU revision.
 unsigned int variant:4
 Implementation defined CPU variant.
 unsigned int reserved:4
 Reserved, undefined value.
 unsigned int sec_ext:1
 Indicates if security extensions are implemented.
 unsigned int pcsr_legacy_addr:1
 Indicates if PCSR is present at the legacy address.
 unsigned int no_secure_halt:1
 NO secure halting debug.
 unsigned int has_dbgdevid:1
 True if DBGDEVID is implemented.
 ARMDebugArchVersion debug_arch_version:4

```

```

 Debug arch version.
 unsigned int context_bpoints_minus_one:4
 Number of breakpoints supporting context matching, zero based (0 means 1 implemented, etc)
 unsigned int bpoints_minus_one:4
 Number of breakpoints, zero based (0 means 1 implemented, etc)
 unsigned int wpoints_minus_one:4
 Number of watchpoints, zero based (0 means 1 implemented, etc)
} __attribute__ ((packed)) bits

```

## Public Attributes

- `uint32_t` [word](#)  
*The raw register value.*

### 7.18.1 Detailed Description

ARM debug ID register (see ARMv7 Architecture Reference Manual, C11.11.15)

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

- [ARMv7Processor.h](#)

## 7.19 ARMv7DebugStatusControlRegister Union Reference

ARM debug status/control register (see ARMv7 Architecture Reference Manual, C11.11.20)

```
#include <ARMv7Processor.h>
```

## Public Member Functions

- ```

struct {
    unsigned int halted:1
        Set by the CPU when the processor is halted.
    unsigned int restarted:1
        Processor restarted flag.
    unsigned int entry\_method:4
        Method of debug entry (TODO)
    unsigned int sticky\_sync\_abt:1
        Sticky sync abort.
    unsigned int sticky\_async\_abt:1
        Sticky async abort.
    unsigned int sticky\_undef\_instr:1
        Sticky undefined instruction.
    unsigned int reserved\_sbz2:1
        Reserved.
    unsigned int force\_dbg\_ack:1
        Force debug acks regardless of cpu settings.
    unsigned int int\_dis:1

```

```

    Disable interrupts.
unsigned int user_dcc:1
    Enable user-mode access to the debug channel.
unsigned int inst_txfr:1
    Enable instruction transfer.
unsigned int halting_debug:1
    Enable halting-mode debug.
unsigned int monitor_debug:1
    Set high by the CPU if it allows monitor-mode debugging.
unsigned int secure_ni_debug:1
    Set high by the CPU if it allows invasive debug in secure mode.
unsigned int deprecated:1
    Deprecated "secure noninvasive debug" bit.
unsigned int nonsec:1
    Set high by the CPU if it is not in secure mode.
unsigned int discard_async_abort:1
    Set high to discard async aborts.
unsigned int ext_dcc_mode:2
    DCC access mode (TODO enum)
unsigned int instr_complete:1
    Latching instruction-complete bit for single instruction issue.
unsigned int pipelined_advancing:1
    Sticky "pipeline advancing" bit, set at unpredictable intervals when not halted.
unsigned int tx_full_latch:1
    Latching TX-full bit.
unsigned int rx_full_latch:1
    Latching RX-full bit.
unsigned int tx_full:1
    Indicates DBGDTRTX has valid data.
unsigned int rx_full:1
    Indicates DBGDTRRX has valid data.
unsigned int reserved_sbz:1
    Reserved, should be zero.
} __attribute__((packed)) bits

```

Public Attributes

- `uint32_t word`
The raw register value.

7.19.1 Detailed Description

ARM debug status/control register (see ARMv7 Architecture Reference Manual, C11.11.20)

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

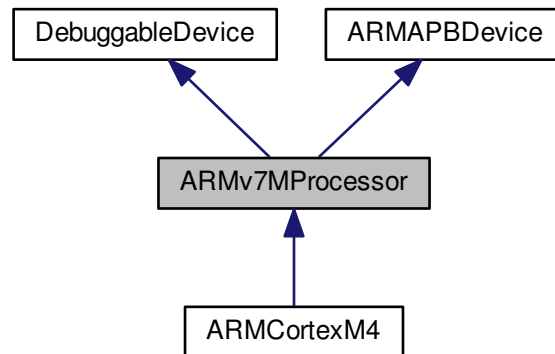
- [ARMv7Processor.h](#)

7.20 ARMv7MPProcessor Class Reference

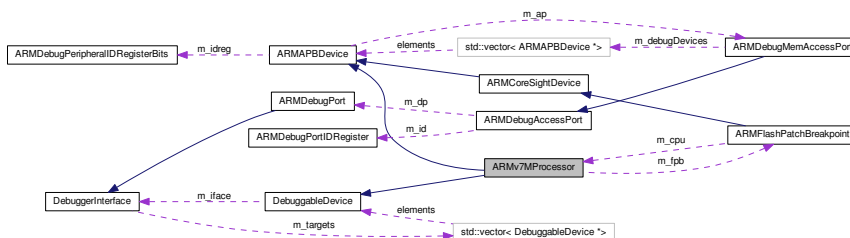
An ARMv7 Cortex-M CPU core, as seen over a CoreSight APB bus.

```
#include <ARMv7MPProcessor.h>
```

Inheritance diagram for ARMv7MPProcessor:



Collaboration diagram for ARMv7MPProcessor:



Public Types

- enum **ARM_V7M_SCS_REGISTERS** {
 - ACTLR** = 0x0002, **STCSR** = 0x0004, **STRVR** = 0x0005, **STCVR** = 0x0006,
 - STCR** = 0x0007, **CPUID** = 0x0340, **ICSR** = 0x0341, **VTOR** = 0x0342,
 - AIRCR** = 0x0343, **SCR** = 0x0344, **CCR** = 0x0345, **SHPR1** = 0x0346,
 - SHPR2** = 0x0347, **SHPR3** = 0x0348, **SHCSR** = 0x0349, **CFSR** = 0x034a,
 - HFSR** = 0x034b, **DFSR** = 0x034c, **MMFAR** = 0x034d, **BFAR** = 0x034e,
 - AFSR** = 0x034f, **ID_PFR0** = 0x0350, **ID_PFR1** = 0x0351, **ID_DFR0** = 0x0352,
 - ID_AFR0** = 0x0353, **ID_MMFAR0** = 0x0354, **ID_MMFR1** = 0x0355, **ID_MMFR2** = 0x0356,
 - ID_MMFR3** = 0x0357, **ID_ISAR0** = 0x0358, **ID_ISAR1** = 0x0359, **ID_ISAR2** = 0x035a,
 - ID_ISAR3** = 0x035b, **ID_ISAR4** = 0x035c, **CPACR** = 0x0362, **DHCSR** = 0x037c,
 - DCRSR** = 0x037d, **DCRDR** = 0x037e, **DEMCR** = 0x037f, **STIR** = 0x03c0 }

- enum **ARM_V7M_CPU_REGISTERS** {
R0 = 0, **R1** = 1, **R2** = 2, **R3** = 3,
R4 = 4, **R5** = 5, **R6** = 6, **R7** = 7,
R8 = 8, **R9** = 9, **R10** = 10, **R11** = 11,
R12 = 12, **SP** = 13, **LR** = 14, **DBGRA** = 15,
XPSR = 16, **MSP** = 17, **PSP** = 18, **CTRL** = 20 }

Public Member Functions

- **ARMv7MProcessor** ([DebuggerInterface](#) *iface, [ARMDebugMemAccessPort](#) *ap, uint32_t address, [ARMDebugPeripheralIDRegisterBits](#) idreg)
- virtual std::string **GetDescription** ()=0
- virtual void **PrintInfo** ()=0
- bool **HaltedDueToUnrecoverableException** ()
Checks if the CPU is halted due to a fatal error.
- uint32_t **ReadCPURegister** (ARM_V7M_CPU_REGISTERS reg)
- const char * **GetRegisterName** (ARM_V7M_CPU_REGISTERS reg)
- virtual void **DebugHalt** ()
Halts the CPU and enters debug state.
- virtual void **DebugResume** ()
- virtual void **PrintRegisters** ()
Prints out all CPU registers.
- [ARMFlashPatchBreakpoint](#) * **GetFlashPatchBreakpoint** ()
- void **AddFlashPatchUnit** ([ARMFlashPatchBreakpoint](#) *fpb)

Protected Attributes

- [ARMFlashPatchBreakpoint](#) * **m_fpb**

Additional Inherited Members

7.20.1 Detailed Description

An ARMv7 Cortex-M CPU core, as seen over a CoreSight APB bus.

7.20.2 Member Function Documentation

7.20.2.1 DebugHalt()

```
void ARMv7MProcessor::DebugHalt ( ) [virtual]
```

Halts the CPU and enters debug state.

See ARMv7-M arch manual C1-6

Implements [DebuggableDevice](#).

The documentation for this class was generated from the following files:

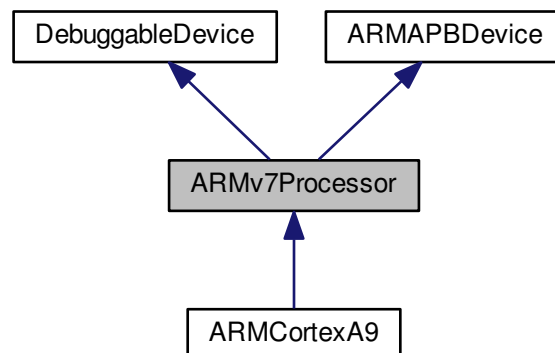
- [ARMv7MProcessor.h](#)
- [ARMv7MProcessor.cpp](#)

7.21 ARmv7Processor Class Reference

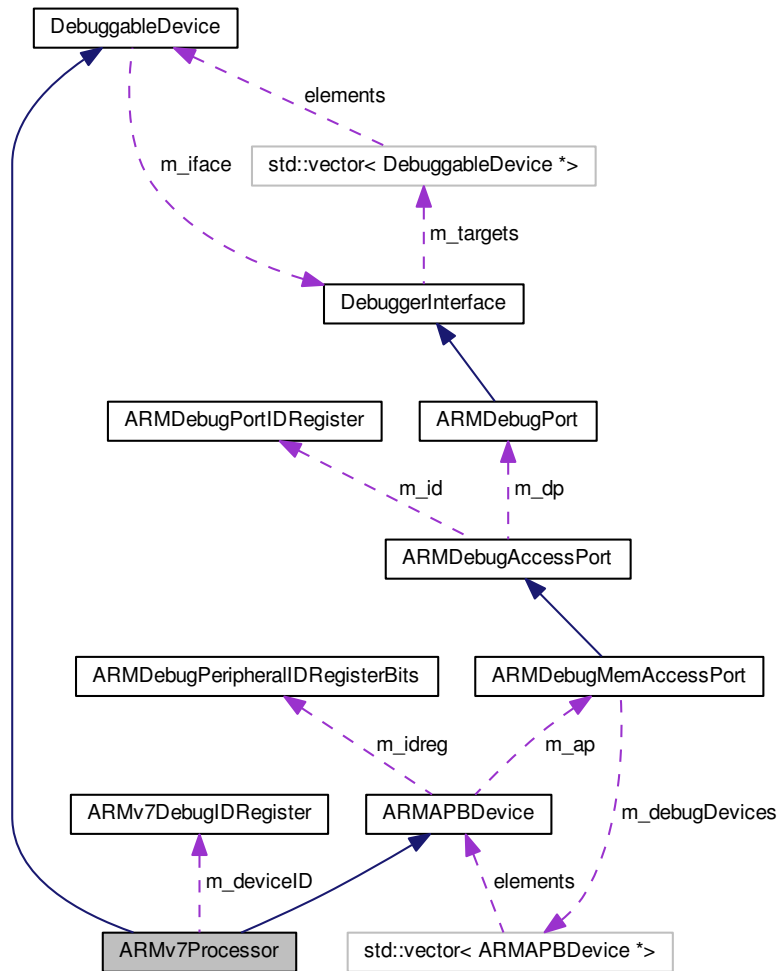
An ARmv7 Cortex-A CPU core, as seen over a CoreSight APB bus.

```
#include <ARmv7Processor.h>
```

Inheritance diagram for ARmv7Processor:



Collaboration diagram for ARMv7Processor:



Public Types

- enum **ARM_V7_DEBUG_REGISTERS** {

DBGDIDR = 0, DBGDSCR_INT = 1, DBGDTRRX_INT = 5, DBGDTRTX_INT = 5,

DBGWFAR = 6, DBGVCR = 7, DBGECCR = 9, DBGDSCCR = 10,

DBGDSMCR = 11, DBGDTRRX_EXT = 32, DBGITR = 33, DBGPCSR_LEGACY = 33,

DBGDSCR_EXT = 34, DBGDTRTX_EXT = 35, DBGDRCR = 36, DBGEACR = 37,

DBGPCSR = 40, DBGCIDSR = 41, DBGVIDSR = 42, DBGBVR_BASE = 64,

DBGBCR_BASE = 80, DBGWVR_BASE = 96, DBGWCR_BASE = 112, DBGDRAR = 128,

DBGBXVR_BASE = 144, DBGOSLAR = 192, DBGOSLSR = 193, DBGOSRRR = 194,

DBGOSDLR = 195, DBGPRCR = 196, DBGPRSR = 197, DBGDSAR = 256,

DBGPRID_BASE = 832, DBGITCTRL = 960, DBGCLAIMSET = 1000, DBGCLAIMCLR = 1001,

DBGLAR = 1004, DBGLSR = 1005, DBGAUTHSTATUS = 1006, DBGDEVID = 1010 }

Public Member Functions

- ARMv7Processor** (*DebuggerInterface* *iface, *ARMDebugMemAccessPort* *ap, uint32_t address, *ARMDebugPeripheralIDRegisterBits* idreg)

- virtual std::string **GetDescription** ()=0
- virtual void **PrintInfo** ()=0

Public Attributes

- enum ARMv7Processor::ARM_V7_DEBUG_REGISTERS **__attribute__**

Protected Member Functions

- void **PrintIDRegister** ([ARMv7DebugIDRegister](#) did)
- virtual void [DebugHalt](#) ()
 - *Halts the CPU and enters debug state.*
- virtual void **DebugResume** ()
- virtual void **PrintRegisters** ()

Protected Attributes

- unsigned int **m_breakpoints**
- unsigned int **m_context_breakpoints**
- unsigned int **m_watchpoints**
- bool **m_hasDevid**
- bool **m_hasSecExt**
- bool **m_hasSecureHalt**
- unsigned int **m_revision**
- unsigned int **m_variant**
- [ARMv7DebugIDRegister](#) **m_deviceID**
- ARM_V7_DEBUG_REGISTERS [m_pcsrIndex](#)
 - *Device-dependent address of the program counter sample register (PCSR)*

7.21.1 Detailed Description

An ARMv7 Cortex-A CPU core, as seen over a CoreSight APB bus.

7.21.2 Member Function Documentation

7.21.2.1 DebugHalt()

```
void ARMv7Processor::DebugHalt ( ) [protected], [virtual]
```

Halts the CPU and enters debug state.

See ARMv7-A/R arch ref manual, C11-2236

Implements [DebuggableDevice](#).

The documentation for this class was generated from the following files:

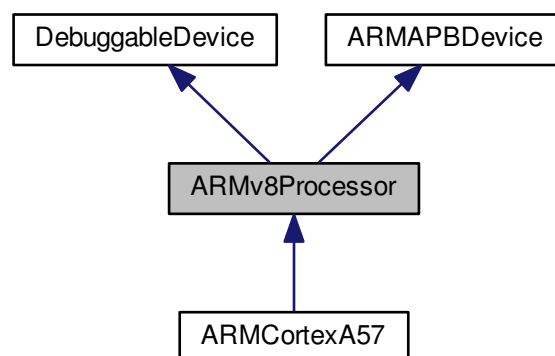
- [ARMv7Processor.h](#)
- [ARMv7Processor.cpp](#)

7.22 ARMv8Processor Class Reference

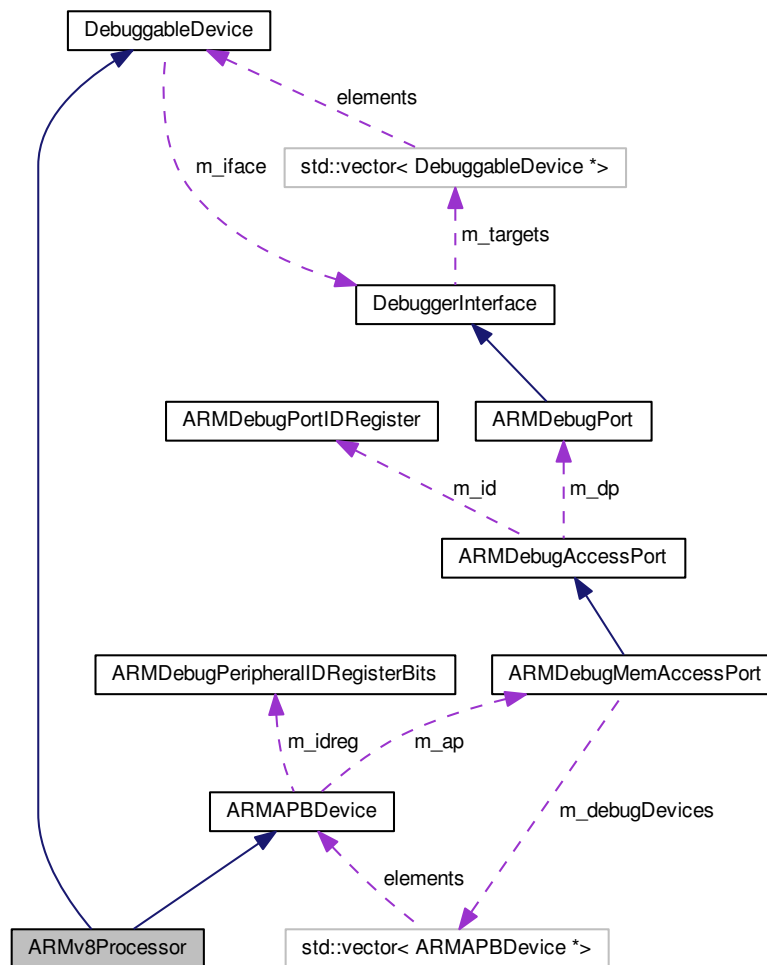
An ARMv8 Cortex-A CPU core, as seen over a CoreSight APB bus.

```
#include <ARMv8Processor.h>
```

Inheritance diagram for ARMv8Processor:



Collaboration diagram for ARMv8Processor:



Public Types

- enum **ARM_V8_DEBUG_REGISTERS** {
DBGDTRRX_EL0 = 0x020, **DBGDTRTX_EL0** = 0x023, **MIDR_EL1** = 0x340, **DBGAUTHSTATUS_EL1** = 0x3ee,
DBGCLAIMSET_EL1 = 0x3e8, **DBGCLAIMCLR_EL1** = 0x3e9 }

Public Member Functions

- ARMv8Processor** ([DebuggerInterface](#) *iface, [ARMDebugMemAccessPort](#) *ap, uint32_t address, [ARMDebugPeripheralIDRegisterBits](#) idreg)
- virtual std::string **GetDescription** ()=0
- virtual void **PrintInfo** ()=0

Protected Member Functions

- virtual void [DebugHalt](#) ()
Halts the CPU and enters debug state.
- virtual void **DebugResume** ()
- virtual void **PrintRegisters** ()

Additional Inherited Members

7.22.1 Detailed Description

An ARMv8 Cortex-A CPU core, as seen over a CoreSight APB bus.

ARM debug ID register (see ARMv8 ARM, C11.11.15) ARM debug status/control register (see ARMv8 ARM, C11.11.20)

The documentation for this class was generated from the following files:

- [ARMv8Processor.h](#)
- [ARMv8Processor.cpp](#)

7.23 AttachedMemoryDevice Class Reference

Base classes for devices which can connect to external memory devices.

```
#include <AttachedMemoryDevice.h>
```

Public Member Functions

- virtual size_t **GetNumAttachedMemories** ()=0
- virtual [ProgrammableDevice](#) * **GetAttachedMemoryDevice** (size_t i)=0

7.23.1 Detailed Description

Base classes for devices which can connect to external memory devices.

Example: [FPGA](#) with attached QSPI/BPI flash, MCU with external memory bus

The documentation for this class was generated from the following files:

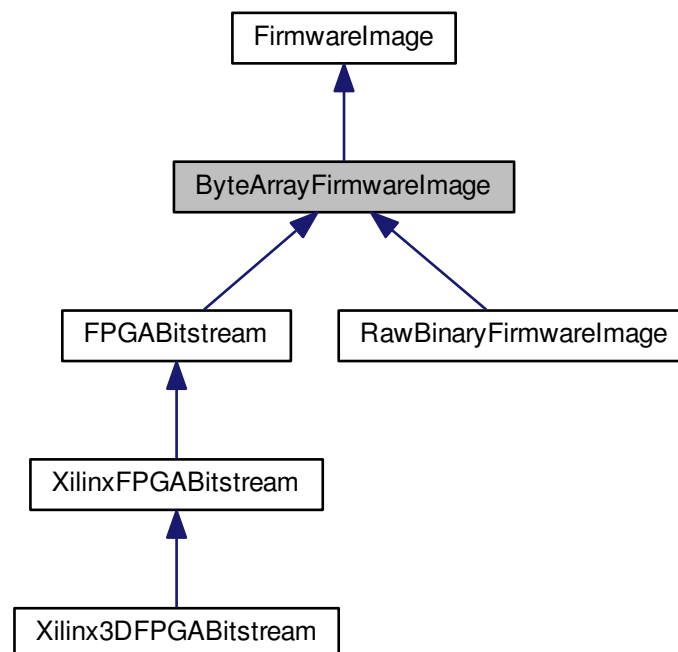
- [AttachedMemoryDevice.h](#)
- [AttachedMemoryDevice.cpp](#)

7.24 ByteArrayFirmwareImage Class Reference

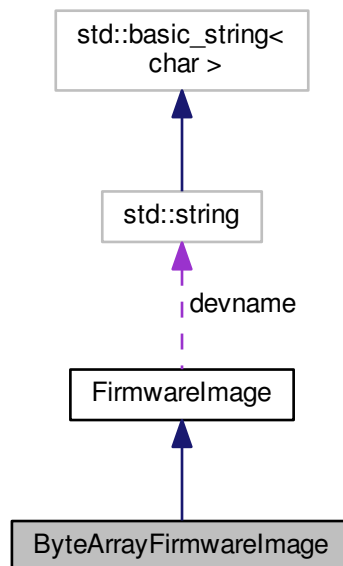
Generic base class for all firmware images consisting of an array of bytes.

```
#include <ByteArrayFirmwareImage.h>
```

Inheritance diagram for ByteArrayFirmwareImage:



Collaboration diagram for ByteArrayFirmwareImage:



Public Member Functions

- [ByteArrayFirmwareImage \(\)](#)
Initializes this object to empty.
- [virtual ~ByteArrayFirmwareImage \(\)](#)
Free bitstream memory.

Public Attributes

- `uint8_t * raw_bitstream`
Raw (header-less) bitstream data ready for sending to the device.
- `size_t raw_bitstream_len`
Length of the raw bitstream, in bytes.

7.24.1 Detailed Description

Generic base class for all firmware images consisting of an array of bytes.

The documentation for this class was generated from the following files:

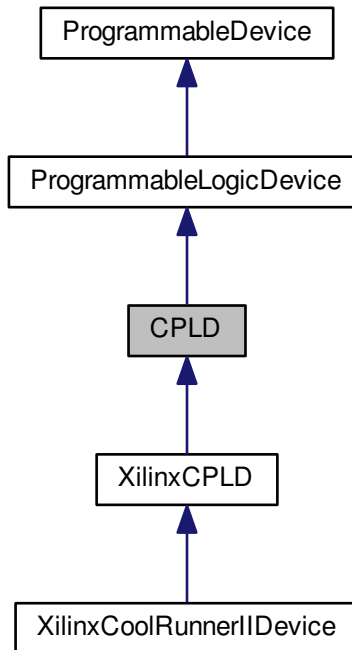
- [ByteArrayFirmwareImage.h](#)
- [ByteArrayFirmwareImage.cpp](#)

7.25 CPLD Class Reference

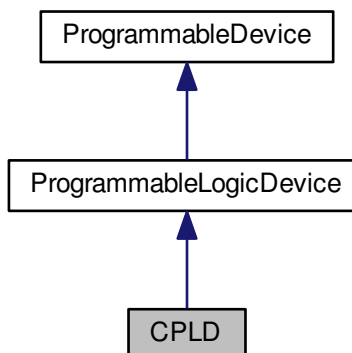
Generic base class for all complex programmable logic devices.

```
#include <CPLD.h>
```

Inheritance diagram for CPLD:



Collaboration diagram for CPLD:



Public Member Functions

- virtual `~CPLD ()`
Empty virtual destructor.

Static Public Member Functions

- static void `ParseJEDFile (CPLDBitstream *bit, const unsigned char *data, size_t len)`
Parses a JED file.

Static Protected Member Functions

- static int `ReadIntLine (const char *cdata, size_t &pos, size_t len)`
*Reads a line containing an integer terminated by a *.*

7.25.1 Detailed Description

Generic base class for all complex programmable logic devices.

7.25.2 Member Function Documentation

7.25.2.1 ParseJEDFile()

```
void CPLD::ParseJEDFile (
    CPLDBitstream * bit,
    const unsigned char * data,
    size_t len ) [static]
```

Parses a JED file.

Reference: JEDEC Standard 3-C

Exceptions

JtagException	if the file is malformed
-------------------------------	--------------------------

Parameters

<i>bit</i>	Output bitstream
<i>data</i>	Data to load
<i>len</i>	Length of the file

7.25.2.2 ReadIntLine()

```
int CPLD::ReadIntLine (
    const char * cdata,
    size_t & pos,
    size_t len ) [static], [protected]
```

Reads a line containing an integer terminated by a *.

Exceptions

JtagException	if malformed input
-------------------------------	--------------------

Parameters

<i>cdata</i>	Buffer to read
<i>pos</i>	Position to read from (updated at function return)
<i>len</i>	Length of the entire buffer

Returns

The value

The documentation for this class was generated from the following files:

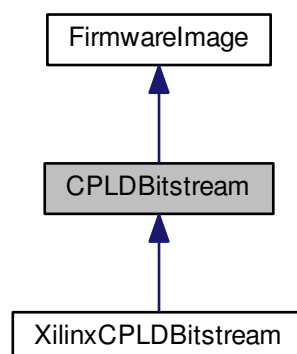
- [CPLD.h](#)
- [CPLD.cpp](#)

7.26 CPLDBitstream Class Reference

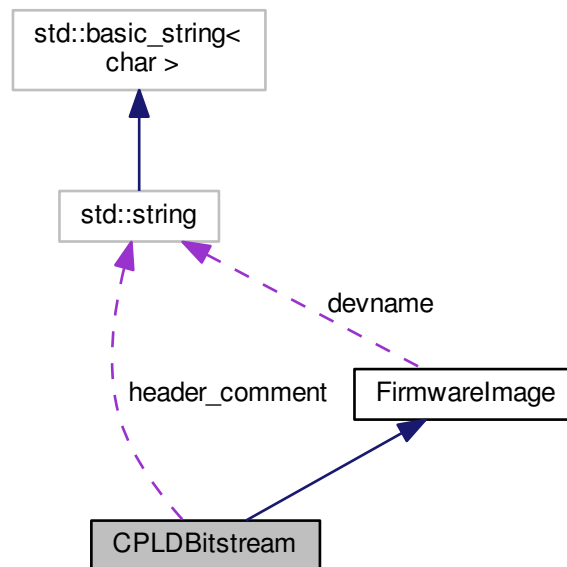
Abstract base class for [CPLD](#) configuration bitstreams.

```
#include <CPLDBitstream.h>
```

Inheritance diagram for CPLDBitstream:



Collaboration diagram for CPLDBitstream:



Public Member Functions

- [CPLDBitstream \(\)](#)
Default constructor.
- [virtual ~CPLDBitstream \(\)](#)
Empty virtual destructor.

Public Attributes

- `std::string` [header_comment](#)
Header comment.
- `uint16_t` [file_checksum](#)
JED file checksum.
- `uint16_t` [fuse_checksum](#)
Fuse array checksum.
- `unsigned int` [fuse_count](#)
Number of fuses.
- `unsigned int` [pin_count](#)
Number of pins.
- `bool *` [fuse_data](#)
Fuse data.

7.26.1 Detailed Description

Abstract base class for [CPLD](#) configuration bitstreams.

The documentation for this class was generated from the following files:

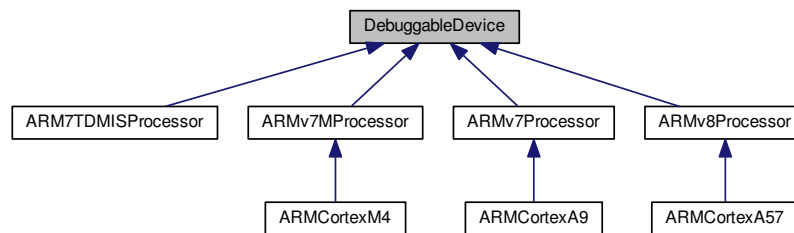
- [CPLDBitstream.h](#)
- [CPLDBitstream.cpp](#)

7.27 DebuggableDevice Class Reference

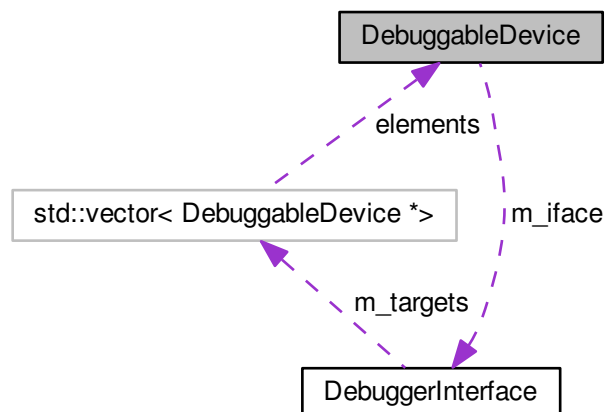
Generic base class for all debuggable devices (CPU cores etc)

```
#include <DebuggableDevice.h>
```

Inheritance diagram for DebuggableDevice:



Collaboration diagram for DebuggableDevice:



Public Member Functions

- **DebuggableDevice** ([DebuggerInterface](#) *iface)
- virtual std::string **GetDescription** ()=0
- virtual uint32_t **ReadMemory** (uint32_t addr)
- virtual void **WriteMemory** (uint32_t addr, uint32_t value)
- virtual void **DebugHalt** ()=0
- virtual void **DebugResume** ()=0
- virtual void **PrintRegisters** ()=0

Protected Attributes

- [DebuggerInterface](#) * **m_iface**

7.27.1 Detailed Description

Generic base class for all debuggable devices (CPU cores etc)

The documentation for this class was generated from the following files:

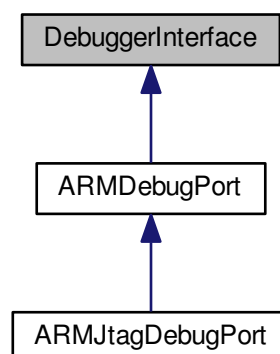
- [DebuggableDevice.h](#)
- [DebuggableDevice.cpp](#)

7.28 DebuggerInterface Class Reference

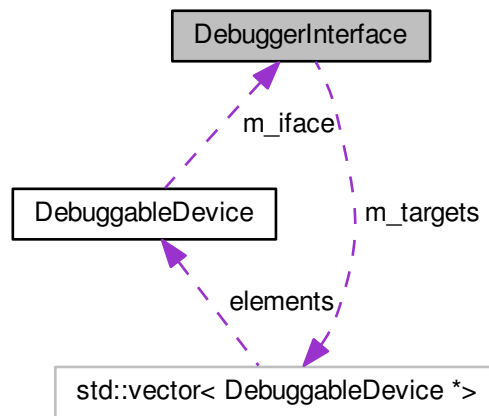
Generic base class for all debugger interfaces (may connect to multiple [DebuggableDevice](#)'s in a SoC)

```
#include <DebuggerInterface.h>
```

Inheritance diagram for DebuggerInterface:



Collaboration diagram for DebuggerInterface:



Public Member Functions

- virtual `size_t` [GetNumTargets](#) ()
Returns the number of [DebuggableDevice](#)'s attached to this debugger.
- virtual `DebuggableDevice *` [GetTarget](#) (size_t i)
Returns a specific [DebuggableDevice](#).
- void [AddTarget](#) (`DebuggableDevice *`target)
Adds a new debuggable device to this interface (called during topology discovery)
- virtual `uint32_t` [ReadMemory](#) (`uint32_t` address)=0
Read a single 32-bit word of memory (TODO support smaller sizes)
- virtual void [WriteMemory](#) (`uint32_t` address, `uint32_t` value)=0
Writes a single 32-bit word of memory (TODO support smaller sizes)

Protected Attributes

- `std::vector< DebuggableDevice * >` [m_targets](#)
The devices (NOT automatically deleted at destruction time)

7.28.1 Detailed Description

Generic base class for all debugger interfaces (may connect to multiple [DebuggableDevice](#)'s in a SoC)

The documentation for this class was generated from the following files:

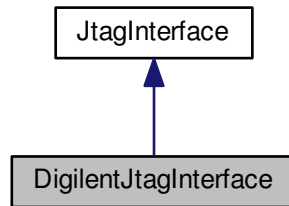
- [DebuggerInterface.h](#)
- [DebuggerInterface.cpp](#)

7.29 DigilentJtagInterface Class Reference

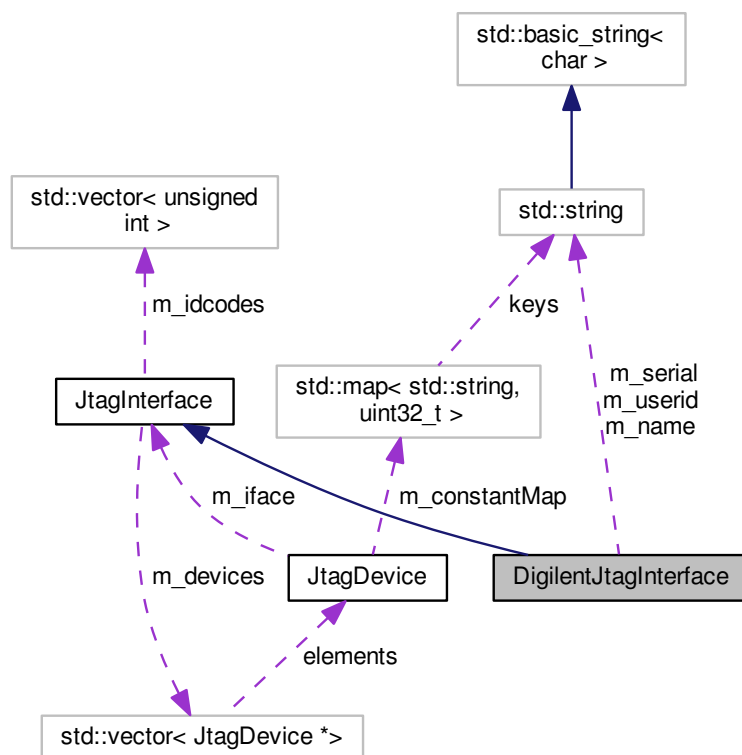
A JTAG adapter exposed through the Digilent Adept SDK.

```
#include <DigilentJtagInterface.h>
```

Inheritance diagram for DigilentJtagInterface:



Collaboration diagram for DigilentJtagInterface:



Public Member Functions

- [DigilentJtagInterface](#) (int ndev)
Connects to a Digilent JTAG interface.
- virtual [~DigilentJtagInterface](#) ()
Interface destructor.
- virtual std::string [GetName](#) ()
Gets the manufacturer-assigned name for this programming adapter.
- virtual std::string [GetSerial](#) ()
Gets the manufacturer-assigned serial number for this programming adapter, if any.
- virtual std::string [GetUserID](#) ()
Gets the user-assigned name for this JTAG adapter, if any.
- virtual int [GetFrequency](#) ()
Gets the clock frequency, in Hz, of the JTAG interface.
- virtual void [ShiftData](#) (bool last_tms, const unsigned char *send_data, unsigned char *rcv_data, size_t count)
Shifts data through TDI to TDO.
- virtual void [SendDummyClocks](#) (size_t n)
Sends the requested number of dummy clocks with TMS=0 and flushes the command to the interface.

Static Public Member Functions

- static std::string [GetAPIVersion](#) ()
Gets the version number of the Digilent JTAG API.
- static int [GetInterfaceCount](#) ()
Gets the number of interfaces on the system.
- static std::string [GetName](#) (int i)
- static std::string [GetSerial](#) (int i)
- static std::string [GetUserID](#) (int i)
- static int [GetDefaultFrequency](#) (int i)

Protected Member Functions

- virtual void [ShiftTMS](#) (bool tdi, const unsigned char *send_data, size_t count)
Shifts data into TMS to change TAP state.

Static Protected Member Functions

- static std::string [GetLibraryError](#) ()
Gets the last-error code of the Digilent API.

Protected Attributes

- std::string [m_name](#)
The adapter's name.
- std::string [m_serial](#)
The adapter's serial number.
- std::string [m_userid](#)
The adapter's user ID.
- unsigned int [m_hif](#)
Digilent API interface handle.
- int [m_freq](#)
The adapter's clock frequency.

7.29.1 Detailed Description

A JTAG adapter exposed through the Digilent Adept SDK.

This includes Digilent-branded dev boards, Digilent-adapter cables such as the HS1, and OEM modules such as the JTAG-SMT2 which may be integrated into third party boards.

7.29.2 Constructor & Destructor Documentation

7.29.2.1 DigilentJtagInterface()

```
DigilentJtagInterface::DigilentJtagInterface (
    int ndev )
```

Connects to a Digilent JTAG interface.

Exceptions

JtagException	if the connection could not be establishes or the index is invalid
-------------------------------	--

Parameters

<i>ndev</i>	Zero-based index of the device to connect to
-------------	--

7.29.2.2 ~DigilentJtagInterface()

```
DigilentJtagInterface::~DigilentJtagInterface ( ) [virtual]
```

Interface destructor.

Closes handles and disconnects from the adapter.

7.29.3 Member Function Documentation

7.29.3.1 GetFrequency()

```
int DigilentJtagInterface::GetFrequency ( ) [virtual]
```

Gets the clock frequency, in Hz, of the JTAG interface.

Returns

The clock frequency

Implements [JtagInterface](#).

7.29.3.2 GetName()

```
std::string DigilentJtagInterface::GetName ( ) [virtual]
```

Gets the manufacturer-assigned name for this programming adapter.

This is usually the model number but is sometimes something more generic like "Digilent Adept USB Device".

Returns

The device name

Implements [JtagInterface](#).

7.29.3.3 GetSerial()

```
std::string DigilentJtagInterface::GetSerial ( ) [virtual]
```

Gets the manufacturer-assigned serial number for this programming adapter, if any.

Derived classes may choose to return the user ID, an empty string, or another default value if no serial number has been assigned.

Returns

The serial number

Implements [JtagInterface](#).

7.29.3.4 GetUserID()

```
std::string DigilentJtagInterface::GetUserID ( ) [virtual]
```

Gets the user-assigned name for this JTAG adapter, if any.

Derived classes may choose to return the serial number, an empty string, or another default value if no name has been assigned.

Returns

The name for this adapter.

Implements [JtagInterface](#).

7.29.3.5 SendDummyClocks()

```
void DigilentJtagInterface::SendDummyClocks (
    size_t n ) [virtual]
```

Sends the requested number of dummy clocks with TMS=0 and flushes the command to the interface.

Since dummy clocks are often used as a delay element for programming algorithms etc, this function flushes the write buffer to ensure immediate execution.

Exceptions

JtagException	may be thrown if the scan operation fails
-------------------------------	---

Parameters

<i>n</i>	Number of dummy clocks to send
----------	--------------------------------

Implements [JtagInterface](#).

7.29.3.6 ShiftData()

```
void DigilentJtagInterface::ShiftData (
    bool last_tms,
    const unsigned char * send_data,
    unsigned char * rcv_data,
    size_t count ) [virtual]
```

Shifts data through TDI to TDO.

The LSB of send_data[0] is sent first; the MSB of send_data[0] is followed by the LSB of send_data[1].

Parameters

<i>last_tms</i>	Different TMS value to use for last bit
<i>send_data</i>	Data to shift into TDI
<i>rcv_data</i>	Data to shift out of TDO (may be NULL)
<i>count</i>	Number of bits to shift

Implements [JtagInterface](#).

7.29.3.7 ShiftTMS()

```
void DigilentJtagInterface::ShiftTMS (
    bool tdi,
    const unsigned char * send_data,
    size_t count ) [protected], [virtual]
```

Shifts data into TMS to change TAP state.

This is no longer a public API operation. It can only be accessed via the state-level interface.

Implementations of this class may choose to implement EITHER this function (and use the default JtagInterface-provided state-level functions) OR override this function with a private no-op stub and override the state-level functions instead.

Exceptions

JtagException	may be thrown if the scan operation fails
-------------------------------	---

Parameters

<i>tdi</i>	Constant tdi value (normally 0)
<i>send_data</i>	Data to shift into TMS. Bit ordering is the same as for ShiftData() .
<i>count</i>	Number of bits to shift

Implements [JtagInterface](#).

The documentation for this class was generated from the following files:

- [DigilentJtagInterface.h](#)
- [DigilentJtagInterface.cpp](#)

7.30 EjtagControlRegister Union Reference

PIC32 EJTAG control register.

```
#include <MicrochipPIC32Device.h>
```

Public Member Functions

```

•
  struct {
    unsigned int reserved1: 3
    unsigned int debug_mode: 1
    unsigned int reserved2: 8
    unsigned int debug_irq: 1
    unsigned int reserved3: 1
    unsigned int debug_vector_pos: 1
    unsigned int probe_enable: 1
    unsigned int proc_reset: 1
    unsigned int reserved4: 1
    unsigned int proc_access: 1
    unsigned int proc_we: 1
    unsigned int periph_reset: 1
    unsigned int bus_halted: 1
    unsigned int low_power: 1
    unsigned int vpe_disable: 1
    unsigned int reserved5: 5
    unsigned int access_size: 2
    unsigned int reset_occurred: 1
  } __attribute__((packed)) bits

```

Public Attributes

- uint32_t [word](#)
The raw status register value.

7.30.1 Detailed Description

PIC32 EJTAG control register.

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

- [MicrochipPIC32Device.h](#)

7.31 EhtagImplementationCodeRegister Union Reference

MIPS EJTAG implementation register.

```
#include <MicrochipPIC32Device.h>
```

Public Member Functions

- ```
struct {
 unsigned int processor_is_64: 1
 unsigned int reserved1: 13
 unsigned int no_ejtag_dma: 1
 unsigned int reserved2: 1
 unsigned int mips16_supported: 1
 unsigned int reserved3: 4
 unsigned int asid_size: 2
 unsigned int reserved4: 1
 unsigned int dint_supported: 1
 unsigned int reserved5: 3
 unsigned int r3k_priv: 1
 unsigned int ejtag_version: 3
} __attribute__((packed)) bits
```

## Public Attributes

- uint32\_t [word](#)  
*The raw status register value.*

### 7.31.1 Detailed Description

MIPS EJTAG implementation register.

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

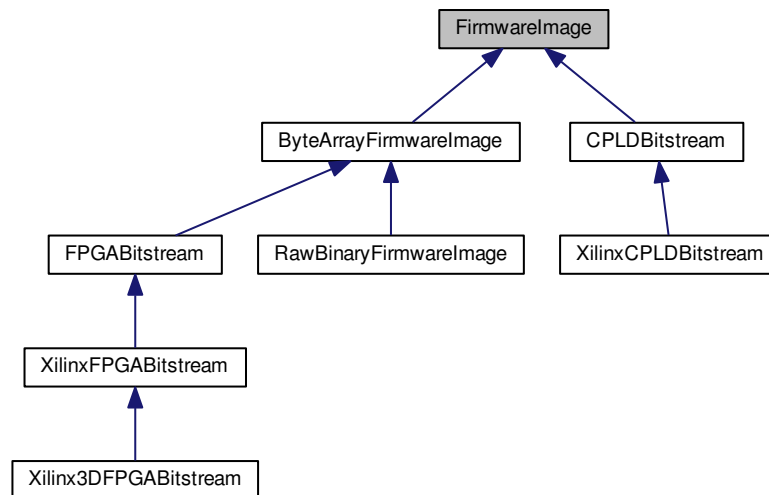
- [MicrochipPIC32Device.h](#)

## 7.32 FirmwareImage Class Reference

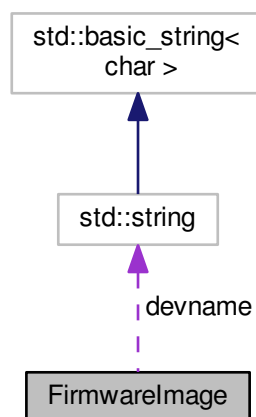
Generic base class for all firmware images for any kind of programmable device.

```
#include <FirmwareImage.h>
```

Inheritance diagram for FirmwareImage:



Collaboration diagram for FirmwareImage:



### Public Member Functions

- virtual `std::string` **GetDescription** ()

## Public Attributes

- unsigned int [idcode](#)  
*JTAG ID code of the device this firmware image is intended for.*
- std::string [devname](#)  
*Human-readable name of the device this bitstream is intended for.*

### 7.32.1 Detailed Description

Generic base class for all firmware images for any kind of programmable device.

### 7.32.2 Member Data Documentation

#### 7.32.2.1 idcode

```
unsigned int FirmwareImage::idcode
```

JTAG ID code of the device this firmware image is intended for.

The special value of all zeroes is not a valid JEDEC ID code and is used for softcores, ROM images, and other devices which are not directly exposed by JTAG.

The documentation for this class was generated from the following files:

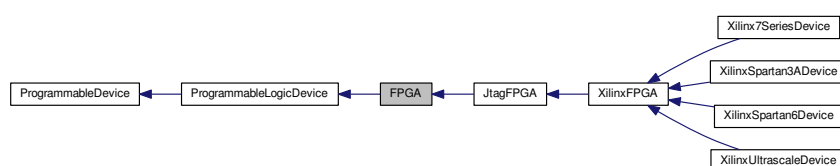
- [FirmwareImage.h](#)
- [FirmwareImage.cpp](#)

## 7.33 FPGA Class Reference

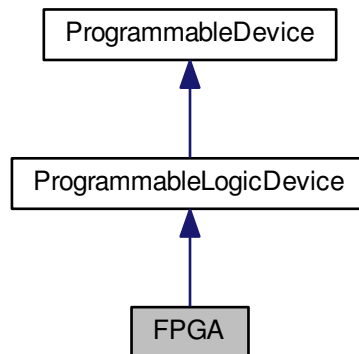
Generic base class for all field-programmable gate array devices.

```
#include <FPGA.h>
```

Inheritance diagram for FPGA:



Collaboration diagram for FPGA:



## Public Member Functions

- virtual `~FPGA ()`  
*Empty virtual destructor.*

### 7.33.1 Detailed Description

Generic base class for all field-programmable gate array devices.

The documentation for this class was generated from the following files:

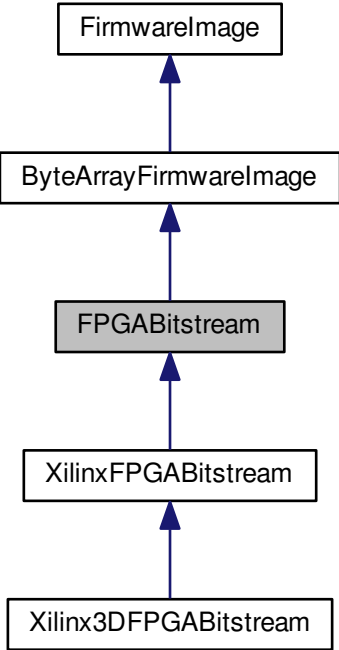
- [FPGA.h](#)
- [FPGA.cpp](#)

## 7.34 FPGABitstream Class Reference

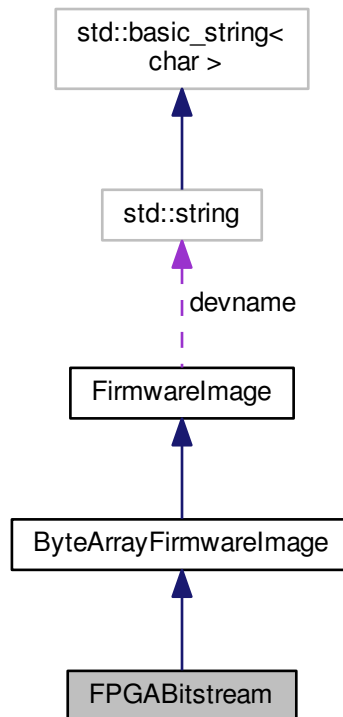
Abstract base class for [FPGA](#) configuration bitstreams.

```
#include <FPGABitstream.h>
```

Inheritance diagram for FPGABitstream:



Collaboration diagram for FPGABitstream:



## Public Member Functions

- [FPGABitstream \(\)](#)  
*Default constructor.*
- [virtual ~FPGABitstream \(\)](#)  
*Empty virtual destructor.*

## Additional Inherited Members

### 7.34.1 Detailed Description

Abstract base class for [FPGA](#) configuration bitstreams.

The documentation for this class was generated from the following files:

- [FPGABitstream.h](#)
- [FPGABitstream.cpp](#)

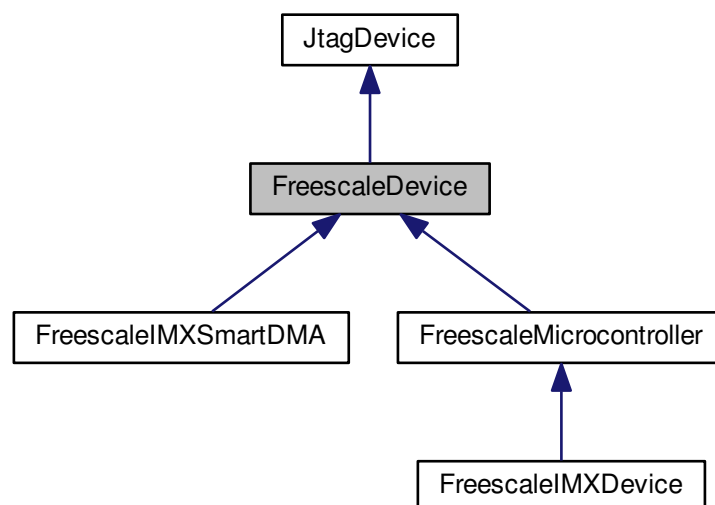


## 7.35 FreescaleDevice Class Reference

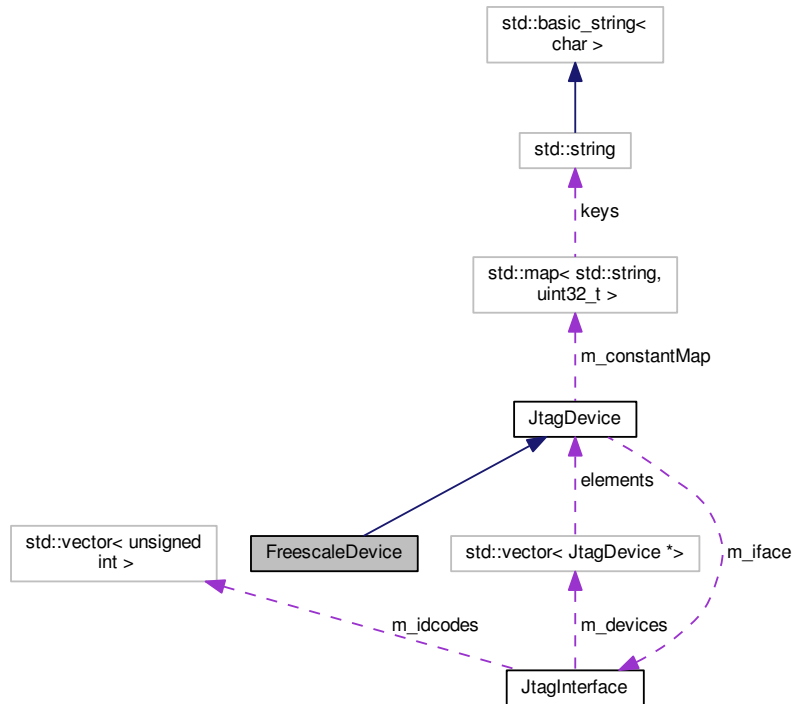
Abstract base class for all Freescale devices (typically MCUs or parts thereof)

```
#include <FreescaleDevice.h>
```

Inheritance diagram for FreescaleDevice:



Collaboration diagram for FreescaleDevice:



## Public Member Functions

- `FreescaleDevice` (unsigned int idcode, `JtagInterface` \*iface, size\_t pos, size\_t irlength)  
*Initializes this device.*
- virtual `~FreescaleDevice` ()  
*Default virtual destructor.*

## Static Public Member Functions

- static `JtagDevice` \* `CreateDevice` (unsigned int idcode, `JtagInterface` \*iface, size\_t pos)  
*Creates a `FreescaleDevice` given an ID code.*

## Additional Inherited Members

### 7.35.1 Detailed Description

Abstract base class for all Freescale devices (typically MCUs or parts thereof)

### 7.35.2 Constructor & Destructor Documentation

### 7.35.2.1 FreescaleDevice()

```
FreescaleDevice::FreescaleDevice (
 unsigned int idcode,
 JtagInterface * iface,
 size_t pos,
 size_t irlength)
```

Initializes this device.

#### Parameters

|               |                                                       |
|---------------|-------------------------------------------------------|
| <i>idcode</i> | The ID code of this device                            |
| <i>iface</i>  | The JTAG adapter this device was discovered on        |
| <i>pos</i>    | Position in the chain that this device was discovered |

## 7.35.3 Member Function Documentation

### 7.35.3.1 CreateDevice()

```
JtagDevice * FreescaleDevice::CreateDevice (
 unsigned int idcode,
 JtagInterface * iface,
 size_t pos) [static]
```

Creates a [FreescaleDevice](#) given an ID code.

#### Exceptions

|                               |                                                                                       |
|-------------------------------|---------------------------------------------------------------------------------------|
| <a href="#">JtagException</a> | if the ID code supplied is not a valid Freescale device, or not a known family number |
|-------------------------------|---------------------------------------------------------------------------------------|

#### Parameters

|               |                                                       |
|---------------|-------------------------------------------------------|
| <i>idcode</i> | The ID code of this device                            |
| <i>iface</i>  | The JTAG adapter this device was discovered on        |
| <i>pos</i>    | Position in the chain that this device was discovered |

#### Returns

A valid [JtagDevice](#) object, or NULL if the vendor ID was not recognized.

The documentation for this class was generated from the following files:

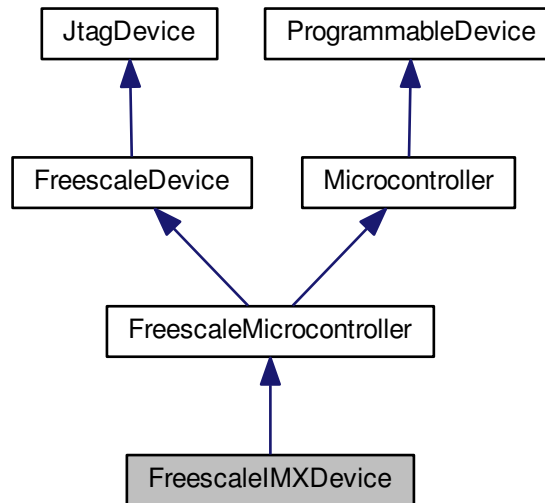
- [FreescaleDevice.h](#)
- [FreescaleDevice.cpp](#)

## 7.36 FreescaleIMXDevice Class Reference

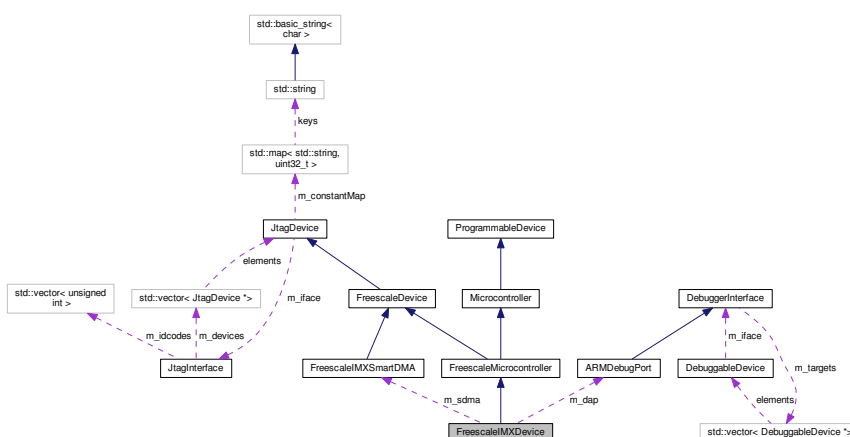
A Freescale i.mx applications processor.

