

DVB-TX-IRIS

0.1.1

Generated by Doxygen 1.8.6

Fri Oct 28 2016 13:01:11



# Contents

<b>1</b>	<b>DVB-TX-IRIS</b>	<b>1</b>
1.1	Introduction	1
1.1.1	The DVB-T waveform	1
1.1.2	IRIS	2
1.2	Compilation and installation	3
1.3	Choosing a bit rate and the transmission parameters	4
1.4	Choosing frequency, power, gain	4
1.5	Choosing an input TS file	5
1.6	Preparing an Iris XML file	5
1.7	USRP setup	8
1.8	Transmit	8
1.9	Receive and validate	9
1.10	Off-line validation	9
1.11	Bibliography	9
<b>2</b>	<b>Namespace Index</b>	<b>11</b>
2.1	Namespace List	11
<b>3</b>	<b>Hierarchical Index</b>	<b>13</b>
3.1	Class Hierarchy	13
<b>4</b>	<b>Class Index</b>	<b>15</b>
4.1	Class List	15
<b>5</b>	<b>File Index</b>	<b>17</b>
5.1	File List	17
<b>6</b>	<b>Namespace Documentation</b>	<b>19</b>
6.1	iris Namespace Reference	19
6.2	iris::phy Namespace Reference	19
6.2.1	Function Documentation	20
6.2.1.1	IRIS_COMPONENT_EXPORTS	20
6.2.1.2	IRIS_COMPONENT_EXPORTS	20

6.2.1.3	IRIS_COMPONENT_EXPORTS . . . . .	20
6.2.1.4	IRIS_COMPONENT_EXPORTS . . . . .	20
6.2.1.5	IRIS_COMPONENT_EXPORTS . . . . .	20
6.2.1.6	IRIS_COMPONENT_EXPORTS . . . . .	20
6.2.1.7	IRIS_COMPONENT_EXPORTS . . . . .	20
6.2.1.8	IRIS_COMPONENT_EXPORTS . . . . .	20
6.2.1.9	IRIS_COMPONENT_EXPORTS . . . . .	20
6.2.1.10	IRIS_COMPONENT_EXPORTS . . . . .	20
6.2.1.11	IRIS_COMPONENT_EXPORTS . . . . .	20
6.2.1.12	IRIS_COMPONENT_EXPORTS . . . . .	20
6.2.1.13	IRIS_COMPONENT_EXPORTS . . . . .	20
6.2.1.14	IRIS_COMPONENT_EXPORTS . . . . .	20
6.2.1.15	IRIS_COMPONENT_EXPORTS . . . . .	20
<b>7</b>	<b>Class Documentation</b>	<b>21</b>
7.1	iris::phy::Dvbt1BitInterleaverComponent Class Reference . . . . .	21
7.1.1	Detailed Description . . . . .	23
7.1.2	Member Typedef Documentation . . . . .	24
7.1.2.1	ByteVec . . . . .	24
7.1.2.2	ByteVecIt . . . . .	24
7.1.3	Constructor & Destructor Documentation . . . . .	24
7.1.3.1	Dvbt1BitInterleaverComponent . . . . .	24
7.1.3.2	~Dvbt1BitInterleaverComponent . . . . .	25
7.1.4	Member Function Documentation . . . . .	25
7.1.4.1	begin . . . . .	25
7.1.4.2	calculateOutputTypes . . . . .	25
7.1.4.3	destroy . . . . .	25
7.1.4.4	end . . . . .	26
7.1.4.5	initialize . . . . .	26
7.1.4.6	parameterHasChanged . . . . .	26
7.1.4.7	process . . . . .	26
7.1.4.8	registerPorts . . . . .	27
7.1.4.9	setup . . . . .	28
7.1.5	Member Data Documentation . . . . .	28
7.1.5.1	address_v2 . . . . .	28
7.1.5.2	address_v4 . . . . .	29
7.1.5.3	address_v6 . . . . .	29
7.1.5.4	debug_x . . . . .	29
7.1.5.5	hierarchyMode_x . . . . .	30
7.1.5.6	intLength_ . . . . .	30

7.1.5.7	<a href="#">intOffset_</a>	30
7.1.5.8	<a href="#">intRegister_</a>	30
7.1.5.9	<a href="#">nu_</a>	30
7.1.5.10	<a href="#">qamMapping_x</a>	30
7.1.5.11	<a href="#">sampleRate_</a>	30
7.1.5.12	<a href="#">timeStamp_</a>	30
7.2	<a href="#">iris::phy::Dvbt1ConvEncoderComponent Class Reference</a>	31
7.2.1	<a href="#">Detailed Description</a>	32
7.2.2	<a href="#">Member Typedef Documentation</a>	33
7.2.2.1	<a href="#">ByteVec</a>	33
7.2.2.2	<a href="#">ByteVecIt</a>	33
7.2.3	<a href="#">Constructor &amp; Destructor Documentation</a>	33
7.2.3.1	<a href="#">Dvbt1ConvEncoderComponent</a>	33
7.2.3.2	<a href="#">~Dvbt1ConvEncoderComponent</a>	34
7.2.4	<a href="#">Member Function Documentation</a>	34
7.2.4.1	<a href="#">begin</a>	34
7.2.4.2	<a href="#">calculateOutputTypes</a>	34
7.2.4.3	<a href="#">destroy</a>	34
7.2.4.4	<a href="#">end</a>	35
7.2.4.5	<a href="#">initialize</a>	35
7.2.4.6	<a href="#">parameterHasChanged</a>	35
7.2.4.7	<a href="#">process</a>	35
7.2.4.8	<a href="#">registerPorts</a>	36
7.2.4.9	<a href="#">setup</a>	36
7.2.5	<a href="#">Member Data Documentation</a>	36
7.2.5.1	<a href="#">debug_x</a>	36
7.2.5.2	<a href="#">parity_</a>	36
7.2.5.3	<a href="#">sampleRate_</a>	37
7.2.5.4	<a href="#">status_</a>	37
7.2.5.5	<a href="#">timeStamp_</a>	37
7.3	<a href="#">iris::phy::Dvbt1ConvInterleaverComponent Class Reference</a>	37
7.3.1	<a href="#">Detailed Description</a>	40
7.3.2	<a href="#">Member Typedef Documentation</a>	40
7.3.2.1	<a href="#">ByteVec</a>	40
7.3.2.2	<a href="#">ByteVecIt</a>	41
7.3.3	<a href="#">Constructor &amp; Destructor Documentation</a>	41
7.3.3.1	<a href="#">Dvbt1ConvInterleaverComponent</a>	41
7.3.3.2	<a href="#">~Dvbt1ConvInterleaverComponent</a>	41
7.3.4	<a href="#">Member Function Documentation</a>	41
7.3.4.1	<a href="#">begin</a>	41

7.3.4.2	<a href="#">calculateOutputTypes</a>	42
7.3.4.3	<a href="#">destroy</a>	42
7.3.4.4	<a href="#">end</a>	42
7.3.4.5	<a href="#">initialize</a>	42
7.3.4.6	<a href="#">parameterHasChanged</a>	42
7.3.4.7	<a href="#">process</a>	43
7.3.4.8	<a href="#">registerPorts</a>	45
7.3.4.9	<a href="#">setup</a>	45
7.3.5	<a href="#">Member Data Documentation</a>	45
7.3.5.1	<a href="#">b_</a>	45
7.3.5.2	<a href="#">debug_x</a>	45
7.3.5.3	<a href="#">l0_</a>	46
7.3.5.4	<a href="#">l10_</a>	46
7.3.5.5	<a href="#">l11_</a>	46
7.3.5.6	<a href="#">l1_</a>	46
7.3.5.7	<a href="#">l2_</a>	46
7.3.5.8	<a href="#">l3_</a>	46
7.3.5.9	<a href="#">l4_</a>	46
7.3.5.10	<a href="#">l5_</a>	46
7.3.5.11	<a href="#">l6_</a>	47
7.3.5.12	<a href="#">l7_</a>	47
7.3.5.13	<a href="#">l8_</a>	47
7.3.5.14	<a href="#">l9_</a>	47
7.3.5.15	<a href="#">rsOffset_</a>	47
7.3.5.16	<a href="#">sampleRate_</a>	47
7.3.5.17	<a href="#">timeStamp_</a>	47
7.4	<a href="#">iris::phy::Dvbt1FilterComponent Class Reference</a>	47
7.4.1	<a href="#">Detailed Description</a>	50
7.4.2	<a href="#">Member Typedef Documentation</a>	51
7.4.2.1	<a href="#">ByteVec</a>	51
7.4.2.2	<a href="#">ByteVecIt</a>	51
7.4.2.3	<a href="#">Cplx</a>	51
7.4.2.4	<a href="#">CplxVec</a>	51
7.4.2.5	<a href="#">CplxVecIt</a>	51
7.4.2.6	<a href="#">FloatVec</a>	51
7.4.2.7	<a href="#">FloatVecIt</a>	51
7.4.2.8	<a href="#">IntVec</a>	51
7.4.2.9	<a href="#">IntVecIt</a>	51
7.4.3	<a href="#">Constructor &amp; Destructor Documentation</a>	52
7.4.3.1	<a href="#">Dvbt1FilterComponent</a>	52

7.4.3.2	<a href="#">~Dvbt1FilterComponent</a>	52
7.4.4	<a href="#">Member Function Documentation</a>	52
7.4.4.1	<a href="#">begin</a>	52
7.4.4.2	<a href="#">bessel_l0</a>	53
7.4.4.3	<a href="#">calculateOutputTypes</a>	54
7.4.4.4	<a href="#">destroy</a>	54
7.4.4.5	<a href="#">end</a>	54
7.4.4.6	<a href="#">factorial</a>	54
7.4.4.7	<a href="#">filter_design</a>	55
7.4.4.8	<a href="#">initialize</a>	55
7.4.4.9	<a href="#">kaiser_design</a>	56
7.4.4.10	<a href="#">kaiser_window</a>	56
7.4.4.11	<a href="#">parameterHasChanged</a>	57
7.4.4.12	<a href="#">process</a>	57
7.4.4.13	<a href="#">registerPorts</a>	58
7.4.4.14	<a href="#">setup</a>	58
7.4.4.15	<a href="#">sinc</a>	60
7.4.5	<a href="#">Member Data Documentation</a>	60
7.4.5.1	<a href="#">coeffp_</a>	60
7.4.5.2	<a href="#">coeffsFile_x</a>	60
7.4.5.3	<a href="#">debug_x</a>	60
7.4.5.4	<a href="#">filterLength_</a>	60
7.4.5.5	<a href="#">sampleRate_</a>	60
7.4.5.6	<a href="#">sampleRate_x</a>	61
7.4.5.7	<a href="#">sBAttenuation_x</a>	61
7.4.5.8	<a href="#">stopBand_x</a>	61
7.4.5.9	<a href="#">symmetric_</a>	61
7.4.5.10	<a href="#">timeStamp_</a>	61
7.4.5.11	<a href="#">work_</a>	61
7.5	<a href="#">iris::phy::Dvbt1FormatterComponent Class Reference</a>	61
7.5.1	<a href="#">Detailed Description</a>	63
7.5.2	<a href="#">Member Typedef Documentation</a>	63
7.5.2.1	<a href="#">ByteVec</a>	63
7.5.2.2	<a href="#">ByteVecIt</a>	63
7.5.2.3	<a href="#">Cplx</a>	63
7.5.2.4	<a href="#">CplxVec</a>	63
7.5.2.5	<a href="#">CplxVecIt</a>	63
7.5.2.6	<a href="#">ShortVec</a>	64
7.5.2.7	<a href="#">ShortVecIt</a>	64
7.5.3	<a href="#">Constructor &amp; Destructor Documentation</a>	64

7.5.3.1	Dvbt1FormatterComponent	64
7.5.3.2	~Dvbt1FormatterComponent	64
7.5.4	Member Function Documentation	64
7.5.4.1	begin	64
7.5.4.2	calculateOutputTypes	64
7.5.4.3	destroy	65
7.5.4.4	end	65
7.5.4.5	initialize	65
7.5.4.6	parameterHasChanged	65
7.5.4.7	process	65
7.5.4.8	registerPorts	66
7.5.4.9	setup	66
7.5.5	Member Data Documentation	66
7.5.5.1	debug_x	66
7.5.5.2	sampleRate_	66
7.5.5.3	timeStamp_	66
7.6	iris::phy::Dvbt1FramerComponent Class Reference	67
7.6.1	Detailed Description	69
7.6.2	Member Typedef Documentation	71
7.6.2.1	ByteVec	71
7.6.2.2	ByteVecIt	71
7.6.2.3	Cplx	71
7.6.2.4	CplxVec	71
7.6.2.5	CplxVecIt	71
7.6.3	Constructor & Destructor Documentation	71
7.6.3.1	Dvbt1FramerComponent	71
7.6.3.2	~Dvbt1FramerComponent	72
7.6.4	Member Function Documentation	72
7.6.4.1	begin	72
7.6.4.2	calculateOutputTypes	73
7.6.4.3	destroy	73
7.6.4.4	end	73
7.6.4.5	initialize	73
7.6.4.6	parameterHasChanged	74
7.6.4.7	process	74
7.6.4.8	registerPorts	75
7.6.4.9	setup	76
7.6.4.10	t1_tps_generate	76
7.6.5	Member Data Documentation	81
7.6.5.1	blockIndex_	81



7.6.5.2	<a href="#">cellId_x</a>	81
7.6.5.3	<a href="#">cont_pilot_position</a>	81
7.6.5.4	<a href="#">debug_x</a>	82
7.6.5.5	<a href="#">deltaMode_x</a>	82
7.6.5.6	<a href="#">fraOffset_</a>	82
7.6.5.7	<a href="#">fraRegister_</a>	82
7.6.5.8	<a href="#">hpCodeRate_x</a>	82
7.6.5.9	<a href="#">hierarchyMode_x</a>	82
7.6.5.10	<a href="#">inDepthInterleaver_x</a>	82
7.6.5.11	<a href="#">kMax_</a>	83
7.6.5.12	<a href="#">lpCodeRate_x</a>	83
7.6.5.13	<a href="#">nMax_</a>	83
7.6.5.14	<a href="#">ofdmMode_x</a>	83
7.6.5.15	<a href="#">prbs_pilot</a>	83
7.6.5.16	<a href="#">qamMapping_x</a>	83
7.6.5.17	<a href="#">sampleRate_</a>	83
7.6.5.18	<a href="#">timeStamp_</a>	83
7.6.5.19	<a href="#">tps_</a>	84
7.6.5.20	<a href="#">tps_position</a>	84
7.6.5.21	<a href="#">tpsAmpl_</a>	84
7.7	<a href="#">iris::phy::Dvbt1InterpolatorComponent Class Reference</a>	84
7.7.1	<a href="#">Detailed Description</a>	87
7.7.2	<a href="#">Member Typedef Documentation</a>	87
7.7.2.1	<a href="#">ByteVec</a>	87
7.7.2.2	<a href="#">ByteVecIt</a>	87
7.7.2.3	<a href="#">Cplx</a>	87
7.7.2.4	<a href="#">CplxVec</a>	88
7.7.2.5	<a href="#">CplxVecIt</a>	88
7.7.2.6	<a href="#">FloatVec</a>	88
7.7.2.7	<a href="#">FloatVecIt</a>	88
7.7.2.8	<a href="#">IntVec</a>	88
7.7.2.9	<a href="#">IntVecIt</a>	88
7.7.3	<a href="#">Constructor &amp; Destructor Documentation</a>	88
7.7.3.1	<a href="#">Dvbt1InterpolatorComponent</a>	88
7.7.3.2	<a href="#">~Dvbt1InterpolatorComponent</a>	89
7.7.4	<a href="#">Member Function Documentation</a>	89
7.7.4.1	<a href="#">begin</a>	89
7.7.4.2	<a href="#">blackman_sinc</a>	89
7.7.4.3	<a href="#">calculateOutputTypes</a>	90
7.7.4.4	<a href="#">destroy</a>	90

7.7.4.5	end . . . . .	90
7.7.4.6	find_rational_approximation . . . . .	90
7.7.4.7	initialize . . . . .	91
7.7.4.8	interp_response . . . . .	91
7.7.4.9	parameterHasChanged . . . . .	92
7.7.4.10	process . . . . .	92
7.7.4.11	registerPorts . . . . .	93
7.7.4.12	setup . . . . .	93
7.7.4.13	sinc . . . . .	94
7.7.4.14	time_buffer_size . . . . .	94
7.7.5	Member Data Documentation . . . . .	95
7.7.5.1	debug_x . . . . .	95
7.7.5.2	inLength_ . . . . .	95
7.7.5.3	inOffset_ . . . . .	95
7.7.5.4	inReg_ . . . . .	95
7.7.5.5	inSampleRate_x . . . . .	95
7.7.5.6	outSampleRate_x . . . . .	95
7.7.5.7	responseFile_x . . . . .	96
7.7.5.8	sampleRate_ . . . . .	96
7.7.5.9	tiBasepointIndex_ . . . . .	96
7.7.5.10	tiHI_ . . . . .	96
7.7.5.11	tiInsize_ . . . . .	96
7.7.5.12	timeStamp_ . . . . .	96
7.7.5.13	tiOutsize_ . . . . .	96
7.8	iris::phy::Dvbt1MapperComponent Class Reference . . . . .	96
7.8.1	Detailed Description . . . . .	98
7.8.2	Member Typedef Documentation . . . . .	100
7.8.2.1	ByteVec . . . . .	100
7.8.2.2	ByteVecIt . . . . .	100
7.8.2.3	Cplx . . . . .	100
7.8.2.4	CplxVec . . . . .	100
7.8.2.5	CplxVecIt . . . . .	100
7.8.3	Constructor & Destructor Documentation . . . . .	100
7.8.3.1	Dvbt1MapperComponent . . . . .	100
7.8.3.2	~Dvbt1MapperComponent . . . . .	101
7.8.4	Member Function Documentation . . . . .	101
7.8.4.1	begin . . . . .	101
7.8.4.2	calculateOutputTypes . . . . .	101
7.8.4.3	destroy . . . . .	101
7.8.4.4	end . . . . .	101

7.8.4.5	initialize	102
7.8.4.6	parameterHasChanged	102
7.8.4.7	process	102
7.8.4.8	registerPorts	103
7.8.4.9	setup	103
7.8.5	Member Data Documentation	104
7.8.5.1	constel_	104
7.8.5.2	debug_x	104
7.8.5.3	hierarchyMode_x	105
7.8.5.4	qamMapping_x	105
7.8.5.5	sampleRate_	105
7.8.5.6	timeStamp_	105
7.9	iris::phy::Dvbt1NoiseGeneratorComponent Class Reference	105
7.9.1	Detailed Description	107
7.9.2	Member Typedef Documentation	107
7.9.2.1	ByteVec	107
7.9.2.2	ByteVecIt	107
7.9.2.3	Cplx	107
7.9.2.4	CplxVec	107
7.9.2.5	CplxVecIt	107
7.9.2.6	FloatVec	108
7.9.2.7	FloatVecIt	108
7.9.2.8	GaussianGenerator	108
7.9.2.9	IntVec	108
7.9.2.10	IntVecIt	108
7.9.2.11	NormalDistribution	108
7.9.2.12	RandomGenerator	108
7.9.3	Constructor & Destructor Documentation	108
7.9.3.1	Dvbt1NoiseGeneratorComponent	108
7.9.3.2	~Dvbt1NoiseGeneratorComponent	109
7.9.4	Member Function Documentation	109
7.9.4.1	begin	109
7.9.4.2	calculateOutputTypes	109
7.9.4.3	destroy	109
7.9.4.4	end	109
7.9.4.5	initialize	110
7.9.4.6	parameterHasChanged	110
7.9.4.7	process	110
7.9.4.8	registerPorts	110
7.9.4.9	setup	111

7.9.5	Member Data Documentation . . . . .	111
7.9.5.1	blockSize_x . . . . .	111
7.9.5.2	debug_x . . . . .	111
7.9.5.3	frequency_x . . . . .	111
7.9.5.4	gen . . . . .	111
7.9.5.5	sampleRate_ . . . . .	111
7.9.5.6	theta0_ . . . . .	111
7.9.5.7	timeStamp_ . . . . .	112
7.9.5.8	variance_x . . . . .	112
7.10	iris::phy::Dvbt1OfdmModComponent Class Reference . . . . .	112
7.10.1	Detailed Description . . . . .	115
7.10.2	Member Typedef Documentation . . . . .	116
7.10.2.1	ByteVec . . . . .	116
7.10.2.2	ByteVecIt . . . . .	116
7.10.2.3	Cplx . . . . .	116
7.10.2.4	CplxVec . . . . .	116
7.10.2.5	CplxVecIt . . . . .	116
7.10.2.6	FloatVec . . . . .	116
7.10.2.7	FloatVecIt . . . . .	117
7.10.2.8	IntVec . . . . .	117
7.10.2.9	IntVecIt . . . . .	117
7.10.3	Constructor & Destructor Documentation . . . . .	117
7.10.3.1	Dvbt1OfdmModComponent . . . . .	117
7.10.3.2	~Dvbt1OfdmModComponent . . . . .	118
7.10.4	Member Function Documentation . . . . .	118
7.10.4.1	begin . . . . .	118
7.10.4.2	blackman_sinc . . . . .	118
7.10.4.3	calculateOutputTypes . . . . .	119
7.10.4.4	destroy . . . . .	119
7.10.4.5	end . . . . .	119
7.10.4.6	frequency_response_modulus . . . . .	119
7.10.4.7	initialize . . . . .	120
7.10.4.8	parameterHasChanged . . . . .	120
7.10.4.9	powerProcedure_ . . . . .	120
7.10.4.10	process . . . . .	121
7.10.4.11	registerPorts . . . . .	122
7.10.4.12	setup . . . . .	122
7.10.4.13	sinc . . . . .	124
7.10.5	Member Data Documentation . . . . .	124
7.10.5.1	_ampliFactor_ . . . . .	124

7.10.5.2	<a href="#">_precorrFactor_</a>	124
7.10.5.3	<a href="#">ampliFactor_</a>	125
7.10.5.4	<a href="#">dacSampleRate_x</a>	125
7.10.5.5	<a href="#">debug_x</a>	125
7.10.5.6	<a href="#">deltaMode_x</a>	125
7.10.5.7	<a href="#">fft_</a>	125
7.10.5.8	<a href="#">fftBins_</a>	125
7.10.5.9	<a href="#">fftReg_</a>	125
7.10.5.10	<a href="#">inOffset_</a>	125
7.10.5.11	<a href="#">inReg_</a>	126
7.10.5.12	<a href="#">kMax_</a>	126
7.10.5.13	<a href="#">multFactor_</a>	126
7.10.5.14	<a href="#">nBit_</a>	126
7.10.5.15	<a href="#">nBlock_</a>	126
7.10.5.16	<a href="#">nDelta_</a>	126
7.10.5.17	<a href="#">nFft_</a>	126
7.10.5.18	<a href="#">nMax_</a>	126
7.10.5.19	<a href="#">ofdmMode_x</a>	126
7.10.5.20	<a href="#">outPower_x</a>	127
7.10.5.21	<a href="#">powerFile_x</a>	127
7.10.5.22	<a href="#">powerInterval_x</a>	127
7.10.5.23	<a href="#">powerThread_</a>	127
7.10.5.24	<a href="#">precorrFactor_</a>	127
7.10.5.25	<a href="#">runPower_</a>	127
7.10.5.26	<a href="#">sampleRate_</a>	127
7.10.5.27	<a href="#">timeStamp_</a>	127
7.10.5.28	<a href="#">tpsNum_</a>	128
7.11	<a href="#">iris::phy::Dvbt1PuncturerComponent Class Reference</a>	128
7.11.1	<a href="#">Detailed Description</a>	130
7.11.2	<a href="#">Member Typedef Documentation</a>	130
7.11.2.1	<a href="#">ByteVec</a>	130
7.11.2.2	<a href="#">ByteVecIt</a>	131
7.11.3	<a href="#">Constructor &amp; Destructor Documentation</a>	131
7.11.3.1	<a href="#">Dvbt1PuncturerComponent</a>	131
7.11.3.2	<a href="#">~Dvbt1PuncturerComponent</a>	131
7.11.4	<a href="#">Member Function Documentation</a>	131
7.11.4.1	<a href="#">begin</a>	131
7.11.4.2	<a href="#">calculateOutputTypes</a>	132
7.11.4.3	<a href="#">destroy</a>	132
7.11.4.4	<a href="#">end</a>	132

7.11.4.5	initialize	132
7.11.4.6	parameterHasChanged	132
7.11.4.7	process	133
7.11.4.8	registerPorts	134
7.11.4.9	setup	134
7.11.5	Member Data Documentation	134
7.11.5.1	codeRate_x	134
7.11.5.2	debug_x	135
7.11.5.3	punOffset_	135
7.11.5.4	punPeriodIn_	135
7.11.5.5	punPeriodOut_	135
7.11.5.6	punRegister_	135
7.11.5.7	sampleRate_	135
7.11.5.8	timeStamp_	135
7.12	iris::phy::Dvbt1RSEncoderComponent Class Reference	135
7.12.1	Detailed Description	138
7.12.2	Member Typedef Documentation	139
7.12.2.1	ByteVec	139
7.12.2.2	ByteVecIt	139
7.12.3	Constructor & Destructor Documentation	139
7.12.3.1	Dvbt1RSEncoderComponent	139
7.12.3.2	~Dvbt1RSEncoderComponent	139
7.12.4	Member Function Documentation	139
7.12.4.1	begin	140
7.12.4.2	calculateOutputTypes	140
7.12.4.3	destroy	140
7.12.4.4	end	140
7.12.4.5	initialize	140
7.12.4.6	modnn	141
7.12.4.7	packetEncode	141
7.12.4.8	parameterHasChanged	141
7.12.4.9	process	142
7.12.4.10	registerPorts	142
7.12.4.11	setup	143
7.12.5	Member Data Documentation	143
7.12.5.1	alpha_	143
7.12.5.2	debug_x	143
7.12.5.3	gg_	143
7.12.5.4	index_	144
7.12.5.5	rsCodeWord_	144

7.12.5.6	sampleRate_ . . . . .	144
7.12.5.7	timeStamp_ . . . . .	144
7.12.5.8	tsOffset_ . . . . .	144
7.13	iris::phy::Dvbt1ScramblerComponent Class Reference . . . . .	145
7.13.1	Detailed Description . . . . .	147
7.13.2	Member Typedef Documentation . . . . .	147
7.13.2.1	ByteVec . . . . .	147
7.13.2.2	ByteVecIt . . . . .	148
7.13.3	Constructor & Destructor Documentation . . . . .	148
7.13.3.1	Dvbt1ScramblerComponent . . . . .	148
7.13.3.2	~Dvbt1ScramblerComponent . . . . .	148
7.13.4	Member Function Documentation . . . . .	148
7.13.4.1	begin . . . . .	148
7.13.4.2	calculateOutputTypes . . . . .	148
7.13.4.3	destroy . . . . .	149
7.13.4.4	end . . . . .	149
7.13.4.5	initialize . . . . .	149
7.13.4.6	parameterHasChanged . . . . .	149
7.13.4.7	process . . . . .	150
7.13.4.8	registerPorts . . . . .	150
7.13.4.9	setup . . . . .	150
7.13.5	Member Data Documentation . . . . .	151
7.13.5.1	debug_x . . . . .	151
7.13.5.2	doneBytes_ . . . . .	151
7.13.5.3	reportInterval_x . . . . .	151
7.13.5.4	sampleRate_ . . . . .	151
7.13.5.5	scramblerOffset_ . . . . .	151
7.13.5.6	scramblerPrbs_ . . . . .	151
7.13.5.7	start_ . . . . .	152
7.13.5.8	timeStamp_ . . . . .	152
7.14	iris::phy::Dvbt1SymbolInterleaverComponent Class Reference . . . . .	152
7.14.1	Detailed Description . . . . .	154
7.14.2	Member Typedef Documentation . . . . .	155
7.14.2.1	ByteVec . . . . .	155
7.14.2.2	ByteVecIt . . . . .	155
7.14.3	Constructor & Destructor Documentation . . . . .	155
7.14.3.1	Dvbt1SymbolInterleaverComponent . . . . .	155
7.14.3.2	~Dvbt1SymbolInterleaverComponent . . . . .	155
7.14.4	Member Function Documentation . . . . .	156
7.14.4.1	begin . . . . .	156

7.14.4.2	<a href="#">calculateOutputTypes</a>	156
7.14.4.3	<a href="#">destroy</a>	156
7.14.4.4	<a href="#">end</a>	156
7.14.4.5	<a href="#">initialize</a>	157
7.14.4.6	<a href="#">parameterHasChanged</a>	157
7.14.4.7	<a href="#">process</a>	157
7.14.4.8	<a href="#">registerPorts</a>	158
7.14.4.9	<a href="#">setup</a>	158
7.14.5	<a href="#">Member Data Documentation</a>	158
7.14.5.1	<a href="#">debug_x</a>	158
7.14.5.2	<a href="#">eo_</a>	159
7.14.5.3	<a href="#">H_2K_</a>	159
7.14.5.4	<a href="#">H_4K_</a>	159
7.14.5.5	<a href="#">H_8K_</a>	159
7.14.5.6	<a href="#">ofdmMode_x</a>	159
7.14.5.7	<a href="#">sampleRate_</a>	159
7.14.5.8	<a href="#">siLength_</a>	159
7.14.5.9	<a href="#">siOffset_</a>	159
7.14.5.10	<a href="#">siRegister_</a>	160
7.14.5.11	<a href="#">timeStamp_</a>	160
7.15	<a href="#">iris::phy::Dvbt1UsrpTxComponent Class Reference</a>	160
7.15.1	<a href="#">Detailed Description</a>	162
7.15.2	<a href="#">Constructor &amp; Destructor Documentation</a>	163
7.15.2.1	<a href="#">Dvbt1UsrpTxComponent</a>	163
7.15.2.2	<a href="#">~Dvbt1UsrpTxComponent</a>	164
7.15.3	<a href="#">Member Function Documentation</a>	164
7.15.3.1	<a href="#">calculateOutputTypes</a>	164
7.15.3.2	<a href="#">initialize</a>	165
7.15.3.3	<a href="#">parameterHasChanged</a>	166
7.15.3.4	<a href="#">process</a>	167
7.15.3.5	<a href="#">registerPorts</a>	168
7.15.3.6	<a href="#">usrpThreadProcedure</a>	168
7.15.4	<a href="#">Member Data Documentation</a>	169
7.15.4.1	<a href="#">antenna_x</a>	169
7.15.4.2	<a href="#">args_x</a>	169
7.15.4.3	<a href="#">bufferSize_x</a>	169
7.15.4.4	<a href="#">bufs_</a>	169
7.15.4.5	<a href="#">bw_x</a>	169
7.15.4.6	<a href="#">condR_</a>	170
7.15.4.7	<a href="#">condW_</a>	170



7.15.4.8	<a href="#">currentRead_</a>	170
7.15.4.9	<a href="#">currentWrite_</a>	170
7.15.4.10	<a href="#">fixLoOffset_x</a>	170
7.15.4.11	<a href="#">fmt_x</a>	170
7.15.4.12	<a href="#">frequency_x</a>	170
7.15.4.13	<a href="#">fulls_</a>	170
7.15.4.14	<a href="#">gain_x</a>	171
7.15.4.15	<a href="#">inBuf_</a>	171
7.15.4.16	<a href="#">mut_</a>	171
7.15.4.17	<a href="#">mutR_</a>	171
7.15.4.18	<a href="#">mutW_</a>	171
7.15.4.19	<a href="#">numBuffers_x</a>	171
7.15.4.20	<a href="#">rate_x</a>	171
7.15.4.21	<a href="#">ref_x</a>	171
7.15.4.22	<a href="#">runUsrcp_</a>	172
7.15.4.23	<a href="#">streaming_x</a>	172
7.15.4.24	<a href="#">subDev_x</a>	172
7.15.4.25	<a href="#">txStream_</a>	172
7.15.4.26	<a href="#">usrp_</a>	172
7.15.4.27	<a href="#">usrpThread_</a>	172
<b>8</b>	<b>File Documentation</b>	<b>173</b>
8.1	<a href="#">/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1BitInterleaver/Dvbt1BitInterleaver-Component.cpp File Reference</a>	173
8.1.1	<a href="#">Detailed Description</a>	173
8.1.2	<a href="#">COPYRIGHT</a>	174
8.1.3	<a href="#">LICENSE</a>	174
8.1.4	<a href="#">DESCRIPTION</a>	174
8.2	<a href="#">/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1BitInterleaver/Dvbt1BitInterleaver-Component.h File Reference</a>	174
8.2.1	<a href="#">Detailed Description</a>	175
8.2.2	<a href="#">COPYRIGHT</a>	175
8.2.3	<a href="#">LICENSE</a>	175
8.2.4	<a href="#">DESCRIPTION</a>	176
8.3	<a href="#">/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1ConvEncoder/Dvbt1ConvEncoder-Component.cpp File Reference</a>	176
8.3.1	<a href="#">Detailed Description</a>	176
8.3.2	<a href="#">COPYRIGHT</a>	177
8.3.3	<a href="#">LICENSE</a>	177
8.3.4	<a href="#">DESCRIPTION</a>	177
8.3.5	<a href="#">Macro Definition Documentation</a>	177

8.3.5.1	g1	177
8.3.5.2	g2	177
8.4	/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1ConvEncoder/Dvbt1ConvEncoder- Component.h File Reference	177
8.4.1	Detailed Description	178
8.4.2	COPYRIGHT	179
8.4.3	LICENSE	179
8.4.4	DESCRIPTION	179
8.5	/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1ConvInterleaver/Dvbt1ConvInterleaver- Component.cpp File Reference	179
8.5.1	Detailed Description	180
8.5.2	COPYRIGHT	180
8.5.3	LICENSE	180
8.5.4	DESCRIPTION	180
8.6	/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1ConvInterleaver/Dvbt1ConvInterleaver- Component.h File Reference	180
8.6.1	Detailed Description	181
8.6.2	COPYRIGHT	182
8.6.3	LICENSE	182
8.6.4	DESCRIPTION	182
8.7	/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1Filter/Dvbt1FilterComponent.cpp File Reference	182
8.7.1	Detailed Description	183
8.7.2	COPYRIGHT	183
8.7.3	LICENSE	183
8.7.4	DESCRIPTION	183
8.7.5	Macro Definition Documentation	183
8.7.5.1	MAX_FILTER_LENGTH	183
8.7.5.2	MAX_FILTER_LENGTH_2	184
8.8	/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1Filter/Dvbt1FilterComponent.h File Refer- ence	184
8.8.1	Detailed Description	185
8.8.2	COPYRIGHT	185
8.8.3	LICENSE	185
8.8.4	DESCRIPTION	185
8.9	/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1Formatter/Dvbt1FormatterComponent.cpp File Reference	185
8.9.1	Detailed Description	186
8.9.2	COPYRIGHT	186
8.9.3	LICENSE	186
8.9.4	DESCRIPTION	186

8.10	/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1Formatter/Dvbt1FormatterComponent.h File Reference	187
8.10.1	Detailed Description	188
8.10.2	COPYRIGHT	188
8.10.3	LICENSE	188
8.10.4	DESCRIPTION	188
8.11	/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1Framer/Dvbt1FramerComponent.cpp File Reference	188
8.11.1	Detailed Description	189
8.11.2	COPYRIGHT	189
8.11.3	LICENSE	189
8.11.4	DESCRIPTION	189
8.11.5	Macro Definition Documentation	189
8.11.5.1	T1_K_BCH	189
8.11.5.2	T1_N_BCH	190
8.11.5.3	T1_PIL_AMPL	190
8.11.5.4	T1_TPS_AMPL	190
8.12	/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1Framer/Dvbt1FramerComponent.h File Reference	190
8.12.1	Detailed Description	191
8.12.2	COPYRIGHT	191
8.12.3	LICENSE	191
8.12.4	DESCRIPTION	192
8.12.5	Macro Definition Documentation	192
8.12.5.1	T1_BLOCKS_PER_FRAME	192
8.12.5.2	T1_FRAMES_PER_SUPERFRAME	192
8.13	/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1Interpolator/Dvbt1InterpolatorComponent.cpp File Reference	192
8.13.1	Detailed Description	193
8.13.2	COPYRIGHT	193
8.13.3	LICENSE	193
8.13.4	DESCRIPTION	193
8.14	/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1Interpolator/Dvbt1InterpolatorComponent.h File Reference	193
8.14.1	Detailed Description	195
8.14.2	COPYRIGHT	195
8.14.3	LICENSE	195
8.14.4	DESCRIPTION	195
8.14.5	Macro Definition Documentation	195
8.14.5.1	T1_RESAMPLE_ORDER	195
8.15	/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1Mapper/Dvbt1MapperComponent.cpp File Reference	195

8.15.1 Detailed Description . . . . .	196
8.15.2 COPYRIGHT . . . . .	196
8.15.3 LICENSE . . . . .	196
8.15.4 DESCRIPTION . . . . .	196
8.16 /home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1Mapper/Dvbt1MapperComponent.h File Reference . . . . .	197
8.16.1 Detailed Description . . . . .	198
8.16.2 COPYRIGHT . . . . .	198
8.16.3 LICENSE . . . . .	198
8.16.4 DESCRIPTION . . . . .	198
8.17 /home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1NoiseGenerator/Dvbt1NoiseGenerator-Component.cpp File Reference . . . . .	198
8.17.1 Detailed Description . . . . .	199
8.17.2 COPYRIGHT . . . . .	199
8.17.3 LICENSE . . . . .	199
8.17.4 DESCRIPTION . . . . .	199
8.18 /home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1NoiseGenerator/Dvbt1NoiseGenerator-Component.h File Reference . . . . .	199
8.18.1 Detailed Description . . . . .	200
8.18.2 COPYRIGHT . . . . .	201
8.18.3 LICENSE . . . . .	201
8.18.4 DESCRIPTION . . . . .	201
8.19 /home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1OfdmMod/Dvbt1OfdmModComponent.cpp File Reference . . . . .	201
8.19.1 Detailed Description . . . . .	202
8.19.2 COPYRIGHT . . . . .	202
8.19.3 LICENSE . . . . .	202
8.19.4 DESCRIPTION . . . . .	202
8.19.5 Macro Definition Documentation . . . . .	202
8.19.5.1 WAKEUPINTERVALMS . . . . .	202
8.20 /home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1OfdmMod/Dvbt1OfdmModComponent.h File Reference . . . . .	202
8.20.1 Detailed Description . . . . .	204
8.20.2 COPYRIGHT . . . . .	204
8.20.3 LICENSE . . . . .	204
8.20.4 DESCRIPTION . . . . .	204
8.20.5 Macro Definition Documentation . . . . .	204
8.20.5.1 T1_BLOCKS_PER_FRAME . . . . .	204
8.20.5.2 T1_FRAMES_PER_SUPERFRAME . . . . .	204
8.20.5.3 T1_RESAMPLE_ORDER . . . . .	204
8.21 /home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1Puncturer/Dvbt1PuncturerComponent.cpp File Reference . . . . .	205

8.21.1 Detailed Description . . . . .	205
8.21.2 COPYRIGHT . . . . .	205
8.21.3 LICENSE . . . . .	205
8.21.4 DESCRIPTION . . . . .	206
8.22 /home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1Puncturer/Dvbt1PuncturerComponent.h File Reference . . . . .	206
8.22.1 Detailed Description . . . . .	207
8.22.2 COPYRIGHT . . . . .	207
8.22.3 LICENSE . . . . .	207
8.22.4 DESCRIPTION . . . . .	207
8.23 /home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1REncoder/Dvbt1REncoderComponent.cpp File Reference . . . . .	207
8.23.1 Detailed Description . . . . .	208
8.23.2 COPYRIGHT . . . . .	208
8.23.3 LICENSE . . . . .	208
8.23.4 DESCRIPTION . . . . .	208
8.24 /home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1REncoder/Dvbt1REncoderComponent.h File Reference . . . . .	209
8.24.1 Detailed Description . . . . .	210
8.24.2 COPYRIGHT . . . . .	210
8.24.3 LICENSE . . . . .	210
8.24.4 DESCRIPTION . . . . .	210
8.24.5 Macro Definition Documentation . . . . .	211
8.24.5.1 RS_PACKET_SIZE . . . . .	211
8.24.5.2 T1_A0 . . . . .	211
8.24.5.3 T1_CLEAR . . . . .	211
8.24.5.4 T1_KK . . . . .	211
8.24.5.5 T1_MM . . . . .	211
8.24.5.6 T1_NN . . . . .	211
8.24.5.7 T1_NN_KK . . . . .	211
8.24.5.8 TS_PACKET_SIZE . . . . .	212
8.25 /home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1Scrambler/Dvbt1ScramblerComponent.cpp File Reference . . . . .	212
8.25.1 Detailed Description . . . . .	212
8.25.2 COPYRIGHT . . . . .	212
8.25.3 LICENSE . . . . .	213
8.25.4 DESCRIPTION . . . . .	213
8.26 /home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1Scrambler/Dvbt1ScramblerComponent.h File Reference . . . . .	213
8.26.1 Detailed Description . . . . .	214
8.26.2 COPYRIGHT . . . . .	214

8.26.3	LICENSE	214
8.26.4	DESCRIPTION	215
8.27	/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1SymbolInterleaver/Dvbt1SymbolInterleaver-Component.cpp File Reference	215
8.27.1	Detailed Description	215
8.27.2	COPYRIGHT	215
8.27.3	LICENSE	216
8.27.4	DESCRIPTION	216
8.28	/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1SymbolInterleaver/Dvbt1SymbolInterleaver-Component.h File Reference	216
8.28.1	Detailed Description	217
8.28.2	COPYRIGHT	217
8.28.3	LICENSE	217
8.28.4	DESCRIPTION	218
8.29	/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1UsrpTx/Dvbt1UsrpTxComponent.cpp File Reference	218
8.29.1	Detailed Description	218
8.29.2	COPYRIGHT	219
8.29.3	LICENSE	219
8.29.4	Macro Definition Documentation	219
8.29.4.1	dbgprintf	219
8.29.4.2	DUMP_STATUS	219
8.30	/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1UsrpTx/Dvbt1UsrpTxComponent.h File Reference	219
8.30.1	Detailed Description	220
8.30.2	COPYRIGHT	221
8.30.3	LICENSE	221
8.30.4	DESCRIPTION	221
8.31	Main_Page.txt File Reference	221
<b>9</b>	<b>Example Documentation</b>	<b>223</b>
9.1	dvbt1chain_ofdmmod_filter_spectrum.xml	223
<b>Index</b>		<b>226</b>

# Chapter 1

## DVB-TX-IRIS

### 1.1 Introduction

This package contains the components which may be used to create DVB-T radios with the IRIS software radio framework.

#### 1.1.1 The DVB-T waveform

DVB-T uses orthogonal frequency division multiplexing (OFDM) symbols with cyclic prefix in order to deliver the transmitted data over the communication channel. OFDM symbols are grouped in frames (composed of  $N_F = 68$  OFDM symbols) and superframes (composed of 4 frames): the superframe can be considered to represent a basic group of data, as it always carries an integer number of transport stream (TS) packets, which constitute the payload of DVB-T and carry compressed video and audio streams. The base-band (BB) signal samples can be expressed as

$$\tilde{s}[n] = \sum_{m=0}^{+\infty} \sum_{l=0}^{N_F-1} z_{m,l}[n] = \sum_{m=0}^{+\infty} \sum_{l=0}^{N_F-1} \sum_{k=0}^{K-1} c_{m,l,k} G_k \psi_{m,l,k}[n]$$

where  $m$  represents the frame index,  $l$  is the OFDM symbol index,  $k$  is the subcarrier index,  $K$  is the number of active carriers (depending on the transmission mode), and  $N_F$  is the number of OFDM symbols per frame; the data transported over each carrier is given by  $c_{m,l,k}$  and it is a QAM (quadrature amplitude modulation) mapped constellation symbol, carrying  $v$  bits per symbol;  $G_k$  is a carrier amplitude weighting factor that can be used to precompensate linear distortions introduced by the transmitter ( $G_k = 1$  in case of no distortions), and  $z_{m,l}[n]$  is the OFDM symbol in time. The modulation is performed using  $K$  out of NFFT orthogonal carriers  $\psi_{m,l,k}[n]$ , expressed as

$$\psi_{m,l,k}[n] = e^{j2\pi \frac{k-K_2}{N_{\text{FFT}}}(n-N_G-(l+mN_F)N_S)} \cdot \Pi_{N_S}[n - (l + mN_F)N_S]$$

where  $K_2 = K/2$ ,  $N_G$  is the number of samples of the guard interval,  $N_S = N_{\text{FFT}} + N_G$  is the total number of samples of the OFDM symbol, and  $\Pi_{N_S}[n]$  is the boxcar window, which is equal to 1 in  $[0, N_S - 1]$  and to 0 elsewhere. The BB samples are then converted into the analog domain using a sample time  $T_{s,\text{DVBT}}$  that depends on the bandwidth of the DVB-T configuration. The sample rate  $f_{s,\text{DVBT}} = 1/T_{s,\text{DVBT}}$ , can be replaced by the DAC sample rate  $f_{s,\text{DAC}} = 1/T_{s,\text{DAC}}$ , as expressed by

$$\tilde{s}(t) = \sum_{n=0}^{\infty} \tilde{s}[n] h(t - nT_{s,\text{DVBT}}) = \sum_{n=0}^{\infty} \tilde{y}[n] h_I(t - nT_{s,\text{DAC}})$$

where  $h(t) = T_{s,\text{DVBT}} \text{sinc}(\Pi t / T_{s,\text{DVBT}})$  is the ideal BB reconstruction filter,  $h_I(t)$  is the DAC output filter, and  $\tilde{y}[n]$  is the signal  $\tilde{s}[n]$  resampled to the DAC sample rate. Eventually, the analog signal is up-converted, using a quadrature modulator, to the RF carrier frequency,  $f_c$ , as

$$s(t) = \text{Re} \{ \tilde{s}(t) e^{j2\pi f_c t} \}$$

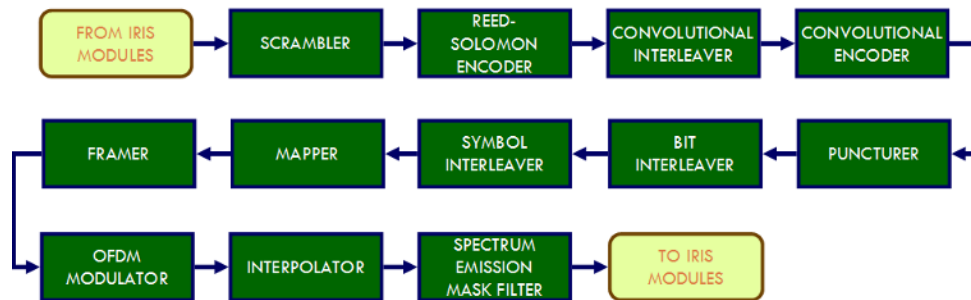


Figure 1.1: DVB-T transmission scheme.

### 1.1.2 IRIS

The DVB-TX-IRIS extension is an extension of the Iris framework. Iris is a software architecture for building highly reconfigurable radio networks using a component-based design. The architecture comprises two repositories - Iris\_Core and Iris\_Modules. Iris\_Core contains the core part of the architecture such as parsers, managers, and engines. Iris\_Modules contain the components which can be used to create a software radio such as PHY-layer components and radio controllers. The Iris architecture, written in C++, supports all layers of the network stack and provides a platform for the development of not only reconfigurable point-to-point radio links, but complete networks of reconfigurable radios. Individual radios are described using an XML document. This lists the components which comprise the radio, gives the values to be used for their parameters and describes the connections between them. Iris was originally developed by CTVR, The Telecommunications Research Centre, based at University of Dublin, Trinity College. In 2013, it was released under the LGPL v3 license and is currently managed by Software Radio Systems (<http://www.softwareradiosystems.com>).

Since DVB-TX-IRIS extends Iris functionalities, there are shared requirements that should be satisfied from the software point of view. In particular, they are:

- Essential SW
  - Ubuntu Linux OS 32/64 bit (<http://www.ubuntu.org>), release 14.04 or later
  - CMake 2.6 or later (<http://www.cmake.org/>), an automated software build and test environment for C/C++
  - Boost 1.46 or later (<http://www.boost.org/>), an extensive collection of C++ libraries for accelerating common software tasks
  - Iris\_Core ([http://www.hostedredmine.com/projects/iris\\_software\\_radio/wiki](http://www.hostedredmine.com/projects/iris_software_radio/wiki)), the core system of the Iris framework
  - Iris\_Modules ([http://www.hostedredmine.com/projects/iris\\_software\\_radio/wiki](http://www.hostedredmine.com/projects/iris_software_radio/wiki)), additional modules for the Iris framework
  - FFTW (<http://www.fftw.org/>), a powerful C/C++ library for FFT transforms
  - UHD (<http://code.ettus.com/redmine/ettus/projects/uhd/wiki>), needed for the connection to USRP hardware
- Optional SW
  - Qt 4.8 (<http://qt-project.org/>), used for building graphical widgets
  - Qwt 6 (<http://qwt.sourceforge.net/>), used for building graphical widgets
  - Liquid-DSP (<https://github.com/jgaeddert/liquid-dsp>), for some PHY components



- Google Protocol Buffers (<https://developers.google.com/protocol-buffers/>), for some Stack components
- Python (<http://www.python.org/>), for the PythonPlotter widget
- Octave (<http://www.octave.org/>), for recreating the test vectors used during the testing phase of the build, and for running complete TX/RX simulations
- Matlab (<http://www.mathworks.com/>), for the MatlabTemplate PHY component and Matlab-Plotter widget
- Doxygen (<http://www.doxygen.org/>), for the documentation
- tzap (dvb-apps package) and w\_scan (w-scan package), used for real-time stream quality testing with DVB-T USB receivers

From the hardware point of view, the following items are required:

- Essential HW
  - A workstation or laptop PC equipped with a multicore CPU clocked at 2 GHz or more, 4 GB of RAM, 20 GB of free disk space, and a free Gigabit Ethernet connection
  - An Ettus USRP N210 equipped with an UHF/VHF capable daughterboard (such as the SBX or SBX120)
  - A UHF/VHF antenna (preferably directional antenna for longer communication range)
  - A DVB-T capable receiver (such as a TV set, a set-top box, or an USB dongle, provided with indoor reception antenna)
- Optional HW
  - A spectrum analyzer for verifying the spectrum of the emitted DVB-T signal

## 1.2 Compilation and installation

Provided that the essential SW requirements are satisfied, the following steps are needed to successfully compile DVB-TX-IRIS:

- The DVB-TX-IRIS code can be downloaded using SVN or GIT from the official GitHub page of the WiSHFUL project, <https://github.com/wishful-project>. For instance, using subversion, the command is

```
svn checkout http://url.iris.repository/dti_wishful
```

- Now cd into the project directory, and create a folder named "build", where the extension will be compiled

```
cd dti_wishful
```

```
mkdir build
```

```
cd build
```

- Invoke cmake to prepare the build environment

```
cmake ..
```

- Invoke make to build the extension

```
make
```

- Optionally, you can benchmark the speed of the extension components and test their correct operation with

```
make benchmark
```

```
make test
```

- If the previous steps ended with success, now you can install the extension by typing

```
sudo make install
```

- If you want to build also the documentation HTML manual, then you must execute doxygen in the "doc" folder

```
cd . .
cd doc
doxygen
```

### 1.3 Choosing a bit rate and the transmission parameters

The DVB-T system is designed to convey a constant bit rate payload. Thus, if the video and audio sources, after compression, have a bit rate higher or lower than the expected value, the SDR modulator will fail in delivering an high quality stream to the receivers. The connection among bit rates and transmission parameters is shown in Table III. There, for any combination of modulation, cyclic prefix, and code rate, there is a corresponding value that should be used to generate the TS. We also highlight that the OFDM mode (2K, 8K) does not influence the bit rate. Summarizing, the transmission parameters could be chosen according to the following guidelines:

- **Cyclic prefix:** this parameter determines the ruggedness of the DVB-T signal with respect to RF channel impairments such as multipath and echoes. Choose a large value (1/4) if delayed signal echoes are expected, especially when covering large areas (width on the order of kms). Differently, the smallest value (1/32) is sufficient for a localized transmission covering a range of few hundreds of metres.
- **OFDM mode:** as explained before, this parameter does not impact on the payload bit rate. However, a 2K OFDM mode is to be preferred if the receivers are preferentially of nomadic or mobile type, since in this case the harmful effect of the Doppler spread (due to the relative motion of the transmitter and receiver) on OFDM carriers orthogonality is minimized. Moreover, 2K is preferred for covering relatively small areas. On the other side, 8K can be used when covering large areas and the receivers are expected to be of a static, fixed type.
- **Modulation order:** this parameter selects the spectral efficiency of the system, thus larger modulation orders (64-QAM) will allow higher bit rate payloads to be delivered. Concurrently, higher modulation orders also require important values of S/N ratio at the receivers, in order to have a successful reception of the TV signal. On the other side, smaller modulation orders (4-QAM, 16-QAM) have lower requests in terms of S/N ratio, but they are not as spectrally efficient.
- **Code rate:** this is the other parameter that concurs to determine the spectral efficiency of the system. High values of code rate (e.g., 7/8) allow very high bit rate payloads, but protect less the signal from unwanted interferences. Lower code rates (such as 1/2) protect the signal very well from noise and other disruptive impairments, but they also lower the spectral efficiency.

We conclude this subsection by recalling that, anyway, the choice of the bit rate depends also on the quality and quantity of the TV programs that will be included in the multiplexed TS.

### 1.4 Choosing frequency, power, gain

These three parameters do not concur in the modification of the payload bit rate, but they are important as well.

- **Frequency:** the emission frequency should be chosen in the range that is commonly used by TV receivers in the VHF-UHF bands. Choosing an emission frequency outside of this range could result in the impossibility to receive this signal on common TV receivers. However, it should be noted that an official permit of the National or Regional Communications Authority must be accorded before broadcasting at such frequencies. The relationship between a DVB-T UHF frequency channel  $i$  and its central frequency (for 8 MHz systems) is

$$474 + i \times 8\text{MHz}, i = 0, 1, 2, \dots$$

- **Power:** this parameter decides the power of the digital signal generated by the SDR DVB-T modulator. This value can be overridden by applying proper amplification gains in the transmission chain, nonetheless it should be chosen carefully. In fact, a typical value of 30-50 (%) is recommended, since this will result in a digital signal with a smaller dynamic range, which will produce fewer distortions during the D/A conversion stage (i.e., less clipping noise).

- Gain: this parameter is related to the analog amplification stage of the USRP. Smaller values (0-5 dB) will result in a scarcely amplified signal, that will not cover a large area, but it will be less distorted since the amplifier is working in the linear portion of its amplification characteristic curve. Higher values (15-20 dB) will allow covering larger areas at the expense of the spectral flatness and purity of the emitted RF signal. The highest values should be avoided, since the additional emitted power is obtained by operating in a highly nonlinear portion of the amplifier characteristic, degrading completely the main spectrum and its surrounding frequencies.

## 1.5 Choosing an input TS file

Once the transmission configuration has been selected, the input TS can be chosen among one of those that are already provided for this purpose, and that can be downloaded from

<http://dante.diei.unipg.it/~baruffa/WiSHFUL/>

These 9 TS files have been generated with a payload bit rate and transmission parameters configuration that can be extracted directly from the file names. In case that none of the files above satisfies the selected payload bit rate, then it is possible to generate a corresponding TS using the OpenCaster and FFmpeg based procedure. To this purpose, example scripts that can be used to compress and multiplex the input clips are included in the "script/dvbt-TS" folder: there is a batch file (ts\_example.bat) that can be used to generate the program and transport streams using the selected compression methods, as well as a python file (ts\_example.py) that can be used to generate the DVB-T TS tables.

## 1.6 Preparing an Iris XML file

The chosen transmission configuration (input file and transmission parameters) must be specified into an XML document, following the Iris framework XML syntax. It is suggested to copy and modify one of the configuration files that are included in the "examples/dvbt" folder. For instance, the file named "demo\_typical\_8K.xml" already contains the configuration parameters necessary to perform the transmission of the typical TS. The syntax of the XML file is pretty self-explanatory, and the values associated to parameters such as OFDM mode, cyclic prefix, code rate, etc., can be modified to suit one's needs: during this step, please be advised that the same parameters could have to be modified inside of several components/engines. The following configuration is excerpted from the XML file used to configure DVB-TX-IRIS for performing the transmission of the typical TS, "demo\_typical\_8K.xml".

```

001 <?xml version="1.0" encoding="utf-8" ?>
002
003 <softwareradio name="Radio1">
004
005   <engine name="phyengine1" class="phyengine">
006
007     <component name="filerawreader1" class="filerawreader">
008       <parameter name="filename" value="hd3typ.ts">
009       <parameter name="blocksize" value="18800">
010       <parameter name="datatype" value="uint8_t">
011       <port name="output1" class="output">
012     </component>
013
014   </engine>
015   <engine name="phyengine2" class="phyengine">
016
017     <component name="dvbt1scrambler1" class="dvbt1scrambler">
018       <parameter name="debug" value="false">
019       <parameter name="reportinterval" value="5">
020       <port name="input1" class="input">
021       <port name="output1" class="output">
022     </component>
023
024   </engine>

```

```

025 <engine name="phyengine3" class="phyengine">
026
027   <component name="dvbtlrsencoder1" class="dvbtlrsencoder">
028     <parameter name="debug" value="false">
029     <port name="input1" class="input">
030     <port name="output1" class="output">
031   </component>
032
033 </engine>
034 <engine name="phyengine4" class="phyengine">
035
036   <component name="dvbtlconvinterleaver1" class="dvbtlconvinterleaver">
037     <parameter name="debug" value="false">
038     <port name="input1" class="input">
039     <port name="output1" class="output">
040   </component>
041
042 </engine>
043 <engine name="phyengine5" class="phyengine">
044
045   <component name="dvbtlconvencoder1" class="dvbtlconvencoder">
046     <parameter name="debug" value="false">
047     <port name="input1" class="input">
048     <port name="output1" class="output">
049   </component>
050
051 </engine>
052 <engine name="phyengine6" class="phyengine">
053
054   <component name="dvbtlpuncturer1" class="dvbtlpuncturer">
055     <parameter name="debug" value="false">
056     <parameter name="coderate" value="34">
057     <port name="input1" class="input">
058     <port name="output1" class="output">
059   </component>
060
061 </engine>
062 <engine name="phyengine7" class="phyengine">
063
064   <component name="dvbtlbitinterleaver1" class="dvbtlbitinterleaver">
065     <parameter name="debug" value="false">
066     <parameter name="qammapping" value="64">
067     <parameter name="hierarchymode" value="0">
068     <port name="input1" class="input">
069     <port name="output1" class="output">
070   </component>
071
072 </engine>
073 <engine name="phyengine8" class="phyengine">
074
075   <component name="dvbtlsymbolinterleaver1" class="dvbtlsymbolinterleaver">
076     <parameter name="debug" value="false">
077     <parameter name="ofdmmode" value="8192">
078     <port name="input1" class="input">
079     <port name="output1" class="output">
080   </component>
081
082 </engine>
083 <engine name="phyengine9" class="phyengine">
084
085   <component name="dvbtlmapper1" class="dvbtlmapper">
086     <parameter name="debug" value="false">
087     <parameter name="qammapping" value="64">

```

```

088     <parameter name="hierarchymode" value="0">
089     <port name="input1" class="input">
090     <port name="output1" class="output">
091 </component>
092
093 </engine>
094 <engine name="phyengine10" class="phyengine">
095
096   <component name="dvbtlframer1" class="dvbtlframer">
097     <parameter name="debug" value="false">
098     <parameter name="ofdmmod" value="8192">
099     <parameter name="qammapping" value="64">
100     <parameter name="hierarchymode" value="0">
101     <parameter name="cellid" value="-1">
102     <parameter name="hpcoderate" value="34">
103     <parameter name="indepthinterleaver" value="false">
104     <parameter name="deltamode" value="4">
105     <port name="input1" class="input">
106     <port name="output1" class="output">
107   </component>
108
109 </engine>
110 <engine name="phyengine11" class="phyengine">
111
112   <component name="dvbtlofdmmod1" class="dvbtlofdmmod">
113     <parameter name="debug" value="false">
114     <parameter name="ofdmmod" value="8192">
115     <parameter name="deltamode" value="4">
116     <parameter name="outpower" value="30">
117     <parameter name="dacsamplerate" value="12500000">
118     <port name="input1" class="input">
119     <port name="output1" class="output">
120   </component>
121
122 </engine>
123 <engine name="phyengine12" class="phyengine">
124
125   <component name="dvbtlininterpolator1" class="dvbtlininterpolator">
126     <parameter name="debug" value="false">
127     <parameter name="insamplerate" value="0">
128     <parameter name="outsamplerate" value="12500000">
129     <parameter name="responsefile" value="interp.txt">
130     <port name="input1" class="input">
131     <port name="output1" class="output">
132   </component>
133
134 </engine>
135 <engine name="phyengine13" class="phyengine">
136
137   <component name="dvbt1filter1" class="dvbt1filter">
138     <parameter name="debug" value="false">
139     <parameter name="samplerate" value="12500000">
140     <parameter name="attenuation" value="25">
141     <parameter name="stopband" value="4500000">
142     <port name="input1" class="input">
143     <port name="output1" class="output">
144   </component>
145
146 </engine>
147 <engine name="phyengine14" class="phyengine">
148
149   <component name="usrptx1" class="dvbt1usrptx">
150     <parameter name="frequency" value="666000000">

```

```

151     <parameter name="fixloffset" value="5000000">
152     <parameter name="rate" value="12500000">
153     <parameter name="streaming" value="false">
154     <parameter name="gain" value="10">
155     <parameter name="numbuffers" value="4">
156     <parameter name="args" value="addr=192.168.10.3">
157     <port name="input1" class="input">
158 </component>
159
160 </engine>
161
162 <link source="filerawreader1.output1" sink="dvbtlscrambler1.input1">
163 <link source="dvbtlscrambler1.output1" sink="dvbtlrsencoder1.input1">
164 <link source="dvbtlrsencoder1.output1" sink="dvbtlconvinterleaver1.input1">
165 <link source="dvbtlconvinterleaver1.output1" sink="dvbtlconvencoder1.input1">
166 <link source="dvbtlconvencoder1.output1" sink="dvbtlpuncturer1.input1">
167 <link source="dvbtlpuncturer1.output1" sink="dvbtlbitinterleaver1.input1">
168 <link source="dvbtlbitinterleaver1.output1" sink="dvbtlsymbolinterleaver1.input1">
169 <link source="dvbtlsymbolinterleaver1.output1" sink="dvbtlmapper1.input1">
170 <link source="dvbtlmapper1.output1" sink="dvbtlframer1.input1">
171 <link source="dvbtlframer1.output1" sink="dvbtlofdmmod1.input1">
172 <link source="dvbtlofdmmod1.output1" sink="dvbtlinterpolator1.input1">
173 <link source="dvbtlinterpolator1.output1" sink="dvbtlfilter1.input1">
174 <link source="dvbtlfilter1.output1" sink="usrptx1.input1">
175
176</softwareradio>

```

## 1.7 USRP setup

The USRP device must be connected via Gigabit Ethernet to the host PC where Iris and DVB-TX-IRIS are installed. The address of the used USRP device, once it is verified to be reachable via ping commands, can be written in the configuration file. The compatibility between the device firmware revision and the Ettus UHD drivers used to communicate with the USRP should be verified; differently, the device will refuse to work. Either a directional or an omni-directional antenna can be used to transmit the signal, after taking proper care of the USRP device gain value in the XML file.

## 1.8 Transmit

At this point the RF signal broadcasting can be initiated. The command line to give is

```
iris -f config.iris demo_typical_8K.xml
```

or different if the modified file name is different. If the transmission is proceeding correctly, the command line output should be devoid of USRP communication errors ("U" characters are printed in case of buffer under-run, typically happening when BB digital samples are not being fed sufficiently fast to the USRP) and the SDR modulator should print, periodically, a report of the actually processed bit rate: this value should match the theoretical payload bit rate.

```

[INFO]   usrptx1: Gain range: (0, 31.5, 0.5)

[INFO]   usrptx1: Setting TX Gain: 5 dB...
[INFO]   usrptx1: Actual TX Gain: 5 dB...
[INFO]   usrptx1: Using TX Antenna: TX/RX
[INFO]   usrptx1: Checking TX: LO: locked ...
[INFO]   System: Starting radio

Stack Repository :
Phy Repository  : /usr/local/lib/iris_modules/components/gpp/phy
SDF Repository  :
Controller Repository : /usr/local/lib/iris_modules/controllers
Log level : debug
Radio Config: demo_typical_8K.xml

      Iris Software Radio
      ~~~~~

      U Unload Radio      S Stop Radio
      R Reconfigure       Q Quit

(Radio running), Selection: [INFO]   dvbt1scrambler1: Current TS bitrate: 13.0523 Mbps
[INFO]   dvbt1scrambler1: Current TS bitrate: 22.3221 Mbps
[INFO]   dvbt1scrambler1: Current TS bitrate: 22.3525 Mbps
[INFO]   dvbt1scrambler1: Current TS bitrate: 22.4676 Mbps

```

Figure 1.2: Output of the Iris command line during correct operation.

## 1.9 Receive and validate

Any standard compliant DVB-T receiver that is in the range covered by the transmitting hardware should be able to pick-up and decode the signal. To this purpose, a full channel scan should be performed in the receiver's setup menu, and one or more TV channels from the WiSHFUL transmission network should be now present in the receiver channel list. The same thing can also be carried out using a DVB-T USB dongle: by this means, it can be possible also to analyze the quality and validate the received signal by a number of parameters such as the signal power, the residual bit error rate, the number of uncorrected packets, etc.

Power loading can also be tested: the "examples/dvbt" folder already contains a demonstration XML configuration file, `demo_typical_8K_PL_USRP.xml`, as well as a pre-computed power profile file, `logo_profile.txt`. Additionally, each user can recreate a logo-resembling power profile by running the MATLAB/Octave script `powerload_logo.m`, which is saved in the "scripts/dvbt/MATLAB" folder

## 1.10 Off-line validation

This step is not generally required, since if the TV signal is correctly received on a TV set, this should be more than sufficient. Anyway, during the compilation step, it is possible to carry out a validation of the DVB-TX-IRIS module C++ components against a MATLAB/Octave implementation of the same component. For every component, there is an associated M-script that can be used to generate random input and output test vectors for the specified component (the M-script and already generated test vectors are present in the "test" folder inside the main component folder). During the build process, the `ctest` command triggered during the invocation of "make test" executes an automated check of the components correct operation: the input test vector is loaded by the component and transformed into the respective output vector (as per the processing performed by the block); then, the MATLAB output test vector and the Iris output test vector are compared. The test passes or fails depending on the correspondence between these two test vectors.

## 1.11 Bibliography

1. G. Baruffa, L. Rugini, and P. Banelli, *Design and validation of a Software Defined Radio testbed for DVB-T transmission*, Radioengineering, vol. 23, no. 1, pp. 387–398, Apr. 2014.

2. DVB PROJECT. *Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for digital terrestrial television*, ETSI EN 300 744 V1.6.1 (2009-01). 2009. Available at: <http://www.dvb.org>.



