

objectDump

Generated by Doxygen 1.8.5

Wed Jun 10 2015 14:57:38



# Contents

<b>1</b>	<b>objectDump application</b>	<b>1</b>
<b>2</b>	<b>Hierarchical Index</b>	<b>3</b>
2.1	Class Hierarchy . . . . .	3
<b>3</b>	<b>Class Index</b>	<b>5</b>
3.1	Class List . . . . .	5
<b>4</b>	<b>File Index</b>	<b>7</b>
4.1	File List . . . . .	7
<b>5</b>	<b>Class Documentation</b>	<b>9</b>
5.1	ApplicationSetup Class Reference . . . . .	9
5.1.1	Detailed Description . . . . .	10
5.1.2	Member Function Documentation . . . . .	10
5.1.2.1	ApplicationSetupDataFileModify . . . . .	10
5.1.2.2	ApplicationSetupGetDataFilePunt . . . . .	10
5.1.2.3	ApplicationSetupGetDataFileSizePunt . . . . .	11
5.1.2.4	ApplicationSetupSet . . . . .	11
5.1.2.5	ArgumentsParsing . . . . .	11
5.1.2.6	FetchInputMode . . . . .	11
5.1.2.7	Instance . . . . .	11
5.1.3	Member Data Documentation . . . . .	11
5.1.3.1	application_setup_conf_file . . . . .	11
5.1.3.2	application_setup_conf_file_path . . . . .	12
5.1.3.3	application_setup_conf_object . . . . .	12
5.1.3.4	application_setup_data_file_path . . . . .	12
5.1.3.5	application_setup_data_file_punt . . . . .	12
5.1.3.6	application_setup_data_file_size_path . . . . .	12
5.1.3.7	application_setup_data_file_size_punt . . . . .	12
5.1.3.8	application_setup_input_mode . . . . .	12
5.1.3.9	application_setup_log_file_path . . . . .	12
5.1.3.10	argc . . . . .	12

5.1.3.11	<a href="#">argv</a>	13
5.1.3.12	<a href="#">channel_visualized</a>	13
5.1.3.13	<a href="#">imset</a>	13
5.1.3.14	<a href="#">input_mode</a>	13
5.2	<a href="#">ChannelObject Class Reference</a>	13
5.2.1	<a href="#">Detailed Description</a>	14
5.2.2	<a href="#">Constructor &amp; Destructor Documentation</a>	14
5.2.2.1	<a href="#">ChannelObject</a>	14
5.2.3	<a href="#">Member Function Documentation</a>	14
5.2.3.1	<a href="#">PrintChannel</a>	14
5.2.4	<a href="#">Member Data Documentation</a>	14
5.2.4.1	<a href="#">channel_trigger</a>	14
5.2.4.2	<a href="#">dc_offset</a>	14
5.2.4.3	<a href="#">enable_input</a>	15
5.2.4.4	<a href="#">numChannel</a>	15
5.2.4.5	<a href="#">set</a>	15
5.2.4.6	<a href="#">trigger_threshold</a>	15
5.3	<a href="#">CommunicationObject Class Reference</a>	15
5.3.1	<a href="#">Detailed Description</a>	16
5.3.2	<a href="#">Constructor &amp; Destructor Documentation</a>	16
5.3.2.1	<a href="#">CommunicationObject</a>	16
5.3.3	<a href="#">Member Function Documentation</a>	16
5.3.3.1	<a href="#">CommunicationObjectInit</a>	16
5.3.3.2	<a href="#">Finish</a>	17
5.3.3.3	<a href="#">GetCommand</a>	17
5.3.3.4	<a href="#">GetCommand</a>	17
5.3.3.5	<a href="#">Main</a>	17
5.3.3.6	<a href="#">Worker</a>	17
5.3.4	<a href="#">Member Data Documentation</a>	18
5.3.4.1	<a href="#">Acquisition_Cond1</a>	18
5.3.4.2	<a href="#">Acquisition_Cond2</a>	18
5.3.4.3	<a href="#">Acquisition_Mutex</a>	18
5.3.4.4	<a href="#">BlockedConsumerInput</a>	18
5.3.4.5	<a href="#">BlockedProducerInput</a>	18
5.3.4.6	<a href="#">coda</a>	18
5.3.4.7	<a href="#">command</a>	18
5.3.4.8	<a href="#">go</a>	18
5.3.4.9	<a href="#">main_thread</a>	19
5.3.4.10	<a href="#">num_mex</a>	19
5.3.4.11	<a href="#">ReservedKeyBoardInputArea</a>	19

5.3.4.12	testa	19
5.3.4.13	worker_thread	19
5.4	ConfigurationConsistence Class Reference	19
5.4.1	Detailed Description	20
5.4.2	Member Function Documentation	20
5.4.2.1	ConfigurationConsistenceConfFileInit	20
5.4.2.2	ConfigurationConsistenceConfFileInitNoPrint	20
5.4.2.3	ConfigurationConsistenceConfFileSetupEssentialWithInitCheck	20
5.4.2.4	ConfigurationConsistenceConfFileSetupEssentialWithInitCheckNoPrint	21
5.4.2.5	ConfigurationConsistenceConfFileSetupEssentialWithoutInitCheck	21
5.4.2.6	ConfigurationConsistenceConfFileSetupEssentialWithoutInitCheckNoPrint	21
5.5	ConfObject Class Reference	22
5.5.1	Detailed Description	24
5.5.2	Constructor & Destructor Documentation	24
5.5.2.1	ConfObject	24
5.5.3	Member Function Documentation	24
5.5.3.1	PrintAll	24
5.5.3.2	PrintAllHuman	24
5.5.4	Member Data Documentation	24
5.5.4.1	Address_register	24
5.5.4.2	channel_enable_mask	24
5.5.4.3	channels	25
5.5.4.4	ConetNode	25
5.5.4.5	Data_register	25
5.5.4.6	dc_offset	25
5.5.4.7	decimation_factor	25
5.5.4.8	desmod	25
5.5.4.9	DSR4_Frequency	25
5.5.4.10	enable_fast_trigger_digitizing	25
5.5.4.11	enable_input	26
5.5.4.12	external_trigger_acquisition_mode	26
5.5.4.13	fast_trigger_acquisition_mode	26
5.5.4.14	fasts	26
5.5.4.15	gnuplot	26
5.5.4.16	group_enable_mask	26
5.5.4.17	groups	26
5.5.4.18	header_yes_no	26
5.5.4.19	LinkNumber	26
5.5.4.20	LinkType	27
5.5.4.21	Mask_register	27

5.5.4.22	max_num_events_BLT	27
5.5.4.23	nim_ttl	27
5.5.4.24	output_file_format	27
5.5.4.25	post_trigger	27
5.5.4.26	record_length	27
5.5.4.27	rising_falling	28
5.5.4.28	self_trigger_enable_mask	28
5.5.4.29	self_trigger_enable_mask_mode	28
5.5.4.30	test_pattern	28
5.5.4.31	trigger_threshold	28
5.5.4.32	use_interrupt	28
5.5.4.33	VMEBaseAddress	28
5.6	DigitizerErrorObject Class Reference	29
5.6.1	Detailed Description	29
5.6.2	Member Function Documentation	29
5.6.2.1	DigitizerErrorObjectDebugging	29
5.6.2.2	DigitizerErrorObjectDebuggingLog	29
5.6.2.3	DigitizerErrorObjectPrintError	30
5.7	DigitizerFlowControl Class Reference	30
5.7.1	Detailed Description	31
5.7.2	Constructor & Destructor Documentation	31
5.7.2.1	DigitizerFlowControl	31
5.7.3	Member Function Documentation	31
5.7.3.1	DigitizerFlowControlStart	31
5.7.3.2	Help	31
5.7.3.3	ParseCommand	32
5.7.3.4	reg_matches	33
5.7.4	Member Data Documentation	33
5.7.4.1	digitizer_flow_control_application_setup	33
5.7.4.2	input_flow_cond	33
5.7.4.3	input_flow_mutex	33
5.7.4.4	mioinput	33
5.7.4.5	mioTCP	33
5.8	DigitizerObject Class Reference	34
5.8.1	Detailed Description	36
5.8.2	Constructor & Destructor Documentation	36
5.8.2.1	DigitizerObject	36
5.8.2.2	DigitizerObject	36
5.8.2.3	DigitizerObject	36
5.8.3	Member Function Documentation	36

5.8.3.1	DigitizerObjectClose	36
5.8.3.2	DigitizerObjectGetDRS4SamplingFrequency	37
5.8.3.3	DigitizerObjectGetInfo	38
5.8.3.4	DigitizerObjectOpen	38
5.8.3.5	DigitizerObjectReadRegister	38
5.8.3.6	DigitizerObjectReset	38
5.8.3.7	DigitizerObjectSetAcquisitionMode	38
5.8.3.8	DigitizerObjectSetChannelDCOffset	39
5.8.3.9	DigitizerObjectSetConfigStructureConfObject	39
5.8.3.10	DigitizerObjectSetConfigStructureInit	39
5.8.3.11	DigitizerObjectSetConfigStructureSetup	40
5.8.3.12	DigitizerObjectSetDRS4SamplingFrequency	40
5.8.3.13	DigitizerObjectSetExtTriggerInputMode	40
5.8.3.14	DigitizerObjectSetGroupDCOffset	40
5.8.3.15	DigitizerObjectSetGroupEnableMask	41
5.8.3.16	DigitizerObjectSetMaxNumEventsBLT	42
5.8.3.17	DigitizerObjectSetRecordLength	42
5.8.3.18	DigitizerObjectSetSWTriggerMode	42
5.8.3.19	DigitizerObjectSWStartAcquisition	42
5.8.3.20	DigitizerObjectSWStopAcquisition	43
5.8.3.21	DigitizerObjectWriteRegister	43
5.8.3.22	GetBoardInfo	43
5.8.3.23	GetFamilyCode	43
5.8.3.24	GetFormFactorCode	43
5.8.3.25	PrintBoardInfo	43
5.8.4	Member Data Documentation	44
5.8.4.1	BoardInfo	44
5.8.4.2	handle	44
5.8.4.3	internal_config	44
5.8.4.4	logfile	44
5.8.4.5	ret	44
5.8.4.6	ret_error	44
5.8.4.7	set_board_info	44
5.9	DigitizerObjectGeneric Class Reference	45
5.9.1	Detailed Description	46
5.9.2	Constructor & Destructor Documentation	46
5.9.2.1	DigitizerObjectGeneric	46
5.9.3	Member Function Documentation	47
5.9.3.1	DigitizerObjectGenericGetDRS4SamplingFrequency	47
5.9.3.2	DigitizerObjectGenericReadRegister	47

5.9.3.3	<a href="#">DigitizerObjectGenericSetAcquisitionMode</a>	47
5.9.3.4	<a href="#">DigitizerObjectGenericSetAllInformations</a>	47
5.9.3.5	<a href="#">DigitizerObjectGenericSetChannelDCOffset</a>	47
5.9.3.6	<a href="#">DigitizerObjectGenericSetChannelSelfTriggerThreshold</a>	47
5.9.3.7	<a href="#">DigitizerObjectGenericSetDCOffset</a>	48
5.9.3.8	<a href="#">DigitizerObjectGenericSetDecimationFactor</a>	48
5.9.3.9	<a href="#">DigitizerObjectGenericSetDesMode</a>	48
5.9.3.10	<a href="#">DigitizerObjectGenericSetDRS4SamplingFrequency</a>	48
5.9.3.11	<a href="#">DigitizerObjectGenericSetEnableMask</a>	48
5.9.3.12	<a href="#">DigitizerObjectGenericSetExtTriggerInputMode</a>	49
5.9.3.13	<a href="#">DigitizerObjectGenericSetFastTriggerDigitizing</a>	49
5.9.3.14	<a href="#">DigitizerObjectGenericSetGroupDCOffset</a>	49
5.9.3.15	<a href="#">DigitizerObjectGenericSetIOLevel</a>	49
5.9.3.16	<a href="#">DigitizerObjectGenericSetMaxNumEventsBLT</a>	49
5.9.3.17	<a href="#">DigitizerObjectGenericSetPostTriggerSize</a>	50
5.9.3.18	<a href="#">DigitizerObjectGenericSetRecordLength</a>	50
5.9.3.19	<a href="#">DigitizerObjectGenericSetSelfTrigger</a>	50
5.9.3.20	<a href="#">DigitizerObjectGenericSetSWTriggerMode</a>	50
5.9.3.21	<a href="#">DigitizerObjectGenericSetTestPattern</a>	50
5.9.3.22	<a href="#">DigitizerObjectGenericSetTriggerPolarity</a>	50
5.9.3.23	<a href="#">DigitizerObjectGenericWriteRegister</a>	51
5.10	<a href="#">DigitizerStateMachine Class Reference</a>	51
5.10.1	<a href="#">Detailed Description</a>	53
5.10.2	<a href="#">Constructor &amp; Destructor Documentation</a>	53
5.10.2.1	<a href="#">DigitizerStateMachine</a>	53
5.10.3	<a href="#">Member Function Documentation</a>	54
5.10.3.1	<a href="#">Consumatore_Dispatcher</a>	54
5.10.3.2	<a href="#">DigitizerStateMachineInit</a>	54
5.10.3.3	<a href="#">DigitizerStateMachinePrintStatus</a>	54
5.10.3.4	<a href="#">DigitizerStateMachineQuit</a>	54
5.10.3.5	<a href="#">DigitizerStateMachineRawDataInit</a>	54
5.10.3.6	<a href="#">DigitizerStateMachineSendSWTrigger</a>	55
5.10.3.7	<a href="#">DigitizerStateMachineSetup</a>	55
5.10.3.8	<a href="#">DigitizerStateMachineStartReading</a>	55
5.10.3.9	<a href="#">DigitizerStateMachineStopReading</a>	55
5.10.3.10	<a href="#">DigitizerStateStartPreprocessing</a>	56
5.10.3.11	<a href="#">DigitizerStateStartRawDataWriting</a>	56
5.10.3.12	<a href="#">DigitizerStateStartVisualization</a>	56
5.10.3.13	<a href="#">DigitizerStateStopPreprocessing</a>	56
5.10.3.14	<a href="#">DigitizerStateStopRawDataWriting</a>	56



5.10.3.15 DigitizerStateStopVisualization . . . . .	56
5.10.3.16 Preprocessing . . . . .	57
5.10.3.17 Produttore . . . . .	57
5.10.3.18 Visualization . . . . .	57
5.10.4 Member Data Documentation . . . . .	57
5.10.4.1 Acquisition_Cond1 . . . . .	57
5.10.4.2 Acquisition_Cond2 . . . . .	57
5.10.4.3 BlockedConsumerInput . . . . .	57
5.10.4.4 BlockedProducerInput . . . . .	57
5.10.4.5 circular_buffer_preprocessing . . . . .	58
5.10.4.6 circular_buffer_raw_data . . . . .	58
5.10.4.7 circular_buffer_visualization . . . . .	58
5.10.4.8 coda_preprocessing . . . . .	58
5.10.4.9 coda_raw_data . . . . .	58
5.10.4.10 coda_visualization . . . . .	58
5.10.4.11 consumatore_thread . . . . .	58
5.10.4.12 digitizer . . . . .	59
5.10.4.13 go_general . . . . .	59
5.10.4.14 go_preprocessing . . . . .	59
5.10.4.15 go_raw_data . . . . .	59
5.10.4.16 go_visualization . . . . .	59
5.10.4.17 imset . . . . .	59
5.10.4.18 imstarted . . . . .	59
5.10.4.19 num_mex_preprocessing . . . . .	59
5.10.4.20 num_mex_raw_data . . . . .	59
5.10.4.21 num_mex_visualization . . . . .	60
5.10.4.22 pre_cond . . . . .	60
5.10.4.23 preprocessing_thread . . . . .	60
5.10.4.24 produttore_thread . . . . .	60
5.10.4.25 raw_cond . . . . .	60
5.10.4.26 ReservedConsumerDispatcherInputArea . . . . .	60
5.10.4.27 ReservedPreprocessingInputArea . . . . .	60
5.10.4.28 ReservedVisualizationInputArea . . . . .	60
5.10.4.29 testa_preprocessing . . . . .	60
5.10.4.30 testa_raw_data . . . . .	61
5.10.4.31 testa_visualization . . . . .	61
5.10.4.32 vis_cond . . . . .	61
5.10.4.33 visualization_thread . . . . .	61
5.11 FastObject Class Reference . . . . .	61
5.11.1 Detailed Description . . . . .	62

5.11.2	Constructor & Destructor Documentation	62
5.11.2.1	FastObject	62
5.11.3	Member Function Documentation	62
5.11.3.1	PrintFast	62
5.11.4	Member Data Documentation	62
5.11.4.1	dc_offset	62
5.11.4.2	numFast	62
5.11.4.3	set	62
5.11.4.4	trigger_threshold	63
5.12	GroupObject Class Reference	63
5.12.1	Detailed Description	63
5.12.2	Constructor & Destructor Documentation	64
5.12.2.1	GroupObject	64
5.12.3	Member Function Documentation	64
5.12.3.1	PrintGroup	64
5.12.4	Member Data Documentation	64
5.12.4.1	dc_offset	64
5.12.4.2	enable_input	64
5.12.4.3	group_trg_enable_mask	64
5.12.4.4	numGroup	64
5.12.4.5	set	64
5.12.4.6	trigger_threshold	65
5.13	Input Class Reference	65
5.13.1	Detailed Description	65
5.13.2	Constructor & Destructor Documentation	65
5.13.2.1	Input	65
5.13.3	Member Function Documentation	66
5.13.3.1	Finish	66
5.13.3.2	GetInput	66
5.13.3.3	Producer	66
5.13.4	Member Data Documentation	66
5.13.4.1	go	66
5.13.4.2	input_buffer	66
5.13.4.3	mutex1	66
5.13.4.4	num_mex	67
5.13.4.5	producer_thread	67
5.14	LogFile Class Reference	67
5.14.1	Detailed Description	68
5.14.2	Member Function Documentation	68
5.14.2.1	Instance	68

5.14.2.2	LogFileMessageOff	68
5.14.2.3	LogFileMessageOn	68
5.14.2.4	LogFileRead	68
5.14.2.5	LogFileSet	68
5.14.2.6	LogFileWrite	68
5.14.2.7	LogFileWriteString	69
5.14.3	Member Data Documentation	69
5.14.3.1	log_file_path	69
5.15	OutputModule Class Reference	69
5.15.1	Detailed Description	70
5.15.2	Member Function Documentation	70
5.15.2.1	Instance	70
5.15.2.2	Output	71
5.15.2.3	OutputFlex	71
5.15.2.4	OutputModuleSockidOff	71
5.15.2.5	OutputModuleSockidOn	71
5.15.2.6	OutputModuleStdoutOff	71
5.15.2.7	OutputModuleStdoutOn	71
5.15.2.8	StdOutInsert	71
5.15.2.9	StdOutInsertLex	71
5.15.2.10	StdOutPrint	72
5.15.2.11	TcpUserArrayDelete	72
5.15.2.12	TcpUserArrayInsert	72
5.15.2.13	TcpUserArraySendStdOut	72
5.15.3	Member Data Documentation	72
5.15.3.1	buffer	72
5.15.3.2	output_module_application_setup	72
5.15.3.3	sockid_array	72
5.16	RawData Class Reference	72
5.16.1	Detailed Description	74
5.16.2	Constructor & Destructor Documentation	74
5.16.2.1	RawData	74
5.16.3	Member Function Documentation	74
5.16.3.1	operator=	74
5.16.3.2	RawDataDecode	74
5.16.3.3	RawDataDel	74
5.16.3.4	RawDataPlot	75
5.16.3.5	RawDataRead	75
5.16.3.6	RawDataSet	75
5.16.3.7	RawDataWriteDecodeEventOnPlotFile	75

5.16.3.8	RawDataWriteOnFile	76
5.16.3.9	RawDataWriteOnFile	77
5.16.4	Member Data Documentation	77
5.16.4.1	bsize	77
5.16.4.2	buffer	77
5.16.4.3	digitizer	77
5.16.4.4	eventInfo	77
5.16.4.5	file	77
5.16.4.6	handle	78
5.16.4.7	imset	78
5.16.4.8	Mode	78
5.16.4.9	ret	78
5.16.4.10	ret_error	78
5.16.4.11	size	78
5.17	TcpUser Class Reference	78
5.17.1	Detailed Description	79
5.17.2	Constructor & Destructor Documentation	79
5.17.2.1	TcpUser	79
5.17.3	Member Data Documentation	79
5.17.3.1	command_sent_by_user	79
5.17.3.2	first_parameter	79
5.17.3.3	register_address	79
5.17.3.4	register_data	80
5.17.3.5	second_parameter	80
5.17.3.6	user_sockid	80
<b>6</b>	<b>File Documentation</b>	<b>81</b>
6.1	Analizzatore.h File Reference	81
6.1.1	Detailed Description	81
6.1.2	Function Documentation	81
6.1.2.1	AnalizzaInit	81
6.1.2.2	AnalizzaInitPrint	83
6.1.2.3	AnalizzaPrint	83
6.1.2.4	AnalizzaSetup	83
6.1.2.5	AnalizzaSetupPrint	83
6.2	Analizzatore.h	84
6.3	AnalizzatoreUtils.c File Reference	84
6.3.1	Detailed Description	85
6.3.2	Function Documentation	85
6.3.2.1	AllInformation	85

6.3.2.2	ChannelEnableMask	86
6.3.2.3	ChannelTriggerEnableMask	86
6.3.2.4	ChInformation	86
6.3.2.5	FastInformation	87
6.3.2.6	FindIntegerValue	87
6.3.2.7	FindPointer	87
6.3.2.8	GetAcquisitionMode	88
6.3.2.9	GetOpenInformation	88
6.3.2.10	GetWriteRegisterInformation	88
6.3.2.11	GroupEnableMask	89
6.3.2.12	GroupInformation	89
6.3.2.13	OutputFileFormat	89
6.3.2.14	OutputNIMTTL	89
6.3.2.15	OutputRisingFalling	90
6.3.2.16	reg_matches	90
6.3.2.17	YesNoAnswer	90
6.4	AnalizzatoreUtils.c	90
6.5	AnalizzatoreUtils.h File Reference	96
6.5.1	Detailed Description	98
6.5.2	Function Documentation	98
6.5.2.1	AllInformation	98
6.5.2.2	ChannelEnableMask	98
6.5.2.3	ChannelTriggerEnableMask	98
6.5.2.4	ChInformation	99
6.5.2.5	FastInformation	99
6.5.2.6	FindIntegerValue	99
6.5.2.7	FindPointer	100
6.5.2.8	GetAcquisitionMode	100
6.5.2.9	GetOpenInformation	100
6.5.2.10	GetWriteRegisterInformation	101
6.5.2.11	GroupEnableMask	101
6.5.2.12	GroupInformation	101
6.5.2.13	OutputFileFormat	101
6.5.2.14	OutputNIMTTL	103
6.5.2.15	OutputRisingFalling	103
6.5.2.16	YesNoAnswer	103
6.6	AnalizzatoreUtils.h	104
6.7	ApplicationSetup.cpp File Reference	104
6.7.1	Detailed Description	104
6.8	ApplicationSetup.cpp	105

6.9	ApplicationSetup.h File Reference	107
6.9.1	Detailed Description	107
6.10	ApplicationSetup.h	107
6.11	ClientApplication.c File Reference	108
6.11.1	Detailed Description	109
6.11.2	Function Documentation	109
6.11.2.1	command_parser	109
6.11.2.2	FindPointer	109
6.11.2.3	Help	109
6.11.2.4	main	110
6.11.2.5	reg_matches	111
6.11.2.6	ricevitore_function	111
6.11.3	Variable Documentation	111
6.11.3.1	comando_da_inviare	111
6.11.3.2	go	111
6.11.3.3	i	111
6.11.3.4	inputline	111
6.11.3.5	n	111
6.11.3.6	no_new_line	111
6.11.3.7	receiveline	111
6.11.3.8	sendline	112
6.11.3.9	sockfd	112
6.12	ClientApplication.c	112
6.13	CommunicationObject.cpp File Reference	117
6.13.1	Detailed Description	117
6.14	CommunicationObject.cpp	117
6.15	CommunicationObject.h File Reference	120
6.16	CommunicationObject.h	120
6.17	ConfigurationConsistence.cpp File Reference	121
6.17.1	Detailed Description	121
6.18	ConfigurationConsistence.cpp	121
6.19	ConfigurationConsistence.h File Reference	124
6.20	ConfigurationConsistence.h	125
6.21	ConfObject.cpp File Reference	125
6.22	ConfObject.cpp	125
6.23	ConfObject.h File Reference	131
6.24	ConfObject.h	131
6.25	DefineCommands.h File Reference	133
6.25.1	Detailed Description	133
6.25.2	Macro Definition Documentation	133

6.25.2.1	CHANGECONF	133
6.25.2.2	CHANGEDATA	134
6.25.2.3	CHANGELOG	134
6.25.2.4	CHECK	134
6.25.2.5	CHKCONF	134
6.25.2.6	CLOSE	134
6.25.2.7	INIT	134
6.25.2.8	MORE	134
6.25.2.9	PRESTART	134
6.25.2.10	PRESTOP	134
6.25.2.11	PRINT	134
6.25.2.12	PRINTFILES	134
6.25.2.13	QUIT	134
6.25.2.14	RAWSTART	135
6.25.2.15	RAWSTOP	135
6.25.2.16	READREGISTER	135
6.25.2.17	SEND	135
6.25.2.18	SETUP	135
6.25.2.19	START	135
6.25.2.20	STATUS	135
6.25.2.21	STOP	135
6.25.2.22	VISTART	135
6.25.2.23	VISTOP	135
6.25.2.24	WRITEREGISTER	135
6.26	DefineCommands.h	135
6.27	DefineGeneral.h File Reference	136
6.27.1	Detailed Description	136
6.27.2	Macro Definition Documentation	137
6.27.2.1	ALL	137
6.27.2.2	MAXCHANNELOBJECT	137
6.27.2.3	MAXCOMMAND	137
6.27.2.4	MAXFASTOBJECT	137
6.27.2.5	MAXGROUPOBJECT	137
6.27.2.6	ONLYTCP	137
6.27.2.7	ONLYUSER	137
6.27.2.8	PREPROCESSINGQUEUE	137
6.27.2.9	RAWDATAQUEUE	137
6.27.2.10	STANDARDBUFFERLIMIT	137
6.27.2.11	VISUALIZATIONQUEUE	138
6.28	DefineGeneral.h	138

6.29 DigitizerErrorObject.cpp File Reference . . . . .	138
6.30 DigitizerErrorObject.cpp . . . . .	138
6.31 DigitizerErrorObject.h File Reference . . . . .	146
6.32 DigitizerErrorObject.h . . . . .	146
6.33 DigitizerFlowControl.cpp File Reference . . . . .	146
6.34 DigitizerFlowControl.cpp . . . . .	147
6.35 DigitizerFlowControl.h File Reference . . . . .	159
6.36 DigitizerFlowControl.h . . . . .	159
6.37 DigitizerObject.cpp File Reference . . . . .	159
6.37.1 Detailed Description . . . . .	159
6.38 DigitizerObject.cpp . . . . .	160
6.39 DigitizerObject.h File Reference . . . . .	165
6.39.1 Detailed Description . . . . .	165
6.40 DigitizerObject.h . . . . .	165
6.41 DigitizerObjectGeneric.cpp File Reference . . . . .	167
6.42 DigitizerObjectGeneric.cpp . . . . .	167
6.43 DigitizerObjectGeneric.h File Reference . . . . .	174
6.44 DigitizerObjectGeneric.h . . . . .	174
6.45 DigitizerStateMachine.cpp File Reference . . . . .	175
6.46 DigitizerStateMachine.cpp . . . . .	175
6.47 DigitizerStateMachine.h File Reference . . . . .	183
6.48 DigitizerStateMachine.h . . . . .	183
6.49 Input.cpp File Reference . . . . .	185
6.50 Input.cpp . . . . .	185
6.51 Input.h File Reference . . . . .	186
6.52 Input.h . . . . .	187
6.53 LogFile.cpp File Reference . . . . .	187
6.54 LogFile.cpp . . . . .	187
6.55 LogFile.h File Reference . . . . .	189
6.56 LogFile.h . . . . .	190
6.57 Main.c File Reference . . . . .	190
6.57.1 Detailed Description . . . . .	191
6.57.2 Macro Definition Documentation . . . . .	191
6.57.2.1 ONLY_TCP . . . . .	191
6.57.3 Function Documentation . . . . .	191
6.57.3.1 main . . . . .	191
6.58 Main.c . . . . .	191
6.59 OutputModule.cpp File Reference . . . . .	192
6.60 OutputModule.cpp . . . . .	192
6.61 OutputModule.h File Reference . . . . .	194



6.62	OutputModule.h . . . . .	195
6.63	RawData.cpp File Reference . . . . .	195
6.64	RawData.cpp . . . . .	196
6.65	RawData.h File Reference . . . . .	201
6.66	RawData.h . . . . .	202
6.67	TcpUser.cpp File Reference . . . . .	202
6.68	TcpUser.cpp . . . . .	203
6.69	TcpUser.h File Reference . . . . .	203
6.70	TcpUser.h . . . . .	203
6.71	X742DecodeRoutines.c File Reference . . . . .	203
6.71.1	Function Documentation . . . . .	204
6.71.1.1	GetEventPtr . . . . .	204
6.71.1.2	GetNumEvents . . . . .	204
6.71.1.3	X742_DecodeEvent . . . . .	204
6.72	X742DecodeRoutines.c . . . . .	204
6.73	X742DecodeRoutines.h File Reference . . . . .	209
6.73.1	Macro Definition Documentation . . . . .	209
6.73.1.1	EVENT_HEADER_SIZE . . . . .	209
6.73.1.2	X742_FIXED_SIZE . . . . .	209
6.73.1.3	X742_MAX_GROUPS . . . . .	209
6.73.2	Function Documentation . . . . .	209
6.73.2.1	GetEventPtr . . . . .	209
6.73.2.2	GetNumEvents . . . . .	209
6.73.2.3	X742_DecodeEvent . . . . .	209
6.74	X742DecodeRoutines.h . . . . .	210



# Chapter 1

## objectDump application

objectDump is an useful support to data acquisition with CAEN digitizers. The application has been tested with a x742 board. objectDump is widely modular so extending it is easy. The user can adapt the application to his purpose.

**Author**

Daniele Berto



## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

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

ApplicationSetup . . . . .	9
ChannelObject . . . . .	13
CommunicationObject . . . . .	15
ConfigurationConsistence . . . . .	19
ConfObject . . . . .	22
DigitizerErrorObject . . . . .	29
DigitizerFlowControl . . . . .	30
DigitizerObject . . . . .	34
DigitizerObjectGeneric . . . . .	45
DigitizerStateMachine . . . . .	51
FastObject . . . . .	61
GroupObject . . . . .	63
Input . . . . .	65
LogFile . . . . .	67
OutputModule . . . . .	69
RawData . . . . .	72
TcpUser . . . . .	78



## Chapter 3

# Class Index

### 3.1 Class List

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

<a href="#">ApplicationSetup</a>	Gets the application settings from the main parameters . . . . .	9
<a href="#">ChannelObject</a>	Picks up all the settings taken from the configuration file regarding one channel of the digitizer .	13
<a href="#">CommunicationObject</a>	. . . . .	15
<a href="#">ConfigurationConsistence</a>	Useful methods for checking the consistence of a configuration file . . . . .	19
<a href="#">ConfObject</a>	Picks up all the settings taken from the configuration file . . . . .	22
<a href="#">DigitizerErrorObject</a>	Two methods that print the meaning of the CAEN_DGTZ_ErrorCode . . . . .	29
<a href="#">DigitizerFlowControl</a>	Controls the flow of execution of the program . . . . .	30
<a href="#">DigitizerObject</a>	Envelops CAEN_DGTZ functions from CAENDigitizer library . . . . .	34
<a href="#">DigitizerObjectGeneric</a>	Extends the <a href="#">DigitizerObject</a> class whith methods for setting the digitizer using "internal_config" attribute . . . . .	45
<a href="#">DigitizerStateMachine</a>	. . . . .	51
<a href="#">FastObject</a>	Picks up all the settings taken from the configuration file regarding a TRn input channel of the digitizer (only for x742 series) . . . . .	61
<a href="#">GroupObject</a>	Picks up all the settings taken from the configuration file regarding one group of the digitizer . .	63
<a href="#">Input</a>	. . . . .	65
<a href="#">LogFile</a>	The <a href="#">LogFile</a> singleton is used to write the error codes retrieved by the CAENDigitizer function to the logfile indicates by the log_file_path attribute . . . . .	67
<a href="#">OutputModule</a>	Useful way to manage the output of the program . . . . .	69
<a href="#">RawData</a>	Manages the readout from the digitizer . . . . .	72
<a href="#">TcpUser</a>	Useful way to store data about the users of the server . . . . .	78





## Chapter 4

# File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

<a href="#">Analizzatore.h</a>	
This file contains the declarations of the functions used to scan the config file . . . . .	81
<a href="#">AnalizzatoreUtils.c</a>	
This file contains the implementation of the utility functions used by the scanner . . . . .	84
<a href="#">AnalizzatoreUtils.h</a>	
This file contains the declaration of the utility functions used by the scanner . . . . .	96
<a href="#">ApplicationSetup.cpp</a> . . . . .	104
<a href="#">ApplicationSetup.h</a> . . . . .	107
<a href="#">ClientApplication.c</a>	
This file contains the client_application . . . . .	108
<a href="#">CommunicationObject.cpp</a> . . . . .	117
<a href="#">CommunicationObject.h</a> . . . . .	120
<a href="#">ConfigurationConsistence.cpp</a> . . . . .	121
<a href="#">ConfigurationConsistence.h</a> . . . . .	124
<a href="#">ConfObject.cpp</a> . . . . .	125
<a href="#">ConfObject.h</a> . . . . .	131
<a href="#">DefineCommands.h</a>	
This file contains the codes of the commands used to send command via TCP/IP . . . . .	133
<a href="#">DefineGeneral.h</a>	
This file contains same parameters used by the program . . . . .	136
<a href="#">DigitizerErrorObject.cpp</a> . . . . .	138
<a href="#">DigitizerErrorObject.h</a> . . . . .	146
<a href="#">DigitizerFlowControl.cpp</a> . . . . .	146
<a href="#">DigitizerFlowControl.h</a> . . . . .	159
<a href="#">DigitizerObject.cpp</a> . . . . .	159
<a href="#">DigitizerObject.h</a> . . . . .	165
<a href="#">DigitizerObjectGeneric.cpp</a> . . . . .	167
<a href="#">DigitizerObjectGeneric.h</a> . . . . .	174
<a href="#">DigitizerStateMachine.cpp</a> . . . . .	175
<a href="#">DigitizerStateMachine.h</a> . . . . .	183
<a href="#">Input.cpp</a> . . . . .	185
<a href="#">Input.h</a> . . . . .	186
<a href="#">LogFile.cpp</a> . . . . .	187
<a href="#">LogFile.h</a> . . . . .	189
<a href="#">Main.c</a>	
This file contains the program main . . . . .	190
<a href="#">OutputModule.cpp</a> . . . . .	192
<a href="#">OutputModule.h</a> . . . . .	194

<a href="#">RawData.cpp</a>	195
<a href="#">RawData.h</a>	201
<a href="#">TcpUser.cpp</a>	202
<a href="#">TcpUser.h</a>	203
<a href="#">X742DecodeRoutines.c</a>	203
<a href="#">X742DecodeRoutines.h</a>	209

## Chapter 5

# Class Documentation

### 5.1 ApplicationSetup Class Reference

The [ApplicationSetup](#) class gets the application settings from the main parameters.

```
#include <ApplicationSetup.h>
```

#### Public Member Functions

- int [ApplicationSetupDataFileModify](#) (const char \*application\_setup\_data\_file\_path\_arg)  
*The ApplicationSetupDataFileModify method modifies the path of the data file and opens it.*
- FILE \* [ApplicationSetupGetDataFilePunt](#) ()  
*The ApplicationSetupGetDataFilePunt method modifies copies in application\_setup\_data\_file\_punt\_arg the private attribute application\_setup\_data\_file\_punt.*
- FILE \* [ApplicationSetupGetDataFileSizePunt](#) ()  
*The ApplicationSetupGetDataFileSziePunt method modifies copies in application\_setup\_data\_file\_punt\_arg the private attribute application\_setup\_data\_file\_size\_punt.*
- void [ArgumentsParsing](#) ()  
*The ArgumentsParsing() method extracts informations from argc and argv.*
- void [ApplicationSetupSet](#) (int argc, char \*\*argv)  
*The ApplicationSetupSet method picks up settings informations from the main arguments using ArgumentsParsing() method.*
- void [FetchInputMode](#) (const char \*application\_setup\_input\_mode)  
*The FetchInputMode method picks up the input mode from argv and puts it in input\_mode variable.*

#### Static Public Member Functions

- static [ApplicationSetup](#) \* [Instance](#) ()  
*Instance() method is used to implement the singleton design pattern: it returns a reference to ApplicationSetup.*

#### Public Attributes

- int [channel\\_visualized](#)  
*The channel\_visualized variable indicates the channel being visualized.*
- [ConfObject](#) [application\\_setup\\_conf\\_object](#)  
*The application\_setup\_conf\_object variable is used for application settings.*
- FILE \* [application\\_setup\\_conf\\_file](#)  
*The application\_setup\_conf\_file variable is used to open the configuration file.*

- int [argc](#)  
*The argc variable is used for counting the number of the main arguments.*
- int [imset](#)  
*The imset variable indicates if the object has been set.*
- char \*\* [argv](#)  
*The argv variable contains the main arguments.*
- const char \* [application\\_setup\\_conf\\_file\\_path](#)  
*The application\_setup\_conf\_file\_path variable contains the path of the configuration file.*
- const char \* [application\\_setup\\_log\\_file\\_path](#)  
*The application\_setup\_log\_file\_path variable contains the path of the log file.*
- const char \* [application\\_setup\\_data\\_file\\_path](#)  
*The application\_setup\_data\_file\_path variable contains the path of the rawdata file.*
- const char \* [application\\_setup\\_data\\_file\\_size\\_path](#)  
*The application\_setup\_data\_file\_size\_path variable contains the path of the file contains the rawdata sizes.*
- const char \* [application\\_setup\\_input\\_mode](#)  
*The application\_setup\_input\_mode variable contains the input mode (as it is inserted with -m flag).*
- FILE \* [application\\_setup\\_data\\_file\\_punt](#)  
*The application\_setup\_data\_file\_punt variable contains a pointer to the rawdata file.*
- FILE \* [application\\_setup\\_data\\_file\\_size\\_punt](#)  
*The application\_setup\_data\_file\_size\_punt variable contains a pointer to the file contains the rawdata sizes.*
- int [input\\_mode](#)  
*The input\_mode variable contains the input mode chosed by the user.*

### 5.1.1 Detailed Description

The [ApplicationSetup](#) class gets the application settings from the main parameters.

Il singleton [ApplicationSetup](#) permette di salvare e rendere disponibili in tutte le parti del programma le informazioni necessarie per il funzionamento dell'applicazione. Esse sono inserite dall'utente lanciando il programma o inserendo l'apposito input ad applicazione avviata. Dunque, [ApplicationSetup](#) ricava le informazioni da argc e argv, cioe' dagli argomenti del main. Occorre che la funzione main chiami il metodo ApplicationSetupSet(argc, argv) per riempire [ApplicationSetup](#). Gli argomenti del main vengono analizzati in modo classico, utilizzando la funzione "getopt".

#### Author

Daniele Berto

Definition at line 15 of file [ApplicationSetup.h](#).

### 5.1.2 Member Function Documentation

#### 5.1.2.1 int ApplicationSetup::ApplicationSetupDataFileModify ( const char \* [application\\_setup\\_data\\_file\\_path\\_arg](#) )

The ApplicationSetupDataFileModify method modifies the path of the data file and opens it.

Definition at line 82 of file [ApplicationSetup.cpp](#).

#### 5.1.2.2 FILE \* ApplicationSetup::ApplicationSetupGetDataFilePunt ( )

The ApplicationSetupGetDataFilePunt method modifies copies in [application\\_setup\\_data\\_file\\_punt\\_arg](#) the private attrivute [application\\_setup\\_data\\_file\\_punt](#).

Definition at line 108 of file [ApplicationSetup.cpp](#).

### 5.1.2.3 FILE \* ApplicationSetup::ApplicationSetupGetDataFileSizePunt ( )

The ApplicationSetupGetDataFileSziePunt method modifies copies in application\_setup\_data\_file\_punt\_arg the private attribute application\_setup\_data\_file\_size\_punt.

Definition at line 115 of file [ApplicationSetup.cpp](#).

### 5.1.2.4 void ApplicationSetup::ApplicationSetupSet ( int argc, char \*\* argv )

The ApplicationSetupSet method picks up settings informations from the main arguments using [ArgumentsParsing\(\)](#) method.

La funzione ApplicationSetupSet salva nelle variabili argc e argv gli argomenti del main.

The caller has to pass argc and argv of the main function.

#### Parameters

<i>argc</i>	is the number of the main arguments
<i>argv</i>	are the main arguments.

#### Returns

void

Definition at line 66 of file [ApplicationSetup.cpp](#).

### 5.1.2.5 void ApplicationSetup::ArgumentsParsing ( )

The [ArgumentsParsing\(\)](#) method extracts informations from argc and argv.

Definition at line 155 of file [ApplicationSetup.cpp](#).

### 5.1.2.6 void ApplicationSetup::FetchInputMode ( const char \* application\_setup\_input\_mode )

The FetchInputMode method picks up the input mode from argv and puts it in input\_mode variable.

This method is called by application\_setup\_set method.

#### Returns

void

Definition at line 122 of file [ApplicationSetup.cpp](#).

### 5.1.2.7 ApplicationSetup \* ApplicationSetup::Instance ( ) [static]

[Instance\(\)](#) method is used to implement the singleton design pattern: it returns a reference to [ApplicationSetup](#).

Definition at line 26 of file [ApplicationSetup.cpp](#).

## 5.1.3 Member Data Documentation

### 5.1.3.1 FILE\* ApplicationSetup::application\_setup\_conf\_file

The application\_setup\_conf\_file variable is used to open the configuration file.

Definition at line 48 of file [ApplicationSetup.h](#).

#### 5.1.3.2 `const char* ApplicationSetup::application_setup_conf_file_path`

The `application_setup_conf_file_path` variable contains the path of the configuration file.

Definition at line 84 of file [ApplicationSetup.h](#).

#### 5.1.3.3 `ConfObject ApplicationSetup::application_setup_conf_object`

The `application_setup_conf_object` variable is used for application settings.

Definition at line 43 of file [ApplicationSetup.h](#).

#### 5.1.3.4 `const char* ApplicationSetup::application_setup_data_file_path`

The `application_setup_data_file_path` variable contains the path of the rawdata file.

Definition at line 94 of file [ApplicationSetup.h](#).

#### 5.1.3.5 `FILE* ApplicationSetup::application_setup_data_file_punt`

The `application_setup_data_file_punt` variable contains a pointer to the rawdata file.

Definition at line 109 of file [ApplicationSetup.h](#).

#### 5.1.3.6 `const char* ApplicationSetup::application_setup_data_file_size_path`

The `application_setup_data_file_size_path` variable contains the path of the file contains the rawdata sizes.

Definition at line 99 of file [ApplicationSetup.h](#).

#### 5.1.3.7 `FILE* ApplicationSetup::application_setup_data_file_size_punt`

The `application_setup_data_file_size_punt` variable contains a pointer to the file contains the rawdata sizes.

Definition at line 114 of file [ApplicationSetup.h](#).

#### 5.1.3.8 `const char* ApplicationSetup::application_setup_input_mode`

The `application_setup_input_mode` variable contains the input mode (as it is inserted with -m flag).

Definition at line 104 of file [ApplicationSetup.h](#).

#### 5.1.3.9 `const char* ApplicationSetup::application_setup_log_file_path`

The `application_setup_log_file_path` variable contains the path of the log file.

Definition at line 89 of file [ApplicationSetup.h](#).

#### 5.1.3.10 `int ApplicationSetup::argc`

The `argc` variable is used for counting the number of the main arguments.

Definition at line 53 of file [ApplicationSetup.h](#).

#### 5.1.3.11 char\*\* ApplicationSetup::argv

The argv variable contains the main arguments.

Definition at line 63 of file [ApplicationSetup.h](#).

#### 5.1.3.12 int ApplicationSetup::channel\_visualized

The channel\_visualized variable indicates the channel being visualized.

The variable is modified by the command "vistart [channelnumber]".

Definition at line 33 of file [ApplicationSetup.h](#).

#### 5.1.3.13 int ApplicationSetup::imset

The imset variable indicates if the object has been set.

Definition at line 58 of file [ApplicationSetup.h](#).

#### 5.1.3.14 int ApplicationSetup::input\_mode

The input\_mode variable contains the input mode chosed by the user.

Definition at line 119 of file [ApplicationSetup.h](#).

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

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

## 5.2 ChannelObject Class Reference

The [ChannelObject](#) class picks up all the settings taken from the configuration file regarding one channel of the digitizer.

```
#include <ConfObject.h>
```

### Public Member Functions

- [ChannelObject](#) ()  
*The [ChannelObject\(\)](#) constructor sets all the values of the channel\_object attributes to -1.*
- void [PrintChannel](#) ()  
*The [PrintChannel\(\)](#) method prints all the channel\_object attributes.*

### Public Attributes

- int [set](#)  
*The set variable indicates if a channel has been set or not.*
- int [numChannel](#)  
*The numChannel variable indicates the number of the channel.*
- int [enable\\_input](#)  
*The enable\_input variable indicates if a channel can receive input or not.*
- int [dc\\_offset](#)

*The variable `dc_offset` indicates the `dc_offset` being applicated to the channel.*

- int [trigger\\_threshold](#)

*The variable `trigger_threshold` contains the threshold for the channel auto trigger.*

- int [channel\\_trigger](#)

*The variable `channel_trigger` contains the channel auto trigger settings.*

### 5.2.1 Detailed Description

The [ChannelObject](#) class picks up all the settings taken from the configuration file regarding one channel of the digitizer.

The number of the channel is indicated by `numChannel` variable. If the channel has not been set, the set variable is set to -1 like the other unsetted variables.

#### Author

Daniele Berto

Definition at line 17 of file [ConfObject.h](#).

### 5.2.2 Constructor & Destructor Documentation

#### 5.2.2.1 ChannelObject::ChannelObject ( )

The [ChannelObject\(\)](#) constructor sets all the values of the `channel_object` attributes to -1.

Definition at line 56 of file [ConfObject.cpp](#).

### 5.2.3 Member Function Documentation

#### 5.2.3.1 void ChannelObject::PrintChannel ( )

The [PrintChannel\(\)](#) method prints all the `channel_object` attributes.

#### Returns

void

Definition at line 88 of file [ConfObject.cpp](#).

### 5.2.4 Member Data Documentation

#### 5.2.4.1 int ChannelObject::channel\_trigger

The variable `channel_trigger` contains the channel auto trigger settings.

Values: 0 = ACQUISITION\_ONLY, 1 = ACQUISITION\_AND\_TRGOUT, 2 = DISABLED, other=UNSET.

Definition at line 50 of file [ConfObject.h](#).

#### 5.2.4.2 int ChannelObject::dc\_offset

The variable `dc_offset` indicates the `dc_offset` being applicated to the channel.

Definition at line 39 of file [ConfObject.h](#).



## 5.2.4.3 int ChannelObject::enable\_input

The enable\_input variable indicates if a channel can receive input or not.

Definition at line 34 of file [ConfObject.h](#).

## 5.2.4.4 int ChannelObject::numChannel

The numChannel variable indicates the number of the channel.

Definition at line 29 of file [ConfObject.h](#).

## 5.2.4.5 int ChannelObject::set

The set variable indicates if a channel has been set or not.

Definition at line 24 of file [ConfObject.h](#).

## 5.2.4.6 int ChannelObject::trigger\_threshold

The variable trigger\_threshold contains the threshold for the channel auto trigger.

Definition at line 44 of file [ConfObject.h](#).

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

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

## 5.3 CommunicationObject Class Reference

```
#include <CommunicationObject.h>
```

### Public Member Functions

- void [GetCommand](#) (int \*socketid, int \*command)  
*The GetCommand method copies in the parameters the socket id of the sender and the command sent by it.*
- [TcpUser](#) [GetCommand](#) ()  
*The GetCommand method with no arguments return a command sent by someone to the application.*
- void [CommunicationObjectInit](#) ()  
*The CommunicationObjectInit method creates all the things needed by the threads.*
- void [Main](#) ()  
*The main\_thread initializes a socket master and listen to connection requests.*
- void [Worker](#) (void \*socket\_desc)  
*The worker\_thread manage the incoming input from a client and put it in the circular buffer command[[MAXCOMMA-ND](#)].*
- void [Finish](#) ()  
*The Finish method sets to 0 the go variable for calling forth the threads to exit.*
- [CommunicationObject](#) ()  
*The CommunicationObject constructor sets to 0 num\_mex and coda variables.*

## Public Attributes

- int [num\\_mex](#)  
The `num_mex` variable records the number of messages stored in the circular buffer `command[MAXCOMMAND]`.
- int [testa](#)  
The `testa` variable is the head pointer.
- int [coda](#)  
The `coda` variable is the tail pointer.
- int [go](#)  
The `go` variable controls the main and the worker threads.
- [TcpUser command](#) [3]  
`command[MAXCOMMAND]` is the circular buffer where the worker threads put the input sent by the clients.
- thread \* [main\\_thread](#)  
The `main_thread` variable is the main thread handler.
- thread \* [worker\\_thread](#)  
The thread id of the Worker thread.
- mutex [ReservedKeyBoardInputArea](#)  
A mutex used to implement producer-consumer model.
- condition\_variable [BlockedProducerInput](#)  
A cond variable used to implement producer-consumer model.
- condition\_variable [BlockedConsumerInput](#)  
A cond variable used to implement producer-consumer model.
- mutex [Acquisition\\_Mutex](#)  
A mutex used to implement producer-consumer model.
- condition\_variable [Acquisition\\_Cond1](#)  
A cond variable used to implement producer-consumer model.
- condition\_variable [Acquisition\\_Cond2](#)  
A cond variable used to implement producer-consumer model.

### 5.3.1 Detailed Description

Definition at line 21 of file [CommunicationObject.h](#).

### 5.3.2 Constructor & Destructor Documentation

#### 5.3.2.1 [CommunicationObject::CommunicationObject \( \)](#)

The `CommunicationObject` constructor sets to 0 `num_mex` and `coda` variables.

It also sets to 1 `testa` variable.

Definition at line 203 of file [CommunicationObject.cpp](#).

### 5.3.3 Member Function Documentation

#### 5.3.3.1 [void CommunicationObject::CommunicationObjectInit \( \)](#)

The `CommunicationObjectInit` method creates all the things needed by the threads.

This method initializes the mutexes and the cond variables and sets `go` to 1 and creates the `main_thread`.

#### Returns

void

Definition at line 150 of file [CommunicationObject.cpp](#).

#### 5.3.3.2 void CommunicationObject::Finish ( )

The Finish method sets to 0 the go variable for calling forth the threads to exit.

##### Returns

void

Definition at line 191 of file [CommunicationObject.cpp](#).

#### 5.3.3.3 void CommunicationObject::GetCommand ( int \* *socketid*, int \* *command* )

The GetCommand method copies in the parameters the socket id of the sender and the command sent by it.

##### Parameters

<i>socketid</i>	is the socket id of the sender.
<i>command</i>	is the command sent by the sender.

##### Returns

void

#### 5.3.3.4 TcpUser CommunicationObject::GetCommand ( )

The GetCommand method with no arguments return a command sent by someone to the application.

##### Returns

The command sent by someone.  
[TcpUser](#)

Definition at line 160 of file [CommunicationObject.cpp](#).

#### 5.3.3.5 void CommunicationObject::Main ( )

The main\_thread initializes a socket master and listen to connection requests.

It creates a worker\_thread for each incoming connection. Where variable go is set to zero it closes the socket master and finish.

Definition at line 38 of file [CommunicationObject.cpp](#).

#### 5.3.3.6 void CommunicationObject::Worker ( void \* *socket\_desc* )

The worker\_thread manage the incoming input from a client and put it in the circular buffer command[MAXCOMM-AND].

Every worker\_thread is created by the thread\_main.

##### Parameters

<i>socket_desc</i>	is the socket id of the client served by the worker_thread
--------------------	--

Definition at line 94 of file [CommunicationObject.cpp](#).

### 5.3.4 Member Data Documentation

#### 5.3.4.1 `condition_variable CommunicationObject::Acquisition_Cond1`

A cond variable used to implement producer-consumer model.

Definition at line 128 of file [CommunicationObject.h](#).

#### 5.3.4.2 `condition_variable CommunicationObject::Acquisition_Cond2`

A cond variable used to implement producer-consumer model.

Definition at line 133 of file [CommunicationObject.h](#).

#### 5.3.4.3 `mutex CommunicationObject::Acquisition_Mutex`

A mutex used to implement producer-consumer model.

Definition at line 123 of file [CommunicationObject.h](#).

#### 5.3.4.4 `condition_variable CommunicationObject::BlockedConsumerInput`

A cond variable used to implement producer-consumer model.

Definition at line 118 of file [CommunicationObject.h](#).

#### 5.3.4.5 `condition_variable CommunicationObject::BlockedProducerInput`

A cond variable used to implement producer-consumer model.

Definition at line 113 of file [CommunicationObject.h](#).

#### 5.3.4.6 `int CommunicationObject::coda`

The coda variable is the tail pointer.

Definition at line 38 of file [CommunicationObject.h](#).

#### 5.3.4.7 `TcpUser CommunicationObject::command[3]`

`command[MAXCOMMAND]` is the circular buffer where the worker threads put the input sent by the clients.

When `get_command` method is called, a value is picked by the circular buffer and returned to the caller. The access to the buffer is managed by a producers-consumer schema: the worker threads are the producers, the `get_command` method is the consumer.

Definition at line 87 of file [CommunicationObject.h](#).

#### 5.3.4.8 `int CommunicationObject::go`

The go variable controls the main and the worker threads.

Definition at line 43 of file [CommunicationObject.h](#).

5.3.4.9 `thread*` `CommunicationObject::main_thread`

The `main_thread` variable is the main thread handler.

Definition at line 98 of file [CommunicationObject.h](#).

5.3.4.10 `int` `CommunicationObject::num_mex`

The `num_mex` variable records the number of messages stored in the circular buffer `command[MAXCOMMAND]`.

Definition at line 28 of file [CommunicationObject.h](#).

5.3.4.11 `mutex` `CommunicationObject::ReservedKeyBoardInputArea`

A mutex used to implement producer-consumer model.

Definition at line 108 of file [CommunicationObject.h](#).

5.3.4.12 `int` `CommunicationObject::testa`

The `testa` variable is the head pointer.

Definition at line 33 of file [CommunicationObject.h](#).

5.3.4.13 `thread*` `CommunicationObject::worker_thread`

The thread id of the Worker thread.

Definition at line 103 of file [CommunicationObject.h](#).

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

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

## 5.4 ConfigurationConsistence Class Reference

The [ConfigurationConsistence](#) class provides useful methods for checking the consistence of a configuration file.

```
#include <ConfigurationConsistence.h>
```

### Public Member Functions

- `int` [ConfigurationConsistenceConfFileInit](#) (`const char *``conf_file_path`)  
*The ConfigurationConsistenceConfFileInit checks if the OPEN attribute in the configuration file is well formed.*
- `int` [ConfigurationConsistenceConfFileInitNoPrint](#) (`const char *``conf_file_path`)  
*The ConfigurationConsistenceConfFileInit checks if the OPEN attribute in the configuration file is well formed.*
- `int` [ConfigurationConsistenceConfFileSetupEssentialWithInitCheck](#) (`const char *``conf_file_path`)  
*The ConfigurationConsistenceConfFileSetupEssentialWithInitCheck checks if the OPEN attribute in the configuration file is well formed and if the other essentials setup attribute are well formed too.*
- `int` [ConfigurationConsistenceConfFileSetupEssentialWithInitCheckNoPrint](#) (`const char *``conf_file_path`)  
*The ConfigurationConsistenceConfFileSetupEssentialWithInitCheck checks if the other essentials setup attribute are well formed too.*
- `int` [ConfigurationConsistenceConfFileSetupEssentialWithoutInitCheck](#) (`const char *``conf_file_path`)

*The ConfigurationConsistenceConfFileSetupEssentialWithInitCheck checks if the other essentials setup attribute are well formed.*

- int [ConfigurationConsistenceConfFileSetupEssentialWithoutInitCheckNoPrint](#) (const char \*conf\_file\_path)

*The ConfigurationConsistenceConfFileSetupEssentialWithInitCheck checks if the other essentials setup attribute are well formed.*

### 5.4.1 Detailed Description

The [ConfigurationConsistence](#) class provides useful methods for checking the consistence of a configuration file.

This class require [Analizzatore.h](#) parser.

#### Author

Daniele Berto

Definition at line 12 of file [ConfigurationConsistence.h](#).

### 5.4.2 Member Function Documentation

#### 5.4.2.1 int ConfigurationConsistence::ConfigurationConsistenceConfFileInit ( const char \* conf\_file\_path )

The ConfigurationConsistenceConfFileInit checks if the OPEN attribute in the configuration file is well formed.

It prints recognized line.

#### Parameters

<i>conf_file_path</i>	is the path of the configuration file
-----------------------	---------------------------------------

#### Returns

int 0 if no errors are detected.

Definition at line 21 of file [ConfigurationConsistence.cpp](#).

#### 5.4.2.2 int ConfigurationConsistence::ConfigurationConsistenceConfFileInitNoPrint ( const char \* conf\_file\_path )

The ConfigurationConsistenceConfFileInit checks if the OPEN attribute in the configuration file is well formed.

It does not print recognized line.

#### Parameters

<i>conf_file_path</i>	is the path of the configuration file
-----------------------	---------------------------------------

#### Returns

int 0 if no errors are detected.

Definition at line 93 of file [ConfigurationConsistence.cpp](#).

#### 5.4.2.3 int ConfigurationConsistence::ConfigurationConsistenceConfFileSetupEssentialWithInitCheck ( const char \* conf\_file\_path )

The ConfigurationConsistenceConfFileSetupEssentialWithInitCheck checks if the OPEN attribute in the configuration file is well formed and if the other essentials setup attribute are well formed too.

It prints recognized line.

## Parameters

<i>conf_file_path</i>	is the path of the configuration file
-----------------------	---------------------------------------

## Returns

int 0 if no errors are detected.

Definition at line 147 of file [ConfigurationConsistence.cpp](#).

**5.4.2.4** `int ConfigurationConsistence::ConfigurationConsistenceConfFileSetupEssentialWithInitCheckNoPrint ( const char *  
conf_file_path )`

The ConfigurationConsistenceConfFileSetupEssentialWithInitCheck checks if the other essentials setup attribute are well formed too.

It does not print recognized line.

## Parameters

<i>conf_file_path</i>	is the path of the configuration file
-----------------------	---------------------------------------

## Returns

int 0 if no errors are detected.

Definition at line 184 of file [ConfigurationConsistence.cpp](#).

**5.4.2.5** `int ConfigurationConsistence::ConfigurationConsistenceConfFileSetupEssentialWithoutInitCheck ( const char *  
conf_file_path )`

The ConfigurationConsistenceConfFileSetupEssentialWithInitCheck checks if the other essentials setup attribute are well formed.

It prints recognized line.

## Parameters

<i>conf_file_path</i>	is the path of the configuration file
-----------------------	---------------------------------------

## Returns

int 0 if no errors are detected.

Definition at line 220 of file [ConfigurationConsistence.cpp](#).

**5.4.2.6** `int ConfigurationConsistence::ConfigurationConsistenceConfFileSetupEssentialWithoutInitCheckNoPrint ( const char *  
conf_file_path )`

The ConfigurationConsistenceConfFileSetupEssentialWithInitCheck checks if the other essentials setup attribute are well formed.

It does not print recognized line.

## Parameters

<code>conf_file_path</code>	is the path of the configuration file
-----------------------------	---------------------------------------

#### Returns

int 0 if no errors are detected.

Definition at line 254 of file [ConfigurationConsistence.cpp](#).