```
#include <FreescaleIMXDevice.h>
```

Inheritance diagram for FreescaleIMXDevice:



Collaboration diagram for FreescaleIMXDevice:



### Public Types

- enum `instructions` {  
`INST_IDCODE` = 0x00, `INST_SAMPLE_PRELOAD` = 0x01, `INST_EXTEST` = 0x02, `INST_HIZ` = 0x03,

```
INST_EXTEST_PULSE = 0x08, INST_EXTEST_TRAIN = 0x09, INST_EXTRADEBUG = 0x04, INST_EN←
TER_DEBUG = 0x05,
INST_SECURE_CHALL = 0x0c, INST_SECURE_RESP = 0x0d, INST_TAP_SELECT = 0x07, INST_BY←
PASS = 0x1f }
```

*5-bit-wide JTAG instructions (from datasheet table 56-3)*

## Public Member Functions

- **FreescaleIMXDevice** (unsigned int devid, unsigned int stepping, unsigned int idcode, [JtagInterface](#) \*iface, size\_t pos)
- virtual [~FreescaleIMXDevice](#) ()  
*Destructor.*
- virtual void [PostInitProbes](#) (bool quiet)  
*Does a post-initialization probe of the device to read debug ROMs etc.*
- virtual std::string [GetDescription](#) ()  
*Gets a human-readable description of this device.*
- virtual bool [IsProgrammed](#) ()  
*Determines if this device is programmed or blank.*
- virtual void [Erase](#) ()  
*Erases the device configuration and restores the device to a blank state.*
- virtual void [Program](#) ([FirmwareImage](#) \*image)  
*Loads a new firmware image onto the device.*
- void [SetIR](#) (unsigned char irval)

## Static Public Member Functions

- static [JtagDevice](#) \* [CreateDevice](#) (unsigned int devid, unsigned int stepping, unsigned int idcode, [JtagInterface](#) \*iface, size\_t pos)

## Protected Attributes

- ImxDeviceIDs [m\\_devid](#)  
*Device ID code.*
- unsigned int [m\\_stepping](#)  
*Stepping number.*
- [ARMDebugPort](#) \* [m\\_dap](#)
- [FreescaleIMXSmartDMA](#) \* [m\\_sdma](#)

### 7.36.1 Detailed Description

A Freescale i.mx applications processor.

### 7.36.2 Member Enumeration Documentation

#### 7.36.2.1 instructions

enum [FreescaleIMXDevice::instructions](#)

5-bit-wide JTAG instructions (from datasheet table 56-3)

## Enumerator

|                     |                      |
|---------------------|----------------------|
| INST_IDCODE         | Read ID code.        |
| INST_SAMPLE_PRELOAD | Boundary scan stuff. |

### 7.36.3 Member Function Documentation

#### 7.36.3.1 Erase()

```
void FreescaleIMXDevice::Erase () [virtual]
```

Erases the device configuration and restores the device to a blank state.

After this function is called, regardless of success or failure, all existing connections to on-chip code become invalid.

#### Exceptions

|                               |                              |
|-------------------------------|------------------------------|
| <a href="#">JtagException</a> | if the erase operation fails |
|-------------------------------|------------------------------|

Implements [ProgrammableDevice](#).

#### 7.36.3.2 GetDescription()

```
string FreescaleIMXDevice::GetDescription () [virtual]
```

Gets a human-readable description of this device.

Example: "Xilinx XC6SLX45 stepping 3"

#### Returns

Device description

Implements [JtagDevice](#).

#### 7.36.3.3 IsProgrammed()

```
bool FreescaleIMXDevice::IsProgrammed () [virtual]
```

Determines if this device is programmed or blank.

#### Returns

true if programmed, false if blank

Implements [ProgrammableDevice](#).

## 7.36.3.4 PostInitProbes()

```
void FreescaleIMXDevice::PostInitProbes (
 bool quiet) [virtual]
```

Does a post-initialization probe of the device to read debug ROMs etc.

## Parameters

|              |                                                           |
|--------------|-----------------------------------------------------------|
| <i>quiet</i> | Do minimal probing to avoid triggering security lockdowns |
|--------------|-----------------------------------------------------------|

Implements [JtagDevice](#).

## 7.36.3.5 Program()

```
void FreescaleIMXDevice::Program (
 FirmwareImage * image) [virtual]
```

Loads a new firmware image onto the device.

After this function is called, regardless of success or failure, all existing connections to on-chip code become invalid.

## Exceptions

|                               |                              |
|-------------------------------|------------------------------|
| <a href="#">JtagException</a> | if the erase operation fails |
|-------------------------------|------------------------------|

## Parameters

|              |                          |
|--------------|--------------------------|
| <i>image</i> | The parsed image to load |
|--------------|--------------------------|

Implements [ProgrammableDevice](#).

The documentation for this class was generated from the following files:

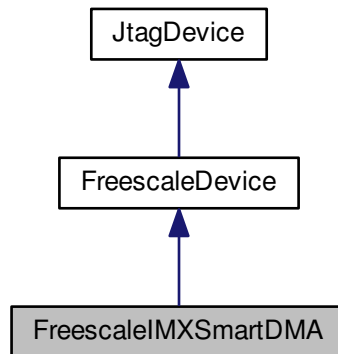
- [FreescaleIMXDevice.h](#)
- [FreescaleIMXDevice.cpp](#)

## 7.37 FreescaleIMXSmartDMA Class Reference

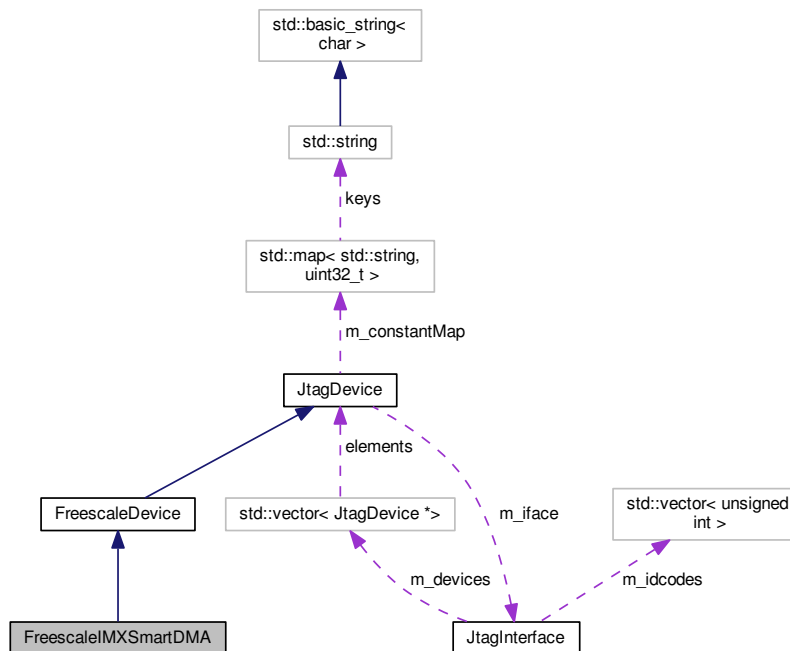
The SDMA in a Freescale i.mx SoC (Chapter 55 of i.mx6 reference manual)

```
#include <FreescaleIMXSmartDMA.h>
```

Inheritance diagram for FreescaleIMXSmartDMA:



Collaboration diagram for FreescaleIMXSmartDMA:



## Public Member Functions

- **FreescaleIMXSmartDMA** (unsigned int devid, unsigned int stepping, unsigned int idcode, [JtagInterface](#) \*iface, size\_t pos)
- virtual `~FreescaleIMXSmartDMA` ()  
*Destructor.*



- virtual void [PostInitProbes](#) (bool quiet)  
*Does a post-initialization probe of the device to read debug ROMs etc.*
- virtual std::string [GetDescription](#) ()  
*Gets a human-readable description of this device.*
- void **SetIR** (unsigned char irval)

### Static Public Member Functions

- static [JtagDevice](#) \* **CreateDevice** (unsigned int devid, unsigned int stepping, unsigned int idcode, [JtagInterface](#) \*iface, size\_t pos)

### Additional Inherited Members

#### 7.37.1 Detailed Description

The SDMA in a Freescale i.mx SoC (Chapter 55 of i.mx6 reference manual)

#### 7.37.2 Member Function Documentation

##### 7.37.2.1 GetDescription()

```
string FreescaleIMXSmartDMA::GetDescription () [virtual]
```

Gets a human-readable description of this device.

Example: "Xilinx XC6SLX45 stepping 3"

##### Returns

Device description

Implements [JtagDevice](#).

##### 7.37.2.2 PostInitProbes()

```
void FreescaleIMXSmartDMA::PostInitProbes (
 bool quiet) [virtual]
```

Does a post-initialization probe of the device to read debug ROMs etc.

## Parameters

|              |                                                           |
|--------------|-----------------------------------------------------------|
| <i>quiet</i> | Do minimal probing to avoid triggering security lockdowns |
|--------------|-----------------------------------------------------------|

Implements [JtagDevice](#).

The documentation for this class was generated from the following files:

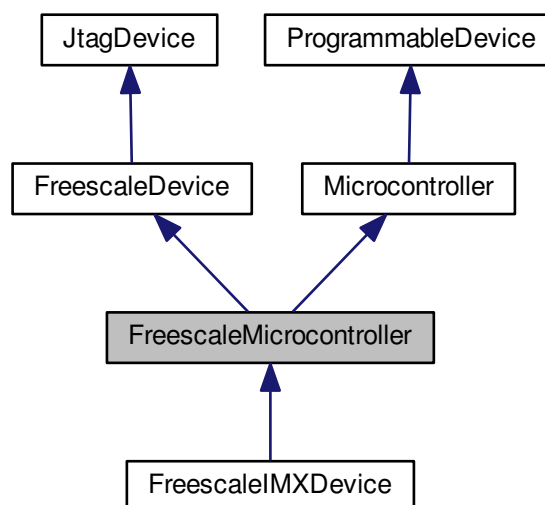
- [FreescaleMXSmartDMA.h](#)
- [FreescaleMXSmartDMA.cpp](#)

## 7.38 FreescaleMicrocontroller Class Reference

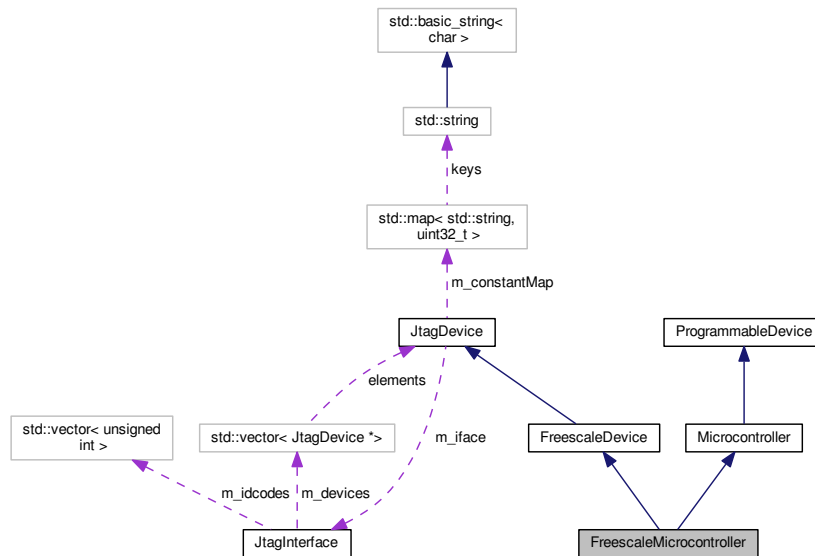
Generic base class for all Freescale MCUs.

```
#include <FreescaleMicrocontroller.h>
```

Inheritance diagram for FreescaleMicrocontroller:



Collaboration diagram for FreescaleMicrocontroller:



## Public Member Functions

- **FreescaleMicrocontroller** (unsigned int idcode, [JtagInterface](#) \*iface, size\_t pos, size\_t irlength)

## Additional Inherited Members

### 7.38.1 Detailed Description

Generic base class for all Freescale MCUs.

The documentation for this class was generated from the following files:

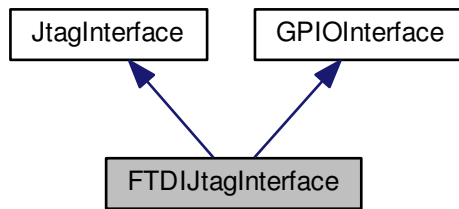
- [FreescaleMicrocontroller.h](#)
- [FreescaleMicrocontroller.cpp](#)

## 7.39 FTDIJtagInterface Class Reference

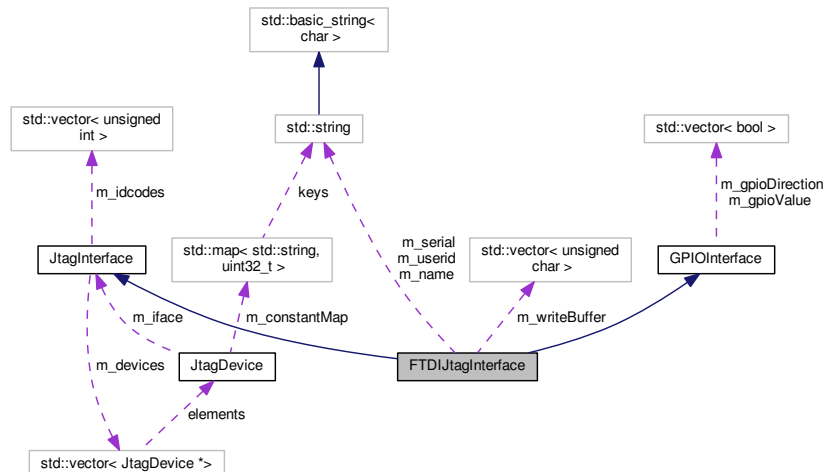
A JTAG adapter using the FTDI chipset, accessed through libftd2xx (proprietary driver from FTDI)

```
#include <FTDIJtagInterface.h>
```

Inheritance diagram for FTDIJtagInterface:



Collaboration diagram for FTDIJtagInterface:



## Public Member Functions

- [FTDIJtagInterface](#) (const std::string &serial, const std::string &layout)  
*Connects to an FTDI JTAG interface.*
- virtual [~FTDIJtagInterface](#) ()  
*Interface destructor.*
- virtual std::string [GetName](#) ()  
*Gets the manufacturer-assigned name for this programming adapter.*
- virtual std::string [GetSerial](#) ()  
*Gets the manufacturer-assigned serial number for this programming adapter.*
- virtual std::string [GetUserID](#) ()  
*Gets the user-assigned name for this programming adapter.*
- virtual int [GetFrequency](#) ()  
*Gets the clock frequency, in Hz, of the JTAG interface.*
- virtual void [ShiftData](#) (bool last\_tms, const unsigned char \*send\_data, unsigned char \*rcv\_data, size\_t count)

- Shifts data through TDI to TDO.*

  - virtual void [SendDummyClocks](#) (size\_t n)
 

*Sends the requested number of dummy clocks with TMS=0 and flushes the command to the interface.*
  - virtual void [SendDummyClocksDeferred](#) (size\_t n)
 

*Sends the requested number of dummy clocks with TMS=0 and does not flush the write pipeline.*
  - virtual void [Commit](#) ()
 

*Commits the outstanding transactions to the adapter.*
  - virtual bool [IsSplitScanSupported](#) ()
 

*Indicates if split (pipelined) DR scanning is supported.*
  - virtual bool [ShiftDataWriteOnly](#) (bool last\_tms, const unsigned char \*send\_data, unsigned char \*rcv\_data, size\_t count)
 

*Shifts data through TDI to TDO.*
  - virtual bool [ShiftDataReadOnly](#) (unsigned char \*rcv\_data, size\_t count)
 

*Reads data from a [ShiftDataWriteOnly\(\)](#) call.*
  - virtual void [ReadGpioState](#) ()
 

*Reads all of the device's GPIO pins into the internal buffer.*
  - virtual void [WriteGpioState](#) ()
 

*Writes all of the device's GPIO pin values to the device.*

### Static Public Member Functions

- static int [GetDefaultFrequency](#) (int index)
 

*Returns the default clock frequency of the Nth device.*
- static bool [IsJtagCapable](#) (int index)
 

*Checks if the requested FTDI device has a MPSSE (and is thus capable of being used for JTAG)*
- static std::string [GetSerialNumber](#) (int index)
 

*Returns the description of the Nth device.*
- static std::string [GetDescription](#) (int index)
 

*Returns the description of the Nth device.*
- static std::string [GetAPIVersion](#) ()
 

*Gets the version of the API.*
- static int [GetInterfaceCount](#) ()
 

*Gets the number of FTDI devices on the system (may include non-JTAG-capable devices)*

### Protected Member Functions

- virtual void [ShiftTMS](#) (bool tdi, const unsigned char \*send\_data, size\_t count)
 

*Shifts data into TMS to change TAP state.*
- void [GenerateShiftPacket](#) (const unsigned char \*send\_data, size\_t count, bool want\_read, bool last\_tms, std::vector< unsigned char > &cmd\_out)
 

*Generates the MPSSE commands for a shift operation.*
- void [DoReadback](#) (unsigned char \*rcv\_data, size\_t count)
 

*Reads back data from a prior transaction.*
- void [SharedCtorInit](#) (uint32\_t type, const std::string &layout)
 

*Shared initialization used by all constructors.*
- void [SyncCheck](#) ()
 

*Verifies that we're still in sync with the MPSSE.*
- void [ReadData](#) (void \*data, size\_t bytesToRead)
 

*Wrapper around FT\_Read() to work around some driver / API bugs.*
- void [WriteDataRaw](#) (const void \*data, size\_t bytesToWrite)

*Wrapper around FT\_Write() to push the provided data buffer to hardware.*

- void [WriteData](#) (const void \*data, size\_t bytesToWrite)

*Writes FTDI MPSSE data to the interface.*

- void [WriteData](#) (unsigned char cmd)

*Wrapper around [WriteData\(\)](#) to send a single byte.*

## Protected Attributes

- std::vector< unsigned char > [m\\_writeBuffer](#)  
*Buffer of data queued for the adapter, but not yet sent.*
- std::string [m\\_name](#)  
*Cached name of this adapter.*
- std::string [m\\_serial](#)  
*Cached serial number of this adapter.*
- std::string [m\\_userid](#)  
*Cached user ID of this adapter.*
- int [m\\_freq](#)  
*Cached clock frequency of this adapter.*
- void \* [m\\_context](#)  
*Libftdi interface handle.*

### 7.39.1 Detailed Description

A JTAG adapter using the FTDI chipset, accessed through libftd2xx (proprietary driver from FTDI)

This adapter supports split scanning and queues up to 4096 bytes of command+data before committing to hardware.

GPIO pin mapping:

| Index | Name            |
|-------|-----------------|
| 0     | GPIOL0 (ADBUS4) |
| 1     | GPIOL1 (ADBUS5) |
| 2     | GPIOL2 (ADBUS6) |
| 3     | GPIOL3 (ADBUS7) |
| 4     | GPIOH0 (ACBUS0) |
| 5     | GPIOH1 (ACBUS1) |
| 6     | GPIOH2 (ACBUS2) |
| 7     | GPIOH3 (ACBUS3) |
| 8     | GPIOH4 (ACBUS4) |
| 9     | GPIOH5 (ACBUS5) |
| 10    | GPIOH6 (ACBUS6) |
| 11    | GPIOH7 (ACBUS7) |

Supported layouts:

| Name | Example hardware                                                   | Pin configuration                   |
|------|--------------------------------------------------------------------|-------------------------------------|
| hs1  | Digilent JTAG-HS1, Digilent JTAG-SMT2, azo-nenberg's usb-jtag-mini | ADBUS7 is active-high output enable |

| Name    | Example hardware                                          | Pin configuration                                                                                   |
|---------|-----------------------------------------------------------|-----------------------------------------------------------------------------------------------------|
| jtagkey | Amontec JTAGkey, Bus Blaster w/ JTAGkey compatible buffer | ADBUS4 is active-low output enable, ACBUS0 is TRST_N, ACBUS2 is active-low output enable for TRST_N |

## 7.39.2 Constructor & Destructor Documentation

### 7.39.2.1 FTDIJtagInterface()

```
FTDIJtagInterface::FTDIJtagInterface (
 const std::string & serial,
 const std::string & layout)
```

Connects to an FTDI JTAG interface.

#### Exceptions

|                               |                                                                            |
|-------------------------------|----------------------------------------------------------------------------|
| <a href="#">JtagException</a> | if the connection could not be established or the serial number is invalid |
|-------------------------------|----------------------------------------------------------------------------|

#### Parameters

|               |                                           |
|---------------|-------------------------------------------|
| <i>serial</i> | Serial number of the device to connect to |
| <i>layout</i> | Adapter layout to use                     |

### 7.39.2.2 ~FTDIJtagInterface()

```
FTDIJtagInterface::~FTDIJtagInterface () [virtual]
```

Interface destructor.

Closes handles and disconnects from the adapter.

## 7.39.3 Member Function Documentation

### 7.39.3.1 Commit()

```
void FTDIJtagInterface::Commit () [virtual]
```

Commits the outstanding transactions to the adapter.

No-op unless the adapter supports queueing of multiple writes.

This function is automatically called when [SendDummyClocks\(\)](#) is called or any readback is performed. Most adapter classes will automatically call it when the transmit queue reaches a certain size.

This function can be called at any time to ensure all pending operations have executed.

## Exceptions

|                               |                  |
|-------------------------------|------------------|
| <a href="#">JtagException</a> | in case of error |
|-------------------------------|------------------|

Reimplemented from [JtagInterface](#).

## 7.39.3.2 DoReadback()

```
void FTDIJtagInterface::DoReadback (
 unsigned char * rcv_data,
 size_t count) [protected]
```

Reads back data from a prior transaction.

## Parameters

|                 |                        |
|-----------------|------------------------|
| <i>rcv_data</i> | Output data buffer     |
| <i>count</i>    | Number of bits to read |

## 7.39.3.3 GenerateShiftPacket()

```
void FTDIJtagInterface::GenerateShiftPacket (
 const unsigned char * send_data,
 size_t count,
 bool want_read,
 bool last_tms,
 std::vector< unsigned char > & cmd_out) [protected]
```

Generates the MPSSE commands for a shift operation.

## Parameters

|                  |                                                                                |
|------------------|--------------------------------------------------------------------------------|
| <i>send_data</i> | Data to send                                                                   |
| <i>count</i>     | Number of bits to send (not bytes)                                             |
| <i>want_read</i> | True if read data is needed, false for a write-only transaction                |
| <i>last_tms</i>  | TMS value to use at the end of the shift operation (all other bits have TMS=0) |
| <i>cmd_out</i>   | The generated command buffer                                                   |

## 7.39.3.4 GetAPIVersion()

```
string FTDIJtagInterface::GetAPIVersion () [static]
```

Gets the version of the API.



**Exceptions**

|                               |                          |
|-------------------------------|--------------------------|
| <a href="#">JtagException</a> | if the FTD2xx call fails |
|-------------------------------|--------------------------|

**Returns**

FTDI driver and API version

**7.39.3.5 GetDefaultFrequency()**

```
int FTDIJtagInterface::GetDefaultFrequency (
 int index) [static]
```

Returns the default clock frequency of the Nth device.

**Exceptions**

|                               |                                                   |
|-------------------------------|---------------------------------------------------|
| <a href="#">JtagException</a> | if the index is invalid or data could not be read |
|-------------------------------|---------------------------------------------------|

**Returns**

Clock frequency

**7.39.3.6 GetDescription()**

```
string FTDIJtagInterface::GetDescription (
 int index) [static]
```

Returns the description of the Nth device.

**Exceptions**

|                               |                                                   |
|-------------------------------|---------------------------------------------------|
| <a href="#">JtagException</a> | if the index is invalid or data could not be read |
|-------------------------------|---------------------------------------------------|

**Returns**

Description string

**7.39.3.7 GetInterfaceCount()**

```
int FTDIJtagInterface::GetInterfaceCount () [static]
```

Gets the number of FTDI devices on the system (may include non-JTAG-capable devices)

**Exceptions**

|                               |                          |
|-------------------------------|--------------------------|
| <a href="#">JtagException</a> | if the FTD2xx call fails |
|-------------------------------|--------------------------|

**Returns**

Number of interfaces found

**7.39.3.8 GetSerialNumber()**

```
string FTDIJtagInterface::GetSerialNumber (
 int index) [static]
```

Returns the description of the Nth device.

**Parameters**

|              |                                        |
|--------------|----------------------------------------|
| <i>index</i> | Zero-based index of the device to test |
|--------------|----------------------------------------|

**Exceptions**

|                               |                                                   |
|-------------------------------|---------------------------------------------------|
| <a href="#">JtagException</a> | if the index is invalid or data could not be read |
|-------------------------------|---------------------------------------------------|

**Returns**

Serial number string

**7.39.3.9 IsJtagCapable()**

```
bool FTDIJtagInterface::IsJtagCapable (
 int index) [static]
```

Checks if the requested FTDI device has a MPSSE (and is thus capable of being used for JTAG)

Note that this function cannot tell if an MPSSE-capable chipset is actually configured for use as JTAG or as something else.

**Exceptions**

|                               |                                                   |
|-------------------------------|---------------------------------------------------|
| <a href="#">JtagException</a> | if the index is invalid or data could not be read |
|-------------------------------|---------------------------------------------------|

**Parameters**

|              |                                        |
|--------------|----------------------------------------|
| <i>index</i> | Zero-based index of the device to test |
|--------------|----------------------------------------|

**Returns**

True if the device has a MPSSE, false otherwise.

**7.39.3.10 IsSplitScanSupported()**

```
bool FTDIJtagInterface::IsSplitScanSupported () [virtual]
```

Indicates if split (pipelined) DR scanning is supported.

Split scanning allows the write halves of several scan operations to take place in one driver-level write call, followed by the read halves in order, to reduce the impact of driver/bus latency on throughput.

If split scanning is not supported, [ScanDRSplitWrite\(\)](#) will behave identically to [ScanDR\(\)](#) and [ScanDRSplitRead\(\)](#) will be a no-op. This ensures that using the split write commands will work correctly regardless of whether the adapter supports split scanning in hardware.

Reimplemented from [JtagInterface](#).

**7.39.3.11 ReadData()**

```
void FTDIJtagInterface::ReadData (
 void * data,
 size_t bytesToRead) [protected]
```

Wrapper around FT\_Read() to work around some driver / API bugs.

**Exceptions**

|                               |            |
|-------------------------------|------------|
| <a href="#">JtagException</a> | on failure |
|-------------------------------|------------|

**Parameters**

|                    |                         |
|--------------------|-------------------------|
| <i>data</i>        | Data to write           |
| <i>bytesToRead</i> | Number of bytes to read |

**7.39.3.12 SendDummyClocks()**

```
void FTDIJtagInterface::SendDummyClocks (
 size_t n) [virtual]
```

Sends the requested number of dummy clocks with TMS=0 and flushes the command to the interface.

Since dummy clocks are often used as a delay element for programming algorithms etc, this function flushes the write buffer to ensure immediate execution.

**Exceptions**

|                               |                                           |
|-------------------------------|-------------------------------------------|
| <a href="#">JtagException</a> | may be thrown if the scan operation fails |
|-------------------------------|-------------------------------------------|

**Parameters**

|          |                                |
|----------|--------------------------------|
| <i>n</i> | Number of dummy clocks to send |
|----------|--------------------------------|

Implements [JtagInterface](#).

**7.39.3.13 SendDummyClocksDeferred()**

```
void FTDIJtagInterface::SendDummyClocksDeferred (
 size_t n) [virtual]
```

Sends the requested number of dummy clocks with TMS=0 and does not flush the write pipeline.

**Exceptions**

|                               |                                           |
|-------------------------------|-------------------------------------------|
| <a href="#">JtagException</a> | may be thrown if the scan operation fails |
|-------------------------------|-------------------------------------------|

**Parameters**

|          |                                |
|----------|--------------------------------|
| <i>n</i> | Number of dummy clocks to send |
|----------|--------------------------------|

Reimplemented from [JtagInterface](#).

**7.39.3.14 ShiftData()**

```
void FTDIJtagInterface::ShiftData (
 bool last_tms,
 const unsigned char * send_data,
 unsigned char * rcv_data,
 size_t count) [virtual]
```

Shifts data through TDI to TDO.

The LSB of `send_data[0]` is sent first; the MSB of `send_data[0]` is followed by the LSB of `send_data[1]`.

**Parameters**

|                  |                                         |
|------------------|-----------------------------------------|
| <i>last_tms</i>  | Different TMS value to use for last bit |
| <i>send_data</i> | Data to shift into TDI                  |
| <i>rcv_data</i>  | Data to shift out of TDO (may be NULL)  |
| <i>count</i>     | Number of bits to shift                 |

Implements [JtagInterface](#).

### 7.39.3.15 ShiftDataReadOnly()

```
bool FTDIJtagInterface::ShiftDataReadOnly (
 unsigned char * rcv_data,
 size_t count) [virtual]
```

Reads data from a [ShiftDataWriteOnly\(\)](#) call.

For more information on split (pipelined) scan operations see [ShiftDataWriteOnly\(\)](#).

#### Returns

True if the read was executed, false if a no-op

#### Parameters

|                 |                                        |
|-----------------|----------------------------------------|
| <i>rcv_data</i> | Data to shift out of TDO (may be NULL) |
| <i>count</i>    | Number of bits to shift                |

Reimplemented from [JtagInterface](#).

### 7.39.3.16 ShiftDataWriteOnly()

```
bool FTDIJtagInterface::ShiftDataWriteOnly (
 bool last_tms,
 const unsigned char * send_data,
 unsigned char * rcv_data,
 size_t count) [virtual]
```

Shifts data through TDI to TDO.

The LSB of `send_data[0]` is sent first; the MSB of `send_data[0]` is followed by the LSB of `send_data[1]`.

If split (pipelined) scanning is supported by the adapter, this function performs the write half of the shift operation only; the read is buffered in the JTAG adapter and no readback is performed until [ShiftDataReadOnly\(\)](#) is called. This allows several shift operations to occur in sequence without incurring a USB turnaround delay or other driver latency overhead for each shift operation.

If split scanning is not supported this call is equivalent to [ShiftData\(\)](#) and [ShiftDataReadOnly\(\)](#) is a no-op.

This function MUST be followed by either another [ShiftDataWriteOnly\(\)](#) call, a [ShiftTMS\(\)](#) call, or a [ShiftDataReadOnly\(\)](#) call. There must be exactly one [ShiftDataReadOnly\(\)](#) call for each [ShiftDataWriteOnly\(\)](#) call and they must be in order with the same `rcv_data` and count values. The result of doing otherwise is undefined.

#### Returns

True if the read was deferred, false if not

## Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <i>last_tms</i>  | Different TMS value to use for last bit |
| <i>send_data</i> | Data to shift into TDI                  |
| <i>rcv_data</i>  | Data to shift out of TDO (may be NULL)  |
| <i>count</i>     | Number of bits to shift                 |

Reimplemented from [JtagInterface](#).

## 7.39.3.17 ShiftTMS()

```
void FTDIJtagInterface::ShiftTMS (
 bool tdi,
 const unsigned char * send_data,
 size_t count) [protected], [virtual]
```

Shifts data into TMS to change TAP state.

This is no longer a public API operation. It can only be accessed via the state-level interface.

Implementations of this class may choose to implement EITHER this function (and use the default JtagInterface-provided state-level functions) OR override this function with a private no-op stub and override the state-level functions instead.

## Exceptions

|                               |                                           |
|-------------------------------|-------------------------------------------|
| <a href="#">JtagException</a> | may be thrown if the scan operation fails |
|-------------------------------|-------------------------------------------|

## Parameters

|                  |                                                                                       |
|------------------|---------------------------------------------------------------------------------------|
| <i>tdi</i>       | Constant tdi value (normally 0)                                                       |
| <i>send_data</i> | Data to shift into TMS. Bit ordering is the same as for <a href="#">ShiftData()</a> . |
| <i>count</i>     | Number of bits to shift                                                               |

Implements [JtagInterface](#).

## 7.39.3.18 SyncCheck()

```
void FTDIJtagInterface::SyncCheck () [protected]
```

Verifies that we're still in sync with the MPSSE.

## Exceptions

|                               |                           |
|-------------------------------|---------------------------|
| <a href="#">JtagException</a> | if an FTDI API call fails |
|-------------------------------|---------------------------|

**7.39.3.19 WriteData()** [1/2]

```
void FTDIJtagInterface::WriteData (
 const void * data,
 size_t bytesToWrite) [protected]
```

Writes FTDI MPSSE data to the interface.

Writes may be deferred until [Commit\(\)](#) is called to improve performance.

**Exceptions**

|                               |            |
|-------------------------------|------------|
| <a href="#">JtagException</a> | on failure |
|-------------------------------|------------|

**Parameters**

|                     |                          |
|---------------------|--------------------------|
| <i>data</i>         | Data to write            |
| <i>bytesToWrite</i> | Number of bytes to write |

**7.39.3.20 WriteData()** [2/2]

```
void FTDIJtagInterface::WriteData (
 unsigned char cmd) [protected]
```

Wrapper around [WriteData\(\)](#) to send a single byte.

**Exceptions**

|                               |            |
|-------------------------------|------------|
| <a href="#">JtagException</a> | on failure |
|-------------------------------|------------|

**Parameters**

|            |                          |
|------------|--------------------------|
| <i>cmd</i> | The single byte to write |
|------------|--------------------------|

**7.39.3.21 WriteDataRaw()**

```
void FTDIJtagInterface::WriteDataRaw (
 const void * data,
 size_t bytesToWrite) [protected]
```

Wrapper around [FT\\_Write\(\)](#) to push the provided data buffer to hardware.

Performs repeated write calls as needed to ensure the entire buffer is written.

## Exceptions

|                               |            |
|-------------------------------|------------|
| <a href="#">JtagException</a> | on failure |
|-------------------------------|------------|

## Parameters

|                     |                          |
|---------------------|--------------------------|
| <i>data</i>         | Data to write            |
| <i>bytesToWrite</i> | Number of bytes to write |

The documentation for this class was generated from the following files:

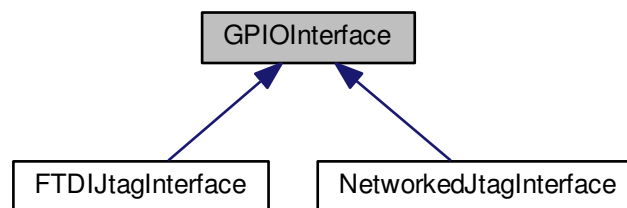
- [FTDIJtagInterface.h](#)
- [FTDIJtagInterface.cpp](#)

## 7.40 GPIOInterface Class Reference

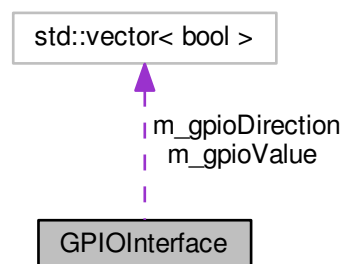
A GPIO bitbang interface. Many JTAG adapters have uncommitted GPIOs which may be used for test purposes.

```
#include <GPIOInterface.h>
```

Inheritance diagram for GPIOInterface:



Collaboration diagram for GPIOInterface:





## Public Member Functions

- int [GetGpioCount](#) ()  
*Gets the number of GPIO pins on the device.*
- virtual void [ReadGpioState](#) ()=0  
*Reads all of the device's GPIO pins into the internal buffer.*
- virtual void [WriteGpioState](#) ()=0  
*Writes all of the device's GPIO pin values to the device.*
- void [SetGpioDirectionDeferred](#) (int pin, bool output)  
*Updates the direction of a GPIO pin but does not push the changes to the device.*
- void [SetGpioValueDeferred](#) (int pin, bool value)  
*Updates the value of a GPIO pin but does not push the changes to the device.*
- bool [GetGpioValueCached](#) (int pin)  
*Reads the cached value of a GPIO pin but does not poll the device.*
- void [SetGpioDirection](#) (int pin, bool output)  
*Updates the direction of a GPIO pin and pushes changes to the device immediately.*
- void [SetGpioValue](#) (int pin, bool value)  
*Updates the value of a GPIO pin and pushes changes to the device immediately.*
- bool [GetGpioValue](#) (int pin)  
*Reads the current value of a GPIO pin, polling the device.*
- bool [GetGpioDirection](#) (int pin)  
*Reads the current direction of a GPIO pin.*

## Protected Attributes

- std::vector< bool > [m\\_gpioValue](#)  
*Value bits (1=high, contains the read value for inputs and the write value for outputs)*
- std::vector< bool > [m\\_gpioDirection](#)  
*Direction bits (1=output)*

### 7.40.1 Detailed Description

A GPIO bitbang interface. Many JTAG adapters have uncommitted GPIOs which may be used for test purposes.

The documentation for this class was generated from the following files:

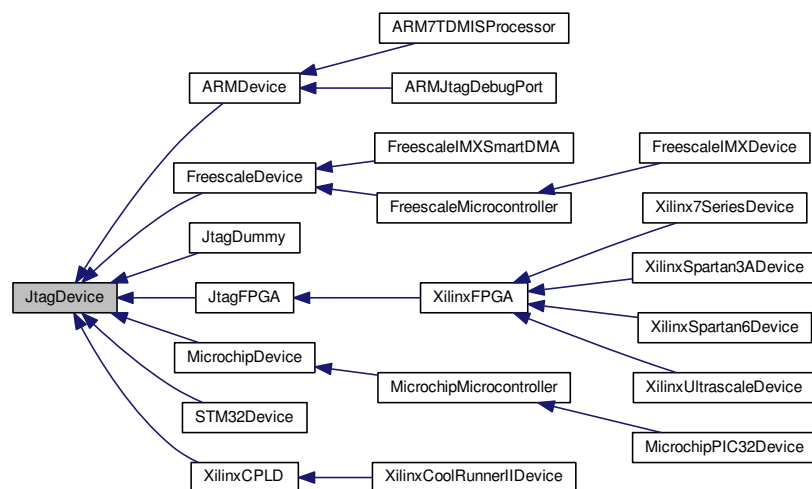
- [GPIOInterface.h](#)
- [GPIOInterface.cpp](#)

## 7.41 JtagDevice Class Reference

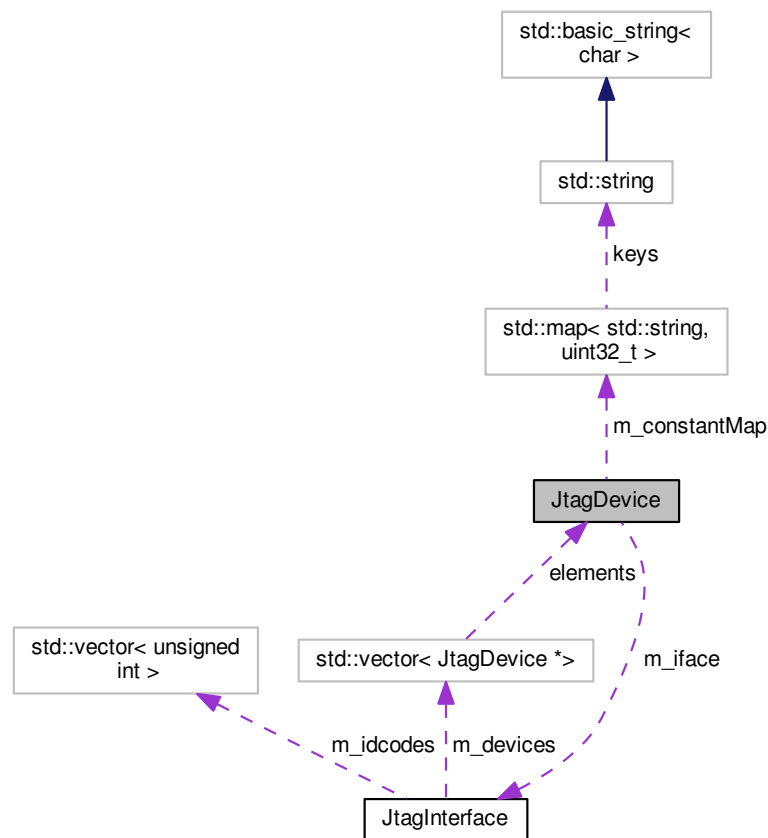
A single TAP in the JTAG chain. May not correspond 1:1 with physical silicon dies.

```
#include <JtagDevice.h>
```

Inheritance diagram for JtagDevice:



Collaboration diagram for JtagDevice:



## Public Member Functions

- `JtagDevice` (`uint32_t` idcode, `JtagInterface` \*iface, `size_t` pos, `size_t` irlength)  
*Initializes this device.*
- virtual `~JtagDevice` ()  
*Default virtual destructor.*
- virtual `std::string` `GetDescription` ()=0  
*Gets a human-readable description of this device.*
- `unsigned int` `GetIDCode` ()  
*Returns the JEDEC ID code of this device.*
- virtual void `PrintInfo` ()
- virtual void `PostInitProbes` (`bool` quiet)=0  
*Does a post-initialization probe of the device to read debug ROMs etc.*
- `bool` `LookupConstant` (`std::string` name, `uint32_t` &value)
- void `SetIR` (`const unsigned char` \*data)
- void `SetIRDeferred` (`const unsigned char` \*data)
- void `SetIR` (`const unsigned char` \*data, `int` count)  
*Wrapper around `JtagInterface::SetIR()`*
- void `SetIRDeferred` (`const unsigned char` \*data, `int` count)

- void [SetIR](#) (const unsigned char \*data, unsigned char \*data\_out, int count)  
*Wrapper around [JtagInterface::SetIR\(\)](#)*
- void [ScanDR](#) (const unsigned char \*send\_data, unsigned char \*rcv\_data, int count)  
*Wrapper around [JtagInterface::ScanDR\(\)](#)*
- void [ScanDRDeferred](#) (const unsigned char \*send\_data, int count)  
*Wrapper around [JtagInterface::ScanDRDeferred\(\)](#)*
- bool [IsSplitScanSupported](#) ()  
*Wrapper around [JtagInterface::IsSplitScanSupported\(\)](#)*
- void [ScanDRSplitWrite](#) (const unsigned char \*send\_data, unsigned char \*rcv\_data, int count)  
*Wrapper around [JtagInterface::ScanDRSplitWrite\(\)](#)*
- void [ScanDRSplitRead](#) (unsigned char \*rcv\_data, int count)  
*Wrapper around [JtagInterface::ScanDRSplitRead\(\)](#)*
- void [SendDummyClocks](#) (int n)  
*Wrapper around [JtagInterface::SendDummyClocks\(\)](#)*
- void [SendDummyClocksDeferred](#) (int n)  
*Wrapper around [JtagInterface::SendDummyClocksDeferred\(\)](#)*
- void [ResetToldle](#) ()  
*Wrapper around [JtagInterface::ResetToldle\(\)](#)*
- void [Commit](#) ()  
*Wrapper around [JtagInterface::Commit\(\)](#)*
- size\_t [GetIRLength](#) ()
- size\_t [GetChainIndex](#) ()
- uint32\_t [GetIdcode](#) ()
- void [EnterShiftDR](#) ()  
*Wrapper around [JtagInterface::EnterShiftDR\(\)](#)*
- void [ShiftData](#) (const unsigned char \*send\_data, unsigned char \*rcv\_data, int count)  
*Wrapper around [JtagInterface::ShiftData\(\)](#)*

### Static Public Member Functions

- static [JtagDevice](#) \* [CreateDevice](#) (uint32\_t idcode, [JtagInterface](#) \*iface, size\_t pos)  
*Creates a [JtagDevice](#) given an ID code.*

### Protected Attributes

- size\_t [m\\_irlength](#)  
*Length of this device's instruction register, in bits.*
- uint32\_t [m\\_idcode](#)  
*32-bit JEDEC ID code of this device*
- [JtagInterface](#) \* [m\\_iface](#)  
*The JTAGInterface associated with this device.*
- size\_t [m\\_pos](#)  
*Position of this device in the interface's scan chain.*
- unsigned char [m\\_cachedIR](#) [4]  
*Cached IR.*
- std::map< std::string, uint32\_t > [m\\_constantMap](#)

### 7.41.1 Detailed Description

A single TAP in the JTAG chain. May not correspond 1:1 with physical silicon dies.

### 7.41.2 Constructor & Destructor Documentation

#### 7.41.2.1 JtagDevice()

```
JtagDevice::JtagDevice (
 uint32_t idcode,
 JtagInterface * iface,
 size_t pos,
 size_t irlength)
```

Initializes this device.

NOTE: Do not do probes/scans of the chain during the constructor, because we haven't initialized all TAPs yet.

Any initialization that involves querying the chain should be done in [PostInitProbes\(\)](#).

#### Parameters

|                 |                                                       |
|-----------------|-------------------------------------------------------|
| <i>idcode</i>   | The ID code of this device                            |
| <i>iface</i>    | The JTAG adapter this device was discovered on        |
| <i>pos</i>      | Position in the chain that this device was discovered |
| <i>irlength</i> | Length of the JTAG instruction register               |

### 7.41.3 Member Function Documentation

#### 7.41.3.1 Commit()

```
void JtagDevice::Commit ()
```

Wrapper around [JtagInterface::Commit\(\)](#)

See [JtagInterface](#) documentation for more details.

#### 7.41.3.2 CreateDevice()

```
JtagDevice * JtagDevice::CreateDevice (
 uint32_t idcode,
 JtagInterface * iface,
 size_t pos) [static]
```

Creates a [JtagDevice](#) given an ID code.

**Parameters**

|               |                                                       |
|---------------|-------------------------------------------------------|
| <i>idcode</i> | The ID code of this device                            |
| <i>iface</i>  | The JTAG adapter this device was discovered on        |
| <i>pos</i>    | Position in the chain that this device was discovered |

**Returns**

A valid [JtagDevice](#) object, or NULL if the vendor ID was not recognized.

**7.41.3.3 EnterShiftDR()**

```
void JtagDevice::EnterShiftDR ()
```

Wrapper around [JtagInterface::EnterShiftDR\(\)](#)

See [JtagInterface](#) documentation for more details.

**7.41.3.4 GetDescription()**

```
virtual std::string JtagDevice::GetDescription () [pure virtual]
```

Gets a human-readable description of this device.

Example: "Xilinx XC6SLX45 stepping 3"

**Returns**

Device description

Implemented in [MicrochipPIC32Device](#), [Xilinx7SeriesDevice](#), [XilinxUltrascaleDevice](#), [XilinxSpartan6Device](#), [XilinxSpartan3ADevice](#), [XilinxCoolRunnerIIDevice](#), [ARMJtagDebugPort](#), [FreescaleIMXDevice](#), [STM32Device](#), [FreescaleIMXSmartDMA](#), and [JtagDummy](#).

**7.41.3.5 IsSplitScanSupported()**

```
bool JtagDevice::IsSplitScanSupported ()
```

Wrapper around [JtagInterface::IsSplitScanSupported\(\)](#)

See [JtagInterface](#) documentation for more details.

**7.41.3.6 PostInitProbes()**

```
virtual void JtagDevice::PostInitProbes (
 bool quiet) [pure virtual]
```

Does a post-initialization probe of the device to read debug ROMs etc.

## Parameters

|              |                                                           |
|--------------|-----------------------------------------------------------|
| <i>quiet</i> | Do minimal probing to avoid triggering security lockdowns |
|--------------|-----------------------------------------------------------|

Implemented in [MicrochipPIC32Device](#), [ARMJtagDebugPort](#), [XilinxCoolRunnerIIDevice](#), [FreescaleIMXDevice](#), [S←TM32Device](#), [ARM7TDMISProcessor](#), [FreescaleIMXSmartDMA](#), [JtagDummy](#), and [XilinxFPGA](#).

#### 7.41.3.7 ResetToIdle()

```
void JtagDevice::ResetToIdle ()
```

Wrapper around [JtagInterface::ResetToIdle\(\)](#)

See [JtagInterface](#) documentation for more details.

#### 7.41.3.8 ScanDR()

```
void JtagDevice::ScanDR (
 const unsigned char * send_data,
 unsigned char * rcv_data,
 int count)
```

Wrapper around [JtagInterface::ScanDR\(\)](#)

See [JtagInterface](#) documentation for more details.

#### 7.41.3.9 ScanDRDeferred()

```
void JtagDevice::ScanDRDeferred (
 const unsigned char * send_data,
 int count)
```

Wrapper around [JtagInterface::ScanDRDeferred\(\)](#)

See [JtagInterface](#) documentation for more details.

#### 7.41.3.10 ScanDRSplitRead()

```
void JtagDevice::ScanDRSplitRead (
 unsigned char * rcv_data,
 int count)
```

Wrapper around [JtagInterface::ScanDRSplitRead\(\)](#)

See [JtagInterface](#) documentation for more details.

#### 7.41.3.11 ScanDRSplitWrite()

```
void JtagDevice::ScanDRSplitWrite (
 const unsigned char * send_data,
 unsigned char * rcv_data,
 int count)
```

Wrapper around [JtagInterface::ScanDRSplitWrite\(\)](#)

See [JtagInterface](#) documentation for more details.

#### 7.41.3.12 SendDummyClocks()

```
void JtagDevice::SendDummyClocks (
 int n)
```

Wrapper around [JtagInterface::SendDummyClocks\(\)](#)

See [JtagInterface](#) documentation for more details.

#### 7.41.3.13 SendDummyClocksDeferred()

```
void JtagDevice::SendDummyClocksDeferred (
 int n)
```

Wrapper around [JtagInterface::SendDummyClocksDeferred\(\)](#)

See [JtagInterface](#) documentation for more details.

#### 7.41.3.14 SetIR() [1/2]

```
void JtagDevice::SetIR (
 const unsigned char * data,
 int count)
```

Wrapper around [JtagInterface::SetIR\(\)](#)

See [JtagInterface](#) documentation for more details.

#### 7.41.3.15 SetIR() [2/2]

```
void JtagDevice::SetIR (
 const unsigned char * data,
 unsigned char * data_out,
 int count)
```

Wrapper around [JtagInterface::SetIR\(\)](#)

See [JtagInterface](#) documentation for more details.



## 7.41.3.16 ShiftData()

```
void JtagDevice::ShiftData (
 const unsigned char * send_data,
 unsigned char * rcv_data,
 int count)
```

Wrapper around [JtagInterface::ShiftData\(\)](#)

See [JtagInterface](#) documentation for more details.

The documentation for this class was generated from the following files:

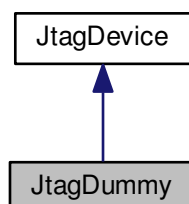
- [JtagDevice.h](#)
- [JtagDevice.cpp](#)

## 7.42 JtagDummy Class Reference

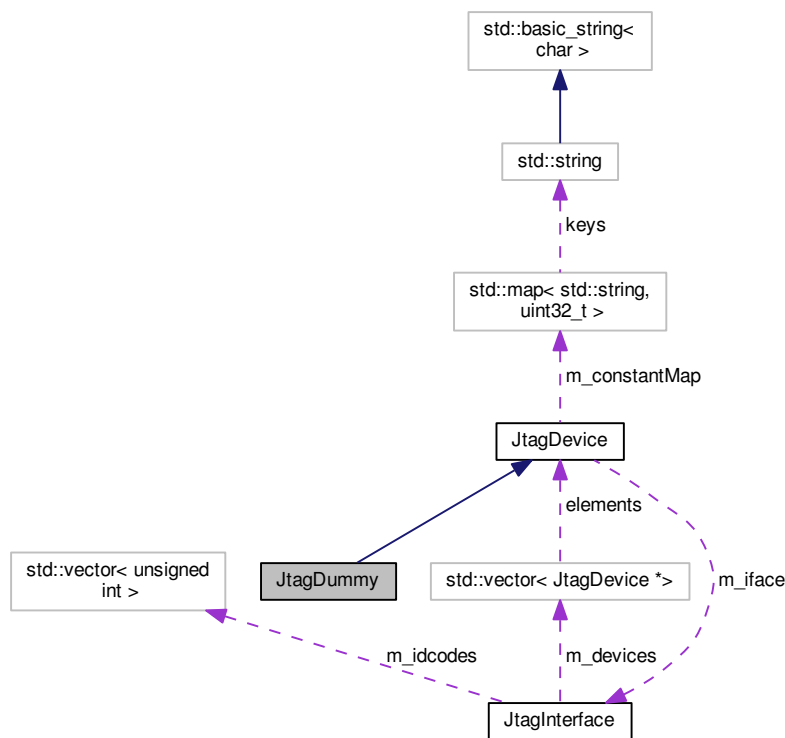
An unknown device (IDCODE not recognized, or no IDCODE present) on a JTAG chain.

```
#include <JtagDummy.h>
```

Inheritance diagram for JtagDummy:



Collaboration diagram for JtagDummy:



## Public Member Functions

- `JtagDummy` (unsigned int idcode, `JtagInterface` \*iface, size\_t pos, size\_t irlength)  
*Initializes this device.*
- virtual `~JtagDummy` ()  
*Default virtual destructor.*
- virtual `std::string` `GetDescription` ()  
*Gets a human-readable description of this device.*
- virtual void `PostInitProbes` (bool quiet)  
*Does a post-initialization probe of the device to read debug ROMs etc.*

## Additional Inherited Members

### 7.42.1 Detailed Description

An unknown device (IDCODE not recognized, or no IDCODE present) on a JTAG chain.

Just exists to take up a slot in the chain and consume IR bits.

## 7.42.2 Constructor & Destructor Documentation

### 7.42.2.1 JtagDummy()

```
JtagDummy::JtagDummy (
 unsigned int idcode,
 JtagInterface * iface,
 size_t pos,
 size_t irlength)
```

Initializes this device.

#### Parameters

|                 |                                                       |
|-----------------|-------------------------------------------------------|
| <i>idcode</i>   | The ID code of this device                            |
| <i>iface</i>    | The JTAG adapter this device was discovered on        |
| <i>pos</i>      | Position in the chain that this device was discovered |
| <i>irlength</i> | Length of the JTAG instruction register               |

## 7.42.3 Member Function Documentation

### 7.42.3.1 GetDescription()

```
string JtagDummy::GetDescription () [virtual]
```

Gets a human-readable description of this device.

Example: "Xilinx XC6SLX45 stepping 3"

#### Returns

Device description

Implements [JtagDevice](#).

### 7.42.3.2 PostInitProbes()

```
void JtagDummy::PostInitProbes (
 bool quiet) [virtual]
```

Does a post-initialization probe of the device to read debug ROMs etc.

## Parameters

|              |                                                           |
|--------------|-----------------------------------------------------------|
| <i>quiet</i> | Do minimal probing to avoid triggering security lockdowns |
|--------------|-----------------------------------------------------------|

Implements [JtagDevice](#).

The documentation for this class was generated from the following files:

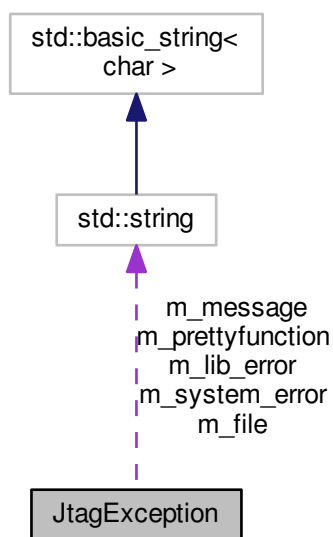
- [JtagDummy.h](#)
- [JtagDummy.cpp](#)

## 7.43 JtagException Class Reference

Base class for all exceptions thrown by libjtaghal.

```
#include <JtagException.h>
```

Collaboration diagram for JtagException:



### Public Member Functions

- [JtagException](#) (std::string message, std::string library\_error, std::string prettyfunction, std::string file, int line)  
*Constructor for an exception object.*
- std::string [GetDescription](#) () const  
*Gets the description of this exception.*

## Static Public Member Functions

- static void [ThrowDummyException](#) ()  
*Throws an exception to test error handling code.*

## Protected Attributes

- `std::string m_message`  
*Error message.*
- `std::string m_system_error`  
*String version of errno.*
- `std::string m_lib_error`  
*String version of library error.*
- `std::string m_prettyfunction`  
*Pretty-printed function name.*
- `std::string m_file`  
*File name.*
- `int m_line`  
*Line number.*

### 7.43.1 Detailed Description

Base class for all exceptions thrown by libjtaghal.

### 7.43.2 Constructor & Destructor Documentation

#### 7.43.2.1 JtagException()

```
JtagException::JtagException (
 std::string message,
 std::string library_error,
 std::string prettyfunction,
 std::string file,
 int line)
```

Constructor for an exception object.

The [JtagExceptionWrapper\(\)](#) macro may be used to pass the last three parameters automatically.

#### Parameters

|                       |                                                                              |
|-----------------------|------------------------------------------------------------------------------|
| <i>message</i>        | Human-readable error message. Include as much detail as reasonably possible. |
| <i>library_error</i>  | Human-readable error string returned from a library (ex: libusb)             |
| <i>prettyfunction</i> | Pretty-printed name of the current function. Pass <b>PRETTY_FUNCTION</b>     |
| <i>file</i>           | The current source file. Pass <b>FILE</b>                                    |
| <i>line</i>           | The current line number. Pass <b>LINE</b>                                    |

### 7.43.3 Member Function Documentation

#### 7.43.3.1 GetDescription()

```
string JtagException::GetDescription () const
```

Gets the description of this exception.

The file name is truncated to the last 3 components for cleaner output.

Example output:

```
JtagException object thrown from static void JtagException::ThrowDummyException()
File : ../src/jtaghal/JtagException.cpp/
Line : 136
Library error:
System error : Permission denied
Message : Test exception
```

#### Returns

Printable exception description

The documentation for this class was generated from the following files:

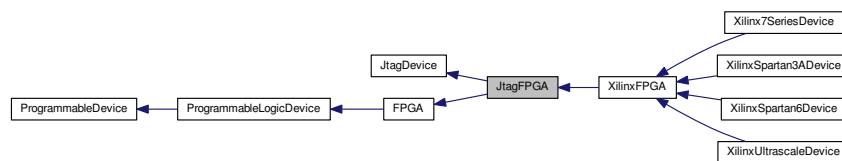
- [JtagException.h](#)
- [JtagException.cpp](#)

## 7.44 JtagFPGA Class Reference

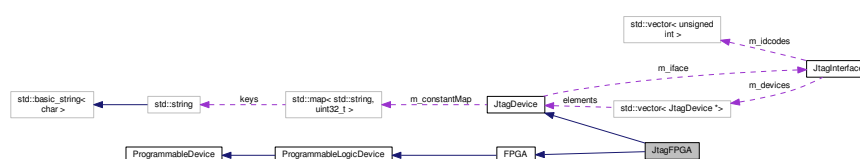
Abstract base class for all JTAG-programmed FPGAs.