## Chapter 2

# Namespace Index

### 2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">iris</a> . . . . .	19
<a href="#">iris::phy</a> . . . . .	19



## Chapter 3

# Hierarchical Index

### 3.1 Class Hierarchy

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

PhyComponent	
iris::phy::Dvbt1BitInterleaverComponent . . . . .	21
iris::phy::Dvbt1ConvEncoderComponent . . . . .	31
iris::phy::Dvbt1ConvInterleaverComponent . . . . .	37
iris::phy::Dvbt1FilterComponent . . . . .	47
iris::phy::Dvbt1FormatterComponent . . . . .	61
iris::phy::Dvbt1FramerComponent . . . . .	67
iris::phy::Dvbt1InterpolatorComponent . . . . .	84
iris::phy::Dvbt1MapperComponent . . . . .	96
iris::phy::Dvbt1NoiseGeneratorComponent . . . . .	105
iris::phy::Dvbt1OfdmModComponent . . . . .	112
iris::phy::Dvbt1PuncturerComponent . . . . .	128
iris::phy::Dvbt1RSEncoderComponent . . . . .	135
iris::phy::Dvbt1ScramblerComponent . . . . .	145
iris::phy::Dvbt1SymbolInterleaverComponent . . . . .	152
iris::phy::Dvbt1UsrcpTxComponent . . . . .	160



## Chapter 4

# Class Index

### 4.1 Class List

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

<a href="#">iris::phy::Dvbt1BitInterleaverComponent</a>	
A DVB-T1 bit interleaver component . . . . .	21
<a href="#">iris::phy::Dvbt1ConvEncoderComponent</a>	
A DVB-T1 convolutional encoder component . . . . .	31
<a href="#">iris::phy::Dvbt1ConvInterleaverComponent</a>	
A DVB-T1 convolutional interleaver component . . . . .	37
<a href="#">iris::phy::Dvbt1FilterComponent</a>	
A DVB-T1 filter component . . . . .	47
<a href="#">iris::phy::Dvbt1FormatterComponent</a>	
A DVB-T1 formatter component . . . . .	61
<a href="#">iris::phy::Dvbt1FramerComponent</a>	
A DVB-T1 framer component . . . . .	67
<a href="#">iris::phy::Dvbt1InterpolatorComponent</a>	
A DVB-T1 interpolator component . . . . .	84
<a href="#">iris::phy::Dvbt1MapperComponent</a>	
A DVB-T1 mapper component . . . . .	96
<a href="#">iris::phy::Dvbt1NoiseGeneratorComponent</a>	
A DVB-T1 noise generator . . . . .	105
<a href="#">iris::phy::Dvbt1OfdmModComponent</a>	
A DVB-T1 OFDM modulator component . . . . .	112
<a href="#">iris::phy::Dvbt1PuncturerComponent</a>	
A DVB-T1 puncturer component . . . . .	128
<a href="#">iris::phy::Dvbt1RSEncoderComponent</a>	
A DVB-T1 R-S Encoder component . . . . .	135
<a href="#">iris::phy::Dvbt1ScramblerComponent</a>	
A DVB-T energy dispersal component . . . . .	145
<a href="#">iris::phy::Dvbt1SymbolInterleaverComponent</a>	
A DVB-T1 symbol interleaver component . . . . .	152
<a href="#">iris::phy::Dvbt1UsrpTxComponent</a>	
The Dvbt1UsrpTx component . . . . .	160



## Chapter 5

# File Index

### 5.1 File List

Here is a list of all files with brief descriptions:

/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1BitInterleaver/Dvbt1BitInterleaverComponent.- cpp . . . . .	173
/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1BitInterleaver/Dvbt1BitInterleaverComponent.h . . . . .	174
/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1ConvEncoder/Dvbt1ConvEncoderComponent.- cpp . . . . .	176
/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1ConvEncoder/Dvbt1ConvEncoderComponent.- h . . . . .	177
/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1ConvInterleaver/Dvbt1ConvInterleaverComponent.- cpp . . . . .	179
/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1ConvInterleaver/Dvbt1ConvInterleaverComponent.- h . . . . .	180
/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1Filter/Dvbt1FilterComponent.cpp . . . . .	182
/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1Filter/Dvbt1FilterComponent.h . . . . .	184
/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1Formatter/Dvbt1FormatterComponent.cpp . . . . .	185
/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1Formatter/Dvbt1FormatterComponent.h . . . . .	187
/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1Framer/Dvbt1FramerComponent.cpp . . . . .	188
/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1Framer/Dvbt1FramerComponent.h . . . . .	190
/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1Interpolator/Dvbt1InterpolatorComponent.cpp . . . . .	192
/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1Interpolator/Dvbt1InterpolatorComponent.h . . . . .	193
/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1Mapper/Dvbt1MapperComponent.cpp . . . . .	195
/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1Mapper/Dvbt1MapperComponent.h . . . . .	197
/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1NoiseGenerator/Dvbt1NoiseGeneratorComponent.- cpp . . . . .	198
/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1NoiseGenerator/Dvbt1NoiseGeneratorComponent.- h . . . . .	199
/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1OfdmMod/Dvbt1OfdmModComponent.cpp . . . . .	201
/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1OfdmMod/Dvbt1OfdmModComponent.h . . . . .	202
/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1Puncturer/Dvbt1PuncturerComponent.cpp . . . . .	205
/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1Puncturer/Dvbt1PuncturerComponent.h . . . . .	206
/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1RSEncoder/Dvbt1RSEncoderComponent.cpp . . . . .	207
/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1RSEncoder/Dvbt1RSEncoderComponent.h . . . . .	209
/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1Scrambler/Dvbt1ScramblerComponent.cpp . . . . .	212
/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1Scrambler/Dvbt1ScramblerComponent.h . . . . .	213
/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1SymbolInterleaver/Dvbt1SymbolInterleaver- Component.cpp . . . . .	215
/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1SymbolInterleaver/Dvbt1SymbolInterleaver- Component.h . . . . .	216
/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1UsrpTx/Dvbt1UsrpTxComponent.cpp . . . . .	218

/home/iris/Projects/dti\_wishful/components/gpp/phy/Dvbt1UsrpTx/[Dvbt1UsrpTxComponent.h](#) . . . . . 219



## Chapter 6

# Namespace Documentation

### 6.1 iris Namespace Reference

#### Namespaces

- [phy](#)

### 6.2 iris::phy Namespace Reference

#### Classes

- class [Dvbt1BitInterleaverComponent](#)  
*A DVB-T1 bit interleaver component.*
- class [Dvbt1ConvEncoderComponent](#)  
*A DVB-T1 convolutional encoder component.*
- class [Dvbt1ConvInterleaverComponent](#)  
*A DVB-T1 convolutional interleaver component.*
- class [Dvbt1FilterComponent](#)  
*A DVB-T1 filter component.*
- class [Dvbt1FormatterComponent](#)  
*A DVB-T1 formatter component.*
- class [Dvbt1FramerComponent](#)  
*A DVB-T1 framer component.*
- class [Dvbt1InterpolatorComponent](#)  
*A DVB-T1 interpolator component.*
- class [Dvbt1MapperComponent](#)  
*A DVB-T1 mapper component.*
- class [Dvbt1NoiseGeneratorComponent](#)  
*A DVB-T1 noise generator.*
- class [Dvbt1OfdmModComponent](#)  
*A DVB-T1 OFDM modulator component.*
- class [Dvbt1PuncturerComponent](#)  
*A DVB-T1 puncturer component.*
- class [Dvbt1RSEncoderComponent](#)  
*A DVB-T1 R-S Encoder component.*
- class [Dvbt1ScramblerComponent](#)

- A DVB-T energy dispersal component.*  
 • class [Dvbt1SymbolInterleaverComponent](#)  
*A DVB-T1 symbol interleaver component.*
- class [Dvbt1UsrpTxComponent](#)  
*The Dvbt1UsrpTx component.*

## Functions

- [IRIS\\_COMPONENT\\_EXPORTS](#) (PhyComponent, [Dvbt1BitInterleaverComponent](#))
- [IRIS\\_COMPONENT\\_EXPORTS](#) (PhyComponent, [Dvbt1ConvEncoderComponent](#))
- [IRIS\\_COMPONENT\\_EXPORTS](#) (PhyComponent, [Dvbt1ConvInterleaverComponent](#))
- [IRIS\\_COMPONENT\\_EXPORTS](#) (PhyComponent, [Dvbt1FilterComponent](#))
- [IRIS\\_COMPONENT\\_EXPORTS](#) (PhyComponent, [Dvbt1FormatterComponent](#))
- [IRIS\\_COMPONENT\\_EXPORTS](#) (PhyComponent, [Dvbt1FramerComponent](#))
- [IRIS\\_COMPONENT\\_EXPORTS](#) (PhyComponent, [Dvbt1InterpolatorComponent](#))
- [IRIS\\_COMPONENT\\_EXPORTS](#) (PhyComponent, [Dvbt1MapperComponent](#))
- [IRIS\\_COMPONENT\\_EXPORTS](#) (PhyComponent, [Dvbt1NoiseGeneratorComponent](#))
- [IRIS\\_COMPONENT\\_EXPORTS](#) (PhyComponent, [Dvbt1OfdmModComponent](#))
- [IRIS\\_COMPONENT\\_EXPORTS](#) (PhyComponent, [Dvbt1PuncturerComponent](#))
- [IRIS\\_COMPONENT\\_EXPORTS](#) (PhyComponent, [Dvbt1REncoderComponent](#))
- [IRIS\\_COMPONENT\\_EXPORTS](#) (PhyComponent, [Dvbt1ScramblerComponent](#))
- [IRIS\\_COMPONENT\\_EXPORTS](#) (PhyComponent, [Dvbt1SymbolInterleaverComponent](#))
- [IRIS\\_COMPONENT\\_EXPORTS](#) (PhyComponent, [Dvbt1UsrpTxComponent](#))

### 6.2.1 Function Documentation

- 6.2.1.1 [iris::phy::IRIS\\_COMPONENT\\_EXPORTS \( PhyComponent , Dvbt1UsrpTxComponent \)](#)
- 6.2.1.2 [iris::phy::IRIS\\_COMPONENT\\_EXPORTS \( PhyComponent , Dvbt1InterpolatorComponent \)](#)
- 6.2.1.3 [iris::phy::IRIS\\_COMPONENT\\_EXPORTS \( PhyComponent , Dvbt1SymbolInterleaverComponent \)](#)
- 6.2.1.4 [iris::phy::IRIS\\_COMPONENT\\_EXPORTS \( PhyComponent , Dvbt1ConvEncoderComponent \)](#)
- 6.2.1.5 [iris::phy::IRIS\\_COMPONENT\\_EXPORTS \( PhyComponent , Dvbt1REncoderComponent \)](#)
- 6.2.1.6 [iris::phy::IRIS\\_COMPONENT\\_EXPORTS \( PhyComponent , Dvbt1FilterComponent \)](#)
- 6.2.1.7 [iris::phy::IRIS\\_COMPONENT\\_EXPORTS \( PhyComponent , Dvbt1MapperComponent \)](#)
- 6.2.1.8 [iris::phy::IRIS\\_COMPONENT\\_EXPORTS \( PhyComponent , Dvbt1NoiseGeneratorComponent \)](#)
- 6.2.1.9 [iris::phy::IRIS\\_COMPONENT\\_EXPORTS \( PhyComponent , Dvbt1PuncturerComponent \)](#)
- 6.2.1.10 [iris::phy::IRIS\\_COMPONENT\\_EXPORTS \( PhyComponent , Dvbt1ConvInterleaverComponent \)](#)
- 6.2.1.11 [iris::phy::IRIS\\_COMPONENT\\_EXPORTS \( PhyComponent , Dvbt1FramerComponent \)](#)
- 6.2.1.12 [iris::phy::IRIS\\_COMPONENT\\_EXPORTS \( PhyComponent , Dvbt1BitInterleaverComponent \)](#)
- 6.2.1.13 [iris::phy::IRIS\\_COMPONENT\\_EXPORTS \( PhyComponent , Dvbt1ScramblerComponent \)](#)
- 6.2.1.14 [iris::phy::IRIS\\_COMPONENT\\_EXPORTS \( PhyComponent , Dvbt1FormatterComponent \)](#)
- 6.2.1.15 [iris::phy::IRIS\\_COMPONENT\\_EXPORTS \( PhyComponent , Dvbt1OfdmModComponent \)](#)

## Chapter 7

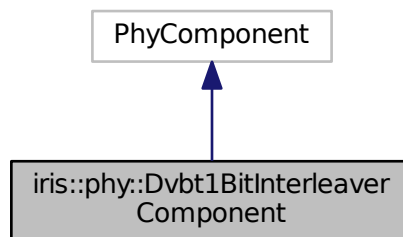
# Class Documentation

### 7.1 iris::phy::Dvbt1BitInterleaverComponent Class Reference

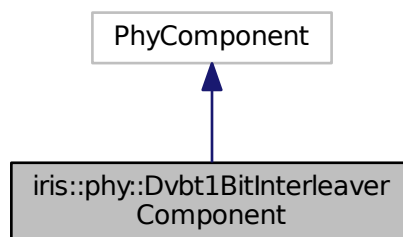
A DVB-T1 bit interleaver component.

```
#include <Dvbt1BitInterleaverComponent.h>
```

Inheritance diagram for iris::phy::Dvbt1BitInterleaverComponent:



Collaboration diagram for iris::phy::Dvbt1BitInterleaverComponent:



## Public Types

- typedef std::vector< uint8\_t > [ByteVec](#)  
*A vector of bytes.*
- typedef ByteVec::iterator [ByteVecIt](#)  
*An iterator for a vector of bytes.*

## Public Member Functions

- [Dvbt1BitInterleaverComponent](#) (std::string name)  
*Default constructor.*
- [~Dvbt1BitInterleaverComponent](#) ()  
*Default destructor.*
- virtual void [calculateOutputTypes](#) (std::map< std::string, int > &inputTypes, std::map< std::string, int > &outputTypes)  
*Calculate the output port types for the IRIS system.*
- virtual void [registerPorts](#) ()  
*Register the interleaver ports with the IRIS system.*
- virtual void [initialize](#) ()  
*Initialize the component.*
- virtual void [process](#) ()  
*Main processing method.*
- virtual void [parameterHasChanged](#) (std::string name)  
*Actions taken when the parameters change.*

## Private Member Functions

- void [setup](#) ()  
*Set up all offsets, clean registers.*
- void [destroy](#) ()  
*Destroy the component.*

## Static Private Member Functions

- template<typename T , size\_t N>  
static T \* [begin](#) (T(&arr)[N])  
*Useful templates.*
- template<typename T , size\_t N>  
static T \* [end](#) (T(&arr)[N])

## Private Attributes

- bool [debug\\_x](#)  
*Debug flag (default = false)*
- int [qamMapping\\_x](#)  
*QAM constellation mapping (default = 16)*
- int [hyerarchyMode\\_x](#)  
*Hyerarchical mode (default = 0)*
- double [timeStamp\\_x](#)  
*Timestamp of current frame.*

- double [sampleRate\\_](#)  
*Sample rate of current frame.*
- int [intOffset\\_](#) [2]  
*Interleaving offsets (HP & LP)*
- int [intLength\\_](#) [2]  
*Interleaving registers length (HP & LP)*
- uint8\_t \* [intRegister\\_](#) [2]  
*Interleaving registers (HP & LP)*
- int [nu\\_](#)  
*Bits per modulated symbol.*

### Static Private Attributes

- static int [address\\_v2](#) [252]  
*The interleaving addresses for QPSK.*
- static int [address\\_v4](#) [504]  
*The interleaving addresses for 16-QAM.*
- static int [address\\_v6](#) [756]  
*The interleaving addresses for 64-QAM.*

#### 7.1.1 Detailed Description

A DVB-T1 bit interleaver component.

[Dvbt1BitInterleaverComponent](#) is the sixth block composing the DVB-T transmission chain. Its purpose, together with the symbol interleaver, is that of reordering the channel encoded bits in order to convert the possible error bursts arising from the communication on the physical channel (due to impulsive noise, multipath, fading) into well-separated single-error events. This way, the channel decoders at the RX side (Viterbi and Reed-Solomon decoder) are able to perform at their best theoretical limit in white Gaussian noise (WGN) conditions.

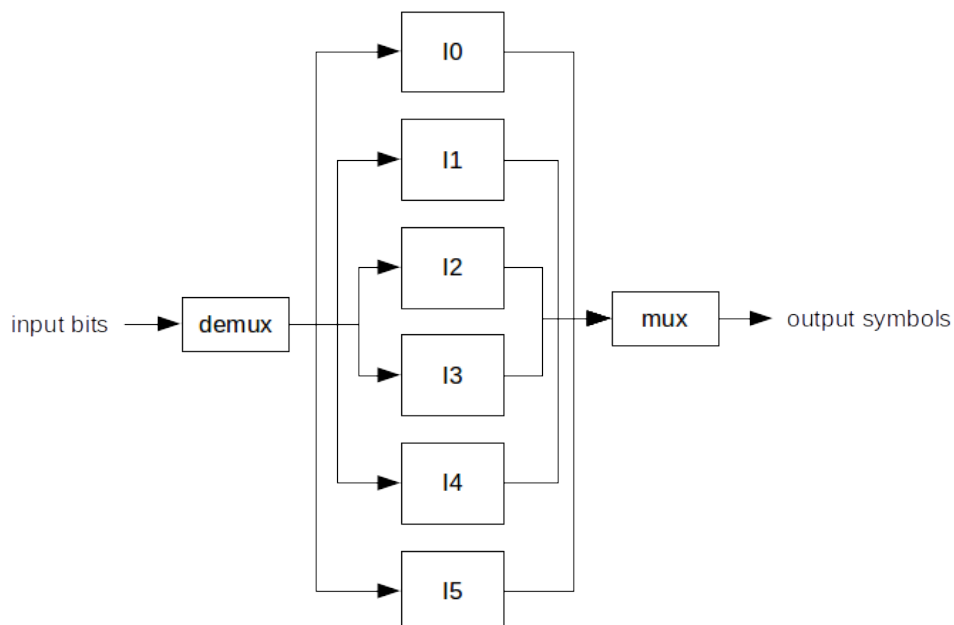


Figure 7.1: DVB-T bit interleaver.

With reference to the figure above, the input demultiplexer routes the incoming bits emitted by the puncturer towards one of the 6 bit interleaving RAMs. Every RAM has a capacity of 126 bits. When all the RAMs are filled, the stored

bits are read out according to a particular cyclic address shift and composed into  $v$ -bit symbols, where  $v$  is the number of bits of the particular M-QAM mapping adopted. Please note that only  $v$  RAM interleavers are adopted, thus the figure above refers to the 64-QAM case. This block accepts in input elements in `uint8_t` (bits) and generates in output  $v$ -bit symbols (`uint8_t`).

There are three parameters that can be changed in the XML configuration file:

- *debug*: by default set to "false", is used to print some small debugging information for the interested developer.
- *qammapping*: by default set to "16", this is used to select one of the three possible QAM mappings. The admitted values are "4", "16", "64".
- *hierarchymode*: by default set to "0", which means "not hierarchical". Hierarchical modes are used to transmit two different transport streams, one with a high priority (HP) information and another one with a low priority (LP) information. The admitted values are "0", "1", "2", "4". NOTE: hierarchical modes are not implemented in the current release of this modulator.

## References

- ETSI Standard: *EN 300 744 V1.5.1, Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for digital terrestrial television*, available at [ETSI Publications Download Area](#)

Definition at line 88 of file `Dvbt1BitInterleaverComponent.h`.

## 7.1.2 Member Typedef Documentation

### 7.1.2.1 `typedef std::vector<uint8_t> iris::phy::Dvbt1BitInterleaverComponent::ByteVec`

A vector of bytes.

Definition at line 94 of file `Dvbt1BitInterleaverComponent.h`.

### 7.1.2.2 `typedef ByteVec::iterator iris::phy::Dvbt1BitInterleaverComponent::ByteVecIt`

An iterator for a vector of bytes.

Definition at line 97 of file `Dvbt1BitInterleaverComponent.h`.

## 7.1.3 Constructor & Destructor Documentation

### 7.1.3.1 `iris::phy::Dvbt1BitInterleaverComponent::Dvbt1BitInterleaverComponent ( std::string name )`

Default constructor.

Registers the block parameters and initializes some variables

Definition at line 57 of file `Dvbt1BitInterleaverComponent.cpp`.

References `begin()`, `debug_x`, `end()`, `hierarchyMode_x`, `intRegister_x`, and `qamMapping_x`.

```

58 : PhyComponent(name,                                // component name
59     "dvbt1bitinterleaver",                          // component type
60     "A DVB-T1 bit interleaver component",           // description
61     "Giuseppe Baruffa",                             // author
62     "0.1")                                           // version
63     ,sampleRate_(0)
64     ,timeStamp_(0)
65 {
66     registerParameter(
67         "debug", "Whether to output debug data",
68         "false", true, debug_x);
69
70     int qamarr[] = {4,16,64};

```

```

71  registerParameter(
72      "qamapping", "QAM constellation mapping",
73      "16", true, qamMapping_x, list<int>(begin(qamarr),end(qamarr)));
74
75  int harr[] = {0,1,2,4};
76  registerParameter(
77      "hyerarchymode", "Hyerarchical mode (0 = NH)",
78      "0", true, hyerarchyMode_x, list<int>(begin(harr),end(harr)));
79
80  intRegister_[0] = NULL;
81  intRegister_[1] = NULL;
82 }

```

### 7.1.3.2 iris::phy::Dvbt1BitInterleaverComponent::~~Dvbt1BitInterleaverComponent ( )

Default destructor.

Just calls [destroy\(\)](#).

Definition at line 87 of file Dvbt1BitInterleaverComponent.cpp.

References [destroy\(\)](#).

```

88 {
89     destroy();
90 }

```

## 7.1.4 Member Function Documentation

### 7.1.4.1 template<typename T , size\_t N> static T\* iris::phy::Dvbt1BitInterleaverComponent::begin ( T(&) arr[N] ) [inline], [static], [private]

Useful templates.

Definition at line 133 of file Dvbt1BitInterleaverComponent.h.

Referenced by [Dvbt1BitInterleaverComponent\(\)](#).

```

133 { return &arr[0]; }

```

### 7.1.4.2 void iris::phy::Dvbt1BitInterleaverComponent::calculateOutputTypes ( std::map< std::string, int > &inputTypes, std::map< std::string, int > &outputTypes ) [virtual]

Calculate the output port types for the IRIS system.

The single output port must provide bytes.

Definition at line 106 of file Dvbt1BitInterleaverComponent.cpp.

```

109 {
110     outputTypes["output1"] = TypeInfo< uint8_t >::identifier;
111 }

```

### 7.1.4.3 void iris::phy::Dvbt1BitInterleaverComponent::destroy ( ) [private]

Destroy the component.

Definition at line 320 of file Dvbt1BitInterleaverComponent.cpp.

References [intRegister\\_](#).

Referenced by [parameterHasChanged\(\)](#), and [~Dvbt1BitInterleaverComponent\(\)](#).

```

321 {
322     // clean
323     delete[] intRegister_[0];
324     delete[] intRegister_[1];
325 }

```

**7.1.4.4** `template<typename T, size_t N> static T* iris::phy::Dvbt1BitInterleaverComponent::end ( T(&) arr[N] )`  
`[inline], [static], [private]`

Definition at line 135 of file Dvbt1BitInterleaverComponent.h.

Referenced by Dvbt1BitInterleaverComponent().

```

135 { return &arr[0]+N; }

```

**7.1.4.5** `void iris::phy::Dvbt1BitInterleaverComponent::initialize ( )` `[virtual]`

Initialize the component.

Just calls `setup()`.

Definition at line 116 of file Dvbt1BitInterleaverComponent.cpp.

References `setup()`.

```

117 {
118     setup();
119 }

```

**7.1.4.6** `void iris::phy::Dvbt1BitInterleaverComponent::parameterHasChanged ( std::string name )` `[virtual]`

Actions taken when the parameters change.

This block has two significant parameters

Definition at line 290 of file Dvbt1BitInterleaverComponent.cpp.

References `destroy()`, and `setup()`.

```

291 {
292     if(name == "qammapping" || name == "hierarchymode")
293     {
294         destroy();
295         setup();
296     }
297 }

```

**7.1.4.7** `void iris::phy::Dvbt1BitInterleaverComponent::process ( )` `[virtual]`

Main processing method.

Definition at line 204 of file Dvbt1BitInterleaverComponent.cpp.

References `address_v2`, `address_v4`, `address_v6`, `debug_x`, `hierarchyMode_x`, `intLength_`, `intOffset_`, `intRegister_`, `nu_`, and `qamMapping_x`.

```

205 {
206     // request input
207     DataSet< uint8_t > *in1 = NULL;
208     getInputDataSet("input1", in1);
209
210     // calculate sizes
211     int in1size = in1 ? (int) in1->data.size() : 0;

```



```

212 int outsize = intLength_[0] * ((inlsize + intOffset_[0]) /
intLength_[0]) / nu_;
213
214 // request output
215 DataSet< uint8_t >* out = NULL;
216 getOutputDataSet("output1", out, outsize);
217
218 // print debug info
219 if(debug_x)
220     LOG(LINFO) << "inl/out: " << inlsize << "/" << outsize;
221
222 // bit by bit
223 for(ByteVecIt inlit = inl->data.begin(), outit = out->data.begin();
224     inlit < inl->data.end(); inlit++)
225 {
226     // copy to register
227     intRegister_[0][intOffset_[0]++] = *inlit;
228
229     // trigger interleaving
230     if(intOffset_[0] == intLength_[0])
231     {
232         // reset offset
233         intOffset_[0] = 0;
234
235         // do the copy
236         if(hierarchyMode_x == 0)
237         {
238             switch(qamMapping_x)
239             {
240                 // read back according to the QAM mode and compose the output symbol
241                 case 4:
242                     for(int b = 0; b < intLength_[0]; b += 2)
243                     {
244                         *outit++ =
245                             (intRegister_[0][address_v2[b + 0]] << 1) |
246                             (intRegister_[0][address_v2[b + 1]] << 0);
247                     }
248                     break;
249                 case 16:
250                     for(int b = 0; b < intLength_[0]; b += 4)
251                     {
252                         *outit++ =
253                             (intRegister_[0][address_v4[b + 0]] << 3) |
254                             (intRegister_[0][address_v4[b + 1]] << 2) |
255                             (intRegister_[0][address_v4[b + 2]] << 1) |
256                             (intRegister_[0][address_v4[b + 3]] << 0);
257                     }
258                     break;
259                 case 64:
260                     for(int b = 0; b < intLength_[0]; b += 6)
261                     {
262                         *outit++ =
263                             (intRegister_[0][address_v6[b + 0]] << 5) |
264                             (intRegister_[0][address_v6[b + 1]] << 4) |
265                             (intRegister_[0][address_v6[b + 2]] << 3) |
266                             (intRegister_[0][address_v6[b + 3]] << 2) |
267                             (intRegister_[0][address_v6[b + 4]] << 1) |
268                             (intRegister_[0][address_v6[b + 5]] << 0);
269                     }
270                     break;
271                 default:
272                     LOG(LERROR) << "Unsupported QAM mapping";
273             }
274         }
275     }
276 }
277
278 // copy the timestamp and sample rate for the DataSets
279 out->timeStamp = inl->timeStamp;
280 out->sampleRate = inl->sampleRate;
281
282 // release input and output
283 releaseInputDataSet("input1", inl);
284 releaseOutputDataSet("output1", out);
285 }

```

#### 7.1.4.8 void iris::phy::Dvbt1BitInterleaverComponent::registerPorts ( ) [virtual]

Register the interleaver ports with the IRIS system.

This component has two inputs that accept bits (one bit per byte) and one output that provides symbols (in one byte).

Definition at line 96 of file Dvbt1BitInterleaverComponent.cpp.

```

97 {
98     registerInputPort("input1", TypeInfo< uint8_t >::identifier);
99     registerInputPort("input2", TypeInfo< uint8_t >::identifier);
100    registerOutputPort("output1", TypeInfo< uint8_t >::identifier);
101 }
```

#### 7.1.4.9 void iris::phy::Dvbt1BitInterleaverComponent::setup ( ) [private]

Set up all offsets, clean registers.

Definition at line 300 of file Dvbt1BitInterleaverComponent.cpp.

References hierarchyMode\_x, intLength\_, intOffset\_, intRegister\_, nu\_, and qamMapping\_x.

Referenced by initialize(), and parameterHasChanged().

```

301 {
302     // clean
303     intOffset_[0] = 0;
304     intOffset_[1] = 0;
305
306     // modulation order
307     nu_ = qamMapping_x == 4 ? 2 : (qamMapping_x == 16 ? 4 : 6);
308
309     // lengths
310     intLength_[0] = (hierarchyMode_x == 0 ? (126 * nu_) : 126 * 2);
311     intLength_[1] = (hierarchyMode_x == 0 ? 1 /* to avoid divide by 0 error */
312                     : (126 * (nu_ - 2)));
313
314     // alloc
315     intRegister_[0] = new uint8_t [intLength_[0]];
316     intRegister_[1] = new uint8_t [intLength_[1]];
317 }
```

### 7.1.5 Member Data Documentation

#### 7.1.5.1 int iris::phy::Dvbt1BitInterleaverComponent::address\_v2 [static], [private]

Initial value:

```

= {
    0, 127,  2, 129,  4, 131,  6, 133,  8, 135, 10, 137, 12, 139, 14, 141, 16, 143, 18, 145, 20,
    147,
    22, 149, 24, 151, 26, 153, 28, 155, 30, 157, 32, 159, 34, 161, 36, 163, 38, 165, 40, 167, 42,
    169,
    44, 171, 46, 173, 48, 175, 50, 177, 52, 179, 54, 181, 56, 183, 58, 185, 60, 187, 62, 189, 64,
    191,
    66, 193, 68, 195, 70, 197, 72, 199, 74, 201, 76, 203, 78, 205, 80, 207, 82, 209, 84, 211, 86,
    213,
    88, 215, 90, 217, 92, 219, 94, 221, 96, 223, 98, 225, 100, 227, 102, 229, 104, 231, 106, 233, 108,
    235,
    110, 237, 112, 239, 114, 241, 116, 243, 118, 245, 120, 247, 122, 249, 124, 251, 126,  1, 128,  3, 130,
    5,
    132,  7, 134,  9, 136, 11, 138, 13, 140, 15, 142, 17, 144, 19, 146, 21, 148, 23, 150, 25, 152,
    27,
    154, 29, 156, 31, 158, 33, 160, 35, 162, 37, 164, 39, 166, 41, 168, 43, 170, 45, 172, 47, 174,
    49,
    176, 51, 178, 53, 180, 55, 182, 57, 184, 59, 186, 61, 188, 63, 190, 65, 192, 67, 194, 69, 196,
    71,
    198, 73, 200, 75, 202, 77, 204, 79, 206, 81, 208, 83, 210, 85, 212, 87, 214, 89, 216, 91, 218,
    93,
    220, 95, 222, 97, 224, 99, 226, 101, 228, 103, 230, 105, 232, 107, 234, 109, 236, 111, 238, 113, 240,
    115,
    242, 117, 244, 119, 246, 121, 248, 123, 250, 125
}
```

The interleaving addresses for QPSK.

Definition at line 127 of file Dvbt1BitInterleaverComponent.h.

Referenced by process().

### 7.1.5.2 int iris::phy::Dvbt1BitInterleaverComponent::address\_v4 [static], [private]

Initial value:

```
= {
    0, 254, 421, 171,    4, 258, 425, 175,    8, 262, 429, 179,    12, 266, 433, 183,    16, 270, 437, 187,    20,
    274,
    441, 191,    24, 278, 445, 195,    28, 282, 449, 199,    32, 286, 453, 203,    36, 290, 457, 207,    40, 294, 461,
    211,
    44, 298, 465, 215,    48, 302, 469, 219,    52, 306, 473, 223,    56, 310, 477, 227,    60, 314, 481, 231,    64,
    318,
    485, 235,    68, 322, 489, 239,    72, 326, 493, 243,    76, 330, 497, 247,    80, 334, 501, 251,    84, 338,    1
    255,
    88, 342,    5, 259,    92, 346,    9, 263,    96, 350,    13, 267,    100, 354,    17, 271,    104, 358,    21, 275,    108,
    362,
    25, 279,    112, 366,    29, 283,    116, 370,    33, 287,    120, 374,    37, 291,    124, 378,    41, 295,    128, 382,    45,
    299,
    132, 386,    49, 303,    136, 390,    53, 307,    140, 394,    57, 311,    144, 398,    61, 315,    148, 402,    65, 319,    152,
    406,
    69, 323,    156, 410,    73, 327,    160, 414,    77, 331,    164, 418,    81, 335,    168, 422,    85, 339,    172, 426,    89,
    343,
    176, 430,    93, 347,    180, 434,    97, 351,    184, 438,    101, 355,    188, 442,    105, 359,    192, 446,    109, 363,    196,
    450,
    113, 367,    200, 454,    117, 371,    204, 458,    121, 375,    208, 462,    125, 379,    212, 466,    129, 383,    216, 470,    133,
    387,
    220, 474,    137, 391,    224, 478,    141, 395,    228, 482,    145, 399,    232, 486,    149, 403,    236, 490,    153, 407,    240,
    494,
    157, 411,    244, 498,    161, 415,    248, 502,    165, 419,    252,    2, 169,    423, 256,    6, 173,    427, 260,    10, 177,
    431,
    264,    14,    181, 435,    268,    18,    185, 439,    272,    22,    189, 443,    276,    26,    193, 447,    280,    30,    197, 451,    284,
    34,
    201, 455,    288,    38,    205, 459,    292,    42,    209, 463,    296,    46,    213, 467,    300,    50,    217, 471,    304,    54,    221,
    475,
    308,    58,    225, 479,    312,    62,    229, 483,    316,    66,    233, 487,    320,    70,    237, 491,    324,    74,    241, 495,    328,
    78,
    245, 499,    332,    82,    249, 503,    336,    86,    253,    3,    340,    90,    257,    7,    344,    94,    261,    11,    348,    98,    265,
    15,
    352,    102,    269,    19,    356,    106,    273,    23,    360,    110,    277,    27,    364,    114,    281,    31,    368,    118,    285,    35,    372,
    122,
    289,    39,    376,    126,    293,    43,    380,    130,    297,    47,    384,    134,    301,    51,    388,    138,    305,    55,    392,    142,    309,
    59,
    396,    146,    313,    63,    400,    150,    317,    67,    404,    154,    321,    71,    408,    158,    325,    75,    412,    162,    329,    79,    416,
    166,
    333,    83,    420,    170,    337,    87,    424,    174,    341,    91,    428,    178,    345,    95,    432,    182,    349,    99,    436,    186,    353,
    103,
    440,    190,    357,    107,    444,    194,    361,    111,    448,    198,    365,    115,    452,    202,    369,    119,    456,    206,    373,    123,    460,
    210,
    377,    127,    464,    214,    381,    131,    468,    218,    385,    135,    472,    222,    389,    139,    476,    226,    393,    143,    480,    230,    397,
    147,
    484,    234,    401,    151,    488,    238,    405,    155,    492,    242,    409,    159,    496,    246,    413,    163,    500,    250,    417,    167
}
```

The interleaving addresses for 16-QAM.

Definition at line 128 of file Dvbt1BitInterleaverComponent.h.

Referenced by process().

### 7.1.5.3 int iris::phy::Dvbt1BitInterleaverComponent::address\_v6 [static], [private]

The interleaving addresses for 64-QAM.

Definition at line 129 of file Dvbt1BitInterleaverComponent.h.

Referenced by process().

### 7.1.5.4 bool iris::phy::Dvbt1BitInterleaverComponent::debug\_x [private]

Debug flag (default = false)

Definition at line 111 of file Dvbt1BitInterleaverComponent.h.

Referenced by Dvbt1BitInterleaverComponent(), and process().

#### 7.1.5.5 `int iris::phy::Dvbt1BitInterleaverComponent::hierarchyMode_x` [private]

Hierarchical mode (default = 0)

Definition at line 113 of file `Dvbt1BitInterleaverComponent.h`.

Referenced by `Dvbt1BitInterleaverComponent()`, `process()`, and `setup()`.

#### 7.1.5.6 `int iris::phy::Dvbt1BitInterleaverComponent::intLength_2` [private]

Interleaving registers length (HP & LP)

Definition at line 122 of file `Dvbt1BitInterleaverComponent.h`.

Referenced by `process()`, and `setup()`.

#### 7.1.5.7 `int iris::phy::Dvbt1BitInterleaverComponent::intOffset_2` [private]

Interleaving offsets (HP & LP)

Definition at line 121 of file `Dvbt1BitInterleaverComponent.h`.

Referenced by `process()`, and `setup()`.

#### 7.1.5.8 `uint8_t* iris::phy::Dvbt1BitInterleaverComponent::intRegister_2` [private]

Interleaving registers (HP & LP)

Definition at line 123 of file `Dvbt1BitInterleaverComponent.h`.

Referenced by `destroy()`, `Dvbt1BitInterleaverComponent()`, `process()`, and `setup()`.

#### 7.1.5.9 `int iris::phy::Dvbt1BitInterleaverComponent::nu_` [private]

Bits per modulated symbol.

Definition at line 124 of file `Dvbt1BitInterleaverComponent.h`.

Referenced by `process()`, and `setup()`.

#### 7.1.5.10 `int iris::phy::Dvbt1BitInterleaverComponent::qamMapping_x` [private]

QAM constellation mapping (default = 16)

Definition at line 112 of file `Dvbt1BitInterleaverComponent.h`.

Referenced by `Dvbt1BitInterleaverComponent()`, `process()`, and `setup()`.

#### 7.1.5.11 `double iris::phy::Dvbt1BitInterleaverComponent::sampleRate_` [private]

Sample rate of current frame.

Definition at line 119 of file `Dvbt1BitInterleaverComponent.h`.

#### 7.1.5.12 `double iris::phy::Dvbt1BitInterleaverComponent::timeStamp_` [private]

Timestamp of current frame.

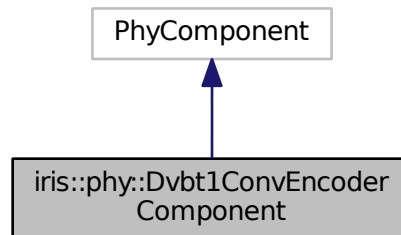
Definition at line 118 of file `Dvbt1BitInterleaverComponent.h`.

## 7.2 iris::phy::Dvbt1ConvEncoderComponent Class Reference

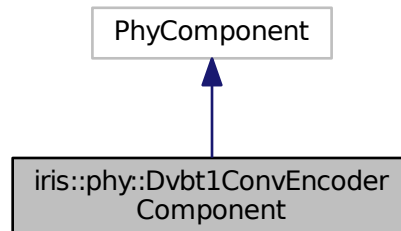
A DVB-T1 convolutional encoder component.

```
#include <Dvbt1ConvEncoderComponent.h>
```

Inheritance diagram for iris::phy::Dvbt1ConvEncoderComponent:



Collaboration diagram for iris::phy::Dvbt1ConvEncoderComponent:



### Public Types

- `typedef std::vector< uint8_t > ByteVec`  
*A vector of bytes.*
- `typedef ByteVec::iterator ByteVecIt`  
*An iterator for a vector of bytes.*

### Public Member Functions

- `Dvbt1ConvEncoderComponent` (std::string name)  
*Default constructor.*
- `~Dvbt1ConvEncoderComponent` ()  
*Default destructor.*

- virtual void [calculateOutputTypes](#) (std::map< std::string, int > &inputTypes, std::map< std::string, int > &outputTypes)  
*Calculate the output port types for the IRIS system.*
- virtual void [registerPorts](#) ()  
*Register the scrambler ports with the IRIS system.*
- virtual void [initialize](#) ()  
*Initialize the component.*
- virtual void [process](#) ()  
*Main processing method.*
- virtual void [parameterHasChanged](#) (std::string name)  
*Actions taken when the parameters change.*

### Private Member Functions

- void [setup](#) ()  
*Clean variables.*
- void [destroy](#) ()  
*Destroy the component.*

### Static Private Member Functions

- template<typename T, size\_t N>  
static T \* [begin](#) (T(&arr)[N])  
*Useful templates.*
- template<typename T, size\_t N>  
static T \* [end](#) (T(&arr)[N])

### Private Attributes

- bool [debug\\_x](#)  
*Debug flag (default = false)*
- double [timeStamp\\_](#)  
*Timestamp of current frame.*
- double [sampleRate\\_](#)  
*Sample rate of current frame.*
- int [status\\_](#)  
*Register with the delayed inputs (state)*

### Static Private Attributes

- static unsigned char [parity\\_](#) [256]  
*LUT containing the parity bits.*

## 7.2.1 Detailed Description

A DVB-T1 convolutional encoder component.

[Dvbt1ConvEncoderComponent](#) is the fourth block composing the DVB-T transmission chain. This block is a binary convolutional encoder of rate  $k/n = 1/2$  with a constraint length of  $L = 6$ . The output bits are generated by proper connections between the shift register cells and the XOR-adders. The connection configuration is represented in octal form by the generator polynomials  $G_1 = 0171$  and  $G_2 = 0133$ .

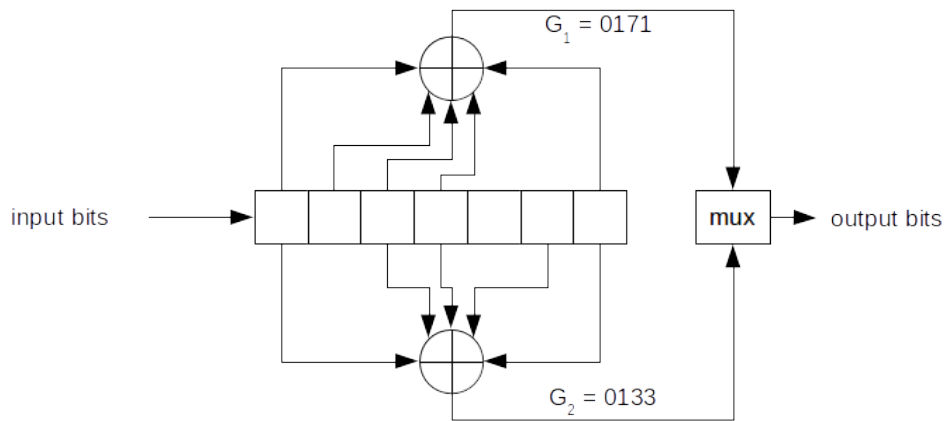


Figure 7.2: DVB-T convolutional encoder.

This block accepts in input elements in `uint8_t` (octets of bits) and generates in output single bits (always formatted as `uint8_t`). There is only one parameter that can be changed in the XML configuration file:

- *debug*: by default set to "false", is used to print some small debugging information for the interested developer.

## References

- ETSI Standard: *EN 300 744 V1.5.1, Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for digital terrestrial television*, available at [ETSI Publications Download Area](#)
- S. Li, D. J. Costello, *Error Control Coding, Second Edition*, Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 2004

Definition at line 72 of file `Dvbt1ConvEncoderComponent.h`.

## 7.2.2 Member Typedef Documentation

### 7.2.2.1 `typedef std::vector<uint8_t> iris::phy::Dvbt1ConvEncoderComponent::ByteVec`

A vector of bytes.

Definition at line 78 of file `Dvbt1ConvEncoderComponent.h`.

### 7.2.2.2 `typedef ByteVec::iterator iris::phy::Dvbt1ConvEncoderComponent::ByteVecIt`

An iterator for a vector of bytes.

Definition at line 81 of file `Dvbt1ConvEncoderComponent.h`.

## 7.2.3 Constructor & Destructor Documentation

### 7.2.3.1 `iris::phy::Dvbt1ConvEncoderComponent::Dvbt1ConvEncoderComponent ( std::string name )`

Default constructor.

Registers the block parameters and initializes some variables

Definition at line 57 of file `Dvbt1ConvEncoderComponent.cpp`.

References `debug_x`.

```

58 : PhyComponent (name,                                // component name
59                 "dvbt1convencoder",                  // component type
60                 "A DVB-T1 convolutional encoder component", // description
61                 "Giuseppe Baruffa",                  // author
62                 "0.1")                                // version
63 ,sampleRate_(0)
64 ,timeStamp_(0)
65 ,status_(0)
66 {
67     registerParameter(
68         "debug", "Whether to output debug data",
69         "false", true, debug_x);
70 }

```

### 7.2.3.2 iris::phy::Dvbt1ConvEncoderComponent::~~Dvbt1ConvEncoderComponent ( )

Default destructor.

Just calls [destroy\(\)](#).

Definition at line 75 of file Dvbt1ConvEncoderComponent.cpp.

References [destroy\(\)](#).

```

76 {
77     destroy\(\);
78 }

```

## 7.2.4 Member Function Documentation

### 7.2.4.1 template<typename T, size\_t N> static T\* iris::phy::Dvbt1ConvEncoderComponent::begin ( T(&) arr[N] ) [inline], [static], [private]

Useful templates.

Definition at line 109 of file Dvbt1ConvEncoderComponent.h.

```

109 { return &arr[0]; }

```

### 7.2.4.2 void iris::phy::Dvbt1ConvEncoderComponent::calculateOutputTypes ( std::map< std::string, int > & inputTypes, std::map< std::string, int > & outputTypes ) [virtual]

Calculate the output port types for the IRIS system.

The single output port must provide bytes.

Definition at line 93 of file Dvbt1ConvEncoderComponent.cpp.

```

96 {
97     outputTypes["output1"] = TypeInfo< uint8_t >::identifier;
98 }

```

### 7.2.4.3 void iris::phy::Dvbt1ConvEncoderComponent::destroy ( ) [private]

Destroy the component.

Definition at line 210 of file Dvbt1ConvEncoderComponent.cpp.

Referenced by [parameterHasChanged\(\)](#), and [~Dvbt1ConvEncoderComponent\(\)](#).

```

211 {
212 }

```



**7.2.4.4** `template<typename T, size_t N> static T* iris::phy::Dvbt1ConvEncoderComponent::end ( T(&) arr[N] )`  
`[inline], [static], [private]`

Definition at line 111 of file Dvbt1ConvEncoderComponent.h.

```
111 { return &arr[0]+N; }
```

**7.2.4.5** `void iris::phy::Dvbt1ConvEncoderComponent::initialize ( )` `[virtual]`

Initialize the component.

Just calls `setup()`.

Definition at line 103 of file Dvbt1ConvEncoderComponent.cpp.

References `setup()`.

```
104 {
105     setup();
106 }
```

**7.2.4.6** `void iris::phy::Dvbt1ConvEncoderComponent::parameterHasChanged ( std::string name )` `[virtual]`

Actions taken when the parameters change.

This block has no significant parameters

Definition at line 194 of file Dvbt1ConvEncoderComponent.cpp.

References `destroy()`, and `setup()`.

```
195 {
196     if(name == "???" )
197     {
198         destroy();
199         setup();
200     }
201 }
```

**7.2.4.7** `void iris::phy::Dvbt1ConvEncoderComponent::process ( )` `[virtual]`

Main processing method.

Definition at line 152 of file Dvbt1ConvEncoderComponent.cpp.

References `debug_x`, `g1`, `g2`, `parity_`, and `status_`.

```
153 {
154     // request input
155     DataSet< uint8_t >* in = NULL;
156     getInputDataSet("input1", in);
157
158     // calculate sizes
159     int insize = in ? (int) in->data.size() : 0;
160     int outsize = 2 * 8 * insize;
161
162     // request output - double size
163     DataSet< uint8_t >* out = NULL;
164     getOutputDataSet("output1", out, outsize);
165
166     // print debug info
167     if(debug_x)
168         LOG(LINFO) << "in/out: " << insize << "/" << outsize;
169
170     // iterate over all input bytes
171     for(ByteVecIt init = in->data.begin(), outit = out->data.begin(); init < in->data.end(); init++)
172     {
```

```

173     // iterate over all the bits of the byte
174     for(int j = 7; j >= 0; j--)
175     {
176         status_ = (status_ << 1) | ((*init >> j) & 0x01); // new status
177         *outit++ = parity_[status_ & g1]; // first parity bit
178         *outit++ = parity_[status_ & g2]; // second parity bit
179     }
180 }
181
182 //Copy the timestamp and sample rate for the DataSets
183 out->timeStamp = in->timeStamp;
184 out->sampleRate = in->sampleRate;
185
186 // release input and output
187 releaseInputDataSet("input1", in);
188 releaseOutputDataSet("output1", out);
189 }

```

#### 7.2.4.8 void iris::phy::Dvbt1ConvEncoderComponent::registerPorts ( ) [virtual]

Register the scrambler ports with the IRIS system.

This component has one input that accepts bytes and one output that provides convolutional encoded bits (one bit per byte).

Definition at line 84 of file Dvbt1ConvEncoderComponent.cpp.

```

85 {
86     registerInputPort("input1", TypeInfo< uint8_t >::identifier);
87     registerOutputPort("output1", TypeInfo< uint8_t >::identifier);
88 }

```

#### 7.2.4.9 void iris::phy::Dvbt1ConvEncoderComponent::setup ( ) [private]

Clean variables.

Definition at line 204 of file Dvbt1ConvEncoderComponent.cpp.

References status\_.

Referenced by initialize(), and parameterHasChanged().

```

205 {
206     status_ = 0;
207 }

```

### 7.2.5 Member Data Documentation

#### 7.2.5.1 bool iris::phy::Dvbt1ConvEncoderComponent::debug\_x [private]

Debug flag (default = false)

Definition at line 95 of file Dvbt1ConvEncoderComponent.h.

Referenced by Dvbt1ConvEncoderComponent(), and process().

#### 7.2.5.2 unsigned char iris::phy::Dvbt1ConvEncoderComponent::parity\_ [static], [private]

LUT containing the parity bits.

This look-up tables contains pairs of convolutional encoder parity bit outputs for all the possible configurations of states (64) and inputs (2)

Definition at line 105 of file Dvbt1ConvEncoderComponent.h.

Referenced by process().

#### 7.2.5.3 double iris::phy::Dvbt1ConvEncoderComponent::sampleRate\_ [private]

Sample rate of current frame.

Definition at line 101 of file Dvbt1ConvEncoderComponent.h.

#### 7.2.5.4 int iris::phy::Dvbt1ConvEncoderComponent::status\_ [private]

Register with the delayed inputs (state)

Definition at line 103 of file Dvbt1ConvEncoderComponent.h.

Referenced by process(), and setup().

#### 7.2.5.5 double iris::phy::Dvbt1ConvEncoderComponent::timeStamp\_ [private]

Timestamp of current frame.

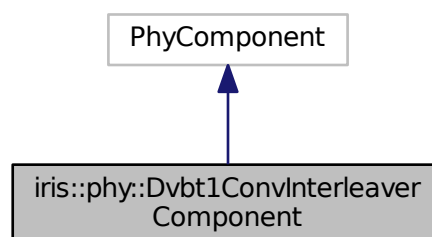
Definition at line 100 of file Dvbt1ConvEncoderComponent.h.

## 7.3 iris::phy::Dvbt1ConvInterleaverComponent Class Reference

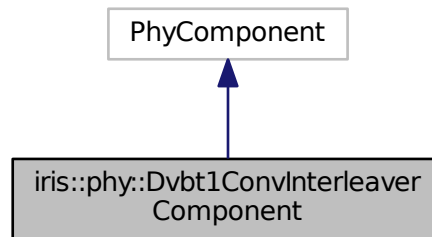
A DVB-T1 convolutional interleaver component.

```
#include <Dvbt1ConvInterleaverComponent.h>
```

Inheritance diagram for iris::phy::Dvbt1ConvInterleaverComponent:



Collaboration diagram for `iris::phy::Dvbt1ConvInterleaverComponent`:



## Public Types

- typedef `std::vector< uint8_t >` [ByteVec](#)  
A vector of bytes.
- typedef `ByteVec::iterator` [ByteVecIt](#)  
An iterator for a vector of bytes.

## Public Member Functions

- [Dvbt1ConvInterleaverComponent](#) (`std::string name`)  
Default constructor.
- [~Dvbt1ConvInterleaverComponent](#) ()  
Default destructor.
- virtual void [calculateOutputTypes](#) (`std::map< std::string, int > &inputTypes`, `std::map< std::string, int > &outputTypes`)  
Calculate the output port types for the IRIS system.
- virtual void [registerPorts](#) ()  
Register the interleaver ports with the IRIS system.
- virtual void [initialize](#) ()  
Initialize the component.
- virtual void [process](#) ()  
Main processing method.
- virtual void [parameterHasChanged](#) (`std::string name`)  
Actions taken when the parameters change.

## Private Member Functions

- void [setup](#) ()  
Set up offsets and clean interleaver registers.
- void [destroy](#) ()  
Destroy the component.

## Static Private Member Functions

- template<typename T , size\_t N>  
static T \* [begin](#) (T(&arr)[N])  
*Useful templates.*
- template<typename T , size\_t N>  
static T \* [end](#) (T(&arr)[N])

## Private Attributes

- bool [debug\\_x](#)  
*Debug flag (default = false)*
- double [timeStamp\\_](#)  
*Timestamp of current frame.*
- double [sampleRate\\_](#)  
*Sample rate of current frame.*
- int [b\\_](#) [12]  
*Interleaving status.*
- uint8\_t [l0\\_](#) [1]  
*First interleaving register, not used.*
- uint8\_t [l1\\_](#) [17]  
*Second interleaving register.*
- uint8\_t [l2\\_](#) [2 \* 17]  
*Third interleaving register.*
- uint8\_t [l3\\_](#) [3 \* 17]  
*Fourth interleaving register.*
- uint8\_t [l4\\_](#) [4 \* 17]  
*Fifth interleaving register.*
- uint8\_t [l5\\_](#) [5 \* 17]  
*Sixth interleaving register.*
- uint8\_t [l6\\_](#) [6 \* 17]  
*Seventh interleaving register.*
- uint8\_t [l7\\_](#) [7 \* 17]  
*Eighth interleaving register.*
- uint8\_t [l8\\_](#) [8 \* 17]  
*Ninth interleaving register.*
- uint8\_t [l9\\_](#) [9 \* 17]  
*Tenth interleaving register.*
- uint8\_t [l10\\_](#) [10 \* 17]  
*Eleventh interleaving register.*
- uint8\_t [l11\\_](#) [11 \* 17]  
*Twelfth interleaving register.*
- int [rsOffset\\_](#)  
*Input offset.*

### 7.3.1 Detailed Description

A DVB-T1 convolutional interleaver component.

[Dvbt1ConvInterleaverComponent](#) is the third block composing the DVB-T transmission chain. The purpose of this interleaver, placed between the R-S encoder and the convolutional encoder, is most useful at decoding time. In fact, the corresponding deinterleaver has the task to shuffle apart consecutive bursts of errors coming out from the Viterbi decoder, so that the R-S error correcting capability (up to 8 bytes in a codeword of 204 bytes) is not exceeded. The convolutional interleaving process is based on the Forney approach, which is compatible with the Ramsey type III approach, with a depth of  $I = 12$ . Each cell in every interleaving delay path is composed by  $M_I = 17$  bytes. The interleaved data bytes are composed of error protected packets and are delimited by inverted or non-inverted MPEG-2 sync bytes (204 bytes periodicity).

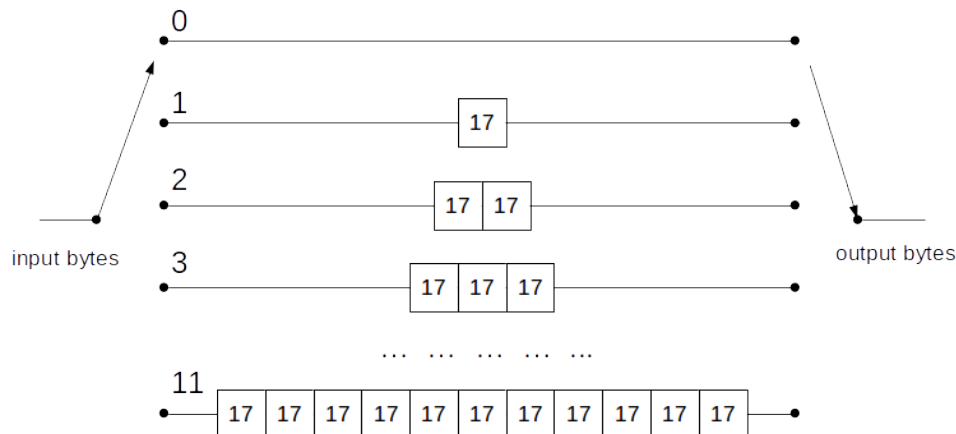


Figure 7.3: DVB-T convolutional interleaver.

Please note that the convolutional interleaver does not operate strictly as a row-column block interleaver, since it keeps memory of older bytes in the current block, and does not emit all the bytes in the current block. For a different implementation of this block operation, please refer also to the testing section implemented in MATLAB. In that case, the operation of this interleaver is performed using a block row-column interleaver that is *slant* after data load and before data dump.

There is only one parameter that can be changed in the XML configuration file:

- *debug*: by default set to "false", is used to print some small debugging information for the interested developer.

### References

- ETSI Standard: *EN 300 744 V1.5.1, Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for digital terrestrial television*, available at [ETSI Publications Download Area](#)
- Forney, G. D., *Burst-Correcting Codes for the Classic Bursty Channel*, IEEE Transactions on Communications, vol. COM-19, October 1971, pp. 772-781.
- Ramsey, J. L., *Realization of Optimum Interleavers*, IEEE Transactions on Information Theory, IT-16 (3), May 1970, pp. 338-345.

Definition at line 85 of file `Dvbt1ConvInterleaverComponent.h`.

### 7.3.2 Member Typedef Documentation

#### 7.3.2.1 `typedef std::vector<uint8_t> iris::phy::Dvbt1ConvInterleaverComponent::ByteVec`

A vector of bytes.

Definition at line 91 of file Dvbt1ConvInterleaverComponent.h.

### 7.3.2.2 typedef ByteVec::iterator iris::phy::Dvbt1ConvInterleaverComponent::ByteVecIt

An iterator for a vector of bytes.

Definition at line 94 of file Dvbt1ConvInterleaverComponent.h.

## 7.3.3 Constructor & Destructor Documentation

### 7.3.3.1 iris::phy::Dvbt1ConvInterleaverComponent::Dvbt1ConvInterleaverComponent ( std::string name )

Default constructor.

Registers the block parameters and initializes some variables

Definition at line 57 of file Dvbt1ConvInterleaverComponent.cpp.

References `debug_x`.

```

58 : PhyComponent (name,                                // component name
59                 "dvbt1convinterleaver",              // component type
60                 "A DVB-T1 convolutional interleaver component", // description
61                 "Giuseppe Baruffa",                  // author
62                 "0.1")                                // version
63 , sampleRate_(0)
64 , timeStamp_(0)
65 , rsOffset_(0)
66 {
67     registerParameter(
68         "debug", "Whether to output debug data",
69         "false", true, debug_x);
70 }
```

### 7.3.3.2 iris::phy::Dvbt1ConvInterleaverComponent::~Dvbt1ConvInterleaverComponent ( )

Default destructor.

Just calls `destroy()`.

Definition at line 75 of file Dvbt1ConvInterleaverComponent.cpp.

References `destroy()`.

```

76 {
77     destroy();
78 }
```

## 7.3.4 Member Function Documentation

### 7.3.4.1 template<typename T, size\_t N> static T\* iris::phy::Dvbt1ConvInterleaverComponent::begin ( T(&) arr[N] ) [inline], [static], [private]

Useful templates.

Definition at line 133 of file Dvbt1ConvInterleaverComponent.h.

```

133 { return &arr[0]; }
```

**7.3.4.2** `void iris::phy::Dvbt1ConvInterleaverComponent::calculateOutputTypes ( std::map< std::string, int > & inputTypes,  
std::map< std::string, int > & outputTypes )` `[virtual]`

Calculate the output port types for the IRIS system.

The single output port must provide bytes.

Definition at line 93 of file `Dvbt1ConvInterleaverComponent.cpp`.

```
96 {
97     outputTypes["output1"] = TypeInfo< uint8_t >::identifier;
98 }
```

**7.3.4.3** `void iris::phy::Dvbt1ConvInterleaverComponent::destroy ( )` `[private]`

Destroy the component.

Definition at line 309 of file `Dvbt1ConvInterleaverComponent.cpp`.

Referenced by `parameterHasChanged()`, and `~Dvbt1ConvInterleaverComponent()`.

```
310 {
311 }
```

**7.3.4.4** `template<typename T , size_t N> static T* iris::phy::Dvbt1ConvInterleaverComponent::end ( T(&) arr[N] )`  
`[inline], [static], [private]`

Definition at line 135 of file `Dvbt1ConvInterleaverComponent.h`.

```
135 { return &arr[0]+N; }
```

**7.3.4.5** `void iris::phy::Dvbt1ConvInterleaverComponent::initialize ( )` `[virtual]`

Initialize the component.

Just calls `setup()`.

Definition at line 103 of file `Dvbt1ConvInterleaverComponent.cpp`.

References `setup()`.

```
104 {
105     setup();
106 }
```

**7.3.4.6** `void iris::phy::Dvbt1ConvInterleaverComponent::parameterHasChanged ( std::string name )` `[virtual]`

Actions taken when the parameters change.

This block has no significant parameters

Definition at line 271 of file `Dvbt1ConvInterleaverComponent.cpp`.

References `destroy()`, and `setup()`.

```
272 {
273     if(name == "???" )
274     {
275         destroy();
276         setup();
277     }
278 }
```



## 7.3.4.7 void iris::phy::Dvbt1ConvInterleaverComponent::process ( ) [virtual]

Main processing method.

Definition at line 109 of file Dvbt1ConvInterleaverComponent.cpp.

References `b_`, `debug_x`, `l10_`, `l11_`, `l1_`, `l2_`, `l3_`, `l4_`, `l5_`, `l6_`, `l7_`, `l8_`, `l9_`, and `rsOffset_`.

```

110 {
111     // request input
112     DataSet< uint8_t >* in = NULL;
113     getInputDataSet("input1", in);
114
115     // calculate sizes
116     int insize = in ? (int) in->data.size() : 0;
117     int outsize = insize;
118
119     // request output
120     DataSet< uint8_t >* out = NULL;
121     getOutputDataSet("output1", out, outsize);
122
123     // print debug info
124     if(debug_x)
125         LOG(LINFO) << "in/out: " << insize << "/" << outsize;
126
127     // process data with a humongous Duff's device!
128     // For more info on what a Duff is, check the Wikipedia page
129     // https://en.wikipedia.org/wiki/Duff's_device
130     ByteVecIt init = in->data.begin();
131     ByteVecIt outit = out->data.begin();
132     switch(rsOffset_)
133     {
134         // in the following, we load a bunch of bytes in the generic path
135         // and advance the relevant pointer
136         case 0:
137         case 12:
138             do
139             {
140                 // first, direct path
141                 *outit++ = *init++;
142                 rsOffset_ = 1;
143                 if(init == in->data.end())
144                     break;
145
146             case 1:
147                 // second path
148                 *outit++ = l1_[b_[1]];
149                 l1_[b_[1]] = *init++;
150                 if(++(b_[1]) == 17 * 1)
151                     b_[1] = 0;
152                 rsOffset_++;
153                 if(init == in->data.end())
154                     break;
155
156             case 2:
157                 // third path
158                 *outit++ = l2_[b_[2]];
159                 l2_[b_[2]] = *init++;
160                 if(++(b_[2]) == 17 * 2)
161                     b_[2] = 0;
162                 rsOffset_++;
163                 if(init == in->data.end())
164                     break;
165
166             case 3:
167                 // fourth path
168                 *outit++ = l3_[b_[3]];
169                 l3_[b_[3]] = *init++;
170                 if(++(b_[3]) == 17 * 3)
171                     b_[3] = 0;
172                 rsOffset_++;
173                 if(init == in->data.end())
174                     break;
175
176             case 4:
177                 // fifth path
178                 *outit++ = l4_[b_[4]];
179                 l4_[b_[4]] = *init++;
180                 if(++(b_[4]) == 17 * 4)
181                     b_[4] = 0;
182                 rsOffset_++;
183                 if(init == in->data.end())
184                     break;
185
186             case 5:

```

```

187     // sixth path
188     *outit++ = I5_[b_[5]];
189     I5_[b_[5]] = *init++;
190     if(++(b_[5]) == 17 * 5)
191         b_[5] = 0;
192     rsOffset_++;
193     if(init == in->data.end())
194         break;
195
196     case 6:
197         // seventh path
198         *outit++ = I6_[b_[6]];
199         I6_[b_[6]] = *init++;
200         if(++(b_[6]) == 17 * 6)
201             b_[6] = 0;
202         rsOffset_++;
203         if(init == in->data.end())
204             break;
205
206     case 7:
207         // eighth path
208         *outit++ = I7_[b_[7]];
209         I7_[b_[7]] = *init++;
210         if(++(b_[7]) == 17 * 7)
211             b_[7] = 0;
212         rsOffset_++;
213         if(init == in->data.end())
214             break;
215
216     case 8:
217         // ninth path
218         *outit++ = I8_[b_[8]];
219         I8_[b_[8]] = *init++;
220         if(++(b_[8]) == 17 * 8)
221             b_[8] = 0;
222         rsOffset_++;
223         if(init == in->data.end())
224             break;
225
226     case 9:
227         // tenth path
228         *outit++ = I9_[b_[9]];
229         I9_[b_[9]] = *init++;
230         if(++(b_[9]) == 17 * 9)
231             b_[9] = 0;
232         rsOffset_++;
233         if(init == in->data.end())
234             break;
235
236     case 10:
237         // eleventh path
238         *outit++ = I10_[b_[10]];
239         I10_[b_[10]] = *init++;
240         if(++(b_[10]) == 17 * 10)
241             b_[10] = 0;
242         rsOffset_++;
243         if(init == in->data.end())
244             break;
245
246     case 11:
247         // twelfth and final path
248         *outit++ = I11_[b_[11]];
249         I11_[b_[11]] = *init++;
250         if(++(b_[11]) == 17 * 11)
251             b_[11] = 0;
252         rsOffset_++;
253         if(init == in->data.end())
254             break;
255
256     } while(true);
257 }
258
259 //Copy the timestamp and sample rate for the DataSets
260 out->timeStamp = in->timeStamp;
261 out->sampleRate = in->sampleRate;
262
263 // release input and output
264 releaseInputDataSet("input1", in);
265 releaseOutputDataSet("output1", out);
266 }

```

**7.3.4.8 void iris::phy::Dvbt1ConvInterleaverComponent::registerPorts ( ) [virtual]**

Register the interleaver ports with the IRIS system.

This component has one input that accepts bytes and one output that provides interleaved bytes.

Definition at line 84 of file Dvbt1ConvInterleaverComponent.cpp.

```
85 {
86     registerInputPort("input1", TypeInfo< uint8_t >::identifier);
87     registerOutputPort("output1", TypeInfo< uint8_t >::identifier);
88 }
```

**7.3.4.9 void iris::phy::Dvbt1ConvInterleaverComponent::setup ( ) [private]**

Set up offsets and clean interleaver registers.

Please note that filling the registers with zeroes, as we do below, generates peaky transients in the final waveform right after the system start-up and up to the moment when all registers are filled by real data bytes. In order to avoid this, we should fill this with a random sequence of bytes, instead of zeroes.

Definition at line 287 of file Dvbt1ConvInterleaverComponent.cpp.

References `b_`, `I0_`, `I10_`, `I11_`, `I1_`, `I2_`, `I3_`, `I4_`, `I5_`, `I6_`, `I7_`, `I8_`, `I9_`, and `rsOffset_`.

Referenced by `initialize()`, and `parameterHasChanged()`.

```
288 {
289     // clean registers
290     memset(b_, 0, sizeof(b_));
291     memset(I0_, 0, sizeof(I0_));
292     memset(I1_, 0, sizeof(I1_));
293     memset(I2_, 0, sizeof(I2_));
294     memset(I3_, 0, sizeof(I3_));
295     memset(I4_, 0, sizeof(I4_));
296     memset(I5_, 0, sizeof(I5_));
297     memset(I6_, 0, sizeof(I6_));
298     memset(I7_, 0, sizeof(I7_));
299     memset(I8_, 0, sizeof(I8_));
300     memset(I9_, 0, sizeof(I9_));
301     memset(I10_, 0, sizeof(I10_));
302     memset(I11_, 0, sizeof(I11_));
303
304     // reset the offset
305     rsOffset_ = 0;
306 }
```

**7.3.5 Member Data Documentation****7.3.5.1 int iris::phy::Dvbt1ConvInterleaverComponent::b\_[12] [private]**

Interleaving ststus.

Definition at line 116 of file Dvbt1ConvInterleaverComponent.h.

Referenced by `process()`, and `setup()`.

**7.3.5.2 bool iris::phy::Dvbt1ConvInterleaverComponent::debug\_x [private]**

Debug flag (default = false)

Definition at line 108 of file Dvbt1ConvInterleaverComponent.h.

Referenced by `Dvbt1ConvInterleaverComponent()`, and `process()`.

#### 7.3.5.3 `uint8_t iris::phy::Dvbt1ConvInterleaverComponent::l0_[1]` [private]

First interleaving register, not used.

Definition at line 117 of file `Dvbt1ConvInterleaverComponent.h`.

Referenced by `setup()`.

#### 7.3.5.4 `uint8_t iris::phy::Dvbt1ConvInterleaverComponent::l10_[10*17]` [private]

Eleventh interleaving register.

Definition at line 127 of file `Dvbt1ConvInterleaverComponent.h`.

Referenced by `process()`, and `setup()`.

#### 7.3.5.5 `uint8_t iris::phy::Dvbt1ConvInterleaverComponent::l11_[11*17]` [private]

Twelfth interleaving register.

Definition at line 128 of file `Dvbt1ConvInterleaverComponent.h`.

Referenced by `process()`, and `setup()`.

#### 7.3.5.6 `uint8_t iris::phy::Dvbt1ConvInterleaverComponent::l1_[17]` [private]

Second interleaving register.

Definition at line 118 of file `Dvbt1ConvInterleaverComponent.h`.

Referenced by `process()`, and `setup()`.

#### 7.3.5.7 `uint8_t iris::phy::Dvbt1ConvInterleaverComponent::l2_[2*17]` [private]

Third interleaving register.

Definition at line 119 of file `Dvbt1ConvInterleaverComponent.h`.

Referenced by `process()`, and `setup()`.

#### 7.3.5.8 `uint8_t iris::phy::Dvbt1ConvInterleaverComponent::l3_[3*17]` [private]

Fourth interleaving register.

Definition at line 120 of file `Dvbt1ConvInterleaverComponent.h`.

Referenced by `process()`, and `setup()`.

#### 7.3.5.9 `uint8_t iris::phy::Dvbt1ConvInterleaverComponent::l4_[4*17]` [private]

Fifth interleaving register.

Definition at line 121 of file `Dvbt1ConvInterleaverComponent.h`.

Referenced by `process()`, and `setup()`.

#### 7.3.5.10 `uint8_t iris::phy::Dvbt1ConvInterleaverComponent::l5_[5*17]` [private]

Sixth interleaving register.

Definition at line 122 of file `Dvbt1ConvInterleaverComponent.h`.

Referenced by process(), and setup().

**7.3.5.11** `uint8_t iris::phy::Dvbt1ConvInterleaverComponent::l6[17]` `[private]`

Seventh interleaving register.

Definition at line 123 of file Dvbt1ConvInterleaverComponent.h.

Referenced by process(), and setup().

**7.3.5.12** `uint8_t iris::phy::Dvbt1ConvInterleaverComponent::l7[17]` `[private]`

Eighth interleaving register.

Definition at line 124 of file Dvbt1ConvInterleaverComponent.h.

Referenced by process(), and setup().

**7.3.5.13** `uint8_t iris::phy::Dvbt1ConvInterleaverComponent::l8[17]` `[private]`

Ninth interleaving register.

Definition at line 125 of file Dvbt1ConvInterleaverComponent.h.

Referenced by process(), and setup().

**7.3.5.14** `uint8_t iris::phy::Dvbt1ConvInterleaverComponent::l9[17]` `[private]`

Tenth interleaving register.

Definition at line 126 of file Dvbt1ConvInterleaverComponent.h.

Referenced by process(), and setup().

**7.3.5.15** `int iris::phy::Dvbt1ConvInterleaverComponent::rsOffset_` `[private]`

Input offset.

Definition at line 129 of file Dvbt1ConvInterleaverComponent.h.

Referenced by process(), and setup().

**7.3.5.16** `double iris::phy::Dvbt1ConvInterleaverComponent::sampleRate_` `[private]`

Sample rate of current frame.

Definition at line 114 of file Dvbt1ConvInterleaverComponent.h.

**7.3.5.17** `double iris::phy::Dvbt1ConvInterleaverComponent::timeStamp_` `[private]`

Timestamp of current frame.

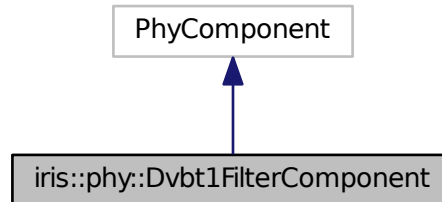
Definition at line 113 of file Dvbt1ConvInterleaverComponent.h.