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

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

## 5.5 ConfObject Class Reference

The [ConfObject](#) class picks up all the settings taken from the configuration file.

```
#include <ConfObject.h>
```

### Public Member Functions

- [ConfObject](#) ()  
*The [ConfObject](#) constructor sets all the values of the `conf_object` attributes to -1.*
- void [PrintAll](#) ()  
*The [PrintAll](#) () method prints all the `conf_object` attributes.*
- void [PrintAllHuman](#) ()  
*The [PrintAllHuman](#) () method prints all the `conf_object` attributes and interprets their means.*

### Public Attributes

- int [LinkType](#)  
*The `LinkType` variable indicates the physical communication channel.*
- int [LinkNumber](#)  
*In case of USB, the link numbers are assigned by the PC when you connect the cable to the device.*
- int [ConetNode](#)  
*The CONET node identifies which device in the Daisy chain is being addressed.*
- int [VMEBaseAddress](#)  
*The VME Base Address of the board (rotary switches setting) expressed as a 32-bit number.*
- int [DSR4\\_Frequency](#)  
*The `DSR4_Frequency` indicates the sampling frequency (for X742 boards only).*
- int [output\\_file\\_format](#)  
*The `output_file_format` variable indicates the format of the output file.*
- char \* [gnuplot](#)  
*The `gnuplot` variable indicates the path to the `gnuplot` program.*
- int [header\\_yes\\_no](#)  
*The `header_yes_no` variable indicates if the output file has to have a header.*
- int [record\\_length](#)  
*The `record_length` variable indicates the size of the acquisition window, that is the number of samples that belong to it.*
- int [test\\_pattern](#)  
*The `test_pattern` variable indicates if the board has to produce a test wave.*
- int [desmod](#)



- The desmod variable indicates if the board has to enable the double edge sampling mode (only for the models 731 and 751).*
- int [external\\_trigger\\_acquisition\\_mode](#)  
*The external\_trigger\_acquisition\_mode variable indicates the operating mode of the external trigger.*
  - int [fast\\_trigger\\_acquisition\\_mode](#)  
*The fast\_trigger\_acquisition\_mode variable indicates the operating mode of the TRn.*
  - int [enable\\_fast\\_trigger\\_digitizing](#)  
*The enable\_fast\_trigger\_digitizing variable (only x742 model) enables/disables the presence of the TRn signal in the data readout.*
  - int [max\\_num\\_events\\_BLT](#)  
*The max\_num\_events\_BLT variable indicates the maximum number of events to read out in one Block Transfer.*
  - int [decimation\\_factor](#)  
*The decimation factor variable (only for 740 model) changes the decimation factor for the acquisition.*
  - int [post\\_trigger](#)  
*post trigger size in percent of the whole acquisition window.*
  - int [rising\\_falling](#)  
*rising\_falling variable decides whether the trigger occurs on the rising or falling edge of the signal.*
  - int [use\\_interrupt](#)  
*The variable use\_interrupt represents the number of events that must be ready for the readout when the IRQ is asserted.*
  - int [nim\\_ttl](#)  
*The variable nim\_ttl indicates the type of the front panel I/O LEMO connectors.*
  - int [Address\\_register](#)  
*The variable Address\_register contains the address of the register which has to be written at the end of the application setup.*
  - int [Mask\\_register](#)  
*The variable Mask\_register contains the bitmask to be used for data masking.*
  - int [Data\\_register](#)  
*The variable Data\_register represents the value being written.*
  - int [enable\\_input](#)  
*The variable enable\_input indicates if the channels can receive input or not.*
  - int [dc\\_offset](#)  
*The variable dc\_offset indicates the dc\_offset being applicated to all channels.*
  - int [trigger\\_threshold](#)  
*The variable trigger\_threshold contains the threshold for the channel auto trigger.*
  - [ChannelObject channels](#) [64]  
*The channels[MAXCHANNELOBJECT] array contains the information settings for each channel.*
  - [GroupObject groups](#) [8]  
*The groups[MAXGROUPOBJECT] array contains the information settings for each group.*
  - [FastObject fasts](#) [2]  
*The fasts[MAXFASTOBJECT] array contains the information settings for each TRn.*
  - int [channel\\_enable\\_mask](#)  
*The channel\_enable\_mask variable contains the mask for enabling the input from the channels.*
  - int [group\\_enable\\_mask](#)  
*The group\_enable\_mask variable contains the mask for enabling the input from the groups.*
  - int [self\\_trigger\\_enable\\_mask](#)  
*The trigger\_enable\_mask variable contains the mask for enabling the self trigger of the channel or of the groups (it depends on the model family).*
  - int [self\\_trigger\\_enable\\_mask\\_mode](#)  
*The trigger\_enable\_mask variable contains the mode for enabling the self trigger of the channel or of the groups (it depends on the model family).*

### 5.5.1 Detailed Description

The [ConfObject](#) class picks up all the settings taken from the configuration file.

A [ConfObject](#) is required by AnalizzaInit and AnalizzaSetup functions (see analizzatore.h file). This class is useful to sum all the setting informations in only one place.

#### Author

Daniele Berto

Definition at line 174 of file [ConfObject.h](#).

### 5.5.2 Constructor & Destructor Documentation

#### 5.5.2.1 ConfObject::ConfObject ( )

The [ConfObject](#) constructor sets all the values of the conf\_object attributes to -1.

Definition at line 14 of file [ConfObject.cpp](#).

### 5.5.3 Member Function Documentation

#### 5.5.3.1 void ConfObject::PrintAll ( )

The PrintAll () method prints all the conf\_object attributes.

#### Returns

void

Definition at line 337 of file [ConfObject.cpp](#).

#### 5.5.3.2 void ConfObject::PrintAllHuman ( )

The PrintAllHuman () method prints all the conf\_object attributes and interprets their means.

Ex: 0=NO, 1=YES, other=UNSET.

#### Returns

void

Definition at line 141 of file [ConfObject.cpp](#).

### 5.5.4 Member Data Documentation

#### 5.5.4.1 int ConfObject::Address\_register

The variable Address\_register contains the address of the register which has to be written at the end of the application setup.

Definition at line 297 of file [ConfObject.h](#).

#### 5.5.4.2 int ConfObject::channel\_enable\_mask

The channel\_enable\_mask variable contains the mask for enabling the input from the channels.

Definition at line 342 of file [ConfObject.h](#).

#### 5.5.4.3 ChannelObject ConfObject::channels[64]

The channels[MAXCHANNELOBJECT] array contains the information settings for each channel.

Definition at line 327 of file [ConfObject.h](#).

#### 5.5.4.4 int ConfObject::ConetNode

The CONET node identifies which device in the Daisy chain is being addressed.

In case of USB, ConetNode must be 0.

Definition at line 193 of file [ConfObject.h](#).

#### 5.5.4.5 int ConfObject::Data\_register

The variable Data\_register represents the value being written.

Definition at line 307 of file [ConfObject.h](#).

#### 5.5.4.6 int ConfObject::dc\_offset

The variable dc\_offset indicates the dc\_offset being applicated to all channels.

Definition at line 317 of file [ConfObject.h](#).

#### 5.5.4.7 int ConfObject::decimation\_factor

The decimation factor variable (only for 740 model) changes the decimation factor for the acquisition.

Options are 1 2 4 8 16 32 64 128.

Definition at line 268 of file [ConfObject.h](#).

#### 5.5.4.8 int ConfObject::desmod

The desmod variable indicates if the board has to enable the double edge sampling mode (only for the models 731 and 751).

Values: 0 = NO, 1 = YES, other=UNSET.

Definition at line 239 of file [ConfObject.h](#).

#### 5.5.4.9 int ConfObject::DSR4\_Frequency

The DSR4\_Frequency indicates the sampling frequency (for X742 boards only).

Values: 0 = 5GHz, 1 = 2.5GHz, 2 = 1GHz.

Definition at line 205 of file [ConfObject.h](#).

#### 5.5.4.10 int ConfObject::enable\_fast\_trigger\_digitizing

The enable\_fast\_trigger\_digitizing variable (only x742 model) enables/disables the presence of the TRn signal in the data readout.

Values: 0 = NO, 1 = YES, other=UNSET.

Definition at line 257 of file [ConfObject.h](#).

#### 5.5.4.11 int ConfObject::enable\_input

The variable enable\_input indicates if the channels can receive input or not.

Definition at line 312 of file [ConfObject.h](#).

#### 5.5.4.12 int ConfObject::external\_trigger\_acquisition\_mode

The external\_trigger\_acquisition\_mode variable indicates the operating mode of the external trigger.

Values: 0 = ACQUISITION\_ONLY, 1 = ACQUISITION\_AND\_TRGOUT, 2 = DISABLED, other=UNSET.

Definition at line 245 of file [ConfObject.h](#).

#### 5.5.4.13 int ConfObject::fast\_trigger\_acquisition\_mode

The fast\_trigger\_acquisition\_mode variable indicates the operating mode of the TRn.

Values: 0 = ACQUISITION\_ONLY, 1 = ACQUISITION\_AND\_TRGOUT, 2 = DISABLED, other=UNSET.

Definition at line 251 of file [ConfObject.h](#).

#### 5.5.4.14 FastObject ConfObject::fasts[2]

The fasts[MAXFASTOBJECT] array contains the information settings for each TRn.

Definition at line 337 of file [ConfObject.h](#).

#### 5.5.4.15 char\* ConfObject::gnuplot

The gnuplot variable indicates the path to the gnuplot program.

Definition at line 216 of file [ConfObject.h](#).

#### 5.5.4.16 int ConfObject::group\_enable\_mask

The group\_enable\_mask variable contains the mask for enabling the input from the groups.

Definition at line 347 of file [ConfObject.h](#).

#### 5.5.4.17 GroupObject ConfObject::groups[8]

The groups[MAXGROUPOBJECT] array contains the information settings for each group.

Definition at line 332 of file [ConfObject.h](#).

#### 5.5.4.18 int ConfObject::header\_yes\_no

The header\_yes\_no variable indicates if the output file has to have a header.

Values: 0 = NO, 1 = YES, other=UNSET.

Definition at line 222 of file [ConfObject.h](#).

#### 5.5.4.19 int ConfObject::LinkNumber

In case of USB, the link numbers are assigned by the PC when you connect the cable to the device.

For other details please consult CAENDigitizer library documentation.

Definition at line 188 of file [ConfObject.h](#).

#### 5.5.4.20 int ConfObject::LinkType

The LinkType variable indicates the physical communication channel.

0 indicates the USB, 1 indicates the Optical Link

Definition at line 182 of file [ConfObject.h](#).

#### 5.5.4.21 int ConfObject::Mask\_register

The variable Mask\_register contains the bitmask to be used for data masking.

Definition at line 302 of file [ConfObject.h](#).

#### 5.5.4.22 int ConfObject::max\_num\_events\_BLT

The max\_num\_events\_BLT variable indicates the maximum number of events to read out in one Block Transfer.

Definition at line 262 of file [ConfObject.h](#).

#### 5.5.4.23 int ConfObject::nim\_ttl

The variable nim\_ttl indicates the type of the front panel I/O LEMO connectors.

Values: 1 = TTL, 0 = NIM, other = UNSET

Definition at line 292 of file [ConfObject.h](#).

#### 5.5.4.24 int ConfObject::output\_file\_format

The output\_file\_format variable indicates the format of the output file.

Values: 0 = BINARY, 1 = ASCII.

Definition at line 211 of file [ConfObject.h](#).

#### 5.5.4.25 int ConfObject::post\_trigger

post trigger size in percent of the whole acquisition window.

Options: 0 to 100. On models 742 there is a delay of about 35nsec on signal Fast Trigger TR; the post trigger is added to this delay.

Definition at line 275 of file [ConfObject.h](#).

#### 5.5.4.26 int ConfObject::record\_length

The record\_length variable indicates the size of the acquisition window, that is the number of samples that belong to it.

Definition at line 227 of file [ConfObject.h](#).

#### 5.5.4.27 int ConfObject::rising\_falling

rising\_falling variable decides whether the trigger occurs on the rising or falling edge of the signal.

Values: 1 = FALLING, 0 = RISING, other=UNSET.

Definition at line 281 of file [ConfObject.h](#).

#### 5.5.4.28 int ConfObject::self\_trigger\_enable\_mask

The trigger\_enable\_mask variable contains the mask for enabling the self trigger of the channel or of the groups (it depends on the model family).

Definition at line 352 of file [ConfObject.h](#).

#### 5.5.4.29 int ConfObject::self\_trigger\_enable\_mask\_mode

The trigger\_enable\_mask variable contains the mode for enabling the self trigger of the channel or of the groups (it depends on the model family).

Definition at line 357 of file [ConfObject.h](#).

#### 5.5.4.30 int ConfObject::test\_pattern

The test\_pattern variable indicates if the board has to produce a test wave.

Values: 0 = NO, 1 = YES, other=UNSET.

Definition at line 233 of file [ConfObject.h](#).

#### 5.5.4.31 int ConfObject::trigger\_threshold

The variable trigger\_threshold contains the threshold for the channel auto trigger.

Definition at line 322 of file [ConfObject.h](#).

#### 5.5.4.32 int ConfObject::use\_interrupt

The variable use\_interrupt represents the number of events that must be ready for the readout when the IRQ is asserted.

Definition at line 286 of file [ConfObject.h](#).

#### 5.5.4.33 int ConfObject::VMEBaseAddress

The VME Base Address of the board (rotary switches setting) expressed as a 32-bit number.

This argument is used only for the VME models accessed through the VME bus and must be 0 in all other cases.

Definition at line 199 of file [ConfObject.h](#).

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

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

## 5.6 DigitizerErrorObject Class Reference

The [DigitizerErrorObject](#) class contains two methods that print the meaning of the CAEN\_DGTZ\_ErrorCode.

```
#include <DigitizerErrorObject.h>
```

### Public Member Functions

- void [DigitizerErrorObjectDebugging](#) (CAEN\_DGTZ\_ErrorCode ritorno, const char \*file, const char \*func, int line)  
*The DigitizerErrorObjectDebugging method prints the meaning of the CAEN\_DGTZ\_ErrorCode.*
- void [DigitizerErrorObjectPrintError](#) (CAEN\_DGTZ\_ErrorCode ritorno)  
*The DigitizerErrorObjectPrintError method prints the meaning of the CAEN\_DGTZ\_ErrorCode.*
- int [DigitizerErrorObjectDebuggingLog](#) (CAEN\_DGTZ\_ErrorCode ritorno, const char \*file, const char \*func, int line, FILE \*log\_file)  
*The DigitizerErrorObjectDebuggingLog method prints the meaning of the CAEN\_DGTZ\_ErrorCode in the logfile.*

### 5.6.1 Detailed Description

The [DigitizerErrorObject](#) class contains two methods that print the meaning of the CAEN\_DGTZ\_ErrorCode.

#### Author

Daniele Berto

Definition at line 13 of file [DigitizerErrorObject.h](#).

### 5.6.2 Member Function Documentation

**5.6.2.1** void DigitizerErrorObject::DigitizerErrorObjectDebugging ( CAEN\_DGTZ\_ErrorCode *ritorno*, const char \* *file*, const char \* *func*, int *line* )

The DigitizerErrorObjectDebugging method prints the meaning of the CAEN\_DGTZ\_ErrorCode.

It is called by other methods using **FILE** preprocessing variable as second paramater, **func** preprocessing variable as third parameter and **LINE** preprocessing variable as fourth parameter.

#### Parameters

<i>ritorno</i>	is the CAEN_DGTZ_ErrorCode being interpreted.
<i>file</i>	is the file which call the method: it should be the <b>FILE</b> preprocessing variable.
<i>func</i>	is the function where the method is called: it should be the <b>func</b> preprocessing variable.
<i>line</i>	is the line where the method is called: it should be the <b>LINE</b> preprocessing variable.

#### Returns

void=

Definition at line 15 of file [DigitizerErrorObject.cpp](#).

**5.6.2.2** int DigitizerErrorObject::DigitizerErrorObjectDebuggingLog ( CAEN\_DGTZ\_ErrorCode *ritorno*, const char \* *file*, const char \* *func*, int *line*, FILE \* *log\_file* )

The DigitizerErrorObjectDebuggingLog method prints the meaning of the CAEN\_DGTZ\_ErrorCode in the logfile.

**Parameters**

<i>ritorno</i>	is the CAEN_DGTZ_ErrorCode being interpreted.
<i>file</i>	is the path of the file in which the functions that generated the error code where located.
<i>func</i>	is the name of the function in which the functions that generated the error code where located.
<i>line</i>	is the line number of the functions that generated the error code.
<i>log_file</i>	is the file pointer being written with the interpretation of the CAEN_DGTZ_ErrorCode.

**Returns**

void

Definition at line 384 of file [DigitizerErrorObject.cpp](#).

### 5.6.2.3 void DigitizerErrorObject::DigitizerErrorObjectPrintError ( CAEN\_DGTZ\_ErrorCode *ritorno* )

The DigitizerErrorObjectPrintError method prints the meaning of the CAEN\_DGTZ\_ErrorCode.

**Parameters**

<i>ritorno</i>	is the CAEN_DGTZ_ErrorCode being interpreted.
----------------	---

**Returns**

void

Definition at line 221 of file [DigitizerErrorObject.cpp](#).

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

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

## 5.7 DigitizerFlowControl Class Reference

The [DigitizerFlowControl](#) class controls the flow of execution of the program.

```
#include <DigitizerFlowControl.h>
```

**Public Member Functions**

- [DigitizerFlowControl](#) ()  
The [DigitizerFlowControl](#) constructor sets the `digitizer_flow_control_application_setup` variable.
- void [DigitizerFlowControlStart](#) ()  
The [DigitizerFlowControlStart](#) method begins the main control cycle.
- const char \* [ParseCommand](#) (int recvline)  
The [ParseCommand](#) method prints the meaning of the command received via TCP.
- void [Help](#) ()  
The [help](#) method lists the commands available.
- bool [reg\\_matches](#) (const char \*str, const char \*pattern)  
The [reg\\_matcher](#) method compares the string `str` with the regex pattern.



## Public Attributes

- [Input mioinput](#)  
*The mioinput variable is used to get the input from the stdin.*
- [CommunicationObject mioTCP](#)  
*The mioTCP pointer is used to get the input from the clients via TCP.*
- [ApplicationSetup \\* digitizer\\_flow\\_control\\_application\\_setup](#)  
*The digitizer\_flow\_control\_application\_setup variable is used to configure the application.*

## Static Public Attributes

- static pthread\_cond\_t [input\\_flow\\_cond](#)  
*The input\_flow\_cond condition variable is used to notify to the object that an input command is ready to be fetched.*
- static pthread\_mutex\_t [input\\_flow\\_mutex](#)  
*The input\_flow\_mutex is used with the input\_flow\_cond to guarantee the consistency of the operations.*

### 5.7.1 Detailed Description

The [DigitizerFlowControl](#) class controls the flow of execution of the program.

#### Author

Daniele Berto

Definition at line 12 of file [DigitizerFlowControl.h](#).

### 5.7.2 Constructor & Destructor Documentation

#### 5.7.2.1 DigitizerFlowControl::DigitizerFlowControl ( )

The [DigitizerFlowControl](#) constructor sets the digitizer\_flow\_control\_application\_setup variable.

Definition at line 34 of file [DigitizerFlowControl.cpp](#).

### 5.7.3 Member Function Documentation

#### 5.7.3.1 void DigitizerFlowControl::DigitizerFlowControlStart ( )

The DigitizerFlowControlStart method begins the main control cycle.

#### Returns

void

Definition at line 43 of file [DigitizerFlowControl.cpp](#).

#### 5.7.3.2 void DigitizerFlowControl::Help ( )

The help method lists the commands available.

#### Returns

void

Definition at line 798 of file [DigitizerFlowControl.cpp](#).

#### 5.7.3.3 `const char * DigitizerFlowControl::ParseCommand ( int recvline )`

The ParseCommand method prints the meaning of the command received via TCP.

## Parameters

<i>recvline</i>	is the command to interpret.
-----------------	------------------------------

## Returns

void

Definition at line 831 of file [DigitizerFlowControl.cpp](#).

#### 5.7.3.4 bool DigitizerFlowControl::reg\_matches ( const char \* *str*, const char \* *pattern* )

The reg\_matcher method compares the string str with the regex pattern.

## Returns

bool

Definition at line 777 of file [DigitizerFlowControl.cpp](#).

### 5.7.4 Member Data Documentation

#### 5.7.4.1 ApplicationSetup\* DigitizerFlowControl::digitizer\_flow\_control\_application\_setup

The digitizer\_flow\_control\_application\_setup variable is used to configure the application.

Definition at line 40 of file [DigitizerFlowControl.h](#).

#### 5.7.4.2 pthread\_cond\_t DigitizerFlowControl::input\_flow\_cond [static]

The input\_flow\_cond condition variable is used to notify to the object that an input command is ready to be fetched.

The [DigitizerFlowControl](#) object waits on it until an [Input](#) object or a [CommunicationObject](#) wakes it up with the pthread\_cond\_signal function.

Definition at line 20 of file [DigitizerFlowControl.h](#).

#### 5.7.4.3 pthread\_mutex\_t DigitizerFlowControl::input\_flow\_mutex [static]

The input\_flow\_mutex is used with the input\_flox\_cond to guarantee the consistency of the operations.

Definition at line 25 of file [DigitizerFlowControl.h](#).

#### 5.7.4.4 Input DigitizerFlowControl::mioinput

The mioinput variable is used to get the input from the stdin.

Definition at line 30 of file [DigitizerFlowControl.h](#).

#### 5.7.4.5 CommunicationObject DigitizerFlowControl::mioTCP

The mioTCP pointer is used to get the input from the clients via TCP.

Definition at line 35 of file [DigitizerFlowControl.h](#).

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

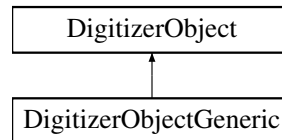
- [DigitizerFlowControl.h](#)
- [DigitizerFlowControl.cpp](#)

## 5.8 DigitizerObject Class Reference

The [DigitizerObject](#) class envelops CAEN\_DGTZ functions from CAENDigitizer library.

```
#include <DigitizerObject.h>
```

Inheritance diagram for DigitizerObject:



### Public Member Functions

- [DigitizerObject](#) ()  
The [DigitizerObject](#) default constructor sets to 0 the variables `set_board_info`, `digitizer_open` and `set_internal_config`.
- [DigitizerObject](#) (const char \*config\_file)  
The [DigitizerObject](#) (const char \*) constructor sets to 0 the variables `set_board_info`, `digitizer_open` and to 1 `set_internal_config`.
- [DigitizerObject](#) (ConfObject config)  
The [DigitizerObject](#) (ConfObject) constructor sets to 0 the variables `set_board_info`, `digitizer_open` and to 1 `set_internal_config`.
- void [DigitizerObjectSetConfigStructureConfObject](#) (ConfObject config)  
The [DigitizerObjectSetConfigStructureConfObject](#) (ConfObject) method copies the [ConfObject](#) parameter in the `internal_config` attribute.
- void [DigitizerObjectSetConfigStructureInit](#) (const char \*config\_file)  
The [DigitizerObjectSetConfigStructureInit](#) (const char \*) method scans the file in the path indicates by the parameter using `AnalizzaInit` and `AnalizzaSetup` functions from `AnalizzatoreLessicale.flex` file and stores the retrieved informations in the `internal_config` attribute.
- void [DigitizerObjectSetConfigStructureSetup](#) (const char \*config\_file)  
The [DigitizerObjectSetConfigStructureSetup](#) (const char \*) method scans the file in the path indicates by the parameter using `AnalizzaInit` and `AnalizzaSetup` functions from `AnalizzatoreLessicale.flex` file and stores the retrieved informations in the `internal_config` attribute.
- int [DigitizerObjectOpen](#) ()  
The [DigitizerObjectOpen](#) method opens the digitizer using the settings stored in the `internal_config` attribute.
- int [DigitizerObjectClose](#) ()  
The [DigitizerObjectOpen](#) method closes the digitizer.
- int [DigitizerObjectReset](#) ()  
The [DigitizerObjectReset](#) method resets the digitizer.
- int [DigitizerObjectGetInfo](#) ()  
The [DigitizerObjectGetInfo](#) method gets factory informations from the digitizer and puts them in the `BoardInfo` attribute.
- CAEN\_DGTZ\_BoardInfo\_t [GetBoardInfo](#) ()  
The [GetBoardInfo](#) method returns factory informations from the digitizer.
- int [GetFamilyCode](#) (int \*FamilyCode)  
The [GetFamilyCode](#) method puts in the parameter `FamilyCode` the family code of the digitizer.
- int [GetFormFactorCode](#) (int \*FormFactor)  
The [GetFormFactorCode](#) method puts in the parameter `FormFactor` the form factor code of the digitizer.
- void [PrintBoardInfo](#) ()  
The [PrintBoardInfo](#) method prints to the stdout the factory informations stored in the `BoardInfo` attribute.
- int [DigitizerObjectSetRecordLength](#) (int recordlength)  
The [DigitizerObjectSetRecordLength](#) method sets the size of the acquisition window.

- int [DigitizerObjectSetGroupEnableMask](#) (int enablemask)  
*The DigitizerObjectSetGroupEnableMask method sets the groups enabled to receive input in accordance with the parameter.*
- int [DigitizerObjectWriteRegister](#) (int registry, int data)  
*The DigitizerObjectWriteRegister method writes a digitizer register in accordance with the parameters.*
- int [DigitizerObjectReadRegister](#) (int registry, int \*data)  
*The DigitizerObjectReadRegister method reads a digitizer register in accordance with the parameters.*
- int [DigitizerObjectGetDRS4SamplingFrequency](#) (CAEN\_DGTZ\_DRS4Frequency\_t \*frequenza)  
*The DigitizerObjectGetDRS4SamplingFrequency method gets the sampling frequency of the digitizer.*
- int [DigitizerObjectSetDRS4SamplingFrequency](#) (CAEN\_DGTZ\_DRS4Frequency\_t frequenza)  
*The DigitizerObjectSetDRS4SamplingFrequency method sets the sampling frequency of the digitizer.*
- int [DigitizerObjectSetChannelDCOffset](#) (int channel\_mask, int dc\_mask)  
*The DigitizerObjectSetChannelDCOffset method sets the DCOffset of the channels in accordance with the parameters.*
- int [DigitizerObjectSetGroupDCOffset](#) (int group\_mask, int dc\_mask)  
*The DigitizerObjectSetGroupDCOffset method sets the DCOffset of the groups in accordance with the parameters.*
- int [DigitizerObjectSetMaxNumEventsBLT](#) (int MaxNumEventsBLT)  
*The DigitizerObjectSetMaxNumEventsBLT method sets the maximum number of events that can be trasferred in a readout cycle.*
- int [DigitizerObjectSetAcquisitionMode](#) (CAEN\_DGTZ\_AcqMode\_t AcqMode)  
*The DigitizerObjectSetAcquisitionMode method sets the data acquisition mode in accordance with the parameter.*
- int [DigitizerObjectSetExtTriggerInputMode](#) (CAEN\_DGTZ\_TriggerMode\_t TriggerMode)  
*The DigitizerObjectSetExtTriggerInputMode method sets how an external trigger has to be used.*
- int [DigitizerObjectSetSWTriggerMode](#) (CAEN\_DGTZ\_TriggerMode\_t TriggerMode)  
*The DigitizerObjectSetSWTriggerMode method sets how a software trigger has to be used.*
- int [DigitizerObjectSWStartAcquisition](#) ()  
*The DigitizerObjectSWStartAcquisition method starts the data acquisition.*
- int [DigitizerObjectSWStopAcquisition](#) ()  
*The DigitizerObjectSWStartAcquisition method stops the data acquisition.*

## Public Attributes

- CAEN\_DGTZ\_ErrorCode [ret](#)  
*The ret variable contains the error code returned by CAEN\_DGTZ library functions.*
- [DigitizerErrorObject](#) [ret\\_error](#)  
*The ret\_error object is used to print the meaning of CAEN\_DGTZ\_ErrorCode.*
- [ConfObject](#) [internal\\_config](#)  
*The internal\_config object is used to store the setting of the digitizer read from the configuration file.*
- int [set\\_board\\_info](#)  
*The set\_board\_info variable indicates if the board informations have been already picked or not.*
- int [handle](#)  
*The handle variable contains the digitizer handler device.*
- CAEN\_DGTZ\_BoardInfo\_t [BoardInfo](#)  
*The BoardInfo variable contains the informations about the board.*
- [LogFile](#) \* [logfile](#)  
*The logfile reference is used to write on logfile.*

### 5.8.1 Detailed Description

The [DigitizerObject](#) class envelops CAEN\_DGTZ functions from CAENDigitizer library.

#### Author

Daniele Berto

Definition at line 15 of file [DigitizerObject.h](#).

### 5.8.2 Constructor & Destructor Documentation

#### 5.8.2.1 DigitizerObject::DigitizerObject ( )

The [DigitizerObject](#) default constructor sets to 0 the variables set\_board\_info, digitizer\_open and set\_internal\_config.

Definition at line 47 of file [DigitizerObject.cpp](#).

#### 5.8.2.2 DigitizerObject::DigitizerObject ( const char \* config\_file )

The [DigitizerObject](#) (const char \*) constructor sets to 0 the variables set\_board\_info, digitizer\_open and to 1 set\_internal\_config.

The method gets the settings from the configuration file specified in the config\_file parameter using AnalizzaInit and AnalizzaSetup functions from AnalizzatoreLessicale.flex file. These settings are copied in the internal\_config attribute.

#### Parameters

<i>config_file</i>	is the path of the configuration file.
--------------------	--

Definition at line 18 of file [DigitizerObject.cpp](#).

#### 5.8.2.3 DigitizerObject::DigitizerObject ( ConfObject config )

The [DigitizerObject](#) (ConfObject) constructor sets to 0 the variables set\_board\_info, digitizer\_open and to 1 set\_internal\_config.

The method gets the settings from the config parameter and stores them in the internal\_config attribute.

#### Parameters

<i>config</i>	is the <a href="#">ConfObject</a> being copied in the internal_config attribute.
---------------	--

Definition at line 32 of file [DigitizerObject.cpp](#).

### 5.8.3 Member Function Documentation

#### 5.8.3.1 int DigitizerObject::DigitizerObjectClose ( )

The DigitizerObjectOpen method closes the digitizer.

#### Returns

int

Definition at line 386 of file [DigitizerObject.cpp](#).

5.8.3.2 `int DigitizerObject::DigitizerObjectGetDRS4SamplingFrequency ( CAEN_DGTZ_DRS4Frequency_t * frequenza )`

The DigitizerObjectGetDRS4SamplingFrequency method gets the sampling frequency of the digitizer.

## Parameters

<i>frequenza</i>	is where the frequency information will be stored.
------------------	--

## Returns

int

Definition at line 314 of file [DigitizerObject.cpp](#).

#### 5.8.3.3 int DigitizerObject::DigitizerObjectGetInfo ( )

The DigitizerObjectGetInfo method gets factory informations from the digitizer and puts them in the BoardInfo attribute.

## Returns

int

Definition at line 118 of file [DigitizerObject.cpp](#).

#### 5.8.3.4 int DigitizerObject::DigitizerObjectOpen ( )

The DigitizerObjectOpen method opens the digitizer using the settings stored in the internal\_config attribute.

## Returns

int

Definition at line 74 of file [DigitizerObject.cpp](#).

#### 5.8.3.5 int DigitizerObject::DigitizerObjectReadRegister ( int *registry*, int \* *data* )

The DigitizerObjectReadRegister method reads a digitizer register in accordance with the parameters.

## Parameters

<i>registry</i>	is the address of the registry being read.
<i>data</i>	are the informations being read from the registry indicated by registry parameter.

## Returns

int

Definition at line 300 of file [DigitizerObject.cpp](#).

#### 5.8.3.6 int DigitizerObject::DigitizerObjectReset ( )

The DigitizerObjectReset method resets the digitizer.

## Returns

int

Definition at line 107 of file [DigitizerObject.cpp](#).

#### 5.8.3.7 int DigitizerObject::DigitizerObjectSetAcquisitionMode ( CAEN\_DGTZ\_AcqMode\_t *AcqMode* )

The DigitizerObjectSetAcquisitionMode method sets the data acquisition mode in accordance with the parameter.



## Parameters

<i>AcqMode</i>	is the data acquisition mode being setted.
----------------	--

## Returns

int

Definition at line 373 of file [DigitizerObject.cpp](#).

#### 5.8.3.8 int DigitizerObject::DigitizerObjectSetChannelDCOffset ( int *channel\_mask*, int *dc\_mask* )

The DigitizerObjectSetChannelDCOffset method sets the DCOffset of the channels in accordance with the parameters.

## Parameters

<i>channel_mask</i>	indicates the channels being influenced by the DCOffset.
<i>dc_mask</i>	is the DAC value.

## Returns

int

Definition at line 339 of file [DigitizerObject.cpp](#).

#### 5.8.3.9 void DigitizerObject::DigitizerObjectSetConfigStructureConfObject ( ConfObject *config* )

The DigitizerObjectSetConfigStructureConfObject ([ConfObject](#)) method copies the [ConfObject](#) parameter in the internal\_config attribute.

## Parameters

<i>config</i>	is the <a href="#">ConfObject</a> being copied in the internal_config attribute.
---------------	--

## Returns

void

Definition at line 41 of file [DigitizerObject.cpp](#).

#### 5.8.3.10 void DigitizerObject::DigitizerObjectSetConfigStructureInit ( const char \* *config\_file* )

The DigitizerObjectSetConfigStructureInit (const char \*) method scans the file in the path indicates by the parameter using AnalizzaInit and AnalizzaSetup functions from AnalizzatoreLessicale.flex file and stores the retrieved informations in the internal\_config attribute.

## Parameters

<i>config_file</i>	is the path of the configuration file.
--------------------	--

## Returns

void

Definition at line 55 of file [DigitizerObject.cpp](#).

#### 5.8.3.11 void DigitizerObject::DigitizerObjectSetConfigStructureSetup ( const char \* *config\_file* )

The DigitizerObjectSetConfigStructureSetup (const char \*) method scans the file in the path indicates by the parameter using AnalizzaInit and AnalizzaSetup functions from AnalizzatoreLessicale.flex file and stores the retrieved informations in the internal\_config attribute.

##### Parameters

<i>config_file</i>	is the path of the configuration file.
--------------------	--

##### Returns

void

Definition at line 65 of file [DigitizerObject.cpp](#).

#### 5.8.3.12 int DigitizerObject::DigitizerObjectSetDRS4SamplingFrequency ( CAEN\_DGTZ\_DRS4Frequency\_t *frequenza* )

The DigitizerObjectSetDRS4SamplingFrequency method sets the sampling frequency of the digitizer.

##### Parameters

<i>frequenza</i>	is the frequency being setted.
------------------	--------------------------------

##### Returns

int

Definition at line 327 of file [DigitizerObject.cpp](#).

#### 5.8.3.13 int DigitizerObject::DigitizerObjectSetExtTriggerInputMode ( CAEN\_DGTZ\_TriggerMode\_t *TriggerMode* )

The DigitizerObjectSetExtTriggerInputMode method sets how an external trigger has to be used.

##### Parameters

<i>TriggerMode</i>	indicates how an external trigger has to be used.
--------------------	---

##### Returns

int

Definition at line 399 of file [DigitizerObject.cpp](#).

#### 5.8.3.14 int DigitizerObject::DigitizerObjectSetGroupDCOffset ( int *group\_mask*, int *dc\_mask* )

The DigitizerObjectSetGroupDCOffset method sets the DCOffset of the groups in accordance with the parameters.

##### Parameters

<i>group_mask</i>	indicates the groups being influenced by the DCOffset.
<i>dc_mask</i>	is the DAC value.

##### Returns

int

Definition at line 351 of file [DigitizerObject.cpp](#).

#### 5.8.3.15 int DigitizerObject::DigitizerObjectSetGroupEnableMask ( int *enablemask* )

The DigitizerObjectSetGroupEnableMask method sets the groups enabled to receive input in accordance with the parameter.

## Parameters

<i>enablemask</i>	is the mask that indicates which groups will receive input.
-------------------	---

## Returns

int

Definition at line 278 of file [DigitizerObject.cpp](#).

#### 5.8.3.16 int DigitizerObject::DigitizerObjectSetMaxNumEventsBLT ( int *MaxNumEventsBLT* )

The DigitizerObjectSetMaxNumEventsBLT method sets the maximum number of events that can be trasferred in a readout cycle.

## Parameters

<i>MaxNumEvents- BLT</i>	is the maximum number of events that can be trasferred in a readout cycle.
------------------------------	--

## Returns

int

Definition at line 361 of file [DigitizerObject.cpp](#).

#### 5.8.3.17 int DigitizerObject::DigitizerObjectSetRecordLength ( int *recordlength* )

The DigitizerObjectSetRecordLength method sets the size of the acquisition window.

## Parameters

<i>recordlength</i>	is the size of the acquisition window.
---------------------	--

## Returns

int

Definition at line 266 of file [DigitizerObject.cpp](#).

#### 5.8.3.18 int DigitizerObject::DigitizerObjectSetSWTriggerMode ( CAEN\_DGTZ\_TriggerMode\_t *TriggerMode* )

The DigitizerObjectSetSWTriggerMode method sets how a software trigger has to be used.

## Parameters

<i>TriggerMode</i>	indicates how a software trigger has to be used.
--------------------	--

## Returns

int

Definition at line 411 of file [DigitizerObject.cpp](#).

#### 5.8.3.19 int DigitizerObject::DigitizerObjectSWStartAcquisition ( )

The DigitizerObjectSWStartAcquisition method starts the data acquisition.

**Returns**

int

Definition at line 424 of file [DigitizerObject.cpp](#).

**5.8.3.20 int DigitizerObject::DigitizerObjectSWStopAcquisition ( )**

The DigitizerObjectSWStartAcquisition method stops the data acquisition.

**Returns**

int

Definition at line 436 of file [DigitizerObject.cpp](#).

**5.8.3.21 int DigitizerObject::DigitizerObjectWriteRegister ( int *registry*, int *data* )**

The DigitizerObjectWriteRegister method writes a digitizer register in accordance with the parameters.

**Parameters**

<i>registry</i>	is the address of the registry being written.
<i>data</i>	are the informations being written into the registry indicated by registry parameter.

**Returns**

int

Definition at line 288 of file [DigitizerObject.cpp](#).

**5.8.3.22 CAEN\_DGTZ\_BoardInfo\_t DigitizerObject::GetBoardInfo ( )**

The GetBoardInfo method returns factory informations from the digitizer.

**Returns**

CAEN\_DGTZ\_BoardInfo\_t

**5.8.3.23 int DigitizerObject::GetFamilyCode ( int \* *FamilyCode* )**

The GetFamilyCode method puts in the parameter FamilyCode the family code of the digitizer.

Definition at line 131 of file [DigitizerObject.cpp](#).

**5.8.3.24 int DigitizerObject::GetFormFactorCode ( int \* *FormFactor* )**

The GetFormFactorCode method puts in the parameter FormFactor the form factor code of the digitizer.

Definition at line 151 of file [DigitizerObject.cpp](#).

**5.8.3.25 void DigitizerObject::PrintBoardInfo ( )**

The PrintBoardInfo method prints to the stdout the factory informations stored in the BoardInfo attribute.

## Returns

void

Definition at line 171 of file [DigitizerObject.cpp](#).

## 5.8.4 Member Data Documentation

### 5.8.4.1 CAEN\_DGTZ\_BoardInfo\_t DigitizerObject::BoardInfo

The BoardInfo variable contains the informations about the board.

Definition at line 48 of file [DigitizerObject.h](#).

### 5.8.4.2 int DigitizerObject::handle

The handle variable contains the digitizer handler device.

Definition at line 43 of file [DigitizerObject.h](#).

### 5.8.4.3 ConfObject DigitizerObject::internal\_config

The internal\_config object is used to store the setting of the digitizer read from the configuration file.

Definition at line 33 of file [DigitizerObject.h](#).

### 5.8.4.4 LogFile\* DigitizerObject::logfile

The logfile reference is used to write on logfile.

Definition at line 245 of file [DigitizerObject.h](#).

### 5.8.4.5 CAEN\_DGTZ\_ErrorCode DigitizerObject::ret

The ret variable contains the error code returned by CAEN\_DGTZ library functions.

Definition at line 23 of file [DigitizerObject.h](#).

### 5.8.4.6 DigitizerErrorObject DigitizerObject::ret\_error

The ret\_error object is used to print the meaning of CAEN\_DGTZ\_ErrorCode.

Definition at line 28 of file [DigitizerObject.h](#).

### 5.8.4.7 int DigitizerObject::set\_board\_info

The set\_board\_info variable indicates if the board informations have been already picked or not.

Definition at line 38 of file [DigitizerObject.h](#).

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

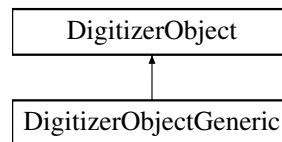
- [DigitizerObject.h](#)
- [DigitizerObject.cpp](#)

## 5.9 DigitizerObjectGeneric Class Reference

The [DigitizerObjectGeneric](#) class extends the [DigitizerObject](#) class with methods for setting the digitizer using "internal\_config" attribute.

```
#include <DigitizerObjectGeneric.h>
```

Inheritance diagram for DigitizerObjectGeneric:



### Public Member Functions

- [int DigitizerObjectGenericSetRecordLength \(\)](#)  
*The DigitizerObjectGenericSetRecordLength method sets the size of the acquisition window.*
- [int DigitizerObjectGenericSetMaxNumEventsBLT \(\)](#)  
*The DigitizerObjectGenericSetMaxNumEventsBLT method sets the maximum number of events that can be transferred in a readout cycle.*
- [int DigitizerObjectGenericSetEnableMask \(\)](#)  
*The DigitizerObjectGenericSetEnableMask method sets the groups or the channels (it depends on the model family) able to receive input in accordance with the config file settings.*
- [int DigitizerObjectGenericSetFastTriggerDigitizing \(\)](#)  
*The DigitizerObjectGenericSetFastTriggerDigitizing method sets the TRn input channel (only for the x742 family) in accordance with the config file settings.*
- [int DigitizerObjectGenericSetSelfTrigger \(\)](#)  
*The DigitizerObjectGenericSetChannelSelfTrigger method sets the groups or the channels (it depends on the model family) self trigger.*
- [int DigitizerObjectGenericSetChannelSelfTriggerThreshold \(\)](#)  
*The DigitizerObjectGenericSetChannelSelfTriggerThreshold method sets the self trigger threshold of the groups or of the channels (it depends on the model family).*
- [int DigitizerObjectGenericSetDCOffset \(\)](#)  
*The DigitizerObjectGenericSetEnableMask method sets the DCOffset of the groups or of the channels (it depends on the model family).*
- [int DigitizerObjectGenericWriteRegister \(\)](#)  
*The DigitizerObjectWriteRegister method writes a digitizer register in accordance with the config file settings.*
- [int DigitizerObjectGenericReadRegister \(\)](#)  
*The DigitizerObjectGenericReadRegister method reads a digitizer register in accordance with the config file settings.*
- [int DigitizerObjectGenericGetDRS4SamplingFrequency \(\)](#)  
*The DigitizerObjectGenericGetDRS4SamplingFrequency method gets the sampling frequency of the digitizer.*
- [int DigitizerObjectGenericSetDRS4SamplingFrequency \(\)](#)  
*The DigitizerObjectGenericSetDRS4SamplingFrequency method sets the sampling frequency of the digitizer.*
- [int DigitizerObjectGenericSetChannelDCOffset \(\)](#)  
*The DigitizerObjectGenericSetChannelDCOffset method sets the DCOffset of the channels in accordance with the config file settings.*
- [int DigitizerObjectGenericSetGroupDCOffset \(\)](#)  
*The DigitizerObjectGenericSetGroupDCOffset method sets the DCOffset of the groups in accordance with the config file settings.*
- [int DigitizerObjectGenericSetTriggerPolarity \(\)](#)  
*The DigitizerObjectGenericSetTriggerPolarity method sets the trigger polarity of a specified channel in accordance with the config file settings.*

- `int DigitizerObjectGenericSetPostTriggerSize ()`  
*The DigitizerObjectGenericSetPostTriggerSize method sets the position of the trigger within the acquisition window in accordance with the config file settings.*
- `int DigitizerObjectGenericSetIOLevel ()`  
*The DigitizerObjectGenericSetIOLevel method sets the I/O level in accordance with the config file settings.*
- `int DigitizerObjectGenericSetAcquisitionMode ()`  
*The DigitizerObjectGenericSetAcquisitionMode method sets the data acquisition mode in accordance with the config file settings.*
- `int DigitizerObjectGenericSetExtTriggerInputMode ()`  
*The DigitizerObjectGenericSetExtTriggerInputMode method sets how an external trigger has to be used.*
- `int DigitizerObjectGenericSetSWTriggerMode ()`  
*The DigitizerObjectGenericSetSWTriggerMode method sets how a software trigger has to be used.*
- `int DigitizerObjectGenericSetDecimationFactor ()`  
*The DigitizerObjectGenericSetDecimationFactor method sets the decimation factor.*
- `int DigitizerObjectGenericSetDesMode ()`  
*The DigitizerObjectGenericSetDesMode method enables the dual edge sampling (DES) mode (for 731 and 751 series only).*
- `int DigitizerObjectGenericSetTestPattern ()`  
*The DigitizerObjectGenericSetTestPattern method sets the waveform test bit for debugging.*
- `int DigitizerObjectGenericSetAllInformations ()`  
*The DigitizerObjectGenericSetAllInformations method sets the digitizer with the settings marked with "all" in the configuration file.*
- `DigitizerObjectGeneric ()`  
*The DigitizerObjectGeneric constructor sets to 0 the set\_board\_info attribute and gets an instance of the LogFile singleton.*

## Additional Inherited Members

### 5.9.1 Detailed Description

The [DigitizerObjectGeneric](#) class extends the [DigitizerObject](#) class with methods for setting the digitizer using "internal\_config" attribute.

[DigitizerObject](#) class provides methods for setting the digitizer specifying the settings in their parameters. The methods of [DigitizerObjectGeneric](#) class are parameterless because they use settings provided by internal\_config attribute.

#### Author

Daniele Berto

Definition at line 16 of file [DigitizerObjectGeneric.h](#).

### 5.9.2 Constructor & Destructor Documentation

#### 5.9.2.1 DigitizerObjectGeneric::DigitizerObjectGeneric ( )

The [DigitizerObjectGeneric](#) constructor sets to 0 the set\_board\_info attribute and gets an instance of the [LogFile](#) singleton.

Definition at line 17 of file [DigitizerObjectGeneric.cpp](#).



### 5.9.3 Member Function Documentation

#### 5.9.3.1 `int DigitizerObjectGeneric::DigitizerObjectGenericGetDRS4SamplingFrequency ( )`

The `DigitizerObjectGenericGetDRS4SamplingFrequency` method gets the sampling frequency of the digitizer.

##### Returns

`int`

#### 5.9.3.2 `int DigitizerObjectGeneric::DigitizerObjectGenericReadRegister ( )`

The `DigitizerObjectGenericReadRegister` method reads a digitizer register in accordance with the config file settings.

##### Returns

`int`

#### 5.9.3.3 `int DigitizerObjectGeneric::DigitizerObjectGenericSetAcquisitionMode ( )`

The `DigitizerObjectGenericSetAcquisitionMode` method sets the data acquisition mode in accordance with the config file settings.

##### Returns

`int`

#### 5.9.3.4 `int DigitizerObjectGeneric::DigitizerObjectGenericSetAllInformations ( )`

The `DigitizerObjectGenericSetAllInformations` method sets the digitizer with the settings marked with "all" in the configuration file.

##### Returns

`int`

Definition at line 25 of file [DigitizerObjectGeneric.cpp](#).

#### 5.9.3.5 `int DigitizerObjectGeneric::DigitizerObjectGenericSetChannelDCOffset ( )`

The `DigitizerObjectGenericSetChannelDCOffset` method sets the `DCOffset` of the channels in accordance with the config file settings.

##### Returns

`int`

#### 5.9.3.6 `int DigitizerObjectGeneric::DigitizerObjectGenericSetChannelSelfTriggerThreshold ( )`

The `DigitizerObjectGenericSetChannelSelfTriggerThreshold` method sets the self trigger threshold of the groups or of the channels (it depends on the model family).

##### Returns

`int`

Definition at line 447 of file [DigitizerObjectGeneric.cpp](#).

#### 5.9.3.7 int DigitizerObjectGeneric::DigitizerObjectGenericSetDCOffset ( )

The DigitizerObjectGenericSetEnableMask method sets the DCOffset of the groups or of the channels (it depends on the model family).

##### Returns

int

Definition at line 332 of file [DigitizerObjectGeneric.cpp](#).

#### 5.9.3.8 int DigitizerObjectGeneric::DigitizerObjectGenericSetDecimationFactor ( )

The DigitizerObjectGenericSetDecimationFactor method sets the decimation factor.

This method produces effects only for 740 series.

##### Returns

int

Definition at line 148 of file [DigitizerObjectGeneric.cpp](#).

#### 5.9.3.9 int DigitizerObjectGeneric::DigitizerObjectGenericSetDesMode ( )

The DigitizerObjectGenericSetDesMode method enables the dual edge sampling (DES) mode (for 731 and 751 series only).

##### Returns

int

Definition at line 160 of file [DigitizerObjectGeneric.cpp](#).

#### 5.9.3.10 int DigitizerObjectGeneric::DigitizerObjectGenericSetDRS4SamplingFrequency ( )

The DigitizerObjectGenericSetDRS4SamplingFrequency method sets the sampling frequency of the digitizer.

##### Returns

int

Definition at line 269 of file [DigitizerObjectGeneric.cpp](#).

#### 5.9.3.11 int DigitizerObjectGeneric::DigitizerObjectGenericSetEnableMask ( )

The DigitizerObjectGenericSetEnableMask method sets the groups or the channels (it depends on the model family) able to receive input in accordance with the config file settings.

##### Returns

int

Definition at line 214 of file [DigitizerObjectGeneric.cpp](#).

#### 5.9.3.12 int DigitizerObjectGeneric::DigitizerObjectGenericSetExtTriggerInputMode ( )

The DigitizerObjectGenericSetExtTriggerInputMode method sets how an external trigger has to be used.

##### Returns

int

Definition at line 239 of file [DigitizerObjectGeneric.cpp](#).

#### 5.9.3.13 int DigitizerObjectGeneric::DigitizerObjectGenericSetFastTriggerDigitizing ( )

The DigitizerObjectGenericSetFastTriggerDigitizing method sets the TRn input channel (only for the x742 family) in accordance with the config file settings.

##### Returns

int

Definition at line 487 of file [DigitizerObjectGeneric.cpp](#).

#### 5.9.3.14 int DigitizerObjectGeneric::DigitizerObjectGenericSetGroupDCOffset ( )

The DigitizerObjectGenericSetGroupDCOffset method sets the DCOffset of the groups in accordance with the config file settings.

##### Returns

int

#### 5.9.3.15 int DigitizerObjectGeneric::DigitizerObjectGenericSetIOLevel ( )

The DigitizerObjectGenericSetIOLevel method sets the I/O level in accordance with the config file settings.

##### Returns

int

Definition at line 311 of file [DigitizerObjectGeneric.cpp](#).

#### 5.9.3.16 int DigitizerObjectGeneric::DigitizerObjectGenericSetMaxNumEventsBLT ( )

The DigitizerObjectGenericSetMaxNumEventsBLT method sets the maximum number of events that can be transferred in a readout cycle.

##### Returns

int

Definition at line 204 of file [DigitizerObjectGeneric.cpp](#).

#### 5.9.3.17 int DigitizerObjectGeneric::DigitizerObjectGenericSetPostTriggerSize ( )

The DigitizerObjectGenericSetPostTriggerSize method sets the position of the trigger within the acquisition window in accordance with the config file settings.

##### Returns

int

Definition at line 299 of file [DigitizerObjectGeneric.cpp](#).

#### 5.9.3.18 int DigitizerObjectGeneric::DigitizerObjectGenericSetRecordLength ( )

The DigitizerObjectGenericSetRecordLength method sets the size of the acquisition window.

##### Returns

int

Definition at line 195 of file [DigitizerObjectGeneric.cpp](#).

#### 5.9.3.19 int DigitizerObjectGeneric::DigitizerObjectGenericSetSelfTrigger ( )

The DigitizerObjectGenericSetChannelSelfTrigger method sets the groups or the channels (it depends on the model family) self trigger.

##### Returns

int

Definition at line 369 of file [DigitizerObjectGeneric.cpp](#).

#### 5.9.3.20 int DigitizerObjectGeneric::DigitizerObjectGenericSetSWTriggerMode ( )

The DigitizerObjectGenericSetSWTriggerMode method sets how a software trigger has to be used.

##### Returns

int

#### 5.9.3.21 int DigitizerObjectGeneric::DigitizerObjectGenericSetTestPattern ( )

The DigitizerObjectGenericSetTestPattern method sets the waveform test bit for debugging.

##### Returns

int

Definition at line 183 of file [DigitizerObjectGeneric.cpp](#).

#### 5.9.3.22 int DigitizerObjectGeneric::DigitizerObjectGenericSetTriggerPolarity ( )

The DigitizerObjectGenericSetTriggerPolarity method sets the trigger polarity of a specified channel in accordance with the config file settings.

##### Returns

int

## 5.9.3.23 int DigitizerObjectGeneric::DigitizerObjectGenericWriteRegister ( )

The DigitizerObjectWriteRegister method writes a digitizer register in accordance with the config file settings.

## Returns

int

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

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

## 5.10 DigitizerStateMachine Class Reference

```
#include <DigitizerStateMachine.h>
```

### Public Member Functions

- void [DigitizerStateMachineSetup](#) (const char \*conf\_file)  
The DigitizerStateMachineSetup method sets the digitizer with the settings specified in the configuration file.
- void [DigitizerStateMachineStartReading](#) ()  
The DigitizerStateMachineStartReading method begins the data acquisition from the digitizer.
- void [DigitizerStateMachinePrintStatus](#) ()  
The DigitizerStateMachinePrintStatus method prints some informations about the threads of the object.
- void [DigitizerStateMachineStopReading](#) ()  
The DigitizerStateMachineStopReading method stops the data acquisition from the digitizer.
- void [DigitizerStateMachineRawDataInit](#) ()  
The DigitizerStateMachineRawDataInit method creates the Produttore and the Consumatore threads.
- void [DigitizerStateMachineQuit](#) ()  
The DigitizerStateMachineQuit method closes the digitizer.
- void [DigitizerStateMachineSendSWTrigger](#) ()  
The DigitizerStateMachineSendSWTrigger method sends one software trigger to the digitizer.
- void [DigitizerStateStartPreprocessing](#) ()  
The DigitizerStartPreprocessing method starts the preprocessing actions.
- void [DigitizerStateStartVisualization](#) ()  
The DigitizerStartVisualization method starts the visualization actions.
- void [DigitizerStateStopPreprocessing](#) ()  
The DigitizerStopPreprocessing method starts the preprocessing actions.
- void [DigitizerStateStartRawDataWriting](#) ()  
The DigitizerStartRawDataWriting method starts the visualization actions.
- void [DigitizerStateStopRawDataWriting](#) ()  
The DigitizerStopRawDataWriting method starts the preprocessing actions.
- void [DigitizerStateStopVisualization](#) ()  
The DigitizerStopVisualization method starts the visualization actions.
- void [Produttore](#) ()  
The Produttore thread reads data from the digitizer and puts them in the buffers.
- void [Consumatore\\_Dispatcher](#) ()  
The Consumatore thread picks data from the circular\_buffer\_raw\_data[RAWDATAQUEUE] and writes them into the hard disk.
- void [Preprocessing](#) ()

The Preprocessing thread picks data from the `circular_buffer_preprocessing[PREPROCESSINGQUEUE]` and performs the preprocessing data actions.

- void [Visualization](#) ()

The Visualization thread picks data from the `circular_buffer_visualization[VISUALIZATIONQUEUE]` and visualizes them using the `gnuplot` program.

- void [DigitizerStateMachineInit](#) (const char \*conf\_file)

The [DigitizerStateMachineInit\(const char \\*\)](#) configures the digitizer.

- [DigitizerStateMachine](#) ()

The [DigitizerStateMachine\(const char \\*\)](#) constructor sets `go_general` variable to zero.

## Public Attributes

- int [num\\_mex\\_raw\\_data](#)

The `num_mex_raw_data` variable represents the number of messages stored in the `circular_buffer_raw_data[RAW-DATAQUEUE]` attribute.

- int [num\\_mex\\_preprocessing](#)

The `num_mex_preprocessing` variable represents the number of messages stored in the `circular_buffer_preprocessing[PREPROCESSINGQUEUE]` attribute.

- int [num\\_mex\\_visualization](#)

The `num_mex_visualization` variable represents the number of messages stored in the `circular_buffer_visualization[VISUALIZATIONQUEUE]` attribute.

- int [testa\\_raw\\_data](#)

The `testa_raw_data` variable is the pointer to the head of the `circular_buffer_raw_data[RAWDATAQUEUE]` attribute.

- int [testa\\_preprocessing](#)

The `testa_preprocessing` variable is the pointer to the head of the `circular_buffer_preprocessing[PREPROCESSING-QUEUE]` attribute.

- int [testa\\_visualization](#)

The `testa_visualization` variable is the pointer to the head of the `circular_buffer_visualization[VISUALIZATIONQUEUE]` attribute.

- int [coda\\_raw\\_data](#)

The `coda_raw_data` variable is the pointer to the tail of the `circular_buffer_raw_data[RAWDATAQUEUE]` attribute.

- int [coda\\_preprocessing](#)

The `coda_preprocessing` variable is the pointer to the tail of the `circular_buffer_preprocessing[PREPROCESSING-QUEUE]` attribute.

- int [coda\\_visualization](#)

The `coda_visualization` variable is the pointer to the tail of the `circular_buffer_visualization[VISUALIZATIONQUEUE]` attribute.

- int [go\\_general](#)

The `go_general` variable controls the main cycle of the threads.

- int [go\\_preprocessing](#)

The `go_preprocessing` variable controls the main cycle of the threads.

- int [go\\_raw\\_data](#)

The `go_raw_data` variable controls the main cycle of the threads.

- int [go\\_visualization](#)

The `go_visualization` variable controls the main cycle of the threads.

- [DigitizerObjectGeneric](#) digitizer

The `digitizer` variable represents the digitizer.

- [RawData](#) `circular_buffer_raw_data` [10]

The `circular_buffer_raw_data[RAWDATAQUEUE]` variable is used to temporary store the data read from the digitizer.

- [RawData](#) `circular_buffer_preprocessing` [10]

The `circular_buffer_preprocessing[PREPROCESSINGQUEUE]` variable is used to temporary store the data read from the digitizer.

- [RawData circular\\_buffer\\_visualization](#) [10]

*The circular\_buffer\_visualization[VISUALIZATIONQUEUE] variable is used to temporary store the data read from the digitizer.*

- thread \* [produttore\\_thread](#)

*The thread id of the Produttore thread.*

- thread \* [consumatore\\_thread](#)

*The thread id of the Consumatore thread.*

- thread \* [preprocessing\\_thread](#)

*The thread id of the Preprocessing thread.*

- thread \* [visualization\\_thread](#)

*The thread id of the Visualization thread.*

- mutex [ReservedConsumerDispatcherInputArea](#)

*A mutex used to implement producer-consumer model.*

- condition\_variable [BlockedProducerInput](#)

*A cond variable used to implement producer-consumer model.*

- condition\_variable [BlockedConsumerInput](#)

*A cond variable used to implement producer-consumer model.*

- mutex [ReservedPreprocessingInputArea](#)

*A mutex used to implement producer-consumer model.*

- mutex [ReservedVisualizationInputArea](#)

*A mutex used to implement producer-consumer model.*

- condition\_variable [Acquisition\\_Cond1](#)

*A cond variable used to implement producer-consumer model.*

- condition\_variable [Acquisition\\_Cond2](#)

*A cond variable used to implement producer-consumer model.*

- condition\_variable [raw\\_cond](#)

*When the rawdata thread is not active, it is suspended in the raw\_cond condition variable.*

- condition\_variable [pre\\_cond](#)

*When the preprocessing thread is not active, it is suspended in the pre\_cond condition variable.*

- condition\_variable [vis\\_cond](#)

*When the visualization thread is not active, it is suspended in the vis\_cond condition variable.*

- int [imstarted](#)

*The imstarted variable indicates if the user has already used "start" command.*

- int [imset](#)

*The imset variable indicates if the user has already init the object with [DigitizerStateMachineInit\(const char \\*conf\\_file\)](#) method.*

### 5.10.1 Detailed Description

Definition at line 22 of file [DigitizerStateMachine.h](#).

### 5.10.2 Constructor & Destructor Documentation

#### 5.10.2.1 DigitizerStateMachine::DigitizerStateMachine ( )

The [DigitizerStateMachine\(const char \\*\)](#) constructor sets go\_general variable to zero.

The constructor is private in order to implement singleton design pattern.

Definition at line 359 of file [DigitizerStateMachine.cpp](#).

### 5.10.3 Member Function Documentation

#### 5.10.3.1 void DigitizerStateMachine::Consumatore\_Dispatcher ( )

The Consumatore thread picks data from the circular\_buffer\_raw\_data[RAWDATAQUEUE] and writes them into the hard disk.

##### Returns

void

Definition at line 154 of file [DigitizerStateMachine.cpp](#).

#### 5.10.3.2 void DigitizerStateMachine::DigitizerStateMachinelnit ( const char \* *conf\_file* )

The [DigitizerStateMachinelnit\(const char \\*\)](#) configures the digitizer.

##### Returns

void.

##### Parameters

<i>conf_file</i>	is the path of the configuration file.
------------------	--

Definition at line 370 of file [DigitizerStateMachine.cpp](#).

#### 5.10.3.3 void DigitizerStateMachine::DigitizerStateMachinePrintStatus ( )

The DigitizerStateMachinePrintStatus method prints some informations about the threads of the object.

##### Returns

void

Definition at line 606 of file [DigitizerStateMachine.cpp](#).

#### 5.10.3.4 void DigitizerStateMachine::DigitizerStateMachineQuit ( )

The DigitizerStateMachineQuit method closes the digitizer.

##### Returns

void

Definition at line 482 of file [DigitizerStateMachine.cpp](#).

#### 5.10.3.5 void DigitizerStateMachine::DigitizerStateMachineRawDataInit ( )

The DigitizerStateMachineRawDataInit method creates the Produttore and the Consumatore threads.

Il metodo DigitizerStateMachineRawDataInit fa riferimento ai parametri RAWDATAQUEUE, PREPROCESSINGQUEUE e VISUALIZATIONQUEUE definiti in [DefineGeneral.h](#).

It also: initializes the mutexes and the cond variables; sets to 0 num\_mex, testa\_raw\_data, coda\_raw\_data and sets to 1 the go\_general variable.



## Returns

void

Si consiglia di non superare il valore di 100 per non intasare le risorse del calcolatore.

Definition at line 411 of file [DigitizerStateMachine.cpp](#).

#### 5.10.3.6 void DigitizerStateMachine::DigitizerStateMachineSendSWTrigger ( )

The DigitizerStateMachineSendSWTrigger method sends one software trigger to the digitizer.

## Returns

void

Definition at line 526 of file [DigitizerStateMachine.cpp](#).

#### 5.10.3.7 void DigitizerStateMachine::DigitizerStateMachineSetup ( const char \* *conf\_file* )

The DigitizerStateMachineSetup method sets the digitizer with the settings specified in the configuration file.

Il metodo DigitizerStateMachineSetup e' fondamentale perche' esegue tutte le funzioni per impostare il digitizer.

## Parameters

<i>conf_file</i>	is the path of the configuration file
------------------	---------------------------------------

## Returns

void

Definition at line 385 of file [DigitizerStateMachine.cpp](#).

#### 5.10.3.8 void DigitizerStateMachine::DigitizerStateMachineStartReading ( )

The DigitizerStateMachineStartReading method begins the data acquisition from the digitizer.

## Returns

void

Definition at line 455 of file [DigitizerStateMachine.cpp](#).