```
#include <JtagFPGA.h>
```

Inheritance diagram for JtagFPGA:



Collaboration diagram for JtagFPGA:



## Public Member Functions

- [JtagFPGA](#) (unsigned int idcode, [JtagInterface](#) \*iface, size\_t pos, size\_t irlength)  
*Initializes this device.*
- virtual [~JtagFPGA](#) ()  
*Default virtual destructor.*
- virtual size\_t [GetNumUserInstructions](#) ()=0  
*Get the number of JTAG instructions which are routed to [FPGA](#) fabric.*
- virtual void [SelectUserInstruction](#) (size\_t index)=0  
*Sets the instruction register to the specified user instruction.*
- bool [GetUserVIDPID](#) (unsigned int &idVendor, unsigned int &idProduct)  
*Gets the vendor/product code in USER1 (if implemented)*

## Additional Inherited Members

### 7.44.1 Detailed Description

Abstract base class for all JTAG-programmed FPGAs.

### 7.44.2 Constructor & Destructor Documentation

#### 7.44.2.1 JtagFPGA()

```
JtagFPGA::JtagFPGA (
 unsigned int idcode,
 JtagInterface * iface,
 size_t pos,
 size_t irlength)
```

Initializes this device.

#### Parameters

|                 |                                                       |
|-----------------|-------------------------------------------------------|
| <i>idcode</i>   | The ID code of this device                            |
| <i>iface</i>    | The JTAG adapter this device was discovered on        |
| <i>pos</i>      | Position in the chain that this device was discovered |
| <i>irlength</i> | Length of the JTAG instruction register               |

### 7.44.3 Member Function Documentation

### 7.44.3.1 GetUserVIDPID()

```
bool JtagFPGA::GetUserVIDPID (
 unsigned int & idVendor,
 unsigned int & idProduct)
```

Gets the vendor/product code in USER1 (if implemented)

Reference: <https://github.com/azonenberg/jtaghal/wiki/FPGA-debug>

#### Returns

false on error, or if no USER instructions

The documentation for this class was generated from the following files:

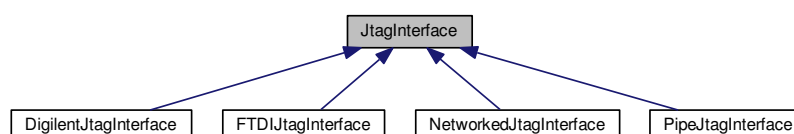
- [JtagFPGA.h](#)
- [JtagFPGA.cpp](#)

## 7.45 JtagInterface Class Reference

Abstract representation of a JTAG adapter.

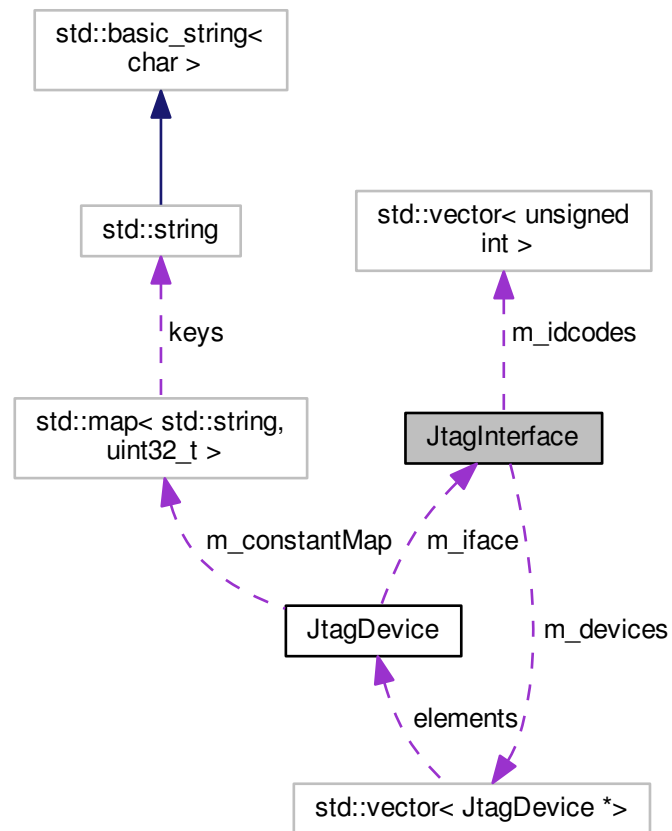
```
#include <JtagInterface.h>
```

Inheritance diagram for JtagInterface:





Collaboration diagram for JtagInterface:



## Public Member Functions

- [JtagInterface \(\)](#)  
*Default constructor.*
- virtual [~JtagInterface \(\)](#)  
*Interface destructor.*
- virtual std::string [GetName \(\)](#)=0  
*Gets the manufacturer-assigned name for this programming adapter.*
- virtual std::string [GetSerial \(\)](#)=0  
*Gets the manufacturer-assigned serial number for this programming adapter, if any.*
- virtual std::string [GetUserID \(\)](#)=0  
*Gets the user-assigned name for this JTAG adapter, if any.*
- virtual int [GetFrequency \(\)](#)=0  
*Gets the clock frequency, in Hz, of the JTAG interface.*
- virtual void [Commit \(\)](#)  
*Commits the outstanding transactions to the adapter.*
- virtual void [ShiftData](#) (bool last\_tms, const unsigned char \*send\_data, unsigned char \*rcv\_data, size\_t count)=0

- Shifts data through TDI to TDO.*

  - virtual void [SendDummyClocks](#) (size\_t n)=0

*Sends the requested number of dummy clocks with TMS=0 and flushes the command to the interface.*
- virtual void [SendDummyClocksDeferred](#) (size\_t n)

*Sends the requested number of dummy clocks with TMS=0 and does not flush the write pipeline.*
- virtual bool [IsSplitScanSupported](#) ()

*Indicates if split (pipelined) DR scanning is supported.*
- virtual bool [ShiftDataWriteOnly](#) (bool last\_tms, const unsigned char \*send\_data, unsigned char \*rcv\_data, size\_t count)

*Shifts data through TDI to TDO.*
- virtual bool [ShiftDataReadOnly](#) (unsigned char \*rcv\_data, size\_t count)

*Reads data from a [ShiftDataWriteOnly\(\)](#) call.*
- virtual void [TestLogicReset](#) ()

*Enters Test-Logic-Reset state by shifting six ones into TMS.*
- virtual void [EnterShiftIR](#) ()

*Enters Shift-IR state from Run-Test-Idle state.*
- virtual void [LeaveExit1IR](#) ()

*Leaves Exit1-IR state and returns to Run-Test-Idle.*
- virtual void [EnterShiftDR](#) ()

*Enters Shift-DR state from Run-Test-Idle state.*
- virtual void [LeaveExit1DR](#) ()

*Leaves Exit1-DR state and returns to Run-Test-Idle.*
- virtual void [ResetToIdle](#) ()

*Resets the TAP and enters Run-Test-Idle state.*
- void [InitializeChain](#) (bool quiet=false)

*Initializes the scan chain and gets the number of devices present.*
- size\_t [GetDeviceCount](#) ()

*Returns the number of devices in the scan chain.*
- unsigned int [GetIDCode](#) (unsigned int device)

*Returns the ID for the supplied device (zero-based indexing)*
- [JtagDevice](#) \* [GetDevice](#) (unsigned int device)

*Gets the Nth device in the chain.*
- void [SetIR](#) (unsigned int device, const unsigned char \*data, size\_t count)

*Sets the IR for a specific device in the chain.*
- void [SetIRDeferred](#) (unsigned int device, const unsigned char \*data, size\_t count)

*Sets the IR for a specific device in the chain.*
- void [SetIR](#) (unsigned int device, const unsigned char \*data, unsigned char \*data\_out, size\_t count)

*Sets the IR for a specific device in the chain and returns the IR capture value.*
- void [ScanDR](#) (unsigned int device, const unsigned char \*send\_data, unsigned char \*rcv\_data, size\_t count)

*Sets the DR for a specific device in the chain and optionally returns the previous DR contents.*
- void [ScanDRDeferred](#) (unsigned int device, const unsigned char \*send\_data, size\_t count)

*Sets the DR for a specific device in the chain and defers the operation if possible.*
- void [ScanDRSplitWrite](#) (unsigned int device, const unsigned char \*send\_data, unsigned char \*rcv\_data, size\_t count)

*Sets the DR for a specific device in the chain and optionally returns the previous DR contents.*
- void [ScanDRSplitRead](#) (unsigned int device, unsigned char \*rcv\_data, size\_t count)

*Sets the DR for a specific device in the chain and optionally returns the previous DR contents.*
- void [SwapOutDummy](#) (size\_t pos, [JtagDevice](#) \*realdev)

*Swap out a dummy device with a real device, once we've figured out by context/heuristics what it does.*
- virtual size\_t [GetShiftOpCount](#) ()

*Gets the number of shift operations performed on this interface.*

- virtual `size_t` [GetRecoverableErrorCount](#) ()  
*Gets the number of errors this interface has recovered from (USB retransmits, etc)*
- virtual `size_t` [GetDataBitCount](#) ()  
*Gets the number of data bits this interface has shifted.*
- virtual `size_t` [GetModeBitCount](#) ()  
*Gets the number of mode bits this interface has shifted.*
- virtual `size_t` [GetDummyClockCount](#) ()  
*Gets the number of dummy clocks this interface has sent.*
- virtual `double` [GetShiftTime](#) ()  
*Gets the number of dummy clocks this interface has sent.*

### Protected Member Functions

- virtual `void` [ShiftTMS](#) (bool tdi, const unsigned char \*send\_data, `size_t` count)=0  
*Shifts data into TMS to change TAP state.*
- `void` [CreateDummyDevices](#) ()  
*Creates dummy devices to fill out an incomplete chain.*
- `void` [PrintChainFaultMessage](#) ()  
*Prints an error when we fail to initialize the scan chain.*

### Protected Attributes

- `size_t` [m\\_devicecount](#)  
*Number of devices in the scan chain.*
- `size_t` [m\\_irtotal](#)  
*Total IR length of the chain.*
- `std::vector< unsigned int >` [m\\_idcodes](#)  
*Array of device ID codes.*
- `std::vector< JtagDevice * >` [m\\_devices](#)  
*Array of devices.*
- `size_t` [m\\_perfShiftOps](#)  
*Number of shift operations performed on this interface.*
- `size_t` [m\\_perfRecoverableErrs](#)  
*Number of link errors successfully recovered from.*
- `size_t` [m\\_perfDataBits](#)  
*Number of data bits shifted.*
- `size_t` [m\\_perfModeBits](#)  
*Number of mode bits shifted.*
- `size_t` [m\\_perfDummyClocks](#)  
*Number of dummy clocks shifted.*
- `double` [m\\_perfShiftTime](#)  
*Total time spent on shift operations.*

#### 7.45.1 Detailed Description

Abstract representation of a JTAG adapter.

A JTAG adapter provides access to a single scan chain containing zero or more [JtagDevice](#) objects.

The [JtagInterface](#) class provides three levels of abstraction for scan chain access.

### Low level (bit level)

This is the most basic way to access a JTAG adapter - clocking raw bits in and out.

In order to support a new "dumb" JTAG adapter without any higher level protocol offload, create a new derived class and implement each of the following functions:

- [GetName\(\)](#)
- [GetSerial\(\)](#)
- [GetUserID\(\)](#)
- [GetFrequency\(\)](#)
- [ShiftData\(\)](#)
- [ShiftTMS\(\)](#)
- [SendDummyClocks\(\)](#)

The low-level interface also includes support for pipelined / queued command execution. This can improve performance when using adapters connected to high-latency links such as USB.

The default implementations simply call the non-deferred versions and execute immediately. If an adapter supports queuing of commands, overriding these functions can result in higher performance.

- [IsSplitScanSupported\(\)](#)
- [SendDummyClocksDeferred\(\)](#)
- [ShiftDataWriteOnly\(\)](#)
- [ShiftDataReadOnly\(\)](#)
- [Commit\(\)](#)

### Mid level (state level)

By default, these functions are simple wrappers around [ShiftTMS\(\)](#) for changing between chain states.

If the adapter supports server-side management of chain state, these can be overridden to simply push a command to the adapter.

- [ResetTidle\(\)](#)
- [TestLogicReset\(\)](#)
- [EnterShiftIR\(\)](#)
- [LeaveExit1IR\(\)](#)
- [EnterShiftDR\(\)](#)
- [LeaveExit1DR\(\)](#)

### High level (register level)

These functions provide access to individual registers of TAPs, providing padding as necessary.

By default these functions are simple wrappers around [ShiftData\(\)](#) and the mid-level state functions.

If the adapter supports server-side padding insertion/removal, these can be overridden to reduce overhead.

The "deferred" versions of these functions may queue commands. To ensure that all previous queued commands have executed, call [Commit\(\)](#) or any function which returns readback data from a scan transaction.

- [SetIR\(\)](#)
- [SetIRDeferred\(\)](#)
- [ScanDR\(\)](#)
- [ScanDRDeferred\(\)](#)
- [ScanDRSplitRead\(\)](#)
- [ScanDRSplitWrite\(\)](#)

### Device management

These functions provide access to individual devices on the chain.

- [InitializeChain\(\)](#)
- [GetDeviceCount\(\)](#)
- [GetIDCode\(\)](#)
- [GetDevice\(\)](#)
- [SwapOutDummy\(\)](#)

### Performance profiling

These functions return stats on the amount of data shifted and the time spent waiting for the adapter.

These may be useful to compare different programming algorithms and optimizations to reduce unnecessary activity.

- [GetShiftOpCount\(\)](#)
- [GetRecoverableErrorCount\(\)](#)
- [GetDataBitCount\(\)](#)
- [GetModeBitCount\(\)](#)
- [GetDummyClockCount\(\)](#)
- [GetShiftTime\(\)](#)

### NOTES

Devices are numbered such that TDI goes to device N and TDO goes to device 0.

## 7.45.2 Constructor & Destructor Documentation

### 7.45.2.1 JtagInterface()

```
JtagInterface::JtagInterface ()
```

Default constructor.

Initializes the interface to the empty state.

You should generally call [InitializeChain\(\)](#) to detect devices before doing much of anything else.

### 7.45.2.2 ~JtagInterface()

```
JtagInterface::~JtagInterface () [virtual]
```

Interface destructor.

Destroys all [JtagDevice](#) objects in the scan chain

## 7.45.3 Member Function Documentation

### 7.45.3.1 Commit()

```
void JtagInterface::Commit () [virtual]
```

Commits the outstanding transactions to the adapter.

No-op unless the adapter supports queueing of multiple writes.

This function is automatically called when [SendDummyClocks\(\)](#) is called or any readback is performed. Most adapter classes will automatically call it when the transmit queue reaches a certain size.

This function can be called at any time to ensure all pending operations have executed.

#### Exceptions

|                               |                  |
|-------------------------------|------------------|
| <a href="#">JtagException</a> | in case of error |
|-------------------------------|------------------|

Reimplemented in [FTDIJtagInterface](#), [NetworkedJtagInterface](#), and [PipeJtagInterface](#).

### 7.45.3.2 CreateDummyDevices()

```
void JtagInterface::CreateDummyDevices () [protected]
```

Creates dummy devices to fill out an incomplete chain.

If this chain has exactly one device which is not supported or missing an IDCODE, create a dummy device to take up the space so we can correctly calculate IR padding.

If multiple unknown devices are present, it's impossible to recover since we don't know how long the IRs for each one are, and thus can't figure out boundaries.

### 7.45.3.3 EnterShiftDR()

```
void JtagInterface::EnterShiftDR () [virtual]
```

Enters Shift-DR state from Run-Test-Idle state.

#### Exceptions

|                               |                                     |
|-------------------------------|-------------------------------------|
| <a href="#">JtagException</a> | if <a href="#">ShiftTMS()</a> fails |
|-------------------------------|-------------------------------------|

Reimplemented in [NetworkedJtagInterface](#), and [PipeJtagInterface](#).

### 7.45.3.4 EnterShiftIR()

```
void JtagInterface::EnterShiftIR () [virtual]
```

Enters Shift-IR state from Run-Test-Idle state.

#### Exceptions

|                               |                                     |
|-------------------------------|-------------------------------------|
| <a href="#">JtagException</a> | if <a href="#">ShiftTMS()</a> fails |
|-------------------------------|-------------------------------------|

Reimplemented in [NetworkedJtagInterface](#), and [PipeJtagInterface](#).

### 7.45.3.5 GetDataBitCount()

```
size_t JtagInterface::GetDataBitCount () [virtual]
```

Gets the number of data bits this interface has shifted.

#### Exceptions

|                               |            |
|-------------------------------|------------|
| <a href="#">JtagException</a> | on failure |
|-------------------------------|------------|

**Returns**

Number of data bits shifted

Reimplemented in [NetworkedJtagInterface](#), and [PipeJtagInterface](#).

**7.45.3.6 GetDevice()**

```
JtagDevice * JtagInterface::GetDevice (
 unsigned int device)
```

Gets the Nth device in the chain.

**Exceptions**

|                               |                              |
|-------------------------------|------------------------------|
| <a href="#">JtagException</a> | if the index is out of range |
|-------------------------------|------------------------------|

**Parameters**

|               |              |
|---------------|--------------|
| <i>device</i> | Device index |
|---------------|--------------|

**Returns**

The device object

**7.45.3.7 GetDeviceCount()**

```
size_t JtagInterface::GetDeviceCount ()
```

Returns the number of devices in the scan chain.

**Returns**

Device count

**7.45.3.8 GetDummyClockCount()**

```
size_t JtagInterface::GetDummyClockCount () [virtual]
```

Gets the number of dummy clocks this interface has sent.



**Exceptions**

|                               |            |
|-------------------------------|------------|
| <a href="#">JtagException</a> | on failure |
|-------------------------------|------------|

**Returns**

Number of dummy clocks sent

Reimplemented in [NetworkedJtagInterface](#), and [PipeJtagInterface](#).

**7.45.3.9 GetFrequency()**

```
virtual int JtagInterface::GetFrequency () [pure virtual]
```

Gets the clock frequency, in Hz, of the JTAG interface.

**Returns**

The clock frequency

Implemented in [FTDIJtagInterface](#), [DigilentJtagInterface](#), [NetworkedJtagInterface](#), and [PipeJtagInterface](#).

**7.45.3.10 GetIDCode()**

```
unsigned int JtagInterface::GetIDCode (
 unsigned int device)
```

Returns the ID for the supplied device (zero-based indexing)

**Exceptions**

|                               |                              |
|-------------------------------|------------------------------|
| <a href="#">JtagException</a> | if the index is out of range |
|-------------------------------|------------------------------|

**Parameters**

|               |              |
|---------------|--------------|
| <i>device</i> | Device index |
|---------------|--------------|

**Returns**

The 32-bit JTAG ID code

#### 7.45.3.11 GetModeBitCount()

```
size_t JtagInterface::GetModeBitCount () [virtual]
```

Gets the number of mode bits this interface has shifted.

##### Exceptions

|                               |            |
|-------------------------------|------------|
| <a href="#">JtagException</a> | on failure |
|-------------------------------|------------|

##### Returns

Number of mode bits shifted

Reimplemented in [NetworkedJtagInterface](#), and [PipeJtagInterface](#).

#### 7.45.3.12 GetName()

```
virtual std::string JtagInterface::GetName () [pure virtual]
```

Gets the manufacturer-assigned name for this programming adapter.

This is usually the model number but is sometimes something more generic like "Digilent Adept USB Device".

##### Returns

The device name

Implemented in [FTDIJtagInterface](#), [DigilentJtagInterface](#), [NetworkedJtagInterface](#), and [PipeJtagInterface](#).

#### 7.45.3.13 GetRecoverableErrorCount()

```
size_t JtagInterface::GetRecoverableErrorCount () [virtual]
```

Gets the number of errors this interface has recovered from (USB retransmits, etc)

##### Exceptions

|                               |            |
|-------------------------------|------------|
| <a href="#">JtagException</a> | on failure |
|-------------------------------|------------|

##### Returns

Number of recoverable errors

Reimplemented in [NetworkedJtagInterface](#), and [PipeJtagInterface](#).

#### 7.45.3.14 GetSerial()

```
virtual std::string JtagInterface::GetSerial () [pure virtual]
```

Gets the manufacturer-assigned serial number for this programming adapter, if any.

Derived classes may choose to return the user ID, an empty string, or another default value if no serial number has been assigned.

##### Returns

The serial number

Implemented in [FTDIJtagInterface](#), [DigilentJtagInterface](#), [NetworkedJtagInterface](#), and [PipeJtagInterface](#).

#### 7.45.3.15 GetShiftOpCount()

```
size_t JtagInterface::GetShiftOpCount () [virtual]
```

Gets the number of shift operations performed on this interface.

##### Exceptions

|                               |            |
|-------------------------------|------------|
| <a href="#">JtagException</a> | on failure |
|-------------------------------|------------|

##### Returns

Number of shift operations

Reimplemented in [NetworkedJtagInterface](#), and [PipeJtagInterface](#).

#### 7.45.3.16 GetShiftTime()

```
double JtagInterface::GetShiftTime () [virtual]
```

Gets the number of dummy clocks this interface has sent.

##### Exceptions

|                               |            |
|-------------------------------|------------|
| <a href="#">JtagException</a> | on failure |
|-------------------------------|------------|

**Returns**

Number of dummy clocks sent

**7.45.3.17 GetUserID()**

```
virtual std::string JtagInterface::GetUserID () [pure virtual]
```

Gets the user-assigned name for this JTAG adapter, if any.

Derived classes may choose to return the serial number, an empty string, or another default value if no name has been assigned.

**Returns**

The name for this adapter.

Implemented in [FTDIJtagInterface](#), [DigilentJtagInterface](#), [NetworkedJtagInterface](#), and [PipeJtagInterface](#).

**7.45.3.18 InitializeChain()**

```
void JtagInterface::InitializeChain (
 bool quiet = false)
```

Initializes the scan chain and gets the number of devices present.

Assumes less than 1024 bits of total IR length.

**Exceptions**

|                               |                                      |
|-------------------------------|--------------------------------------|
| <a href="#">JtagException</a> | if any of the scan operations fails. |
|-------------------------------|--------------------------------------|

**7.45.3.19 IsSplitScanSupported()**

```
bool JtagInterface::IsSplitScanSupported () [virtual]
```

Indicates if split (pipelined) DR scanning is supported.

Split scanning allows the write halves of several scan operations to take place in one driver-level write call, followed by the read halves in order, to reduce the impact of driver/bus latency on throughput.

If split scanning is not supported, [ScanDRSplitWrite\(\)](#) will behave identically to [ScanDR\(\)](#) and [ScanDRSplitRead\(\)](#) will be a no-op. This ensures that using the split write commands will work correctly regardless of whether the adapter supports split scanning in hardware.

Reimplemented in [FTDIJtagInterface](#), [NetworkedJtagInterface](#), and [PipeJtagInterface](#).

### 7.45.3.20 LeaveExit1DR()

```
void JtagInterface::LeaveExit1DR () [virtual]
```

Leaves Exit1-DR state and returns to Run-Test-Idle.

#### Exceptions

|                               |                                     |
|-------------------------------|-------------------------------------|
| <a href="#">JtagException</a> | if <a href="#">ShiftTMS()</a> fails |
|-------------------------------|-------------------------------------|

Reimplemented in [NetworkedJtagInterface](#), and [PipeJtagInterface](#).

### 7.45.3.21 LeaveExit1IR()

```
void JtagInterface::LeaveExit1IR () [virtual]
```

Leaves Exit1-IR state and returns to Run-Test-Idle.

#### Exceptions

|                               |                                     |
|-------------------------------|-------------------------------------|
| <a href="#">JtagException</a> | if <a href="#">ShiftTMS()</a> fails |
|-------------------------------|-------------------------------------|

Reimplemented in [NetworkedJtagInterface](#), and [PipeJtagInterface](#).

### 7.45.3.22 ResetToIdle()

```
void JtagInterface::ResetToIdle () [virtual]
```

Resets the TAP and enters Run-Test-Idle state.

#### Exceptions

|                               |                                     |
|-------------------------------|-------------------------------------|
| <a href="#">JtagException</a> | if <a href="#">ShiftTMS()</a> fails |
|-------------------------------|-------------------------------------|

Reimplemented in [NetworkedJtagInterface](#), and [PipeJtagInterface](#).

### 7.45.3.23 ScanDR()

```
void JtagInterface::ScanDR (
 unsigned int device,
 const unsigned char * send_data,
```

```

 unsigned char * rcv_data,
 size_t count)

```

Sets the DR for a specific device in the chain and optionally returns the previous DR contents.

Starts and ends in Run-Test-Idle state.

#### Exceptions

|                               |                               |
|-------------------------------|-------------------------------|
| <a href="#">JtagException</a> | if any shift operation fails. |
|-------------------------------|-------------------------------|

#### Parameters

|                  |                                                                                                                        |
|------------------|------------------------------------------------------------------------------------------------------------------------|
| <i>device</i>    | Zero-based index of the target device. All other devices are assumed to be in BYPASS mode and their DR is set to zero. |
| <i>send_data</i> | The data value to scan (see <a href="#">ShiftData()</a> for bit/byte ordering)                                         |
| <i>rcv_data</i>  | Output data to scan, or NULL if no output is desired (faster)                                                          |
| <i>count</i>     | Number of bits to scan                                                                                                 |

#### 7.45.3.24 ScanDRDeferred()

```

void JtagInterface::ScanDRDeferred (
 unsigned int device,
 const unsigned char * send_data,
 size_t count)

```

Sets the DR for a specific device in the chain and defers the operation if possible.

The scan operation may not actually execute until [Commit\(\)](#) is called. When the operation executes is dependent on whether the interface supports deferred writes, how full the interface's buffer is, and when the next operation forcing a commit (call to [Commit\(\)](#) or a read operation) takes place.

Starts and ends in Run-Test-Idle state.

#### Exceptions

|                               |                               |
|-------------------------------|-------------------------------|
| <a href="#">JtagException</a> | if any shift operation fails. |
|-------------------------------|-------------------------------|

#### Parameters

|                  |                                                                                                                        |
|------------------|------------------------------------------------------------------------------------------------------------------------|
| <i>device</i>    | Zero-based index of the target device. All other devices are assumed to be in BYPASS mode and their DR is set to zero. |
| <i>send_data</i> | The data value to scan (see <a href="#">ShiftData()</a> for bit/byte ordering)                                         |
| <i>count</i>     | Number of bits to scan                                                                                                 |

## 7.45.3.25 ScanDRSplitRead()

```
void JtagInterface::ScanDRSplitRead (
 unsigned int device,
 unsigned char * rcv_data,
 size_t count)
```

Sets the DR for a specific device in the chain and optionally returns the previous DR contents.

Starts and ends in Run-Test-Idle state.

If split (pipelined) scanning is supported, this call performs the read half of the scan only.

## Exceptions

|                               |                               |
|-------------------------------|-------------------------------|
| <a href="#">JtagException</a> | if any shift operation fails. |
|-------------------------------|-------------------------------|

## Parameters

|                 |                                                                                                                        |
|-----------------|------------------------------------------------------------------------------------------------------------------------|
| <i>device</i>   | Zero-based index of the target device. All other devices are assumed to be in BYPASS mode and their DR is set to zero. |
| <i>rcv_data</i> | Output data to scan, or NULL if no output is desired (faster)                                                          |
| <i>count</i>    | Number of bits to scan                                                                                                 |

## 7.45.3.26 ScanDRSplitWrite()

```
void JtagInterface::ScanDRSplitWrite (
 unsigned int device,
 const unsigned char * send_data,
 unsigned char * rcv_data,
 size_t count)
```

Sets the DR for a specific device in the chain and optionally returns the previous DR contents.

Starts and ends in Run-Test-Idle state.

If split (pipelined) scanning is supported, this call performs the write half of the scan only; the read is performed by [ScanDRSplitRead\(\)](#). Several writes may occur in a row, and must be followed by an equivalent number of reads with matching length values.

## Exceptions

|                               |                               |
|-------------------------------|-------------------------------|
| <a href="#">JtagException</a> | if any shift operation fails. |
|-------------------------------|-------------------------------|

## Parameters

|                  |                                                                                                                        |
|------------------|------------------------------------------------------------------------------------------------------------------------|
| <i>device</i>    | Zero-based index of the target device. All other devices are assumed to be in BYPASS mode and their DR is set to zero. |
| <i>send_data</i> | The data value to scan (see <a href="#">ShiftData()</a> for bit/byte ordering)                                         |

## Parameters

|                 |                                                               |
|-----------------|---------------------------------------------------------------|
| <i>rcv_data</i> | Output data to scan, or NULL if no output is desired (faster) |
| <i>count</i>    | Number of bits to scan                                        |

## 7.45.3.27 SendDummyClocks()

```
virtual void JtagInterface::SendDummyClocks (
 size_t n) [pure virtual]
```

Sends the requested number of dummy clocks with TMS=0 and flushes the command to the interface.

Since dummy clocks are often used as a delay element for programming algorithms etc, this function flushes the write buffer to ensure immediate execution.

## Exceptions

|                               |                                           |
|-------------------------------|-------------------------------------------|
| <a href="#">JtagException</a> | may be thrown if the scan operation fails |
|-------------------------------|-------------------------------------------|

## Parameters

|          |                                |
|----------|--------------------------------|
| <i>n</i> | Number of dummy clocks to send |
|----------|--------------------------------|

Implemented in [FTDIJtagInterface](#), [DigilentJtagInterface](#), [NetworkedJtagInterface](#), and [PipeJtagInterface](#).

## 7.45.3.28 SendDummyClocksDeferred()

```
void JtagInterface::SendDummyClocksDeferred (
 size_t n) [virtual]
```

Sends the requested number of dummy clocks with TMS=0 and does not flush the write pipeline.

## Exceptions

|                               |                                           |
|-------------------------------|-------------------------------------------|
| <a href="#">JtagException</a> | may be thrown if the scan operation fails |
|-------------------------------|-------------------------------------------|

## Parameters

|          |                                |
|----------|--------------------------------|
| <i>n</i> | Number of dummy clocks to send |
|----------|--------------------------------|

Reimplemented in [FTDIJtagInterface](#), [NetworkedJtagInterface](#), and [PipeJtagInterface](#).



## 7.45.3.29 SetIR() [1/2]

```
void JtagInterface::SetIR (
 unsigned int device,
 const unsigned char * data,
 size_t count)
```

Sets the IR for a specific device in the chain.

Starts and ends in Run-Test-Idle state.

## Exceptions

|                                      |                               |
|--------------------------------------|-------------------------------|
| <a href="#"><i>JtagException</i></a> | if any shift operation fails. |
|--------------------------------------|-------------------------------|

## Parameters

|               |                                                                                  |
|---------------|----------------------------------------------------------------------------------|
| <i>device</i> | Zero-based index of the target device. All other devices are set to BYPASS mode. |
| <i>data</i>   | The IR value to scan (see <a href="#">ShiftData()</a> for bit/byte ordering)     |
| <i>count</i>  | Instruction register length, in bits                                             |

## 7.45.3.30 SetIR() [2/2]

```
void JtagInterface::SetIR (
 unsigned int device,
 const unsigned char * data,
 unsigned char * data_out,
 size_t count)
```

Sets the IR for a specific device in the chain and returns the IR capture value.

Starts and ends in Run-Test-Idle state.

## Exceptions

|                                      |                               |
|--------------------------------------|-------------------------------|
| <a href="#"><i>JtagException</i></a> | if any shift operation fails. |
|--------------------------------------|-------------------------------|

## Parameters

|                 |                                                                                  |
|-----------------|----------------------------------------------------------------------------------|
| <i>device</i>   | Zero-based index of the target device. All other devices are set to BYPASS mode. |
| <i>data</i>     | The IR value to scan (see <a href="#">ShiftData()</a> for bit/byte ordering)     |
| <i>data_out</i> | IR capture value                                                                 |
| <i>count</i>    | Instruction register length, in bits                                             |

### 7.45.3.31 SetIRDeferred()

```
void JtagInterface::SetIRDeferred (
 unsigned int device,
 const unsigned char * data,
 size_t count)
```

Sets the IR for a specific device in the chain.

Starts and ends in Run-Test-Idle state.

#### Exceptions

|                               |                               |
|-------------------------------|-------------------------------|
| <a href="#">JtagException</a> | if any shift operation fails. |
|-------------------------------|-------------------------------|

#### Parameters

|               |                                                                                  |
|---------------|----------------------------------------------------------------------------------|
| <i>device</i> | Zero-based index of the target device. All other devices are set to BYPASS mode. |
| <i>data</i>   | The IR value to scan (see <a href="#">ShiftData()</a> for bit/byte ordering)     |
| <i>count</i>  | Instruction register length, in bits                                             |

### 7.45.3.32 ShiftData()

```
virtual void JtagInterface::ShiftData (
 bool last_tms,
 const unsigned char * send_data,
 unsigned char * rcv_data,
 size_t count) [pure virtual]
```

Shifts data through TDI to TDO.

The LSB of `send_data[0]` is sent first; the MSB of `send_data[0]` is followed by the LSB of `send_data[1]`.

#### Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <i>last_tms</i>  | Different TMS value to use for last bit |
| <i>send_data</i> | Data to shift into TDI                  |
| <i>rcv_data</i>  | Data to shift out of TDO (may be NULL)  |
| <i>count</i>     | Number of bits to shift                 |

Implemented in [FTDIJtagInterface](#), [DigilentJtagInterface](#), [NetworkedJtagInterface](#), and [PipeJtagInterface](#).

### 7.45.3.33 ShiftDataReadOnly()

```
bool JtagInterface::ShiftDataReadOnly (
 unsigned char * rcv_data,
 size_t count) [virtual]
```

Reads data from a [ShiftDataWriteOnly\(\)](#) call.

For more information on split (pipelined) scan operations see [ShiftDataWriteOnly\(\)](#).

#### Returns

True if the read was executed, false if a no-op

#### Parameters

|                 |                                        |
|-----------------|----------------------------------------|
| <i>rcv_data</i> | Data to shift out of TDO (may be NULL) |
| <i>count</i>    | Number of bits to shift                |

Reimplemented in [FTDIJtagInterface](#), [NetworkedJtagInterface](#), and [PipeJtagInterface](#).

#### 7.45.3.34 ShiftDataWriteOnly()

```
bool JtagInterface::ShiftDataWriteOnly (
 bool last_tms,
 const unsigned char * send_data,
 unsigned char * rcv_data,
 size_t count) [virtual]
```

Shifts data through TDI to TDO.

The LSB of `send_data[0]` is sent first; the MSB of `send_data[0]` is followed by the LSB of `send_data[1]`.

If split (pipelined) scanning is supported by the adapter, this function performs the write half of the shift operation only; the read is buffered in the JTAG adapter and no readback is performed until [ShiftDataReadOnly\(\)](#) is called. This allows several shift operations to occur in sequence without incurring a USB turnaround delay or other driver latency overhead for each shift operation.

If split scanning is not supported this call is equivalent to [ShiftData\(\)](#) and [ShiftDataReadOnly\(\)](#) is a no-op.

This function MUST be followed by either another [ShiftDataWriteOnly\(\)](#) call, a [ShiftTMS\(\)](#) call, or a [ShiftDataReadOnly\(\)](#) call. There must be exactly one [ShiftDataReadOnly\(\)](#) call for each [ShiftDataWriteOnly\(\)](#) call and they must be in order with the same `rcv_data` and count values. The result of doing otherwise is undefined.

#### Returns

True if the read was deferred, false if not

#### Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <i>last_tms</i>  | Different TMS value to use for last bit |
| <i>send_data</i> | Data to shift into TDI                  |
| <i>rcv_data</i>  | Data to shift out of TDO (may be NULL)  |
| <i>count</i>     | Number of bits to shift                 |

Reimplemented in [FTDIJtagInterface](#), [NetworkedJtagInterface](#), and [PipeJtagInterface](#).

#### 7.45.3.35 ShiftTMS()

```
virtual void JtagInterface::ShiftTMS (
 bool tdi,
 const unsigned char * send_data,
 size_t count) [protected], [pure virtual]
```

Shifts data into TMS to change TAP state.

This is no longer a public API operation. It can only be accessed via the state-level interface.

Implementations of this class may choose to implement EITHER this function (and use the default JtagInterface-provided state-level functions) OR override this function with a private no-op stub and override the state-level functions instead.

#### Exceptions

|                               |                                           |
|-------------------------------|-------------------------------------------|
| <a href="#">JtagException</a> | may be thrown if the scan operation fails |
|-------------------------------|-------------------------------------------|

#### Parameters

|                  |                                                                                       |
|------------------|---------------------------------------------------------------------------------------|
| <i>tdi</i>       | Constant tdi value (normally 0)                                                       |
| <i>send_data</i> | Data to shift into TMS. Bit ordering is the same as for <a href="#">ShiftData()</a> . |
| <i>count</i>     | Number of bits to shift                                                               |

Implemented in [FTDIJtagInterface](#), and [DigilentJtagInterface](#).

#### 7.45.3.36 SwapOutDummy()

```
void JtagInterface::SwapOutDummy (
 size_t pos,
 JtagDevice * realdev)
```

Swap out a dummy device with a real device, once we've figured out by context/heuristics what it does.

Often when the chain is first being walked, unknown devices cannot be identified - but later on, once the remainder of the chain has been discovered, the unknown devices can be identified contextually (for example a no-idcode debug TAP followed by an IDCODE-capable boundary scan TAP).

This function allows a dummy device in the chain to be replaced with a non-dummy device.

#### 7.45.3.37 TestLogicReset()

```
void JtagInterface::TestLogicReset () [virtual]
```

Enters Test-Logic-Reset state by shifting six ones into TMS.

## Exceptions

|                               |                                  |
|-------------------------------|----------------------------------|
| <a href="#">JtagException</a> | if <code>ShiftTMS()</code> fails |
|-------------------------------|----------------------------------|

Reimplemented in [NetworkedJtagInterface](#), and [PipeJtagInterface](#).

The documentation for this class was generated from the following files:

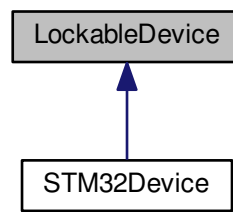
- [JtagInterface.h](#)
- [JtagInterface.cpp](#)

## 7.46 LockableDevice Class Reference

Generic base class for all devices which have some kind of read/write protection.

```
#include <LockableDevice.h>
```

Inheritance diagram for LockableDevice:



### Public Types

- enum [AccessLevel](#) { **ACCESS\_EXECUTE** = 1, **ACCESS\_WRITE** = 2, **ACCESS\_READ** = 4 }  
Levels of access being requested (may be ORed together)

### Public Member Functions

- virtual void [ProbeLocksNondestructive](#) ()=0  
Queries lock status in a non-destructive fashion (contents of the chip are untouched)
- virtual void [ProbeLocksDestructive](#) ()=0  
Queries lock status in a more invasive fashion. Gives more accurate data but may involve write transactions to memory.
- virtual [UncertainBoolean CheckMemoryAccess](#) (uint32\_t ptr, unsigned int access)=0  
Checks if a given physical address has a given protection applied.
- virtual [UncertainBoolean IsDeviceReadLocked](#) ()=0  
Checks if the device is globally read protected or not.
- virtual void [PrintLockProbeDetails](#) ()=0  
Prints detailed information regarding the state of the read lock.
- virtual void [SetReadLock](#) ()=0  
Sets a global read-protection lock on the entire device.
- virtual void [ClearReadLock](#) ()=0  
Clears the global read-protection lock, if set in a non-permanent fashion.

### 7.46.1 Detailed Description

Generic base class for all devices which have some kind of read/write protection.

Note that sometimes due to protections, it's not possible to get definite answers to all queries.

### 7.46.2 Member Function Documentation

#### 7.46.2.1 ClearReadLock()

```
virtual void LockableDevice::ClearReadLock () [pure virtual]
```

Clears the global read-protection lock, if set in a non-permanent fashion.

In most parts, this will trigger a bulk flash erase.

Implemented in [STM32Device](#).

#### 7.46.2.2 SetReadLock()

```
virtual void LockableDevice::SetReadLock () [pure virtual]
```

Sets a global read-protection lock on the entire device.

This function only performs reversible locks that can be cleared with a bulk erase. Thus, it should not be able to brick the chip entirely.

Implemented in [STM32Device](#).

The documentation for this class was generated from the following files:

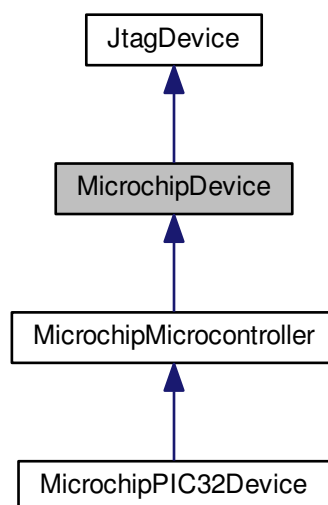
- [LockableDevice.h](#)
- [LockableDevice.cpp](#)

## 7.47 MicrochipDevice Class Reference

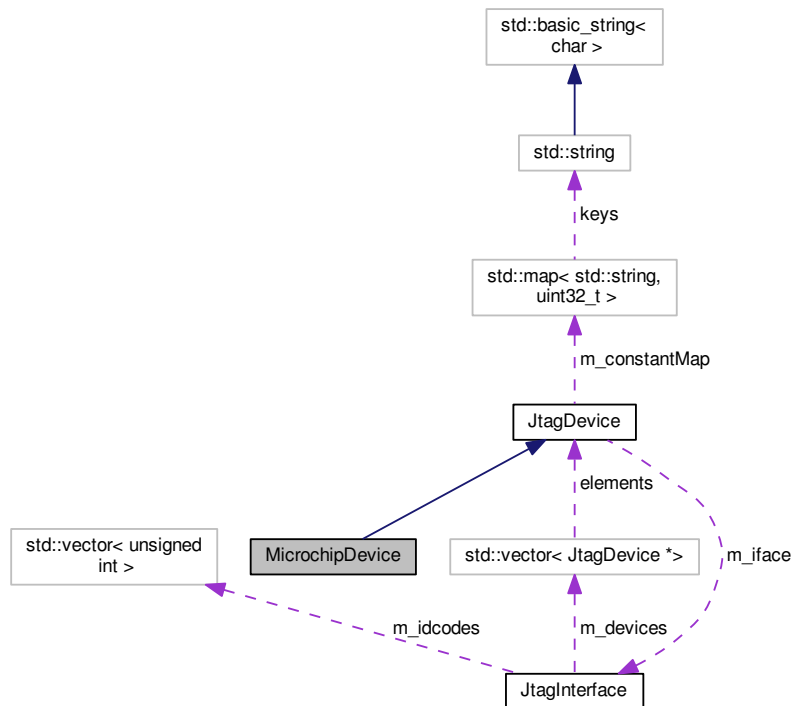
Abstract base class for all Microchip devices (typically MCUs)

```
#include <MicrochipDevice.h>
```

Inheritance diagram for MicrochipDevice:



Collaboration diagram for MicrochipDevice:



## Public Member Functions

- `MicrochipDevice` (unsigned int idcode, `JtagInterface` \*iface, size\_t pos, size\_t irlength)  
*Initializes this device.*
- virtual `~MicrochipDevice` ()  
*Default virtual destructor.*

## Static Public Member Functions

- static `JtagDevice` \* `CreateDevice` (unsigned int idcode, `JtagInterface` \*iface, size\_t pos)  
*Creates a `MicrochipDevice` given an ID code.*

## Additional Inherited Members

### 7.47.1 Detailed Description

Abstract base class for all Microchip devices (typically MCUs)

### 7.47.2 Constructor & Destructor Documentation



## 7.47.2.1 MicrochipDevice()

```
MicrochipDevice::MicrochipDevice (
 unsigned int idcode,
 JtagInterface * iface,
 size_t pos,
 size_t irlength)
```

Initializes this device.

## Parameters

|                 |                                                       |
|-----------------|-------------------------------------------------------|
| <i>idcode</i>   | The ID code of this device                            |
| <i>iface</i>    | The JTAG adapter this device was discovered on        |
| <i>pos</i>      | Position in the chain that this device was discovered |
| <i>irlength</i> | Length of the JTAG instruction register               |

## 7.47.3 Member Function Documentation

## 7.47.3.1 CreateDevice()

```
JtagDevice * MicrochipDevice::CreateDevice (
 unsigned int idcode,
 JtagInterface * iface,
 size_t pos) [static]
```

Creates a [MicrochipDevice](#) given an ID code.

## Exceptions

|                               |                                                                                       |
|-------------------------------|---------------------------------------------------------------------------------------|
| <a href="#">JtagException</a> | if the ID code supplied is not a valid Microchip device, or not a known family number |
|-------------------------------|---------------------------------------------------------------------------------------|

## Parameters

|               |                                                       |
|---------------|-------------------------------------------------------|
| <i>idcode</i> | The ID code of this device                            |
| <i>iface</i>  | The JTAG adapter this device was discovered on        |
| <i>pos</i>    | Position in the chain that this device was discovered |

## Returns

A valid [JtagDevice](#) object, or NULL if the vendor ID was not recognized.

The documentation for this class was generated from the following files:

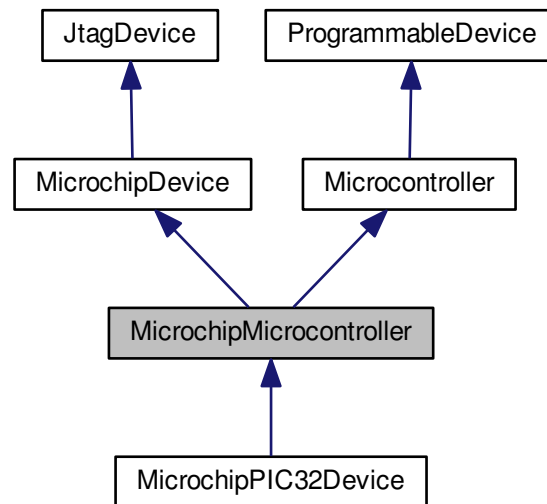
- [MicrochipDevice.h](#)
- [MicrochipDevice.cpp](#)

## 7.48 MicrochipMicrocontroller Class Reference

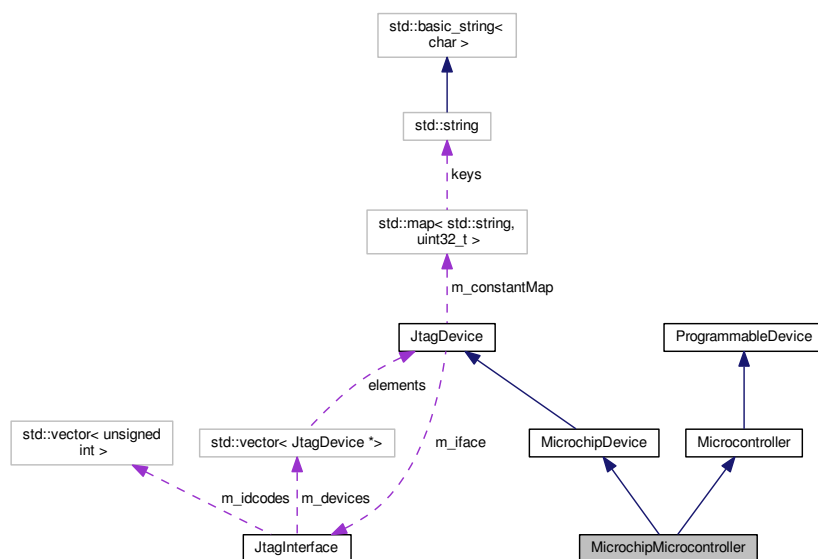
Generic base class for all Microchip MCUs.

```
#include <MicrochipMicrocontroller.h>
```

Inheritance diagram for MicrochipMicrocontroller:



Collaboration diagram for MicrochipMicrocontroller:



## Public Member Functions

- **MicrochipMicrocontroller** (unsigned int idcode, [JtagInterface](#) \*iface, size\_t pos, size\_t irlength)

## Additional Inherited Members

### 7.48.1 Detailed Description

Generic base class for all Microchip MCUs.

The documentation for this class was generated from the following files:

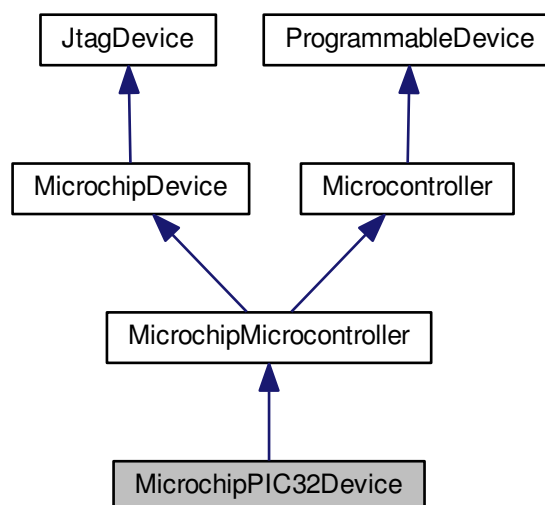
- [MicrochipMicrocontroller.h](#)
- [MicrochipMicrocontroller.cpp](#)

## 7.49 MicrochipPIC32Device Class Reference

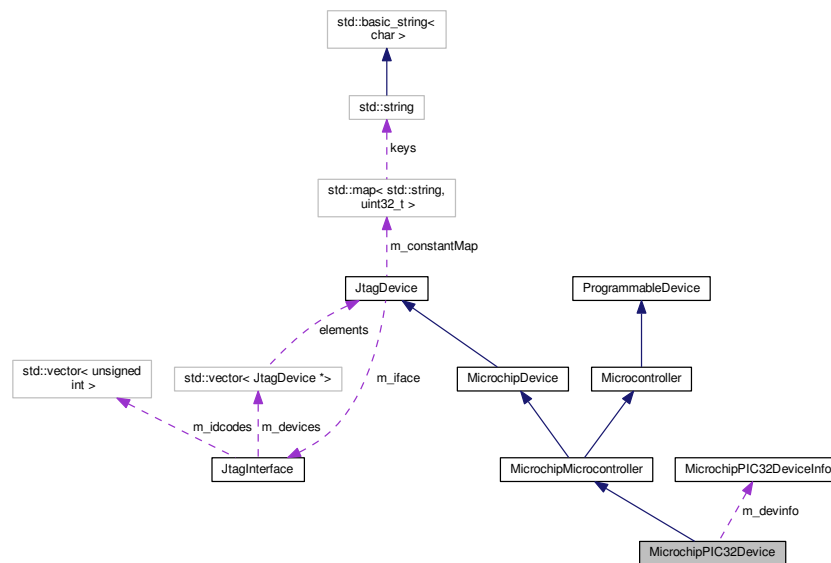
A Microchip PIC32 microcontroller (MX, MZ, MM, etc)

```
#include <MicrochipPIC32Device.h>
```

Inheritance diagram for MicrochipPIC32Device:



Collaboration diagram for MicrochipPIC32Device:



## Public Types

- enum [families](#) {
  - FAMILY\_MX12**, **FAMILY\_MX34**, **FAMILY\_MX567**, **FAMILY\_MM**, **FAMILY\_MZ** }
  - Device families.*
- enum [cpus](#) { **CPU\_M4K**, **CPU\_MAPTIV** }
  - CPU types.*
- enum [deviceids](#) {
  - PIC32MX110F016B** = 0x4a07, **PIC32MX110F016C** = 0x4a09, **PIC32MX110F016D** = 0x4a0b, **PIC32MX120F032B** = 0x4a06,
  - PIC32MX120F032C** = 0x4a08, **PIC32MX120F032D** = 0x4a0a, **PIC32MX130F064B** = 0x4d07, **PIC32MX130F064C** = 0x4d09,
  - PIC32MX130F064D** = 0x4d0b, **PIC32MX150F128B** = 0x4d06, **PIC32MX150F128C** = 0x4d08, **PIC32MX150F128D** = 0x4d0a,
  - PIC32MX210F016B** = 0x4a01, **PIC32MX210F016C** = 0x4a03, **PIC32MX210F016D** = 0x4a05, **PIC32MX220F032B** = 0x4a00,
  - PIC32MX220F032C** = 0x4a02, **PIC32MX220F032D** = 0x4a04, **PIC32MX230F064B** = 0x4d01, **PIC32MX230F064C** = 0x4d03,
  - PIC32MX230F064D** = 0x4d05, **PIC32MX250F128B** = 0x4d00, **PIC32MX250F128C** = 0x4d02, **PIC32MX250F128D** = 0x4d04,
  - PIC32MX330F064H** = 0x5600, **PIC32MX330F064L** = 0x5601, **PIC32MX340F512H** = 0x0916, **PIC32MX350F128H** = 0x570c,
  - PIC32MX350F256H** = 0x570d, **PIC32MX350F256L** = 0x5705, **PIC32MX430F064H** = 0x5602, **PIC32MX430F064L** = 0x5603,
  - PIC32MX450F128H** = 0x570e, **PIC32MX450F128L** = 0x570f, **PIC32MX450F256H** = 0x5706, **PIC32MX450F256L** = 0x5707,
  - PIC32MX534F064H** = 0x440c, **PIC32MX564F064H** = 0x4401, **PIC32MX564F064L** = 0x440d, **PIC32MX564F128H** = 0x4403,
  - PIC32MX564F128L** = 0x440f, **PIC32MX664F064H** = 0x4405, **PIC32MX664F064L** = 0x4411, **PIC32MX664F128H** = 0x4407,

**PIC32MX664F128L** = 0x4413, **PIC32MX695F512L** = 0x4341, **PIC32MX764F128H** = 0x440b, **PIC32MX764F128L** = 0x4417,  
**PIC32MX795F512L** = 0x4307, **PIC32MM0016GPL020** = 0x6b04, **PIC32MM0032GPL020** = 0x6b0c, **PIC32MM0064GPL020** = 0x6b14,  
**PIC32MM0016GPL028** = 0x6b02, **PIC32MM0032GPL028** = 0x6b0a, **PIC32MM0064GPL028** = 0x6b12, **PIC32MM0016GPL036** = 0x6b06,  
**PIC32MM0032GPL036** = 0x6b0b, **PIC32MM0064GPL036** = 0x6b16, **PIC32MM0064GPM028** = 0x7708, **PIC32MM0128GPM028** = 0x7710,  
**PIC32MM0256GPM028** = 0x7718, **PIC32MM0064GPM036** = 0x770a, **PIC32MM0128GPM036** = 0x7712, **PIC32MM0256GPM036** = 0x771a,  
**PIC32MM0064GPM048** = 0x772c, **PIC32MM0128GPM048** = 0x7734, **PIC32MM0256GPM048** = 0x773c, **PIC32MM0064GPM064** = 0x770e,  
**PIC32MM0128GPM064** = 0x7716, **PIC32MM0256GPM064** = 0x771e }

*JTAG device IDs (from BSDL files and/or flash programming spec)*

- enum `instructions` {
  - `INST_BYPASS` = 0x1F, `INST_IDCODE` = 0x01, `INST_IMPCODE` = 0x03, `INST_MTAP_SW_MCHP` = 0x04, `INST_MTAP_SW_EJTAG` = 0x05, `INST_MTAP_COMMAND` = 0x07, `INST_ADDRESS` = 0x08, `INST_DATA` = 0x09,
  - `INST_CONTROL` = 0x0A, `INST_ALL` = 0x0B, `INST_DEBUGBOOT` = 0x0C, `INST_NORMALBOOT` = 0x0D, `INST_FASTDATA` = 0x0E, `INST_PCSAMPLE` = 0x14 }

*5-bit-wide JTAG instructions (from BSDL file and datasheet)*

- enum `mtap_instructions` {
  - `MCHP_STATUS` = 0x00, `MCHP_ASSERT_RST` = 0xD1, `MCHP_DE_ASSERT_RST` = 0xD0, `MCHP_ERASE` = 0xFC,
  - `MCHP_FLASH_ENABLE` = 0xFE, `MCHP_FLASH_DISABLE` = 0xFD, `MCHP_READ_CONFIG` = 0xFF }

*8-bit instructions for Microchip virtual TAP (write to INST\_MTAP\_COMMAND data register)*

## Public Member Functions

- **MicrochipPIC32Device** (unsigned int devid, unsigned int stepping, unsigned int idcode, [JtagInterface](#) \*iface, size\_t pos)
- virtual `~MicrochipPIC32Device` ()
  - Destructor.*
- virtual void `PostInitProbes` (bool quiet)
  - Does a post-initialization probe of the device to read debug ROMs etc.*
- virtual std::string `GetDescription` ()
  - Gets a human-readable description of this device.*
- virtual bool `IsProgrammed` ()
  - Determines if this device is programmed or blank.*
- virtual void `Erase` ()
  - Erases the device configuration and restores the device to a blank state.*
- virtual void `Program` ([FirmwareImage](#) \*image)
  - Loads a new firmware image onto the device.*
- void `SetIR` (unsigned char irval)

## Static Public Member Functions

- static [JtagDevice](#) \* `CreateDevice` (unsigned int devid, unsigned int stepping, unsigned int idcode, [JtagInterface](#) \*iface, size\_t pos)

## Public Attributes

- enum `MicrochipPIC32Device::families` `__attribute__`

## Protected Member Functions

- void **EnterMtapMode** ()
- uint8\_t **SendMchpCommand** (uint8\_t cmd)  
*Sends a MTAP command. Requires TAP to be in MCHP mode, not EJTAG mode.*
- void **EnterEjtagMode** ()
- void **EnterSerialExecMode** ()
- void **SerialExecuteInstruction** (uint32\_t insn, bool first=false)  
*Executes a single MIPS32 instruction in serial-exec mode.*
- void **SerialExecuteMemoryWrite** (uint32\_t addr, uint32\_t data)
- uint32\_t **SerialExecuteMemoryRead** (uint32\_t addr)
- **EjtagControlRegister WaitForEjtagMemoryOperation** (bool first=false)
- void **SerialExecHelper** ()
- **MicrochipPIC32DeviceStatusRegister GetStatus** ()
- **EjtagImplementationCodeRegister GetImpCode** ()

## Protected Attributes

- unsigned int **m\_devid**  
*Device ID code.*
- unsigned int **m\_stepping**  
*Stepping number.*
- const **MicrochipPIC32DeviceInfo \* m\_devinfo**  
*Device info.*

### 7.49.1 Detailed Description

A Microchip PIC32 microcontroller (MX, MZ, MM, etc)

### 7.49.2 Member Enumeration Documentation

#### 7.49.2.1 instructions

enum **MicrochipPIC32Device::instructions**

5-bit-wide JTAG instructions (from BSDL file and datasheet)

#### Enumerator

|                    |                                                      |
|--------------------|------------------------------------------------------|
| INST_BYPASS        | Standard JTAG bypass.                                |
| INST_IDCODE        | Read ID code.                                        |
| INST_IMPCODE       | Read implementation code.                            |
| INST_MTAP_SW_MCHP  | Selects Microchip scan chain.                        |
| INST_MTAP_SW_EJTAG | Selects EJTAG scan chain.                            |
| INST_MTAP_COMMAND  | Command to Microchip virtualized JTAG.               |
| INST_ADDRESS       | Select address register for memory ops.              |
| INST_DATA          | Select data register for memory ops.                 |
| INST_CONTROL       | Control register of some sort?                       |
| INST_ALL           | Selects address, data, control end to end in one DR. |
| INST_DEBUGBOOT     | Makes the CPU trap to debugger after a reset.        |
| INST_NORMALBOOT    | ...                                                  |

### 7.49.2.2 mtap\_instructions

enum `MicrochipPIC32Device::mtap_instructions`

8-bit instructions for Microchip virtual TAP (write to INST\_MTAP\_COMMAND data register)

#### Enumerator

|                    |                                     |
|--------------------|-------------------------------------|
| MCHP_STATUS        | Get status.                         |
| MCHP_ASSERT_RST    | Begin chip reset.                   |
| MCHP_DE_ASSERT_RST | End chip reset.                     |
| MCHP_ERASE         | Bulk-erase flash.                   |
| MCHP_FLASH_ENABLE  | Enable connecting the CPU to flash. |
| MCHP_FLASH_DISABLE | Disconnect the CPU from flash.      |
| MCHP_READ_CONFIG   | Force re-read of device config.     |

## 7.49.3 Member Function Documentation

### 7.49.3.1 Erase()

```
void MicrochipPIC32Device::Erase () [virtual]
```

Erases the device configuration and restores the device to a blank state.

After this function is called, regardless of success or failure, all existing connections to on-chip code become invalid.

#### Exceptions

|                      |                              |
|----------------------|------------------------------|
| <i>JtagException</i> | if the erase operation fails |
|----------------------|------------------------------|

Implements [ProgrammableDevice](#).

### 7.49.3.2 GetDescription()

```
std::string MicrochipPIC32Device::GetDescription () [virtual]
```

Gets a human-readable description of this device.

Example: "Xilinx XC6SLX45 stepping 3"

**Returns**

Device description

Implements [JtagDevice](#).

**7.49.3.3 IsProgrammed()**

```
bool MicrochipPIC32Device::IsProgrammed () [virtual]
```

Determines if this device is programmed or blank.

**Returns**

true if programmed, false if blank

Implements [ProgrammableDevice](#).

**7.49.3.4 PostInitProbes()**

```
void MicrochipPIC32Device::PostInitProbes (
 bool quiet) [virtual]
```

Does a post-initialization probe of the device to read debug ROMs etc.

**Parameters**

|              |                                                           |
|--------------|-----------------------------------------------------------|
| <i>quiet</i> | Do minimal probing to avoid triggering security lockdowns |
|--------------|-----------------------------------------------------------|

Implements [JtagDevice](#).

**7.49.3.5 Program()**

```
void MicrochipPIC32Device::Program (
 FirmwareImage * image) [virtual]
```

Loads a new firmware image onto the device.

After this function is called, regardless of success or failure, all existing connections to on-chip code become invalid.

**Exceptions**

|                               |                              |
|-------------------------------|------------------------------|
| <a href="#">JtagException</a> | if the erase operation fails |
|-------------------------------|------------------------------|



## Parameters

|              |                          |
|--------------|--------------------------|
| <i>image</i> | The parsed image to load |
|--------------|--------------------------|

Implements [ProgrammableDevice](#).

The documentation for this class was generated from the following files:

- [MicrochipPIC32Device.h](#)
- [MicrochipPIC32Device.cpp](#)

## 7.50 MicrochipPIC32DeviceInfo Struct Reference

Internal data structure storing properties of a single SKU in the PIC32 family.

```
#include <MicrochipPIC32Device.h>
```

### Public Attributes

- `uint16_t` [devid](#)  
*JTAG device ID.*
- `const char *` [name](#)  
*String name of device.*
- `unsigned int` [family](#)  
*Device family.*
- `unsigned int` [cpu](#)  
*CPU type.*
- `unsigned int` [sram\\_size](#)  
*SRAM capacity (kB)*
- `unsigned int` [program\\_flash\\_size](#)  
*Main program flash size (kB)*
- `float` [boot\\_flash\\_size](#)  
*Boot flash size (kB)*

### 7.50.1 Detailed Description

Internal data structure storing properties of a single SKU in the PIC32 family.

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

- [MicrochipPIC32Device.h](#)

## 7.51 MicrochipPIC32DeviceStatusRegister Union Reference

Status register for a Microchip PIC32 device.

```
#include <MicrochipPIC32Device.h>
```

## Public Member Functions

- ```

struct {
    unsigned int reset_active:1
    unsigned int flash_en:1
    unsigned int flash_busy:1
    unsigned int cfg_rdy:1
    unsigned int reserved2:1
    unsigned int nvm_error:1
    unsigned int reserved1:1
    unsigned int code_protect_off:1
} __attribute__((packed)) bits

```

Public Attributes

- uint8_t **word**
The raw status register value.

7.51.1 Detailed Description

Status register for a Microchip PIC32 device.

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

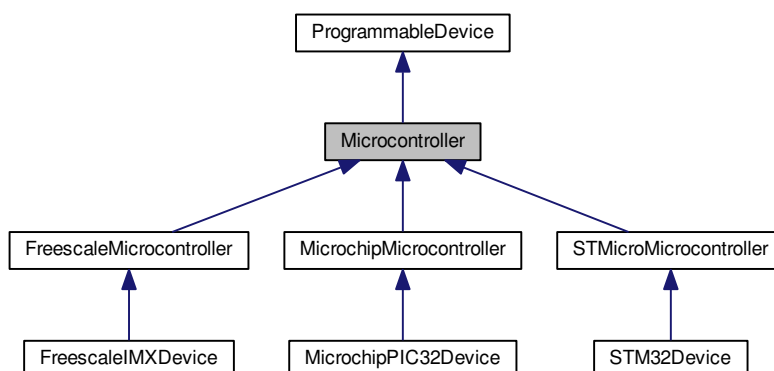
- [MicrochipPIC32Device.h](#)

7.52 Microcontroller Class Reference

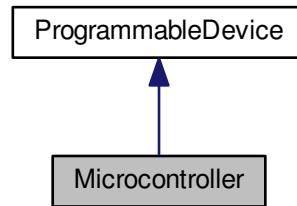
Generic base class for all microcontrollers.