## 7.4 iris::phy::Dvbt1FilterComponent Class Reference

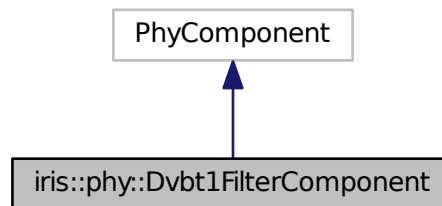
A DVB-T1 filter component.

```
#include <Dvbt1FilterComponent.h>
```

Inheritance diagram for `iris::phy::Dvbt1FilterComponent`:



Collaboration diagram for `iris::phy::Dvbt1FilterComponent`:



## Public Types

- typedef `std::vector< uint8_t >` [ByteVec](#)  
*A vector of bytes.*
- typedef `ByteVec::iterator` [ByteVecIt](#)  
*An iterator for a vector of bytes.*
- typedef `std::complex< float >` [Cplx](#)  
*A complex type.*
- typedef `std::vector< Cplx >` [CplxVec](#)  
*A vector of complex.*
- typedef `CplxVec::iterator` [CplxVecIt](#)
- typedef `std::vector< float >` [FloatVec](#)  
*A vector of float.*
- typedef `FloatVec::iterator` [FloatVecIt](#)  
*An iterator for a vector of float.*
- typedef `std::vector< int >` [IntVec](#)  
*A vector of integers.*
- typedef `IntVec::iterator` [IntVecIt](#)  
*An iterator for a vector of integers.*

## Public Member Functions

- [Dvbt1FilterComponent](#) (std::string name)  
*Default constructor.*
- [~Dvbt1FilterComponent](#) ()  
*Default destructor.*
- virtual void [calculateOutputTypes](#) (std::map< std::string, int > &inputTypes, std::map< std::string, int > &outputTypes)  
*Calculate the output port types for the IRIS system.*
- virtual void [registerPorts](#) ()  
*Register the mapper ports with the IRIS system.*
- virtual void [initialize](#) ()  
*Initialize the component.*
- virtual void [process](#) ()  
*Main processing method.*
- virtual void [parameterHasChanged](#) (std::string name)  
*Actions taken when the parameters change.*

## Private Member Functions

- void [setup](#) ()  
*Set up all our index vectors and containers.*
- void [destroy](#) ()  
*Destroy the component.*
- int [kaiser\\_design](#) (int \*order, double \*beta, double ripple, double width)  
*Find Kaiser parameters.*
- int [filter\\_design](#) (FloatVec &h, int order, double fc)  
*Design a Kaiser-windowed low-pass filter.*
- double [kaiser\\_window](#) (int n, int order, double beta)  
*Find Kaiser window coefficients.*
- double [sinc](#) (double x)  
 *$\sin(x)/x$  function*
- double [factorial](#) (int n)  
*factorial function*
- double [bessel\\_I0](#) (double x)  
*Zeroth Order Modified Bessel Function.*

## Static Private Member Functions

- template<typename T, size\_t N>  
static T \* [begin](#) (T(&arr)[N])  
*Useful templates.*
- template<typename T, size\_t N>  
static T \* [end](#) (T(&arr)[N])

## Private Attributes

- bool `debug_x`  
*Debug flag (default = false)*
- double `sampleRate_x`  
*Sampling rate (default = 0)*
- double `stopBand_x`  
*Filter stop-band (default = 4000000)*
- double `sBAttenuation_x`  
*Filter stop-band attenuation (default = 35)*
- std::string `coeffsFile_x`  
*Text file with impulse response (default = none)*
- double `timeStamp_x`  
*Timestamp of current frame.*
- double `sampleRate_`  
*Sample rate of current frame.*
- bool `symmetric_`
- int `filterLength_`
- FloatVec `coeffp_`
- CplxVec `work_`

### 7.4.1 Detailed Description

A DVB-T1 filter component.

`Dvbt1FilterComponent` is the second optional block composing the DVB-T transmission chain. It is required only if the spectrum emission mask (SEM) has to be obeyed directly at the BB level and cannot be modified operating on the RF emitted signal. This filter also helps to reduce the IF images resulting from the interpolation process, if the DAC sampling rate is not directly compatible with the DVB-T sampling rate.

This block implements a Kaiser-designed FIR lowpass filter, whose number of taps is decided by the attenuation and transition bandwidth values. Please note that setting high values of attenuation or a steep transition bandwidth could result in a high number of taps, and the filter could not be able to operate in real time.

This block accepts in input complex float values and generates in output complex float values.

There are parameters several that can be changed in the XML configuration file:

- *debug*: by default set to "false", is used to print some small debugging information for the interested developer.
- *samplerate*: by default set to "0", a placeholder for 64e6/7 Hz. This represents the sampling rate of the DAC signal and, consequently, the whole bandwidth over which the filter may operate.
- *stopband*: by default set to "4000000.0", it represents the frequency at which the specified attenuation is achieved. This frequency is given relatively to the centre frequency of the RF emitted signal. The transition bandwidth of the filter ends at this frequency, and it begins right after the last active OFDM carrier, which happens to be at 3.805 MHz for an 8K system.
- *attenuation*: by default set to "35.0", it is the attenuation (in dB) of the filter at the specified stop frequency.
- *coeffsfile*: by default set to "", which means not enabled, this is the name of a text file where the impulse response of the filter is saved, line after line.

## References

- ETSI Standard: *EN 300 744 V1.5.1, Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for digital terrestrial television*, available at [ETSI Publications Download Area](#)

Definition at line 90 of file `Dvbt1FilterComponent.h`.



## 7.4.2 Member Typedef Documentation

### 7.4.2.1 `typedef std::vector<uint8_t> iris::phy::Dvbt1FilterComponent::ByteVec`

A vector of bytes.

Definition at line 96 of file Dvbt1FilterComponent.h.

### 7.4.2.2 `typedef ByteVec::iterator iris::phy::Dvbt1FilterComponent::ByteVecIt`

An iterator for a vector of bytes.

Definition at line 99 of file Dvbt1FilterComponent.h.

### 7.4.2.3 `typedef std::complex<float> iris::phy::Dvbt1FilterComponent::Cplx`

A complex type.

Definition at line 102 of file Dvbt1FilterComponent.h.

### 7.4.2.4 `typedef std::vector<Cplx> iris::phy::Dvbt1FilterComponent::CplxVec`

A vector of complex.

Definition at line 105 of file Dvbt1FilterComponent.h.

### 7.4.2.5 `typedef CplxVec::iterator iris::phy::Dvbt1FilterComponent::CplxVecIt`

Definition at line 108 of file Dvbt1FilterComponent.h.

### 7.4.2.6 `typedef std::vector<float> iris::phy::Dvbt1FilterComponent::FloatVec`

A vector of float.

Definition at line 111 of file Dvbt1FilterComponent.h.

### 7.4.2.7 `typedef FloatVec::iterator iris::phy::Dvbt1FilterComponent::FloatVecIt`

An iterator for a vector of float.

Definition at line 114 of file Dvbt1FilterComponent.h.

### 7.4.2.8 `typedef std::vector<int> iris::phy::Dvbt1FilterComponent::IntVec`

A vector of integers.

Definition at line 117 of file Dvbt1FilterComponent.h.

### 7.4.2.9 `typedef IntVec::iterator iris::phy::Dvbt1FilterComponent::IntVecIt`

An iterator for a vector of integers.

Definition at line 120 of file Dvbt1FilterComponent.h.

### 7.4.3 Constructor & Destructor Documentation

#### 7.4.3.1 iris::phy::Dvbt1FilterComponent::Dvbt1FilterComponent ( std::string name )

Default constructor.

Registers the block parameters and initializes some variables

Definition at line 57 of file Dvbt1FilterComponent.cpp.

References coeffsFile\_x, debug\_x, sampleRate\_x, sBAttenuation\_x, and stopBand\_x.

```

58 : PhyComponent (name,                                // component name
59                 "dvbt1filter",                        // component type
60                 "A DVB-T1 filter component",          // description
61                 "Giuseppe Baruffa",                  // author
62                 "0.1")                                // version
63 , sampleRate_(0)
64 , timeStamp_(0)
65 {
66     registerParameter(
67         "debug", "Whether to output debug data",
68         "false", true, debug_x);
69
70     registerParameter(
71         "samplerate", "Sampling rate (use 0 for 9142857)",
72         "0.0", true, sampleRate_x, Interval<double>(0.0,15000000.0));
73
74     registerParameter(
75         "stopband", "Stop-band of the filter, in Hz relative to the "
76         "centre frequency", "4000000.0", true, stopBand_x,
77         Interval<double>(2000000.0,10000000.0));
78
79     registerParameter(
80         "attenuation", "Attenuation in the stop-band, in dB: 0 disables "
81         "filtering (35 is the value tested at Electrosys)", "35.0", true,
82         sBAttenuation_x, Interval<double>(0.0,90.0));
83
84     registerParameter(
85         "coeffsfile", "Text file with the filter impulse response",
86         "", true, coeffsFile_x);
87 }
```

#### 7.4.3.2 iris::phy::Dvbt1FilterComponent::~~Dvbt1FilterComponent ( )

Default destructor.

Just calls [destroy\(\)](#).

Definition at line 92 of file Dvbt1FilterComponent.cpp.

References [destroy\(\)](#).

```

93 {
94     destroy\(\);
95 }
```

### 7.4.4 Member Function Documentation

#### 7.4.4.1 template<typename T , size\_t N> static T\* iris::phy::Dvbt1FilterComponent::begin ( T(&) arr[N] ) [inline], [static], [private]

Useful templates.

Definition at line 159 of file Dvbt1FilterComponent.h.

```

159 { return &arr[0]; }
```

7.4.4.2 double iris::phy::Dvbt1FilterComponent::bessel\_I0 ( double x ) [private]

Zeroth Order Modified Bessel Function.

**Parameters**

$x$	The input value
-----	-----------------

**Returns**

The function evaluated on  $x$

Definition at line 254 of file Dvbt1FilterComponent.cpp.

References `factorial()`.

Referenced by `kaiser_window()`.

```

255 {
256     double I0 = 1.0;
257     int i = 0;
258     for (i = 1; i <= 20; i++) {
259         I0 += pow(x / 2.0, (double) (2 * i)) / pow(factorial(i), 2.0);
260     }
261     return I0;
262 }
```

**7.4.4.3** `void iris::phy::Dvbt1FilterComponent::calculateOutputTypes ( std::map< std::string, int > &inputTypes, std::map< std::string, int > &outputTypes )` `[virtual]`

Calculate the output port types for the IRIS system.

The single output port must provide complex values.

Definition at line 110 of file Dvbt1FilterComponent.cpp.

```

113 {
114     outputTypes["output1"] = TypeInfo< Cplx >::identifier;
115 }
```

**7.4.4.4** `void iris::phy::Dvbt1FilterComponent::destroy ( )` `[private]`

Destroy the component.

Definition at line 443 of file Dvbt1FilterComponent.cpp.

Referenced by `parameterHasChanged()`, and `~Dvbt1FilterComponent()`.

```

444 {
445 }
```

**7.4.4.5** `template<typename T, size_t N> static T* iris::phy::Dvbt1FilterComponent::end ( T(&) arr[N] )` `[inline]`, `[static]`, `[private]`

Definition at line 161 of file Dvbt1FilterComponent.h.

```

161 { return &arr[0]+N; }
```

**7.4.4.6** `double iris::phy::Dvbt1FilterComponent::factorial ( int n )` `[private]`

factorial function

## Parameters

<i>n</i>	the integer on which to apply the factorial
----------	---

## Returns

the factorial of *n*

Definition at line 241 of file Dvbt1FilterComponent.cpp.

Referenced by `bessel_I0()`.

```

242 {
243     double fact = 1.0;
244     int i = 0;
245     for (i = 1; i <= n; i++)
246         fact *= (double) i;
247     return fact;
248 }
```

#### 7.4.4.7 int iris::phy::Dvbt1FilterComponent::filter\_design ( FloatVec & *h*, int *order*, double *fc* ) [private]

Design a Kaiser-windowed low-pass filter.

Implementation inspired from <http://www.labbookpages.co.uk/audio/firWindowing.html>

## Parameters

<i>fc</i>	The cut frequency is normalized to the sampling frequency
<i>order</i>	Order of the filter (as calculated by <code>kaiser_design</code> )
<i>h</i>	The array of filter taps

## Returns

0 in case of success

Definition at line 304 of file Dvbt1FilterComponent.cpp.

References `sinc()`.

Referenced by `setup()`.

```

305 {
306     h.resize(order + 1);
307     for (int i = 0; i <= order; i++) {
308         h[i] = (float) (2.0 * fc * sinc(2.0 * M_PI * fc * ((double) i
309             - (double) order / 2.0)));
310     }
311
312     return 0;
313 }
```

#### 7.4.4.8 void iris::phy::Dvbt1FilterComponent::initialize ( ) [virtual]

Initialize the component.

Just calls `setup()`.

Definition at line 120 of file Dvbt1FilterComponent.cpp.

References `setup()`.

```

121 {
122     setup();
123 }
```

**7.4.4.9** `int iris::phy::Dvbt1FilterComponent::kaiser_design ( int * order, double * beta, double ripple, double width )`  
`[private]`

Find Kaiser parameters.

Implementation inspired from <http://www.labbookpages.co.uk/audio/firWindowing.html>

Parameters

<i>ripple</i>	Ripple of the filter (linear)
<i>width</i>	Bandwidth normalized to sampling frequency
<i>beta</i>	The $\beta$ of the Kaiser window
<i>order</i>	The order of the filter (number of taps minus one)

Returns

0 in case of success

Definition at line 274 of file Dvbt1FilterComponent.cpp.

Referenced by `setup()`.

```

276 {
277     double A = -20.0 * log(ripple) / log(10.0);
278     double tw = 2.0 * M_PI * width;
279
280     if (A > 21.0)
281         *order = (int) ceil((A - 7.95) / (2.285 * tw));
282     else
283         *order = (int) ceil(5.79 / tw);
284
285     if (A <= 21.0)
286         *beta = 0.0;
287     else if (21.0 < A && A <= 50.0)
288         *beta = 0.5842 * pow(A - 21.0, 0.4) + 0.07886 * (A - 21.0);
289     else
290         *beta = 0.1102 * (A - 8.7);
291
292     return 0;
293 }
```

**7.4.4.10** `double iris::phy::Dvbt1FilterComponent::kaiser_window ( int n, int order, double beta )` `[private]`

Find Kaiser window coefficients.

Inspired from <http://www.labbookpages.co.uk/audio/firWindowing.html>

Parameters

<i>n</i>	The lag at which to evaluate the Kaiser function
<i>beta</i>	The $\beta$ of the Kaiser window
<i>order</i>	The order of the filter (number of taps minus one)

Returns

The amplitude of the filter tap

Definition at line 323 of file Dvbt1FilterComponent.cpp.

References `bessel_I0()`.

Referenced by `setup()`.

```

324 {
325     return bessel_I0(beta * sqrt(1.0 - pow(((double) (2 * n) / (double) order)
326         - 1.0, 2.0))) / bessel_I0(beta);
327 }
```

**7.4.4.11 void iris::phy::Dvbt1FilterComponent::parameterHasChanged ( std::string *name* ) [virtual]**

Actions taken when the parameters change.

This block has several significant parameters

Definition at line 219 of file Dvbt1FilterComponent.cpp.

References `destroy()`, and `setup()`.

```

220 {
221     if(name == "stopband" || name == "attenuation")
222     {
223         destroy();
224         setup();
225     }
226 }
```

**7.4.4.12 void iris::phy::Dvbt1FilterComponent::process ( ) [virtual]**

Main processing method.

Definition at line 126 of file Dvbt1FilterComponent.cpp.

References `coeffp_`, `debug_x`, `filterLength_`, `symmetric_`, and `work_`.

```

127 {
128     // request input
129     DataSet< Cplx > *in = NULL;
130     getInputDataSet("input1", in);
131
132     // calculate sizes
133     int insize = in ? (int) in->data.size() : 0;
134     int outsize = insize;
135
136     // request output and pre-fill with zeroes
137     DataSet< Cplx > *out = NULL;
138     getOutputDataSet("output1", out, insize);
139     fill(out->data.begin(), out->data.end(), Cplx(0,0));
140
141     // print debug info
142     if(debug_x)
143         LOG(LINFO) << "in/out: " << insize << "/" << outsize;
144
145     // copy head
146     CplxVecIt workit = work_.begin() + filterLength_ - 1;
147     copy(in->data.begin(), in->data.end() + filterLength_ - 1, workit);
148
149     // filter!
150     CplxVecIt outit = out->data.begin();
151     if(symmetric_)
152     {
153         // symmetric filter
154         for(int n = 0; n < filterLength_ - 1; n++, outit++)
155         {
156             CplxVecIt init = workit + n;
157             CplxVecIt inlastit = workit + n - filterLength_ + 1;
158             FloatVecIt coeffit = coeffp_.begin();
159             for(; init > inlastit; init--, inlastit++, coeffit++)
160             {
161                 outit->real(outit->real() + *coeffit * (init->real() + inlastit->real()));
162                 outit->imag(outit->imag() + *coeffit * (init->imag() + inlastit->imag()));
163             }
164             outit->real(outit->real() + *coeffit * init->real());
165             outit->imag(outit->imag() + *coeffit * init->imag());
166         }
167         for(int n = filterLength_ - 1; n < insize; n++, outit++)
168         {
169             CplxVecIt init = in->data.begin() + n;
170             CplxVecIt inlastit = in->data.begin() + n - filterLength_ + 1;
171             FloatVecIt coeffit = coeffp_.begin();
172             for(; init > inlastit; init--, inlastit++, coeffit++)
173             {
174                 outit->real(outit->real() + *coeffit * (init->real() + inlastit->real()));
175                 outit->imag(outit->imag() + *coeffit * (init->imag() + inlastit->imag()));
176             }
177             outit->real(outit->real() + *coeffit * init->real());
178             outit->imag(outit->imag() + *coeffit * init->imag());
179         }
180     }
181 }
```

```

180     } else {
181         // asymmetric filter - double work
182         for(int n = 0; n < filterLength_ - 1; n++, outit++)
183         {
184             CplxVecIt init = workit + n;
185             for(FloatVecIt coeffit = coefffp_.begin(); coeffit <
coefffp_.end();
186                 coeffit++, init--)
187             {
188                 outit->real(outit->real() + *coeffit * init->real());
189                 outit->imag(outit->imag() + *coeffit * init->imag());
190             }
191         }
192         for(int n = filterLength_ - 1; n < insize; n++, outit++)
193         {
194             CplxVecIt init = in->data.begin() + n;
195             for(FloatVecIt coeffit = coefffp_.begin(); coeffit <
coefffp_.end();
196                 coeffit++, init--)
197             {
198                 outit->real(outit->real() + *coeffit * init->real());
199                 outit->imag(outit->imag() + *coeffit * init->imag());
200             }
201         }
202     }
203
204     // copy tail in previous
205     copy(in->data.end() - (filterLength_ - 1), in->data.end(), work_.begin());
206
207     // Copy the timestamp and sample rate for the DataSets
208     out->timeStamp = in->timeStamp;
209     out->sampleRate = in->sampleRate;
210
211     // release input and output
212     releaseOutputDataSet("output1", out);
213     releaseInputDataSet("input1", in);
214 }

```

#### 7.4.4.13 void iris::phy::Dvbt1FilterComponent::registerPorts ( ) [virtual]

Register the mapper ports with the IRIS system.

This component has one input that accept complex float values and one output that provides complex float values.

Definition at line 101 of file Dvbt1FilterComponent.cpp.

```

102 {
103     registerInputPort("input1", TypeInfo< Cplx >::identifier);
104     registerOutputPort("output1", TypeInfo< Cplx >::identifier);
105 }

```

#### 7.4.4.14 void iris::phy::Dvbt1FilterComponent::setup ( ) [private]

Set up all our index vectors and containers.

Definition at line 336 of file Dvbt1FilterComponent.cpp.

References coefffp\_, coeffsFile\_x, filter\_design(), filterLength\_, kaiser\_design(), kaiser\_window(), MAX\_FILTER\_LENGTH, sampleRate\_x, sBAttenuation\_x, stopBand\_x, symmetric\_, and work\_.

Referenced by initialize(), and parameterHasChanged().

```

337 {
338     // replace the DVB-T sample rate with its real value
339     if(sampleRate_x == 0)
340         sampleRate_x = 64.0e6/7.0;
341
342     // clear
343     symmetric_ = true;
344     filterLength_ = 1;
345     coefffp_.resize(1);
346     coefffp_[0] = 1;
347
348     // test section, leave disabled
349     if(false)

```



```

350 {
351     symmetric_ = true;
352     filterLength_ = 123;
353     coeffp_.resize(filterLength_);
354     for(int i = 0; i < filterLength_; i++)
355         coeffp_[i] = ((double) i / 4.0) / filterLength_;
356 }
357
358 // design the transmission filter if requested
359 if(true && sBAttenuation_x)
360 {
361     // checks (you can try to modify the limits, but long filters could result
362     if(stopBand_x < 0.515 * (64.0e6/7.0) * (1705.0 / 2048.0))
363         LOG(LError) << "The selected stopband is too next to the passband: "
364             << stopBand_x;
365     if(stopBand_x > 0.485 * sampleRate_x)
366         LOG(LError) << "The selected stopband is too next to the sampling band: "
367             << sampleRate_x;
368     if(sBAttenuation_x > 40)
369         LOG(LError) << "A maximum attenuation of 40 dB can be specified";
370     if (sBAttenuation_x < 5)
371         LOG(LError) << "A minimum attenuation of 5 dB can be specified";
372
373     // the transition width is between the last carrier edge and the stopband
374     double tw = stopBand_x - 0.5 * (64.0e6/7.0) * (1705.0 / 2048.0);
375
376     // the cutoff frequency is at the last carrier edge plus half transition width
377     double fc = 0.501 * (64.0e6/7.0) * (1705.0 / 2048.0) + tw / 2;
378
379     // normalize to sample frequency
380     tw /= sampleRate_x;
381     fc /= sampleRate_x;
382
383     // the ripple
384     double ripple = pow(10.0, - sBAttenuation_x / 20.0);
385
386     // find kaiser parameters
387     double beta = 0.0;
388     int order = 0;
389     int status = kaiser_design(&order, &beta, ripple, tw);
390     if(status)
391         LOG(LError) << "Could not design the Kaiser window";
392
393     // ensure an integer-delay filter is designed (odd length)
394     filterLength_ = (2 * ((order + 1) / 2)) + 1;
395
396     // check
397     if(filterLength_ > MAX_FILTER_LENGTH)
398         LOG(LError) << "The maximum filter length has been exceeded: relax the "
399             "filtering performance";
400
401     // design base filter
402     status = filter_design(coeffp_, filterLength_ - 1, fc);
403     if(status)
404         LOG(LError) << "Could not design the base filter";
405
406     // windowed filter
407     for(int m = 0; m < filterLength_; m++)
408         coeffp_[m] *= (float) kaiser_window(m, filterLength_ - 1, beta);
409
410     // dump filter coefficients to file
411     if(!coeffsFile_x.empty())
412     {
413         FILE *fp = fopen(coeffsFile_x.c_str(), "wt");
414         if(fp)
415         {
416             setlocale(LC_NUMERIC, "C");
417             for(int m = 0; m < filterLength_; m++)
418                 fprintf(fp, "%.8f\n", coeffp_[m]);
419             fclose(fp);
420         }
421     }
422
423     // discover if the filter is symmetric or asymmetric
424     // This isn't really needed, since the filter will always be symmetrical
425     double maxtol = 1.0E-8, tol = 0.0;
426     for (int m = 0; m < filterLength_ / 2; m++)
427         tol += fabs(coeffp_[m] - coeffp_[filterLength_ - 1 - m]);
428     if (tol < maxtol)
429         symmetric_ = true;
430     else
431         symmetric_ = false;
432 }
433
434 // resume filter characteristics
435 LOG(LINFO) << (symmetric_ ? "Symmetric" : "Asymmetric") << " filter, "
436     << filterLength_ << " taps";

```

```

437
438 // working initial array
439 work_.resize(filterLength_ - 1 + filterLength_ - 1);
440 }

```

#### 7.4.4.15 double iris::phy::Dvbt1FilterComponent::sinc ( double x ) [private]

sin(x)/x function

##### Parameters

x	Input value
---	-------------

##### Returns

The sinc of the input

Definition at line 232 of file Dvbt1FilterComponent.cpp.

Referenced by filter\_design().

```

233 {
234     return x == 0.0 ? 1.0 : (sin(x) / x);
235 }

```

### 7.4.5 Member Data Documentation

#### 7.4.5.1 FloatVec iris::phy::Dvbt1FilterComponent::coeffp\_ [private]

Definition at line 147 of file Dvbt1FilterComponent.h.

Referenced by process(), and setup().

#### 7.4.5.2 std::string iris::phy::Dvbt1FilterComponent::coeffsFile\_x [private]

Text file with impulse response (default = none)

Definition at line 138 of file Dvbt1FilterComponent.h.

Referenced by Dvbt1FilterComponent(), and setup().

#### 7.4.5.3 bool iris::phy::Dvbt1FilterComponent::debug\_x [private]

Debug flag (default = false)

Definition at line 134 of file Dvbt1FilterComponent.h.

Referenced by Dvbt1FilterComponent(), and process().

#### 7.4.5.4 int iris::phy::Dvbt1FilterComponent::filterLength\_ [private]

Definition at line 146 of file Dvbt1FilterComponent.h.

Referenced by process(), and setup().

#### 7.4.5.5 double iris::phy::Dvbt1FilterComponent::sampleRate\_ [private]

Sample rate of current frame.

Definition at line 144 of file Dvbt1FilterComponent.h.

#### 7.4.5.6 double iris::phy::Dvbt1FilterComponent::sampleRate\_x [private]

Sampling rate (default = 0)

Definition at line 135 of file Dvbt1FilterComponent.h.

Referenced by Dvbt1FilterComponent(), and setup().

#### 7.4.5.7 double iris::phy::Dvbt1FilterComponent::sBAttenuation\_x [private]

Filter stop-band attenuation (default = 35)

Definition at line 137 of file Dvbt1FilterComponent.h.

Referenced by Dvbt1FilterComponent(), and setup().

#### 7.4.5.8 double iris::phy::Dvbt1FilterComponent::stopBand\_x [private]

Filter stop-band (default = 4000000)

Definition at line 136 of file Dvbt1FilterComponent.h.

Referenced by Dvbt1FilterComponent(), and setup().

#### 7.4.5.9 bool iris::phy::Dvbt1FilterComponent::symmetric\_ [private]

Definition at line 145 of file Dvbt1FilterComponent.h.

Referenced by process(), and setup().

#### 7.4.5.10 double iris::phy::Dvbt1FilterComponent::timeStamp\_ [private]

Timestamp of current frame.

Definition at line 143 of file Dvbt1FilterComponent.h.

#### 7.4.5.11 CplxVec iris::phy::Dvbt1FilterComponent::work\_ [private]

Definition at line 148 of file Dvbt1FilterComponent.h.

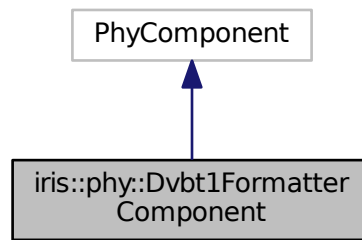
Referenced by process(), and setup().

## 7.5 iris::phy::Dvbt1FormatterComponent Class Reference

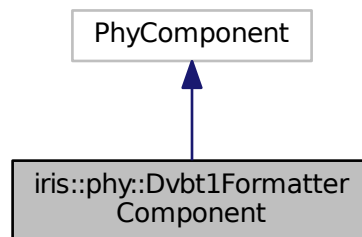
A DVB-T1 formatter component.

```
#include <Dvbt1FormatterComponent.h>
```

Inheritance diagram for `iris::phy::Dvbt1FormatterComponent`:



Collaboration diagram for `iris::phy::Dvbt1FormatterComponent`:



## Public Types

- typedef `std::vector< uint8_t >` [ByteVec](#)
- typedef `ByteVec::iterator` [ByteVecIt](#)
- typedef `std::complex< float >` [Cplx](#)
- typedef `std::vector< Cplx >` [CplxVec](#)
- typedef `CplxVec::iterator` [CplxVecIt](#)
- typedef `std::vector< int16_t >` [ShortVec](#)
- typedef `ShortVec::iterator` [ShortVecIt](#)

## Public Member Functions

- [Dvbt1FormatterComponent](#) (`std::string name`)
- [~Dvbt1FormatterComponent](#) ()
- virtual void [calculateOutputTypes](#) (`std::map< std::string, int > &inputTypes, std::map< std::string, int > &outputTypes`)
- virtual void [registerPorts](#) ()
- virtual void [initialize](#) ()
- virtual void [process](#) ()
- virtual void [parameterHasChanged](#) (`std::string name`)

## Private Member Functions

- void `setup` ()  
*Set up all our index vectors and containers.*
- void `destroy` ()

## Static Private Member Functions

- template<typename T, size\_t N>  
static T \* `begin` (T(&arr)[N])
- template<typename T, size\_t N>  
static T \* `end` (T(&arr)[N])

## Private Attributes

- bool `debug_x`  
*Debug flag (default = false)*
- double `timeStamp_`  
*Timestamp of current frame.*
- double `sampleRate_`  
*Sample rate of current frame.*

### 7.5.1 Detailed Description

A DVB-T1 formatter component.

Definition at line 48 of file Dvbt1FormatterComponent.h.

### 7.5.2 Member Typedef Documentation

#### 7.5.2.1 typedef std::vector<uint8\_t> iris::phy::Dvbt1FormatterComponent::ByteVec

Definition at line 53 of file Dvbt1FormatterComponent.h.

#### 7.5.2.2 typedef ByteVec::iterator iris::phy::Dvbt1FormatterComponent::ByteVecIt

Definition at line 54 of file Dvbt1FormatterComponent.h.

#### 7.5.2.3 typedef std::complex<float> iris::phy::Dvbt1FormatterComponent::Cplx

Definition at line 55 of file Dvbt1FormatterComponent.h.

#### 7.5.2.4 typedef std::vector<Cplx> iris::phy::Dvbt1FormatterComponent::CplxVec

Definition at line 56 of file Dvbt1FormatterComponent.h.

#### 7.5.2.5 typedef CplxVec::iterator iris::phy::Dvbt1FormatterComponent::CplxVecIt

Definition at line 57 of file Dvbt1FormatterComponent.h.

### 7.5.2.6 `typedef std::vector<int16_t> iris::phy::Dvbt1FormatterComponent::ShortVec`

Definition at line 58 of file Dvbt1FormatterComponent.h.

### 7.5.2.7 `typedef ShortVec::iterator iris::phy::Dvbt1FormatterComponent::ShortVecIt`

Definition at line 59 of file Dvbt1FormatterComponent.h.

## 7.5.3 Constructor & Destructor Documentation

### 7.5.3.1 `iris::phy::Dvbt1FormatterComponent::Dvbt1FormatterComponent ( std::string name )`

Definition at line 55 of file Dvbt1FormatterComponent.cpp.

References `debug_x`.

```

56 : PhyComponent(name,                                // component name
57     "dvbt1formatter",                               // component type
58     "A DVB-T1 formatter component", // description
59     "Giuseppe Baruffa",           // author
60     "0.1")                         // version
61 ,sampleRate_(0)
62 ,timeStamp_(0)
63 {
64     registerParameter(
65         "debug", "Whether to output debug data",
66         "false", true, debug_x);
67 }
```

### 7.5.3.2 `iris::phy::Dvbt1FormatterComponent::~~Dvbt1FormatterComponent ( )`

Definition at line 69 of file Dvbt1FormatterComponent.cpp.

References `destroy()`.

```

70 {
71     destroy();
72 }
```

## 7.5.4 Member Function Documentation

### 7.5.4.1 `template<typename T, size_t N> static T* iris::phy::Dvbt1FormatterComponent::begin ( T(&) arr[N] ) [inline], [static], [private]`

Definition at line 82 of file Dvbt1FormatterComponent.h.

```

82 { return &arr[0]; }
```

### 7.5.4.2 `void iris::phy::Dvbt1FormatterComponent::calculateOutputTypes ( std::map< std::string, int > & inputTypes, std::map< std::string, int > & outputTypes ) [virtual]`

Definition at line 80 of file Dvbt1FormatterComponent.cpp.

```

83 {
84     outputTypes["output1"] = TypeInfo< int16_t >::identifier;
85 }
```

**7.5.4.3 void iris::phy::Dvbt1FormatterComponent::destroy ( ) [private]**

Definition at line 145 of file Dvbt1FormatterComponent.cpp.

Referenced by parameterHasChanged(), and ~Dvbt1FormatterComponent().

```
146 {
147 }
```

**7.5.4.4 template<typename T, size\_t N> static T\* iris::phy::Dvbt1FormatterComponent::end ( T(&) arr[N] ) [inline], [static], [private]**

Definition at line 84 of file Dvbt1FormatterComponent.h.

```
84 { return &arr[0]+N; }
```

**7.5.4.5 void iris::phy::Dvbt1FormatterComponent::initialize ( ) [virtual]**

Definition at line 87 of file Dvbt1FormatterComponent.cpp.

References setup().

```
88 {
89     setup();
90 }
```

**7.5.4.6 void iris::phy::Dvbt1FormatterComponent::parameterHasChanged ( std::string name ) [virtual]**

Definition at line 131 of file Dvbt1FormatterComponent.cpp.

References destroy(), and setup().

```
132 {
133     if(name == "???" )
134     {
135         destroy();
136         setup();
137     }
138 }
```

**7.5.4.7 void iris::phy::Dvbt1FormatterComponent::process ( ) [virtual]**

Definition at line 92 of file Dvbt1FormatterComponent.cpp.

References debug\_x.

```
93 {
94     DataSet< Cplx >* in = NULL;
95     getInputDataSet("input1", in);
96     int insize = in ? (int) in->data.size() : 0;
97     int outsize = 2 * insize;
98
99     if(debug_x)
100         LOG(LINFO) << "in/out: " << insize << "/" << outsize;
101
102     DataSet< int16_t >* out = NULL;
103     getOutputDataSet("output1", out, outsize);
104
105     // do the formatting
106     ShortVecIt outit = out->data.begin();
107     for(CplxVecIt init = in->data.begin(); init < in->data.end(); init++)
108     {
109         if(init->real() > 1.0)
```

```

110     *outit++ = 32767;
111     else if (init->real() < -1)
112         *outit++ = -32768;
113     else
114         *outit++ = (int16_t) (0.5 + init->real() * 32768.0);
115     if (init->imag() > 1.0)
116         *outit++ = 32767;
117     else if (init->imag() < -1)
118         *outit++ = -32768;
119     else
120         *outit++ = (int16_t) (0.5 + init->imag() * 32768.0);
121 }
122
123 //Copy the timestamp and sample rate for the DataSets
124 out->timeStamp = in->timeStamp;
125 out->sampleRate = in->sampleRate;
126
127 releaseInputDataSet("input1", in);
128 releaseOutputDataSet("output1", out);
129 }

```

#### 7.5.4.8 void iris::phy::Dvbt1FormatterComponent::registerPorts ( ) [virtual]

Definition at line 74 of file Dvbt1FormatterComponent.cpp.

```

75 {
76     registerInputPort("input1", TypeInfo< Cplx >::identifier);
77     registerOutputPort("output1", TypeInfo< int16_t >::identifier);
78 }

```

#### 7.5.4.9 void iris::phy::Dvbt1FormatterComponent::setup ( ) [private]

Set up all our index vectors and containers.

Definition at line 141 of file Dvbt1FormatterComponent.cpp.

Referenced by initialize(), and parameterHasChanged().

```

142 {
143 }

```

### 7.5.5 Member Data Documentation

#### 7.5.5.1 bool iris::phy::Dvbt1FormatterComponent::debug\_x [private]

Debug flag (default = false)

Definition at line 73 of file Dvbt1FormatterComponent.h.

Referenced by Dvbt1FormatterComponent(), and process().

#### 7.5.5.2 double iris::phy::Dvbt1FormatterComponent::sampleRate\_ [private]

Sample rate of current frame.

Definition at line 79 of file Dvbt1FormatterComponent.h.

#### 7.5.5.3 double iris::phy::Dvbt1FormatterComponent::timeStamp\_ [private]

Timestamp of current frame.

Definition at line 78 of file Dvbt1FormatterComponent.h.

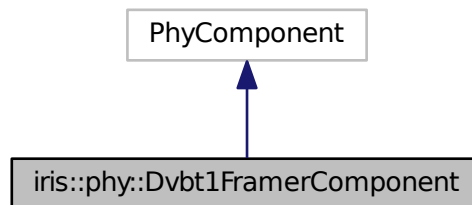


## 7.6 iris::phy::Dvbt1FramerComponent Class Reference

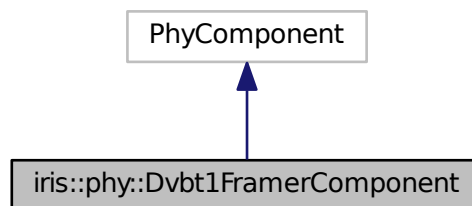
A DVB-T1 framer component.

```
#include <Dvbt1FramerComponent.h>
```

Inheritance diagram for iris::phy::Dvbt1FramerComponent:



Collaboration diagram for iris::phy::Dvbt1FramerComponent:



### Public Types

- typedef std::vector< uint8\_t > [ByteVec](#)  
*A vector of bytes.*
- typedef ByteVec::iterator [ByteVecIt](#)  
*An iterator for a vector of bytes.*
- typedef std::complex< float > [Cplx](#)  
*A complex type.*
- typedef std::vector< [Cplx](#) > [CplxVec](#)  
*A vector of complex.*
- typedef CplxVec::iterator [CplxVecIt](#)

### Public Member Functions

- [Dvbt1FramerComponent](#) (std::string name)

- *Default constructor.*
- `~Dvbt1FramerComponent ()`
- *Default destructor.*
- virtual void `calculateOutputTypes` (std::map< std::string, int > &inputTypes, std::map< std::string, int > &outputTypes)
- *Calculate the output port types for the IRIS system.*
- virtual void `registerPorts` ()
- *Register the mapper ports with the IRIS system.*
- virtual void `initialize` ()
- *Initialize the component.*
- virtual void `process` ()
- *Main processing method.*
- virtual void `parameterHasChanged` (std::string name)
- *Actions taken when the parameters change.*

### Private Member Functions

- void `setup` ()
- *Set up all needed constants.*
- void `destroy` ()
- *Destroy the component.*
- int `t1_tps_generate` (unsigned char \*tps, int block\_in\_frame, int frame\_in\_superframe)
- *This functions generates the modulated TPS carriers.*

### Static Private Member Functions

- template<typename T, size\_t N>  
static T \* `begin` (T(&arr)[N])
- *Useful templates.*
- template<typename T, size\_t N>  
static T \* `end` (T(&arr)[N])

### Private Attributes

- bool `debug_x`
- *Debug flag (default = false)*
- int `ofdmMode_x`
- *OFDM mode (default = 2048)*
- int `cellId_x`
- *Cell ID for DVB-H mode (default = -1)*
- int `qamMapping_x`
- *QAM constellation mapping (default = 16)*
- bool `inDepthInterleaver_x`
- *In-depth interleaver for DVB-H mode (default = false)*
- int `hierarchyMode_x`
- *Hyerarchical mode (default = 0)*
- int `hpCodeRate_x`
- *HP stream channel coding rate (default = 34)*
- int `lpCodeRate_x`
- *LP stream channel coding rate (default = 34)*

- int [deltaMode\\_x](#)  
*Cyclic prefix ratio (default = 32)*
- double [timeStamp\\_](#)  
*Timestamp of current frame.*
- double [sampleRate\\_](#)  
*Sample rate of current frame.*
- int [nMax\\_](#)  
*data carriers*
- int [kMax\\_](#)  
*active carriers*
- int [fraOffset\\_](#)  
*framer offset*
- [CplxVec](#) [fraRegister\\_](#)  
*framer template*
- int [blockIndex\\_](#)  
*OFDM block index.*
- float [tpsAmpl\\_](#) [6817]  
*tps\_amplitudes*
- uint8\_t [tps\\_](#) [T1\_BLOCKS\_PER\_FRAME]  
*tps data*

### Static Private Attributes

- static int [cont\\_pilot\\_position](#) [178]
- static int [tps\\_position](#) [69]
- static unsigned char [prbs\\_pilot](#) [6817]

#### 7.6.1 Detailed Description

A DVB-T1 framer component.

[Dvbt1FramerComponent](#) is the ninth block composing the DVB-T transmission chain. The framer has the task to assemble together QAM data cells, pilot data cells, and transmission parameters signaling (TPS) data cells into a frame structure that will be mapped onto OFDM symbols.

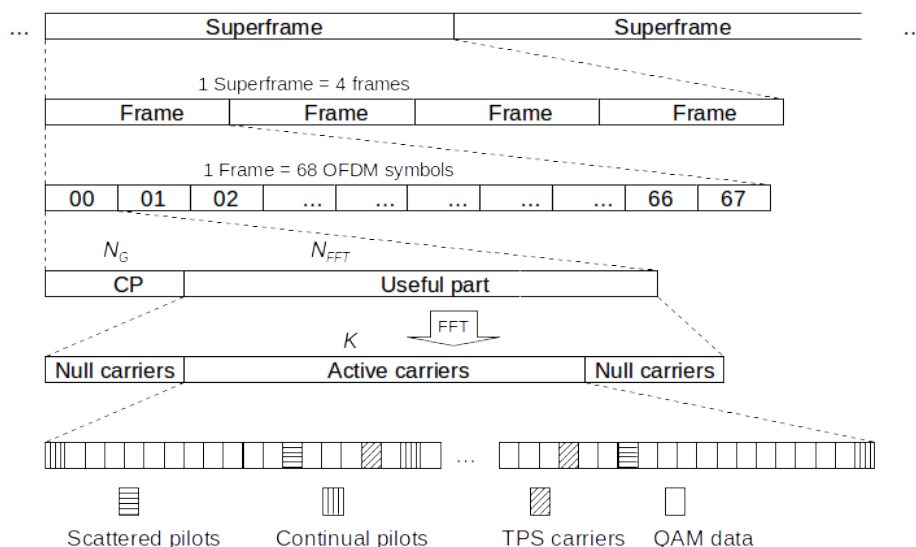


Figure 7.4: DVB-T framing structure.

The basic frame structure starts from the OFDM symbol: 68 OFDM symbols constitute one frame, and 4 frames build up a superframe. Each OFDM symbol is composed by an useful portion, which comes from an IFFT operation, and by a cyclic prefix (CP). The carriers of the useful portion are composed by active and null carriers, which are switched off and are guard bands.

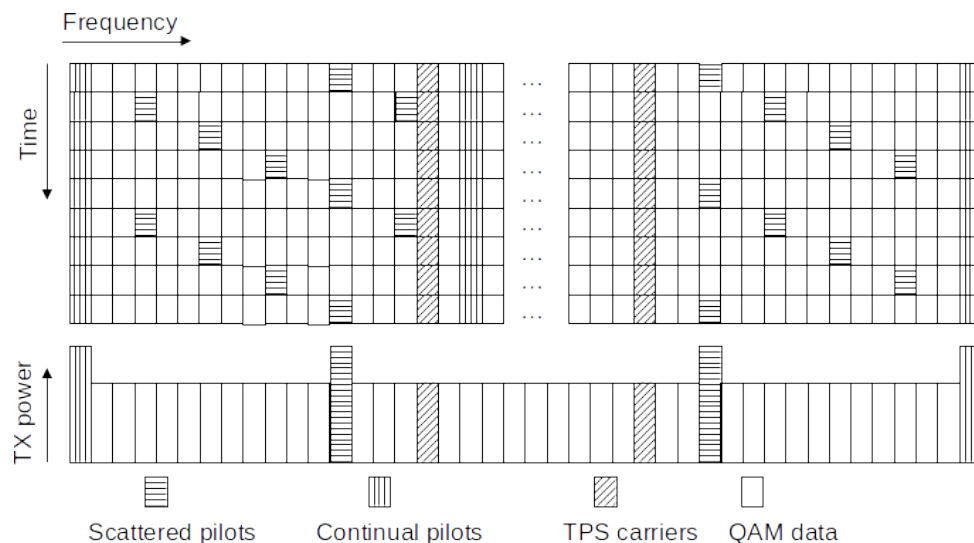


Figure 7.5: DVB-T pilots structure.

Pilot carriers are divided between continual pilots, which occur on every OFDM symbol at the same carrier position, and scattered pilots, which are cyclically shifted of three positions at each new OFDM symbol. Additionally, there are a number of carriers that are used to convey TPS data, useful for purposes of frame synchronization and signalling. As displayed in the figure above, the carriers are not created at the same power: while data and TPS carriers have a unitary power, all pilot carriers are transmitted at a power of 16/9.

This block accepts in input complex float values and generates in output complex float values. The block is capable to generate internally all the required frame timing and modulation for the pilot and TPS cells.

There are several parameters that can be changed in the XML configuration file:

- *debug*: by default set to "false", is used to print some small debugging information for the interested developer.
- *hpcoderate*: by default set to "34", this is used to select one of the five possible coding rates. The admitted values are "12", "23", "34", "56", and "78", which are easily recognizable as the real coding ratios written without the separating slash. This parameter refers to the high priority (HP) stream in case of hierarchical transmission, differently it refers to the coderate of the single stream for nonhierarchical transmission.
- *lpcoderate*: by default set to "34", this is used to select one of the five possible coding rates. The admitted values are "12", "23", "34", "56", and "78", which are easily recognizable as the real coding ratios written without the separating slash. This parameter refers to the low priority (LP) stream in case of hierarchical transmission, differently it is not used for nonhierarchical transmission.
- *qammapping*: by default set to "16", this is used to select one of the three possible QAM mappings. The admitted values are "4", "16", "64".
- *hierarchymode*: by default set to "0", which means "not hierarchical". Hierarchical modes are used to transmit two different transport streams, one with a high priority (HP) information and another one with a low priority (LP) information. The admitted values are "0", "1", "2", "4". NOTE: hierarchical modes are not implemented in the current release of this modulator.
- *ofdmmode*: by default set to "2048", this is used to select one of the three possible OFDM modes. The admitted values are "2048", "4096", "8192", respectively for 2K, 4K (DVB-H, unused), and 8K.

- *deltamode*: by default set to "32", this is used to select one of the four possible cyclic prefix lengths. The admitted values are "32", "16", "8", and "4", which are directly derived from the denominator of the cyclic prefix fraction (1/32, 1/16, 1/8, 1/4).
- *cellicd*: by default set to "-1", which means it is disabled. The Cell Identifier is used to identify transmission towers with a 16 bit numeric identifier, and is used only in case of DVB-H transmission. NOTE: DVB-H is not implemented in this software modulator.
- *indephtinterleaver*: by default set to "false", which means it is disabled. This additional interleaver is used only in DVB-H mode and should improve the diversity of the received signal in case of transmission over time varying channels. NOTE: DVB-H is not implemented in this software modulator.

## References

- ETSI Standard: *EN 300 744 V1.5.1, Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for digital terrestrial television*, available at [ETSI Publications Download Area](#)

Definition at line 132 of file Dvbt1FramerComponent.h.

## 7.6.2 Member Typedef Documentation

### 7.6.2.1 typedef std::vector<uint8\_t> iris::phy::Dvbt1FramerComponent::ByteVec

A vector of bytes.

Definition at line 138 of file Dvbt1FramerComponent.h.

### 7.6.2.2 typedef ByteVec::iterator iris::phy::Dvbt1FramerComponent::ByteVecIt

An iterator for a vector of bytes.

Definition at line 141 of file Dvbt1FramerComponent.h.

### 7.6.2.3 typedef std::complex<float> iris::phy::Dvbt1FramerComponent::Cplx

A complex type.

Definition at line 144 of file Dvbt1FramerComponent.h.

### 7.6.2.4 typedef std::vector<Cplx> iris::phy::Dvbt1FramerComponent::CplxVec

A vector of complex.

Definition at line 147 of file Dvbt1FramerComponent.h.

### 7.6.2.5 typedef CplxVec::iterator iris::phy::Dvbt1FramerComponent::CplxVecIt

Definition at line 150 of file Dvbt1FramerComponent.h.

## 7.6.3 Constructor & Destructor Documentation

### 7.6.3.1 iris::phy::Dvbt1FramerComponent::Dvbt1FramerComponent ( std::string name )

Default constructor.

Registers the block parameters and initializes some variables

Definition at line 57 of file Dvbt1FramerComponent.cpp.

References `begin()`, `cellId_x`, `debug_x`, `deltaMode_x`, `end()`, `hpCodeRate_x`, `hyerarchyMode_x`, `inDepthInterleaver_x`, `lpCodeRate_x`, `ofdmMode_x`, and `qamMapping_x`.

```

58 : PhyComponent(name,                                // component name
59               "dvbt1framer",                        // component type
60               "A DVB-T1 framer component",           // description
61               "Giuseppe Baruffa",                   // author
62               "0.1")                                 // version
63     ,sampleRate_(0)
64     ,timeStamp_(0)
65 {
66     registerParameter(
67         "debug", "Whether to output debug data",
68         "false", true, debug_x);
69
70     int codearr[] = {12,23,34,56,78};
71     registerParameter(
72         "hpcoderate", "HP stream channel coding rate",
73         "34", true, hpCodeRate_x, list<int>(begin(codearr),end(codearr)));
74
75     registerParameter(
76         "lpcoderate", "LP stream channel coding rate",
77         "34", true, lpCodeRate_x, list<int>(begin(codearr),end(codearr)));
78
79     int qamarr[] = {4,16,64};
80     registerParameter(
81         "qammapping", "QAM constellation mapping",
82         "16", true, qamMapping_x, list<int>(begin(qamarr),end(qamarr)));
83
84     int harr[] = {0,1,2,4};
85     registerParameter(
86         "hyerarchymode", "Hyerarchical mode (0 = NH)",
87         "0", true, hyerarchyMode_x, list<int>(begin(harr),end(harr)));
88
89     int ofdmarr[] = {2048,4096,8192};
90     registerParameter(
91         "ofdmmode", "OFDM mode",
92         "2048", true, ofdmMode_x, list<int>(begin(ofdmarr),end(ofdmarr)));
93
94     int deltaarr[] = {32,16,8,4};
95     registerParameter(
96         "deltamode", "Cyclic prefix ratio",
97         "32", true, deltaMode_x, list<int>(begin(deltaarr),end(deltaarr)));
98
99     registerParameter(
100        "cellid", "Cell ID for DVB-H mode",
101        "-1", true, cellId_x, Interval<int>(-1,65535));
102
103     registerParameter(
104         "indepthinterleaver", "In-depth interleaver for DVB-H mode",
105         "false", true, inDepthInterleaver_x);
106 }
```

### 7.6.3.2 `iris::phy::Dvbt1FramerComponent::~~Dvbt1FramerComponent ( )`

Default destructor.

Just calls `destroy()`.

Definition at line 111 of file Dvbt1FramerComponent.cpp.

References `destroy()`.

```

112 {
113     destroy();
114 }
```

## 7.6.4 Member Function Documentation

**7.6.4.1** `template<typename T, size_t N> static T* iris::phy::Dvbt1FramerComponent::begin ( T(&) arr[N] )` `[inline]`, `[static]`, `[private]`

Useful templates.

Definition at line 197 of file Dvbt1FramerComponent.h.

Referenced by Dvbt1FramerComponent().

```
197 { return &arr[0]; }
```

**7.6.4.2** void iris::phy::Dvbt1FramerComponent::calculateOutputTypes ( std::map< std::string, int > & *inputTypes*, std::map< std::string, int > & *outputTypes* ) [virtual]

Calculate the output port types for the IRIS system.

The single output port must provide complex values.

Definition at line 129 of file Dvbt1FramerComponent.cpp.

```
132 {
133     outputTypes["output1"] = TypeInfo< Cplx >::identifier;
134 }
```

**7.6.4.3** void iris::phy::Dvbt1FramerComponent::destroy ( ) [private]

Destroy the component.

Definition at line 943 of file Dvbt1FramerComponent.cpp.

Referenced by parameterHasChanged(), and ~Dvbt1FramerComponent().

```
944 {
945 }
```

**7.6.4.4** template<typename T, size\_t N> static T\* iris::phy::Dvbt1FramerComponent::end ( T(&) *arr[N]* ) [inline], [static], [private]

Definition at line 199 of file Dvbt1FramerComponent.h.

Referenced by Dvbt1FramerComponent().

```
199 { return &arr[0]+N; }
```

**7.6.4.5** void iris::phy::Dvbt1FramerComponent::initialize ( ) [virtual]

Initialize the component.

Just calls [setup\(\)](#).

Definition at line 139 of file Dvbt1FramerComponent.cpp.

References [setup\(\)](#).

```
140 {
141     setup();
142 }
```

#### 7.6.4.6 void iris::phy::Dvbt1FramerComponent::parameterHasChanged ( std::string name ) [virtual]

Actions taken when the parameters change.

This block has several significant parameters

Definition at line 904 of file Dvbt1FramerComponent.cpp.

References `destroy()`, and `setup()`.

```

905 {
906     if(name == "hpcoderate" || name == "deltamode" ||
907        name == "lpcoderate" || name == "gammapping" ||
908        name == "hyerarchymode" || name == "ofdmmode" ||
909        name == "cellid" || name == "indepthinterleaver")
910     {
911         destroy();
912         setup();
913     }
914 }
```

#### 7.6.4.7 void iris::phy::Dvbt1FramerComponent::process ( ) [virtual]

Main processing method.

Definition at line 783 of file Dvbt1FramerComponent.cpp.

References `blockIndex_`, `cont_pilot_position`, `debug_x`, `fraOffset_`, `fraRegister_`, `kMax_`, `nMax_`, `prbs_pilot`, `T1_BLOCKS_PER_FRAME`, `T1_FRAMES_PER_SUPERFRAME`, `T1_PIL_AMPL`, `T1_TPS_AMPL`, `t1_tps_generate()`, `tps_`, `tps_position`, and `tpsAmpl_`.

```

784 {
785     // request input
786     DataSet< Cplx > *in = NULL;
787     getInputDataSet("input1", in);
788
789     // calculate sizes
790     int insize = in ? (int) in->data.size() : 0;
791     int outsize = kMax_ * ((insize + fraOffset_) / nMax_);
792
793     // request output
794     DataSet< Cplx > *out = NULL;
795     getOutputDataSet("output1", out, outsize);
796
797     // print debug info
798     if(debug_x)
799         LOG(LINFO) << "in/out: " << insize << "/" << outsize;
800
801     // fill register
802     for(CplxVecIt init = in->data.begin(), outit = out->data.begin();
803        init < in->data.end(); init++)
804     {
805         // copy
806         fraRegister_[fraOffset_++] = *init;
807
808         // ready for new block - trigger
809         if(fraOffset_ == nMax_)
810         {
811             // reset offset
812             fraOffset_ = 0;
813
814             // initial position values
815             int scatt_pil_pos = 3 * (blockIndex_ & 0x03);
816             int cpp = 0;
817             int cont_pil_pos = cont_pilot_position[cpp];
818             int tp = 0;
819             int tps_pos = tps_position[tp];
820
821             // counters
822             int frameInSuperFrame = blockIndex_ / T1_BLOCKS_PER_FRAME;
823             int blockInFrame = blockIndex_ - frameInSuperFrame *
T1_BLOCKS_PER_FRAME;
824
825             // generate the tps information for this frame
826             if(blockInFrame == 0)
827             {
828                 int status = t1_tps_generate(tps_, blockInFrame, frameInSuperFrame);
829                 if(status)
```



```

830         LOG(LError) << "Error in TPS parity generation";
831     }
832
833     // populate the frame
834     CplxVecIt regit = fraRegister_.begin();
835     for(int k = 0; k < kMax_; k++, outit++)
836     {
837         if(k == scatt_pil_pos) {
838             // scattered pilot
839             outit->real(prbs_pilot[k] ? -Tl_PIL_AMPL :
Tl_PIL_AMPL);
840             outit->imag(0);
841             scatt_pil_pos += 12;
842             if (k == cont_pil_pos)
843             {
844                 // coincidence with continual pilot
845                 cont_pil_pos = cont_pilot_position[+cpp];
846             }
847         }
848         else if(k == cont_pil_pos)
849         {
850             // continual pilot
851             outit->real(prbs_pilot[k] ? -Tl_PIL_AMPL : Tl_PIL_AMPL);
852             outit->imag(0);
853             cont_pil_pos = cont_pilot_position[+cpp];
854         }
855         else if(k == tps_pos)
856         {
857             // TPS
858             // first symbol in frame: absolute reference for differential encoding
859             if (blockInFrame == 0)
860             {
861                 tpsAmpl_[k] = prbs_pilot[k] ? -
Tl_TPS_AMPL : Tl_TPS_AMPL;
862             }
863             else
864             {
865                 // subsequent symbols in frame
866                 // differentially encoded bits with respect to the first frame bits
867                 tpsAmpl_[k] = tps_[blockInFrame] ? -tpsAmpl_[k] :
tpsAmpl_[k];
868             }
869             outit->real(tpsAmpl_[k]);
870             outit->imag(0);
871
872             tps_pos = tps_position[+tp];
873         }
874         else
875         {
876             // real data
877             *outit = *regit++;
878         }
879     }
880
881     // advance block index
882     if(++blockIndex_ == Tl_BLOCKS_PER_FRAME *
Tl_FRAMES_PER_SUPERFRAME)
883     {
884         // reset
885         blockIndex_ = 0;
886     }
887
888     // Copy the timestamp and sample rate for the DataSets
889     out->timeStamp = in->timeStamp;
890     out->sampleRate = in->sampleRate;
891
892     // release input and output
893     releaseInputDataSet("input1", in);
894     releaseOutputDataSet("output1", out);
895 }

```

#### 7.6.4.8 void iris::phy::Dvbt1FramerComponent::registerPorts ( ) [virtual]

Register the mapper ports with the IRIS system.

This component has one input that accept complex float symbols and one output that provides complex float symbols.

Definition at line 120 of file Dvbt1FramerComponent.cpp.

```
121 {
122     registerInputPort("input1", TypeInfo< Cplx >::identifier);
123     registerOutputPort("output1", TypeInfo< Cplx >::identifier);
124 }
```

#### 7.6.4.9 void iris::phy::Dvbt1FramerComponent::setup ( ) [private]

Set up all needed constants.

Definition at line 917 of file Dvbt1FramerComponent.cpp.

References `blockIndex_`, `fraOffset_`, `fraRegister_`, `kMax_`, `nMax_`, `ofdmMode_x`, `tps_`, and `tpsAmpl_`.

Referenced by `initialize()`, and `parameterHasChanged()`.

```
918 {
919     // prepare
920     blockIndex_ = 0;
921     fraOffset_ = 0;
922     switch(ofdmMode_x)
923     {
924         case 2048:
925             kMax_ = 1705;
926             nMax_ = 1512;
927             break;
928         case 4096:
929             kMax_ = 3409;
930             nMax_ = 3024;
931             break;
932         case 8192:
933             kMax_ = 6817;
934             nMax_ = 6048;
935             break;
936     }
937     fraRegister_.resize(nMax_);
938     memset(tpsAmpl_, 0, sizeof tpsAmpl_);
939     memset(tps_, 0, sizeof tps_);
940 }
```

#### 7.6.4.10 int iris::phy::Dvbt1FramerComponent::t1\_tps\_generate ( unsigned char \* tps, int block\_in\_frame, int frame\_in\_superframe ) [private]

This functions generates the modulated TPS carriers.

##### Parameters

<i>tps</i>	Preallocated array that will contain the bit to be transmitted in the TPS carriers
<i>block_in_frame</i>	Index of the block in the current frame (0..67)
<i>frame_in_superframe</i>	Index of the frame in the current superframe (0..3)

##### Returns

0 if all went well, else errors happened

Definition at line 390 of file Dvbt1FramerComponent.cpp.

References `cellId_x`, `deltaMode_x`, `hpCodeRate_x`, `hyerarchyMode_x`, `inDepthInterleaver_x`, `lpCodeRate_x`, `ofdmMode_x`, `qamMapping_x`, `T1_K_BCH`, and `T1_N_BCH`.

Referenced by `process()`.

```
392 {
393     int i = 0, j = 0;
394     unsigned char feedback = 0;
395     unsigned char x[T1_K_BCH] = {0}, b[T1_N_BCH - T1_K_BCH] = {0};
```

```

396
397 // Code generator polynomial
398 static unsigned char g[] = {1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1};
399
400 // Predefined TPS bits
401
402 // Very first bit is given by PRBS and it is used
403 // as a reference in differential BPSK modulation
404 // The present value is set to zero, but it reall
405 // doesn't care
406 tps[0] = 0;
407
408 // Synchronization
409 switch(frame_in_superframe)
410 {
411     // Frame 1 and 3
412     case 0:
413     case 2:
414         tps[1] = 0;
415         tps[2] = 0;
416         tps[3] = 1;
417         tps[4] = 1;
418         tps[5] = 0;
419         tps[6] = 1;
420         tps[7] = 0;
421         tps[8] = 1;
422         tps[9] = 1;
423         tps[10] = 1;
424         tps[11] = 1;
425         tps[12] = 0;
426         tps[13] = 1;
427         tps[14] = 1;
428         tps[15] = 1;
429         tps[16] = 0;
430         break;
431     // Frame 2 and 4
432     case 1:
433     case 3:
434         tps[1] = 1;
435         tps[2] = 1;
436         tps[3] = 0;
437         tps[4] = 0;
438         tps[5] = 1;
439         tps[6] = 0;
440         tps[7] = 1;
441         tps[8] = 0;
442         tps[9] = 0;
443         tps[10] = 0;
444         tps[11] = 0;
445         tps[12] = 1;
446         tps[13] = 0;
447         tps[14] = 0;
448         tps[15] = 0;
449         tps[16] = 1;
450         break;
451 }
452
453 // TPS length indicator
454 // Full DVB-H option wants 33 bits
455 // NOT USED NOW !!!
456 if(cellId_x >= 0)
457 {
458     // Cell Id is set, 31 bits
459     tps[17] = 0;
460     tps[18] = 1;
461     tps[19] = 1;
462     tps[20] = 1;
463     tps[21] = 1;
464     tps[22] = 1;
465 }
466 else
467 {
468     // Cell Id is not set, 23 bits
469     tps[17] = 0;
470     tps[18] = 1;
471     tps[19] = 0;
472     tps[20] = 1;
473     tps[21] = 1;
474     tps[22] = 1;
475 }
476
477 // Variable TPS bits
478
479 // Counts the frame number in superframe
480 switch(frame_in_superframe)
481 {

```

```

483     case 0:
484         tps[23] = 0;
485         tps[24] = 0;
486         break;
487     case 1:
488         tps[23] = 0;
489         tps[24] = 1;
490         break;
491     case 2:
492         tps[23] = 1;
493         tps[24] = 0;
494         break;
495     case 3:
496         tps[23] = 1;
497         tps[24] = 1;
498     }
499
500     // Constellation
501     switch(qamMapping_x)
502     {
503         // QPSK
504         case 4:
505             tps[25] = 0;
506             tps[26] = 0;
507             break;
508         // 16-QAM
509         case 16:
510             tps[25] = 0;
511             tps[26] = 1;
512             break;
513         // 64-QAM
514         case 64:
515             tps[25] = 1;
516             tps[26] = 0;
517             break;
518     }
519
520     // In-depth inner interleaver
521     if(inDepthInterleaver_x == false)
522     {
523         // Native
524         tps[27] = 0;
525     }
526     else
527     {
528         tps[27] = 1;
529     }
530
531     // Hierarchy
532     switch(hierarchyMode_x)
533     {
534         // Not hierarchical
535         case 0:
536             tps[28] = 0;
537             tps[29] = 0;
538             break;
539         // Alpha = 1
540         case 1:
541             tps[28] = 0;
542             tps[29] = 1;
543             break;
544         // Alpha = 2
545         case 2:
546             tps[28] = 1;
547             tps[29] = 0;
548             break;
549         // Alpha = 4
550         case 4:
551             tps[28] = 1;
552             tps[29] = 1;
553             break;
554     }
555
556     // Code rate
557     if(hierarchyMode_x == 0)
558     {
559         // Code rate, NH
560         switch (hpCodeRate_x) {
561             case 12:
562                 tps[30] = 0;
563                 tps[31] = 0;
564                 tps[32] = 0;
565                 break;
566             case 23:
567                 tps[30] = 0;
568                 tps[31] = 0;
569                 tps[32] = 1;

```

```
570         break;
571     case 34:
572         tps[30] = 0;
573         tps[31] = 1;
574         tps[32] = 0;
575         break;
576     case 56:
577         tps[30] = 0;
578         tps[31] = 1;
579         tps[32] = 1;
580         break;
581     case 78:
582         tps[30] = 1;
583         tps[31] = 0;
584         tps[32] = 0;
585         break;
586     default:
587         break;
588     }
589 }
590 else
591 {
592     // Code rate, HP
593     switch (hpCodeRate_x)
594     {
595     case 12:
596         tps[30] = 0;
597         tps[31] = 0;
598         tps[32] = 0;
599         break;
600     case 23:
601         tps[30] = 0;
602         tps[31] = 0;
603         tps[32] = 1;
604         break;
605     case 34:
606         tps[30] = 0;
607         tps[31] = 1;
608         tps[32] = 0;
609         break;
610     case 56:
611         tps[30] = 0;
612         tps[31] = 1;
613         tps[32] = 1;
614         break;
615     case 78:
616         tps[30] = 1;
617         tps[31] = 0;
618         tps[32] = 0;
619         break;
620     default:
621         break;
622     }
623
624     // Code rate, LP
625     switch (lpCodeRate_x)
626     {
627     case 12:
628         tps[33] = 0;
629         tps[34] = 0;
630         tps[35] = 0;
631         break;
632     case 23:
633         tps[33] = 0;
634         tps[34] = 0;
635         tps[35] = 1;
636         break;
637     case 34:
638         tps[33] = 0;
639         tps[34] = 1;
640         tps[35] = 0;
641         break;
642     case 56:
643         tps[33] = 0;
644         tps[34] = 1;
645         tps[35] = 1;
646         break;
647     case 78:
648         tps[33] = 1;
649         tps[34] = 0;
650         tps[35] = 0;
651         break;
652     default:
653         break;
654     }
655 }
656
```

```

657     // Guard interval
658     switch(deltaMode_x)
659     {
660         case 32:
661             tps[36] = 0;
662             tps[37] = 0;
663             break;
664         case 16:
665             tps[36] = 0;
666             tps[37] = 1;
667             break;
668         case 8:
669             tps[36] = 1;
670             tps[37] = 0;
671             break;
672         case 4:
673             tps[36] = 1;
674             tps[37] = 1;
675             break;
676     }
677
678     // Transmission mode
679     switch (ofdmMode_x)
680     {
681         case 2048:
682             tps[38] = 0;
683             tps[39] = 0;
684             break;
685         case 8192:
686             tps[38] = 0;
687             tps[39] = 1;
688             break;
689         case 4096:
690             tps[38] = 1;
691             tps[39] = 0;
692             break;
693     }
694
695     // Cell identification bits
696     if(cellId_x >= 0)
697     {
698         // Compute Cell Id bits
699         unsigned char cidv[16];
700         for(i = 0; i < 16; i++)
701             cidv[i] = (unsigned char) (((unsigned short int) cellId_x) >> i) & 0x0001;
702
703         switch(frame_in_superframe)
704         {
705             // First half in frames 1 and 3
706             case (0):
707             case (2):
708                 for (i = 40; i <= 47; i++)
709                     tps[i] = cidv[55 - i];
710                 break;
711
712             // Second half in frames 2 and 4
713             case (1):
714             case (3):
715                 for (i = 40; i <= 47; i++)
716                     tps[i] = cidv[47 - i];
717                 break;
718         }
719     }
720     else
721     {
722         // Cell Id not set
723         for(i = 40; i <= 47; i++)
724             tps[i] = 0;
725     }
726
727     // Bits 48 and 49 are used in DVB-H
728     // Not set currently
729     tps[48] = 0;
730     tps[49] = 0;
731
732     // Bits from 50 to 53 are all set to zero
733     tps[50] = 0;
734     tps[51] = 0;
735     tps[52] = 0;
736     tps[53] = 0;
737
738     // BCH encoding
739
740     // Empty the parity register
741     for(i = 0; i < (T1_N_BCH - T1_K_BCH); i++)
742         b[i] = 0;
743

```

```

744 // Reverse copy data into x, considering the shortening zeroes
745 for(i = 53; i > 0; i--)
746     x[53 - i] = tps[i];
747
748 // Compute redundancy bb[], the coefficients of b(x). The redundancy
749 // polynomial b(x) is the remainder after dividing x^(n-k)*data(x)
750 // by the generator polynomial g(x).
751 for(i = (Tl_K_BCH - 1); i >= 0; i--)
752 {
753     feedback = x[i] ^ b[Tl_N_BCH - Tl_K_BCH - 1];
754     if(feedback != 0)
755     {
756         for(j = (Tl_N_BCH - Tl_K_BCH - 1); j > 0; j--)
757             if (g[j] != 0)
758                 b[j] = b[j - 1] ^ feedback;
759             else
760                 b[j] = b[j - 1];
761
762         b[0] = g[0] && feedback;
763     }
764     else
765     {
766         for(j = (Tl_N_BCH - Tl_K_BCH - 1); j > 0; j--)
767             b[j] = b[j - 1];
768         b[0] = 0;
769     }
770 }
771
772 /* Back copy parity bits into s */
773 for(i = 0; i < 14; i++)
774     tps[54 + i] = b[13 - i];
775
776 return 0;
777 }

```

## 7.6.5 Member Data Documentation

### 7.6.5.1 int iris::phy::Dvbt1FramerComponent::blockIndex\_ [private]

OFDM block index.

Definition at line 185 of file Dvbt1FramerComponent.h.

Referenced by process(), and setup().