#### 5.10.3.9 void DigitizerStateMachine::DigitizerStateMachineStopReading ( )

The DigitizerStateMachineStopReading method stops the data acquisition from the digitizer.

## Returns

void

Definition at line 494 of file [DigitizerStateMachine.cpp](#).

#### 5.10.3.10 void DigitizerStateMachine::DigitizerStateStartPreprocessing ( )

The DigitizerStartPreprocessing method starts the preprocessing actions.

##### Returns

void

Definition at line 537 of file [DigitizerStateMachine.cpp](#).

#### 5.10.3.11 void DigitizerStateMachine::DigitizerStateStartRawDataWriting ( )

The DigitizerStartRawDataWriting method starts the visualization actions.

##### Returns

void

Definition at line 572 of file [DigitizerStateMachine.cpp](#).

#### 5.10.3.12 void DigitizerStateMachine::DigitizerStateStartVisualization ( )

The DigitizerStartVisualization method starts the visualization actions.

##### Returns

void

Definition at line 549 of file [DigitizerStateMachine.cpp](#).

#### 5.10.3.13 void DigitizerStateMachine::DigitizerStateStopPreprocessing ( )

The DigitizerStopPreprocessing method starts the preprocessing actions.

##### Returns

void

Definition at line 561 of file [DigitizerStateMachine.cpp](#).

#### 5.10.3.14 void DigitizerStateMachine::DigitizerStateStopRawDataWriting ( )

The DigitizerStopRawDataWriting method starts the preprocessing actions.

##### Returns

void

Definition at line 584 of file [DigitizerStateMachine.cpp](#).

#### 5.10.3.15 void DigitizerStateMachine::DigitizerStateStopVisualization ( )

The DigitizerStopVisualization method starts the visualization actions.

##### Returns

void

Definition at line 595 of file [DigitizerStateMachine.cpp](#).

#### 5.10.3.16 void DigitizerStateMachine::Preprocessing ( )

The Preprocessing thread picks data from the `circular_buffer_preprocessing[PREPROCESSINGQUEUE]` and performs the preprocessing data actions.

##### Returns

void

Definition at line 219 of file [DigitizerStateMachine.cpp](#).

#### 5.10.3.17 void DigitizerStateMachine::Produttore ( )

The Produttore thread reads data from the digitizer and puts them in the buffers.

##### Returns

void

Definition at line 29 of file [DigitizerStateMachine.cpp](#).

#### 5.10.3.18 void DigitizerStateMachine::Visualization ( )

The Visualization thread picks data from the `circular_buffer_visualization[VISUALIZATIONQUEUE]` and visualizes them using the gnuplot program.

##### Returns

void

Definition at line 281 of file [DigitizerStateMachine.cpp](#).

### 5.10.4 Member Data Documentation

#### 5.10.4.1 condition\_variable DigitizerStateMachine::Acquisition\_Cond1

A cond variable used to implement producer-consumer model.

Definition at line 267 of file [DigitizerStateMachine.h](#).

#### 5.10.4.2 condition\_variable DigitizerStateMachine::Acquisition\_Cond2

A cond variable used to implement producer-consumer model.

Definition at line 272 of file [DigitizerStateMachine.h](#).

#### 5.10.4.3 condition\_variable DigitizerStateMachine::BlockedConsumerInput

A cond variable used to implement producer-consumer model.

Definition at line 252 of file [DigitizerStateMachine.h](#).

#### 5.10.4.4 condition\_variable DigitizerStateMachine::BlockedProducerInput

A cond variable used to implement producer-consumer model.

Definition at line 247 of file [DigitizerStateMachine.h](#).

#### 5.10.4.5 RawData DigitizerStateMachine::circular\_buffer\_preprocessing[10]

The circular\_buffer\_preprocessing[PREPROCESSINGQUEUE] variable is used to temporary store the data read from the digitizer.

The data are put in the circular\_buffer\_preprocessing[PREPROCESSINGQUEUE] by Produttore thread and they are picked up by Preprocessing thread.

Definition at line 107 of file [DigitizerStateMachine.h](#).

#### 5.10.4.6 RawData DigitizerStateMachine::circular\_buffer\_raw\_data[10]

The circular\_buffer\_raw\_data[RAWDATAQUEUE] variable is used to temporary store the data read from the digitizer.

The data are put in the circular\_buffer\_raw\_data[RAWDATAQUEUE] by Produttore thread and they are picked up by Consumatore thread.

Definition at line 101 of file [DigitizerStateMachine.h](#).

#### 5.10.4.7 RawData DigitizerStateMachine::circular\_buffer\_visualization[10]

The circular\_buffer\_visualization[VISUALIZATIONQUEUE] variable is used to temporary store the data read from the digitizer.

The data are put in the circular\_buffer\_visualization[VISUALIZATIONQUEUE] by Produttore thread and they are picked up by Visualization thread.

Definition at line 113 of file [DigitizerStateMachine.h](#).

#### 5.10.4.8 int DigitizerStateMachine::coda\_preprocessing

The coda\_preprocessing variable is the pointer to the tail of the circular\_buffer\_preprocessing[PREPROCESSING-QUEUE] attribute.

Definition at line 65 of file [DigitizerStateMachine.h](#).

#### 5.10.4.9 int DigitizerStateMachine::coda\_raw\_data

The coda\_raw\_data variable is the pointer to the tail of the circular\_buffer\_raw\_data[RAWDATAQUEUE] attribute.

Definition at line 60 of file [DigitizerStateMachine.h](#).

#### 5.10.4.10 int DigitizerStateMachine::coda\_visualization

The coda\_visualization variable is the pointer to the tail of the circular\_buffer\_visualization[VISUALIZATIONQUEUE] attribute.

Definition at line 70 of file [DigitizerStateMachine.h](#).

#### 5.10.4.11 thread\* DigitizerStateMachine::consumatore\_thread

The thread id of the Consumatore thread.

Definition at line 227 of file [DigitizerStateMachine.h](#).

#### 5.10.4.12 DigitizerObjectGeneric DigitizerStateMachine::digitizer

The digitizer variable represents the digitizer.

Definition at line 95 of file [DigitizerStateMachine.h](#).

#### 5.10.4.13 int DigitizerStateMachine::go\_general

The go\_general variable controls the main cycle of the threads.

Definition at line 75 of file [DigitizerStateMachine.h](#).

#### 5.10.4.14 int DigitizerStateMachine::go\_preprocessing

The go\_preprocessing variable controls the main cycle of the threads.

Definition at line 80 of file [DigitizerStateMachine.h](#).

#### 5.10.4.15 int DigitizerStateMachine::go\_raw\_data

The go\_raw\_data variable controls the main cycle of the threads.

Definition at line 85 of file [DigitizerStateMachine.h](#).

#### 5.10.4.16 int DigitizerStateMachine::go\_visualization

The go\_visualization variable controls the main cycle of the threads.

Definition at line 90 of file [DigitizerStateMachine.h](#).

#### 5.10.4.17 int DigitizerStateMachine::imset

The imset variable indicates if the user has already init the object with [DigitizerStateMachineInit\(const char \\*conf\\_file\)](#) method.

Definition at line 304 of file [DigitizerStateMachine.h](#).

#### 5.10.4.18 int DigitizerStateMachine::imstarted

The imstarted variable indicates if the user has already used "start" command.

Definition at line 299 of file [DigitizerStateMachine.h](#).

#### 5.10.4.19 int DigitizerStateMachine::num\_mex\_preprocessing

The num\_mex\_preprocessing variable represents the number of messages stored in the circular\_buffer\_preprocessing[PREPROCESSINGQUEUE] attribute.

Definition at line 35 of file [DigitizerStateMachine.h](#).

#### 5.10.4.20 int DigitizerStateMachine::num\_mex\_raw\_data

The num\_mex\_raw\_data variable represents the number of messages stored in the circular\_buffer\_raw\_data[RAWDATAQUEUE] attribute.

Definition at line 30 of file [DigitizerStateMachine.h](#).

#### 5.10.4.21 `int DigitizerStateMachine::num_mex_visualization`

The `num_mex_visualization` variable represents the number of messages stored in the `circular_buffer_visualization[VISUALIZATIONQUEUE]` attribute.

Definition at line 40 of file [DigitizerStateMachine.h](#).

#### 5.10.4.22 `condition_variable DigitizerStateMachine::pre_cond`

When the preprocessing thread is not active, it is suspended in the `pre_cond` condition variable.

Definition at line 282 of file [DigitizerStateMachine.h](#).

#### 5.10.4.23 `thread* DigitizerStateMachine::preprocessing_thread`

The thread id of the Preprocessing thread.

Definition at line 232 of file [DigitizerStateMachine.h](#).

#### 5.10.4.24 `thread* DigitizerStateMachine::produttore_thread`

The thread id of the Produttore thread.

Definition at line 222 of file [DigitizerStateMachine.h](#).

#### 5.10.4.25 `condition_variable DigitizerStateMachine::raw_cond`

When the rawdata thread is not active, it is suspended in the `raw_cond` condition variable.

Definition at line 277 of file [DigitizerStateMachine.h](#).

#### 5.10.4.26 `mutex DigitizerStateMachine::ReservedConsumerDispatcherInputArea`

A mutex used to implement producer-consumer model.

Definition at line 242 of file [DigitizerStateMachine.h](#).

#### 5.10.4.27 `mutex DigitizerStateMachine::ReservedPreprocessingInputArea`

A mutex used to implement producer-consumer model.

Definition at line 257 of file [DigitizerStateMachine.h](#).

#### 5.10.4.28 `mutex DigitizerStateMachine::ReservedVisualizationInputArea`

A mutex used to implement producer-consumer model.

Definition at line 262 of file [DigitizerStateMachine.h](#).

#### 5.10.4.29 `int DigitizerStateMachine::testa_preprocessing`

The `testa_preprocessing` variable is the pointer to the head of the `circular_buffer_preprocessing[PREPROCESSINGQUEUE]` attribute.

Definition at line 50 of file [DigitizerStateMachine.h](#).

## 5.10.4.30 int DigitizerStateMachine::testa\_raw\_data

The testa\_raw\_data variable is the pointer to the head of the circular\_buffer\_raw\_data[RAWDATAQUEUE] attribute.  
Definition at line 45 of file [DigitizerStateMachine.h](#).

## 5.10.4.31 int DigitizerStateMachine::testa\_visualization

The testa\_visualization variable is the pointer to the head of the circular\_buffer\_visualization[VISUALIZATIONQUEUE] attribute.  
Definition at line 55 of file [DigitizerStateMachine.h](#).

## 5.10.4.32 condition\_variable DigitizerStateMachine::vis\_cond

When the visualization thread is not active, it is suspended in the vis\_cond condition variable.  
Definition at line 287 of file [DigitizerStateMachine.h](#).

## 5.10.4.33 thread\* DigitizerStateMachine::visualization\_thread

The thread id of the Visualization thread.  
Definition at line 237 of file [DigitizerStateMachine.h](#).  
The documentation for this class was generated from the following files:

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

## 5.11 FastObject Class Reference

The [FastObject](#) class picks up all the settings taken from the configuration file regarding a TRn input channel of the digitizer (only for x742 series).

```
#include <ConfObject.h>
```

### Public Member Functions

- [FastObject](#) ()  
*The [FastObject\(\)](#) constructor sets all the values of the fast\_object attributes to -1.*
- void [PrintFast](#) ()  
*The [PrintFast\(\)](#) method prints all the fast\_object attributes.*

### Public Attributes

- int [set](#)  
*The set variable indicates if a TRn has been set or not.*
- int [numFast](#)  
*The numFast variable indicates the number of the TRn.*
- int [dc\\_offset](#)  
*The variable dc\_offset indicates the dc\_offset being applicated to the TRn.*
- int [trigger\\_threshold](#)  
*The variable trigger\_threshold contains the threshold for the TRn auto trigger.*

### 5.11.1 Detailed Description

The [FastObject](#) class picks up all the settings taken from the configuration file regarding a TRn input channel of the digitizer (only for x742 series).

The number of the channel is indicated by numFast variable. If the channel has not been set, the set variable is set to -1 like the other variables.

#### Author

Daniele Berto

Definition at line 130 of file [ConfObject.h](#).

### 5.11.2 Constructor & Destructor Documentation

#### 5.11.2.1 FastObject::FastObject ( )

The [FastObject\(\)](#) constructor sets all the values of the fast\_object attributes to -1.

Definition at line 78 of file [ConfObject.cpp](#).

### 5.11.3 Member Function Documentation

#### 5.11.3.1 void FastObject::PrintFast ( )

The [PrintFast\(\)](#) method prints all the fast\_object attributes.

#### Returns

void

Definition at line 124 of file [ConfObject.cpp](#).

### 5.11.4 Member Data Documentation

#### 5.11.4.1 int FastObject::dc\_offset

The variable dc\_offset indicates the dc\_offset being applied to the TRn.

Definition at line 147 of file [ConfObject.h](#).

#### 5.11.4.2 int FastObject::numFast

The numFast variable indicates the number of the TRn.

Definition at line 142 of file [ConfObject.h](#).

#### 5.11.4.3 int FastObject::set

The set variable indicates if a TRn has been set or not.

Definition at line 137 of file [ConfObject.h](#).



#### 5.11.4.4 int FastObject::trigger\_threshold

The variable `trigger_threshold` contains the threshold for the TRn auto trigger.

Definition at line 152 of file [ConfObject.h](#).

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

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

## 5.12 GroupObject Class Reference

The [GroupObject](#) class picks up all the settings taken from the configuration file regarding one group of the digitizer.

```
#include <ConfObject.h>
```

### Public Member Functions

- [GroupObject](#) ()  
*The [GroupObject\(\)](#) constructor sets all the values of the group\_object attributes to -1.*
- void [PrintGroup](#) ()  
*The [PrintGroup\(\)](#) method prints all the group\_object attributes.*

### Public Attributes

- int [set](#)  
*The set variable indicates if a group has been set or not.*
- int [numGroup](#)  
*The numGroup variable indicates the number of the group.*
- int [enable\\_input](#)  
*The enable\_input variable indicates if a group can receive input or not.*
- int [dc\\_offset](#)  
*The variable dc\_offset indicates the dc\_offset being applicated to the group.*
- int [trigger\\_threshold](#)  
*The variable trigger\_threshold contains the threshold for the group auto trigger.*
- int [group\\_trg\\_enable\\_mask](#)  
*this option is used only for the Models x740.*

### 5.12.1 Detailed Description

The [GroupObject](#) class picks up all the settings taken from the configuration file regarding one group of the digitizer.

The number of the group is indicated by numGroup variable. If the group has not been set, the set variable is set to -1 like the other variables.

#### Author

Daniele Berto

Definition at line 73 of file [ConfObject.h](#).

## 5.12.2 Constructor & Destructor Documentation

### 5.12.2.1 GroupObject::GroupObject ( )

The [GroupObject\(\)](#) constructor sets all the values of the group\_object attributes to -1.

Definition at line 67 of file [ConfObject.cpp](#).

## 5.12.3 Member Function Documentation

### 5.12.3.1 void GroupObject::PrintGroup ( )

The [PrintGroup\(\)](#) method prints all the group\_object attributes.

Returns

void

Definition at line 106 of file [ConfObject.cpp](#).

## 5.12.4 Member Data Documentation

### 5.12.4.1 int GroupObject::dc\_offset

The variable dc\_offset indicates the dc\_offset being applicated to the group.

Definition at line 95 of file [ConfObject.h](#).

### 5.12.4.2 int GroupObject::enable\_input

The enable\_input variable indicates if a group can receive input or not.

Definition at line 90 of file [ConfObject.h](#).

### 5.12.4.3 int GroupObject::group\_trg\_enable\_mask

this option is used only for the Models x740.

These models have the channels grouped 8 by 8; one group of 8 channels has a common trigger that is generated as the OR of the self trigger of the channels in the group that are enabled by this mask.

Definition at line 108 of file [ConfObject.h](#).

### 5.12.4.4 int GroupObject::numGroup

The numGroup variable indicates the number of the group.

Definition at line 85 of file [ConfObject.h](#).

### 5.12.4.5 int GroupObject::set

The set variable indicates if a group has been set or not.

Definition at line 80 of file [ConfObject.h](#).

#### 5.12.4.6 int GroupObject::trigger\_threshold

The variable trigger\_threshold contains the threshold for the group auto trigger.

Definition at line 100 of file [ConfObject.h](#).

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

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

## 5.13 Input Class Reference

```
#include <Input.h>
```

### Public Member Functions

- void [GetInput](#) (char \*input\_buffer\_sending)  
*The GetInput method copies the input from the stdin to the parameter input\_buffer\_sending.*
- void [Finish](#) ()  
*The Finish method shuts down the object threads.*
- void [Producer](#) ()  
*The Producer method gets the input from the stdin and puts it in the input\_buffer attribute.*
- [Input](#) ()  
*The Input constructor sets num\_mex to 0 and go to 1.*

### Public Attributes

- char \* [input\\_buffer](#)  
*The input\_buffer variable contains the informations shared by the threads.*
- thread \* [producer\\_thread](#)  
*The producer\_thread gets the input from the stdin and puts it in the input\_buffer attribute.*
- mutex [mutex1](#)  
*The mutex1 mutex guarantees the consistency of the informations retrieved by the get\_input method.*
- int [go](#)  
*The go variable controls the threads.*
- int [num\\_mex](#)  
*The num\_mex variable indicates if a message is ready to be picked by the get\_input method.*

#### 5.13.1 Detailed Description

Definition at line 17 of file [Input.h](#).

#### 5.13.2 Constructor & Destructor Documentation

##### 5.13.2.1 Input::Input ( )

The [Input](#) constructor sets num\_mex to 0 and go to 1.

It also initializes input\_buffer and creates the producer thread.

Definition at line 32 of file [Input.cpp](#).

### 5.13.3 Member Function Documentation

#### 5.13.3.1 void Input::Finish ( )

The Finish method shuts down the object threads.

##### Returns

void

Definition at line 109 of file [Input.cpp](#).

#### 5.13.3.2 void Input::GetInput ( char \* *input\_buffer\_sending* )

The GetInput method copies the input from the stdin to the parameter `input_buffer_sending`.

The caller of this method gets the user input: the method copies it in the parameter.

##### Parameters

<i>input_buffer_sending</i>	is where the informations are copied
-----------------------------	--------------------------------------

##### Returns

void

Definition at line 86 of file [Input.cpp](#).

#### 5.13.3.3 void Input::Producer ( )

The Producer method gets the input from the stdin and puts it in the `input_buffer` attribute.

##### Returns

void

Definition at line 44 of file [Input.cpp](#).

### 5.13.4 Member Data Documentation

#### 5.13.4.1 int Input::go

The go variable controls the threads.

Definition at line 53 of file [Input.h](#).

#### 5.13.4.2 char\* Input::input\_buffer

The `input_buffer` variable contains the informations shared by the threads.

Definition at line 24 of file [Input.h](#).

#### 5.13.4.3 mutex Input::mutex1

The `mutex1` mutex guarantees the consistency of the informations retrieved by the `get_input` method.

Definition at line 48 of file [Input.h](#).

## 5.13.4.4 int Input::num\_mex

The num\_mex variable indicates if a message is ready to be picked by the get\_input method.

Definition at line 58 of file [Input.h](#).

## 5.13.4.5 thread\* Input::producer\_thread

The producer\_thread gets the input from the stdin and puts it in the input\_buffer attribute.

Definition at line 43 of file [Input.h](#).

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

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

## 5.14 LogFile Class Reference

The [LogFile](#) singleton is used to write the error codes retrieved by the CAENDigitizer function to the logfile indicates by the log\_file\_path attribute.

```
#include <LogFile.h>
```

### Public Member Functions

- void [LogFileSet](#) (const char \*log\_file\_arg)  
*The LogFileSet method copies the log\_file\_arg parameter to the log\_file\_path and opens it using the log\_file\_punt file pointer.*
- void [LogFileWrite](#) (CAEN\_DGTZ\_ErrorCode ret\_arg, const char \*file, const char \*func, int line)  
*The LogFileWrite method prints the meaning of the CAEN\_DGTZ\_ErrorCode ret\_arg to the log file.*
- void [LogFileWriteString](#) (const char \*string)  
*The LogFileWriteString method prints the string parameter to the log file.*
- void [LogFileRead](#) ()  
*The LogFileRead method prints the content of the log file.*
- void [LogFileMessageOn](#) ()  
*The LogFileMessageOn method sets to 1 the flag attribute allowing the object to print the message "Some error occurred".*
- void [LogFileMessageOff](#) ()  
*The LogFileMessageOff method sets to 0 the flag attribute disallowing the object to print the message "Some error occurred".*

### Static Public Member Functions

- static [LogFile](#) \* [Instance](#) ()  
*The "Instance()" method returns a pointer to the [LogFile](#) instance.*

### Public Attributes

- const char \* [log\\_file\\_path](#)  
*The log\_file\_path stores the path of the log file.*

### 5.14.1 Detailed Description

The [LogFile](#) singleton is used to write the error codes retrieved by the CAENDigitizer function to the logfile indicates by the `log_file_path` attribute.

Using the "more" command, the user prints the content of the log file.

#### Author

Daniele Berto

Definition at line 13 of file [LogFile.h](#).

### 5.14.2 Member Function Documentation

#### 5.14.2.1 `LogFile * LogFile::Instance ( ) [static]`

The "Instance()" method returns a pointer to the [LogFile](#) instance.

Definition at line 24 of file [LogFile.cpp](#).

#### 5.14.2.2 `void LogFile::LogFileMessageOff ( )`

The `LogFileMessageOff` method sets to 0 the flag attribute disallowing the object to print the message "Some error occurred".

Definition at line 43 of file [LogFile.cpp](#).

#### 5.14.2.3 `void LogFile::LogFileMessageOn ( )`

The `LogFileMessageOn` method sets to 1 the flag attribute allowing the object to print the message "Some error occurred".

Definition at line 34 of file [LogFile.cpp](#).

#### 5.14.2.4 `void LogFile::LogFileRead ( )`

The `LogFileRead` method prints the content of the log file.

Definition at line 109 of file [LogFile.cpp](#).

#### 5.14.2.5 `void LogFile::LogFileSet ( const char * log_file_arg )`

The `LogFileSet` method copies the `log_file_arg` parameter to the `log_file_path` and opens it using the `log_file_punt` file pointer.

Definition at line 51 of file [LogFile.cpp](#).

#### 5.14.2.6 `void LogFile::LogFileWrite ( CAEN_DGTZ_ErrorCode ret_arg, const char * file, const char * func, int line )`

The `LogFileWrite` method prints the meaning of the `CAEN_DGTZ_ErrorCode` `ret_arg` to the log file.

#### Parameters

---

<i>ret_arg</i>	is the CAEN_DGTZ_ErrorCode being written on the log file.
<i>file</i>	is the path of the file in which the functions that generated the error code where located.
<i>func</i>	is the name of the function in which the functions that generated the error code where located.
<i>line</i>	is the line number of the functions that generated the error code.

#### Returns

void.

Definition at line 88 of file [LogFile.cpp](#).

#### 5.14.2.7 void LogFile::LogFileWriteString ( const char \* *string* )

The LogFileWriteString method prints the string parameter to the log file.

Definition at line 78 of file [LogFile.cpp](#).

### 5.14.3 Member Data Documentation

#### 5.14.3.1 const char\* LogFile::log\_file\_path

The log\_file\_path stores the path of the log file.

Definition at line 50 of file [LogFile.h](#).

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

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

## 5.15 OutputModule Class Reference

The [OutputModule](#) class provides an useful way to manage the output of the program.

```
#include <OutputModule.h>
```

### Public Member Functions

- void [StdOutInsert](#) (const char \*string)  
*The "StdOutInsert" method copies the parameter "string" to the "buffer" attribute.*
- void [StdOutInsertLex](#) (char \*string, int length)  
*The "StdOutInsertLex" method copies the parameter "string" to the "buffer" attribute.*
- void [StdOutPrint](#) ()  
*The "StdOutPrint" method prints buffer to the stdout.*
- void [OutputModuleStdoutOn](#) ()  
*The "OutputModuleStdoutOn" method enables the object to print "buffer" to the stdout.*
- void [OutputModuleStdoutOff](#) ()  
*The "OutputModuleStdoutOff" method disables the object to print "buffer" to the stdout.*
- void [OutputModuleSockidOn](#) (int sockid)  
*The "OutputModuleSockidOn" method enables the object to send "buffer" to the TCP/IP user identified by the "sockid" parameter.*
- void [OutputModuleSockidOff](#) ()  
*The "OutputModuleSockidOn" method disables the object to send "buffer" via TCP/IP.*

- int [TcpUserArrayInsert](#) (int sockid)  
The "TcpUserArrayInsert" method inserts the parameter "sockid" in the "sockid\_array".
- int [TcpUserArrayDelete](#) (int sockid)  
The "TcpUserArrayDelete" method deletes the parameter "sockid" from the "sockid\_array".
- int [TcpUserArraySendStdOut](#) ()  
The "TcpUserArraySendStdOut" method sends the "buffer" attribute to the TCP/IP user identified by the output\_module\_sockid attribute.
- void [Output](#) (const char \*string)  
The "Output" method prints the "string" parameter to the stdout or send it via TCP/IP in accordance with the "output\_module\_stdout" and the "output\_module\_sockid" (these variable could be modified by the "OutputModuleStdoutOn"/"Off" and "OutputModuleSockidOn"/"Off" methods).
- void [OutputFlex](#) (const char \*string, int length)  
The "OutputFlex" method prints the "string" parameter to the stdout or send it via TCP/IP in accordance with the "output\_module\_stdout" and the "output\_module\_sockid" (these variable could be modified by the "OutputModuleStdoutOn"/"Off" and "OutputModuleSockidOn"/"Off" methods).

## Static Public Member Functions

- static [OutputModule \\* Instance](#) ()  
The "Instance ()" method is used to implement the singleton design pattern.

## Public Attributes

- [ApplicationSetup \\* output\\_module\\_application\\_setup](#)  
The "output\_module\_application\_setup" is a pointer to the [ApplicationSetup](#) singleton.
- char [buffer](#) [1000]  
The "buffer" array stores the output being printed.
- int [sockid\\_array](#) [100]  
The "sockid\_array" array takes note of the users connected to the server via TCP/IP.

### 5.15.1 Detailed Description

The [OutputModule](#) class provides an useful way to manage the output of the program.

The program has to send output not only to stdout but also to the tcp users. The [OutputModule](#) class takes note of the output target with the "OutputModuleStdoutOn/Off" and the "OutputModuleSockidOn/Off" methods. It also takes note of the TCP users active on the server storing their sockid in sockid\_array. It is possible to insert/delete a sockid with "TcpUserArrayInsert"/"Delete" methods.

#### Author

Daniele Berto

Definition at line 17 of file [OutputModule.h](#).

### 5.15.2 Member Function Documentation

#### 5.15.2.1 [OutputModule \\* OutputModule::Instance](#) ( ) [static]

The "Instance ()" method is used to implement the singleton design pattern.

Definition at line 20 of file [OutputModule.cpp](#).



#### 5.15.2.2 void OutputModule::Output ( const char \* *string* )

The "Output" method prints the "string" parameter to the stdout or send it via TCP/IP in accordance with the "output\_module\_stdout" and the "output\_module\_sockid" (these variable could be modified by the "OutputModuleStdoutOn"/"Off" and "OutputModuleSockidOn"/"Off" methods).

Definition at line 129 of file [OutputModule.cpp](#).

#### 5.15.2.3 void OutputModule::OutputFlex ( const char \* *string*, int *length* )

The "OutputFlex" method prints the "string" parameter to the stdout or send it via TCP/IP in accordance with the "output\_module\_stdout" and the "output\_module\_sockid" (these variable could be modified by the "OutputModuleStdoutOn"/"Off" and "OutputModuleSockidOn"/"Off" methods).

The maximum number of characters being displayed is indicated by the "length" parameter. This method is used to send the information read from the configuration file by the lessical scanner.

Definition at line 138 of file [OutputModule.cpp](#).

#### 5.15.2.4 void OutputModule::OutputModuleSockidOff ( )

The "OutputModuleSockidOn" method disables the object to send "buffer" via TCP/IP.

Definition at line 164 of file [OutputModule.cpp](#).

#### 5.15.2.5 void OutputModule::OutputModuleSockidOn ( int *sockid* )

The "OutputModuleSockidOn" method enables the object to send "buffer" to the TCP/IP user identified by the "sockid" parameter.

The "sockid" parameter is copied to the "output\_module\_sockid" attribute.

Definition at line 159 of file [OutputModule.cpp](#).

#### 5.15.2.6 void OutputModule::OutputModuleStdoutOff ( )

The "OutputModuleStdoutOff" method disables the object to print "buffer" to the stdout.

Definition at line 155 of file [OutputModule.cpp](#).

#### 5.15.2.7 void OutputModule::OutputModuleStdoutOn ( )

The "OutputModuleStdoutOn" method enables the object to print "buffer" to the stdout.

Definition at line 151 of file [OutputModule.cpp](#).

#### 5.15.2.8 void OutputModule::StdOutInsert ( const char \* *string* )

The "StdOutInsert" method copies the parameter "string" to the "buffer" attribute.

Definition at line 41 of file [OutputModule.cpp](#).

#### 5.15.2.9 void OutputModule::StdOutInsertLex ( char \* *string*, int *length* )

The "StdOutInsertLex" method copies the parameter "string" to the "buffer" attribute.

Length indicates the dimension of "string".

Definition at line 48 of file [OutputModule.cpp](#).

#### 5.15.2.10 void OutputModule::StdOutPrint ( )

The "StdOutPrint" method prints buffer to the stdout.

Definition at line 55 of file [OutputModule.cpp](#).

#### 5.15.2.11 int OutputModule::TcpUserArrayDelete ( int sockid )

The "TcpUserArrayDelete" method deletes the parameter "sockid" from the "sockid\_array".

Definition at line 92 of file [OutputModule.cpp](#).

#### 5.15.2.12 int OutputModule::TcpUserArrayInsert ( int sockid )

The "TcpUserArrayInsert" method inserts the parameter "sockid" in the "sockid\_array".

Definition at line 75 of file [OutputModule.cpp](#).

#### 5.15.2.13 int OutputModule::TcpUserArraySendStdOut ( )

The "TcpUserArraySendStdOut" method sends the "buffer" attribute to the TCP/IP user identified by the output\_module\_sockid attribute.

Definition at line 108 of file [OutputModule.cpp](#).

### 5.15.3 Member Data Documentation

#### 5.15.3.1 char OutputModule::buffer[1000]

The "buffer" array stores the output being printed.

It is filled by the "Output(const char \* string)" method.

Definition at line 59 of file [OutputModule.h](#).

#### 5.15.3.2 ApplicationSetup\* OutputModule::output\_module\_application\_setup

The "output\_module\_application\_setup" is a pointer to the [ApplicationSetup](#) singleton.

Definition at line 49 of file [OutputModule.h](#).

#### 5.15.3.3 int OutputModule::sockid\_array[100]

The "sockid\_array" array takes note of the users connected to the server via TCP/IP.

Definition at line 100 of file [OutputModule.h](#).

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

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

## 5.16 RawData Class Reference

The [RawData](#) class manages the readout from the digitizer.

```
#include <RawData.h>
```

## Public Member Functions

- [RawData](#) ()  
The [RawData](#) constructor set "imset" to 0.
- void [RawDataSet](#) ([DigitizerObjectGeneric](#) digitizer\_arg)  
The [RawDataSet](#) method copies the parameter handler\_raw to the private attribute handler.
- void [RawDataDel](#) ()  
The [RawDataDel](#) method deallocates buffer attribute.
- void [RawDataRead](#) ()  
The [RawDataRead](#) method reads data from the digitizer and puts it in the buffer variable.
- void [RawDataWriteOnFile](#) (const char \*file\_arg)  
The [RawWriteOnFile](#) method writes the events read from the digitizer into the hard disk.
- void [RawDataWriteOnFile](#) (FILE \*file, FILE \*file\_size)  
The [RawWriteOnFile](#) method writes the events read from the digitizer into the hard disk.
- void [RawDataWriteDecodeEventOnPlotFile](#) (const char \*file\_arg)  
The [RawDataWriteDecodeEventOnPlotFile](#) method decodes the events stored in the buffer attribute and writes the in the file specified by file\_arg.
- void [RawDataPlot](#) (const char \*file\_arg, FILE \*gnuplot)  
The [RawDataPlot](#) method plots the decoded events stored in the file specified by file\_arg.
- void [RawDataDecode](#) ()  
The [RawDataDecode](#) method decodes the events stored in the buffer attributes.
- [RawData](#) & operator= (const [RawData](#) &p)  
This overloading is necessary in order to perform a deep copy of the object.

## Public Attributes

- CAEN\_DGTZ\_ErrorCode [ret](#)  
The [ret](#) variable contains the error code returned by CAEN\_DGTZ library functions.
- CAEN\_DGTZ\_EventInfo\_t [eventInfo](#)  
The [eventInfo](#) variable contains the information about an event.
- [DigitizerErrorObject](#) [ret\\_error](#)  
The [ret\\_error](#) object is used to print the meaning of CAEN\_DGTZ\_ErrorCode.
- int [handle](#)  
The [handle](#) variable represents the device handler of the digitizer.
- int [size](#)  
The [size](#) variable represents the dimension of the readout buffer.
- CAEN\_DGTZ\_ReadMode\_t [Mode](#)  
The [Mode](#) variable represents the mode of readout.
- int [imset](#)  
The [imset](#) variable indicates if the object has been set or not.
- FILE \* [file](#)  
The [file](#) pointer is used to write the raw data to the hard disk.
- int [bsize](#)  
The [bsize](#) variable represents the dimension of the data read from the digitizer.
- char \* [buffer](#)  
The [buffer](#) pointer contains the data read from the digitizer.
- [DigitizerObjectGeneric](#) [digitizer](#)  
The [digitizer](#) object is fundamental to get the informations to decode correctly the events read from the digitizer.

### 5.16.1 Detailed Description

The [RawData](#) class manages the readout from the digitizer.

#### Author

Daniele Berto

Definition at line 16 of file [RawData.h](#).

### 5.16.2 Constructor & Destructor Documentation

#### 5.16.2.1 [RawData::RawData](#) ( )

The [RawData](#) constructor set "imset" to 0.

Definition at line 48 of file [RawData.cpp](#).

### 5.16.3 Member Function Documentation

#### 5.16.3.1 [RawData & RawData::operator=](#) ( const [RawData](#) & *p* )

This overloading is necessary in order to perform a deep copy of the object.

Now the assignment operator copies all the values of a [RawData](#) object, including the buffer attribute (not only the pointer of memory area).

#### Parameters

<i>p</i>	is the <a href="#">RawData</a> being copied.
----------	--

#### Returns

[RawData](#) &

Definition at line 23 of file [RawData.cpp](#).

#### 5.16.3.2 [void RawData::RawDataDecode](#) ( )

The [RawDataDecode](#) method decodes the events stored in the buffer attributes.

#### Returns

void

Definition at line 324 of file [RawData.cpp](#).

#### 5.16.3.3 [void RawData::RawDataDel](#) ( )

The [RawDataDel](#) method deallocates buffer attribute.

The method uses the free function from the stdlib, not the [CAEN\\_DGTZ\\_FreeReadoutBuffer](#) function.

#### Returns

void

Definition at line 67 of file [RawData.cpp](#).

#### 5.16.3.4 void RawData::RawDataPlot ( const char \* *file\_arg*, FILE \* *gnuplot* )

The RawDataPlot method plots the decoded events stored in the file specified by *file\_arg*.

The method calls RawDataWriteDecodeEventOnPlotFile method. The code line "fflush(gnuplot)" is fundamental in order to send data to the gnuplot program in real-time.

##### Parameters

<i>file_arg</i>	is the path of the file being written with the raw data.
<i>gnuplot</i>	is the pipe communicating with the gnuplot program.

##### Returns

void

Definition at line 291 of file [RawData.cpp](#).

#### 5.16.3.5 void RawData::RawDataRead ( )

The RawDataRead method reads data from the digitizer and puts it in the buffer variable.

The method uses CAEN\_DGTZ\_ReadData function from CAENDigitizer library.

##### Returns

void

Definition at line 313 of file [RawData.cpp](#).

#### 5.16.3.6 void RawData::RawDataSet ( DigitizerObjectGeneric *digitizer\_arg* )

The RawDataSet method copies the parameter handler\_raw to the private attribute handler.

It also allocates the buffer public attribute using MallocReadoutBuffer function.

##### Returns

void

Definition at line 56 of file [RawData.cpp](#).

#### 5.16.3.7 void RawData::RawDataWriteDecodeEventOnPlotFile ( const char \* *file\_arg* )

The RawDataWriteDecodeEventOnPlotFile method decodes the events stored in the buffer attribute and writes the in the file specified by *file\_arg*.

The method is called by the RawDataPlot method.

##### Parameters

<i>file_arg</i>	is the path of file being written with the raw data.
-----------------	--

##### Returns

void

Definition at line 111 of file [RawData.cpp](#).

#### 5.16.3.8 void RawData::RawDataWriteOnFile ( const char \* *file\_arg* )

The RawWriteOnFile method writes the events read from the digitizer into the hard disk.

The method doesn't decode the events.

## Parameters

<i>file_arg</i>	is the path of file being written with the raw data.
-----------------	--

## Returns

void

Definition at line 77 of file [RawData.cpp](#).

#### 5.16.3.9 void RawData::RawDataWriteOnFile ( FILE \* *file*, FILE \* *file\_size* )

The RawWriteOnFile method writes the events read from the digitizer into the hard disk.

The method doesn't decode the events.

## Parameters

<i>file</i>	is the pointer to the file being written with the raw data.
<i>file_size</i>	is the pointer to the file being written with the raw data size.

## Returns

void

Definition at line 100 of file [RawData.cpp](#).

### 5.16.4 Member Data Documentation

#### 5.16.4.1 int RawData::bsize

The bsize variable represents the dimension of the data read from the digitizer.

Definition at line 70 of file [RawData.h](#).

#### 5.16.4.2 char\* RawData::buffer

The buffer pointer contains the data read from the digitizer.

Definition at line 75 of file [RawData.h](#).

#### 5.16.4.3 DigitizerObjectGeneric RawData::digitizer

The digitizer object is fundamental to get the informations to decode correctly the events read from the digitizer.

Definition at line 80 of file [RawData.h](#).

#### 5.16.4.4 CAEN\_DGTZ\_EventInfo\_t RawData::eventInfo

The eventInfo variable contains the information about an event.

Definition at line 29 of file [RawData.h](#).

#### 5.16.4.5 FILE\* RawData::file

The file pointer is used to write the raw data to the hard disk.

Definition at line 60 of file [RawData.h](#).

#### 5.16.4.6 int RawData::handle

The handle variable represents the device handler of the digitizer.

Definition at line 39 of file [RawData.h](#).

#### 5.16.4.7 int RawData::imset

The imset variable indicates if the object has been set or not.

Definition at line 55 of file [RawData.h](#).

#### 5.16.4.8 CAEN\_DGTZ\_ReadMode\_t RawData::Mode

The Mode variable represents the mode of readout.

Definition at line 50 of file [RawData.h](#).

#### 5.16.4.9 CAEN\_DGTZ\_ErrorCode RawData::ret

The ret variable contains the error code returned by CAEN\_DGTZ library functions.

Definition at line 24 of file [RawData.h](#).

#### 5.16.4.10 DigitizerErrorObject RawData::ret\_error

The ret\_error object is used to print the meaning of CAEN\_DGTZ\_ErrorCode.

Definition at line 34 of file [RawData.h](#).

#### 5.16.4.11 int RawData::size

The size variable represents the dimension of the readout buffer.

The dimension is determined by CAEN\_DGTZ\_MallocReadoutBuffer function.

Definition at line 45 of file [RawData.h](#).

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

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

## 5.17 TcpUser Class Reference

The [TcpUser](#) class provides an useful way to store data about the users of the server.

```
#include <TcpUser.h>
```

### Public Member Functions

- [TcpUser](#) ()

*The [TcpUser](#) constructor sets to 0 the attribute "command\_sent\_by\_user", to -1 "register\_address" and "register\_data" and to 0 first\_parameter and second\_parameter.*



## Public Attributes

- char [command\\_sent\\_by\\_user](#)  
*The command\_sent\_by\_user variable contains the informations about the command sent by the user.*
- char [first\\_parameter](#) [1000]  
*Some commands need more than one parameter to be executed (ex.*
- char [second\\_parameter](#) [1000]  
*The second\_parameter array stores the second parameter of a composite command.*
- int [register\\_address](#)  
*The register\_address variable stores the address of the register being written or being read.*
- int [register\\_data](#)  
*The register\_data variable stores the data being written in the register indicated by register\_address.*
- int [user\\_sockid](#)  
*The user\_sockid variable contains the sockid of the user who sent the command.*

### 5.17.1 Detailed Description

The [TcpUser](#) class provides an useful way to store data about the users of the server.

#### Author

Daniele Berto

Definition at line 13 of file [TcpUser.h](#).

### 5.17.2 Constructor & Destructor Documentation

#### 5.17.2.1 [TcpUser::TcpUser \( \)](#)

The [TcpUser](#) constructor sets to 0 the attribute "command\_sent\_by\_user", to -1 "register\_address" and "register\_data" and to 0 first\_parameter and second\_parameter.

Definition at line 11 of file [TcpUser.cpp](#).

### 5.17.3 Member Data Documentation

#### 5.17.3.1 [char TcpUser::command\\_sent\\_by\\_user](#)

The command\_sent\_by\_user variable contains the informations about the command sent by the user.

See "DefineGeneral.h" for other informations about the available commands.

Definition at line 21 of file [TcpUser.h](#).

#### 5.17.3.2 [char TcpUser::first\\_parameter\[1000\]](#)

Some commands need more than one parameter to be executed (ex.

write register address data). The first\_parameter array stores the first parameter of a composite command.

Definition at line 27 of file [TcpUser.h](#).

#### 5.17.3.3 [int TcpUser::register\\_address](#)

The register\_address variable stores the address of the register being written or being read.

Definition at line 37 of file [TcpUser.h](#).

#### 5.17.3.4 int TcpUser::register\_data

The register\_data variable stores the data being written in the register indicated by register\_address.

Definition at line 42 of file [TcpUser.h](#).

#### 5.17.3.5 char TcpUser::second\_parameter[1000]

The second\_parameter array stores the second parameter of a composite command.

Definition at line 32 of file [TcpUser.h](#).

#### 5.17.3.6 int TcpUser::user\_sockid

The user\_sockid variable contains the sockid of the user who sent the command.

Definition at line 47 of file [TcpUser.h](#).

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

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

## Chapter 6

# File Documentation

### 6.1 Analizzatore.h File Reference

This file contains the declarations of the functions used to scan the config file.

#### Functions

- int [AnalizzaInit](#) ([ConfObject](#) \*mioconfig, const char \*file)  
*AnalizzaInit function scans the file indicated in the second parameter searching the informations to open the digitizer and puts them in the conf\_object \* mioconfig.*
- int [AnalizzaSetup](#) ([ConfObject](#) \*mioconfig, const char \*file)  
*AnalizzaSetup function scans the file indicated in the second parameter searching the information to setup the digitizer and puts the informations in the conf\_object \* mioconfig.*
- int [AnalizzaInitPrint](#) ([ConfObject](#) \*mioconfig, const char \*file)  
*AnalizzaInitPrint function scans the file indicated in the second parameter searching the information to open the digitizer and puts the informations in the conf\_object \* mioconfig.*
- int [AnalizzaSetupPrint](#) ([ConfObject](#) \*mioconfig, const char \*file)  
*AnalizzaSetupPrint function scans the file indicated in the second parameter searching the information to setup the digitizer and puts the informations in the conf\_object \* mioconfig.*
- void [AnalizzaPrint](#) (const char \*file)  
*AnalizzaPrint function scans the file indicated in the second parameter searching the information to setup the digitizer and prints them to the stdout.*

#### 6.1.1 Detailed Description

This file contains the declarations of the functions used to scan the config file. These functions are implemented in analizzatore.c file (generated by flex).

##### Author

Daniele Berto

Definition in file [Analizzatore.h](#).

#### 6.1.2 Function Documentation

##### 6.1.2.1 int AnalizzaInit ( ConfObject \* mioconfig, const char \* file )

AnalizzaInit function scans the file indicated in the second parameter searching the informations to open the digitizer and puts them in the conf\_object \* mioconfig.

It does not print recognized lines.

## Parameters

<i>mioconfig</i>	is where the informations are stored.
<i>file</i>	is the path of the configuration file.

## Returns

int

**6.1.2.2 int AnalizzaInitPrint ( ConfObject \* *mioconfig*, const char \* *file* )**

AnalizzaInitPrint function scans the file indicated in the second parameter searching the information to open the digitizer and puts the informations in the conf\_object \* mioconfig.

It prints recognized lines.

## Parameters

<i>mioconfig</i>	is where the informations are stored.
<i>file</i>	is the path of the configuration file.

## Returns

int

**6.1.2.3 void AnalizzaPrint ( const char \* *file* )**

AnalizzaPrint function scans the file indicated in the second parameter searching the information to setup the digitizer and prints them to the stdout.

## Parameters

<i>file</i>	is the path of the configuration file.
-------------	--

## Returns

void

**6.1.2.4 int AnalizzaSetup ( ConfObject \* *mioconfig*, const char \* *file* )**

AnalizzaSetup function scans the file indicated in the second parameter searching the information to setup the digitizer and puts the informations in the conf\_object \* mioconfig.

It does not print recognized lines.

## Parameters

<i>mioconfig</i>	is where the informations are stored.
<i>file</i>	is the path of the configuration file.

## Returns

int

**6.1.2.5 int AnalizzaSetupPrint ( ConfObject \* *mioconfig*, const char \* *file* )**

AnalizzaSetupPrint function scans the file indicated in the second parameter searching the information to setup the digitizer and puts the informations in the conf\_object \* mioconfig.

It prints recognized lines.

**Parameters**

<i>mioconfig</i>	is where the informations are stored.
<i>file</i>	is the path of the configuration file.

**Returns**

int

## 6.2 Analizzatore.h

```

00001
00017 int AnalizzaInit (ConfObject * mioconfig, const char *file);
00018
00027 int AnalizzaSetup (ConfObject * mioconfig, const char *file);
00028
00037 int AnalizzaInitPrint (ConfObject * mioconfig, const char *file);
00038
00047 int AnalizzaSetupPrint (ConfObject * mioconfig, const char *file);
00048
00055 void AnalizzaPrint (const char *file);

```

## 6.3 AnalizzatoreUtils.c File Reference

This file contains the implementation of the utility functions used by the scanner.

```

#include <stdlib.h>
#include <string.h>
#include <assert.h>
#include <regex.h>
#include <stdbool.h>
#include "DefineGeneral.h"
#include "ConfObject.h"

```

**Functions**

- bool [reg\\_matches](#) (const char \*str, const char \*pattern)  
*La funzione reg\_matches serve per comparare una stringa con una espressione regolare. Restituisce true se e' stata trovata l'espressione regolare pattern nella stringa str.*
- char \* [FindPointer](#) (char \*yytext)  
*The FindPointer function returns a pointer to the first useful character after the command.*
- int [FindIntegerValue](#) (char \*yytext)  
*The FindIntegerValue function returns the value of the first integer after the command.*
- int [OutputFileFormat](#) (char \*yytext)  
*The OutputFileFormat function returns an integer that represents the format of the output file of the application.*
- int [OutputRisingFalling](#) (char \*yytext)  
*The OutputRisingFalling function returns an integer that decides whether the trigger occurs on the rising or falling edge of the signal.*
- int [OutputNIMTTL](#) (char \*yytext)  
*The OutputNIMTTL function returns an integer that represents the type of the front panel I/O LEMO connectors.*
- int [YesNoAnswer](#) (char \*yytext)  
*The YesNoAnswer function returns an integer that represents yes or no.*
- int [GetAcquisitionMode](#) (char \*yytext)  
*The GetAcquisitionMode function returns an integer that represents the acquisition mode.*
- void [GetOpenInformation](#) (char \*yytext, ConfObject \*mioconfig)

The `GetOpenInformation` function gets the information for opening the digitizer from the `yytext` string and puts it in a `ConfObject`.

- void `GetWriteRegisterInformation` (char \*yytext, `ConfObject` \*mioconfig)

The `GetWriteRegisterInformation` function gets the information for writing a specific register of the digitizer from the `yytext` string and puts it in a `ConfObject`.

- void `ChInformation` (char \*yytext, `ConfObject` \*mioconfig)

The `ChInformation` function gets the information for setting a specific channel of the digitizer from the `yytext` string and puts it in a `ConfObject`.

- void `GroupInformation` (char \*yytext, `ConfObject` \*mioconfig)

The `GroupInformation` function gets the information for setting a specific group of the digitizer from the `yytext` string and puts it in a `ConfObject`.

- void `AllInformation` (char \*yytext, `ConfObject` \*mioconfig)

The `AllInformation` function gets the information for setting all the channels of the digitizer from the `yytext` string and puts it in a `ConfObject`.

- void `FastInformation` (char \*yytext, `ConfObject` \*mioconfig)

The `FastInformation` function gets the information for setting TRn channels of the digitizer from the `yytext` string and puts it in a `ConfObject`.

- void `ChannelEnableMask` (char \*yytext, `ConfObject` \*mioconfig)

The `ChannelEnableMask` function gets the information for enabling or not the input from the channels.

- void `GroupEnableMask` (char \*yytext, `ConfObject` \*mioconfig)

The `GroupEnableMask` function gets the information for enabling or not the input from the groups.

- void `ChannelTriggerEnableMask` (char \*yytext, `ConfObject` \*mioconfig)

The `ChannelTriggerEnableMask` function gets the information for setting TRn channels of the digitizer from the `yytext` string and puts it in a `ConfObject`.

### 6.3.1 Detailed Description

This file contains the implementation of the utility functions used by the scanner. Il file `AnalizzatoreLessicale.-flex` puo' generare, usando `flex`, uno scanner per il file di configurazione. Il contenuto delle stringhe riconosciute deve essere poi estrapolato e messo nel `ConfObject`: le funzioni seguenti eseguono questo compito. Ogni funzione esegue un compito molto specifico: sono progettate guardando alla singola stringa di configurazione da cui ricavare il contenuto.

#### Author

Daniele Berto

Definition in file `AnalizzatoreUtils.c`.

### 6.3.2 Function Documentation

#### 6.3.2.1 void AllInformation ( char \* yytext, ConfObject \* mioconfig )

The `AllInformation` function gets the information for setting all the channels of the digitizer from the `yytext` string and puts it in a `ConfObject`.

Ex. `yytext` contains the string "ALL TRIGGER\_THRESHOLD 0x10000000", so we can call `AllInformation` (`yytext`, `mioconfig`).

#### Parameters

<code>yytext</code>	contains a matched string.
---------------------	----------------------------

<i>mioconfig</i>	is where the retrieved informations are stored.
------------------	---

**Returns**

void

Definition at line 347 of file [AnalizzatoreUtils.c](#).

**6.3.2.2 void ChannelEnableMask ( char \* yytext, ConfObject \* mioconfig )**

The ChannelEnableMask function gets the information for enabling or not the input from the channels.

It gets them from the yytext string and puts it in a [ConfObject](#).

Ex. yytext contains the string "CHANNEL\_ENABLE\_MASK 0x00000111", so we can call ChannelEnableMask (yytext, mioconfig).

**Parameters**

<i>yytext</i>	contains a matched string.
<i>mioconfig</i>	is where the retrieved informations are stored.

**Returns**

void

Definition at line 447 of file [AnalizzatoreUtils.c](#).

**6.3.2.3 void ChannelTriggerEnableMask ( char \* yytext, ConfObject \* mioconfig )**

The ChannelTriggerEnableMask function gets the information for setting TRn channels of the digitizer from the yytext string and puts it in a [ConfObject](#).

Ex. yytext contains the string "FAST TRIGGER\_THRESHOLD 0x10000000", so we can call ChannelTriggerEnableMask (yytext, mioconfig).

**Parameters**

<i>yytext</i>	contains a matched string.
<i>mioconfig</i>	is where the retrieved informations are stored.

**Returns**

void

Definition at line 499 of file [AnalizzatoreUtils.c](#).

**6.3.2.4 void ChInformation ( char \* yytext, ConfObject \* mioconfig )**

The ChInformation function gets the information for setting a specific channel of the digitizer from the yytext string and puts it in a [ConfObject](#).

Ex. yytext contains the string "CH 1 CHANNEL\_TRIGGER DISABLED", so we can call ChInformation (yytext, mioconfig).



## Parameters

<i>yytext</i>	contains a matched string.
<i>mioconfig</i>	is where the retrieved informations are stored.

## Returns

void

Definition at line 183 of file [AnalizzatoreUtils.c](#).

#### 6.3.2.5 void FastInformation ( char \* *yytext*, ConfObject \* *mioconfig* )

The FastInformation function gets the information for setting TRn channels of the digitizer from the *yytext* string and puts it in a [ConfObject](#).

Ex. *yytext* contains the string "FAST TRIGGER\_THRESHOLD 0x10000000", so we can call FastInformation (*yytext*, *mioconfig*).

## Parameters

<i>yytext</i>	contains a matched string.
<i>mioconfig</i>	is where the retrieved informations are stored.

## Returns

void

Definition at line 401 of file [AnalizzatoreUtils.c](#).

#### 6.3.2.6 int FindIntegerValue ( char \* *yytext* )

The FindIntegerValue function returns the value of the first integer after the command.

Ex: the string "POST\_TRIGGER 20" is in *yytext* and we call `value = find_integer_value(yytext)`. After that value contains the integer 20.

## Parameters

<i>yytext</i>	contains a matched string.
---------------	----------------------------

## Returns

int

Definition at line 51 of file [AnalizzatoreUtils.c](#).

#### 6.3.2.7 char\* FindPointer ( char \* *yytext* )

The FindPointer function returns a pointer to the first useful character after the command.

Ex: the string "POST\_TRIGGER 20" is in *yytext* and we call `punt = find_pointer(yytext)`. After that punt contains a pointer to the character '2'. This function is used by many other functions of this file.

## Parameters

<i>yytext</i>	contains a matched string.
---------------	----------------------------

**Returns**

char

Definition at line 38 of file [AnalizzatoreUtils.c](#).

**6.3.2.8 int GetAcquisitionMode ( char \* yytext )**

The GetAcquisitionMode function returns an integer that represents the acquisition mode.

Values: 0 = ACQUISITION\_ONLY, 1 = ACQUISITION\_AND\_TRGOUT, 2 = DISABLED, other=UNSET.

**Parameters**

<i>yytext</i>	contains a matched string.
---------------	----------------------------

**Returns**

int

Definition at line 113 of file [AnalizzatoreUtils.c](#).

**6.3.2.9 void GetOpenInformation ( char \* yytext, ConfObject \* mioconfig )**

The GetOpenInformation function gets the information for opening the digitizer from the yytext string and puts it in a [ConfObject](#).

Ex. yytext contains the string "OPEN PCI 0 0 0x11110000", so we can call GetOpenInformation (yytext, mioconfig).

**Parameters**

<i>yytext</i>	contains a matched string.
<i>mioconfig</i>	is where the retrieved informations are stored.

**Returns**

void

Definition at line 132 of file [AnalizzatoreUtils.c](#).

**6.3.2.10 void GetWriteRegisterInformation ( char \* yytext, ConfObject \* mioconfig )**

The GetWriteRegisterInformation function gets the information for writing a specific register of the digitizer from the yytext string and puts it in a [ConfObject](#).

Ex. yytext contains the string "WRITE\_REGISTER 1080 0000 0100", so we can call GetWriteRegisterInformation (yytext, mioconfig).

**Parameters**

<i>yytext</i>	contains a matched string.
<i>mioconfig</i>	is where the retrieved informations are stored.

**Returns**

void

Definition at line 162 of file [AnalizzatoreUtils.c](#).

#### 6.3.2.11 void GroupEnableMask ( char \* *yytext*, ConfObject \* *mioconfig* )

The GroupEnableMask function gets the information for enabling or not the input from the groups.

It gets them from the *yytext* string and puts it in a [ConfObject](#).

Ex. *yytext* contains the string "GROUP\_ENABLE\_MASK 0x10000000", so we can call GroupEnableMask (*yytext*, *mioconfig*).

##### Parameters

<i>yytext</i>	contains a matched string.
<i>mioconfig</i>	is where the retrieved informations are stored.

##### Returns

void

Definition at line 473 of file [AnalizzatoreUtils.c](#).

#### 6.3.2.12 void GroupInformation ( char \* *yytext*, ConfObject \* *mioconfig* )

The GroupInformation function gets the information for setting a specific group of the digitizer from the *yytext* string and puts it in a [ConfObject](#).

Ex. *yytext* contains the string "GR 1 CHANNEL\_TRIGGER DISABLED", so we can call GroupInformation (*yytext*, *mioconfig*).

##### Parameters

<i>yytext</i>	contains a matched string.
<i>mioconfig</i>	is where the retrieved informations are stored.

##### Returns

void

Definition at line 277 of file [AnalizzatoreUtils.c](#).

#### 6.3.2.13 int OutputFileFormat ( char \* *yytext* )

The OutputFileFormat function returns an integer that represents the format of the output file of the application.

See [ConfObject](#) documentation for other informations.

##### Parameters

<i>yytext</i>	contains a matched string.
---------------	----------------------------

##### Returns

int

Definition at line 61 of file [AnalizzatoreUtils.c](#).

#### 6.3.2.14 int OutputNIMTTL ( char \* *yytext* )

The OutputNIMTTL function returns an integer that represents the type of the front panel I/O LEMO connectors.

See [ConfObject](#) documentation for other informations.

**Parameters**

<i>yytext</i>	contains a matched string.
---------------	----------------------------

**Returns**

int

Definition at line 87 of file [AnalizzatoreUtils.c](#).

**6.3.2.15 int OutputRisingFalling ( char \* *yytext* )**

The OutputRisingFalling function returns an integer that decides whether the trigger occurs on the rising or falling edge of the signal.

See [ConfObject](#) documentation for other informations.

**Parameters**

<i>yytext</i>	contains a matched string.
---------------	----------------------------

**Returns**

int

Definition at line 74 of file [AnalizzatoreUtils.c](#).

**6.3.2.16 bool reg\_matches ( const char \* *str*, const char \* *pattern* )**

La funzione reg\_matches serve per comparare una stringa con una espressione regolare. Restituisce true se e' stata trovata l'espressione regolare pattern nella stringa str.

Definition at line 20 of file [AnalizzatoreUtils.c](#).

**6.3.2.17 int YesNoAnswer ( char \* *yytext* )**

The YesNoAnswer function returns an integer that represents yes or no.

Values: 0 = NO, 1 = YES, other=UNSET.

**Parameters**

<i>yytext</i>	contains a matched string.
---------------	----------------------------

**Returns**

int

Definition at line 100 of file [AnalizzatoreUtils.c](#).