```
#include <Microcontroller.h>
```

Inheritance diagram for Microcontroller:



Collaboration diagram for Microcontroller:



Public Member Functions

- virtual `FirmwareImage * LoadFirmwareImage (const unsigned char *data, size_t len)`
Parses an in-memory image of a firmware image into a format suitable for loading into the device.

7.52.1 Detailed Description

Generic base class for all microcontrollers.

7.52.2 Member Function Documentation

7.52.2.1 LoadFirmwareImage()

```
FirmwareImage * Microcontroller::LoadFirmwareImage (
    const unsigned char * data,
    size_t len ) [virtual]
```

Parses an in-memory image of a firmware image into a format suitable for loading into the device.

Exceptions

<code>JtagException</code>	if the image is malformed
----------------------------	---------------------------

Parameters

<code>data</code>	Pointer to the start of the firmware image, including headers
<code>len</code>	Length of the firmware image

Returns

Pointer to an [FirmwareImage](#) object suitable for passing to `Configure()`.

Implements [ProgrammableDevice](#).

Reimplemented in [STM32Device](#).

The documentation for this class was generated from the following files:

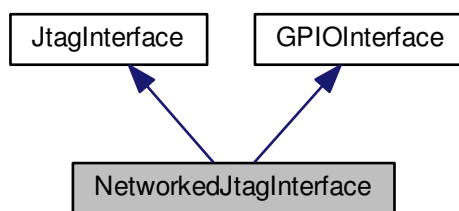
- [Microcontroller.h](#)
- [Microcontroller.cpp](#)

7.53 NetworkedJtagInterface Class Reference

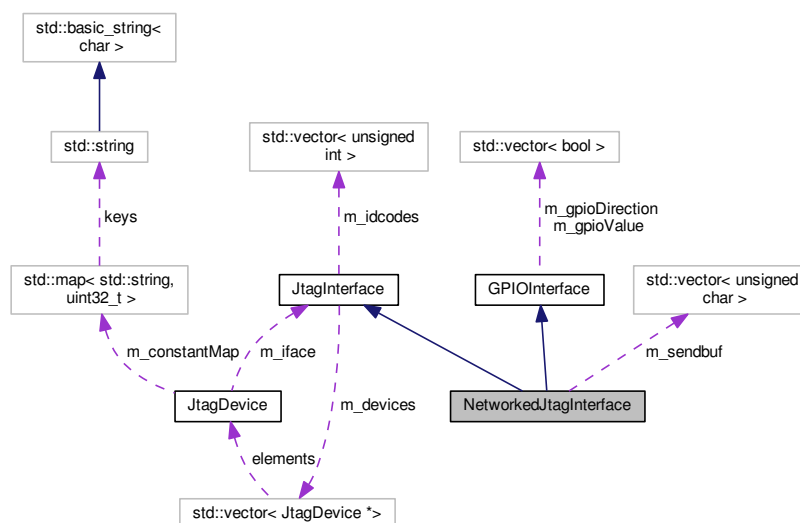
Thin wrapper around TCP sockets for talking to a jtagd instance.

```
#include <NetworkedJtagInterface.h>
```

Inheritance diagram for NetworkedJtagInterface:



Collaboration diagram for NetworkedJtagInterface:



Public Member Functions

- [NetworkedJtagInterface](#) ()
Creates the interface object but does not connect to a server.
- virtual [~NetworkedJtagInterface](#) ()
Disconnects from the server.
- void [Connect](#) (const std::string &server, uint16_t port)
Connects to a jtagd server.
- virtual std::string [GetName](#) ()
Gets the manufacturer-assigned name for this programming adapter.
- virtual std::string [GetSerial](#) ()
Gets the manufacturer-assigned serial number for this programming adapter, if any.
- virtual std::string [GetUserID](#) ()
Gets the user-assigned name for this JTAG adapter, if any.
- virtual int [GetFrequency](#) ()
Gets the clock frequency, in Hz, of the JTAG interface.
- virtual void [ShiftData](#) (bool last_tms, const unsigned char *send_data, unsigned char *rcv_data, size_t count)
Shifts data through TDI to TDO.
- virtual void [SendDummyClocks](#) (size_t n)
Sends the requested number of dummy clocks with TMS=0 and flushes the command to the interface.
- virtual void [SendDummyClocksDeferred](#) (size_t n)
Sends the requested number of dummy clocks with TMS=0 and does not flush the write pipeline.
- virtual void [Commit](#) ()
Commits the outstanding transactions to the adapter.
- virtual bool [IsSplitScanSupported](#) ()
Indicates if split (pipelined) DR scanning is supported.
- virtual bool [ShiftDataWriteOnly](#) (bool last_tms, const unsigned char *send_data, unsigned char *rcv_data, size_t count)
Shifts data through TDI to TDO.
- virtual bool [ShiftDataReadOnly](#) (unsigned char *rcv_data, size_t count)
Reads data from a [ShiftDataWriteOnly\(\)](#) call.
- virtual void [TestLogicReset](#) ()
Enters Test-Logic-Reset state by shifting six ones into TMS.
- virtual void [EnterShiftIR](#) ()
Enters Shift-IR state from Run-Test-Idle state.
- virtual void [LeaveExit1IR](#) ()
Leaves Exit1-IR state and returns to Run-Test-Idle.
- virtual void [EnterShiftDR](#) ()
Enters Shift-DR state from Run-Test-Idle state.
- virtual void [LeaveExit1DR](#) ()
Leaves Exit1-DR state and returns to Run-Test-Idle.
- virtual void [ResetToIdle](#) ()
Resets the TAP and enters Run-Test-Idle state.
- virtual void [ReadGpioState](#) ()
Reads all of the device's GPIO pins into the internal buffer.
- virtual void [WriteGpioState](#) ()
Writes all of the device's GPIO pin values to the device.
- bool [IsGPIOCapable](#) ()

Static Public Member Functions

- static std::string [GetAPIVersion](#) ()
Returns the protocol version.
- static int [GetInterfaceCount](#) ()
Returns the constant 1.

Protected Member Functions

- virtual size_t [GetShiftOpCount](#) ()
Gets the number of shift operations performed on this interface.
- virtual size_t [GetRecoverableErrorCount](#) ()
Gets the number of errors this interface has recovered from (USB retransmits, etc)
- virtual size_t [GetDataBitCount](#) ()
Gets the number of data bits this interface has shifted.
- virtual size_t [GetModeBitCount](#) ()
Gets the number of mode bits this interface has shifted.
- virtual size_t [GetDummyClockCount](#) ()
Gets the number of dummy clocks this interface has sent.
- void **BufferedSend** (const unsigned char *buf, int count)
- void **SendFlush** ()

Protected Attributes

- Socket [m_socket](#)
Our socket.
- std::vector< unsigned char > **m_sendbuf**

7.53.1 Detailed Description

Thin wrapper around TCP sockets for talking to a jtagd instance.

7.53.2 Member Function Documentation

7.53.2.1 Commit()

```
void NetworkedJtagInterface::Commit ( ) [virtual]
```

Commits the outstanding transactions to the adapter.

No-op unless the adapter supports queueing of multiple writes.

This function is automatically called when [SendDummyClocks\(\)](#) is called or any readback is performed. Most adapter classes will automatically call it when the transmit queue reaches a certain size.

This function can be called at any time to ensure all pending operations have executed.

Exceptions

JtagException	in case of error
-------------------------------	------------------

Reimplemented from [JtagInterface](#).

7.53.2.2 Connect()

```
void NetworkedJtagInterface::Connect (
    const std::string & server,
    uint16_t port )
```

Connects to a jtagd server.

Exceptions

JtagException	if the connection could not be established
-------------------------------	--

Parameters

<i>server</i>	Hostname of the server to connect to
<i>port</i>	Port number (in host byte ordering) the server is running on

7.53.2.3 EnterShiftDR()

```
void NetworkedJtagInterface::EnterShiftDR ( ) [virtual]
```

Enters Shift-DR state from Run-Test-Idle state.

Exceptions

JtagException	if ShiftTMS() fails
-------------------------------	---------------------

Reimplemented from [JtagInterface](#).

7.53.2.4 EnterShiftIR()

```
void NetworkedJtagInterface::EnterShiftIR ( ) [virtual]
```

Enters Shift-IR state from Run-Test-Idle state.

Exceptions

JtagException	if ShiftTMS() fails
-------------------------------	---------------------

Reimplemented from [JtagInterface](#).

7.53.2.5 GetDataBitCount()

```
size_t NetworkedJtagInterface::GetDataBitCount ( ) [protected], [virtual]
```

Gets the number of data bits this interface has shifted.

Exceptions

JtagException	on failure
-------------------------------	------------

Returns

Number of data bits shifted

Reimplemented from [JtagInterface](#).

7.53.2.6 GetDummyClockCount()

```
size_t NetworkedJtagInterface::GetDummyClockCount ( ) [protected], [virtual]
```

Gets the number of dummy clocks this interface has sent.

Exceptions

JtagException	on failure
-------------------------------	------------

Returns

Number of dummy clocks sent

Reimplemented from [JtagInterface](#).

7.53.2.7 GetFrequency()

```
int NetworkedJtagInterface::GetFrequency ( ) [virtual]
```

Gets the clock frequency, in Hz, of the JTAG interface.

Returns

The clock frequency

Implements [JtagInterface](#).

7.53.2.8 GetModeBitCount()

```
size_t NetworkedJtagInterface::GetModeBitCount ( ) [protected], [virtual]
```

Gets the number of mode bits this interface has shifted.

Exceptions

JtagException	on failure
-------------------------------	------------

Returns

Number of mode bits shifted

Reimplemented from [JtagInterface](#).

7.53.2.9 GetName()

```
std::string NetworkedJtagInterface::GetName ( ) [virtual]
```

Gets the manufacturer-assigned name for this programming adapter.

This is usually the model number but is sometimes something more generic like "Digilent Adept USB Device".

Returns

The device name

Implements [JtagInterface](#).

7.53.2.10 GetRecoverableErrorCount()

```
size_t NetworkedJtagInterface::GetRecoverableErrorCount ( ) [protected], [virtual]
```

Gets the number of errors this interface has recovered from (USB retransmits, etc)

Exceptions

JtagException	on failure
-------------------------------	------------

Returns

Number of recoverable errors

Reimplemented from [JtagInterface](#).

7.53.2.11 GetSerial()

```
std::string NetworkedJtagInterface::GetSerial ( ) [virtual]
```

Gets the manufacturer-assigned serial number for this programming adapter, if any.

Derived classes may choose to return the user ID, an empty string, or another default value if no serial number has been assigned.

Returns

The serial number

Implements [JtagInterface](#).

7.53.2.12 GetShiftOpCount()

```
size_t NetworkedJtagInterface::GetShiftOpCount ( ) [protected], [virtual]
```

Gets the number of shift operations performed on this interface.

Exceptions

JtagException	on failure
-------------------------------	------------

Returns

Number of shift operations

Reimplemented from [JtagInterface](#).

7.53.2.13 GetUserID()

```
std::string NetworkedJtagInterface::GetUserID ( ) [virtual]
```

Gets the user-assigned name for this JTAG adapter, if any.

Derived classes may choose to return the serial number, an empty string, or another default value if no name has been assigned.

Returns

The name for this adapter.

Implements [JtagInterface](#).

7.53.2.14 IsSplitScanSupported()

```
bool NetworkedJtagInterface::IsSplitScanSupported ( ) [virtual]
```

Indicates if split (pipelined) DR scanning is supported.

Split scanning allows the write halves of several scan operations to take place in one driver-level write call, followed by the read halves in order, to reduce the impact of driver/bus latency on throughput.

If split scanning is not supported, [ScanDRSplitWrite\(\)](#) will behave identically to [ScanDR\(\)](#) and [ScanDRSplitRead\(\)](#) will be a no-op. This ensures that using the split write commands will work correctly regardless of whether the adapter supports split scanning in hardware.

Reimplemented from [JtagInterface](#).

7.53.2.15 LeaveExit1DR()

```
void NetworkedJtagInterface::LeaveExit1DR ( ) [virtual]
```

Leaves Exit1-DR state and returns to Run-Test-Idle.

Exceptions

JtagException	if ShiftTMS() fails
-------------------------------	---------------------

Reimplemented from [JtagInterface](#).

7.53.2.16 LeaveExit1IR()

```
void NetworkedJtagInterface::LeaveExit1IR ( ) [virtual]
```

Leaves Exit1-IR state and returns to Run-Test-Idle.

Exceptions

JtagException	if ShiftTMS() fails
-------------------------------	---------------------

Reimplemented from [JtagInterface](#).

7.53.2.17 ResetToIdle()

```
void NetworkedJtagInterface::ResetToIdle ( ) [virtual]
```

Resets the TAP and enters Run-Test-Idle state.

Exceptions

JtagException	if ShiftTMS() fails
-------------------------------	---------------------

Reimplemented from [JtagInterface](#).

7.53.2.18 SendDummyClocks()

```
void NetworkedJtagInterface::SendDummyClocks (
    size_t n ) [virtual]
```

Sends the requested number of dummy clocks with TMS=0 and flushes the command to the interface.

Since dummy clocks are often used as a delay element for programming algorithms etc, this function flushes the write buffer to ensure immediate execution.

Exceptions

JtagException	may be thrown if the scan operation fails
-------------------------------	---

Parameters

<i>n</i>	Number of dummy clocks to send
----------	--------------------------------

Implements [JtagInterface](#).

7.53.2.19 SendDummyClocksDeferred()

```
void NetworkedJtagInterface::SendDummyClocksDeferred (
    size_t n ) [virtual]
```

Sends the requested number of dummy clocks with TMS=0 and does not flush the write pipeline.

Exceptions

JtagException	may be thrown if the scan operation fails
-------------------------------	---

Parameters

<i>n</i>	Number of dummy clocks to send
----------	--------------------------------

Reimplemented from [JtagInterface](#).

7.53.2.20 ShiftData()

```
void NetworkedJtagInterface::ShiftData (
    bool last_tms,
    const unsigned char * send_data,
    unsigned char * rcv_data,
    size_t count ) [virtual]
```

Shifts data through TDI to TDO.

The LSB of send_data[0] is sent first; the MSB of send_data[0] is followed by the LSB of send_data[1].

Parameters

<i>last_tms</i>	Different TMS value to use for last bit
<i>send_data</i>	Data to shift into TDI
<i>rcv_data</i>	Data to shift out of TDO (may be NULL)
<i>count</i>	Number of bits to shift

Implements [JtagInterface](#).

7.53.2.21 ShiftDataReadOnly()

```
bool NetworkedJtagInterface::ShiftDataReadOnly (
    unsigned char * rcv_data,
    size_t count ) [virtual]
```

Reads data from a [ShiftDataWriteOnly\(\)](#) call.

For more information on split (pipelined) scan operations see [ShiftDataWriteOnly\(\)](#).

Returns

True if the read was executed, false if a no-op

Parameters

<i>rcv_data</i>	Data to shift out of TDO (may be NULL)
<i>count</i>	Number of bits to shift

Reimplemented from [JtagInterface](#).

7.53.2.22 ShiftDataWriteOnly()

```
bool NetworkedJtagInterface::ShiftDataWriteOnly (
    bool last_tms,
    const unsigned char * send_data,
    unsigned char * rcv_data,
    size_t count ) [virtual]
```

Shifts data through TDI to TDO.

The LSB of `send_data[0]` is sent first; the MSB of `send_data[0]` is followed by the LSB of `send_data[1]`.

If split (pipelined) scanning is supported by the adapter, this function performs the write half of the shift operation only; the read is buffered in the JTAG adapter and no readback is performed until [ShiftDataReadOnly\(\)](#) is called. This allows several shift operations to occur in sequence without incurring a USB turnaround delay or other driver latency overhead for each shift operation.

If split scanning is not supported this call is equivalent to [ShiftData\(\)](#) and [ShiftDataReadOnly\(\)](#) is a no-op.

This function MUST be followed by either another [ShiftDataWriteOnly\(\)](#) call, a [ShiftTMS\(\)](#) call, or a [ShiftDataReadOnly\(\)](#) call. There must be exactly one [ShiftDataReadOnly\(\)](#) call for each [ShiftDataWriteOnly\(\)](#) call and they must be in order with the same `rcv_data` and `count` values. The result of doing otherwise is undefined.

Returns

True if the read was deferred, false if not

Parameters

<i>last_tms</i>	Different TMS value to use for last bit
<i>send_data</i>	Data to shift into TDI
<i>rcv_data</i>	Data to shift out of TDO (may be NULL)
<i>count</i>	Number of bits to shift

Reimplemented from [JtagInterface](#).

7.53.2.23 TestLogicReset()

```
void NetworkedJtagInterface::TestLogicReset ( ) [virtual]
```

Enters Test-Logic-Reset state by shifting six ones into TMS.

Exceptions

JtagException	if ShiftTMS() fails
-------------------------------	---------------------

Reimplemented from [JtagInterface](#).

The documentation for this class was generated from the following files:

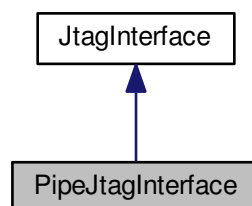
- [NetworkedJtagInterface.h](#)
- [NetworkedJtagInterface.cpp](#)

7.54 PipeJtagInterface Class Reference

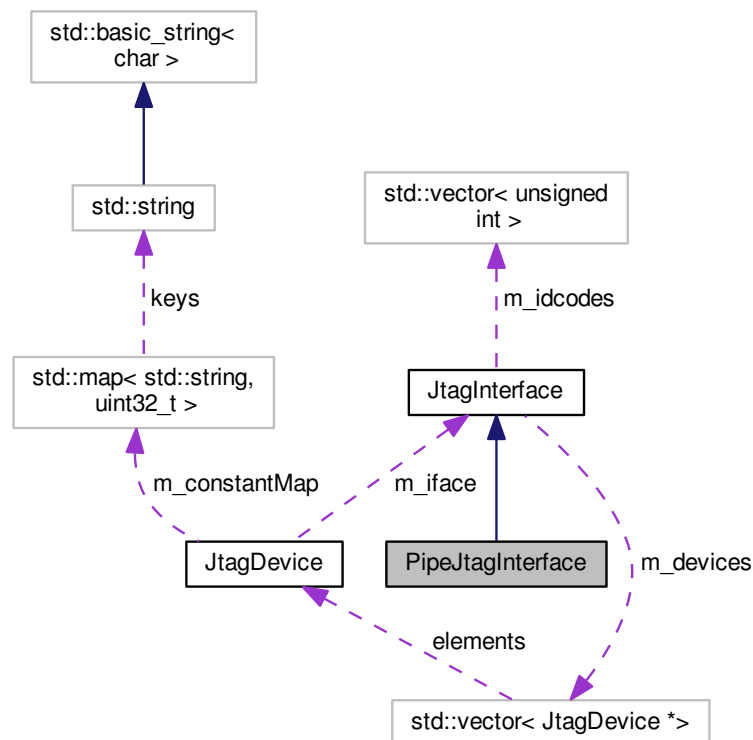
Thin wrapper around pipes for talking to an openfpga JtagPipeBridge.

```
#include <PipeJtagInterface.h>
```

Inheritance diagram for PipeJtagInterface:



Collaboration diagram for PipeJtagInterface:



Public Member Functions

- [PipeJtagInterface](#) ()
Creates the interface object and connects to the pipes (TODO: support more than one)
- virtual [~PipeJtagInterface](#) ()
Disconnects from the server.
- virtual [std::string GetName](#) ()
Gets the manufacturer-assigned name for this programming adapter.
- virtual [std::string GetSerial](#) ()
Gets the manufacturer-assigned serial number for this programming adapter, if any.
- virtual [std::string GetUserID](#) ()
Gets the user-assigned name for this JTAG adapter, if any.
- virtual [int GetFrequency](#) ()
Gets the clock frequency, in Hz, of the JTAG interface.
- virtual void [ShiftData](#) (bool last_tms, const unsigned char *send_data, unsigned char *rcv_data, size_t count)
Shifts data through TDI to TDO.
- virtual void [SendDummyClocks](#) (size_t n)
Sends the requested number of dummy clocks with TMS=0 and flushes the command to the interface.
- virtual void [SendDummyClocksDeferred](#) (size_t n)
Sends the requested number of dummy clocks with TMS=0 and does not flush the write pipeline.
- virtual void [Commit](#) ()

- Commits the outstanding transactions to the adapter.*

 - virtual bool [IsSplitScanSupported](#) ()
 - Indicates if split (pipelined) DR scanning is supported.*
 - virtual bool [ShiftDataWriteOnly](#) (bool last_tms, const unsigned char *send_data, unsigned char *rcv_data, size_t count)
 - Shifts data through TDI to TDO.*
 - virtual bool [ShiftDataReadOnly](#) (unsigned char *rcv_data, size_t count)
 - Reads data from a [ShiftDataWriteOnly\(\)](#) call.*
 - virtual void [TestLogicReset](#) ()
 - Enters Test-Logic-Reset state by shifting six ones into TMS.*
 - virtual void [EnterShiftIR](#) ()
 - Enters Shift-IR state from Run-Test-Idle state.*
 - virtual void [LeaveExit1IR](#) ()
 - Leaves Exit1-IR state and returns to Run-Test-Idle.*
 - virtual void [EnterShiftDR](#) ()
 - Enters Shift-DR state from Run-Test-Idle state.*
 - virtual void [LeaveExit1DR](#) ()
 - Leaves Exit1-DR state and returns to Run-Test-Idle.*
 - virtual void [ResetToldle](#) ()
 - Resets the TAP and enters Run-Test-Idle state.*

Static Public Member Functions

- static std::string [GetAPIVersion](#) ()
 - Returns the protocol version.*
- static int [GetInterfaceCount](#) ()
 - Returns the constant 1.*

Protected Member Functions

- virtual size_t [GetShiftOpCount](#) ()
 - Gets the number of shift operations performed on this interface.*
- virtual size_t [GetRecoverableErrorCount](#) ()
 - Gets the number of errors this interface has recovered from (USB retransmits, etc)*
- virtual size_t [GetDataBitCount](#) ()
 - Gets the number of data bits this interface has shifted.*
- virtual size_t [GetModeBitCount](#) ()
 - Gets the number of mode bits this interface has shifted.*
- virtual size_t [GetDummyClockCount](#) ()
 - Gets the number of dummy clocks this interface has sent.*

Protected Attributes

- FILE * [m_readpipe](#)
 - Our pipe.*
- FILE * [m_writepipe](#)

7.54.1 Detailed Description

Thin wrapper around pipes for talking to an openfpga JtagPipeBridge.

7.54.2 Member Function Documentation

7.54.2.1 Commit()

```
void PipeJtagInterface::Commit ( ) [virtual]
```

Commits the outstanding transactions to the adapter.

No-op unless the adapter supports queueing of multiple writes.

This function is automatically called when [SendDummyClocks\(\)](#) is called or any readback is performed. Most adapter classes will automatically call it when the transmit queue reaches a certain size.

This function can be called at any time to ensure all pending operations have executed.

Exceptions

JtagException	in case of error
-------------------------------	------------------

Reimplemented from [JtagInterface](#).

7.54.2.2 EnterShiftDR()

```
void PipeJtagInterface::EnterShiftDR ( ) [virtual]
```

Enters Shift-DR state from Run-Test-Idle state.

Exceptions

JtagException	if ShiftTMS() fails
-------------------------------	---------------------

Reimplemented from [JtagInterface](#).

7.54.2.3 EnterShiftIR()

```
void PipeJtagInterface::EnterShiftIR ( ) [virtual]
```

Enters Shift-IR state from Run-Test-Idle state.

Exceptions

JtagException	if ShiftTMS() fails
-------------------------------	---------------------

Reimplemented from [JtagInterface](#).

7.54.2.4 GetDataBitCount()

```
size_t PipeJtagInterface::GetDataBitCount ( ) [protected], [virtual]
```

Gets the number of data bits this interface has shifted.

Exceptions

JtagException	on failure
-------------------------------	------------

Returns

Number of data bits shifted

Reimplemented from [JtagInterface](#).

7.54.2.5 GetDummyClockCount()

```
size_t PipeJtagInterface::GetDummyClockCount ( ) [protected], [virtual]
```

Gets the number of dummy clocks this interface has sent.

Exceptions

JtagException	on failure
-------------------------------	------------

Returns

Number of dummy clocks sent

Reimplemented from [JtagInterface](#).

7.54.2.6 GetFrequency()

```
int PipeJtagInterface::GetFrequency ( ) [virtual]
```

Gets the clock frequency, in Hz, of the JTAG interface.

Returns

The clock frequency

Implements [JtagInterface](#).

7.54.2.7 GetModeBitCount()

```
size_t PipeJtagInterface::GetModeBitCount ( ) [protected], [virtual]
```

Gets the number of mode bits this interface has shifted.

Exceptions

JtagException	on failure
-------------------------------	------------

Returns

Number of mode bits shifted

Reimplemented from [JtagInterface](#).

7.54.2.8 GetName()

```
string PipeJtagInterface::GetName ( ) [virtual]
```

Gets the manufacturer-assigned name for this programming adapter.

This is usually the model number but is sometimes something more generic like "Digilent Adept USB Device".

Returns

The device name

Implements [JtagInterface](#).

7.54.2.9 GetRecoverableErrorCount()

```
size_t PipeJtagInterface::GetRecoverableErrorCount ( ) [protected], [virtual]
```

Gets the number of errors this interface has recovered from (USB retransmits, etc)

Exceptions

JtagException	on failure
-------------------------------	------------

Returns

Number of recoverable errors

Reimplemented from [JtagInterface](#).

7.54.2.10 GetSerial()

```
string PipeJtagInterface::GetSerial ( ) [virtual]
```

Gets the manufacturer-assigned serial number for this programming adapter, if any.

Derived classes may choose to return the user ID, an empty string, or another default value if no serial number has been assigned.

Returns

The serial number

Implements [JtagInterface](#).

7.54.2.11 GetShiftOpCount()

```
size_t PipeJtagInterface::GetShiftOpCount ( ) [protected], [virtual]
```

Gets the number of shift operations performed on this interface.

Exceptions

JtagException	on failure
-------------------------------	------------

Returns

Number of shift operations

Reimplemented from [JtagInterface](#).

7.54.2.12 GetUserID()

```
string PipeJtagInterface::GetUserID ( ) [virtual]
```

Gets the user-assigned name for this JTAG adapter, if any.

Derived classes may choose to return the serial number, an empty string, or another default value if no name has been assigned.

Returns

The name for this adapter.

Implements [JtagInterface](#).

7.54.2.13 IsSplitScanSupported()

```
bool PipeJtagInterface::IsSplitScanSupported ( ) [virtual]
```

Indicates if split (pipelined) DR scanning is supported.

Split scanning allows the write halves of several scan operations to take place in one driver-level write call, followed by the read halves in order, to reduce the impact of driver/bus latency on throughput.

If split scanning is not supported, [ScanDRSplitWrite\(\)](#) will behave identically to [ScanDR\(\)](#) and [ScanDRSplitRead\(\)](#) will be a no-op. This ensures that using the split write commands will work correctly regardless of whether the adapter supports split scanning in hardware.

Reimplemented from [JtagInterface](#).

7.54.2.14 LeaveExit1DR()

```
void PipeJtagInterface::LeaveExit1DR ( ) [virtual]
```

Leaves Exit1-DR state and returns to Run-Test-Idle.

Exceptions

JtagException	if ShiftTMS() fails
-------------------------------	---------------------

Reimplemented from [JtagInterface](#).

7.54.2.15 LeaveExit1IR()

```
void PipeJtagInterface::LeaveExit1IR ( ) [virtual]
```

Leaves Exit1-IR state and returns to Run-Test-Idle.

Exceptions

JtagException	if ShiftTMS() fails
-------------------------------	---------------------

Reimplemented from [JtagInterface](#).

7.54.2.16 ResetToIdle()

```
void PipeJtagInterface::ResetToIdle ( ) [virtual]
```

Resets the TAP and enters Run-Test-Idle state.

Exceptions

JtagException	if ShiftTMS() fails
-------------------------------	---------------------

Reimplemented from [JtagInterface](#).

7.54.2.17 SendDummyClocks()

```
void PipeJtagInterface::SendDummyClocks (
    size_t n ) [virtual]
```

Sends the requested number of dummy clocks with TMS=0 and flushes the command to the interface.

Since dummy clocks are often used as a delay element for programming algorithms etc, this function flushes the write buffer to ensure immediate execution.

Exceptions

JtagException	may be thrown if the scan operation fails
-------------------------------	---

Parameters

<i>n</i>	Number of dummy clocks to send
----------	--------------------------------

Implements [JtagInterface](#).

7.54.2.18 SendDummyClocksDeferred()

```
void PipeJtagInterface::SendDummyClocksDeferred (
    size_t n ) [virtual]
```

Sends the requested number of dummy clocks with TMS=0 and does not flush the write pipeline.

Exceptions

JtagException	may be thrown if the scan operation fails
-------------------------------	---

Parameters

<i>n</i>	Number of dummy clocks to send
----------	--------------------------------

Reimplemented from [JtagInterface](#).

7.54.2.19 ShiftData()

```
void PipeJtagInterface::ShiftData (
    bool last_tms,
    const unsigned char * send_data,
    unsigned char * rcv_data,
    size_t count ) [virtual]
```

Shifts data through TDI to TDO.

The LSB of `send_data[0]` is sent first; the MSB of `send_data[0]` is followed by the LSB of `send_data[1]`.

Parameters

<i>last_tms</i>	Different TMS value to use for last bit
<i>send_data</i>	Data to shift into TDI
<i>rcv_data</i>	Data to shift out of TDO (may be NULL)
<i>count</i>	Number of bits to shift

Implements [JtagInterface](#).

7.54.2.20 ShiftDataReadOnly()

```
bool PipeJtagInterface::ShiftDataReadOnly (
    unsigned char * rcv_data,
    size_t count ) [virtual]
```

Reads data from a [ShiftDataWriteOnly\(\)](#) call.

For more information on split (pipelined) scan operations see [ShiftDataWriteOnly\(\)](#).

Returns

True if the read was executed, false if a no-op

Parameters

<i>rcv_data</i>	Data to shift out of TDO (may be NULL)
<i>count</i>	Number of bits to shift

Reimplemented from [JtagInterface](#).

7.54.2.21 ShiftDataWriteOnly()

```
bool PipeJtagInterface::ShiftDataWriteOnly (
    bool last_tms,
    const unsigned char * send_data,
    unsigned char * rcv_data,
    size_t count ) [virtual]
```

Shifts data through TDI to TDO.

The LSB of `send_data[0]` is sent first; the MSB of `send_data[0]` is followed by the LSB of `send_data[1]`.

If split (pipelined) scanning is supported by the adapter, this function performs the write half of the shift operation only; the read is buffered in the JTAG adapter and no readback is performed until [ShiftDataReadOnly\(\)](#) is called. This allows several shift operations to occur in sequence without incurring a USB turnaround delay or other driver latency overhead for each shift operation.

If split scanning is not supported this call is equivalent to [ShiftData\(\)](#) and [ShiftDataReadOnly\(\)](#) is a no-op.

This function MUST be followed by either another [ShiftDataWriteOnly\(\)](#) call, a [ShiftTMS\(\)](#) call, or a [ShiftDataReadOnly\(\)](#) call. There must be exactly one [ShiftDataReadOnly\(\)](#) call for each [ShiftDataWriteOnly\(\)](#) call and they must be in order with the same `rcv_data` and `count` values. The result of doing otherwise is undefined.

Returns

True if the read was deferred, false if not

Parameters

<i>last_tms</i>	Different TMS value to use for last bit
<i>send_data</i>	Data to shift into TDI
<i>rcv_data</i>	Data to shift out of TDO (may be NULL)
<i>count</i>	Number of bits to shift

Reimplemented from [JtagInterface](#).

7.54.2.22 TestLogicReset()

```
void PipeJtagInterface::TestLogicReset ( ) [virtual]
```

Enters Test-Logic-Reset state by shifting six ones into TMS.

Exceptions

JtagException	if ShiftTMS() fails
-------------------------------	---------------------

Reimplemented from [JtagInterface](#).

The documentation for this class was generated from the following files:

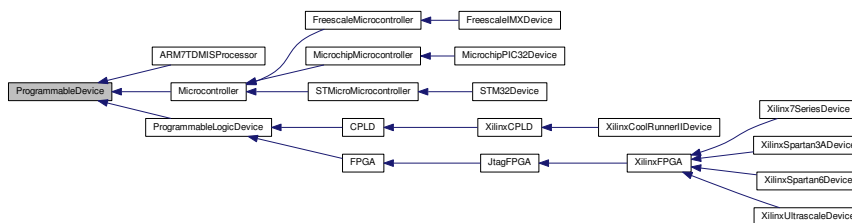
- [PipeJtagInterface.h](#)
- [PipeJtagInterface.cpp](#)

7.55 ProgrammableDevice Class Reference

Generic base class for all programmable devices (PLD, MCU, flash, etc)

```
#include <ProgrammableDevice.h>
```

Inheritance diagram for ProgrammableDevice:



Public Member Functions

- virtual bool [IsProgrammed](#) ()=0
Determines if this device is programmed or blank.
- [FirmwareImage](#) * [LoadFirmwareImage](#) (std::string fname)
Wrapper for [LoadFirmwareImage](#)().
- virtual [FirmwareImage](#) * [LoadFirmwareImage](#) (const unsigned char *data, size_t len)=0
Parses an in-memory image of a firmware image into a format suitable for loading into the device.
- virtual void [Erase](#) ()=0
Erases the device configuration and restores the device to a blank state.
- virtual void [Program](#) ([FirmwareImage](#) *image)=0
Loads a new firmware image onto the device.

7.55.1 Detailed Description

Generic base class for all programmable devices (PLD, MCU, flash, etc)

7.55.2 Member Function Documentation

7.55.2.1 Erase()

```
virtual void ProgrammableDevice::Erase ( ) [pure virtual]
```

Erases the device configuration and restores the device to a blank state.

After this function is called, regardless of success or failure, all existing connections to on-chip code become invalid.

Exceptions

JtagException	if the erase operation fails
-------------------------------	------------------------------

Implemented in [MicrochipPIC32Device](#), [Xilinx7SeriesDevice](#), [XilinxUltrascaleDevice](#), [XilinxSpartan6Device](#), [XilinxSpartan3ADevice](#), [XilinxCoolRunnerIIDevice](#), [ARM7TDMISProcessor](#), [FreescaleIMXDevice](#), and [STM32←Device](#).

7.55.2.2 IsProgrammed()

```
virtual bool ProgrammableDevice::IsProgrammed ( ) [pure virtual]
```

Determines if this device is programmed or blank.

Returns

true if programmed, false if blank

Implemented in [MicrochipPIC32Device](#), [Xilinx7SeriesDevice](#), [XilinxUltrascaleDevice](#), [XilinxSpartan6Device](#), [XilinxSpartan3ADevice](#), [XilinxCoolRunnerIIDevice](#), [ARM7TDMISProcessor](#), [FreescaleIMXDevice](#), and [STM32←Device](#).

7.55.2.3 LoadFirmwareImage() [1/2]

```
FirmwareImage * ProgrammableDevice::LoadFirmwareImage (
    std::string fname )
```

Wrapper for [LoadFirmwareImage\(\)](#).

Loads the file and passes it to [LoadFirmwareImage\(\)](#)

Exceptions

JtagException	if the file could not be opened or the image is invalid
-------------------------------	---

Parameters

<i>fname</i>	Name of the image to load
--------------	---------------------------

Returns

Pointer to an [FirmwareImage](#) object suitable for passing to [Program\(\)](#).

7.55.2.4 LoadFirmwareImage() [2/2]

```
virtual FirmwareImage* ProgrammableDevice::LoadFirmwareImage (
    const unsigned char * data,
    size_t len ) [pure virtual]
```

Parses an in-memory image of a firmware image into a format suitable for loading into the device.

Exceptions

JtagException	if the image is malformed
-------------------------------	---------------------------

Parameters

<i>data</i>	Pointer to the start of the firmware image, including headers
<i>len</i>	Length of the firmware image

Returns

Pointer to an [FirmwareImage](#) object suitable for passing to [Configure\(\)](#).

Implemented in [Xilinx7SeriesDevice](#), [XilinxUltrascaleDevice](#), [XilinxSpartan6Device](#), [XilinxSpartan3ADevice](#), [XilinxCoolRunnerIIDevice](#), [ARM7TDMISProcessor](#), [STM32Device](#), and [Microcontroller](#).

7.55.2.5 Program()

```
virtual void ProgrammableDevice::Program (
    FirmwareImage * image ) [pure virtual]
```

Loads a new firmware image onto the device.

After this function is called, regardless of success or failure, all existing connections to on-chip code become invalid.

Exceptions

JtagException	if the erase operation fails
-------------------------------	------------------------------

Parameters

<i>image</i>	The parsed image to load
--------------	--------------------------

Implemented in [MicrochipPIC32Device](#), [Xilinx7SeriesDevice](#), [XilinxUltrascaleDevice](#), [XilinxSpartan6Device](#), [XilinxSpartan3ADevice](#), [XilinxCoolRunnerIIDevice](#), [ARM7TDMISProcessor](#), [FreescaleIMXDevice](#), and [STM32Device](#).

The documentation for this class was generated from the following files:

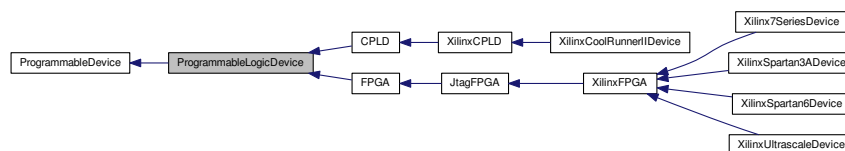
- [ProgrammableDevice.h](#)
- [ProgrammableDevice.cpp](#)

7.56 ProgrammableLogicDevice Class Reference

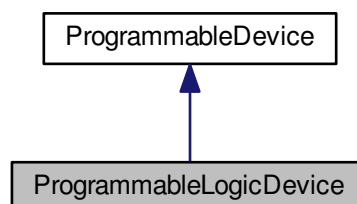
Generic base class for all programmable logic devices (FPGA and CPLD)

```
#include <ProgrammableLogicDevice.h>
```

Inheritance diagram for ProgrammableLogicDevice:



Collaboration diagram for ProgrammableLogicDevice:



Additional Inherited Members

7.56.1 Detailed Description

Generic base class for all programmable logic devices ([FPGA](#) and [CPLD](#))

The documentation for this class was generated from the following files:

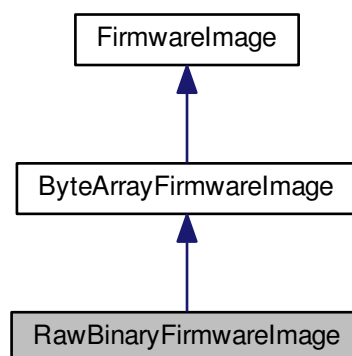
- [ProgrammableLogicDevice.h](#)
- [ProgrammableLogicDevice.cpp](#)

7.57 RawBinaryFirmwareImage Class Reference

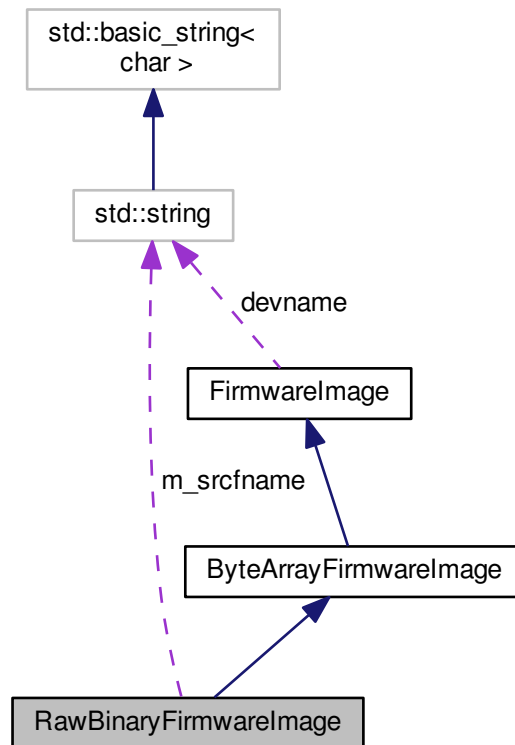
Raw binary firmware image loaded from a file.

```
#include <RawBinaryFirmwareImage.h>
```

Inheritance diagram for RawBinaryFirmwareImage:



Collaboration diagram for RawBinaryFirmwareImage:



Public Member Functions

- [RawBinaryFirmwareImage](#) (std::string fname, std::string sdevname)
Initializes this object to empty.
- virtual [~RawBinaryFirmwareImage](#) ()
Free bitstream memory.
- virtual std::string **GetDescription** ()

Public Attributes

- std::string [m_srcfname](#)
Source file name.

7.57.1 Detailed Description

Raw binary firmware image loaded from a file.

The documentation for this class was generated from the following files:

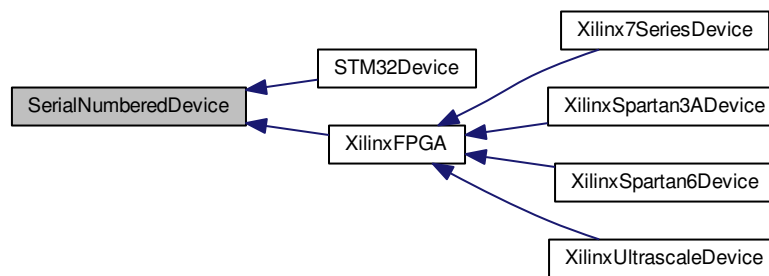
- [RawBinaryFirmwareImage.h](#)
- [RawBinaryFirmwareImage.cpp](#)

7.58 SerialNumberedDevice Class Reference

Abstract base class for all devices that have a unique die serial number.

```
#include <SerialNumberedDevice.h>
```

Inheritance diagram for SerialNumberedDevice:



Public Member Functions

- virtual bool [ReadingSerialRequiresReset](#) ()=0
True if reading this serial number requires a device reset.
- virtual int [GetSerialNumberLength](#) ()=0
Gets the length of the device's unique serial number, in bytes (rounded up to the nearest whole byte).
- virtual int [GetSerialNumberLengthBits](#) ()=0
Gets the length of the device's unique serial number, in bits.
- virtual void [GetSerialNumber](#) (unsigned char *data)=0
Gets the device's unique serial number.
- virtual std::string [GetPrettyPrintedSerialNumber](#) ()
Returns a pretty-printed serial number.

7.58.1 Detailed Description

Abstract base class for all devices that have a unique die serial number.

7.58.2 Member Function Documentation

7.58.2.1 GetPrettyPrintedSerialNumber()

```
string SerialNumberedDevice::GetPrettyPrintedSerialNumber ( ) [virtual]
```

Returns a pretty-printed serial number.

Most serial numbers have no inherent meaning but some contain encoded lot numbers etc.

This function returns a human-readable version of the serial number, if such exists. Otherwise, a hex encoding of [GetSerialNumber\(\)](#) is returned.

Reimplemented in [STM32Device](#).

7.58.2.2 GetSerialNumber()

```
virtual void SerialNumberedDevice::GetSerialNumber (
    unsigned char * data ) [pure virtual]
```

Gets the device's unique serial number.

Note that some architectures, such as Spartan-6, cannot read the serial number over JTAG without erasing the [FPGA](#) configuration. If this is the case, calling this function will automatically erase the [FPGA](#).

Call [ReadingSerialRequiresReset\(\)](#) to see if this is the case.

Exceptions

JtagException	if an error occurs during the read operation
-------------------------------	--

Parameters

<i>data</i>	Buffer to store the serial number into. Must be at least as large as the size given by GetSerialNumberLength() .
-------------	--

Implemented in [Xilinx7SeriesDevice](#), [XilinxUltrascaleDevice](#), [XilinxSpartan6Device](#), [XilinxSpartan3ADevice](#), and [STM32Device](#).

7.58.2.3 GetSerialNumberLength()

```
virtual int SerialNumberedDevice::GetSerialNumberLength ( ) [pure virtual]
```

Gets the length of the device's unique serial number, in bytes (rounded up to the nearest whole byte).

Returns

Serial number length

Implemented in [Xilinx7SeriesDevice](#), [XilinxUltrascaleDevice](#), [XilinxSpartan6Device](#), [XilinxSpartan3ADevice](#), and [STM32Device](#).

7.58.2.4 GetSerialNumberLengthBits()

```
virtual int SerialNumberedDevice::GetSerialNumberLengthBits ( ) [pure virtual]
```

Gets the length of the device's unique serial number, in bits.

Returns

Serial number length

Implemented in [Xilinx7SeriesDevice](#), [XilinxUltrascaleDevice](#), [XilinxSpartan6Device](#), [XilinxSpartan3ADevice](#), and [STM32Device](#).

7.58.2.5 ReadingSerialRequiresReset()

```
virtual bool SerialNumberedDevice::ReadingSerialRequiresReset ( ) [pure virtual]
```

True if reading this serial number requires a device reset.

Applications may choose not to display the serial number to avoid disrupting the running code.

Implemented in [STM32Device](#), and [XilinxFPGA](#).

The documentation for this class was generated from the following files:

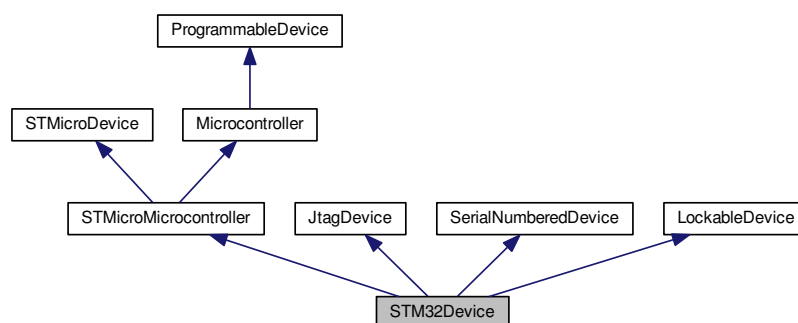
- [SerialNumberedDevice.h](#)
- [SerialNumberedDevice.cpp](#)

7.59 STM32Device Class Reference

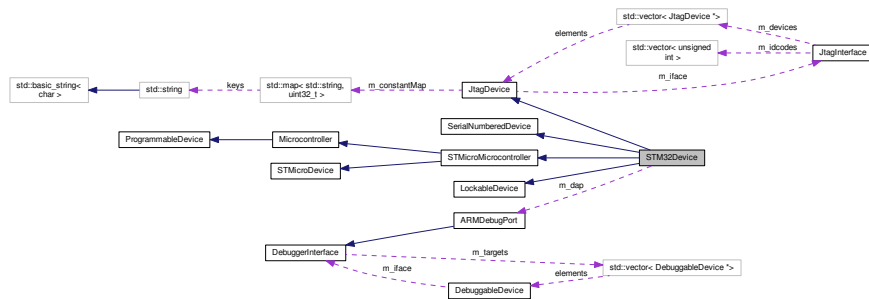
A STM32 microcontroller.

```
#include <STM32Device.h>
```

Inheritance diagram for STM32Device:



Collaboration diagram for STM32Device:



Public Types

- enum **FlashStrOffsets** {
FLASH_KEYR = 0x04, **FLASH_OPTKEYR** = 0x08, **FLASH_SR** = 0x0c, **FLASH_CR** = 0x10,
FLASH_OPTCR = 0x14 }

Public Member Functions

- STM32Device** (unsigned int devid, unsigned int stepping, unsigned int idcode, [JtagInterface](#) *iface, size_t pos)
- virtual [~STM32Device](#) ()
Destructor.
- virtual void [PostInitProbes](#) (bool quiet)
Does a post-initialization probe of the device to read debug ROMs etc.
- virtual std::string [GetDescription](#) ()
Gets a human-readable description of this device.
- virtual bool [IsProgrammed](#) ()
Determines if this device is programmed or blank.
- virtual void [Erase](#) ()
Erases the device configuration and restores the device to a blank state.
- virtual void [Program](#) ([FirmwareImage](#) *image)
Programs a firmware image to the device.
- virtual [FirmwareImage](#) * [LoadFirmwareImage](#) (const unsigned char *data, size_t len)
Loads the firmware.
- virtual bool [ReadingSerialRequiresReset](#) ()
True if reading this serial number requires a device reset.
- virtual int [GetSerialNumberLength](#) ()
Gets the length of the device's unique serial number, in bytes (rounded up to the nearest whole byte).
- virtual int [GetSerialNumberLengthBits](#) ()
Gets the length of the device's unique serial number, in bits.
- virtual void [GetSerialNumber](#) (unsigned char *data)
Gets the device's unique serial number.
- virtual std::string [GetPrettyPrintedSerialNumber](#) ()
Returns a pretty-printed serial number.
- virtual void [ProbeLocksNondestructive](#) ()
Queries lock status in a non-destructive fashion (contents of the chip are untouched)

- virtual void [ProbeLocksDestructive](#) ()
Queries lock status in a more invasive fashion. Gives more accurate data but may involve write transactions to memory.
- virtual [UncertainBoolean CheckMemoryAccess](#) (uint32_t ptr, unsigned int access)
Checks if a given physical address has a given protection applied.
- virtual [UncertainBoolean IsDeviceReadLocked](#) ()
Checks if the device is globally read protected or not.
- virtual void [SetReadLock](#) ()
Sets a global read-protection lock on the entire device.
- virtual void [ClearReadLock](#) ()
Clears the global read-protection lock, if set in a non-permanent fashion.
- virtual void [PrintLockProbeDetails](#) ()
Prints detailed information regarding the state of the read lock.
- int [GetProtectionLevel](#) ()
- [ARMv7MPProcessor](#) * [GetCPU](#) ()
- void [SetIR](#) (unsigned char irval)

Static Public Member Functions

- static [JtagDevice](#) * [CreateDevice](#) (unsigned int devid, unsigned int stepping, unsigned int idcode, [JtagInterface](#) *iface, size_t pos)

Protected Member Functions

- void [UnlockFlash](#) ()
- void [PollUntilFlashNotBusy](#) ()
- bool [BlankCheck](#) ()
- void [UnlockFlashOptions](#) ()

Protected Attributes

- [ARMDebugPort](#) * [m_dap](#)
- unsigned int [m_deviceID](#)
- unsigned int [m_flashKB](#)
- unsigned int [m_ramKB](#)
- uint32_t [m_waferX](#)
- uint32_t [m_waferY](#)
- int [m_waferNum](#)
- char [m_waferLot](#) [8]
- uint8_t [m_serialRaw](#) [12]
- uint32_t [m_flashSfrBase](#)
- uint32_t [m_flashMemoryBase](#)
- uint32_t [m_sramMemoryBase](#)
- bool [m_locksProbed](#)
- int [m_protectionLevel](#)

7.59.1 Detailed Description

A STM32 microcontroller.

7.59.2 Member Function Documentation

7.59.2.1 ClearReadLock()

```
void STM32Device::ClearReadLock ( ) [virtual]
```

Clears the global read-protection lock, if set in a non-permanent fashion.

In most parts, this will trigger a bulk flash erase.

Implements [LockableDevice](#).

7.59.2.2 Erase()

```
void STM32Device::Erase ( ) [virtual]
```

Erases the device configuration and restores the device to a blank state.

After this function is called, regardless of success or failure, all existing connections to on-chip code become invalid.

Exceptions

JtagException	if the erase operation fails
-------------------------------	------------------------------

Implements [ProgrammableDevice](#).

7.59.2.3 GetDescription()

```
string STM32Device::GetDescription ( ) [virtual]
```

Gets a human-readable description of this device.

Example: "Xilinx XC6SLX45 stepping 3"

Returns

Device description

Implements [JtagDevice](#).

7.59.2.4 GetPrettyPrintedSerialNumber()

```
string STM32Device::GetPrettyPrintedSerialNumber ( ) [virtual]
```

Returns a pretty-printed serial number.

Most serial numbers have no inherent meaning but some contain encoded lot numbers etc.

This function returns a human-readable version of the serial number, if such exists. Otherwise, a hex encoding of [GetSerialNumber\(\)](#) is returned.

Reimplemented from [SerialNumberedDevice](#).

7.59.2.5 GetSerialNumber()

```
void STM32Device::GetSerialNumber (
    unsigned char * data ) [virtual]
```

Gets the device's unique serial number.

Note that some architectures, such as Spartan-6, cannot read the serial number over JTAG without erasing the [FPGA](#) configuration. If this is the case, calling this function will automatically erase the [FPGA](#).

Call [ReadingSerialRequiresReset\(\)](#) to see if this is the case.

Exceptions

JtagException	if an error occurs during the read operation
-------------------------------	--

Parameters

<i>data</i>	Buffer to store the serial number into. Must be at least as large as the size given by GetSerialNumberLength() .
-------------	--

Implements [SerialNumberedDevice](#).

7.59.2.6 GetSerialNumberLength()

```
int STM32Device::GetSerialNumberLength ( ) [virtual]
```

Gets the length of the device's unique serial number, in bytes (rounded up to the nearest whole byte).

Returns

Serial number length

Implements [SerialNumberedDevice](#).

7.59.2.7 GetSerialNumberLengthBits()

```
int STM32Device::GetSerialNumberLengthBits ( ) [virtual]
```

Gets the length of the device's unique serial number, in bits.

Returns

Serial number length

Implements [SerialNumberedDevice](#).

7.59.2.8 IsProgrammed()

```
bool STM32Device::IsProgrammed ( ) [virtual]
```

Determines if this device is programmed or blank.

Returns

true if programmed, false if blank

Implements [ProgrammableDevice](#).

7.59.2.9 LoadFirmwareImage()

```
FirmwareImage * STM32Device::LoadFirmwareImage (
    const unsigned char * data,
    size_t len ) [virtual]
```

Loads the firmware.

For now assume it's a raw ROM image, no ELF etc supported

Reimplemented from [Microcontroller](#).

7.59.2.10 PostInitProbes()

```
void STM32Device::PostInitProbes (
    bool quiet ) [virtual]
```

Does a post-initialization probe of the device to read debug ROMs etc.

Parameters

<i>quiet</i>	Do minimal probing to avoid triggering security lockdowns
--------------	---

Implements [JtagDevice](#).

7.59.2.11 Program()

```
void STM32Device::Program (
    FirmwareImage * image ) [virtual]
```

Programs a firmware image to the device.

For now, assume we are blank when we start.

For now, assume the image is a flat binary to be burned to flash.

Implements [ProgrammableDevice](#).

7.59.2.12 ReadingSerialRequiresReset()

```
bool STM32Device::ReadingSerialRequiresReset ( ) [virtual]
```

True if reading this serial number requires a device reset.

Applications may choose not to display the serial number to avoid disrupting the running code.

Implements [SerialNumberedDevice](#).

7.59.2.13 SetReadLock()

```
void STM32Device::SetReadLock ( ) [virtual]
```

Sets a global read-protection lock on the entire device.

This function only performs reversible locks that can be cleared with a bulk erase. Thus, it should not be able to brick the chip entirely.

Implements [LockableDevice](#).

The documentation for this class was generated from the following files:

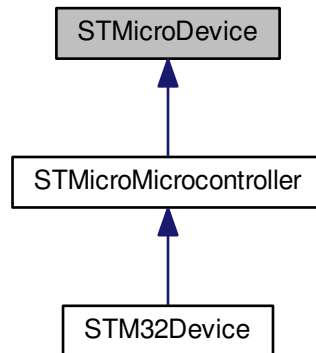
- [STM32Device.h](#)
- [STM32Device.cpp](#)

7.60 STMicroDevice Class Reference

Abstract base class for all STMicro devices.

```
#include <STMicroDevice.h>
```

Inheritance diagram for STMicroDevice:



Public Member Functions

- **STMicroDevice** (unsigned int devicetype, unsigned int stepping)
- virtual `~STMicroDevice` ()
Default virtual destructor.

Static Public Member Functions

- static `JtagDevice * CreateDevice` (unsigned int idcode, `JtagInterface *iface`, `size_t pos`)
Creates a `STMicroDevice` given an ID code.

Protected Attributes

- unsigned int **m_devicetype**
- unsigned int **m_stepping**

7.60.1 Detailed Description

Abstract base class for all STMicro devices.

7.60.2 Member Function Documentation

7.60.2.1 CreateDevice()

```
JtagDevice * STMicroDevice::CreateDevice (
    unsigned int idcode,
    JtagInterface * iface,
    size_t pos ) [static]
```

Creates a [STMicroDevice](#) given an ID code.

Exceptions

JtagException	if the ID code supplied is not a valid STMicro device, or not a known family number
-------------------------------	---

Parameters

<i>idcode</i>	The ID code of this device
<i>iface</i>	The JTAG adapter this device was discovered on
<i>pos</i>	Position in the chain that this device was discovered

Returns

A valid [JtagDevice](#) object, or NULL if the vendor ID was not recognized.

The documentation for this class was generated from the following files:

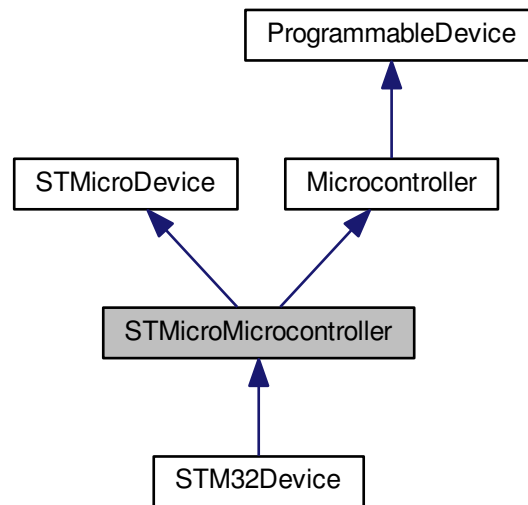
- [STMicroDevice.h](#)
- [STMicroDevice.cpp](#)

7.61 STMicroMicrocontroller Class Reference

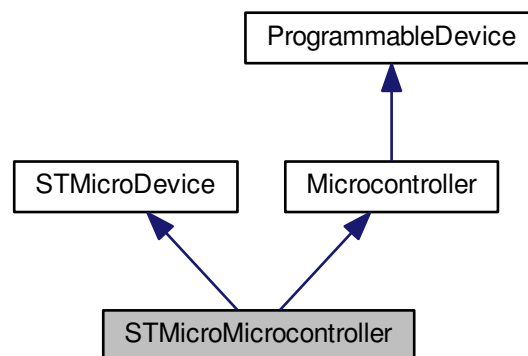
Generic base class for all STMicro MCUs.

```
#include <STMicroMicrocontroller.h>
```

Inheritance diagram for STMicroMicrocontroller:



Collaboration diagram for STMicroMicrocontroller:



Public Member Functions

- **STMicroMicrocontroller** (unsigned int devicetype, unsigned int stepping, unsigned int idcode, [JtagInterface](#) *iface, size_t pos)

Additional Inherited Members

7.61.1 Detailed Description

Generic base class for all STMicro MCUs.

The documentation for this class was generated from the following files:

- [STMicroMicrocontroller.h](#)
- [STMicroMicrocontroller.cpp](#)

7.62 UncertainBoolean Class Reference

A boolean value with an attached level of uncertainty.