### 7.6.5.2 int iris::phy::Dvbt1FramerComponent::cellId\_x [private]

Cell ID for DVB-H mode (default = -1)

Definition at line 166 of file Dvbt1FramerComponent.h.

Referenced by Dvbt1FramerComponent(), and t1\_tps\_generate().

### 7.6.5.3 int iris::phy::Dvbt1FramerComponent::cont\_pilot\_position [static], [private]

**Initial value:**

```

= {
    0,    48,    54,    87,    141,    156,    192,    201,    255,    279,    282,    333,    432,    450,    483,
    525,    531,    618,    636,    714,    759,    765,    780,    804,    873,    888,    918,    939,    942,    969,
    984,    1050,    1101,    1107,    1110,    1137,    1140,    1146,    1206,    1269,    1323,    1377,    1491,    1683,    1704,
    1752,    1758,    1791,    1845,    1860,    1896,    1905,    1959,    1983,    1986,    2037,    2136,    2154,    2187,    2229,
    2235,    2322,    2340,    2418,    2463,    2469,    2484,    2508,    2577,    2592,    2622,    2643,    2646,    2673,    2688,
    2754,    2805,    2811,    2814,    2841,    2844,    2850,    2910,    2973,    3027,    3081,    3195,    3387,    3408,    3456,
    3462,    3495,    3549,    3564,    3600,    3609,    3663,    3687,    3690,    3741,    3840,    3858,    3891,    3933,    3939,
    4026,    4044,    4122,    4167,    4173,    4188,    4212,    4281,    4296,    4326,    4347,    4350,    4377,    4392,    4458,
    4509,    4515,    4518,    4545,    4548,    4554,    4614,    4677,    4731,    4785,    4899,    5091,    5112,    5160,    5166,
    5199,    5253,    5268,    5304,    5313,    5367,    5391,    5394,    5445,    5544,    5562,    5595,    5637,    5643,    5730,
    5748,    5826,    5871,    5877,    5892,    5916,    5985,    6000,    6030,    6051,    6054,    6081,    6096,    6162,    6213,
    6219,    6222,    6249,    6252,    6258,    6318,    6381,    6435,    6489,    6603,    6795,    6816,    -1
}

```

Definition at line 189 of file Dvbt1FramerComponent.h.

Referenced by process().

#### 7.6.5.4 `bool iris::phy::Dvbt1FramerComponent::debug_x` [private]

Debug flag (default = false)

Definition at line 164 of file Dvbt1FramerComponent.h.

Referenced by Dvbt1FramerComponent(), and process().

#### 7.6.5.5 `int iris::phy::Dvbt1FramerComponent::deltaMode_x` [private]

Cyclic prefix ratio (default = 32)

Definition at line 172 of file Dvbt1FramerComponent.h.

Referenced by Dvbt1FramerComponent(), and t1\_tps\_generate().

#### 7.6.5.6 `int iris::phy::Dvbt1FramerComponent::fraOffset_` [private]

framer offset

Definition at line 182 of file Dvbt1FramerComponent.h.

Referenced by process(), and setup().

#### 7.6.5.7 `CplxVec iris::phy::Dvbt1FramerComponent::fraRegister_` [private]

framer template

Definition at line 183 of file Dvbt1FramerComponent.h.

Referenced by process(), and setup().

#### 7.6.5.8 `int iris::phy::Dvbt1FramerComponent::hpCodeRate_x` [private]

HP stream channel coding rate (default = 34)

Definition at line 170 of file Dvbt1FramerComponent.h.

Referenced by Dvbt1FramerComponent(), and t1\_tps\_generate().

#### 7.6.5.9 `int iris::phy::Dvbt1FramerComponent::hierarchyMode_x` [private]

Hierarchical mode (default = 0)

Definition at line 169 of file Dvbt1FramerComponent.h.

Referenced by Dvbt1FramerComponent(), and t1\_tps\_generate().

#### 7.6.5.10 `bool iris::phy::Dvbt1FramerComponent::inDepthInterleaver_x` [private]

In-depth interleaver for DVB-H mode (default = false)

Definition at line 168 of file Dvbt1FramerComponent.h.

Referenced by Dvbt1FramerComponent(), and t1\_tps\_generate().



**7.6.5.11** `int iris::phy::Dvbt1FramerComponent::kMax_` `[private]`

active carriers

Definition at line 181 of file Dvbt1FramerComponent.h.

Referenced by `process()`, and `setup()`.

**7.6.5.12** `int iris::phy::Dvbt1FramerComponent::lpCodeRate_x` `[private]`

LP stream channel coding rate (default = 34)

Definition at line 171 of file Dvbt1FramerComponent.h.

Referenced by `Dvbt1FramerComponent()`, and `t1_tps_generate()`.

**7.6.5.13** `int iris::phy::Dvbt1FramerComponent::nMax_` `[private]`

data carriers

Definition at line 180 of file Dvbt1FramerComponent.h.

Referenced by `process()`, and `setup()`.

**7.6.5.14** `int iris::phy::Dvbt1FramerComponent::ofdmMode_x` `[private]`

OFDM mode (default = 2048)

Definition at line 165 of file Dvbt1FramerComponent.h.

Referenced by `Dvbt1FramerComponent()`, `setup()`, and `t1_tps_generate()`.

**7.6.5.15** `unsigned char iris::phy::Dvbt1FramerComponent::prbs_pilot` `[static], [private]`

Definition at line 191 of file Dvbt1FramerComponent.h.

Referenced by `process()`.

**7.6.5.16** `int iris::phy::Dvbt1FramerComponent::qamMapping_x` `[private]`

QAM constellation mapping (default = 16)

Definition at line 167 of file Dvbt1FramerComponent.h.

Referenced by `Dvbt1FramerComponent()`, and `t1_tps_generate()`.

**7.6.5.17** `double iris::phy::Dvbt1FramerComponent::sampleRate_` `[private]`

Sample rate of current frame.

Definition at line 178 of file Dvbt1FramerComponent.h.

**7.6.5.18** `double iris::phy::Dvbt1FramerComponent::timeStamp_` `[private]`

Timestamp of current frame.

Definition at line 177 of file Dvbt1FramerComponent.h.

#### 7.6.5.19 `uint8_t iris::phy::Dvbt1FramerComponent::tps_[T1_BLOCKS_PER_FRAME]` [private]

tps data

Definition at line 187 of file `Dvbt1FramerComponent.h`.

Referenced by `process()`, and `setup()`.

#### 7.6.5.20 `int iris::phy::Dvbt1FramerComponent::tps_position` [static],[private]

**Initial value:**

```
= {
    34,  50,  209,  346,  413,  569,  595,  688,  790,  901, 1073, 1219, 1262, 1286, 1469,
 1594, 1687, 1738, 1754, 1913, 2050, 2117, 2273, 2299, 2392, 2494, 2605, 2777, 2923, 2966,
 2990, 3173, 3298, 3391, 3442, 3458, 3617, 3754, 3821, 3977, 4003, 4096, 4198, 4309, 4481,
 4627, 4670, 4694, 4877, 5002, 5095, 5146, 5162, 5321, 5458, 5525, 5681, 5707, 5800, 5902,
 6013, 6185, 6331, 6374, 6398, 6581, 6706, 6799, -1
}
```

Definition at line 190 of file `Dvbt1FramerComponent.h`.

Referenced by `process()`.

#### 7.6.5.21 `float iris::phy::Dvbt1FramerComponent::tpsAmpl_[6817]` [private]

tps\_amplitudes

Definition at line 186 of file `Dvbt1FramerComponent.h`.

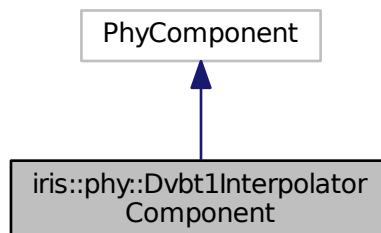
Referenced by `process()`, and `setup()`.

## 7.7 `iris::phy::Dvbt1InterpolatorComponent` Class Reference

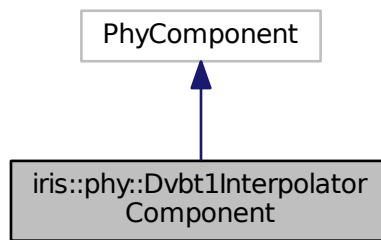
A DVB-T1 interpolator component.

```
#include <Dvbt1InterpolatorComponent.h>
```

Inheritance diagram for `iris::phy::Dvbt1InterpolatorComponent`:



Collaboration diagram for iris::phy::Dvbt1InterpolatorComponent:



## Public Types

- `typedef std::vector< uint8_t > ByteVec`  
A vector of bytes.
- `typedef ByteVec::iterator ByteVecIt`  
An iterator for a vector of bytes.
- `typedef std::complex< float > Cplx`  
A complex type.
- `typedef std::vector< Cplx > CplxVec`  
A vector of complex.
- `typedef CplxVec::iterator CplxVecIt`
- `typedef std::vector< float > FloatVec`  
A vector of float.
- `typedef FloatVec::iterator FloatVecIt`  
An iterator for a vector of float.
- `typedef std::vector< int > IntVec`  
A vector of integers.
- `typedef IntVec::iterator IntVecIt`  
An iterator for a vector of typedef.

## Public Member Functions

- `Dvbt1InterpolatorComponent` (std::string name)  
Default constructor.
- `~Dvbt1InterpolatorComponent` ()  
Default destructor.
- virtual void `calculateOutputTypes` (std::map< std::string, int > &inputTypes, std::map< std::string, int > &outputTypes)  
Calculate the output port types for the IRIS system.
- virtual void `registerPorts` ()  
Register the mapper ports with the IRIS system.
- virtual void `initialize` ()  
Initialize the component.
- virtual void `process` ()

*Main processing method.*

- virtual void [parameterHasChanged](#) (std::string name)  
*Actions taken when the parameters change.*

## Private Member Functions

- void [setup](#) ()  
*Set up all our index vectors and containers.*
- void [destroy](#) ()  
*Destroy the component.*
- int [time\\_buffer\\_size](#) (int input\_samples)  
*size correctly the interpolation buffers*
- int [find\\_rational\\_approximation](#) (int \*num, int \*den, double x, int N)  
*find a rational approximation of a real value*
- double \* [blackman\\_sinc](#) (int \*n\_order, double T, double dt, int order)  
*Calculate a Blackman-windowed sinc.*
- double [interp\\_response](#) (double \*h, int n, double dt, double t)  
*interpolate a base response*
- double [sinc](#) (double x)  
*sin(x)/x function*

## Static Private Member Functions

- template<typename T , size\_t N>  
static T \* [begin](#) (T(&arr)[N])  
*Useful templates.*
- template<typename T , size\_t N>  
static T \* [end](#) (T(&arr)[N])

## Private Attributes

- bool [debug\\_x](#)  
*Debug flag (default = false)*
- double [inSampleRate\\_x](#)  
*Input sampling rate (default = 0)*
- double [outSampleRate\\_x](#)  
*Output sampling rate (default = 0)*
- std::string [responseFile\\_x](#)  
*Text file with impulse response (default = none)*
- double [timeStamp\\_](#)  
*Timestamp of current frame.*
- double [sampleRate\\_](#)  
*Sample rate of current frame.*
- int [tiInsize\\_](#)
- int [tiOutsize\\_](#)
- int [inOffset\\_](#)
- CplxVec [inReg\\_](#)
- int [inLength\\_](#)
- IntVec [tiBasepointIndex\\_](#)
- FloatVec [tiHI\\_](#)

### 7.7.1 Detailed Description

A DVB-T1 interpolator component.

[Dvbt1InterpolatorComponent](#) is the first optional block composing the DVB-T transmission chain. It is required only if the analog conversion module following in the transmission chain has a rate different than that of the natural DVB-T sampling rate (64/7 MHz).

The conversion between the input DVB-T sampling rate and the output sampling rate is performed via a very simple sinc-shaped interpolator. The memory of the interpolating response should be kept short, in order to achieve the best processing speed. Clearly, this block distorts the original signal spectrum, and proper actions should be taken to override this detrimental effect. If you have used the DVB-T OFDM modulator previously on the transmission chain, then this effect has already been taken into account and the generated signal spectrum has been linearly predistorted in order to compensate for the distortion that is generated by the interpolator block.

This block accepts in input complex float values and generates in output complex float values.

There are parameters several that can be changed in the XML configuration file:

- *debug*: by default set to "false", is used to print some small debugging information for the interested developer.
- *insamplerate*: by default set to "0", a placeholder for 64e6/7 Hz. This represents the sampling rate of the entering signal. **Please note that if you are using the Dvbt1OFDM block, then you need to leave this parameter at 0.**
- *outsamplerate*: by default set to "0", a placeholder for 64e6/7 Hz. This represents the sampling rate adopted by the DAC for emitting the BB analog signal
- *responsefile*: by default set to "", which means not enabled, this is the name of a text file where the impulse response of the interpolating filter is saved, line after line.

### References

- ETSI Standard: *EN 300 744 V1.5.1, Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for digital terrestrial television*, available at [ETSI Publications Download Area](#)

Definition at line 90 of file Dvbt1InterpolatorComponent.h.

### 7.7.2 Member Typedef Documentation

#### 7.7.2.1 typedef std::vector<uint8\_t> iris::phy::Dvbt1InterpolatorComponent::ByteVec

A vector of bytes.

Definition at line 96 of file Dvbt1InterpolatorComponent.h.

#### 7.7.2.2 typedef ByteVec::iterator iris::phy::Dvbt1InterpolatorComponent::ByteVecIt

An iterator for a vector of bytes.

Definition at line 99 of file Dvbt1InterpolatorComponent.h.

#### 7.7.2.3 typedef std::complex<float> iris::phy::Dvbt1InterpolatorComponent::Cplx

A complex type.

Definition at line 102 of file Dvbt1InterpolatorComponent.h.

#### 7.7.2.4 `typedef std::vector<Cplx> iris::phy::Dvbt1InterpolatorComponent::CplxVec`

A vector of complex.

Definition at line 105 of file Dvbt1InterpolatorComponent.h.

#### 7.7.2.5 `typedef CplxVec::iterator iris::phy::Dvbt1InterpolatorComponent::CplxVecIt`

Definition at line 108 of file Dvbt1InterpolatorComponent.h.

#### 7.7.2.6 `typedef std::vector<float> iris::phy::Dvbt1InterpolatorComponent::FloatVec`

A vector of float.

Definition at line 111 of file Dvbt1InterpolatorComponent.h.

#### 7.7.2.7 `typedef FloatVec::iterator iris::phy::Dvbt1InterpolatorComponent::FloatVecIt`

An iterator for a vector of float.

Definition at line 114 of file Dvbt1InterpolatorComponent.h.

#### 7.7.2.8 `typedef std::vector<int> iris::phy::Dvbt1InterpolatorComponent::IntVec`

A vector of integers.

Definition at line 117 of file Dvbt1InterpolatorComponent.h.

#### 7.7.2.9 `typedef IntVec::iterator iris::phy::Dvbt1InterpolatorComponent::IntVecIt`

An iterator for a vector of typedef.

Definition at line 120 of file Dvbt1InterpolatorComponent.h.

### 7.7.3 Constructor & Destructor Documentation

#### 7.7.3.1 `iris::phy::Dvbt1InterpolatorComponent::Dvbt1InterpolatorComponent ( std::string name )`

Default constructor.

Registers the block parameters and initializes some variables

Definition at line 57 of file Dvbt1InterpolatorComponent.cpp.

References `debug_x`, `inSampleRate_x`, `outSampleRate_x`, and `responseFile_x`.

```

58 : PhyComponent(name,                                     // component name
59               "dvbt1interpolator",                       // component type
60               "A DVB-T1 OFDM interpolator component",    // description
61               "Giuseppe Baruffa",                       // author
62               "0.1")                                     // version
63 , sampleRate_(0)
64 , timeStamp_(0)
65 {
66     registerParameter(
67         "debug", "Whether to output debug data",
68         "false", true, debug_x);
69
70     registerParameter(
71         "insamplerate", "Input sampling rate (use 0 for 9142857)",
72         "0.0", true, inSampleRate_x, Interval<double>(0.0,15000000.0));
73
74     registerParameter(

```

```

75     "outsamplerate", "Output sampling rate (use 0 for 9142857)",
76     "0.0", true, outSampleRate_x, Interval<double>(0.0,15000000.0));
77
78     registerParameter(
79         "responsefile", "Text file with the interpolating impulse response",
80         "", true, responseFile_x);
81 }

```

### 7.7.3.2 iris::phy::Dvbt1InterpolatorComponent::~~Dvbt1InterpolatorComponent ( )

Default destructor.

Just calls [destroy\(\)](#).

Definition at line 86 of file Dvbt1InterpolatorComponent.cpp.

References [destroy\(\)](#).

```

87 {
88     destroy\(\);
89 }

```

## 7.7.4 Member Function Documentation

### 7.7.4.1 template<typename T, size\_t N> static T\* iris::phy::Dvbt1InterpolatorComponent::begin ( T(&) arr[N] ) [inline], [static], [private]

Useful templates.

Definition at line 161 of file Dvbt1InterpolatorComponent.h.

```

161 { return &arr[0]; }

```

### 7.7.4.2 double \* iris::phy::Dvbt1InterpolatorComponent::blackman\_sinc ( int \* n\_order, double T, double dt, int order ) [private]

Calculate a Blackman-windowed sinc.

Parameters

<i>n_order</i>	order of the calculated window
<i>T</i>	time extension of the window
<i>dt</i>	sampling time
<i>order</i>	preferred order of the window

Returns

array containing the window taps, please remember to free when this is not needed anymore

Definition at line 272 of file Dvbt1InterpolatorComponent.cpp.

References [sinc\(\)](#).

Referenced by [setup\(\)](#).

```

273 {
274     int n0 = (int) floor(T / dt);
275     int i;
276     double *h_order = NULL, w = 0.0;
277     double a0 = 7938.0 / 18608.0, a1 = 9240.0 / 18608.0, a2 = 1430.0 / 18608.0;
278     double accum = 0.0;
279     *n_order = (order + 1) * n0;
280     h_order = (double *) calloc(*n_order, sizeof(double));
281     for (i = 0; i < *n_order; i++) {

```

```

282         w = a0 - a1 * cos(2.0 * M_PI * i / (*n_order - 1)) + a2 * cos(4.0 * M_PI * i / (*n_order - 1));
283         h_order[i] = w * sinc(M_PI * (i - *n_order / 2) * dt / T);
284         accum += h_order[i] * dt;
285     }
286     /*for (i = 0; i < *n_order; i++)
287         h_order[i] /= accum;*/
288
289     return h_order;
290 }

```

**7.7.4.3** `void iris::phy::Dvbt1InterpolatorComponent::calculateOutputTypes ( std::map< std::string, int > & inputTypes, std::map< std::string, int > & outputTypes ) [virtual]`

Calculate the output port types for the IRIS system.

The single output port must provide complex values.

Definition at line 104 of file Dvbt1InterpolatorComponent.cpp.

```

107 {
108     outputTypes["output1"] = TypeInfo< Cplx >::identifier;
109 }

```

**7.7.4.4** `void iris::phy::Dvbt1InterpolatorComponent::destroy ( ) [private]`

Destroy the component.

Definition at line 374 of file Dvbt1InterpolatorComponent.cpp.

Referenced by parameterHasChanged(), and ~Dvbt1InterpolatorComponent().

```

375 {
376 }

```

**7.7.4.5** `template<typename T, size_t N> static T* iris::phy::Dvbt1InterpolatorComponent::end ( T(& arr[N]) ) [inline], [static], [private]`

Definition at line 163 of file Dvbt1InterpolatorComponent.h.

```

163 { return &arr[0]+N; }

```

**7.7.4.6** `int iris::phy::Dvbt1InterpolatorComponent::find_rational_approximation ( int * num, int * den, double x, int N ) [private]`

find a rational approximation of a real value

#### Parameters

<i>x</i>	Input value to be approximated as ratio of integers
<i>N</i>	maximum value for the integer at the denominator
<i>num</i>	the integer at the numerator
<i>den</i>	the integer at the denominator

#### Returns

0 for no errors

Definition at line 202 of file Dvbt1InterpolatorComponent.cpp.

Referenced by time\_buffer\_size().



```

203 {
204     int a = 0, b = 1;
205     int c = 1, d = 0;
206     while (b <= N && d <= N) {
207         double mediant = (double) (a + c) / (double) (b + d);
208         if (x == mediant) {
209             if (b + d <= N) {
210                 *num = a + c;
211                 *den = b + d;
212                 return 0;
213             } else if (d > b) {
214                 *num = c;
215                 *den = d;
216                 return 0;
217             } else {
218                 *num = a;
219                 *den = b;
220                 return 0;
221             }
222         } else if (x > mediant) {
223             a += c;
224             b += d;
225         } else {
226             c += a;
227             d += b;
228         }
229     }
230
231     if (b > N) {
232         *num = c;
233         *den = d;
234     } else {
235         *num = a;
236         *den = b;
237     }
238
239     return 0;
240 }

```

#### 7.7.4.7 void iris::phy::Dvbt1InterpolatorComponent::initialize ( ) [virtual]

Initialize the component.

Just calls [setup\(\)](#).

Definition at line 114 of file Dvbt1InterpolatorComponent.cpp.

References [setup\(\)](#).

```

115 {
116     setup();
117 }

```

#### 7.7.4.8 double iris::phy::Dvbt1InterpolatorComponent::interp\_response ( double \* *h*, int *n*, double *dt*, double *t* ) [private]

interpolate a base response

**Parameters**

<i>h</i>	array of the taps of the base impulse response
<i>n</i>	number of taps of the base impulse response
<i>dt</i>	sampling time of the base impulse response
<i>t</i>	time at which to interpolate the base response

**Returns**

the value of the base response linearly interpolated at the requested time

Definition at line 309 of file Dvbt1InterpolatorComponent.cpp.

Referenced by [setup\(\)](#).

```

310 {
311     if (t < 0.0)
312         return 0.0;
313     else if (t >= n * dt)
314         return 0.0;
315     else {
316         int n0 = (int) floor(t / dt);
317         double h0 = h[n0];
318         double h1 = n0 == (n - 1) ? 0.0 : h[n0 + 1];
319         return h0 + ((h1 - h0) / dt) * (t - n0 * dt);
320     }
321 }

```

#### 7.7.4.9 void iris::phy::Dvbt1InterpolatorComponent::parameterHasChanged ( std::string name ) [virtual]

Actions taken when the parameters change.

This block has several significant parameters

Definition at line 186 of file Dvbt1InterpolatorComponent.cpp.

References `destroy()`, and `setup()`.

```

187 {
188     if(name == "insamplerate" || name == "outsamplerate")
189     {
190         destroy();
191         setup();
192     }
193 }

```

#### 7.7.4.10 void iris::phy::Dvbt1InterpolatorComponent::process ( ) [virtual]

Main processing method.

Definition at line 120 of file Dvbt1InterpolatorComponent.cpp.

References `debug_x`, `inLength_`, `inOffset_`, `inReg_`, `T1_RESAMPLE_ORDER`, `tiBasepointIndex_`, `tiHl_`, `tiInsize_`, and `tiOutsize_`.

```

121 {
122     // request input
123     DataSet< Cplx > *in = NULL;
124     getInputDataSet("input1", in);
125
126     // calculate sizes
127     int insize = in ? (int) in->data.size() : 0;
128     int numbufs = (insize + inOffset_) / tiInsize_;
129     int outsize = tiOutsize_ * numbufs;
130
131     // print debug info
132     if(debug_x)
133         LOG(LINFO) << "in/out: " << insize << "/" << outsize;
134
135     // request output
136     DataSet< Cplx > *out = NULL;
137     getOutputDataSet("output1", out, outsize);
138
139     // copy
140     for(CplxVecIt init = in->data.begin(), outit = out->data.begin(),
141         inRegEff_ = inReg_.begin() + T1_RESAMPLE_ORDER + 1; init < in->data.end();
142         init++)
143     {
144         // copy
145         inRegEff_[inOffset_++] = *init;
146
147         // do the trick
148         if(inOffset_ == inLength_)
149         {
150             // reset
151             inOffset_ = 0;
152
153             // fractional filter
154             for(int j = 0; j < tiOutsize_; j++, outit++)
155             {

```

```

156         // current base point
157         int currbp = tiBasepointIndex_[j];
158
159         // interpolate
160         Cplx temp(0,0);
161         for(int k = 0; k < T1_RESAMPLE_ORDER + 1; k++)
162         {
163             temp.real(temp.real() + inRegEff_[currbp - k].real() * tiHI_[k * tiOutsize_ + j]);
164             temp.imag(temp.imag() + inRegEff_[currbp - k].imag() * tiHI_[k * tiOutsize_ + j]);
165         }
166         *outit = temp;
167     }
168
169     // copy last values at the beginning
170     copy(inReg_.end() - (T1_RESAMPLE_ORDER + 1), inReg_.end(),
inReg_.begin());
171 }
172 }
173
174 //Copy the timestamp and sample rate for the DataSets
175 out->timeStamp = in->timeStamp;
176 out->sampleRate = in->sampleRate; // not sure about this: it should change!
177
178 // release input and output
179 releaseOutputDataSet("output1", out);
180 releaseInputDataSet("input1", in);
181 }

```

#### 7.7.4.11 void iris::phy::Dvbt1InterpolatorComponent::registerPorts ( ) [virtual]

Register the mapper ports with the IRIS system.

This component has one input that accept complex float values and one output that provides complex float values.

Definition at line 95 of file Dvbt1InterpolatorComponent.cpp.

```

96 {
97     registerInputPort("input1", TypeInfo< Cplx >::identifier);
98     registerOutputPort("output1", TypeInfo< Cplx >::identifier);
99 }

```

#### 7.7.4.12 void iris::phy::Dvbt1InterpolatorComponent::setup ( ) [private]

Set up all our index vectors and containers.

Definition at line 324 of file Dvbt1InterpolatorComponent.cpp.

References blackman\_sinc(), inLength\_, inOffset\_, inReg\_, inSampleRate\_x, interp\_response(), outSampleRate\_x, responseFile\_x, T1\_RESAMPLE\_ORDER, tiBasepointIndex\_, tiHI\_, tiInsize\_, time\_buffer\_size(), and tiOutsize\_.

Referenced by initialize(), and parameterHasChanged().

```

325 {
326     // calculate factors
327     if(inSampleRate_x == 0)
328         inSampleRate_x = 64.0e6/7.0;
329     if(outSampleRate_x == 0)
330         outSampleRate_x = 64.0e6/7.0;
331     time_buffer_size(0);
332
333     // clear
334     inOffset_ = 0;
335     inLength_ = tiInsize_;
336     inReg_.resize(inLength_ + T1_RESAMPLE_ORDER + 1);
337
338     // interpolator basepoint
339     tiBasepointIndex_.resize(tiOutsize_);
340     for(int i = 0; i < tiOutsize_; i++)
341         tiBasepointIndex_[i] = (int) floor(inSampleRate_x * ((double) i /
342             outSampleRate_x));
343
344     // interpolator response
345     tiHI_.resize(tiOutsize_ * (T1_RESAMPLE_ORDER + 1));
346     double dtbase = (1 / inSampleRate_x) / 100.0;
347     int nbase = 0;

```

```

348     double *hbase = blackman_sinc(&nbase, 1 / inSampleRate_x, dtbase,
349         TI_RESAMPLE_ORDER);
350     for(int i = 0; i < TI_RESAMPLE_ORDER + 1; i++)
351     {
352         for(int j = 0; j < tiOutsize_; j++)
353         {
354             tiHI_[i * tiOutsize_ + j] =
355                 (float) interp_response(hbase, nbase, dtbase,
356                     ((double) j / outSampleRate_x) -
357                     ((double) (tiBasepointIndex_[j] - i) / inSampleRate_x));
358         }
359     }
360
361     // dump to file
362     if(!responseFile_x.empty())
363     {
364         FILE *ffp = fopen(responseFile_x.c_str(), "wt");
365         for(int i = 0; i < nbase; i++)
366             fprintf(ffp, "%.10f\n", hbase[i]);
367         fclose(ffp);
368     }
369     free(hbase);
370
371 }

```

#### 7.7.4.13 double iris::phy::Dvbt1InterpolatorComponent::sinc ( double x ) [private]

$\sin(x)/x$  function

##### Parameters

x	Input value
---	-------------

##### Returns

The sinc of the input

Definition at line 296 of file Dvbt1InterpolatorComponent.cpp.

Referenced by blackman\_sinc().

```

297 {
298     return x == 0.0 ? 1.0 : (sin(x) / x);
299 }

```

#### 7.7.4.14 int iris::phy::Dvbt1InterpolatorComponent::time\_buffer\_size ( int input\_samples ) [private]

size correctly the interpolation buffers

##### Parameters

input_samples	Size, in samples, of the input buffer
---------------	---------------------------------------

##### Returns

Size, in samples, of the output buffer

Definition at line 246 of file Dvbt1InterpolatorComponent.cpp.

References find\_rational\_approximation(), inSampleRate\_x, outSampleRate\_x, tilnsize\_, and tiOutsize\_.

Referenced by setup().

```

247 {
248     int output_samples = 0;
249     int status = 0;
250
251     // find the best rational approximation

```

```

252     tiOutsize_ = 0;
253     tiInsize_ = 0;
254     status = find_rational_approximation(&tiOutsize_, &
tiInsize_, outSampleRate_x / inSampleRate_x, 2000);
255     if (status)
256         LOG(ERROR) << "Could not find a rational approximation for " <<
outSampleRate_x << "/" << inSampleRate_x << "=" <<
outSampleRate_x / inSampleRate_x;
257
258     LOG(INFO) << "Original sampling rate: " << inSampleRate_x << " sps";
259     LOG(INFO) << "Effective sampling rate (x" << tiOutsize_ << "/" <<
tiInsize_ << "): " << inSampleRate_x * (double)
tiOutsize_ / (double) tiInsize_ << " sps";
260
261     return output_samples;
262 }

```

## 7.7.5 Member Data Documentation

### 7.7.5.1 bool iris::phy::Dvbt1InterpolatorComponent::debug\_x [private]

Debug flag (default = false)

Definition at line 134 of file Dvbt1InterpolatorComponent.h.

Referenced by Dvbt1InterpolatorComponent(), and process().

### 7.7.5.2 int iris::phy::Dvbt1InterpolatorComponent::inLength\_ [private]

Definition at line 149 of file Dvbt1InterpolatorComponent.h.

Referenced by process(), and setup().

### 7.7.5.3 int iris::phy::Dvbt1InterpolatorComponent::inOffset\_ [private]

Definition at line 147 of file Dvbt1InterpolatorComponent.h.

Referenced by process(), and setup().

### 7.7.5.4 CplxVec iris::phy::Dvbt1InterpolatorComponent::inReg\_ [private]

Definition at line 148 of file Dvbt1InterpolatorComponent.h.

Referenced by process(), and setup().

### 7.7.5.5 double iris::phy::Dvbt1InterpolatorComponent::inSampleRate\_x [private]

Input sampling rate (default = 0)

Definition at line 135 of file Dvbt1InterpolatorComponent.h.

Referenced by Dvbt1InterpolatorComponent(), setup(), and time\_buffer\_size().

### 7.7.5.6 double iris::phy::Dvbt1InterpolatorComponent::outSampleRate\_x [private]

Output sampling rate (default = 0)

Definition at line 136 of file Dvbt1InterpolatorComponent.h.

Referenced by Dvbt1InterpolatorComponent(), setup(), and time\_buffer\_size().

#### 7.7.5.7 `std::string iris::phy::Dvbt1InterpolatorComponent::responseFile_x` [private]

Text file with impulse response (default = none)

Definition at line 137 of file `Dvbt1InterpolatorComponent.h`.

Referenced by `Dvbt1InterpolatorComponent()`, and `setup()`.

#### 7.7.5.8 `double iris::phy::Dvbt1InterpolatorComponent::sampleRate_` [private]

Sample rate of current frame.

Definition at line 143 of file `Dvbt1InterpolatorComponent.h`.

#### 7.7.5.9 `IntVec iris::phy::Dvbt1InterpolatorComponent::tiBasepointIndex_` [private]

Definition at line 150 of file `Dvbt1InterpolatorComponent.h`.

Referenced by `process()`, and `setup()`.

#### 7.7.5.10 `FloatVec iris::phy::Dvbt1InterpolatorComponent::tiHl_` [private]

Definition at line 151 of file `Dvbt1InterpolatorComponent.h`.

Referenced by `process()`, and `setup()`.

#### 7.7.5.11 `int iris::phy::Dvbt1InterpolatorComponent::tiInsize_` [private]

Definition at line 145 of file `Dvbt1InterpolatorComponent.h`.

Referenced by `process()`, `setup()`, and `time_buffer_size()`.

#### 7.7.5.12 `double iris::phy::Dvbt1InterpolatorComponent::timeStamp_` [private]

Timestamp of current frame.

Definition at line 142 of file `Dvbt1InterpolatorComponent.h`.

#### 7.7.5.13 `int iris::phy::Dvbt1InterpolatorComponent::tiOutsize_` [private]

Definition at line 146 of file `Dvbt1InterpolatorComponent.h`.

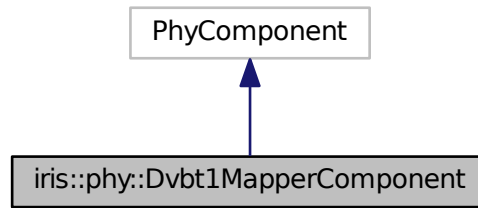
Referenced by `process()`, `setup()`, and `time_buffer_size()`.

## 7.8 `iris::phy::Dvbt1MapperComponent` Class Reference

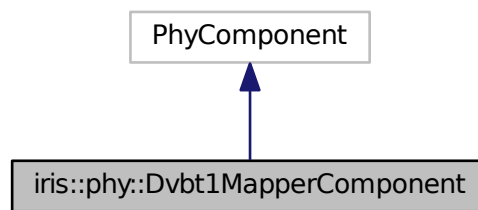
A DVB-T1 mapper component.

```
#include <Dvbt1MapperComponent.h>
```

Inheritance diagram for iris::phy::Dvbt1MapperComponent:



Collaboration diagram for iris::phy::Dvbt1MapperComponent:



## Public Types

- typedef std::vector< uint8\_t > [ByteVec](#)  
*A vector of bytes.*
- typedef ByteVec::iterator [ByteVecIt](#)  
*An iterator for a vector of bytes.*
- typedef std::complex< float > [Cplx](#)  
*A complex type.*
- typedef std::vector< [Cplx](#) > [CplxVec](#)  
*A vector of complex.*
- typedef CplxVec::iterator [CplxVecIt](#)

## Public Member Functions

- [Dvbt1MapperComponent](#) (std::string name)  
*Default constructor.*
- [~Dvbt1MapperComponent](#) ()  
*Default destructor.*
- virtual void [calculateOutputTypes](#) (std::map< std::string, int > &inputTypes, std::map< std::string, int > &outputTypes)

- *Calculate the output port types for the IRIS system.*
- virtual void [registerPorts](#) ()  
*Register the mapper ports with the IRIS system.*
- virtual void [initialize](#) ()  
*Initialize the component.*
- virtual void [process](#) ()  
*Main processing method.*
- virtual void [parameterHasChanged](#) (std::string name)  
*Actions taken when the parameters change.*

## Private Member Functions

- void [setup](#) ()  
*Set up all our constellations.*
- void [destroy](#) ()  
*Destroy the component.*

## Static Private Member Functions

- template<typename T , size\_t N>  
static T \* [begin](#) (T(&arr)[N])  
*Useful templates.*
- template<typename T , size\_t N>  
static T \* [end](#) (T(&arr)[N])

## Private Attributes

- bool [debug\\_x](#)  
*Debug flag (default = false)*
- int [qamMapping\\_x](#)  
*QAM constellation mapping (default = 16)*
- int [hyerarchyMode\\_x](#)  
*Hyerarchical mode (default = 0)*
- double [timeStamp\\_](#)  
*Timestamp of current frame.*
- double [sampleRate\\_](#)  
*Sample rate of current frame.*
- [CplxVec](#) [constel\\_](#)  
*actual constellation*

### 7.8.1 Detailed Description

A DVB-T1 mapper component.

[Dvbt1MapperComponent](#) is the eighth block composing the DVB-T transmission chain. The mapper uses the QAM constellations mandated in the standard, to transform the data symbols into complex numbers that can be eventually delivered over I&Q analog waveforms. The constellations are Gray-encoded, that is, adjacent points in the complex plane only differ in one bit among their represented symbols (indeed, this is only partially true, as there can also be a difference of more bits, but at larger distances).



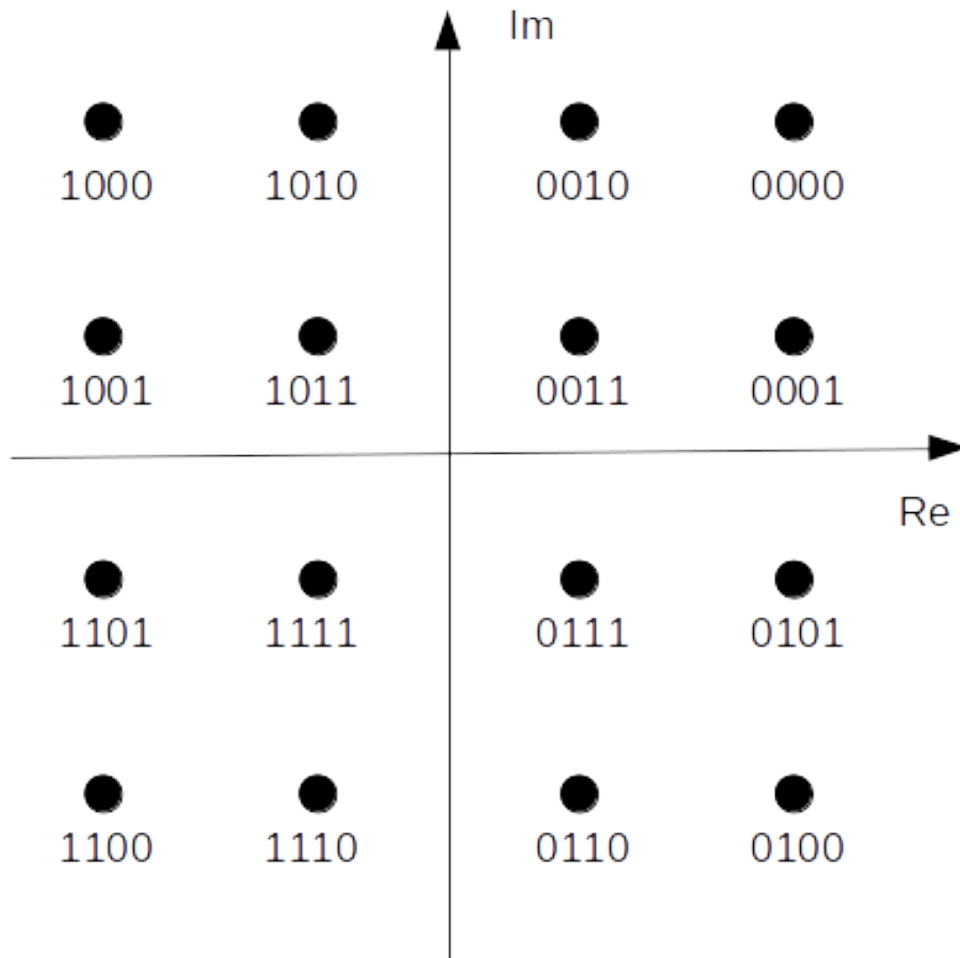


Figure 7.6: DVB-T 16-QAM constellation.

The constellation points are statically written in the source files.

This blocks accepts in input elements in `uint8_t` (  $v$ -bit symbols) and generates in output complex values (complex float).

There are three parameters that can be changed in the XML configuration file:

- *debug*: by default set to "false", is used to print some small debugging information for the interested developer.
- *qammapping*: by default set to "16", this is used to select one of the three possible QAM mappings. The admitted values are "4", "16", "64".
- *hyerarchymode*: by default set to "0", which means "not hyerarchical". Hierarchical modes are used to transmit two different transport streams, one with a high priority (HP) information and another one with a low priority (LP) information. The admitted values are "0", "1", "2", "4". NOTE: hyerarchical modes are not implemented in the current release of this modulator.

## References

- ETSI Standard: *EN 300 744 V1.5.1, Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for digital terrestrial television*, available at [ETSI Publications Download Area](#)

Definition at line 83 of file `Dvbt1MapperComponent.h`.

## 7.8.2 Member Typedef Documentation

### 7.8.2.1 `typedef std::vector<uint8_t> iris::phy::Dvbt1MapperComponent::ByteVec`

A vector of bytes.

Definition at line 89 of file `Dvbt1MapperComponent.h`.

### 7.8.2.2 `typedef ByteVec::iterator iris::phy::Dvbt1MapperComponent::ByteVecIt`

An iterator for a vector of bytes.

Definition at line 92 of file `Dvbt1MapperComponent.h`.

### 7.8.2.3 `typedef std::complex<float> iris::phy::Dvbt1MapperComponent::Cplx`

A complex type.

Definition at line 95 of file `Dvbt1MapperComponent.h`.

### 7.8.2.4 `typedef std::vector<Cplx> iris::phy::Dvbt1MapperComponent::CplxVec`

A vector of complex.

Definition at line 98 of file `Dvbt1MapperComponent.h`.

### 7.8.2.5 `typedef CplxVec::iterator iris::phy::Dvbt1MapperComponent::CplxVecIt`

Definition at line 101 of file `Dvbt1MapperComponent.h`.

## 7.8.3 Constructor & Destructor Documentation

### 7.8.3.1 `iris::phy::Dvbt1MapperComponent::Dvbt1MapperComponent ( std::string name )`

Default constructor.

Registers the block parameters and initializes some variables

Definition at line 57 of file `Dvbt1MapperComponent.cpp`.

References `begin()`, `debug_x`, `end()`, `hyerarchyMode_x`, and `qamMapping_x`.

```

58 : PhyComponent (name,                                // component name
59                 "dvbt1mapper",                        // component type
60                 "A DVB-T1 mapper component",          // description
61                 "Giuseppe Baruffa",                  // author
62                 "0.1")                                // version
63 ,sampleRate_(0)
64 ,timeStamp_(0)
65 {
66     registerParameter(
67         "debug", "Whether to output debug data",
68         "false", true, debug_x);
69
70     int qamarr[] = {4,16,64};
71     registerParameter(
72         "qammapping", "QAM constellation mapping",
73         "16", true, qamMapping_x, list<int>(begin(qamarr),end(qamarr)));
74
75     int harr[] = {0,1,2,4};
76     registerParameter(
77         "hyerarchymode", "Hyerarchical mode (0 = NH)",
78         "0", true, hyerarchyMode_x, list<int>(begin(harr),end(harr)));
79 }
```

**7.8.3.2** iris::phy::Dvbt1MapperComponent::~~Dvbt1MapperComponent ( )

Default destructor.

Just calls [destroy\(\)](#).

Definition at line 84 of file Dvbt1MapperComponent.cpp.

References [destroy\(\)](#).

```
85 {
86     destroy\(\);
87 }
```

**7.8.4 Member Function Documentation****7.8.4.1** template<typename T , size\_t N> static T\* iris::phy::Dvbt1MapperComponent::begin ( T(&) arr[N] ) [inline], [static], [private]

Useful templates.

Definition at line 129 of file Dvbt1MapperComponent.h.

Referenced by [Dvbt1MapperComponent\(\)](#).

```
129 { return &arr[0]; }
```

**7.8.4.2** void iris::phy::Dvbt1MapperComponent::calculateOutputTypes ( std::map< std::string, int > & inputTypes, std::map< std::string, int > & outputTypes ) [virtual]

Calculate the output port types for the IRIS system.

The single output port must provide complex values.

Definition at line 102 of file Dvbt1MapperComponent.cpp.

```
105 {
106     outputTypes["output1"] = TypeInfo< Cplx >::identifier;
107 }
```

**7.8.4.3** void iris::phy::Dvbt1MapperComponent::destroy ( ) [private]

Destroy the component.

Definition at line 284 of file Dvbt1MapperComponent.cpp.

Referenced by [parameterHasChanged\(\)](#), and [~Dvbt1MapperComponent\(\)](#).

```
285 {
286 }
```

**7.8.4.4** template<typename T , size\_t N> static T\* iris::phy::Dvbt1MapperComponent::end ( T(&) arr[N] ) [inline], [static], [private]

Definition at line 131 of file Dvbt1MapperComponent.h.

Referenced by [Dvbt1MapperComponent\(\)](#).

```
131 { return &arr[0]+N; }
```

#### 7.8.4.5 void iris::phy::Dvbt1MapperComponent::initialize ( ) [virtual]

Initialize the component.

Just calls [setup\(\)](#).

Definition at line 112 of file Dvbt1MapperComponent.cpp.

References [setup\(\)](#).

```
113 {
114     setup();
115 }
```

#### 7.8.4.6 void iris::phy::Dvbt1MapperComponent::parameterHasChanged ( std::string name ) [virtual]

Actions taken when the parameters change.

This block has two significant parameters

Definition at line 155 of file Dvbt1MapperComponent.cpp.

References [destroy\(\)](#), and [setup\(\)](#).

```
156 {
157     if(name == "qammapping" || name == "hierarchyMode")
158     {
159         destroy();
160         setup();
161     }
162 }
```

#### 7.8.4.7 void iris::phy::Dvbt1MapperComponent::process ( ) [virtual]

Main processing method.

Definition at line 118 of file Dvbt1MapperComponent.cpp.

References [constel\\_](#), and [debug\\_x](#).

```
119 {
120     // request input
121     DataSet< uint8_t > *in = NULL;
122     getInputDataSet("input1", in);
123
124     // calculate sizes
125     int insize = in ? (int) in->data.size() : 0;
126     int outsize = insize;
127
128     // request output
129     DataSet< Cplx > *out = NULL;
130     getOutputDataSet("output1", out, outsize);
131
132     // print debug info
133     if(debug_x)
134         LOG(LINFO) << "in/out: " << insize << "/" << outsize;
135
136     // assign
137     CplxVecIt outit = out->data.begin();
138     for(ByteVecIt init = in->data.begin(); init < in->data.end(); init++, outit++)
139     {
140         *outit = constel_[*init];
141     }
142
143     //Copy the timestamp and sample rate for the DataSets
144     out->timeStamp = in->timeStamp;
145     out->sampleRate = in->sampleRate;
146
147     // release input and output
148     releaseInputDataSet("input1", in);
149     releaseOutputDataSet("output1", out);
150 }
```

**7.8.4.8 void iris::phy::Dvbt1MapperComponent::registerPorts ( ) [virtual]**

Register the mapper ports with the IRIS system.

This component has one input that accept symbols (some bits per byte) and one output that provides complex symbols (in floats).

Definition at line 93 of file Dvbt1MapperComponent.cpp.

```

94 {
95     registerInputPort("input1", TypeInfo< uint8_t >::identifier);
96     registerOutputPort("output1", TypeInfo< Cplx >::identifier);
97 }
```

**7.8.4.9 void iris::phy::Dvbt1MapperComponent::setup ( ) [private]**

Set up all our constellations.

Definition at line 165 of file Dvbt1MapperComponent.cpp.

References `constel_`, `hyerarchyMode_x`, and `qamMapping_x`.

Referenced by `initialize()`, and `parameterHasChanged()`.

```

166 {
167     // nonuniformity value
168     float alpha = hyerarchyMode_x == 0 ? 1 : (float) ceil((double) (1 <<
169         hyerarchyMode_x) / 2.0);
170     // constellation array
171     switch(qamMapping_x)
172     {
173     case 4:
174         constel_.push_back(Cplx(1, 1));
175         constel_.push_back(Cplx(1, -1));
176         constel_.push_back(Cplx(-1, 1));
177         constel_.push_back(Cplx(-1, -1));
178         break;
179     case 16:
180         constel_.push_back(Cplx(3, 3));
181         constel_.push_back(Cplx(3, 1));
182         constel_.push_back(Cplx(1, 3));
183         constel_.push_back(Cplx(1, 1));
184         constel_.push_back(Cplx(3, -3));
185         constel_.push_back(Cplx(3, -1));
186         constel_.push_back(Cplx(1, -3));
187         constel_.push_back(Cplx(1, -1));
188         constel_.push_back(Cplx(-3, 3));
189         constel_.push_back(Cplx(-3, 1));
190         constel_.push_back(Cplx(-1, 3));
191         constel_.push_back(Cplx(-1, 1));
192         constel_.push_back(Cplx(-3, -3));
193         constel_.push_back(Cplx(-3, -1));
194         constel_.push_back(Cplx(-1, -3));
195         constel_.push_back(Cplx(-1, -1));
196         break;
197     case 64:
198         constel_.push_back(Cplx(7, 7));
199         constel_.push_back(Cplx(7, 5));
200         constel_.push_back(Cplx(5, 7));
201         constel_.push_back(Cplx(5, 5));
202         constel_.push_back(Cplx(7, 1));
203         constel_.push_back(Cplx(7, 3));
204         constel_.push_back(Cplx(5, 1));
205         constel_.push_back(Cplx(5, 3));
206         constel_.push_back(Cplx(1, 7));
207         constel_.push_back(Cplx(1, 5));
208         constel_.push_back(Cplx(3, 7));
209         constel_.push_back(Cplx(3, 5));
210         constel_.push_back(Cplx(1, 1));
211         constel_.push_back(Cplx(1, 3));
212         constel_.push_back(Cplx(3, 1));
213         constel_.push_back(Cplx(3, 3));
214         constel_.push_back(Cplx(7, -7));
215         constel_.push_back(Cplx(7, -5));
216         constel_.push_back(Cplx(5, -7));
217         constel_.push_back(Cplx(5, -5));
218         constel_.push_back(Cplx(7, -1));
219         constel_.push_back(Cplx(7, -3));
```

```

220     constel_.push_back(Cplx(5, -1));
221     constel_.push_back(Cplx(5, -3));
222     constel_.push_back(Cplx(1, -7));
223     constel_.push_back(Cplx(1, -5));
224     constel_.push_back(Cplx(3, -7));
225     constel_.push_back(Cplx(3, -5));
226     constel_.push_back(Cplx(1, -1));
227     constel_.push_back(Cplx(1, -3));
228     constel_.push_back(Cplx(3, -1));
229     constel_.push_back(Cplx(3, -3));
230     constel_.push_back(Cplx(-7, 7));
231     constel_.push_back(Cplx(-7, 5));
232     constel_.push_back(Cplx(-5, 7));
233     constel_.push_back(Cplx(-5, 5));
234     constel_.push_back(Cplx(-7, 1));
235     constel_.push_back(Cplx(-7, 3));
236     constel_.push_back(Cplx(-5, 1));
237     constel_.push_back(Cplx(-5, 3));
238     constel_.push_back(Cplx(-1, 7));
239     constel_.push_back(Cplx(-1, 5));
240     constel_.push_back(Cplx(-3, 7));
241     constel_.push_back(Cplx(-3, 5));
242     constel_.push_back(Cplx(-1, 1));
243     constel_.push_back(Cplx(-1, 3));
244     constel_.push_back(Cplx(-3, 1));
245     constel_.push_back(Cplx(-3, 3));
246     constel_.push_back(Cplx(-7, -7));
247     constel_.push_back(Cplx(-7, -5));
248     constel_.push_back(Cplx(-5, -7));
249     constel_.push_back(Cplx(-5, -5));
250     constel_.push_back(Cplx(-7, -1));
251     constel_.push_back(Cplx(-7, -3));
252     constel_.push_back(Cplx(-5, -1));
253     constel_.push_back(Cplx(-5, -3));
254     constel_.push_back(Cplx(-1, -7));
255     constel_.push_back(Cplx(-1, -5));
256     constel_.push_back(Cplx(-3, -7));
257     constel_.push_back(Cplx(-3, -5));
258     constel_.push_back(Cplx(-1, -1));
259     constel_.push_back(Cplx(-1, -3));
260     constel_.push_back(Cplx(-3, -1));
261     constel_.push_back(Cplx(-3, -3));
262     break;
263 }
264
265 // add alpha and find energy
266 float energy = 0;
267 for(int m = 0; m < constel_.size(); m++)
268 {
269     constel_[m].real(constel_[m].real() + (alpha - 1) * (
constel_[m].real() >= 0 ? 1 : -1));
270     constel_[m].imag(constel_[m].imag() + (alpha - 1) * (
constel_[m].imag() >= 0 ? 1 : -1));
271     energy += constel_[m].real() * constel_[m].real() + constel_[m].imag() *
constel_[m].imag();
272 }
273 energy = sqrtf(energy / constel_.size());
274
275 // normalize to have unit energy
276 for(int m = 0; m < constel_.size(); m++)
277 {
278     constel_[m].real(constel_[m].real() / energy);
279     constel_[m].imag(constel_[m].imag() / energy);
280 }
281 }

```

## 7.8.5 Member Data Documentation

### 7.8.5.1 CplxVec iris::phy::Dvbt1MapperComponent::constel\_ [private]

actual constellation

Definition at line 125 of file Dvbt1MapperComponent.h.

Referenced by process(), and setup().

### 7.8.5.2 bool iris::phy::Dvbt1MapperComponent::debug\_x [private]

Debug flag (default = false)

Definition at line 115 of file Dvbt1MapperComponent.h.

Referenced by Dvbt1MapperComponent(), and process().

#### 7.8.5.3 int iris::phy::Dvbt1MapperComponent::hyerarchyMode\_x [private]

Hyerarchical mode (default = 0)

Definition at line 117 of file Dvbt1MapperComponent.h.

Referenced by Dvbt1MapperComponent(), and setup().

#### 7.8.5.4 int iris::phy::Dvbt1MapperComponent::qamMapping\_x [private]

QAM constellation mapping (default = 16)

Definition at line 116 of file Dvbt1MapperComponent.h.

Referenced by Dvbt1MapperComponent(), and setup().

#### 7.8.5.5 double iris::phy::Dvbt1MapperComponent::sampleRate\_ [private]

Sample rate of current frame.

Definition at line 123 of file Dvbt1MapperComponent.h.

#### 7.8.5.6 double iris::phy::Dvbt1MapperComponent::timeStamp\_ [private]

Timestamp of current frame.

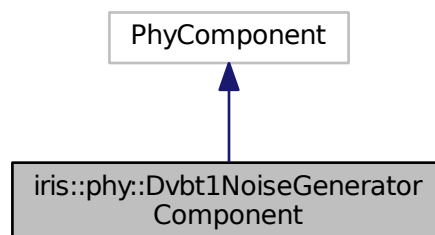
Definition at line 122 of file Dvbt1MapperComponent.h.

## 7.9 iris::phy::Dvbt1NoiseGeneratorComponent Class Reference

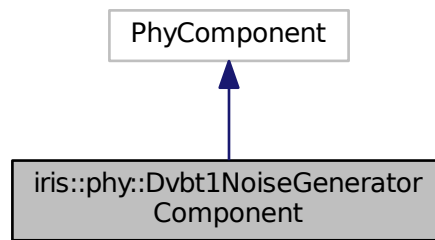
A DVB-T1 noise generator.

```
#include <Dvbt1NoiseGeneratorComponent.h>
```

Inheritance diagram for iris::phy::Dvbt1NoiseGeneratorComponent:



Collaboration diagram for `iris::phy::Dvbt1NoiseGeneratorComponent`:



## Public Types

- `typedef std::vector< uint8_t > ByteVec`
- `typedef ByteVec::iterator ByteVecIt`
- `typedef std::complex< float > Cplx`
- `typedef std::vector< Cplx > CplxVec`
- `typedef CplxVec::iterator CplxVecIt`
- `typedef std::vector< float > FloatVec`
- `typedef FloatVec::iterator FloatVecIt`
- `typedef std::vector< int > IntVec`
- `typedef IntVec::iterator IntVecIt`
- `typedef`  
`boost::normal_distribution`  
`< double > NormalDistribution`
- `typedef boost::mt19937 RandomGenerator`
- `typedef`  
`boost::variate_generator`  
`< RandomGenerator,`  
`NormalDistribution > GaussianGenerator`

## Public Member Functions

- `Dvbt1NoiseGeneratorComponent` (`std::string name`)
- `~Dvbt1NoiseGeneratorComponent` ()
- virtual void `calculateOutputTypes` (`std::map< std::string, int > &inputTypes, std::map< std::string, int > &outputTypes`)
- virtual void `registerPorts` ()
- virtual void `initialize` ()
- virtual void `process` ()
- virtual void `parameterHasChanged` (`std::string name`)

## Private Member Functions

- void `setup` ()  
*Set up all our index vectors and containers.*
- void `destroy` ()



## Static Private Member Functions

- template<typename T , size\_t N>  
static T \* [begin](#) (T(&arr)[N])
- template<typename T , size\_t N>  
static T \* [end](#) (T(&arr)[N])

## Private Attributes

- bool [debug\\_x](#)  
*Debug flag (default = false)*
- double [variance\\_x](#)  
*Noise variance (default = 1)*
- int [blockSize\\_x](#)  
*Size of blocks to generate.*
- double [frequency\\_x](#)  
*Size of blocks to generate.*
- double [timeStamp\\_](#)  
*Timestamp of current frame.*
- double [sampleRate\\_](#)  
*Sample rate of current frame.*
- double [theta0\\_](#)
- [GaussianGenerator](#) \* [gen](#)

### 7.9.1 Detailed Description

A DVB-T1 noise generator.

Definition at line 50 of file Dvbt1NoiseGeneratorComponent.h.

### 7.9.2 Member Typedef Documentation

#### 7.9.2.1 typedef std::vector<uint8\_t> iris::phy::Dvbt1NoiseGeneratorComponent::ByteVec

Definition at line 55 of file Dvbt1NoiseGeneratorComponent.h.

#### 7.9.2.2 typedef ByteVec::iterator iris::phy::Dvbt1NoiseGeneratorComponent::ByteVecIt

Definition at line 56 of file Dvbt1NoiseGeneratorComponent.h.

#### 7.9.2.3 typedef std::complex<float> iris::phy::Dvbt1NoiseGeneratorComponent::Cplx

Definition at line 57 of file Dvbt1NoiseGeneratorComponent.h.

#### 7.9.2.4 typedef std::vector<Cplx> iris::phy::Dvbt1NoiseGeneratorComponent::CplxVec

Definition at line 58 of file Dvbt1NoiseGeneratorComponent.h.

#### 7.9.2.5 typedef CplxVec::iterator iris::phy::Dvbt1NoiseGeneratorComponent::CplxVecIt

Definition at line 59 of file Dvbt1NoiseGeneratorComponent.h.

### 7.9.2.6 `typedef std::vector<float> iris::phy::Dvbt1NoiseGeneratorComponent::FloatVec`

Definition at line 60 of file `Dvbt1NoiseGeneratorComponent.h`.

### 7.9.2.7 `typedef FloatVec::iterator iris::phy::Dvbt1NoiseGeneratorComponent::FloatVecIt`

Definition at line 61 of file `Dvbt1NoiseGeneratorComponent.h`.

### 7.9.2.8 `typedef boost::variate_generator<RandomGenerator,NormalDistribution> iris::phy::Dvbt1NoiseGeneratorComponent::GaussianGenerator`

Definition at line 66 of file `Dvbt1NoiseGeneratorComponent.h`.

### 7.9.2.9 `typedef std::vector<int> iris::phy::Dvbt1NoiseGeneratorComponent::IntVec`

Definition at line 62 of file `Dvbt1NoiseGeneratorComponent.h`.

### 7.9.2.10 `typedef IntVec::iterator iris::phy::Dvbt1NoiseGeneratorComponent::IntVecIt`

Definition at line 63 of file `Dvbt1NoiseGeneratorComponent.h`.

### 7.9.2.11 `typedef boost::normal_distribution<double> iris::phy::Dvbt1NoiseGeneratorComponent::Normal- Distribution`

Definition at line 64 of file `Dvbt1NoiseGeneratorComponent.h`.

### 7.9.2.12 `typedef boost::mt19937 iris::phy::Dvbt1NoiseGeneratorComponent::RandomGenerator`

Definition at line 65 of file `Dvbt1NoiseGeneratorComponent.h`.

## 7.9.3 Constructor & Destructor Documentation

### 7.9.3.1 `iris::phy::Dvbt1NoiseGeneratorComponent::Dvbt1NoiseGeneratorComponent ( std::string name )`

Definition at line 54 of file `Dvbt1NoiseGeneratorComponent.cpp`.

References `blockSize_x`, `debug_x`, `frequency_x`, and `variance_x`.

```

55 : PhyComponent (name,                                // component name
56                 "dvbtlnoisegenerator",              // component type
57                 "A DVB-T1 noise generator component", // description
58                 "Giuseppe Baruffa",                 // author
59                 "0.1")                               // version
60 ,sampleRate_(0)
61 ,timeStamp_(0)
62 ,theta0_(0)
63 {
64     registerParameter(
65         "variance", "Noise variance (default = 1)",
66         "1", true, variance_x, Interval<double>(0.0,1000000.0));
67
68     registerParameter(
69         "frequency", "Cosine frequency",
70         "3e6", true, frequency_x, Interval<double>(0.0,100000000.0));
71
72     registerParameter("blocksize",
73                       "Size of generated blocks",
74                       "1024",
75                       true,
76                       blockSize_x,
```

```

77             Interval<int>(1, 128*1024*1024));
78
79     registerParameter(
80         "debug", "Whether to output debug data",
81         "false", true, debug_x);
82 }

```

### 7.9.3.2 iris::phy::Dvbt1NoiseGeneratorComponent::~~Dvbt1NoiseGeneratorComponent ( )

Definition at line 84 of file Dvbt1NoiseGeneratorComponent.cpp.

References `destroy()`.

```

85 {
86     destroy();
87 }

```

## 7.9.4 Member Function Documentation

### 7.9.4.1 template<typename T, size\_t N> static T\* iris::phy::Dvbt1NoiseGeneratorComponent::begin ( T(&) arr[N] ) [inline], [static], [private]

Definition at line 95 of file Dvbt1NoiseGeneratorComponent.h.

```

95 { return &arr[0]; }

```

### 7.9.4.2 void iris::phy::Dvbt1NoiseGeneratorComponent::calculateOutputTypes ( std::map< std::string, int > & inputTypes, std::map< std::string, int > & outputTypes ) [virtual]

Definition at line 94 of file Dvbt1NoiseGeneratorComponent.cpp.

```

97 {
98     outputTypes["output1"] = TypeInfo< Cplx >::identifier;
99 }

```

### 7.9.4.3 void iris::phy::Dvbt1NoiseGeneratorComponent::destroy ( ) [private]

Definition at line 151 of file Dvbt1NoiseGeneratorComponent.cpp.

References `gen`.

Referenced by `parameterHasChanged()`, and `~Dvbt1NoiseGeneratorComponent()`.

```

152 {
153     delete gen;
154 }

```

### 7.9.4.4 template<typename T, size\_t N> static T\* iris::phy::Dvbt1NoiseGeneratorComponent::end ( T(&) arr[N] ) [inline], [static], [private]

Definition at line 97 of file Dvbt1NoiseGeneratorComponent.h.

```

97 { return &arr[0]+N; }

```

#### 7.9.4.5 void iris::phy::Dvbt1NoiseGeneratorComponent::initialize ( ) [virtual]

Definition at line 101 of file Dvbt1NoiseGeneratorComponent.cpp.

References setup().

```
102 {
103     setup();
104 }
```

#### 7.9.4.6 void iris::phy::Dvbt1NoiseGeneratorComponent::parameterHasChanged ( std::string name ) [virtual]

Definition at line 133 of file Dvbt1NoiseGeneratorComponent.cpp.

References destroy(), and setup().

```
134 {
135     if(name == "variance")
136     {
137         destroy();
138         setup();
139     }
140 }
```

#### 7.9.4.7 void iris::phy::Dvbt1NoiseGeneratorComponent::process ( ) [virtual]

Definition at line 106 of file Dvbt1NoiseGeneratorComponent.cpp.

References blockSize\_x, debug\_x, frequency\_x, theta0\_, and variance\_x.

```
107 {
108     DataSet< Cplx >* out = NULL;
109     getOutputDataSet("output1", out, blockSize_x);
110
111     if(debug_x)
112         LOG(LINFO) << "out: " << blockSize_x;
113
114     float dtheta = 2*M_PI*frequency_x/12.5e6;
115     float A = sqrt(variance_x * 2);
116     int n = 0;
117     for(CplxVecIt outit = out->data.begin(); outit < out->data.end(); outit++)
118     {
119         /*outit->real((*gen)());
120         outit->imag((*gen)());*/
121         outit->real(A*cos(theta0_ + dtheta*(n++)));
122         outit->imag(0);
123     }
124     theta0_ += dtheta*n;
125
126     //Copy the timestamp and sample rate for the DataSets
127     out->timeStamp = 0;
128     out->sampleRate = 0;
129
130     releaseOutputDataSet("output1", out);
131 }
```

#### 7.9.4.8 void iris::phy::Dvbt1NoiseGeneratorComponent::registerPorts ( ) [virtual]

Definition at line 89 of file Dvbt1NoiseGeneratorComponent.cpp.

```
90 {
91     registerOutputPort("output1", TypeInfo< Cplx >::identifier);
92 }
```

**7.9.4.9 void iris::phy::Dvbt1NoiseGeneratorComponent::setup ( ) [private]**

Set up all our index vectors and containers.

Definition at line 143 of file Dvbt1NoiseGeneratorComponent.cpp.

References `gen`, `theta0_`, and `variance_x`.

Referenced by `initialize()`, and `parameterHasChanged()`.

```

144 {
145     RandomGenerator eng;
146     NormalDistribution dist(0, variance_x / 2);
147     gen = new GaussianGenerator(eng,dist);
148     theta0_ = 0;
149 }
```

**7.9.5 Member Data Documentation****7.9.5.1 int iris::phy::Dvbt1NoiseGeneratorComponent::blockSize\_x [private]**

Size of blocks to generate.

Definition at line 82 of file Dvbt1NoiseGeneratorComponent.h.

Referenced by `Dvbt1NoiseGeneratorComponent()`, and `process()`.

**7.9.5.2 bool iris::phy::Dvbt1NoiseGeneratorComponent::debug\_x [private]**

Debug flag (default = false)

Definition at line 80 of file Dvbt1NoiseGeneratorComponent.h.

Referenced by `Dvbt1NoiseGeneratorComponent()`, and `process()`.

**7.9.5.3 double iris::phy::Dvbt1NoiseGeneratorComponent::frequency\_x [private]**

Size of blocks to generate.

Definition at line 83 of file Dvbt1NoiseGeneratorComponent.h.

Referenced by `Dvbt1NoiseGeneratorComponent()`, and `process()`.

**7.9.5.4 GaussianGenerator\* iris::phy::Dvbt1NoiseGeneratorComponent::gen [private]**

Definition at line 92 of file Dvbt1NoiseGeneratorComponent.h.

Referenced by `destroy()`, and `setup()`.

**7.9.5.5 double iris::phy::Dvbt1NoiseGeneratorComponent::sampleRate\_ [private]**

Sample rate of current frame.

Definition at line 89 of file Dvbt1NoiseGeneratorComponent.h.

**7.9.5.6 double iris::phy::Dvbt1NoiseGeneratorComponent::theta0\_ [private]**

Definition at line 90 of file Dvbt1NoiseGeneratorComponent.h.

Referenced by `process()`, and `setup()`.

7.9.5.7 `double iris::phy::Dvbt1NoiseGeneratorComponent::timeStamp_` [private]

Timestamp of current frame.

Definition at line 88 of file `Dvbt1NoiseGeneratorComponent.h`.

7.9.5.8 `double iris::phy::Dvbt1NoiseGeneratorComponent::variance_x` [private]

Noise variance (default = 1)

Definition at line 81 of file `Dvbt1NoiseGeneratorComponent.h`.

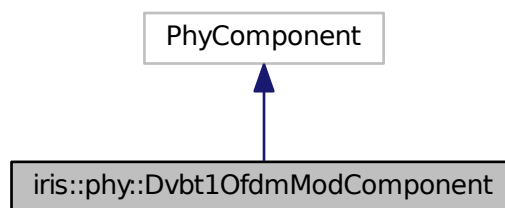
Referenced by `Dvbt1NoiseGeneratorComponent()`, `process()`, and `setup()`.