## 6.4 AnalizzatoreUtils.c

```
00001
00010 #include <stdlib.h>
00011 #include <string.h>
00012 #include <assert.h>
00013 #include <regex.h>
00014 #include <stdbool.h>
00015 #include "DefineGeneral.h"
00016 #include "ConfObject.h"
```

```

00017
00019 bool
00020 reg_matches (const char *str, const char *pattern)
00021 {
00022     regex_t re;
00023     int ret;
00024
00025     if (regcomp (&re, pattern, REG_EXTENDED) != 0)
00026         return false;
00027
00028     ret = regexec (&re, str, (size_t) 0, NULL, 0);
00029     regfree (&re);
00030
00031     if (ret == 0)
00032         return true;
00033
00034     return false;
00035 }
00036
00037 char *
00038 FindPointer (char *yytext)
00039 {
00040     int i = 0;
00041     char *punt;
00042     while (yytext[i] != ' ' && yytext[i] != '\t')
00043         i++;
00044     while (yytext[i] == ' ' || yytext[i] == '\t')
00045         i++;
00046     punt = yytext + i;
00047     return punt;
00048 }
00049
00050 int
00051 FindIntegerValue (char *yytext)
00052 {
00053     char *punt;
00054     int integer_value;
00055     punt = FindPointer (yytext);
00056     integer_value = atoi (punt);
00057     return integer_value;
00058 }
00059
00060 int
00061 OutputFileFormat (char *yytext)
00062 {
00063     char *punt;
00064     punt = FindPointer (yytext);
00065     if (reg_matches (punt, "[Bb][Ii][Nn][Aa][Rr][Yy]")
00066         return 0;
00067     else if (reg_matches (punt, "[Aa][Ss][Cc][Ii][Ii]")
00068         return 1;
00069     else
00070         return -1;
00071 }
00072
00073 int
00074 OutputRisingFalling (char *yytext)
00075 {
00076     char *punt;
00077     punt = FindPointer (yytext);
00078     if (reg_matches (punt, "[rR][iI][sS][iI][nN][gG]")
00079         return 0;
00080     else if (reg_matches (punt, "[fF][aA][lL][lL][iI][nN][gG]")
00081         return 1;
00082     else
00083         return -1;
00084 }
00085
00086 int
00087 OutputNIMTTL (char *yytext)
00088 {
00089     char *punt;
00090     punt = FindPointer (yytext);
00091     if (reg_matches (punt, "[Nn][Ii][Mm]")
00092         return 0;
00093     else if (reg_matches (punt, "[Tt][Tt][Ll]")
00094         return 1;
00095     else
00096         return -1;
00097 }
00098
00099 int
00100 YesNoAnswer (char *yytext)
00101 {
00102     char *punt;
00103     punt = FindPointer (yytext);
00104     if (reg_matches (punt, "[Yy][Ee][Ss]")

```

```

00105     return 1;
00106 else if (reg_matches (punt, "[Nn][Oo]"))
00107     return 0;
00108 else
00109     return -1;
00110 }
00111
00112 int
00113 GetAcquisitionMode (char *yytext)
00114 {
00115     char *punt;
00116     punt = FindPointer (yytext);
00117     if (reg_matches
00118         (punt, "[aA][cC][qQ][uU][iI][sS][iI][tT][iI][oO][nN]_[oO][nN][lL][yY]"))
00119         return 0;
00120     else
00121         if (reg_matches
00122             (punt,
00123              "[aA][cC][qQ][uU][iI][sS][iI][tT][iI][oO][nN]_[aA][nN][dD]_[tT][rR][gG][oO][uU][tT]"))
00124             return 1;
00125     else if (reg_matches (punt, "[dD][iI][sS][aA][bB][lL][eE][dD]"))
00126         return 2;
00127     else
00128         return -1;
00129 }
00130
00131 void
00132 GetOpenInformation (char *yytext, ConfObject * mioconfig)
00133 {
00134     char *punt;
00135     int Type;
00136     int LinkNumber;
00137     int NodeNumber;
00138     int BaseAddress;
00139     punt = FindPointer (yytext);
00140     if (reg_matches (punt, "[Uu][Ss][Bb]"))
00141         Type = 0;
00142     else if (reg_matches (punt, "[Pp][Cc][Ii]"))
00143         Type = 1;
00144     else
00145         Type = -1;
00146     LinkNumber = FindIntegerValue (punt);
00147     punt = FindPointer (punt);
00148     NodeNumber = FindIntegerValue (punt);
00149     punt = FindPointer (punt);
00150     punt = FindPointer (punt);
00151     BaseAddress = strtoul (punt, NULL, 16);
00152
00153     mioconfig->LinkType = Type;
00154     mioconfig->LinkNumber = LinkNumber;
00155     mioconfig->ConetNode = NodeNumber;
00156     mioconfig->VMEBaseAddress = BaseAddress;
00157
00158     //printf ("%d %d %d %d\n", Type, LinkNumber, NodeNumber, BaseAddress);
00159 }
00160
00161 void
00162 GetWriteRegisterInformation (char *yytext, ConfObject * mioconfig)
00163 {
00164     char *punt;
00165     int Address;
00166     int Data;
00167     int Mask;
00168     punt = FindPointer (yytext);
00169     Address = strtoul (punt, NULL, 16);
00170     punt = FindPointer (punt);
00171     Data = strtoul (punt, NULL, 16);
00172     punt = FindPointer (punt);
00173     Mask = strtoul (punt, NULL, 16);
00174     mioconfig->Address_register = Address;
00175     mioconfig->Data_register = Data;
00176     mioconfig->Mask_register = Mask;
00177 }
00178
00179
00180 //*****
00181
00182 void
00183 ChInformation (char *yytext, ConfObject * mioconfig)
00184 {
00185     char *punt;
00186     int mod;
00187     int yes_no;
00188     int channel;
00189     int channel_trigger;
00190
00191

```

```

00192     channel = FindIntegerValue (yytext);
00193
00194
00195     if (channel < MAXCHANNELOBJECT)
00196     {
00197         mioconfig->channels[channel].set = 1;
00198         mioconfig->channels[channel].numChannel = channel;
00199
00200         punt = FindPointer (yytext);
00201         punt = FindPointer (punt);
00202
00203
00204         if (reg_matches (punt, "[Ee][Nn][Aa][Bb][Ll][Ee]_[iI][nN][pP][uU][tT]"))
00205             mod = 0;
00206         else if (reg_matches (punt, "[dD][cC]_[oO][fF][fF][sS][eE][tT]"))
00207             mod = 1;
00208         else if (reg_matches (punt, "[tT][rR][iI][gG][gG][eE][rR]_[tT][hH][rR][eE][sS][hH][oO][lL][dD]"))
00209             mod = 2;
00210         else if (reg_matches
00211             (punt, "[cC][hH][aA][nN][nN][eE][lL]_[tT][rR][iI][gG][gG][eE][rR]"))
00212             mod = 3;
00213         else
00214             mod = -1;
00215
00216
00217         if (mod == 0)
00218         {
00219             //ENABLE_INPUT
00220             punt = FindPointer (punt);
00221             if (reg_matches (punt, "[Yy][Ee][Ss]"))
00222             {
00223                 yes_no = 1;
00224                 mioconfig->channels[channel].enable_input = yes_no;
00225             }
00226             else if (reg_matches (punt, "[Nn][Oo]"))
00227             {
00228                 yes_no = 0;
00229                 mioconfig->channels[channel].enable_input = yes_no;
00230             }
00231         }
00232         else if (mod == 1)
00233         {
00234             //DC_OFFSET
00235             punt = FindPointer (punt);
00236             int dc_offset = strtoul (punt, NULL, 16);
00237             mioconfig->channels[channel].dc_offset = dc_offset;
00238         }
00239         else if (mod == 2)
00240         {
00241             //TRIGGER_THRESHOLD
00242             punt = FindPointer (punt);
00243             int trigger_threshold = strtoul (punt, NULL, 16);
00244             mioconfig->channels[channel].trigger_threshold = trigger_threshold;
00245             //mioconfig->channels[channel].trigger_threshold;
00246         }
00247         else if (mod == 3)
00248         {
00249             //CHANNEL_TRIGGER
00250             punt = FindPointer (punt);
00251             if (reg_matches
00252                 (punt, "[aA][cC][qQ][uU][iI][sS][iI][tT][iI][oO][nN]_[oO][nN][lL][yY]"))
00253             {
00254                 channel_trigger = 0;
00255                 mioconfig->channels[channel].channel_trigger = channel_trigger;
00256             }
00257             else
00258             if (reg_matches
00259                 (punt, "[aA][cC][qQ][uU][iI][sS][iI][tT][iI][oO][nN]_[aA][nN][dD]_[tT][rR][gG][oO][uU][tT]"))
00260             {
00261                 channel_trigger = 1;
00262                 mioconfig->channels[channel].channel_trigger = channel_trigger;
00263             }
00264             else if (reg_matches (punt, "[dD][iI][sS][aA][bB][lL][eE][dD]"))
00265             {
00266                 channel_trigger = 2;
00267                 mioconfig->channels[channel].channel_trigger = channel_trigger;
00268             }
00269         }
00270     } // else if (mod == 3)
00271 } //if (channel < MAXCHANNELOBJECT)
00272 } //END_FUNCTION
00273
00274 //*****
00275
00276 void
00277 GroupInformation (char *yytext, ConfObject * mioconfig)

```

```

00278 {
00279     char *punt;
00280     int mod;
00281     int yes_no;
00282     int group;
00283     //int group_mask;
00284
00285     group = FindIntegerValue (yytext);
00286
00287     if (group < MAXGROUPOBJECT)
00288     {
00289
00290         mioconfig->groups[group].set = 1;
00291         mioconfig->groups[group].numGroup = group;
00292
00293         punt = FindPointer (yytext);
00294         punt = FindPointer (punt);
00295         if (reg_matches (punt, "[Ee][Nn][Aa][Bb][Ll][Ee]_[iI][nN][pP][uU][tT]"))
00296             mod = 0;
00297         else if (reg_matches (punt, "[dD][cC]_[oO][fF][fF][sS][eE][tT]"))
00298             mod = 1;
00299         else if (reg_matches (punt, "[tT][rR][iI][gG][gG][eE][rR]_[tT][hH][rR][eE][sS][hH][oO][lL][dD]"))
00300             mod = 2;
00301         else if (reg_matches (punt, "[gG][rR][oO][uU][pP]_[tT][rR][gG]_[eE][nN][aA][bB][lL][eE]_[mM][aA][sS][kK]"))
00302             mod = 3;
00303         else
00304             mod = -1;
00305
00306         if (mod == 0)
00307         {
00308             //ENABLE_INPUT
00309             punt = FindPointer (punt);
00310             if (reg_matches (punt, "[Yy][Ee][Ss]"))
00311             {
00312                 yes_no = 1;
00313                 mioconfig->groups[group].enable_input = yes_no;
00314             }
00315             else if (reg_matches (punt, "[Nn][Oo]"))
00316             {
00317                 yes_no = 0;
00318                 mioconfig->groups[group].enable_input = yes_no;
00319             }
00320         }
00321         else if (mod == 1)
00322         {
00323             //DC_OFFSET
00324             punt = FindPointer (punt);
00325             int dc_offset = strtoul (punt, NULL, 16);
00326             mioconfig->groups[group].dc_offset = dc_offset;
00327         }
00328         else if (mod == 2)
00329         {
00330             //TRIGGER_THRESHOLD
00331             punt = FindPointer (punt);
00332             int trigger_threshold = strtoul (punt, NULL, 16);
00333             mioconfig->groups[group].trigger_threshold = trigger_threshold;
00334         }
00335         else if (mod == 3)
00336         {
00337             //GROUP_TRG_ENABLE_MASK
00338             punt = FindPointer (punt);
00339             int group_mask = strtoul (punt, NULL, 16);
00340             mioconfig->groups[group].group_trg_enable_mask = group_mask;
00341         } // else if (mod == 3)
00342     } // if (group < MAXGROUPOBJECT)
00343 } //END_FUNCTION
00344 //*****
00345
00346 void
00347 AllInformation (char *yytext, ConfObject * mioconfig)
00348 {
00349     char *punt;
00350     int mod;
00351     int yes_no;
00352     punt = FindPointer (yytext);
00353     if (reg_matches (punt, "[Ee][Nn][Aa][Bb][Ll][Ee]_[iI][nN][pP][uU][tT]"))
00354         mod = 0;
00355     else if (reg_matches (punt, "[dD][cC]_[oO][fF][fF][sS][eE][tT]"))
00356         mod = 1;
00357     else
00358         if (reg_matches
00359             (punt,
00360              "[tT][rR][iI][gG][gG][eE][rR]_[tT][hH][rR][eE][sS][hH][oO][lL][dD]"))
00361             mod = 2;
00362     else

```



```

00363     mod = -1;
00364
00365     if (mod == 0)
00366     {
00367         //ENABLE_INPUT
00368         punt = FindPointer (punt);
00369         if (reg_matches (punt, "[Yy][Ee][Ss]"))
00370         {
00371             yes_no = 1;
00372             mioconfig->enable_input = yes_no;
00373         }
00374         else if (reg_matches (punt, "[Nn][Oo]"))
00375         {
00376             yes_no = 0;
00377             mioconfig->enable_input = yes_no;
00378         }
00379     }
00380
00381
00382     else if (mod == 1)
00383     {
00384         //DC_OFFSET
00385         punt = FindPointer (punt);
00386         int dc_offset = strtoul (punt, NULL, 16);
00387         mioconfig->dc_offset = dc_offset;
00388     }
00389     else if (mod == 2)
00390     {
00391         //TRIGGER_THRESHOLD
00392         punt = FindPointer (punt);
00393         int trigger_threshold = strtoul (punt, NULL, 16);
00394         mioconfig->trigger_threshold = trigger_threshold;
00395     }
00396 } //END_FUNCTION
00397
00398 //*****
00399
00400 void
00401 FastInformation (char *yytext, ConfObject * mioconfig)
00402 {
00403     char *punt;
00404     int mod;
00405     //int yes_no;
00406     int fast;
00407     fast = FindIntegerValue (yytext);
00408
00409     if (fast < MAXFASTOBJECT)
00410     {
00411
00412         mioconfig->fasts[fast].set = 1;
00413         mioconfig->fasts[fast].numFast = fast;
00414
00415         punt = FindPointer (yytext);
00416         punt = FindPointer (punt);
00417
00418         if (reg_matches (punt, "[dD][cC]_[oO][fF][fF][sS][eE][tT]"))
00419             mod = 0;
00420         else
00421             if (reg_matches
00422                 (punt,
00423                  "[tT][rR][iI][gG][gG][eE][rR]_[tT][hH][rR][eE][sS][hH][oO][lL][dD]"))
00424                 mod = 1;
00425             else
00426                 mod = -1;
00427
00428         if (mod == 0)
00429         {
00430             //DC_OFFSET
00431             punt = FindPointer (punt);
00432             int dc_offset = strtoul (punt, NULL, 16);
00433             mioconfig->fasts[fast].dc_offset = dc_offset;
00434         }
00435
00436         else if (mod == 1)
00437         {
00438             //TRIGGER_THRESHOLD
00439             punt = FindPointer (punt);
00440             int trigger_threshold = strtoul (punt, NULL, 16);
00441             mioconfig->fasts[fast].trigger_threshold = trigger_threshold;
00442         } // if (fast < MAXFASTOBJECT)
00443     }
00444 } //END_FUNCTION
00445
00446 void
00447 ChannelEnableMask (char *yytext, ConfObject * mioconfig)
00448 {
00449

```

```

00450     int i;
00451     unsigned int bit = 1;
00452     unsigned int maschera;
00453
00454     char *punt;
00455     int channel_enable_mask;
00456     punt = FindPointer (yytext);
00457     mioconfig->channel_enable_mask = strtoul (punt, NULL, 16);
00458
00459     maschera = (unsigned int) mioconfig->channel_enable_mask;
00460     for (i = 0; i < MAXCHANNELOBJECT; i++)
00461     {
00462         if (maschera & bit)
00463         {
00464             mioconfig->channels[i].enable_input = 1;
00465             mioconfig->channels[i].set = 1;
00466             mioconfig->channels[i].numChannel = i;
00467         }
00468         maschera >>= 1;
00469     }
00470 }
00471
00472 void
00473 GroupEnableMask (char *yytext, ConfObject * mioconfig)
00474 {
00475     int i;
00476     unsigned int bit = 1;
00477     unsigned int maschera;
00478
00479     char *punt;
00480     int channel_enable_mask;
00481     punt = FindPointer (yytext);
00482     mioconfig->group_enable_mask = strtoul (punt, NULL, 16);
00483
00484     maschera = (unsigned int) mioconfig->group_enable_mask;
00485     for (i = 0; i < MAXGROUPOBJECT; i++)
00486     {
00487         if (maschera & bit)
00488         {
00489             mioconfig->groups[i].enable_input = 1;
00490             mioconfig->groups[i].set = 1;
00491             mioconfig->groups[i].numGroup = i;
00492         }
00493         maschera >>= 1;
00494     }
00495 }
00496 }
00497
00498 void
00499 ChannelTriggerEnableMask (char *yytext, ConfObject * mioconfig)
00500 {
00501     char *punt;
00502     int channel_trigger_enable_mask;
00503     punt = FindPointer (yytext);
00504     mioconfig->self_trigger_enable_mask = strtoul (punt, NULL, 16);
00505     punt = FindPointer (yytext);
00506
00507     if (reg_matches
00508         (punt, "[aA][cC][qQ][uU][iI][sS][iI][tT][iI][oO][nN]_[oO][nN][lL][yY]"))
00509     {
00510         mioconfig->self_trigger_enable_mask_mode = 0;
00511     }
00512     else
00513     if (reg_matches
00514         (punt, "[aA][cC][qQ][uU][iI][sS][iI][tT][iI][oO][nN]_[aA][nN][dD]_[tT][rR][gG][oO][uU][tT]"))
00515     {
00516         mioconfig->self_trigger_enable_mask_mode = 1;
00517     }
00518     else if (reg_matches (punt, "[dD][iI][sS][aA][bB][lL][eE][dD]"))
00519     {
00520         mioconfig->self_trigger_enable_mask_mode = 2;
00521     }
00522 }
00523 }
00524 }

```

## 6.5 AnalizzatoreUtils.h File Reference

This file contains the declaration of the utility functions used by the scanner.

```
#include <stdlib.h>
#include <string.h>
#include <assert.h>
```

## Functions

- char \* [FindPointer](#) (char \*yytext)  
*The FindPointer function returns a pointer to the first useful character after the command.*
- int [FindIntegerValue](#) (char \*yytext)  
*The FindIntegerValue function returns the value of the first integer after the command.*
- int [OutputFileFormat](#) (char \*yytext)  
*The OutputFileFormat function returns an integer that represents the format of the output file of the application.*
- int [OutputRisingFalling](#) (char \*yytext)  
*The OutputRisingFalling function returns an integer that decides whether the trigger occurs on the rising or falling edge of the signal.*
- int [OutputNIMTTL](#) (char \*yytext)  
*The OutputNIMTTL function returns an integer that represents the type of the front panel I/O LEMO connectors.*
- int [YesNoAnswer](#) (char \*yytext)  
*The YesNoAnswer function returns an integer that represents yes or no.*
- int [GetAcquisitionMode](#) (char \*yytext)  
*The GetAcquisitionMode function returns an integer that represents the acquisition mode.*
- void [GetOpenInformation](#) (char \*yytext, [ConfObject](#) \*mioconfig)  
*The GetOpenInformation function gets the information for opening the digitizer from the yytext string and puts it in a [ConfObject](#).*
- void [GetWriteRegisterInformation](#) (char \*yytext, [ConfObject](#) \*mioconfig)  
*The GetWriteRegisterInformation function gets the information for writing a specific register of the digitizer from the yytext string and puts it in a [ConfObject](#).*
- void [ChInformation](#) (char \*yytext, [ConfObject](#) \*mioconfig)  
*The ChInformation function gets the information for setting a specific channel of the digitizer from the yytext string and puts it in a [ConfObject](#).*
- void [GroupInformation](#) (char \*yytext, [ConfObject](#) \*mioconfig)  
*The GroupInformation function gets the information for setting a specific group of the digitizer from the yytext string and puts it in a [ConfObject](#).*
- void [AllInformation](#) (char \*yytext, [ConfObject](#) \*mioconfig)  
*The AllInformation function gets the information for setting all the channels of the digitizer from the yytext string and puts it in a [ConfObject](#).*
- void [FastInformation](#) (char \*yytext, [ConfObject](#) \*mioconfig)  
*The FastInformation function gets the information for setting TRn channels of the digitizer from the yytext string and puts it in a [ConfObject](#).*
- void [ChannelEnableMask](#) (char \*yytext, [ConfObject](#) \*mioconfig)  
*The ChannelEnableMask function gets the information for enabling or not the input from the channels.*
- void [GroupEnableMask](#) (char \*yytext, [ConfObject](#) \*mioconfig)  
*The GroupEnableMask function gets the information for enabling or not the input from the groups.*
- void [ChannelTriggerEnableMask](#) (char \*yytext, [ConfObject](#) \*mioconfig)  
*The ChannelTriggerEnableMask function gets the information for setting TRn channels of the digitizer from the yytext string and puts it in a [ConfObject](#).*

### 6.5.1 Detailed Description

This file contains the declaration of the utility functions used by the scanner. These functions are called in the flex file.

They help the `yylex()` function to extract the information from the matched strings.

#### Author

Daniele Berto

Definition in file [AnalizzatoreUtils.h](#).

### 6.5.2 Function Documentation

#### 6.5.2.1 void AllInformation ( char \* yytext, ConfObject \* mioconfig )

The AllInformation function gets the information for setting all the channels of the digitizer from the yytext string and puts it in a [ConfObject](#).

Ex. yytext contains the string "ALL TRIGGER\_THRESHOLD 0x10000000", so we can call AllInformation (yytext, mioconfig).

#### Parameters

<i>yytext</i>	contains a matched string.
<i>mioconfig</i>	is where the retrieved informations are stored.

#### Returns

void

Definition at line 347 of file [AnalizzatoreUtils.c](#).

#### 6.5.2.2 void ChannelEnableMask ( char \* yytext, ConfObject \* mioconfig )

The ChannelEnableMask function gets the information for enabling or not the input from the channels.

It gets them from the yytext string and puts it in a [ConfObject](#).

Ex. yytext contains the string "CHANNEL\_ENABLE\_MASK 0x00000111", so we can call ChannelEnableMask (yytext, mioconfig).

#### Parameters

<i>yytext</i>	contains a matched string.
<i>mioconfig</i>	is where the retrieved informations are stored.

#### Returns

void

Definition at line 447 of file [AnalizzatoreUtils.c](#).

#### 6.5.2.3 void ChannelTriggerEnableMask ( char \* yytext, ConfObject \* mioconfig )

The ChannelTriggerEnableMask function gets the information for setting TRn channels of the digitizer from the yytext string and puts it in a [ConfObject](#).

Ex. yytext contains the string "FAST TRIGGER\_THRESHOLD 0x10000000", so we can call ChannelTriggerEnableMask (yytext, mioconfig).

## Parameters

<i>yytext</i>	contains a matched string.
<i>mioconfig</i>	is where the retrieved informations are stored.

## Returns

void

Definition at line 499 of file [AnalizzatoreUtils.c](#).

#### 6.5.2.4 void ChInformation ( char \* *yytext*, **ConfObject** \* *mioconfig* )

The ChInformation function gets the information for setting a specific channel of the digitizer from the *yytext* string and puts it in a [ConfObject](#).

Ex. *yytext* contains the string "CH 1 CHANNEL\_TRIGGER DISABLED", so we can call ChInformation (*yytext*, *mioconfig*).

## Parameters

<i>yytext</i>	contains a matched string.
<i>mioconfig</i>	is where the retrieved informations are stored.

## Returns

void

Definition at line 183 of file [AnalizzatoreUtils.c](#).

#### 6.5.2.5 void FastInformation ( char \* *yytext*, **ConfObject** \* *mioconfig* )

The FastInformation function gets the information for setting TRn channels of the digitizer from the *yytext* string and puts it in a [ConfObject](#).

Ex. *yytext* contains the string "FAST\_TRIGGER\_THRESHOLD 0x10000000", so we can call FastInformation (*yytext*, *mioconfig*).

## Parameters

<i>yytext</i>	contains a matched string.
<i>mioconfig</i>	is where the retrieved informations are stored.

## Returns

void

Definition at line 401 of file [AnalizzatoreUtils.c](#).

#### 6.5.2.6 int FindIntegerValue ( char \* *yytext* )

The FindIntegerValue function returns the value of the first integer after the command.

Ex: the string "POST\_TRIGGER 20" is in *yytext* and we call `value = find_integer_value(yytext)`. After that value contains the integer 20.

**Parameters**

<i>yytext</i>	contains a matched string.
---------------	----------------------------

**Returns**

int

Definition at line 51 of file [AnalizzatoreUtils.c](#).

**6.5.2.7 char\* FindPointer ( char \* yytext )**

The FindPointer function returns a pointer to the first useful character after the command.

Ex: the string "POST\_TRIGGER 20" is in yytext and we call punt = find\_pointer(yytext). After that punt contains a pointer to the character '2'. This function is used by many other functions of this file.

**Parameters**

<i>yytext</i>	contains a matched string.
---------------	----------------------------

**Returns**

char

Definition at line 38 of file [AnalizzatoreUtils.c](#).

**6.5.2.8 int GetAcquisitionMode ( char \* yytext )**

The GetAcquisitionMode function returns an integer that represents the acquisition mode.

Values: 0 = ACQUISITION\_ONLY, 1 = ACQUISITION\_AND\_TRGOUT, 2 = DISABLED, other=UNSET.

**Parameters**

<i>yytext</i>	contains a matched string.
---------------	----------------------------

**Returns**

int

Definition at line 113 of file [AnalizzatoreUtils.c](#).

**6.5.2.9 void GetOpenInformation ( char \* yytext, ConfObject \* mioconfig )**

The GetOpenInformation function gets the information for opening the digitizer from the yytext string and puts it in a [ConfObject](#).

Ex. yytext contains the string "OPEN PCI 0 0 0x11110000", so we can call GetOpenInformation (yytext, mioconfig).

**Parameters**

<i>yytext</i>	contains a matched string.
<i>mioconfig</i>	is where the retrieved informations are stored.

**Returns**

void

Definition at line 132 of file [AnalizzatoreUtils.c](#).

**6.5.2.10 void GetWriteRegisterInformation ( char \* yytext, ConfObject \* mioconfig )**

The GetWriteRegisterInformation function gets the information for writing a specific register of the digitizer from the yytext string and puts it in a [ConfObject](#).

Ex. yytext contains the string "WRITE\_REGISTER 1080 0000 0100", so we can call GetWriteRegisterInformation (yytext, mioconfig).

**Parameters**

<i>yytext</i>	contains a matched string.
<i>mioconfig</i>	is where the retrieved informations are stored.

**Returns**

void

Definition at line 162 of file [AnalizzatoreUtils.c](#).

**6.5.2.11 void GroupEnableMask ( char \* yytext, ConfObject \* mioconfig )**

The GroupEnableMask function gets the information for enabling or not the input from the groups.

It gets them from the yytext string and puts it in a [ConfObject](#).

Ex. yytext contains the string "GROUP\_ENABLE\_MASK 0x10000000", so we can call GroupEnableMask (yytext, mioconfig).

**Parameters**

<i>yytext</i>	contains a matched string.
<i>mioconfig</i>	is where the retrieved informations are stored.

**Returns**

void

Definition at line 473 of file [AnalizzatoreUtils.c](#).

**6.5.2.12 void GroupInformation ( char \* yytext, ConfObject \* mioconfig )**

The GroupInformation function gets the information for setting a specific group of the digitizer from the yytext string and puts it in a [ConfObject](#).

Ex. yytext contains the string "GR 1 CHANNEL\_TRIGGER DISABLED", so we can call GroupInformation (yytext, mioconfig).

**Parameters**

<i>yytext</i>	contains a matched string.
<i>mioconfig</i>	is where the retrieved informations are stored.

**Returns**

void

Definition at line 277 of file [AnalizzatoreUtils.c](#).

**6.5.2.13 int OutputFileFormat ( char \* yytext )**

The OutputFileFormat function returns an integer that represents the format of the output file of the application.

See [ConfObject](#) documentation for other informations.



## Parameters

<i>yytext</i>	contains a matched string.
---------------	----------------------------

## Returns

int

Definition at line 61 of file [AnalizzatoreUtils.c](#).

**6.5.2.14 int OutputNIMTTL ( char \* *yytext* )**

The OutputNIMTTL function returns an integer that represents the type of the front panel I/O LEMO connectors.

See [ConfObject](#) documentation for other informations.

## Parameters

<i>yytext</i>	contains a matched string.
---------------	----------------------------

## Returns

int

Definition at line 87 of file [AnalizzatoreUtils.c](#).

**6.5.2.15 int OutputRisingFalling ( char \* *yytext* )**

The OutputRisingFalling function returns an integer that decides whether the trigger occurs on the rising or falling edge of the signal.

See [ConfObject](#) documentation for other informations.

## Parameters

<i>yytext</i>	contains a matched string.
---------------	----------------------------

## Returns

int

Definition at line 74 of file [AnalizzatoreUtils.c](#).

**6.5.2.16 int YesNoAnswer ( char \* *yytext* )**

The YesNoAnswer function returns an integer that represents yes or no.

Values: 0 = NO, 1 = YES, other=UNSET.

## Parameters

<i>yytext</i>	contains a matched string.
---------------	----------------------------

## Returns

int

Definition at line 100 of file [AnalizzatoreUtils.c](#).

## 6.6 AnalizzatoreUtils.h

```

00001
00008 #include <stdlib.h>
00009 #include <string.h>
00010 #include <assert.h>
00011
00020 char *FindPointer (char *yytext);
00021
00029 int FindIntegerValue (char *yytext);
00030
00037 int OutputFileFormat (char *yytext);
00038
00045 int OutputRisingFalling (char *yytext);
00046
00053 int OutputNIMTTL (char *yytext);
00054
00061 int YesNoAnswer (char *yytext);
00062
00069 int GetAcquisitionMode (char *yytext);
00070
00078 void GetOpenInformation (char *yytext, ConfObject * mioconfig);
00079
00087 void GetWriteRegisterInformation (char *yytext,
    ConfObject * mioconfig);
00088
00096 void ChInformation (char *yytext, ConfObject * mioconfig);
00097
00105 void GroupInformation (char *yytext, ConfObject * mioconfig);
00106
00114 void AllInformation (char *yytext, ConfObject * mioconfig);
00115
00123 void FastInformation (char *yytext, ConfObject * mioconfig);
00124
00133 void ChannelEnableMask (char *yytext, ConfObject * mioconfig);
00134
00143 void GroupEnableMask (char *yytext, ConfObject * mioconfig);
00144
00152 void ChannelTriggerEnableMask (char *yytext, ConfObject * mioconfig);

```

## 6.7 ApplicationSetup.cpp File Reference

```

#include "ConfObject.h"
#include "ApplicationSetup.h"
#include "OutputModule.h"
#include "DefineGeneral.h"
#include <CAENDigitizer.h>
#include <stdio.h>
#include <unistd.h>
#include <ctype.h>

```

### 6.7.1 Detailed Description

Il singleton [ApplicationSetup](#) permette di salvare e rendere disponibili in tutte le parti del programma le informazioni necessarie per il funzionamento dell'applicazione. Esse sono inserite dall'utente lanciando il programma o inserendo l'apposito input ad applicazione avviata. Dunque, [ApplicationSetup](#) ricava le informazioni da argc e argv, cioè dagli argomenti del main. Occorre che la funzione main chiami il metodo ApplicationSetupSet(argc, argv) per riempire [ApplicationSetup](#). Gli argomenti del main vengono analizzati in modo classico, utilizzando la funzione "getopt".

#### Author

Daniele Berto

Definition in file [ApplicationSetup.cpp](#).

## 6.8 ApplicationSetup.cpp

```

00001
00010 #include "ConfObject.h"
00011 #include "ApplicationSetup.h"
00012 #include "OutputModule.h"
00013 #include "DefineGeneral.h"
00014 #include <CAENDigitizer.h>
00015 #include <stdio.h>
00016 #include <unistd.h>
00017 #include <ctype.h>
00018
00019 /*
00020 Implementazione standard dei metodi per implementare il singleton design pattern
00021 */
00022 ApplicationSetup *
00023 ApplicationSetup::application_setup_pInstance = NULL;
00024
00025 ApplicationSetup *
00026 ApplicationSetup::Instance ()
00027 {
00028     if (!application_setup_pInstance) // Only allow one instance of class to be generated.
00029         application_setup_pInstance = new ApplicationSetup ();
00030
00031     return application_setup_pInstance;
00032 }
00033
00034
00036 ApplicationSetup::ApplicationSetup ()
00037 {
00038     channel_visualized = 0;
00039
00040     imset = 0;
00041
00042     application_setup_log_file_path = (const char *)malloc ((strlen ("
00043 ./LogFile") + 1));
00044     strcpy ((char *) application_setup_log_file_path, "./LogFile");
00045
00046     application_setup_data_file_path = (const char *) malloc ((strlen ("
00047 ./RawData/data.txt") + 1));
00048     strcpy ((char *) application_setup_data_file_path, "./RawData/data.txt");
00049
00050     application_setup_conf_file_path = (const char *)malloc ((strlen ("
00051 ./ConfigurationFile") + 1));
00052     strcpy ((char *) application_setup_conf_file_path, "./ConfigurationFile")
00053 ;
00054
00055     application_setup_input_mode = (const char *) malloc ((strlen ("default") + 1
00056 ));
00057     strcpy ((char *) application_setup_input_mode, "default");
00058
00059     application_setup_data_file_size_path = (const char *) malloc (
00060 strlen (application_setup_data_file_path) + 3);
00061     strcpy ((char *) application_setup_data_file_size_path,
00062 application_setup_data_file_path);
00063     strcat ((char *) application_setup_data_file_size_path, "sz");
00064
00065     application_setup_data_file_punt = fopen (
00066 application_setup_data_file_path, "a");
00067     application_setup_data_file_size_punt = fopen (
00068 application_setup_data_file_size_path, "a");
00069 }
00070
00071 void
00072 ApplicationSetup::ApplicationSetupSet (int argc_arg, char **argv_arg)
00073 {
00074     int i;
00075     argc = argc_arg;
00076     argv = (char **) malloc (argc * sizeof (char *));
00077     for (i = 0; i < argc; i++)
00078     {
00079         argv[i] = (char *) malloc (sizeof (char) * (strlen (argv_arg[i]) + 1));
00080         strcpy (argv[i], argv_arg[i]);
00081     }
00082     ApplicationSetup::ArgumentsParsing ();
00083     ApplicationSetup::FetchInputMode (
00084 application_setup_input_mode);
00085 }
00086
00087
00088 int
00089 ApplicationSetup:: ApplicationSetupDataFileModify (const
00090 char *application_setup_data_file_path_arg)
00091 {

```

```

00084 free ((void *) application_setup_data_file_path);
00085 application_setup_data_file_path = NULL;
00086 application_setup_data_file_path = (char *) malloc (strlen (
application_setup_data_file_path_arg) + 1);
00087 strcpy ((char *) application_setup_data_file_path,
application_setup_data_file_path_arg);
00088
00089 if (application_setup_data_file_punt != NULL)
00090     fclose (application_setup_data_file_punt);
00091
00092 application_setup_data_file_punt = fopen (
application_setup_data_file_path, "a");
00093
00094 free ((void *) application_setup_data_file_size_path);
00095 application_setup_data_file_size_path = NULL;
00096 application_setup_data_file_size_path = (char *) malloc (strlen (
application_setup_data_file_path_arg) + 3);
00097 strcpy ((char *) application_setup_data_file_size_path,
application_setup_data_file_path_arg);
00098 strcat ((char *) application_setup_data_file_size_path, "sz");
00099
00100 if (application_setup_data_file_size_punt != NULL)
00101     fclose (application_setup_data_file_size_punt);
00102
00103 application_setup_data_file_size_punt = fopen (
application_setup_data_file_size_path, "a");
00104 }
00105
00106
00107 FILE *
00108 ApplicationSetup::ApplicationSetupGetDataFilePunt ()
00109 {
00110     return application_setup_data_file_punt;
00111 }
00112
00113
00114 FILE *
00115 ApplicationSetup::ApplicationSetupGetDataFileSizePunt
()
00116 {
00117     return application_setup_data_file_size_punt;
00118 }
00119
00120
00121 void
00122 ApplicationSetup::FetchInputMode (const char *application_setup_input_mode)
00123 {
00124
00125     OutputModule *output_module;
00126     output_module = OutputModule::Instance ();
00127
00128     if (!strcmp ("user", application_setup_input_mode))
00129     {
00130         input_mode = 0;
00131         output_module->Output ("User command mode activated\n");
00132     }
00133
00134     else if (!strcmp ("tcp", application_setup_input_mode))
00135     {
00136         input_mode = 1;
00137         output_module->Output ("Tcp command mode activated\n");
00138     }
00139
00140     else if (!strcmp ("all", application_setup_input_mode))
00141     {
00142         input_mode = 2;
00143         output_module->Output ("Tcp and User command mode activated\n");
00144     }
00145
00146     else
00147     {
00148         output_module->Output ("Tcp and User command mode activated\n");
00149         input_mode = 2;
00150     }
00151 }
00152
00153
00154 void
00155 ApplicationSetup::ArgumentsParsing ()
00156 {
00157
00158     int c = 0;
00159
00160     opterr = 0;
00161
00162     while ((c = getopt (argc, argv, "f:m:d:l:")) != -1)
00163         switch (c)

```

```

00164     {
00165         case 'f':
00166             application_setup_conf_file_path =
00167                 (char *) malloc (strlen (optarg) + 1);
00168             strcpy ((char *) application_setup_conf_file_path, optarg);
00169             break;
00170         case 'm':
00171             application_setup_input_mode = (char *) malloc (strlen (optarg) + 1);
00172             strcpy ((char *) application_setup_input_mode, optarg);
00173             break;
00174         case 'd':
00175             ApplicationSetup::ApplicationSetupDataFileModify (
00176                 optarg);
00177             break;
00178         case 'l':
00179             application_setup_log_file_path =
00180                 (char *) malloc (strlen (optarg) + 1);
00181             strcpy ((char *) application_setup_log_file_path, optarg);
00182             break;
00183         case '?':
00184             if (optopt == 'f' || optopt == 'm' || optopt == 'd' || optopt == 'l')
00185                 fprintf (stderr, "Option -%c requires an argument.\n", optopt);
00186             else if (isprint (optopt))
00187                 fprintf (stderr, "Unknown option '-%c'.\n", optopt);
00188             else
00189                 fprintf (stderr, "Unknown option character '\\x%x'.\n", optopt);
00190             break;
00191     }

```

## 6.9 ApplicationSetup.h File Reference

### Classes

- class [ApplicationSetup](#)

The [ApplicationSetup](#) class gets the application settings from the main parameters.

### 6.9.1 Detailed Description

#### Author

Daniele Berto

Definition in file [ApplicationSetup.h](#).

## 6.10 ApplicationSetup.h

```

00001
00015 class ApplicationSetup
00016 {
00017 private:
00018
00022     static ApplicationSetup *application_setup_pInstance;
00023
00027     ApplicationSetup ();
00028 public:
00029
00033     int channel_visualized;
00034
00038     static ApplicationSetup *Instance ();
00039
00043     ConfObject application_setup_conf_object;
00044
00048     FILE *application_setup_conf_file;
00049
00053     int argc;
00054
00058     int imset;
00059
00063     char **argv;
00064
00068     int ApplicationSetupDataFileModify (const char

```

```

00069             *application_setup_data_file_path_arg);
00070
00074 FILE *ApplicationSetupGetDataFilePunt ();
00075
00079 FILE *ApplicationSetupGetDataFileSizePunt ();
00080
00084 const char *application_setup_conf_file_path;
00085
00089 const char *application_setup_log_file_path;
00090
00094 const char *application_setup_data_file_path;
00095
00099 const char *application_setup_data_file_size_path;
00100
00104 const char *application_setup_input_mode;
00105
00109 FILE *application_setup_data_file_punt;
00110
00114 FILE *application_setup_data_file_size_punt;
00115
00119 int input_mode;
00120
00124 void ArgumentsParsing ();
00125
00133 void ApplicationSetupSet (int argc, char **argv);
00134
00140 void FetchInputMode (const char *application_setup_input_mode);
00141 };

```

## 6.11 ClientApplication.c File Reference

This file contains the client\_application.

```

#include "DefineGeneral.h"
#include "DefineCommands.h"
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <string.h>
#include <regex.h>
#include <stdbool.h>
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>
#include <ctype.h>

```

### Functions

- char \* [FindPointer](#) (char \*yytext)

*The FindPointer function returns a pointer to the first useful character after the command.*

- void \* [ricevitore\\_function](#) (void \*nothing)
- bool [reg\\_matches](#) (const char \*str, const char \*pattern)

*Confronta la stringa puntata da str con il pattern indicato da pattern. Restituisci true se e' possibile trovare pattern in str, false al contrario.*

- void [Help](#) ()
- int [command\\_parser](#) (char \*inputline, char \*sendline)

*The command\_parser function returns a number which correspond to the command inserted by the user in accordance with the our conventions.*

- int [main](#) (int argc, char \*\*argv)

*The main function of the client application.*

## Variables

- int `i`
- int `sockfd`
- int `n`
- int `comando_da_inviare`
- char `inputline` [1000]
- char `sendline` [1000]
- char `receiveline` [1000]
- int `go` = 0
- int `no_new_line` = 0

### 6.11.1 Detailed Description

This file contains the `client_application`.

Definition in file [ClientApplication.c](#).

### 6.11.2 Function Documentation

#### 6.11.2.1 `int command_parser ( char * inputline, char * sendline )`

The `command_parser` function returns a number which correspond to the command inserted by the user in accordance with the our conventions.

See [DefineCommands.h](#) file for informations about the codes of the available commands.

##### Parameters

<i>inputline</i>	contains the command being interpreted
<i>sendline</i>	contains the informations being sent via TCP/IP for composite commands.

##### Returns

int

Definition at line 142 of file [ClientApplication.c](#).

#### 6.11.2.2 `char* FindPointer ( char * yytext )`

The `FindPointer` function returns a pointer to the first useful character after the command.

Ex: the string "POST\_TRIGGER 20" is in `yytext` and we call `punt = find_pointer(yytext)`. After that `punt` contains a pointer to the character '2'. This function is used by many other functions of this file.

##### Parameters

<i>yytext</i>	contains a matched string.
---------------	----------------------------

##### Returns

char

Definition at line 44 of file [ClientApplication.c](#).

#### 6.11.2.3 `void Help ( )`

Definition at line 102 of file [ClientApplication.c](#).

#### 6.11.2.4 `int main ( int argc, char ** argv )`

The main function of the client application.



## Parameters

<i>argc</i>	is the number of the main arguments
<i>argv</i>	are the main arguments

## Returns

int

Definition at line 286 of file [ClientApplication.c](#).

#### 6.11.2.5 bool reg\_matches ( const char \* *str*, const char \* *pattern* )

Confronta la stringa puntata da *str* con il pattern indicato da *pattern*. Restituisci true se e' possibile trovare pattern in *str*, false al contrario.

Definition at line 84 of file [ClientApplication.c](#).

#### 6.11.2.6 void\* ricevitore\_function ( void \* *nothing* )

Definition at line 57 of file [ClientApplication.c](#).

### 6.11.3 Variable Documentation

#### 6.11.3.1 int comando\_da\_inviare

Definition at line 35 of file [ClientApplication.c](#).

#### 6.11.3.2 int go = 0

Definition at line 39 of file [ClientApplication.c](#).

#### 6.11.3.3 int i

Definition at line 33 of file [ClientApplication.c](#).

#### 6.11.3.4 char inputline[1000]

Definition at line 36 of file [ClientApplication.c](#).

#### 6.11.3.5 int n

Definition at line 34 of file [ClientApplication.c](#).

#### 6.11.3.6 int no\_new\_line = 0

Definition at line 40 of file [ClientApplication.c](#).

#### 6.11.3.7 char receiveline[1000]

Definition at line 38 of file [ClientApplication.c](#).

### 6.11.3.8 char sendline[1000]

Definition at line 37 of file [ClientApplication.c](#).

### 6.11.3.9 int sockfd

Definition at line 34 of file [ClientApplication.c](#).

## 6.12 ClientApplication.c

```

00001
00008 /*
00009 Questo sorgente, se compilato con il make file contenuto nella cartella principale del progetto, produce il
00010 programma objectdumpclient, il lato client del programma
00011 objectdump.
00012 Objectdumpclient consente di inviare ad objectdump comandi in input via TCP/IP. La lista dei codici che il
00013 client puo' inviare e' contenuta nel file DefineCommands.h.
00014 Ricordo che affinche' il server riconosca i comandi inviati via TCP/IP, i dati inviati devono essere un
00015 array di caratteri contenente nelle prime tre posizioni i numeri
00016 25 21 17 e poi il codice del comando da inviare.
00017 L'output mandato dal server e' ascoltato da un thread separato che esegue la funzione
00018 "ricevitore_function".
00019 Per lanciare il programma occorre aggiungere l'opzione -i serveraddress dopo il nome dell'eseguibile.
00020 */
00021
00022 #include "DefineGeneral.h"
00023 #include "DefineCommands.h"
00024 #include <sys/socket.h>
00025 #include <netinet/in.h>
00026 #include <arpa/inet.h>
00027 #include <stdio.h>
00028 #include <string.h>
00029 #include <regex.h>
00030 #include <stdbool.h>
00031 #include <stdlib.h>
00032 #include <pthread.h>
00033 #include <unistd.h>
00034 #include <ctype.h>
00035
00036 //Variabili globali contenenti la socket di comunicazione col server e i dati da inviare e ricevere.
00037 int i;
00038 int sockfd, n;
00039 int comando_da_inviare;
00040 char inputline[STANDARDBUFFERLIMIT];
00041 char sendline[STANDARDBUFFERLIMIT];
00042 char receiveline[STANDARDBUFFERLIMIT];
00043 int go = 0;
00044 int no_new_line = 0;
00045
00046 //Funzione che restituisce un puntatore alla prima parola contenuta in yytext
00047 char *
00048 FindPointer (char *yytext)
00049 {
00050     int i = 0;
00051     char *punt;
00052     while (yytext[i] != ' ' && yytext[i] != '\t')
00053         i++;
00054     while (yytext[i] == ' ' || yytext[i] == '\t')
00055         i++;
00056     punt = yytext + i;
00057     return punt;
00058 }
00059
00060 //Funzione eseguita dal thread che riceve l'output dal server
00061 void * ricevitore_function (void * nothing)
00062 {
00063     int i;
00064     while (go)
00065     {
00066         bzero(receiveline, STANDARDBUFFERLIMIT);
00067         if (recv (sockfd, receiveline, STANDARDBUFFERLIMIT, 0) <= 0)
00068         {
00069             fprintf (stdout, "Comunicazione col server interrotta\n");
00070             shutdown (sockfd, 2);
00071             go = 0;
00072         } else {
00073             //If necessario per stampare correttamente a video l'output del server.

```

```

00071         if (receiveline[strlen(receiveline)-1] == 10)
00072             fprintf(stdout, "%s", receiveline);
00073         else if (receiveline[strlen(receiveline)-1] != 10)
00074             {
00075                 fprintf(stdout, "%s", receiveline);
00076                 if (strlen(receiveline) > 1 &&
no_new_line == 0)
00077                     fprintf (stdout, "\n");
00078             } // else if (receiveline[strlen(receiveline)-1] != 10)
00079         } //} else {
00080     } //while (go)
00081 } //void * ricevitore_function (void * nothing)
00082
00084 bool reg_matches(const char *str, const char *pattern)
00085 {
00086     regex_t re;
00087     int ret;
00088
00089     if (regcomp(&re, pattern, REG_EXTENDED) != 0)
00090         return false;
00091
00092     ret = regexec(&re, str, (size_t) 0, NULL, 0);
00093     regfree(&re);
00094
00095     if (ret == 0)
00096         return true;
00097
00098     return false;
00099 }
00100
00101 //Stampa i comandi disponibile. La funzione e' chiamata quando l'utente digita il comando help.
00102 void Help()
00103 {
00104     printf ("Available command list:\n");
00105     printf ("init: open the digitizer\n");
00106     printf ("setup: setup the digitizer\n");
00107     printf ("start: start the data acquisition\n");
00108     printf ("stop: stop the data acquisition\n");
00109     printf ("prestart: start the preprocessing thread\n");
00110     printf ("prestop: stop the preprocessing thread\n");
00111     printf ("vistart [channelnumber]: start the visualization thread\n");
00112     printf ("vistop: stop the visualization thread\n");
00113     printf ("rawstart: start the raw data writing thread\n");
00114     printf ("rawstop: stop the raw data writing thread\n");
00115     printf ("close: close the digitizer\n");
00116     printf ("send: send a software trigger\n");
00117     printf ("help\n");
00118     printf ("check: check the correctness of the configuration file\n");
00119     printf ("chkconf: print the content of the configuration file\n");
00120     printf ("write register 0x[register] 0x[data]\n");
00121     printf ("read register 0x[register]\n");
00122     printf ("-f [conf file path]: change the configuration file path\n");
00123     printf ("-d [data file path]: change the data file path\n");
00124     printf ("-l [log file path]: change the log file path\n");
00125     printf ("print: print the internal configuration object used to configure the digitizer\n");
00126     printf ("print files: print the path of the configuration file, of the data file and of the log file\n");
00127     printf ("status: print the status of the threads acquisition, preprocessing, raw data and visualization\n");
00128     printf ("more: display the content of the logfile\n");
00129     printf ("exit: quit program\n");
00130     printf ("quit: quit program\n");
00131 }
00132
00133
00141 int
00142 command_parser (char *inputline , char *sendline)
00143 { char * my_punt;
00144   const char *my_punt_const;
00145   no_new_line = 0;
00146
00147   if (reg_matches(inputline, "^[iI][Nn][Ii][Tt][ \\t]*$"))
00148   {
00149       //printf ("1\n");
00150       return INIT;
00151   }
00152   else if (reg_matches (inputline, "^[Ss][Ee][Tt][Uu][Pp][ \\t]*$"))
00153   {
00154       //printf ("2\n");
00155       return SETUP;
00156   }
00157   else if (reg_matches (inputline, "^[pP][rR][eE][sS][tT][aA][rR][tT][ \\t]*$"))
00158   {
00159       //printf ("3\n");
00160       return PRESTART;
00161   }
00162   else if (reg_matches (inputline, "^[pP][rR][eE][sS][tT][Oo][Pp][ \\t]*$"))
00163   {

```

```

00164         //printf ("4\n");
00165         return PRESTOP;
00166     }
00167     else if (reg_matches (inputline, "^[sS][tT][aA][rR][tT][ \\t]*$"))
00168     {
00169         //printf ("7\n");
00170         return START;
00171     }
00172     else if (reg_matches (inputline, "^[sS][tT][oO][pP][ \\t]*$"))
00173     {
00174         //printf ("8\n");
00175         return STOP;
00176     }
00177     else if (reg_matches (inputline, "^[sS][eE][nN][dD][ \\t]*$"))
00178     {
00179         //printf ("9\n");
00180         return SEND;
00181     }
00182     else if (reg_matches (inputline, "^[cC][lL][oO][sS][eE][ \\t]*$"))
00183     {
00184         //printf ("10\n");
00185         return CLOSE;
00186     }
00187     else if (reg_matches (inputline, "^[qQ][uU][iI][tT][ \\t]*$"))
00188     {
00189         //printf ("11\n");
00190         return QUIT;
00191     }
00192     else if (reg_matches (inputline, "^[rR][aA][wW][sS][tT][aA][rR][tT][ \\t]*$"))
00193     {
00194         //printf ("12\n");
00195         return RAWSTART;
00196     }
00197     else if (reg_matches (inputline, "^[Rr][Aa][Ww][Ss][Tt][Oo][Pp][ \\t]*$"))
00198     {
00199         //printf ("13\n");
00200         return RAWSTOP;
00201     }
00202     else if (reg_matches (inputline, "^[pP][rR][iI][nN][tT][ \\t]*$"))
00203     {
00204         //printf ("14\n");
00205         return PRINT;
00206     }
00207     else if (reg_matches (inputline, "^[Cc][Hh][Ee][Cc][Kk][ \\t]*$"))
00208     {
00209         //printf ("16\n");
00210         return CHECK;
00211     }
00212     else if (reg_matches (inputline, "^[Cc][Hh][Kk][Cc][Oo][Nn][Ff][ \\t]*$"))
00213     {
00214         //printf ("17\n");
00215         return CHKCONF;
00216     }
00217     else if (reg_matches (inputline, "^[Mm][Oo][Rr][Ee][ \\t]*$"))
00218     {
00219         no_new_line = 1;
00220         //printf ("18\n");
00221         return MORE;
00222     }
00223     else if (reg_matches (inputline, "^[sS][tT][aA][tT][uU][sS][ \\t]*$"))
00224     {
00225         //printf ("17\n");
00226         return STATUS;
00227     }
00228     else if (reg_matches (inputline, "^[-][Ff][ ]+.$"))
00229     {
00230         //printf ("18\n");
00231         return CHANGECONF;
00232     }
00233     else if (reg_matches (inputline, "^[-][Dd][ ]+.$"))
00234     {
00235         //printf ("18\n");
00236         return CHANGEDATA;
00237     }
00238     else if (reg_matches (inputline, "^[-][Ll][ ]+.$"))
00239     {
00240         //printf ("18\n");
00241         return CHANGELOG;
00242     }
00243     else if (reg_matches (inputline, "^[pP][rR][iI][nN][tT][ \\t]+[Ff][Ii][Ll][Ee][Ss][ \\t]*$"))
00244     {
00245         //printf ("18\n");
00246         return PRINTFILES;
00247     }
00248     else if (reg_matches (inputline, "^[Ww][Rr][iI][tT][Ee][ \\t]+[Rr][Ee][Gg][Ii][Ss][Tt][Ee][Rr][ \\t]+(0x[0-9a-fA-F]{1,16})[ \\t]+(0x[0-9a-fA-F]{1,16})[ \\t]*$")) //5==print
00249     {

```

```

00250         my_punt = FindPointer (inputline);
00251         my_punt = FindPointer (my_punt);
00252         strcpy(sendline + 4, my_punt);
00253         return WRITEREGISTER;
00254     }
00255     else if (reg_matches (inputline, "[Rr][Ee][Aa][Dd][ \\t]+[Rr][Ee][Gg][Ii][Ss][Tt][Ee][Rr][ \\t]
]+(0x[0-9a-fA-F]{1,4})[ \\t]*$")) //5==print
00256     {
00257         my_punt = FindPointer (inputline);
00258         my_punt = FindPointer (my_punt);
00259         strcpy(sendline + 4, my_punt);
00260         return READREGISTER;
00261     }
00262     else if (reg_matches (inputline, "[hH][eE][lL][pP][ \\t]*$"))
00263     {
00264         Help();
00265         return -1;
00266     }
00267     else if (reg_matches (inputline, "[Ee][Xx][Ii][Tt]$"))
00268     {
00269         fprintf(stdout, "Exiting...\n");
00270         return -1;
00271     }
00272     else
00273     {
00274         fprintf (stderr, "Unrecognized command\n");
00275         return -1;
00276     }
00277 }
00278
00285 int
00286 main (int argc, char **argv)
00287 {
00288
00289     //Variabili per ottenere l'indirizzo del server dagli argomenti del main.
00290     const char *server_address;
00291     server_address == NULL;
00292
00293     int c = 0;
00294
00295     int flag_arg = 0;
00296
00297     opterr = 0;
00298
00299     while ((c = getopt (argc, argv, "i:")) != -1)
00300     switch (c)
00301     {
00302         case 'i':
00303             server_address = (char *)malloc(strlen(optarg) + 1);
00304             strcpy((char *)server_address, optarg);
00305             flag_arg = 1;
00306             break;
00307         case '?':
00308             if (optopt == 'i')
00309                 fprintf (stderr, "Option -%c requires an argument.\n", optopt);
00310             else if (isprint (optopt))
00311                 fprintf (stderr, "Unknown option '-%c'.\n", optopt);
00312             else
00313                 fprintf (stderr,
00314                     "Unknown option character '\\x%x'.\n",
00315                     optopt);
00316             break;
00317     }
00318
00319     if (flag_arg == 0)
00320     {
00321         fprintf(stderr, "You have not insert server address: use -i flag\n");
00322         fprintf(stderr, "usage: [executablepath] -i [serveraddress]\n");
00323         return 1;
00324     }
00325
00326     //Inserisco queste informazioni dentro sendline in modo tale che il server riconosca i dati inviategli.
00327     sendline[0] = 25;
00328     sendline[1] = 21;
00329     sendline[2] = 17;
00330     sendline[4] = '\\0';
00331
00332     const char * my_punt;
00333
00334     //thread id del thread che ascolta i messaggi provenienti dal server
00335     pthread_t ricevitore;
00336
00337     //Codice necessario all'apertura di una socket con il client.
00338     struct sockaddr_in servaddr, cliaddr;
00339     char recvline[STANDARDBUFFERLIMIT];
00340     sockfd = socket (AF_INET, SOCK_STREAM, 0);
00341     bzero (&servaddr, sizeof (servaddr));

```

```

00342     servaddr.sin_family = AF_INET;
00343     servaddr.sin_addr.s_addr = inet_addr (server_address);
00344     servaddr.sin_port = htons (1111);
00345
00346     fprintf(stdout, "Welcome to objectDump tcp service, press help for getting the available command list\n");
00347 ;
00348     connect (sockfd, (struct sockaddr *) &servaddr, sizeof (servaddr));
00349     perror(" ");
00350
00351     //Creo il thread ricevitore, controllato dalla variabile go.
00352     go = 1;
00353     pthread_create(&ricevitore, NULL, ricevitore_function, NULL);
00354
00355     //Ciclo di fetching dell'input da tastiera il ciclo termina quando l'utente inserisce il comando exit.
00356     do
00357     {
00358         fflush(stdout);
00359
00360         //Prelevo un input da tastiera
00361         fgets (inputline, STANDARDBUFFERLIMIT, stdin);
00362         if (inputline[strlen (inputline) - 1] == '\n')
00363             inputline[strlen (inputline) - 1] = '\0';
00364
00365         //Metto in sendline il codice corrispondente al comando inserito dall'utente.
00366         //Se il comando non e' riconosciuto, viene restituito -1 e non e' inviato nulla al server.
00367         sendline[3] = (char) command_parser (inputline,
00368         sendline);
00369
00370         //Se il client invia un comando per modificare il path dei files usati dal server, e' necessario
00371         //ricavare dalla stringa in input il nuovo path
00372         //e inviarlo al server.
00373         if (sendline[3] == CHANGECONF || sendline[3] ==
00374         CHANGEDATA || sendline[3] == CHANGELOG)
00375         {
00376             my_punt = inputline + 3;
00377             strcpy(sendline + 4, my_punt);
00378         }
00379
00380         //Se il comando inserito dall'utente non e' stato riconosciuto, al server non e' inviato nulla.
00381         if (sendline[3] != -1)
00382         {
00383             send (sockfd, sendline, STANDARDBUFFERLIMIT, 0);
00384         }
00385     }
00386     //Il ciclo termina quando l'utente inserisce il comando exit.
00387     while (!reg_matches (inputline, "^[Ee][Xx][Ii][Tt][ \\t]*$" ) != 0);
00388
00389     //Il programma deve terminare, quindi interrompo anche il thread ricevitore
00390     go = 0;
00391
00392     //Chiudo la socket di comunicazione col server.
00393     shutdown (sockfd, 2);
00394 } //int main (int argc, char **argv)
00395
00396
00397
00398
00399
00400
00401
00402
00403
00404
00405
00406
00407
00408
00409
00410
00411
00412
00413
00414
00415
00416
00417
00418
00419
00420
00421
00422
00423
00424
00425
00426
00427
00428
00429
00430
00431
00432
00433
00434
00435
00436
00437
00438
00439
00440
00441
00442
00443
00444
00445
00446
00447
00448
00449
00450
00451
00452
00453
00454
00455
00456
00457
00458
00459
00460
00461
00462
00463
00464
00465
00466
00467
00468
00469
00470
00471
00472
00473
00474
00475
00476
00477
00478
00479
00480
00481
00482
00483
00484
00485
00486
00487
00488
00489
00490
00491
00492
00493
00494
00495
00496
00497
00498
00499
00500
00501
00502
00503
00504
00505
00506
00507
00508
00509
00510
00511
00512
00513
00514
00515
00516
00517
00518
00519
00520
00521
00522
00523
00524
00525
00526
00527
00528
00529
00530
00531
00532
00533
00534
00535
00536
00537
00538
00539
00540
00541
00542
00543
00544
00545
00546
00547
00548
00549
00550
00551
00552
00553
00554
00555
00556
00557
00558
00559
00560
00561
00562
00563
00564
00565
00566
00567
00568
00569
00570
00571
00572
00573
00574
00575
00576
00577
00578
00579
00580
00581
00582
00583
00584
00585
00586
00587
00588
00589
00590
00591
00592
00593
00594
00595
00596
00597
00598
00599
00600
00601
00602
00603
00604
00605
00606
00607
00608
00609
00610
00611
00612
00613
00614
00615
00616
00617
00618
00619
00620
00621
00622
00623
00624
00625
00626
00627
00628
00629
00630
00631
00632
00633
00634
00635
00636
00637
00638
00639
00640
00641
00642
00643
00644
00645
00646
00647
00648
00649
00650
00651
00652
00653
00654
00655
00656
00657
00658
00659
00660
00661
00662
00663
00664
00665
00666
00667
00668
00669
00670
00671
00672
00673
00674
00675
00676
00677
00678
00679
00680
00681
00682
00683
00684
00685
00686
00687
00688
00689
00690
00691
00692
00693
00694
00695
00696
00697
00698
00699
00700
00701
00702
00703
00704
00705
00706
00707
00708
00709
00710
00711
00712
00713
00714
00715
00716
00717
00718
00719
00720
00721
00722
00723
00724
00725
00726
00727
00728
00729
00730
00731
00732
00733
00734
00735
00736
00737
00738
00739
00740
00741
00742
00743
00744
00745
00746
00747
00748
00749
00750
00751
00752
00753
00754
00755
00756
00757
00758
00759
00760
00761
00762
00763
00764
00765
00766
00767
00768
00769
00770
00771
00772
00773
00774
00775
00776
00777
00778
00779
00780
00781
00782
00783
00784
00785
00786
00787
00788
00789
00790
00791
00792
00793
00794
00795
00796
00797
00798
00799
00800
00801
00802
00803
00804
00805
00806
00807
00808
00809
00810
00811
00812
00813
00814
00815
00816
00817
00818
00819
00820
00821
00822
00823
00824
00825
00826
00827
00828
00829
00830
00831
00832
00833
00834
00835
00836
00837
00838
00839
00840
00841
00842
00843
00844
00845
00846
00847
00848
00849
00850
00851
00852
00853
00854
00855
00856
00857
00858
00859
00860
00861
00862
00863
00864
00865
00866
00867
00868
00869
00870
00871
00872
00873
00874
00875
00876
00877
00878
00879
00880
00881
00882
00883
00884
00885
00886
00887
00888
00889
00890
00891
00892
00893
00894
00895
00896
00897
00898
00899
00900
00901
00902
00903
00904
00905
00906
00907
00908
00909
00910
00911
00912
00913
00914
00915
00916
00917
00918
00919
00920
00921
00922
00923
00924
00925
00926
00927
00928
00929
00930
00931
00932
00933
00934
00935
00936
00937
00938
00939
00940
00941
00942
00943
00944
00945
00946
00947
00948
00949
00950
00951
00952
00953
00954
00955
00956
00957
00958
00959
00960
00961
00962
00963
00964
00965
00966
00967
00968
00969
00970
00971
00972
00973
00974
00975
00976
00977
00978
00979
00980
00981
00982
00983
00984
00985
00986
00987
00988
00989
00990
00991
00992
00993
00994
00995
00996
00997
00998
00999

```

## 6.13 CommunicationObject.cpp File Reference

```
#include "DefineGeneral.h"
#include "DefineCommands.h"
#include "TcpUser.h"
#include "ConfObject.h"
#include "DigitizerErrorObject.h"
#include "LogFile.h"
#include "DigitizerObject.h"
#include "DigitizerObjectGeneric.h"
#include "RawData.h"
#include "DigitizerStateMachine.h"
#include "ApplicationSetup.h"
#include "Input.h"
#include "CommunicationObject.h"
#include "DigitizerFlowControl.h"
#include "ConfigurationConsistence.h"
#include "AnalizzatoreUtils.h"
#include "Analizzatore.h"
#include "OutputModule.h"
#include <pthread.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <thread>
#include <mutex>
#include <condition_variable>
```

### 6.13.1 Detailed Description

#### Author

Daniele Berto

Definition in file [CommunicationObject.cpp](#).