```
#include <LockableDevice.h>
```

Public Types

- enum **CertaintyLevel** { **USELESS**, **INCONSISTENT**, **VERY_LIKELY**, **CERTAIN** }

Public Member Functions

- **UncertainBoolean** (bool b, CertaintyLevel level)
- CertaintyLevel **GetCertainty** ()
- bool **GetValue** ()
- const char * **GetCertaintyAsText** ()

Protected Attributes

- bool **m_value**
- CertaintyLevel **m_certainty**

7.62.1 Detailed Description

A boolean value with an attached level of uncertainty.

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

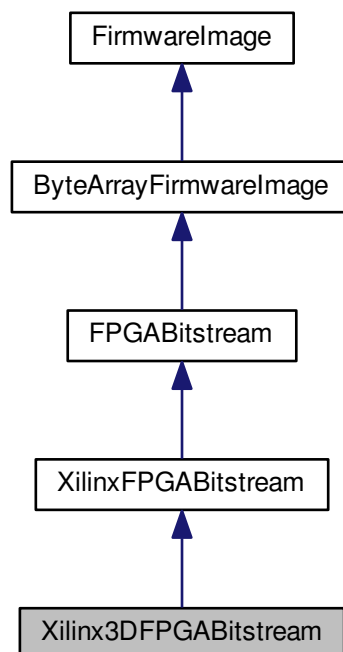
- [LockableDevice.h](#)

7.63 Xilinx3DFPGABitstream Class Reference

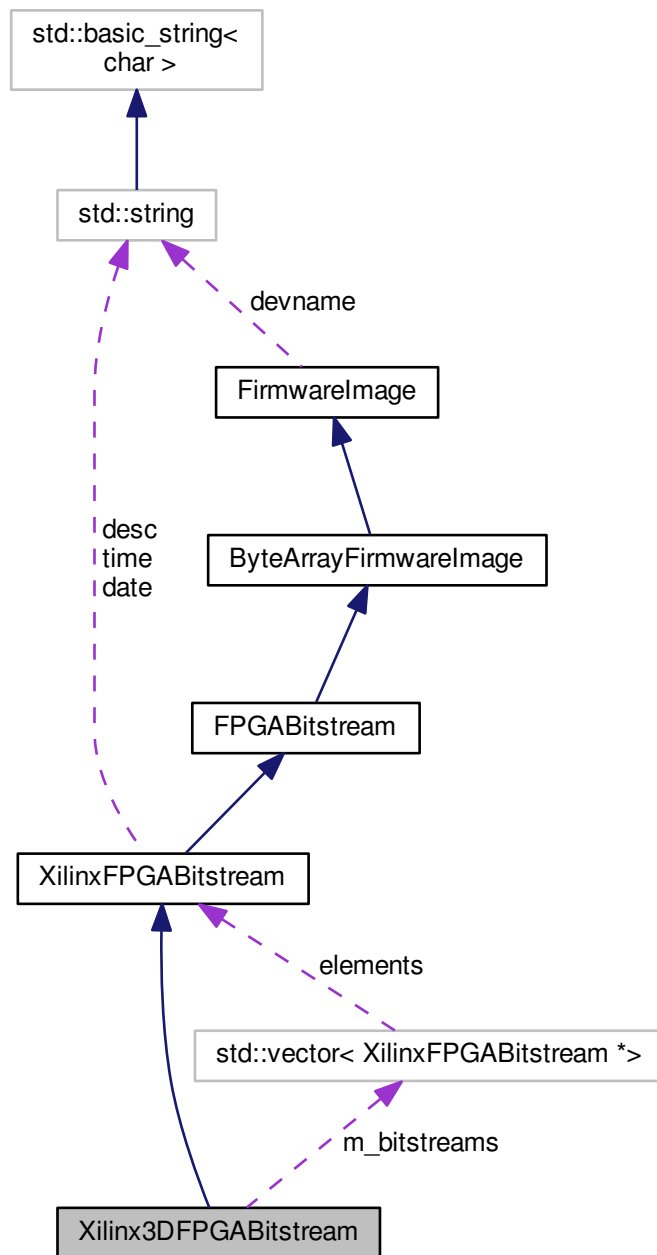
A bitstream for Xilinx 3D FPGAs (multiple dies on a passive interposer, each with their own bitstream)

```
#include <Xilinx3DFPGABitstream.h>
```

Inheritance diagram for Xilinx3DFPGABitstream:



Collaboration diagram for Xilinx3DFPGABitstream:



Public Member Functions

- [Xilinx3DFPGABitstream \(\)](#)
Initializes this object to empty.
- [virtual ~Xilinx3DFPGABitstream \(\)](#)
Free bitstream memory.
- [virtual std::string GetDescription \(\)](#)

Public Attributes

- `std::vector< XilinxFPGABitstream * > m_bitstreams`

7.63.1 Detailed Description

A bitstream for Xilinx 3D FPGAs (multiple dies on a passive interposer, each with their own bitstream)

The documentation for this class was generated from the following files:

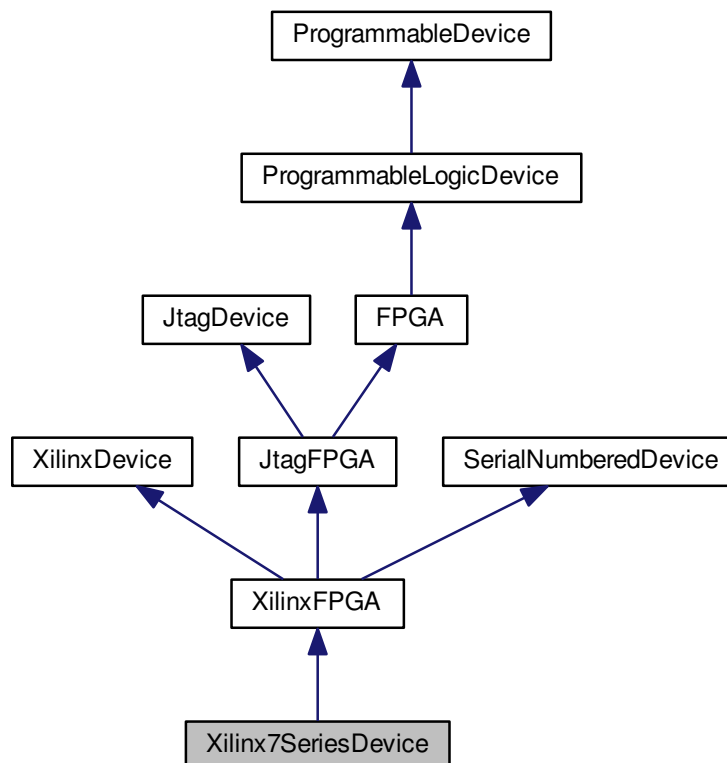
- [Xilinx3DFPGABitstream.h](#)
- [Xilinx3DFPGABitstream.cpp](#)

7.64 Xilinx7SeriesDevice Class Reference

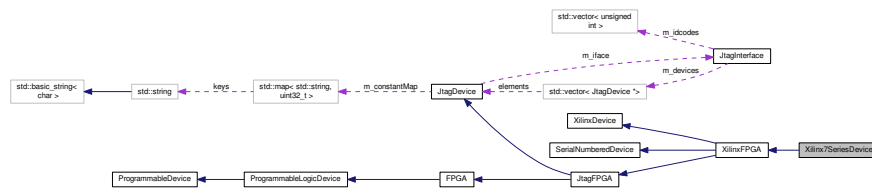
A Xilinx 7-series [FPGA](#) device.

```
#include <Xilinx7SeriesDevice.h>
```

Inheritance diagram for Xilinx7SeriesDevice:



Collaboration diagram for Xilinx7SeriesDevice:



Public Types

- enum [deviceids](#) {
SPARTAN7_6 = 0x022, **SPARTAN7_15** = 0x020, **SPARTAN7_25** = 0x1c4, **SPARTAN7_50** = 0x02f,
SPARTAN7_75 = 0x1c8, **SPARTAN7_100** = 0x1c7, **ARTIX7_12T** = 0x1c3, **ARTIX7_15T** = 0x02e,
ARTIX7_25T = 0x1c2, **ARTIX7_35T** = 0x02d, **ARTIX7_50T** = 0x02c, **ARTIX7_75T** = 0x032,
ARTIX7_100T = 0x031, **ARTIX7_200T** = 0x036, **KINTEX7_70T** = 0x047, **KINTEX7_160T** = 0x04c,
ZYNQ_010 = 0x122 }
JTAG device IDs.
- enum [instructions](#) {
INST_USER1 = 0x02, **INST_USER2** = 0x03, **INST_USERS3** = 0x22, **INST_USER4** = 0x23,
INST_CFG_OUT = 0x04, **INST_CFG_IN** = 0x05, **INST_USERCODE** = 0x08, **INST_IDCODE** = 0x09,
INST_JPROGRAM = 0x0B, **INST_JSTART** = 0x0C, **INST_JSHUTDOWN** = 0x0D, **INST_ISC_ENABLE** =
0x10,
INST_ISC_DISABLE = 0x16, **INST_XSC_DNA** = 0x17, **INST_XADC_DRP** = 0x37, **INST_BYPASS** = 0x3F }
6-bit-wide JTAG instructions (see BSDL file). Mostly, but not entirely, same as Spartan-6.

Public Member Functions

- [Xilinx7SeriesDevice](#) (unsigned int arraysize, unsigned int rev, unsigned int idcode, [JtagInterface](#) *iface, size_t pos)
Initializes this device.
- virtual [~Xilinx7SeriesDevice](#) ()
Empty virtual destructor.
- virtual std::string [GetDescription](#) ()
Gets a human-readable description of this device.
- virtual void [PrintStatusRegister](#) ()
- virtual bool [IsProgrammed](#) ()
Determines if this device is programmed or blank.
- virtual int [GetSerialNumberLength](#) ()
Gets the length of the device's unique serial number, in bytes (rounded up to the nearest whole byte).
- virtual int [GetSerialNumberLengthBits](#) ()
Gets the length of the device's unique serial number, in bits.
- virtual void [GetSerialNumber](#) (unsigned char *data)
Gets the device's unique serial number.
- virtual size_t [GetNumUserInstructions](#) ()
Get the number of JTAG instructions which are routed to [FPGA](#) fabric.
- virtual void [SelectUserInstruction](#) (size_t index)
Sets the instruction register to the specified user instruction.
- virtual void [Erase](#) ()

- virtual void **InternalErase** ()
Erases the device configuration and restores the device to a blank state.
- virtual **FirmwareImage** * **LoadFirmwareImage** (const unsigned char *data, size_t len)
Parses an in-memory image of a firmware image into a format suitable for loading into the device.
- virtual void **Program** (**FirmwareImage** *image)
Loads a new firmware image onto the device.
- virtual void **Reboot** ()
*Reboots the **FPGA** and loads from external memory, if possible.*

Static Public Member Functions

- static **JtagDevice** * **CreateDevice** (unsigned int arraysize, unsigned int rev, unsigned int idcode, **JtagInterface** *iface, size_t pos)
Factory method.

Public Attributes

- enum **Xilinx7SeriesDevice::deviceids** **__attribute__**

Protected Types

- enum **x7_config_opcodes** { **X7_CONFIG_OP_NOP** = 0, **X7_CONFIG_OP_READ** = 1, **X7_CONFIG_OP_WRITE** = 2 }
7-series configuration opcodes (see UG470 page 87). Same as for Spartan-6.
- enum **x7_config_frame_types** { **X7_CONFIG_FRAME_TYPE_1** = 1, **X7_CONFIG_FRAME_TYPE_2** = 2 }
7-series configuration frame types (see UG470 page 87). Same as for Spartan-6.
- enum **x7_config_regs** {
CONFIG_CRC = 0x00, **CONFIG_FAR** = 0x01, **CONFIG_FDRI** = 0x02, **CONFIG_FDRO** = 0x03,
CONFIG_CMD = 0x04, **CONFIG_CTL0** = 0x05, **CONFIG_MASK** = 0x06, **CONFIG_STAT** = 0x07,
CONFIG_LOUT = 0x08, **CONFIG_COR0** = 0x09, **CONFIG_MFWR** = 0x0A, **CONFIG_CBC** = 0x0B,
CONFIG_IDCODE = 0x0C, **CONFIG_AXSS** = 0x0D, **CONFIG_COR1** = 0x0E, **CONFIG_WBSTAR** = 0x10,
CONFIG_TIMER = 0x11, **CONFIG_BOOTSTS** = 0x16, **CONFIG_CTL1** = 0x18, **CONFIG_BSPI** = 0x1F,
CONFIG_MAX }
7-series configuration registers (see UG470 page 104). Not same as Spartan-6.
- enum **x7_cmd_values** {
X7_CMD_NULL = 0x00, **X7_CMD_WCFG** = 0x01, **X7_CMD_MFW** = 0x02, **X7_CMD_LFRM** = 0x03,
X7_CMD_RCFG = 0x04, **X7_CMD_START** = 0x05, **X7_CMD_RCAP** = 0x06, **X7_CMD_RCRC** = 0x07,
X7_CMD_AGHIGH = 0x08, **X7_CMD_SWITCH** = 0x09, **X7_CMD_GRESTORE** = 0x0a, **X7_CMD_SHUTDOWN** = 0x0b,
X7_CMD_GCAPTURE = 0x0c, **X7_CMD_DESYNC** = 0x0d, **X7_CMD_IPROG** = 0x0f, **X7_CMD_CRCC** = 0x10,
X7_CMD_LTIMER = 0x11, **X7_CMD_MAX** }
7-series CMD register values (see UG470 page 89-90)

Protected Member Functions

- uint32_t **ReadWordConfigRegister** (unsigned int reg)
Reads a single 32-bit word from a config register.
- void **ReadWordsConfigRegister** (unsigned int reg, uint32_t *dout, unsigned int count)
- void **WriteWordConfigRegister** (unsigned int reg, uint32_t value)
- virtual void **ParseBitstreamInternals** (const unsigned char *data, size_t len, **XilinxFPGABitstream** *bitstream, size_t fpos)
Reads several 16-bit words from a config register.
- void **SetIR** (unsigned char irval)
- void **SetIRDeferred** (unsigned char irval)

Protected Attributes

- unsigned `m_arraysize`
Array size (the specific 7-series device we are)
- unsigned int `m_rev`
Stepping number.

7.64.1 Detailed Description

A Xilinx 7-series [FPGA](#) device.

7.64.2 Member Enumeration Documentation

7.64.2.1 instructions

```
enum Xilinx7SeriesDevice::instructions
```

6-bit-wide JTAG instructions (see BSDL file). Mostly, but not entirely, same as Spartan-6.

Enumerator

INST_USER1	User-defined instruction 1.
INST_USER2	User-defined instruction 2.
INST_USER3	User-defined instruction 3 Not same as Spartan-6
INST_USER4	User-defined instruction 4 Not same as Spartan-6
INST_CFG_OUT	Read configuration register.
INST_CFG_IN	Write configuration register.
INST_USERCODE	Read user ID code.
INST_IDCODE	Read ID code.
INST_JPROGRAM	Enters programming mode (erases FPGA configuration)
INST_JSTART	Runs the FPGA startup sequence (must supply dummy clocks after)
INST_JSHUTDOWN	Runs the FPGA shutdown sequence (must supply dummy clocks after)
INST_ISC_ENABLE	Enters In-System Configuration mode (must load INST_JPROGRAM before)
INST_ISC_DISABLE	Leaves In-System Configuration mode.
INST_XSC_DNA	Read device DNA (must load INST_ISC_ENABLE before and INST_ISC_DISABLE after) Not same as Spartan-6
INST_XADC_DRP	Access to the ADC Not present in Spartan-6
INST_BYPASS	Standard JTAG bypass.

7.64.3 Constructor & Destructor Documentation

7.64.3.1 Xilinx7SeriesDevice()

```
Xilinx7SeriesDevice::Xilinx7SeriesDevice (
    unsigned int arraysize,
    unsigned int rev,
    unsigned int idcode,
    JtagInterface * iface,
    size_t pos )
```

Initializes this device.

Parameters

<i>arraysize</i>	Array size from JTAG ID code
<i>rev</i>	Revision number from JTAG ID code
<i>idcode</i>	The ID code of this device
<i>iface</i>	The JTAG adapter this device was discovered on
<i>pos</i>	Position in the chain that this device was discovered

7.64.4 Member Function Documentation

7.64.4.1 Erase()

```
void Xilinx7SeriesDevice::Erase ( ) [virtual]
```

Erases the device configuration and restores the device to a blank state.

After this function is called, regardless of success or failure, all existing connections to on-chip code become invalid.

Exceptions

<i>JtagException</i>	if the erase operation fails
--------------------------------------	------------------------------

Implements [ProgrammableDevice](#).

7.64.4.2 GetDescription()

```
string Xilinx7SeriesDevice::GetDescription ( ) [virtual]
```

Gets a human-readable description of this device.

Example: "Xilinx XC6SLX45 stepping 3"

Returns

Device description

Implements [JtagDevice](#).

7.64.4.3 GetSerialNumber()

```
void Xilinx7SeriesDevice::GetSerialNumber (
    unsigned char * data ) [virtual]
```

Gets the device's unique serial number.

Note that some architectures, such as Spartan-6, cannot read the serial number over JTAG without erasing the [FPGA](#) configuration. If this is the case, calling this function will automatically erase the [FPGA](#).

Call [ReadingSerialRequiresReset\(\)](#) to see if this is the case.

Exceptions

JtagException	if an error occurs during the read operation
-------------------------------	--

Parameters

<i>data</i>	Buffer to store the serial number into. Must be at least as large as the size given by GetSerialNumberLength() .
-------------	--

Implements [SerialNumberedDevice](#).

7.64.4.4 GetSerialNumberLength()

```
int Xilinx7SeriesDevice::GetSerialNumberLength ( ) [virtual]
```

Gets the length of the device's unique serial number, in bytes (rounded up to the nearest whole byte).

Returns

Serial number length

Implements [SerialNumberedDevice](#).

7.64.4.5 GetSerialNumberLengthBits()

```
int Xilinx7SeriesDevice::GetSerialNumberLengthBits ( ) [virtual]
```

Gets the length of the device's unique serial number, in bits.

Returns

Serial number length

Implements [SerialNumberedDevice](#).

7.64.4.6 IsProgrammed()

```
bool Xilinx7SeriesDevice::IsProgrammed ( ) [virtual]
```

Determines if this device is programmed or blank.

Returns

true if programmed, false if blank

Implements [ProgrammableDevice](#).

7.64.4.7 LoadFirmwareImage()

```
FirmwareImage * Xilinx7SeriesDevice::LoadFirmwareImage (
    const unsigned char * data,
    size_t len ) [virtual]
```

Parses an in-memory image of a firmware image into a format suitable for loading into the device.

Exceptions

JtagException	if the image is malformed
-------------------------------	---------------------------

Parameters

<i>data</i>	Pointer to the start of the firmware image, including headers
<i>len</i>	Length of the firmware image

Returns

Pointer to an [FirmwareImage](#) object suitable for passing to [Configure\(\)](#).

Implements [ProgrammableDevice](#).

7.64.4.8 ParseBitstreamInternals()

```
void Xilinx7SeriesDevice::ParseBitstreamInternals (
    const unsigned char * data,
    size_t len,
    XilinxFPGABitstream * bitstream,
    size_t fpos ) [protected], [virtual]
```

Reads several 16-bit words from a config register.

The current implementation uses type 1 packets and is thus limited to reading less than 32 words.

Exceptions

JtagException	if the read fails
-------------------------------	-------------------

Parameters

<i>reg</i>	The configuration register to read
<i>dout</i>	Buffer to read into
<i>count</i>	Number of 16-bit words to read

Implements [XilinxFPGA](#).

7.64.4.9 Program()

```
void Xilinx7SeriesDevice::Program (
    FirmwareImage * image ) [virtual]
```

Loads a new firmware image onto the device.

After this function is called, regardless of success or failure, all existing connections to on-chip code become invalid.

Exceptions

JtagException	if the erase operation fails
-------------------------------	------------------------------

Parameters

<i>image</i>	The parsed image to load
--------------	--------------------------

Implements [ProgrammableDevice](#).

7.64.4.10 ReadWordConfigRegister()

```
uint32_t Xilinx7SeriesDevice::ReadWordConfigRegister (
    unsigned int reg ) [protected], [virtual]
```

Reads a single 32-bit word from a config register.

Reference: UG470 page 87

Note that 7-series devices expect data clocked in MSB first but the JTAG API clocks data LSB first. Some swapping is required as a result.

Clock data into CFG_IN register Synchronization word Nop Read STAT register Two dummy words to flush packet buffer

Read from CFG_OUT register

Exceptions

JtagException	if the read fails
-------------------------------	-------------------

Parameters

<i>reg</i>	The configuration register to read
------------	------------------------------------

Implements [XilinxFPGA](#).

The documentation for this class was generated from the following files:

- [Xilinx7SeriesDevice.h](#)
- [Xilinx7SeriesDevice.cpp](#)

7.65 Xilinx7SeriesDeviceConfigurationFrame Union Reference

7-series configuration frame (see UG470 page 87)

```
#include <Xilinx7SeriesDevice.h>
```

Public Member Functions

- ```
struct {
 unsigned int count:11
 Count field.
 unsigned int reserved:2
 Reserved, must be zero.
 unsigned int reg_addr:14
 Register address.
 unsigned int op:2
 Opcode.
 unsigned int type:3
 Frame type.
} __attribute__((packed)) bits
```
- ```
struct {
    unsigned int count:27
        Count field.
    unsigned int op:2
        Opcode.
    unsigned int type:3
        Frame type.
} __attribute__((packed)) bits_type2
```

Public Attributes

- `uint32_t word`
The raw configuration word.

7.65.1 Detailed Description

7-series configuration frame (see UG470 page 87)

7.65.2 Member Data Documentation

7.65.2.1 `op`

```
unsigned int Xilinx7SeriesDeviceConfigurationFrame::op
```

Opcode.

Must be one of the following:

- `Xilinx7SeriesDevice::X7_CONFIG_OP_NOP`
- `Xilinx7SeriesDevice::X7_CONFIG_OP_READ`
- `Xilinx7SeriesDevice::X7_CONFIG_OP_WRITE`

Must be zero

7.65.2.2 `type`

```
unsigned int Xilinx7SeriesDeviceConfigurationFrame::type
```

Frame type.

Must be `Xilinx7SeriesDevice::X7_CONFIG_FRAME_TYPE_1`

Must be `Xilinx7SeriesDevice::X7_CONFIG_FRAME_TYPE_2`

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

- [Xilinx7SeriesDevice.h](#)

7.66 Xilinx7SeriesDeviceStatusRegister Union Reference

7-series status register (see UG470 table 5-28)

```
#include <Xilinx7SeriesDevice.h>
```


Public Member Functions

- ```

struct {
 unsigned int crc_err:1
 Indicates that the device failed to configure due to a CRC error.
 unsigned int part_secured:1
 Indicates that the device is in secure mode (encrypted bitstream)
 unsigned int mmcm_lock:1
 Indicates MMCMs are locked.
 unsigned int dci_match:1
 Indicates DCI is matched.
 unsigned int eos:1
 End-of-Startup signal.
 unsigned int gts_cfg_b:1
 Status of GTS_CFG net.
 unsigned int gwe:1
 Status of GWE net.
 unsigned int ghigh_b:1
 Status of GHIGH_B net.
 unsigned int mode_pins:3
 Status of mode pins.
 unsigned int init_complete:1
 Internal init-finished signal.
 unsigned int init_b:1
 Status of INIT_B pin.
 unsigned int release_done:1
 Indicates DONE was released.
 unsigned int done:1
 Actual value on DONE pin.
 unsigned int id_error:1
 Indicates an ID code error occurred (write with wrong bitstream)
 unsigned int dec_error:1
 Decryption error.
 unsigned int xadc_over_temp:1
 Indicates board is too hot.
 unsigned int startup_state:3
 Status of startup state machine.
 unsigned int reserved_1:4
 Reserved.
 unsigned int bus_width:2
 Config bus width (see table 5-26)
 unsigned int reserved_2:5
 Reserved.
} __attribute__((packed)) bits

```

## Public Attributes

- `uint32_t` [word](#)  
*The raw status register value.*

### 7.66.1 Detailed Description

7-series status register (see UG470 table 5-28)

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

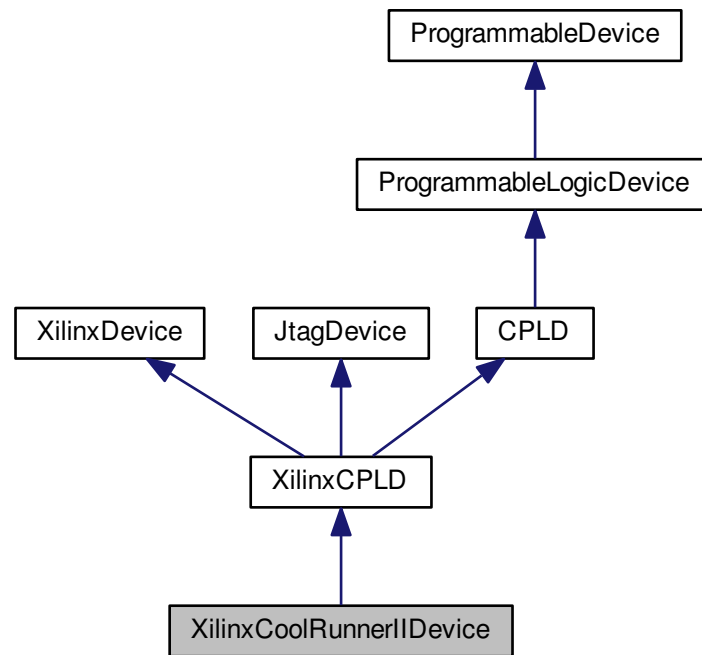
- [Xilinx7SeriesDevice.h](#)

## 7.67 XilinxCoolRunnerIIDevice Class Reference

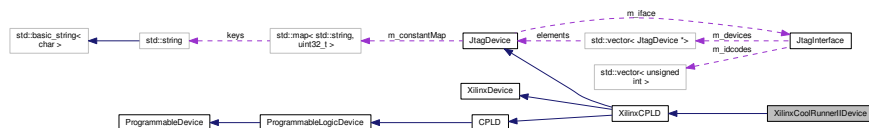
A Xilinx CoolRunner-II device.

```
#include <XilinxCoolRunnerIIDevice.h>
```

Inheritance diagram for XilinxCoolRunnerIIDevice:



Collaboration diagram for XilinxCoolRunnerIIDevice:



### Public Types

- enum `deviceids` {
  - `XC2C32` = 0x01, `XC2C32A` = 0x21, `XC2C64` = 0x05, `XC2C64A` = 0x25,
  - `XC2C128` = 0x18, `XC2C256` = 0x14, `XC2C384` = 0x15, `XC2C512` = 0x17 }

*JTAG device IDs.*

- enum `packages` {  
`QFG32 = 1, VQG44 = 2, QFG48 = 3, CPG56 = 4,`  
`VQG100 = 5, CPG132 = 6, TQG144 = 7, PQG208 = 8,`  
`FTG256 = 9, FGG324 = 10` }
- enum `instructions` {  
`INST_IDCODE = 0x01, INST_BYPASS = 0xFF, INST_ISC_ENABLE = 0xE8, INST_ISC_ENABLEOTF = 0xE4,`  
`INST_ISC_SRAM_READ = 0xE7, INST_ISC_SRAM_WRITE = 0xE6, INST_ISC_ERASE = 0xED, INST_ISC_PROGRAM = 0xEA,`  
`INST_ISC_INIT = 0xF0, INST_ISC_DISABLE = 0xC0, INST_ISC_READ = 0xEE` }  
*6-bit-wide JTAG instructions (from BSDL file)*
- enum `fusevalues` { `FUSE_VALUE_TRANSFER = -1, FUSE_VALUE_DONTCARE = -2` }

## Public Member Functions

- **XilinxCoolRunnerIIDevice** (unsigned int devid, unsigned int package\_decoded, unsigned int stepping, unsigned int idcode, `JtagInterface` \*iface, size\_t pos)
- virtual `~XilinxCoolRunnerIIDevice` ()  
*Destructor.*
- virtual void `PostInitProbes` (bool quiet)  
*Does a post-initialization probe of the device to read debug ROMs etc.*
- virtual std::string `GetDescription` ()  
*Gets a human-readable description of this device.*
- std::string `GetDeviceName` ()  
*Returns the device name.*
- std::string `GetDevicePackage` ()  
*Returns the device package.*
- virtual bool `IsProgrammed` ()  
*Determines if this device is programmed or blank.*
- virtual void `Erase` ()  
*Erases the device configuration and restores the device to a blank state.*
- virtual `FirmwareImage` \* `LoadFirmwareImage` (const unsigned char \*data, size\_t len)  
*Parses an in-memory image of a firmware image into a format suitable for loading into the device.*
- virtual void `Program` (`FirmwareImage` \*image)  
*Loads a new firmware image onto the device.*
- `XilinxCoolRunnerIIDeviceStatusRegister` `GetStatusRegister` ()  
*Returns the device status register.*
- void `SetIR` (unsigned char irval)
- int `GetShiftRegisterWidth` ()  
*Gets the width of the shift register for this device.*
- int `GetShiftRegisterDepth` ()  
*Gets the depth of the shift register for this device.*
- int `GetFuseCount` ()  
*Gets the number of fuses in the device.*
- int `GetAddressSize` ()  
*Gets the number of address bits.*
- int `GetPaddingSize` ()  
*Gets the number of padding bits to add.*
- unsigned char \* `GeneratePermutedFuseData` (`XilinxCPLDBitstream` \*bit, int \*permtable)  
*Permutes the bitstream and adds transfer bits if the device needs them.*
- unsigned char \* `GenerateVerificationTable` ()

- Generates the verification table.*

  - int \* [GeneratePermutationTable](#) ()

*Generates the permutation table.*
- int [GetZIAWidth](#) ()

*Returns the width of one function block's ZIA in bits.*
- int [GetFunctionBlockCount](#) ()

*Returns the number of function blocks in the device.*
- int [GetFunctionBlockPairCount](#) ()

*Returns the number of function block pairs in the device.*
- int [GetFunctionBlockGridWidth](#) ()

*Returns the width of the FB grid, in FB pairs.*
- int [GetFunctionBlockGridHeight](#) ()

*Returns the height of the FB grid, in FBs.*
- int [MirrorCoordinate](#) (int x, int end, bool mirror)

*Mirrors a coordinate within a certain range.*
- int [GrayEncode](#) (int address)

*Gray code encoder.*
- unsigned int [GetDensity](#) ()

### Static Public Member Functions

- static [JtagDevice](#) \* [CreateDevice](#) (unsigned int idcode, [JtagInterface](#) \*iface, size\_t pos)
  - static std::string [GetPackageName](#) (int pknum)
- Gets the name of a given package enum.*

### Public Attributes

- enum [XilinxCoolRunnerIIDevice::deviceids](#) `__attribute__`

### Protected Attributes

- unsigned int [m\\_devid](#)
- Device ID code.*
- unsigned int [m\\_package](#)
- Package code.*
- unsigned int [m\\_stepping](#)
- Stepping number.*

### Additional Inherited Members

#### 7.67.1 Detailed Description

A Xilinx CoolRunner-II device.

#### 7.67.2 Member Enumeration Documentation

##### 7.67.2.1 instructions

enum [XilinxCoolRunnerIIDevice::instructions](#)

6-bit-wide JTAG instructions (from BSDL file)

## Enumerator

|                  |                                                                |
|------------------|----------------------------------------------------------------|
| INST_BYPASS      | Standard JTAG bypass.                                          |
| INST_ISC_ENABLE  | Enter in-system configuration mode.                            |
| INST_ISC_ERASE   | Erase the EEPROM array.                                        |
| INST_ISC_PROGRAM | Program the EEPROM array.                                      |
| INST_ISC_INIT    | Discharge high voltage and/or boot device (depends on context) |
| INST_ISC_DISABLE | Leave ISC mode.                                                |
| INST_ISC_READ    | Verify.                                                        |

## 7.67.2.2 packages

```
enum XilinxCoolRunnerIIDevice::packages
```

## Enumerator

|        |                              |
|--------|------------------------------|
| QFG32  | 32-pin QFN (0.5mm pitch)     |
| VQG44  | 44-pin VQFP (0.8mm pitch)    |
| QFG48  | 48-pin QFN (0.5mm pitch)     |
| CPG56  | 56-ball CSBGA (0.5mm pitch)  |
| VQG100 | 100-pin VQFP (0.5mm pitch)   |
| CPG132 | 132-ball CSBGA (0.5mm pitch) |
| TQG144 | 144-pin TQFP (0.5mm pitch)   |
| PQG208 | 208-pin PQFP (0.5mm pitch)   |
| FTG256 | 256-ball FTBGA (1mm pitch)   |
| FGG324 | 324-ball FGBGA (1mm pitch)   |

## 7.67.3 Member Function Documentation

## 7.67.3.1 Erase()

```
void XilinxCoolRunnerIIDevice::Erase () [virtual]
```

Erases the device configuration and restores the device to a blank state.

After this function is called, regardless of success or failure, all existing connections to on-chip code become invalid.

## Exceptions

|                                      |                              |
|--------------------------------------|------------------------------|
| <a href="#"><i>JtagException</i></a> | if the erase operation fails |
|--------------------------------------|------------------------------|

Implements [ProgrammableDevice](#).

#### 7.67.3.2 GeneratePermutationTable()

```
int * XilinxCoolRunnerIIDevice::GeneratePermutationTable ()
```

Generates the permutation table.

The table is generated in *row major* order to simplify the code, but for typical use should probably be column major.

#### 7.67.3.3 GetDescription()

```
std::string XilinxCoolRunnerIIDevice::GetDescription () [virtual]
```

Gets a human-readable description of this device.

Example: "Xilinx XC6SLX45 stepping 3"

#### Returns

Device description

Implements [JtagDevice](#).

#### 7.67.3.4 GetPaddingSize()

```
int XilinxCoolRunnerIIDevice::GetPaddingSize ()
```

Gets the number of padding bits to add.

See table 10 of programmer spec

#### 7.67.3.5 GetShiftRegisterDepth()

```
int XilinxCoolRunnerIIDevice::GetShiftRegisterDepth ()
```

Gets the depth of the shift register for this device.

Does not include sec/done or UES words.

#### 7.67.3.6 GetShiftRegisterWidth()

```
int XilinxCoolRunnerIIDevice::GetShiftRegisterWidth ()
```

Gets the width of the shift register for this device.

Includes transfer bits.

## 7.67.3.7 IsProgrammed()

```
bool XilinxCoolRunnerIIDevice::IsProgrammed () [virtual]
```

Determines if this device is programmed or blank.

## Returns

true if programmed, false if blank

Implements [ProgrammableDevice](#).

## 7.67.3.8 LoadFirmwareImage()

```
FirmwareImage * XilinxCoolRunnerIIDevice::LoadFirmwareImage (
 const unsigned char * data,
 size_t len) [virtual]
```

Parses an in-memory image of a firmware image into a format suitable for loading into the device.

## Exceptions

|                               |                           |
|-------------------------------|---------------------------|
| <a href="#">JtagException</a> | if the image is malformed |
|-------------------------------|---------------------------|

## Parameters

|             |                                                               |
|-------------|---------------------------------------------------------------|
| <i>data</i> | Pointer to the start of the firmware image, including headers |
| <i>len</i>  | Length of the firmware image                                  |

## Returns

Pointer to an [FirmwareImage](#) object suitable for passing to [Configure\(\)](#).

Implements [ProgrammableDevice](#).

## 7.67.3.9 PostInitProbes()

```
void XilinxCoolRunnerIIDevice::PostInitProbes (
 bool quiet) [virtual]
```

Does a post-initialization probe of the device to read debug ROMs etc.

## Parameters

|              |                                                           |
|--------------|-----------------------------------------------------------|
| <i>quiet</i> | Do minimal probing to avoid triggering security lockdowns |
|--------------|-----------------------------------------------------------|

Implements [JtagDevice](#).

### 7.67.3.10 Program()

```
void XilinxCoolRunnerIIDevice::Program (
 FirmwareImage * image) [virtual]
```

Loads a new firmware image onto the device.

After this function is called, regardless of success or failure, all existing connections to on-chip code become invalid.

#### Exceptions

|                               |                              |
|-------------------------------|------------------------------|
| <a href="#">JtagException</a> | if the erase operation fails |
|-------------------------------|------------------------------|

#### Parameters

|              |                          |
|--------------|--------------------------|
| <i>image</i> | The parsed image to load |
|--------------|--------------------------|

Implements [ProgrammableDevice](#).

The documentation for this class was generated from the following files:

- [XilinxCoolRunnerIIDevice.h](#)
- [XilinxCoolRunnerIIDevice.cpp](#)

## 7.68 XilinxCoolRunnerIIDeviceStatusRegister Union Reference

Status register for a Xilinx CoolRunner-II device.

```
#include <XilinxCoolRunnerIIDevice.h>
```

#### Public Member Functions

- ```
struct {
    unsigned int padding_one:2
        Constant '01'.
    unsigned int done:1
        True if configured.
    unsigned int sec:1
        True if secured.
    unsigned int isc_en:1
        True if in ISC_ENABLE state.
    unsigned int isc_dis:1
        True if in ISC_DISABLE state.
    unsigned int padding_zero:2
        Constant '00'.
} __attribute__((packed)) bits
```


Public Attributes

- `uint8_t word`
The raw status register value.

7.68.1 Detailed Description

Status register for a Xilinx CoolRunner-II device.

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

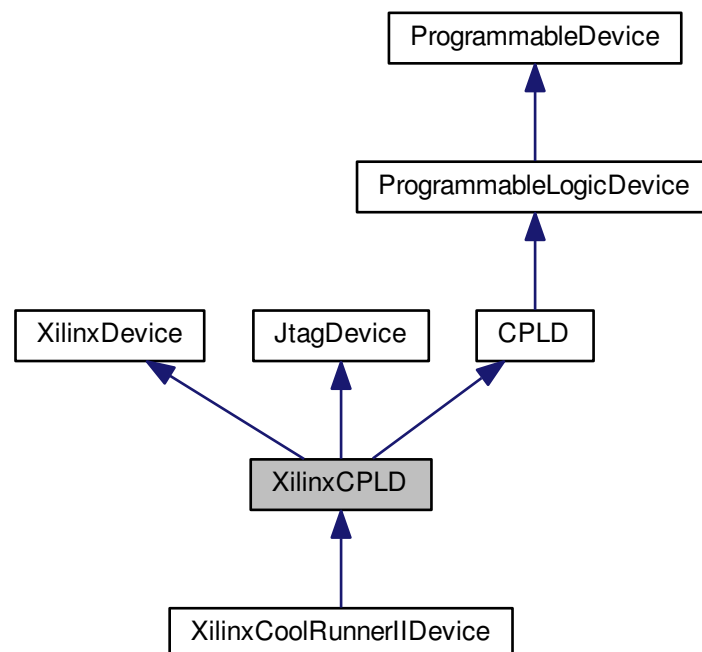
- [XilinxCoolRunnerIIDevice.h](#)

7.69 XilinxCPLD Class Reference

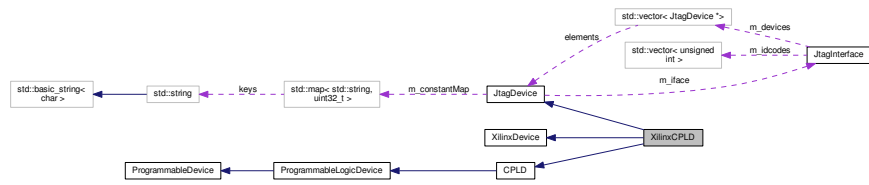
Generic base class for all Xilinx [CPLD](#) devices.

```
#include <XilinxCPLD.h>
```

Inheritance diagram for XilinxCPLD:



Collaboration diagram for XilinxCPLD:



Public Member Functions

- **XilinxCPLD** (unsigned int idcode, [JtagInterface](#) *iface, size_t pos, size_t irlength)

Additional Inherited Members

7.69.1 Detailed Description

Generic base class for all Xilinx [CPLD](#) devices.

The documentation for this class was generated from the following files:

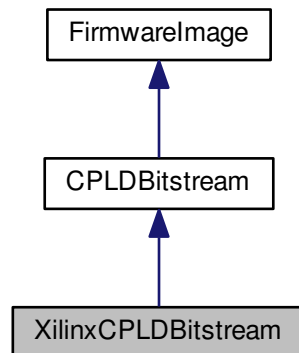
- [XilinxCPLD.h](#)
- [XilinxCPLD.cpp](#)

7.70 XilinxCPLDBitstream Class Reference

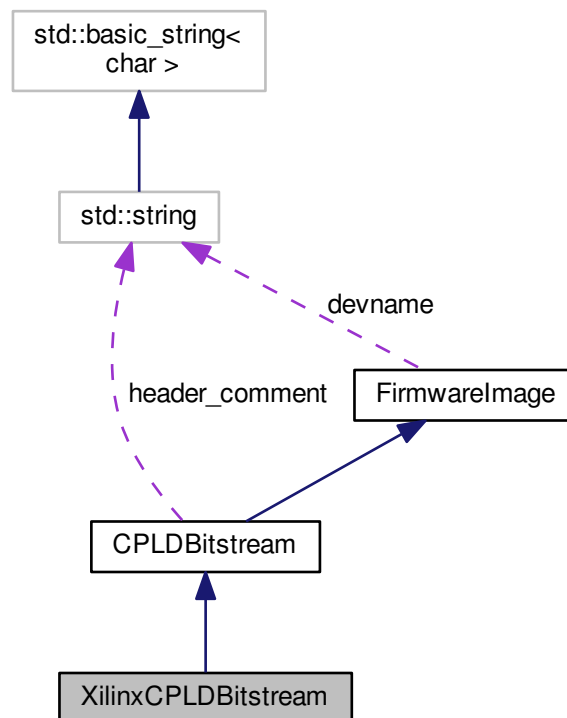
A bitstream for Xilinx CPLDs.

```
#include <XilinxCPLDBitstream.h>
```

Inheritance diagram for XilinxCPLDBitstream:



Collaboration diagram for XilinxCPLDBitstream:



Public Member Functions

- [XilinxCPLDBitstream \(\)](#)
Initializes this object to empty.
- [virtual ~XilinxCPLDBitstream \(\)](#)
Free bitstream memory.
- [virtual std::string GetDescription \(\)](#)

Additional Inherited Members

7.70.1 Detailed Description

A bitstream for Xilinx CPLDs.

The documentation for this class was generated from the following files:

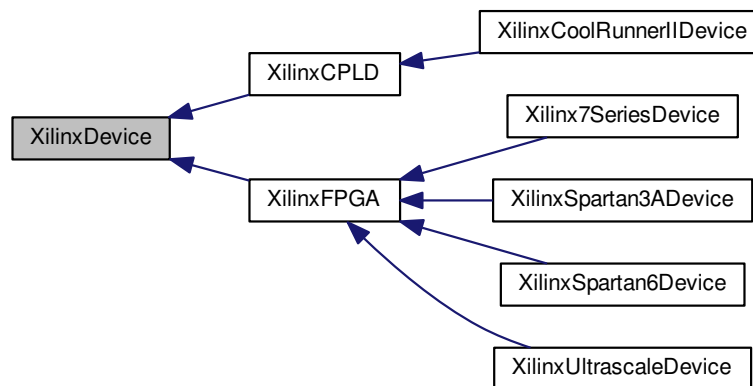
- [XilinxCPLDBitstream.h](#)
- [XilinxCPLDBitstream.cpp](#)

7.71 XilinxDevice Class Reference

Abstract base class for all Xilinx devices (FPGA, CPLD, flash, etc)

```
#include <XilinxDevice.h>
```

Inheritance diagram for XilinxDevice:



Public Member Functions

- virtual `~XilinxDevice()`
Default virtual destructor.

Static Public Member Functions

- static `JtagDevice * CreateDevice` (unsigned int idcode, `JtagInterface *iface`, size_t pos)
Creates a [XilinxDevice](#) given an ID code.

7.71.1 Detailed Description

Abstract base class for all Xilinx devices (FPGA, CPLD, flash, etc)

7.71.2 Member Function Documentation

7.71.2.1 CreateDevice()

```
JtagDevice * XilinxDevice::CreateDevice (
    unsigned int idcode,
    JtagInterface * iface,
    size_t pos ) [static]
```

Creates a [XilinxDevice](#) given an ID code.

Exceptions

JtagException	if the ID code supplied is not a valid Xilinx device, or not a known family number
-------------------------------	--

Parameters

<i>idcode</i>	The ID code of this device
<i>iface</i>	The JTAG adapter this device was discovered on
<i>pos</i>	Position in the chain that this device was discovered

Returns

A valid [JtagDevice](#) object, or NULL if the vendor ID was not recognized.

The documentation for this class was generated from the following files:

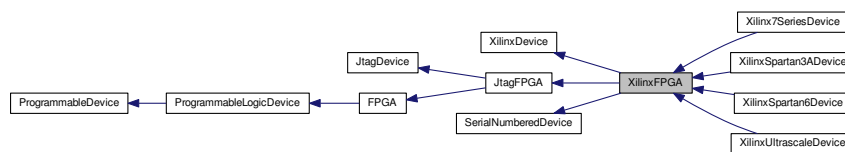
- [XilinxDevice.h](#)
- [XilinxDevice.cpp](#)

7.72 XilinxFPGA Class Reference

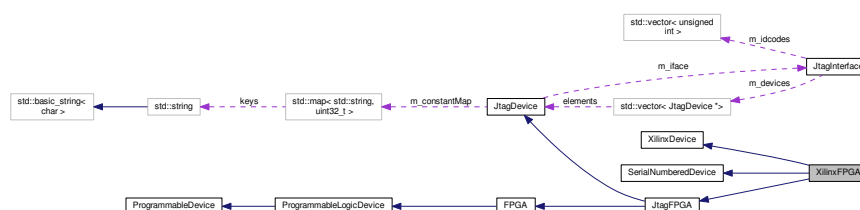
Abstract base class for all Xilinx FPGAs.

```
#include <XilinxFPGA.h>
```

Inheritance diagram for XilinxFPGA:



Collaboration diagram for XilinxFPGA:



Public Member Functions

- [XilinxFPGA](#) (unsigned int idcode, [JtagInterface](#) *iface, size_t pos, size_t irlength)
Initializes this device.
- virtual [~XilinxFPGA](#) ()
Default virtual destructor.
- virtual void [PostInitProbes](#) (bool quiet)
Does a post-initialization probe of the device to read debug ROMs etc.
- virtual uint32_t [ReadWordConfigRegister](#) (unsigned int reg)=0
- virtual void [PrintStatusRegister](#) ()=0
- virtual void [Reboot](#) ()=0
Reboots the [FPGA](#) and loads from external memory, if possible.

Protected Member Functions

- void [ParseBitstreamCore](#) ([XilinxFPGABitstream](#) *bitstream, const unsigned char *data, size_t len)
Parse a bitstream image (common to all Xilinx devices)
- virtual void [ParseBitstreamInternals](#) (const unsigned char *data, size_t len, [XilinxFPGABitstream](#) *bitstream, size_t fpos)=0
Parse a full bitstream image (specific to the derived [FPGA](#) family)
- virtual bool [ReadingSerialRequiresReset](#) ()
True if reading this serial number requires a device reset.

Additional Inherited Members

7.72.1 Detailed Description

Abstract base class for all Xilinx FPGAs.

7.72.2 Constructor & Destructor Documentation

7.72.2.1 XilinxFPGA()

```
XilinxFPGA::XilinxFPGA (
    unsigned int idcode,
    JtagInterface * iface,
    size_t pos,
    size_t irlength )
```

Initializes this device.

Parameters

<i>idcode</i>	The ID code of this device
<i>iface</i>	The JTAG adapter this device was discovered on
<i>pos</i>	Position in the chain that this device was discovered

7.72.3 Member Function Documentation

7.72.3.1 ParseBitstreamCore()

```
void XilinxFPGA::ParseBitstreamCore (
    XilinxFPGABitstream * bitstream,
    const unsigned char * data,
    size_t len ) [protected]
```

Parse a bitstream image (common to all Xilinx devices)

Exceptions

JtagException	if the bitstream is malformed or for the wrong device family
-------------------------------	--

Parameters

<i>bitstream</i>	The bitstream object being initialized
<i>data</i>	Pointer to the bitstream data
<i>len</i>	Length of the bitstream

Returns

A bitstream suitable for loading into this device

Bitstream format

13 unknown bytes (magic number?) 00 09 0f f0 0f f0 0f f0 0f f0 00 00 01 Records Record type (1 byte, lowercase letter) Null byte Record length (1 byte)

7.72.3.2 ParseBitstreamInternals()

```
virtual void XilinxFPGA::ParseBitstreamInternals (
    const unsigned char * data,
    size_t len,
    XilinxFPGABitstream * bitstream,
    size_t fpos ) [protected], [pure virtual]
```

Parse a full bitstream image (specific to the derived [FPGA](#) family)

Exceptions

JtagException	if the bitstream is malformed or for the wrong device family
-------------------------------	--

Parameters

<i>data</i>	Pointer to the bitstream data
-------------	-------------------------------

Parameters

<i>len</i>	Length of the bitstream
<i>bitstream</i>	The bitstream object to load into
<i>fpos</i>	Position in the bitstream image to start parsing (after the end of headers)
<i>bVerbose</i>	Set to true for verbose debug output on bitstream internals

Implemented in [Xilinx7SeriesDevice](#), [XilinxUltrascaleDevice](#), [XilinxSpartan6Device](#), and [XilinxSpartan3ADevice](#).

7.72.3.3 PostInitProbes()

```
void XilinxFPGA::PostInitProbes (
    bool quiet ) [virtual]
```

Does a post-initialization probe of the device to read debug ROMs etc.

Parameters

<i>quiet</i>	Do minimal probing to avoid triggering security lockdowns
--------------	---

Implements [JtagDevice](#).

7.72.3.4 ReadingSerialRequiresReset()

```
bool XilinxFPGA::ReadingSerialRequiresReset ( ) [protected], [virtual]
```

True if reading this serial number requires a device reset.

Applications may choose not to display the serial number to avoid disrupting the running code.

Implements [SerialNumberedDevice](#).

The documentation for this class was generated from the following files:

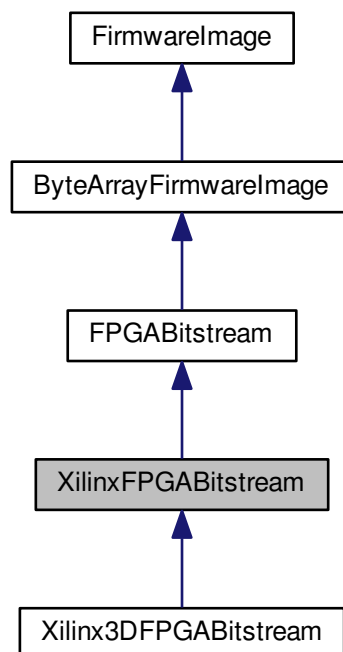
- [XilinxFPGA.h](#)
- [XilinxFPGA.cpp](#)

7.73 XilinxFPGABitstream Class Reference

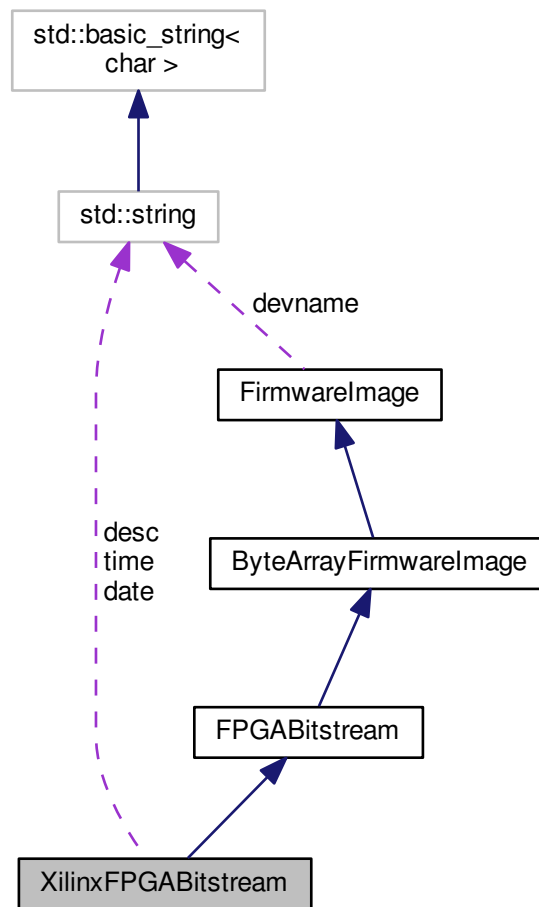
A bitstream for Xilinx FPGAs.

```
#include <XilinxFPGABitstream.h>
```

Inheritance diagram for XilinxFPGABitstream:



Collaboration diagram for XilinxFPGABitstream:



Public Member Functions

- [XilinxFPGABitstream \(\)](#)
Initializes this object to empty.
- [virtual ~XilinxFPGABitstream \(\)](#)
Free bitstream memory.
- [virtual std::string GetDescription \(\)](#)

Public Attributes

- [std::string desc](#)
Description of the bitstream, inserted by bitgen. Format is "ncdfile.ncd;UserID=0xdeadbeef".
- [std::string date](#)
Date the bitstream was created, inserted by bitgen.
- [std::string time](#)
Time the bitstream was created, inserted by bitgen.

7.73.1 Detailed Description

A bitstream for Xilinx FPGAs.

The documentation for this class was generated from the following files:

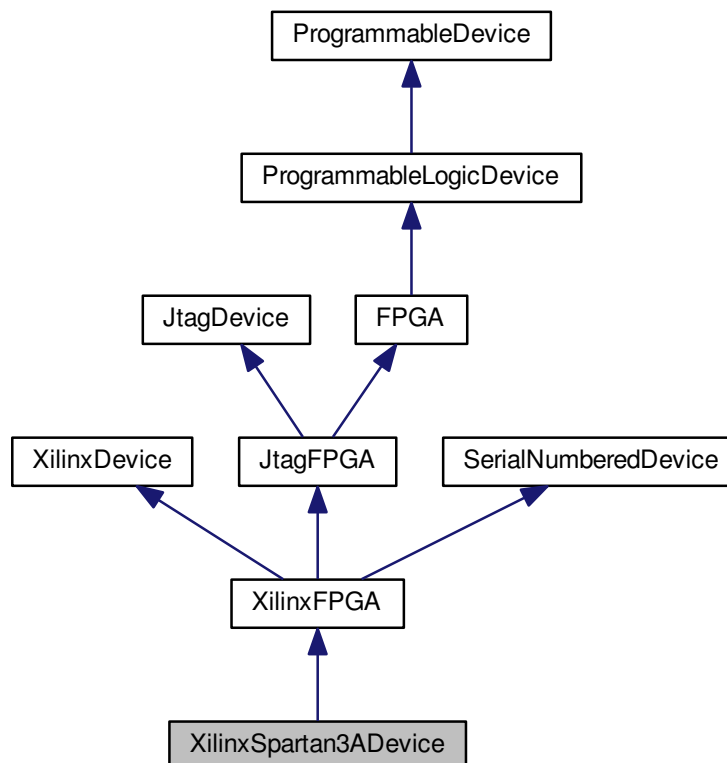
- [XilinxFPGABitstream.h](#)
- [XilinxFPGABitstream.cpp](#)

7.74 XilinxSpartan3ADevice Class Reference

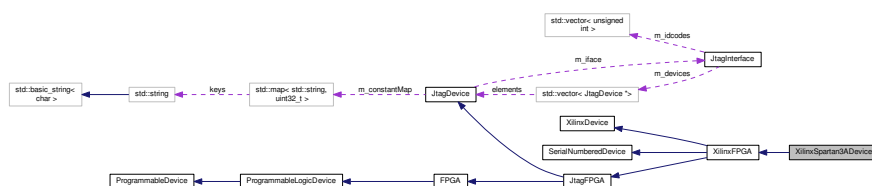
A Xilinx Spartan-3A [FPGA](#) device.