## 7.10 `iris::phy::Dvbt1OfdmModComponent` Class Reference

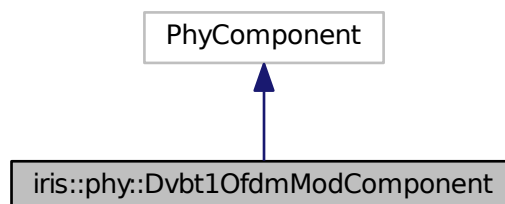
A DVB-T1 OFDM modulator component.

```
#include <Dvbt1OfdmModComponent.h>
```

Inheritance diagram for `iris::phy::Dvbt1OfdmModComponent`:



Collaboration diagram for `iris::phy::Dvbt1OfdmModComponent`:



### Public Types

- `typedef std::vector< uint8_t > ByteVec`

- A vector of bytes.*

  - typedef ByteVec::iterator [ByteVecIt](#)

*An iterator for a vector of bytes.*
- typedef std::complex< float > [Cplx](#)

*A complex type.*
- typedef std::vector< [Cplx](#) > [CplxVec](#)

*A vector of complex.*
- typedef CplxVec::iterator [CplxVecIt](#)
- typedef std::vector< float > [FloatVec](#)

*A vector of float.*
- typedef FloatVec::iterator [FloatVecIt](#)

*An iterator for a vector of float.*
- typedef std::vector< int > [IntVec](#)

*A vector of integers.*
- typedef IntVec::iterator [IntVecIt](#)

*An iterator for a vector of typedef.*

## Public Member Functions

- [Dvbt1OfdmModComponent](#) (std::string name)
 

*Default constructor.*
- [~Dvbt1OfdmModComponent](#) ()
 

*Default destructor.*
- virtual void [calculateOutputTypes](#) (std::map< std::string, int > &inputTypes, std::map< std::string, int > &outputTypes)
 

*Calculate the output port types for the IRIS system.*
- virtual void [registerPorts](#) ()
 

*Register the mapper ports with the IRIS system.*
- virtual void [initialize](#) ()
 

*Initialize the component.*
- virtual void [process](#) ()
 

*Main processing method.*
- virtual void [parameterHasChanged](#) (std::string name)
 

*Actions taken when the parameters change.*

## Private Member Functions

- void [setup](#) ()
 

*Set up all needed constants.*
- void [destroy](#) ()
 

*Destroy the component.*
- void [powerProcedure\\_](#) ()
 

*Separate thread for power loading.*
- double [sinc](#) (double x)
 

*sin(x)/x function*
- double [frequency\\_response\\_modulus](#) (double \*h, int n, double dt, double f)
 

*Calculate the frequency response modulus of an impulse response.*
- double \* [blackman\\_sinc](#) (int \*n\_order, double T, double dt, int order)
 

*Calculate a Blackman-windowed sinc.*

## Static Private Member Functions

- `template<typename T, size_t N>`  
`static T * begin (T(&arr)[N])`

*Useful templates.*

- `template<typename T, size_t N>`  
`static T * end (T(&arr)[N])`

## Private Attributes

- `bool debug\_x`  
*Debug flag (default = false)*
- `int ofdmMode\_x`  
*OFDM mode (default = 2048)*
- `int deltaMode\_x`  
*Cyclic prefix ratio (default = 32)*
- `float outPower\_x`  
*Output power indicator (default = 10)*
- `double dacSampleRate\_x`  
*Sampling rate used by the DAC.*
- `std::string powerFile\_x`  
*Text file with power loading (default = none)*
- `double powerInterval\_x`  
*Power update interval in seconds (default = 0)*
- `double timeStamp\_`  
*Timestamp of current frame.*
- `double sampleRate\_`  
*Sample rate of current frame.*
- `int nFft\_`
- `int nDelta\_`
- `int nBlock\_`
- `int inOffset\_`
- `CplxVec inReg\_`
- `CplxVec fftReg\_`
- `int nMax\_`
- `int kMax\_`
- `int tpsNum\_`
- `int nBit\_`
- `float multFactor\_`
- `FloatVec \_precorrFactor\_`
- `FloatVec precorrFactor\_`
- `FloatVec \_ampliFactor\_`
- `FloatVec ampliFactor\_`
- `fftwf_plan fft\_`  
*Our FFT object pointer.*
- `Cplx * fftBins\_`  
*Allocated using `fftwf_malloc` (SIMD aligned)*
- `boost::thread * powerThread\_`
- `bool runPower\_`



### 7.10.1 Detailed Description

A DVB-T1 OFDM modulator component.

[Dvbt1OfdmModComponent](#) is the tenth block composing the DVB-T transmission chain. The OFDM block takes the modulated QAM cells, assembled in frames together with the pilot and TPS cells, and converts them from a *virtual* frequency domain sequence to a time domain signal, which can be transmitted on a channel.

Not all the carriers are modulated, but some of them are left untouched for purposes of guard bandwidth implementation (*virtual* carriers). Due to the peculiar way frequencies are structured in the sampled frequency domain, the central part of the spectrum is left to the virtual carriers, whereas the outer portions are occupied by the active carriers.

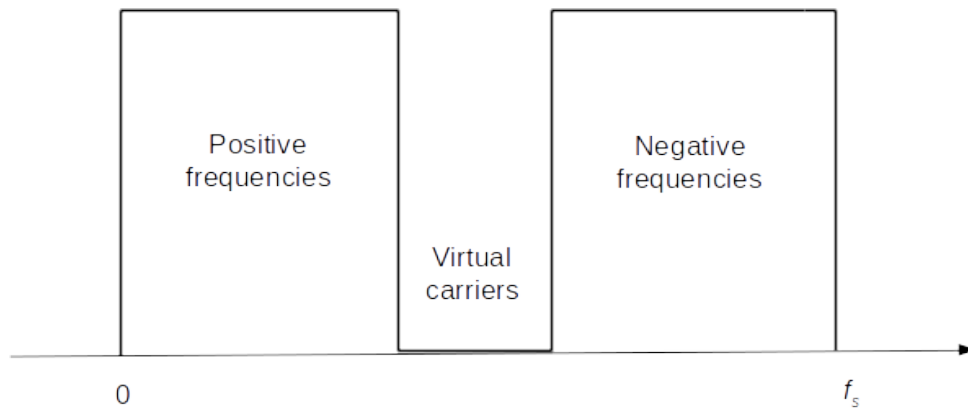


Figure 7.7: OFDM carriers arrangement.

The conversion between the frequency and time domains can be done in several ways, either by using a bank of quadrature modulators or an inverse Discrete Frequency Transform algorithm. In our case, we use an inverse Fast Fourier Transform algorithm, and the signal generated starting from the active cells  $\Psi_k$  can be written as

$$x[n] = \frac{1}{N_{\text{FFT}}} \sum_{k=0}^{N_{\text{FFT}}-1} \rho_k \Lambda_k \Psi_k e^{j \frac{2\pi}{N_{\text{FFT}}} kn}, \quad n = -L, -(L-1), \dots, 1, 0, 1, 2, \dots, (N_{\text{FFT}} - 1),$$

where  $L$  is the cyclic prefix size,  $\rho_k$  is a frequency amplitude linear precorrection term and  $\Lambda_k$  is a power-loading factor: the purpose of these terms will be clarified below.

This block accepts in input complex float values and generates in output complex float values.

There are several parameters that can be changed in the XML configuration file:

- *debug*: by default set to "false", is used to print some small debugging information for the interested developer.
- *ofdmmode*: by default set to "2048", this is used to select one of the three possible OFDM modes. The admitted values are "2048", "4096", "8192", respectively for 2K, 4K (DVB-H, unused), and 8K.
- *deltamode*: by default set to "32", this is used to select one of the four possible cyclic prefix lengths. The admitted values are "32", "16", "8", and "4", which are directly derived from the denominator of the cyclic prefix fraction (1/32, 1/16, 1/8, 1/4).
- *outpower*: by default set to "10", this parameter represents the scaling factor used for rescaling the IFFT output into the wanted range. In particular, this parameter is a percentage. A percentage of 100 means that the output signal real and imaginary parts have an amplitude distribution that concentrates the values into a interval between -1 and 1 with the 99.7% of probability. Since the OFDM signal is Gaussian, this means that the  $\pm 3\sigma$  interval of amplitudes falls in the span  $[-1, +1]$ . When the digital signal is mapped onto analog values by the USRP DAC, for example, the valid range is that enclosed in the  $[-1, +1]$  interval, all other values will be clipped.

- *dacsamplerate*: by default set to "0", a placeholder for 64e6/7 Hz. This represents the sampling rate adopted by the DAC for emitting the BB analog signal. It is used internally to precorrect, linearly, with a multiplicative factor  $\rho_k$ , the amplitude of the OFDM carriers that will be distorted by the Dvbt1Interpolator block. The type of distortion is decided by the algorithm adopted internally by the interpolator block. **Please note that if you are not using the Dvbt1Interpolator block, then you need to leave this parameter at 0.**
- *powerfile*: by default empty, this is the name of a text file that can be read, at periodic intervals, to generate a powerloading configuration for the OFDM carriers. This file contains, line by line, the value of power correction, expressed in dB, for each one of the OFDM carriers. For instance, for 8K OFDM, the file is composed by 8192 lines. A value of 0 means that the power of the carrier is left untouched, a positive value means that there will be a power increase, a negative value will result into a power decrease. The positioning of the carrier indices starts from the first, lowest frequency carrier up to the last, highest frequency carrier.
- *powerinterval*: by default it is set to "1". This is the number of seconds among consecutive reads of the power loading file.

## References

- ETSI Standard: *EN 300 744 V1.5.1, Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for digital terrestrial television*, available at [ETSI Publications Download Area](#)

Definition at line 139 of file Dvbt1OfdmModComponent.h.

## 7.10.2 Member Typedef Documentation

### 7.10.2.1 `typedef std::vector<uint8_t> iris::phy::Dvbt1OfdmModComponent::ByteVec`

A vector of bytes.

Definition at line 145 of file Dvbt1OfdmModComponent.h.

### 7.10.2.2 `typedef ByteVec::iterator iris::phy::Dvbt1OfdmModComponent::ByteVecIt`

An iterator for a vector of bytes.

Definition at line 148 of file Dvbt1OfdmModComponent.h.

### 7.10.2.3 `typedef std::complex<float> iris::phy::Dvbt1OfdmModComponent::Cplx`

A complex type.

Definition at line 151 of file Dvbt1OfdmModComponent.h.

### 7.10.2.4 `typedef std::vector<Cplx> iris::phy::Dvbt1OfdmModComponent::CplxVec`

A vector of complex.

Definition at line 154 of file Dvbt1OfdmModComponent.h.

### 7.10.2.5 `typedef CplxVec::iterator iris::phy::Dvbt1OfdmModComponent::CplxVecIt`

Definition at line 157 of file Dvbt1OfdmModComponent.h.

### 7.10.2.6 `typedef std::vector<float> iris::phy::Dvbt1OfdmModComponent::FloatVec`

A vector of float.

Definition at line 160 of file Dvbt1OfdmModComponent.h.

### 7.10.2.7 typedef FloatVec::iterator iris::phy::Dvbt1OfdmModComponent::FloatVecIt

An iterator for a vector of float.

Definition at line 163 of file Dvbt1OfdmModComponent.h.

### 7.10.2.8 typedef std::vector<int> iris::phy::Dvbt1OfdmModComponent::IntVec

A vector of integers.

Definition at line 166 of file Dvbt1OfdmModComponent.h.

### 7.10.2.9 typedef IntVec::iterator iris::phy::Dvbt1OfdmModComponent::IntVecIt

An iterator for a vector of typedef.

Definition at line 169 of file Dvbt1OfdmModComponent.h.

## 7.10.3 Constructor & Destructor Documentation

### 7.10.3.1 iris::phy::Dvbt1OfdmModComponent::Dvbt1OfdmModComponent ( std::string name )

Default constructor.

Registers the block parameters and initializes some variables

Definition at line 59 of file Dvbt1OfdmModComponent.cpp.

References begin(), dacSampleRate\_x, debug\_x, deltaMode\_x, end(), ofdmMode\_x, outPower\_x, powerFile\_x, and powerInterval\_x.

```

60 : PhyComponent(name,                                // component name
61               "dvbt1ofdmmod",                        // component type
62               "A DVB-T1 OFDM modulator component",    // description
63               "Giuseppe Baruffa",                    // author
64               "0.1")                                  // version
65 , sampleRate_(0)
66 , timeStamp_(0)
67 , fft_(NULL)
68 , fftBins_(NULL)
69 , powerThread_(NULL)
70 , runPower_(false)
71 {
72     registerParameter(
73         "debug", "Whether to output debug data",
74         "false", true, debug_x);
75
76     int ofdmarr[] = {2048, 4096, 8192};
77     registerParameter(
78         "ofdmmode", "OFDM mode",
79         "2048", true, ofdmMode_x, list<int>(begin(ofdmarr), end(ofdmarr)));
80
81     int deltaarr[] = {32, 16, 8, 4};
82     registerParameter(
83         "deltamode", "Cyclic prefix ratio",
84         "32", true, deltaMode_x, list<int>(begin(deltaarr), end(deltaarr)));
85
86     registerParameter(
87         "outpower", "Output power, in percentage: note that a value of 100 will "
88         "result in signal clipping only below -3 sigma and above +3 sigma",
89         "10", true, outPower_x, Interval<float>(0, 300));
90
91     registerParameter(
92         "dacsamplerate", "Sampling rate at the DAC (default = 0, means 64e6/7)",
93         "0", true, dacSampleRate_x, Interval<double>(0, 15000000));
94
95     registerParameter(
96         "powerfile", "Text file with the power loading profile (default = none)",
97         "", true, powerFile_x);
98
99     registerParameter(
100        "powerinterval", "Power update interval in seconds (default = 1)",
101        "0", true, powerInterval_x);

```

```
102 }
```

### 7.10.3.2 iris::phy::Dvbt1OfdmModComponent::~~Dvbt1OfdmModComponent ( )

Default destructor.

Just calls [destroy\(\)](#).

Definition at line 107 of file Dvbt1OfdmModComponent.cpp.

References [destroy\(\)](#).

```
108 {
109     destroy();
110 }
```

## 7.10.4 Member Function Documentation

### 7.10.4.1 template<typename T, size\_t N> static T\* iris::phy::Dvbt1OfdmModComponent::begin ( T(&) arr[N] ) [inline], [static], [private]

Useful templates.

Definition at line 225 of file Dvbt1OfdmModComponent.h.

Referenced by [Dvbt1OfdmModComponent\(\)](#).

```
225 { return &arr[0]; }
```

### 7.10.4.2 double \* iris::phy::Dvbt1OfdmModComponent::blackman\_sinc ( int \* n\_order, double T, double dt, int order ) [private]

Calculate a Blackman-windowed sinc.

**Parameters**

<i>n_order</i>	order of the calculated window
<i>T</i>	time extension of the window
<i>dt</i>	sampling time
<i>order</i>	preferred order of the window

**Returns**

array containing the window taps, please remember to free when this is not needed anymore

Definition at line 274 of file Dvbt1OfdmModComponent.cpp.

References [sinc\(\)](#).

Referenced by [setup\(\)](#).

```
275 {
276     int n0 = (int) floor(T / dt);
277     int i;
278     double *h_order = NULL, w = 0.0;
279     double a0 = 7938.0 / 18608.0, a1 = 9240.0 / 18608.0, a2 = 1430.0 / 18608.0;
280     double accum = 0.0;
281     *n_order = (order + 1) * n0;
282     h_order = (double *) calloc(*n_order, sizeof(double));
283     for (i = 0; i < *n_order; i++) {
284         w = a0 - a1 * cos(2.0 * M_PI * i / (*n_order - 1)) + a2 * cos(4.0 * M_PI * i / (*n_order - 1));
285         h_order[i] = w * sinc(M_PI * (i - *n_order / 2) * dt / T);
286         accum += h_order[i] * dt;
```

```

287     }
288     /*for (i = 0; i < *n_order; i++)
289         h_order[i] /= accum;*/
290
291     return h_order;
292 }

```

**7.10.4.3** `void iris::phy::Dvbt1OfdmModComponent::calculateOutputTypes ( std::map< std::string, int > & inputTypes, std::map< std::string, int > & outputTypes ) [virtual]`

Calculate the output port types for the IRIS system.

The single output port must provide complex values.

Definition at line 125 of file Dvbt1OfdmModComponent.cpp.

```

128 {
129     outputTypes["output1"] = TypeInfo< Cplx >::identifier;
130 }

```

**7.10.4.4** `void iris::phy::Dvbt1OfdmModComponent::destroy ( ) [private]`

Destroy the component.

Definition at line 446 of file Dvbt1OfdmModComponent.cpp.

References `fft_`, `fftBins_`, `powerThread_`, and `runPower_`.

Referenced by `parameterHasChanged()`, and `~Dvbt1OfdmModComponent()`.

```

447 {
448     if(fftBins_ != NULL)
449         fftwf_free(fftBins_);
450     if(fft_ != NULL)
451         fftwf_destroy_plan(fft_);
452
453     // stop thread
454     runPower_ = false;
455     if (powerThread_) {
456         powerThread_>join();
457         delete powerThread_;
458     }
459 }

```

**7.10.4.5** `template<typename T, size_t N> static T* iris::phy::Dvbt1OfdmModComponent::end ( T(&) arr[N] ) [inline], [static], [private]`

Definition at line 227 of file Dvbt1OfdmModComponent.h.

Referenced by `Dvbt1OfdmModComponent()`.

```

227 { return &arr[0]+N; }

```

**7.10.4.6** `double iris::phy::Dvbt1OfdmModComponent::frequency_response_modulus ( double * h, int n, double dt, double f ) [private]`

Calculate the frequency response modulus of an impulse response.

**Parameters**

$h$	array of impulse response taps
$n$	number of taps
$dt$	sampling period of the impulse response
$f$	frequency at which the modulus of the frequency response is calculated

**Returns**

the modulus of the frequency response at the indicated frequency

Definition at line 251 of file Dvbt1OfdmModComponent.cpp.

Referenced by `setup()`.

```

252 {
253     double H_re = 0.0, H_im = 0.0;
254     double arg = 2.0 * M_PI * f * dt;
255     int i;
256
257     // just use plain old DFT, no FFT here
258     for (i = 0; i < n; i++) {
259         H_re += h[i] * cos(arg * i) * dt;
260         H_im += h[i] * (-sin(arg * i)) * dt;
261     }
262
263     return sqrt(H_re * H_re + H_im * H_im);
264 }
```

**7.10.4.7 void iris::phy::Dvbt1OfdmModComponent::initialize ( ) [virtual]**

Initialize the component.

Just calls `setup()`.

Definition at line 135 of file Dvbt1OfdmModComponent.cpp.

References `setup()`.

```

136 {
137     setup();
138 }
```

**7.10.4.8 void iris::phy::Dvbt1OfdmModComponent::parameterHasChanged ( std::string name ) [virtual]**

Actions taken when the parameters change.

This block has several significant parameters

Definition at line 225 of file Dvbt1OfdmModComponent.cpp.

References `destroy()`, and `setup()`.

```

226 {
227     if(name == "deltamode" || name == "ofdmmode" ||
228         name == "outpower")
229     {
230         destroy();
231         setup();
232     }
233 }
```

**7.10.4.9 void iris::phy::Dvbt1OfdmModComponent::powerProcedure\_ ( ) [private]**

Separate thread for power loading.

Definition at line 405 of file Dvbt1OfdmModComponent.cpp.

References `_ampliFactor_`, `nFft_`, `powerFile_x`, `powerInterval_x`, `runPower_`, and `WAKEUPINTERVALMS`.

Referenced by `setup()`.

```

406 {
407     int currentTick = 0, nextTick = currentTick;
408
409     while(runPower_)
410     {
411         if(currentTick == nextTick)
412         {
413             // advance ticks
414             nextTick = currentTick + 1000 * powerInterval_x /
WAKEUPINTERVALMS;
415
416             // open the file
417             std::ifstream myfile(powerFile_x.c_str());
418             if(myfile)
419             {
420                 // read it line after line
421                 std::string line;
422                 int l = 0;
423                 while (std::getline(myfile, line) && l < nFft_)
424                 {
425                     // parse the line and get the dB, then convert it to linear
426                     float val = 0.0f;
427                     std::istringstream istr(line);
428                     istr.imbue(std::locale("C"));
429                     istr >> val;
430                     _ampliFactor_[l++] = powf(10.0F, val / 20.0F);
431                 }
432             }
433             else
434             {
435                 LOG(LINFO) << "Power loading file '" << powerFile_x << "' not found";
436             }
437         }
438
439         // this is to give responsivity
440         boost::this_thread::sleep(boost::posix_time::milliseconds(WAKEUPINTERVALMS));
441         currentTick++;
442     }
443 }

```

#### 7.10.4.10 void iris::phy::Dvbt1OfdmModComponent::process ( ) [virtual]

Main processing method.

Definition at line 141 of file Dvbt1OfdmModComponent.cpp.

References `ampliFactor_`, `debug_x`, `fft_`, `fftBins_`, `inOffset_`, `inReg_`, `kMax_`, `multFactor_`, `nBlock_`, `nDelta_`, `nFft_`, `precorrFactor_`, `sampleRate_`, and `timeStamp_`.

```

142 {
143     // request input
144     DataSet< Cplx > *in = NULL;
145     getInputDataSet("input1", in);
146
147     // calculate sizes
148     int insize = in ? (int) in->data.size() : 0;
149     int outsize = nBlock_ * ((insize + inOffset_) / kMax_);
150
151     // request output
152     DataSet< Cplx > *out = NULL;
153     getOutputDataSet("output1", out, outsize);
154
155     // print debug info
156     if(debug_x)
157         LOG(LINFO) << "in/out: " << insize << "/" << outsize;
158
159     // fill the input register
160     CplxVecIt outit = out->data.begin();
161     for(CplxVecIt init = in->data.begin(); init < in->data.end(); init++)
162     {
163         // copy datum
164         inReg_[inOffset_++] = *init;
165
166         // trigger IFFT

```

```

167     if(inOffset_ == kMax_)
168     {
169         int num_pos = kMax_ / 2 + 1;
170         int num_neg = num_pos - 1;
171         int neg_start = nFft_ - num_neg;
172
173         // reset offset
174         inOffset_ = 0;
175
176         // copy positive frequencies
177         for(int i = 0; i < num_pos; i++)
178         {
179             fftBins_[i] = inReg_[num_neg + i] * precorrFactor_[i] *
ampliFactor_[i];
180         }
181
182         // copy negative frequencies
183         for(int i = 0; i < num_neg; i++)
184         {
185             fftBins_[neg_start + i] = inReg_[i] * precorrFactor_[-num_neg + i] *
ampliFactor_[-num_neg + i];
186         }
187
188         // set null frequencies
189         for(int i = 0; i < nFft_ - kMax_; i++)
190         {
191             fftBins_[num_pos + i] = Cplx(0,0);
192         }
193
194         // call FFTW
195         fftwf_execute(fft_);
196
197         // apply multiplicative factor, for the power
198         for(int i = 0; i < nFft_; i++)
199         {
200             fftBins_[i].real(fftBins_[i].real() * multFactor_);
201             fftBins_[i].imag(fftBins_[i].imag() * multFactor_);
202         }
203
204         // copy to output
205         CplxVecIt it = copy(&fftBins_[nFft_ - nDelta_], &
fftBins_[nFft_], outit);
206         copy(&fftBins_[0], &fftBins_[nFft_], it);
207
208         outit += nBlock_;
209     }
210 }
211
212 //set the timestamp and sample rate for the DataSets
213 out->timeStamp = timeStamp_;
214 out->sampleRate = sampleRate_;
215 timeStamp_ += (double) outsize / sampleRate_;
216
217 // release input and output
218 releaseInputDataSet("input1", in);
219 releaseOutputDataSet("output1", out);
220 }

```

#### 7.10.4.11 void iris::phy::Dvbt1OfdmModComponent::registerPorts ( ) [virtual]

Register the mapper ports with the IRIS system.

This component has one input that accept complex float values and one output that provides complex float values.

Definition at line 116 of file Dvbt1OfdmModComponent.cpp.

```

117 {
118     registerInputPort("input1", TypeInfo< Cplx >::identifier);
119     registerOutputPort("output1", TypeInfo< Cplx >::identifier);
120 }

```

#### 7.10.4.12 void iris::phy::Dvbt1OfdmModComponent::setup ( ) [private]

Set up all needed constants.

Definition at line 295 of file Dvbt1OfdmModComponent.cpp.



References `_ampliFactor_`, `_precorrFactor_`, `ampliFactor_`, `blackman_sinc()`, `dacSampleRate_x`, `deltaMode_x`, `fft_`, `fftBins_`, `fftReg_`, `frequency_response_modulus()`, `inOffset_`, `inReg_`, `kMax_`, `multFactor_`, `nBit_`, `nBlock_`, `nDelta_`, `nFft_`, `nMax_`, `ofdmMode_x`, `outPower_x`, `powerFile_x`, `powerProcedure_()`, `powerThread_`, `precorrFactor_`, `runPower_`, `sampleRate_`, `T1_RESAMPLE_ORDER`, `timeStamp_`, and `tpsNum_`.

Referenced by `initialize()`, and `parameterHasChanged()`.

```

296 {
297     if(dacSampleRate_x == 0)
298         dacSampleRate_x = 64.0e6/7.0;
299     sampleRate_ = 64.0e6/7;
300     timeStamp_ = 0;
301     // clean registers
302     switch(ofdmMode_x)
303     {
304         case 2048:
305             tpsNum_ = 17;
306             nMax_ = 1512;
307             kMax_ = 1705;
308             nBit_ = 11;
309             nFft_ = 2048;
310             break;
311         case 4096:
312             tpsNum_ = 34;
313             nMax_ = 3024;
314             kMax_ = 3409;
315             nBit_ = 12;
316             nFft_ = 4096;
317             break;
318         case 8192:
319             tpsNum_ = 68;
320             nMax_ = 6048;
321             kMax_ = 6817;
322             nBit_ = 13;
323             nFft_ = 8192;
324             break;
325     }
326     nDelta_ = (int) floor((double) nFft_ / (double) deltaMode_x);
327     nBlock_ = nFft_ + nDelta_;
328     inOffset_ = 0;
329     inReg_.resize(kMax_);
330     fftReg_.resize(nFft_);
331
332     int num_pos = kMax_ / 2 + 1;
333     int num_neg = num_pos - 1;
334     float temp = 0.0F;
335
336     // multiplicative factor
337     float power = (1.0F * (float) nMax_ /* data carriers */
338         + (16.0F / 9.0F) * (float) (kMax_ - nMax_ - tpsNum_) /* pilot carriers */
339         + 1.0F * (float) tpsNum_ /* tps carriers */)
340         / (float) nFft_;
341     multFactor_ = (float) sqrt((outPower_x / 100.0F) / (power * (float)
342         nFft_)) /
343         3.0F;
344
345     // linear precorrection
346     double dtbase = (1 / (64.0e6/7.0)) / 100.0;
347     int nbase = 0;
348     double *hbase = blackman_sinc(&nbase, 1 / (64.0e6/7.0), dtbase,
349         T1_RESAMPLE_ORDER);
350     _precorrFactor_.resize(nFft_);
351     precorrFactor_ = _precorrFactor_.begin() + nFft_ / 2;
352     for(int i = -num_neg; i < num_pos; i++)
353     {
354         //printf("dacSampleRate_x = %f\n", dacSampleRate_x);
355         if(dacSampleRate_x == 64.0e6/7.0)
356         {
357             precorrFactor_[i] = 1.0F; // no precorrection
358         }
359         else
360         {
361             precorrFactor_[i] = (float) (1.0 /
362                 frequency_response_modulus(hbase,
363                     nbase, dtbase, (double) i * (64.0e6/7.0) / (double) nFft_));
364
365             if (i == 0)
366                 temp = precorrFactor_[i];
367         }
368     }
369
370     // normalize
371     for(int i = -num_neg; i < num_pos; i++)
372         precorrFactor_[i] /= temp;
373     free(hbase);
374
375     // amplitude power loading factor

```

```

371  _ampliFactor_.resize(nFft_);
372  ampliFactor_ = _ampliFactor_.begin() + nFft_ / 2;
373  for(int i = -num_neg; i < num_pos; i++)
374      ampliFactor_[i] = 1.0F; // default powerloading
375  if(!powerFile_x.empty())
376  {
377      // stop in case it's running
378      if(powerThread_)
379      {
380          runPower_ = false;
381          powerThread_>join();
382          delete powerThread_;
383      }
384
385      // start thread
386      runPower_ = true;
387      powerThread_ = new boost::thread(boost::bind(&
Dvbt1OfdmModComponent::powerProcedure_, this));
388  }
389
390  // Set up containers for FFTW
391  fftBins_ = reinterpret_cast<Cplx*>(
392      fftwf_malloc(sizeof(fftwf_complex) * ofdmMode_x));
393  fill(&fftBins_[0], &fftBins_[ofdmMode_x], Cplx(0,0));
394  fft_ = fftwf_plan_dft_1d(ofdmMode_x,
395                          (fftwf_complex*)fftBins_,
396                          (fftwf_complex*)fftBins_,
397                          FFTW_BACKWARD,
398                          FFTW_MEASURE);
399
400 }

```

#### 7.10.4.13 double iris::phy::Dvbt1OfdmModComponent::sinc ( double x ) [private]

$\sin(x)/x$  function

Parameters

x	Input value
---	-------------

Returns

The sinc of the input

Definition at line 239 of file Dvbt1OfdmModComponent.cpp.

Referenced by blackman\_sinc().

```

240 {
241     return x == 0.0 ? 1.0 : (sin(x) / x);
242 }

```

## 7.10.5 Member Data Documentation

### 7.10.5.1 FloatVec iris::phy::Dvbt1OfdmModComponent::\_ampliFactor\_ [private]

Definition at line 210 of file Dvbt1OfdmModComponent.h.

Referenced by powerProcedure\_(), and setup().

### 7.10.5.2 FloatVec iris::phy::Dvbt1OfdmModComponent::\_precorrFactor\_ [private]

Definition at line 208 of file Dvbt1OfdmModComponent.h.

Referenced by setup().

#### 7.10.5.3 FloatVecIt iris::phy::Dvbt1OfdmModComponent::ampliFactor\_ [private]

Definition at line 211 of file Dvbt1OfdmModComponent.h.

Referenced by process(), and setup().

#### 7.10.5.4 double iris::phy::Dvbt1OfdmModComponent::dacSampleRate\_x [private]

Sampling rate used by the DAC.

Definition at line 187 of file Dvbt1OfdmModComponent.h.

Referenced by Dvbt1OfdmModComponent(), and setup().

#### 7.10.5.5 bool iris::phy::Dvbt1OfdmModComponent::debug\_x [private]

Debug flag (default = false)

Definition at line 183 of file Dvbt1OfdmModComponent.h.

Referenced by Dvbt1OfdmModComponent(), and process().

#### 7.10.5.6 int iris::phy::Dvbt1OfdmModComponent::deltaMode\_x [private]

Cyclic prefix ratio (default = 32)

Definition at line 185 of file Dvbt1OfdmModComponent.h.

Referenced by Dvbt1OfdmModComponent(), and setup().

#### 7.10.5.7 fftwf\_plan iris::phy::Dvbt1OfdmModComponent::fft\_ [private]

Our FFT object pointer.

Definition at line 212 of file Dvbt1OfdmModComponent.h.

Referenced by destroy(), process(), and setup().

#### 7.10.5.8 Cplx\* iris::phy::Dvbt1OfdmModComponent::fftBins\_ [private]

Allocated using fftwf\_malloc (SIMD aligned)

Definition at line 213 of file Dvbt1OfdmModComponent.h.

Referenced by destroy(), process(), and setup().

#### 7.10.5.9 CplxVec iris::phy::Dvbt1OfdmModComponent::fftReg\_ [private]

Definition at line 202 of file Dvbt1OfdmModComponent.h.

Referenced by setup().

#### 7.10.5.10 int iris::phy::Dvbt1OfdmModComponent::inOffset\_ [private]

Definition at line 200 of file Dvbt1OfdmModComponent.h.

Referenced by process(), and setup().

**7.10.5.11 CplxVec iris::phy::Dvbt1OfdmModComponent::inReg\_ [private]**

Definition at line 201 of file Dvbt1OfdmModComponent.h.

Referenced by process(), and setup().

**7.10.5.12 int iris::phy::Dvbt1OfdmModComponent::kMax\_ [private]**

Definition at line 204 of file Dvbt1OfdmModComponent.h.

Referenced by process(), and setup().

**7.10.5.13 float iris::phy::Dvbt1OfdmModComponent::multFactor\_ [private]**

Definition at line 207 of file Dvbt1OfdmModComponent.h.

Referenced by process(), and setup().

**7.10.5.14 int iris::phy::Dvbt1OfdmModComponent::nBit\_ [private]**

Definition at line 206 of file Dvbt1OfdmModComponent.h.

Referenced by setup().

**7.10.5.15 int iris::phy::Dvbt1OfdmModComponent::nBlock\_ [private]**

Definition at line 199 of file Dvbt1OfdmModComponent.h.

Referenced by process(), and setup().

**7.10.5.16 int iris::phy::Dvbt1OfdmModComponent::nDelta\_ [private]**

Definition at line 198 of file Dvbt1OfdmModComponent.h.

Referenced by process(), and setup().

**7.10.5.17 int iris::phy::Dvbt1OfdmModComponent::nFft\_ [private]**

Definition at line 197 of file Dvbt1OfdmModComponent.h.

Referenced by powerProcedure\_(), process(), and setup().

**7.10.5.18 int iris::phy::Dvbt1OfdmModComponent::nMax\_ [private]**

Definition at line 203 of file Dvbt1OfdmModComponent.h.

Referenced by setup().

**7.10.5.19 int iris::phy::Dvbt1OfdmModComponent::ofdmMode\_x [private]**

OFDM mode (default = 2048)

Definition at line 184 of file Dvbt1OfdmModComponent.h.

Referenced by Dvbt1OfdmModComponent(), and setup().

**7.10.5.20** `float iris::phy::Dvbt1OfdmModComponent::outPower_x` `[private]`

Output power indicator (default = 10)

Definition at line 186 of file Dvbt1OfdmModComponent.h.

Referenced by Dvbt1OfdmModComponent(), and setup().

**7.10.5.21** `std::string iris::phy::Dvbt1OfdmModComponent::powerFile_x` `[private]`

Text file with power loading (default = none)

Definition at line 188 of file Dvbt1OfdmModComponent.h.

Referenced by Dvbt1OfdmModComponent(), powerProcedure\_(), and setup().

**7.10.5.22** `double iris::phy::Dvbt1OfdmModComponent::powerInterval_x` `[private]`

Power update interval in seconds (default = 0)

Definition at line 189 of file Dvbt1OfdmModComponent.h.

Referenced by Dvbt1OfdmModComponent(), and powerProcedure\_().

**7.10.5.23** `boost::thread* iris::phy::Dvbt1OfdmModComponent::powerThread_` `[private]`

Definition at line 216 of file Dvbt1OfdmModComponent.h.

Referenced by destroy(), and setup().

**7.10.5.24** `FloatVecIt iris::phy::Dvbt1OfdmModComponent::precorrFactor_` `[private]`

Definition at line 209 of file Dvbt1OfdmModComponent.h.

Referenced by process(), and setup().

**7.10.5.25** `bool iris::phy::Dvbt1OfdmModComponent::runPower_` `[private]`

Definition at line 217 of file Dvbt1OfdmModComponent.h.

Referenced by destroy(), powerProcedure\_(), and setup().

**7.10.5.26** `double iris::phy::Dvbt1OfdmModComponent::sampleRate_` `[private]`

Sample rate of current frame.

Definition at line 195 of file Dvbt1OfdmModComponent.h.

Referenced by process(), and setup().

**7.10.5.27** `double iris::phy::Dvbt1OfdmModComponent::timeStamp_` `[private]`

Timestamp of current frame.

Definition at line 194 of file Dvbt1OfdmModComponent.h.

Referenced by process(), and setup().

7.10.5.28 `int iris::phy::Dvbt1OfdmModComponent::tpsNum_ [private]`

Definition at line 205 of file `Dvbt1OfdmModComponent.h`.

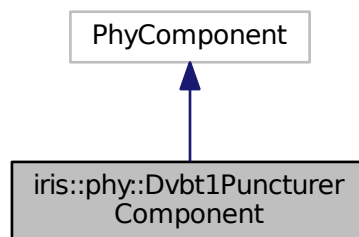
Referenced by `setup()`.

## 7.11 `iris::phy::Dvbt1PuncturerComponent` Class Reference

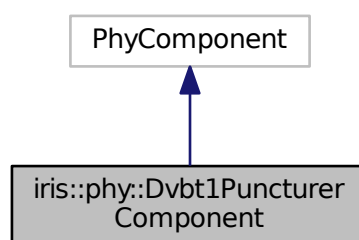
A DVB-T1 puncturer component.

```
#include <Dvbt1PuncturerComponent.h>
```

Inheritance diagram for `iris::phy::Dvbt1PuncturerComponent`:



Collaboration diagram for `iris::phy::Dvbt1PuncturerComponent`:



### Public Types

- `typedef std::vector< uint8_t > ByteVec`  
*A vector of bytes.*
- `typedef ByteVec::iterator ByteVecIt`  
*An iterator for a vector of bytes.*

## Public Member Functions

- [Dvbt1PuncturerComponent](#) (std::string name)  
*Default constructor.*
- [~Dvbt1PuncturerComponent](#) ()  
*Default destructor.*
- virtual void [calculateOutputTypes](#) (std::map< std::string, int > &inputTypes, std::map< std::string, int > &outputTypes)  
*Calculate the output port types for the IRIS system.*
- virtual void [registerPorts](#) ()  
*Register the puncturer ports with the IRIS system.*
- virtual void [initialize](#) ()  
*Initialize the component.*
- virtual void [process](#) ()  
*Main processing method.*
- virtual void [parameterHasChanged](#) (std::string name)  
*Actions taken when the parameters change.*

## Private Member Functions

- void [setup](#) ()  
*Set up all puncturing basic sizes, reset offsets, clean registers.*
- void [destroy](#) ()  
*Destroy the component.*

## Static Private Member Functions

- template<typename T , size\_t N>  
static T \* [begin](#) (T(&arr)[N])  
*Useful templates.*
- template<typename T , size\_t N>  
static T \* [end](#) (T(&arr)[N])

## Private Attributes

- bool [debug\\_x](#)  
*Debug flag (default = false)*
- int [codeRate\\_x](#)  
*stream channel coding rate (default = 34)*
- double [timeStamp\\_](#)  
*Timestamp of current frame.*
- double [sampleRate\\_](#)  
*Sample rate of current frame.*
- int [punOffset\\_](#)  
*Puncturing offset.*
- uint8\_t [punRegister\\_](#) [14]  
*Puncturing register (statically set to the maximum expected size)*
- int [punPeriodIn\\_](#)
- int [punPeriodOut\\_](#)  
*Input and output puncturing periods.*

### 7.11.1 Detailed Description

A DVB-T1 puncturer component.

[Dvbt1PuncturerComponent](#) is the fifth block composing the DVB-T transmission chain. The purpose of the puncturer is that of achieving variable coding rate keeping fixed the properties and complexity of the main *mother* convolutional code, which sticks at a rate of  $k/n = 1/2$ . By properly removing convolutional encoded bits before transmission, one can still expect to take profit of the error correction capabilities of the convolutional decoder (Viterbi algorithm) at the receiving side, although having the possibility to change the overall coding rate to one of the values  $r_c = 1/2, 2/3, 3/4, 5/6, 7/8$ . The block operates by translating the puncturing matrices, that are given in the standard, into a periodic subsampling structure (a sort of nonequispaced bit decimation). Thus, a group of input bits (at the input periodicity) are read into a register, which is dumped into another shorter register, which in turn is read out (at the output periodicity).

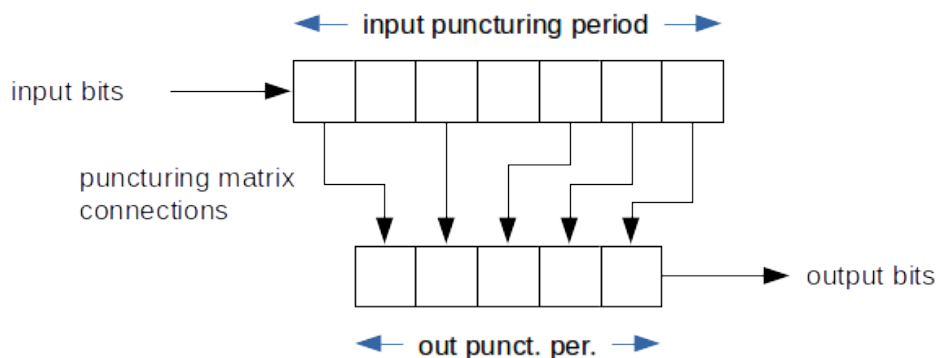


Figure 7.8: DVB-T puncturer.

This block accepts in input elements in `uint8_t` (bits) and generates in output bits (`uint8_t`).

There are two parameters that can be changed in the XML configuration file:

- *debug*: by default set to "false", is used to print some small debugging information for the interested developer.
- *coderate*: by default set to "34", this is used to select one of the five possible coding rates. The admitted values are "12", "23", "34", "56", and "78", which are easily recognizable as the real coding ratios written without the separating slash.

### References

- ETSI Standard: *EN 300 744 V1.5.1, Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for digital terrestrial television*, available at [ETSI Publications Download Area](#)
- S. Li, D. J. Costello, *Error Control Coding, Second Edition*, Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 2004

Definition at line 84 of file `Dvbt1PuncturerComponent.h`.

### 7.11.2 Member Typedef Documentation

#### 7.11.2.1 `typedef std::vector<uint8_t> iris::phy::Dvbt1PuncturerComponent::ByteVec`

A vector of bytes.

Definition at line 90 of file `Dvbt1PuncturerComponent.h`.



### 7.11.2.2 typedef ByteVec::iterator iris::phy::Dvbt1PuncturerComponent::ByteVecIt

An iterator for a vector of bytes.

Definition at line 93 of file Dvbt1PuncturerComponent.h.

## 7.11.3 Constructor & Destructor Documentation

### 7.11.3.1 iris::phy::Dvbt1PuncturerComponent::Dvbt1PuncturerComponent ( std::string *name* )

Default constructor.

Registers the block parameters and initializes some variables

Definition at line 57 of file Dvbt1PuncturerComponent.cpp.

References `begin()`, `codeRate_x`, `debug_x`, and `end()`.

```

58 : PhyComponent (name,                                // component name
59                 "dvbt1puncturer",                    // component type
60                 "A DVB-T1 puncturer component",        // description
61                 "Giuseppe Baruffa",                    // author
62                 "0.1")                                // version
63 , sampleRate_(0)
64 , timeStamp_(0)
65 , punOffset_(0)
66 {
67     registerParameter(
68         "debug", "Whether to output debug data",
69         "false", true, debug_x);
70
71     int codearr[] = {12,23,34,56,78};
72     registerParameter(
73         "coderate", "Channel coding rate",
74         "34", true, codeRate_x, list<int>(begin(codearr),end(codearr)));
75 }
```

### 7.11.3.2 iris::phy::Dvbt1PuncturerComponent::~~Dvbt1PuncturerComponent ( )

Default destructor.

Just calls `destroy()`.

Definition at line 80 of file Dvbt1PuncturerComponent.cpp.

References `destroy()`.

```

81 {
82     destroy();
83 }
```

## 7.11.4 Member Function Documentation

### 7.11.4.1 template<typename T, size\_t N> static T\* iris::phy::Dvbt1PuncturerComponent::begin ( T(&) arr[N] ) [inline], [static], [private]

Useful templates.

Definition at line 123 of file Dvbt1PuncturerComponent.h.

Referenced by `Dvbt1PuncturerComponent()`.

```

123 { return &arr[0]; }
```

**7.11.4.2** `void iris::phy::Dvbt1PuncturerComponent::calculateOutputTypes ( std::map< std::string, int > & inputTypes,  
std::map< std::string, int > & outputTypes ) [virtual]`

Calculate the output port types for the IRIS system.

The single output port must provide bytes.

Definition at line 98 of file Dvbt1PuncturerComponent.cpp.

```
101 {
102     outputTypes["output1"] = TypeInfo< uint8_t >::identifier;
103 }
```

**7.11.4.3** `void iris::phy::Dvbt1PuncturerComponent::destroy ( ) [private]`

Destroy the component.

Definition at line 241 of file Dvbt1PuncturerComponent.cpp.

Referenced by parameterHasChanged(), and ~Dvbt1PuncturerComponent().

```
242 {
243 }
```

**7.11.4.4** `template<typename T, size_t N> static T* iris::phy::Dvbt1PuncturerComponent::end ( T(&) arr[N] ) [inline],  
[static], [private]`

Definition at line 125 of file Dvbt1PuncturerComponent.h.

Referenced by Dvbt1PuncturerComponent().

```
125 { return &arr[0]+N; }
```

**7.11.4.5** `void iris::phy::Dvbt1PuncturerComponent::initialize ( ) [virtual]`

Initialize the component.

Just calls [setup\(\)](#).

Definition at line 108 of file Dvbt1PuncturerComponent.cpp.

References [setup\(\)](#).

```
109 {
110     setup();
111 }
```

**7.11.4.6** `void iris::phy::Dvbt1PuncturerComponent::parameterHasChanged ( std::string name ) [virtual]`

Actions taken when the parameters change.

This block has one significant parameters

Definition at line 199 of file Dvbt1PuncturerComponent.cpp.

References [destroy\(\)](#), and [setup\(\)](#).

```
200 {
201     if(name == "coderate")
202     {
203         destroy();
204         setup();
205     }
206 }
```

## 7.11.4.7 void iris::phy::Dvbt1PuncturerComponent::process ( ) [virtual]

Main processing method.

Definition at line 114 of file Dvbt1PuncturerComponent.cpp.

References `codeRate_x`, `debug_x`, `punOffset_`, `punPeriodIn_`, `punPeriodOut_`, and `punRegister_`.

```

115 {
116     // request input
117     DataSet< uint8_t >* in = NULL;
118     getInputDataSet("input1", in);
119
120     // calculate sizes
121     int insize = in ? (int) in->data.size() : 0;
122     int outsize = ((insize + punOffset_) / punPeriodIn_) *
punPeriodOut_;
123
124     // request output
125     DataSet< uint8_t >* out = NULL;
126     getOutputDataSet("output1", out, outsize);
127
128     // print debug info
129     if(debug_x)
130         LOG(LINFO) << "in/out: " << insize + punOffset_ << "(" << insize << "+" <<
punOffset_ << ")/" << outsize;
131
132     // iterate over input
133     for(ByteVecIt init = in->data.begin(), outit = out->data.begin(); init < in->data.end(); init++)
134     {
135         // fill puncturing register
136         punRegister_[punOffset_++] = *init;
137
138         // trigger puncturing at the output
139         if(punOffset_ == punPeriodIn_)
140         {
141             // reset offset
142             punOffset_ = 0;
143
144             // copy to output
145             switch(codeRate_x)
146             {
147                 // the puncturing matrices are hard-coded for all the five code rates
148                 case 12:
149                     *outit++ = punRegister_[0];
150                     *outit++ = punRegister_[1];
151                     break;
152                 case 23:
153                     *outit++ = punRegister_[0];
154                     *outit++ = punRegister_[1];
155                     *outit++ = punRegister_[3];
156                     break;
157                 case 34:
158                     *outit++ = punRegister_[0];
159                     *outit++ = punRegister_[1];
160                     *outit++ = punRegister_[3];
161                     *outit++ = punRegister_[4];
162                     break;
163                 case 56:
164                     *outit++ = punRegister_[0];
165                     *outit++ = punRegister_[1];
166                     *outit++ = punRegister_[3];
167                     *outit++ = punRegister_[4];
168                     *outit++ = punRegister_[7];
169                     *outit++ = punRegister_[8];
170                     break;
171                 case 78:
172                     *outit++ = punRegister_[0];
173                     *outit++ = punRegister_[1];
174                     *outit++ = punRegister_[3];
175                     *outit++ = punRegister_[5];
176                     *outit++ = punRegister_[7];
177                     *outit++ = punRegister_[8];
178                     *outit++ = punRegister_[11];
179                     *outit++ = punRegister_[12];
180                     break;
181                 default:
182                     LOG(LEERROR) << "Invalid puncturing rate: " << codeRate_x;
183             }
184         }
185     }
186
187     // Copy the timestamp and sample rate for the DataSets
188     out->timeStamp = in->timeStamp;
189     out->sampleRate = in->sampleRate;

```

```

190
191 // release input and output
192 releaseInputDataSet("input1", in);
193 releaseOutputDataSet("output1", out);
194 }

```

#### 7.11.4.8 void iris::phy::Dvbt1PuncturerComponent::registerPorts ( ) [virtual]

Register the puncturer ports with the IRIS system.

This component has one input that accepts bits (one bit per byte) and one output that provides punctured bits (one bit per byte).

Definition at line 89 of file Dvbt1PuncturerComponent.cpp.

```

90 {
91     registerInputPort("input1", TypeInfo< uint8_t >::identifier);
92     registerOutputPort("output1", TypeInfo< uint8_t >::identifier);
93 }

```

#### 7.11.4.9 void iris::phy::Dvbt1PuncturerComponent::setup ( ) [private]

Set up all puncturing basic sizes, reset offsets, clean registers.

Definition at line 209 of file Dvbt1PuncturerComponent.cpp.

References codeRate\_x, punOffset\_, punPeriodIn\_, punPeriodOut\_, and punRegister\_.

Referenced by initialize(), and parameterHasChanged().

```

210 {
211     switch(codeRate_x)
212     {
213         case 12:
214             punPeriodIn_ = 2;
215             punPeriodOut_ = 2;
216             break;
217         case 23:
218             punPeriodIn_ = 4;
219             punPeriodOut_ = 3;
220             break;
221         case 34:
222             punPeriodIn_ = 6;
223             punPeriodOut_ = 4;
224             break;
225         case 56:
226             punPeriodIn_ = 10;
227             punPeriodOut_ = 6;
228             break;
229         case 78:
230             punPeriodIn_ = 14;
231             punPeriodOut_ = 8;
232             break;
233         default:
234             LOG(LERROR) << "Invalid puncturing rate: " << codeRate_x;
235     }
236     punOffset_ = 0;
237     memset(punRegister_, 0, sizeof(punRegister_));
238 }

```

### 7.11.5 Member Data Documentation

#### 7.11.5.1 int iris::phy::Dvbt1PuncturerComponent::codeRate\_x [private]

stream channel coding rate (default = 34)

Definition at line 108 of file Dvbt1PuncturerComponent.h.

Referenced by Dvbt1PuncturerComponent(), process(), and setup().

**7.11.5.2** `bool iris::phy::Dvbt1PuncturerComponent::debug_x` `[private]`

Debug flag (default = false)

Definition at line 107 of file Dvbt1PuncturerComponent.h.

Referenced by Dvbt1PuncturerComponent(), and process().

**7.11.5.3** `int iris::phy::Dvbt1PuncturerComponent::punOffset_` `[private]`

Puncturing offset.

Definition at line 116 of file Dvbt1PuncturerComponent.h.

Referenced by process(), and setup().

**7.11.5.4** `int iris::phy::Dvbt1PuncturerComponent::punPeriodIn_` `[private]`

Definition at line 119 of file Dvbt1PuncturerComponent.h.

Referenced by process(), and setup().

**7.11.5.5** `int iris::phy::Dvbt1PuncturerComponent::punPeriodOut_` `[private]`

Input and output puncturing periods.

Definition at line 119 of file Dvbt1PuncturerComponent.h.

Referenced by process(), and setup().

**7.11.5.6** `uint8_t iris::phy::Dvbt1PuncturerComponent::punRegister_[14]` `[private]`

Puncturing register (statically set to the maximum expected size)

Definition at line 117 of file Dvbt1PuncturerComponent.h.

Referenced by process(), and setup().

**7.11.5.7** `double iris::phy::Dvbt1PuncturerComponent::sampleRate_` `[private]`

Sample rate of current frame.

Definition at line 114 of file Dvbt1PuncturerComponent.h.

**7.11.5.8** `double iris::phy::Dvbt1PuncturerComponent::timeStamp_` `[private]`

Timestamp of current frame.

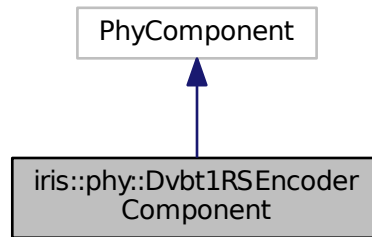
Definition at line 113 of file Dvbt1PuncturerComponent.h.

**7.12** `iris::phy::Dvbt1RSEncoderComponent` Class Reference

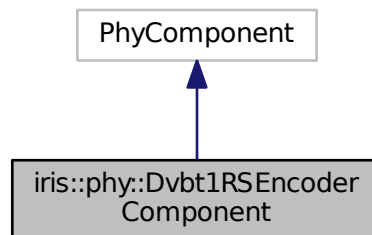
A DVB-T1 R-S Encoder component.

```
#include <Dvbt1RSEncoderComponent.h>
```

Inheritance diagram for `iris::phy::Dvbt1REncoderComponent`:



Collaboration diagram for `iris::phy::Dvbt1REncoderComponent`:



## Public Types

- typedef `std::vector< uint8_t >` [ByteVec](#)  
*A vector of bytes.*
- typedef `ByteVec::iterator` [ByteVecIt](#)  
*An iterator for a vector of bytes.*

## Public Member Functions

- [Dvbt1REncoderComponent](#) (`std::string name`)  
*Default constructor.*
- [~Dvbt1REncoderComponent](#) ()  
*Default destructor.*
- virtual void [calculateOutputTypes](#) (`std::map< std::string, int > &inputTypes, std::map< std::string, int > &outputTypes`)  
*Calculate the output port types for the IRIS system.*
- virtual void [registerPorts](#) ()  
*Register the encoder ports with the IRIS system.*

- virtual void `initialize` ()  
*Initialize the component.*
- virtual void `process` ()  
*Main processing method.*
- virtual void `parameterHasChanged` (std::string name)  
*Actions taken when the parameters change.*

### Private Member Functions

- void `setup` ()  
*Set up offsets and clean variables.*
- void `destroy` ()  
*Destroy the component.*
- int `packetEncode` (unsigned char \*data, unsigned char \*bb)  
*Encodes a single data packet.*
- int `modnn` (int x)  
*Computes the modulo-255 of a number.*

### Static Private Member Functions

- template<typename T, size\_t N>  
static T \* `begin` (T(&arr)[N])  
*Useful templates.*
- template<typename T, size\_t N>  
static T \* `end` (T(&arr)[N])

### Private Attributes

- bool `debug_x`  
*Debug flag (default = false)*
- double `timeStamp_`  
*Timestamp of current frame.*
- double `sampleRate_`  
*Sample rate of current frame.*
- uint8\_t `rsCodeWord_` [T1\_NN]  
*Nonshortened codeword.*
- int `tsOffset_`  
*Current offset in TS input.*

### Static Private Attributes

- static int `index_` [256]  
*LUT containing the base  $\alpha$  logarithm of the field elements.*
- static int `alpha_` [256]  
*LUT containing the powers of  $\alpha$ .*
- static int `gg_` [17]  
*R-S code generator polynomial.*

### 7.12.1 Detailed Description

A DVB-T1 R-S Encoder component.

[Dvbt1RSEncoderComponent](#) is the second block composing the DVB-T transmission chain. This block is a non-binary Reed-Solomon (R-S) encoder operating on the Galois field  $GF(2^8)$  of 256 elements. Every element in the field is either 0 or an integer power of a primitive element  $\alpha$ ; the field is generated by the primitive polynomial  $p(x) = x^8 + x^4 + x^3 + x^2 + 1$ . The code generator polynomial, instead, is generated to have as roots all the first 16 powers (0 to 15) of the primitive element  $\alpha = 2$ , as

$$g(x) = (x + \alpha^0)(x + \alpha^1) \dots (x + \alpha^{15}) .$$

The encoder computes the remainder of the division of the message polynomial  $m(x)$ , of 239 bytes, by the generator polynomial  $g(x)$ , and considers this as the parity polynomial  $p(x)$ , of 16 bytes. The codeword is then composed by appending the parity polynomial and the message polynomial together (255 bytes), as

$$c(x) = p(x) + x^{16}m(x) .$$

This code is capable of correcting  $t = 8$  errated bytes in every codeword. Actually, DVB-T uses shortened codewords of 204 bytes, generated by messages of 188 bytes prepended by a string of 51 zero bytes. The encoder itself can be implemented with a feedback shift register, operating in  $GF(2^8)$ . Please note that the codewords are message-first parity-last ordered.

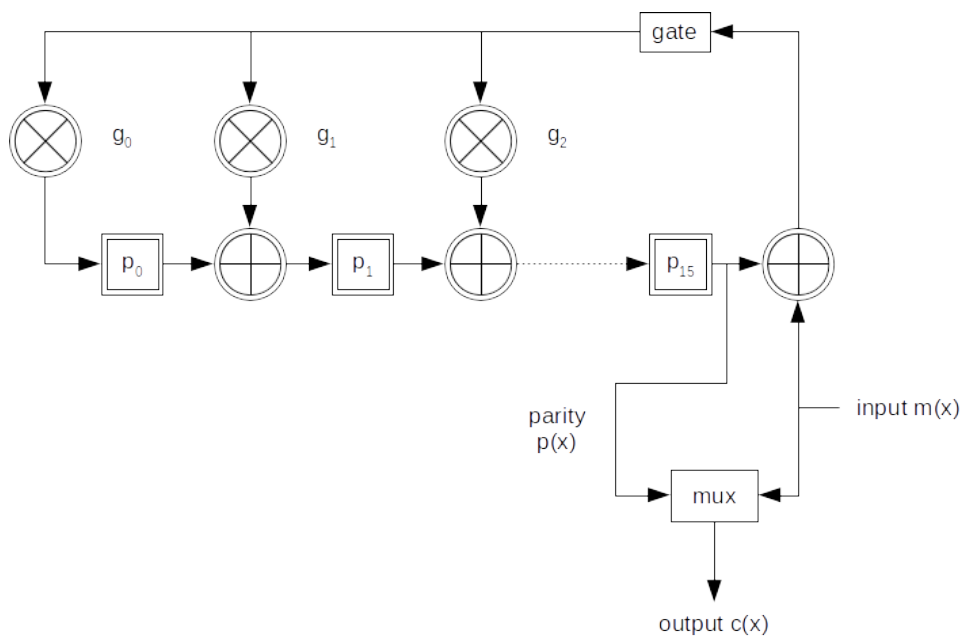


Figure 7.9: DVB-T shortened R-S encoder.

Differently from the simpler multiply-and-add operations in the binary Galois field  $GF(2)$ , in this case we must recur to byte operators, which are practically implemented with look-up tables that perform exponentiation and logarithm of the  $GF(2^8)$  elements. Particular care is taken to consider the zero element.

There is only one parameter that can be changed in the XML configuration file:

- *debug*: by default set to "false", is used to print some small debugging information for the interested developer.

### References

- ETSI Standard: *EN 300 744 V1.5.1, Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for digital terrestrial television*, available at [ETSI Publications Download Area](#)



- S. Li, D. J. Costello, *Error Control Coding, Second Edition*, Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 2004

Definition at line 121 of file Dvbt1RSEncoderComponent.h.

## 7.12.2 Member Typedef Documentation

### 7.12.2.1 typedef std::vector<uint8\_t> iris::phy::Dvbt1RSEncoderComponent::ByteVec

A vector of bytes.

Definition at line 127 of file Dvbt1RSEncoderComponent.h.

### 7.12.2.2 typedef ByteVec::iterator iris::phy::Dvbt1RSEncoderComponent::ByteVecIt

An iterator for a vector of bytes.

Definition at line 130 of file Dvbt1RSEncoderComponent.h.

## 7.12.3 Constructor & Destructor Documentation

### 7.12.3.1 iris::phy::Dvbt1RSEncoderComponent::Dvbt1RSEncoderComponent ( std::string name )

Default constructor.

Registers the block parameters and initializes some variables

Definition at line 57 of file Dvbt1RSEncoderComponent.cpp.

References [debug\\_x](#).

```

58 : PhyComponent (name,                                // component name
59     "dvbt1rsencoder",                                // component type
60     "A DVB-T1 R-S encoder component",                // description
61     "Giuseppe Baruffa",                              // author
62     "0.1")                                            // version
63 , sampleRate_(0)
64 , timeStamp_(0)
65 , tsOffset_(0)
66 {
67     registerParameter(
68         "debug", "Whether to output debug data",
69         "false", true, debug_x);
70 }
```

### 7.12.3.2 iris::phy::Dvbt1RSEncoderComponent::~Dvbt1RSEncoderComponent ( )

Default destructor.

Just calls [destroy\(\)](#).

Definition at line 75 of file Dvbt1RSEncoderComponent.cpp.

References [destroy\(\)](#).

```

76 {
77     destroy();
78 }
```

## 7.12.4 Member Function Documentation

**7.12.4.1** `template<typename T , size_t N> static T* iris::phy::Dvbt1REncoderComponent::begin ( T(&) arr[N] )`  
`[inline], [static], [private]`

Useful templates.

Definition at line 172 of file Dvbt1REncoderComponent.h.

```
172 { return &arr[0]; }
```

**7.12.4.2** `void iris::phy::Dvbt1REncoderComponent::calculateOutputTypes ( std::map< std::string, int > & inputTypes,`  
`std::map< std::string, int > & outputTypes )` `[virtual]`

Calculate the output port types for the IRIS system.

The single output port must provide bytes.

Definition at line 93 of file Dvbt1REncoderComponent.cpp.

```
96 {
97   outputTypes["output1"] = TypeInfo< uint8_t >::identifier;
98 }
```

**7.12.4.3** `void iris::phy::Dvbt1REncoderComponent::destroy ( )` `[private]`

Destroy the component.

Definition at line 264 of file Dvbt1REncoderComponent.cpp.

Referenced by parameterHasChanged(), and ~Dvbt1REncoderComponent().

```
265 {
266 }
```

**7.12.4.4** `template<typename T , size_t N> static T* iris::phy::Dvbt1REncoderComponent::end ( T(&) arr[N] )` `[inline],`  
`[static], [private]`

Definition at line 174 of file Dvbt1REncoderComponent.h.

```
174 { return &arr[0]+N; }
```

**7.12.4.5** `void iris::phy::Dvbt1REncoderComponent::initialize ( )` `[virtual]`

Initialize the component.

Just calls [setup\(\)](#).

Definition at line 103 of file Dvbt1REncoderComponent.cpp.

References [setup\(\)](#).

```
104 {
105   setup();
106 }
```

#### 7.12.4.6 int iris::phy::Dvbt1RSEncoderComponent::modnn ( int x ) [inline],[private]

Computes the modulo-255 of a number.

Definition at line 151 of file Dvbt1RSEncoderComponent.h.

References T1\_MM, and T1\_NN.

Referenced by packetEncode().

```

152 {
153     while (x >= T1_NN) {
154         x -= T1_NN;
155         x = (x >> T1_MM) + (x & T1_NN);
156     }
157     return x;
158 };

```

#### 7.12.4.7 int iris::phy::Dvbt1RSEncoderComponent::packetEncode ( unsigned char \* data, unsigned char \* bb ) [private]

Encodes a single data packet.

Provides, at the output, a systematic encoded codeword where the first 188 bytes are the message, and the last 16 bytes are the parity.

Definition at line 160 of file Dvbt1RSEncoderComponent.cpp.

References alpha\_, gg\_, index\_, modnn(), T1\_A0, T1\_CLEAR, T1\_KK, T1\_NN, and T1\_NN\_KK.

Referenced by process().

```

161 {
162     T1_CLEAR(bb,T1_NN-T1_KK);
163     for(int i = T1_KK - 1; i >= 0; i--)
164     {
165         int feedback = index_[data[i] ^ bb[T1_NN_KK - 1]]; // feedback term
166         if(feedback != T1_A0)
167         {
168             // feedback term is non-zero
169             for(int j = T1_NN_KK - 1; j > 0; j--)
170                 if(gg_[j] != T1_A0)
171                     bb[j] = bb[j - 1] ^ alpha_[modnn(gg_[j] + feedback)];
172             else
173                 bb[j] = bb[j - 1];
174             bb[0] = alpha_[modnn(gg_[0] + feedback)]; // terminal connection
175         }
176         else
177         {
178             // feedback term is zero
179             for(int j = T1_NN_KK - 1; j > 0; j--)
180                 bb[j] = bb[j - 1];
181             bb[0] = 0;
182         }
183     }
184     return 0;
185 }

```

#### 7.12.4.8 void iris::phy::Dvbt1RSEncoderComponent::parameterHasChanged ( std::string name ) [virtual]

Actions taken when the parameters change.

This block has no significant parameters

Definition at line 246 of file Dvbt1RSEncoderComponent.cpp.

References destroy(), and setup().

```

247 {
248     if(name == "???" )
249     {
250         destroy();

```

```

251     setup();
252 }
253 }

```

#### 7.12.4.9 void iris::phy::Dvbt1RSEncoderComponent::process ( ) [virtual]

Main processing method.

Definition at line 188 of file Dvbt1RSEncoderComponent.cpp.

References debug\_x, packetEncode(), RS\_PACKET\_SIZE, rsCodeWord\_, T1\_KK, T1\_NN, T1\_NN\_KK, TS\_PACKET\_SIZE, and tsOffset\_.

```

189 {
190     // request input
191     DataSet< uint8_t >* in = NULL;
192     getInputDataSet("input1", in);
193
194     // calculate sizes
195     int insize = in ? (int) in->data.size() : 0;
196     int numpacks = (insize + tsOffset_) / TS_PACKET_SIZE;
197     int outsize = numpacks * RS_PACKET_SIZE;
198
199     // request output
200     DataSet< uint8_t >* out = NULL;
201     getOutputDataSet("output1", out, outsize);
202
203     // print debug info
204     if(debug_x)
205         LOG(LINFO) << "in/out: " << insize + tsOffset_ << "(" << insize << "+" <<
tsOffset_ << ")" << outsize;
206
207     // fill the messagewords
208     for(ByteVecIt init = in->data.begin(), outit = out->data.begin(); init < in->data.end(); init++)
209     {
210         // copy in reverse order
211         rsCodeWord_[TS_PACKET_SIZE - 1 - tsOffset_] = *init;
212
213         // trigger encoding
214         if(++tsOffset_ == TS_PACKET_SIZE)
215         {
216             int status = packetEncode(rsCodeWord_, rsCodeWord_ +
T1_KK);
217
218             if (status)
219                 LOG(LERROR) << "Problem encoding a R-S word";
220
221             // copy information part
222             for(int b = 0; b < TS_PACKET_SIZE; b++, outit++)
223                 *outit = rsCodeWord_[TS_PACKET_SIZE - 1 - b];
224
225             // copy parity part
226             for(int b = 0; b < T1_NN_KK; b++, outit++)
227                 *outit = rsCodeWord_[T1_NN - 1 - b];
228
229             // reset TS pointer
230             tsOffset_ = 0;
231         }
232     }
233
234     //Copy the timestamp and sample rate for the DataSets
235     out->timeStamp = in->timeStamp;
236     out->sampleRate = in->sampleRate;
237
238     // release input and output
239     releaseInputDataSet("input1", in);
240     releaseOutputDataSet("output1", out);
241 }

```

#### 7.12.4.10 void iris::phy::Dvbt1RSEncoderComponent::registerPorts ( ) [virtual]

Register the encoder ports with the IRIS system.

This component has one input that accepts bytes and one output that provides encoded bytes.

Definition at line 84 of file Dvbt1RSEncoderComponent.cpp.

```

85 {
86     registerInputPort("input1", TypeInfo< uint8_t >::identifier);
87     registerOutputPort("output1", TypeInfo< uint8_t >::identifier);
88 }

```

#### 7.12.4.11 void iris::phy::Dvbt1RSEncoderComponent::setup( ) [private]

Set up offsets and clean variables.

Definition at line 256 of file Dvbt1RSEncoderComponent.cpp.

References rsCodeWord\_, T1\_NN, and tsOffset\_.