## 6.14 CommunicationObject.cpp

```
00001
00006 #include "DefineGeneral.h"
00007 #include "DefineCommands.h"
00008 #include "TcpUser.h"
00009 #include "ConfObject.h"
00010 #include "DigitizerErrorObject.h"
00011 #include "LogFile.h"
00012 #include "DigitizerObject.h"
00013 #include "DigitizerObjectGeneric.h"
00014 #include "RawData.h"
00015 #include "DigitizerStateMachine.h"
00016 #include "ApplicationSetup.h"
00017 #include "Input.h"
00018 #include "CommunicationObject.h"
00019 #include "DigitizerFlowControl.h"
00020 #include "ConfigurationConsistence.h"
00021 #include "AnalizzatoreUtils.h"
00022 #include "Analizzatore.h"
00023 #include "OutputModule.h"
00024 #include <pthread.h>
00025 #include <stdio.h>
```

```

00026 #include<string.h>
00027 #include<stdlib.h>
00028 #include<sys/socket.h>
00029 #include<arpa/inet.h>
00030 #include<unistd.h>
00031 #include <thread>
00032 #include <mutex>
00033 #include <condition_variable>
00034
00035 using namespace std;
00036
00037 void
00038 CommunicationObject::Main ()
00039 {
00040
00041     OutputModule *output_module;
00042     output_module = OutputModule::Instance ();
00043
00044     int client_sock; /*Socket descriptor del client. */
00045     int c; /*Lunghezza di sockaddr_in. */
00046     struct sockaddr_in server, client;
00047     int socket_desc;
00048     socket_desc = socket (AF_INET, SOCK_STREAM, 0);
00049
00050     /*Inizializzo la struttura sockaddr_in */
00051     server.sin_family = AF_INET;
00052     server.sin_addr.s_addr = INADDR_ANY;
00053     server.sin_port = htons (1111);
00054
00055     /*Effettuo il bind */
00056     bind (socket_desc, (struct sockaddr *) &server, sizeof (server));
00057
00058     while (go)
00059     {
00060         listen (socket_desc, 1);
00061
00062         c = sizeof (struct sockaddr_in);
00063
00064         if (go == 0)
00065         {
00066             break;
00067         }
00068
00069         client_sock =
00070         accept (socket_desc, (struct sockaddr *) &client, (socklen_t *) &c);
00071
00072         if (go == 0)
00073         {
00074             break;
00075         }
00076
00077         if (client_sock >= 0)
00078         {
00079             output_module->OutputModuleStdoutOff();
00080             output_module->OutputModuleSockidOn(client_sock);
00081             output_module->TcpUserArrayInsert (client_sock);
00082             output_module->Output ("Comunicazione col server stabilita\n");
00083             worker_thread = new thread (&CommunicationObject::Worker, this,
00084                                         (void *) &client_sock);
00085         }
00086     }
00087     // while go
00088 }
00089
00090 }
00091
00092 void
00093 CommunicationObject::Worker (void *socket_desc)
00094 {
00095     OutputModule *output_module;
00096     output_module = OutputModule::Instance ();
00097
00098     int j;
00099     int sock = *(int *) socket_desc;
00100     int read_size;
00101     char buffer[STANDARDBUFFERLIMIT];
00102     const char *my_punt;
00103     bzero (buffer, STANDARDBUFFERLIMIT);
00104
00105     while (go)
00106     {
00107         bzero (buffer, STANDARDBUFFERLIMIT);
00108         if ((read_size = recv (sock, buffer, STANDARDBUFFERLIMIT, 0)) <= 0)
00109         {
00110             //fprintf (stderr, "Comunicazione con il client %d interrotta\nobjectDump>",
00111                     // sock);

```



```

00113     output_module->TcpUserArrayDelete (sock);
00114 }
00115
00116     unique_lock<mutex> ReservedKeyBoardInputAreaHandle (ReservedKeyBoardInputArea);
00117
00118     if (num_mex == MAXCOMMAND)
00119         BlockedProducerInput.wait (ReservedKeyBoardInputAreaHandle);
00120     coda = (coda + 1) % MAXCOMMAND;
00121     command[coda].command_sent_by_user = buffer[3];
00122
00123     if (buffer[3] == CHANGECONF || buffer[3] == CHANGEDATA
00124         || buffer[3] == CHANGELOG || buffer[3] == WRITEREGISTER
00125         || buffer[3] == READREGISTER)
00126     {
00127         bzero (command[coda].first_parameter, STANDARDBUFFERLIMIT);
00128         my_punt = buffer + 4;
00129         strncpy (command[coda].first_parameter, my_punt, 999);
00130         fprintf (stderr, "command[coda].first_parameter: %s\n",
00131             command[coda].first_parameter);
00132     }
00133
00134     command[coda].user_sockid = sock;
00135     num_mex++;
00136     pthread_mutex_lock (&DigitizerFlowControl::input_flow_mutex);
00137     pthread_cond_signal (&DigitizerFlowControl::input_flow_cond);
00138     pthread_mutex_unlock (&DigitizerFlowControl::input_flow_mutex);
00139
00140     BlockedConsumerInput.notify_one();
00141
00142     } // while go
00143
00144     BlockedConsumerInput.notify_one();
00145
00146 }
00147
00148
00149 void
00150 CommunicationObject::CommunicationObjectInit ()
00151 {
00152
00153     go = 1;
00154
00155     main_thread = new thread (&CommunicationObject::Main, this);
00156
00157 }
00158
00159
00160 TcpUser CommunicationObject::GetCommand ()
00161 {
00162     TcpUser
00163         tmp;
00164     tmp.command_sent_by_user = 0;
00165     tmp.user_sockid = -1;
00166     unique_lock<mutex> ReservedKeyBoardInputAreaHandle (ReservedKeyBoardInputArea);
00167
00168     if (num_mex == 0)
00169     {
00170         tmp.command_sent_by_user = 0;
00171         tmp.user_sockid = -1;
00172     }
00173     else
00174     {
00175         tmp = command[testa];
00176
00177         command[testa].command_sent_by_user = 0;
00178         command[testa].user_sockid = -1;
00179         bzero (command[testa].first_parameter, STANDARDBUFFERLIMIT);
00180
00181         testa = (testa + 1) % MAXCOMMAND;
00182
00183         num_mex--;
00184         BlockedProducerInput.notify_one();
00185     }
00186     return tmp;
00187 }
00188
00189
00190 void
00191 CommunicationObject::Finish ()
00192 {
00193     int i;
00194
00195     go = 0;
00196
00197     for (i = 0; i < 10; i++)
00198         BlockedProducerInput.notify_one();
00199

```

```

00200 }
00201
00202
00203 CommunicationObject::CommunicationObject ()
00204 {
00205     num_mex = 0;
00206     coda = 0;
00207     testa = 1;
00208 }

```

## 6.15 CommunicationObject.h File Reference

```

#include "DefineGeneral.h"
#include <pthread.h>
#include <thread>
#include <mutex>
#include <condition_variable>

```

### Classes

- class [CommunicationObject](#)

## 6.16 CommunicationObject.h

```

00001
00012 #include "DefineGeneral.h"
00013 #include <pthread.h>
00014 #include <thread>
00015 #include <mutex>
00016 #include <condition_variable>
00017
00018 using namespace std;
00019
00020
00021 class CommunicationObject
00022 {
00023 public:
00024
00028     int num_mex;
00029
00033     int testa;
00034
00038     int coda;
00039
00043     int go;
00044
00051     void GetCommand (int *socketid, int *command);
00052
00058     TcpUser GetCommand ();
00059
00065     void CommunicationObjectInit ();
00066
00072     void Main ();
00073
00079     void Worker (void *socket_desc);
00080
00087     TcpUser command[MAXCOMMAND];
00088
00093     void Finish ();
00094
00098     thread * main_thread;
00099
00103     thread * worker_thread;
00104
00108     mutex ReservedKeyBoardInputArea;
00109
00113     condition_variable BlockedProducerInput;
00114
00118     condition_variable BlockedConsumerInput;
00119
00123     mutex Acquisition_Mutex;
00124

```

```

00128     condition_variable Acquisition_Cond1;
00129
00133     condition_variable Acquisition_Cond2;
00134
00138         CommunicationObject ();
00139 };

```

## 6.17 ConfigurationConsistence.cpp File Reference

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <CAENDigitizer.h>
#include "DefineGeneral.h"
#include "ConfObject.h"
#include "ApplicationSetup.h"
#include "Analizzatore.h"
#include "ConfigurationConsistence.h"
#include "OutputModule.h"

```

### 6.17.1 Detailed Description

#### Author

Daniele Berto

Definition in file [ConfigurationConsistence.cpp](#).

## 6.18 ConfigurationConsistence.cpp

```

00001
00006 #include <stdio.h>
00007 #include <stdlib.h>
00008 #include <string.h>
00009 #include <CAENDigitizer.h>
00010 #include "DefineGeneral.h"
00011 #include "ConfObject.h"
00012 #include "ApplicationSetup.h"
00013 #include "Analizzatore.h"
00014 #include "ConfigurationConsistence.h"
00015 #include "OutputModule.h"
00016
00017
00018
00019 int
00020 ConfigurationConsistence::
00021 ConfigurationConsistenceConfFileInit (const char *conf_file)
00022 {
00023
00024     OutputModule *output_module;
00025     output_module = OutputModule::Instance ();
00026
00027     int ret_flag = 0;
00028     ConfObject conf_object_private;
00029
00030
00031     if (AnalizzaInitPrint (&conf_object_private, conf_file))
00032     {
00033         fprintf (stderr, "%s: Errore: il file di configurazione non esiste\n",
00034                 __func__);
00035         ret_flag = 1;
00036         output_module->Output ("Errore: il file di configurazione non esiste\n");
00037         return ret_flag;
00038     }
00039
00040     if (conf_object_private.LinkType == -1)
00041     {
00042         fprintf (stderr,

```

```

00043         "%s: Error: you have not specify LinkType in OPEN parameter\n",
00044         __func__);
00045         ret_flag++;
00046         output_module->Output("Errore: il file di configurazione non esiste\n");
00047     }
00048
00049     if (conf_object_private.LinkNumber == -1)
00050     {
00051         fprintf (stderr,
00052                 "%s: Error: you have not specify LinkNumber in OPEN parameter\n",
00053                 __func__);
00054         ret_flag++;
00055         output_module->Output("Error: you have not specify LinkNumber in OPEN parameter\n");
00056     }
00057
00058     if (conf_object_private.ConetNode == -1)
00059     {
00060         fprintf (stderr,
00061                 "%s: Error: you have not specify ConetNode in OPEN parameter\n",
00062                 __func__);
00063         ret_flag++;
00064         output_module->Output("Error: you have not specify ConetNode in OPEN parameter\n");
00065     }
00066
00067     if (conf_object_private.VMEBaseAddress == -1)
00068     {
00069         fprintf (stderr,
00070                 "%s: Error: you have not specify VMEBaseAddress in OPEN parameter\n",
00071                 __func__);
00072         ret_flag++;
00073         output_module->Output("Error: you have not specify VMEBaseAddress in OPEN parameter\n");
00074     }
00075
00076
00077     if (ret_flag == 4)
00078     {
00079         fprintf (stderr,
00080                 "%s: Ops, have you forgotten to insert OPEN attribute in the configuration file?\n",
00081                 __func__);
00082         output_module->Output("Ops, have you forgotten to insert OPEN attribute in the configuration
00083         file?\n");
00084     }
00085     return ret_flag;
00086
00087 } //int ConfigurationConsistence::ConfigurationConsistenceConfFileInit (const char * conf_file)
00088
00089
00090
00091 int
00092 ConfigurationConsistence::
00093 ConfigurationConsistenceConfFileInitNoPrint (const char *
00094 conf_file)
00095 {
00096     OutputModule *output_module;
00097     output_module = OutputModule::Instance ();
00098
00099     int ret_flag = 0;
00100     ConfObject conf_object_private;
00101
00102
00103     if (AnalizzaInit (&conf_object_private, conf_file))
00104     {
00105         ret_flag = 1;
00106         output_module->Output("Errore: il file di configurazione non esiste\n");
00107         return ret_flag;
00108     }
00109
00110     if (conf_object_private.LinkType == -1)
00111     {
00112         ret_flag++;
00113         output_module->Output("Error: you have not specify LinkType in OPEN parameter\n");
00114     }
00115
00116     if (conf_object_private.LinkNumber == -1)
00117     {
00118         ret_flag++;
00119         output_module->Output("Error: you have not specify LinkNumber in OPEN parameter\n");
00120     }
00121
00122     if (conf_object_private.ConetNode == -1)
00123     {
00124         ret_flag++;
00125         output_module->Output("Error: you have not specify ConetNode in OPEN parameter\n");
00126     }
00127

```

```

00128     if (conf_object_private.VMEBaseAddress == -1)
00129     {
00130         ret_flag++;
00131         output_module->Output("Error: you have not specify VMEBaseAddress in OPEN parameter\n");
00132     }
00133
00134
00135     if (ret_flag == 4)
00136     {
00137         output_module->Output("Ops, have you forgotten to insert OPEN attribute in the configuration
00138 file?\n");
00139     }
00140     return ret_flag;
00141
00142 }
00143
00144
00145
00146 int
00147 ConfigurationConsistence::ConfigurationConsistenceConfFileSetupEssentialWithInitCheck
00148 (const char *conf_file)
00149 {
00150     OutputModule *output_module;
00151     output_module = OutputModule::Instance ();
00152
00153     int ret_flag = 0;
00154     ConfObject conf_object_private;
00155     ret_flag =
00156         ConfigurationConsistence::
00157             ConfigurationConsistenceConfFileInit (conf_file);
00158
00159     if (AnalizzaSetupPrint (&conf_object_private, conf_file))
00160     {
00161         ret_flag++;
00162         output_module->Output("Errore: il file di configurazione non esiste\n");
00163         return ret_flag;
00164     }
00165
00166     if (conf_object_private.record_length == -1)
00167     {
00168         ret_flag++;
00169         output_module->Output("Error: you have not specify RECORD_LENGTH\n");
00170     }
00171
00172     if (conf_object_private.max_num_events_BLT == -1)
00173     {
00174         ret_flag++;
00175         output_module->Output("Error: you have not specify MAX_NUM_EVENTS_BLT\n");
00176     }
00177     return ret_flag;
00178
00179 }
00180
00181
00182
00183 int
00184 ConfigurationConsistence:: ConfigurationConsistenceConfFileSetupEssentialWithInitCheckNoPrint
00185 (const char *conf_file)
00186 {
00187     OutputModule *output_module;
00188     output_module = OutputModule::Instance ();
00189
00190     int ret_flag = 0;
00191     ConfObject conf_object_private;
00192     ret_flag =
00193         ConfigurationConsistence::
00194             ConfigurationConsistenceConfFileInitNoPrint (conf_file);
00195
00196     if (AnalizzaSetup (&conf_object_private, conf_file))
00197     {
00198         ret_flag++;
00199         output_module->Output("Errore: il file di configurazione non esiste\n");
00200         return ret_flag;
00201     }
00202
00203     if (conf_object_private.record_length == -1)
00204     {
00205         ret_flag++;
00206         output_module->Output("Error: you have not specify RECORD_LENGTH\n");
00207     }
00208
00209     if (conf_object_private.max_num_events_BLT == -1)
00210     {

```

```

00210         ret_flag++;
00211         output_module->Output("Error: you have not specify MAX_NUM_EVENTS_BLT\n");
00212     }
00213
00214     return ret_flag;
00215
00216 }
00217
00218
00219 int
00220 ConfigurationConsistence::ConfigurationConsistenceConfFileSetupEssentialWithoutInitCheck
    (const char *conf_file)
00221 {
00222     OutputModule *output_module;
00223     output_module = OutputModule::Instance ();
00224
00225     int ret_flag = 0;
00226     ConfObject conf_object_private;
00227
00228     if (AnalizzaSetupPrint (&conf_object_private, conf_file))
00229     {
00230         ret_flag++;
00231         output_module->Output("Errore: il file di configurazione non esiste\n");
00232         return ret_flag;
00233     }
00234
00235     if (conf_object_private.record_length == -1)
00236     {
00237         ret_flag++;
00238         output_module->Output("Error: you have not specify RECORD_LENGTH\n");
00239     }
00240
00241     if (conf_object_private.max_num_events_BLT == -1)
00242     {
00243         ret_flag++;
00244         output_module->Output("Error: you have not specify MAX_NUM_EVENTS_BLT\n");
00245     }
00246
00247     return ret_flag;
00248 }
00249
00250
00251
00252
00253 int
00254 ConfigurationConsistence:: ConfigurationConsistenceConfFileSetupEssentialWithoutInitCheckNoPrint
    (const char *conf_file)
00255 {
00256     OutputModule *output_module;
00257     output_module = OutputModule::Instance ();
00258
00259     int ret_flag = 0;
00260     ConfObject conf_object_private;
00261
00262     if (AnalizzaSetup (&conf_object_private, conf_file))
00263     {
00264         ret_flag++;
00265         output_module->Output("Errore: il file di configurazione non esiste\n");
00266         return ret_flag;
00267     }
00268
00269     if (conf_object_private.record_length == -1)
00270     {
00271         ret_flag++;
00272         output_module->Output("Error: you have not specify RECORD_LENGTH\n");
00273     }
00274
00275     if (conf_object_private.max_num_events_BLT == -1)
00276     {
00277         ret_flag++;
00278         output_module->Output("Error: you have not specify MAX_NUM_EVENTS_BLT\n");
00279     }
00280
00281     return ret_flag;
00282 }
00283 }

```

## 6.19 ConfigurationConsistence.h File Reference

## Classes

- class [ConfigurationConsistence](#)

The [ConfigurationConsistence](#) class provides useful methods for checking the consistence of a configuration file.

## 6.20 ConfigurationConsistence.h

```

00001
00012 class ConfigurationConsistence
00013 {
00014 public:
00015
00022 int ConfigurationConsistenceConfFileInit (const char *conf_file_path)
00023 ;
00023
00030 int ConfigurationConsistenceConfFileInitNoPrint (const char
00031             *conf_file_path);
00032
00040 int ConfigurationConsistenceConfFileSetupEssentialWithInitCheck
00041 (const char
00042             *conf_file_path);
00042
00050 int
00051 ConfigurationConsistenceConfFileSetupEssentialWithInitCheckNoPrint
00052 (const
00053             char
00054             *conf_file_path);
00062 int ConfigurationConsistenceConfFileSetupEssentialWithoutInitCheck
00063 (const
00064             char
00065             *conf_file_path);
00073 int
00074 ConfigurationConsistenceConfFileSetupEssentialWithoutInitCheckNoPrint
00075 (const char *conf_file_path);
00076 };

```

## 6.21 ConfObject.cpp File Reference

```

#include "DefineGeneral.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "ConfObject.h"
#include "ApplicationSetup.h"
#include "OutputModule.h"

```

## 6.22 ConfObject.cpp

```

00001
00005 #include "DefineGeneral.h"
00006 #include <stdio.h>
00007 #include <stdlib.h>
00008 #include <string.h>
00009 #include "ConfObject.h"
00010 #include "ApplicationSetup.h"
00011 #include "OutputModule.h"
00012
00013
00014 ConfObject::ConfObject ()
00015 {
00016     LinkType = -1;
00017     LinkNumber = -1;
00018     ConetNode = -1;
00019     VMEBaseAddress = -1;
00020     DSR4_Frequency = -1;
00021     output_file_format = -1;

```





Generated on Wed Jun 10 2015 14:57:38 for objectDump by Doxygen



Generated on Wed Jun 10 2015 14:57:38 for objectDump by Doxygen



## 6.23 ConfObject.h File Reference

```
#include "DefineGeneral.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

### Classes

- class [ChannelObject](#)  
The [ChannelObject](#) class picks up all the settings taken from the configuration file regarding one channel of the digitizer.
- class [GroupObject](#)  
The [GroupObject](#) class picks up all the settings taken from the configuration file regarding one group of the digitizer.
- class [FastObject](#)  
The [FastObject](#) class picks up all the settings taken from the configuration file regarding a TRn input channel of the digitizer (only for x742 series).
- class [ConfObject](#)  
The [ConfObject](#) class picks up all the settings taken from the configuration file.

## 6.24 ConfObject.h

```
00001
00005 #include "DefineGeneral.h"
00006 #include <stdio.h>
00007 #include <stdlib.h>
00008 #include <string.h>
00009
00017 class ChannelObject
00018 {
00019 public:
00020
00024     int set;
00025
00029     int numChannel;
00030
00034     int enable_input;
00035
00039     int dc_offset;
00040
00044     int trigger_threshold;
00045
00050     int channel_trigger;
00051
00055     ChannelObject ();
00056
00061     void PrintChannel ();
00062 };
00063
00064
00065
00073 class GroupObject
00074 {
00075 public:
00076
00080     int set;
00081
00085     int numGroup;
00086
00090     int enable_input;
00091
00095     int dc_offset;
00096
00100     int trigger_threshold;
00101
00108     int group_trg_enable_mask;
00109
00113     GroupObject ();
00114
```

```
00119 void PrintGroup ();
00120 };
00121
00122
00130 class FastObject
00131 {
00132 public:
00133
00137 int set;
00138
00142 int numFast;
00143
00147 int dc_offset;
00148
00152 int trigger_threshold;
00153
00157 FastObject ();
00158
00163 void PrintFast ();
00164 };
00165
00166
00174 class ConfObject
00175 {
00176 public:
00177
00182 int LinkType;
00183
00188 int LinkNumber;
00189
00193 int ConetNode;
00194
00199 int VMEBaseAddress;
00200
00205 int DSR4_Frequency;
00206
00211 int output_file_format;
00212
00216 char *gnuplot;
00217
00222 int header_yes_no;
00223
00227 int record_length;
00228
00233 int test_pattern;
00234
00239 int desmod;
00240
00245 int external_trigger_acquisition_mode;
00246
00251 int fast_trigger_acquisition_mode;
00252
00257 int enable_fast_trigger_digitizing;
00258
00262 int max_num_events_BLT;
00263
00268 int decimation_factor;
00269
00275 int post_trigger;
00276
00281 int rising_falling;
00282
00286 int use_interrupt;
00287
00292 int nim_ttl;
00293
00297 int Address_register;
00298
00302 int Mask_register;
00303
00307 int Data_register;
00308
00312 int enable_input;
00313
00317 int dc_offset;
00318
00322 int trigger_threshold;
00323
00327 ChannelObject channels[MAXCHANNELOBJECT];
00328
00332 GroupObject groups[MAXGROUPOBJECT];
00333
00337 FastObject fasts[MAXFASTOBJECT];
00338
00342 int channel_enable_mask;
00343
00347 int group_enable_mask;
```

```
00348
00352  int self_trigger_enable_mask;
00353
00357  int self_trigger_enable_mask_mode;
00358
00362      ConfObject ();
00363
00368  void PrintAll ();
00369
00374  void PrintAllHuman ();
00375 };
```

## 6.25 DefineCommands.h File Reference

This file contains the codes of the commands used to send command via TCP/IP.

### Macros

- `#define INIT 1`
- `#define SETUP 2`
- `#define PRESTART 3`
- `#define PRESTOP 4`
- `#define VISTART 5`
- `#define VISTOP 6`
- `#define START 7`
- `#define STOP 8`
- `#define SEND 9`
- `#define CLOSE 10`
- `#define QUIT 11`
- `#define RAWSTART 12`
- `#define RAWSTOP 13`
- `#define PRINT 14`
- `#define CHECK 16`
- `#define CHKCONF 17`
- `#define MORE 18`
- `#define CHANGECONF 21`
- `#define CHANGEDATA 22`
- `#define CHANGELOG 23`
- `#define WRITEREGISTER 24`
- `#define READREGISTER 25`
- `#define PRINTFILES 28`
- `#define STATUS 30`

### 6.25.1 Detailed Description

This file contains the codes of the commands used to send command via TCP/IP.

#### Author

Daniele Berto

Definition in file [DefineCommands.h](#).

### 6.25.2 Macro Definition Documentation

#### 6.25.2.1 `#define CHANGECONF 21`

Definition at line 24 of file [DefineCommands.h](#).

#### 6.25.2.2 `#define CHANGEDATA 22`

Definition at line 25 of file [DefineCommands.h](#).

#### 6.25.2.3 `#define CHANGELOG 23`

Definition at line 26 of file [DefineCommands.h](#).

#### 6.25.2.4 `#define CHECK 16`

Definition at line 21 of file [DefineCommands.h](#).

#### 6.25.2.5 `#define CHKCONF 17`

Definition at line 22 of file [DefineCommands.h](#).

#### 6.25.2.6 `#define CLOSE 10`

Definition at line 16 of file [DefineCommands.h](#).

#### 6.25.2.7 `#define INIT 1`

Definition at line 7 of file [DefineCommands.h](#).

#### 6.25.2.8 `#define MORE 18`

Definition at line 23 of file [DefineCommands.h](#).

#### 6.25.2.9 `#define PRESTART 3`

Definition at line 9 of file [DefineCommands.h](#).

#### 6.25.2.10 `#define PRESTOP 4`

Definition at line 10 of file [DefineCommands.h](#).

#### 6.25.2.11 `#define PRINT 14`

Definition at line 20 of file [DefineCommands.h](#).

#### 6.25.2.12 `#define PRINTFILES 28`

Definition at line 29 of file [DefineCommands.h](#).

#### 6.25.2.13 `#define QUIT 11`

Definition at line 17 of file [DefineCommands.h](#).



#### 6.25.2.14 #define RAWSTART 12

Definition at line 18 of file [DefineCommands.h](#).

#### 6.25.2.15 #define RAWSTOP 13

Definition at line 19 of file [DefineCommands.h](#).

#### 6.25.2.16 #define READREGISTER 25

Definition at line 28 of file [DefineCommands.h](#).

#### 6.25.2.17 #define SEND 9

Definition at line 15 of file [DefineCommands.h](#).

#### 6.25.2.18 #define SETUP 2

Definition at line 8 of file [DefineCommands.h](#).

#### 6.25.2.19 #define START 7

Definition at line 13 of file [DefineCommands.h](#).

#### 6.25.2.20 #define STATUS 30

Definition at line 30 of file [DefineCommands.h](#).

#### 6.25.2.21 #define STOP 8

Definition at line 14 of file [DefineCommands.h](#).

#### 6.25.2.22 #define VISTART 5

Definition at line 11 of file [DefineCommands.h](#).

#### 6.25.2.23 #define VISTOP 6

Definition at line 12 of file [DefineCommands.h](#).

#### 6.25.2.24 #define WRITEREGISTER 24

Definition at line 27 of file [DefineCommands.h](#).

## 6.26 DefineCommands.h

```
00001
00007 #define INIT 1
00008 #define SETUP 2
00009 #define PRESTART 3
```

```

00010 #define PRESTOP 4
00011 #define VISTART 5
00012 #define VISTOP 6
00013 #define START 7
00014 #define STOP 8
00015 #define SEND 9
00016 #define CLOSE 10
00017 #define QUIT 11
00018 #define RAWSTART 12
00019 #define RAWSTOP 13
00020 #define PRINT 14
00021 #define CHECK 16
00022 #define CHKCONF 17
00023 #define MORE 18
00024 #define CHANGECONF 21
00025 #define CHANGEDATA 22
00026 #define CHANGELOG 23
00027 #define WRITEREGISTER 24
00028 #define READREGISTER 25
00029 #define PRINTFILES 28
00030 #define STATUS 30

```

## 6.27 DefineGeneral.h File Reference

This file contains same parameters used by the program.

### Macros

- #define [STANDARDBUFFERLIMIT](#) 1000
- #define [RAWDATAQUEUE](#) 10  
*The dimension of the rawdata thread queue.*
- #define [VISUALIZATIONQUEUE](#) 10  
*The dimension of the visualization thread queue.*
- #define [PREPROCESSINGQUEUE](#) 10  
*The dimension of the preprocessing thread queue.*
- #define [MAXCHANNELOBJECT](#) 64  
*The max number of channels managed by the program.*
- #define [MAXGROUPOBJECT](#) 8  
*The max number of groups managed by the program.*
- #define [MAXFASTOBJECT](#) 2  
*The max number of fast channels managed by the program.*
- #define [MAXCOMMAND](#) 3  
*The dimension of the command buffer queue (see [CommunicationObject](#) class).*
- #define [ALL](#) 2
- #define [ONLYUSER](#) 0
- #define [ONLYTCP](#) 1

### 6.27.1 Detailed Description

This file contains same parameters used by the program. In particular, it contains the dimension of the queues used by the aquisition threads.

#### Author

Daniele Berto

Definition in file [DefineGeneral.h](#).

## 6.27.2 Macro Definition Documentation

### 6.27.2.1 #define ALL 2

Definition at line 31 of file [DefineGeneral.h](#).

### 6.27.2.2 #define MAXCHANNELOBJECT 64

The max number of channels managed by the program.

Definition at line 19 of file [DefineGeneral.h](#).

### 6.27.2.3 #define MAXCOMMAND 3

The dimension of the command buffer queue (see [CommunicationObject](#) class).

Definition at line 28 of file [DefineGeneral.h](#).

### 6.27.2.4 #define MAXFASTOBJECT 2

The max number of fast channels managed by the program.

Definition at line 25 of file [DefineGeneral.h](#).

### 6.27.2.5 #define MAXGROUPOBJECT 8

The max number of groups managed by the program.

Definition at line 22 of file [DefineGeneral.h](#).

### 6.27.2.6 #define ONLYTCP 1

Definition at line 33 of file [DefineGeneral.h](#).

### 6.27.2.7 #define ONLYUSER 0

Definition at line 32 of file [DefineGeneral.h](#).

### 6.27.2.8 #define PREPROCESSINGQUEUE 10

The dimension of the preprocessing thread queue.

Definition at line 16 of file [DefineGeneral.h](#).

### 6.27.2.9 #define RAWDATAQUEUE 10

The dimension of the rawdata thread queue.

Definition at line 10 of file [DefineGeneral.h](#).

### 6.27.2.10 #define STANDARDBUFFERLIMIT 1000

Definition at line 7 of file [DefineGeneral.h](#).

### 6.27.2.11 #define VISUALIZATIONQUEUE 10

The dimension of the visualization thread queue.

Definition at line 13 of file [DefineGeneral.h](#).

## 6.28 DefineGeneral.h

```

00001
00007 #define STANDARDBUFFERLIMIT 1000
00008
00010 #define RAWDATAQUEUE 10
00011
00013 #define VISUALIZATIONQUEUE 10
00014
00016 #define PREPROCESSINGQUEUE 10
00017
00019 #define MAXCHANNELOBJECT 64
00020
00022 #define MAXGROUPOBJECT 8
00023
00025 #define MAXFASTOBJECT 2
00026
00028 #define MAXCOMMAND 3
00029
00030 //The output modality chosen by the user with the "-m" flag when he launches the program.
00031 #define ALL 2
00032 #define ONLYUSER 0
00033 #define ONLYTCP 1

```

## 6.29 DigitizerErrorObject.cpp File Reference

```

#include "DigitizerErrorObject.h"
#include "DefineGeneral.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <CAENDigitizer.h>
#include <sys/time.h>
#include <time.h>

```

## 6.30 DigitizerErrorObject.cpp

```

00001
00005 #include "DigitizerErrorObject.h"
00006 #include "DefineGeneral.h"
00007 #include <stdio.h>
00008 #include <stdlib.h>
00009 #include <string.h>
00010 #include <CAENDigitizer.h>
00011 #include <sys/time.h>
00012 #include <time.h>
00013
00014 void
00015 DigitizerErrorObject::DigitizerErrorObjectDebugging (
    CAEN_DGTZ_ErrorCode
00016
00017     ritorno,
00018     const char *file,
00019     const char *func,
00020     int line)
00021 {
00022     switch (ritorno)
00023     {
00024     case (CAEN_DGTZ_Success):
00025         fprintf (stderr,
00026             "File: %s Function: %s Line: %d ErrorCode: %d Operation completed successful\n",
00027                 file, func, line, ritorno);

```

```
00028         break;
00029
00030     case (CAEN_DGTZ_CommError):
00031         fprintf (stderr,
00032             "File: %s Function: %s Line: %d ErrorCode: %d Communication error\n",
00033             file, func, line, ritorno);
00034         break;
00035
00036     case (CAEN_DGTZ_GenericError):
00037         fprintf (stderr,
00038             "File: %s Function: %s Line: %d ErrorCode: %d Unspecified error\n",
00039             file, func, line, ritorno);
00040         break;
00041
00042     case (CAEN_DGTZ_InvalidParam):
00043         fprintf (stderr,
00044             "File: %s Function: %s Line: %d ErrorCode: %d Invalid parameter\n",
00045             file, func, line, ritorno);
00046         break;
00047
00048     case (CAEN_DGTZ_InvalidLinkType):
00049         fprintf (stderr,
00050             "File: %s Function: %s Line: %d ErrorCode: %d Invalid Link Type\n",
00051             file, func, line, ritorno);
00052         break;
00053
00054     case (-5):
00055         fprintf (stderr,
00056             "File: %s Function: %s Line: %d ErrorCode: %d Invalid device handler\n",
00057             file, func, line, ritorno);
00058         break;
00059
00060     case (CAEN_DGTZ_MaxDevicesError):
00061         fprintf (stderr,
00062             "File: %s Function: %s Line: %d ErrorCode: %d Maximum number of devices exceeded\n",
00063             file, func, line, ritorno);
00064         break;
00065
00066     case (CAEN_DGTZ_BadBoardType):
00067         fprintf (stderr,
00068             "File: %s Function: %s Line: %d ErrorCode: %d Operation non allowed on this type of board\n",
00069             file, func, line, ritorno);
00070         break;
00071
00072     case (CAEN_DGTZ_BadInterruptLev):
00073         fprintf (stderr,
00074             "File: %s Function: %s Line: %d ErrorCode: %d The interrupt level is not allowed\n",
00075             file, func, line, ritorno);
00076         break;
00077
00078     case (CAEN_DGTZ_BadEventNumber):
00079         fprintf (stderr,
00080             "File: %s Function: %s Line: %d ErrorCode: %d The event number is bad\n",
00081             file, func, line, ritorno);
00082         break;
00083
00084     case (-10):
00085         fprintf (stderr,
00086             "File: %s Function: %s Line: %d ErrorCode: %d Unable to read the registry\n",
00087             file, func, line, ritorno);
00088         break;
00089
00090     case (CAEN_DGTZ_ChannelBusy):
00091         fprintf (stderr,
00092             "File: %s Function: %s Line: %d ErrorCode: %d The channel number is invalid\n",
00093             file, func, line, ritorno);
00094         break;
00095
00096     case (CAEN_DGTZ_FPIOModeInvalid):
00097         fprintf (stderr,
00098             "File: %s Function: %s Line: %d ErrorCode: %d Invalid FPIO mode\n",
00099             file, func, line, ritorno);
00100         break;
00101
00102     case (CAEN_DGTZ_WrongAcqMode):
00103         fprintf (stderr,
00104             "File: %s Function: %s Line: %d ErrorCode: %d Wrong acquisition mode\n",
00105             file, func, line, ritorno);
00106         break;
00107
00108     case (CAEN_DGTZ_FunctionNotAllowed):
00109         fprintf (stderr,
00110             "File: %s Function: %s Line: %d ErrorCode: %d This function is not allowed for this module\n",
00111             file, func, line, ritorno);
00112         break;
00113
00114     case (CAEN_DGTZ_Timeout):
```

```
00115     fprintf (stderr,
00116             "File: %s Function: %s Line: %d ErrorCode: %d Communication Timeout\n",
00117             file, func, line, ritorno);
00118     break;
00119
00120     case (CAEN_DGTZ_InvalidBuffer):
00121     fprintf (stderr,
00122             "File: %s Function: %s Line: %d ErrorCode: %d The buffer is invalid\n",
00123             file, func, line, ritorno);
00124     break;
00125
00126     case (CAEN_DGTZ_EventNotFound):
00127     fprintf (stderr,
00128             "File: %s Function: %s Line: %d ErrorCode: %d The event is not found\n",
00129             file, func, line, ritorno);
00130     break;
00131
00132     case (CAEN_DGTZ_InvalidEvent):
00133     fprintf (stderr,
00134             "File: %s Function: %s Line: %d ErrorCode: %d The event is invalid\n",
00135             file, func, line, ritorno);
00136     break;
00137
00138     case (CAEN_DGTZ_OutOfMemory):
00139     fprintf (stderr,
00140             "File: %s Function: %s Line: %d ErrorCode: %d Out of memory\n",
00141             file, func, line, ritorno);
00142     break;
00143
00144     case (CAEN_DGTZ_CalibrationError):
00145     fprintf (stderr,
00146             "File: %s Function: %s Line: %d ErrorCode: %d Unable to calibrate the board\n",
00147             file, func, line, ritorno);
00148     break;
00149
00150     case (CAEN_DGTZ_DigitizerNotFound):
00151     fprintf (stderr,
00152             "File: %s Function: %s Line: %d ErrorCode: %d Unable to open the digitizer\n",
00153             file, func, line, ritorno);
00154     break;
00155
00156     case (CAEN_DGTZ_DigitizerAlreadyOpen):
00157     fprintf (stderr,
00158             "File: %s Function: %s Line: %d ErrorCode: %d The Digitizer is already open\n",
00159             file, func, line, ritorno);
00160     break;
00161
00162     case (CAEN_DGTZ_DigitizerNotReady):
00163     fprintf (stderr,
00164             "File: %s Function: %s Line: %d ErrorCode: %d The Digitizer is not ready to operate\n",
00165             file, func, line, ritorno);
00166     break;
00167
00168     case (CAEN_DGTZ_InterruptNotConfigured):
00169     fprintf (stderr,
00170             "File: %s Function: %s Line: %d ErrorCode: %d The Digitizer has not the IRQ configured\n",
00171             file, func, line, ritorno);
00172     break;
00173
00174     case (CAEN_DGTZ_DigitizerMemoryCorrupted):
00175     fprintf (stderr,
00176             "File: %s Function: %s Line: %d ErrorCode: %d The Digitizer flash memory is corrupted\n",
00177             file, func, line, ritorno);
00178     break;
00179
00180     case (CAEN_DGTZ_DPPFirmwareNotSupported):
00181     fprintf (stderr,
00182             "File: %s Function: %s Line: %d ErrorCode: %d The digitizer DPP firmware is not supported in
this lib version\n",
00183             file, func, line, ritorno);
00184     break;
00185
00186     case (CAEN_DGTZ_InvalidLicense):
00187     fprintf (stderr,
00188             "File: %s Function: %s Line: %d ErrorCode: %d Invalid Firmware License\n",
00189             file, func, line, ritorno);
00190     break;
00191
00192     case (CAEN_DGTZ_InvalidDigitizerStatus):
00193     fprintf (stderr,
00194             "File: %s Function: %s Line: %d ErrorCode: %d The digitizer is found in a corrupted status\n",
00195             file, func, line, ritorno);
00196     break;
00197
00198     case (CAEN_DGTZ_UnsupportedTrace):
00199     fprintf (stderr,
00200             "File: %s Function: %s Line: %d ErrorCode: %d The given trace is not supported by the digitizer
```

```
\n",
00201         file, func, line, ritorno);
00202     break;
00203
00204     case (CAEN_DGTZ_InvalidProbe):
00205         fprintf (stderr,
00206             "File: %s Function: %s Line: %d ErrorCode: %d The given probe is not supported for the given
digitizer's trace\n",
00207             file, func, line, ritorno);
00208     break;
00209
00210     case (CAEN_DGTZ_NotYetImplemented):
00211         fprintf (stderr,
00212             "File: %s Function: %s Line: %d ErrorCode: %d The function is not yet implemented\n",
00213             file, func, line, ritorno);
00214     break;
00215 }
00216
00217 }
00218
00219 void
00220 DigitizerErrorObject::DigitizerErrorObjectPrintError
00221 (CAEN_DGTZ_ErrorCode ritorno)
00222 {
00223     switch (ritorno)
00224     {
00225
00226         case (CAEN_DGTZ_Success):
00227             fprintf (stderr, "ErrorCode: %d Operation completed successful\n",
00228                 ritorno);
00229             break;
00230
00231         case (CAEN_DGTZ_CommError):
00232             fprintf (stderr, "ErrorCode: %d Communication error\n", ritorno);
00233             break;
00234
00235         case (CAEN_DGTZ_GenericError):
00236             fprintf (stderr, "ErrorCode: %d Unspecified error\n", ritorno);
00237             break;
00238
00239         case (CAEN_DGTZ_InvalidParam):
00240             fprintf (stderr, "ErrorCode: %d Invalid parameter\n", ritorno);
00241             break;
00242
00243         case (CAEN_DGTZ_InvalidLinkType):
00244             fprintf (stderr, "ErrorCode: %d Invalid Link Type\n", ritorno);
00245             break;
00246
00247         case (-5):
00248             fprintf (stderr, "ErrorCode:%d Invalid device handler\n", ritorno);
00249             break;
00250
00251         case (CAEN_DGTZ_MaxDevicesError):
00252             fprintf (stderr, "ErrorCode: %d Maximum number of devices exceeded\n",
00253                 ritorno);
00254             break;
00255
00256         case (CAEN_DGTZ_BadBoardType):
00257             fprintf (stderr,
00258                 "ErrorCode: %d Operation non allowed on this type of board\n",
00259                 ritorno);
00260             break;
00261
00262         case (CAEN_DGTZ_BadInterruptLev):
00263             fprintf (stderr, "ErrorCode: %d The interrupt level is not allowed\n",
00264                 ritorno);
00265             break;
00266
00267         case (CAEN_DGTZ_BadEventNumber):
00268             fprintf (stderr, "ErrorCode: %d The event number is bad\n", ritorno);
00269             break;
00270
00271         case (-10):
00272             fprintf (stderr, "ErrorCode:%d Unable to read the registry\n", ritorno);
00273             break;
00274
00275         case (CAEN_DGTZ_ChannelBusy):
00276             fprintf (stderr, "ErrorCode: %d The channel number is invalid\n",
00277                 ritorno);
00278             break;
00279
00280         case (CAEN_DGTZ_FPIOModeInvalid):
00281             fprintf (stderr, "ErrorCode: %d Invalid FPIO mode\n", ritorno);
00282             break;
00283
00284         case (CAEN_DGTZ_WrongAcqMode):
00285             fprintf (stderr, "ErrorCode: %d Wrong acquisition mode\n", ritorno);
```

```
00286         break;
00287
00288     case (CAEN_DGTZ_FunctionNotAllowed):
00289         fprintf (stderr,
00290             "ErrorCode: %d This function is not allowed for this module\n",
00291             ritorno);
00292         break;
00293
00294     case (CAEN_DGTZ_Timeout):
00295         fprintf (stderr, "ErrorCode: %d Communication Timeout\n", ritorno);
00296         break;
00297
00298     case (CAEN_DGTZ_InvalidBuffer):
00299         fprintf (stderr, "ErrorCode: %d The buffer is invalid\n", ritorno);
00300         break;
00301
00302     case (CAEN_DGTZ_EventNotFound):
00303         fprintf (stderr, "ErrorCode: %d The event is not found\n", ritorno);
00304         break;
00305
00306     case (CAEN_DGTZ_InvalidEvent):
00307         fprintf (stderr, "ErrorCode: %d The event is invalid\n", ritorno);
00308         break;
00309
00310     case (CAEN_DGTZ_OutOfMemory):
00311         fprintf (stderr, "ErrorCode: %d Out of memory\n", ritorno);
00312         break;
00313
00314     case (CAEN_DGTZ_CalibrationError):
00315         fprintf (stderr, "ErrorCode: %d Unable to calibrate the board\n",
00316             ritorno);
00317         break;
00318
00319     case (CAEN_DGTZ_DigitizerNotFound):
00320         fprintf (stderr, "ErrorCode: %d Unable to open the digitizer\n",
00321             ritorno);
00322         break;
00323
00324     case (CAEN_DGTZ_DigitizerAlreadyOpen):
00325         fprintf (stderr, "ErrorCode: %d The Digitizer is already open\n",
00326             ritorno);
00327         break;
00328
00329     case (CAEN_DGTZ_DigitizerNotReady):
00330         fprintf (stderr,
00331             "ErrorCode: %d The Digitizer is not ready to operate\n",
00332             ritorno);
00333         break;
00334
00335     case (CAEN_DGTZ_InterruptNotConfigured):
00336         fprintf (stderr,
00337             "ErrorCode: %d The Digitizer has not the IRQ configured\n",
00338             ritorno);
00339         break;
00340
00341     case (CAEN_DGTZ_DigitizerMemoryCorrupted):
00342         fprintf (stderr,
00343             "ErrorCode: %d The Digitizer flash memory is corrupted\n",
00344             ritorno);
00345         break;
00346
00347     case (CAEN_DGTZ_DPPFirmwareNotSupported):
00348         fprintf (stderr,
00349             "ErrorCode: %d The digitizer DPP firmware is not supported in this lib version\n",
00350             ritorno);
00351         break;
00352
00353     case (CAEN_DGTZ_InvalidLicense):
00354         fprintf (stderr, "ErrorCode: %d Invalid Firmware License\n", ritorno);
00355         break;
00356
00357     case (CAEN_DGTZ_InvalidDigitizerStatus):
00358         fprintf (stderr,
00359             "ErrorCode: %d The digitizer is found in a corrupted status\n",
00360             ritorno);
00361         break;
00362
00363     case (CAEN_DGTZ_UnsupportedTrace):
00364         fprintf (stderr,
00365             "ErrorCode: %d The given trace is not supported by the digitizer\n",
00366             ritorno);
00367         break;
00368
00369     case (CAEN_DGTZ_InvalidProbe):
00370         fprintf (stderr,
00371             "ErrorCode: %d The given probe is not supported for the given digitizer's trace\n",
00372             ritorno);
```



```

00373         break;
00374
00375     case (CAEN_DGTZ_NotYetImplemented):
00376         fprintf (stderr, "ErrorCode: %d The function is not yet implemented\n",
00377                 ritorno);
00378         break;
00379     }
00380 }
00381
00382
00383 int
00384 DigitizerErrorObject::DigitizerErrorObjectDebuggingLog
00385 (CAEN_DGTZ_ErrorCode
00386      ritorno,
00387      const char *file,
00388      const char *func,
00389      int line,
00390      FILE * log_file)
00391 {
00392     char ts[100];
00393     time_t t;
00394     t = time (NULL);
00395     ctime_r (&t, ts);
00396     ts[strlen (ts) - 1] = '\0';
00397
00398     switch (ritorno)
00399     {
00400
00401     case (CAEN_DGTZ_Success):
00402         fprintf (log_file,
00403                 "%s File: %s Function: %s Line: %d ErrorCode: %d Operation completed successfully\n",
00404                 ts, file, func, line, ritorno);
00405         fflush (log_file);
00406         return 0;
00407         break;
00408
00409     case (CAEN_DGTZ_CommError):
00410         fprintf (log_file,
00411                 "%s File: %s Function: %s Line: %d ErrorCode: %d Communication error\n",
00412                 ts, file, func, line, ritorno);
00413         fflush (log_file);
00414         return 1;
00415         break;
00416
00417     case (CAEN_DGTZ_GenericError):
00418         fprintf (log_file,
00419                 "%s File: %s Function: %s Line: %d ErrorCode: %d Unspecified error\n",
00420                 ts, file, func, line, ritorno);
00421         fflush (log_file);
00422         return 1;
00423         break;
00424
00425     case (CAEN_DGTZ_InvalidParam):
00426         fprintf (log_file,
00427                 "%s File: %s Function: %s Line: %d ErrorCode: %d Invalid parameter\n",
00428                 ts, file, func, line, ritorno);
00429         fflush (log_file);
00430         return 1;
00431         break;
00432
00433     case (CAEN_DGTZ_InvalidLinkType):
00434         fprintf (log_file,
00435                 "%s File: %s Function: %s Line: %d ErrorCode: %d Invalid Link Type\n",
00436                 ts, file, func, line, ritorno);
00437         fflush (log_file);
00438         return 1;
00439         break;
00440
00441     case (-5):
00442         fprintf (log_file,
00443                 "%s File: %s Function: %s Line: %d ErrorCode: %d Invalid device handler\n",
00444                 ts, file, func, line, ritorno);
00445         fflush (log_file);
00446         return 1;
00447         break;
00448
00449     case (CAEN_DGTZ_MaxDevicesError):
00450         fprintf (log_file,
00451                 "%s File: %s Function: %s Line: %d ErrorCode: %d Maximum number of devices exceeded\n",
00452                 ts, file, func, line, ritorno);
00453         fflush (log_file);
00454         return 1;
00455         break;
00456
00457     case (CAEN_DGTZ_BadBoardType):
00458         fprintf (log_file,

```

```
00459         "%s File: %s Function: %s Line: %d ErrorCode: %d Operation non allowed on this type of board\n",
00460         ts, file, func, line, ritorno);
00461     fflush (log_file);
00462     return 1;
00463     break;
00464
00465     case (CAEN_DGTZ_BadInterruptLev):
00466         fprintf (log_file,
00467             "%s File: %s Function: %s Line: %d ErrorCode: %d The interrupt level is not allowed\n",
00468             ts, file, func, line, ritorno);
00469         fflush (log_file);
00470         return 1;
00471         break;
00472
00473     case (CAEN_DGTZ_BadEventNumber):
00474         fprintf (log_file,
00475             "%s File: %s Function: %s Line: %d ErrorCode: %d The event number is bad\n",
00476             ts, file, func, line, ritorno);
00477         fflush (log_file);
00478         return 1;
00479         break;
00480
00481     case (-10):
00482         fprintf (log_file,
00483             "%s File: %s Function: %s Line: %d ErrorCode: %d Unable to read the registry\n",
00484             ts, file, func, line, ritorno);
00485         fflush (log_file);
00486         return 1;
00487         break;
00488
00489     case (CAEN_DGTZ_ChannelBusy):
00490         fprintf (log_file,
00491             "%s File: %s Function: %s Line: %d ErrorCode: %d The channel number is invalid\n",
00492             ts, file, func, line, ritorno);
00493         fflush (log_file);
00494         return 1;
00495         break;
00496
00497     case (CAEN_DGTZ_FPIOModeInvalid):
00498         fprintf (log_file,
00499             "%s File: %s Function: %s Line: %d ErrorCode: %d Invalid FPIO mode\n",
00500             ts, file, func, line, ritorno);
00501         fflush (log_file);
00502         return 1;
00503         break;
00504
00505     case (CAEN_DGTZ_WrongAcqMode):
00506         fprintf (log_file,
00507             "%s File: %s Function: %s Line: %d ErrorCode: %d Wrong acquisition mode\n",
00508             ts, file, func, line, ritorno);
00509         fflush (log_file);
00510         return 1;
00511         break;
00512
00513     case (CAEN_DGTZ_FunctionNotAllowed):
00514         fprintf (log_file,
00515             "%s File: %s Function: %s Line: %d ErrorCode: %d This function is not allowed for this module\n",
00516             ts, file, func, line, ritorno);
00517         fflush (log_file);
00518         return 1;
00519         break;
00520
00521     case (CAEN_DGTZ_Timeout):
00522         fprintf (log_file,
00523             "%s File: %s Function: %s Line: %d ErrorCode: %d Communication Timeout\n",
00524             ts, file, func, line, ritorno);
00525         fflush (log_file);
00526         return 1;
00527         break;
00528
00529     case (CAEN_DGTZ_InvalidBuffer):
00530         fprintf (log_file,
00531             "%s File: %s Function: %s Line: %d ErrorCode: %d The buffer is invalid\n",
00532             ts, file, func, line, ritorno);
00533         fflush (log_file);
00534         return 1;
00535         break;
00536
00537     case (CAEN_DGTZ_EventNotFound):
00538         fprintf (log_file,
00539             "%s File: %s Function: %s Line: %d ErrorCode: %d The event is not found\n",
00540             ts, file, func, line, ritorno);
00541         fflush (log_file);
00542         return 1;
00543         break;
00544
```

```
00545     case (CAEN_DGTZ_InvalidEvent):
00546         fprintf (log_file,
00547             "%s File: %s Function: %s Line: %d ErrorCode: %d The event is invalid\n",
00548             ts, file, func, line, ritorno);
00549         fflush (log_file);
00550         return 1;
00551     break;
00552
00553     case (CAEN_DGTZ_OutOfMemory):
00554         fprintf (log_file,
00555             "%s File: %s Function: %s Line: %d ErrorCode: %d Out of memory\n",
00556             ts, file, func, line, ritorno);
00557         fflush (log_file);
00558         return 1;
00559     break;
00560
00561     case (CAEN_DGTZ_CalibrationError):
00562         fprintf (log_file,
00563             "%s File: %s Function: %s Line: %d ErrorCode: %d Unable to calibrate the board\n",
00564             ts, file, func, line, ritorno);
00565         fflush (log_file);
00566         return 1;
00567     break;
00568
00569     case (CAEN_DGTZ_DigitizerNotFound):
00570         fprintf (log_file,
00571             "%s File: %s Function: %s Line: %d ErrorCode: %d Unable to open the digitizer\n",
00572             ts, file, func, line, ritorno);
00573         fflush (log_file);
00574         return 1;
00575     break;
00576
00577     case (CAEN_DGTZ_DigitizerAlreadyOpen):
00578         fprintf (log_file,
00579             "%s File: %s Function: %s Line: %d ErrorCode: %d The Digitizer is already open\n",
00580             ts, file, func, line, ritorno);
00581         fflush (log_file);
00582         return 1;
00583     break;
00584
00585     case (CAEN_DGTZ_DigitizerNotReady):
00586         fprintf (log_file,
00587             "%s File: %s Function: %s Line: %d ErrorCode: %d The Digitizer is not ready to operate\n",
00588             ts, file, func, line, ritorno);
00589         fflush (log_file);
00590         return 1;
00591     break;
00592
00593     case (CAEN_DGTZ_InterruptNotConfigured):
00594         fprintf (log_file,
00595             "%s File: %s Function: %s Line: %d ErrorCode: %d The Digitizer has not the IRQ configured\n",
00596             ts, file, func, line, ritorno);
00597         fflush (log_file);
00598         return 1;
00599     break;
00600
00601     case (CAEN_DGTZ_DigitizerMemoryCorrupted):
00602         fprintf (log_file,
00603             "%s File: %s Function: %s Line: %d ErrorCode: %d The Digitizer flash memory is corrupted\n",
00604             ts, file, func, line, ritorno);
00605         fflush (log_file);
00606         return 1;
00607     break;
00608
00609     case (CAEN_DGTZ_DPPFirmwareNotSupported):
00610         fprintf (log_file,
00611             "%s File: %s Function: %s Line: %d ErrorCode: %d The digitizer DPP firmware is not supported in
00612             this lib version\n",
00613             ts, file, func, line, ritorno);
00614         fflush (log_file);
00615         return 1;
00616     break;
00617
00618     case (CAEN_DGTZ_InvalidLicense):
00619         fprintf (log_file,
00620             "%s File: %s Function: %s Line: %d ErrorCode: %d Invalid Firmware License\n",
00621             ts, file, func, line, ritorno);
00622         fflush (log_file);
00623         return 1;
00624     break;
00625
00626     case (CAEN_DGTZ_InvalidDigitizerStatus):
00627         fprintf (log_file,
00628             "%s File: %s Function: %s Line: %d ErrorCode: %d The digitizer is found in a corrupted status\n",
00629             ts, file, func, line, ritorno);
00629         fflush (log_file);
```

```

00630         return 1;
00631         break;
00632
00633         case (CAEN_DGTZ_UnsupportedTrace):
00634             fprintf (log_file,
00635                 "%s File: %s Function: %s Line: %d ErrorCode: %d The given trace is not supported by the
digitizer\n",
00636                 ts, file, func, line, ritorno);
00637             fflush (log_file);
00638             return 1;
00639             break;
00640
00641         case (CAEN_DGTZ_InvalidProbe):
00642             fprintf (log_file,
00643                 "%s File: %s Function: %s Line: %d ErrorCode: %d The given probe is not supported for the given
digitizer's trace\n",
00644                 ts, file, func, line, ritorno);
00645             fflush (log_file);
00646             return 1;
00647             break;
00648
00649         case (CAEN_DGTZ_NotYetImplemented):
00650             fprintf (log_file,
00651                 "%s File: %s Function: %s Line: %d ErrorCode: %d The function is not yet implemented\n",
00652                 ts, file, func, line, ritorno);
00653             fflush (log_file);
00654             return 1;
00655             break;
00656     }
00657
00658 }

```

## 6.31 DigitizerErrorObject.h File Reference

```

#include <stdio.h>
#include <CAENDigitizer.h>

```

### Classes

- class [DigitizerErrorObject](#)

The [DigitizerErrorObject](#) class contains two methods that print the meaning of the CAEN\_DGTZ\_ErrorCode.

## 6.32 DigitizerErrorObject.h

```

00001
00005 #include <stdio.h>
00006 #include <CAENDigitizer.h>
00007
00013 class DigitizerErrorObject
00014 {
00015 public:
00016
00026 void DigitizerErrorObjectDebugging (CAEN_DGTZ_ErrorCode ritorno,
00027                                     const char *file, const char *func,
00028                                     int line);
00034 void DigitizerErrorObjectPrintError (CAEN_DGTZ_ErrorCode ritorno);
00035
00036
00046 int DigitizerErrorObjectDebuggingLog (CAEN_DGTZ_ErrorCode ritorno,
00047                                       const char *file, const char *func,
00048                                       int line, FILE * log_file);
00049 };

```

## 6.33 DigitizerFlowControl.cpp File Reference

```

#include "DefineGeneral.h"

```

```

#include "DefineCommands.h"
#include "TcpUser.h"
#include "ConfObject.h"
#include "DigitizerErrorObject.h"
#include "LogFile.h"
#include "DigitizerObject.h"
#include "DigitizerObjectGeneric.h"
#include "RawData.h"
#include "DigitizerStateMachine.h"
#include "ApplicationSetup.h"
#include "Input.h"
#include "CommunicationObject.h"
#include "DigitizerFlowControl.h"
#include "ConfigurationConsistence.h"
#include "AnalizzatoreUtils.h"
#include "Analizzatore.h"
#include "OutputModule.h"
#include <CAENDigitizer.h>
#include <stdio.h>
#include <regex.h>
#include <stdbool.h>

```

## 6.34 DigitizerFlowControl.cpp

```

00001
00006 #include "DefineGeneral.h"
00007 #include "DefineCommands.h"
00008 #include "TcpUser.h"
00009 #include "ConfObject.h"
00010 #include "DigitizerErrorObject.h"
00011 #include "LogFile.h"
00012 #include "DigitizerObject.h"
00013 #include "DigitizerObjectGeneric.h"
00014 #include "RawData.h"
00015 #include "DigitizerStateMachine.h"
00016 #include "ApplicationSetup.h"
00017 #include "Input.h"
00018 #include "CommunicationObject.h"
00019 #include "DigitizerFlowControl.h"
00020 #include "ConfigurationConsistence.h"
00021 #include "AnalizzatoreUtils.h"
00022 #include "Analizzatore.h"
00023 #include "OutputModule.h"
00024 #include <CAENDigitizer.h>
00025 #include <stdio.h>
00026 #include <regex.h>
00027 #include <stdbool.h>
00028
00029 pthread_cond_t DigitizerFlowControl::input_flow_cond;
00030
00031 pthread_mutex_t DigitizerFlowControl::input_flow_mutex;
00032
00033 // L'oggetto di tipo DigitizerFlowControl viene inizializzato ottenendo un riferimento all'oggetto
    ApplicationSetup.
00034 DigitizerFlowControl::DigitizerFlowControl ()
00035 {
00036     pthread_mutex_init (&input_flow_mutex, NULL);
00037     pthread_cond_init (&input_flow_cond, NULL);
00038     digitizer_flow_control_application_setup =
    ApplicationSetup::Instance ();
00039 }
00040
00041
00042 void
00043 DigitizerFlowControl::DigitizerFlowControlStart ()
00044 {
00045
00046 //Viene ottenuto il riferimento ad OutputModule.
00047     OutputModule *output_module;
00048     output_module = OutputModule::Instance ();
00049
00050 //E' ottenuto un riferimento al singleton LogFile. LogFile permette di stampare (in modo verboso) il

```

```

        risultato delle funzioni della libreria CAENDigitizer nel file di log
00051 //indicato dal flag -l [logfilepath]. Se l'utente non ne indica uno, viene usato quello di default. Il
        contenuto del log file puo' essere visualizzato digitando il comando "more".
00052 //Questo permette di controllare rapidamente se gli altri comandi che interagiscono col digitizer hanno
        avuto successo.
00053     LogFile *logfile;
00054     logfile = LogFile::Instance ();
00055
00056 //Variabili di supporto
00057     char *my_punt;
00058     char *conf_file_string;
00059     char *data_file_string;
00060     char *log_file_string;
00061     int i;
00062     int Address = 0;
00063     int Data = 0;
00064     char support_string[STANDARDBUFFERLIMIT];
00065     bzero (support_string, STANDARDBUFFERLIMIT);
00066     int support_flag1 = 0;
00067
00068 //Impostazione del log file. Il metodo LogFileSet apre il log file indicato dal flag -l [logfilepath] o
        quello di default.
00069     logfile->LogFileSet (digitizer_flow_control_application_setup
->
        application_setup_log_file_path);
00070
00071
00072 //L'oggetto di tipo ConfigurationConsistence serve per controllare se nel file di configurazione sono
        presenti errori gravi (come l'omissione dell'attributo OPEN).
00073 //Questo tipo di controllo e' possibile farlo con il comando "check".
00074     ConfigurationConsistence my_consistence;
00075
00076 //Se il metodo ConfigurationConsistenceConfFileSetupEssentialWithInitCheckNoPrint torna un valore diverso
        da zero vuol dire che ha trovato un errore nel file di configurazione.
00077     if (my_consistence.
        ConfigurationConsistenceConfFileSetupEssentialWithInitCheckNoPrint
        (digitizer_flow_control_application_setup->
        application_setup_conf_file_path))
00078     {
00079         fprintf (stderr, "%s: Some error detected in %s config file\n",
00080                 __func__,
00081                 digitizer_flow_control_application_setup->
        application_setup_conf_file_path);
00082     }
00083
00084 //DigitizerStateMachine e' il tipo di oggetto che comunica col digitizer. E', insieme a
        DigitizerFlowControl, il tipo di oggetto piu' delicato perche' lancia e gestisce
00085 //il thread che comunica col digitizer e quelli che trattano i dati letti.
00086     DigitizerStateMachine miodigitizer;
00087
00088 //buffer contiene il comando fetchato da stdin o da tcp.
00089     char *buffer;
00090     buffer = (char *) malloc (STANDARDBUFFERLIMIT);
00091     TcpUser command_received_by_user;
00092
00093 //A seconda della modalita' di interazione scelta dall'utente (-m [all|user|tcp]) occorre inizializzare gli
        oggetti per fetchare l'input.
00094
00095 //Se l'utente ha lanciato il programma con il flag -m all, vuol dire che l'input puo' essere inviato sia da
        stdin sia da tcp.
00096     if (digitizer_flow_control_application_setup->
        input_mode == ALL)
00097     {
00098
00099 //Inizializzo CommunicationObject che raccoglie l'input via TCP.
00100         mioTCP.CommunicationObjectInit ();
00101
00102     } //if (digitizer_flow_control_application_setup->input_mode == ALL)
00103
00104 //Se l'utente ha lanciato il programma con il flag -m user, vuol dire che l'input puo' essere inviato solo
        da stdin.
00105     else if (digitizer_flow_control_application_setup->
        input_mode == ONLYUSER)
00106     {
00107
00108     } //if (digitizer_flow_control_application_setup->input_mode == ONLYUSER)
00109
00110 //Se l'utente ha lanciato il programma con il flag -m tcp, vuol dire che l'input puo' essere inviato solo
        via TCP.
00111     else if (digitizer_flow_control_application_setup->
        input_mode == ONLYTCP)
00112     {
00113 //Inizializzo CommunicationObject che raccoglie l'input via TCP.
00114         mioTCP.CommunicationObjectInit ();
00115     } //if (digitizer_flow_control_application_setup->input_mode == ONLYTCP)
00116
00117 //Se il programma e' stato eseguito in una modalita' diversa da ONLYTCP, stampo la stringa di benvenuto.
00118 //I controlli vengono fatti dai singleton StdOut e TcpUserArray.