```
#include <XilinxSpartan3ADevice.h>
```

Inheritance diagram for XilinxSpartan3ADevice:



Collaboration diagram for XilinxSpartan3ADevice:



Public Types

- enum `deviceids` { `SPARTAN3A_50A` = 0x10 }
JTAG device IDs.
- enum `instructions` {
`INST_USER1` = 0x02, `INST_USER2` = 0x03, `INST_CFG_OUT` = 0x04, `INST_CFG_IN` = 0x05,
`INST_USERCODE` = 0x08, `INST_IDCODE` = 0x09, `INST_JPROGRAM` = 0x0B, `INST_JSTART` = 0x0C,
`INST_JSHUTDOWN` = 0x0D, `INST_ISC_ENABLE` = 0x10, `INST_ISC_DISABLE` = 0x16, `INST_ISC_DNA` =
0x31,
`INST_BYPASS` = 0x3F }
6-bit-wide JTAG instructions (see UG332 table 9-5 on page 207)

Public Member Functions

- `XilinxSpartan3ADevice` (unsigned int arraysize, unsigned int rev, unsigned int idcode, `JtagInterface` *iface, size_t pos)
Initializes this device.
- virtual `~XilinxSpartan3ADevice` ()
Empty virtual destructor.
- virtual std::string `GetDescription` ()
Gets a human-readable description of this device.
- virtual void `PrintStatusRegister` ()
- virtual bool `IsProgrammed` ()
Determines if this device is programmed or blank.
- virtual int `GetSerialNumberLength` ()
Gets the length of the device's unique serial number, in bytes (rounded up to the nearest whole byte).
- virtual int `GetSerialNumberLengthBits` ()
Gets the length of the device's unique serial number, in bits.
- virtual void `GetSerialNumber` (unsigned char *data)
Gets the device's unique serial number.
- virtual void `Erase` ()
Erases the device configuration and restores the device to a blank state.
- virtual void `InternalErase` ()
- virtual `FirmwareImage` * `LoadFirmwareImage` (const unsigned char *data, size_t len)
Parses an in-memory image of a firmware image into a format suitable for loading into the device.
- virtual void `Program` (`FirmwareImage` *image)
Loads a new firmware image onto the device.
- virtual void `Reboot` ()
*Reboots the *FPGA* and loads from external memory, if possible.*
- virtual size_t `GetNumUserInstructions` ()
*Get the number of JTAG instructions which are routed to *FPGA* fabric.*
- virtual void `SelectUserInstruction` (size_t index)
Sets the instruction register to the specified user instruction.
- unsigned int `GetArraySize` ()

Static Public Member Functions

- static [JtagDevice](#) * [CreateDevice](#) (unsigned int arraysize, unsigned int rev, unsigned int idcode, [JtagInterface](#) *iface, size_t pos)
Factory method.

Public Attributes

- enum [XilinxSpartan3ADevice::deviceids](#) `__attribute__`

Protected Types

- enum [spartan3a_config_opcodes](#) { [S3A_CONFIG_OP_NOP](#) = 0, [S3A_CONFIG_OP_READ](#) = 1, [S3A_CONFIG_OP_WRITE](#) = 2 }
Spartan-3A configuration opcodes (see UG332 page 323)
- enum [spartan3a_config_frame_types](#) { [S3A_CONFIG_FRAME_TYPE_1](#) = 1, [S3A_CONFIG_FRAME_TYPE_2](#) = 2 }
Spartan-3A configuration frame types (see UG332 page 323)
- enum [spartan3a_config_regs](#) {
[S3A_CONFIG_REG_CRC](#) = 0x00, [S3A_CONFIG_REG_FAR_MAJ](#) = 0x01, [S3A_CONFIG_REG_FAR_MIN](#) = 0x02, [S3A_CONFIG_REG_FDRI](#) = 0x03,
[S3A_CONFIG_REG_FDRO](#) = 0x04, [S3A_CONFIG_REG_CMD](#) = 0x05, [S3A_CONFIG_REG_CTL](#) = 0x06,
[S3A_CONFIG_REG_MASK](#) = 0x07,
[S3A_CONFIG_REG_STAT](#) = 0x08, [S3A_CONFIG_REG_LOUT](#) = 0x09, [S3A_CONFIG_REG_COR1](#) = 0x0a, [S3A_CONFIG_REG_COR2](#) = 0x0b,
[S3A_CONFIG_REG_PWRDN](#) = 0x0c, [S3A_CONFIG_REG_FLR](#) = 0x0d, [S3A_CONFIG_REG_IDCODE](#) = 0x0e, [S3A_CONFIG_REG_HCOPT](#) = 0x10,
[S3A_CONFIG_REG_CSBO](#) = 0x12, [S3A_CONFIG_REG_GENERAL1](#) = 0x13, [S3A_CONFIG_REG_GENERAL2](#) = 0x14, [S3A_CONFIG_REG_MODE_REG](#) = 0x15,
[S3A_CONFIG_REG_PU_GWE](#) = 0x16, [S3A_CONFIG_REG_PU_GTS](#) = 0x17, [S3A_CONFIG_REG_MFWR](#) = 0x18, [S3A_CONFIG_REG_CCLK_FREQ](#) = 0x19,
[S3A_CONFIG_REG_SEU_OPT](#) = 0x1a, [S3A_CONFIG_REG_EXP_SIGN](#) = 0x1b, [S3A_CONFIG_REG_RDBK_SIGN](#) = 0x1c, [S3A_CONFIG_REG_MAX](#) }
Spartan-3A configuration registers (see UG332 page 325)
- enum [spartan3a_cmd_values](#) {
[S3A_CMD_NULL](#) = 0x0, [S3A_CMD_WCFG](#) = 0x1, [S3A_CMD_MFWR](#) = 0x2, [S3A_CMD_LFRM](#) = 0x3,
[S3A_CMD_RCFG](#) = 0x4, [S3A_CMD_START](#) = 0x5, [S3A_CMD_RCAP](#) = 0x6, [S3A_CMD_RCRC](#) = 0x7,
[S3A_CMD_AGHIGH](#) = 0x8, [S3A_CMD_GRESTORE](#) = 0xa, [S3A_CMD_SHUTDOWN](#) = 0xb, [S3A_CMD_GCAPTURE](#) = 0xc,
[S3A_CMD_DESYNC](#) = 0xd, [S3A_CMD_REBOOT](#) = 0xe }
Spartan-3A CMD register values (see UG332 page 325-326)

Protected Member Functions

- virtual uint32_t [ReadWordConfigRegister](#) (unsigned int reg)
Reads a single 32-bit word from a config register.
- virtual void [ParseBitstreamInternals](#) (const unsigned char *data, size_t len, [XilinxFPGABitstream](#) *bitstream, size_t fpos)
Reads several 16-bit words from a config register.
- void [SetIR](#) (unsigned char irval)
- void [SetIRDeferred](#) (unsigned char irval)

Protected Attributes

- unsigned int `m_arraysize`
Array size (the specific Spartan-6 device we are)
- unsigned int `m_rev`
Stepping number.

7.74.1 Detailed Description

A Xilinx Spartan-3A [FPGA](#) device.

7.74.2 Member Enumeration Documentation

7.74.2.1 deviceids

```
enum XilinxSpartan3ADevice::deviceids
```

JTAG device IDs.

Enumerator

SPARTAN3A_50A	XC3S50A.
---------------	----------

7.74.2.2 instructions

```
enum XilinxSpartan3ADevice::instructions
```

6-bit-wide JTAG instructions (see UG332 table 9-5 on page 207)

Enumerator

INST_USER1	User-defined instruction 1.
INST_USER2	User-defined instruction 2.
INST_CFG_OUT	Read configuration register.
INST_CFG_IN	Write configuration register.
INST_USERCODE	Read user ID code.
INST_IDCODE	Read ID code.
INST_JPROGRAM	Enters programming mode (erases FPGA configuration)
INST_JSTART	Runs the FPGA startup sequence (must supply dummy clocks after)
INST_JSHUTDOWN	Runs the FPGA shutdown sequence (must supply dummy clocks after)
INST_ISC_ENABLE	Enters In-System Configuration mode (must load INST_JPROGRAM before)
INST_ISC_DISABLE	Leaves In-System Configuration mode.
INST_ISC_DNA	Read device DNA (must load INST_ISC_ENABLE before and INST_ISC_DISABLE after) Note that this opcode isn't the same as Spartan-6.
INST_BYPASS	Standard JTAG bypass.

7.74.3 Constructor & Destructor Documentation

7.74.3.1 XilinxSpartan3ADevice()

```
XilinxSpartan3ADevice::XilinxSpartan3ADevice (
    unsigned int arraysize,
    unsigned int rev,
    unsigned int idcode,
    JtagInterface * iface,
    size_t pos )
```

Initializes this device.

Parameters

<i>arraysize</i>	Array size from JTAG ID code
<i>rev</i>	Revision number from JTAG ID code
<i>idcode</i>	The ID code of this device
<i>iface</i>	The JTAG adapter this device was discovered on
<i>pos</i>	Position in the chain that this device was discovered

7.74.4 Member Function Documentation

7.74.4.1 Erase()

```
void XilinxSpartan3ADevice::Erase ( ) [virtual]
```

Erases the device configuration and restores the device to a blank state.

After this function is called, regardless of success or failure, all existing connections to on-chip code become invalid.

Exceptions

JtagException	if the erase operation fails
-------------------------------	------------------------------

Implements [ProgrammableDevice](#).

7.74.4.2 GetDescription()

```
string XilinxSpartan3ADevice::GetDescription ( ) [virtual]
```

Gets a human-readable description of this device.

Example: "Xilinx XC6SLX45 stepping 3"

Returns

Device description

Implements [JtagDevice](#).

7.74.4.3 GetSerialNumber()

```
void XilinxSpartan3ADevice::GetSerialNumber (
    unsigned char * data ) [virtual]
```

Gets the device's unique serial number.

Note that some architectures, such as Spartan-6, cannot read the serial number over JTAG without erasing the [FPGA](#) configuration. If this is the case, calling this function will automatically erase the [FPGA](#).

Call [ReadingSerialRequiresReset\(\)](#) to see if this is the case.

Exceptions

JtagException	if an error occurs during the read operation
-------------------------------	--

Parameters

<i>data</i>	Buffer to store the serial number into. Must be at least as large as the size given by GetSerialNumberLength() .
-------------	--

Implements [SerialNumberedDevice](#).

7.74.4.4 GetSerialNumberLength()

```
int XilinxSpartan3ADevice::GetSerialNumberLength ( ) [virtual]
```

Gets the length of the device's unique serial number, in bytes (rounded up to the nearest whole byte).

Returns

Serial number length

Implements [SerialNumberedDevice](#).

7.74.4.5 GetSerialNumberLengthBits()

```
int XilinxSpartan3ADevice::GetSerialNumberLengthBits ( ) [virtual]
```

Gets the length of the device's unique serial number, in bits.

Returns

Serial number length

Implements [SerialNumberedDevice](#).

7.74.4.6 IsProgrammed()

```
bool XilinxSpartan3ADevice::IsProgrammed ( ) [virtual]
```

Determines if this device is programmed or blank.

Returns

true if programmed, false if blank

Implements [ProgrammableDevice](#).

7.74.4.7 LoadFirmwareImage()

```
FirmwareImage * XilinxSpartan3ADevice::LoadFirmwareImage (
    const unsigned char * data,
    size_t len ) [virtual]
```

Parses an in-memory image of a firmware image into a format suitable for loading into the device.

Exceptions

JtagException	if the image is malformed
-------------------------------	---------------------------

Parameters

<i>data</i>	Pointer to the start of the firmware image, including headers
<i>len</i>	Length of the firmware image

Returns

Pointer to an [FirmwareImage](#) object suitable for passing to [Configure\(\)](#).

Implements [ProgrammableDevice](#).

7.74.4.8 ParseBitstreamInternals()

```
void XilinxSpartan3ADevice::ParseBitstreamInternals (
    const unsigned char * data,
    size_t len,
    XilinxFPGABitstream * bitstream,
    size_t fpos ) [protected], [virtual]
```

Reads several 16-bit words from a config register.

The current implementation uses type 1 packets and is thus limited to reading less than 32 words.

Exceptions

JtagException	if the read fails
-------------------------------	-------------------

Parameters

<i>reg</i>	The configuration register to read
<i>dout</i>	Buffer to read into
<i>count</i>	Number of 16-bit words to read

Implements [XilinxFPGA](#).

7.74.4.9 Program()

```
void XilinxSpartan3ADevice::Program (
    FirmwareImage * image ) [virtual]
```

Loads a new firmware image onto the device.

After this function is called, regardless of success or failure, all existing connections to on-chip code become invalid.

Exceptions

JtagException	if the erase operation fails
-------------------------------	------------------------------

Parameters

<i>image</i>	The parsed image to load
--------------	--------------------------

Implements [ProgrammableDevice](#).

7.74.4.10 ReadWordConfigRegister()

```
uint32_t XilinxSpartan3ADevice::ReadWordConfigRegister (
    unsigned int reg ) [protected], [virtual]
```

Reads a single 32-bit word from a config register.

Note that Spartan-3A devices expect data clocked in MSB first but the JTAG API clocks data LSB first. Some swapping is required as a result.

Exceptions

JtagException	if the read fails
-------------------------------	-------------------

Parameters

<i>reg</i>	The configuration register to read
------------	------------------------------------

Implements [XilinxFPGA](#).

The documentation for this class was generated from the following files:

- [XilinxSpartan3ADevice.h](#)
- [XilinxSpartan3ADevice.cpp](#)

7.75 XilinxSpartan3ADeviceConfigurationFrame Union Reference

Spartan-3A configuration frame header (see UG332 page 323)

```
#include <XilinxSpartan3ADevice.h>
```

Public Member Functions

- ```
struct {
 unsigned int count:5
 Count field.
 unsigned int reg_addr:6
 Register address.
 unsigned int op:2
 Opcode.
 unsigned int type:3
 Frame type.
} __attribute__((packed)) bits
```

## Public Attributes

- `uint16_t word`  
*The raw configuration word.*

### 7.75.1 Detailed Description

Spartan-3A configuration frame header (see UG332 page 323)

### 7.75.2 Member Data Documentation

#### 7.75.2.1 count

```
unsigned int XilinxSpartan3ADeviceConfigurationFrame::count
```

Count field.

- Type 1 packets: word count
- Type 2 packets: don't care

#### 7.75.2.2 op

```
unsigned int XilinxSpartan3ADeviceConfigurationFrame::op
```

Opcode.

Must be one of the following:

- XilinxSpartan3ADevice::S3\_CONFIG\_OP\_NOP
- XilinxSpartan3ADevice::S3\_CONFIG\_OP\_READ
- XilinxSpartan3ADevice::S3\_CONFIG\_OP\_WRITE

#### 7.75.2.3 type

```
unsigned int XilinxSpartan3ADeviceConfigurationFrame::type
```

Frame type.

Must be one of the following:

- XilinxSpartan3ADevice::S3A\_CONFIG\_FRAME\_TYPE\_1
- XilinxSpartan3ADevice::S3A\_CONFIG\_FRAME\_TYPE\_2

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

- [XilinxSpartan3ADevice.h](#)

## 7.76 XilinxSpartan3ADeviceStatusRegister Union Reference

Spartan-3A status register (see UG332 table 17-13, pages 327-328)

```
#include <XilinxSpartan3ADevice.h>
```

### Public Member Functions

- ```
struct {
    unsigned int crc\_err:1
        Indicates that the device failed to configure due to a CRC error.
    unsigned int idcode\_err:1
        Indicates that the device failed to configure due to the bitstream having the wrong ID code.
    unsigned int dcm\_lock:1
        Asserted once all DCM/PLL instances used in the design have locked on.
    unsigned int gts\_cfg\_b:1
        Status of global tristate net.
    unsigned int gwe:1
        Status of global write-enable net.
    unsigned int ghigh:1
        Status of GHIGH (TODO: describe what this is)
    unsigned int vsel:3
        Status of the SPI variant select pins.
    unsigned int mode:3
        Status of the mode bits.
    unsigned int init\_b:1
        Status of the INIT_B pin.
    unsigned int done:1
        Status of the DONE pin.
    unsigned int seu\_err:1
        True if there was a post-config CRC error.
    unsigned int sync\_timeout:1
        True if the config watchdog timer ran out.
} __attribute__((packed)) bits
```

Public Attributes

- `uint32_t word`
 - The raw status register value.*

7.76.1 Detailed Description

Spartan-3A status register (see UG332 table 17-13, pages 327-328)

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

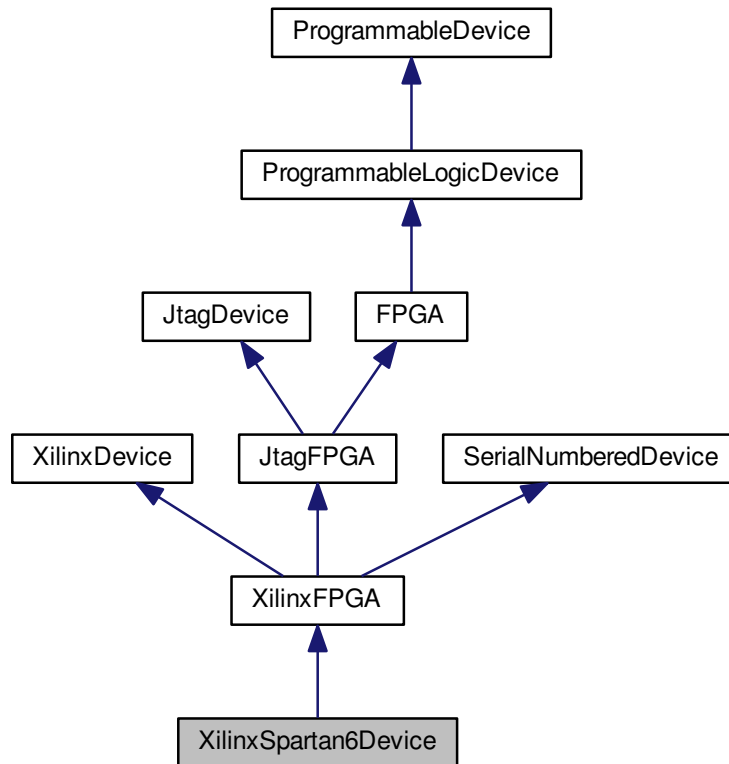
- [XilinxSpartan3ADevice.h](#)

7.77 XilinxSpartan6Device Class Reference

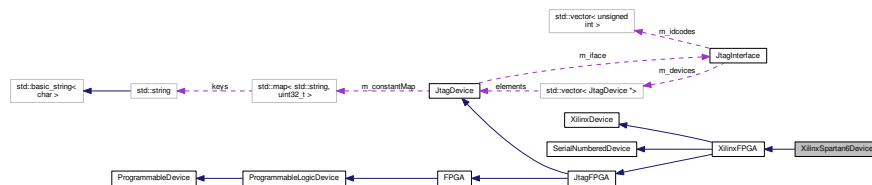
A Xilinx Spartan-6 [FPGA](#) device.

```
#include <XilinxSpartan6Device.h>
```

Inheritance diagram for XilinxSpartan6Device:



Collaboration diagram for XilinxSpartan6Device:



Public Types

- enum [deviceids](#) { [SPARTAN6_LX9](#) = 1, [SPARTAN6_LX16](#) = 2, [SPARTAN6_LX25](#) = 4, [SPARTAN6_LX45](#) = 8 }

JTAG device IDs.

- enum [instructions](#) {
[INST_USER1](#) = 0x02, [INST_USER2](#) = 0x03, [INST_USER3](#) = 0x1A, [INST_USER4](#) = 0x1B,
[INST_CFG_OUT](#) = 0x04, [INST_CFG_IN](#) = 0x05, [INST_IDCODE](#) = 0x09, [INST_JPROGRAM](#) = 0x0B,
[INST_JSTART](#) = 0x0C, [INST_JSHUTDOWN](#) = 0x0D, [INST_ISC_ENABLE](#) = 0x10, [INST_ISC_DISABLE](#) =
0x16,
[INST_ISC_DNA](#) = 0x30, [INST_BYPASS](#) = 0x3F }

6-bit-wide JTAG instructions (see UG380 table 10-2)

Public Member Functions

- [XilinxSpartan6Device](#) (unsigned int arraysize, unsigned int rev, unsigned int idcode, [JtagInterface](#) *iface, size_t pos)
Initializes this device.
- virtual [~XilinxSpartan6Device](#) ()
Empty virtual destructor.
- virtual std::string [GetDescription](#) ()
Gets a human-readable description of this device.
- virtual void [PrintStatusRegister](#) ()
- virtual bool [IsProgrammed](#) ()
Determines if this device is programmed or blank.
- virtual int [GetSerialNumberLength](#) ()
Gets the length of the device's unique serial number, in bytes (rounded up to the nearest whole byte).
- virtual int [GetSerialNumberLengthBits](#) ()
Gets the length of the device's unique serial number, in bits.
- virtual void [GetSerialNumber](#) (unsigned char *data)
Gets the device's unique serial number.
- virtual size_t [GetNumUserInstructions](#) ()
Get the number of JTAG instructions which are routed to [FPGA](#) fabric.
- virtual void [SelectUserInstruction](#) (size_t index)
Sets the instruction register to the specified user instruction.
- virtual void [Erase](#) ()
Erases the device configuration and restores the device to a blank state.
- virtual void [InternalErase](#) ()
- virtual [FirmwareImage](#) * [LoadFirmwareImage](#) (const unsigned char *data, size_t len)
Parses an in-memory image of a firmware image into a format suitable for loading into the device.
- virtual void [Program](#) ([FirmwareImage](#) *image)
Loads a new firmware image onto the device.
- virtual void [Reboot](#) ()
Reboots the [FPGA](#) and loads from external memory, if possible.
- unsigned int [GetArraySize](#) ()

Static Public Member Functions

- static [JtagDevice](#) * [CreateDevice](#) (unsigned int arraysize, unsigned int rev, unsigned int idcode, [JtagInterface](#) *iface, size_t pos)
Factory method.

Public Attributes

- enum [XilinxSpartan6Device::deviceids](#) [__attribute__](#)

Protected Types

- enum [spartan6_config_opcodes](#) { **S6_CONFIG_OP_NOP** = 0, **S6_CONFIG_OP_READ** = 1, **S6_CONFIG_OP_WRITE** = 2 }
Spartan-6 configuration opcodes (see UG380 page 90)
- enum [spartan6_config_frame_types](#) { **S6_CONFIG_FRAME_TYPE_1** = 1, **S6_CONFIG_FRAME_TYPE_2** = 2 }
Spartan-6 configuration frame types (see UG380 page 91)
- enum [spartan6_config_regs](#) {
S6_CONFIG_REG_CRC = 0x00, **S6_CONFIG_REG_FAR_MAJ** = 0x01, **S6_CONFIG_REG_FAR_MIN** = 0x02, **S6_CONFIG_REG_FDRI** = 0x03,
S6_CONFIG_REG_FDRO = 0x04, **S6_CONFIG_REG_CMD** = 0x05, **S6_CONFIG_REG_CTL** = 0x06, **S6_CONFIG_REG_MASK** = 0x07,
S6_CONFIG_REG_STAT = 0x08, **S6_CONFIG_REG_LOUT** = 0x09, **S6_CONFIG_REG_COR1** = 0x0a, **S6_CONFIG_REG_COR2** = 0x0b,
S6_CONFIG_REG_PWRDN = 0x0c, **S6_CONFIG_REG_FLR** = 0x0d, **S6_CONFIG_REG_IDCODE** = 0x0e, **S6_CONFIG_REG_CWDT** = 0x0f,
S6_CONFIG_REG_HC_OPT = 0x10, **S6_CONFIG_REG_CSBO** = 0x12, **S6_CONFIG_REG_GENERAL1** = 0x13, **S6_CONFIG_REG_GENERAL2** = 0x14,
S6_CONFIG_REG_GENERAL3 = 0x15, **S6_CONFIG_REG_GENERAL4** = 0x16, **S6_CONFIG_REG_GENERAL5** = 0x17, **S6_CONFIG_REG_MODE** = 0x18,
S6_CONFIG_REG_PU_GWE = 0x19, **S6_CONFIG_REG_PU_GTS** = 0x1a, **S6_CONFIG_REG_MFWR** = 0x1b, **S6_CONFIG_REG_CCLK_FREQ** = 0x1c,
S6_CONFIG_REG_SEU_OPT = 0x1d, **S6_CONFIG_REG_EXP_SIGN** = 0x1e, **S6_CONFIG_REG_RDBK_SIGN** = 0x1f, **S6_CONFIG_REG_BOOTSTS** = 0x20,
S6_CONFIG_REG_EYE_MASK = 0x21, **S6_CONFIG_REG_CBC** = 0x22, **S6_CONFIG_REG_MAX** }
Spartan-6 configuration registers (see UG380 page 92)
- enum [spartan6_cmd_values](#) {
S6_CMD_NULL = 0x0, **S6_CMD_WCFG** = 0x1, **S6_CMD_MFW** = 0x2, **S6_CMD_LFRM** = 0x3,
S6_CMD_RCFG = 0x4, **S6_CMD_START** = 0x5, **S6_CMD_RCRC** = 0x7, **S6_CMD_AGHIGH** = 0x8,
S6_CMD_GRESTORE = 0xa, **S6_CMD_SHUTDOWN** = 0xb, **S6_CMD_DESYNC** = 0xd, **S6_CMD_IPROG** = 0xe }
Spartan-6 CMD register values (see UG380 page 94-95)

Protected Member Functions

- virtual `uint32_t` [ReadWordConfigRegister](#) (unsigned int reg)
Reads a single 16-bit word from a config register.
- void [ReadWordsConfigRegister](#) (unsigned int reg, `uint16_t *dout`, unsigned int count)
Reads several 16-bit words from a config register.
- void [WriteWordConfigRegister](#) (unsigned int reg, `uint16_t` value)
- virtual void [ParseBitstreamInternals](#) (const unsigned char *data, `size_t` len, `XilinxFPGABitstream *bitstream`, `size_t` fpos)
Parse a full bitstream image (specific to the derived FPGA family)
- void [SetIR](#) (unsigned char irval)
- void [SetIRDeferred](#) (unsigned char irval)

Protected Attributes

- unsigned int [m_arraysize](#)
Array size (the specific Spartan-6 device we are)
- unsigned int [m_rev](#)
Stepping number.

7.77.1 Detailed Description

A Xilinx Spartan-6 [FPGA](#) device.

7.77.2 Member Enumeration Documentation

7.77.2.1 deviceids

```
enum XilinxSpartan6Device::deviceids
```

JTAG device IDs.

Enumerator

SPARTAN6_LX9	XC6SLX9.
SPARTAN6_LX16	XC6SLX16.
SPARTAN6_LX25	XC6SLX25.
SPARTAN6_LX45	XC6SLX45.

7.77.2.2 instructions

```
enum XilinxSpartan6Device::instructions
```

6-bit-wide JTAG instructions (see UG380 table 10-2)

Enumerator

INST_USER1	User-defined instruction 1.
INST_USER2	User-defined instruction 2.
INST_USER3	User-defined instruction 3.
INST_USER4	User-defined instruction 4.
INST_CFG_OUT	Read configuration register.
INST_CFG_IN	Write configuration register.
INST_IDCODE	Read ID code.
INST_JPROGRAM	Enters programming mode (erases FPGA configuration)
INST_JSTART	Runs the FPGA startup sequence (must supply dummy clocks after)
INST_JSHUTDOWN	Runs the FPGA shutdown sequence (must supply dummy clocks after)
INST_ISC_ENABLE	Enters In-System Configuration mode (must load INST_JPROGRAM before)
INST_ISC_DISABLE	Leaves In-System Configuration mode.
INST_ISC_DNA	Read device DNA (must load INST_ISC_ENABLE before and INST_ISC_DISABLE after)
INST_BYPASS	Standard JTAG bypass.

7.77.3 Constructor & Destructor Documentation

7.77.3.1 XilinxSpartan6Device()

```
XilinxSpartan6Device::XilinxSpartan6Device (
    unsigned int arraysize,
    unsigned int rev,
    unsigned int idcode,
    JtagInterface * iface,
    size_t pos )
```

Initializes this device.

Parameters

<i>arraysize</i>	Array size from JTAG ID code
<i>rev</i>	Revision number from JTAG ID code
<i>idcode</i>	The ID code of this device
<i>iface</i>	The JTAG adapter this device was discovered on
<i>pos</i>	Position in the chain that this device was discovered

7.77.4 Member Function Documentation

7.77.4.1 Erase()

```
void XilinxSpartan6Device::Erase ( ) [virtual]
```

Erases the device configuration and restores the device to a blank state.

After this function is called, regardless of success or failure, all existing connections to on-chip code become invalid.

Exceptions

JtagException	if the erase operation fails
-------------------------------	------------------------------

Implements [ProgrammableDevice](#).

7.77.4.2 GetDescription()

```
string XilinxSpartan6Device::GetDescription ( ) [virtual]
```

Gets a human-readable description of this device.

Example: "Xilinx XC6SLX45 stepping 3"

Returns

Device description

Implements [JtagDevice](#).

7.77.4.3 GetSerialNumber()

```
void XilinxSpartan6Device::GetSerialNumber (
    unsigned char * data ) [virtual]
```

Gets the device's unique serial number.

Note that some architectures, such as Spartan-6, cannot read the serial number over JTAG without erasing the [FPGA](#) configuration. If this is the case, calling this function will automatically erase the [FPGA](#).

Call [ReadingSerialRequiresReset\(\)](#) to see if this is the case.

Exceptions

JtagException	if an error occurs during the read operation
-------------------------------	--

Parameters

<i>data</i>	Buffer to store the serial number into. Must be at least as large as the size given by GetSerialNumberLength() .
-------------	--

Implements [SerialNumberedDevice](#).

7.77.4.4 GetSerialNumberLength()

```
int XilinxSpartan6Device::GetSerialNumberLength ( ) [virtual]
```

Gets the length of the device's unique serial number, in bytes (rounded up to the nearest whole byte).

Returns

Serial number length

Implements [SerialNumberedDevice](#).

7.77.4.5 GetSerialNumberLengthBits()

```
int XilinxSpartan6Device::GetSerialNumberLengthBits ( ) [virtual]
```

Gets the length of the device's unique serial number, in bits.

Returns

Serial number length

Implements [SerialNumberedDevice](#).

7.77.4.6 IsProgrammed()

```
bool XilinxSpartan6Device::IsProgrammed ( ) [virtual]
```

Determines if this device is programmed or blank.

Returns

true if programmed, false if blank

Implements [ProgrammableDevice](#).

7.77.4.7 LoadFirmwareImage()

```
FirmwareImage * XilinxSpartan6Device::LoadFirmwareImage (
    const unsigned char * data,
    size_t len ) [virtual]
```

Parses an in-memory image of a firmware image into a format suitable for loading into the device.

Exceptions

JtagException	if the image is malformed
-------------------------------	---------------------------

Parameters

<i>data</i>	Pointer to the start of the firmware image, including headers
<i>len</i>	Length of the firmware image

Returns

Pointer to an [FirmwareImage](#) object suitable for passing to [Configure\(\)](#).

Implements [ProgrammableDevice](#).

7.77.4.8 ParseBitstreamInternals()

```
void XilinxSpartan6Device::ParseBitstreamInternals (
    const unsigned char * data,
    size_t len,
    XilinxFPGABitstream * bitstream,
    size_t fpos ) [protected], [virtual]
```

Parse a full bitstream image (specific to the derived [FPGA](#) family)

Exceptions

JtagException	if the bitstream is malformed or for the wrong device family
-------------------------------	--

Parameters

<i>data</i>	Pointer to the bitstream data
<i>len</i>	Length of the bitstream
<i>bitstream</i>	The bitstream object to load into
<i>fpos</i>	Position in the bitstream image to start parsing (after the end of headers)
<i>bVerbose</i>	Set to true for verbose debug output on bitstream internals

Implements [XilinxFPGA](#).

7.77.4.9 Program()

```
void XilinxSpartan6Device::Program (
    FirmwareImage * image ) [virtual]
```

Loads a new firmware image onto the device.

After this function is called, regardless of success or failure, all existing connections to on-chip code become invalid.

Exceptions

JtagException	if the erase operation fails
-------------------------------	------------------------------

Parameters

<i>image</i>	The parsed image to load
--------------	--------------------------

Implements [ProgrammableDevice](#).

7.77.4.10 ReadWordConfigRegister()

```
uint32_t XilinxSpartan6Device::ReadWordConfigRegister (
    unsigned int reg ) [protected], [virtual]
```

Reads a single 16-bit word from a config register.

Reference: UG380 page 115-116 Table 6-5

Note that Spartan-6 devices expect data clocked in MSB first but the JTAG API clocks data LSB first. Some swapping is required as a result.

Clock data into CFG_IN register Synchronization word Read STAT register Type = 1 001 Op = Read 01 Addr of reg xxxxxx Word count = 1 00001 = 0x2901 Two dummy words to flush packet buffer

Read from CFG_OUT register

Exceptions

JtagException	if the read fails
-------------------------------	-------------------

Parameters

<i>reg</i>	The configuration register to read
------------	------------------------------------

Returns

The register value. Note that only the low 16 bits are valid but [XilinxFPGA](#)'s API has 32 in the return type

Implements [XilinxFPGA](#).

7.77.4.11 ReadWordsConfigRegister()

```
void XilinxSpartan6Device::ReadWordsConfigRegister (
    unsigned int reg,
    uint16_t * dout,
    unsigned int count ) [protected]
```

Reads several 16-bit words from a config register.

The current implementation uses type 1 packets and is thus limited to reading less than 32 words.

Exceptions

JtagException	if the read fails
-------------------------------	-------------------

Parameters

<i>reg</i>	The configuration register to read
<i>dout</i>	Buffer to read into
<i>count</i>	Number of 16-bit words to read

The documentation for this class was generated from the following files:

- [XilinxSpartan6Device.h](#)
- [XilinxSpartan6Device.cpp](#)

7.78 XilinxSpartan6DeviceConfigurationFrame Union Reference

Spartan-6 configuration frame (see UG380 page 91)

```
#include <XilinxSpartan6Device.h>
```

Public Member Functions

- ```
struct {
 unsigned int count:5
 Count field.
 unsigned int reg_addr:6
 Register address.
 unsigned int op:2
 Opcode.
 unsigned int type:3
 Frame type.
} __attribute__((packed)) bits
```

### Public Attributes

- `uint16_t word`

*The raw configuration word.*

#### 7.78.1 Detailed Description

Spartan-6 configuration frame (see UG380 page 91)

For type 2 packets, the header is followed by a 32-bit big-endian length value

#### 7.78.2 Member Data Documentation

### 7.78.2.1 count

```
unsigned int XilinxSpartan6DeviceConfigurationFrame::count
```

Count field.

- Type 1 packets: word count
- Type 2 packets: don't care

### 7.78.2.2 op

```
unsigned int XilinxSpartan6DeviceConfigurationFrame::op
```

Opcode.

Must be one of the following:

- XilinxSpartan6Device::S6\_CONFIG\_OP\_NOP
- XilinxSpartan6Device::S6\_CONFIG\_OP\_READ
- XilinxSpartan6Device::S6\_CONFIG\_OP\_WRITE

### 7.78.2.3 type

```
unsigned int XilinxSpartan6DeviceConfigurationFrame::type
```

Frame type.

Must be one of the following:

- XilinxSpartan6Device::S6\_CONFIG\_FRAME\_TYPE\_1
- XilinxSpartan6Device::S6\_CONFIG\_FRAME\_TYPE\_2

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

- [XilinxSpartan6Device.h](#)

## 7.79 XilinxSpartan6DeviceStatusRegister Union Reference

Spartan-6 status register (see UG380 table 5-35)

```
#include <XilinxSpartan6Device.h>
```



## Public Member Functions

- ```

struct {
    unsigned int crc\_err:1
        Indicates that the device failed to configure due to a CRC error.
    unsigned int idcode\_err:1
        Indicates that the device failed to configure due to the bitstream having the wrong ID code.
    unsigned int dcm\_lock:1
        Asserted once all DCM/PLL instances used in the design have locked on.
    unsigned int gts\_cfg\_b:1
        Status of global tristate net.
    unsigned int gwe:1
        Status of global write-enable net.
    unsigned int ghigh:1
        Status of GHIGH (TODO: describe what this is)
    unsigned int decrypt\_err:1
        Decryption error flag.
    unsigned int decrypt\_en:1
        Bitstream encryption enable flag.
    unsigned int hswapen:1
        Status of the HSWAPEN pin.
    unsigned int m0:1
        Status of the M0 mode bit.
    unsigned int m1:1
        Status of the M1 mode bit.
    unsigned int reserved:1
        Reserved.
    unsigned int init\_b:1
        Status of the INIT_B pin.
    unsigned int done:1
        Status of the DONE pin.
    unsigned int suspend:1
        Suspend state.
    unsigned int fallback:1
        Configuration fallback state.
} \_\_attribute\_\_\(\(packed\)\) bits

```

Public Attributes

- `uint16_t word`
The raw status register value.

7.79.1 Detailed Description

Spartan-6 status register (see UG380 table 5-35)

Typical status register bits:

- [0] CRC ERROR : 0
- [1] IDCODE ERROR : 0
- [2] DCM LOCK STATUS : 1

- [3] GTS_CFG_B STATUS : 1
- [4] GWE STATUS : 1
- [5] GHIGH STATUS : 1
- [6] DECRYPTION ERROR : 0
- [7] DECRYPTOR ENABLE : 0
- [8] HSWAPEN PIN : 1
- [9] MODE PIN M[0] : 1
- [10] MODE PIN M[1] : 1
- [11] RESERVED : 0
- [12] INIT_B PIN : 1
- [13] DONE PIN : 1
- [14] SUSPEND STATUS : 0
- [15] FALLBACK STATUS : 0

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

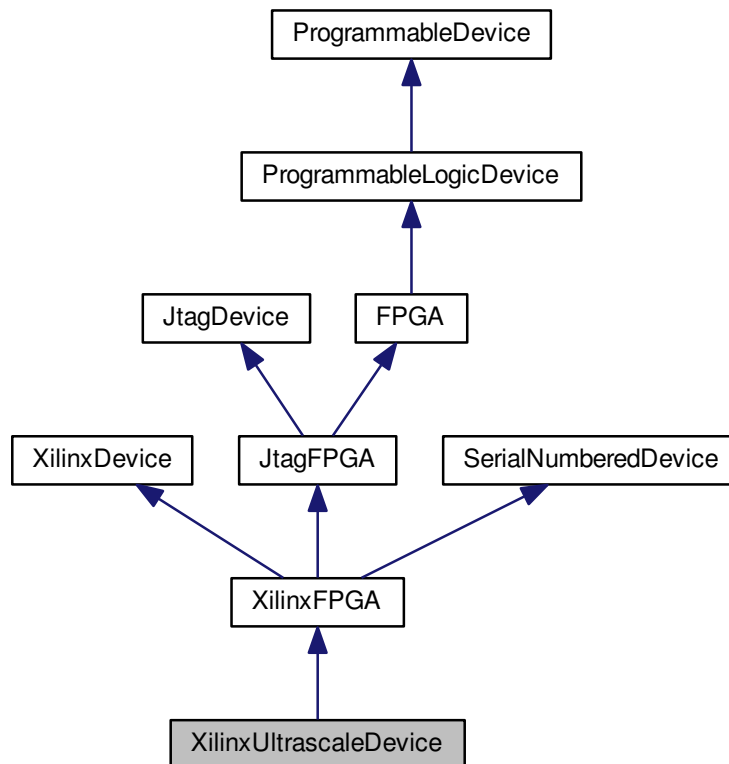
- [XilinxSpartan6Device.h](#)

7.80 XilinxUltrascaleDevice Class Reference

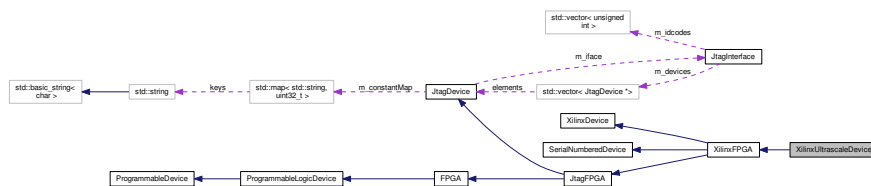
A Xilinx Ultrascale or Ultrascale+ [FPGA](#) device.

```
#include <XilinxUltrascaleDevice.h>
```

Inheritance diagram for XilinxUltrascaleDevice:



Collaboration diagram for XilinxUltrascaleDevice:



Public Types

- enum `deviceids` { **VUPLUS_9** = 0x131 }
JTAG device IDs.
- enum `instructions` {
`INST_SLR_BYPASS` = 0x24, `INST_CFG_OUT` = 0x04, `INST_CFG_IN` = 0x05, `INST_IDCODE` = 0x09,
`INST_JPROGRAM` = 0x0B, `INST_JSTART` = 0x0C, `INST_ISC_ENABLE` = 0x10, `INST_ISC_NOOP` = 0x14,
`INST_ISC_DISABLE` = 0x16, `INST_XSC_DNA` = 0x17, `INST_BYPASS` = 0x3F }
6-bit-wide JTAG instructions (see BSD file). Seems to be mostly same as 7 series
- enum `config_opcodes` { **CONFIG_OP_NOP** = 0, **CONFIG_OP_READ** = 1, **CONFIG_OP_WRITE** = 2 }

UltraScale configuration opcodes (see UG570 page 159). Same as for Spartan-6 and 7 series.

- enum [config_frame_types](#) { **CONFIG_FRAME_TYPE_1** = 1, **CONFIG_FRAME_TYPE_2** = 2 }

UltraScale configuration frame types (see UG570 page 158). Same as for Spartan-6 and 7 series.

- enum [ultrascale_config_regs](#) {
CONFIG_REG_CRC = 0x00, **CONFIG_REG_FAR** = 0x01, **CONFIG_REG_FDRI** = 0x02, **CONFIG_REG_FDRO** = 0x03,
CONFIG_REG_CMD = 0x04, **CONFIG_REG_CTL0** = 0x05, **CONFIG_REG_MASK** = 0x06, **CONFIG_REG_STAT** = 0x07,
CONFIG_REG_LOUT = 0x08, **CONFIG_REG_COR0** = 0x09, **CONFIG_REG_MFWR** = 0x0A, **CONFIG_REG_CBC** = 0x0B,
CONFIG_REG_IDCODE = 0x0C, **CONFIG_REG_AXSS** = 0x0D, **CONFIG_REG_COR1** = 0x0E, **CONFIG_REG_WBSTAR** = 0x10,
CONFIG_REG_TIMER = 0x11, **CONFIG_REG_BOOTSTS** = 0x16, **CONFIG_REG_CTL1** = 0x18, **CONFIG_REG_BSPI** = 0x1F,
CONFIG_REG_MAX }

UltraScale configuration registers (see UG570 page 159). Seems to be same as 7 series.

- enum [cmd_values](#) {
CMD_NULL = 0x00, **CMD_WCFG** = 0x01, **CMD_MFW** = 0x02, **CMD_LFRM** = 0x03,
CMD_RCFG = 0x04, **CMD_START** = 0x05, **CMD_RCRC** = 0x07, **CMD_AGHIGH** = 0x08,
CMD_SWITCH = 0x09, **CMD_GRESTORE** = 0x0a, **CMD_SHUTDOWN** = 0x0b, **CMD_DESYNC** = 0x0d,
CMD_IPROG = 0x0f, **CMD_CRCC** = 0x10, **CMD_LTIMER** = 0x11, **CMD_BSPI_READ** = 0x12,
CMD_FALL_EDGE = 0x13, **CMD_MAX** }

UltraScale CMD register values (see UG570 page table 9-22).

Public Member Functions

- [XilinxUltrascaleDevice](#) (unsigned int arraysize, unsigned int family, unsigned int rev, unsigned int idcode, [JtagInterface](#) *iface, size_t pos)
Initializes this device.
- virtual [~XilinxUltrascaleDevice](#) ()
Empty virtual destructor.
- virtual std::string [GetDescription](#) ()
Gets a human-readable description of this device.
- virtual void [PrintStatusRegister](#) ()
- virtual bool [IsProgrammed](#) ()
Determines if this device is programmed or blank.
- virtual int [GetSerialNumberLength](#) ()
Gets the length of the device's unique serial number, in bytes (rounded up to the nearest whole byte).
- virtual int [GetSerialNumberLengthBits](#) ()
Gets the length of the device's unique serial number, in bits.
- virtual void [GetSerialNumber](#) (unsigned char *data)
Gets the device's unique serial number.
- virtual size_t [GetNumUserInstructions](#) ()
Get the number of JTAG instructions which are routed to [FPGA](#) fabric.
- virtual void [SelectUserInstruction](#) (size_t index)
Sets the instruction register to the specified user instruction.
- virtual void [Erase](#) ()
Erases the device configuration and restores the device to a blank state.
- virtual void [InternalErase](#) ()
- virtual [FirmwareImage](#) * [LoadFirmwareImage](#) (const unsigned char *data, size_t len)
Parses an in-memory image of a firmware image into a format suitable for loading into the device.
- virtual void [Program](#) ([FirmwareImage](#) *image)

- Loads a new firmware image onto the device.*
- virtual void [Reboot](#) ()
 - Reboots the [FPGA](#) and loads from external memory, if possible.*
- void **AnalyzeSVF** (std::string path)

Static Public Member Functions

- static [JtagDevice](#) * [CreateDevice](#) (unsigned int arraysize, unsigned int family, unsigned int rev, unsigned int idcode, [JtagInterface](#) *iface, size_t pos)
 - Factory method.*

Public Attributes

- enum [XilinxUltrascaleDevice::deviceids](#) **__attribute__**

Protected Member Functions

- virtual uint32_t [ReadWordConfigRegister](#) (unsigned int reg)
 - Reads a single 32-bit word from a config register.*
- virtual void [ParseBitstreamInternals](#) (const unsigned char *data, size_t len, [XilinxFPGABitstream](#) *bitstream, size_t fpos)
 - Parse a full bitstream image (specific to the derived [FPGA](#) family)*
- bool **ParseType1ConfigFrame** ([XilinxUltrascaleDeviceConfigurationFrame](#) frame, const unsigned char *data, size_t len, size_t &fpos, uint32_t &idcode, bool &desync, bool flip_bit_order=false)
- bool [GetSVFLine](#) (FILE *fp, std::string &line)
- std::string [GetSVFOpcode](#) (std::string &line)
- void [SetIRForMasterSLR](#) (unsigned char irval, bool defer=false)
- void [SetIRForAllSLRs](#) (unsigned char irval, bool defer=false)
- size_t [InitializePartDimensions](#) (unsigned int arraysize, unsigned int family)
 - Set up SLR count etc and return total IR size.*

Protected Attributes

- unsigned int [m_arraysize](#)
 - Array size (the specific device we are)*
- unsigned int [m_family](#)
 - Family (Ultrascale or Ultrascale+)*
- unsigned int [m_rev](#)
 - Stepping number.*
- unsigned int [m_slrCount](#)
 - Number of SLRs in the device.*
- unsigned int [m_masterSLR](#)
 - Index of the master SLR (zero-based). Always 0 for monolithic devices.*

7.80.1 Detailed Description

A Xilinx Ultrascale or Ultrascale+ [FPGA](#) device.

7.80.2 Member Enumeration Documentation

7.80.2.1 cmd_values

enum `XilinxUltrascaleDevice::cmd_values`

UltraScale CMD register values (see UG570 page table 9-22).

Seems to be mostly same as 7 series but a few things changed (commented)

7.80.2.2 instructions

enum `XilinxUltrascaleDevice::instructions`

6-bit-wide JTAG instructions (see BSDL file). Seems to be mostly same as 7 series

Enumerator

<code>INST_SLR_BYPASS</code>	Turn off the JTAG subsystem for a SLR we're not talking to.
<code>INST_CFG_OUT</code>	Read configuration register.
<code>INST_CFG_IN</code>	Write configuration register.
<code>INST_IDCODE</code>	Read user ID code. Read ID code
<code>INST_JPROGRAM</code>	Enters programming mode (erases FPGA configuration)
<code>INST_JSTART</code>	Runs the FPGA startup sequence (must supply dummy clocks after)
<code>INST_ISC_ENABLE</code>	Runs the FPGA shutdown sequence (must supply dummy clocks after) Enters In-System Configuration mode (must load <code>INST_JPROGRAM</code> before)
<code>INST_ISC_DISABLE</code>	Leaves In-System Configuration mode.
<code>INST_XSC_DNA</code>	Read device DNA (must load <code>INST_ISC_ENABLE</code> before and <code>INST_ISC_DISABLE</code> after) Same as 7-series but not same as Spartan-6
<code>INST_BYPASS</code>	Standard JTAG bypass. Access to the ADC Not present in Spartan-6

7.80.3 Constructor & Destructor Documentation

7.80.3.1 XilinxUltrascaleDevice()

```
XilinxUltrascaleDevice::XilinxUltrascaleDevice (
    unsigned int arraysize,
    unsigned int family,
    unsigned int rev,
    unsigned int idcode,
    JtagInterface * iface,
    size_t pos )
```

Initializes this device.

Parameters

<i>arraysize</i>	Array size from JTAG ID code
<i>family</i>	Family from JTAG ID code (needed since this class handles both ultrascale and ultrascale+)
<i>rev</i>	Revision number from JTAG ID code
<i>idcode</i>	The ID code of this device
<i>iface</i>	The JTAG adapter this device was discovered on
<i>pos</i>	Position in the chain that this device was discovered
<i>irlength</i>	Length of the JTAG instruction register

7.80.4 Member Function Documentation

7.80.4.1 Erase()

```
void XilinxUltrascaleDevice::Erase ( ) [virtual]
```

Erases the device configuration and restores the device to a blank state.

After this function is called, regardless of success or failure, all existing connections to on-chip code become invalid.

Exceptions

<i>JtagException</i>	if the erase operation fails
--------------------------------------	------------------------------

Implements [ProgrammableDevice](#).

7.80.4.2 GetDescription()

```
string XilinxUltrascaleDevice::GetDescription ( ) [virtual]
```

Gets a human-readable description of this device.

Example: "Xilinx XC6SLX45 stepping 3"

Returns

Device description

Implements [JtagDevice](#).

7.80.4.3 GetSerialNumber()

```
void XilinxUltrascaleDevice::GetSerialNumber (
    unsigned char * data ) [virtual]
```

Gets the device's unique serial number.

Note that some architectures, such as Spartan-6, cannot read the serial number over JTAG without erasing the [FPGA](#) configuration. If this is the case, calling this function will automatically erase the [FPGA](#).

Call [ReadingSerialRequiresReset\(\)](#) to see if this is the case.

Exceptions

JtagException	if an error occurs during the read operation
-------------------------------	--

Parameters

<i>data</i>	Buffer to store the serial number into. Must be at least as large as the size given by GetSerialNumberLength() .
-------------	--

Implements [SerialNumberedDevice](#).

7.80.4.4 GetSerialNumberLength()

```
int XilinxUltrascaleDevice::GetSerialNumberLength ( ) [virtual]
```

Gets the length of the device's unique serial number, in bytes (rounded up to the nearest whole byte).

Returns

Serial number length

Implements [SerialNumberedDevice](#).

7.80.4.5 GetSerialNumberLengthBits()

```
int XilinxUltrascaleDevice::GetSerialNumberLengthBits ( ) [virtual]
```

Gets the length of the device's unique serial number, in bits.

Returns

Serial number length

Implements [SerialNumberedDevice](#).

7.80.4.6 IsProgrammed()

```
bool XilinxUltrascaleDevice::IsProgrammed ( ) [virtual]
```

Determines if this device is programmed or blank.

Returns

true if programmed, false if blank

Implements [ProgrammableDevice](#).

7.80.4.7 LoadFirmwareImage()

```
FirmwareImage * XilinxUltrascaleDevice::LoadFirmwareImage (
    const unsigned char * data,
    size_t len ) [virtual]
```

Parses an in-memory image of a firmware image into a format suitable for loading into the device.

Exceptions

JtagException	if the image is malformed
-------------------------------	---------------------------

Parameters

<i>data</i>	Pointer to the start of the firmware image, including headers
<i>len</i>	Length of the firmware image

Returns

Pointer to an [FirmwareImage](#) object suitable for passing to [Configure\(\)](#).

Implements [ProgrammableDevice](#).

7.80.4.8 ParseBitstreamInternals()

```
void XilinxUltrascaleDevice::ParseBitstreamInternals (
    const unsigned char * data,
    size_t len,
    XilinxFPGABitstream * bitstream,
    size_t fpos ) [protected], [virtual]
```

Parse a full bitstream image (specific to the derived [FPGA](#) family)

Exceptions

JtagException	if the bitstream is malformed or for the wrong device family
-------------------------------	--

Parameters

<i>data</i>	Pointer to the bitstream data
<i>len</i>	Length of the bitstream
<i>bitstream</i>	The bitstream object to load into
<i>fpos</i>	Position in the bitstream image to start parsing (after the end of headers)
<i>bVerbose</i>	Set to true for verbose debug output on bitstream internals

Implements [XilinxFPGA](#).

7.80.4.9 Program()

```
void XilinxUltrascaleDevice::Program (
    FirmwareImage * image ) [virtual]
```

Loads a new firmware image onto the device.

After this function is called, regardless of success or failure, all existing connections to on-chip code become invalid.

Exceptions

JtagException	if the erase operation fails
-------------------------------	------------------------------

Parameters

<i>image</i>	The parsed image to load
--------------	--------------------------

Implements [ProgrammableDevice](#).

7.80.4.10 ReadWordConfigRegister()

```
uint32_t XilinxUltrascaleDevice::ReadWordConfigRegister (
    unsigned int reg ) [protected], [virtual]
```

Reads a single 32-bit word from a config register.

Reference: UG570 page 164

Note that UltraScale devices expect data clocked in MSB first but the JTAG API clocks data LSB first. Some swapping is required as a result.

Clock data into CFG_IN register
Synchronization word
Nop
Read STAT register
Two dummy words to flush packet buffer

Read from CFG_OUT register

Exceptions

JtagException	if the read fails
-------------------------------	-------------------

Parameters

<i>reg</i>	The configuration register to read
------------	------------------------------------

Implements [XilinxFPGA](#).

The documentation for this class was generated from the following files:

- [XilinxUltraScaleDevice.h](#)
- [XilinxUltraScaleDevice.cpp](#)

7.81 XilinxUltraScaleDeviceConfigurationFrame Union Reference

UltraScale configuration frame (see UG570 page 158)

```
#include <XilinxUltraScaleDevice.h>
```

Public Member Functions

- ```
struct {
 unsigned int count:11
 Count field.
 unsigned int reserved:2
 Reserved, must be zero.
 unsigned int reg_addr:14
 Register address.
 unsigned int op:2
 Opcode.
 unsigned int type:3
 Frame type.
} __attribute__((packed)) bits
```
- ```
struct {
    unsigned int count:27
        Count field.
    unsigned int op:2
        Opcode.
    unsigned int type:3
        Frame type.
} __attribute__((packed)) bits\_type2
```

Public Attributes

- `uint32_t word`
The raw configuration word.

7.81.1 Detailed Description

UltraScale configuration frame (see UG570 page 158)

Same as 7 series

7.81.2 Member Data Documentation

7.81.2.1 `op`

```
unsigned int XilinxUltrascaleDeviceConfigurationFrame::op
```

Opcode.

Must be one of the following:

- `XilinxUltrascaleDevice::CONFIG_OP_NOP`
- `XilinxUltrascaleDevice::CONFIG_OP_READ`
- `XilinxUltrascaleDevice::CONFIG_OP_WRITE`

Must be zero

7.81.2.2 `type`

```
unsigned int XilinxUltrascaleDeviceConfigurationFrame::type
```

Frame type.

Must be `XilinxUltrascaleDevice::CONFIG_FRAME_TYPE_1`

Must be `XilinxUltrascaleDevice::CONFIG_FRAME_TYPE_2`

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

- [XilinxUltrascaleDevice.h](#)

7.82 XilinxUltrascaleDeviceStatusRegister Union Reference

UltraScale status register (see UG570 table 9-25)

```
#include <XilinxUltrascaleDevice.h>
```

Public Member Functions

- ```
struct {
 unsigned int crc_err:1
 Indicates that the device failed to configure due to a CRC error.
 unsigned int decryptor_enabled:1
 Indicates that the crypto subsystem is active.
 unsigned int mmcm_lock:1
 Indicates MMCMs and PLLs are locked.
 unsigned int dci_match:1
 Indicates DCI is matched.
 unsigned int eos:1
 End-of-Startup signal.
 unsigned int gts_cfg_b:1
 Status of GTS_CFG net.
 unsigned int gwe:1
 Status of GWE net.
 unsigned int ghigh_b:1
 Status of GHIGH_B net.
 unsigned int mode_pins:3
 Status of mode pins.
 unsigned int init_complete:1
 Internal init-finished signal.
 unsigned int init_b:1
 Status of INIT_B pin.
 unsigned int release_done:1
 Indicates DONE was released.
 unsigned int done:1
 Actual value on DONE pin.
 unsigned int id_error:1
 Indicates an ID code error occurred (write with wrong bitstream)
 unsigned int security_error:1
 Security / crypto error.
 unsigned int sysmon_over_temp:1
 Indicates board is too hot.
 unsigned int startup_state:3
 Status of startup state machine.
 unsigned int reserved_1:4
 Reserved.
 unsigned int bus_width:2
 Config bus width (see table 5-26)
 unsigned int reserved_2:5
 Reserved.
} __attribute__((packed)) bits
```

### Public Attributes

- `uint32_t word`  
*The raw status register value.*

### 7.82.1 Detailed Description

UltraScale status register (see UG570 table 9-25)

Very similar to the 7-series status register but with a few fields renamed.

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

- [XilinxUltrascaleDevice.h](#)

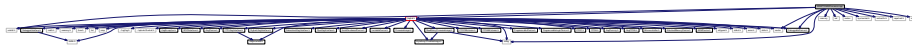
# Chapter 8

## File Documentation

### 8.1 ARM7TDMISProcessor.cpp File Reference

Implementation of [ARM7TDMISProcessor](#).

```
#include "jtaghal.h"
#include "DebuggableDevice.h"
Include dependency graph for ARM7TDMISProcessor.cpp:
```



#### 8.1.1 Detailed Description

Implementation of [ARM7TDMISProcessor](#).

Author

Andrew D. Zonenberg

### 8.2 ARM7TDMISProcessor.h File Reference

Declaration of [ARM7TDMISProcessor](#).

This graph shows which files directly or indirectly include this file:



#### Classes

- class [ARM7TDMISProcessor](#)  
*An ARM7TDMI-S CPU core supporting the ARMv4 architecture, as seen over JTAG (no CoreSight support)*

### 8.2.1 Detailed Description

Declaration of [ARM7TDMISProcessor](#).

Author

Andrew D. Zonenberg

## 8.3 ARMAPBDevice.cpp File Reference

Implementation of [ARMAPBDevice](#).

```
#include "jtaghal.h"
#include "ARMDebugAccessPort.h"
#include "ARMDebugMemAccessPort.h"
#include "ARMAPBDevice.h"
```

Include dependency graph for ARMAPBDevice.cpp:



### 8.3.1 Detailed Description

Implementation of [ARMAPBDevice](#).

Author

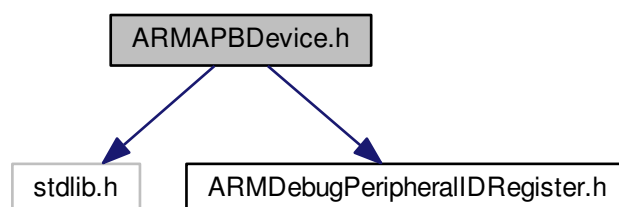
Andrew D. Zonenberg

## 8.4 ARMAPBDevice.h File Reference

Declaration of [ARMAPBDevice](#).

```
#include <stdlib.h>
#include "ARMDebugPeripheralIDRegister.h"
```

Include dependency graph for ARMAPBDevice.h:



This graph shows which files directly or indirectly include this file:





## Classes

- class [ARMAPBDevice](#)  
*A device attached to an ARM APB bus (may be a debug core or something else)*

### 8.4.1 Detailed Description

Declaration of [ARMAPBDevice](#).

#### Author

Andrew D. Zonenberg

## 8.5 ARMCoresightDevice.cpp File Reference

Base class for ARM CoreSight components on a debug APB bus.

```
#include "jtaghal.h"
#include "ARMAPBDevice.h"
#include "ARMCoresightDevice.h"
```

Include dependency graph for ARMCoresightDevice.cpp:



### 8.5.1 Detailed Description

Base class for ARM CoreSight components on a debug APB bus.

#### Author

Andrew D. Zonenberg

## 8.6 ARMCoresightDevice.h File Reference

Declaration of [ARMCoresightDevice](#).

This graph shows which files directly or indirectly include this file:



## Classes

- class [ARMCoresightDevice](#)  
*Base class for ARM CoreSight components (other than CPU cores) on a debug APB bus.*

### 8.6.1 Detailed Description

Declaration of [ARMCoreSightDevice](#).

#### Author

Andrew D. Zonenberg

## 8.7 ARM Cortex A57.cpp File Reference

Implementation of [ARM Cortex A57](#).

```
#include "jtaghal.h"
#include "DebuggableDevice.h"
#include "ARMAPBDevice.h"
#include "ARMDebugAccessPort.h"
#include "ARMDebugMemAccessPort.h"
#include "ARM Cortex A57.h"
Include dependency graph for ARM Cortex A57.cpp:
```



### 8.7.1 Detailed Description

Implementation of [ARM Cortex A57](#).

#### Author

Andrew D. Zonenberg

## 8.8 ARM Cortex A57.h File Reference

Declaration of [ARM Cortex A57](#).

This graph shows which files directly or indirectly include this file:



### Classes

- class [ARM Cortex A57](#)

*An ARM Cortex-A57 CPU core, as seen over a CoreSight APB bus.*

### 8.8.1 Detailed Description

Declaration of [ARM Cortex A9](#).

#### Author

Andrew D. Zonenberg

## 8.9 ARM Cortex A9.cpp File Reference

Implementation of [ARM Cortex A9](#).

```
#include "jtaghal.h"
#include "DebuggableDevice.h"
#include "ARMAPBDevice.h"
#include "ARMDebugAccessPort.h"
#include "ARMDebugMemAccessPort.h"
#include "ARM Cortex A9.h"
Include dependency graph for ARM Cortex A9.cpp:
```



### 8.9.1 Detailed Description

Implementation of [ARM Cortex A9](#).

#### Author

Andrew D. Zonenberg

## 8.10 ARM Cortex A9.h File Reference

Declaration of [ARM Cortex A9](#).

This graph shows which files directly or indirectly include this file:



### Classes

- class [ARM Cortex A9](#)

*An ARM Cortex-A9 CPU core, as seen over a CoreSight APB bus.*

### 8.10.1 Detailed Description

Declaration of [ARMCortexA9](#).

#### Author

Andrew D. Zonenberg

### 8.11 ARMCortexM4.cpp File Reference

Implementation of [ARMCortexM4](#).

```
#include "jtaghal.h"
#include "DebuggableDevice.h"
#include "ARMAPBDevice.h"
#include "ARMDebugAccessPort.h"
#include "ARMDebugMemAccessPort.h"
#include "ARMCortexM4.h"
Include dependency graph for ARMCortexM4.cpp:
```



#### 8.11.1 Detailed Description

Implementation of [ARMCortexM4](#).

#### Author

Andrew D. Zonenberg

### 8.12 ARMCortexM4.h File Reference

Declaration of [ARMCortexM4](#).

This graph shows which files directly or indirectly include this file:



#### Classes

- class [ARMCortexM4](#)  
*An ARM Cortex-M4 CPU core, as seen over a CoreSight APB bus.*

### 8.12.1 Detailed Description

Declaration of [ARMCortexM4](#).

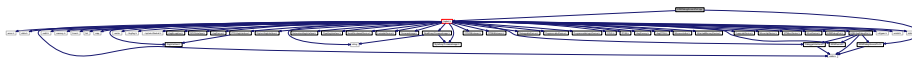
Author

Andrew D. Zonenberg

## 8.13 ARMDebugAccessPort.cpp File Reference

Implementation of [ARMDebugAccessPort](#).

```
#include "jtaghal.h"
#include "ARMDebugAccessPort.h"
Include dependency graph for ARMDebugAccessPort.cpp:
```



### 8.13.1 Detailed Description

Implementation of [ARMDebugAccessPort](#).

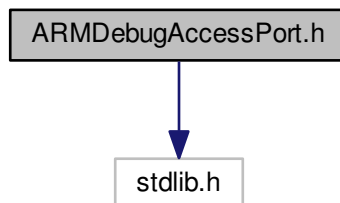
Author

Andrew D. Zonenberg

## 8.14 ARMDebugAccessPort.h File Reference

Declaration of [ARMDebugAccessPort](#).

```
#include <stdlib.h>
Include dependency graph for ARMDebugAccessPort.h:
```



This graph shows which files directly or indirectly include this file:



## Classes

- union [ARMDebugPortIDRegister](#)  
*ARM debug port identification register (see ADiv5 Architecture Specification figure 6-3)*
- class [ARMDebugAccessPort](#)  
*Base class for all access ports within an [ARMDebugPort](#).*

## Functions

- union [ARMDebugPortIDRegister](#) `__attribute__((packed))`

## Variables

- unsigned int [type](#)  
*Type of AP.*
- unsigned int [variant](#)  
*Variant of AP.*
- unsigned int [reserved\\_zero](#)  
*Reserved, SBZ.*
- unsigned int [is\\_mem\\_ap](#)  
*Class (1 = mem-AP, 0=not mem-AP)*
- unsigned int [identity](#)  
*Identity code (must be 0x3B)*
- unsigned int [continuation](#)  
*Continuation code (must be 0x4)*
- unsigned int [revision](#)  
*Revision of the AP design.*
- `uint32_t` [word](#)  
*The raw status register value.*
- class [ARMDebugAccessPort](#) `__attribute__((packed))`

### 8.14.1 Detailed Description

Declaration of [ARMDebugAccessPort](#).

#### Author

Andrew D. Zonenberg

### 8.14.2 Variable Documentation

#### 8.14.2.1 `reserved_zero`

```
unsigned int reserved_zero
```

Reserved, SBZ.

Reserved, should be zero.

### 8.14.2.2 revision

unsigned int revision

Revision of the AP design.

Implementation defined CPU revision.

### 8.14.2.3 type

unsigned int type

Type of AP.

Frame type.

Must be Xilinx7SeriesDevice::X7\_CONFIG\_FRAME\_TYPE\_2

Must be Xilinx7SeriesDevice::X7\_CONFIG\_FRAME\_TYPE\_1

Must be one of the following:

- XilinxSpartan3ADevice::S3A\_CONFIG\_FRAME\_TYPE\_1
- XilinxSpartan3ADevice::S3A\_CONFIG\_FRAME\_TYPE\_2

Must be one of the following:

- XilinxSpartan6Device::S6\_CONFIG\_FRAME\_TYPE\_1
- XilinxSpartan6Device::S6\_CONFIG\_FRAME\_TYPE\_2

Must be XilinxUltrascaleDevice::CONFIG\_FRAME\_TYPE\_2

Must be XilinxUltrascaleDevice::CONFIG\_FRAME\_TYPE\_1

### 8.14.2.4 variant

unsigned int variant

Variant of AP.

Implementation defined CPU variant.

### 8.14.2.5 word

uint32\_t word

The raw status register value.

The raw register value.

## 8.15 ARMDebugMemAccessPort.cpp File Reference

Implementation of [ARMDebugMemAccessPort](#).