Referenced by initialize(), and parameterHasChanged().

```

257 {
258     // clean
259     memset(rsCodeWord_, 0, T1_NN);
260     tsOffset_ = 0;
261 }

```

### 7.12.5 Member Data Documentation

#### 7.12.5.1 int iris::phy::Dvbt1RSEncoderComponent::alpha\_ [static],[private]

**Initial value:**

```

=
{
    1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768, 65536, 131072, 262144, 524288, 1048576, 2097152, 4194304, 8388608, 16777216, 33554432, 67108864, 134217728, 268435456, 536870912, 1073741824, 2147483648, 4294967296, 8589934592, 17179869184, 34359738368, 68719476736, 137438953472, 274877906944, 549755813888, 1099511627776, 2199023255552, 4398046511104, 8796093022208, 17592186044416, 35184372088832, 70368744177664, 140737488355328, 281474976710656, 562949953421312, 1125899906842624, 2251799813685248, 4503599627370496, 9007199254740992, 18014398509481984, 36028797018963968, 72057594037927936, 144115188075855872, 288230376151711744, 576460752303423488, 1152921504606846976, 2305843009213693952, 4611686018427387904, 9223372036854775808, 18446744073709551616, 36893488147419103232, 73786976294838206464, 147573952589676412928, 295147905179352825856, 590295810358705651712, 1180591620717411303424, 2361183241434822606848, 4722366482869645213696, 9444732965739290427392, 18889465931478580854784, 37778931862957161709568, 75557863725914323419136, 151115727451828646838272, 302231454903657293676544, 604462909807314587353088, 1208925819614629174706176, 2417851639229258349412352, 4835703278458516698824704, 9671406556917033397649408, 19342813113834066795298816, 38685626227668133590597632, 77371252455336267181195264, 154742504910672534362390528, 309485009821345068724781056, 618970019642690137449562112, 1237940039285380274899124224, 2475880078570760549798248448, 4951760157141521099596496896, 9903520314283042199192993792, 19807040628566084398385987584, 39614081257132168796771975168, 79228162514264337593543950336, 158456325028528675187087900672, 316912650057057350374175801344, 633825300114114700748351602688, 1267650600228229401496703205376, 2535301200456458802993406410752, 5070602400912917605986812821504, 10141204801825835211973625643008, 20282409603651670423947251286016, 40564819207303340847894502572032, 81129638414606681695789005144064, 162259276829213363391578010288128, 324518553658426726783156020576256, 649037107316853453566312041152512, 1298074214633706907132624082305024, 2596148429267413814265248164610048, 5192296858534827628530496329220096, 10384593717069655257060992658440192, 20769187434139310514121985316880384, 41538374868278621028243970633760768, 83076749736557242056487941267521536, 166153499473114484112975882535043072, 332306998946228968225951765070086144, 664613997892457936451903530140172288, 1329227995784915872903807060280344576, 2658455991569831745807614120560689152, 5316911983139663491615228241121378304, 10633823966279326983230456482242756608, 21267647932558653966460912964485513216, 42535295865117307932921825928971026432, 85070591730234615865843651857942052864, 170141183460469231731687303715884105728, 340282366920938463463374607431768211456, 680564733841876926926749214863536422912, 1361129467683753853853498429727072845824, 2722258935367507707706996859454145691648, 5444517870735015415413993718908291383296, 10889035741470030830827987437816582766592, 21778071482940061661655974875633165533184, 43556142965880123323311949751266331066368, 87112285931760246646623899502532662132736, 174224571863520493293247799005065324265472, 348449143727040986586495598010130648530944, 696898287454081973172991196020261297061888, 1393796574908163946345982392040522594123776, 2787593149816327892691964784081045188247552, 5575186299632655785383929568162090376495104, 11150372599265311570767859136324180752990208, 22300745198530623141535718272648361505980416, 44601490397061246283071436545296723011960832, 89202980794122492566142873090593446023921664, 178405961588244985132285746181186892047843328, 356811923176489970264571492362373784095686656, 713623846352979940529142984724747568191373312, 1427247692705959881058285969449495136382746624, 2854495385411919762116571938898990272765493248, 5708990770823839524233143877797980545530986496, 11417981541647679048466287755595961091061972992, 22835963083295358096932575511191922182123945984, 45671926166590716193865151022383844364247891968, 91343852333181432387730302044767688728495783936, 182687704666362864775460604089535377456991567872, 365375409332725729550921208179070754913983135744, 730750818665451459101842416358141509827966271488, 1461501637330902918203684832716283019655932542976, 2923003274661805836407369665432566039311865085952, 5846006549323611672814739330865132078623730171904, 11692013098647223345629478661730264157247460343808, 23384026197294446691258957323460528314494920687616, 46768052394588893382517914646921056628989841375232, 93536104789177786765035829293842113257979682750464, 187072209578355573530071658587684226515959365500928, 374144419156711147060143317175368453031918731001856, 748288838313422294120286634350736906063837462003712, 1496577676626844588240573268701473812127674924007424, 2993155353253689176481146537402947624255349848014848, 5986310706507378352962293074805895248510699696029696, 11972621413014756705924586149611790497021399392059392, 23945242826029513411849172299223580994042798784118784, 47890485652059026823698344598447161988085597568237568, 95780971304118053647396689196894323976171195136475136, 191561942608236107294793378393788647952342390272950272, 383123885216472214589586756787577295904684780545900544, 766247770432944429179173513575154591809369561091801088, 1532495540865888858358347027150309183618739122183602176, 3064991081731777716716694054300618367237478244367204352, 6129982163463555433433388108601236734474956488734408704, 12259964326927110866866776217202473468949912977468817408, 24519928653854221733733552434404946937899825954937634816, 49039857307708443467467104868809893875799651909875269632, 98079714615416886934934209737619787751599303819750539264, 196159429230833773869868419475239575503198607639501078528, 392318858461667547739736838950479151006397215279002157056, 784637716923335095479473677900958302012794430558004314112, 1569275433846670190958947355801916604025588861116008628224, 3138550867693340381917894711603833208051177722232017256448, 6277101735386680763835789423207666416102355444464034512896, 12554203470773361527671578846415332832204710888928069025792, 25108406941546723055343157692830665664409421777856138051584, 50216813883093446110686315385661331328818843555712276103168, 100433627766186892221372630771322662657637687111424552206336, 200867255532373784442745261542645325315275374222849104412672, 401734511064747568885490523085290650630550748445698208825344, 803469022129495137770981046170581301261101496891396417650688, 1606938044258990275541962092341162602522202993782792835301376, 3213876088517980551083924184682325205044405987565585670602752, 6427752177035961102167848369364650410088811975131171341205504, 12855504354071922204335696738729300820177623950262342682411008, 25711008708143844408671393477458601640355247900524685364822016, 51422017416287688817342786954917203280710495801049370729644032, 102844034832575377634685573909834406561420991602098741459288064, 205688069665150755269371147819668813122841983204197482918576128, 411376139330301510538742295639337626245683966408394965837152256, 822752278660603021077484591278675252491367932816789931674304512, 1645504557321206042154969182557350504982735865633579863348609024, 3291009114642412084309938365114701009965471731267159726697218048, 6582018229284824168619876730229402019930943462534319453394436096, 13164036458569648337239753460458804039861886925068638906788872192, 26328072917139296674479506920917608079723773850137277813577744384, 52656145834278593348959013841835216159447547700274555627155488768, 105312291668557186697918027683670432318895095400549111254310977536, 210624583337114373395836055367340864637790190801098222508621955072, 421249166674228746791672110734681729275580381602196445017243910144, 842498333348457493583344221469363458551160763204392890034487820288, 1684996666696914987166688442938726917102321526408785780068975640576, 3369993333393829974333376885877453834204643052817571560137951281152, 6739986666787659948666753771754907668409286105635143120275902562304, 13479973333575319897333507543509815336818572211270286240551805124608, 26959946667150639794667015087019630673637144422540572481103610249216, 53919893334301279589334030174039261347274288845081144962207220498432, 107839786668602559178668060348078522694548577690162289924414440996864, 215679573337205118357336120696157045389097155380324579848828881993728, 431359146674410236714672241392314090778194310760649159697657763987456, 862718293348820473429344482784628181556388621521298319395315527974912, 1725436586697640946858688965569256363112777243042596638790631055949824, 3450873173395281893717377931138512726225554486085193277581262111899648, 6901746346790563787434755862277025452451108972170386555162524223799296, 13803492693581127574869511724554050904902217944340773110325048447598592, 27606985387162255149739023449108101809804435888681546220650096895197184, 55213970774324510299478046898216203619608871777363092441300193790394368, 110427941548649020598956093796432407239217743554726184882600387580788736, 220855883097298041197912187592864814478435487109452369765200775161577472, 441711766194596082395824375185729628956870974218904739530401550323154944, 883423532389192164791648750371459257913741948437809479060803100646309888, 1766847064778384329583297500742918515827483896875618958121606201292619776, 3533694129556768659166595001485837031654967793751237916243212402585239552, 7067388259113537318333190002971674063309935587502475832486424805170479104, 14134776518227074636666380005943348126619871175004951664972849610340958208, 28269553036454149273332760011886696253239742350009903329945699220681916416, 56539106072908298546665520023773392506479484700019806659891398441363832832, 113078212145816597093331040047546785012958969400039613319782796882727665664, 226156424291633194186662080095093570025917938800079226639565593765455331328, 452312848583266388373324160190187140051835877600158453279131187530910662656, 904625697166532776746648320380374280103671755200316906558262375061821325312, 1809251394333065553493296640760748560207343510400633813116524750123642650624, 3618502788666131106986593281521497120414687020801267626233049500247285301248, 7237005577332262213973186563042994240829374041602535252466099000494570602496, 14474011154664524427946373126085988481658748083205070504932198000989141204992, 28948022309329048855892746252171976963317496166410141009864396001978282409984, 57896044618658097711785492504343953926634992332820282019728792003956564819968, 115792089237316195423570985008687907853269984665640564039457584007913129639936, 231584178474632390847141970017375815706539969331281128078915168015826259279872, 463168356949264781694283940034751631413079938662562256157830336031652518559744, 926336713898529563388567880069503262826159877325124512315660672063305037119488, 1852673427797059126777135760139006525652319754650249024631321344126610074238976, 3705346855594118253554271520278013051304639509300498049262642688253220148477952, 7410693711188236507108543040556026102609279018600996098525285376506440296955904, 14821387422376473014217086081112052205218558037201992197050570753012880593911808, 29642774844752946028434172162224104410437116074403984394101141506025761187823616, 59285549689505892056868344324448208820874232148807968788202283012051522375647232, 118571099379011784113736688648896417641748464297615937576404566024103044751294464, 237142198758023568227473377297792835283496928595231875152809132048206089502588928, 474284397516047136454946754595585670566993857190463750305618264096412179005177856, 948568795032094272909893509191171341133987714380927500611236528192824358010355712, 18971375900641885458197870183823426822679754287618550012
```

```
=
{
    120, 225, 194, 182, 169, 147, 191, 91, 3, 76, 161, 102, 109, 107, 104, 120, 0
}
```

R-S code generator polynomial.

Definition at line 168 of file Dvbt1RSEncoderComponent.h.

Referenced by packetEncode().

**7.12.5.4** `int iris::phy::Dvbt1RSEncoderComponent::index_ [static],[private]`

**Initial value:**

```
=
{
    255, 0, 1, 25, 2, 50, 26, 198, 3, 223, 51, 238, 27, 104, 199, 75,
    4, 100, 224, 14, 52, 141, 239, 129, 28, 193, 105, 248, 200, 8, 76, 113,
    5, 138, 101, 47, 225, 36, 15, 33, 53, 147, 142, 218, 240, 18, 130, 69,
    29, 181, 194, 125, 106, 39, 249, 185, 201, 154, 9, 120, 77, 228, 114, 166,
    6, 191, 139, 98, 102, 221, 48, 253, 226, 152, 37, 179, 16, 145, 34, 136,
    54, 208, 148, 206, 143, 150, 219, 189, 241, 210, 19, 92, 131, 56, 70, 64,
    30, 66, 182, 163, 195, 72, 126, 110, 107, 58, 40, 84, 250, 133, 186, 61,
    202, 94, 155, 159, 10, 21, 121, 43, 78, 212, 229, 172, 115, 243, 167, 87,
    7, 112, 192, 247, 140, 128, 99, 13, 103, 74, 222, 237, 49, 197, 254, 24,
    227, 165, 153, 119, 38, 184, 180, 124, 17, 68, 146, 217, 35, 32, 137, 46,
    55, 63, 209, 91, 149, 188, 207, 205, 144, 135, 151, 178, 220, 252, 190, 97,
    242, 86, 211, 171, 20, 42, 93, 158, 132, 60, 57, 83, 71, 109, 65, 162,
    31, 45, 67, 216, 183, 123, 164, 118, 196, 23, 73, 236, 127, 12, 111, 246,
    108, 161, 59, 82, 41, 157, 85, 170, 251, 96, 134, 177, 187, 204, 62, 90,
    203, 89, 95, 176, 156, 169, 160, 81, 11, 245, 22, 235, 122, 117, 44, 215,
    79, 174, 213, 233, 230, 231, 173, 232, 116, 214, 244, 234, 168, 80, 88, 175
}
```

LUT containing the base  $\alpha$  logarithm of the field elements.

Definition at line 165 of file Dvbt1RSEncoderComponent.h.

Referenced by packetEncode().

**7.12.5.5** `uint8_t iris::phy::Dvbt1RSEncoderComponent::rsCodeWord_[T1_NN] [private]`

Nonshortened codeword.

Definition at line 163 of file Dvbt1RSEncoderComponent.h.

Referenced by process(), and setup().

**7.12.5.6** `double iris::phy::Dvbt1RSEncoderComponent::sampleRate_ [private]`

Sample rate of current frame.

Definition at line 161 of file Dvbt1RSEncoderComponent.h.

**7.12.5.7** `double iris::phy::Dvbt1RSEncoderComponent::timeStamp_ [private]`

Timestamp of current frame.

Definition at line 158 of file Dvbt1RSEncoderComponent.h.

**7.12.5.8** `int iris::phy::Dvbt1RSEncoderComponent::tsOffset_ [private]`

Current offset in TS input.

Definition at line 164 of file Dvbt1RSEncoderComponent.h.

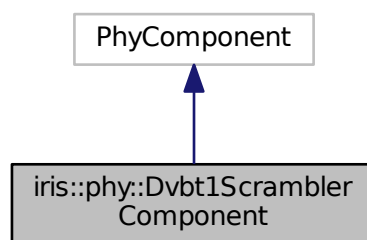
Referenced by process(), and setup().

## 7.13 iris::phy::Dvbt1ScramblerComponent Class Reference

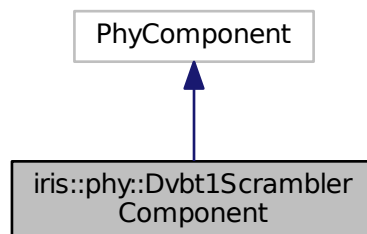
A DVB-T energy dispersal component.

```
#include <Dvbt1ScramblerComponent.h>
```

Inheritance diagram for iris::phy::Dvbt1ScramblerComponent:



Collaboration diagram for iris::phy::Dvbt1ScramblerComponent:



### Public Types

- typedef std::vector< uint8\_t > [ByteVec](#)  
*A vector of bytes.*
- typedef ByteVec::iterator [ByteVecIt](#)  
*An iterator for a vector of bytes.*

### Public Member Functions

- [Dvbt1ScramblerComponent](#) (std::string name)  
*Default constructor.*

- [~Dvbt1ScramblerComponent](#) ()  
*Default destructor.*
- virtual void [calculateOutputTypes](#) (std::map< std::string, int > &inputTypes, std::map< std::string, int > &outputTypes)  
*Calculate the output port types for the IRIS system.*
- virtual void [registerPorts](#) ()  
*Register the scrambler ports with the IRIS system.*
- virtual void [initialize](#) ()  
*Initialize the component.*
- virtual void [process](#) ()  
*Main processing method.*
- virtual void [parameterHasChanged](#) (std::string name)  
*Actions taken when the parameters change.*

### Private Member Functions

- void [setup](#) ()  
*Set up counters, offsets, etc.*
- void [destroy](#) ()  
*Destroy the component.*

### Static Private Member Functions

- template<typename T , size\_t N>  
static T \* [begin](#) (T(&arr)[N])  
*Useful templates.*
- template<typename T , size\_t N>  
static T \* [end](#) (T(&arr)[N])

### Private Attributes

- bool [debug\\_x](#)  
*Debug flag (default = false)*
- double [reportInterval\\_x](#)  
*Reporting interval in seconds (default = 0)*
- double [timeStamp\\_](#)  
*Timestamp of current frame.*
- double [sampleRate\\_](#)  
*Sample rate of current frame.*
- int [scramblerOffset\\_](#)  
*Current scrambling offset.*
- boost::posix\_time::ptime [start\\_](#)  
*Timestamp used for frame error rate reports.*
- uint64\_t [doneBytes\\_](#)  
*currently processed bytes*

### Static Private Attributes

- static uint8\_t [scramblerPrbs\\_](#) [1504]  
*Scrambling PRBS bytes.*



### 7.13.1 Detailed Description

A DVB-T energy dispersal component.

[Dvbt1ScramblerComponent](#) is the first block composing the DVB-T transmission chain. This block takes an MPEG-2 Transport Stream (TS) of data bytes in `uint8_t` format and outputs a scrambled stream of `uint8_t` data. Per the DVB-T standard, the PRBS generator polynomial is  $1 + X^{14} + X^{15}$ . The scrambler operates on a group of eight TS packets: each packet is 188-byte long and begins with the SYNC byte, 0x47. The PRBS register is loaded with the sequence "100101010000000" and is shift-enabled after the eighth bit, thus the ninth bit is the first to be scrambled. The other 7 SYNC bytes in the group are then bitwise-inverted to 0xB8, so as to provide a viable means for recovering scrambling synchrony at the receiver. The process is then repeated for the following groups of eight packets. The full period of the scrambling sequence is thus of  $188 * 8 - 1 = 1503$  bytes.

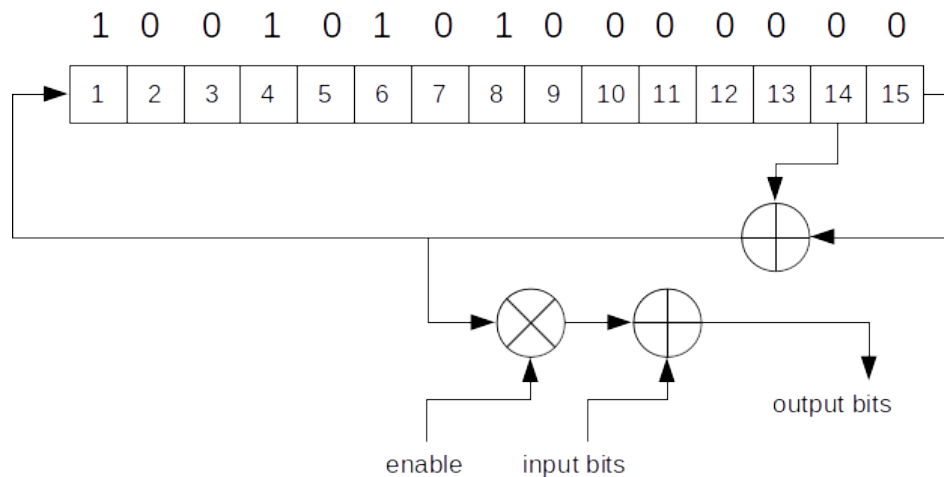


Figure 7.10: DVB-T energy dispersal.

There are two parameters that can be changed in the XML configuration file:

- *debug*: by default set to "false", is used to print some small debugging information for the interested developer.
- *reportinterval*: by default set to "0", which means it is disabled. If a number greater than zero is used, then it will be the number of seconds between which the block reports the computed processing speed. This can be useful to benchmark on-the-fly the processing speed of a complete DVB-T modulator graph that uses this block as source: if the graph is free-running, i.e., not terminated into an USRP block or similar, it will provide the maximum TS bitrate that the CPU is capable to process. Differently, if terminated into an USRP, this can be used to verify if the expected bitrate value (for that particular combination of DVB-T modulation and coding parameters) is honored.

### References

- ETSI Standard: *EN 300 744 V1.5.1, Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for digital terrestrial television*, available at [ETSI Publications Download Area](#)

Definition at line 87 of file `Dvbt1ScramblerComponent.h`.

### 7.13.2 Member Typedef Documentation

#### 7.13.2.1 `typedef std::vector<uint8_t> iris::phy::Dvbt1ScramblerComponent::ByteVec`

A vector of bytes.

Definition at line 93 of file `Dvbt1ScramblerComponent.h`.

### 7.13.2.2 typedef ByteVec::iterator iris::phy::Dvbt1ScramblerComponent::ByteVecIt

An iterator for a vector of bytes.

Definition at line 96 of file Dvbt1ScramblerComponent.h.

## 7.13.3 Constructor & Destructor Documentation

### 7.13.3.1 iris::phy::Dvbt1ScramblerComponent::Dvbt1ScramblerComponent ( std::string name )

Default constructor.

Registers the block parameters and initializes some variables

Definition at line 58 of file Dvbt1ScramblerComponent.cpp.

References `debug_x`, and `reportInterval_x`.

```

59 : PhyComponent (name,                                     // component name
60                 "dvbt1scrambler",                       // component type
61                 "A DVB-T1 scrambler component",          // description
62                 "Giuseppe Baruffa",                     // author
63                 "0.1")                                   // version
64     ,sampleRate_(0)
65     ,timeStamp_(0)
66     ,scramblerOffset_(0)
67 {
68     registerParameter(
69         "debug", "Whether to output debug data",
70         "false", true, debug_x);
71
72     registerParameter(
73         "reportinterval", "Report interval in seconds",
74         "0", true, reportInterval_x);
75 }
```

### 7.13.3.2 iris::phy::Dvbt1ScramblerComponent::~~Dvbt1ScramblerComponent ( )

Default destructor.

Just calls `destroy()`.

Definition at line 80 of file Dvbt1ScramblerComponent.cpp.

References `destroy()`.

```

81 {
82     destroy();
83 }
```

## 7.13.4 Member Function Documentation

### 7.13.4.1 template<typename T , size\_t N> static T\* iris::phy::Dvbt1ScramblerComponent::begin ( T(&) arr[N] ) [inline], [static], [private]

Useful templates.

Definition at line 125 of file Dvbt1ScramblerComponent.h.

```

125 { return &arr[0]; }
```

### 7.13.4.2 void iris::phy::Dvbt1ScramblerComponent::calculateOutputTypes ( std::map< std::string, int > & inputTypes, std::map< std::string, int > & outputTypes ) [virtual]

Calculate the output port types for the IRIS system.

The single output port must provide bytes.

Definition at line 98 of file Dvbt1ScramblerComponent.cpp.

```
101 {
102     outputTypes["output1"] = TypeInfo< uint8_t >::identifier;
103 }
```

#### 7.13.4.3 void iris::phy::Dvbt1ScramblerComponent::destroy ( ) [private]

Destroy the component.

Definition at line 280 of file Dvbt1ScramblerComponent.cpp.

Referenced by parameterHasChanged(), and ~Dvbt1ScramblerComponent().

```
281 {
282 }
```

#### 7.13.4.4 template<typename T, size\_t N> static T\* iris::phy::Dvbt1ScramblerComponent::end ( T(&) arr[N] ) [inline], [static], [private]

Definition at line 127 of file Dvbt1ScramblerComponent.h.

```
127 { return &arr[0]+N; }
```

#### 7.13.4.5 void iris::phy::Dvbt1ScramblerComponent::initialize ( ) [virtual]

Initialize the component.

Just calls [setup\(\)](#).

Definition at line 108 of file Dvbt1ScramblerComponent.cpp.

References [setup\(\)](#).

```
109 {
110     setup();
111 }
```

#### 7.13.4.6 void iris::phy::Dvbt1ScramblerComponent::parameterHasChanged ( std::string name ) [virtual]

Actions taken when the parameters change.

This block is not to be reset if parameters change

Definition at line 262 of file Dvbt1ScramblerComponent.cpp.

References [destroy\(\)](#), and [setup\(\)](#).

```
263 {
264     if(name == "???" )
265     {
266         destroy();
267         setup();
268     }
269 }
```

#### 7.13.4.7 void iris::phy::Dvbt1ScramblerComponent::process ( ) [virtual]

Main processing method.

Definition at line 212 of file Dvbt1ScramblerComponent.cpp.

References debug\_x\_, doneBytes\_, reportInterval\_x\_, scramblerOffset\_, scramblerPrbs\_, and start\_.

```

213 {
214     // request input
215     DataSet< uint8_t >* in = NULL;
216     getInputDataSet("input1", in);
217     int size = in ? (int) in->data.size() : 0;
218
219     // print debug info
220     if(debug_x)
221         LOG(LINFO) << "in/out: " << size << "/" << size;
222
223     // request output - same size
224     DataSet< uint8_t >* out = NULL;
225     getOutputDataSet("output1", out, size);
226
227     // do the scrambling using the static array above
228     for(ByteVecIt init = in->data.begin(), outit = out->data.begin(); init < in->data.end(); init++,
        outit++) {
229         *outit = *init ^ scramblerPrbs_[scramblerOffset_];
230         if(++scramblerOffset_ == 1504)
231             scramblerOffset_ = 0;
232     }
233
234     // Copy the timestamp and sample rate for the DataSets
235     out->timeStamp = in->timeStamp;
236     out->sampleRate = in->sampleRate;
237
238     // release input and output
239     releaseInputDataSet("input1", in);
240     releaseOutputDataSet("output1", out);
241
242     // print the calculated bitrate
243     if(reportInterval_x)
244     {
245         ptime t = microsec_clock::local_time(); // current time
246         doneBytes_ += size; // increase processed bytes since last report
247         time_duration delta = t - start_; // time elapsed from last report
248         if(delta > seconds(reportInterval_x))
249         {
250             // interval is triggered, compute speed and report
251             LOG(LINFO) << "Current TS bitrate: " << 8.0 * (double) doneBytes_ / (delta.
                total_microseconds()) << " Mbps";
252             // reset counters
253             start_ = t;
254             doneBytes_ = 0;
255         }
256     }
257 }
```

#### 7.13.4.8 void iris::phy::Dvbt1ScramblerComponent::registerPorts ( ) [virtual]

Register the scrambler ports with the IRIS system.

This component has one input that accepts TS bytes and one output that provides scrambled bytes.

Definition at line 89 of file Dvbt1ScramblerComponent.cpp.

```

90 {
91     registerInputPort("input1", TypeInfo< uint8_t >::identifier);
92     registerOutputPort("output1", TypeInfo< uint8_t >::identifier);
93 }
```

#### 7.13.4.9 void iris::phy::Dvbt1ScramblerComponent::setup ( ) [private]

Set up counters, offsets, etc.

Definition at line 272 of file Dvbt1ScramblerComponent.cpp.

References doneBytes\_, scramblerOffset\_, and start\_.

Referenced by initialize(), and parameterHasChanged().

```
273 {
274     scramblerOffset_ = 0;
275     start_ = microsec_clock::local_time();
276     doneBytes_ = 0L;
277 }
```

### 7.13.5 Member Data Documentation

#### 7.13.5.1 bool iris::phy::Dvbt1ScramblerComponent::debug\_x [private]

Debug flag (default = false)

Definition at line 110 of file Dvbt1ScramblerComponent.h.

Referenced by Dvbt1ScramblerComponent(), and process().

#### 7.13.5.2 uint64\_t iris::phy::Dvbt1ScramblerComponent::doneBytes\_ [private]

currently processed bytes

Definition at line 121 of file Dvbt1ScramblerComponent.h.

Referenced by process(), and setup().

#### 7.13.5.3 double iris::phy::Dvbt1ScramblerComponent::reportInterval\_x [private]

Reporting interval in seconds (default = 0)

Definition at line 111 of file Dvbt1ScramblerComponent.h.

Referenced by Dvbt1ScramblerComponent(), and process().

#### 7.13.5.4 double iris::phy::Dvbt1ScramblerComponent::sampleRate\_ [private]

Sample rate of current frame.

Definition at line 117 of file Dvbt1ScramblerComponent.h.

#### 7.13.5.5 int iris::phy::Dvbt1ScramblerComponent::scramblerOffset\_ [private]

Current scrambling offset.

Definition at line 119 of file Dvbt1ScramblerComponent.h.

Referenced by process(), and setup().

#### 7.13.5.6 uint8\_t iris::phy::Dvbt1ScramblerComponent::scramblerPrbs\_ [static], [private]

Scrambling PRBS bytes.

Scrambling sequence bytes.

Definition at line 129 of file Dvbt1ScramblerComponent.h.

Referenced by process().

#### 7.13.5.7 `boost::posix_time::ptime iris::phy::Dvbt1ScramblerComponent::start_` [private]

Timestamp used for frame error rate reports.

Definition at line 120 of file `Dvbt1ScramblerComponent.h`.

Referenced by `process()`, and `setup()`.

#### 7.13.5.8 `double iris::phy::Dvbt1ScramblerComponent::timeStamp_` [private]

Timestamp of current frame.

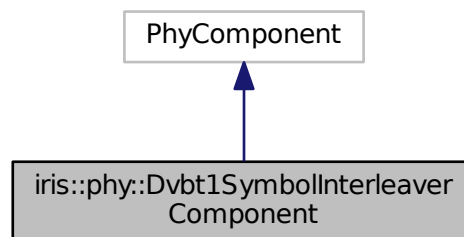
Definition at line 116 of file `Dvbt1ScramblerComponent.h`.

## 7.14 `iris::phy::Dvbt1SymbolInterleaverComponent` Class Reference

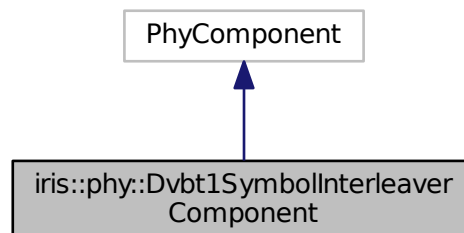
A DVB-T1 symbol interleaver component.

```
#include <Dvbt1SymbolInterleaverComponent.h>
```

Inheritance diagram for `iris::phy::Dvbt1SymbolInterleaverComponent`:



Collaboration diagram for `iris::phy::Dvbt1SymbolInterleaverComponent`:



## Public Types

- typedef std::vector< uint8\_t > [ByteVec](#)  
*A vector of bytes.*
- typedef ByteVec::iterator [ByteVecIt](#)  
*An iterator for a vector of bytes.*

## Public Member Functions

- [Dvbt1SymbolInterleaverComponent](#) (std::string name)  
*Default constructor.*
- [~Dvbt1SymbolInterleaverComponent](#) ()  
*Default destructor.*
- virtual void [calculateOutputTypes](#) (std::map< std::string, int > &inputTypes, std::map< std::string, int > &outputTypes)  
*Calculate the output port types for the IRIS system.*
- virtual void [registerPorts](#) ()  
*Register the interleaver ports with the IRIS system.*
- virtual void [initialize](#) ()  
*Initialize the component.*
- virtual void [process](#) ()  
*Main processing method.*
- virtual void [parameterHasChanged](#) (std::string name)  
*Actions taken when the parameters change.*

## Private Member Functions

- void [setup](#) ()  
*Set up the register space and initialize.*
- void [destroy](#) ()  
*Destroy the component.*

## Static Private Member Functions

- template<typename T , size\_t N>  
static T \* [begin](#) (T(&arr)[N])  
*Useful templates.*
- template<typename T , size\_t N>  
static T \* [end](#) (T(&arr)[N])

## Private Attributes

- bool [debug\\_x](#)  
*Debug flag (default = false)*
- int [ofdmMode\\_x](#)  
*OFDM mode (default = 2048)*
- double [timeStamp\\_](#)  
*Timestamp of current frame.*
- double [sampleRate\\_](#)  
*Sample rate of current frame.*

- int [siOffset\\_](#)  
*Interleaving offset.*
- int [siLength\\_](#)  
*Interleaving register length.*
- uint8\_t \* [siRegister\\_](#)  
*Actual interleaving register.*
- int [eo\\_](#)  
*Even/odd numbered OFDM block.*

### Static Private Attributes

- static int [H\\_2K\\_](#)[1512]  
*Interleaving addresses for 2K.*
- static int [H\\_4K\\_](#)[3024]  
*Interleaving addresses for 4K.*
- static int [H\\_8K\\_](#)[6048]  
*Interleaving addresses for 8K.*

#### 7.14.1 Detailed Description

A DVB-T1 symbol interleaver component.

[Dvbt1SymbolInterleaverComponent](#) is the seventh block composing the DVB-T transmission chain. Its purpose, together with the bit interleaver, is that of reordering the channel encoded bits in order to convert the possible error bursts arising from the communication on the physical channel (due to impulsive noise, multipath, fading) into well-separated single-error events. This way, the channel decoders at the RX side (Viterbi and Reed-Solomon decoder) are able to perform at their best theoretical limit in white Gaussian noise (WGN) conditions.

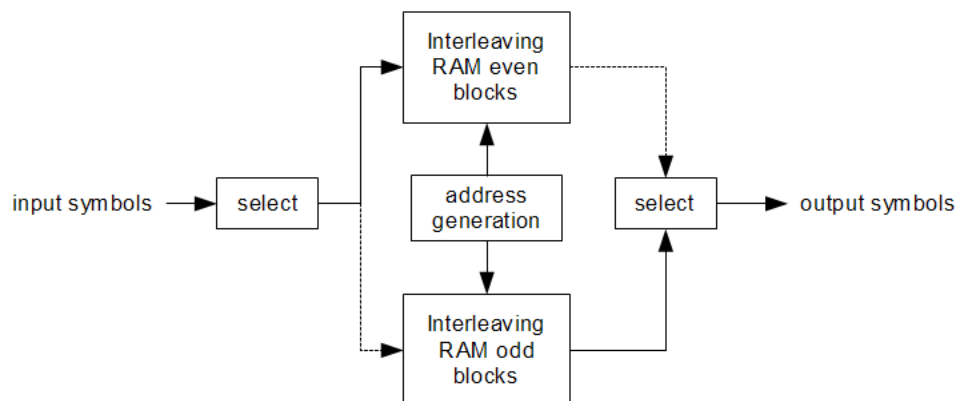


Figure 7.11: DVB-T symbol interleaver.

The symbol interleaver is a block-based interleaver, i.e., a block of consecutive symbols is written in the interleaving RAM, and then the same symbols are read into an output block with pseudo-random read addresses. Every output block of symbols is mapped into an OFDM block. For instance, in the 8K case, the symbol interleaver memory is of 6048 cells (1512 for the 2K case). The pseudo-random interleaving law is generated by means of linear feedback registers, whose state is turned into a valid interleaving address with the help of a bit mapping between the register bits and the addressing bits. Every second interleaving block, the interleaving law is exchanged between the reading and writing processes. In the practical implementation used in IRIS, however, the interleaving addresses are statically embedded in the source files, and a simple address mapping law is applied.

This block accepts in input elements in uint8\_t ( v-bit symbols) and generates in output v-bit symbols (uint8\_t).

There are three parameters that can be changed in the XML configuration file:



- *debug*: by default set to "false", is used to print some small debugging information for the interested developer.
- *ofdmMode*: by default set to "2048", this is used to select one of the three possible OFDM modes. The admitted values are "2048", "4096", "8192", respectively for 2K, 4K (DVB-H, unused), and 8K.

## References

- ETSI Standard: *EN 300 744 V1.5.1, Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for digital terrestrial television*, available at [ETSI Publications Download Area](#)

Definition at line 89 of file Dvbt1SymbolInterleaverComponent.h.

## 7.14.2 Member Typedef Documentation

### 7.14.2.1 typedef std::vector<uint8\_t> iris::phy::Dvbt1SymbolInterleaverComponent::ByteVec

A vector of bytes.

Definition at line 95 of file Dvbt1SymbolInterleaverComponent.h.

### 7.14.2.2 typedef ByteVec::iterator iris::phy::Dvbt1SymbolInterleaverComponent::ByteVecIt

An iterator for a vector of bytes.

Definition at line 98 of file Dvbt1SymbolInterleaverComponent.h.

## 7.14.3 Constructor & Destructor Documentation

### 7.14.3.1 iris::phy::Dvbt1SymbolInterleaverComponent::Dvbt1SymbolInterleaverComponent ( std::string name )

Default constructor.

Registers the block parameters and initializes some variables

Definition at line 57 of file Dvbt1SymbolInterleaverComponent.cpp.

References `begin()`, `debug_x`, `end()`, and `ofdmMode_x`.

```

58 : PhyComponent(name,                                // component name
59     "dvbt1symbolinterleaver",                        // component type
60     "A DVB-T1 symbol interleaver component", // description
61     "Giuseppe Baruffa",                          // author
62     "0.1")                                         // version
63     ,sampleRate_(0)
64     ,timeStamp_(0)
65     ,siRegister_(NULL)
66 {
67     registerParameter(
68         "debug", "Whether to output debug data",
69         "false", true, debug_x);
70
71     int ofdmarr[] = {2048, 4096, 8192};
72     registerParameter(
73         "ofdmMode", "OFDM mode",
74         "2048", true, ofdmMode_x, list<int>(begin(ofdmarr), end(ofdmarr)));
75 }
```

### 7.14.3.2 iris::phy::Dvbt1SymbolInterleaverComponent::~~Dvbt1SymbolInterleaverComponent ( )

Default destructor.

Just calls `destroy()`.

Definition at line 80 of file Dvbt1SymbolInterleaverComponent.cpp.

References `destroy()`.

```
81 {
82     destroy();
83 }
```

#### 7.14.4 Member Function Documentation

**7.14.4.1** `template<typename T, size_t N> static T* iris::phy::Dvbt1SymbolInterleaverComponent::begin ( T(&) arr[N] )`  
`[inline], [static], [private]`

Useful templates.

Definition at line 130 of file Dvbt1SymbolInterleaverComponent.h.

Referenced by `Dvbt1SymbolInterleaverComponent()`.

```
130 { return &arr[0]; }
```

**7.14.4.2** `void iris::phy::Dvbt1SymbolInterleaverComponent::calculateOutputTypes ( std::map< std::string, int > & inputTypes, std::map< std::string, int > & outputTypes )` `[virtual]`

Calculate the output port types for the IRIS system.

The single output port must provide bytes.

Definition at line 98 of file Dvbt1SymbolInterleaverComponent.cpp.

```
101 {
102     outputTypes["output1"] = TypeInfo< uint8_t >::identifier;
103 }
```

**7.14.4.3** `void iris::phy::Dvbt1SymbolInterleaverComponent::destroy ( )` `[private]`

Destroy the component.

Definition at line 1095 of file Dvbt1SymbolInterleaverComponent.cpp.

References `siRegister_`.

Referenced by `parameterHasChanged()`, and `~Dvbt1SymbolInterleaverComponent()`.

```
1096 {
1097     // clean
1098     delete [] siRegister_;
1099 }
```

**7.14.4.4** `template<typename T, size_t N> static T* iris::phy::Dvbt1SymbolInterleaverComponent::end ( T(&) arr[N] )`  
`[inline], [static], [private]`

Definition at line 132 of file Dvbt1SymbolInterleaverComponent.h.

Referenced by `Dvbt1SymbolInterleaverComponent()`.

```
132 { return &arr[0]+N; }
```

**7.14.4.5 void iris::phy::Dvbt1SymbolInterleaverComponent::initialize ( ) [virtual]**

Initialize the component.

Just calls [setup\(\)](#).

Definition at line 108 of file Dvbt1SymbolInterleaverComponent.cpp.

References [setup\(\)](#).

```
109 {
110     setup();
111 }
```

**7.14.4.6 void iris::phy::Dvbt1SymbolInterleaverComponent::parameterHasChanged ( std::string name ) [virtual]**

Actions taken when the parameters change.

This block has one significant parameter

Definition at line 1073 of file Dvbt1SymbolInterleaverComponent.cpp.

References [destroy\(\)](#), and [setup\(\)](#).

```
1074 {
1075     if(name == "ofdmmode")
1076     {
1077         destroy();
1078         setup();
1079     }
1080 }
```

**7.14.4.7 void iris::phy::Dvbt1SymbolInterleaverComponent::process ( ) [virtual]**

Main processing method.

Definition at line 1008 of file Dvbt1SymbolInterleaverComponent.cpp.

References [debug\\_x](#), [eo\\_](#), [H\\_2K\\_](#), [H\\_4K\\_](#), [H\\_8K\\_](#), [ofdmMode\\_x](#), [siLength\\_](#), [siOffset\\_](#), and [siRegister\\_](#).

```
1009 {
1010     // request input
1011     DataSet< uint8_t > *in = NULL;
1012     getInputDataSet("input1", in);
1013
1014     // calculate sizes
1015     int insize = in ? (int) in->data.size() : 0;
1016     int outsize = siLength_ * ((insize + siOffset_) / siLength_);
1017
1018     // request output
1019     DataSet< uint8_t > *out = NULL;
1020     getOutputDataSet("output1", out, outsize);
1021
1022     // print debug info
1023     if(debug_x)
1024         LOG(LINFO) << "in/out: " << insize << "/" << outsize;
1025
1026     // symbol by symbol
1027     for(ByteVecIt init = in->data.begin(), outit = out->data.begin(); init < in->data.end(); init++)
1028     {
1029         // copy to register
1030         siRegister_[siOffset_++] = *init;
1031
1032         // trigger interleaving
1033         if(siOffset_ == siLength_)
1034         {
1035             // reset offset
1036             siOffset_ = 0;
1037
1038             // actual interleaving matrix
1039             int *H = ofdmMode_x == 2048 ? H_2K_ : (ofdmMode_x == 4096 ?
H_4K_ : H_8K_);
1040 }
```

```

1041         // copy
1042         switch(eo_++ & 0x01)
1043         {
1044             case 0:
1045                 // even numbered symbol
1046                 for(int s = 0; s < siLength_; s++)
1047                     outit[H[s]] = siRegister_[s];
1048                 break;
1049             case 1:
1050                 // odd numbered symbol
1051                 for(int s = 0; s < siLength_; s++)
1052                     outit[s] = siRegister_[H[s]];
1053                 break;
1054         }
1055
1056         // advance output
1057         outit += siLength_;
1058     }
1059 }
1060
1061 // Copy the timestamp and sample rate for the DataSets
1062 out->timeStamp = in->timeStamp;
1063 out->sampleRate = in->sampleRate;
1064
1065 // release input and output
1066 releaseInputDataSet("input1", in);
1067 releaseOutputDataSet("output1", out);
1068 }

```

#### 7.14.4.8 void iris::phy::Dvbt1SymbolInterleaverComponent::registerPorts ( ) [virtual]

Register the interleaver ports with the IRIS system.

This component has two inputs that accept symbols (some bits per byte) and one output that provides symbols (in one byte).

Definition at line 89 of file Dvbt1SymbolInterleaverComponent.cpp.

```

90 {
91     registerInputPort("input1", TypeInfo< uint8_t >::identifier);
92     registerOutputPort("output1", TypeInfo< uint8_t >::identifier);
93 }

```

#### 7.14.4.9 void iris::phy::Dvbt1SymbolInterleaverComponent::setup ( ) [private]

Set up the register space and initialize.

Definition at line 1083 of file Dvbt1SymbolInterleaverComponent.cpp.

References eo\_, ofdmMode\_x, siLength\_, siOffset\_, and siRegister\_.

Referenced by initialize(), and parameterHasChanged().

```

1084 {
1085     // clean
1086     eo_ = 0;
1087     siOffset_ = 0;
1088     siLength_ = ofdmMode_x == 2048 ? 1512 : (ofdmMode_x == 4096 ? 3024 : 6048);
1089
1090     // get room
1091     siRegister_ = new uint8_t [siLength_];
1092 }

```

### 7.14.5 Member Data Documentation

#### 7.14.5.1 bool iris::phy::Dvbt1SymbolInterleaverComponent::debug\_x [private]

Debug flag (default = false)

Definition at line 112 of file Dvbt1SymbolInterleaverComponent.h.

Referenced by Dvbt1SymbolInterleaverComponent(), and process().

**7.14.5.2** `int iris::phy::Dvbt1SymbolInterleaverComponent::eo_ [private]`

Even/odd numbered OFDM block.

Definition at line 124 of file Dvbt1SymbolInterleaverComponent.h.

Referenced by `process()`, and `setup()`.

**7.14.5.3** `int iris::phy::Dvbt1SymbolInterleaverComponent::H_2K_ [static],[private]`

Interleaving addresses for 2K.

Definition at line 126 of file Dvbt1SymbolInterleaverComponent.h.

Referenced by `process()`.

**7.14.5.4** `int iris::phy::Dvbt1SymbolInterleaverComponent::H_4K_ [static],[private]`

Interleaving addresses for 4K.

Definition at line 126 of file Dvbt1SymbolInterleaverComponent.h.

Referenced by `process()`.

**7.14.5.5** `int iris::phy::Dvbt1SymbolInterleaverComponent::H_8K_ [static],[private]`

Interleaving addresses for 8K.

Definition at line 126 of file Dvbt1SymbolInterleaverComponent.h.

Referenced by `process()`.

**7.14.5.6** `int iris::phy::Dvbt1SymbolInterleaverComponent::ofdmMode_x [private]`

OFDM mode (default = 2048)

Definition at line 113 of file Dvbt1SymbolInterleaverComponent.h.

Referenced by `Dvbt1SymbolInterleaverComponent()`, `process()`, and `setup()`.

**7.14.5.7** `double iris::phy::Dvbt1SymbolInterleaverComponent::sampleRate_ [private]`

Sample rate of current frame.

Definition at line 119 of file Dvbt1SymbolInterleaverComponent.h.

**7.14.5.8** `int iris::phy::Dvbt1SymbolInterleaverComponent::siLength_ [private]`

Interleaving register length.

Definition at line 122 of file Dvbt1SymbolInterleaverComponent.h.

Referenced by `process()`, and `setup()`.

**7.14.5.9** `int iris::phy::Dvbt1SymbolInterleaverComponent::siOffset_ [private]`

Interleaving offset.

Definition at line 121 of file Dvbt1SymbolInterleaverComponent.h.

Referenced by `process()`, and `setup()`.

7.14.5.10 `uint8_t* iris::phy::Dvbt1SymbolInterleaverComponent::siRegister_` [private]

Actual interleaving register.

Definition at line 123 of file `Dvbt1SymbolInterleaverComponent.h`.

Referenced by `destroy()`, `process()`, and `setup()`.

7.14.5.11 `double iris::phy::Dvbt1SymbolInterleaverComponent::timeStamp_` [private]

Timestamp of current frame.

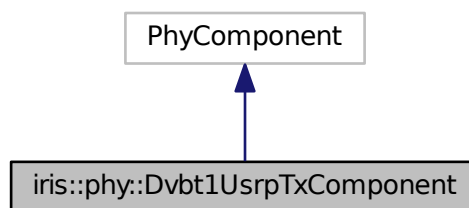
Definition at line 118 of file `Dvbt1SymbolInterleaverComponent.h`.

## 7.15 `iris::phy::Dvbt1UsrpTxComponent` Class Reference

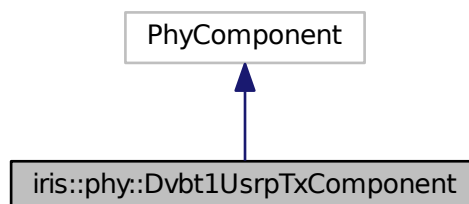
The `Dvbt1UsrpTx` component.

```
#include <Dvbt1UsrpTxComponent.h>
```

Inheritance diagram for `iris::phy::Dvbt1UsrpTxComponent`:



Collaboration diagram for `iris::phy::Dvbt1UsrpTxComponent`:



### Public Member Functions

- [`Dvbt1UsrpTxComponent`](#) (`std::string` name)

- virtual `~Dvbt1UsrpTxComponent ()`
- virtual void `calculateOutputTypes` (std::map< std::string, int > &inputTypes, std::map< std::string, int > &outputTypes)
- virtual void `registerPorts ()`
- virtual void `initialize ()`  
*Do any initialization required.*
- virtual void `process ()`
- virtual void `parameterHasChanged` (std::string name)  
*This gets called whenever a parameter is reconfigured.*
- void `usrpThreadProcedure ()`

### Private Attributes

- std::string `args_x`  
*See [http://files.ettus.com/uhd\\_docs/manual/html/identification.html](http://files.ettus.com/uhd_docs/manual/html/identification.html).*
- double `rate_x`  
*Rate of outgoing samples.*
- double `frequency_x`  
*Tx frequency.*
- double `fixLoOffset_x`  
*Fix the local oscillator offset (defaults to 2\*rate)*
- float `gain_x`  
*Overall tx gain.*
- std::string `antenna_x`  
*Daughterboard antenna selection.*
- std::string `subDev_x`  
*Daughterboard subdevice specification.*
- double `bw_x`  
*Daughterboard IF filter bandwidth (Hz)*
- std::string `ref_x`  
*Reference waveform (internal, external, mimo)*
- bool `streaming_x`  
*Streaming or bursty traffic?*
- std::string `fmt_x`  
*Data format (fc64, fc32 or sc16)*
- int `bufferSize_x`  
*Size (in samples) of a single buffer.*
- int `numBuffers_x`  
*Number of buffers.*
- ReadBuffer< std::complex  
< float > > \* `inBuf_`  
*Convenience pointer to input buffer.*
- uhd::usrp::multi\_usrp::sptr `usrp_`  
*The device.*
- uhd::tx\_streamer::sptr `txStream_`
- std::vector< std::vector  
< std::complex< float > > > `bufs_`
- std::vector< int > `fulls_`
- boost::condition\_variable `condW_`
- boost::condition\_variable `condR_`
- boost::mutex `mutW_`
- boost::mutex `mutR_`

- boost::mutex `mut_`
- int `currentRead_`
- int `currentWrite_`
- bool `runUsrc_`
- boost::thread \* `usrpThread_`

### 7.15.1 Detailed Description

The Dvbt1UsrcTx component.

A sink component which writes to a USRP transmitter using the Universal Hardware Driver (UHD). This component supports streaming data by default. This component is derived from the component used by the Iris modules. In addition to that, we use here additional threading and buffering to allow high data rate streams to be continuously and uninterruptedly transmitted from the input coming from other blocks to the real USRP device.

This block accepts in input complex float values.

There are several parameters that can be changed in the XML configuration file:

- *args*: by default set to "". This is a string that can be used to address one particular USRP device by specifying its IP address.
- *rate*: by default set to "1000000", it represents the sampling rate at which the digital samples are sent to the device.
- *frequency*: by default set to "2400000000", it is the frequency at which the BB signal, after digital to analog conversion, will be modulated.
- *gain*: by default set to "1", it is the gain of the final amplifier in the USRP TX chain.
- *streaming*: by default set to true, it states whether a continuous stream of samples has to be expected.
- *fixloffset*: by default set to "0", this is the offset at which the analog oscillator will up-convert the BB signal, with respect to the specified frequency. This offset is recovered in digital by means of a digitally controlled oscillator implemented in the USRP FPGA.
- *antenna*: by default set to "", which means automatic selection. This parameter can be used to select one particular antenna, if more than one is present.
- *subdev*: by default set to "", which means automatic selection. This parameter allows selecting one particular subdevice inside of the device, if more than one is present.
- *bw*: by default set to "0" Hz, which means automatic selection. This parameter selects the bandwidth of the daughterboard IF filter, if present.
- *ref*: can be one of "internal", "external", "mimo", by default it is set to "internal". This parameter represents the type of clock reference signal used for the synchronization of all the device chips.
- *fmt*: can be one of "fc64" (double), "fc32" (float) or "sc16" (short), by default set to "fc32". This is the default sample precision used for sample representation.
- *numbuffers*: by default set to "4". A number of internal buffers is required to temporarily store the samples that are sent to the USRP. If buffers are not used, there could be moments during which the USRP (which runs inside of an asynchronous thread) is lacking input samples, thus degrading the quality of the emitted signal.
- *buffersize*: by default set to "1000000" samples, this is the number of samples contained in one of the buffers mentioned above.

Definition at line 99 of file Dvbt1UsrcTxComponent.h.



## 7.15.2 Constructor & Destructor Documentation

### 7.15.2.1 iris::phy::Dvbt1UsrcTxComponent::Dvbt1UsrcTxComponent ( std::string *name* )

#### Constructor

Call the constructor on PhyComponent and pass in all details about the component. Register all parameters and events in the constructor.

#### Parameters

<i>name</i>	The name assigned to this component when loaded
-------------	---

Definition at line 57 of file Dvbt1UsrcTxComponent.cpp.

References antenna\_x, args\_x, bufferSize\_x, bw\_x, fixLoOffset\_x, fmt\_x, frequency\_x, gain\_x, numBuffers\_x, rate\_x, ref\_x, streaming\_x, and subDev\_x.

```

58 : PhyComponent (name,
59                 "dvbt1usrptx",
60                 "UsrcTx with buffering and sustained TX rate",
61                 "Giuseppe Baruffa",
62                 "0.1")
63 , currentRead_(0)
64 , currentWrite_(0)
65 , runUsrc_(true)
66 , usrpThread_(NULL)
67 {
68     /*
69     * format:
70     * registerParameter(name,
71     *                   description,
72     *                   default value,
73     *                   dynamic?,
74     *                   parameter,
75     *                   allowed values);
76     */
77     registerParameter("args",
78                     "A delimited string which may be used to specify a particular usrp",
79                     "",
80                     false,
81                     args_x);
82     registerParameter("rate",
83                     "The transmit rate",
84                     "1000000",
85                     true,
86                     rate_x);
87     registerParameter("frequency",
88                     "The transmit frequency",
89                     "2400000000",
90                     true,
91                     frequency_x);
92     registerParameter("gain",
93                     "The transmit gain",
94                     "1",
95                     true,
96                     gain_x);
97     registerParameter("streaming",
98                     "Whether we're streaming data to tx",
99                     "true",
100                    true,
101                    streaming_x);
102     registerParameter("fixlooffset",
103                     "Value to fix LO offset to in Hz",
104                     "0",
105                     false,
106                     fixLoOffset_x);
107     registerParameter("antenna",
108                     "Daughterboard antenna selection",
109                     "",
110                     false,
111                     antenna_x);
112     registerParameter("subdev",
113                     "Daughterboard subdevice specification",
114                     "",
115                     false,
116                     subDev_x);
117     registerParameter("bw",
118                     "Daughterboard IF filter bandwidth (Hz)",
119                     "0",
120                     false,
121                     bw_x);

```

```

122   registerParameter("ref",
123                     "Reference waveform (internal, external, mimo)",
124                     "internal",
125                     false,
126                     ref_x);
127   registerParameter("fmt",
128                     "Data format (fc64, fc32 or sc16)",
129                     "fc32",
130                     false,
131                     fmt_x);
132   registerParameter("numbuffers",
133                     "Number of buffers",
134                     "4",
135                     false,
136                     numBuffers_x);
137   registerParameter("buffersize",
138                     "Size of a buffer (in samples)",
139                     "1000000",
140                     false,
141                     bufferSize_x);
142 }

```

#### 7.15.2.2 iris::phy::Dvbt1UsrpTxComponent::~Dvbt1UsrpTxComponent ( ) [virtual]

Destructor

Send an EOB packet to stop the Usrp

Definition at line 148 of file Dvbt1UsrpTxComponent.cpp.

References `runUsrp_`, `txStream_`, and `usrpThread_`.

```

149 {
150     // stop thread
151     runUsrp_ = false;
152     usrpThread_>join();
153     delete usrpThread_;
154
155     //Send a mini EOB packet
156     uhd::tx_metadata_t md;
157     md.start_of_burst = false;
158     md.end_of_burst   = true;
159     vector< complex<float> > v;
160     #if 1
161     if(txStream_ != NULL)
162     {
163         txStream_>send(&v.front(), 0, md);
164     }
165     #endif
166 }

```

### 7.15.3 Member Function Documentation

#### 7.15.3.1 void iris::phy::Dvbt1UsrpTxComponent::calculateOutputTypes ( std::map< std::string, int > & inputTypes, std::map< std::string, int > & outputTypes ) [virtual]

Calculate output data types

Based on the input data types, tell the system what output data types will be provided.

Parameters

<i>inputTypes</i>	The data types of the inputs which will be passed to this component
<i>outputTypes</i>	The data types of the outputs which will be generated by this component

Definition at line 189 of file Dvbt1UsrpTxComponent.cpp.

```

192 {
193     //No output types
194 }

```

## 7.15.3.2 void iris::phy::Dvbt1UsrpTxComponent::initialize ( ) [virtual]

Do any initialization required.

Definition at line 197 of file Dvbt1UsrpTxComponent.cpp.

References antenna\_x, args\_x, bufferSize\_x, bufs\_, bw\_x, currentRead\_, currentWrite\_, fixLoOffset\_x, fmt\_x, frequency\_x, fulls\_, gain\_x, inBuf\_, numBuffers\_x, rate\_x, ref\_x, runUsrp\_, subDev\_x, txStream\_, usrp\_, usrpThread\_, and usrpThreadProcedure().

```

198 {
199     //uhd::set_thread_priority_safe();
200
201     //Set up the input DataBuffer
202     inBuf_ = castToType< complex<float> >(inputBuffers.at(0));
203
204     // prepare buffers
205     bufs_.resize(numBuffers_x);
206     fulls_.resize(numBuffers_x);
207     bufs_.resize(numBuffers_x);
208     for(int i = 0; i < numBuffers_x; i++)
209     {
210         bufs_[i].resize(bufferSize_x);
211         fulls_[i] = 0;
212     }
213     currentRead_ = numBuffers_x - 1;
214     currentWrite_ = 0;
215
216     // the thread
217     if(usrpThread_)
218     {
219         runUsrp_ = false;
220         usrpThread_>join();
221         delete usrpThread_;
222     }
223     runUsrp_ = true;
224     usrpThread_ = new boost::thread(boost::bind(&
225         Dvbt1UsrpTxComponent::usrpThreadProcedure, this));
226
227     #if 1
228     //Set up the usrp
229     {
230         //Create the device
231         LOG(LINFO) << "Creating the usrp device with args: " << args_x;
232         usrp_ = uhd::usrp::multi_usrp::make(args_x);
233         //Lock mboard clocks
234         usrp_>set_clock_source(ref_x);
235         //always select the subdevice first, the channel mapping affects the other settings
236         if (subDev_x!="")
237             usrp_>set_tx_subdev_spec(subDev_x);
238         LOG(LINFO) << "Using Device: " << usrp_>get_pp_string();
239
240         //Set rate
241         LOG(LINFO) << "Setting TX Rate: " << (rate_x/1e6) << "MSPS...";
242         usrp_>set_tx_rate(rate_x);
243         LOG(LINFO) << "Actual TX Rate: " << (usrp_>get_tx_rate()/1e6) << "MSPS...";
244
245         //Set frequency
246         LOG(LINFO) << "Setting TX Frequency: " << (frequency_x/1e6) << "MHz...";
247         double lo_offset = 0; //Set LO offset to zero by default
248         if(fixLoOffset_x >= 0)
249             lo_offset = fixLoOffset_x;
250         usrp_>set_tx_freq(tune_request_t(frequency_x, lo_offset));
251         LOG(LINFO) << "Actual TX Frequency: " << (usrp_>get_tx_freq()/1e6) << "MHz";
252         LOG(LINFO) << "RX LO offset: " << (lo_offset/1e6) << "MHz...";
253
254         //We can only set the time on usrp2 devices
255         if(usrp_>get_mboard_name().find("usrp1") == string::npos)
256         {
257             LOG(LINFO) << "Setting device timestamp to 0...";
258             usrp_>set_time_now(uhd::time_spec_t((double)0));
259         }
260
261         //set the rf gain
262         gain_range_t range = usrp_>get_tx_gain_range();
263         LOG(LINFO) << "Gain range: " << range.to_pp_string();
264         LOG(LINFO) << "Setting TX Gain: " << gain_x << " dB...";
265         usrp_>set_tx_gain(gain_x);
266         LOG(LINFO) << "Actual TX Gain: " << usrp_>get_tx_gain() << " dB...";
267
268         //set the IF filter bandwidth
269         if(bw_x!=0)
270         {

```

```

271     LOG(LINFO) << "Setting TX Bandwidth: " << bw_x << " MHz...";
272     usrp_>set_tx_bandwidth(bw_x);
273     LOG(LINFO) << "Actual TX Bandwidth: " << usrp_>get_tx_bandwidth() << " MHz...";
274 }
275
276 //Set the antenna
277 if(antenna_x!="")
278     usrp_>set_tx_antenna(boost::to_upper_copy(antenna_x));
279 LOG(LINFO) << "Using TX Antenna: " << usrp_>get_tx_antenna();
280
281 boost::this_thread::sleep(boost::posix_time::seconds(1)); //allow for some setup time
282
283 //Check Ref and LO Lock detect
284 std::vector<std::string> sensor_names;
285 sensor_names = usrp_>get_tx_sensor_names(0);
286 if (std::find(sensor_names.begin(),
287             sensor_names.end(),
288             "lo_locked") != sensor_names.end())
289 {
290     uhd::sensor_value_t lo_locked = usrp_>get_tx_sensor("lo_locked",0);
291     LOG(LINFO) << "Checking TX: " << lo_locked.to_pp_string() << " ...";
292     if(!lo_locked.to_bool())
293         throw IrisException("Failed to lock LO");
294 }
295 sensor_names = usrp_>get_mboard_sensor_names(0);
296 if ((ref_x == "mimo") and (std::find(sensor_names.begin(),
297                                     sensor_names.end(),
298                                     "mimo_locked") != sensor_names.end()))
299 {
300     uhd::sensor_value_t mimo_locked = usrp_>get_mboard_sensor("mimo_locked",0);
301     LOG(LINFO) << "Checking TX: " << mimo_locked.to_pp_string() << " ...";
302     if(!mimo_locked.to_bool())
303         throw IrisException("Failed to lock LO");
304 }
305 if ((ref_x == "external") and (std::find(sensor_names.begin(),
306                                           sensor_names.end(),
307                                           "ref_locked") != sensor_names.end()))
308 {
309     uhd::sensor_value_t ref_locked = usrp_>get_mboard_sensor("ref_locked",0);
310     LOG(LINFO) << "Checking TX: " << ref_locked.to_pp_string() << " ...";
311     if(!ref_locked.to_bool())
312         throw IrisException("Failed to lock LO");
313 }
314
315 //create a transmit streamer
316 uhd::stream_args_t stream_args(fmt_x);
317 txStream_ = usrp_>get_tx_stream(stream_args);
318 }
319 catch(std::exception& e)
320 {
321     throw IrisException(e.what());
322 }
323 #endif
324 }

```

### 7.15.3.3 void iris::phy::Dvbt1UsrpTxComponent::parameterHasChanged ( std::string name ) [virtual]

This gets called whenever a parameter is reconfigured.

Definition at line 476 of file Dvbt1UsrpTxComponent.cpp.

References fixLoOffset\_x, frequency\_x, gain\_x, rate\_x, and usrp\_.

```

477 {
478     #if 1
479     try
480     {
481         if(name == "frequency")
482         {
483             LOG(LINFO) << "Setting TX Frequency: " << (frequency_x/1e6) << "MHz...";
484             double lo_offset = 2*rate_x; //Set LO offset to twice signal rate by default
485             if(fixLoOffset_x >= 0)
486             {
487                 lo_offset = fixLoOffset_x;
488             }
489             usrp_>set_tx_freq(tune_request_t(frequency_x, lo_offset));
490             LOG(LINFO) << "LOG TX Frequency: " << (usrp_>get_tx_freq()/1e6) << "MHz";
491         }
492         else if(name == "rate")
493         {
494             LOG(LINFO) << "Setting TX Rate: " << (rate_x/1e6) << "Msps...";
495             usrp_>set_tx_rate(rate_x);

```

```

496     LOG(LINFO) << "Actual TX Rate: " << (usrp_>get_tx_rate()/1e6) << "Mpsps...";
497 }
498 else if(name == "gain")
499 {
500     gain_range_t range = usrp_>get_tx_gain_range();
501     LOG(LINFO) << "Gain range: " << range.to_pp_string();
502     LOG(LINFO) << "Setting TX Gain: " << gain_x << " dB...";
503     usrp_>set_tx_gain(gain_x);
504     LOG(LINFO) << "Actual TX Gain: " << usrp_>get_tx_gain() << " dB...";
505 }
506 }
507 catch(std::exception &e)
508 {
509     throw IrisException(e.what());
510 }
511 #endif
512 }

```

#### 7.15.3.4 void iris::phy::Dvbt1UsrpTxComponent::process ( ) [virtual]

The main work of the component is carried out here

Take a DataSet from the input buffer and send to the usrp

Definition at line 345 of file Dvbt1UsrpTxComponent.cpp.

References bufferSize\_x, bufs\_, condR\_, condW\_, currentRead\_, currentWrite\_, dbgprintf, DUMP\_STATUS, fulls\_, inBuf\_, mut\_, mutR\_, and numBuffers\_x.

```

346 {
347     //Get a DataSet from the input DataBuffer
348     DataSet< complex<float> >* readDataSet = NULL;
349     inBuf_>getReadData(readDataSet);
350
351     size_t insize = readDataSet->data.size();
352
353     // check buffers
354     int remSize = insize;
355     while(remSize > 0)
356     {
357         bool have_to_wait = true;
358         int availSize = bufferSize_x - fulls_[currentWrite_];
359         if(availSize > 0)
360         {
361             // there is room in this buffer
362             dbgprintf("filling W(%d)...\n", currentWrite_);
363             DUMP_STATUS();
364             int dasize = min(availSize, remSize);
365             copy(&(readDataSet->data[insize - remSize]), &(readDataSet->data[insize - remSize + dasize]), &(
366             bufs_[currentWrite_][fulls_[currentWrite_]));
367             fulls_[currentWrite_] += dasize;
368             remSize -= dasize;
369         }
370         else
371         {
372             // try to change buffer
373             boost::lock_guard<boost::mutex> lock(mut_);
374             int nextWrite_ = currentWrite_ == (numBuffers_x - 1) ? 0 : (
375             currentWrite_ + 1);
376             dbgprintf("looking for W(%d)...\n", nextWrite_);
377             DUMP_STATUS();
378             if(nextWrite_ != currentRead_)
379             {
380                 // update buffer
381                 currentWrite_ = nextWrite_;
382                 // notify the reader that the writer has done something
383                 condW_.notify_one();
384                 dbgprintf("notified a write\n");
385                 have_to_wait = false;
386             }
387         }
388         // wait for a free read buffer to write into
389         if(have_to_wait)
390         {
391             dbgprintf("awaiting a read...\n");
392             boost::unique_lock<boost::mutex> lockR(mutR_);
393             condR_.wait(lockR);
394             dbgprintf("awaited a read\n");
395         }
396     }
397 }

```

```

396     }
397 }
398
399 //Release the DataSet
400 inBuf_>releaseReadData(readDataSet);
401 }

```

### 7.15.3.5 void iris::phy::Dvbt1UsrpTxComponent::registerPorts ( ) [virtual]

Register the ports of this component

Ports are registered by name with a vector of valid data types permitted on those ports. This example has one input port with a single valid data type - complex<float>.

Definition at line 173 of file Dvbt1UsrpTxComponent.cpp.

```

174 {
175     //Register all ports
176     vector<int> validTypes;
177     validTypes.push_back(TypeInfo< complex<float> >::identifier);
178
179     //format: (name, vector of valid types)
180     registerInputPort("input1", validTypes);
181 }

```

### 7.15.3.6 void iris::phy::Dvbt1UsrpTxComponent::usrpThreadProcedure ( )

Definition at line 404 of file Dvbt1UsrpTxComponent.cpp.

References bufferSize\_x, bufs\_, condR\_, condW\_, currentRead\_, currentWrite\_, dbgprintf, DUMP\_STATUS, fulls\_, mut\_, mutW\_, numBuffers\_x, runUsrp\_, and txStream\_.

Referenced by initialize().

```

405 {
406     uhd::tx_metadata_t md;
407     md.start_of_burst = false;
408     md.end_of_burst = false;
409     md.has_time_spec = false;
410     double max_waiting_time = 0.5;
411
412     uhd::set_thread_priority_safe();
413
414     // wait to avoid premature death
415     {
416         dbgprintf("awaiting a write...\n");
417         boost::unique_lock<boost::mutex> lockW(mutW_);
418         condW_.wait(lockW);
419         dbgprintf("awaited a write\n");
420     }
421
422     while(runUsrp_)
423     {
424         DUMP_STATUS();
425
426         // data available? send data
427         size_t num_tx_samps = 0;
428         while(fulls_[currentRead_])
429         {
430             dbgprintf("pouring R(%d)...\n", currentRead_);
431
432             num_tx_samps = txStream_>send(
433                 &bufs_[currentRead_][num_tx_samps], bufferSize_x - num_tx_samps, md,
434                 max_waiting_time
435             );
436             fulls_[currentRead_] -= num_tx_samps;
437
438             /*boost::this_thread::sleep(boost::posix_time::milliseconds(200));
439             DUMP_STATUS();
440             fulls_[currentRead_] -= 300000;
441             if(fulls_[currentRead_] < 0)
442                 fulls_[currentRead_] = 0;*/
443
444             /*boost::this_thread::sleep(boost::posix_time::milliseconds(1));
445             fulls_[currentRead_] = 0;*/

```

```

445     }
446
447     // go to next read
448     bool have_to_wait = true;
449     {
450         boost::lock_guard<boost::mutex> lock(mut_);
451         int nextRead_ = currentRead_ == (numBuffers_x - 1) ? 0 : (currentRead_ + 1);
452         dbgprintf("advancing R(%d)...\n", nextRead_);
453         DUMP_STATUS();
454         if(nextRead_ != currentWrite_)
455         {
456             // change and notify the writer that the reader has done something
457             currentRead_ = nextRead_;
458             condR_.notify_one();
459             dbgprintf("notified a read\n");
460             have_to_wait = false;
461         }
462     }
463
464     // wait the writer
465     if(have_to_wait)
466     {
467         dbgprintf("awaiting a write\n");
468         boost::unique_lock<boost::mutex> lockW(mutW_);
469         condW_.wait(lockW);
470         dbgprintf("awaited a write\n");
471     }
472 }
473 }

```

## 7.15.4 Member Data Documentation

### 7.15.4.1 std::string iris::phy::Dvbt1UsrpTxComponent::antenna\_x [private]

Daughterboard antenna selection.

Definition at line 121 of file Dvbt1UsrpTxComponent.h.

Referenced by Dvbt1UsrpTxComponent(), and initialize().

### 7.15.4.2 std::string iris::phy::Dvbt1UsrpTxComponent::args\_x [private]

See [http://files.ettus.com/uhd\\_docs/manual/html/identification.html](http://files.ettus.com/uhd_docs/manual/html/identification.html).

Definition at line 116 of file Dvbt1UsrpTxComponent.h.

Referenced by Dvbt1UsrpTxComponent(), and initialize().

### 7.15.4.3 int iris::phy::Dvbt1UsrpTxComponent::bufferSize\_x [private]

Size (in samples) of a single buffer.

Definition at line 127 of file Dvbt1UsrpTxComponent.h.

Referenced by Dvbt1UsrpTxComponent(), initialize(), process(), and usrpThreadProcedure().

### 7.15.4.4 std::vector< std::vector< std::complex<float> > > iris::phy::Dvbt1UsrpTxComponent::bufs\_ [private]

Definition at line 133 of file Dvbt1UsrpTxComponent.h.

Referenced by initialize(), process(), and usrpThreadProcedure().

### 7.15.4.5 double iris::phy::Dvbt1UsrpTxComponent::bw\_x [private]

Daughterboard IF filter bandwidth (Hz)

Definition at line 123 of file Dvbt1UsrpTxComponent.h.

Referenced by Dvbt1UsrpTxComponent(), and initialize().

#### 7.15.4.6 `boost::condition_variable iris::phy::Dvbt1UsrpTxComponent::condR_ [private]`

Definition at line 135 of file Dvbt1UsrpTxComponent.h.

Referenced by process(), and usrpThreadProcedure().

#### 7.15.4.7 `boost::condition_variable iris::phy::Dvbt1UsrpTxComponent::condW_ [private]`

Definition at line 135 of file Dvbt1UsrpTxComponent.h.

Referenced by process(), and usrpThreadProcedure().

#### 7.15.4.8 `int iris::phy::Dvbt1UsrpTxComponent::currentRead_ [private]`

Definition at line 138 of file Dvbt1UsrpTxComponent.h.

Referenced by initialize(), process(), and usrpThreadProcedure().

#### 7.15.4.9 `int iris::phy::Dvbt1UsrpTxComponent::currentWrite_ [private]`

Definition at line 138 of file Dvbt1UsrpTxComponent.h.

Referenced by initialize(), process(), and usrpThreadProcedure().

#### 7.15.4.10 `double iris::phy::Dvbt1UsrpTxComponent::fixLoOffset_x [private]`

Fix the local oscillator offset (defaults to 2\*rate)

Definition at line 119 of file Dvbt1UsrpTxComponent.h.

Referenced by Dvbt1UsrpTxComponent(), initialize(), and parameterHasChanged().

#### 7.15.4.11 `std::string iris::phy::Dvbt1UsrpTxComponent::fmt_x [private]`

Data format (fc64, fc32 or sc16)

Definition at line 126 of file Dvbt1UsrpTxComponent.h.

Referenced by Dvbt1UsrpTxComponent(), and initialize().

#### 7.15.4.12 `double iris::phy::Dvbt1UsrpTxComponent::frequency_x [private]`

Tx frequency.

Definition at line 118 of file Dvbt1UsrpTxComponent.h.

Referenced by Dvbt1UsrpTxComponent(), initialize(), and parameterHasChanged().

#### 7.15.4.13 `std::vector<int> iris::phy::Dvbt1UsrpTxComponent::fulls_ [private]`

Definition at line 134 of file Dvbt1UsrpTxComponent.h.

Referenced by initialize(), process(), and usrpThreadProcedure().



**7.15.4.14** float iris::phy::Dvbt1UsrpTxComponent::gain\_x [private]

Overall tx gain.