```

```

00119     output_module->Output("objectDump, press help to get the available command list\n");
00120
00121 //Inizia il ciclo di fetch-decode-execute-writeback del programma.
00122     while (1)
00123     {
00124
00125 //Se il programma e' stato lanciato in una modalita' diversa da quella ONLYTCP e' effettuato un flush del
00126 //stdout per stampare l'eventuale output bufferizzato.
00127 //Per evitare la bufferizzazione e' anche possibile stampare su stderr invece che su stdout, ma
00128 //risulterebbe essere un utilizzo non appropriato di stderr.
00129         if (digitizer_flow_control_application_setup->
00130             input_mode != ONLYTCP)
00131         {
00132             fflush (stdout);
00133         }
00134
00135 //Settiamo a zero buffer: in esso viene salvato il contenuto dell'input ottenuto dagli oggetti di tipo
00136 //CommunicationObject (per il TCP) e Input (per l'stdout).
00137         bzero (buffer, STANDARDBUFFERLIMIT);
00138
00139 //Azioni da intraprende per fetchare l'input se il programma e' stato lanciato sia in modalita' tcp che in
00140 //modalita' user
00141         if (digitizer_flow_control_application_setup->
00142             input_mode == ALL)
00143         {
00144             //Chiedo l'input da stdin. Se non c'e' il programma prosegue e in buffer e' salvata la stringa vuota.
00145             mioinput.GetInput (buffer);
00146
00147             //Se e' stato ricevuto un input dall'stdin, e' stampata una stringa che lo riproduce.
00148             if (strlen (buffer) != 0)
00149             {
00150                 output_module->OutputModuleStdoutOn();
00151                 output_module->OutputModuleSockidOff();
00152             }
00153
00154             //Chiedo l'input inviato da TCP.
00155             command_received_by_user = mioTCP.GetCommand ();
00156
00157             //Se e' stato ricevuto un input via tcp.
00158             if (command_received_by_user.command_sent_by_user != 0)
00159             {
00160                 //E' stampata una stringa che scrive l'input ricevuto e il sockid di chi l'ha inviato.
00161                 //Questo avviene solo se il programma e' lanciato in modalita' -m all: se e' lanciato in modalita' -m tcp,
00162                 //esso e' eseguito in modalita' demone.
00163                 //Di conseguenza, in quest'ultimo caso non stampa niente a video.
00164                 fprintf (stdout, "Command sent by user %d: %d which means %s\n",
00165                     command_received_by_user.user_sockid,
00166                     command_received_by_user.command_sent_by_user,
00167                     DigitizerFlowControl::
00168                     ParseCommand (command_received_by_user.
00169                         command_sent_by_user));
00170
00171                 output_module->OutputModuleStdoutOff();
00172                 output_module->OutputModuleSockidOn(command_received_by_user.
00173                     user_sockid);
00174             }
00175         } //if (digitizer_flow_control_application_setup->input_mode == ALL)
00176
00177 //Azioni da intraprende per fetchare l'input se il programma e' stato lanciato in modalita' user.
00178         else if (digitizer_flow_control_application_setup->
00179             input_mode ==
00180                 ONLYUSER)
00181         {
00182             //Chiedo l'input dall'stdin. Se non c'e' il programma prosegue e in buffer e' salvata la stringa vuota.
00183             mioinput.GetInput (buffer);
00184
00185             //Se e' stato ricevuto un input dall'stdin, e' stampata una stringa che lo riproduce.
00186             if (strlen (buffer) != 0)
00187             {
00188                 output_module->OutputModuleStdoutOn();
00189                 output_module->OutputModuleSockidOff();
00190             }
00191         } //if (digitizer_flow_control_application_setup->input_mode == ONLYUSER)
00192
00193 //Azioni da intraprende per fetchare l'input se il programma e' stato lanciato in modalita' tcp.
00194         else if (digitizer_flow_control_application_setup->
00195             input_mode ==
00196                 ONLYTCP)

```

```

00202     {
00203
00204     //Chiedo l'input inviato da TCP.
00205         command_received_by_user = mioTCP.GetCommand ();
00206
00207     //Se e' stato ricevuto un input via TCP.
00208         if (command_received_by_user.command_sent_by_user != 0)
00209         {
00210
00212             output_module->OutputModuleStdoutOff();
00213             output_module->OutputModuleSockidOn(command_received_by_user.
user_sockid);
00215         }
00216     }
00217
00218     } //if (digitizer_flow_control_application_setup->input_mode == ONLYTCP)
00219
00220     pthread_mutex_lock (&input_flow_mutex);
00221
00222     //Se e' stato ricevuto un input via tcp o via stdin.
00223     if (strlen (buffer) != 0
00224         || command_received_by_user.command_sent_by_user != 0)
00225     {
00226         //Se e' stato ricevuto il comando di exit o di quit
00227         if (DigitizerFlowControl::reg_matches (buffer, "[Ee][Xx][iI][tT][
\t]*$") || DigitizerFlowControl::reg_matches (buffer, "[qQ][uU][iI][tT]$")
00228             || command_received_by_user.command_sent_by_user == QUIT) //ll==quit
00229         {
00230             output_module->OutputModuleStdoutOn();
00232
00233             //Spendo CommunicationObject
00234             if (digitizer_flow_control_application_setup->
input_mode !=
00235                 ONLYUSER)
00236                 mioTCP.Finish ();
00237
00238             //Spendo Input.
00239             if (digitizer_flow_control_application_setup->
input_mode !=
00240                 ONLYTCP)
00241                 mioinput.Finish ();
00242
00243             //Spendo DigitizerStateMachine.
00244             miogdigitizer.DigitizerStateMachineQuit ();
00245             return;
00246         }
00247
00248         //Se e' stato ricevuto il comando init.
00249         else if (DigitizerFlowControl::
reg_matches (buffer, "[iI][Nn][iI][tT][
\t]*$")
00250             || command_received_by_user.command_sent_by_user ==
INIT)
00251         {
00252             //Controllo se il file di configurazione contiene l'attributo OPEN e se e' ben settato. Il
NoPrint del nome del metodo indica che non viene stampato a video
00254             //il parametro OPEN del file di configurazione.
00255             if (!my_consistence.
ConfigurationConsistenceConfFileInitNoPrint
00256                 (digitizer_flow_control_application_setup->
application_setup_conf_file_path))
00257             {
00258                 //Se non ci sono errori nell'attributo OPEN del file di configurazione, il digitizer viene
aperto.
00261                 output_module->Output ("Digitizer initialization...\n");
00262
00263                 logfile->LogFileMessageOn ();
00264
00265                 miogdigitizer.
DigitizerStateMachineInit
00266                     (digitizer_flow_control_application_setup->
application_setup_conf_file_path);
00269                 logfile->LogFileMessageOff();
00271             }
00272             else
00273             {
00274                 //Altrimenti non viene intrapresa nessuna azione
00275                 output_module->Output ("Some error occurred, I cannot execute init command\n");
00276             }
00277         }
00278
00279         //Se e' stato ricevuto il comando setup.
00280         else if (DigitizerFlowControl::
reg_matches (buffer, "[Ss][Ee][Tt][Uu][Pp][
\t]*$")
00281             || command_received_by_user.command_sent_by_user ==
SETUP)
00282         {
00283

```



```

00284         //Controllo se il file di configurazione contiene i parametri fondamentali per settare il
digitizer.
00285         if (!my_consistence.
00286             ConfigurationConsistenceConfFileSetupEssentialWithInitCheckNoPrint
00287             (digitizer_flow_control_application_setup->
00288                 application_setup_conf_file_path))
00289         {
00290             //Se non ci sono errori nell'attributo OPEN del file di configurazione, il digitizer viene
impostato.
00291             output_module->Output("Digitizer setupping...\n");
00292
00293             logfile->LogFileMessageOn();
00294
00295             mioidigitizer.
00296                 DigitizerStateMachineSetup
00297                 (digitizer_flow_control_application_setup->
00298                     application_setup_conf_file_path);
00299
00300             logfile->LogFileMessageOff();
00301         }
00302     }
00303
00304     //Se e' stato ricevuto il comando start.
00305     else if (DigitizerFlowControl::
00306         reg_matches (buffer, "[sS][tT][aA][rR][tT][ \\t]*$")
00307         || command_received_by_user.command_sent_by_user ==
START)
00308     {
00309         //Il programma inizia ad acquisire dati dal digitizer. Di default nessun thread di trattamento
dei dati e' attivo.
00310         output_module->Output("Data acquisition starting...\n");
00311
00312         mioidigitizer.DigitizerStateMachineStartReading ();
00313
00314     }
00315
00316     //Se e' stato ricevuto il comando stop.
00317     else if (DigitizerFlowControl::
00318         reg_matches (buffer, "[sS][tT][oO][pP][ \\t]*$")
00319         || command_received_by_user.command_sent_by_user ==
STOP)
00320     {
00321         //Il programma termina l'acquisizione dei dati dal digitizer.
00322         output_module->Output("Stopping data acquisition...\n");
00323
00324         mioidigitizer.DigitizerStateMachineStopReading ();
00325
00326     }
00327
00328     //Se e' stato ricevuto il comando prestart.
00329     else if (DigitizerFlowControl::
00330         reg_matches (buffer, "[pP][rR][eE][sS][tT][aA][rR][tT][ \\t]*$")
00331         || command_received_by_user.command_sent_by_user ==
PRESTART)
00332     {
00333         //Il programma inizia il preprocessing dei dati raccolti.
00334         output_module->Output("Starting Thread Preprocessing...\n");
00335         mioidigitizer.DigitizerStateStartPreprocessing ();
00336     }
00337
00338     //Se e' stato ricevuto il comando prestop.
00339     else if (DigitizerFlowControl::
00340         reg_matches (buffer, "[pP][rR][eE][sS][tT][oO][pP][ \\t]*$")
00341         || command_received_by_user.command_sent_by_user ==
PRESTOP)
00342     {
00343         //Il programma termina il preprocessing dei dati raccolti.
00344         output_module->Output("Stopping Thread Preprocessing...\n");
00345         mioidigitizer.DigitizerStateStopPreprocessing ();
00346     }
00347
00348     //Se e' stato ricevuto il comando rawstart.
00349     else if (DigitizerFlowControl::
00350         reg_matches (buffer, "[rR][aA][wW][sS][tT][aA][rR][tT][ \\t]*$")
00351         || command_received_by_user.command_sent_by_user ==
RAWSTART)
00352     {
00353         //Il programma inizia la scrittura su disco dei dati raccolti.
00354         output_module->Output("Starting Thread RawDataWriting...\n");
00355         mioidigitizer.DigitizerStateStartRawDataWriting ();
00356     }
00357
00358     //Se e' stato ricevuto il comando rawstop.
00359     else if (DigitizerFlowControl::
00360         reg_matches (buffer, "[rR][aA][wW][sS][tT][oO][pP][ \\t]*$")
00361         || command_received_by_user.command_sent_by_user ==
RAWSTOP)
00362     {
00363         //Il programma termina la scrittura su disco dei dati raccolti.
00364         output_module->Output("Stopping Thread RawDataWriting...\n");
00365         mioidigitizer.DigitizerStateStopRawDataWriting ();
00366     }

```

```

00366     {
00367         //Il programma termina la scrittura su disco dei dati raccolti.
00368         output_module->Output("Stopping Thread RawDataWriting...\n");
00369         mioidigitizer.DigitizerStateStopRawDataWriting ();
00370     }
00371
00372     //Se e' stato ricevuto il comando vistart.
00373     else if (DigitizerFlowControl::
00374         reg_matches (buffer, "^[Vv][Ii][Ss][Tt][Aa][Rr][Tt][ \t]+[0-9]{1,3}[ \t]*$"))
00375     {
00376         digitizer_flow_control_application_setup ->
channel_visualized = FindIntegerValue(buffer);
00377         //Il programma inizia la visualizzazione dei dati raccolti.
00378         output_module->Output("Starting Thread Visualization...\n");
00379         mioidigitizer.DigitizerStateStartVisualization ();
00380     }
00381
00382     //Se e' stato ricevuto il comando vistop.
00383     else if (DigitizerFlowControl::
00384         reg_matches (buffer, "^[Vv][Ii][Ss][Tt][Oo][Pp][ \t]*$"))
00385     {
00386         //Il programma termina la visualizzazione dei dati raccolti.
00387         output_module->Output("Stopping Thread Visualization...\n");
00388         mioidigitizer.DigitizerStateStopVisualization ();
00389     }
00390
00391     //Se e' stato ricevuto il comando send.
00392     else if (DigitizerFlowControl::
00393         reg_matches (buffer, "^[Ss][Ee][Nn][Dd][ \t]*$"))
00394         || command_received_by_user.command_sent_by_user ==
SEND)
00395     {
00396         //Il programma invia un software trigger
00397         output_module->Output("Sending software trigger...\n");
00398         mioidigitizer.DigitizerStateMachineSendSWTrigger ();
00399     }
00400
00401     //Se e' stato ricevuto il comando close
00402     else if (DigitizerFlowControl::
00403         reg_matches (buffer, "^[cC][lL][oO][sS][eE][ \t]*$"))
00404         || command_received_by_user.command_sent_by_user ==
CLOSE)
00405     {
00406         //Il programma chiude il digitizer
00407         output_module->Output("Closing Digitizer...\n");
00408         mioidigitizer.DigitizerStateMachineQuit ();
00409     }
00410
00411     //Se e' stato ricevuto il comando print.
00412     else if (DigitizerFlowControl::
00413         reg_matches (buffer, "^[pP][rR][iI][nN][tT][ \t]*$"))
00414         || command_received_by_user.command_sent_by_user ==
PRINT)
00415     {
00416         //Il programma stampa in formato comprensibile le informazioni contenuto nell'oggetto di tipo
ConfObject che viene usato per configurare il
00417         //digitizer. Le informazioni contenuto nell'oggetto sono prese dal file di configurazione con i
comandi init (attributo open) e setup (tutti gli
00418         //altri).
00419         output_module->Output("Printing ConfObject parameters in human readable format...\n");
00420         mioidigitizer.digitizer.internal_config.
PrintAllHuman ();
00421     }
00422
00423     //Se e' stato ricevuto il comando print files.
00424     else if (DigitizerFlowControl::
00425         reg_matches (buffer,
00426             "^[pP][rR][iI][nN][tT][ \t]+[Ff][Ii][Ll][Ee][Ss][ \t]*$"))
00427         || command_received_by_user.command_sent_by_user ==
PRINTFILES)
00428     {
00429         //Viene stampato il path dei file di configurazione, di log e di salvataggio dei rawdata.
00430
00431         //Path del file di configurazione.
00432         if (digitizer_flow_control_application_setup->
application_setup_conf_file_path != NULL)
00433         {
00434             output_module->Output(digitizer_flow_control_application_setup
->
00437             application_setup_conf_file_path);
00438             output_module->Output("\n");
00439         }
00440
00441         //Path del file di salvataggio dei rawdata.
00442         if (digitizer_flow_control_application_setup->
application_setup_data_file_path != NULL)
00443         {
00444

```

```

00445         output_module->Output (digitizer_flow_control_application_setup
->
00446             application_setup_data_file_path);
00447         output_module->Output ("\n");
00448     }
00449
00450     //Path del file di log.
00451     if (digitizer_flow_control_application_setup->
00452         application_setup_log_file_path != NULL)
00453     {
00454         output_module->Output (digitizer_flow_control_application_setup
->
00455             application_setup_log_file_path);
00456         output_module->Output ("\n");
00457     }
00458 }
00459
00460 //Se e' stato ricevuto il comando help (solo da stdin perche' i clients connessi via tcp hanno il
loro comando help).
00461 else if (DigitizerFlowControl::
00462     reg_matches (buffer, "^[hH][eE][lL][pP][\t]*$"))
00463 {
00464     //Viene stampata la lista dei comandi disponibili.
00465     printf ("Listing commands available...\n");
00466     DigitizerFlowControl::Help ();
00467 }
00468
00469 //Se e' stato ricevuto il comando -f [pathdellogfile]
00470 else if (DigitizerFlowControl::
00471     reg_matches (buffer, "^[-][Ff][\t]*$"))
00472     || command_received_by_user.command_sent_by_user ==
00473     CHANGECONF)
00474 {
00475     //Se il comando e' stato inviato da tcp, viene ricavato il path del file di configurazione dalla
stringa contenuta in buffer.
00476     if (command_received_by_user.command_sent_by_user ==
CHANGECONF)
00477     {
00478         output_module->Output ("Changing conf file...\n");
00479         free ((void *) digitizer_flow_control_application_setup->
00480             application_setup_conf_file_path);
00481         digitizer_flow_control_application_setup->
00482             application_setup_conf_file_path = NULL;
00483         digitizer_flow_control_application_setup->
00484             application_setup_conf_file_path =
00485             (char *)
00486             malloc (strlen (command_received_by_user.first_parameter)
00487                 + 1);
00488         strcpy ((char *) digitizer_flow_control_application_setup
->
00489             application_setup_conf_file_path,
00490             command_received_by_user.first_parameter);
00491     }
00492     else
00493     {
00494         //Se il comando e' stato inviato da stdin, viene ricavato il path del file di configurazione
dalla stringa contenuta in buffer.
00495         //In questo caso la stringa e' un po' diversa rispetto a quella del caso precedente: per questo
e' necessario distinguere le procedure
00496         //di ottenimento della stringa.
00497         my_punt = FindPointer (buffer);
00498         output_module->Output ("Changing conf file...\n");
00499         free ((void *) digitizer_flow_control_application_setup->
00500             application_setup_conf_file_path);
00501         digitizer_flow_control_application_setup->
00502             application_setup_conf_file_path = NULL;
00503         digitizer_flow_control_application_setup->
00504             application_setup_conf_file_path =
00505             (char *) malloc (strlen (my_punt) + 1);
00506         strcpy ((char *) digitizer_flow_control_application_setup
->
00507             application_setup_conf_file_path, my_punt);
00508     }
00509 }
00510
00511 //Se e' stato ricevuto il comando -f [pathdelrawdatafile]
00512 else if (DigitizerFlowControl::
00513     reg_matches (buffer, "^[-][Dd][\t]*$"))
00514     || command_received_by_user.command_sent_by_user ==
00515     CHANGEDATA)
00516 {
00517     //WARNING: modificare il path del data file non e' banale perche' si rischia l'inconsistenza delle
operazioni se l'acquisizione e' attiva.
00518     //Qui si e' preferito interrompere l'acquisizione, modificare il path del data file e riprendere
l'acquisizione.
00519     //Se il comando e' stato inviato da tcp, viene ricavato il path del file di salvataggio dei
rawdata dalla stringa contenuta in buffer.

```

```

00520         if (command_received_by_user.command_sent_by_user ==
CHANGEDATA)
00521     {
00522         output_module->Output("Changing data file...\n");
00523         if (miogitizier.go_raw_data == 1)
00524         {
00525             support_flag1 = 1;
00526             miogitizier.DigitizerStateStopRawDataWriting ();
00527         }
00528         digitizer_flow_control_application_setup->
00529             ApplicationSetupDataFileModify (command_received_by_user.
00530                 first_parameter);
00531         if (support_flag1 == 1)
00532         {
00533             support_flag1 = 0;
00534             miogitizier.DigitizerStateStartRawDataWriting ();
00535         }
00536     }
00537     else
00538     {
00539         //Se il comando e' stato inviato da stdin, viene ricavato il path del file di salvataggio dei
rawdata dalla stringa contenuta in buffer.
00540         //In questo caso la stringa e' un po' diversa rispetto a quella del caso precedente: per questo
e' necessario distinguere le procedure
00541         //di ricavo della stringa.
00542         my_punt = FindPointer (buffer);
00543         output_module->Output("Changing data file...\n");
00544         if (miogitizier.go_raw_data == 1)
00545         {
00546             support_flag1 = 1;
00547             miogitizier.DigitizerStateStopRawDataWriting ();
00548         }
00549         digitizer_flow_control_application_setup->
00550             ApplicationSetupDataFileModify (my_punt);
00551         if (support_flag1 == 1)
00552         {
00553             support_flag1 = 0;
00554             miogitizier.DigitizerStateStartRawDataWriting ();
00555         }
00556     }
00557 }
00558
00559 //Se e' stato ricevuto il comando -l [pathdellogfile]
00560 else if (DigitizerFlowControl::
00561     reg_matches (buffer, "^[^][Ll][ ]+.$")
00562     || command_received_by_user.command_sent_by_user ==
00563     CHANGELOG)
00564     {
00565         //Se il comando e' stato inviato da tcp, viene ricavato il path del log file dalla stringa
contenuta in buffer.
00566         if (command_received_by_user.command_sent_by_user ==
CHANGELOG)
00567         {
00568             output_module->Output("Changing log file...\n");
00569             free ((void *) digitizer_flow_control_application_setup->
00570                 application_setup_log_file_path);
00571             digitizer_flow_control_application_setup->
00572                 application_setup_log_file_path = NULL;
00573             digitizer_flow_control_application_setup->
00574                 application_setup_log_file_path =
00575                 (char *)
00576                 malloc (strlen (command_received_by_user.first_parameter)
00577                     + 1);
00578             strcpy ((char *) digitizer_flow_control_application_setup
->
00579                 application_setup_log_file_path,
00580                 command_received_by_user.first_parameter);
00581             logfile->
00582                 LogFileSet (digitizer_flow_control_application_setup->
00583                     application_setup_log_file_path);
00584         }
00585         else
00586         {
00587             //Se il comando e' stato inviato da stdin, viene ricavato il path del log file dalla stringa
contenuta in buffer.
00588             //In questo caso la stringa e' un po' diversa rispetto a quella del caso precedente: per questo
e' necessario distinguere le procedure
00589             //di ricavo della stringa.
00590             my_punt = FindPointer (buffer);
00591             output_module->Output("Changing log file...\n");
00592             free ((void *) digitizer_flow_control_application_setup->
00593                 application_setup_log_file_path);
00594             digitizer_flow_control_application_setup->
00595                 application_setup_log_file_path = NULL;
00596             digitizer_flow_control_application_setup->
00597                 application_setup_log_file_path =
00598                 (char *) malloc (strlen (my_punt) + 1);

```

```

00599         strcpy ((char *) digitizer_flow_control_application_setup
->
00600             application_setup_log_file_path, my_punt);
00601     logfile->
00602         LogFileSet (digitizer_flow_control_application_setup->
00603             application_setup_log_file_path);
00604     }
00605     }
00606
00607     //Se e' stato inviato il comando check.
00608     else if (DigitizerFlowControl::
00609         reg_matches (buffer, "^[Cc][Hh][Ee][Cc][Kk][ \\t]*$")
00610         || command_received_by_user.command_sent_by_user ==
CHECK)
00611     {
00612         //Viene controllato se il file di configurazione contiene manca dei parametri fondamentali per
impostare il digitizer.
00613         output_module->Output("Checking configuration file...\n");
00614         my_consistence.
00615         ConfigurationConsistenceConfFileSetupEssentialWithInitCheck
00616         (digitizer_flow_control_application_setup->
00617             application_setup_conf_file_path);
00618     }
00619
00620     //Se e' stato inviato il comando chkconf.
00621     else if (DigitizerFlowControl::
00622         reg_matches (buffer, "^[Cc][Hh][Kk][Cc][Oo][Nn][Ff][ \\t]*$")
00623         || command_received_by_user.command_sent_by_user ==
CHKCONF)
00624     {
00625         //Viene stampato il contenuto del file di configurazione: come contenuto si intendono le stringhe
riconosciute dallo scanner generato in flex.
00627         output_module->Output("These are the settings read from the configuration file...\n");
00628         AnalizzaPrint (digitizer_flow_control_application_setup
->
00629             application_setup_conf_file_path);
00630     }
00631
00632     //Se e' stato inviato il comando write register [address] [data].
00633     else if (DigitizerFlowControl::
00634         reg_matches (buffer,
00635             "[Ww][Rr][Ii][Tt][Ee][ \\t]+[Rr][Ee][Gg][Ii][Ss][Tt][Ee][Rr][ \\t]+(0x[0-9a-fA-F]{1,16})[ \\t]+(0x[0-9a-fA-F]{1,16})[ \\t]*$")
00636         || command_received_by_user.command_sent_by_user ==
WRITEREGISTER)
00637     {
00638         Data = -1;
00639
00640         //Se il comando e' stato inviato da tcp, vengono ricavati i dati dalla stringa inviata.
00642
00643         if (command_received_by_user.command_sent_by_user ==
WRITEREGISTER)
00644         {
00645             Address =
00646                 strtoul (command_received_by_user.first_parameter, NULL,
00647                     16);
00648             my_punt =
00649                 FindPointer (command_received_by_user.first_parameter);
00650             Data = strtoul (my_punt, NULL, 16);
00651             bzero (support_string, STANDARDBUFFERLIMIT);
00652             snprintf (support_string, STANDARDBUFFERLIMIT,
00653                 "Writing 0x%x in the 0x%x register\n", Data,
00654                 Address);
00655             output_module->Output (support_string);
00656             miodigitizer.digitizer.
00657                 DigitizerObjectWriteRegister (Address, Data);
00658
00659             //Viene stampato il contenuto del registro modificato per poter rendersi subito conto se la
modifica e' avvenuta con successo.
00661             Data = -1;
00662             miodigitizer.digitizer.
00663                 DigitizerObjectReadRegister (Address, &Data);
00664             bzero (support_string, STANDARDBUFFERLIMIT);
00665             snprintf (support_string, STANDARDBUFFERLIMIT,
00666                 "The data in the 0x%x register are: 0x%x\n",
00667                 Address, Data);
00668             output_module->Output (support_string);
00669         }
00670         else
00671         {
00672             //Se il comando e' stato inviato da stdin, vengono ricavati i dati dalla stringa inviata: la
stringa e' leggermente diversa rispetto a quella del
00673             //caso precedente quindi e' stato necessario distinguere i due casi.
00674             my_punt = FindPointer (buffer);
00675             my_punt = FindPointer (my_punt);
00676             Address = strtoul (my_punt, NULL, 16);

```

```

00677         my_punt = FindPointer (my_punt);
00678         Data = strtoul (my_punt, NULL, 16);
00679         bzero (support_string, STANDARDBUFFERLIMIT);
00680         snprintf (support_string, STANDARDBUFFERLIMIT,
00681                 "Writing 0x%x in the 0x%x register\n", Data,
00682                 Address);
00683         output_module->Output(support_string);
00684         mioidigitizer.digitizer.
00685             DigitizerObjectWriteRegister (Address, Data);
00686
00687         //Viene stampato il contenuto del registro modificato per poter rendersi subito conto se la
modifica e' avvenuto con successo.
00688         Data = -1;
00689         mioidigitizer.digitizer.
00690             DigitizerObjectReadRegister (Address, &Data);
00691         bzero (support_string, STANDARDBUFFERLIMIT);
00692         snprintf (support_string, STANDARDBUFFERLIMIT,
00693                 "The data in the 0x%x register are: 0x%x\n",
00694                 Address, Data);
00695         output_module->Output(support_string);
00696     }
00697 }
00698
00699 //Se e' stato inviato il comando read register [address] [data].
00700 else if (DigitizerFlowControl::
00701     reg_matches (buffer,
00702         "[Rr][Ee][Aa][Dd][ \\t]+[Rr][Ee][Gg][Ii][Ss][Tt][Ee][Rr][ \\t]+(0x[0-9a-fA-F]{1,16})[ \\t]*$"
00703 )
00704     || command_received_by_user.command_sent_by_user ==
00705     READREGISTER)
00706 {
00707     Data = -1;
00708
00709     //Se l'input e' stato ricevuto via tcp.
00710     if (command_received_by_user.command_sent_by_user ==
00711         READREGISTER)
00712     {
00713         //Ricavo dalla stringa contenuta in buffer l'indirizzo del registro da leggere.
00714
00715         Address =
00716             strtoul (command_received_by_user.first_parameter, NULL,
00717                     16);
00718         mioidigitizer.digitizer.
00719             DigitizerObjectReadRegister (Address, &Data);
00720         bzero (support_string, STANDARDBUFFERLIMIT);
00721         snprintf (support_string, STANDARDBUFFERLIMIT,
00722                 "I've found this data in the 0x%x register: 0x%x\n",
00723                 Address, Data);
00724         output_module->Output(support_string);
00725     }
00726     else
00727     {
00728         //Se l'input e' stato ricevuto via stdin, ricavo dalla stringa contenuta in buffer l'indirizzo
del registro da leggere.
00729         my_punt = FindPointer (buffer);
00730         my_punt = FindPointer (my_punt);
00731         Address = strtoul (my_punt, NULL, 16);
00732         mioidigitizer.digitizer.
00733             DigitizerObjectReadRegister (Address, &Data);
00734         bzero (support_string, STANDARDBUFFERLIMIT);
00735         snprintf (support_string, STANDARDBUFFERLIMIT,
00736                 "I've found this data in the 0x%x register: 0x%x\n",
00737                 Address, Data);
00738         output_module->Output(support_string);
00739     }
00740 }
00741
00742 //Se e' stato inviato il comando "more".
00743 else if (DigitizerFlowControl::
00744     reg_matches (buffer, "^[Mm][Oo][Rr][Ee][ \\t]*$"
00745 )
00746     || command_received_by_user.command_sent_by_user ==
MORE)
00747 {
00748     //Viene letto il contenuto del log file.
00749     logfile->LogFileRead ();
00750 }
00751
00752 //Se e' stato ricevuto il comando status.
00753 else if (DigitizerFlowControl::
00754     reg_matches (buffer, "^[Ss][Tt][Aa][Tt][Uu][Ss][ \\t]*$"
00755 )
00756     || command_received_by_user.command_sent_by_user ==
STATUS)
00757 {
00758     //Viene stampato lo stato (ON/OFF) del thread che acquisisce i dati e di quelli che operano sui
dati

```

```

00757         mioidigitizer.DigitizerStateMachinePrintStatus ();
00758     }
00759
00760     //Se il comando non e' stato riconosciuto.
00761     else
00762     {
00763         output_module->StdOutInsert ("Unrecognized command\n");
00764         output_module->StdOutPrint ();
00765     }
00766
00767 }
00768
00769 else
00770 pthread_cond_wait (&input_flow_cond, &input_flow_mutex);
00771 // strlen (buffer) != 0 || command_received_by_user.command_sent_by_user != 0
00772 pthread_mutex_unlock (&input_flow_mutex);
00773 } // while 1
00774 } //void DigitizerFlowControl::DigitizerFlowControlStart ()
00775
00776 //Funzione per confrontare una stringa con un'espressione regolare.
00777 bool DigitizerFlowControl::reg_matches (const char *str, const char *
pattern)
00778 {
00779     regex_t
00780     re;
00781     int
00782     ret;
00783
00784     if (regcomp (&re, pattern, REG_EXTENDED) != 0)
00785         return false;
00786
00787     ret = regexec (&re, str, (size_t) 0, NULL, 0);
00788     regfree (&re);
00789
00790     if (ret == 0)
00791         return true;
00792
00793     return false;
00794 }
00795
00796 //Funzione di aiuto.
00797 void
00798 DigitizerFlowControl::Help ()
00799 {
00800     printf ("Available command list:\n");
00801     printf ("init: open the digitizer\n");
00802     printf ("setup: setup the digitizer\n");
00803     printf ("start: start the data acquisition\n");
00804     printf ("stop: stop the data acquisition\n");
00805     printf ("prestart: start the preprocessing thread\n");
00806     printf ("prestop: stop the preprocessing thread\n");
00807     printf ("vistart [channelnumber]: start the visualization thread\n");
00808     printf ("vistop: stop the visualization thread\n");
00809     printf ("rawstart: start the raw data writing thread\n");
00810     printf ("rawstop: stop the raw data writing thread\n");
00811     printf ("close: close the digitizer\n");
00812     printf ("send: send a software trigger\n");
00813     printf ("help\n");
00814     printf ("check: check the correctness of the configuration file\n");
00815     printf ("chkconf: print the content of the configuration file\n");
00816     printf ("write register 0x[register] 0x[data]\n");
00817     printf ("read register 0x[register]\n");
00818     printf ("-f [conf file path]: change the configuration file path\n");
00819     printf ("-d [data file path]: change the data file path\n");
00820     printf ("-l [log file path]: change the log file path\n");
00821     printf ("print: print the internal configuration object used to configure the digitizer\n");
00822     printf ("print files: print the path of the configuration file, of the data file and of the log file\n");
00823     printf ("status: print the status of the threads acquisition, preprocessing, raw data and visualization\n");
00824     printf ("more: display the content of the logfile\n");
00825     printf ("exit: quit program\n");
00826     printf ("quit: quit program\n");
00827 }
00828
00829 //Funzione per stampare il significato dei comandi inviati via tcp.
00830 const char *
00831 DigitizerFlowControl::ParseCommand (int comando_inviato_da_tcp)
00832 {
00833     switch (comando_inviato_da_tcp)
00834     {
00835     case INIT:
00836         return "init";
00837
00838         return "init";
00839
00840     case SETUP:
00841

```

```
00842         return "setup";
00843
00844     case PRESTART:
00845
00846         return "prestart";
00847
00848     case PRESTOP:
00849
00850         return "prestop";
00851
00852     case VISTART:
00853
00854         return "vistart";
00855
00856     case VISTOP:
00857
00858         return "vistop";
00859
00860     case START:
00861
00862         return "start";
00863
00864     case STOP:
00865
00866         return "stop";
00867
00868     case SEND:
00869
00870         return "send";
00871
00872     case CLOSE:
00873
00874         return "close";
00875
00876     case QUIT:
00877
00878         return "quit";
00879
00880     case RAWSTART:
00881
00882         return "rawstart";
00883
00884     case RAWSTOP:
00885
00886         return "rawstop";
00887
00888     case PRINT:
00889
00890         return "print";
00891
00892     case CHECK:
00893
00894         return "check";
00895
00896     case CHKCONF:
00897
00898         return "chkconf";
00899
00900     case MORE:
00901
00902         return "more";
00903
00904     case CHANGECONF:
00905
00906         return "-f [conf_file_path]";
00907
00908     case CHANGEDATA:
00909
00910         return "-d [data_file_path]";
00911
00912     case CHANGELOG:
00913
00914         return "-l [log_file_path]";
00915
00916     case WRITEREGISTER:
00917
00918         return "write register";
00919
00920     case READREGISTER:
00921
00922         return "read register";
00923
00924     case PRINTFILES:
00925
00926         return "print files";
00927
00928     case STATUS:
```



```

00929
00930     return "status";
00931
00932     default:
00933     return "Unrecognized command";
00934     }
00935 }

```

## 6.35 DigitizerFlowControl.h File Reference

### Classes

- class [DigitizerFlowControl](#)

The *[DigitizerFlowControl](#)* class controls the flow of execution of the program.

## 6.36 DigitizerFlowControl.h

```

00001
00012 class DigitizerFlowControl
00013 {
00014 public:
00015
00020     static pthread_cond_t input_flow_cond;
00021
00025     static pthread_mutex_t input_flow_mutex;
00026
00030     Input mioinput;
00031
00035     CommunicationObject mioTCP;
00036
00040     ApplicationSetup *digitizer_flow_control_application_setup
00041 ;
00045     DigitizerFlowControl ();
00046
00051     void DigitizerFlowControlStart ();
00052
00058     const char *ParseCommand (int recvline);
00059
00064     void Help ();
00065
00070     bool reg_matches (const char *str, const char *pattern);
00071 };

```

## 6.37 DigitizerObject.cpp File Reference

```

#include "DefineGeneral.h"
#include "ConfObject.h"
#include "DigitizerErrorObject.h"
#include "LogFile.h"
#include "DigitizerObject.h"
#include "Analizzatore.h"
#include "ApplicationSetup.h"
#include "OutputModule.h"
#include <assert.h>
#include <stdio.h>
#include <CAENDigitizer.h>

```

### 6.37.1 Detailed Description

**Author**

Daniele Berto

Definition in file [DigitizerObject.cpp](#).**6.38 DigitizerObject.cpp**

```

00001
00006 #include "DefineGeneral.h"
00007 #include "ConfObject.h"
00008 #include "DigitizerErrorObject.h"
00009 #include "LogFile.h"
00010 #include "DigitizerObject.h"
00011 #include "Analizzatore.h"
00012 #include "ApplicationSetup.h"
00013 #include "OutputModule.h"
00014 #include <assert.h>
00015 #include <stdio.h>
00016 #include <CAENDigitizer.h>
00017
00018 DigitizerObject::DigitizerObject (const char *config_file)
00019 {
00020     if (AnalizzaInit (&internal_config, config_file))
00021         logfile->
00022             LogFileWriteString
00023             ("Warning: can't create DigitizerObject correctly. Configuration file does't exists\n");
00024     if (AnalizzaSetup (&internal_config, config_file))
00025         logfile->
00026             LogFileWriteString
00027             ("Warning: can't create DigitizerObject correctly. Configuration file does't exists\n");
00028     set_board_info = 0;
00029     logfile = LogFile::Instance ();
00030 }
00031
00032 DigitizerObject::DigitizerObject (ConfObject config)
00033 {
00034     internal_config = config;
00035     set_board_info = 0;
00036     logfile = LogFile::Instance ();
00037 }
00038
00039 void
00040 DigitizerObject::
00041 DigitizerObjectSetConfigStructureConfObject (
00042     ConfObject config)
00043 {
00044     internal_config = config;
00045     logfile = LogFile::Instance ();
00046 }
00047 DigitizerObject::DigitizerObject ()
00048 {
00049     set_board_info = 0;
00050     logfile = LogFile::Instance ();
00051 }
00052
00053 void
00054 DigitizerObject::
00055 DigitizerObjectSetConfigStructureInit (const char *config_file)
00056 {
00057     if (AnalizzaInit (&internal_config, config_file))
00058         logfile->
00059             LogFileWriteString
00060             ("Warning: can't create DigitizerObject correctly. Configuration file does't exists\n");
00061 }
00062
00063 void
00064 DigitizerObject::
00065 DigitizerObjectSetConfigStructureSetup (const char *config_file)
00066 {
00067     if (AnalizzaSetup (&internal_config, config_file))
00068         logfile->
00069             LogFileWriteString
00070             ("Warning: can't create DigitizerObject correctly. Configuration file does't exists\n");
00071 }
00072
00073 int
00074 DigitizerObject::DigitizerObjectOpen ()
00075 {
00076
00077     char stringa[STANDARDBUFFERLIMIT];

```

```

00078     bzero (stringa, STANDARDBUFFERLIMIT);
00079
00080     CAEN_DGTZ_ConnectionType Connection;
00081
00082     if (internal_config.LinkType == 0)
00083         Connection = CAEN_DGTZ_USB;
00084     else if (internal_config.LinkType == 1)
00085         Connection = CAEN_DGTZ_OpticalLink;
00086     else
00087     {
00088         snprintf (stringa, STANDARDBUFFERLIMIT,
00089             "%s %d Configuration structure invalid: LinkType %d field inconsistent\n",
00090             __FILE__, __LINE__, internal_config.LinkType);
00091         logfile->LogFileWriteString (stringa);
00092         return 1;
00093     }
00094
00095     ret =
00096     CAEN_DGTZ_OpenDigitizer (Connection, internal_config.
LinkNumber,
00097         internal_config.ConetNode,
00098         internal_config.VMEBaseAddress, &
handle);
00099     //ret_error.DigitizerErrorObjectDebugging(ret, __FILE__, __func__, __LINE__);
00100     logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00101     //ret_error.DigitizerErrorObjectPrintError (ret);
00102
00103     return 0;
00104 }
00105
00106 int
00107 DigitizerObject::DigitizerObjectReset ()
00108 {
00109     ret = CAEN_DGTZ_Reset (handle);
00110     //ret_error.digitizer_error_object_debugging(ret, __FILE__, __LINE__);
00111     //ret_error.DigitizerErrorObjectPrintError (ret);
00112     //ret_error.DigitizerErrorObjectDebugging(ret, __FILE__, __func__, __LINE__);
00113     logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00114     return 0;
00115 }
00116
00117 int
00118 DigitizerObject::DigitizerObjectGetInfo ()
00119 {
00120     ret = CAEN_DGTZ_GetInfo (handle, &BoardInfo);
00121     //ret_error.digitizer_error_object_debugging(ret, __FILE__, __LINE__);
00122     //ret_error.DigitizerErrorObjectPrintError (ret);
00123     //ret_error.DigitizerErrorObjectDebugging(ret, __FILE__, __func__, __LINE__);
00124     logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00125     if (ret == CAEN_DGTZ_Success)
00126         set_board_info = 1;
00127     return 0;
00128 }
00129
00130 int
00131 DigitizerObject::GetFamilyCode(int * FamilyCode)
00132 {
00133
00134     char stringa[STANDARDBUFFERLIMIT];
00135
00136     if (set_board_info == 0)
00137     {
00138         bzero (stringa, STANDARDBUFFERLIMIT);
00139         snprintf (stringa, STANDARDBUFFERLIMIT,
00140             "File: %s, Function %s, Line: %d, Error: can't print board info. Use
DigitizerObject::DigitizerObjectgetinfo first. Perhaps you have not open the digitizer?\n",
00141             __FILE__, __func__, __LINE__);
00142         logfile->LogFileWriteString (stringa);
00143         return 1;
00144     }
00145
00146     *FamilyCode = BoardInfo.FamilyCode;
00147
00148 }
00149
00150 int
00151 DigitizerObject::GetFormFactorCode(int * FormFactor)
00152 {
00153
00154     char stringa[STANDARDBUFFERLIMIT];
00155
00156     if (set_board_info == 0)
00157     {
00158         bzero (stringa, STANDARDBUFFERLIMIT);
00159         snprintf (stringa, STANDARDBUFFERLIMIT,
00160             "File: %s, Function %s, Line: %d, Error: can't print board info. Use
DigitizerObject::DigitizerObjectgetinfo first. Perhaps you have not open the digitizer?\n",

```

```

00161     __FILE__, __func__, __LINE__);
00162     logfile->LogFileWriteString (stringa);
00163     return 1;
00164 }
00165
00166 *FormFactor = BoardInfo.FormFactor;
00167 }
00168 }
00169
00170 void
00171 DigitizerObject::PrintBoardInfo ()
00172 {
00173     OutputModule *output_module;
00174     output_module = OutputModule::Instance ();
00175
00176     char stringa[STANDARDBUFFERLIMIT];
00177
00178     if (set_board_info == 0)
00179     {
00180         bzero (stringa, STANDARDBUFFERLIMIT);
00181         snprintf (stringa, STANDARDBUFFERLIMIT,
00182             "File: %s, Function %s, Line: %d, Error: can't print board info. Use
00183 DigitizerObject::DigitizerObjectgetinfo first. Perhaps you have not open the digitizer?\n",
00184             __FILE__, __func__, __LINE__);
00185         logfile->LogFileWriteString (stringa);
00186         return;
00187     }
00188     bzero (stringa, STANDARDBUFFERLIMIT);
00189     snprintf (stringa, STANDARDBUFFERLIMIT, "Board model: %s\n",
00190         BoardInfo.ModelName);
00191     output_module->Output (stringa);
00192
00193     bzero (stringa, STANDARDBUFFERLIMIT);
00194     snprintf (stringa, STANDARDBUFFERLIMIT, "Model: %d\n",
00195         BoardInfo.Model);
00196     output_module->Output (stringa);
00197
00198     bzero (stringa, STANDARDBUFFERLIMIT);
00199     snprintf (stringa, STANDARDBUFFERLIMIT, "Channels: %d\n",
00200         BoardInfo.Channels);
00201     output_module->Output (stringa);
00202
00203
00204     bzero (stringa, STANDARDBUFFERLIMIT);
00205     snprintf (stringa, STANDARDBUFFERLIMIT, "Form factor: %d\n",
00206         BoardInfo.FormFactor);
00207     output_module->Output (stringa);
00208
00209
00210     bzero (stringa, STANDARDBUFFERLIMIT);
00211     snprintf (stringa, STANDARDBUFFERLIMIT, "FamilyCode: %d\n",
00212         BoardInfo.FamilyCode);
00213     output_module->Output (stringa);
00214
00215
00216     bzero (stringa, STANDARDBUFFERLIMIT);
00217     snprintf (stringa, STANDARDBUFFERLIMIT, "ROC_FirmwareRel: %s\n",
00218         BoardInfo.ROC_FirmwareRel);
00219     output_module->Output (stringa);
00220
00221
00222     bzero (stringa, STANDARDBUFFERLIMIT);
00223     snprintf (stringa, STANDARDBUFFERLIMIT, "AMC_FirmwareRel: %s\n",
00224         BoardInfo.AMC_FirmwareRel);
00225     output_module->Output (stringa);
00226
00227
00228     bzero (stringa, STANDARDBUFFERLIMIT);
00229     snprintf (stringa, STANDARDBUFFERLIMIT, "SerialNumber: %d\n",
00230         BoardInfo.SerialNumber);
00231     output_module->Output (stringa);
00232
00233
00234     bzero (stringa, STANDARDBUFFERLIMIT);
00235     snprintf (stringa, STANDARDBUFFERLIMIT, "PCB_Revision: %d\n",
00236         BoardInfo.PCB_Revision);
00237     output_module->Output (stringa);
00238
00239
00240     bzero (stringa, STANDARDBUFFERLIMIT);
00241     snprintf (stringa, STANDARDBUFFERLIMIT, "ADC_NBits: %d\n",
00242         BoardInfo.ADC_NBits);
00243     output_module->Output (stringa);
00244
00245

```

```

00246     bzero (stringa, STANDARDBUFFERLIMIT);
00247     snprintf (stringa, STANDARDBUFFERLIMIT, "SAMCorrectionDataLoaded: %d\n",
00248             BoardInfo.SAMCorrectionDataLoaded);
00249     output_module->Output (stringa);
00250
00251
00252     bzero (stringa, STANDARDBUFFERLIMIT);
00253     snprintf (stringa, STANDARDBUFFERLIMIT, "CommHandle: %d\n",
00254             BoardInfo.CommHandle);
00255     output_module->Output (stringa);
00256
00257
00258     bzero (stringa, STANDARDBUFFERLIMIT);
00259     snprintf (stringa, STANDARDBUFFERLIMIT, "License: %s\n",
00260             BoardInfo.License);
00261     output_module->Output (stringa);
00262
00263 }
00264
00265 int
00266 DigitizerObject::DigitizerObjectSetRecordLength (int
00267     recordlength)
00268 {
00269     ret = CAEN_DGTZ_SetRecordLength (handle, recordlength);
00270     //ret_error.DigitizerErrorObjectPrintError (ret);
00271     //ret_error.DigitizerErrorObjectDebugging(ret, __FILE__, __func__, __LINE__);
00272     logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00273     return 0;
00274 }
00275
00276
00277 int
00278 DigitizerObject::DigitizerObjectSetGroupEnableMask (int
00279     enablemask)
00280 {
00281     ret = CAEN_DGTZ_SetGroupEnableMask (handle, enablemask);
00282     //ret_error.DigitizerErrorObjectPrintError (ret);
00283     //ret_error.DigitizerErrorObjectDebugging(ret, __FILE__, __func__, __LINE__);
00284     logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00285     return 0;
00286 }
00287
00288 int
00289 DigitizerObject::DigitizerObjectWriteRegister (int registry,
00290     int data)
00291 {
00292     ret = CAEN_DGTZ_WriteRegister (handle, registry, data);
00293     //ret_error.DigitizerErrorObjectPrintError (ret);
00294     //ret_error.DigitizerErrorObjectDebugging(ret, __FILE__, __func__, __LINE__);
00295     logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00296     return 0;
00297 }
00298
00299 int
00300 DigitizerObject::DigitizerObjectReadRegister (int registry, int
00301     *data)
00302 {
00303     ret =
00304         CAEN_DGTZ_ReadRegister (handle, (uint32_t) registry, (uint32_t *) data);
00305     //ret_error.DigitizerErrorObjectPrintError (ret);
00306     //ret_error.DigitizerErrorObjectDebugging(ret, __FILE__, __func__, __LINE__);
00307     logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00308     return 0;
00309 }
00310
00311
00312 int
00313 DigitizerObject::DigitizerObjectGetDRS4SamplingFrequency
00314     (CAEN_DGTZ_DRS4Frequency_t * frequenza)
00315 {
00316
00317     ret = CAEN_DGTZ_GetDRS4SamplingFrequency (handle, frequenza);
00318     //ret_error.DigitizerErrorObjectPrintError (ret);
00319     //ret_error.DigitizerErrorObjectDebugging(ret, __FILE__, __func__, __LINE__);
00320     logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00321     return 0;
00322 }
00323
00324
00325 int
00326 DigitizerObject::DigitizerObjectSetDRS4SamplingFrequency
00327     (CAEN_DGTZ_DRS4Frequency_t frequenza)

```

```

00328 {
00329
00330     ret = CAEN_DGTZ_SetDRS4SamplingFrequency (handle, frequenza);
00331     //ret_error.DigitizerErrorObjectPrintError (ret);
00332     //ret_error.DigitizerErrorObjectDebugging(ret, __FILE__, __func__, __LINE__);
00333     logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00334     return 0;
00335 }
00336 }
00337
00338 int
00339 DigitizerObject::DigitizerObjectSetChannelDCOffset (int
channel, int dc_mask)
00340 {
00341
00342     ret = CAEN_DGTZ_SetChannelDCOffset (handle, channel, dc_mask);
00343     //ret_error.DigitizerErrorObjectPrintError (ret);
00344     //ret_error.DigitizerErrorObjectDebugging(ret, __FILE__, __func__, __LINE__);
00345     logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00346     return 0;
00347 }
00348 }
00349
00350 int
00351 DigitizerObject::DigitizerObjectSetGroupDCOffset (int group
, int dc_mask)
00352 {
00353
00354     ret = CAEN_DGTZ_SetGroupDCOffset (handle, group, dc_mask);
00355     ret_error.DigitizerErrorObjectPrintError (
ret);
00356     return 0;
00357 }
00358 }
00359
00360 int
00361 DigitizerObject::DigitizerObjectSetMaxNumEventsBLT (int
MaxNumEventsBLT)
00362 {
00363
00364     ret = CAEN_DGTZ_SetMaxNumEventsBLT (handle, MaxNumEventsBLT);
00365     //ret_error.DigitizerErrorObjectPrintError (ret);
00366     //ret_error.DigitizerErrorObjectDebugging(ret, __FILE__, __func__, __LINE__);
00367     logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00368     return 0;
00369 }
00370 }
00371
00372 int
00373 DigitizerObject::DigitizerObjectSetAcquisitionMode (
CAEN_DGTZ_AcqMode_t
AcqMode)
00374 {
00375
00376     ret = CAEN_DGTZ_SetAcquisitionMode (handle, AcqMode);
00377     //ret_error.DigitizerErrorObjectPrintError (ret);
00378     //ret_error.DigitizerErrorObjectDebugging(ret, __FILE__, __func__, __LINE__);
00379     logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00380     return 0;
00381 }
00382 }
00383 }
00384
00385 int
00386 DigitizerObject::DigitizerObjectClose ()
00387 {
00388
00389     ret = CAEN_DGTZ_CloseDigitizer (handle);
00390     //ret_error.DigitizerErrorObjectPrintError (ret);
00391     //ret_error.DigitizerErrorObjectDebugging(ret, __FILE__, __func__, __LINE__);
00392     logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00393     return 0;
00394 }
00395 }
00396
00397 int
00398 DigitizerObject::DigitizerObjectSetExtTriggerInputMode
(CAEN_DGTZ_TriggerMode_t TriggerMode)
00399 {
00400
00401     ret = CAEN_DGTZ_SetExtTriggerInputMode (handle, TriggerMode);
00402     //ret_error.DigitizerErrorObjectPrintError (ret);
00403     //ret_error.DigitizerErrorObjectDebugging(ret, __FILE__, __func__, __LINE__);
00404     logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00405     return 0;
00406 }
00407 }
00408 }
00409

```

```

00410 int
00411 DigitizerObject::DigitizerObjectSetSWTriggerMode (
    CAEN_DGTZ_TriggerMode_t
00412     TriggerMode)
00413 {
00414
00415     ret = CAEN_DGTZ_SetSWTriggerMode (handle, TriggerMode);
00416     //ret_error.DigitizerErrorObjectPrintError (ret);
00417     //ret_error.DigitizerErrorObjectDebugging(ret, __FILE__, __func__, __LINE__);
00418     logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00419     return 0;
00420
00421 }
00422
00423 int
00424 DigitizerObject::DigitizerObjectSWStartAcquisition ()
00425 {
00426
00427     ret = CAEN_DGTZ_SWStartAcquisition (handle);
00428     //ret_error.DigitizerErrorObjectPrintError (ret);
00429     //ret_error.DigitizerErrorObjectDebugging(ret, __FILE__, __func__, __LINE__);
00430     logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00431     return 0;
00432
00433 }
00434
00435 int
00436 DigitizerObject::DigitizerObjectSWStopAcquisition ()
00437 {
00438
00439     ret = CAEN_DGTZ_SWStopAcquisition (handle);
00440     //ret_error.DigitizerErrorObjectPrintError (ret);
00441     //ret_error.DigitizerErrorObjectDebugging(ret, __FILE__, __func__, __LINE__);
00442     logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00443     return 0;
00444
00445 }

```

## 6.39 DigitizerObject.h File Reference

```

#include <assert.h>
#include <stdio.h>
#include <CAENDigitizer.h>

```

### Classes

- class [DigitizerObject](#)

The [DigitizerObject](#) class envelops CAEN\_DGTZ functions from CAENDigitizer library.

### 6.39.1 Detailed Description

#### Author

Daniele Berto

Definition in file [DigitizerObject.h](#).