```
#include "jtaghal.h"
#include "ARMAPBDevice.h"
#include "ARMDebugAccessPort.h"
#include "ARMDebugMemAccessPort.h"
#include "ARMCoreSightDevice.h"
```

Include dependency graph for ARMDebugMemAccessPort.cpp:



### 8.15.1 Detailed Description

Implementation of [ARMDebugMemAccessPort](#).

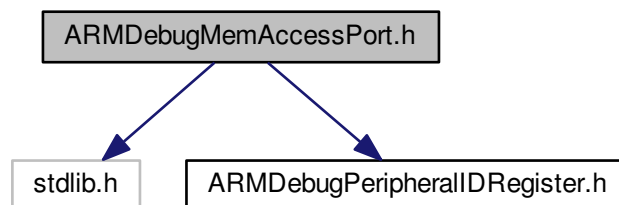
Author

Andrew D. Zonenberg

## 8.16 ARMDebugMemAccessPort.h File Reference

Declaration of [ARMDebugMemAccessPort](#).

```
#include <stdlib.h>
#include "ARMDebugPeripheralIDRegister.h"
Include dependency graph for ARMDebugMemAccessPort.h:
```



This graph shows which files directly or indirectly include this file:





## Classes

- union [ARMDebugMemAPControlStatusWord](#)  
*Contents of the CSW register in a MEM-AP (see ADIV5 Architecture Specification 7.6.4)*
- class [ARMDebugMemAccessPort](#)  
*A bridge from an [ARMDebugPort](#) to an ARM memory bus.*

## Functions

- union [ARMDebugMemAPControlStatusWord](#) **\_\_attribute\_\_** ((packed))

## Variables

- unsigned int [size](#)  
*Size of the access to perform.*
- unsigned int [reserved\\_zero\\_1](#)  
*Reserved, should be zero.*
- unsigned int [auto\\_increment](#)  
*Address increment/pack mode.*
- unsigned int [enable](#)  
*Debug port enable (RO)*
- unsigned int [busy](#)  
*Transfer in progress.*
- unsigned int [mode](#)  
*Operating mode (write as zero, read undefined)*
- unsigned int [reserved\\_zero\\_2](#)  
*Reserved, should be zero.*
- unsigned int [secure\\_priv\\_debug](#)  
*Secure privileged debug flag (not sure what this is)*
- unsigned int [bus\\_protect](#)  
*Bus access protection (implementation defined)*
- unsigned int [nonsecure\\_transfer](#)  
*Secure transfer (high=nonsecure)*
- unsigned int **reserved\_zero\_3**
- uint32\_t [word](#)  
*The raw status register value.*
- [ARMDebugMemAccessPort](#) **\_\_attribute\_\_**

### 8.16.1 Detailed Description

Declaration of [ARMDebugMemAccessPort](#).

#### Author

Andrew D. Zonenberg

### 8.16.2 Variable Documentation

### 8.16.2.1 mode

unsigned int mode

Operating mode (write as zero, read undefined)

Status of the mode bits.

## 8.17 ARMDebugPeripheralIDRegister.h File Reference

Declaration of [ARMDebugPeripheralIDRegister](#).

This graph shows which files directly or indirectly include this file:



### Classes

- class [ARMDebugPeripheralIDRegisterBits](#)  
*ADI component ID register bitfield.*
- union [ARMDebugPeripheralIDRegister](#)  
*ADI component ID register.*

### Functions

- class [ARMDebugPeripheralIDRegisterBits](#) **\_\_attribute\_\_** ((packed))

### Variables

- unsigned int [partnum](#)  
*Part number (TODO)*
- unsigned int [jep106\\_id](#)  
*JEP106 identity code.*
- unsigned int [jep106\\_used](#)  
*Indicates if JEP106 code is valid.*
- unsigned int [revnum](#)  
*Peripheral revision number.*
- unsigned int [cust\\_mod](#)  
*Customer modification ID.*
- unsigned int [revand](#)  
*Manufacturer rev number (stepping)*
- unsigned int [jep106\\_cont](#)  
*JEP106 continuation code.*
- unsigned int [log\\_4k\\_blocks](#)  
*Log2(#4K address space blocks)*
- unsigned int [reserved\\_zero](#)  
*Unmapped.*
- [ARMDebugPeripheralIDRegisterBits](#) bits  
*The bitfield.*
- uint64\_t [word](#)  
*The raw status register value.*

### 8.17.1 Detailed Description

Declaration of [ARMDebugPeripheralIDRegister](#).

#### Author

Andrew D. Zonenberg

## 8.18 ARMDebugPort.h File Reference

Declaration of [ARMDebugPort](#).

This graph shows which files directly or indirectly include this file:



### Classes

- class [ARMDebugPort](#)  
*Base class for ARM debug ports (JTAG-DP, SWJ-DP, etc)*

### Variables

- [ARMDebugPort](#) `__attribute__`

### 8.18.1 Detailed Description

Declaration of [ARMDebugPort](#).

#### Author

Andrew D. Zonenberg

## 8.19 ARMDevice.cpp File Reference

Implementation of [ARMDevice](#).

```
#include "jtagahal.h"
#include "JEDECVendorID_enum.h"
Include dependency graph for ARMDevice.cpp:
```



### 8.19.1 Detailed Description

Implementation of [ARMDevice](#).

#### Author

Andrew D. Zonenberg

## 8.20 ARMDevice.h File Reference

Declaration of [ARMDevice](#).

This graph shows which files directly or indirectly include this file:



### Classes

- class [ARMDevice](#)  
*Abstract base class for all ARM Ltd JTAG devices (ADiv5 DAP or legacy CPUs with their own JTAG TAPs)*

### Enumerations

- enum [ARM\\_IDCODES](#) { `IDCODE_ARM_DAP_JTAG = 0xBA00`, `IDCODE_ARM_7TDMI_S = 0xF1F0` }  
*JTAG part number for ARM JTAG DAP.*

### 8.20.1 Detailed Description

Declaration of [ARMDevice](#).

#### Author

Andrew D. Zonenberg

## 8.21 ARMFlashPatchBreakpoint.cpp File Reference

ARM Cortex-M Flash Patch/Breakpoint.

```
#include "jtaghal.h"
#include "ARMAPBDevice.h"
#include "ARMFlashPatchBreakpoint.h"
Include dependency graph for ARMFlashPatchBreakpoint.cpp:
```



### 8.21.1 Detailed Description

ARM Cortex-M Flash Patch/Breakpoint.

#### Author

Andrew D. Zonenberg

## 8.22 ARMFlashPatchBreakpoint.h File Reference

Declaration of [ARMFlashPatchBreakpoint](#).

This graph shows which files directly or indirectly include this file:



### Classes

- class [ARMFlashPatchBreakpoint](#)  
*Cortex-M Flash Patch/Breakpoint Unit (see ARMv7-M architecture ref C1.11)*

### 8.22.1 Detailed Description

Declaration of [ARMFlashPatchBreakpoint](#).

#### Author

Andrew D. Zonenberg

## 8.23 ARMJtagDebugPort.cpp File Reference

Implementation of [ARMJtagDebugPort](#).

```
#include "jtaghal.h"
#include "ARMDebugMemAccessPort.h"
#include "ARMAPBDevice.h"
```

Include dependency graph for ARMJtagDebugPort.cpp:



### 8.23.1 Detailed Description

Implementation of [ARMJtagDebugPort](#).

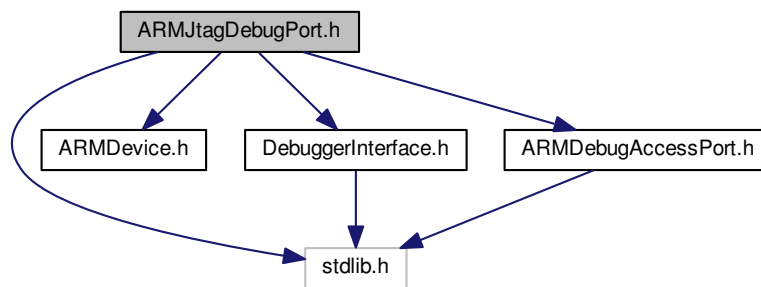
#### Author

Andrew D. Zonenberg

## 8.24 ARMJtagDebugPort.h File Reference

Declaration of [ARMJtagDebugPort](#).

```
#include <stdlib.h>
#include "ARMDevice.h"
#include "DebuggerInterface.h"
#include "ARMDebugAccessPort.h"
Include dependency graph for ARMJtagDebugPort.h:
```



This graph shows which files directly or indirectly include this file:



### Classes

- union [ARMJtagDebugPortStatusRegister](#)  
*ARM debug port status register (see ADIV5 Architecture Specification figure 6-3)*
- class [ARMJtagDebugPort](#)  
*An ARM JTAG-DP (contains one or more APs and a DP)*

### Functions

- union [ARMJtagDebugPortStatusRegister](#) **\_\_attribute\_\_** ((packed))

### Variables

- unsigned int [sticky\\_ouerrun\\_en](#)  
*Set to 1 to enable ouerrun detection.*
- unsigned int [sticky\\_ouerrun](#)  
*Sticky buffer ouerrun (if enabled)*
- unsigned int [transfer\\_mode](#)  
*Transfer mode.*
- unsigned int [sticky\\_compare](#)

- *Sticky compare bit.*  
unsigned int [sticky\\_err](#)
- *Sticky error bit.*  
unsigned int [read\\_ok](#)
- *Read status flag.*  
unsigned int [wr\\_data\\_err](#)
- *Write data error flag.*  
unsigned int [mask\\_lane](#)
- *Byte mask.*  
unsigned int [trans\\_count](#)
- *Transaction counter.*  
unsigned int [reserved\\_zero](#)
- *Reserved, should be zero.*  
unsigned int [debug\\_reset\\_req](#)
- *Debug reset request.*  
unsigned int [debug\\_reset\\_ack](#)
- *Debug reset acknowledgement.*  
unsigned int [debug\\_pwrup\\_req](#)
- *Powerup request.*  
unsigned int [debug\\_pwrup\\_ack](#)
- *Powerup acknowledgement.*  
unsigned int [sys\\_pwrup\\_req](#)
- *Powerup request.*  
unsigned int [sys\\_pwrup\\_ack](#)
- *Powerup acknowledgement.*  
uint32\_t [word](#)
- *The raw status register value.*  
[ARMJtagDebugPort](#) **\_\_attribute\_\_**

### 8.24.1 Detailed Description

Declaration of [ARMJtagDebugPort](#).

#### Author

Andrew D. Zonenberg

## 8.25 ARMv7MProcessor.cpp File Reference

Implementation of [ARMv7MProcessor](#).

```
#include "jtagahal.h"
#include "DebuggableDevice.h"
#include "ARMAPBDevice.h"
#include "ARMDebugAccessPort.h"
#include "ARMDebugMemAccessPort.h"
#include "ARMv7MProcessor.h"
Include dependency graph for ARMv7MProcessor.cpp:
```



### 8.25.1 Detailed Description

Implementation of [ARMv7MProcessor](#).

#### Author

Andrew D. Zonenberg

## 8.26 ARMv7MProcessor.h File Reference

Declaration of [ARMv7MProcessor](#).

This graph shows which files directly or indirectly include this file:



### Classes

- class [ARMv7MProcessor](#)  
An ARMv7 Cortex-M CPU core, as seen over a CoreSight APB bus.

### 8.26.1 Detailed Description

Declaration of [ARMv7MProcessor](#).

#### Author

Andrew D. Zonenberg

## 8.27 ARMv7Processor.cpp File Reference

Implementation of [ARMv7Processor](#).

```
#include "jtaghal.h"
#include "DebuggableDevice.h"
#include "ARMAPBDevice.h"
#include "ARMDebugAccessPort.h"
#include "ARMDebugMemAccessPort.h"
#include "ARMv7Processor.h"
Include dependency graph for ARMv7Processor.cpp:
```





### 8.27.1 Detailed Description

Implementation of [ARMv7Processor](#).

#### Author

Andrew D. Zonenberg

## 8.28 ARMv7Processor.h File Reference

Declaration of [ARMv7Processor](#).

This graph shows which files directly or indirectly include this file:



### Classes

- union [ARMv7DebugIDRegister](#)  
*ARM debug ID register (see ARMv7 Architecture Reference Manual, C11.11.15)*
- union [ARMv7DebugStatusControlRegister](#)  
*ARM debug status/control register (see ARMv7 Architecture Reference Manual, C11.11.20)*
- class [ARMv7Processor](#)  
*An ARMv7 Cortex-A CPU core, as seen over a CoreSight APB bus.*

### Enumerations

- enum [ARMDebugArchVersion](#) {  
[ARM\\_DEBUG\\_V6](#) = 1, [ARM\\_DEBUG\\_V6\\_P1](#) = 2, [ARM\\_DEBUG\\_V7\\_FULL](#) = 3, [ARM\\_DEBUG\\_V7\\_MIN](#) =  
4,  
[ARM\\_DEBUG\\_V7\\_P1](#) = 5 }

### Functions

- union [ARMv7DebugIDRegister](#) `__attribute__((packed))`

## Variables

- unsigned int [revision](#)  
*Implementation defined CPU revision.*
- unsigned int [variant](#)  
*Implementation defined CPU variant.*
- unsigned int [reserved](#)  
*Reserved, undefined value.*
- unsigned int [sec\\_ext](#)  
*Indicates if security extensions are implemented.*
- unsigned int [pcsr\\_legacy\\_addr](#)  
*Indicates if PCSR is present at the legacy address.*
- unsigned int [no\\_secure\\_halt](#)  
*NO secure halting debug.*
- unsigned int [has\\_dbgdevid](#)  
*True if DBGDEVID is implemented.*
- [ARMDebugArchVersion](#) [debug\\_arch\\_version](#)  
*Debug arch version.*
- unsigned int [context\\_bpoints\\_minus\\_one](#)  
*Number of breakpoints supporting context matching, zero based (0 means 1 implemented, etc)*
- unsigned int [bpoints\\_minus\\_one](#)  
*Number of breakpoints, zero based (0 means 1 implemented, etc)*
- unsigned int [wpoints\\_minus\\_one](#)  
*Number of watchpoints, zero based (0 means 1 implemented, etc)*
- uint32\_t [word](#)  
*The raw register value.*
- unsigned int [halted](#)  
*Set by the CPU when the processor is halted.*
- unsigned int [restarted](#)  
*Processor restarted flag.*
- unsigned int [entry\\_method](#)  
*Method of debug entry (TODO)*
- unsigned int [sticky\\_sync\\_abt](#)  
*Sticky sync abort.*
- unsigned int [sticky\\_async\\_abt](#)  
*Sticky async abort.*
- unsigned int [sticky\\_undef\\_instr](#)  
*Sticky undefined instruction.*
- unsigned int [reserved\\_sbz2](#)  
*Reserved.*
- unsigned int [force\\_dbg\\_ack](#)  
*Force debug acks regardless of cpu settings.*
- unsigned int [int\\_dis](#)  
*Disable interrupts.*
- unsigned int [user\\_dcc](#)  
*Enable user-mode access to the debug channel.*
- unsigned int [inst\\_txfr](#)  
*Enable instruction transfer.*
- unsigned int [halting\\_debug](#)  
*Enable halting-mode debug.*
- unsigned int [monitor\\_debug](#)

- Set high by the CPU if it allows monitor-mode debugging.*

  - unsigned int [secure\\_ni\\_debug](#)
- Set high by the CPU if it allows invasive debug in secure mode.*

  - unsigned int [deprecated](#)

*Deprecated "secure noninvasive debug" bit.*
- unsigned int [nonsec](#)

*Set high by the CPU if it is not in secure mode.*
- unsigned int [discard\\_async\\_abort](#)

*Set high to discard async aborts.*
- unsigned int [ext\\_dcc\\_mode](#)

*DCC access mode (TODO enum)*
- unsigned int [instr\\_complete](#)

*Latching instruction-complete bit for single instruction issue.*
- unsigned int [pipelined\\_advancing](#)

*Sticky "pipeline advancing" bit, set at unpredictable intervals when not halted.*
- unsigned int [tx\\_full\\_latch](#)

*Latching TX-full bit.*
- unsigned int [rx\\_full\\_latch](#)

*Latching RX-full bit.*
- unsigned int [tx\\_full](#)

*Indicates DBGDTRTX has valid data.*
- unsigned int [rx\\_full](#)

*Indicates DBGDTRRX has valid data.*
- unsigned int [reserved\\_sbz](#)

*Reserved, should be zero.*
- [ARMv7Processor](#) `__attribute__`

### 8.28.1 Detailed Description

Declaration of [ARMv7Processor](#).

#### Author

Andrew D. Zonenberg

### 8.28.2 Enumeration Type Documentation

#### 8.28.2.1 ARMDebugArchVersion

enum [ARMDebugArchVersion](#)

#### Enumerator

|                                |                                      |
|--------------------------------|--------------------------------------|
| <code>ARM_DEBUG_V6</code>      | ARMv6, v6 debug arch.                |
| <code>ARM_DEBUG_V6_P1</code>   | ARMv6, v6.1 debug arch.              |
| <code>ARM_DEBUG_V7_FULL</code> | ARMv7, v7 debug, full CP14.          |
| <code>ARM_DEBUG_V7_MIN</code>  | ARMv7, v7 debug, only baseline cp14. |
| <code>ARM_DEBUG_V7_P1</code>   | ARMv7, v7.1 debug.                   |

### 8.28.3 Variable Documentation

#### 8.28.3.1 reserved

`unsigned int reserved`

Reserved, undefined value.

Reserved.

Reserved, must be zero.

## 8.29 ARMv8Processor.cpp File Reference

Implementation of [ARMv8Processor](#).

```
#include "jtaghal.h"
#include "DebuggableDevice.h"
#include "ARMAPBDevice.h"
#include "ARMDebugAccessPort.h"
#include "ARMDebugMemAccessPort.h"
#include "ARMv8Processor.h"
Include dependency graph for ARMv8Processor.cpp:
```



### 8.29.1 Detailed Description

Implementation of [ARMv8Processor](#).

#### Author

Andrew D. Zonenberg

## 8.30 ARMv8Processor.h File Reference

Declaration of [ARMv8Processor](#).

This graph shows which files directly or indirectly include this file:



## Classes

- class [ARMv8Processor](#)  
*An ARMv8 Cortex-A CPU core, as seen over a CoreSight APB bus.*

### 8.30.1 Detailed Description

Declaration of [ARMv8Processor](#).

#### Author

Andrew D. Zonenberg

## 8.31 AttachedMemoryDevice.h File Reference

Declaration of [AttachedMemoryDevice](#).

This graph shows which files directly or indirectly include this file:



## Classes

- class [AttachedMemoryDevice](#)  
*Base classes for devices which can connect to external memory devices.*

### 8.31.1 Detailed Description

Declaration of [AttachedMemoryDevice](#).

#### Author

Andrew D. Zonenberg

## 8.32 ByteArrayFirmwareImage.cpp File Reference

Implementation of [ByteArrayFirmwareImage](#).

```
#include "jtaghal.h"
```

Include dependency graph for ByteArrayFirmwareImage.cpp:



### 8.32.1 Detailed Description

Implementation of [ByteArrayFirmwareImage](#).

Author

Andrew D. Zonenberg

### 8.33 ByteArrayFirmwareImage.h File Reference

Declaration of [ByteArrayFirmwareImage](#).

This graph shows which files directly or indirectly include this file:



#### Classes

- class [ByteArrayFirmwareImage](#)  
*Generic base class for all firmware images consisting of an array of bytes.*

### 8.33.1 Detailed Description

Declaration of [ByteArrayFirmwareImage](#).

Author

Andrew D. Zonenberg

### 8.34 CPLD.cpp File Reference

Implementation of [CPLD](#).

```
#include "jtagahal.h"
Include dependency graph for CPLD.cpp:
```



### 8.34.1 Detailed Description

Implementation of [CPLD](#).

Author

Andrew D. Zonenberg

## 8.35 CPLD.h File Reference

Declaration of [CPLD](#).

This graph shows which files directly or indirectly include this file:



### Classes

- class [CPLD](#)  
*Generic base class for all complex programmable logic devices.*

### 8.35.1 Detailed Description

Declaration of [CPLD](#).

Author

Andrew D. Zonenberg

## 8.36 CPLDBitstream.cpp File Reference

Implementation of [CPLDBitstream](#).

```
#include "jtagahal.h"
```

Include dependency graph for CPLDBitstream.cpp:



### 8.36.1 Detailed Description

Implementation of [CPLDBitstream](#).

Author

Andrew D. Zonenberg

## 8.37 CPLDBitstream.h File Reference

Declaration of [CPLDBitstream](#).

This graph shows which files directly or indirectly include this file:



## Classes

- class [CPLDBitstream](#)  
*Abstract base class for CPLD configuration bitstreams.*

### 8.37.1 Detailed Description

Declaration of [CPLDBitstream](#).

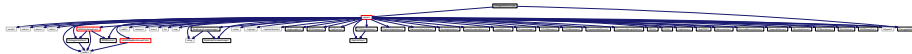
#### Author

Andrew D. Zonenberg

## 8.38 DebuggableDevice.cpp File Reference

Implementation of [DebuggableDevice](#).

```
#include "jtagahal.h"
#include "DebuggableDevice.h"
Include dependency graph for DebuggableDevice.cpp:
```



### 8.38.1 Detailed Description

Implementation of [DebuggableDevice](#).

#### Author

Andrew D. Zonenberg

## 8.39 DebuggableDevice.h File Reference

Declaration of [DebuggableDevice](#).

This graph shows which files directly or indirectly include this file:



## Classes

- class [DebuggableDevice](#)  
*Generic base class for all debuggable devices (CPU cores etc)*



### 8.39.1 Detailed Description

Declaration of [DebuggableDevice](#).

#### Author

Andrew D. Zonenberg

## 8.40 DebuggerInterface.cpp File Reference

Implementation of [DebuggerInterface](#).

```
#include "jtaghal.h"
#include "DebuggerInterface.h"
Include dependency graph for DebuggerInterface.cpp:
```



### 8.40.1 Detailed Description

Implementation of [DebuggerInterface](#).

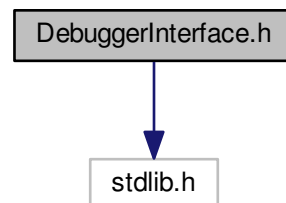
#### Author

Andrew D. Zonenberg

## 8.41 DebuggerInterface.h File Reference

Declaration of [DebuggerInterface](#).

```
#include <stdlib.h>
Include dependency graph for DebuggerInterface.h:
```



This graph shows which files directly or indirectly include this file:



## Classes

- class [DebuggerInterface](#)

*Generic base class for all debugger interfaces (may connect to multiple [DebuggableDevice](#)'s in a SoC)*

### 8.41.1 Detailed Description

Declaration of [DebuggerInterface](#).

#### Author

Andrew D. Zonenberg

## 8.42 DigilentJtagInterface.cpp File Reference

Implementation of [DigilentJtagInterface](#).

```
#include "jtaghal.h"
#include <digilent/adept/dpcdecl.h>
#include <digilent/adept/dpcdefs.h>
#include <digilent/adept/dpcutil.h>
#include <digilent/adept/djtg.h>
#include <digilent/adept/dmgr.h>
Include dependency graph for DigilentJtagInterface.cpp:
```



### 8.42.1 Detailed Description

Implementation of [DigilentJtagInterface](#).

#### Author

Andrew D. Zonenberg

## 8.43 DigilentJtagInterface.h File Reference

Declaration of [DigilentJtagInterface](#).

This graph shows which files directly or indirectly include this file:



## Classes

- class [DigilentJtagInterface](#)  
*A JTAG adapter exposed through the Digilent Adept SDK.*

### 8.43.1 Detailed Description

Declaration of [DigilentJtagInterface](#).

#### Author

Andrew D. Zonenberg

## 8.44 FirmwareImage.cpp File Reference

Implementation of [FirmwareImage](#).

```
#include "jtagahal.h"
Include dependency graph for FirmwareImage.cpp:
```



### 8.44.1 Detailed Description

Implementation of [FirmwareImage](#).

#### Author

Andrew D. Zonenberg

## 8.45 FirmwareImage.h File Reference

Declaration of [FirmwareImage](#).

This graph shows which files directly or indirectly include this file:



## Classes

- class [FirmwareImage](#)  
*Generic base class for all firmware images for any kind of programmable device.*

### 8.45.1 Detailed Description

Declaration of [FirmwareImage](#).

Author

Andrew D. Zonenberg

### 8.46 FPGA.cpp File Reference

Implementation of [FPGA](#).

```
#include "jtaghal.h"
Include dependency graph for FPGA.cpp:
```



#### 8.46.1 Detailed Description

Implementation of [FPGA](#).

Author

Andrew D. Zonenberg

### 8.47 FPGA.h File Reference

Declaration of [FPGA](#).

This graph shows which files directly or indirectly include this file:



#### Classes

- class [FPGA](#)  
*Generic base class for all field-programmable gate array devices.*

#### 8.47.1 Detailed Description

Declaration of [FPGA](#).

Author

Andrew D. Zonenberg

## 8.48 FPGABitstream.cpp File Reference

Implementation of [FPGABitstream](#).

```
#include "jtaghal.h"
```

Include dependency graph for FPGABitstream.cpp:



### 8.48.1 Detailed Description

Implementation of [FPGABitstream](#).

Author

Andrew D. Zonenberg

## 8.49 FPGABitstream.h File Reference

Declaration of [FPGABitstream](#).

This graph shows which files directly or indirectly include this file:



### Classes

- class [FPGABitstream](#)  
*Abstract base class for [FPGA](#) configuration bitstreams.*

### 8.49.1 Detailed Description

Declaration of [FPGABitstream](#).

Author

Andrew D. Zonenberg

## 8.50 FreescaleDevice.cpp File Reference

Implementation of [FreescaleDevice](#).

```
#include "jtaghal.h"
```

```
#include "JEDECVendorID_enum.h"
```

Include dependency graph for FreescaleDevice.cpp:



### 8.50.1 Detailed Description

Implementation of [FreescaleDevice](#).

#### Author

Andrew D. Zonenberg

## 8.51 FreescaleDevice.h File Reference

Declaration of [FreescaleDevice](#).

This graph shows which files directly or indirectly include this file:



### Classes

- class [FreescaleDevice](#)  
*Abstract base class for all Freescale devices (typically MCUs or parts thereof)*

### 8.51.1 Detailed Description

Declaration of [FreescaleDevice](#).

#### Author

Andrew D. Zonenberg

## 8.52 FreescaleIMXDevice.cpp File Reference

Implementation of [FreescaleIMXDevice](#).

```
#include "jtaghal.h"
#include "FreescaleIMXDevice.h"
#include "memory.h"
```

Include dependency graph for FreescaleIMXDevice.cpp:



### 8.52.1 Detailed Description

Implementation of [FreescaleIMXDevice](#).

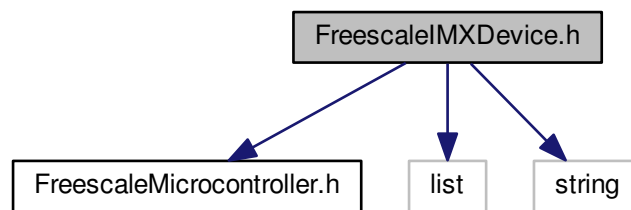
#### Author

Andrew D. Zonenberg

## 8.53 FreescaleIMXDevice.h File Reference

Declaration of [FreescaleIMXDevice](#).

```
#include "FreescaleMicrocontroller.h"
#include <list>
#include <string>
Include dependency graph for FreescaleIMXDevice.h:
```



This graph shows which files directly or indirectly include this file:



### Classes

- class [FreescaleIMXDevice](#)  
*A Freescale i.mx applications processor.*

### Enumerations

- enum `ImxDeviceIDs` { `IMX_6_SOLO` = 0x891B, `IMX_6_DUAL_LITE` = 0x891A }

#### 8.53.1 Detailed Description

Declaration of [FreescaleIMXDevice](#).

#### Author

Andrew D. Zonenberg

## 8.54 FreescaleIMXSmartDMA.cpp File Reference

Implementation of [FreescaleIMXSmartDMA](#).

```
#include "jtagahal.h"
#include "FreescaleIMXSmartDMA.h"
#include "STMicroDeviceID_enum.h"
#include "memory.h"
Include dependency graph for FreescaleIMXSmartDMA.cpp:
```



### 8.54.1 Detailed Description

Implementation of [FreescaleIMXSmartDMA](#).

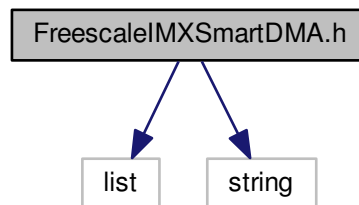
Author

Andrew D. Zonenberg

## 8.55 FreescaleIMXSmartDMA.h File Reference

Declaration of [FreescaleIMXSmartDMA](#).

```
#include <list>
#include <string>
Include dependency graph for FreescaleIMXSmartDMA.h:
```



This graph shows which files directly or indirectly include this file:





## Classes

- class [FreescaleIMXSmartDMA](#)

*The SDMA in a Freescale i.mx SoC (Chapter 55 of i.mx6 reference manual)*

### 8.55.1 Detailed Description

Declaration of [FreescaleIMXSmartDMA](#).

#### Author

Andrew D. Zonenberg

## 8.56 FreescaleMicrocontroller.cpp File Reference

Implementation of [FreescaleMicrocontroller](#).

```
#include "jtaghal.h"
```

Include dependency graph for FreescaleMicrocontroller.cpp:



### 8.56.1 Detailed Description

Implementation of [FreescaleMicrocontroller](#).

#### Author

Andrew D. Zonenberg

## 8.57 FreescaleMicrocontroller.h File Reference

Declaration of [FreescaleMicrocontroller](#).

This graph shows which files directly or indirectly include this file:



## Classes

- class [FreescaleMicrocontroller](#)

*Generic base class for all Freescale MCUs.*

### 8.57.1 Detailed Description

Declaration of [FreescaleMicrocontroller](#).

Author

Andrew D. Zonenberg

## 8.58 FTDIJtagInterface.cpp File Reference

Implementation of [FTDIJtagInterface](#).

```
#include "jtaghal.h"
#include <ftd2xx/ftd2xx.h>
Include dependency graph for FTDIJtagInterface.cpp:
```



### Macros

- `#define FTDI_VID 0x0403 /* FTDI's USB vendor ID */`
- `#define PID_232H_JTAG 0x8028 /* Product ID for azonenberg's FT232H based JTAG system */`
- `#define BIT_MODE_RESET 0x00 /* Reset the MPSSE */`
- `#define BIT_MODE_MPSSE 0x02 /* MPSSE mode */`

### Enumerations

- enum `MPSSE_Commands` {  
`MPSSE_TX_BYTES = 0x19, MPSSE_TX_BITS = 0x1b, MPSSE_TXRX_BYTES = 0x39, MPSSE_TXRX_↔`  
`BITS = 0x3b,`  
`MPSSE_TX_TMS_BITS = 0x4b, MPSSE_TXRX_TMS_BITS = 0x6b, MPSSE_SET_DATA_LOW = 0x80,`  
`MPSSE_GET_DATA_LOW = 0x81,`  
`MPSSE_SET_DATA_HIGH = 0x82, MPSSE_GET_DATA_HIGH = 0x83, MPSSE_DISABLE_LOOPBACK`  
`= 0x85, MPSSE_SET_CLKDIV = 0x86,`  
`MPSSE_FLUSH = 0x87, MPSSE_DISABLE_DIV5 = 0x8a, MPSSE_DISABLE_3PHA = 0x8d, MPSSE_D_↔`  
`UMMY_CLOCK_BITS = 0x8e,`  
`MPSSE_DUMMY_CLOCK_BYTES = 0x8f, MPSSE_DISABLE_ADAPTIVE_CLK = 0x97, MPSSE_INVA_↔`  
`LID_COMMAND = 0xAA, MPSSE_INVALID_COMMAND_RESPONSE = 0xFA }`

### 8.58.1 Detailed Description

Implementation of [FTDIJtagInterface](#).

Author

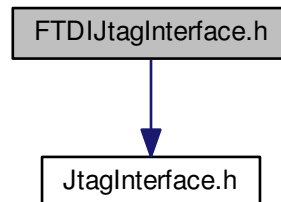
Andrew D. Zonenberg

## 8.59 FTDIJtagInterface.h File Reference

Declaration of [FTDIJtagInterface](#).

```
#include "JtagInterface.h"
```

Include dependency graph for FTDIJtagInterface.h:



This graph shows which files directly or indirectly include this file:



### Classes

- class [FTDIJtagInterface](#)

*A JTAG adapter using the FTDI chipset, accessed through libftd2xx (proprietary driver from FTDI)*

### 8.59.1 Detailed Description

Declaration of [FTDIJtagInterface](#).

#### Author

Andrew D. Zonenberg

## 8.60 GPIOInterface.cpp File Reference

Implementation of [GPIOInterface](#).

```
#include "jtagahal.h"
```

Include dependency graph for GPIOInterface.cpp:



### 8.60.1 Detailed Description

Implementation of [GPIOInterface](#).

#### Author

Andrew D. Zonenberg

## 8.61 GPIOInterface.h File Reference

Declaration of [GPIOInterface](#).

This graph shows which files directly or indirectly include this file:



### Classes

- class [GPIOInterface](#)  
*A GPIO bitbang interface. Many JTAG adapters have uncommitted GPIOs which may be used for test purposes.*

### 8.61.1 Detailed Description

Declaration of [GPIOInterface](#).

#### Author

Andrew D. Zonenberg

## 8.62 JtagDevice.cpp File Reference

Implementation of [JtagDevice](#).

```
#include "jtagahal.h"
#include "JEDECVendorID_enum.h"
#include "UserVID_enum.h"
#include "UserPID_enum.h"
Include dependency graph for JtagDevice.cpp:
```



### 8.62.1 Detailed Description

Implementation of [JtagDevice](#).

#### Author

Andrew D. Zonenberg

## 8.63 JtagDevice.h File Reference

Declaration of [JtagDevice](#).

This graph shows which files directly or indirectly include this file:



### Classes

- class [JtagDevice](#)  
*A single TAP in the JTAG chain. May not correspond 1:1 with physical silicon dies.*

### Macros

- `#define RegisterConstant(c) m_constantMap[(#c)] = c`

### 8.63.1 Detailed Description

Declaration of [JtagDevice](#).

#### Author

Andrew D. Zonenberg

## 8.64 JtagDummy.cpp File Reference

Implementation of [JtagDummy](#).

```
#include "jtagahal.h"
```

Include dependency graph for JtagDummy.cpp:



### 8.64.1 Detailed Description

Implementation of [JtagDummy](#).

Author

Andrew D. Zonenberg

### 8.65 JtagDummy.h File Reference

Declaration of [JtagDummy](#).

This graph shows which files directly or indirectly include this file:



#### Classes

- class [JtagDummy](#)  
*An unknown device (IDCODE not recognized, or no IDCODE present) on a JTAG chain.*

### 8.65.1 Detailed Description

Declaration of [JtagDummy](#).

Author

Andrew D. Zonenberg

### 8.66 JtagException.cpp File Reference

Implementation of [JtagException](#).

```
#include "jtagahal.h"
Include dependency graph for JtagException.cpp:
```



### 8.66.1 Detailed Description

Implementation of [JtagException](#).

Author

Andrew D. Zonenberg

## 8.67 JtagException.h File Reference

Declaration of [JtagException](#).

This graph shows which files directly or indirectly include this file:



### Classes

- class [JtagException](#)  
*Base class for all exceptions thrown by libjtaghal.*

### Macros

- `#define JtagExceptionWrapper(err, lib_err) JtagException(err, lib_err, __PRETTY_FUNCTION__, __FILE__, __LINE__)`  
*Wrapper for [JtagException](#) constructor that passes function, file, and line number automatically.*

### 8.67.1 Detailed Description

Declaration of [JtagException](#).

#### Author

Andrew D. Zonenberg

### 8.67.2 Macro Definition Documentation

#### 8.67.2.1 JtagExceptionWrapper

```
#define JtagExceptionWrapper(
 err,
 lib_err) JtagException(err, lib_err, __PRETTY_FUNCTION__, __FILE__, __LINE__)
```

Wrapper for [JtagException](#) constructor that passes function, file, and line number automatically.

#### Parameters

|                |                                                                              |
|----------------|------------------------------------------------------------------------------|
| <i>err</i>     | Human-readable error message. Include as much detail as reasonably possible. |
| <i>lib_err</i> | Human-readable error string returned from a library (ex: libusb)             |

## 8.68 JtagFPGA.cpp File Reference

Implementation of [JtagFPGA](#).

```
#include "jtagahal.h"
```

Include dependency graph for JtagFPGA.cpp:



### 8.68.1 Detailed Description

Implementation of [JtagFPGA](#).

Author

Andrew D. Zonenberg

## 8.69 JtagFPGA.h File Reference

Declaration of [JtagFPGA](#).

This graph shows which files directly or indirectly include this file:



### Classes

- class [JtagFPGA](#)  
*Abstract base class for all JTAG-programmed FPGAs.*

### 8.69.1 Detailed Description

Declaration of [JtagFPGA](#).

Author

Andrew D. Zonenberg

## 8.70 jtagahal.cpp File Reference

Implementation of global functions.

```
#include "jtagahal.h"
```

Include dependency graph for jtagahal.cpp:





## Functions

- bool [PeekBit](#) (const unsigned char \*data, int nbit)  
*Extracts a bit from a bit string.*
- void [PokeBit](#) (unsigned char \*data, int nbit, bool val)  
*Writes a bit to a bit string.*
- unsigned char [FlipByte](#) (unsigned char c)  
*Flips the bits in a byte.*
- void [FlipByteArray](#) (unsigned char \*data, int len)  
*Reverses an array of bytes in place without changing bit ordering.*
- void [FlipBitArray](#) (unsigned char \*data, int len)  
*Reverses the bit ordering in an array of bytes, but does not change byte ordering.*
- void [MirrorBitArray](#) (unsigned char \*data, int bitlen)  
*Reverses the bit ordering in an array of bits (need not be integer byte size)*
- void [FlipEndianArray](#) (unsigned char \*data, int len)  
*Swaps endianness in an array of 16-bit values.*
- void [FlipEndian32Array](#) (unsigned char \*data, int len)  
*Swaps endianness in an array of 32-bit values.*
- void [FlipBitAndEndianArray](#) (unsigned char \*data, int len)  
*Reverses the bit ordering in an array of bytes, as well as 16-bit endianness.*
- void [FlipBitAndEndian32Array](#) (unsigned char \*data, int len)  
*Reverses the bit ordering in an array of bytes, as well as 32-bit endianness.*
- uint16\_t [GetBigEndianUint16FromByteArray](#) (const unsigned char \*data, size\_t offset)  
*Casts (data+offset) to a uint16\_t and dereferences it with big-endian ordering.*
- uint32\_t [GetBigEndianUint32FromByteArray](#) (const unsigned char \*data, size\_t offset)  
*Casts (data+offset) to a uint32\_t and dereferences it with big-endian ordering.*
- double [GetTime](#) ()  
*Returns a timestamp suitable for performance measurement.*

### 8.70.1 Detailed Description

Implementation of global functions.

#### Author

Andrew D. Zonenberg

### 8.70.2 Function Documentation

#### 8.70.2.1 GetBigEndianUint16FromByteArray()

```
uint16_t GetBigEndianUint16FromByteArray (
 const unsigned char * data,
 size_t offset)
```

Casts (data+offset) to a uint16\_t and dereferences it with big-endian ordering.

Byte-level accesses are used to ensure safety for machines requiring aligned access to words.

### 8.70.2.2 GetBigEndianUint32FromByteArray()

```
uint32_t GetBigEndianUint32FromByteArray (
 const unsigned char * data,
 size_t offset)
```

Casts (data+offset) to a uint32\_t and dereferences it with big-endian ordering.

Byte-level accesses are used to ensure safety for machines requiring aligned access to words.

## 8.71 jtaghal.h File Reference

Main library include file.

```
#include <inttypes.h>
#include <unistd.h>
#include <stdint.h>
#include <errno.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <memory.h>
#include <time.h>
#include <list>
#include <map>
#include <string>
#include <vector>
#include "../log/log.h"
#include "../xptools/Socket.h"
#include "JtagException.h"
#include "GPIOInterface.h"
#include "JtagDevice.h"
#include "JtagInterface.h"
#include "DigilentJtagInterface.h"
#include "FTDIJtagInterface.h"
#include "NetworkedJtagInterface.h"
#include "PipeJtagInterface.h"
#include "SerialNumberedDevice.h"
#include "LockableDevice.h"
#include "FirmwareImage.h"
#include "ByteArrayFirmwareImage.h"
#include "RawBinaryFirmwareImage.h"
#include "CPLDBitstream.h"
#include "FPGABitstream.h"
#include "DebuggerInterface.h"
#include "DebuggableDevice.h"
#include "ProgrammableDevice.h"
#include "ProgrammableLogicDevice.h"
#include "CPLD.h"
#include "FPGA.h"
#include "JtagDummy.h"
#include "JtagFPGA.h"
#include "Microcontroller.h"
#include "AttachedMemoryDevice.h"
#include "ARMDevice.h"
```

```
#include "FreescaleDevice.h"
#include "MicrochipDevice.h"
#include "STMicroDevice.h"
#include "XilinxDevice.h"
#include "ARMDebugPort.h"
#include "ARMJtagDebugPort.h"
#include "ARMDebugMemAccessPort.h"
#include "ARMAPBDevice.h"
#include "ARMCoreSightDevice.h"
#include "ARMFlashPatchBreakpoint.h"
#include "ARMv7Processor.h"
#include "ARMv8Processor.h"
#include "ARMv7MProcessor.h"
#include "ARMCortexA57.h"
#include "ARMCortexA9.h"
#include "ARMCortexM4.h"
#include "ARM7TDMISProcessor.h"
#include "FreescaleMicrocontroller.h"
#include "FreescaleIMXDevice.h"
#include "FreescaleIMXSmartDMA.h"
#include "MicrochipMicrocontroller.h"
#include "STMicroMicrocontroller.h"
#include "STM32Device.h"
#include "XilinxCPLD.h"
#include "XilinxCPLDBitstream.h"
#include "XilinxCoolRunnerIIDevice.h"
#include "XilinxFPGABitstream.h"
#include "Xilinx3DFPGABitstream.h"
#include "XilinxFPGA.h"
#include "Xilinx7SeriesDevice.h"
#include "XilinxUltrascaleDevice.h"
#include "XilinxSpartan6Device.h"
#include "XilinxSpartan3ADevice.h"
```

## Macros

- #define **\_\_STDC\_FORMAT\_MACROS**
- #define **ZFILE\_DESCRIPTOR** int

## Functions

- bool [PeekBit](#) (const unsigned char \*data, int nbit)  
*Extracts a bit from a bit string.*
- void [PokeBit](#) (unsigned char \*data, int nbit, bool val)  
*Writes a bit to a bit string.*
- unsigned char [FlipByte](#) (unsigned char c)  
*Flips the bits in a byte.*
- void [FlipByteArray](#) (unsigned char \*data, int len)  
*Reverses an array of bytes in place without changing bit ordering.*
- void [FlipBitArray](#) (unsigned char \*data, int len)  
*Reverses the bit ordering in an array of bytes, but does not change byte ordering.*
- void [FlipEndianArray](#) (unsigned char \*data, int len)  
*Swaps endianness in an array of 16-bit values.*

- void [FlipEndian32Array](#) (unsigned char \*data, int len)  
*Swaps endianness in an array of 32-bit values.*
- void [FlipBitAndEndianArray](#) (unsigned char \*data, int len)  
*Reverses the bit ordering in an array of bytes, as well as 16-bit endianness.*
- void [FlipBitAndEndian32Array](#) (unsigned char \*data, int len)  
*Reverses the bit ordering in an array of bytes, as well as 32-bit endianness.*
- void [MirrorBitArray](#) (unsigned char \*data, int bitlen)  
*Reverses the bit ordering in an array of bits (need not be integer byte size)*
- uint16\_t [GetBigEndianUint16FromByteArray](#) (const unsigned char \*data, size\_t offset)  
*Casts (data+offset) to a uint16\_t and dereferences it with big-endian ordering.*
- uint32\_t [GetBigEndianUint32FromByteArray](#) (const unsigned char \*data, size\_t offset)  
*Casts (data+offset) to a uint32\_t and dereferences it with big-endian ordering.*
- double [GetTime](#) ()  
*Returns a timestamp suitable for performance measurement.*

### 8.71.1 Detailed Description

Main library include file.

#### Author

Andrew D. Zonenberg

### 8.71.2 Function Documentation

#### 8.71.2.1 GetBigEndianUint16FromByteArray()

```
uint16_t GetBigEndianUint16FromByteArray (
 const unsigned char * data,
 size_t offset)
```

Casts (data+offset) to a uint16\_t and dereferences it with big-endian ordering.

Byte-level accesses are used to ensure safety for machines requiring aligned access to words.

#### 8.71.2.2 GetBigEndianUint32FromByteArray()

```
uint32_t GetBigEndianUint32FromByteArray (
 const unsigned char * data,
 size_t offset)
```

Casts (data+offset) to a uint32\_t and dereferences it with big-endian ordering.

Byte-level accesses are used to ensure safety for machines requiring aligned access to words.

## 8.72 JtagInterface.cpp File Reference

Implementation of [JtagInterface](#).

```
#include "jtaghal.h"
```

Include dependency graph for JtagInterface.cpp:



### 8.72.1 Detailed Description

Implementation of [JtagInterface](#).

Author

Andrew D. Zonenberg

## 8.73 JtagInterface.h File Reference

Declaration of [JtagInterface](#).

This graph shows which files directly or indirectly include this file:



### Classes

- class [JtagInterface](#)  
*Abstract representation of a JTAG adapter.*

### 8.73.1 Detailed Description

Declaration of [JtagInterface](#).

Author

Andrew D. Zonenberg

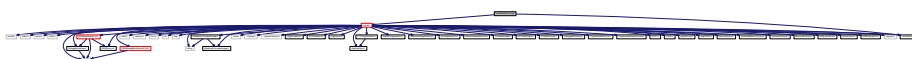
## 8.74 LockableDevice.cpp File Reference

Implementation of [LockableDevice](#).

```
#include "jtaghal.h"
```

```
#include "LockableDevice.h"
```

Include dependency graph for LockableDevice.cpp:



### 8.74.1 Detailed Description

Implementation of [LockableDevice](#).

#### Author

Andrew D. Zonenberg

## 8.75 LockableDevice.h File Reference

Declaration of [LockableDevice](#).

This graph shows which files directly or indirectly include this file:



### Classes

- class [UncertainBoolean](#)  
*A boolean value with an attached level of uncertainty.*
- class [LockableDevice](#)  
*Generic base class for all devices which have some kind of read/write protection.*

### 8.75.1 Detailed Description

Declaration of [LockableDevice](#).

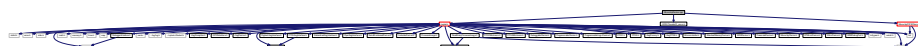
#### Author

Andrew D. Zonenberg

## 8.76 MicrochipDevice.cpp File Reference

Implementation of [MicrochipDevice](#).

```
#include "jtaghal.h"
#include "JEDECVendorID_enum.h"
#include "MicrochipPIC32Device.h"
Include dependency graph for MicrochipDevice.cpp:
```



### 8.76.1 Detailed Description

Implementation of [MicrochipDevice](#).

#### Author

Andrew D. Zonenberg

## 8.77 MicrochipDevice.h File Reference

Declaration of [MicrochipDevice](#).

This graph shows which files directly or indirectly include this file:



### Classes

- class [MicrochipDevice](#)  
*Abstract base class for all Microchip devices (typically MCUs)*

### 8.77.1 Detailed Description

Declaration of [MicrochipDevice](#).

#### Author

Andrew D. Zonenberg

## 8.78 MicrochipMicrocontroller.cpp File Reference

Implementation of [MicrochipMicrocontroller](#).

```
#include "jtaghal.h"
```

Include dependency graph for MicrochipMicrocontroller.cpp:



### 8.78.1 Detailed Description

Implementation of [MicrochipMicrocontroller](#).

#### Author

Andrew D. Zonenberg

## 8.79 MicrochipMicrocontroller.h File Reference

Declaration of [MicrochipMicrocontroller](#).

This graph shows which files directly or indirectly include this file:



### Classes

- class [MicrochipMicrocontroller](#)  
*Generic base class for all Microchip MCUs.*

### 8.79.1 Detailed Description

Declaration of [MicrochipMicrocontroller](#).

#### Author

Andrew D. Zonenberg

## 8.80 MicrochipPIC32Device.cpp File Reference

Implementation of [MicrochipPIC32Device](#).

```
#include "jtaghal.h"
#include "MicrochipPIC32Device.h"
#include "memory.h"
Include dependency graph for MicrochipPIC32Device.cpp:
```



### 8.80.1 Detailed Description

Implementation of [MicrochipPIC32Device](#).

#### Author

Andrew D. Zonenberg

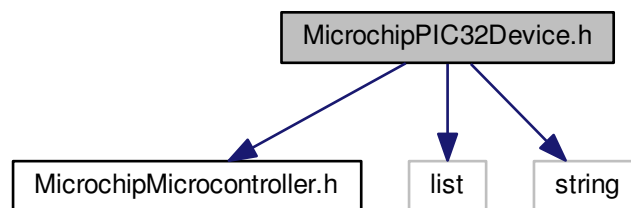


## 8.81 MicrochipPIC32Device.h File Reference

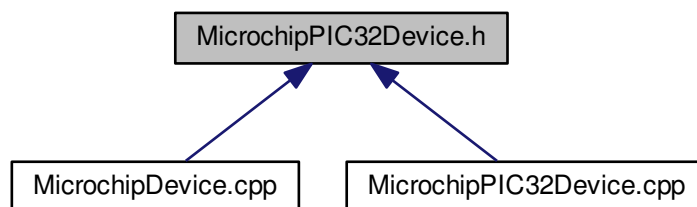
Declaration of [MicrochipPIC32Device](#).

```
#include "MicrochipMicrocontroller.h"
#include <list>
#include <string>
```

Include dependency graph for MicrochipPIC32Device.h:



This graph shows which files directly or indirectly include this file:



### Classes

- union [MicrochipPIC32DeviceStatusRegister](#)  
*Status register for a Microchip PIC32 device.*
- union [EjtagImplementationCodeRegister](#)  
*MIPS EJTAG implementation register.*
- union [EjtagControlRegister](#)  
*PIC32 EJTAG control register.*
- struct [MicrochipPIC32DeviceInfo](#)  
*Internal data structure storing properties of a single SKU in the PIC32 family.*
- class [MicrochipPIC32Device](#)  
*A Microchip PIC32 microcontroller (MX, MZ, MM, etc)*

## Functions

- union [MicrochipPIC32DeviceStatusRegister](#) `__attribute__((packed))`

## Variables

- unsigned int `reset_active`
- unsigned int `flash_en`
- unsigned int `flash_busy`
- unsigned int `cfg_rdy`
- unsigned int `reserved2`
- unsigned int `nvm_error`
- unsigned int `reserved1`
- unsigned int `code_protect_off`
- uint8\_t `word`  
*The raw status register value.*
- unsigned int `processor_is_64`
- unsigned int `no_ejtag_dma`
- unsigned int `mips16_supported`
- unsigned int `reserved3`
- unsigned int `asid_size`
- unsigned int `reserved4`
- unsigned int `dint_supported`
- unsigned int `reserved5`
- unsigned int `r3k_priv`
- unsigned int `ejtag_version`
- unsigned int `debug_mode`
- unsigned int `debug_irq`
- unsigned int `debug_vector_pos`
- unsigned int `probe_enable`
- unsigned int `proc_reset`
- unsigned int `proc_access`
- unsigned int `proc_we`
- unsigned int `periph_reset`
- unsigned int `bus_halted`
- unsigned int `low_power`
- unsigned int `vpe_disable`
- unsigned int `access_size`
- unsigned int `reset_occurred`
- struct [MicrochipPIC32DeviceInfo](#) `__attribute__((packed))`

### 8.81.1 Detailed Description

Declaration of [MicrochipPIC32Device](#).

#### Author

Andrew D. Zonenberg

## 8.82 Microcontroller.cpp File Reference

Implementation of [Microcontroller](#).

```
#include "jtagahal.h"
```

Include dependency graph for Microcontroller.cpp:



### 8.82.1 Detailed Description

Implementation of [Microcontroller](#).

Author

Andrew D. Zonenberg

## 8.83 Microcontroller.h File Reference

Declaration of [Microcontroller](#).

This graph shows which files directly or indirectly include this file:



### Classes

- class [Microcontroller](#)  
*Generic base class for all microcontrollers.*

### 8.83.1 Detailed Description

Declaration of [Microcontroller](#).

Author

Andrew D. Zonenberg

## 8.84 NetworkedJtagInterface.cpp File Reference

Implementation of [NetworkedJtagInterface](#).

```
#include "jtagahal.h"
```

```
#include "jtagd_opcodes_enum.h"
```

Include dependency graph for NetworkedJtagInterface.cpp:



### 8.84.1 Detailed Description

Implementation of [NetworkedJtagInterface](#).

Author

Andrew D. Zonenberg

### 8.85 NetworkedJtagInterface.h File Reference

Declaration of [NetworkedJtagInterface](#).

This graph shows which files directly or indirectly include this file:



#### Classes

- class [NetworkedJtagInterface](#)  
*Thin wrapper around TCP sockets for talking to a jtagd instance.*

### 8.85.1 Detailed Description

Declaration of [NetworkedJtagInterface](#).

Author

Andrew D. Zonenberg

### 8.86 PipeJtagInterface.cpp File Reference

Implementation of [PipeJtagInterface](#).

```
#include "jtagahal.h"
#include "jtagd_opcodes_enum.h"
Include dependency graph for PipeJtagInterface.cpp:
```



### 8.86.1 Detailed Description

Implementation of [PipeJtagInterface](#).

Author

Andrew D. Zonenberg

## 8.87 PipeJtagInterface.h File Reference

Declaration of [PipeJtagInterface](#).

This graph shows which files directly or indirectly include this file:



### Classes

- class [PipeJtagInterface](#)  
*Thin wrapper around pipes for talking to an openfpga JtagPipeBridge.*

### 8.87.1 Detailed Description

Declaration of [PipeJtagInterface](#).

#### Author

Andrew D. Zonenberg

## 8.88 ProgrammableDevice.cpp File Reference

Implementation of [ProgrammableDevice](#).

```
#include "jtagahal.h"
Include dependency graph for ProgrammableDevice.cpp:
```



### 8.88.1 Detailed Description

Implementation of [ProgrammableDevice](#).

#### Author

Andrew D. Zonenberg

## 8.89 ProgrammableDevice.h File Reference

Declaration of [ProgrammableDevice](#).

This graph shows which files directly or indirectly include this file:



## Classes

- class [ProgrammableDevice](#)  
*Generic base class for all programmable devices (PLD, MCU, flash, etc)*

### 8.89.1 Detailed Description

Declaration of [ProgrammableDevice](#).

#### Author

Andrew D. Zonenberg

## 8.90 ProgrammableLogicDevice.cpp File Reference

Implementation of [ProgrammableLogicDevice](#).

```
#include "jtaghal.h"
Include dependency graph for ProgrammableLogicDevice.cpp:
```



### 8.90.1 Detailed Description

Implementation of [ProgrammableLogicDevice](#).

#### Author

Andrew D. Zonenberg

## 8.91 ProgrammableLogicDevice.h File Reference

Declaration of [ProgrammableLogicDevice](#).

This graph shows which files directly or indirectly include this file:



## Classes

- class [ProgrammableLogicDevice](#)  
*Generic base class for all programmable logic devices (FPGA and CPLD)*

### 8.91.1 Detailed Description

Declaration of [ProgrammableLogicDevice](#).

Author

Andrew D. Zonenberg

## 8.92 RawBinaryFirmwareImage.cpp File Reference

Implementation of [RawBinaryFirmwareImage](#).

```
#include "jtaghal.h"
Include dependency graph for RawBinaryFirmwareImage.cpp:
```



### 8.92.1 Detailed Description

Implementation of [RawBinaryFirmwareImage](#).

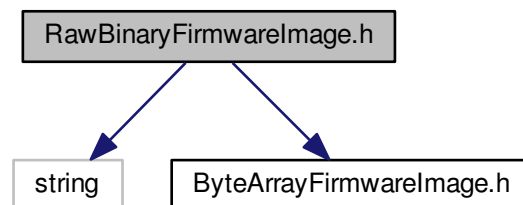
Author

Andrew D. Zonenberg

## 8.93 RawBinaryFirmwareImage.h File Reference

Declaration of [RawBinaryFirmwareImage](#).

```
#include <string>
#include "ByteArrayFirmwareImage.h"
Include dependency graph for RawBinaryFirmwareImage.h:
```



This graph shows which files directly or indirectly include this file:



## Classes

- class [RawBinaryFirmwareImage](#)  
*Raw binary firmware image loaded from a file.*

### 8.93.1 Detailed Description

Declaration of [RawBinaryFirmwareImage](#).

#### Author

Andrew D. Zonenberg

## 8.94 SerialNumberedDevice.cpp File Reference

Implementation of [SerialNumberedDevice](#).

```
#include "jtaghal.h"
```

Include dependency graph for SerialNumberedDevice.cpp:



### 8.94.1 Detailed Description

Implementation of [SerialNumberedDevice](#).

#### Author

Andrew D. Zonenberg

## 8.95 SerialNumberedDevice.h File Reference

Declaration of [SerialNumberedDevice](#).

This graph shows which files directly or indirectly include this file:



## Classes

- class [SerialNumberedDevice](#)  
*Abstract base class for all devices that have a unique die serial number.*



### 8.95.1 Detailed Description

Declaration of [SerialNumberedDevice](#).

Author

Andrew D. Zonenberg

## 8.96 STM32Device.cpp File Reference

Implementation of [STM32Device](#).

```
#include "jtaghal.h"
#include "STM32Device.h"
#include "STMicroDeviceID_enum.h"
#include "memory.h"
```

Include dependency graph for STM32Device.cpp:



### 8.96.1 Detailed Description

Implementation of [STM32Device](#).

Author

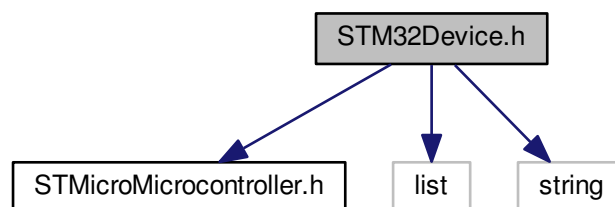
Andrew D. Zonenberg

## 8.97 STM32Device.h File Reference

Declaration of [STM32Device](#).

```
#include "STMicroMicrocontroller.h"
#include <list>
#include <string>
```

Include dependency graph for STM32Device.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [STM32Device](#)  
A STM32 microcontroller.

### 8.97.1 Detailed Description

Declaration of [STM32Device](#).

#### Author

Andrew D. Zonenberg

## 8.98 STMIODevice.cpp File Reference

Implementation of [STMIODevice](#).

```
#include "jtaghal.h"
#include "JEDECVendorID_enum.h"
#include "STMIODeviceID_enum.h"
Include dependency graph for STMIODevice.cpp:
```



### 8.98.1 Detailed Description

Implementation of [STMIODevice](#).

#### Author

Andrew D. Zonenberg

## 8.99 STMIODevice.h File Reference

Declaration of [STMIODevice](#).

This graph shows which files directly or indirectly include this file:



## Classes

- class [STMIODevice](#)  
Abstract base class for all STMIO devices.

### 8.99.1 Detailed Description

Declaration of [STMicroDevice](#).

Author

Andrew D. Zonenberg

## 8.100 STMicromicrocontroller.cpp File Reference

Implementation of [STMicromicrocontroller](#).

```
#include "jtaghal.h"
Include dependency graph for STMicromicrocontroller.cpp:
```



### 8.100.1 Detailed Description

Implementation of [STMicromicrocontroller](#).

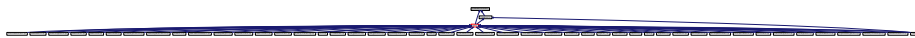
Author

Andrew D. Zonenberg

## 8.101 STMicromicrocontroller.h File Reference

Declaration of [STMicromicrocontroller](#).

This graph shows which files directly or indirectly include this file:



### Classes

- class [STMicromicrocontroller](#)  
*Generic base class for all STMicromicro MCUs.*

### 8.101.1 Detailed Description

Declaration of [STMicromicrocontroller](#).

Author

Andrew D. Zonenberg

## 8.102 Xilinx3DFPGABitstream.cpp File Reference

Implementation of [Xilinx3DFPGABitstream](#).