Definition at line 120 of file Dvbt1UsrpTxComponent.h.

Referenced by Dvbt1UsrpTxComponent(), initialize(), and parameterHasChanged().

**7.15.4.15** ReadBuffer< std::complex<float> >\* iris::phy::Dvbt1UsrpTxComponent::inBuf\_ [private]

Convenience pointer to input buffer.

Definition at line 130 of file Dvbt1UsrpTxComponent.h.

Referenced by initialize(), and process().

**7.15.4.16** boost::mutex iris::phy::Dvbt1UsrpTxComponent::mut\_ [private]

Definition at line 137 of file Dvbt1UsrpTxComponent.h.

Referenced by process(), and usrpThreadProcedure().

**7.15.4.17** boost::mutex iris::phy::Dvbt1UsrpTxComponent::mutR\_ [private]

Definition at line 136 of file Dvbt1UsrpTxComponent.h.

Referenced by process().

**7.15.4.18** boost::mutex iris::phy::Dvbt1UsrpTxComponent::mutW\_ [private]

Definition at line 136 of file Dvbt1UsrpTxComponent.h.

Referenced by usrpThreadProcedure().

**7.15.4.19** int iris::phy::Dvbt1UsrpTxComponent::numBuffers\_x [private]

Number of buffers.

Definition at line 128 of file Dvbt1UsrpTxComponent.h.

Referenced by Dvbt1UsrpTxComponent(), initialize(), process(), and usrpThreadProcedure().

**7.15.4.20** double iris::phy::Dvbt1UsrpTxComponent::rate\_x [private]

Rate of outgoing samples.

Definition at line 117 of file Dvbt1UsrpTxComponent.h.

Referenced by Dvbt1UsrpTxComponent(), initialize(), and parameterHasChanged().

**7.15.4.21** std::string iris::phy::Dvbt1UsrpTxComponent::ref\_x [private]

Reference waveform (internal, external, mimo)

Definition at line 124 of file Dvbt1UsrpTxComponent.h.

Referenced by Dvbt1UsrpTxComponent(), and initialize().

**7.15.4.22** `bool iris::phy::Dvbt1UsrpTxComponent::runUsrp_` `[private]`

Definition at line 139 of file Dvbt1UsrpTxComponent.h.

Referenced by `initialize()`, `usrpThreadProcedure()`, and `~Dvbt1UsrpTxComponent()`.

**7.15.4.23** `bool iris::phy::Dvbt1UsrpTxComponent::streaming_x` `[private]`

Streaming or bursty traffic?

Definition at line 125 of file Dvbt1UsrpTxComponent.h.

Referenced by `Dvbt1UsrpTxComponent()`.

**7.15.4.24** `std::string iris::phy::Dvbt1UsrpTxComponent::subDev_x` `[private]`

Daughterboard subdevice specification.

Definition at line 122 of file Dvbt1UsrpTxComponent.h.

Referenced by `Dvbt1UsrpTxComponent()`, and `initialize()`.

**7.15.4.25** `uhd::tx_streamer::sptr iris::phy::Dvbt1UsrpTxComponent::txStream_` `[private]`

Definition at line 132 of file Dvbt1UsrpTxComponent.h.

Referenced by `initialize()`, `usrpThreadProcedure()`, and `~Dvbt1UsrpTxComponent()`.

**7.15.4.26** `uhd::usrp::multi_usrp::sptr iris::phy::Dvbt1UsrpTxComponent::usrp_` `[private]`

The device.

Definition at line 131 of file Dvbt1UsrpTxComponent.h.

Referenced by `initialize()`, and `parameterHasChanged()`.

**7.15.4.27** `boost::thread* iris::phy::Dvbt1UsrpTxComponent::usrpThread_` `[private]`

Definition at line 140 of file Dvbt1UsrpTxComponent.h.

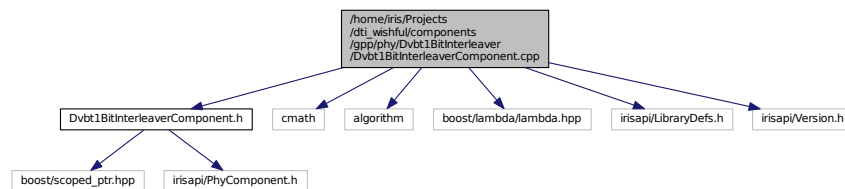
Referenced by `initialize()`, and `~Dvbt1UsrpTxComponent()`.

## Chapter 8

# File Documentation

### 8.1 /home/iris/Projects/dti\_wishful/components/gpp/phy/Dvbt1BitInterleaver/Dvbt1BitInterleaverComponent.cpp File Reference

```
#include "Dvbt1BitInterleaverComponent.h"
#include <cmath>
#include <algorithm>
#include <boost/lambda/lambda.hpp>
#include "irisapi/LibraryDefs.h"
#include "irisapi/Version.h"
Include dependency graph for Dvbt1BitInterleaverComponent.cpp:
```



### Namespaces

- [iris](#)
- [iris::phy](#)

### Functions

- [iris::phy::IRIS\\_COMPONENT\\_EXPORTS](#) (PhyComponent, Dvbt1BitInterleaverComponent)

#### 8.1.1 Detailed Description

**Version**

0.1

**8.1.2 COPYRIGHT**

Copyright 2012-2016 The Iris Project Developers. See the COPYRIGHT file at the top-level directory of this distribution and at <http://www.softwareradiosystems.com/iris/copyright.html>.

**8.1.3 LICENSE**

This file is part of the Iris Project.

Iris is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

Iris is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

A copy of the GNU Lesser General Public License can be found in the LICENSE file in the top-level directory of this distribution and at <http://www.gnu.org/licenses/>.

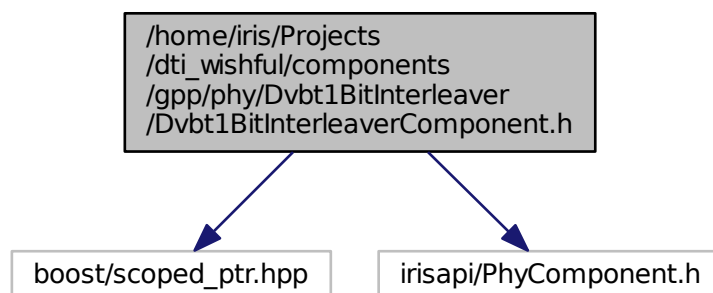
**8.1.4 DESCRIPTION**

Implementation of the Dvbt1BitInterleaver component.

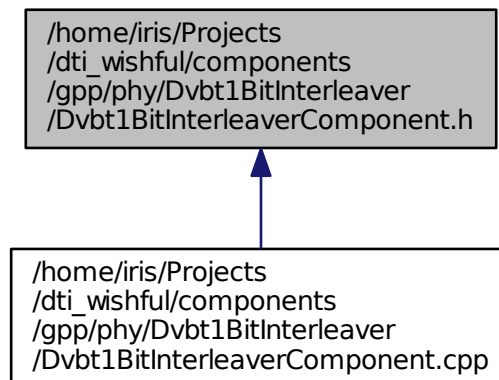
Definition in file [Dvbt1BitInterleaverComponent.cpp](#).

**8.2 /home/iris/Projects/dti\_wishful/components/gpp/phy/Dvbt1BitInterleaver/Dvbt1BitInterleaverComponent.h File Reference**

```
#include <boost/scoped_ptr.hpp>
#include "irisapi/PhyComponent.h"
Include dependency graph for Dvbt1BitInterleaverComponent.h:
```



This graph shows which files directly or indirectly include this file:



## Classes

- class [iris::phy::Dvbt1BitInterleaverComponent](#)  
A DVB-T1 bit interleaver component.

## Namespaces

- [iris](#)
- [iris::phy](#)

### 8.2.1 Detailed Description

#### Version

0.1

### 8.2.2 COPYRIGHT

Copyright 2012-2016 The Iris Project Developers. See the COPYRIGHT file at the top-level directory of this distribution and at <http://www.softwareradiosystems.com/iris/copyright.html>.

### 8.2.3 LICENSE

This file is part of the Iris Project.

Iris is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

Iris is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

A copy of the GNU Lesser General Public License can be found in the LICENSE file in the top-level directory of this distribution and at <http://www.gnu.org/licenses/>.

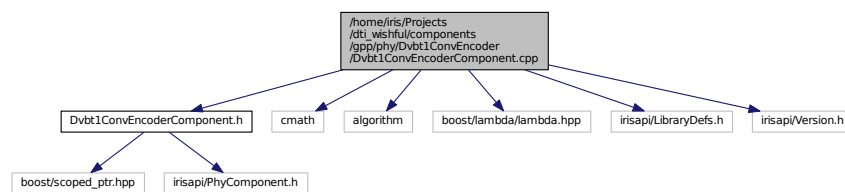
### 8.2.4 DESCRIPTION

The Dvbt1BitInterleaver component.

Definition in file [Dvbt1BitInterleaverComponent.h](#).

## 8.3 /home/iris/Projects/dti\_wishful/components/gpp/phy/Dvbt1ConvEncoder/Dvbt1ConvEncoderComponent.cpp File Reference

```
#include "Dvbt1ConvEncoderComponent.h"
#include <cmath>
#include <algorithm>
#include <boost/lambda/lambda.hpp>
#include "irisapi/LibraryDefs.h"
#include "irisapi/Version.h"
Include dependency graph for Dvbt1ConvEncoderComponent.cpp:
```



### Namespaces

- [iris](#)
- [iris::phy](#)

### Macros

- `#define g1 0x4f`  
*First polynomial (bit-reversed)*
- `#define g2 0x6d`  
*Second polynomial (bit-reversed)*

### Functions

- [iris::phy::IRIS\\_COMPONENT\\_EXPORTS](#) (PhyComponent, Dvbt1ConvEncoderComponent)

### 8.3.1 Detailed Description

#### Version

0.1

**8.3.2 COPYRIGHT**

Copyright 2012-2016 The Iris Project Developers. See the COPYRIGHT file at the top-level directory of this distribution and at <http://www.softwareradiosystems.com/iris/copyright.html>.

**8.3.3 LICENSE**

This file is part of the Iris Project.

Iris is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

Iris is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

A copy of the GNU Lesser General Public License can be found in the LICENSE file in the top-level directory of this distribution and at <http://www.gnu.org/licenses/>.

**8.3.4 DESCRIPTION**

Implementation of the Dvbt1ConvEncoder component.

Definition in file [Dvbt1ConvEncoderComponent.cpp](#).

**8.3.5 Macro Definition Documentation****8.3.5.1 #define g1 0x4f**

First polynomial (bit-reversed)

Definition at line 146 of file Dvbt1ConvEncoderComponent.cpp.

Referenced by `iris::phy::Dvbt1ConvEncoderComponent::process()`.

**8.3.5.2 #define g2 0x6d**

Second polynomial (bit-reversed)

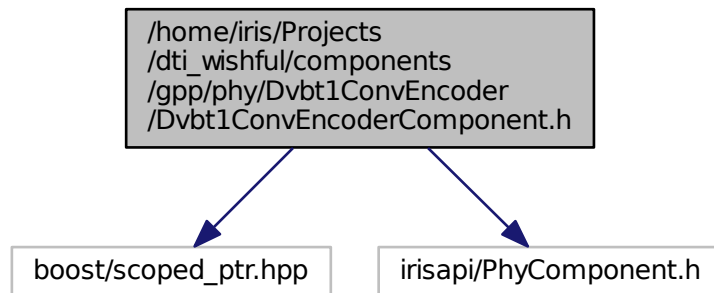
Definition at line 149 of file Dvbt1ConvEncoderComponent.cpp.

Referenced by `iris::phy::Dvbt1ConvEncoderComponent::process()`.

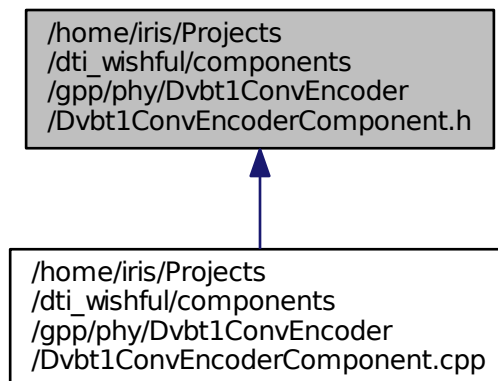
**8.4 /home/iris/Projects/dti\_wishful/components/gpp/phy/Dvbt1ConvEncoder/Dvbt1ConvEncoderComponent.h File Reference**

```
#include <boost/scoped_ptr.hpp>
#include "irisapi/PhyComponent.h"
```

Include dependency graph for Dvbt1ConvEncoderComponent.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class `iris::phy::Dvbt1ConvEncoderComponent`  
A DVB-T1 convolutional encoder component.

## Namespaces

- `iris`
- `iris::phy`

### 8.4.1 Detailed Description



0.1

## 8.4.2 COPYRIGHT

Copyright 2012-2016 The Iris Project Developers. See the COPYRIGHT file at the top-level directory of this distribution and at <http://www.software radiosystems.com/iris/copyright.html>.

## 8.4.3 LICENSE

This file is part of the Iris Project.

Iris is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

Iris is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

A copy of the GNU Lesser General Public License can be found in the LICENSE file in the top-level directory of this distribution and at <http://www.gnu.org/licenses/>.

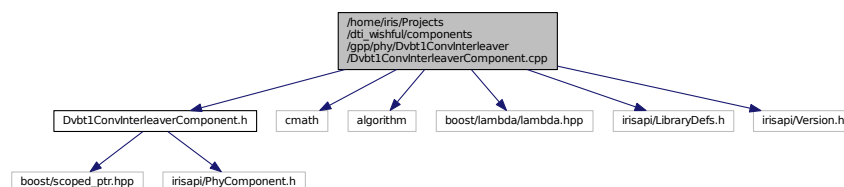
## 8.4.4 DESCRIPTION

The Dvbt1ConvEncoder component.

Definition in file [Dvbt1ConvEncoderComponent.h](#).

## 8.5 /home/iris/Projects/dti\_wishful/components/gpp/phy/Dvbt1ConvInterleaver/Dvbt1-ConvInterleaverComponent.cpp File Reference

```
#include "Dvbt1ConvInterleaverComponent.h"
#include <cmath>
#include <algorithm>
#include <boost/lambda/lambda.hpp>
#include "irisapi/LibraryDefs.h"
#include "irisapi/Version.h"
Include dependency graph for Dvbt1ConvInterleaverComponent.cpp:
```



## Namespaces

- [iris](#)
- [iris::phy](#)

## Functions

- [iris::phy::IRIS\\_COMPONENT\\_EXPORTS](#) (PhyComponent, Dvbt1ConvInterleaverComponent)

### 8.5.1 Detailed Description

#### Version

0.1

### 8.5.2 COPYRIGHT

Copyright 2012-2016 The Iris Project Developers. See the COPYRIGHT file at the top-level directory of this distribution and at <http://www.software radiosystems.com/iris/copyright.html>.

### 8.5.3 LICENSE

This file is part of the Iris Project.

Iris is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

Iris is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

A copy of the GNU Lesser General Public License can be found in the LICENSE file in the top-level directory of this distribution and at <http://www.gnu.org/licenses/>.

### 8.5.4 DESCRIPTION

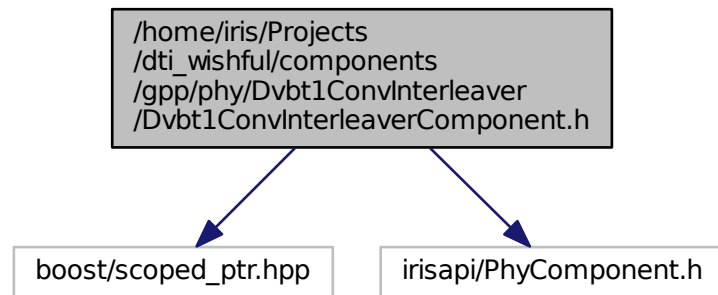
Implementation of the Dvbt1ConvInterleaver component.

Definition in file [Dvbt1ConvInterleaverComponent.cpp](#).

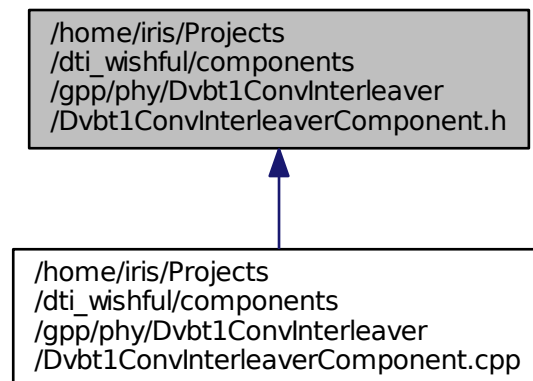
## 8.6 /home/iris/Projects/dti\_wishful/components/gpp/phy/Dvbt1ConvInterleaver/Dvbt1-ConvInterleaverComponent.h File Reference

```
#include <boost/scoped_ptr.hpp>
#include "irisapi/PhyComponent.h"
```

Include dependency graph for Dvbt1ConvInterleaverComponent.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [iris::phy::Dvbt1ConvInterleaverComponent](#)  
A DVB-T1 convolutional interleaver component.

## Namespaces

- [iris](#)
- [iris::phy](#)

### 8.6.1 Detailed Description

## Version

0.1

## 8.6.2 COPYRIGHT

Copyright 2012-2016 The Iris Project Developers. See the COPYRIGHT file at the top-level directory of this distribution and at <http://www.software-radiosystems.com/iris/copyright.html>.

## 8.6.3 LICENSE

This file is part of the Iris Project.

Iris is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

Iris is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

A copy of the GNU Lesser General Public License can be found in the LICENSE file in the top-level directory of this distribution and at <http://www.gnu.org/licenses/>.

## 8.6.4 DESCRIPTION

The Dvbt1ConvInterleaver component.

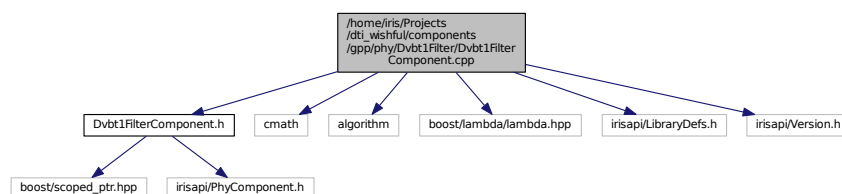
Definition in file [Dvbt1ConvInterleaverComponent.h](#).

## 8.7 /home/iris/Projects/dti\_wishful/components/gpp/phy/Dvbt1Filter/Dvbt1FilterComponent.cpp

### File Reference

```
#include "Dvbt1FilterComponent.h"
#include <cmath>
#include <algorithm>
#include <boost/lambda/lambda.hpp>
#include "irisapi/LibraryDefs.h"
#include "irisapi/Version.h"
```

Include dependency graph for Dvbt1FilterComponent.cpp:



## Namespaces

- `iris`
- `iris::phy`

## Macros

- #define `MAX_FILTER_LENGTH` 127  
*Change this to suit your needs.*
- #define `MAX_FILTER_LENGTH_2` 50001  
*unused*

## Functions

- `iris::phy::IRIS_COMPONENT_EXPORTS` (PhyComponent, Dvbt1FilterComponent)

### 8.7.1 Detailed Description

#### Version

0.1

### 8.7.2 COPYRIGHT

Copyright 2012-2016 The Iris Project Developers. See the COPYRIGHT file at the top-level directory of this distribution and at <http://www.software radiosystems.com/iris/copyright.html>.

### 8.7.3 LICENSE

This file is part of the Iris Project.

Iris is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

Iris is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

A copy of the GNU Lesser General Public License can be found in the LICENSE file in the top-level directory of this distribution and at <http://www.gnu.org/licenses/>.

### 8.7.4 DESCRIPTION

Implementation of the Dvbt1Filter component.

Definition in file [Dvbt1FilterComponent.cpp](#).

### 8.7.5 Macro Definition Documentation

#### 8.7.5.1 #define `MAX_FILTER_LENGTH` 127

Change this to suit your needs.

Definition at line 330 of file `Dvbt1FilterComponent.cpp`.

Referenced by `iris::phy::Dvbt1FilterComponent::setup()`.

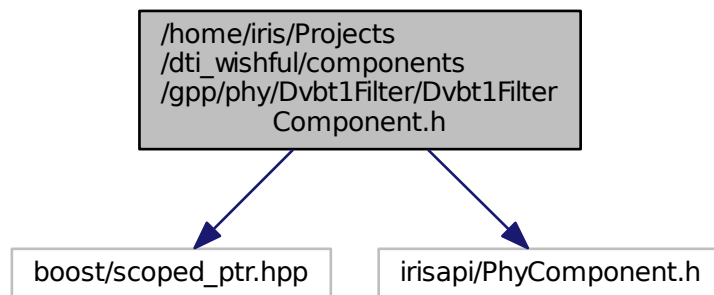
8.7.5.2 `#define MAX_FILTER_LENGTH_2 50001`

unused

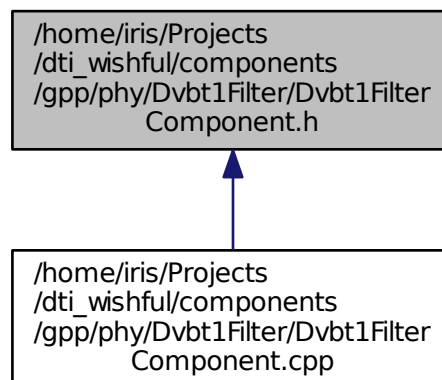
Definition at line 333 of file Dvbt1FilterComponent.cpp.

## 8.8 `/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1Filter/Dvbt1FilterComponent.h` File Reference

```
#include <boost/scoped_ptr.hpp>
#include "irisapi/PhyComponent.h"
Include dependency graph for Dvbt1FilterComponent.h:
```



This graph shows which files directly or indirectly include this file:



## Classes

- class [iris::phy::Dvbt1FilterComponent](#)  
*A DVB-T1 filter component.*

## Namespaces

- [iris](#)
- [iris::phy](#)

### 8.8.1 Detailed Description

#### Version

0.1

### 8.8.2 COPYRIGHT

Copyright 2012-2016 The Iris Project Developers. See the COPYRIGHT file at the top-level directory of this distribution and at <http://www.software radiosystems.com/iris/copyright.html>.

### 8.8.3 LICENSE

This file is part of the Iris Project.

Iris is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

Iris is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

A copy of the GNU Lesser General Public License can be found in the LICENSE file in the top-level directory of this distribution and at <http://www.gnu.org/licenses/>.

### 8.8.4 DESCRIPTION

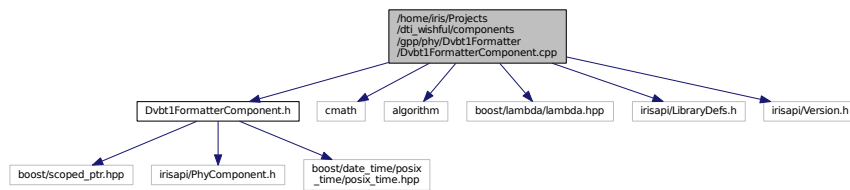
The Dvbt1Filter component.

Definition in file [Dvbt1FilterComponent.h](#).

## 8.9 /home/iris/Projects/dti\_wishful/components/gpp/phy/Dvbt1Formatter/Dvbt1FormatterComponent.cpp File Reference

```
#include "Dvbt1FormatterComponent.h"
#include <cmath>
#include <algorithm>
#include <boost/lambda/lambda.hpp>
#include "irisapi/LibraryDefs.h"
#include "irisapi/Version.h"
```

Include dependency graph for Dvbt1FormatterComponent.cpp:



## Namespaces

- [iris](#)
- [iris::phy](#)

## Functions

- [iris::phy::IRIS\\_COMPONENT\\_EXPORTS](#) (PhyComponent, Dvbt1FormatterComponent)

### 8.9.1 Detailed Description

#### Version

0.1

### 8.9.2 COPYRIGHT

Copyright 2012-2016 The Iris Project Developers. See the COPYRIGHT file at the top-level directory of this distribution and at <http://www.software radiosystems.com/iris/copyright.html>.

### 8.9.3 LICENSE

This file is part of the Iris Project.

Iris is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

Iris is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

A copy of the GNU Lesser General Public License can be found in the LICENSE file in the top-level directory of this distribution and at <http://www.gnu.org/licenses/>.

### 8.9.4 DESCRIPTION

Implementation of the Dvbt1Formatter component.

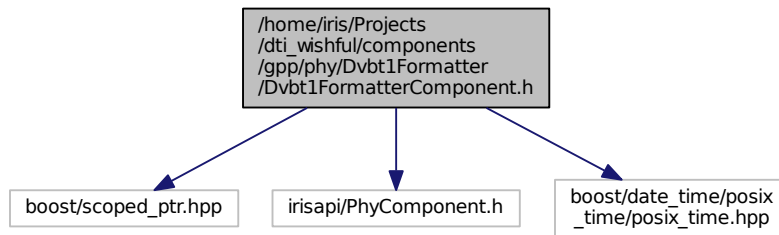
Definition in file [Dvbt1FormatterComponent.cpp](#).



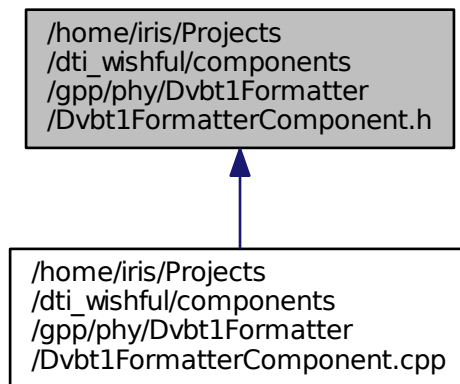
## 8.10 /home/iris/Projects/dti\_wishful/components/gpp/phy/Dvbt1Formatter/Dvbt1FormatterComponent.h File Reference

```
#include <boost/scoped_ptr.hpp>
#include "irisapi/PhyComponent.h"
#include <boost/date_time/posix_time/posix_time.hpp>
```

Include dependency graph for Dvbt1FormatterComponent.h:



This graph shows which files directly or indirectly include this file:



### Classes

- class [iris::phy::Dvbt1FormatterComponent](#)  
*A DVB-T1 formatter component.*

### Namespaces

- [iris](#)
- [iris::phy](#)

### 8.10.1 Detailed Description

#### Version

0.1

### 8.10.2 COPYRIGHT

Copyright 2012-2016 The Iris Project Developers. See the COPYRIGHT file at the top-level directory of this distribution and at <http://www.softwareradiosystems.com/iris/copyright.html>.

### 8.10.3 LICENSE

This file is part of the Iris Project.

Iris is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

Iris is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

A copy of the GNU Lesser General Public License can be found in the LICENSE file in the top-level directory of this distribution and at <http://www.gnu.org/licenses/>.

### 8.10.4 DESCRIPTION

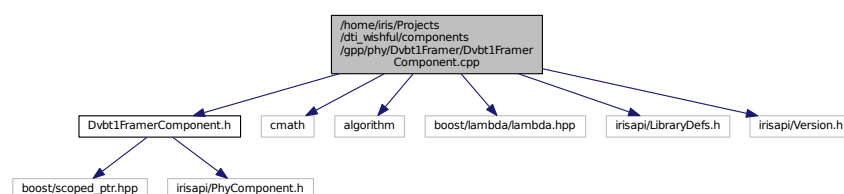
A formatter.

Definition in file [Dvbt1FormatterComponent.h](#).

## 8.11 /home/iris/Projects/dti\_wishful/components/gpp/phy/Dvbt1Framer/Dvbt1Framer-Component.cpp File Reference

```
#include "Dvbt1FramerComponent.h"
#include <cmath>
#include <algorithm>
#include <boost/lambda/lambda.hpp>
#include "irisapi/LibraryDefs.h"
#include "irisapi/Version.h"
```

Include dependency graph for Dvbt1FramerComponent.cpp:



## Namespaces

- [iris](#)
- [iris::phy](#)

## Macros

- `#define T1_PIL_AMPL 1.3333333333333F`
- `#define T1_TPS_AMPL 1.0F`
- `#define T1_N_BCH 127`
- `#define T1_K_BCH 113`

## Functions

- [iris::phy::IRIS\\_COMPONENT\\_EXPORTS](#) (PhyComponent, Dvbt1FramerComponent)

### 8.11.1 Detailed Description

#### Version

0.1

### 8.11.2 COPYRIGHT

Copyright 2012-2016 The Iris Project Developers. See the COPYRIGHT file at the top-level directory of this distribution and at <http://www.software radiosystems.com/iris/copyright.html>.

### 8.11.3 LICENSE

This file is part of the Iris Project.

Iris is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

Iris is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

A copy of the GNU Lesser General Public License can be found in the LICENSE file in the top-level directory of this distribution and at <http://www.gnu.org/licenses/>.

### 8.11.4 DESCRIPTION

Implementation of the Dvbt1Framer component.

Definition in file [Dvbt1FramerComponent.cpp](#).

### 8.11.5 Macro Definition Documentation

#### 8.11.5.1 `#define T1_K_BCH 113`

Definition at line 381 of file Dvbt1FramerComponent.cpp.

Referenced by `iris::phy::Dvbt1FramerComponent::t1_tps_generate()`.

#### 8.11.5.2 `#define T1_N_BCH 127`

Definition at line 380 of file `Dvbt1FramerComponent.cpp`.

Referenced by `iris::phy::Dvbt1FramerComponent::t1_tps_generate()`.

#### 8.11.5.3 `#define T1_PIL_AMPL 1.333333333333F`

Definition at line 377 of file `Dvbt1FramerComponent.cpp`.

Referenced by `iris::phy::Dvbt1FramerComponent::process()`.

#### 8.11.5.4 `#define T1_TPS_AMPL 1.0F`

Definition at line 378 of file `Dvbt1FramerComponent.cpp`.

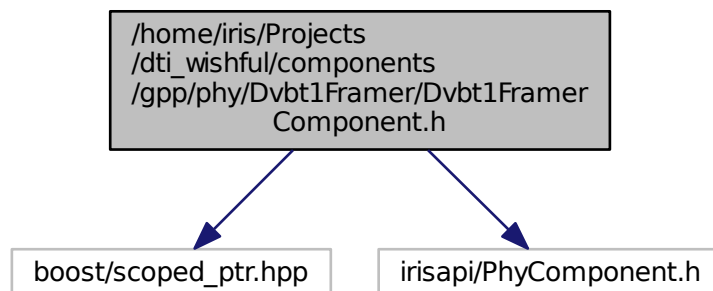
Referenced by `iris::phy::Dvbt1FramerComponent::process()`.

## 8.12 `/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1Framer/Dvbt1FramerComponent.h` File Reference

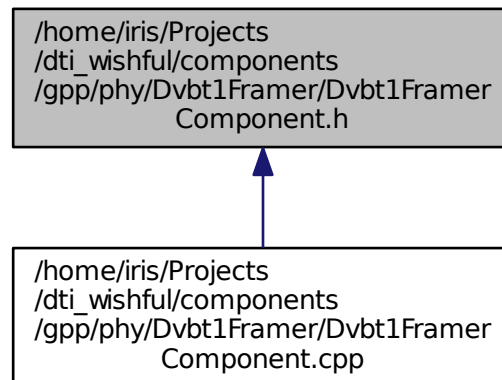
```
#include <boost/scoped_ptr.hpp>
```

```
#include "irisapi/PhyComponent.h"
```

Include dependency graph for `Dvbt1FramerComponent.h`:



This graph shows which files directly or indirectly include this file:



## Classes

- class `iris::phy::Dvbt1FramerComponent`  
*A DVB-T1 framer component.*

## Namespaces

- `iris`
- `iris::phy`

## Macros

- `#define T1_BLOCKS_PER_FRAME 68`
- `#define T1_FRAMES_PER_SUPERFRAME 4`

### 8.12.1 Detailed Description

#### Version

0.1

### 8.12.2 COPYRIGHT

Copyright 2012-2016 The Iris Project Developers. See the COPYRIGHT file at the top-level directory of this distribution and at <http://www.softwareradiosystems.com/iris/copyright.html>.

### 8.12.3 LICENSE

This file is part of the Iris Project.

Iris is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

Iris is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

A copy of the GNU Lesser General Public License can be found in the LICENSE file in the top-level directory of this distribution and at <http://www.gnu.org/licenses/>.

## 8.12.4 DESCRIPTION

The Dvbt1Framer component.

Definition in file [Dvbt1FramerComponent.h](#).

## 8.12.5 Macro Definition Documentation

### 8.12.5.1 #define T1\_BLOCKS\_PER\_FRAME 68

Definition at line 40 of file Dvbt1FramerComponent.h.

Referenced by `iris::phy::Dvbt1FramerComponent::process()`.

### 8.12.5.2 #define T1\_FRAMES\_PER\_SUPERFRAME 4

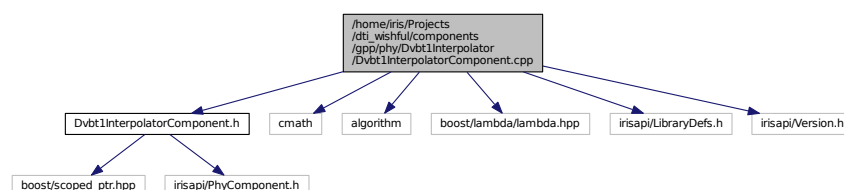
Definition at line 41 of file Dvbt1FramerComponent.h.

Referenced by `iris::phy::Dvbt1FramerComponent::process()`.

## 8.13 /home/iris/Projects/dti\_wishful/components/gpp/phy/Dvbt1Interpolator/Dvbt1Interpolator-Component.cpp File Reference

```
#include "Dvbt1InterpolatorComponent.h"
#include <cmath>
#include <algorithm>
#include <boost/lambda/lambda.hpp>
#include "irisapi/LibraryDefs.h"
#include "irisapi/Version.h"
```

Include dependency graph for Dvbt1InterpolatorComponent.cpp:



## Namespaces

- [iris](#)

- [iris::phy](#)

## Functions

- [iris::phy::IRIS\\_COMPONENT\\_EXPORTS](#) (PhyComponent, Dvbt1InterpolatorComponent)

### 8.13.1 Detailed Description

#### Version

0.1

### 8.13.2 COPYRIGHT

Copyright 2012-2016 The Iris Project Developers. See the COPYRIGHT file at the top-level directory of this distribution and at <http://www.software radiosystems.com/iris/copyright.html>.

### 8.13.3 LICENSE

This file is part of the Iris Project.

Iris is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

Iris is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

A copy of the GNU Lesser General Public License can be found in the LICENSE file in the top-level directory of this distribution and at <http://www.gnu.org/licenses/>.

### 8.13.4 DESCRIPTION

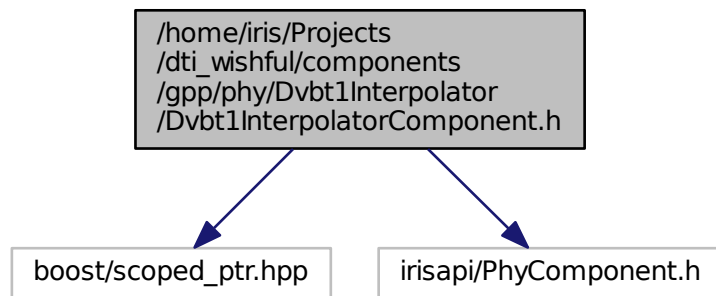
Implementation of the Dvbt1Interpolator component.

Definition in file [Dvbt1InterpolatorComponent.cpp](#).

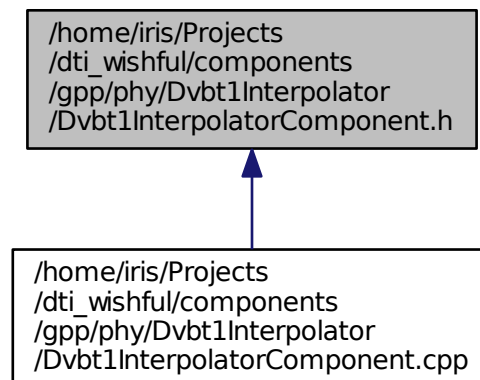
## 8.14 /home/iris/Projects/dti\_wishful/components/gpp/phy/Dvbt1Interpolator/Dvbt1InterpolatorComponent.h File Reference

```
#include <boost/scoped_ptr.hpp>
#include "irisapi/PhyComponent.h"
```

Include dependency graph for Dvbt1InterpolatorComponent.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [iris::phy::Dvbt1InterpolatorComponent](#)  
A DVB-T1 interpolator component.

## Namespaces

- [iris](#)
- [iris::phy](#)

## Macros

- `#define` [T1\\_RESAMPLE\\_ORDER](#) 4



*this defines the memory of the interpolator - keep low to have a good speed*

### 8.14.1 Detailed Description

Version

0.1

### 8.14.2 COPYRIGHT

Copyright 2012-2016 The Iris Project Developers. See the COPYRIGHT file at the top-level directory of this distribution and at <http://www.softwareradiosystems.com/iris/copyright.html>.

### 8.14.3 LICENSE

This file is part of the Iris Project.

Iris is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

Iris is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

A copy of the GNU Lesser General Public License can be found in the LICENSE file in the top-level directory of this distribution and at <http://www.gnu.org/licenses/>.

### 8.14.4 DESCRIPTION

The Dvbt1Interpolator component.

Definition in file [Dvbt1InterpolatorComponent.h](#).

### 8.14.5 Macro Definition Documentation

#### 8.14.5.1 `#define T1_RESAMPLE_ORDER 4`

this defines the memory of the interpolator - keep low to have a good speed

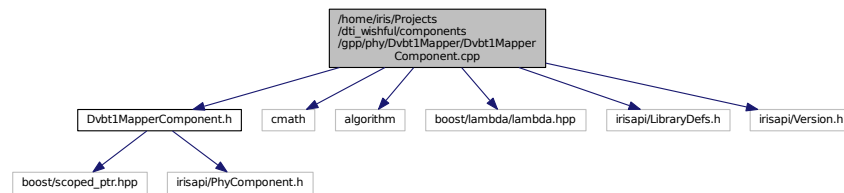
Definition at line 41 of file Dvbt1InterpolatorComponent.h.

Referenced by `iris::phy::Dvbt1InterpolatorComponent::process()`, `iris::phy::Dvbt1InterpolatorComponent::setup()`, and `iris::phy::Dvbt1OfdmModComponent::setup()`.

## 8.15 /home/iris/Projects/dti\_wishful/components/gpp/phy/Dvbt1Mapper/Dvbt1MapperComponent.cpp File Reference

```
#include "Dvbt1MapperComponent.h"
#include <cmath>
#include <algorithm>
#include <boost/lambda/lambda.hpp>
#include "irisapi/LibraryDefs.h"
#include "irisapi/Version.h"
```

Include dependency graph for Dvbt1MapperComponent.cpp:



## Namespaces

- [iris](#)
- [iris::phy](#)

## Functions

- [iris::phy::IRIS\\_COMPONENT\\_EXPORTS](#) (PhyComponent, Dvbt1MapperComponent)

### 8.15.1 Detailed Description

#### Version

0.1

### 8.15.2 COPYRIGHT

Copyright 2012-2016 The Iris Project Developers. See the COPYRIGHT file at the top-level directory of this distribution and at <http://www.software radiosystems.com/iris/copyright.html>.

### 8.15.3 LICENSE

This file is part of the Iris Project.

Iris is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

Iris is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

A copy of the GNU Lesser General Public License can be found in the LICENSE file in the top-level directory of this distribution and at <http://www.gnu.org/licenses/>.

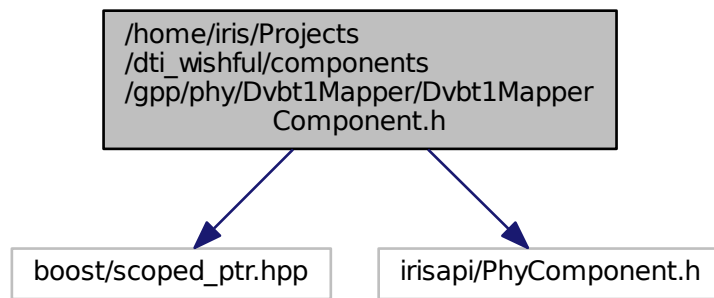
### 8.15.4 DESCRIPTION

Implementation of the Dvbt1Mapper component.

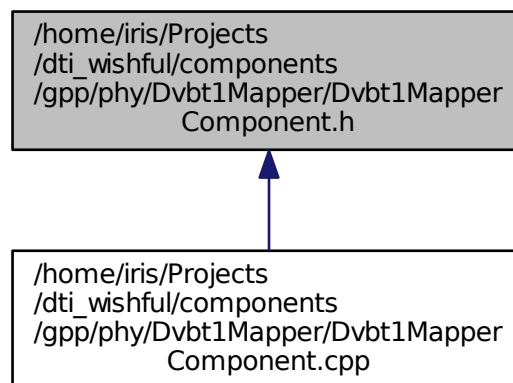
Definition in file [Dvbt1MapperComponent.cpp](#).

## 8.16 /home/iris/Projects/dti\_wishful/components/gpp/phy/Dvbt1Mapper/Dvbt1MapperComponent.h File Reference

```
#include <boost/scoped_ptr.hpp>
#include "irisapi/PhyComponent.h"
Include dependency graph for Dvbt1MapperComponent.h:
```



This graph shows which files directly or indirectly include this file:



### Classes

- class [iris::phy::Dvbt1MapperComponent](#)  
*A DVB-T1 mapper component.*

### Namespaces

- [iris](#)

- [iris::phy](#)

### 8.16.1 Detailed Description

Version

0.1

### 8.16.2 COPYRIGHT

Copyright 2012-2016 The Iris Project Developers. See the COPYRIGHT file at the top-level directory of this distribution and at <http://www.softwareradiosystems.com/iris/copyright.html>.

### 8.16.3 LICENSE

This file is part of the Iris Project.

Iris is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

Iris is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

A copy of the GNU Lesser General Public License can be found in the LICENSE file in the top-level directory of this distribution and at <http://www.gnu.org/licenses/>.

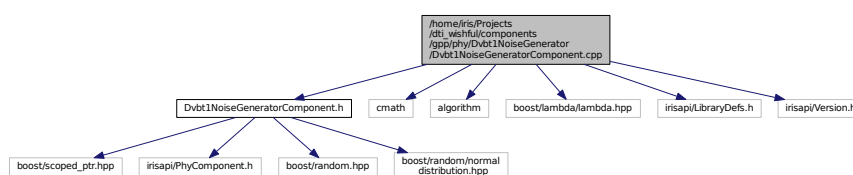
### 8.16.4 DESCRIPTION

The Dvbt1Mapper component.

Definition in file [Dvbt1MapperComponent.h](#).

## 8.17 /home/iris/Projects/dti\_wishful/components/gpp/phy/Dvbt1NoiseGenerator/Dvbt1NoiseGeneratorComponent.cpp File Reference

```
#include "Dvbt1NoiseGeneratorComponent.h"
#include <cmath>
#include <algorithm>
#include <boost/lambda/lambda.hpp>
#include "irisapi/LibraryDefs.h"
#include "irisapi/Version.h"
Include dependency graph for Dvbt1NoiseGeneratorComponent.cpp:
```



- [iris](#)
- [iris::phy](#)

## Functions

- [iris::phy::IRIS\\_COMPONENT\\_EXPORTS](#) (PhyComponent, Dvbt1NoiseGeneratorComponent)

### 8.17.1 Detailed Description

#### Version

0.1

### 8.17.2 COPYRIGHT

Copyright 2012-2016 The Iris Project Developers. See the COPYRIGHT file at the top-level directory of this distribution and at <http://www.softwareradiosystems.com/iris/copyright.html>.

### 8.17.3 LICENSE

This file is part of the Iris Project.

Iris is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

Iris is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

A copy of the GNU Lesser General Public License can be found in the LICENSE file in the top-level directory of this distribution and at <http://www.gnu.org/licenses/>.

### 8.17.4 DESCRIPTION

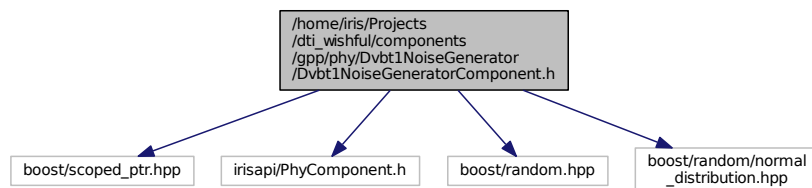
Implementation of the Dvbt1NoiseGenerator component.

Definition in file [Dvbt1NoiseGeneratorComponent.cpp](#).

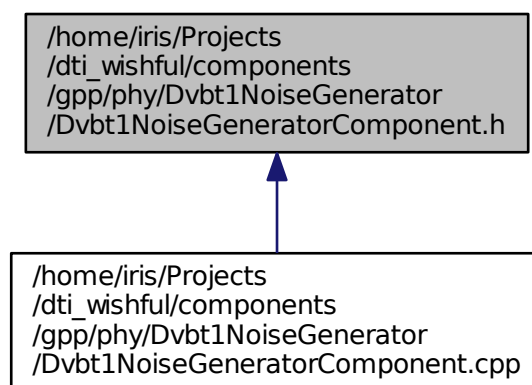
## 8.18 /home/iris/Projects/dti\_wishful/components/gpp/phy/Dvbt1NoiseGenerator/Dvbt1NoiseGeneratorComponent.h File Reference

```
#include <boost/scoped_ptr.hpp>
#include "irisapi/PhyComponent.h"
#include <boost/random.hpp>
#include <boost/random/normal_distribution.hpp>
```

Include dependency graph for Dvbt1NoiseGeneratorComponent.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [iris::phy::Dvbt1NoiseGeneratorComponent](#)

*A DVB-T1 noise generator.*

## Namespaces

- [iris](#)
- [iris::phy](#)

### 8.18.1 Detailed Description

## Version

0.1

## 8.18.2 COPYRIGHT

Copyright 2012-2016 The Iris Project Developers. See the COPYRIGHT file at the top-level directory of this distribution and at <http://www.softwareradiosystems.com/iris/copyright.html>.

## 8.18.3 LICENSE

This file is part of the Iris Project.

Iris is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

Iris is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

A copy of the GNU Lesser General Public License can be found in the LICENSE file in the top-level directory of this distribution and at <http://www.gnu.org/licenses/>.

## 8.18.4 DESCRIPTION

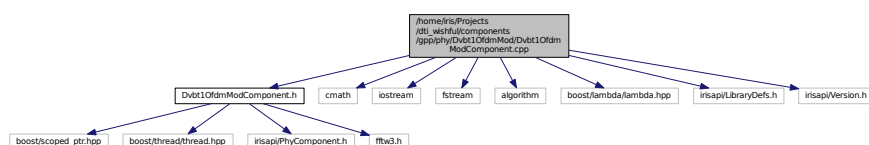
A DVB-T1 noise generator component.

Definition in file [Dvbt1NoiseGeneratorComponent.h](#).

## 8.19 /home/iris/Projects/dti\_wishful/components/gpp/phy/Dvbt1OfdmMod/Dvbt1OfdmModComponent.cpp File Reference

```
#include "Dvbt1OfdmModComponent.h"
#include <cmath>
#include <iostream>
#include <fstream>
#include <algorithm>
#include <boost/lambda/lambda.hpp>
#include "irisapi/LibraryDefs.h"
#include "irisapi/Version.h"
```

Include dependency graph for Dvbt1OfdmModComponent.cpp:



## Namespaces

- [iris](#)
- [iris::phy](#)

## Macros

- `#define WAKEUPINTERVALMS 200`

## Functions

- `iris::phy::IRIS_COMPONENT_EXPORTS` (PhyComponent, Dvbt1OfdmModComponent)

### 8.19.1 Detailed Description

#### Version

0.1

### 8.19.2 COPYRIGHT

Copyright 2012-2016 The Iris Project Developers. See the COPYRIGHT file at the top-level directory of this distribution and at <http://www.software radiosystems.com/iris/copyright.html>.

### 8.19.3 LICENSE

This file is part of the Iris Project.

Iris is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

Iris is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

A copy of the GNU Lesser General Public License can be found in the LICENSE file in the top-level directory of this distribution and at <http://www.gnu.org/licenses/>.

### 8.19.4 DESCRIPTION

Implementation of the Dvbt1OfdmMod component.

Definition in file [Dvbt1OfdmModComponent.cpp](#).

### 8.19.5 Macro Definition Documentation

#### 8.19.5.1 `#define WAKEUPINTERVALMS 200`

Definition at line 402 of file [Dvbt1OfdmModComponent.cpp](#).

Referenced by `iris::phy::Dvbt1OfdmModComponent::powerProcedure_()`.

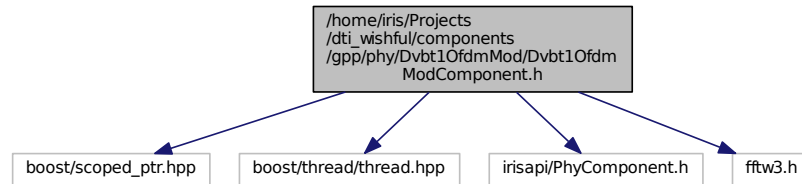
## 8.20 `/home/iris/Projects/dti_wishful/components/gpp/phy/Dvbt1OfdmMod/Dvbt1OfdmModComponent.h` File Reference

```
#include <boost/scoped_ptr.hpp>
```

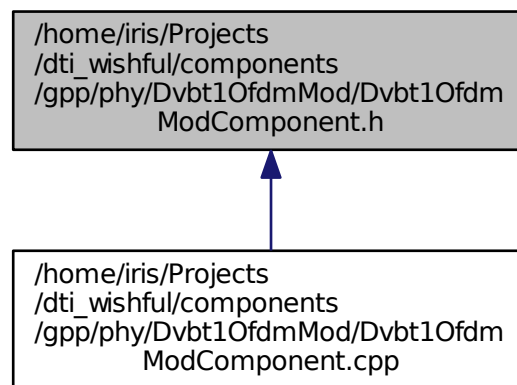


```
#include <boost/thread/thread.hpp>
#include "irisapi/PhyComponent.h"
#include "fftw3.h"
```

Include dependency graph for Dvbt1OfdmModComponent.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class `iris::phy::Dvbt1OfdmModComponent`  
*A DVB-T1 OFDM modulator component.*

## Namespaces

- `iris`
- `iris::phy`

## Macros

- `#define T1_BLOCKS_PER_FRAME 68`
- `#define T1_FRAMES_PER_SUPERFRAME 4`
- `#define T1_RESAMPLE_ORDER 4`

### 8.20.1 Detailed Description

Version

0.1

### 8.20.2 COPYRIGHT

Copyright 2012-2016 The Iris Project Developers. See the COPYRIGHT file at the top-level directory of this distribution and at <http://www.softwareradiosystems.com/iris/copyright.html>.

### 8.20.3 LICENSE

This file is part of the Iris Project.

Iris is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

Iris is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

A copy of the GNU Lesser General Public License can be found in the LICENSE file in the top-level directory of this distribution and at <http://www.gnu.org/licenses/>.

### 8.20.4 DESCRIPTION

The Dvbt1OfdmMod component.

Definition in file [Dvbt1OfdmModComponent.h](#).

### 8.20.5 Macro Definition Documentation

#### 8.20.5.1 `#define T1_BLOCKS_PER_FRAME 68`

Definition at line 42 of file Dvbt1OfdmModComponent.h.

#### 8.20.5.2 `#define T1_FRAMES_PER_SUPERFRAME 4`

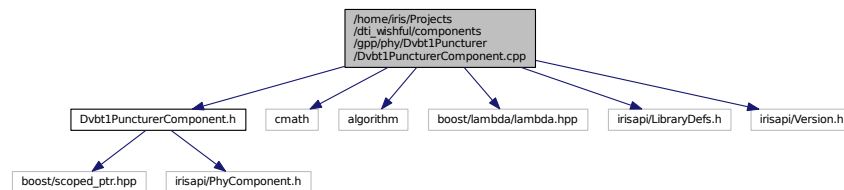
Definition at line 43 of file Dvbt1OfdmModComponent.h.

#### 8.20.5.3 `#define T1_RESAMPLE_ORDER 4`

Definition at line 44 of file Dvbt1OfdmModComponent.h.

## 8.21 /home/iris/Projects/dti\_wishful/components/gpp/phy/Dvbt1Puncturer/Dvbt1Puncturer-Component.cpp File Reference

```
#include "Dvbt1PuncturerComponent.h"
#include <cmath>
#include <algorithm>
#include <boost/lambda/lambda.hpp>
#include "irisapi/LibraryDefs.h"
#include "irisapi/Version.h"
Include dependency graph for Dvbt1PuncturerComponent.cpp:
```



### Namespaces

- [iris](#)
- [iris::phy](#)

### Functions

- [iris::phy::IRIS\\_COMPONENT\\_EXPORTS](#) (PhyComponent, Dvbt1PuncturerComponent)

#### 8.21.1 Detailed Description

##### Version

0.1

#### 8.21.2 COPYRIGHT

Copyright 2012-2016 The Iris Project Developers. See the COPYRIGHT file at the top-level directory of this distribution and at <http://www.softwareradiosystems.com/iris/copyright.html>.

#### 8.21.3 LICENSE

This file is part of the Iris Project.

Iris is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

Iris is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

A copy of the GNU Lesser General Public License can be found in the LICENSE file in the top-level directory of this distribution and at <http://www.gnu.org/licenses/>.

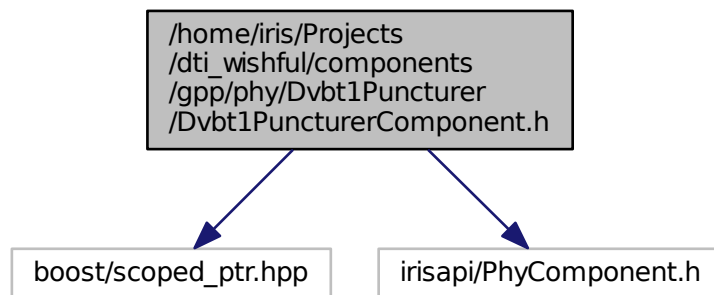
### 8.21.4 DESCRIPTION

Implementation of the Dvbt1Puncturer component.

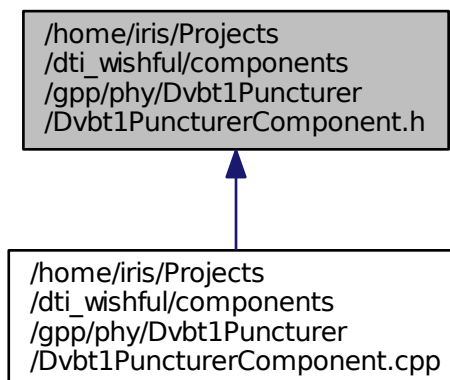
Definition in file [Dvbt1PuncturerComponent.cpp](#).

## 8.22 /home/iris/Projects/dti\_wishful/components/gpp/phy/Dvbt1Puncturer/Dvbt1Puncturer-Component.h File Reference

```
#include <boost/scoped_ptr.hpp>
#include "irisapi/PhyComponent.h"
Include dependency graph for Dvbt1PuncturerComponent.h:
```



This graph shows which files directly or indirectly include this file:



- class [iris::phy::Dvbt1PuncturerComponent](#)

*A DVB-T1 puncturer component.*

## Namespaces

- [iris](#)
- [iris::phy](#)

### 8.22.1 Detailed Description

#### Version

0.1

### 8.22.2 COPYRIGHT

Copyright 2012-2016 The Iris Project Developers. See the COPYRIGHT file at the top-level directory of this distribution and at <http://www.software radiosystems.com/iris/copyright.html>.

### 8.22.3 LICENSE

This file is part of the Iris Project.

Iris is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

Iris is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

A copy of the GNU Lesser General Public License can be found in the LICENSE file in the top-level directory of this distribution and at <http://www.gnu.org/licenses/>.

### 8.22.4 DESCRIPTION

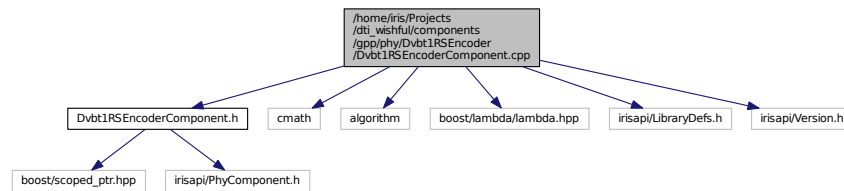
The Dvbt1Puncturer component.

Definition in file [Dvbt1PuncturerComponent.h](#).

## 8.23 /home/iris/Projects/dti\_wishful/components/gpp/phy/Dvbt1RSEncoder/Dvbt1RSEncoderComponent.cpp File Reference

```
#include "Dvbt1RSEncoderComponent.h"
#include <cmath>
#include <algorithm>
#include <boost/lambda/lambda.hpp>
#include "irisapi/LibraryDefs.h"
#include "irisapi/Version.h"
```

Include dependency graph for Dvbt1RSEncoderComponent.cpp:



## Namespaces

- [iris](#)
- [iris::phy](#)

## Functions

- [iris::phy::IRIS\\_COMPONENT\\_EXPORTS](#) (PhyComponent, Dvbt1RSEncoderComponent)

### 8.23.1 Detailed Description

#### Version

0.1

### 8.23.2 COPYRIGHT

Copyright 2012-2016 The Iris Project Developers. See the COPYRIGHT file at the top-level directory of this distribution and at <http://www.softwareradiosystems.com/iris/copyright.html>.

### 8.23.3 LICENSE

This file is part of the Iris Project.

Iris is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

Iris is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

A copy of the GNU Lesser General Public License can be found in the LICENSE file in the top-level directory of this distribution and at <http://www.gnu.org/licenses/>.

### 8.23.4 DESCRIPTION

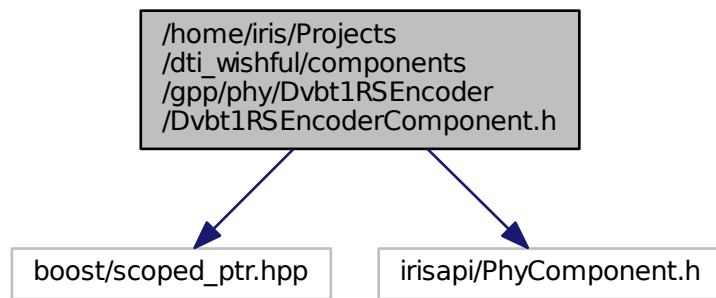
Implementation of the Dvbt1RSEncoder component.

Definition in file [Dvbt1RSEncoderComponent.cpp](#).

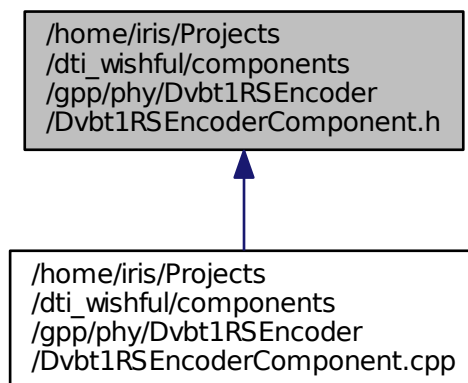
## 8.24 /home/iris/Projects/dti\_wishful/components/gpp/phy/Dvbt1REncoder/Dvbt1REncoderComponent.h File Reference

```
#include <boost/scoped_ptr.hpp>
#include "irisapi/PhyComponent.h"
```

Include dependency graph for Dvbt1REncoderComponent.h:



This graph shows which files directly or indirectly include this file:



### Classes

- class [iris::phy::Dvbt1REncoderComponent](#)  
*A DVB-T1 R-S Encoder component.*

### Namespaces

- [iris](#)

- [iris::phy](#)

## Macros

- `#define TS_PACKET_SIZE 188`  
*TS packet size.*
- `#define RS_PACKET_SIZE 204`
- `#define T1_MM 8`  
*R-S code over  $GF(2^8)$*
- `#define T1_KK 239`  
*Nonshortened message size.*
- `#define T1_NN 255`  
*Nonshortened codeword size.*
- `#define T1_NN_KK 16`  
*Parity bytes size.*
- `#define T1_CLEAR(a, n)`  
*Clear an array from a point towards the beginning.*
- `#define T1_A0 (T1_NN)`  
*Placeholder for zero.*

### 8.24.1 Detailed Description

#### Version

0.1

### 8.24.2 COPYRIGHT

Copyright 2012-2016 The Iris Project Developers. See the COPYRIGHT file at the top-level directory of this distribution and at <http://www.software radiosystems.com/iris/copyright.html>.

### 8.24.3 LICENSE

This file is part of the Iris Project.

Iris is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

Iris is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

A copy of the GNU Lesser General Public License can be found in the LICENSE file in the top-level directory of this distribution and at <http://www.gnu.org/licenses/>.

### 8.24.4 DESCRIPTION

The Dvbt1RSEncoder component.

Definition in file [Dvbt1RSEncoderComponent.h](#).



## 8.24.5 Macro Definition Documentation

### 8.24.5.1 #define RS\_PACKET\_SIZE 204

Definition at line 49 of file Dvbt1REncoderComponent.h.

Referenced by iris::phy::Dvbt1REncoderComponent::process().

### 8.24.5.2 #define T1\_A0 (T1\_NN)

Placeholder for zero.

Definition at line 70 of file Dvbt1REncoderComponent.h.

Referenced by iris::phy::Dvbt1REncoderComponent::packetEncode().

### 8.24.5.3 #define T1\_CLEAR( a, n )

**Value:**

```
{\n    for(int ci=(n)-1;ci >=0;ci--)\n        (a)[ci] = 0;\n}
```

Clear an array from a point towards the beginning.

Definition at line 64 of file Dvbt1REncoderComponent.h.

Referenced by iris::phy::Dvbt1REncoderComponent::packetEncode().

### 8.24.5.4 #define T1\_KK 239

Nonshortened message size.

Definition at line 55 of file Dvbt1REncoderComponent.h.

Referenced by iris::phy::Dvbt1REncoderComponent::packetEncode(), and iris::phy::Dvbt1REncoderComponent::process().

### 8.24.5.5 #define T1\_MM 8

R-S code over GF(2<sup>8</sup>)

Definition at line 52 of file Dvbt1REncoderComponent.h.

Referenced by iris::phy::Dvbt1REncoderComponent::modnn().

### 8.24.5.6 #define T1\_NN 255

Nonshortened codeword size.

Definition at line 58 of file Dvbt1REncoderComponent.h.

Referenced by iris::phy::Dvbt1REncoderComponent::modnn(), iris::phy::Dvbt1REncoderComponent::packetEncode(), iris::phy::Dvbt1REncoderComponent::process(), and iris::phy::Dvbt1REncoderComponent::setup().

### 8.24.5.7 #define T1\_NN\_KK 16

Parity bytes size.

Definition at line 61 of file Dvbt1REncoderComponent.h.

Referenced by `iris::phy::Dvbt1REncoderComponent::packetEncode()`, and `iris::phy::Dvbt1REncoderComponent::process()`.

#### 8.24.5.8 #define TS\_PACKET\_SIZE 188

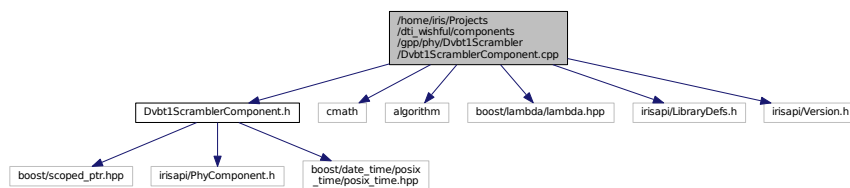
TS packet size.

Definition at line 46 of file Dvbt1REncoderComponent.h.

Referenced by `iris::phy::Dvbt1REncoderComponent::process()`.

## 8.25 /home/iris/Projects/dti\_wishful/components/gpp/phy/Dvbt1Scrambler/Dvbt1Scrambler-Component.cpp File Reference

```
#include "Dvbt1ScramblerComponent.h"
#include <cmath>
#include <algorithm>
#include <boost/lambda/lambda.hpp>
#include "irisapi/LibraryDefs.h"
#include "irisapi/Version.h"
Include dependency graph for Dvbt1ScramblerComponent.cpp:
```



## Namespaces

- [iris](#)
- [iris::phy](#)

## Functions

- [iris::phy::IRIS\\_COMPONENT\\_EXPORTS](#) (PhyComponent, Dvbt1ScramblerComponent)

### 8.25.1 Detailed Description

#### Version

0.1

### 8.25.2 COPYRIGHT

Copyright 2012-2016 The Iris Project Developers. See the COPYRIGHT file at the top-level directory of this distribution and at <http://www.softwareradiosystems.com/iris/copyright.html>.

### 8.25.3 LICENSE

This file is part of the Iris Project.

Iris is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

Iris is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

A copy of the GNU Lesser General Public License can be found in the LICENSE file in the top-level directory of this distribution and at <http://www.gnu.org/licenses/>.

### 8.25.4 DESCRIPTION

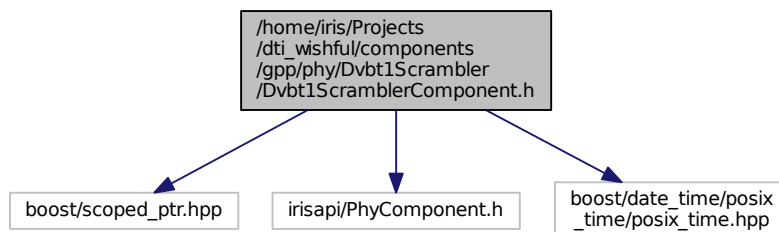
Implementation of the Dvbt1Scrambler component.

Definition in file [Dvbt1ScramblerComponent.cpp](#).

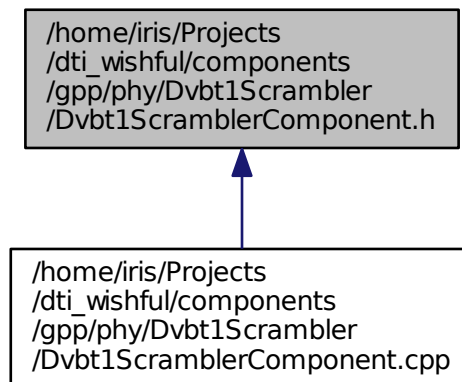
## 8.26 /home/iris/Projects/dti\_wishful/components/gpp/phy/Dvbt1Scrambler/Dvbt1ScramblerComponent.h File Reference

```
#include <boost/scoped_ptr.hpp>
#include "irisapi/PhyComponent.h"
#include <boost/date_time/posix_time/posix_time.hpp>
```

Include dependency graph for Dvbt1ScramblerComponent.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [iris::phy::Dvbt1ScramblerComponent](#)  
*A DVB-T energy dispersal component.*

## Namespaces

- [iris](#)
- [iris::phy](#)

### 8.26.1 Detailed Description

#### Version

0.1

### 8.26.2 COPYRIGHT

Copyright 2012-2016 The Iris Project Developers. See the COPYRIGHT file at the top-level directory of this distribution and at <http://www.softwareradiosystems.com/iris/copyright.html>.

### 8.26.3 LICENSE

This file is part of the Iris Project.

Iris is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

Iris is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

## 8.27 /home/iris/Projects/dti\_wishful/components/gpp/phy/Dvbt1SymbolInterleaver/Dvbt1SymbolInterleaverComponent.cpp File

### Reference

215

A copy of the GNU Lesser General Public License can be found in the LICENSE file in the top-level directory of this distribution and at <http://www.gnu.org/licenses/>.

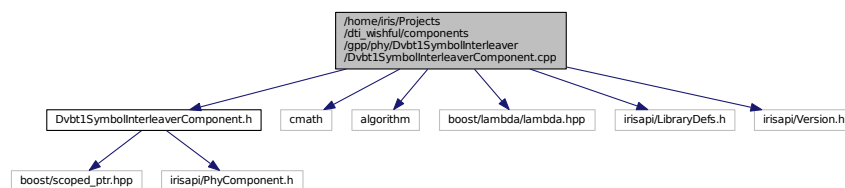
### 8.26.4 DESCRIPTION

The Dvbt1Scrambler component.

Definition in file [Dvbt1ScramblerComponent.h](#).

## 8.27 /home/iris/Projects/dti\_wishful/components/gpp/phy/Dvbt1SymbolInterleaver/-Dvbt1SymbolInterleaverComponent.cpp File Reference

```
#include "Dvbt1SymbolInterleaverComponent.h"
#include <cmath>
#include <algorithm>
#include <boost/lambda/lambda.hpp>
#include "irisapi/LibraryDefs.h"
#include "irisapi/Version.h"
Include dependency graph for Dvbt1SymbolInterleaverComponent.cpp:
```



### Namespaces

- [iris](#)
- [iris::phy](#)

### Functions

- [iris::phy::IRIS\\_COMPONENT\\_EXPORTS](#) (PhyComponent, Dvbt1SymbolInterleaverComponent)

### 8.27.1 Detailed Description

#### Version

0.1

### 8.27.2 COPYRIGHT

Copyright 2012-2016 The Iris Project Developers. See the COPYRIGHT file at the top-level directory of this distribution and at <http://www.softwareradiosystems.com/iris/copyright.html>.

### 8.27.3 LICENSE

This file is part of the Iris Project.

Iris is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

Iris is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

A copy of the GNU Lesser General Public License can be found in the LICENSE file in the top-level directory of this distribution and at <http://www.gnu.org/licenses/>.

### 8.27.4 DESCRIPTION

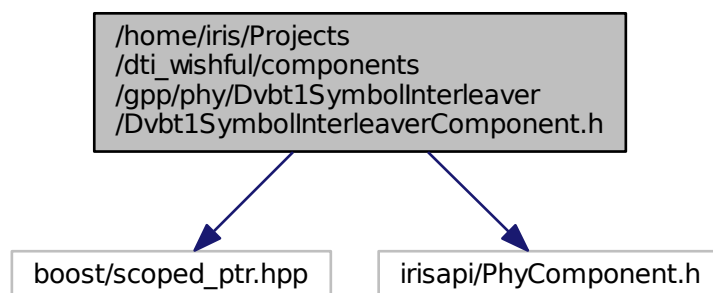
Implementation of the Dvbt1SymbolInterleaver component.