## 6.40 DigitizerObject.h

```

00001
00011 #include <assert.h>
00012 #include <stdio.h>
00013 #include <CAENDigitizer.h>
00014
00015 class DigitizerObject
00016 {
00017

```

```

00018 public:
00019
00023     CAEN_DGTZ_ErrorCode ret;
00024
00028     DigitizerErrorObject ret_error;
00029
00033     ConfObject internal_config;
00034
00038     int set_board_info;
00039
00043     int handle;
00044
00048     CAEN_DGTZ_BoardInfo_t BoardInfo;
00049
00053     DigitizerObject ();
00054
00061     DigitizerObject (const char *config_file);
00062
00068     DigitizerObject (ConfObject config);
00069
00075     void DigitizerObjectSetConfigStructureConfObject (
00076     ConfObject config);
00076
00083     void DigitizerObjectSetConfigStructureInit (const char *config_file)
00084     ;
00084
00091     void DigitizerObjectSetConfigStructureSetup (const char *
00092     config_file);
00092
00097     int DigitizerObjectOpen ();
00098
00103     int DigitizerObjectClose ();
00104
00109     int DigitizerObjectReset ();
00110
00115     int DigitizerObjectGetInfo ();
00116
00121     CAEN_DGTZ_BoardInfo_t GetBoardInfo ();
00122
00126     int GetFamilyCode(int * FamilyCode);
00127
00131     int GetFormFactorCode(int * FormFactor);
00132
00137     void PrintBoardInfo ();
00138
00144     int DigitizerObjectSetRecordLength (int recordlength);
00145
00151     int DigitizerObjectSetGroupEnableMask (int enablemask);
00152
00159     int DigitizerObjectWriteRegister (int registry, int data);
00160
00167     int DigitizerObjectReadRegister (int registry, int *data);
00168
00174     int DigitizerObjectGetDRS4SamplingFrequency (
00175     CAEN_DGTZ_DRS4Frequency_t *
00176     frequenza);
00176
00182     int DigitizerObjectSetDRS4SamplingFrequency (
00183     CAEN_DGTZ_DRS4Frequency_t
00184     frequenza);
00184
00191     int DigitizerObjectSetChannelDCOffset (int channel_mask, int dc_mask);
00192
00199     int DigitizerObjectSetGroupDCOffset (int group_mask, int dc_mask);
00200
00206     int DigitizerObjectSetMaxNumEventsBLT (int MaxNumEventsBLT);
00207
00213     int DigitizerObjectSetAcquisitionMode (CAEN_DGTZ_AcqMode_t AcqMode);
00214
00220     int DigitizerObjectSetExtTriggerInputMode (CAEN_DGTZ_TriggerMode_t
00221     TriggerMode);
00222
00228     int DigitizerObjectSetSWTriggerMode (CAEN_DGTZ_TriggerMode_t TriggerMode);
00229
00234     int DigitizerObjectSWStartAcquisition ();
00235
00240     int DigitizerObjectSWStopAcquisition ();
00241
00245     LogFile *logfile;
00246 };

```



## 6.41 DigitizerObjectGeneric.cpp File Reference

```
#include "ConfObject.h"
#include "DigitizerErrorObject.h"
#include "LogFile.h"
#include "DigitizerObject.h"
#include "DigitizerObjectGeneric.h"
#include "Analizzatore.h"
#include "DefineGeneral.h"
#include <assert.h>
#include <stdio.h>
#include <CAENDigitizer.h>
```

## 6.42 DigitizerObjectGeneric.cpp

```
00001
00005 #include "ConfObject.h"
00006 #include "DigitizerErrorObject.h"
00007 #include "LogFile.h"
00008 #include "DigitizerObject.h"
00009 #include "DigitizerObjectGeneric.h"
00010 #include "Analizzatore.h"
00011 #include "DefineGeneral.h"
00012 #include <assert.h>
00013 #include <stdio.h>
00014 #include <CAENDigitizer.h>
00015
00016
00017 DigitizerObjectGeneric::DigitizerObjectGeneric ()
00018 {
00019     set_board_info = 0;
00020     logfile = LogFile::Instance ();
00021 }
00022
00023
00024 int
00025 DigitizerObjectGeneric::DigitizerObjectGenericSetAllInformations
00026 ()
00027 {
00028     int max_channels = 0;
00029     int max_groups = 0;
00030
00031     int FamilyCode = BoardInfo.FamilyCode;
00032
00033     int FormFactor = BoardInfo.FormFactor;
00034
00035     int tmp;
00036
00037     int i = 0;
00038
00039     switch(FamilyCode) {
00040     case CAEN_DGTZ_XX724_FAMILY_CODE:
00041     case CAEN_DGTZ_XX781_FAMILY_CODE:
00042     case CAEN_DGTZ_XX720_FAMILY_CODE:
00043     case CAEN_DGTZ_XX721_FAMILY_CODE:
00044     case CAEN_DGTZ_XX751_FAMILY_CODE:
00045     case CAEN_DGTZ_XX761_FAMILY_CODE:
00046     case CAEN_DGTZ_XX731_FAMILY_CODE:
00047         switch(FormFactor) {
00048         case CAEN_DGTZ_VME64_FORM_FACTOR:
00049             max_channels = 8;
00050             break;
00051         case CAEN_DGTZ_DESKTOP_FORM_FACTOR:
00052         case CAEN_DGTZ_NIM_FORM_FACTOR:
00053             max_channels = 4;
00054             break;
00055         }
00056         break;
00057     case CAEN_DGTZ_XX730_FAMILY_CODE:
00058         switch(FormFactor) {
00059         case CAEN_DGTZ_VME64_FORM_FACTOR:
00060         case CAEN_DGTZ_VME64X_FORM_FACTOR:
00061             max_channels = 16;
00062             break;
00063         case CAEN_DGTZ_DESKTOP_FORM_FACTOR:
```

```

00064         case CAEN_DGTZ_NIM_FORM_FACTOR:
00065             max_channels = 8;
00066             break;
00067     }
00068     break;
00069     case CAEN_DGTZ_XX740_FAMILY_CODE:
00070         switch(FormFactor) {
00071             case CAEN_DGTZ_VME64_FORM_FACTOR:
00072             case CAEN_DGTZ_VME64X_FORM_FACTOR:
00073                 max_channels = 64;
00074                 break;
00075             case CAEN_DGTZ_DESKTOP_FORM_FACTOR:
00076             case CAEN_DGTZ_NIM_FORM_FACTOR:
00077                 max_channels = 32;
00078                 break;
00079         }
00080     break;
00081     case CAEN_DGTZ_XX742_FAMILY_CODE:
00082         switch(FormFactor) {
00083             case CAEN_DGTZ_VME64_FORM_FACTOR:
00084             case CAEN_DGTZ_VME64X_FORM_FACTOR:
00085                 //max_channels = 36; ---> sbagliato!!!
00086                 max_groups = 4;
00087                 break;
00088             case CAEN_DGTZ_DESKTOP_FORM_FACTOR:
00089             case CAEN_DGTZ_NIM_FORM_FACTOR:
00090                 //max_channels = 16; ---> sbagliato!!!
00091                 max_groups = 2;
00092                 break;
00093         }
00094     break;
00095     default:
00096         assert("This program cannot be used with this digitizer family\n");
00097     }
00098
00099     if (FamilyCode == CAEN_DGTZ_XX740_FAMILY_CODE)
00100     {
00101
00102         tmp = max_channels/8;
00103
00104         if (internal_config.dc_offset != -1)
00105         {
00106             for (i=0; i<tmp; i++)
00107             {
00108                 DigitizerObject::DigitizerObjectSetGroupDCOffset
00109                 (i, internal_config.dc_offset);
00110             }
00111             if (internal_config.trigger_threshold != -1)
00112             {
00113                 for (i=0; i<tmp; i++)
00114                 {
00115                     ret = CAEN_DGTZ_SetGroupTriggerThreshold (handle, i,
00116                     internal_config.trigger_threshold);
00117                     logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00118                 }
00119             }
00120         }
00121     else
00122     {
00123
00124         if (FamilyCode == CAEN_DGTZ_XX742_FAMILY_CODE)
00125             max_channels = max_groups*8;
00126
00127         if (internal_config.dc_offset != -1)
00128         {
00129             for (i=0; i<max_channels; i++)
00130             {
00131                 DigitizerObject::DigitizerObjectSetChannelDCOffset
00132                 (i, internal_config.dc_offset);
00133             }
00134             if (internal_config.trigger_threshold != -1)
00135             {
00136                 for (i=0; i<max_channels; i++)
00137                 {
00138                     ret = CAEN_DGTZ_SetChannelTriggerThreshold (handle, i,
00139                     internal_config.trigger_threshold);
00140                     logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00141                 }
00142             }
00143         }
00144     }
00145 }
00146

```

```

00147 int
00148 DigitizerObjectGeneric::DigitizerObjectGenericSetDecimationFactor
00149 ()
00150 {
00151     if (BoardInfo.FamilyCode == CAEN_DGTZ_XX740_FAMILY_CODE &&
00152         internal_config.decimation_factor != -1)
00153     {
00154         ret = CAEN_DGTZ_SetDecimationFactor (handle, internal_config.
00155             decimation_factor);
00156         logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00157         return 0;
00158     }
00159 }
00160 int
00161 DigitizerObjectGeneric::DigitizerObjectGenericSetDesMode
00162 ()
00163 {
00164     if (((BoardInfo.FamilyCode == CAEN_DGTZ_XX751_FAMILY_CODE) || (
00165         BoardInfo.FamilyCode == CAEN_DGTZ_XX731_FAMILY_CODE)) &&
00166         internal_config.desmod != -1)
00167     {
00168         CAEN_DGTZ_EnaDis_t desmodtype;
00169         if (internal_config.desmod == 1)
00170         {
00171             desmodtype = CAEN_DGTZ_ENABLE;
00172         }
00173         else
00174         {
00175             desmodtype = CAEN_DGTZ_DISABLE;
00176         }
00177         ret = CAEN_DGTZ_SetDESMODE(handle, desmodtype);
00178         logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00179         return 0;
00180     }
00181 }
00182 int
00183 DigitizerObjectGeneric::DigitizerObjectGenericSetTestPattern
00184 ()
00185 {
00186     if (internal_config.test_pattern == 1)
00187     {
00188         ret = CAEN_DGTZ_WriteRegister(handle, CAEN_DGTZ_BROAD_CH_CONFIGBIT_SET_ADD, 1<<3);
00189         logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00190         return 0;
00191     }
00192 }
00193 int
00194 DigitizerObjectGeneric::DigitizerObjectGenericSetRecordLength
00195 ()
00196 {
00197     ret = CAEN_DGTZ_SetRecordLength (handle, internal_config.
00198         record_length);
00199     logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00200     return 0;
00201 }
00202 int
00203 DigitizerObjectGeneric::DigitizerObjectGenericSetMaxNumEventsBLT
00204 ()
00205 {
00206     ret =
00207         CAEN_DGTZ_SetMaxNumEventsBLT (handle, internal_config.
00208             max_num_events_BLT);
00209     logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00210     return 0;
00211 }
00212 int
00213 DigitizerObjectGeneric::DigitizerObjectGenericSetEnableMask
00214 ()
00215 {
00216     //Occorre scrivere le informazioni direttamente sui registri perche' le funzioni di libreria non
00217     funzionano
00218     if (BoardInfo.FamilyCode == CAEN_DGTZ_XX740_FAMILY_CODE
00219         || BoardInfo.FamilyCode == CAEN_DGTZ_XX742_FAMILY_CODE)
00220     {
00221         if (internal_config.group_enable_mask != -1)

```

```

00221     {
00222         DigitizerObjectWriteRegister (0x8120,
00223             internal_config.group_enable_mask);
00224     }
00225 }
00226 else
00227 {
00228     if (internal_config.channel_enable_mask != -1)
00229     {
00230         DigitizerObjectWriteRegister (0x8120,
00231             internal_config.channel_enable_mask);
00232     }
00233 }
00234 } //int DigitizerObjectGeneric::DigitizerObjectGenericSetEnableMask ()
00235
00236
00237
00238 int
00239 DigitizerObjectGeneric::DigitizerObjectGenericSetExtTriggerInputMode
00240 ()
00241 {
00242     switch (internal_config.external_trigger_acquisition_mode
00243 )
00244     {
00245     case 0:
00246         ret =
00247         CAEN_DGTZ_SetExtTriggerInputMode (handle, CAEN_DGTZ_TRGMODE_ACQ_ONLY);
00248         logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00249         break;
00250     case 1:
00251         ret =
00252         CAEN_DGTZ_SetExtTriggerInputMode (handle,
00253             CAEN_DGTZ_TRGMODE_ACQ_AND_EXTOUT);
00254         logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00255         break;
00256     case 2:
00257         ret =
00258         CAEN_DGTZ_SetExtTriggerInputMode (handle, CAEN_DGTZ_TRGMODE_DISABLED);
00259         logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00260         break;
00261     }
00262 }
00263 } //int DigitizerObjectGeneric::DigitizerObjectGenericSetExtTriggerInputMode ()
00264
00265
00266
00267
00268 int
00269 DigitizerObjectGeneric::DigitizerObjectGenericSetDRS4SamplingFrequency
00270 ()
00271 {
00272     if (BoardInfo.FamilyCode == CAEN_DGTZ_XX742_FAMILY_CODE)
00273     {
00274         switch (internal_config.DSR4_Frequency)
00275         {
00276         case 0:
00277             ret =
00278             CAEN_DGTZ_SetDRS4SamplingFrequency (handle, CAEN_DGTZ_DRS4_5GHz);
00279             logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00280             break;
00281         case 1:
00282             ret =
00283             CAEN_DGTZ_SetDRS4SamplingFrequency (handle,
00284                 CAEN_DGTZ_DRS4_2_5GHz);
00285             logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00286             break;
00287         case 2:
00288             ret =
00289             CAEN_DGTZ_SetDRS4SamplingFrequency (handle, CAEN_DGTZ_DRS4_1GHz);
00290             logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00291             break;
00292         } //switch(internal_config.DSR4_Frequency)
00293     } //if (BoardInfo.FamilyCode == CAEN_DGTZ_XX742_FAMILY_CODE)
00294 } //int DigitizerObjectGeneric::DigitizerObjectGenericSetDRS4SamplingFrequency ()
00295
00296
00297
00298 int
00299 DigitizerObjectGeneric::DigitizerObjectGenericSetPostTriggerSize
00300 ()
00301 {
00302     if (internal_config.post_trigger != -1)
00303     {
00304         ret =

```

```

00304     CAEN_DGTZ_SetPostTriggerSize (handle, internal_config.
post_trigger);
00305     logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00306 }
00307 } //int DigitizerObjectGeneric::DigitizerObjectGenericSetPostTriggerSize ()
00308
00309
00310 int
00311 DigitizerObjectGeneric::DigitizerObjectGenericSetIOLevel
()
00312 {
00313
00314     switch (internal_config.nim_ttl)
00315     {
00316     case 0: //NIM
00317         ret = CAEN_DGTZ_SetIOLevel (handle, CAEN_DGTZ_IOLevel_NIM);
00318         logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00319         break;
00320
00321     case 1: //TTL
00322         ret = CAEN_DGTZ_SetIOLevel (handle, CAEN_DGTZ_IOLevel_TTL);
00323         logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00324         break;
00325     }
00326
00327 } // int DigitizerObjectGeneric::DigitizerObjectGenericSetIOLevel ()
00328
00329
00330
00331 int
00332 DigitizerObjectGeneric::DigitizerObjectGenericSetDCOffset
()
00333 {
00334
00335     int i;
00336
00337     if (BoardInfo.FamilyCode == CAEN_DGTZ_XX740_FAMILY_CODE)
00338     {
00339         for (i = 0; i < MAXGROUPOBJECT; i++)
00340         {
00341             if (internal_config.groups[i].set != -1
00342                 && internal_config.groups[i].dc_offset != -1)
00343             {
00344                 DigitizerObject::DigitizerObjectSetGroupDCOffset
(internal_config.groups[i].numGroup,
00345                     internal_config.groups[i].dc_offset);
00346             }
00347         } //for (i=0;i<MAXGROUPOBJECT;i++)
00348     }
00349
00350     else
00351     {
00352         for (i = 0; i < MAXCHANNELOBJECT; i++)
00353         {
00354             if (internal_config.channels[i].set != -1
00355                 && internal_config.channels[i].dc_offset != -1)
00356             {
00357                 DigitizerObject::DigitizerObjectSetChannelDCOffset
(internal_config.channels[i].numChannel,
00358                     internal_config.channels[i].dc_offset);
00359             }
00360         } //for (i=0;i<MAXCHANNELOBJECT;i++)
00361     }
00362
00363
00364 } //int DigitizerObjectGeneric::DigitizerObjectGenericSetDCOffset ()
00365
00366
00367
00368 int
00369 DigitizerObjectGeneric::DigitizerObjectGenericSetSelfTrigger
()
00370 {
00371
00372     if (BoardInfo.FamilyCode == CAEN_DGTZ_XX740_FAMILY_CODE)
00373     {
00374         switch (internal_config.self_trigger_enable_mask_mode)
00375         {
00376         case 0:
00377             ret =
00378                 CAEN_DGTZ_SetGroupSelfTrigger (handle, CAEN_DGTZ_TRGMODE_ACQ_ONLY,
00379                     internal_config.
00380                         self_trigger_enable_mask);
00381             logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00382             break;
00383
00384         case 1:
00385             ret =

```

```

00387         CAEN_DGTZ_SetGroupSelfTrigger (handle,
00388                                         CAEN_DGTZ_TRGMODE_ACQ_AND_EXTOUT,
00389                                         internal_config.
00390                                         self_trigger_enable_mask);
00391         logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00392         break;
00393
00394     case 2:
00395         ret =
00396             CAEN_DGTZ_SetGroupSelfTrigger (handle, CAEN_DGTZ_TRGMODE_DISABLED,
00397                                             internal_config.
00398                                             self_trigger_enable_mask);
00399         logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00400         break;
00401     }
00402 }
00403 }
00404 else
00405 {
00406
00407
00408     switch (internal_config.self_trigger_enable_mask_mode)
00409     {
00410     case 0:
00411         ret =
00412             CAEN_DGTZ_SetChannelSelfTrigger (handle,
00413                                              CAEN_DGTZ_TRGMODE_ACQ_ONLY,
00414                                              internal_config.
00415                                              self_trigger_enable_mask);
00416         logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00417         break;
00418
00419     case 1:
00420         ret =
00421             CAEN_DGTZ_SetChannelSelfTrigger (handle,
00422                                              CAEN_DGTZ_TRGMODE_ACQ_AND_EXTOUT,
00423                                              internal_config.
00424                                              self_trigger_enable_mask);
00425         logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00426         break;
00427
00428     case 2:
00429         ret =
00430             CAEN_DGTZ_SetChannelSelfTrigger (handle,
00431                                              CAEN_DGTZ_TRGMODE_DISABLED,
00432                                              internal_config.
00433                                              self_trigger_enable_mask);
00434         logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00435         break;
00436     }
00437 }
00438 }
00439
00440 } //int DigitizerObjectGeneric::DigitizerObjectGenericSetSelfTrigger ()
00441
00442
00443
00444
00445 int
00446 DigitizerObjectGeneric::
00447 DigitizerObjectGenericSetChannelSelfTriggerThreshold ()
00448 {
00449
00450     int i;
00451
00452     if (BoardInfo.FamilyCode == CAEN_DGTZ_XX740_FAMILY_CODE)
00453     {
00454         for (i = 0; i < MAXGROUPOBJECT; i++)
00455         {
00456             if (internal_config.groups[i].set != -1
00457                 && internal_config.groups[i].trigger_threshold != -1)
00458             {
00459                 ret = CAEN_DGTZ_SetGroupTriggerThreshold
00460                     (handle, internal_config.groups[i].numGroup,
00461                      internal_config.groups[i].trigger_threshold);
00462                 logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00463             }
00464         } //for (i=0; i<MAXGROUPOBJECT; i++)
00465     }
00466     else
00467     {
00468
00469         for (i = 0; i < MAXCHANNELOBJECT; i++)
00470         {
00471             if (internal_config.channels[i].set != -1
00472                 && internal_config.channels[i].
00473                 trigger_threshold != -1)

```

```

00473     {
00474         ret = CAEN_DGTZ_SetChannelTriggerThreshold
00475         (handle, internal_config.channels[i].
numChannel,
00476         internal_config.channels[i].trigger_threshold);
00477         logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00478     }
00479 } //for (i=0;i<MAXCHANNELOBJECT;i++)
00480 } //else
00481
00482 } //int DigitizerObjectGeneric::DigitizerObjectGenericSetChannelSelfTriggerThreshold ()
00483
00484
00485
00486 int
00487 DigitizerObjectGeneric::DigitizerObjectGenericSetFastTriggerDigitizing
()
00488 {
00489
00490     if (BoardInfo.FamilyCode == CAEN_DGTZ_XX742_FAMILY_CODE)
00491     {
00492
00493         if (internal_config.enable_fast_trigger_digitizing == 1)
00494         {
00495             ret = CAEN_DGTZ_SetFastTriggerDigitizing (handle, CAEN_DGTZ_ENABLE);
00496             logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00497         }
00498
00499         if (internal_config.enable_fast_trigger_digitizing == 0)
00500         {
00501             ret =
00502             CAEN_DGTZ_SetFastTriggerDigitizing (handle, CAEN_DGTZ_DISABLE);
00503             logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00504         }
00505
00506         if (internal_config.fast_trigger_acquisition_mode == 0)
00507         {
00508             ret =
00509             CAEN_DGTZ_SetFastTriggerMode (handle, CAEN_DGTZ_TRGMODE_ACQ_ONLY);
00510             logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00511         }
00512
00513         if (internal_config.fast_trigger_acquisition_mode == 2)
00514         {
00515             ret =
00516             CAEN_DGTZ_SetFastTriggerMode (handle, CAEN_DGTZ_TRGMODE_DISABLED);
00517             logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00518         }
00519
00520         if (internal_config.fast[0].set != -1
00521         && internal_config.fast[0].dc_offset != -1)
00522         {
00523             ret =
00524             CAEN_DGTZ_SetGroupFastTriggerDCOffset (handle,
00525             internal_config.fast[0].
numFast,
00526             internal_config.fast[0].
dc_offset);
00527             logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00528         }
00529
00530         if (internal_config.fast[1].set != -1
00531         && internal_config.fast[1].dc_offset != -1)
00532         {
00533             ret =
00534             CAEN_DGTZ_SetGroupFastTriggerDCOffset (handle,
00535             internal_config.fast[1].
numFast,
00536             internal_config.fast[1].
dc_offset);
00537             logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00538         }
00539
00540         if (internal_config.fast[0].set != -1
00541         && internal_config.fast[0].trigger_threshold != -1)
00542         {
00543             ret =
00544             CAEN_DGTZ_SetGroupFastTriggerThreshold (handle,
00545             internal_config.fast[0].
numFast,
00546             internal_config.fast[0].
trigger_threshold);
00547             logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00548         }
00549
00550         if (internal_config.fast[1].set != -1
00551         && internal_config.fast[1].trigger_threshold != -1)
00552         {
00553             ret =
00554             CAEN_DGTZ_SetGroupFastTriggerThreshold (handle,
00555             internal_config.fast[1].
numFast,
00556             internal_config.fast[1].
trigger_threshold);
00557             logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00558         }
00559     }
00560 }

```

```

00558     {
00559         ret = CAEN_DGTZ_SetGroupFastTriggerThreshold
00560             (handle, internal_config.fasts[1].numFast,
00561              internal_config.fasts[1].trigger_threshold);
00562         logfile->LogFileWrite (ret, __FILE__, __func__, __LINE__);
00563     }
00564 }
00565 } //if (internal_config.BoardInfo.FamilyCode == CAEN_DGTZ_XX742_FAMILY_CODE) {
00566
00567 } //int DigitizerObjectGeneric::DigitizerObjectGenericSetFastTriggerDigitizing ()

```

## 6.43 DigitizerObjectGeneric.h File Reference

```

#include <assert.h>
#include <stdio.h>
#include <CAENDigitizer.h>

```

### Classes

- class [DigitizerObjectGeneric](#)

The *DigitizerObjectGeneric* class extends the *DigitizerObject* class with methods for setting the digitizer using "internal\_config" attribute.

## 6.44 DigitizerObjectGeneric.h

```

00001
00012 #include <assert.h>
00013 #include <stdio.h>
00014 #include <CAENDigitizer.h>
00015
00016 class DigitizerObjectGeneric:public DigitizerObject
00017 {
00018 public:
00019
00024     int DigitizerObjectGenericSetRecordLength ();
00025
00030     int DigitizerObjectGenericSetMaxNumEventsBLT ();
00031
00037     int DigitizerObjectGenericSetEnableMask ();
00038
00044     int DigitizerObjectGenericSetFastTriggerDigitizing ();
00045
00050     int DigitizerObjectGenericSetSelfTrigger ();
00051
00056     int DigitizerObjectGenericSetChannelSelfTriggerThreshold
00057     ();
00058
00063     int DigitizerObjectGenericSetDCOffset ();
00064
00069     int DigitizerObjectGenericWriteRegister ();
00070
00075     int DigitizerObjectGenericReadRegister ();
00076
00081     int DigitizerObjectGenericGetDRS4SamplingFrequency ();
00082
00087     int DigitizerObjectGenericSetDRS4SamplingFrequency ();
00088
00093     int DigitizerObjectGenericSetChannelDCOffset ();
00094
00099     int DigitizerObjectGenericSetGroupDCOffset ();
00100
00105     int DigitizerObjectGenericSetTriggerPolarity ();
00106
00112     int DigitizerObjectGenericSetPostTriggerSize ();
00113
00118     int DigitizerObjectGenericSetIOLevel ();
00119
00124     int DigitizerObjectGenericSetAcquisitionMode ();
00125
00130     int DigitizerObjectGenericSetExtTriggerInputMode ();
00131

```



```

00136     int DigitizerObjectGenericSetSWTriggerMode ();
00137
00142     int DigitizerObjectGenericSetDecimationFactor ();
00143
00148     int DigitizerObjectGenericSetDesMode ();
00149
00154     int DigitizerObjectGenericSetTestPattern ();
00155
00160     int DigitizerObjectGenericSetAllInformations ();
00161
00165     DigitizerObjectGeneric ();
00166 };

```

## 6.45 DigitizerStateMachine.cpp File Reference

```

#include "DefineGeneral.h"
#include "X742DecodeRoutines.h"
#include "ConfObject.h"
#include "ApplicationSetup.h"
#include "DigitizerErrorObject.h"
#include "LogFile.h"
#include "DigitizerObject.h"
#include "Analizzatore.h"
#include "DigitizerObjectGeneric.h"
#include "RawData.h"
#include "DigitizerStateMachine.h"
#include "OutputModule.h"
#include <time.h>
#include <pthread.h>
#include <assert.h>
#include <stdio.h>
#include <CAENDigitizer.h>
#include <thread>
#include <mutex>
#include <condition_variable>

```

## 6.46 DigitizerStateMachine.cpp

```

00001
00005 #include "DefineGeneral.h"
00006 #include "X742DecodeRoutines.h"
00007 #include "ConfObject.h"
00008 #include "ApplicationSetup.h"
00009 #include "DigitizerErrorObject.h"
00010 #include "LogFile.h"
00011 #include "DigitizerObject.h"
00012 #include "Analizzatore.h"
00013 #include "DigitizerObjectGeneric.h"
00014 #include "RawData.h"
00015 #include "DigitizerStateMachine.h"
00016 #include "OutputModule.h"
00017 #include <time.h>
00018 #include <pthread.h>
00019 #include <assert.h>
00020 #include <stdio.h>
00021 #include <CAENDigitizer.h>
00022 #include <thread>
00023 #include <mutex>
00024 #include <condition_variable>
00025
00026 using namespace std;
00027
00028 void
00029 DigitizerStateMachine::Produttore ()
00030 {
00031     //Codice per testare il tempo.
00032     /*
00033

```

```

00034     struct timespec start, stop;
00035     unsigned long accum;
00036     unsigned long parz_clock;
00037     unsigned long parz=1000000000L;
00038
00039     double accum2;
00040     /*
00041     unique_lock<mutex> ReservedConsumerDispatcherInputAreaHandle(ReservedConsumerDispatcherInputArea,
00042     defer_lock);
00043     unique_lock<mutex> ReservedPreprocessingInputAreaHandle(ReservedPreprocessingInputArea, defer_lock);
00044     unique_lock<mutex> ReservedVisualizationInputAreaHandle(ReservedVisualizationInputArea, defer_lock);
00045
00046     int i;
00047     RawData tmp;
00048     tmp.RawDataSet (digitizer);
00049
00050     //Codice per testare il tempo.
00051     //clock_gettime(CLOCK_REALTIME, &start);
00052
00053     while (go_general)
00054     {
00055     //Codice per testare il tempo.
00056     /*clock_gettime(CLOCK_REALTIME, &stop);
00057
00058         parz_clock = ((unsigned long)stop.tv_sec - (unsigned long)start.tv_sec) - 1 ;
00059         if (parz_clock>=0) {accum=parz_clock*1000000000L+((parz-start.tv_nsec)+stop.tv_nsec);}
00060         else accum = stop.tv_sec - start.tv_sec;
00061         //printf( "%lf\n", accum );
00062         accum2=(double)accum;
00063         accum2=accum2/1000000L;
00064         if(tmp.bsize>0)
00065             fprintf(stderr, "%t%.6lf\n", accum2);
00066
00067         clock_gettime(CLOCK_REALTIME, &start);
00068     */
00069
00070     tmp.RawDataRead ();
00071
00072     if (tmp.bsize > 0)
00073     {
00074
00075         ReservedVisualizationInputAreaHandle.lock();
00076         //fprintf(stderr, "num_mex_raw_data:%d\n", num_mex_raw_data);
00077         if (go_raw_data && go_general)
00078         {
00079             if (num_mex_raw_data < RAWDATAQUEUE)
00080             {
00081                 circular_buffer_raw_data[coda_raw_data] = tmp;
00082                 if (coda_raw_data == RAWDATAQUEUE - 1)
00083                     coda_raw_data = 0;
00084                 else
00085                     coda_raw_data = coda_raw_data + 1;
00086                 num_mex_raw_data++;
00087                 BlockedConsumerInput.notify_one();
00088             }
00089             // go_raw_data
00090         }
00091         ReservedVisualizationInputAreaHandle.unlock();
00092
00093
00094
00095         ReservedPreprocessingInputAreaHandle.lock();
00096         //fprintf(stderr, "num_mex_preprocessing:%d\n", num_mex_preprocessing);
00097         if (go_preprocessing && go_general)
00098         {
00099             if (num_mex_preprocessing < PREPROCESSINGQUEUE)
00100             {
00101                 circular_buffer_preprocessing[coda_preprocessing] = tmp;
00102
00103                 if (coda_preprocessing == PREPROCESSINGQUEUE - 1)
00104                     coda_preprocessing = 0;
00105                 else
00106                     coda_preprocessing = coda_preprocessing + 1;
00107                 num_mex_preprocessing++;
00108                 Acquisition_Cond1.notify_one();
00109             }
00110             // go_preprocessing
00111         }
00112         ReservedPreprocessingInputAreaHandle.unlock();
00113
00114
00115
00116
00117         ReservedVisualizationInputAreaHandle.lock();
00118         //fprintf(stderr, "num_mex_visualization:%d\n", num_mex_visualization);
00119         if (go_visualization && go_general)
00120         {
00121             if (num_mex_visualization < VISUALIZATIONQUEUE)
00122             {
00123                 circular_buffer_visualization[coda_visualization] = tmp;

```

```

00125
00126     if (coda_visualization == VISUALIZATIONQUEUE - 1)
00127         coda_visualization = 0;
00128     else
00129         coda_visualization = coda_visualization + 1;
00130     num_mex_visualization++;
00131     Acquisition_Cond2.notify_one();
00132 }
00133 } //go_visualization
00134     ReservedVisualizationInputAreaHandle.unlock();
00135 } //if (tmp.bsize > 0)
00136 } // go_general
00137
00138
00139 //segnalo la fine dell'acquisizione a tutti i threads
00140 for (i = 0; i < 10; i++)
00141 {
00142     Acquisition_Cond2.notify_one();
00143     Acquisition_Cond1.notify_one();
00144     BlockedConsumerInput.notify_one();
00145     raw_cond.notify_one();
00146     pre_cond.notify_one();
00147     vis_cond.notify_one();
00148 }
00149 //fprintf (stderr, "Esco dal Reader Thread\n");
00150
00151 }
00152
00153 void
00154 DigitizerStateMachine::Consumatore_Dispatcher ()
00155 {
00156     ApplicationSetup *application_setup;
00157     application_setup = ApplicationSetup::Instance ();
00158
00159     RawData tmp;
00160     tmp.RawDataSet (digitizer);
00161
00162     unique_lock<mutex> ReservedConsumerDispatcherInputAreaHandle (ReservedConsumerDispatcherInputArea,
00163 std::defer_lock);
00164
00165     while (go_general)
00166     {
00167         if (go_raw_data && go_general)
00168         {
00169
00170
00171             ReservedConsumerDispatcherInputAreaHandle.lock();
00172
00173             if (num_mex_raw_data == 0)
00174             {
00175                 BlockedConsumerInput.wait (ReservedConsumerDispatcherInputAreaHandle);
00176             }
00177
00178             if (go_general != 0 && go_raw_data != 0)
00179             {
00180
00181                 tmp = circular_buffer_raw_data[testa_raw_data];
00182
00183                 ReservedConsumerDispatcherInputAreaHandle.unlock();
00184
00185
00186                 //fprintf(stderr, "scrivo\n");
00187                 if (tmp.bsize > 0)
00188                     tmp.RawDataWriteOnFile (application_setup->
00189 ApplicationSetupGetDataFilePunt (),
00190 application_setup->
00191 ApplicationSetupGetDataFileSizePunt ());
00192
00193
00194                 ReservedConsumerDispatcherInputAreaHandle.lock();
00195
00196                 if (testa_raw_data == RAWDATAQUEUE - 1)
00197                     testa_raw_data = 0;
00198                 else
00199                     testa_raw_data = testa_raw_data + 1;
00200
00201                 num_mex_raw_data--;
00202             }
00203             ReservedConsumerDispatcherInputAreaHandle.unlock();
00204
00205         }
00206     }
00207     else
00208     {
00209         ReservedConsumerDispatcherInputAreaHandle.lock();
00210         raw_cond.wait (ReservedConsumerDispatcherInputAreaHandle);
00211         ReservedConsumerDispatcherInputAreaHandle.unlock();

```

```

00212     }
00213     } //go_general
00214     //fprintf (stderr, "Esco dal Thread RawData\n");
00215 }
00216
00217
00218 void
00219 DigitizerStateMachine::Preprocessing ()
00220 {
00221     int i, j;
00222     uint32_t numEvents;
00223     int conta = 0;
00224     char *evtptr;
00225     CAEN_DGTZ_EventInfo_t eventInfo;
00226     RawData tmp;
00227     tmp.RawDataSet (digitizer);
00228
00229     unique_lock<mutex> ReservedPreprocessingInputAreaHandle(ReservedPreprocessingInputArea, defer_lock);
00230
00231     while (go_general)
00232     {
00233         if (go_preprocessing && go_general)
00234         {
00235             ReservedPreprocessingInputAreaHandle.lock();
00236
00237             if (num_mex_preprocessing == 0)
00238             {
00239                 Acquisition_Cond1.wait(ReservedPreprocessingInputAreaHandle);
00240             }
00241
00242             if (go_general != 0 && go_preprocessing != 0)
00243             {
00244                 tmp = circular_buffer_preprocessing[testa_preprocessing];
00245                 ReservedPreprocessingInputAreaHandle.unlock();
00246
00247                 //fprintf(stderr, "Decodifico\n");
00248                 tmp.RawDataDecode ();
00249
00250                 ReservedPreprocessingInputAreaHandle.lock();
00251
00252                 if (testa_preprocessing == PREPROCESSINGQUEUE - 1)
00253                     testa_preprocessing = 0;
00254                 else
00255                     testa_preprocessing = testa_preprocessing + 1;
00256                 num_mex_preprocessing--; }
00257                 ReservedPreprocessingInputAreaHandle.unlock();
00258             }
00259             else
00260             {
00261                 ReservedPreprocessingInputAreaHandle.lock();
00262                 pre_cond.wait(ReservedPreprocessingInputAreaHandle);
00263                 ReservedPreprocessingInputAreaHandle.unlock();
00264             }
00265             // go_general
00266         }
00267     }
00268 }
00269
00270 void
00271 DigitizerStateMachine::Visualization ()
00272 {
00273     OutputModule *output_module;
00274     output_module = OutputModule::Instance ();
00275
00276     RawData tmp;
00277     tmp.RawDataSet (digitizer);
00278
00279     FILE *gnuplot;
00280
00281     char stringa[STANDARDBUFFERLIMIT];
00282     bzero (stringa, STANDARDBUFFERLIMIT);
00283     snprintf(stringa, STANDARDBUFFERLIMIT, "%s -persist 2>/dev/null", digitizer.
00284             internal_config.gnuplot);
00285
00286     gnuplot = NULL;
00287     gnuplot = popen (stringa, "w");
00288 }

```

```

00298     unique_lock<mutex> ReservedVisualizationInputAreaHandle(ReservedVisualizationInputArea, defer_lock);
00299
00300     //Il codice seguente serve per gestire il caso in cui l'utente non abbia installato gnuplot o abbia
    specificato male il comando
00301     //per lanciarlo nel configuration file. Occorre gestire questa situazione perche' si rischia, altrimenti,
    l'arresto del programma senza nessun
00302     //messaggio di errore rendendo il debugging difficile.
00303     int gnuplot_error_code;
00304     gnuplot_error_code = pclose(gnuplot);
00305
00306     if (gnuplot_error_code != 0)
00307     {
00308         output_module->Output("Error, you have not specify gnuplot command correctly in the configuration
    file\n");
00309     }
00310
00311     gnuplot = NULL;
00312     gnuplot = popen (stringa, "w");
00313
00314     while (go_general)
00315     {
00316
00317         if (go_visualization && go_general)
00318         {
00319
00320             ReservedVisualizationInputAreaHandle.lock();
00321
00322             if (num_mex_visualization == 0)
00323                 Acquisition_Cond2.wait(ReservedVisualizationInputAreaHandle);
00324             if (go_general != 0 && go_visualization != 0)
00325             {
00326                 tmp = circular_buffer_visualization[testa_visualization];
00327
00328                 ReservedVisualizationInputAreaHandle.unlock();
00329
00330
00331                 //fprintf(stderr, "plotto\n");
00332                 if (tmp.bsize > 0 && gnuplot_error_code == 0)
00333                     tmp.RawDataPlot ("plot_data.txt", gnuplot);
00334
00335                 ReservedVisualizationInputAreaHandle.lock();
00336
00337                 if (testa_visualization == VISUALIZATIONQUEUE - 1)
00338                     testa_visualization = 0;
00339                 else
00340                     testa_visualization = testa_visualization + 1;
00341
00342                 num_mex_visualization--; }
00343
00344                 ReservedVisualizationInputAreaHandle.unlock();
00345             }
00346         else
00347         {
00348             ReservedVisualizationInputAreaHandle.lock();
00349             vis_cond.wait(ReservedVisualizationInputAreaHandle);
00350             ReservedVisualizationInputAreaHandle.unlock();
00351         }
00352     } // go_general
00353
00354     pclose (gnuplot);
00355 }
00356
00357 DigitizerStateMachine::DigitizerStateMachine ()
00358 {
00359     imset = 0;
00360     imstarted = 0;
00361     go_general = 0;
00362     go_raw_data = 0;
00363     go_preprocessing = 0;
00364     go_visualization = 0;
00365 }
00366
00367 void
00368 DigitizerStateMachine::DigitizerStateMachineInit (const
    char *conf_file)
00369 {
00370     digitizer.DigitizerObjectSetConfigStructureInit (conf_file);
00371     //Se la variabile imset e' impostata a 0 e viene eseguito il metodo
    DigitizerStateMachine::DigitizerStateMachineStartReading,
00372     //viene stampato un messaggio di errore.
00373     imset = 1;
00374
00375     digitizer.DigitizerObjectOpen ();
00376     digitizer.DigitizerObjectReset ();
00377     digitizer.DigitizerObjectGetInfo ();

```

```

00380 digitizer.PrintBoardInfo ();
00381 }
00382
00384 void
00385 DigitizerStateMachine::DigitizerStateMachineSetup (const
char *conf_file)
00386 {
00387     int data;
00388     digitizer.DigitizerObjectReset ();
00389     digitizer.DigitizerObjectSetConfigStructureSetup (conf_file);
00390     digitizer.DigitizerObjectGenericSetRecordLength ();
00391     digitizer.DigitizerObjectGenericSetMaxNumEventsBLT ();
00392     digitizer.DigitizerObjectSetAcquisitionMode (CAEN_DGTZ_SW_CONTROLLED);
00393     digitizer.DigitizerObjectGenericSetExtTriggerInputMode ();
00394     digitizer.DigitizerObjectGenericSetEnableMask ();
00395     digitizer.DigitizerObjectGenericSetDRS4SamplingFrequency ();
00396     digitizer.DigitizerObjectGenericSetPostTriggerSize ();
00397     digitizer.DigitizerObjectGenericSetChannelSelfTriggerThreshold ();
00398     digitizer.DigitizerObjectGenericSetIOLevel ();
00399     digitizer.DigitizerObjectGenericSetDCOffset ();
00400     digitizer.DigitizerObjectGenericSetSelfTrigger ();
00401     digitizer.DigitizerObjectGenericSetFastTriggerDigitizing ();
00402     digitizer.DigitizerObjectGenericSetDecimationFactor ();
00403     digitizer.DigitizerObjectGenericSetDesMode ();
00404     digitizer.DigitizerObjectGenericSetTestPattern ();
00405     digitizer.DigitizerObjectGenericSetAllInformations ();
00406 }
00407
00410 void
00411 DigitizerStateMachine::DigitizerStateMachineRawDataInit
()
00412 {
00413     int i = 0;
00414     num_mex_raw_data = 0;
00415     coda_raw_data = 0;
00416     testa_raw_data = 0;
00417
00418     num_mex_preprocessing = 0;
00419     coda_preprocessing = 0;
00420     testa_preprocessing = 0;
00421
00422     num_mex_visualization = 0;
00423     coda_visualization = 0;
00424     testa_visualization = 0;
00425
00426     for (i = 0; i < RAWDATAQUEUE; i++)
00427     {
00428         circular_buffer_raw_data[i].RawDataSet (digitizer);
00429     }
00430
00431     for (i = 0; i < PREPROCESSINGQUEUE; i++)
00432     {
00433         circular_buffer_preprocessing[i].RawDataSet (digitizer);
00434     }
00435
00436     for (i = 0; i < VISUALIZATIONQUEUE; i++)
00437     {
00438         circular_buffer_visualization[i].RawDataSet (digitizer);
00439     }
00440
00441     go_general = 1;
00442
00443     visualization_thread = new thread (&DigitizerStateMachine::Visualization
, this);
00444
00445     preprocessing_thread = new thread (&DigitizerStateMachine::Preprocessing
, this);
00446
00447     consumatore_thread = new thread (&
DigitizerStateMachine::Consumatore_Dispatcher, this);
00448
00449     produttore_thread = new thread (&DigitizerStateMachine::Produttore, this
);
00450
00451 }
00452
00453
00454 void
00455 DigitizerStateMachine::DigitizerStateMachineStartReading
()
00456 {
00457     OutputModule *output_module;
00458     output_module = OutputModule::Instance ();
00459
00460     if (imset == 0)
00461     {

```

```

00463     output_module->Output("DigitizerStateMachine: Error, you have not set the object, use
DigitizerStateMachineInit(const char *conf_file) method.\n");
00464 }
00465
00466     if (imstarted == 0)
00467     {
00468         imstarted = 1;
00469         digitizer.DigitizerObjectSWStartAcquisition ();
00470         DigitizerStateMachine::DigitizerStateMachineRawDataInit
00471     };
00471     output_module->Output("DigitizerStateMachine: Inizio la lettura dei dati\n");
00472     }
00473     else
00474     {
00475         output_module->Output("DigitizerStateMachine: Error, you have already insert start command \n")
00476     };
00477 }
00478 }
00479
00480
00481 void
00482 DigitizerStateMachine::DigitizerStateMachineQuit ()
00483 {
00484     OutputModule *output_module;
00485     output_module = OutputModule::Instance ();
00486     output_module->Output("DigitizerStateMachine: Esco dalla sessione di acquisizione\n");
00487     DigitizerStateMachine::DigitizerStateMachineStopReading
00488 };
00489     digitizer.DigitizerObjectClose ();
00490 }
00491 }
00492
00493 void
00494 DigitizerStateMachine::DigitizerStateMachineStopReading
00495 ()
00496 {
00497     OutputModule *output_module;
00498     output_module = OutputModule::Instance ();
00499     output_module->Output("DigitizerStateMachine: Interrompo la lettura dei dati\n");
00500     imstarted = 0;
00501     go_general = 0;
00502
00503     //DigitizerStateMachine::DigitizerStateStopRawDataWriting();
00504     BlockedConsumerInput.notify_one();
00505
00506     //DigitizerStateMachine::DigitizerStateStopPreprocessing();
00507     Acquisition_Cond1.notify_one();
00508
00509     //DigitizerStateMachine::DigitizerStateStopVisualization();
00510     Acquisition_Cond2.notify_one();
00511
00512     raw_cond.notify_one();
00513     vis_cond.notify_one();
00514     pre_cond.notify_one();
00515
00516     digitizer.DigitizerObjectSWStopAcquisition ();
00517 }
00518 }
00519
00520 void
00521 DigitizerStateMachine::DigitizerStateMachineSendSWTrigger
00522 ()
00523 {
00524     OutputModule *output_module;
00525     output_module = OutputModule::Instance ();
00526     output_module->Output("DigitizerStateMachine: invio il software trigger\n");
00527     CAEN_DGTZ_SendSWtrigger (digitizer.handle);
00528 }
00529
00530 void
00531 DigitizerStateMachine::DigitizerStateStartPreprocessing
00532 ()
00533 {
00534     OutputModule *output_module;
00535     output_module = OutputModule::Instance ();
00536     output_module->Output("DigitizerStateMachine: Starting Preprocessing thread\n");

```

```
00543
00544     go_preprocessing = 1;
00545     pre_cond.notify_one();
00546 }
00547
00548 void
00549 DigitizerStateMachine::DigitizerStateStartVisualization
00550 ()
00551 {
00552     OutputModule *output_module;
00553     output_module = OutputModule::Instance ();
00554     output_module->Output ("Starting Visualization thread\n");
00555     go_visualization = 1;
00556     vis_cond.notify_one();
00557 }
00558
00559 void
00560 DigitizerStateMachine::DigitizerStateStopPreprocessing
00561 ()
00562 {
00563     OutputModule *output_module;
00564     output_module = OutputModule::Instance ();
00565     output_module->Output ("DigitizerStateMachine: Stopping Preprocessing thread\n");
00566     go_preprocessing = 0;
00567 }
00568
00569 void
00570 DigitizerStateMachine::DigitizerStateStartRawDataWriting
00571 ()
00572 {
00573     OutputModule *output_module;
00574     output_module = OutputModule::Instance ();
00575     output_module->Output ("DigitizerStateMachine: Starting RawData thread\n");
00576     go_raw_data = 1;
00577     raw_cond.notify_one();
00578 }
00579
00580 void
00581 DigitizerStateMachine::DigitizerStateStopRawDataWriting
00582 ()
00583 {
00584     OutputModule *output_module;
00585     output_module = OutputModule::Instance ();
00586     output_module->Output ("DigitizerStateMachine: Stopping RawData thread\n");
00587     go_raw_data = 0;
00588 }
00589
00590 void
00591 DigitizerStateMachine::DigitizerStateStopVisualization
00592 ()
00593 {
00594     OutputModule *output_module;
00595     output_module = OutputModule::Instance ();
00596     output_module->Output ("DigitizerStateMachine: Stopping Visualization thread\n");
00597     go_visualization = 0;
00598 }
00599
00600 void
00601 DigitizerStateMachine::DigitizerStateMachinePrintStatus
00602 ()
00603 {
00604     OutputModule *output_module;
00605     output_module = OutputModule::Instance ();
00606     if (go_general)
00607     {
00608         output_module->Output ("Data acquisition thread ON\n");
00609     }
00610     else
00611     {
00612         output_module->Output ("Data acquisition thread OFF\n");
00613     }
00614     if (go_preprocessing)
00615     {
00616         output_module->Output ("Preprocessing thread ON\n");
00617     }
00618 }
```



```

00624     else
00625     {
00626         output_module->Output("Preprocessing thread OFF\n");
00627     }
00628
00629     if (go_raw_data)
00630     {
00631         output_module->Output("Raw data thread ON\n");
00632     }
00633     else
00634     {
00635         output_module->Output("Raw data thread OFF\n");
00636     }
00637
00638     if (go_visualization)
00639     {
00640         output_module->Output("Visualization thread ON\n");
00641     }
00642     else
00643     {
00644         output_module->Output("Visualization thread OFF\n");
00645     }
00646
00647 }

```

## 6.47 DigitizerStateMachine.h File Reference

```

#include "DefineGeneral.h"
#include <assert.h>
#include <stdio.h>
#include <pthread.h>
#include <CAENDigitizer.h>
#include <thread>
#include <mutex>
#include <condition_variable>

```

### Classes

- class [DigitizerStateMachine](#)

## 6.48 DigitizerStateMachine.h

```

00001
00011 #include "DefineGeneral.h"
00012 #include <assert.h>
00013 #include <stdio.h>
00014 #include <pthread.h>
00015 #include <CAENDigitizer.h>
00016 #include <thread>
00017 #include <mutex>
00018 #include <condition_variable>
00019
00020 using namespace std;
00021
00022 class DigitizerStateMachine
00023 {
00024
00025 public:
00026
00030     int num_mex_raw_data;
00031
00035     int num_mex_preprocessing;
00036
00040     int num_mex_visualization;
00041
00045     int testa_raw_data;
00046
00050     int testa_preprocessing;
00051
00055     int testa_visualization;

```

```
00056
00060 int coda_raw_data;
00061
00065 int coda_preprocessing;
00066
00070 int coda_visualization;
00071
00075 int go_general;
00076
00080 int go_preprocessing;
00081
00085 int go_raw_data;
00086
00090 int go_visualization;
00091
00095 DigitizerObjectGeneric digitizer;
00096
00101 RawData circular_buffer_raw_data[RAWDATAQUEUE];
00102
00107 RawData circular_buffer_preprocessing[PREPROCESSINGQUEUE];
00108
00113 RawData circular_buffer_visualization[VISUALIZATIONQUEUE];
00114
00120 void DigitizerStateMachineSetup (const char *conf_file);
00121
00126 void DigitizerStateMachineStartReading ();
00127
00132 void DigitizerStateMachinePrintStatus ();
00133
00138 void DigitizerStateMachineStopReading ();
00139
00145 void DigitizerStateMachineRawDataInit ();
00146
00151 void DigitizerStateMachineQuit ();
00152
00157 void DigitizerStateMachineSendSWTrigger ();
00158
00163 void DigitizerStateStartPreprocessing ();
00164
00169 void DigitizerStateStartVisualization ();
00170
00175 void DigitizerStateStopPreprocessing ();
00176
00181 void DigitizerStateStartRawDataWriting ();
00182
00187 void DigitizerStateStopRawDataWriting ();
00188
00193 void DigitizerStateStopVisualization ();
00194
00199 void Produttore ();
00200
00205 void Consumatore_Dispatcher ();
00206
00211 void Preprocessing ();
00212
00217 void Visualization ();
00218
00222 thread * produttore_thread;
00223
00227 thread * consumatore_thread;
00228
00232 thread * preprocessing_thread;
00233
00237 thread * visualization_thread;
00238
00242 mutex ReservedConsumerDispatcherInputArea;
00243
00247 condition_variable BlockedProducerInput;
00248
00252 condition_variable BlockedConsumerInput;
00253
00257 mutex ReservedPreprocessingInputArea;
00258
00262 mutex ReservedVisualizationInputArea;
00263
00267 condition_variable Acquisition_Cond1;
00268
00272 condition_variable Acquisition_Cond2;
00273
00277 condition_variable raw_cond;
00278
00282 condition_variable pre_cond;
00283
00287 condition_variable vis_cond;
00288
00294 void DigitizerStateMachineInit (const char *conf_file);
00295
```

```

00299     int imstarted;
00300
00304     int imset;
00305
00310     DigitizerStateMachine ();
00311 };

```

## 6.49 Input.cpp File Reference

```

#include "DefineGeneral.h"
#include "DefineCommands.h"
#include "TcpUser.h"
#include "ConfObject.h"
#include "DigitizerErrorObject.h"
#include "LogFile.h"
#include "DigitizerObject.h"
#include "DigitizerObjectGeneric.h"
#include "RawData.h"
#include "DigitizerStateMachine.h"
#include "ApplicationSetup.h"
#include "Input.h"
#include "CommunicationObject.h"
#include "DigitizerFlowControl.h"
#include "ConfigurationConsistence.h"
#include "AnalizzatoreUtils.h"
#include "Analizzatore.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <thread>
#include <mutex>
#include <pthread.h>
#include <condition_variable>

```

## 6.50 Input.cpp

```

00001
00005 #include "DefineGeneral.h"
00006 #include "DefineCommands.h"
00007 #include "TcpUser.h"
00008 #include "ConfObject.h"
00009 #include "DigitizerErrorObject.h"
00010 #include "LogFile.h"
00011 #include "DigitizerObject.h"
00012 #include "DigitizerObjectGeneric.h"
00013 #include "RawData.h"
00014 #include "DigitizerStateMachine.h"
00015 #include "ApplicationSetup.h"
00016 #include "Input.h"
00017 #include "CommunicationObject.h"
00018 #include "DigitizerFlowControl.h"
00019 #include "ConfigurationConsistence.h"
00020 #include "AnalizzatoreUtils.h"
00021 #include "Analizzatore.h"
00022 #include <stdio.h>
00023 #include <stdlib.h>
00024 #include <string.h>
00025 #include <thread>
00026 #include <mutex>
00027 #include <pthread.h>
00028 #include <condition_variable>
00029
00030 using namespace std;
00031
00032 Input::Input ()
00033 {

```

```

00034     int i;
00035     num_mex = 0;
00036     go = 1;
00037     input_buffer = (char *) malloc (STANDARDBUFFERLIMIT);
00038     for (i = 0; i < STANDARDBUFFERLIMIT; i++)
00039         input_buffer[i] = '\0';
00040     producer_thread = new thread (&Input::Producer, this);
00041 }
00042
00043 void
00044 Input::Producer ()
00045 {
00046     int i;
00047     char input_buffer_private[STANDARDBUFFERLIMIT];
00048
00049     //go is set to 1 by the costructor. The finish() method set it to 0.
00050     while (go)
00051     {
00052         for (i = 0; i < STANDARDBUFFERLIMIT; i++)
00053         {
00054             input_buffer_private[i] = '\0';
00055         }
00056
00057         fgets (input_buffer_private, 99, stdin); //getting the input from stdin
00058
00059         for (i = 0; i < STANDARDBUFFERLIMIT; i++)
00060         {
00061             if (input_buffer_private[i] == '\n')
00062                 input_buffer_private[i] = '\0';
00063         }
00064
00065         unique_lock<mutex> lk(mutex1); //assure data consistency
00066
00067         for (i = 0; i < STANDARDBUFFERLIMIT; i++)
00068         {
00069             input_buffer[i] = input_buffer_private[i];
00070         }
00071
00072         pthread_mutex_lock (&DigitizerFlowControl::input_flow_mutex);
00073         pthread_cond_signal (&DigitizerFlowControl::input_flow_cond);
00074         pthread_mutex_unlock (&DigitizerFlowControl::input_flow_mutex);
00075
00076         num_mex = 1; // a message is ready
00077
00078         lk.unlock();
00079     }
00080
00081 }
00082
00083
00084
00085 void
00086 Input::GetInput (char *input_buffer_sending)
00087 {
00088     int i;
00089     unique_lock<mutex> lk(mutex1); //assure data consistency
00090
00091     if (num_mex == 0) // if no message set the input_buffer_sending to zero
00092     {
00093         for (i = 0; i < STANDARDBUFFERLIMIT; i++)
00094             input_buffer_sending[i] = '\0';
00095     }
00096
00097     else // if there is a message, copy it into input_buffer_sending
00098     {
00099         for (i = 0; i < STANDARDBUFFERLIMIT; i++)
00100         {
00101             input_buffer_sending[i] = input_buffer[i];
00102         }
00103         num_mex = 0; //no message ready
00104     }
00105 }
00106
00107
00108 void
00109 Input::Finish ()
00110 {
00111     go = 0; // stop the producer thread
00112 }

```

## 6.51 Input.h File Reference

```
#include <thread>
```

```
#include <mutex>
#include <condition_variable>
```

## Classes

- class [Input](#)

## 6.52 Input.h

```
00001
00011 #include <thread>
00012 #include <mutex>
00013 #include <condition_variable>
00014
00015 using namespace std;
00016
00017 class Input
00018 {
00019 public:
00020
00024     char *input_buffer;
00025
00032     void GetInput (char *input_buffer_sending);
00033
00038     void Finish ();
00039
00043     thread * producer_thread;
00044
00048     mutex mutex1;
00049
00053     int go;
00054
00058     int num_mex;
00059
00064     void Producer ();
00065
00070     Input ();
00071 };
```

## 6.53 LogFile.cpp File Reference

```
#include "ConfObject.h"
#include "ApplicationSetup.h"
#include "DigitizerErrorObject.h"
#include "LogFile.h"
#include "DefineGeneral.h"
#include "OutputModule.h"
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

## 6.54 LogFile.cpp

```
00001
00005 #include "ConfObject.h"
00006 #include "ApplicationSetup.h"
00007 #include "DigitizerErrorObject.h"
00008 #include "LogFile.h"
00009 #include "DefineGeneral.h"
```

```

00010 #include "OutputModule.h"
00011 #include<sys/socket.h>
00012 #include<arpa/inet.h>
00013 #include<unistd.h>
00014 #include <stdio.h>
00015 #include <stdlib.h>
00016 #include <string.h>
00017
00018
00019 LogFile *
00020     LogFile::log_file_pInstance = NULL;
00021
00022
00023 LogFile *
00024 LogFile::Instance ()
00025 {
00026     if (!log_file_pInstance) // Only allow one instance of class to be generated.
00027         log_file_pInstance = new LogFile ();
00028     return log_file_pInstance;
00029 }
00030
00031
00032
00033 void
00034 LogFile::LogFileMessageOn ()
00035 {
00036
00037     flag = 1;
00038 }
00039
00040
00041
00042 void
00043 LogFile::LogFileMessageOff ()
00044 {
00045
00046     flag = 1;
00047 }
00048
00049
00050 void
00051 LogFile::LogFileSet (const char *log_file_arg)
00052 {
00053
00054     flag = 0;
00055
00056     OutputModule *output_module;
00057     output_module = OutputModule::Instance ();
00058
00059     log_file_path = (const char *) malloc (strlen (log_file_arg) + 1);
00060     strcpy ((char *) log_file_path, (char *) log_file_arg);
00061     log_file_punt = fopen (log_file_path, "w");
00062
00063     if (log_file_punt == NULL)
00064     {
00065         output_module->Output("Warning, I can't access log file.\n");
00066         can_write = 0;
00067     }
00068     else
00069     {
00070         output_module->Output("Accessing log file...\n");
00071         can_write = 1;
00072     }
00073 }
00074 //void LogFile::LogFileSet(const char * log_file_arg)
00075
00076
00077 void
00078 LogFile::LogFileWriteString (const char *string)
00079 {
00080     if (can_write)
00081     {
00082         fprintf (log_file_punt, string);
00083     }
00084 }
00085
00086
00087 void
00088 LogFile::LogFileWrite (CAEN_DGTZ_ErrorCode ret_arg, const char *file,
00089                       const char *func, int line)
00090 {
00091     OutputModule *output_module;
00092     output_module = OutputModule::Instance ();
00093
00094     if (can_write)
00095         ret_debug =
00096             my_error.DigitizerErrorObjectDebuggingLog (ret_arg, file, func, line,

```

```

00097             log_file_punt);
00098
00099     if (ret_debug && flag == 1)
00100     {
00101         output_module->Output("Some error occurred, digit \"more\" for informations\n");
00102         flag = 0;
00103     }
00104
00105 } //void LogFile::LogFileWrite(CAEN_DGTZ_ErrorCode ret_arg, const char *file, const char *func, int line)
00106
00107
00108 void
00109 LogFile::LogFileRead ()
00110 {
00111     OutputModule *output_module;
00112     output_module = OutputModule::Instance ();
00113
00114     char buffer[999];
00115     bzero (buffer, 999);
00116     char ch;
00117
00118     int i = 0;
00119     int flag = 0;
00120
00121     FILE *read_log;
00122     read_log = fopen (log_file_path, "r");
00123
00124
00125     if (read_log != NULL)
00126     {
00127         fseek (read_log, 0, SEEK_SET);
00128         while ((ch = fgetc (read_log)) != EOF)
00129         {
00130             if (i < 997)
00131             {
00132                 buffer[i] = ch;
00133                 i++;
00134                 flag = 1;
00135             }
00136             else
00137             {
00138                 buffer[i] = ch;
00139                 buffer[998] = '\0';
00140                 output_module->Output((const char *) buffer);
00141                 bzero (buffer, 999);
00142                 flag = 0;
00143                 i = 0;
00144             } // } else {
00145         } //while( ( ch = fgetc(read_log) ) != EOF )
00146
00147         if (flag == 1)
00148         {
00149             buffer[i] = '\0';
00150             output_module->Output((const char *) buffer);
00151             bzero (buffer, 999);
00152         }
00153         fclose (read_log); // N.B. la fclose deve stare dentro senno' capita una segmentation fault!!!
00154     } // if (read_log != NULL)
00155
00156 } //void LogFile::LogFileRead()

```

## 6.55 LogFile.h File Reference

```
#include <stdio.h>
```

### Classes

- class [LogFile](#)

The *LogFile* singleton is used to write the error codes retrieved by the CAENDigitizer function to the logfile indicates by the *log\_file\_path* attribute.

## 6.56 LogFile.h

```

00001
00011 #include <stdio.h>
00012
00013 class LogFile
00014 {
00015 private:
00019     int ret_debug;
00020
00024     int can_write;
00025
00029     int flag;
00030
00034     FILE *log_file_punt;
00035
00039     static LogFile *log_file_pInstance;
00040
00044     DigitizerErrorObject my_error;
00045
00046 public:
00050     const char *log_file_path;
00051
00055     static LogFile *Instance ();
00056
00060     void LogFileSet (const char *log_file_arg);
00061
00070     void LogFileWrite (CAEN_DGTZ_ErrorCode ret_arg, const char *file,
00071                       const char *func, int line);
00072
00076     void LogFileWriteString (const char *string);
00077
00081     void LogFileRead ();
00082
00086     void LogFileMessageOn ();
00087
00091     void LogFileMessageOff ();
00092 };

```

## 6.57 Main.c File Reference

This file contains the program main.