```
#include "jtagahal.h"
```

Include dependency graph for Xilinx3DFPGABitstream.cpp:



### 8.102.1 Detailed Description

Implementation of [Xilinx3DFPGABitstream](#).

Author

Andrew D. Zonenberg

## 8.103 Xilinx3DFPGABitstream.h File Reference

Declaration of [Xilinx3DFPGABitstream](#).

This graph shows which files directly or indirectly include this file:



### Classes

- class [Xilinx3DFPGABitstream](#)

*A bitstream for Xilinx 3D FPGAs (multiple dies on a passive interposer, each with their own bitstream)*

### 8.103.1 Detailed Description

Declaration of [Xilinx3DFPGABitstream](#).

Author

Andrew D. Zonenberg

## 8.104 Xilinx7SeriesDevice.cpp File Reference

Implementation of [Xilinx7SeriesDevice](#).

```
#include "jtagahal.h"
```

Include dependency graph for Xilinx7SeriesDevice.cpp:



### 8.104.1 Detailed Description

Implementation of [Xilinx7SeriesDevice](#).

#### Author

Andrew D. Zonenberg

## 8.105 Xilinx7SeriesDevice.h File Reference

Declaration of [Xilinx7SeriesDevice](#).

This graph shows which files directly or indirectly include this file:



### Classes

- union [Xilinx7SeriesDeviceConfigurationFrame](#)  
*7-series configuration frame (see UG470 page 87)*
- union [Xilinx7SeriesDeviceStatusRegister](#)  
*7-series status register (see UG470 table 5-28)*
- class [Xilinx7SeriesDevice](#)  
*A Xilinx 7-series FPGA device.*

### Functions

- union [MicrochipPIC32DeviceStatusRegister](#) **\_\_attribute\_\_** ((packed))
  -
- ```

struct {
    unsigned int count:11
        Count field.
    unsigned int reserved:2
        Reserved, must be zero.
    unsigned int reg\_addr:14
        Register address.
    unsigned int op:2
        Opcode.
    unsigned int type:3
        Frame type.
} __attribute__ ((packed)) bits

```

Variables

- unsigned int `count`
Count field.
- unsigned int `reserved`
Reserved, must be zero.
- unsigned int `reg_addr`
Register address.
- unsigned int `op`
Opcode.
- unsigned int `type`
Frame type.
- uint32_t `word`
The raw configuration word.
- unsigned int `crc_err`
Indicates that the device failed to configure due to a CRC error.
- unsigned int `part_secured`
Indicates that the device is in secure mode (encrypted bitstream)
- unsigned int `mmcm_lock`
Indicates MMCMs are locked.
- unsigned int `dci_match`
Indicates DCI is matched.
- unsigned int `eos`
End-of-Startup signal.
- unsigned int `gts_cfg_b`
Status of GTS_CFG net.
- unsigned int `gwe`
Status of GWE net.
- unsigned int `ghigh_b`
Status of GHIGH_B net.
- unsigned int `mode_pins`
Status of mode pins.
- unsigned int `init_complete`
Internal init-finished signal.
- unsigned int `init_b`
Status of INIT_B pin.
- unsigned int `release_done`
Indicates DONE was released.
- unsigned int `done`
Actual value on DONE pin.
- unsigned int `id_error`
Indicates an ID code error occurred (write with wrong bitstream)
- unsigned int `dec_error`
Decryption error.
- unsigned int `xadc_over_temp`
Indicates board is too hot.
- unsigned int `startup_state`
Status of startup state machine.
- unsigned int `reserved_1`
Reserved.
- unsigned int `bus_width`
Config bus width (see table 5-26)
- unsigned int `reserved_2`
Reserved.
- `Xilinx7SeriesDevice` `__attribute__`

8.105.1 Detailed Description

Declaration of [Xilinx7SeriesDevice](#).

Author

Andrew D. Zonenberg

8.105.2 Variable Documentation

8.105.2.1 count

```
unsigned int count
```

Count field.

- Type 1 packets: word count
- Type 2 packets: don't care

8.105.2.2 done

```
unsigned int done
```

Actual value on DONE pin.

Status of the DONE pin.

True if configured.

8.105.2.3 gts_cfg_b

```
unsigned int gts_cfg_b
```

Status of GTS_CFG net.

Status of global tristate net.

8.105.2.4 gwe

```
unsigned int gwe
```

Status of GWE net.

Status of global write-enable net.

8.105.2.5 init_b

unsigned int init_b

Status of INIT_B pin.

Status of the INIT_B pin.

8.105.2.6 mmcm_lock

unsigned int mmcm_lock

Indicates MMCMs are locked.

Indicates MMCMs and PLLs are locked.

8.105.2.7 op

unsigned int op

Opcode.

Must be one of the following:

- Xilinx7SeriesDevice::X7_CONFIG_OP_NOP
- Xilinx7SeriesDevice::X7_CONFIG_OP_READ
- Xilinx7SeriesDevice::X7_CONFIG_OP_WRITE

Must be zero

Must be one of the following:

- XilinxSpartan3ADevice::S3_CONFIG_OP_NOP
- XilinxSpartan3ADevice::S3_CONFIG_OP_READ
- XilinxSpartan3ADevice::S3_CONFIG_OP_WRITE

Must be one of the following:

- XilinxSpartan6Device::S6_CONFIG_OP_NOP
- XilinxSpartan6Device::S6_CONFIG_OP_READ
- XilinxSpartan6Device::S6_CONFIG_OP_WRITE

Must be one of the following:

- XilinxUltrascaleDevice::CONFIG_OP_NOP
- XilinxUltrascaleDevice::CONFIG_OP_READ
- XilinxUltrascaleDevice::CONFIG_OP_WRITE

8.105.2.8 type

unsigned int type

Frame type.

Must be Xilinx7SeriesDevice::X7_CONFIG_FRAME_TYPE_1

Must be Xilinx7SeriesDevice::X7_CONFIG_FRAME_TYPE_2

8.105.2.9 word

uint32_t word

The raw configuration word.

The raw status register value.

8.106 XilinxCoolRunnerIIDevice.cpp File Reference

Implementation of [XilinxCoolRunnerIIDevice](#).

```
#include "jtaghal.h"
#include "XilinxCoolRunnerIIDevice.h"
#include "XilinxCPLDBitstream.h"
#include "memory.h"
Include dependency graph for XilinxCoolRunnerIIDevice.cpp:
```



8.106.1 Detailed Description

Implementation of [XilinxCoolRunnerIIDevice](#).

Author

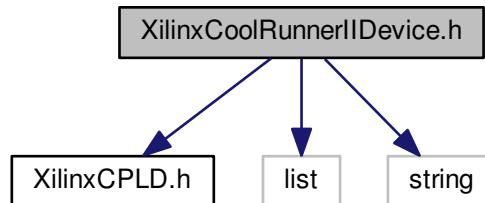
Andrew D. Zonenberg

8.107 XilinxCoolRunnerIIDevice.h File Reference

Declaration of [XilinxCoolRunnerIIDevice](#).

```
#include "XilinxCPLD.h"
#include <list>
#include <string>
```

Include dependency graph for XilinxCoolRunnerIIDevice.h:



This graph shows which files directly or indirectly include this file:



Classes

- union [XilinxCoolRunnerIIDeviceStatusRegister](#)
Status register for a Xilinx CoolRunner-II device.
- class [XilinxCoolRunnerIIDevice](#)
A Xilinx CoolRunner-II device.

Functions

- union [MicrochipPIC32DeviceStatusRegister](#) **__attribute__** ((packed))
- ```

struct {
 unsigned int padding_one:2
 Constant '01'.
 unsigned int done:1
 True if configured.
 unsigned int sec:1
 True if secured.
 unsigned int isc_en:1
 True if in ISC_ENABLE state.
 unsigned int isc_dis:1
 True if in ISC_DISABLE state.
 unsigned int padding_zero:2
 Constant '00'.
} __attribute__ ((packed)) bits

```

## Variables

- unsigned int [padding\\_one](#)  
*Constant '01'.*
- unsigned int [done](#)  
*True if configured.*
- unsigned int [sec](#)  
*True if secured.*
- unsigned int [isc\\_en](#)  
*True if in ISC\_ENABLE state.*
- unsigned int [isc\\_dis](#)  
*True if in ISC\_DISABLE state.*
- unsigned int [padding\\_zero](#)  
*Constant '00'.*
- uint8\_t [word](#)  
*The raw status register value.*
- [XilinxCoolRunnerIIDevice](#) `__attribute__`

### 8.107.1 Detailed Description

Declaration of [XilinxCoolRunnerIIDevice](#).

#### Author

Andrew D. Zonenberg

## 8.108 XilinxCPLD.cpp File Reference

Implementation of [XilinxCPLD](#).

```
#include "jtaghal.h"
```

Include dependency graph for XilinxCPLD.cpp:



### 8.108.1 Detailed Description

Implementation of [XilinxCPLD](#).

#### Author

Andrew D. Zonenberg

## 8.109 XilinxCPLD.h File Reference

Declaration of [XilinxCPLD](#).

This graph shows which files directly or indirectly include this file:



### Classes

- class [XilinxCPLD](#)  
*Generic base class for all Xilinx CPLD devices.*

### 8.109.1 Detailed Description

Declaration of [XilinxCPLD](#).

#### Author

Andrew D. Zonenberg

## 8.110 XilinxCPLDBitstream.cpp File Reference

Implementation of [XilinxCPLDBitstream](#).

```
#include "jtagahal.h"
#include "XilinxCPLDBitstream.h"
#include <stdio.h>
Include dependency graph for XilinxCPLDBitstream.cpp:
```



### 8.110.1 Detailed Description

Implementation of [XilinxCPLDBitstream](#).

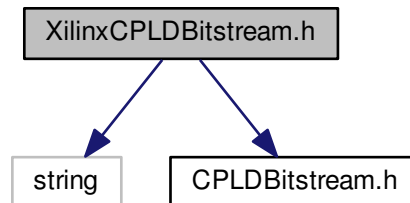
#### Author

Andrew D. Zonenberg

## 8.111 XilinxCPLDBitstream.h File Reference

Declaration of [XilinxCPLDBitstream](#).

```
#include <string>
#include "CPLDBitstream.h"
Include dependency graph for XilinxCPLDBitstream.h:
```



This graph shows which files directly or indirectly include this file:



### Classes

- class [XilinxCPLDBitstream](#)  
*A bitstream for Xilinx CPLDs.*

#### 8.111.1 Detailed Description

Declaration of [XilinxCPLDBitstream](#).

#### Author

Andrew D. Zonenberg

## 8.112 XilinxDevice.cpp File Reference

Implementation of [XilinxDevice](#).

```
#include "jtagahal.h"
#include "JEDECVendorID_enum.h"
#include "XilinxDeviceID_enum.h"
Include dependency graph for XilinxDevice.cpp:
```



### 8.112.1 Detailed Description

Implementation of [XilinxDevice](#).

#### Author

Andrew D. Zonenberg

### 8.113 XilinxDevice.h File Reference

Declaration of [XilinxDevice](#).

This graph shows which files directly or indirectly include this file:



#### Classes

- class [XilinxDevice](#)  
*Abstract base class for all Xilinx devices (FPGA, CPLD, flash, etc)*

### 8.113.1 Detailed Description

Declaration of [XilinxDevice](#).

#### Author

Andrew D. Zonenberg

### 8.114 XilinxFPGA.cpp File Reference

Implementation of [XilinxFPGA](#).

```
#include "jtagahal.h"
Include dependency graph for XilinxFPGA.cpp:
```



### 8.114.1 Detailed Description

Implementation of [XilinxFPGA](#).

#### Author

Andrew D. Zonenberg

## 8.115 XilinxFPGA.h File Reference

Declaration of [XilinxFPGA](#).

This graph shows which files directly or indirectly include this file:



### Classes

- class [XilinxFPGA](#)  
*Abstract base class for all Xilinx FPGAs.*

### 8.115.1 Detailed Description

Declaration of [XilinxFPGA](#).

Author

Andrew D. Zonenberg

## 8.116 XilinxFPGABitstream.cpp File Reference

Implementation of [XilinxFPGABitstream](#).

```
#include "jtagahal.h"
Include dependency graph for XilinxFPGABitstream.cpp:
```



### 8.116.1 Detailed Description

Implementation of [XilinxFPGABitstream](#).

Author

Andrew D. Zonenberg

## 8.117 XilinxFPGABitstream.h File Reference

Declaration of [XilinxFPGABitstream](#).

This graph shows which files directly or indirectly include this file:



## Classes

- class [XilinxFPGABitstream](#)  
*A bitstream for Xilinx FPGAs.*

### 8.117.1 Detailed Description

Declaration of [XilinxFPGABitstream](#).

#### Author

Andrew D. Zonenberg

## 8.118 XilinxSpartan3ADevice.cpp File Reference

Implementation of [XilinxSpartan3ADevice](#).

```
#include "jtaghal.h"
#include <stdio.h>
#include <memory.h>
#include "XilinxSpartan3ADevice.h"
#include "XilinxFPGABitstream.h"
Include dependency graph for XilinxSpartan3ADevice.cpp:
```



### 8.118.1 Detailed Description

Implementation of [XilinxSpartan3ADevice](#).

#### Author

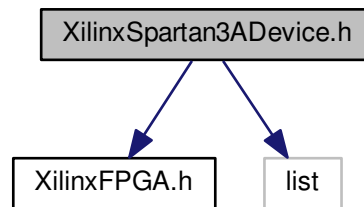
Andrew D. Zonenberg

## 8.119 XilinxSpartan3ADevice.h File Reference

Declaration of [XilinxSpartan3ADevice](#).



```
#include "XilinxFPGA.h"
#include <list>
Include dependency graph for XilinxSpartan3ADevice.h:
```



This graph shows which files directly or indirectly include this file:



## Classes

- union [XilinxSpartan3ADeviceConfigurationFrame](#)  
*Spartan-3A configuration frame header (see UG332 page 323)*
- union [XilinxSpartan3ADeviceStatusRegister](#)  
*Spartan-3A status register (see UG332 table 17-13, pages 327-328)*
- class [XilinxSpartan3ADevice](#)  
*A Xilinx Spartan-3A FPGA device.*

## Functions

- union [MicrochipPIC32DeviceStatusRegister](#) **\_\_attribute\_\_** ((packed))
  -
- ```
struct {
    unsigned int count:5
        Count field.
    unsigned int reg\_addr:6
        Register address.
    unsigned int op:2
        Opcode.
    unsigned int type:3
        Frame type.
} __attribute__ ((packed)) bits
```

Variables

- unsigned int `count`
Count field.
- unsigned int `reg_addr`
Register address.
- unsigned int `op`
Opcode.
- unsigned int `type`
Frame type.
- uint16_t `word`
The raw configuration word.
- unsigned int `crc_err`
Indicates that the device failed to configure due to a CRC error.
- unsigned int `idcode_err`
Indicates that the device failed to configure due to the bitstream having the wrong ID code.
- unsigned int `dcm_lock`
Asserted once all DCM/PLL instances used in the design have locked on.
- unsigned int `gts_cfg_b`
Status of global tristate net.
- unsigned int `gwe`
Status of global write-enable net.
- unsigned int `ghigh`
Status of GHIGH (TODO: describe what this is)
- unsigned int `vsel`
Status of the SPI variant select pins.
- unsigned int `mode`
Status of the mode bits.
- unsigned int `init_b`
Status of the INIT_B pin.
- unsigned int `done`
Status of the DONE pin.
- unsigned int `seu_err`
True if there was a post-config CRC error.
- unsigned int `sync_timeout`
True if the config watchdog timer ran out.
- `XilinxSpartan3ADevice` `__attribute__`

8.119.1 Detailed Description

Declaration of `XilinxSpartan3ADevice`.

Author

Andrew D. Zonenberg

8.119.2 Variable Documentation

8.119.2.1 count

unsigned int count

Count field.

- Type 1 packets: word count
- Type 2 packets: don't care

8.119.2.2 op

unsigned int op

Opcode.

Must be one of the following:

- XilinxSpartan3ADevice::S3_CONFIG_OP_NOP
- XilinxSpartan3ADevice::S3_CONFIG_OP_READ
- XilinxSpartan3ADevice::S3_CONFIG_OP_WRITE

8.119.2.3 type

unsigned int type

Frame type.

Must be one of the following:

- XilinxSpartan3ADevice::S3A_CONFIG_FRAME_TYPE_1
- XilinxSpartan3ADevice::S3A_CONFIG_FRAME_TYPE_2

8.119.2.4 word

uint32_t word

The raw configuration word.

The raw status register value.

8.120 XilinxSpartan6Device.cpp File Reference

Implementation of [XilinxSpartan6Device](#).

```
#include "jtagahal.h"
```

Include dependency graph for XilinxSpartan6Device.cpp:



8.120.1 Detailed Description

Implementation of [XilinxSpartan6Device](#).

Author

Andrew D. Zonenberg

8.121 XilinxSpartan6Device.h File Reference

Declaration of [XilinxSpartan6Device](#).

This graph shows which files directly or indirectly include this file:



Classes

- union [XilinxSpartan6DeviceConfigurationFrame](#)
Spartan-6 configuration frame (see UG380 page 91)
- union [XilinxSpartan6DeviceStatusRegister](#)
Spartan-6 status register (see UG380 table 5-35)
- class [XilinxSpartan6Device](#)
A Xilinx Spartan-6 FPGA device.

Functions

- union [MicrochipPIC32DeviceStatusRegister](#) **__attribute__** ((packed))
 -
- ```
struct {
 unsigned int count:5
 Count field.
 unsigned int reg_addr:6
 Register address.
 unsigned int op:2
 Opcode.
 unsigned int type:3
 Frame type.
} __attribute__ ((packed)) bits
```

## Variables

- unsigned int `count`  
*Count field.*
- unsigned int `reg_addr`  
*Register address.*
- unsigned int `op`  
*Opcode.*
- unsigned int `type`  
*Frame type.*
- uint16\_t `word`  
*The raw configuration word.*
- unsigned int `crc_err`  
*Indicates that the device failed to configure due to a CRC error.*
- unsigned int `idcode_err`  
*Indicates that the device failed to configure due to the bitstream having the wrong ID code.*
- unsigned int `dcm_lock`  
*Asserted once all DCM/PLL instances used in the design have locked on.*
- unsigned int `gts_cfg_b`  
*Status of global tristate net.*
- unsigned int `gwe`  
*Status of global write-enable net.*
- unsigned int `ghigh`  
*Status of GHIGH (TODO: describe what this is)*
- unsigned int `decrypt_err`  
*Decryption error flag.*
- unsigned int `decrypt_en`  
*Bitstream encryption enable flag.*
- unsigned int `hswapen`  
*Status of the HSWAPEN pin.*
- unsigned int `m0`  
*Status of the M0 mode bit.*
- unsigned int `m1`  
*Status of the M1 mode bit.*
- unsigned int `reserved`  
*Reserved.*
- unsigned int `init_b`  
*Status of the INIT\_B pin.*
- unsigned int `done`  
*Status of the DONE pin.*
- unsigned int `suspend`  
*Suspend state.*
- unsigned int `fallback`  
*Configuration fallback state.*
- `XilinxSpartan6Device` `__attribute__`

### 8.121.1 Detailed Description

Declaration of `XilinxSpartan6Device`.

#### Author

Andrew D. Zonenberg

## 8.121.2 Variable Documentation

### 8.121.2.1 count

unsigned int count

Count field.

- Type 1 packets: word count
- Type 2 packets: don't care

### 8.121.2.2 op

unsigned int op

Opcode.

Must be one of the following:

- XilinxSpartan6Device::S6\_CONFIG\_OP\_NOP
- XilinxSpartan6Device::S6\_CONFIG\_OP\_READ
- XilinxSpartan6Device::S6\_CONFIG\_OP\_WRITE

### 8.121.2.3 type

unsigned int type

Frame type.

Must be one of the following:

- XilinxSpartan6Device::S6\_CONFIG\_FRAME\_TYPE\_1
- XilinxSpartan6Device::S6\_CONFIG\_FRAME\_TYPE\_2

## 8.121.2.4 word

```
uint16_t word
```

The raw configuration word.

The raw status register value.

## 8.122 XilinxUltrascaleDevice.cpp File Reference

Implementation of [XilinxUltrascaleDevice](#).

```
#include "jtagahal.h"
#include "XilinxDeviceID_enum.h"
Include dependency graph for XilinxUltrascaleDevice.cpp:
```



### 8.122.1 Detailed Description

Implementation of [XilinxUltrascaleDevice](#).

#### Author

Andrew D. Zonenberg

## 8.123 XilinxUltrascaleDevice.h File Reference

Declaration of [XilinxUltrascaleDevice](#).

This graph shows which files directly or indirectly include this file:



### Classes

- union [XilinxUltrascaleDeviceConfigurationFrame](#)  
*UltraScale configuration frame (see UG570 page 158)*
- union [XilinxUltrascaleDeviceStatusRegister](#)  
*UltraScale status register (see UG570 table 9-25)*
- class [XilinxUltrascaleDevice](#)  
*A Xilinx UltraScale or UltraScale+ [FPGA](#) device.*

## Functions

- union [MicrochipPIC32DeviceStatusRegister](#) `__attribute__((packed))`
- 
- struct {
  - unsigned int [count](#):11  
*Count field.*
  - unsigned int [reserved](#):2  
*Reserved, must be zero.*
  - unsigned int [reg\\_addr](#):14  
*Register address.*
  - unsigned int [op](#):2  
*Opcode.*
  - unsigned int [type](#):3  
*Frame type.*
- } `__attribute__((packed))` [bits](#)

## Variables

- unsigned int [count](#)  
*Count field.*
- unsigned int [reserved](#)  
*Reserved, must be zero.*
- unsigned int [reg\\_addr](#)  
*Register address.*
- unsigned int [op](#)  
*Opcode.*
- unsigned int [type](#)  
*Frame type.*
- uint32\_t [word](#)  
*The raw configuration word.*
- unsigned int [crc\\_err](#)  
*Indicates that the device failed to configure due to a CRC error.*
- unsigned int [decryptor\\_enabled](#)  
*Indicates that the crypto subsystem is active.*
- unsigned int [mmcm\\_lock](#)  
*Indicates MMCMs and PLLs are locked.*
- unsigned int [dci\\_match](#)  
*Indicates DCI is matched.*
- unsigned int [eos](#)  
*End-of-Startup signal.*
- unsigned int [gts\\_cfg\\_b](#)  
*Status of GTS\_CFG net.*
- unsigned int [gwe](#)  
*Status of GWE net.*
- unsigned int [ghigh\\_b](#)  
*Status of GHIGH\_B net.*
- unsigned int [mode\\_pins](#)  
*Status of mode pins.*
- unsigned int [init\\_complete](#)  
*Internal init-finished signal.*



- unsigned int [init\\_b](#)  
*Status of INIT\_B pin.*
- unsigned int [release\\_done](#)  
*Indicates DONE was released.*
- unsigned int [done](#)  
*Actual value on DONE pin.*
- unsigned int [id\\_error](#)  
*Indicates an ID code error occurred (write with wrong bitstream)*
- unsigned int [security\\_error](#)  
*Security / crypto error.*
- unsigned int [sysmon\\_over\\_temp](#)  
*Indicates board is too hot.*
- unsigned int [startup\\_state](#)  
*Status of startup state machine.*
- unsigned int [reserved\\_1](#)  
*Reserved.*
- unsigned int [bus\\_width](#)  
*Config bus width (see table 5-26)*
- unsigned int [reserved\\_2](#)  
*Reserved.*
- [XilinxUltrascaleDevice](#) `__attribute__`

### 8.123.1 Detailed Description

Declaration of [XilinxUltrascaleDevice](#).

#### Author

Andrew D. Zonenberg

### 8.123.2 Variable Documentation

#### 8.123.2.1 op

unsigned int op

Opcode.

Must be one of the following:

- `XilinxUltrascaleDevice::CONFIG_OP_NOP`
- `XilinxUltrascaleDevice::CONFIG_OP_READ`
- `XilinxUltrascaleDevice::CONFIG_OP_WRITE`

Must be zero

**8.123.2.2 type**

unsigned int type

Frame type.

Must be XilinxUltrascaleDevice::CONFIG\_FRAME\_TYPE\_1

Must be XilinxUltrascaleDevice::CONFIG\_FRAME\_TYPE\_2

**8.123.2.3 word**

uint32\_t word

The raw configuration word.

The raw status register value.

# Index

- ~DigilentJtagInterface
  - DigilentJtagInterface, 81
- ~FTDIJtagInterface
  - FTDIJtagInterface, 105
- ~JtagInterface
  - JtagInterface, 136
- APRegisterRead
  - ARMJtagDebugPort, 56
- APRegisterWrite
  - ARMJtagDebugPort, 56
- ARM7TDMISProcessor, 29
  - Erase, 31
  - IsProgrammed, 31
  - LoadFirmwareImage, 32
  - PostInitProbes, 32
  - Program, 33
- ARM7TDMISProcessor.cpp, 273
- ARM7TDMISProcessor.h, 273
- ARMAPBDevice, 33
- ARMAPBDevice.cpp, 274
- ARMAPBDevice.h, 274
- ARMCoreSightDevice, 35
- ARMCoreSightDevice.cpp, 275
- ARMCoreSightDevice.h, 275
- ARMcortexA57, 36
- ARMcortexA57.cpp, 276
- ARMcortexA57.h, 276
- ARMcortexA9, 38
- ARMcortexA9.cpp, 277
- ARMcortexA9.h, 277
- ARMcortexM4, 40
- ARMcortexM4.cpp, 278
- ARMcortexM4.h, 278
- ARMDebugAccessPort, 41
- ARMDebugAccessPort.cpp, 279
- ARMDebugAccessPort.h, 279
  - reserved\_zero, 280
  - revision, 280
  - type, 281
  - variant, 281
  - word, 281
- ARMDebugArchVersion
  - ARMv7Processor.h, 293
- ARMDebugMemAPControlStatusWord, 44
- ARMDebugMemAccessPort, 43
- ARMDebugMemAccessPort.cpp, 282
- ARMDebugMemAccessPort.h, 282
  - mode, 283
- ARMDebugPeripheralIDRegister, 45
- ARMDebugPeripheralIDRegister.h, 284
- ARMDebugPeripheralIDRegisterBits, 46
- ARMDebugPort, 47
- ARMDebugPort.h, 285
- ARMDebugPortIDRegister, 49
- ARMDevice, 50
  - ARMDevice, 51
  - CreateDevice, 51
- ARMDevice.cpp, 285
- ARMDevice.h, 286
- ARMFlashPatchBreakpoint, 52
- ARMFlashPatchBreakpoint.cpp, 286
- ARMFlashPatchBreakpoint.h, 287
- ARMJtagDebugPort, 54
  - APRegisterRead, 56
  - APRegisterWrite, 56
  - DPRegisterRead, 57
  - GetDescription, 57
  - PostInitProbes, 57
- ARMJtagDebugPort.cpp, 287
- ARMJtagDebugPort.h, 288
- ARMJtagDebugPortStatusRegister, 58
- ARMv7DebugIDRegister, 59
- ARMv7DebugStatusControlRegister, 60
- ARMv7MPProcessor, 62
  - DebugHalt, 63
- ARMv7MPProcessor.cpp, 289
- ARMv7MPProcessor.h, 290
- ARMv7Processor, 64
  - DebugHalt, 66
- ARMv7Processor.cpp, 290
- ARMv7Processor.h, 291
  - ARMDebugArchVersion, 293
  - reserved, 294
- ARMv8Processor, 67
- ARMv8Processor.cpp, 294
- ARMv8Processor.h, 294
- AttachedMemoryDevice, 69
- AttachedMemoryDevice.h, 295
- ByteArrayFirmwareImage, 70
- ByteArrayFirmwareImage.cpp, 295
- ByteArrayFirmwareImage.h, 296
- CPLD.cpp, 296
- CPLD.h, 297
- CPLDBitstream, 74
- CPLDBitstream.cpp, 297
- CPLDBitstream.h, 297
- CPLD, 72

- ParseJEDFile, [73](#)
- ReadIntLine, [73](#)
- ClearReadLock
  - LockableDevice, [152](#)
  - STM32Device, [199](#)
- cmd\_values
  - XilinxUltrascaleDevice, [264](#)
- Commit
  - FTDIJtagInterface, [105](#)
  - JtagDevice, [119](#)
  - JtagInterface, [136](#)
  - NetworkedJtagInterface, [168](#)
  - PipeJtagInterface, [180](#)
- Connect
  - NetworkedJtagInterface, [169](#)
- count
  - Xilinx7SeriesDevice.h, [337](#)
  - XilinxSpartan3ADevice.h, [348](#)
  - XilinxSpartan3ADeviceConfigurationFrame, [246](#)
  - XilinxSpartan6Device.h, [352](#)
  - XilinxSpartan6DeviceConfigurationFrame, [257](#)
- CreateDevice
  - ARMDevice, [51](#)
  - FreescaleDevice, [93](#)
  - JtagDevice, [119](#)
  - MicrochipDevice, [155](#)
  - STMicroDevice, [203](#)
  - XilinxDevice, [230](#)
- CreateDummyDevices
  - JtagInterface, [136](#)
- DPRRegisterRead
  - ARMJtagDebugPort, [57](#)
- DebugHalt
  - ARMv7MProcessor, [63](#)
  - ARMv7Processor, [66](#)
- DebuggableDevice, [76](#)
- DebuggableDevice.cpp, [298](#)
- DebuggableDevice.h, [298](#)
- DebuggerInterface, [77](#)
- DebuggerInterface.cpp, [299](#)
- DebuggerInterface.h, [299](#)
- deviceids
  - XilinxSpartan3ADevice, [240](#)
  - XilinxSpartan6Device, [251](#)
- DigilentJtagInterface, [79](#)
  - ~DigilentJtagInterface, [81](#)
  - DigilentJtagInterface, [81](#)
  - GetFrequency, [81](#)
  - GetName, [81](#)
  - GetSerial, [82](#)
  - GetUserID, [82](#)
  - SendDummyClocks, [82](#)
  - ShiftData, [83](#)
  - ShiftTMS, [83](#)
- DigilentJtagInterface.cpp, [300](#)
- DigilentJtagInterface.h, [300](#)
- DoReadback
  - FTDIJtagInterface, [106](#)
- done
  - Xilinx7SeriesDevice.h, [337](#)
- EjtagControlRegister, [84](#)
- EjtagImplementationCodeRegister, [85](#)
- EnterShiftDR
  - JtagDevice, [120](#)
  - JtagInterface, [137](#)
  - NetworkedJtagInterface, [169](#)
  - PipeJtagInterface, [180](#)
- EnterShiftIR
  - JtagInterface, [137](#)
  - NetworkedJtagInterface, [169](#)
  - PipeJtagInterface, [180](#)
- Erase
  - ARM7TDMISProcessor, [31](#)
  - FreescaleIMXDevice, [96](#)
  - MicrochipPIC32Device, [161](#)
  - ProgrammableDevice, [189](#)
  - STM32Device, [199](#)
  - Xilinx7SeriesDevice, [213](#)
  - XilinxCoolRunnerIIDevice, [223](#)
  - XilinxSpartan3ADevice, [241](#)
  - XilinxSpartan6Device, [252](#)
  - XilinxUltrascaleDevice, [265](#)
- FPGA.cpp, [302](#)
- FPGA.h, [302](#)
- FPGABitstream, [88](#)
- FPGABitstream.cpp, [303](#)
- FPGABitstream.h, [303](#)
- FPGA, [87](#)
- FTDIJtagInterface, [101](#)
  - ~FTDIJtagInterface, [105](#)
  - Commit, [105](#)
  - DoReadback, [106](#)
  - FTDIJtagInterface, [105](#)
  - GenerateShiftPacket, [106](#)
  - GetAPIVersion, [106](#)
  - GetDefaultFrequency, [107](#)
  - GetDescription, [107](#)
  - GetInterfaceCount, [107](#)
  - GetSerialNumber, [108](#)
  - IsJtagCapable, [108](#)
  - IsSplitScanSupported, [109](#)
  - ReadData, [109](#)
  - SendDummyClocks, [109](#)
  - SendDummyClocksDeferred, [110](#)
  - ShiftData, [110](#)
  - ShiftDataReadOnly, [111](#)
  - ShiftDataWriteOnly, [111](#)
  - ShiftTMS, [112](#)
  - SyncCheck, [112](#)
  - WriteData, [113](#)
  - WriteDataRaw, [113](#)
- FTDIJtagInterface.cpp, [308](#)
- FTDIJtagInterface.h, [309](#)
- FirmwareImage, [86](#)
- idcode, [87](#)

- FirmwareImage.cpp, 301
- FirmwareImage.h, 301
- FlipBitAndEndian32Array
  - Stuff not in another group yet, 23
- FlipBitAndEndianArray
  - Stuff not in another group yet, 23
- FlipBitArray
  - Stuff not in another group yet, 24
- FlipByte
  - Stuff not in another group yet, 24
- FlipByteArray
  - Stuff not in another group yet, 24
- FlipEndian32Array
  - Stuff not in another group yet, 25
- FlipEndianArray
  - Stuff not in another group yet, 25
- FreescaleDevice, 91
  - CreateDevice, 93
  - FreescaleDevice, 92
- FreescaleDevice.cpp, 303
- FreescaleDevice.h, 304
- FreescaleIMXDevice, 94
  - Erase, 96
  - GetDescription, 96
  - instructions, 95
  - IsProgrammed, 96
  - PostInitProbes, 96
  - Program, 97
- FreescaleIMXDevice.cpp, 304
- FreescaleIMXDevice.h, 305
- FreescaleIMXSmartDMA.cpp, 306
- FreescaleIMXSmartDMA.h, 306
- FreescaleIMXSmartDMA, 97
  - GetDescription, 99
  - PostInitProbes, 99
- FreescaleMicrocontroller, 100
- FreescaleMicrocontroller.cpp, 307
- FreescaleMicrocontroller.h, 307
  
- GPIOInterface, 114
- GPIOInterface.cpp, 309
- GPIOInterface.h, 310
- GeneratePermutationTable
  - XilinxCoolRunnerIIDevice, 224
- GenerateShiftPacket
  - FTDIJtagInterface, 106
- GetAPIVersion
  - FTDIJtagInterface, 106
- GetBigEndianUInt16FromByteArray
  - jtaghal.cpp, 315
  - jtaghal.h, 318
- GetBigEndianUInt32FromByteArray
  - jtaghal.cpp, 315
  - jtaghal.h, 318
- GetDataBitCount
  - JtagInterface, 137
  - NetworkedJtagInterface, 170
  - PipeJtagInterface, 181
- GetDefaultFrequency
  - FTDIJtagInterface, 107
- GetDescription
  - ARMJtagDebugPort, 57
  - FTDIJtagInterface, 107
  - FreescaleIMXDevice, 96
  - FreescaleIMXSmartDMA, 99
  - JtagDevice, 120
  - JtagDummy, 125
  - JtagException, 128
  - MicrochipPIC32Device, 161
  - STM32Device, 199
  - Xilinx7SeriesDevice, 213
  - XilinxCoolRunnerIIDevice, 224
  - XilinxSpartan3ADevice, 241
  - XilinxSpartan6Device, 252
  - XilinxUltrascaleDevice, 265
- GetDevice
  - JtagInterface, 138
- GetDeviceCount
  - JtagInterface, 138
- GetDummyClockCount
  - JtagInterface, 138
  - NetworkedJtagInterface, 170
  - PipeJtagInterface, 181
- GetFrequency
  - DigilentJtagInterface, 81
  - JtagInterface, 139
  - NetworkedJtagInterface, 170
  - PipeJtagInterface, 181
- GetIDCode
  - JtagInterface, 139
- GetInterfaceCount
  - FTDIJtagInterface, 107
- GetModeBitCount
  - JtagInterface, 139
  - NetworkedJtagInterface, 171
  - PipeJtagInterface, 182
- GetName
  - DigilentJtagInterface, 81
  - JtagInterface, 140
  - NetworkedJtagInterface, 171
  - PipeJtagInterface, 182
- GetPaddingSize
  - XilinxCoolRunnerIIDevice, 224
- GetPrettyPrintedSerialNumber
  - STM32Device, 199
  - SerialNumberedDevice, 194
- GetRecoverableErrorCount
  - JtagInterface, 140
  - NetworkedJtagInterface, 171
  - PipeJtagInterface, 182
- GetSerial
  - DigilentJtagInterface, 82
  - JtagInterface, 141
  - NetworkedJtagInterface, 172
  - PipeJtagInterface, 183
- GetSerialNumber
  - FTDIJtagInterface, 108

- STM32Device, [200](#)
- SerialNumberedDevice, [195](#)
- Xilinx7SeriesDevice, [213](#)
- XilinxSpartan3ADevice, [242](#)
- XilinxSpartan6Device, [253](#)
- XilinxUltrascaleDevice, [265](#)
- GetSerialNumberLength
  - STM32Device, [200](#)
  - SerialNumberedDevice, [195](#)
  - Xilinx7SeriesDevice, [214](#)
  - XilinxSpartan3ADevice, [242](#)
  - XilinxSpartan6Device, [253](#)
  - XilinxUltrascaleDevice, [266](#)
- GetSerialNumberLengthBits
  - STM32Device, [200](#)
  - SerialNumberedDevice, [195](#)
  - Xilinx7SeriesDevice, [214](#)
  - XilinxSpartan3ADevice, [242](#)
  - XilinxSpartan6Device, [253](#)
  - XilinxUltrascaleDevice, [266](#)
- GetShiftOpCount
  - JtagInterface, [141](#)
  - NetworkedJtagInterface, [172](#)
  - PipeJtagInterface, [183](#)
- GetShiftRegisterDepth
  - XilinxCoolRunnerIIDevice, [224](#)
- GetShiftRegisterWidth
  - XilinxCoolRunnerIIDevice, [224](#)
- GetShiftTime
  - JtagInterface, [141](#)
- GetTime
  - Stuff not in another group yet, [25](#)
- GetUserID
  - DigilentJtagInterface, [82](#)
  - JtagInterface, [142](#)
  - NetworkedJtagInterface, [172](#)
  - PipeJtagInterface, [183](#)
- GetUserVIDPID
  - JtagFPGA, [129](#)
- gts\_cfg\_b
  - Xilinx7SeriesDevice.h, [337](#)
- gwe
  - Xilinx7SeriesDevice.h, [337](#)
- idcode
  - FirmwareImage, [87](#)
- init\_b
  - Xilinx7SeriesDevice.h, [337](#)
- InitializeChain
  - JtagInterface, [142](#)
- instructions
  - FreescaleIMXDevice, [95](#)
  - MicrochipPIC32Device, [160](#)
  - Xilinx7SeriesDevice, [212](#)
  - XilinxCoolRunnerIIDevice, [222](#)
  - XilinxSpartan3ADevice, [240](#)
  - XilinxSpartan6Device, [251](#)
  - XilinxUltrascaleDevice, [264](#)
- IsJtagCapable
  - FTDIJtagInterface, [108](#)
- IsProgrammed
  - ARM7TDMISProcessor, [31](#)
  - FreescaleIMXDevice, [96](#)
  - MicrochipPIC32Device, [162](#)
  - ProgrammableDevice, [189](#)
  - STM32Device, [201](#)
  - Xilinx7SeriesDevice, [214](#)
  - XilinxCoolRunnerIIDevice, [224](#)
  - XilinxSpartan3ADevice, [243](#)
  - XilinxSpartan6Device, [254](#)
  - XilinxUltrascaleDevice, [266](#)
- IsSplitScanSupported
  - FTDIJtagInterface, [109](#)
  - JtagDevice, [120](#)
  - JtagInterface, [142](#)
  - NetworkedJtagInterface, [173](#)
  - PipeJtagInterface, [184](#)
- JTAG interface layer, [19](#)
- JtagDevice, [116](#)
  - Commit, [119](#)
  - CreateDevice, [119](#)
  - EnterShiftDR, [120](#)
  - GetDescription, [120](#)
  - IsSplitScanSupported, [120](#)
  - JtagDevice, [119](#)
  - PostInitProbes, [120](#)
  - ResetToIdle, [121](#)
  - ScanDRDeferred, [121](#)
  - ScanDRSplitRead, [121](#)
  - ScanDRSplitWrite, [121](#)
  - ScanDR, [121](#)
  - SendDummyClocks, [122](#)
  - SendDummyClocksDeferred, [122](#)
  - SetIR, [122](#)
  - ShiftData, [122](#)
- JtagDevice.cpp, [310](#)
- JtagDevice.h, [311](#)
- JtagDummy, [123](#)
  - GetDescription, [125](#)
  - JtagDummy, [125](#)
  - PostInitProbes, [125](#)
- JtagDummy.cpp, [311](#)
- JtagDummy.h, [312](#)
- JtagException, [126](#)
  - GetDescription, [128](#)
  - JtagException, [127](#)
- JtagException.cpp, [312](#)
- JtagException.h, [313](#)
  - JtagExceptionWrapper, [313](#)
- JtagExceptionWrapper
  - JtagException.h, [313](#)
- JtagFPGA.cpp, [314](#)
- JtagFPGA.h, [314](#)
- JtagFPGA, [128](#)
  - GetUserVIDPID, [129](#)
  - JtagFPGA, [129](#)
- JtagInterface, [130](#)

- ~JtagInterface, [136](#)
- Commit, [136](#)
- CreateDummyDevices, [136](#)
- EnterShiftDR, [137](#)
- EnterShiftIR, [137](#)
- GetDataBitCount, [137](#)
- GetDevice, [138](#)
- GetDeviceCount, [138](#)
- GetDummyClockCount, [138](#)
- GetFrequency, [139](#)
- GetIDCode, [139](#)
- GetModeBitCount, [139](#)
- GetName, [140](#)
- GetRecoverableErrorCount, [140](#)
- GetSerial, [141](#)
- GetShiftOpCount, [141](#)
- GetShiftTime, [141](#)
- GetUserID, [142](#)
- InitializeChain, [142](#)
- IsSplitScanSupported, [142](#)
- JtagInterface, [136](#)
- LeaveExit1DR, [142](#)
- LeaveExit1IR, [143](#)
- ResetToldle, [143](#)
- ScanDRDeferred, [144](#)
- ScanDRSplitRead, [144](#)
- ScanDRSplitWrite, [145](#)
- ScanDR, [143](#)
- SendDummyClocks, [146](#)
- SendDummyClocksDeferred, [146](#)
- SetIRDeferred, [147](#)
- SetIR, [146](#), [147](#)
- ShiftData, [148](#)
- ShiftDataReadOnly, [148](#)
- ShiftDataWriteOnly, [149](#)
- ShiftTMS, [150](#)
- SwapOutDummy, [150](#)
- TestLogicReset, [150](#)
- JtagInterface.cpp, [319](#)
- JtagInterface.h, [319](#)
- jtaghal.cpp, [314](#)
  - GetBigEndianUInt16FromByteArray, [315](#)
  - GetBigEndianUInt32FromByteArray, [315](#)
- jtaghal.h, [316](#)
  - GetBigEndianUInt16FromByteArray, [318](#)
  - GetBigEndianUInt32FromByteArray, [318](#)
- LeaveExit1DR
  - JtagInterface, [142](#)
  - NetworkedJtagInterface, [173](#)
  - PipeJtagInterface, [184](#)
- LeaveExit1IR
  - JtagInterface, [143](#)
  - NetworkedJtagInterface, [173](#)
  - PipeJtagInterface, [184](#)
- LoadFirmwareImage
  - ARM7TDMISProcessor, [32](#)
  - Microcontroller, [165](#)
  - ProgrammableDevice, [189](#), [190](#)
  - STM32Device, [201](#)
  - Xilinx7SeriesDevice, [215](#)
  - XilinxCoolRunnerIIDevice, [225](#)
  - XilinxSpartan3ADevice, [243](#)
  - XilinxSpartan6Device, [254](#)
  - XilinxUltrascaleDevice, [267](#)
- LockableDevice, [151](#)
  - ClearReadLock, [152](#)
  - SetReadLock, [152](#)
- LockableDevice.cpp, [319](#)
- LockableDevice.h, [320](#)
- MicrochipDevice, [153](#)
  - CreateDevice, [155](#)
  - MicrochipDevice, [154](#)
- MicrochipDevice.cpp, [320](#)
- MicrochipDevice.h, [321](#)
- MicrochipMicrocontroller, [156](#)
- MicrochipMicrocontroller.cpp, [321](#)
- MicrochipMicrocontroller.h, [322](#)
- MicrochipPIC32Device, [157](#)
  - Erase, [161](#)
  - GetDescription, [161](#)
  - instructions, [160](#)
  - IsProgrammed, [162](#)
  - mtap\_instructions, [161](#)
  - PostInitProbes, [162](#)
  - Program, [162](#)
- MicrochipPIC32Device.cpp, [322](#)
- MicrochipPIC32Device.h, [323](#)
- MicrochipPIC32DeviceInfo, [163](#)
- MicrochipPIC32DeviceStatusRegister, [163](#)
- Microcontroller, [164](#)
  - LoadFirmwareImage, [165](#)
- Microcontroller.cpp, [325](#)
- Microcontroller.h, [325](#)
- MirrorBitArray
  - Stuff not in another group yet, [26](#)
- mmcm\_lock
  - Xilinx7SeriesDevice.h, [338](#)
- mode
  - ARMDebugMemAccessPort.h, [283](#)
- mtap\_instructions
  - MicrochipPIC32Device, [161](#)
- NetworkedJtagInterface, [166](#)
  - Commit, [168](#)
  - Connect, [169](#)
  - EnterShiftDR, [169](#)
  - EnterShiftIR, [169](#)
  - GetDataBitCount, [170](#)
  - GetDummyClockCount, [170](#)
  - GetFrequency, [170](#)
  - GetModeBitCount, [171](#)
  - GetName, [171](#)
  - GetRecoverableErrorCount, [171](#)
  - GetSerial, [172](#)
  - GetShiftOpCount, [172](#)
  - GetUserID, [172](#)

- IsSplitScanSupported, [173](#)
- LeaveExit1DR, [173](#)
- LeaveExit1IR, [173](#)
- ResetToldle, [174](#)
- SendDummyClocks, [174](#)
- SendDummyClocksDeferred, [174](#)
- ShiftData, [175](#)
- ShiftDataReadOnly, [175](#)
- ShiftDataWriteOnly, [176](#)
- TestLogicReset, [176](#)
- NetworkedJtagInterface.cpp, [325](#)
- NetworkedJtagInterface.h, [326](#)
- op
  - Xilinx7SeriesDevice.h, [338](#)
  - Xilinx7SeriesDeviceConfigurationFrame, [218](#)
  - XilinxSpartan3ADevice.h, [349](#)
  - XilinxSpartan3ADeviceConfigurationFrame, [246](#)
  - XilinxSpartan6Device.h, [352](#)
  - XilinxSpartan6DeviceConfigurationFrame, [258](#)
  - XilinxUltrascaleDevice.h, [355](#)
  - XilinxUltrascaleDeviceConfigurationFrame, [270](#)
- packages
  - XilinxCoolRunnerIIDevice, [223](#)
- ParseBitstreamCore
  - XilinxFPGA, [233](#)
- ParseBitstreamInternals
  - Xilinx7SeriesDevice, [215](#)
  - XilinxFPGA, [233](#)
  - XilinxSpartan3ADevice, [244](#)
  - XilinxSpartan6Device, [255](#)
  - XilinxUltrascaleDevice, [267](#)
- ParseJEDFile
  - CPLD, [73](#)
- PeekBit
  - Stuff not in another group yet, [26](#)
- PipeJtagInterface, [177](#)
  - Commit, [180](#)
  - EnterShiftDR, [180](#)
  - EnterShiftIR, [180](#)
  - GetDataBitCount, [181](#)
  - GetDummyClockCount, [181](#)
  - GetFrequency, [181](#)
  - GetModeBitCount, [182](#)
  - GetName, [182](#)
  - GetRecoverableErrorCount, [182](#)
  - GetSerial, [183](#)
  - GetShiftOpCount, [183](#)
  - GetUserID, [183](#)
  - IsSplitScanSupported, [184](#)
  - LeaveExit1DR, [184](#)
  - LeaveExit1IR, [184](#)
  - ResetToldle, [185](#)
  - SendDummyClocks, [185](#)
  - SendDummyClocksDeferred, [185](#)
  - ShiftData, [186](#)
  - ShiftDataReadOnly, [186](#)
  - ShiftDataWriteOnly, [187](#)
  - TestLogicReset, [187](#)
  - PipeJtagInterface.cpp, [326](#)
  - PipeJtagInterface.h, [327](#)
- PokeBit
  - Stuff not in another group yet, [26](#)
- PostInitProbes
  - ARM7TDMISProcessor, [32](#)
  - ARMJtagDebugPort, [57](#)
  - FreescaleIMXDevice, [96](#)
  - FreescaleIMXSmartDMA, [99](#)
  - JtagDevice, [120](#)
  - JtagDummy, [125](#)
  - MicrochipPIC32Device, [162](#)
  - STM32Device, [201](#)
  - XilinxCoolRunnerIIDevice, [225](#)
  - XilinxFPGA, [234](#)
- Program
  - ARM7TDMISProcessor, [33](#)
  - FreescaleIMXDevice, [97](#)
  - MicrochipPIC32Device, [162](#)
  - ProgrammableDevice, [190](#)
  - STM32Device, [202](#)
  - Xilinx7SeriesDevice, [216](#)
  - XilinxCoolRunnerIIDevice, [226](#)
  - XilinxSpartan3ADevice, [244](#)
  - XilinxSpartan6Device, [255](#)
  - XilinxUltrascaleDevice, [268](#)
- ProgrammableDevice, [188](#)
  - Erase, [189](#)
  - IsProgrammed, [189](#)
  - LoadFirmwareImage, [189](#), [190](#)
  - Program, [190](#)
- ProgrammableDevice.cpp, [327](#)
- ProgrammableDevice.h, [327](#)
- ProgrammableLogicDevice, [191](#)
- ProgrammableLogicDevice.cpp, [328](#)
- ProgrammableLogicDevice.h, [328](#)
- RawBinaryFirmwareImage, [192](#)
- RawBinaryFirmwareImage.cpp, [329](#)
- RawBinaryFirmwareImage.h, [329](#)
- ReadData
  - FTDIJtagInterface, [109](#)
- ReadIntLine
  - CPLD, [73](#)
- ReadWordConfigRegister
  - Xilinx7SeriesDevice, [216](#)
  - XilinxSpartan3ADevice, [244](#)
  - XilinxSpartan6Device, [256](#)
  - XilinxUltrascaleDevice, [268](#)
- ReadWordsConfigRegister
  - XilinxSpartan6Device, [256](#)
- ReadingSerialRequiresReset
  - STM32Device, [202](#)
  - SerialNumberedDevice, [196](#)
  - XilinxFPGA, [234](#)
- reserved
  - ARMv7Processor.h, [294](#)
- reserved\_zero



- ARMDebugAccessPort.h, 280
- ResetToldle
  - JtagDevice, 121
  - JtagInterface, 143
  - NetworkedJtagInterface, 174
  - PipeJtagInterface, 185
- revision
  - ARMDebugAccessPort.h, 280
- STM32Device, 196
  - ClearReadLock, 199
  - Erase, 199
  - GetDescription, 199
  - GetPrettyPrintedSerialNumber, 199
  - GetSerialNumber, 200
  - GetSerialNumberLength, 200
  - GetSerialNumberLengthBits, 200
  - IsProgrammed, 201
  - LoadFirmwareImage, 201
  - PostInitProbes, 201
  - Program, 202
  - ReadingSerialRequiresReset, 202
  - SetReadLock, 202
- STM32Device.cpp, 331
- STM32Device.h, 331
- STMMicroDevice, 203
  - CreateDevice, 203
- STMMicroDevice.cpp, 332
- STMMicroDevice.h, 332
- STMMicroMicrocontroller, 204
- STMMicroMicrocontroller.cpp, 333
- STMMicroMicrocontroller.h, 333
- ScanDRDeferred
  - JtagDevice, 121
  - JtagInterface, 144
- ScanDRSplitRead
  - JtagDevice, 121
  - JtagInterface, 144
- ScanDRSplitWrite
  - JtagDevice, 121
  - JtagInterface, 145
- ScanDR
  - JtagDevice, 121
  - JtagInterface, 143
- SendDummyClocks
  - DigilentJtagInterface, 82
  - FTDIJtagInterface, 109
  - JtagDevice, 122
  - JtagInterface, 146
  - NetworkedJtagInterface, 174
  - PipeJtagInterface, 185
- SendDummyClocksDeferred
  - FTDIJtagInterface, 110
  - JtagDevice, 122
  - JtagInterface, 146
  - NetworkedJtagInterface, 174
  - PipeJtagInterface, 185
- SerialNumberedDevice, 194
  - GetPrettyPrintedSerialNumber, 194
  - GetSerialNumber, 195
  - GetSerialNumberLength, 195
  - GetSerialNumberLengthBits, 195
  - ReadingSerialRequiresReset, 196
- SerialNumberedDevice.cpp, 330
- SerialNumberedDevice.h, 330
- SetIRDeferred
  - JtagInterface, 147
- SetIR
  - JtagDevice, 122
  - JtagInterface, 146, 147
- SetReadLock
  - LockableDevice, 152
  - STM32Device, 202
- ShiftData
  - DigilentJtagInterface, 83
  - FTDIJtagInterface, 110
  - JtagDevice, 122
  - JtagInterface, 148
  - NetworkedJtagInterface, 175
  - PipeJtagInterface, 186
- ShiftDataReadOnly
  - FTDIJtagInterface, 111
  - JtagInterface, 148
  - NetworkedJtagInterface, 175
  - PipeJtagInterface, 186
- ShiftDataWriteOnly
  - FTDIJtagInterface, 111
  - JtagInterface, 149
  - NetworkedJtagInterface, 176
  - PipeJtagInterface, 187
- ShiftTMS
  - DigilentJtagInterface, 83
  - FTDIJtagInterface, 112
  - JtagInterface, 150
- Stuff not in another group yet, 20
  - FlipBitAndEndian32Array, 23
  - FlipBitAndEndianArray, 23
  - FlipBitArray, 24
  - FlipByte, 24
  - FlipByteArray, 24
  - FlipEndian32Array, 25
  - FlipEndianArray, 25
  - GetTime, 25
  - MirrorBitArray, 26
  - PeekBit, 26
  - PokeBit, 26
- SwapOutDummy
  - JtagInterface, 150
- SyncCheck
  - FTDIJtagInterface, 112
- TestLogicReset
  - JtagInterface, 150
  - NetworkedJtagInterface, 176
  - PipeJtagInterface, 187
- type
  - ARMDebugAccessPort.h, 281
  - Xilinx7SeriesDevice.h, 338

- Xilinx7SeriesDeviceConfigurationFrame, 218
- XilinxSpartan3ADevice.h, 349
- XilinxSpartan3ADeviceConfigurationFrame, 246
- XilinxSpartan6Device.h, 352
- XilinxSpartan6DeviceConfigurationFrame, 258
- XilinxUltrascaleDevice.h, 355
- XilinxUltrascaleDeviceConfigurationFrame, 270
- UncertainBoolean, 206
- variant
  - ARMDebugAccessPort.h, 281
- word
  - ARMDebugAccessPort.h, 281
  - Xilinx7SeriesDevice.h, 339
  - XilinxSpartan3ADevice.h, 349
  - XilinxSpartan6Device.h, 352
  - XilinxUltrascaleDevice.h, 356
- WriteData
  - FTDIJtagInterface, 113
- WriteDataRaw
  - FTDIJtagInterface, 113
- Xilinx3DFPGABitstream, 207
- Xilinx3DFPGABitstream.cpp, 334
- Xilinx3DFPGABitstream.h, 334
- Xilinx7SeriesDevice, 209
  - Erase, 213
  - GetDescription, 213
  - GetSerialNumber, 213
  - GetSerialNumberLength, 214
  - GetSerialNumberLengthBits, 214
  - instructions, 212
  - IsProgrammed, 214
  - LoadFirmwareImage, 215
  - ParseBitstreamInternals, 215
  - Program, 216
  - ReadWordConfigRegister, 216
  - Xilinx7SeriesDevice, 212
- Xilinx7SeriesDevice.cpp, 334
- Xilinx7SeriesDevice.h, 335
  - count, 337
  - done, 337
  - gts\_cfg\_b, 337
  - gwe, 337
  - init\_b, 337
  - mmcm\_lock, 338
  - op, 338
  - type, 338
  - word, 339
- Xilinx7SeriesDeviceConfigurationFrame, 217
  - op, 218
  - type, 218
- Xilinx7SeriesDeviceStatusRegister, 218
- XilinxCPLD.cpp, 341
- XilinxCPLD.h, 342
- XilinxCPLDBitstream, 228
- XilinxCPLDBitstream.cpp, 342
- XilinxCPLDBitstream.h, 343
- XilinxCPLD, 227
- XilinxCoolRunnerIIDevice, 220
  - Erase, 223
  - GeneratePermutationTable, 224
  - GetDescription, 224
  - GetPaddingSize, 224
  - GetShiftRegisterDepth, 224
  - GetShiftRegisterWidth, 224
  - instructions, 222
  - IsProgrammed, 224
  - LoadFirmwareImage, 225
  - packages, 223
  - PostInitProbes, 225
  - Program, 226
- XilinxCoolRunnerIIDevice.cpp, 339
- XilinxCoolRunnerIIDevice.h, 340
- XilinxCoolRunnerIIDeviceStatusRegister, 226
- XilinxDevice, 230
  - CreateDevice, 230
- XilinxDevice.cpp, 343
- XilinxDevice.h, 344
- XilinxFPGA.cpp, 344
- XilinxFPGA.h, 345
- XilinxFPGABitstream, 235
- XilinxFPGABitstream.cpp, 345
- XilinxFPGABitstream.h, 345
- XilinxFPGA, 231
  - ParseBitstreamCore, 233
  - ParseBitstreamInternals, 233
  - PostInitProbes, 234
  - ReadingSerialRequiresReset, 234
  - XilinxFPGA, 232
- XilinxSpartan3ADevice, 237
  - deviceids, 240
  - Erase, 241
  - GetDescription, 241
  - GetSerialNumber, 242
  - GetSerialNumberLength, 242
  - GetSerialNumberLengthBits, 242
  - instructions, 240
  - IsProgrammed, 243
  - LoadFirmwareImage, 243
  - ParseBitstreamInternals, 244
  - Program, 244
  - ReadWordConfigRegister, 244
  - XilinxSpartan3ADevice, 241
- XilinxSpartan3ADevice.cpp, 346
- XilinxSpartan3ADevice.h, 346
  - count, 348
  - op, 349
  - type, 349
  - word, 349
- XilinxSpartan3ADeviceConfigurationFrame, 245
  - count, 246
  - op, 246
  - type, 246
- XilinxSpartan3ADeviceStatusRegister, 247

- XilinxSpartan6Device, [248](#)
  - deviceids, [251](#)
  - Erase, [252](#)
  - GetDescription, [252](#)
  - GetSerialNumber, [253](#)
  - GetSerialNumberLength, [253](#)
  - GetSerialNumberLengthBits, [253](#)
  - instructions, [251](#)
  - IsProgrammed, [254](#)
  - LoadFirmwareImage, [254](#)
  - ParseBitstreamInternals, [255](#)
  - Program, [255](#)
  - ReadWordConfigRegister, [256](#)
  - ReadWordsConfigRegister, [256](#)
  - XilinxSpartan6Device, [252](#)
- XilinxSpartan6Device.cpp, [350](#)
- XilinxSpartan6Device.h, [350](#)
  - count, [352](#)
  - op, [352](#)
  - type, [352](#)
  - word, [352](#)
- XilinxSpartan6DeviceConfigurationFrame, [257](#)
  - count, [257](#)
  - op, [258](#)
  - type, [258](#)
- XilinxSpartan6DeviceStatusRegister, [258](#)
- XilinxUltrascaleDevice, [260](#)
  - cmd\_values, [264](#)
  - Erase, [265](#)
  - GetDescription, [265](#)
  - GetSerialNumber, [265](#)
  - GetSerialNumberLength, [266](#)
  - GetSerialNumberLengthBits, [266](#)
  - instructions, [264](#)
  - IsProgrammed, [266](#)
  - LoadFirmwareImage, [267](#)
  - ParseBitstreamInternals, [267](#)
  - Program, [268](#)
  - ReadWordConfigRegister, [268](#)
  - XilinxUltrascaleDevice, [264](#)
- XilinxUltrascaleDevice.cpp, [353](#)
- XilinxUltrascaleDevice.h, [353](#)
  - op, [355](#)
  - type, [355](#)
  - word, [356](#)
- XilinxUltrascaleDeviceConfigurationFrame, [269](#)
  - op, [270](#)
  - type, [270](#)
- XilinxUltrascaleDeviceStatusRegister, [271](#)