Definition in file [Dvbt1SymbolInterleaverComponent.cpp](#).

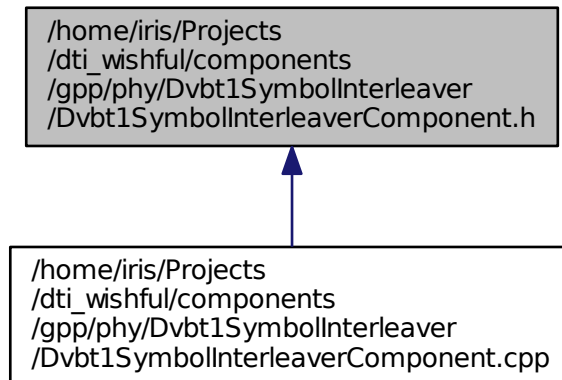
## 8.28 /home/iris/Projects/dti\_wishful/components/gpp/phy/Dvbt1SymbolInterleaver/- Dvbt1SymbolInterleaverComponent.h File Reference

```
#include <boost/scoped_ptr.hpp>
#include "irisapi/PhyComponent.h"
```

Include dependency graph for Dvbt1SymbolInterleaverComponent.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [iris::phy::Dvbt1SymbolInterleaverComponent](#)  
*A DVB-T1 symbol interleaver component.*

## Namespaces

- [iris](#)
- [iris::phy](#)

### 8.28.1 Detailed Description

#### Version

0.1

### 8.28.2 COPYRIGHT

Copyright 2012-2016 The Iris Project Developers. See the COPYRIGHT file at the top-level directory of this distribution and at <http://www.softwareradiosystems.com/iris/copyright.html>.

### 8.28.3 LICENSE

This file is part of the Iris Project.

Iris is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

Iris is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

A copy of the GNU Lesser General Public License can be found in the LICENSE file in the top-level directory of this distribution and at <http://www.gnu.org/licenses/>.

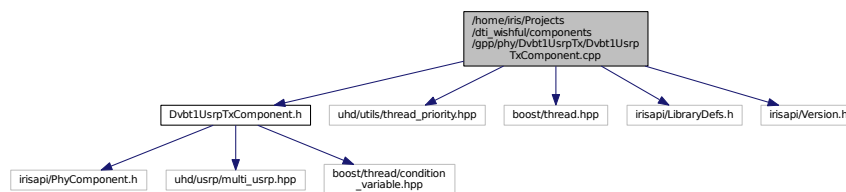
#### 8.28.4 DESCRIPTION

The Dvbt1SymbolInterleaver component.

Definition in file [Dvbt1SymbolInterleaverComponent.h](#).

### 8.29 /home/iris/Projects/dti\_wishful/components/gpp/phy/Dvbt1UsrcTx/Dvbt1UsrcTx-Component.cpp File Reference

```
#include "Dvbt1UsrcTxComponent.h"
#include <uhd/utils/thread_priority.hpp>
#include <boost/thread.hpp>
#include "irisapi/LibraryDefs.h"
#include "irisapi/Version.h"
Include dependency graph for Dvbt1UsrcTxComponent.cpp:
```



#### Namespaces

- [iris](#)
- [iris::phy](#)

#### Macros

- `#define DUMP_STATUS() {}`
- `#define dbgprintf(...) {}`

#### Functions

- [iris::phy::IRIS\\_COMPONENT\\_EXPORTS](#) (PhyComponent, Dvbt1UsrcTxComponent)

#### 8.29.1 Detailed Description

##### Version

1.0



## 8.29.2 COPYRIGHT

Copyright 2012-2013 The Iris Project Developers. See the COPYRIGHT file at the top-level directory of this distribution and at <http://www.softwareradiosystems.com/iris/copyright.html>.

## 8.29.3 LICENSE

This file is part of the Iris Project.

Iris is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

Iris is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

A copy of the GNU Lesser General Public License can be found in the LICENSE file in the top-level directory of this distribution and at <http://www.gnu.org/licenses/>.

Definition in file [Dvbt1UsrcTxComponent.cpp](#).

## 8.29.4 Macro Definition Documentation

### 8.29.4.1 #define dbgprintf( ... ) {}

Definition at line 339 of file Dvbt1UsrcTxComponent.cpp.

Referenced by `iris::phy::Dvbt1UsrcTxComponent::process()`, and `iris::phy::Dvbt1UsrcTxComponent::usrpThreadProcedure()`.

### 8.29.4.2 #define DUMP\_STATUS( ) {}

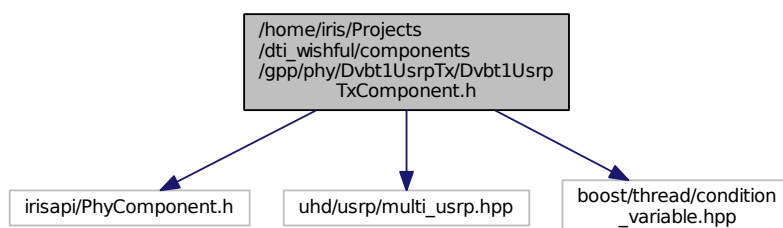
Definition at line 335 of file Dvbt1UsrcTxComponent.cpp.

Referenced by `iris::phy::Dvbt1UsrcTxComponent::process()`, and `iris::phy::Dvbt1UsrcTxComponent::usrpThreadProcedure()`.

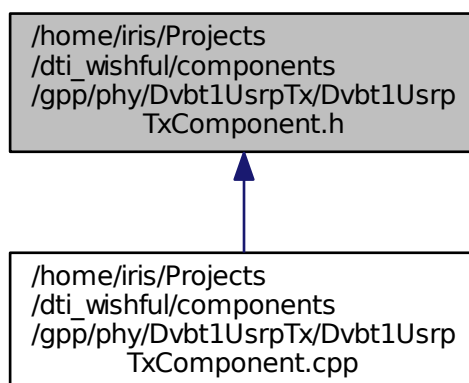
## 8.30 /home/iris/Projects/dti\_wishful/components/gpp/phy/Dvbt1UsrcTx/Dvbt1UsrcTx-Component.h File Reference

```
#include "irisapi/PhyComponent.h"
#include <uhd/usrp/multi_usrp.hpp>
#include <boost/thread/condition_variable.hpp>
```

Include dependency graph for Dvbt1UsrcTxComponent.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class `iris::phy::Dvbt1UsrcTxComponent`

*The Dvbt1UsrcTx component.*

## Namespaces

- `iris`
- `iris::phy`

## 8.30.1 Detailed Description

### Version

1.0

### 8.30.2 COPYRIGHT

Copyright 2012-2013 The Iris Project Developers. See the COPYRIGHT file at the top-level directory of this distribution and at <http://www.softwareradiosystems.com/iris/copyright.html>.

### 8.30.3 LICENSE

This file is part of the Iris Project.

Iris is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

Iris is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

A copy of the GNU Lesser General Public License can be found in the LICENSE file in the top-level directory of this distribution and at <http://www.gnu.org/licenses/>.

### 8.30.4 DESCRIPTION

The Dvbt1UsrpTx component.

Definition in file [Dvbt1UsrpTxComponent.h](#).

## 8.31 Main\_Page.txt File Reference



## Chapter 9

# Example Documentation

### 9.1 dvbt1chain\_ofdmmod\_filter\_spectrum.xml

This is an example of how to use the DVB-TX-IRIS modules to show the spectrum of the generated DVB-T signal.

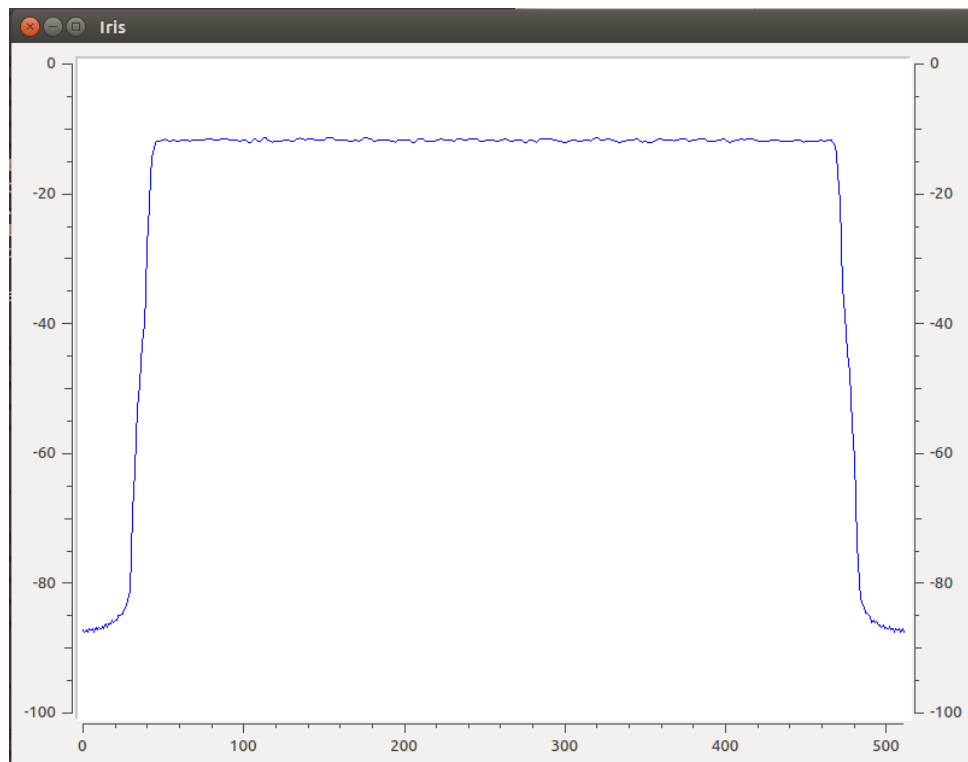


Figure 9.1: Spectrum of the generated DVB-T signal.

The transmission chain has been split among several physical engines (i.e., separate threads), so as to have the maximum parallelized performance.

```
<?xml version="1.0" encoding="utf-8" ?>
<softwareradio name="Radio1">
  <controller class="spectrogramdisplay">
    <parameter name="spectrogramcomponent" value="spectrogram1"/>
  </controller>
  <engine name="phyengine1" class="phyengine">
```

```

<component name="filerawreader1" class="filerawreader">
  <parameter name="filename" value="mux4800000.ts"/>
  <parameter name="blocksize" value="4096"/>
  <parameter name="datatype" value="uint8_t"/>
  <port name="output1" class="output"/>
</component>

<component name="dvbtlscrambler1" class="dvbtlscrambler">
  <parameter name="debug" value="false"/>
  <port name="input1" class="input"/>
  <port name="output1" class="output"/>
</component>

<component name="dvbtlrsencoder1" class="dvbtlrsencoder">
  <parameter name="debug" value="false"/>
  <port name="input1" class="input"/>
  <port name="output1" class="output"/>
</component>

<component name="dvbt1convinterleaver1" class="dvbt1convinterleaver">
  <parameter name="debug" value="false"/>
  <port name="input1" class="input"/>
  <port name="output1" class="output"/>
</component>

<component name="dvbt1convencoder1" class="dvbt1convencoder">
  <parameter name="debug" value="false"/>
  <port name="input1" class="input"/>
  <port name="output1" class="output"/>
</component>

<component name="dvbt1puncturer1" class="dvbt1puncturer">
  <parameter name="debug" value="false"/>
  <parameter name="coderate" value="34"/>
  <port name="input1" class="input"/>
  <port name="output1" class="output"/>
</component>

<component name="dvbt1bitinterleaver1" class="dvbt1bitinterleaver">
  <parameter name="debug" value="false"/>
  <parameter name="qammapping" value="64"/>
  <parameter name="hyerarchymode" value="0"/>
  <port name="input1" class="input"/>
  <port name="output1" class="output"/>
</component>

<component name="dvbt1symbolinterleaver1" class="dvbt1symbolinterleaver">
  <parameter name="debug" value="false"/>
  <parameter name="ofdmmod" value="2048"/>
  <port name="input1" class="input"/>
  <port name="output1" class="output"/>
</component>

<component name="dvbt1mapper1" class="dvbt1mapper">
  <parameter name="debug" value="false"/>
  <parameter name="qammapping" value="64"/>
  <parameter name="hyerarchymode" value="0"/>
  <port name="input1" class="input"/>
  <port name="output1" class="output"/>
</component>

<component name="dvbt1framer1" class="dvbt1framer">
  <parameter name="debug" value="false"/>
  <parameter name="ofdmmod" value="2048"/>
  <parameter name="qammapping" value="64"/>
  <parameter name="hyerarchymode" value="0"/>
  <parameter name="cellid" value="-1"/>
  <parameter name="hpcoderate" value="34"/>
  <parameter name="indepthinterleaver" value="false"/>
  <parameter name="deltamode" value="32"/>
  <port name="input1" class="input"/>
  <port name="output1" class="output"/>
</component>

<component name="dvbt1ofdmmod1" class="dvbt1ofdmmod">
  <parameter name="debug" value="false"/>
  <parameter name="ofdmmod" value="2048"/>
  <parameter name="deltamode" value="32"/>
  <parameter name="outpower" value="50"/>
  <port name="input1" class="input"/>
  <port name="output1" class="output"/>
</component>

<component name="dvbt1filter1" class="dvbt1filter">
  <parameter name="debug" value="false"/>
  <parameter name="samplerate" value="9142857.143"/>

```

```
<parameter name="stopband" value="4000000"/>
<parameter name="attenuation" value="40"/>
<port name="input1" class="input"/>
<port name="output1" class="output"/>
</component>

<component name="spectrogram1" class="spectrogram">
  <parameter name="isprobe" value="true"/>
  <parameter name="issink" value="true"/>
  <parameter name="nwindows" value="512"/>
  <parameter name="nfft" value="512"/>
  <port name="input1" class="input"/>
  <port name="output1" class="output"/>
</component>

</engine>

<link source="filerawreader1.output1" sink="dvbt1scrambler1.input1" />
<link source="dvbt1scrambler1.output1" sink="dvbt1rsencoder1.input1" />
<link source="dvbt1rsencoder1.output1" sink="dvbt1convinterleaver1.input1" />
<link source="dvbt1convinterleaver1.output1" sink="dvbt1convencoder1.input1" />
<link source="dvbt1convencoder1.output1" sink="dvbt1puncturer1.input1" />
<link source="dvbt1puncturer1.output1" sink="dvbt1bitinterleaver1.input1" />
<link source="dvbt1bitinterleaver1.output1" sink="dvbt1symbolinterleaver1.input1" />
<link source="dvbt1symbolinterleaver1.output1" sink="dvbt1mapper1.input1" />
<link source="dvbt1mapper1.output1" sink="dvbt1framer1.input1" />
<link source="dvbt1framer1.output1" sink="dvbt1ofdmmod1.input1" />
<link source="dvbt1ofdmmod1.output1" sink="dvbt1filter1.input1" />
<link source="dvbt1filter1.output1" sink="spectrogram1.input1" />

</softwareradio>
```

# Index

- ~Dvbt1BitInterleaverComponent
  - iris::phy::Dvbt1BitInterleaverComponent, [25](#)
- ~Dvbt1ConvEncoderComponent
  - iris::phy::Dvbt1ConvEncoderComponent, [34](#)
- ~Dvbt1ConvInterleaverComponent
  - iris::phy::Dvbt1ConvInterleaverComponent, [41](#)
- ~Dvbt1FilterComponent
  - iris::phy::Dvbt1FilterComponent, [52](#)
- ~Dvbt1FormatterComponent
  - iris::phy::Dvbt1FormatterComponent, [64](#)
- ~Dvbt1FramerComponent
  - iris::phy::Dvbt1FramerComponent, [72](#)
- ~Dvbt1InterpolatorComponent
  - iris::phy::Dvbt1InterpolatorComponent, [89](#)
- ~Dvbt1MapperComponent
  - iris::phy::Dvbt1MapperComponent, [100](#)
- ~Dvbt1NoiseGeneratorComponent
  - iris::phy::Dvbt1NoiseGeneratorComponent, [109](#)
- ~Dvbt1OfdmModComponent
  - iris::phy::Dvbt1OfdmModComponent, [118](#)
- ~Dvbt1PuncturerComponent
  - iris::phy::Dvbt1PuncturerComponent, [131](#)
- ~Dvbt1REncoderComponent
  - iris::phy::Dvbt1REncoderComponent, [139](#)
- ~Dvbt1ScramblerComponent
  - iris::phy::Dvbt1ScramblerComponent, [148](#)
- ~Dvbt1SymbolInterleaverComponent
  - iris::phy::Dvbt1SymbolInterleaverComponent, [155](#)
- ~Dvbt1UsrpTxComponent
  - iris::phy::Dvbt1UsrpTxComponent, [164](#)
- /home/iris/Projects/dti\_wishful/components/gpp/phy/-  
Dvbt1BitInterleaver/Dvbt1BitInterleaver-  
Component.cpp, [173](#)
- /home/iris/Projects/dti\_wishful/components/gpp/phy/-  
Dvbt1BitInterleaver/Dvbt1BitInterleaver-  
Component.h, [174](#)
- /home/iris/Projects/dti\_wishful/components/gpp/phy/-  
Dvbt1ConvEncoder/Dvbt1ConvEncoder-  
Component.cpp, [176](#)
- /home/iris/Projects/dti\_wishful/components/gpp/phy/-  
Dvbt1ConvEncoder/Dvbt1ConvEncoder-  
Component.h, [177](#)
- /home/iris/Projects/dti\_wishful/components/gpp/phy/-  
Dvbt1ConvInterleaver/Dvbt1ConvInterleaver-  
Component.cpp, [179](#)
- /home/iris/Projects/dti\_wishful/components/gpp/phy/-  
Dvbt1ConvInterleaver/Dvbt1ConvInterleaver-  
Component.h, [180](#)
- /home/iris/Projects/dti\_wishful/components/gpp/phy/-  
Dvbt1Filter/Dvbt1FilterComponent.cpp, [182](#)
- /home/iris/Projects/dti\_wishful/components/gpp/phy/-  
Dvbt1Filter/Dvbt1FilterComponent.h, [184](#)
- /home/iris/Projects/dti\_wishful/components/gpp/phy/-  
Dvbt1Formatter/Dvbt1FormatterComponent.-  
cpp, [185](#)
- /home/iris/Projects/dti\_wishful/components/gpp/phy/-  
Dvbt1Formatter/Dvbt1FormatterComponent.-  
h, [187](#)
- /home/iris/Projects/dti\_wishful/components/gpp/phy/-  
Dvbt1Framer/Dvbt1FramerComponent.cpp,  
[188](#)
- /home/iris/Projects/dti\_wishful/components/gpp/phy/-  
Dvbt1Framer/Dvbt1FramerComponent.h, [190](#)
- /home/iris/Projects/dti\_wishful/components/gpp/phy/-  
Dvbt1Interpolator/Dvbt1InterpolatorComponent.-  
cpp, [192](#)
- /home/iris/Projects/dti\_wishful/components/gpp/phy/-  
Dvbt1Interpolator/Dvbt1InterpolatorComponent.-  
h, [193](#)
- /home/iris/Projects/dti\_wishful/components/gpp/phy/-  
Dvbt1Mapper/Dvbt1MapperComponent.cpp,  
[195](#)
- /home/iris/Projects/dti\_wishful/components/gpp/phy/-  
Dvbt1Mapper/Dvbt1MapperComponent.h,  
[197](#)
- /home/iris/Projects/dti\_wishful/components/gpp/phy/-  
Dvbt1NoiseGenerator/Dvbt1NoiseGenerator-  
Component.cpp, [198](#)
- /home/iris/Projects/dti\_wishful/components/gpp/phy/-  
Dvbt1NoiseGenerator/Dvbt1NoiseGenerator-  
Component.h, [199](#)
- /home/iris/Projects/dti\_wishful/components/gpp/phy/-  
Dvbt1OfdmMod/Dvbt1OfdmModComponent.-  
cpp, [201](#)
- /home/iris/Projects/dti\_wishful/components/gpp/phy/-  
Dvbt1OfdmMod/Dvbt1OfdmModComponent.-  
h, [202](#)
- /home/iris/Projects/dti\_wishful/components/gpp/phy/-  
Dvbt1Puncturer/Dvbt1PuncturerComponent.-  
cpp, [205](#)
- /home/iris/Projects/dti\_wishful/components/gpp/phy/-  
Dvbt1Puncturer/Dvbt1PuncturerComponent.-  
h, [206](#)
- /home/iris/Projects/dti\_wishful/components/gpp/phy/-  
Dvbt1REncoder/Dvbt1REncoderComponent.-  
cpp, [207](#)
- /home/iris/Projects/dti\_wishful/components/gpp/phy/-  
Dvbt1REncoder/Dvbt1REncoderComponent.-



- h, 209
- /home/iris/Projects/dti\_wishful/components/gpp/phy/-  
Dvbt1Scrambler/Dvbt1ScramblerComponent.  
cpp, 212
- /home/iris/Projects/dti\_wishful/components/gpp/phy/-  
Dvbt1Scrambler/Dvbt1ScramblerComponent.  
h, 213
- /home/iris/Projects/dti\_wishful/components/gpp/phy/-  
Dvbt1SymbolInterleaver/Dvbt1Symbol-  
InterleaverComponent.cpp, 215
- /home/iris/Projects/dti\_wishful/components/gpp/phy/-  
Dvbt1SymbolInterleaver/Dvbt1Symbol-  
InterleaverComponent.h, 216
- /home/iris/Projects/dti\_wishful/components/gpp/phy/-  
Dvbt1UsrpTx/Dvbt1UsrpTxComponent.cpp,  
218
- /home/iris/Projects/dti\_wishful/components/gpp/phy/-  
Dvbt1UsrpTx/Dvbt1UsrpTxComponent.h, 219
- \_ampliFactor\_  
iris::phy::Dvbt1OfdmModComponent, 124
- \_precorrFactor\_  
iris::phy::Dvbt1OfdmModComponent, 124
- address\_v2  
iris::phy::Dvbt1BitInterleaverComponent, 28
- address\_v4  
iris::phy::Dvbt1BitInterleaverComponent, 28
- address\_v6  
iris::phy::Dvbt1BitInterleaverComponent, 29
- alpha\_  
iris::phy::Dvbt1RSEncoderComponent, 143
- ampliFactor\_  
iris::phy::Dvbt1OfdmModComponent, 124
- antenna\_x  
iris::phy::Dvbt1UsrpTxComponent, 169
- args\_x  
iris::phy::Dvbt1UsrpTxComponent, 169
- b\_  
iris::phy::Dvbt1ConvInterleaverComponent, 45
- begin  
iris::phy::Dvbt1BitInterleaverComponent, 25  
iris::phy::Dvbt1ConvEncoderComponent, 34  
iris::phy::Dvbt1ConvInterleaverComponent, 41  
iris::phy::Dvbt1FilterComponent, 52  
iris::phy::Dvbt1FormatterComponent, 64  
iris::phy::Dvbt1FramerComponent, 72  
iris::phy::Dvbt1InterpolatorComponent, 89  
iris::phy::Dvbt1MapperComponent, 101  
iris::phy::Dvbt1NoiseGeneratorComponent, 109  
iris::phy::Dvbt1OfdmModComponent, 118  
iris::phy::Dvbt1PuncturerComponent, 131  
iris::phy::Dvbt1RSEncoderComponent, 139  
iris::phy::Dvbt1ScramblerComponent, 148  
iris::phy::Dvbt1SymbolInterleaverComponent, 156
- bessel\_l0  
iris::phy::Dvbt1FilterComponent, 52
- blackman\_sinc  
iris::phy::Dvbt1InterpolatorComponent, 89
- iris::phy::Dvbt1OfdmModComponent, 118
- blockIndex\_  
iris::phy::Dvbt1FramerComponent, 81
- blockSize\_x  
iris::phy::Dvbt1NoiseGeneratorComponent, 111
- bufferSize\_x  
iris::phy::Dvbt1UsrpTxComponent, 169
- bufs\_  
iris::phy::Dvbt1UsrpTxComponent, 169
- bw\_x  
iris::phy::Dvbt1UsrpTxComponent, 169
- ByteVec  
iris::phy::Dvbt1BitInterleaverComponent, 24  
iris::phy::Dvbt1ConvEncoderComponent, 33  
iris::phy::Dvbt1ConvInterleaverComponent, 40  
iris::phy::Dvbt1FilterComponent, 51  
iris::phy::Dvbt1FormatterComponent, 63  
iris::phy::Dvbt1FramerComponent, 71  
iris::phy::Dvbt1InterpolatorComponent, 87  
iris::phy::Dvbt1MapperComponent, 100  
iris::phy::Dvbt1NoiseGeneratorComponent, 107  
iris::phy::Dvbt1OfdmModComponent, 116  
iris::phy::Dvbt1PuncturerComponent, 130  
iris::phy::Dvbt1RSEncoderComponent, 139  
iris::phy::Dvbt1ScramblerComponent, 147  
iris::phy::Dvbt1SymbolInterleaverComponent, 155
- ByteVecIt  
iris::phy::Dvbt1BitInterleaverComponent, 24  
iris::phy::Dvbt1ConvEncoderComponent, 33  
iris::phy::Dvbt1ConvInterleaverComponent, 41  
iris::phy::Dvbt1FilterComponent, 51  
iris::phy::Dvbt1FormatterComponent, 63  
iris::phy::Dvbt1FramerComponent, 71  
iris::phy::Dvbt1InterpolatorComponent, 87  
iris::phy::Dvbt1MapperComponent, 100  
iris::phy::Dvbt1NoiseGeneratorComponent, 107  
iris::phy::Dvbt1OfdmModComponent, 116  
iris::phy::Dvbt1PuncturerComponent, 130  
iris::phy::Dvbt1RSEncoderComponent, 139  
iris::phy::Dvbt1ScramblerComponent, 147  
iris::phy::Dvbt1SymbolInterleaverComponent, 155
- calculateOutputTypes  
iris::phy::Dvbt1BitInterleaverComponent, 25  
iris::phy::Dvbt1ConvEncoderComponent, 34  
iris::phy::Dvbt1ConvInterleaverComponent, 41  
iris::phy::Dvbt1FilterComponent, 54  
iris::phy::Dvbt1FormatterComponent, 64  
iris::phy::Dvbt1FramerComponent, 73  
iris::phy::Dvbt1InterpolatorComponent, 90  
iris::phy::Dvbt1MapperComponent, 101  
iris::phy::Dvbt1NoiseGeneratorComponent, 109  
iris::phy::Dvbt1OfdmModComponent, 119  
iris::phy::Dvbt1PuncturerComponent, 131  
iris::phy::Dvbt1RSEncoderComponent, 140  
iris::phy::Dvbt1ScramblerComponent, 148  
iris::phy::Dvbt1SymbolInterleaverComponent, 156  
iris::phy::Dvbt1UsrpTxComponent, 164
- cellId\_x

- iris::phy::Dvbt1FramerComponent, [81](#)
- codeRate\_x
  - iris::phy::Dvbt1PuncturerComponent, [134](#)
- coeffp\_
  - iris::phy::Dvbt1FilterComponent, [60](#)
- coeffsFile\_x
  - iris::phy::Dvbt1FilterComponent, [60](#)
- condR\_
  - iris::phy::Dvbt1UsrpTxComponent, [170](#)
- condW\_
  - iris::phy::Dvbt1UsrpTxComponent, [170](#)
- constel\_
  - iris::phy::Dvbt1MapperComponent, [104](#)
- cont\_pilot\_position
  - iris::phy::Dvbt1FramerComponent, [81](#)
- Cplx
  - iris::phy::Dvbt1FilterComponent, [51](#)
  - iris::phy::Dvbt1FormatterComponent, [63](#)
  - iris::phy::Dvbt1FramerComponent, [71](#)
  - iris::phy::Dvbt1InterpolatorComponent, [87](#)
  - iris::phy::Dvbt1MapperComponent, [100](#)
  - iris::phy::Dvbt1NoiseGeneratorComponent, [107](#)
  - iris::phy::Dvbt1OfdmModComponent, [116](#)
- CplxVec
  - iris::phy::Dvbt1FilterComponent, [51](#)
  - iris::phy::Dvbt1FormatterComponent, [63](#)
  - iris::phy::Dvbt1FramerComponent, [71](#)
  - iris::phy::Dvbt1InterpolatorComponent, [87](#)
  - iris::phy::Dvbt1MapperComponent, [100](#)
  - iris::phy::Dvbt1NoiseGeneratorComponent, [107](#)
  - iris::phy::Dvbt1OfdmModComponent, [116](#)
- CplxVect
  - iris::phy::Dvbt1FilterComponent, [51](#)
  - iris::phy::Dvbt1FormatterComponent, [63](#)
  - iris::phy::Dvbt1FramerComponent, [71](#)
  - iris::phy::Dvbt1InterpolatorComponent, [88](#)
  - iris::phy::Dvbt1MapperComponent, [100](#)
  - iris::phy::Dvbt1NoiseGeneratorComponent, [107](#)
  - iris::phy::Dvbt1OfdmModComponent, [116](#)
- currentRead\_
  - iris::phy::Dvbt1UsrpTxComponent, [170](#)
- currentWrite\_
  - iris::phy::Dvbt1UsrpTxComponent, [170](#)
- DUMP\_STATUS
  - Dvbt1UsrpTxComponent.cpp, [219](#)
- dacSampleRate\_x
  - iris::phy::Dvbt1OfdmModComponent, [125](#)
- dbgprintf
  - Dvbt1UsrpTxComponent.cpp, [219](#)
- debug\_x
  - iris::phy::Dvbt1BitInterleaverComponent, [29](#)
  - iris::phy::Dvbt1ConvEncoderComponent, [36](#)
  - iris::phy::Dvbt1ConvInterleaverComponent, [45](#)
  - iris::phy::Dvbt1FilterComponent, [60](#)
  - iris::phy::Dvbt1FormatterComponent, [66](#)
  - iris::phy::Dvbt1FramerComponent, [82](#)
  - iris::phy::Dvbt1InterpolatorComponent, [95](#)
  - iris::phy::Dvbt1MapperComponent, [104](#)
  - iris::phy::Dvbt1NoiseGeneratorComponent, [111](#)
  - iris::phy::Dvbt1OfdmModComponent, [125](#)
  - iris::phy::Dvbt1PuncturerComponent, [134](#)
  - iris::phy::Dvbt1RSEncoderComponent, [143](#)
  - iris::phy::Dvbt1ScramblerComponent, [151](#)
  - iris::phy::Dvbt1SymbolInterleaverComponent, [158](#)
- deltaMode\_x
  - iris::phy::Dvbt1FramerComponent, [82](#)
  - iris::phy::Dvbt1OfdmModComponent, [125](#)
- destroy
  - iris::phy::Dvbt1BitInterleaverComponent, [25](#)
  - iris::phy::Dvbt1ConvEncoderComponent, [34](#)
  - iris::phy::Dvbt1ConvInterleaverComponent, [42](#)
  - iris::phy::Dvbt1FilterComponent, [54](#)
  - iris::phy::Dvbt1FormatterComponent, [64](#)
  - iris::phy::Dvbt1FramerComponent, [73](#)
  - iris::phy::Dvbt1InterpolatorComponent, [90](#)
  - iris::phy::Dvbt1MapperComponent, [101](#)
  - iris::phy::Dvbt1NoiseGeneratorComponent, [109](#)
  - iris::phy::Dvbt1OfdmModComponent, [119](#)
  - iris::phy::Dvbt1PuncturerComponent, [132](#)
  - iris::phy::Dvbt1RSEncoderComponent, [140](#)
  - iris::phy::Dvbt1ScramblerComponent, [149](#)
  - iris::phy::Dvbt1SymbolInterleaverComponent, [156](#)
- doneBytes\_
  - iris::phy::Dvbt1ScramblerComponent, [151](#)
- Dvbt1BitInterleaverComponent
  - iris::phy::Dvbt1BitInterleaverComponent, [24](#)
- Dvbt1ConvEncoderComponent
  - iris::phy::Dvbt1ConvEncoderComponent, [33](#)
- Dvbt1ConvEncoderComponent.cpp
  - g1, [177](#)
  - g2, [177](#)
- Dvbt1ConvInterleaverComponent
  - iris::phy::Dvbt1ConvInterleaverComponent, [41](#)
- Dvbt1FilterComponent
  - iris::phy::Dvbt1FilterComponent, [52](#)
- Dvbt1FormatterComponent
  - iris::phy::Dvbt1FormatterComponent, [64](#)
- Dvbt1FramerComponent
  - iris::phy::Dvbt1FramerComponent, [71](#)
- Dvbt1FramerComponent.cpp
  - T1\_K\_BCH, [189](#)
  - T1\_N\_BCH, [189](#)
  - T1\_PIL\_AMPL, [190](#)
  - T1\_TPS\_AMPL, [190](#)
- Dvbt1InterpolatorComponent
  - iris::phy::Dvbt1InterpolatorComponent, [88](#)
- Dvbt1InterpolatorComponent.h
  - T1\_RESAMPLE\_ORDER, [195](#)
- Dvbt1MapperComponent
  - iris::phy::Dvbt1MapperComponent, [100](#)
- Dvbt1NoiseGeneratorComponent
  - iris::phy::Dvbt1NoiseGeneratorComponent, [108](#)
- Dvbt1OfdmModComponent
  - iris::phy::Dvbt1OfdmModComponent, [117](#)
- Dvbt1OfdmModComponent.cpp
  - WAKEUPINTERVALMS, [202](#)

- Dvbt1PuncturerComponent
  - iris::phy::Dvbt1PuncturerComponent, [131](#)
- Dvbt1REncoderComponent
  - iris::phy::Dvbt1REncoderComponent, [139](#)
- Dvbt1REncoderComponent.h
  - RS\_PACKET\_SIZE, [211](#)
  - T1\_A0, [211](#)
  - T1\_CLEAR, [211](#)
  - T1\_KK, [211](#)
  - T1\_MM, [211](#)
  - T1\_NN, [211](#)
  - T1\_NN\_KK, [211](#)
  - TS\_PACKET\_SIZE, [212](#)
- Dvbt1ScramblerComponent
  - iris::phy::Dvbt1ScramblerComponent, [148](#)
- Dvbt1SymbolInterleaverComponent
  - iris::phy::Dvbt1SymbolInterleaverComponent, [155](#)
- Dvbt1UsrpTxComponent
  - iris::phy::Dvbt1UsrpTxComponent, [163](#)
- Dvbt1UsrpTxComponent.cpp
  - DUMP\_STATUS, [219](#)
  - dbgprintf, [219](#)
- end
  - iris::phy::Dvbt1BitInterleaverComponent, [26](#)
  - iris::phy::Dvbt1ConvEncoderComponent, [34](#)
  - iris::phy::Dvbt1ConvInterleaverComponent, [42](#)
  - iris::phy::Dvbt1FilterComponent, [54](#)
  - iris::phy::Dvbt1FormatterComponent, [65](#)
  - iris::phy::Dvbt1FramerComponent, [73](#)
  - iris::phy::Dvbt1InterpolatorComponent, [90](#)
  - iris::phy::Dvbt1MapperComponent, [101](#)
  - iris::phy::Dvbt1NoiseGeneratorComponent, [109](#)
  - iris::phy::Dvbt1OfdmModComponent, [119](#)
  - iris::phy::Dvbt1PuncturerComponent, [132](#)
  - iris::phy::Dvbt1REncoderComponent, [140](#)
  - iris::phy::Dvbt1ScramblerComponent, [149](#)
  - iris::phy::Dvbt1SymbolInterleaverComponent, [156](#)
- eo\_
  - iris::phy::Dvbt1SymbolInterleaverComponent, [158](#)
- factorial
  - iris::phy::Dvbt1FilterComponent, [54](#)
- fft\_
  - iris::phy::Dvbt1OfdmModComponent, [125](#)
- fftBins\_
  - iris::phy::Dvbt1OfdmModComponent, [125](#)
- fftReg\_
  - iris::phy::Dvbt1OfdmModComponent, [125](#)
- filter\_design
  - iris::phy::Dvbt1FilterComponent, [55](#)
- filterLength\_
  - iris::phy::Dvbt1FilterComponent, [60](#)
- find\_rational\_approximation
  - iris::phy::Dvbt1InterpolatorComponent, [90](#)
- fixLoOffset\_x
  - iris::phy::Dvbt1UsrpTxComponent, [170](#)
- FloatVec
  - iris::phy::Dvbt1FilterComponent, [51](#)
- iris::phy::Dvbt1InterpolatorComponent, [88](#)
- iris::phy::Dvbt1NoiseGeneratorComponent, [107](#)
- iris::phy::Dvbt1OfdmModComponent, [116](#)
- FloatVecIt
  - iris::phy::Dvbt1FilterComponent, [51](#)
  - iris::phy::Dvbt1InterpolatorComponent, [88](#)
  - iris::phy::Dvbt1NoiseGeneratorComponent, [108](#)
  - iris::phy::Dvbt1OfdmModComponent, [116](#)
- fmt\_x
  - iris::phy::Dvbt1UsrpTxComponent, [170](#)
- fraOffset\_
  - iris::phy::Dvbt1FramerComponent, [82](#)
- fraRegister\_
  - iris::phy::Dvbt1FramerComponent, [82](#)
- frequency\_response\_modulus
  - iris::phy::Dvbt1OfdmModComponent, [119](#)
- frequency\_x
  - iris::phy::Dvbt1NoiseGeneratorComponent, [111](#)
  - iris::phy::Dvbt1UsrpTxComponent, [170](#)
- fulls\_
  - iris::phy::Dvbt1UsrpTxComponent, [170](#)
- g1
  - Dvbt1ConvEncoderComponent.cpp, [177](#)
- g2
  - Dvbt1ConvEncoderComponent.cpp, [177](#)
- gain\_x
  - iris::phy::Dvbt1UsrpTxComponent, [170](#)
- GaussianGenerator
  - iris::phy::Dvbt1NoiseGeneratorComponent, [108](#)
- gen
  - iris::phy::Dvbt1NoiseGeneratorComponent, [111](#)
- gg\_
  - iris::phy::Dvbt1REncoderComponent, [143](#)
- H\_2K\_
  - iris::phy::Dvbt1SymbolInterleaverComponent, [159](#)
- H\_4K\_
  - iris::phy::Dvbt1SymbolInterleaverComponent, [159](#)
- H\_8K\_
  - iris::phy::Dvbt1SymbolInterleaverComponent, [159](#)
- hpCodeRate\_x
  - iris::phy::Dvbt1FramerComponent, [82](#)
- hierarchyMode\_x
  - iris::phy::Dvbt1BitInterleaverComponent, [29](#)
  - iris::phy::Dvbt1FramerComponent, [82](#)
  - iris::phy::Dvbt1MapperComponent, [105](#)
- l0\_
  - iris::phy::Dvbt1ConvInterleaverComponent, [45](#)
- l10\_
  - iris::phy::Dvbt1ConvInterleaverComponent, [46](#)
- l11\_
  - iris::phy::Dvbt1ConvInterleaverComponent, [46](#)
- l1\_
  - iris::phy::Dvbt1ConvInterleaverComponent, [46](#)
- l2\_
  - iris::phy::Dvbt1ConvInterleaverComponent, [46](#)
- l3\_

- iris::phy::Dvbt1ConvInterleaverComponent, 46
- 14\_
  - iris::phy::Dvbt1ConvInterleaverComponent, 46
- 15\_
  - iris::phy::Dvbt1ConvInterleaverComponent, 46
- 16\_
  - iris::phy::Dvbt1ConvInterleaverComponent, 47
- 17\_
  - iris::phy::Dvbt1ConvInterleaverComponent, 47
- 18\_
  - iris::phy::Dvbt1ConvInterleaverComponent, 47
- 19\_
  - iris::phy::Dvbt1ConvInterleaverComponent, 47
- inBuf\_
  - iris::phy::Dvbt1UsrpTxComponent, 171
- inDepthInterleaver\_x
  - iris::phy::Dvbt1FramerComponent, 82
- inLength\_
  - iris::phy::Dvbt1InterpolatorComponent, 95
- inOffset\_
  - iris::phy::Dvbt1InterpolatorComponent, 95
  - iris::phy::Dvbt1OfdmModComponent, 125
- inReg\_
  - iris::phy::Dvbt1InterpolatorComponent, 95
  - iris::phy::Dvbt1OfdmModComponent, 125
- inSampleRate\_x
  - iris::phy::Dvbt1InterpolatorComponent, 95
- index\_
  - iris::phy::Dvbt1RSEncoderComponent, 144
- initialize
  - iris::phy::Dvbt1BitInterleaverComponent, 26
  - iris::phy::Dvbt1ConvEncoderComponent, 35
  - iris::phy::Dvbt1ConvInterleaverComponent, 42
  - iris::phy::Dvbt1FilterComponent, 55
  - iris::phy::Dvbt1FormatterComponent, 65
  - iris::phy::Dvbt1FramerComponent, 73
  - iris::phy::Dvbt1InterpolatorComponent, 91
  - iris::phy::Dvbt1MapperComponent, 101
  - iris::phy::Dvbt1NoiseGeneratorComponent, 109
  - iris::phy::Dvbt1OfdmModComponent, 120
  - iris::phy::Dvbt1PuncturerComponent, 132
  - iris::phy::Dvbt1RSEncoderComponent, 140
  - iris::phy::Dvbt1ScramblerComponent, 149
  - iris::phy::Dvbt1SymbolInterleaverComponent, 156
  - iris::phy::Dvbt1UsrpTxComponent, 164
- intLength\_
  - iris::phy::Dvbt1BitInterleaverComponent, 30
- intOffset\_
  - iris::phy::Dvbt1BitInterleaverComponent, 30
- intRegister\_
  - iris::phy::Dvbt1BitInterleaverComponent, 30
- IntVec
  - iris::phy::Dvbt1FilterComponent, 51
  - iris::phy::Dvbt1InterpolatorComponent, 88
  - iris::phy::Dvbt1NoiseGeneratorComponent, 108
  - iris::phy::Dvbt1OfdmModComponent, 117
- IntVecIt
  - iris::phy::Dvbt1FilterComponent, 51
- iris::phy::Dvbt1InterpolatorComponent, 88
- iris::phy::Dvbt1NoiseGeneratorComponent, 108
- iris::phy::Dvbt1OfdmModComponent, 117
- interp\_response
  - iris::phy::Dvbt1InterpolatorComponent, 91
- iris, 19
- iris::phy, 19
- iris::phy::Dvbt1BitInterleaverComponent, 21
  - ~Dvbt1BitInterleaverComponent, 25
  - address\_v2, 28
  - address\_v4, 28
  - address\_v6, 29
  - begin, 25
  - ByteVec, 24
  - ByteVecIt, 24
  - calculateOutputTypes, 25
  - debug\_x, 29
  - destroy, 25
  - Dvbt1BitInterleaverComponent, 24
  - end, 26
  - hierarchyMode\_x, 29
  - initialize, 26
  - intLength\_, 30
  - intOffset\_, 30
  - intRegister\_, 30
  - nu\_, 30
  - parameterHasChanged, 26
  - process, 26
  - qamMapping\_x, 30
  - registerPorts, 27
  - sampleRate\_, 30
  - setup, 28
  - timeStamp\_, 30
- iris::phy::Dvbt1ConvEncoderComponent, 31
  - ~Dvbt1ConvEncoderComponent, 34
  - begin, 34
  - ByteVec, 33
  - ByteVecIt, 33
  - calculateOutputTypes, 34
  - debug\_x, 36
  - destroy, 34
  - Dvbt1ConvEncoderComponent, 33
  - end, 34
  - initialize, 35
  - parameterHasChanged, 35
  - parity\_, 36
  - process, 35
  - registerPorts, 36
  - sampleRate\_, 36
  - setup, 36
  - status\_, 37
  - timeStamp\_, 37
- iris::phy::Dvbt1ConvInterleaverComponent, 37
  - ~Dvbt1ConvInterleaverComponent, 41
  - b\_, 45
  - begin, 41
  - ByteVec, 40
  - ByteVecIt, 41

- calculateOutputTypes, 41
- debug\_x, 45
- destroy, 42
- Dvbt1ConvInterleaverComponent, 41
- end, 42
- I0\_, 45
- I10\_, 46
- I11\_, 46
- I1\_, 46
- I2\_, 46
- I3\_, 46
- I4\_, 46
- I5\_, 46
- I6\_, 47
- I7\_, 47
- I8\_, 47
- I9\_, 47
- initialize, 42
- parameterHasChanged, 42
- process, 42
- registerPorts, 44
- rsOffset\_, 47
- sampleRate\_, 47
- setup, 45
- timeStamp\_, 47
- iris::phy::Dvbt1FilterComponent, 47
  - ~Dvbt1FilterComponent, 52
  - begin, 52
  - bessel\_I0, 52
  - ByteVec, 51
  - ByteVecIt, 51
  - calculateOutputTypes, 54
  - coeffp\_, 60
  - coeffsFile\_x, 60
  - Cplx, 51
  - CplxVec, 51
  - CplxVecIt, 51
  - debug\_x, 60
  - destroy, 54
  - Dvbt1FilterComponent, 52
  - end, 54
  - factorial, 54
  - filter\_design, 55
  - filterLength\_, 60
  - FloatVec, 51
  - FloatVecIt, 51
  - initialize, 55
  - IntVec, 51
  - IntVecIt, 51
  - kaiser\_design, 55
  - kaiser\_window, 56
  - parameterHasChanged, 56
  - process, 57
  - registerPorts, 58
  - sBAttenuation\_x, 61
  - sampleRate\_, 60
  - sampleRate\_x, 60
  - setup, 58
  - sinc, 60
  - stopBand\_x, 61
  - symmetric\_, 61
  - timeStamp\_, 61
  - work\_, 61
- iris::phy::Dvbt1FormatterComponent, 61
  - ~Dvbt1FormatterComponent, 64
  - begin, 64
  - ByteVec, 63
  - ByteVecIt, 63
  - calculateOutputTypes, 64
  - Cplx, 63
  - CplxVec, 63
  - CplxVecIt, 63
  - debug\_x, 66
  - destroy, 64
  - Dvbt1FormatterComponent, 64
  - end, 65
  - initialize, 65
  - parameterHasChanged, 65
  - process, 65
  - registerPorts, 66
  - sampleRate\_, 66
  - setup, 66
  - ShortVec, 63
  - ShortVecIt, 64
  - timeStamp\_, 66
- iris::phy::Dvbt1FramerComponent, 67
  - ~Dvbt1FramerComponent, 72
  - begin, 72
  - blockIndex\_, 81
  - ByteVec, 71
  - ByteVecIt, 71
  - calculateOutputTypes, 73
  - cellId\_x, 81
  - cont\_pilot\_position, 81
  - Cplx, 71
  - CplxVec, 71
  - CplxVecIt, 71
  - debug\_x, 82
  - deltaMode\_x, 82
  - destroy, 73
  - Dvbt1FramerComponent, 71
  - end, 73
  - fraOffset\_, 82
  - fraRegister\_, 82
  - hpCodeRate\_x, 82
  - hierarchyMode\_x, 82
  - inDepthInterleaver\_x, 82
  - initialize, 73
  - kMax\_, 82
  - lpCodeRate\_x, 83
  - nMax\_, 83
  - ofdmMode\_x, 83
  - parameterHasChanged, 73
  - prbs\_pilot, 83
  - process, 74
  - qamMapping\_x, 83

- registerPorts, 75
- sampleRate\_, 83
- setup, 76
- t1\_tps\_generate, 76
- timeStamp\_, 83
- tps\_, 83
- tps\_position, 84
- tpsAmpl\_, 84
- iris::phy::Dvbt1InterpolatorComponent, 84
  - ~Dvbt1InterpolatorComponent, 89
  - begin, 89
  - blackman\_sinc, 89
  - ByteVec, 87
  - ByteVecIt, 87
  - calculateOutputTypes, 90
  - Cplx, 87
  - CplxVec, 87
  - CplxVecIt, 88
  - debug\_x, 95
  - destroy, 90
  - Dvbt1InterpolatorComponent, 88
  - end, 90
  - find\_rational\_approximation, 90
  - FloatVec, 88
  - FloatVecIt, 88
  - inLength\_, 95
  - inOffset\_, 95
  - inReg\_, 95
  - inSampleRate\_x, 95
  - initialize, 91
  - IntVec, 88
  - IntVecIt, 88
  - interp\_response, 91
  - outSampleRate\_x, 95
  - parameterHasChanged, 92
  - process, 92
  - registerPorts, 93
  - responseFile\_x, 95
  - sampleRate\_, 96
  - setup, 93
  - sinc, 94
  - tiBasepointIndex\_, 96
  - tiHI\_, 96
  - tiInSize\_, 96
  - tiOutSize\_, 96
  - time\_buffer\_size, 94
  - timeStamp\_, 96
- iris::phy::Dvbt1MapperComponent, 96
  - ~Dvbt1MapperComponent, 100
  - begin, 101
  - ByteVec, 100
  - ByteVecIt, 100
  - calculateOutputTypes, 101
  - constel\_, 104
  - Cplx, 100
  - CplxVec, 100
  - CplxVecIt, 100
  - debug\_x, 104
  - destroy, 101
  - Dvbt1MapperComponent, 100
  - end, 101
  - hyerarchyMode\_x, 105
  - initialize, 101
  - parameterHasChanged, 102
  - process, 102
  - qamMapping\_x, 105
  - registerPorts, 102
  - sampleRate\_, 105
  - setup, 103
  - timeStamp\_, 105
- iris::phy::Dvbt1NoiseGeneratorComponent, 105
  - ~Dvbt1NoiseGeneratorComponent, 109
  - begin, 109
  - blockSize\_x, 111
  - ByteVec, 107
  - ByteVecIt, 107
  - calculateOutputTypes, 109
  - Cplx, 107
  - CplxVec, 107
  - CplxVecIt, 107
  - debug\_x, 111
  - destroy, 109
  - Dvbt1NoiseGeneratorComponent, 108
  - end, 109
  - FloatVec, 107
  - FloatVecIt, 108
  - frequency\_x, 111
  - GaussianGenerator, 108
  - gen, 111
  - initialize, 109
  - IntVec, 108
  - IntVecIt, 108
  - NormalDistribution, 108
  - parameterHasChanged, 110
  - process, 110
  - RandomGenerator, 108
  - registerPorts, 110
  - sampleRate\_, 111
  - setup, 110
  - theta0\_, 111
  - timeStamp\_, 111
  - variance\_x, 112
- iris::phy::Dvbt1OfdmModComponent, 112
  - ~Dvbt1OfdmModComponent, 118
  - \_ampliFactor\_, 124
  - \_precorrFactor\_, 124
  - ampliFactor\_, 124
  - begin, 118
  - blackman\_sinc, 118
  - ByteVec, 116
  - ByteVecIt, 116
  - calculateOutputTypes, 119
  - Cplx, 116
  - CplxVec, 116
  - CplxVecIt, 116
  - dacSampleRate\_x, 125



- debug\_x, 125
- deltaMode\_x, 125
- destroy, 119
- Dvbt1OfdmModComponent, 117
- end, 119
- fft\_, 125
- fftBins\_, 125
- fftReg\_, 125
- FloatVec, 116
- FloatVecIt, 116
- frequency\_response\_modulus, 119
- inOffset\_, 125
- inReg\_, 125
- initialize, 120
- IntVec, 117
- IntVecIt, 117
- kMax\_, 126
- multFactor\_, 126
- nBit\_, 126
- nBlock\_, 126
- nDelta\_, 126
- nFft\_, 126
- nMax\_, 126
- ofdmMode\_x, 126
- outPower\_x, 126
- parameterHasChanged, 120
- powerFile\_x, 127
- powerInterval\_x, 127
- powerProcedure\_, 120
- powerThread\_, 127
- precorrFactor\_, 127
- process, 121
- registerPorts, 122
- runPower\_, 127
- sampleRate\_, 127
- setup, 122
- sinc, 124
- timeStamp\_, 127
- tpsNum\_, 127
- iris::phy::Dvbt1PuncturerComponent, 128
  - ~Dvbt1PuncturerComponent, 131
  - begin, 131
  - ByteVec, 130
  - ByteVecIt, 130
  - calculateOutputTypes, 131
  - codeRate\_x, 134
  - debug\_x, 134
  - destroy, 132
  - Dvbt1PuncturerComponent, 131
  - end, 132
  - initialize, 132
  - parameterHasChanged, 132
  - process, 132
  - punOffset\_, 135
  - punPeriodIn\_, 135
  - punPeriodOut\_, 135
  - punRegister\_, 135
  - registerPorts, 134
  - sampleRate\_, 135
  - setup, 134
  - timeStamp\_, 135
- iris::phy::Dvbt1RSEncoderComponent, 135
  - ~Dvbt1RSEncoderComponent, 139
  - alpha\_, 143
  - begin, 139
  - ByteVec, 139
  - ByteVecIt, 139
  - calculateOutputTypes, 140
  - debug\_x, 143
  - destroy, 140
  - Dvbt1RSEncoderComponent, 139
  - end, 140
  - gg\_, 143
  - index\_, 144
  - initialize, 140
  - modnn, 140
  - packetEncode, 141
  - parameterHasChanged, 141
  - process, 142
  - registerPorts, 142
  - rsCodeWord\_, 144
  - sampleRate\_, 144
  - setup, 143
  - timeStamp\_, 144
  - tsOffset\_, 144
- iris::phy::Dvbt1ScramblerComponent, 145
  - ~Dvbt1ScramblerComponent, 148
  - begin, 148
  - ByteVec, 147
  - ByteVecIt, 147
  - calculateOutputTypes, 148
  - debug\_x, 151
  - destroy, 149
  - doneBytes\_, 151
  - Dvbt1ScramblerComponent, 148
  - end, 149
  - initialize, 149
  - parameterHasChanged, 149
  - process, 149
  - registerPorts, 150
  - reportInterval\_x, 151
  - sampleRate\_, 151
  - scramblerOffset\_, 151
  - scramblerPrbs\_, 151
  - setup, 150
  - start\_, 151
  - timeStamp\_, 152
- iris::phy::Dvbt1SymbolInterleaverComponent, 152
  - ~Dvbt1SymbolInterleaverComponent, 155
  - begin, 156
  - ByteVec, 155
  - ByteVecIt, 155
  - calculateOutputTypes, 156
  - debug\_x, 158
  - destroy, 156
  - Dvbt1SymbolInterleaverComponent, 155

- end, 156
- eo\_, 158
- H\_2K\_, 159
- H\_4K\_, 159
- H\_8K\_, 159
- initialize, 156
- ofdmMode\_x, 159
- parameterHasChanged, 157
- process, 157
- registerPorts, 158
- sampleRate\_, 159
- setup, 158
- siLength\_, 159
- siOffset\_, 159
- siRegister\_, 159
- timeStamp\_, 160
- iris::phy::Dvbt1UsrpTxComponent, 160
  - ~Dvbt1UsrpTxComponent, 164
  - antenna\_x, 169
  - args\_x, 169
  - bufferSize\_x, 169
  - bufs\_, 169
  - bw\_x, 169
  - calculateOutputTypes, 164
  - condR\_, 170
  - condW\_, 170
  - currentRead\_, 170
  - currentWrite\_, 170
  - Dvbt1UsrpTxComponent, 163
  - fixLoOffset\_x, 170
  - fmt\_x, 170
  - frequency\_x, 170
  - fulls\_, 170
  - gain\_x, 170
  - inBuf\_, 171
  - initialize, 164
  - mut\_, 171
  - mutR\_, 171
  - mutW\_, 171
  - numBuffers\_x, 171
  - parameterHasChanged, 166
  - process, 167
  - rate\_x, 171
  - ref\_x, 171
  - registerPorts, 168
  - runUsrp\_, 171
  - streaming\_x, 172
  - subDev\_x, 172
  - txStream\_, 172
  - usrp\_, 172
  - usrpThread\_, 172
  - usrpThreadProcedure, 168
- kMax\_
  - iris::phy::Dvbt1FramerComponent, 82
  - iris::phy::Dvbt1OfdmModComponent, 126
- kaiser\_design
  - iris::phy::Dvbt1FilterComponent, 55
- kaiser\_window
  - iris::phy::Dvbt1FilterComponent, 56
- lpCodeRate\_x
  - iris::phy::Dvbt1FramerComponent, 83
- MAX\_FILTER\_LENGTH
  - Dvbt1FilterComponent.cpp, 183
- Main\_Page.txt, 221
- modnn
  - iris::phy::Dvbt1RSEncoderComponent, 140
- multFactor\_
  - iris::phy::Dvbt1OfdmModComponent, 126
- mut\_
  - iris::phy::Dvbt1UsrpTxComponent, 171
- mutR\_
  - iris::phy::Dvbt1UsrpTxComponent, 171
- mutW\_
  - iris::phy::Dvbt1UsrpTxComponent, 171
- nBit\_
  - iris::phy::Dvbt1OfdmModComponent, 126
- nBlock\_
  - iris::phy::Dvbt1OfdmModComponent, 126
- nDelta\_
  - iris::phy::Dvbt1OfdmModComponent, 126
- nFft\_
  - iris::phy::Dvbt1OfdmModComponent, 126
- nMax\_
  - iris::phy::Dvbt1FramerComponent, 83
  - iris::phy::Dvbt1OfdmModComponent, 126
- NormalDistribution
  - iris::phy::Dvbt1NoiseGeneratorComponent, 108
- nu\_
  - iris::phy::Dvbt1BitInterleaverComponent, 30
- numBuffers\_x
  - iris::phy::Dvbt1UsrpTxComponent, 171
- ofdmMode\_x
  - iris::phy::Dvbt1FramerComponent, 83
  - iris::phy::Dvbt1OfdmModComponent, 126
  - iris::phy::Dvbt1SymbolInterleaverComponent, 159
- outPower\_x
  - iris::phy::Dvbt1OfdmModComponent, 126
- outSampleRate\_x
  - iris::phy::Dvbt1InterpolatorComponent, 95
- packetEncode
  - iris::phy::Dvbt1RSEncoderComponent, 141
- parameterHasChanged
  - iris::phy::Dvbt1BitInterleaverComponent, 26
  - iris::phy::Dvbt1ConvEncoderComponent, 35
  - iris::phy::Dvbt1ConvInterleaverComponent, 42
  - iris::phy::Dvbt1FilterComponent, 56
  - iris::phy::Dvbt1FormatterComponent, 65
  - iris::phy::Dvbt1FramerComponent, 73
  - iris::phy::Dvbt1InterpolatorComponent, 92
  - iris::phy::Dvbt1MapperComponent, 102
  - iris::phy::Dvbt1NoiseGeneratorComponent, 110
  - iris::phy::Dvbt1OfdmModComponent, 120



- iris::phy::Dvbt1PuncturerComponent, 132
- iris::phy::Dvbt1RSEncoderComponent, 141
- iris::phy::Dvbt1ScramblerComponent, 149
- iris::phy::Dvbt1SymbolInterleaverComponent, 157
- iris::phy::Dvbt1UsrpTxComponent, 166
- parity\_
  - iris::phy::Dvbt1ConvEncoderComponent, 36
- powerFile\_x
  - iris::phy::Dvbt1OfdmModComponent, 127
- powerInterval\_x
  - iris::phy::Dvbt1OfdmModComponent, 127
- powerProcedure\_
  - iris::phy::Dvbt1OfdmModComponent, 120
- powerThread\_
  - iris::phy::Dvbt1OfdmModComponent, 127
- prbs\_pilot
  - iris::phy::Dvbt1FramerComponent, 83
- precorrFactor\_
  - iris::phy::Dvbt1OfdmModComponent, 127
- process
  - iris::phy::Dvbt1BitInterleaverComponent, 26
  - iris::phy::Dvbt1ConvEncoderComponent, 35
  - iris::phy::Dvbt1ConvInterleaverComponent, 42
  - iris::phy::Dvbt1FilterComponent, 57
  - iris::phy::Dvbt1FormatterComponent, 65
  - iris::phy::Dvbt1FramerComponent, 74
  - iris::phy::Dvbt1InterpolatorComponent, 92
  - iris::phy::Dvbt1MapperComponent, 102
  - iris::phy::Dvbt1NoiseGeneratorComponent, 110
  - iris::phy::Dvbt1OfdmModComponent, 121
  - iris::phy::Dvbt1PuncturerComponent, 132
  - iris::phy::Dvbt1RSEncoderComponent, 142
  - iris::phy::Dvbt1ScramblerComponent, 149
  - iris::phy::Dvbt1SymbolInterleaverComponent, 157
  - iris::phy::Dvbt1UsrpTxComponent, 167
- punOffset\_
  - iris::phy::Dvbt1PuncturerComponent, 135
- punPeriodIn\_
  - iris::phy::Dvbt1PuncturerComponent, 135
- punPeriodOut\_
  - iris::phy::Dvbt1PuncturerComponent, 135
- punRegister\_
  - iris::phy::Dvbt1PuncturerComponent, 135
- qamMapping\_x
  - iris::phy::Dvbt1BitInterleaverComponent, 30
  - iris::phy::Dvbt1FramerComponent, 83
  - iris::phy::Dvbt1MapperComponent, 105
- RS\_PACKET\_SIZE
  - Dvbt1RSEncoderComponent.h, 211
- RandomGenerator
  - iris::phy::Dvbt1NoiseGeneratorComponent, 108
- rate\_x
  - iris::phy::Dvbt1UsrpTxComponent, 171
- ref\_x
  - iris::phy::Dvbt1UsrpTxComponent, 171
- registerPorts
  - iris::phy::Dvbt1BitInterleaverComponent, 27
- iris::phy::Dvbt1ConvEncoderComponent, 36
- iris::phy::Dvbt1ConvInterleaverComponent, 44
- iris::phy::Dvbt1FilterComponent, 58
- iris::phy::Dvbt1FormatterComponent, 66
- iris::phy::Dvbt1FramerComponent, 75
- iris::phy::Dvbt1InterpolatorComponent, 93
- iris::phy::Dvbt1MapperComponent, 102
- iris::phy::Dvbt1NoiseGeneratorComponent, 110
- iris::phy::Dvbt1OfdmModComponent, 122
- iris::phy::Dvbt1PuncturerComponent, 134
- iris::phy::Dvbt1RSEncoderComponent, 142
- iris::phy::Dvbt1ScramblerComponent, 150
- iris::phy::Dvbt1SymbolInterleaverComponent, 158
- iris::phy::Dvbt1UsrpTxComponent, 168
- reportInterval\_x
  - iris::phy::Dvbt1ScramblerComponent, 151
- responseFile\_x
  - iris::phy::Dvbt1InterpolatorComponent, 95
- rsCodeWord\_
  - iris::phy::Dvbt1RSEncoderComponent, 144
- rsOffset\_
  - iris::phy::Dvbt1ConvInterleaverComponent, 47
- runPower\_
  - iris::phy::Dvbt1OfdmModComponent, 127
- runUsrp\_
  - iris::phy::Dvbt1UsrpTxComponent, 171
- sBAttenuation\_x
  - iris::phy::Dvbt1FilterComponent, 61
- sampleRate\_
  - iris::phy::Dvbt1BitInterleaverComponent, 30
  - iris::phy::Dvbt1ConvEncoderComponent, 36
  - iris::phy::Dvbt1ConvInterleaverComponent, 47
  - iris::phy::Dvbt1FilterComponent, 60
  - iris::phy::Dvbt1FormatterComponent, 66
  - iris::phy::Dvbt1FramerComponent, 83
  - iris::phy::Dvbt1InterpolatorComponent, 96
  - iris::phy::Dvbt1MapperComponent, 105
  - iris::phy::Dvbt1NoiseGeneratorComponent, 111
  - iris::phy::Dvbt1OfdmModComponent, 127
  - iris::phy::Dvbt1PuncturerComponent, 135
  - iris::phy::Dvbt1RSEncoderComponent, 144
  - iris::phy::Dvbt1ScramblerComponent, 151
  - iris::phy::Dvbt1SymbolInterleaverComponent, 159
- sampleRate\_x
  - iris::phy::Dvbt1FilterComponent, 60
- scramblerOffset\_
  - iris::phy::Dvbt1ScramblerComponent, 151
- scramblerPrbs\_
  - iris::phy::Dvbt1ScramblerComponent, 151
- setup
  - iris::phy::Dvbt1BitInterleaverComponent, 28
  - iris::phy::Dvbt1ConvEncoderComponent, 36
  - iris::phy::Dvbt1ConvInterleaverComponent, 45
  - iris::phy::Dvbt1FilterComponent, 58
  - iris::phy::Dvbt1FormatterComponent, 66
  - iris::phy::Dvbt1FramerComponent, 76
  - iris::phy::Dvbt1InterpolatorComponent, 93
  - iris::phy::Dvbt1MapperComponent, 103

- iris::phy::Dvbt1NoiseGeneratorComponent, [110](#)
- iris::phy::Dvbt1OfdmModComponent, [122](#)
- iris::phy::Dvbt1PuncturerComponent, [134](#)
- iris::phy::Dvbt1RSEncoderComponent, [143](#)
- iris::phy::Dvbt1ScramblerComponent, [150](#)
- iris::phy::Dvbt1SymbolInterleaverComponent, [158](#)
- ShortVec
  - iris::phy::Dvbt1FormatterComponent, [63](#)
- ShortVecIt
  - iris::phy::Dvbt1FormatterComponent, [64](#)
- siLength\_
  - iris::phy::Dvbt1SymbolInterleaverComponent, [159](#)
- siOffset\_
  - iris::phy::Dvbt1SymbolInterleaverComponent, [159](#)
- siRegister\_
  - iris::phy::Dvbt1SymbolInterleaverComponent, [159](#)
- sinc
  - iris::phy::Dvbt1FilterComponent, [60](#)
  - iris::phy::Dvbt1InterpolatorComponent, [94](#)
  - iris::phy::Dvbt1OfdmModComponent, [124](#)
- start\_
  - iris::phy::Dvbt1ScramblerComponent, [151](#)
- status\_
  - iris::phy::Dvbt1ConvEncoderComponent, [37](#)
- stopBand\_x
  - iris::phy::Dvbt1FilterComponent, [61](#)
- streaming\_x
  - iris::phy::Dvbt1UsrpTxComponent, [172](#)
- subDev\_x
  - iris::phy::Dvbt1UsrpTxComponent, [172](#)
- symmetric\_
  - iris::phy::Dvbt1FilterComponent, [61](#)
- T1\_A0
  - Dvbt1RSEncoderComponent.h, [211](#)
- T1\_CLEAR
  - Dvbt1RSEncoderComponent.h, [211](#)
- T1\_K\_BCH
  - Dvbt1FramerComponent.cpp, [189](#)
- T1\_KK
  - Dvbt1RSEncoderComponent.h, [211](#)
- T1\_MM
  - Dvbt1RSEncoderComponent.h, [211](#)
- T1\_N\_BCH
  - Dvbt1FramerComponent.cpp, [189](#)
- T1\_NN
  - Dvbt1RSEncoderComponent.h, [211](#)
- T1\_NN\_KK
  - Dvbt1RSEncoderComponent.h, [211](#)
- T1\_PIL\_AMPL
  - Dvbt1FramerComponent.cpp, [190](#)
- T1\_RESAMPLE\_ORDER
  - Dvbt1InterpolatorComponent.h, [195](#)
  - Dvbt1OfdmModComponent.h, [204](#)
- T1\_TPS\_AMPL
  - Dvbt1FramerComponent.cpp, [190](#)
- t1\_tps\_generate
  - iris::phy::Dvbt1FramerComponent, [76](#)
- TS\_PACKET\_SIZE
  - Dvbt1RSEncoderComponent.h, [212](#)
- theta0\_
  - iris::phy::Dvbt1NoiseGeneratorComponent, [111](#)
- tiBasepointIndex\_
  - iris::phy::Dvbt1InterpolatorComponent, [96](#)
- tiHl\_
  - iris::phy::Dvbt1InterpolatorComponent, [96](#)
- tiInSize\_
  - iris::phy::Dvbt1InterpolatorComponent, [96](#)
- tiOutsize\_
  - iris::phy::Dvbt1InterpolatorComponent, [96](#)
- time\_buffer\_size
  - iris::phy::Dvbt1InterpolatorComponent, [94](#)
- timeStamp\_
  - iris::phy::Dvbt1BitInterleaverComponent, [30](#)
  - iris::phy::Dvbt1ConvEncoderComponent, [37](#)
  - iris::phy::Dvbt1ConvInterleaverComponent, [47](#)
  - iris::phy::Dvbt1FilterComponent, [61](#)
  - iris::phy::Dvbt1FormatterComponent, [66](#)
  - iris::phy::Dvbt1FramerComponent, [83](#)
  - iris::phy::Dvbt1InterpolatorComponent, [96](#)
  - iris::phy::Dvbt1MapperComponent, [105](#)
  - iris::phy::Dvbt1NoiseGeneratorComponent, [111](#)
  - iris::phy::Dvbt1OfdmModComponent, [127](#)
  - iris::phy::Dvbt1PuncturerComponent, [135](#)
  - iris::phy::Dvbt1RSEncoderComponent, [144](#)
  - iris::phy::Dvbt1ScramblerComponent, [152](#)
  - iris::phy::Dvbt1SymbolInterleaverComponent, [160](#)
- tps\_
  - iris::phy::Dvbt1FramerComponent, [83](#)
- tps\_position
  - iris::phy::Dvbt1FramerComponent, [84](#)
- tpsAmpl\_
  - iris::phy::Dvbt1FramerComponent, [84](#)
- tpsNum\_
  - iris::phy::Dvbt1OfdmModComponent, [127](#)
- tsOffset\_
  - iris::phy::Dvbt1RSEncoderComponent, [144](#)
- txStream\_
  - iris::phy::Dvbt1UsrpTxComponent, [172](#)
- usrp\_
  - iris::phy::Dvbt1UsrpTxComponent, [172](#)
- usrpThread\_
  - iris::phy::Dvbt1UsrpTxComponent, [172](#)
- usrpThreadProcedure
  - iris::phy::Dvbt1UsrpTxComponent, [168](#)
- variance\_x
  - iris::phy::Dvbt1NoiseGeneratorComponent, [112](#)
- WAKEUPINTERVALMS
  - Dvbt1OfdmModComponent.cpp, [202](#)
- work\_
  - iris::phy::Dvbt1FilterComponent, [61](#)