```

#include "X742DecodeRoutines.h"
#include "TcpUser.h"
#include "ConfObject.h"
#include "DigitizerErrorObject.h"
#include "LogFile.h"
#include "DigitizerObject.h"
#include "DigitizerObjectGeneric.h"
#include "RawData.h"
#include "DigitizerStateMachine.h"
#include "Analizzatore.h"
#include "ApplicationSetup.h"
#include "Input.h"
#include "CommunicationObject.h"
#include "DigitizerFlowControl.h"
#include <CAENDigitizer.h>
#include <stdio.h>
#include <unistd.h>
#include <ctype.h>

```

### Macros

- #define ONLY\_TCP 1



## Functions

- `int main (int argc, char **argv)`

*The main function of the application.*

### 6.57.1 Detailed Description

This file contains the program main.

Definition in file [Main.c](#).

### 6.57.2 Macro Definition Documentation

#### 6.57.2.1 `#define ONLY_TCP 1`

Definition at line 37 of file [Main.c](#).

### 6.57.3 Function Documentation

#### 6.57.3.1 `int main ( int argc, char ** argv )`

The main function of the application.

##### Parameters

<i>argc</i>	is the number of the main arguments
<i>argv</i>	are the main arguments

##### Returns

int

Definition at line 46 of file [Main.c](#).

## 6.58 Main.c

```

00001
00018 #include "X742DecodeRoutines.h"
00019 #include "TcpUser.h"
00020 #include "ConfObject.h"
00021 #include "DigitizerErrorObject.h"
00022 #include "LogFile.h"
00023 #include "DigitizerObject.h"
00024 #include "DigitizerObjectGeneric.h"
00025 #include "RawData.h"
00026 #include "DigitizerStateMachine.h"
00027 #include "Analizzatore.h"
00028 #include "ApplicationSetup.h"
00029 #include "Input.h"
00030 #include "CommunicationObject.h"
00031 #include "DigitizerFlowControl.h"
00032 #include <CAENDigitizer.h>
00033 #include <stdio.h>
00034 #include <unistd.h>
00035 #include <ctype.h>
00036
00037 #define ONLY_TCP 1
00038
00045 int
00046 main (int argc, char **argv)
00047 {
00048
00049     //Se l'utente sceglie di avviare il programma in modalita' ONLY_TCP (flag -m tcp), occorre eseguire il
        programma in modalita' demone.

```

```

00050 //Di default il valore della variabile e' zero cosi' l'espressione process_id == 0 e' verificata anche
00051 //senza assegnare a process_id il risultato
00052 //della system call fork() che viene chiamata solo se il programma viene eseguito con il flag -m tcp
00053 // (cioe' l'unica modalita' di accettazione dell'input e di
00054 //produzione dell'output e' quella tramite tcp).
00055 pid_t process_id = 0;
00056 //L'oggetto di tipo ApplicationSetup serve per salvare le impostazioni che l'utente ha inserito tramite
00057 //flags nella stringa di esecuzione del programma.
00058 //ApplicationSetup e' un singleton, quindi e' necessario ottenerne un riferimento con il metodo
00059 Instance().
00060 ApplicationSetup *main_application_setup;
00061 main_application_setup = ApplicationSetup::Instance ();
00062 //Il metodo ApplicationSetupSet copia nello scope di ApplicationSetup gli argomenti del main argc e argv,
00063 //li analizza e salva le informazioni ottenute.
00064 //Gli altri oggetti del programma potranno accedere alle informazioni ottenute ottenendo un'istanza di
00065 ApplicationSetup.
00066 main_application_setup->ApplicationSetupSet (argc, argv);
00067 //Se l'utente ha avviato il programma con il flag -m tcp, esso deve essere eseguito in modalita' demone.
00068 if (main_application_setup->input_mode == ONLY_TCP)
00069 {
00070     process_id = fork ();
00071 }
00072 //L'espressione risulta vera se e' letta dal processo figlio o da processo padre eseguito senza il flag
00073 -m tcp
00074 if (process_id == 0)
00075 {
00076     //Crea e avvia l'oggetto digitizer_flow_control_main di tipo DigitizerFlowControl: l'oggetto in
00077     //questione gestisce l'intero flusso di esecuzione del programma.
00078     DigitizerFlowControl digitizer_flow_control_main;
00079     digitizer_flow_control_main.DigitizerFlowControlStart ();
00080 }
00081 remove(".plot_data.txt");
00082 }

```

## 6.59 OutputModule.cpp File Reference

```

#include "DefineGeneral.h"
#include "ConfObject.h"
#include "ApplicationSetup.h"
#include "OutputModule.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>

```

## 6.60 OutputModule.cpp

```

00001
00002 #include "DefineGeneral.h"
00003 #include "ConfObject.h"
00004 #include "ApplicationSetup.h"
00005 #include "OutputModule.h"
00006 #include <stdio.h>
00007 #include <stdlib.h>
00008 #include <string.h>
00009 #include <sys/socket.h>
00010 #include <arpa/inet.h>
00011 #include <unistd.h>
00012
00013 OutputModule *
00014 OutputModule::outputmodule_pInstance = NULL;
00015
00016 OutputModule *
00017 OutputModule::Instance ()
00018 {

```

```

00022     if (!outputmodule_pInstance) // Only allow one instance of class to be generated.
00023         outputmodule_pInstance = new OutputModule ();
00024
00025     return outputmodule_pInstance;
00026 }
00027
00028 OutputModule::OutputModule ()
00029 {
00030     OutputModule::OutputModuleSockidOff();
00031     OutputModule::OutputModuleStdoutOn();
00032
00033     bzero (buffer, STANDARDBUFFERLIMIT);
00034
00035     int i;
00036     for (i = 0; i < 100; i++)
00037         sockid_array[i] = -1;
00038 }
00039
00040 void
00041 OutputModule::StdOutInsert (const char *string)
00042 {
00043     bzero (buffer, STANDARDBUFFERLIMIT);
00044     strncpy (buffer, string, STANDARDBUFFERLIMIT);
00045 }
00046
00047 void
00048 OutputModule::StdOutInsertLex (char *string, int length)
00049 {
00050     bzero (buffer, STANDARDBUFFERLIMIT);
00051     strncpy (buffer, string, STANDARDBUFFERLIMIT);
00052 }
00053
00054 void
00055 OutputModule::StdOutPrint ()
00056 {
00057     output_module_application_setup =
00058         ApplicationSetup::Instance ();
00059     //fprintf(stderr, "Codice dell'ultimo carattere: %d\n", buffer[strlen(buffer)-1]);
00060
00061     if ((output_module_application_setup->
00062         input_mode == 2
00063         || output_module_application_setup->
00064         input_mode == 0)
00065         && output_module_stdout == 1)
00066     {
00067         //if (buffer[strlen(buffer)-1] == 10)
00068             fprintf (stdout, "%s", buffer);
00069         //else if (buffer[strlen(buffer)-1] != 10){
00070             fprintf (stdout, "%s", buffer);
00071         //if (strlen(buffer) > 1) fprintf (stdout, "\n");
00072         //}
00073     } //void OutputModule::StdOutPrint ()
00074
00075 int
00076 OutputModule::TcpUserArrayInsert (int sockid)
00077 {
00078     int i;
00079     for (i = 0; i < 100; i++)
00080     {
00081         if (sockid_array[i] == -1)
00082         {
00083             sockid_array[i] = sockid;
00084             return 0;
00085         }
00086     }
00087     return 1;
00088 } //int OutputModule::TcpUserArrayInsert(int sockid)
00089
00090
00091 int
00092 OutputModule::TcpUserArrayDelete (int sockid)
00093 {
00094     int i;
00095     for (i = 0; i < 100; i++)
00096     {
00097         if (sockid_array[i] == sockid)
00098         {
00099             sockid_array[i] = -1;
00100             return 0;
00101         }
00102     }
00103     return 1;
00104 } //int OutputModule::TcpUserArrayDelete(int sockid)
00105

```

```

00106
00107 int
00108 OutputModule::TcpUserArraySendStdOut ()
00109 {
00110     int i;
00111     output_module_application_setup =
        ApplicationSetup::Instance ();
00112
00113     if (output_module_application_setup->input_mode == 2
00114         || output_module_application_setup->
            input_mode == 1)
00115     {
00116         for (i = 0; i < 100; i++)
00117         {
00118             if (sockid_array[i] > -1
00119                 && sockid_array[i] == output_module_sockid)
00120             {
00121                 send (sockid_array[i], buffer, 1000, 0);    //usleep(5000);
00122             } //if(sockid_array[i]>-1)
00123         } //for(i=0; i<100; i++)
00124     }
00125
00126     return 0;
00127 } //int OutputModule::TcpUserArraySendStdOut (StdOut * stdout_arg)
00128
00129 void OutputModule::Output(const char * string){
00130
00131     OutputModule::StdOutInsert (string);
00132     OutputModule::StdOutPrint ();
00133     OutputModule::TcpUserArraySendStdOut ();
00134 }
00135
00136
00137
00138 void OutputModule::OutputFlex(const char * string, int length)
00139 {
00140     char * new_line_string = (char*)malloc(length+1);
00141     strcpy(new_line_string, string);
00142     new_line_string[length] = new_line_string[length -1];
00143     new_line_string[length -1] = '\n';
00144     OutputModule::StdOutInsertLex(new_line_string, length);
00145     usleep(5000);
00146     OutputModule::StdOutPrint ();
00147     OutputModule::TcpUserArraySendStdOut ();
00148     free(new_line_string);
00149 }
00150
00151 void OutputModule::OutputModuleStdoutOn(){
00152     output_module_stdout = 1;
00153 }
00154
00155 void OutputModule::OutputModuleStdoutOff(){
00156     output_module_stdout = -1;
00157 }
00158
00159 void OutputModule::OutputModuleSockidOn(int sockid)
00160 {
00161     output_module_sockid = sockid;
00162 }
00163
00164 void OutputModule::OutputModuleSockidOff(){
00165     output_module_sockid = -1;
00166 }

```

## 6.61 OutputModule.h File Reference

```
#include "DefineGeneral.h"
```

### Classes

- class [OutputModule](#)

The *[OutputModule](#)* class provides an useful way to manage the output of the program.

## 6.62 OutputModule.h

```

00001
00015 #include "DefineGeneral.h"
00016
00017 class OutputModule
00018 {
00019
00020 private:
00027     OutputModule ();
00028
00032     static OutputModule *outputmodule_pInstance;
00033
00037     int output_module_stdout;
00038
00042     int output_module_sockid;
00043
00044 public:
00045
00049     ApplicationSetup * output_module_application_setup;
00050
00054     static OutputModule *Instance ();
00055
00059     char buffer[STANDARDBUFFERLIMIT];
00060
00064     void StdOutInsert (const char *string);
00065
00069     void StdOutInsertLex (char *string, int length);
00070
00074     void StdOutPrint ();
00075
00079     void OutputModuleStdoutOn();
00080
00084     void OutputModuleStdoutOff();
00085
00090     void OutputModuleSockidOn(int sockid);
00091
00095     void OutputModuleSockidOff();
00096
00100     int sockid_array[100];
00101
00105     int TcpUserArrayInsert (int sockid);
00106
00110     int TcpUserArrayDelete (int sockid);
00111
00115     int TcpUserArraySendStdOut ();
00116
00121     void Output(const char * string);
00122
00129     void OutputFlex(const char * string, int length);
00130 };

```

## 6.63 RawData.cpp File Reference

```

#include "DefineGeneral.h"
#include "X742DecodeRoutines.h"
#include "ConfObject.h"
#include "ApplicationSetup.h"
#include "DigitizerErrorObject.h"
#include "LogFile.h"
#include "DigitizerObject.h"
#include "Analizzatore.h"
#include "DigitizerObjectGeneric.h"
#include "RawData.h"
#include "DigitizerStateMachine.h"
#include "OutputModule.h"
#include <assert.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <CAENDigitizer.h>

```

## 6.64 RawData.cpp

```

00001
00005 #include "DefineGeneral.h"
00006 #include "X742DecodeRoutines.h"
00007 #include "ConfObject.h"
00008 #include "ApplicationSetup.h"
00009 #include "DigitizerErrorObject.h"
00010 #include "LogFile.h"
00011 #include "DigitizerObject.h"
00012 #include "Analizzatore.h"
00013 #include "DigitizerObjectGeneric.h"
00014 #include "RawData.h"
00015 #include "DigitizerStateMachine.h"
00016 #include "OutputModule.h"
00017 #include <assert.h>
00018 #include <stdio.h>
00019 #include <stdlib.h>
00020 #include <string.h>
00021 #include <CAENDigitizer.h>
00022
00023 RawData & RawData::operator= (const RawData & p)
00024 {
00025     int
00026     i;
00027     if (this != &p)
00028     {
00029         free (buffer);
00030         buffer = NULL;
00031         buffer = (char *) malloc (size);
00032         for (i = 0; i < p.bsize; i++)
00033             buffer[i] = p.buffer[i];
00034         imset = p.imset;
00035         handle = p.handle;
00036         bsize = p.bsize;
00037         size = p.size;
00038         ret = p.ret;
00039         eventInfo = p.eventInfo;
00040         ret_error = p.ret_error;
00041         Mode = p.Mode;
00042         file = p.file;
00043     }
00044     return *this;
00045 }
00046
00047
00048 RawData::RawData ()
00049 {
00050     buffer = NULL;
00051     imset = 0;
00052 }
00053
00054
00055 void
00056 RawData::RawDataSet (DigitizerObjectGeneric digitizer_arg)
00057 {
00058     digitizer = digitizer_arg;
00059
00060     handle = digitizer_arg.handle;
00061     ret = CAEN_DGTZ_MallocReadoutBuffer (handle, &buffer, (uint32_t *) &
size);
00062     //ret_error.digitizer_error_object_print_error (ret);
00063 }
00064
00065
00066 void
00067 RawData::RawDataDel ()
00068 {
00069     //ret = CAEN_DGTZ_FreeReadoutBuffer (&buffer);
00070     free (buffer);
00071     buffer = NULL;
00072     //ret_error.digitizer_error_object_print_error (ret);
00073 }
00074
00075
00076 void
00077 RawData::RawDataWriteOnFile (const char *file_arg)
00078 {
00079     if (bsize > 0)
00080     {
00081         FILE *file_size;
00082         char *file_size_path;
00083         file_size_path = (char *) malloc (strlen (file_arg) + 3);
00084         strcpy (file_size_path, file_arg);
00085         strcat (file_size_path, "sz");
00086         file = fopen (file_arg, "a");

```

```

00087     file_size = fopen (file_size_path, "a");
00088     if (file != NULL && file_size != NULL)
00089     {
00090         fprintf (file_size, "%d\n", bsize);
00091         fwrite (buffer, bsize, 1, file);
00092         fclose (file);
00093         fclose (file_size);
00094     } //if (file != NULL && file_size != NULL)
00095     } //if (bsize>0)
00096 }
00097
00098
00099 void
00100 RawData::RawDataWriteOnFile (FILE * file, FILE * file_size)
00101 {
00102     if (file != NULL && file_size != NULL)
00103     {
00104         fprintf (file_size, "%d\n", bsize);
00105         fwrite (buffer, bsize, 1, file);
00106     }
00107 }
00108
00109
00110 void
00111 RawData::RawDataWriteDecodeEventOnPlotFile (const char *file_arg)
00112 {
00113     FILE *file_private_punt;
00114     file_private_punt = fopen (file_arg, "w");
00115
00116     ApplicationSetup *application_setup;
00117     application_setup = ApplicationSetup::Instance ();
00118
00119     if (file_private_punt != NULL)
00120     {
00121         CAEN_DGTZ_UINT8_EVENT_t *Evt8 = NULL;
00122         CAEN_DGTZ_UINT16_EVENT_t *Evt16 = NULL;
00123         CAEN_DGTZ_X742_EVENT_t *Evt742 = NULL;
00124         int max_channels = 0;
00125         int max_groups = 0;
00126
00127         int FamilyCode = digitizer.BoardInfo.FamilyCode;
00128
00129         int FormFactor = digitizer.BoardInfo.FormFactor;
00130
00131         int i, j, k, o;
00132         uint32_t numEvents;
00133         int conta = 0;
00134         char *evtptr;
00135
00136         switch(FamilyCode) {
00137             case CAEN_DGTZ_XX724_FAMILY_CODE:
00138             case CAEN_DGTZ_XX781_FAMILY_CODE:
00139             case CAEN_DGTZ_XX720_FAMILY_CODE:
00140             case CAEN_DGTZ_XX721_FAMILY_CODE:
00141             case CAEN_DGTZ_XX751_FAMILY_CODE:
00142             case CAEN_DGTZ_XX761_FAMILY_CODE:
00143             case CAEN_DGTZ_XX731_FAMILY_CODE:
00144                 switch(FormFactor) {
00145                     case CAEN_DGTZ_VME64_FORM_FACTOR:
00146                     case CAEN_DGTZ_VME64X_FORM_FACTOR:
00147                         max_channels = 8;
00148                         break;
00149                     case CAEN_DGTZ_DESKTOP_FORM_FACTOR:
00150                     case CAEN_DGTZ_NIM_FORM_FACTOR:
00151                         max_channels = 4;
00152                         break;
00153                 }
00154             }
00155             break;
00156             case CAEN_DGTZ_XX730_FAMILY_CODE:
00157                 switch(FormFactor) {
00158                     case CAEN_DGTZ_VME64_FORM_FACTOR:
00159                     case CAEN_DGTZ_VME64X_FORM_FACTOR:
00160                         max_channels = 16;
00161                         break;
00162                     case CAEN_DGTZ_DESKTOP_FORM_FACTOR:
00163                     case CAEN_DGTZ_NIM_FORM_FACTOR:
00164                         max_channels = 8;
00165                         break;
00166                 }
00167             }
00168             break;
00169             case CAEN_DGTZ_XX740_FAMILY_CODE:
00170                 switch(FormFactor) {
00171                     case CAEN_DGTZ_VME64_FORM_FACTOR:
00172                     case CAEN_DGTZ_VME64X_FORM_FACTOR:
00173                         max_channels = 64;
00174                         break;

```

```

00174         case CAEN_DGTZ_DESKTOP_FORM_FACTOR:
00175         case CAEN_DGTZ_NIM_FORM_FACTOR:
00176             max_channels = 32;
00177             break;
00178     }
00179     break;
00180     case CAEN_DGTZ_XX743_FAMILY_CODE:
00181         switch(FormFactor) {
00182             case CAEN_DGTZ_VME64_FORM_FACTOR:
00183             case CAEN_DGTZ_VME64X_FORM_FACTOR:
00184                 max_channels = 16;
00185                 break;
00186             case CAEN_DGTZ_DESKTOP_FORM_FACTOR:
00187             case CAEN_DGTZ_NIM_FORM_FACTOR:
00188                 max_channels = 8;
00189                 break;
00190         }
00191     break;
00192     case CAEN_DGTZ_XX742_FAMILY_CODE:
00193         switch(FormFactor) {
00194             case CAEN_DGTZ_VME64_FORM_FACTOR:
00195             case CAEN_DGTZ_VME64X_FORM_FACTOR:
00196                 max_groups = 4;
00197                 break;
00198             case CAEN_DGTZ_DESKTOP_FORM_FACTOR:
00199             case CAEN_DGTZ_NIM_FORM_FACTOR:
00200                 max_groups = 2;
00201                 break;
00202         }
00203     break;
00204     //default:
00205     //assert("This program cannot be used with this digitizer family\n");
00206 }
00207
00209 ret = CAEN_DGTZ_GetNumEvents (handle, buffer, bsize, &numEvents);
00210 conta += numEvents;
00211
00213 for (i = 0; (unsigned int) i < numEvents; i++)
00214 {
00215     ret = CAEN_DGTZ_GetEventInfo (handle, buffer, bsize, i, &
eventInfo,
00216                                     &evtptr);
00217
00219     if (FamilyCode != CAEN_DGTZ_XX742_FAMILY_CODE)
00220     {
00221
00222         if (FamilyCode == CAEN_DGTZ_XX721_FAMILY_CODE || FamilyCode == CAEN_DGTZ_XX731_FAMILY_CODE)
00223         {
00224             ret = CAEN_DGTZ_DecodeEvent (handle, evtptr, (void **)&Evt8);
00225             if (application_setup->channel_visualized<max_channels)
00226             {
00227                 for (j=0; j<Evt8->ChSize[application_setup->
channel_visualized]; j++)
00228                 {
00229                     fprintf(file_private_punt, "%d\n",
00230 (int) Evt8->DataChannel[application_setup->
channel_visualized][j]);
00231                 }
00232             }
00233             CAEN_DGTZ_FreeEvent (handle, (void **) &Evt8);
00234         }
00235         else
00236         {
00237             ret = CAEN_DGTZ_DecodeEvent (handle, evtptr, (void **)&Evt16);
00238             if (application_setup->channel_visualized<max_channels)
00239             {
00240                 for (j=0; j<Evt16->ChSize[application_setup->
channel_visualized]; j++)
00241                 {
00242                     fprintf(file_private_punt, "%d\n",
00243 (int) Evt16->DataChannel[application_setup->
channel_visualized][j]);
00244                 }
00245             }
00246             CAEN_DGTZ_FreeEvent (handle, (void **) &Evt16);
00247         }
00248     }
00249
00251     else
00252     {
00253         X742_DecodeEvent (evtptr, (void **)&Evt742);
00254         printf("gruppo: %d\n canale: %d\n", application_setup->
channel_visualized / MAX_X742_CHANNEL_SIZE, application_setup->
channel_visualized % MAX_X742_CHANNEL_SIZE);
00255         if ((application_setup->channel_visualized / MAX_X742_CHANNEL_SIZE) <
MAX_X742_GROUP_SIZE)
00256     {

```



```

00257         if (Evt742->GrPresent[application_setup->
channel_visualized / MAX_X742_CHANNEL_SIZE] == 1)
00258         {
00259             //bzero(stringa, STANDARDBUFFERLIMIT);
00260             //snprintf(stringa, STANDARDBUFFERLIMIT, "####GROUP: %d\n", o);
00261             //OUTPUT(stringa);
00262             //if (application_setup->channel_visualized % MAX_X742_CHANNEL_SIZE <
MAX_X742_CHANNEL_SIZE)
00263                 //{
00264                 //bzero(stringa, STANDARDBUFFERLIMIT);
00265                 //snprintf(stringa, STANDARDBUFFERLIMIT, "####CHANNEL: %d\n", j);
00266                 //OUTPUT(stringa);
00267                 for (k = 0; k < Evt742->DataGroup[application_setup->
channel_visualized / MAX_X742_CHANNEL_SIZE].
00268                     ChSize[application_setup->
channel_visualized % MAX_X742_CHANNEL_SIZE] ; k++)
00269                 {
00270                     fprintf(file_private_punt, "%d\n",
00271                         (int) Evt742->DataGroup[application_setup->
channel_visualized / MAX_X742_CHANNEL_SIZE].
00272                             DataChannel[application_setup->
channel_visualized % MAX_X742_CHANNEL_SIZE][k]);
00273                     } //for (k = 0; k < Evt->DataGroup[i].ChSize[j] ; k++)
00274                     //} //if (application_setup->channel_visualizedMAX_X742_CHANNEL_SIZE)
00275                     } //if (Evt742->GrPresent[application_setup->channel_visualized /
MAX_X742_CHANNEL_SIZE] == 1)
00276                     } //if ((application_setup->channel_visualized / MAX_X742_CHANNEL_SIZE) <
MAX_X742_GROUP_SIZE)
00277                     CAEN_DGTZ_FreeEvent(handle, (void **) &Evt742);
00278                 } // END OF ELSE
00279
00280                 //fprintf(stderr, "Un segnale: %d\n", (int) Evt->DataChannel[0][0]);
00281
00282                 } //for (i = 0; (unsigned int) i < numEvents; i++)
00283
00284                 fclose (file_private_punt);
00285             } // if (file_private_punt != NULL)
00286         }
00287
00288
00289
00290 void
00291 RawData::RawDataPlot (const char *file_arg, FILE * gnuplot)
00292 {
00293
00294     uint32_t numEvents = 0;
00295
00296     ret = CAEN_DGTZ_GetNumEvents (handle, buffer, bsize, &numEvents);
00297
00298     if (numEvents > 0)
00299     {
00300         RawData::RawDataWriteDecodeEventOnPlotFile (file_arg);
00301         fprintf (gnuplot, "plot '%s'\n", file_arg);
00302         // This line is fundamental in order to send data to the gnuplot program in real-time.
00303         fflush (gnuplot);
00304         sleep (1);
00305
00306     }
00307
00308 }
00309
00310
00311
00312 void
00313 RawData::RawDataRead ()
00314 {
00315     ret =
00316         CAEN_DGTZ_ReadData (handle, CAEN_DGTZ_SLAVE_TERMINATED_READOUT_MBLT,
00317             buffer, (uint32_t *) & bsize);
00318     //ret_error.digitizer_error_object_print_error (ret);
00319 }
00320
00321
00322
00323 void
00324 RawData::RawDataDecode ()
00325 {
00326
00327     CAEN_DGTZ_UINT8_EVENT_t *Evt8 = NULL;
00328     CAEN_DGTZ_UINT16_EVENT_t *Evt16 = NULL;
00329     CAEN_DGTZ_X742_EVENT_t *Evt742 = NULL;
00330
00331     char stringa[STANDARDBUFFERLIMIT];
00332     bzero (stringa, STANDARDBUFFERLIMIT);
00333
00334     OutputModule *output_module;
00335     output_module = OutputModule::Instance ();

```

```
00336
00337 int max_channels = 0;
00338 int max_groups = 0;
00339
00341
00342 int FamilyCode = digitizer.BoardInfo.FamilyCode;
00343
00344 int FormFactor = digitizer.BoardInfo.FormFactor;
00345
00347
00348 int i, j, k, o;
00349 uint32_t numEvents;
00350 int conta = 0;
00351 char *evtptr;
00352
00353     switch(FamilyCode) {
00354     case CAEN_DGTZ_XX724_FAMILY_CODE:
00355     case CAEN_DGTZ_XX781_FAMILY_CODE:
00356     case CAEN_DGTZ_XX720_FAMILY_CODE:
00357     case CAEN_DGTZ_XX721_FAMILY_CODE:
00358     case CAEN_DGTZ_XX751_FAMILY_CODE:
00359     case CAEN_DGTZ_XX761_FAMILY_CODE:
00360     case CAEN_DGTZ_XX731_FAMILY_CODE:
00361         switch(FormFactor) {
00362         case CAEN_DGTZ_VME64_FORM_FACTOR:
00363         case CAEN_DGTZ_VME64X_FORM_FACTOR:
00364             max_channels = 8;
00365             break;
00366         case CAEN_DGTZ_DESKTOP_FORM_FACTOR:
00367         case CAEN_DGTZ_NIM_FORM_FACTOR:
00368             max_channels = 4;
00369             break;
00370         }
00371         break;
00372     case CAEN_DGTZ_XX730_FAMILY_CODE:
00373         switch(FormFactor) {
00374         case CAEN_DGTZ_VME64_FORM_FACTOR:
00375         case CAEN_DGTZ_VME64X_FORM_FACTOR:
00376             max_channels = 16;
00377             break;
00378         case CAEN_DGTZ_DESKTOP_FORM_FACTOR:
00379         case CAEN_DGTZ_NIM_FORM_FACTOR:
00380             max_channels = 8;
00381             break;
00382         }
00383         break;
00384     case CAEN_DGTZ_XX740_FAMILY_CODE:
00385         switch(FormFactor) {
00386         case CAEN_DGTZ_VME64_FORM_FACTOR:
00387         case CAEN_DGTZ_VME64X_FORM_FACTOR:
00388             max_channels = 64;
00389             break;
00390         case CAEN_DGTZ_DESKTOP_FORM_FACTOR:
00391         case CAEN_DGTZ_NIM_FORM_FACTOR:
00392             max_channels = 32;
00393             break;
00394         }
00395         break;
00396     case CAEN_DGTZ_XX743_FAMILY_CODE:
00397         switch(FormFactor) {
00398         case CAEN_DGTZ_VME64_FORM_FACTOR:
00399         case CAEN_DGTZ_VME64X_FORM_FACTOR:
00400             max_channels = 16;
00401             break;
00402         case CAEN_DGTZ_DESKTOP_FORM_FACTOR:
00403         case CAEN_DGTZ_NIM_FORM_FACTOR:
00404             max_channels = 8;
00405             break;
00406         }
00407         break;
00408     case CAEN_DGTZ_XX742_FAMILY_CODE:
00409         switch(FormFactor) {
00410         case CAEN_DGTZ_VME64_FORM_FACTOR:
00411         case CAEN_DGTZ_VME64X_FORM_FACTOR:
00412             max_groups = 4;
00413             break;
00414         case CAEN_DGTZ_DESKTOP_FORM_FACTOR:
00415         case CAEN_DGTZ_NIM_FORM_FACTOR:
00416             max_groups = 2;
00417             break;
00418         }
00419         break;
00420     //default:
00421     //assert("This program cannot be used with this digitizer family\n");
00422     }
00423
00424
```

```

00426     ret = CAEN_DGTZ_GetNumEvents (handle, buffer, bsize, &numEvents);
00427     conta += numEvents;
00428
00430     for (i = 0; (unsigned int) i < numEvents; i++)
00431     {
00432         ret = CAEN_DGTZ_GetEventInfo (handle, buffer, bsize, i, &
eventInfo,
00433                                     &evtptr);
00434
00436         if (FamilyCode != CAEN_DGTZ_XX742_FAMILY_CODE)
00437         {
00438
00439             if (FamilyCode == CAEN_DGTZ_XX721_FAMILY_CODE || FamilyCode == CAEN_DGTZ_XX731_FAMILY_CODE)
00440             {
00441                 ret = CAEN_DGTZ_DecodeEvent (handle, evtptr, (void **)&Evt8);
00442                 for (o=0; o<max_channels; o++)
00443                 {
00444                     for (j=0; j<Evt8->ChSize[o]; j++)
00445                     {
00446                         //bzero(stringa, STANDARDBUFFERLIMIT);
00447                         //snprintf(stringa, STANDARDBUFFERLIMIT, "Valore del digitizer: %d\n",
00448                             //(int) Evt8->DataChannel[o][j]);
00449                         //output_module->Output(stringa);
00450                     }
00451                 }
00452                 CAEN_DGTZ_FreeEvent (handle, (void **) &Evt8);
00453             }
00454             else
00455             {
00456                 ret = CAEN_DGTZ_DecodeEvent (handle, evtptr, (void **)&Evt16);
00457                 for (o=0; o<max_channels; o++)
00458                 {
00459                     for (j=0; j<Evt16->ChSize[o]; j++)
00460                     {
00461                         //bzero(stringa, STANDARDBUFFERLIMIT);
00462                         //snprintf(stringa, STANDARDBUFFERLIMIT, "Valore del digitizer: %d\n",
00463                             //(int) Evt16->DataChannel[o][j]);
00464                         //output_module->Output(stringa);
00465                     }
00466                 }
00467                 CAEN_DGTZ_FreeEvent (handle, (void **) &Evt16);
00468             }
00469         }
00471         else
00472         {
00473             X742_DecodeEvent (evtptr, (void **)&Evt742);
00474             for (o=0; o<max_groups; o++)
00475             {
00476                 if (Evt742->GrPresent[o] == 1)
00477                 {
00478                     //bzero(stringa, STANDARDBUFFERLIMIT);
00479                     //snprintf(stringa, STANDARDBUFFERLIMIT, "####GROUP: %d\n", o);
00480                     //OUTPUT(stringa);
00481                     for (j=0; j<MAX_X742_CHANNEL_SIZE; j++)
00482                     {
00483                         //bzero(stringa, STANDARDBUFFERLIMIT);
00484                         //snprintf(stringa, STANDARDBUFFERLIMIT, "####CHANNEL: %d\n", j);
00485                         //OUTPUT(stringa);
00486                         for (k = 0; k < Evt742->DataGroup[o].ChSize[j] ; k++)
00487                         {
00488                             //bzero(stringa, STANDARDBUFFERLIMIT);
00489                             //snprintf(stringa, STANDARDBUFFERLIMIT, "Valore del digitizer: %d\n",
00490                                 //(int) Evt742->DataGroup[o].DataChannel[j][k]);
00491                             //OUTPUT(stringa);
00492                             } //for (k = 0; k < Evt->DataGroup[i].ChSize[j] ; k++)
00493                         } //for (j=0; j<MAX_X742_CHANNEL_SIZE; j++)
00494                     } //if (Evt->GrPresent[o] == 1)
00495                 } //for (o=0; o<MAX_X742_GROUP_SIZE; o++)
00496                 CAEN_DGTZ_FreeEvent (handle, (void **) &Evt742);
00497             } // END OF ELSE
00498
00499             //fprintf(stderr, "Un segnale: %d\n", (int) Evt->DataChannel[0][0]);
00500
00501         } //for (i = 0; (unsigned int) i < numEvents; i++)
00502 }

```

## 6.65 RawData.h File Reference

```

#include <assert.h>
#include <stdio.h>
#include <CAENDigitizer.h>

```

## Classes

- class [RawData](#)

The [RawData](#) class manages the readout from the digitizer.

## 6.66 RawData.h

```

00001
00012 #include <assert.h>
00013 #include <stdio.h>
00014 #include <CAENDigitizer.h>
00015
00016 class RawData
00017 {
00018
00019 public:
00020
00024     CAEN_DGTZ_ErrorCode ret;
00025
00029     CAEN_DGTZ_EventInfo_t eventInfo;
00030
00034     DigitizerErrorObject ret_error;
00035
00039     int handle;
00040
00045     int size;
00046
00050     CAEN_DGTZ_ReadMode_t Mode;
00051
00055     int imset;
00056
00060     FILE *file;
00061
00065     RawData ();
00066
00070     int bsize;
00071
00075     char *buffer;
00076
00080     DigitizerObjectGeneric digitizer;
00081
00082
00088     void RawDataSet (DigitizerObjectGeneric digitizer_arg);
00089
00095     void RawDataDel ();
00096
00102     void RawDataRead ();
00103
00110     void RawDataWriteOnFile (const char *file_arg);
00111
00119     void RawDataWriteOnFile (FILE * file, FILE * file_size);
00120
00127     void RawDataWriteDecodeEventOnPlotFile (const char *file_arg);
00128
00137     void RawDataPlot (const char *file_arg, FILE * gnuplot);
00138
00143     void RawDataDecode ();
00144
00151     RawData & operator= (const RawData & p);
00152 };

```

## 6.67 TcpUser.cpp File Reference

```

#include "DefineGeneral.h"
#include "TcpUser.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

```

## 6.68 TcpUser.cpp

```

00001
00005 #include "DefineGeneral.h"
00006 #include "TcpUser.h"
00007 #include <stdio.h>
00008 #include <stdlib.h>
00009 #include <string.h>
00010
00011 TcpUser::TcpUser ()
00012 {
00013     command_sent_by_user = 0;
00014     bzero (first_parameter, STANDARDBUFFERLIMIT);
00015     bzero (first_parameter, STANDARDBUFFERLIMIT);
00016     register_address = -1;
00017     register_data = -1;
00018 }

```

## 6.69 TcpUser.h File Reference

```
#include "DefineGeneral.h"
```

### Classes

- class [TcpUser](#)

The [TcpUser](#) class provides an useful way to store data about the users of the server.

## 6.70 TcpUser.h

```

00001
00011 #include "DefineGeneral.h"
00012
00013 class TcpUser
00014 {
00015 public:
00016
00021     char command_sent_by_user;
00022
00027     char first_parameter[STANDARDBUFFERLIMIT];
00028
00032     char second_parameter[STANDARDBUFFERLIMIT];
00033
00037     int register_address;
00038
00042     int register_data;
00043
00047     int user_sockid;
00048
00053     TcpUser ();
00054 };

```

## 6.71 X742DecodeRoutines.c File Reference

```

#include "X742DecodeRoutines.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

```

### Functions

- int32\_t [GetNumEvents](#) (char \*buffer, uint32\_t bufsize, uint32\_t \*numEvents)

- `int32_t GetEventPtr` (char \*buffer, uint32\_t bufsize, int32\_t numEvent, char \*\*EventPtr)
- `int32_t X742_DecodeEvent` (char \*evtPtr, void \*\*Evt)

### 6.71.1 Function Documentation

#### 6.71.1.1 `int32_t GetEventPtr` ( char \* *buffer*, uint32\_t *bufsize*, int32\_t *numEvent*, char \*\* *EventPtr* )

Definition at line 250 of file [X742DecodeRoutines.c](#).

#### 6.71.1.2 `int32_t GetNumEvents` ( char \* *buffer*, uint32\_t *bufsize*, uint32\_t \* *numEvents* )

Definition at line 208 of file [X742DecodeRoutines.c](#).

#### 6.71.1.3 `int32_t X742_DecodeEvent` ( char \* *evtPtr*, void \*\* *Evt* )

Definition at line 301 of file [X742DecodeRoutines.c](#).

## 6.72 X742DecodeRoutines.c

```

00001 #include "X742DecodeRoutines.h"
00002 #include <stdio.h>
00003 #include <stdlib.h>
00004 #include <string.h>
00005
00006
00007 static int32_t
00008 getNumberOfBits (uint8_t byte)
00009 {
00010
00011     uint32_t i, count;
00012
00013     count = 0;
00014
00015     for (i = 0; i < 8; i++)
00016     {
00017
00018         if ((byte >> i) & 0x1)
00019             count++;
00020
00021     }
00022
00023     return count;
00024
00025 }
00026
00027 static int
00028 V1742UnpackEventGroup (uint32_t group, uint32_t * datain,
00029                        CAEN_DGTZ_X742_GROUP_t * dataout)
00030 {
00031
00032
00033
00034     int i, j, rpnt = 0, wpnt = 0, size1, size2, trg = 0, k;
00035
00036     long samples;
00037
00038     float Time[1024], t0;
00039
00040     float Tsamp;
00041
00042     float vcorr;
00043
00044     uint16_t st_ind = 0;
00045
00046     uint32_t freq;
00047
00048     float wave_tmp[1024];
00049
00050
00051     freq = (datain[0] >> 16) & 0x3;
00052
00053     switch (freq)

```

```
00054     {
00055
00056     case CAEN_DGTZ_DRS4_2_5GHz:
00057
00058     Tsamp = (float) ((1.0 / 2500.0) * 1000.0);
00059
00060     break;
00061
00062     case CAEN_DGTZ_DRS4_1GHz:
00063
00064     Tsamp = (float) ((1.0 / 1000.0) * 1000.0);
00065
00066     break;
00067
00068     default:
00069
00070     Tsamp = (float) ((1.0 / 5000.0) * 1000.0);
00071
00072     break;
00073
00074 }
00075
00076
00077 st_ind = (uint16_t) ((datain[0] >> 20) & 0x3FF);
00078
00079 size1 = datain[0] & 0xFFF;
00080
00081 if ((trg = (datain[0] >> 12) & 0x1) == 1)
00082
00083 size2 = (datain[0] >> 3) & 0x1FF;
00084
00085     else
00086
00087 size2 = 0;
00088
00089
00090 dataout->TriggerTimeTag = datain[size1 + size2 + 1] & 0xFFFFFFFF;
00091
00092
00093 samples = ((long) (size1 / 3));
00094
00095
00096 while (rpnt < size1)
00097     {
00098
00099
00100     switch (rpnt % 3)
00101     {
00102
00103     case 0:
00104
00105     dataout->DataChannel[0][wpnt] = (float) (datain[rpnt + 1] & 0x00000FFF); /* S0[11:0] - CH0 */
00106
00107     dataout->DataChannel[1][wpnt] = (float) ((datain[rpnt + 1] & 0x00FFF000) >> 12); /* S0[11:0] - CH1 */
00108
00109     dataout->DataChannel[2][wpnt] = (float) ((datain[rpnt + 1] & 0xFF000000) >> 24); /* S0[ 7:0] - CH2 */
00110
00111     break;
00112
00113     case 1:
00114
00115     dataout->DataChannel[2][wpnt] +=
00116         (float) ((datain[rpnt + 1] & 0x0000000F) << 8);
00117
00118     dataout->DataChannel[3][wpnt] = (float) ((datain[rpnt + 1] & 0x0000FFF0) >> 4); /* S0[11:0] - CH3 */
00119
00120     dataout->DataChannel[4][wpnt] = (float) ((datain[rpnt + 1] & 0x0FFF0000) >> 16); /* S0[11:0] - CH4 */
00121
00122     dataout->DataChannel[5][wpnt] = (float) ((datain[rpnt + 1] & 0xF0000000) >> 28); /* S0[3:0] - CH5 */
00123
00124     break;
00125
00126     case 2:
00127
00128     dataout->DataChannel[5][wpnt] +=
00129         (float) ((datain[rpnt + 1] & 0x000000FF) << 4);
00130
00131     dataout->DataChannel[6][wpnt] = (float) ((datain[rpnt + 1] & 0x000FFF00) >> 8); /* S0[11:0] - CH6 */
00132
00133     dataout->DataChannel[7][wpnt] = (float) ((datain[rpnt + 1] & 0xFFFF0000) >> 20); /* S0[11:0] - CH7 */
00134
00135     wpnt++;
00136
00137     break;
00138
00139     }
00140
```

```

00141 rpnt++;
00142
00143 }
00144
00145 rpnt++;
00146
00147 for (k = 0; k < 8; k++)
00148     dataout->ChSize[k] = wpnt;
00149
00150 wpnt = 0;
00151
00152
00153 for (i = 0; i < size2; i++)
00154 {
00155
00156     switch (i % 3)
00157     {
00158
00159     case 0:
00160
00161         dataout->DataChannel[8][wpnt] = (float) (datain[rpnt + i] & 0x00000FFF);    /* S0 - CH8 */
00162
00163         dataout->DataChannel[8][++wpnt] = (float) ((datain[rpnt + i] & 0x00FFF000) >> 12);    /* S1 - CH8 */
00164
00165         dataout->DataChannel[8][++wpnt] = (float) ((datain[rpnt + i] & 0xFF000000) >> 24);    /* S2[ 7:0] - CH8 */
00166
00167         break;
00168
00169     case 1:
00170
00171         dataout->DataChannel[8][wpnt] +=
00172             (float) ((datain[rpnt + i] & 0x0000000F) << 8);
00173
00174         dataout->DataChannel[8][++wpnt] = (float) ((datain[rpnt + i] & 0x0000FFF0) >> 4);    /* S3 - CH8 */
00175
00176         dataout->DataChannel[8][++wpnt] = (float) ((datain[rpnt + i] & 0x0FFF0000) >> 16);    /* S4 - CH8 */
00177
00178         dataout->DataChannel[8][++wpnt] = (float) ((datain[rpnt + i] & 0xFF000000) >> 28);    /* S5[3:0] - CH8 */
00179
00180         break;
00181
00182     case 2:
00183
00184         dataout->DataChannel[8][wpnt] += (float) ((datain[rpnt + i] & 0x000000FF) << 4);    /* S5[11:4] - CH8 */
00185
00186         dataout->DataChannel[8][++wpnt] = (float) ((datain[rpnt + i] & 0x000FFF00) >> 8);    /* S6[11:0] - CH8 */
00187
00188         dataout->DataChannel[8][++wpnt] = (float) ((datain[rpnt + i] & 0xFFFF0000) >> 20);    /* S7[11:0] - CH8 */
00189
00190         wpnt++;
00191
00192         break;
00193
00194     }
00195
00196     }
00197
00198     dataout->ChSize[8] = wpnt;
00199
00200     dataout->StartIndexCell = (uint16_t) st_ind;
00201
00202     return (size1 + size2 + 2);
00203
00204 }
00205
00206
00207
00208 int32_t GetNumEvents (char *buffer, uint32_t buffsize,
00209                     uint32_t * numEvents)
00210 {
00211
00212     uint32_t i = 0, evtSize;
00213
00214     int ret;
00215
00216     int32_t counter = -1;
00217
00218     if ((buffsize == 0) || (buffer == NULL))
00219     {
00220
00221         *numEvents = 0;
00222
00223         return 0;
00224
00225     }
00226
00227     if (buffsize < EVENT_HEADER_SIZE)

```



```
00228     return -1;
00229
00230     do
00231     {
00232
00233 counter++;
00234
00235 evtSize = *(long *) (buffer + i) & 0xFFFFFFFF;
00236
00237 i += (uint32_t) (evtSize * 4);
00238
00239 }
00240 while ((i + EVENT_HEADER_SIZE) < buffsize);
00241
00242 *numEvents = counter + 1;
00243
00244 return 0;
00245
00246 }
00247
00248
00249
00250 int32_t GetEventPtr (char *buffer, uint32_t buffsize, int32_t numEvent,
00251                     char **EventPtr)
00252 {
00253
00254 uint32_t i = 0;
00255
00256 int32_t counter = -1;
00257
00258 int ret;
00259
00260 int evtSize;
00261
00262
00263 if ((buffer == NULL) || (buffsize < EVENT_HEADER_SIZE))
00264     return -1;
00265
00266     do
00267     {
00268
00269 counter++;
00270
00271 evtSize = *(long *) (buffer + i) & 0xFFFFFFFF;
00272
00273 if (counter == numEvent)
00274     {
00275
00276 if ((i + (uint32_t) evtSize) < buffsize)
00277     {
00278
00279 *EventPtr = (buffer + i);
00280
00281 return 0;
00282
00283 }
00284
00285         else
00286             return -1;
00287
00288 }
00289
00290 i += (uint32_t) (evtSize * 4);
00291
00292 }
00293 while ((i + EVENT_HEADER_SIZE) < buffsize);
00294
00295 return -1;
00296
00297 }
00298
00299
00300
00301 int32_t X742_DecodeEvent (char *evtPtr, void **Evt)
00302 {
00303
00304 CAEN_DGTZ_X742_EVENT_t * Event;
00305
00306 uint32_t * buffer;
00307
00308 char chanMask;
00309
00310 uint32_t j, g, size;
00311
00312 uint32_t * pBuffer;
00313
00314 uint32_t eventSize;
```

```
00315
00316 int evtSize, h;
00317
00318
00319 evtSize = *(long *) evtPtr & 0xFFFFFFFF;
00320
00321 chanMask = *(long *) (evtPtr + 4) & 0x0000000F;
00322
00323 evtPtr += EVENT_HEADER_SIZE;
00324
00325 buffer = (uint32_t *) evtPtr;
00326
00327 pBuffer = (uint32_t *) evtPtr;
00328
00329 eventSize = (evtSize * 4) - EVENT_HEADER_SIZE;
00330
00331 if (eventSize == 0)
00332     return -1;
00333
00334 Event =
00335     (CAEN_DGTZ_X742_EVENT_t *) malloc (sizeof (CAEN_DGTZ_X742_EVENT_t));
00336
00337 if (Event == NULL)
00338     return -1;
00339
00340 memset (Event, 0, sizeof (CAEN_DGTZ_X742_EVENT_t));
00341
00342 for (g = 0; g < X742_MAX_GROUPS; g++)
00343 {
00344
00345     if ((chanMask >> g) & 0x1)
00346     {
00347
00348         for (j = 0; j < MAX_X742_CHANNEL_SIZE; j++)
00349         {
00350
00351             Event->DataGroup[g].DataChannel[j] =
00352                 (float *) malloc (X742_FIXED_SIZE * sizeof (float));
00353
00354             if (Event->DataGroup[g].DataChannel[j] == NULL)
00355             {
00356
00357                 for (h = j - 1; h > -1; h++)
00358                     free (Event->DataGroup[g].DataChannel[h]);
00359
00360                 return -1;
00361             }
00362         }
00363     }
00364
00365     size = V1742UnpackEventGroup (g, pBuffer, &(Event->DataGroup[g]));
00366
00367     pBuffer += size;
00368
00369     Event->GrPresent[g] = 1;
00370
00371 }
00372
00373     else
00374     {
00375
00376         Event->GrPresent[g] = 0;
00377
00378         for (j = 0; j < MAX_X742_CHANNEL_SIZE; j++)
00379         {
00380
00381             Event->DataGroup[g].DataChannel[j] = NULL;
00382
00383         }
00384     }
00385
00386 }
00387
00388 }
00389
00390 *Evt = Event;
00391
00392 return 0;
00393
00394 }
00395
00396
```

## 6.73 X742DecodeRoutines.h File Reference

```
#include <CAENDigitizer.h>
```

### Macros

- `#define EVENT_HEADER_SIZE 0x10`
- `#define X742_MAX_GROUPS 0x04`
- `#define X742_FIXED_SIZE 0x400`

### Functions

- `int32_t GetNumEvents (char *buffer, uint32_t bufsize, uint32_t *numEvents)`
- `int32_t GetEventPtr (char *buffer, uint32_t bufsize, int32_t numEvent, char **EventPtr)`
- `int32_t X742_DecodeEvent (char *evtPtr, void **Evt)`

#### 6.73.1 Macro Definition Documentation

##### 6.73.1.1 `#define EVENT_HEADER_SIZE 0x10`

Definition at line 3 of file [X742DecodeRoutines.h](#).

##### 6.73.1.2 `#define X742_FIXED_SIZE 0x400`

Definition at line 7 of file [X742DecodeRoutines.h](#).

##### 6.73.1.3 `#define X742_MAX_GROUPS 0x04`

Definition at line 5 of file [X742DecodeRoutines.h](#).

#### 6.73.2 Function Documentation

##### 6.73.2.1 `int32_t GetEventPtr ( char * buffer, uint32_t bufsize, int32_t numEvent, char ** EventPtr )`

Definition at line 250 of file [X742DecodeRoutines.c](#).

##### 6.73.2.2 `int32_t GetNumEvents ( char * buffer, uint32_t bufsize, uint32_t * numEvents )`

Definition at line 208 of file [X742DecodeRoutines.c](#).

##### 6.73.2.3 `int32_t X742_DecodeEvent ( char * evtPtr, void ** Evt )`

Definition at line 301 of file [X742DecodeRoutines.c](#).

## 6.74 X742DecodeRoutines.h

```

00001 #include <CAENDigitizer.h>
00002
00003 #define EVENT_HEADER_SIZE    0x10
00004
00005 #define X742_MAX_GROUPS     0x04
00006
00007 #define X742_FIXED_SIZE     0x400
00008
00009 /*****
00010 * GetNumEvents(char *buffer, uint32_t buffsize, uint32_t *numEvents)
00011 * Gets current number of event stored in the acquisition buffer
00012 *
00013 * [IN] buffer      : Address of the acquisition buffer
00014 * [IN] bufferSize : Size of the data stored in the acquisition buffer
00015 * [OUT] numEvents  : Number of events stored in the acquisition buffer
00016 *                  : return 0 = Success;
00017 *****/
00018 int32_t GetNumEvents (char *buffer, uint32_t buffsize,
00019                     uint32_t * numEvents);
00020
00021
00022 /*****
00023 * GetEventPtr(char *buffer, uint32_t buffsize, int32_t numEvent, char **EventPtr)
00024 * Retrieves the event pointer of a specified event in the acquisition buffer
00025 *
00026 * [IN] buffer      : Address of the acquisition buffer
00027 * [IN] bufferSize : Acquisition buffer size (in samples)
00028 * [IN] numEvents  : Number of events stored in the acquisition buffer
00029 * [OUT] EventPtr  : Pointer to the requested event in the acquisition buffer
00030 *                  : return 0 = Success;
00031 *****/
00032 int32_t GetEventPtr (char *buffer, uint32_t buffsize, int32_t numEvent,
00033                    char **EventPtr);
00034
00035
00036 /*****
00037 * X742_DecodeEvent(char *evtPtr, void **Evt)
00038 * Decodes a specified event stored in the acquisition buffer writing data in Evt memory
00039 * Once used the Evt memory MUST be deallocated by the caller!
00040 *
00041 * [IN] EventPtr : pointer to the requested event in the acquisition buffer (MUST BE NULL)
00042 * [OUT] Evt      : event structure with the requested event data
00043 *                  : return 0 = Success;
00044 *****/
00045 int32_t X742_DecodeEvent (char *evtPtr, void **Evt);
00046

```

# Index

## ALL

DefineGeneral.h, [137](#)

## Acquisition\_Conf1

CommunicationObject, [18](#)

DigitizerStateMachine, [57](#)

## Acquisition\_Conf2

CommunicationObject, [18](#)

DigitizerStateMachine, [57](#)

## Acquisition\_Mutex

CommunicationObject, [18](#)

## Address\_register

ConfObject, [24](#)

## AllInformation

AnalizzatoreUtils.c, [85](#)

AnalizzatoreUtils.h, [98](#)

## AnalizzaInit

Analizzatore.h, [81](#)

## AnalizzaInitPrint

Analizzatore.h, [83](#)

## AnalizzaPrint

Analizzatore.h, [83](#)

## AnalizzaSetup

Analizzatore.h, [83](#)

## AnalizzaSetupPrint

Analizzatore.h, [83](#)

## Analizzatore.h, [81](#)

AnalizzaInit, [81](#)

AnalizzaInitPrint, [83](#)

AnalizzaPrint, [83](#)

AnalizzaSetup, [83](#)

AnalizzaSetupPrint, [83](#)

## AnalizzatoreUtils.c, [84](#)

AllInformation, [85](#)

ChInformation, [86](#)

ChannelEnableMask, [86](#)

ChannelTriggerEnableMask, [86](#)

FastInformation, [87](#)

FindIntegerValue, [87](#)

FindPointer, [87](#)

GetAcquisitionMode, [88](#)

GetOpenInformation, [88](#)

GetWriteRegisterInformation, [88](#)

GroupEnableMask, [88](#)

GroupInformation, [89](#)

OutputFileFormat, [89](#)

OutputNIMTTL, [89](#)

OutputRisingFalling, [90](#)

reg\_matches, [90](#)

YesNoAnswer, [90](#)

## AnalizzatoreUtils.h, [96](#)

AllInformation, [98](#)

ChInformation, [99](#)

ChannelEnableMask, [98](#)

ChannelTriggerEnableMask, [98](#)

FastInformation, [99](#)

FindIntegerValue, [99](#)

FindPointer, [100](#)

GetAcquisitionMode, [100](#)

GetOpenInformation, [100](#)

GetWriteRegisterInformation, [100](#)

GroupEnableMask, [101](#)

GroupInformation, [101](#)

OutputFileFormat, [101](#)

OutputNIMTTL, [103](#)

OutputRisingFalling, [103](#)

YesNoAnswer, [103](#)

## application\_setup\_conf\_file

ApplicationSetup, [11](#)

## application\_setup\_conf\_file\_path

ApplicationSetup, [11](#)

## application\_setup\_conf\_object

ApplicationSetup, [12](#)

## application\_setup\_data\_file\_path

ApplicationSetup, [12](#)

## application\_setup\_data\_file\_punt

ApplicationSetup, [12](#)

## application\_setup\_data\_file\_size\_path

ApplicationSetup, [12](#)

## application\_setup\_data\_file\_size\_punt

ApplicationSetup, [12](#)

## application\_setup\_input\_mode

ApplicationSetup, [12](#)

## application\_setup\_log\_file\_path

ApplicationSetup, [12](#)

## ApplicationSetup, [9](#)

application\_setup\_conf\_file, [11](#)

application\_setup\_conf\_file\_path, [11](#)

application\_setup\_conf\_object, [12](#)

application\_setup\_data\_file\_path, [12](#)

application\_setup\_data\_file\_punt, [12](#)

application\_setup\_data\_file\_size\_path, [12](#)

application\_setup\_data\_file\_size\_punt, [12](#)

application\_setup\_input\_mode, [12](#)

application\_setup\_log\_file\_path, [12](#)

ApplicationSetupDataFileModify, [10](#)

ApplicationSetupGetDataFilePunt, [10](#)

ApplicationSetupGetDataFileSizePunt, [10](#)

ApplicationSetupSet, [11](#)

- argc, [12](#)
  - ArgumentsParsing, [11](#)
  - argv, [12](#)
  - channel\_visualized, [13](#)
  - FetchInputMode, [11](#)
  - imset, [13](#)
  - input\_mode, [13](#)
  - Instance, [11](#)
- ApplicationSetup.cpp, [104](#)
- ApplicationSetup.h, [107](#)
- ApplicationSetupDataFileModify
  - ApplicationSetup, [10](#)
- ApplicationSetupGetDataFilePunt
  - ApplicationSetup, [10](#)
- ApplicationSetupGetDataFileSizePunt
  - ApplicationSetup, [10](#)
- ApplicationSetupSet
  - ApplicationSetup, [11](#)
- argc
  - ApplicationSetup, [12](#)
- ArgumentsParsing
  - ApplicationSetup, [11](#)
- argv
  - ApplicationSetup, [12](#)
- BlockedConsumerInput
  - CommunicationObject, [18](#)
  - DigitizerStateMachine, [57](#)
- BlockedProducerInput
  - CommunicationObject, [18](#)
  - DigitizerStateMachine, [57](#)
- BoardInfo
  - DigitizerObject, [44](#)
- bsize
  - RawData, [77](#)
- buffer
  - OutputModule, [72](#)
  - RawData, [77](#)
- CHANGECONF
  - DefineCommands.h, [133](#)
- CHANGEDATA
  - DefineCommands.h, [133](#)
- CHANGELOG
  - DefineCommands.h, [134](#)
- CHECK
  - DefineCommands.h, [134](#)
- CHKCONF
  - DefineCommands.h, [134](#)
- CLOSE
  - DefineCommands.h, [134](#)
- ChInformation
  - AnalizzatoreUtils.c, [86](#)
  - AnalizzatoreUtils.h, [99](#)
- channel\_enable\_mask
  - ConfObject, [24](#)
- channel\_trigger
  - ChannelObject, [14](#)
- channel\_visualized
  - ApplicationSetup, [13](#)
- ChannelEnableMask
  - AnalizzatoreUtils.c, [86](#)
  - AnalizzatoreUtils.h, [98](#)
- ChannelObject, [13](#)
  - channel\_trigger, [14](#)
  - ChannelObject, [14](#)
  - ChannelObject, [14](#)
  - dc\_offset, [14](#)
  - enable\_input, [14](#)
  - numChannel, [15](#)
  - PrintChannel, [14](#)
  - set, [15](#)
  - trigger\_threshold, [15](#)
- ChannelTriggerEnableMask
  - AnalizzatoreUtils.c, [86](#)
  - AnalizzatoreUtils.h, [98](#)
- channels
  - ConfObject, [24](#)
- circular\_buffer\_preprocessing
  - DigitizerStateMachine, [57](#)
- circular\_buffer\_raw\_data
  - DigitizerStateMachine, [58](#)
- circular\_buffer\_visualization
  - DigitizerStateMachine, [58](#)
- ClientApplication.c, [108](#)
  - comando\_da\_inviare, [111](#)
  - command\_parser, [109](#)
  - FindPointer, [109](#)
  - go, [111](#)
  - Help, [109](#)
  - i, [111](#)
  - inputline, [111](#)
  - main, [109](#)
  - n, [111](#)
  - no\_new\_line, [111](#)
  - receiveline, [111](#)
  - reg\_matches, [111](#)
  - ricevitore\_function, [111](#)
  - sendline, [111](#)
  - sockfd, [112](#)
- coda
  - CommunicationObject, [18](#)
- coda\_preprocessing
  - DigitizerStateMachine, [58](#)
- coda\_raw\_data
  - DigitizerStateMachine, [58](#)
- coda\_visualization
  - DigitizerStateMachine, [58](#)
- comando\_da\_inviare
  - ClientApplication.c, [111](#)
- command
  - CommunicationObject, [18](#)
- command\_parser
  - ClientApplication.c, [109](#)
- command\_sent\_by\_user
  - TcpUser, [79](#)
- CommunicationObject, [15](#)

- Acquisition\_Cond1, [18](#)
- Acquisition\_Cond2, [18](#)
- Acquisition\_Mutex, [18](#)
- BlockedConsumerInput, [18](#)
- BlockedProducerInput, [18](#)
- coda, [18](#)
- command, [18](#)
- CommunicationObject, [16](#)
- CommunicationObjectInit, [16](#)
- CommunicationObject, [16](#)
- Finish, [16](#)
- GetCommand, [17](#)
- go, [18](#)
- Main, [17](#)
- main\_thread, [18](#)
- num\_mex, [19](#)
- ReservedKeyBoardInputArea, [19](#)
- testa, [19](#)
- Worker, [17](#)
- worker\_thread, [19](#)
- CommunicationObject.cpp, [117](#)
- CommunicationObject.h, [120](#)
- CommunicationObjectInit
  - CommunicationObject, [16](#)
- ConetNode
  - ConfObject, [25](#)
- ConfObject, [22](#)
  - Address\_register, [24](#)
  - channel\_enable\_mask, [24](#)
  - channels, [24](#)
  - ConetNode, [25](#)
  - ConfObject, [24](#)
  - ConfObject, [24](#)
  - DSR4\_Frequency, [25](#)
  - Data\_register, [25](#)
  - dc\_offset, [25](#)
  - decimation\_factor, [25](#)
  - desmod, [25](#)
  - enable\_fast\_trigger\_digitizing, [25](#)
  - enable\_input, [25](#)
  - external\_trigger\_acquisition\_mode, [26](#)
  - fast\_trigger\_acquisition\_mode, [26](#)
  - fasts, [26](#)
  - gnuplot, [26](#)
  - group\_enable\_mask, [26](#)
  - groups, [26](#)
  - header\_yes\_no, [26](#)
  - LinkNumber, [26](#)
  - LinkType, [27](#)
  - Mask\_register, [27](#)
  - max\_num\_events\_BLT, [27](#)
  - nim\_ttl, [27](#)
  - output\_file\_format, [27](#)
  - post\_trigger, [27](#)
  - PrintAll, [24](#)
  - PrintAllHuman, [24](#)
  - record\_length, [27](#)
  - rising\_falling, [27](#)
  - self\_trigger\_enable\_mask, [28](#)
  - self\_trigger\_enable\_mask\_mode, [28](#)
  - test\_pattern, [28](#)
  - trigger\_threshold, [28](#)
  - use\_interrupt, [28](#)
  - VMEBaseAddress, [28](#)
- ConfObject.cpp, [125](#)
- ConfObject.h, [131](#)
- ConfigurationConsistence, [19](#)
  - ConfigurationConsistenceConfFileInit, [20](#)
  - ConfigurationConsistenceConfFileInitNoPrint, [20](#)
  - ConfigurationConsistenceConfFileSetupEssential-  
WithInitCheck, [20](#)
  - ConfigurationConsistenceConfFileSetupEssential-  
WithInitCheckNoPrint, [21](#)
  - ConfigurationConsistenceConfFileSetupEssential-  
WithoutInitCheck, [21](#)
  - ConfigurationConsistenceConfFileSetupEssential-  
WithoutInitCheckNoPrint, [21](#)
- ConfigurationConsistence.cpp, [121](#)
- ConfigurationConsistence.h, [124](#)
- ConfigurationConsistenceConfFileInit
  - ConfigurationConsistence, [20](#)
- ConfigurationConsistenceConfFileInitNoPrint
  - ConfigurationConsistence, [20](#)
- ConfigurationConsistenceConfFileSetupEssentialWith-  
InitCheck
  - ConfigurationConsistence, [20](#)
- ConfigurationConsistenceConfFileSetupEssentialWith-  
InitCheckNoPrint
  - ConfigurationConsistence, [21](#)
- ConfigurationConsistenceConfFileSetupEssential-  
WithoutInitCheck
  - ConfigurationConsistence, [21](#)
- ConfigurationConsistenceConfFileSetupEssential-  
WithoutInitCheckNoPrint
  - ConfigurationConsistence, [21](#)
- Consumatore\_Dispatcher
  - DigitizerStateMachine, [54](#)
- consumatore\_thread
  - DigitizerStateMachine, [58](#)
- DSR4\_Frequency
  - ConfObject, [25](#)
- Data\_register
  - ConfObject, [25](#)
- dc\_offset
  - ChannelObject, [14](#)
  - ConfObject, [25](#)
  - FastObject, [62](#)
  - GroupObject, [64](#)
- decimation\_factor
  - ConfObject, [25](#)
- DefineCommands.h, [133](#)
  - CHANGECONF, [133](#)
  - CHANGEDATA, [133](#)
  - CHANGELOG, [134](#)
  - CHECK, [134](#)
  - CHKCONF, [134](#)

- CLOSE, [134](#)
- INIT, [134](#)
- MORE, [134](#)
- PRESTART, [134](#)
- PRESTOP, [134](#)
- PRINT, [134](#)
- PRINTFILES, [134](#)
- QUIT, [134](#)
- RAWSTART, [134](#)
- RAWSTOP, [135](#)
- READREGISTER, [135](#)
- SEND, [135](#)
- SETUP, [135](#)
- START, [135](#)
- STATUS, [135](#)
- STOP, [135](#)
- VISTART, [135](#)
- VISTOP, [135](#)
- WRITEREGISTER, [135](#)
- DefineGeneral.h, [136](#)
  - ALL, [137](#)
  - MAXCHANNELOBJECT, [137](#)
  - MAXCOMMAND, [137](#)
  - MAXFASTOBJECT, [137](#)
  - MAXGROUPOBJECT, [137](#)
  - ONLYTCP, [137](#)
  - ONLYUSER, [137](#)
  - PREPROCESSINGQUEUE, [137](#)
  - RAWDATAQUEUE, [137](#)
  - STANDARDBUFFERLIMIT, [137](#)
  - VISUALIZATIONQUEUE, [137](#)
- desmod
  - ConfObject, [25](#)
- digitizer
  - DigitizerStateMachine, [58](#)
  - RawData, [77](#)
- digitizer\_flow\_control\_application\_setup
  - DigitizerFlowControl, [33](#)
- DigitizerErrorObject, [29](#)
  - DigitizerErrorObjectDebugging, [29](#)
  - DigitizerErrorObjectDebuggingLog, [29](#)
  - DigitizerErrorObjectPrintError, [30](#)
- DigitizerErrorObject.cpp, [138](#)
- DigitizerErrorObject.h, [146](#)
- DigitizerErrorObjectDebugging
  - DigitizerErrorObject, [29](#)
- DigitizerErrorObjectDebuggingLog
  - DigitizerErrorObject, [29](#)
- DigitizerErrorObjectPrintError
  - DigitizerErrorObject, [30](#)
- DigitizerFlowControl, [30](#)
  - digitizer\_flow\_control\_application\_setup, [33](#)
  - DigitizerFlowControl, [31](#)
  - DigitizerFlowControlStart, [31](#)
  - DigitizerFlowControl, [31](#)
  - Help, [31](#)
  - input\_flow\_cond, [33](#)
  - input\_flow\_mutex, [33](#)
  - mioTCP, [33](#)
  - mioinput, [33](#)
  - ParseCommand, [31](#)
  - reg\_matches, [33](#)
- DigitizerFlowControl.cpp, [146](#)
- DigitizerFlowControl.h, [159](#)
- DigitizerFlowControlStart
  - DigitizerFlowControl, [31](#)
- DigitizerObject, [34](#)
  - BoardInfo, [44](#)
  - DigitizerObject, [36](#)
  - DigitizerObjectClose, [36](#)
  - DigitizerObjectGetDRS4SamplingFrequency, [36](#)
  - DigitizerObjectGetInfo, [38](#)
  - DigitizerObjectOpen, [38](#)
  - DigitizerObjectReadRegister, [38](#)
  - DigitizerObjectReset, [38](#)
  - DigitizerObjectSWStartAcquisition, [42](#)
  - DigitizerObjectSWStopAcquisition, [43](#)
  - DigitizerObjectSetAcquisitionMode, [38](#)
  - DigitizerObjectSetChannelDCOffset, [39](#)
  - DigitizerObjectSetConfigStructureConfObject, [39](#)
  - DigitizerObjectSetConfigStructureInit, [39](#)
  - DigitizerObjectSetConfigStructureSetup, [39](#)
  - DigitizerObjectSetDRS4SamplingFrequency, [40](#)
  - DigitizerObjectSetExtTriggerInputMode, [40](#)
  - DigitizerObjectSetGroupDCOffset, [40](#)
  - DigitizerObjectSetGroupEnableMask, [40](#)
  - DigitizerObjectSetMaxNumEventsBLT, [42](#)
  - DigitizerObjectSetRecordLength, [42](#)
  - DigitizerObjectSetSWTriggerMode, [42](#)
  - DigitizerObjectWriteRegister, [43](#)
  - DigitizerObject, [36](#)
  - GetBoardInfo, [43](#)
  - GetFamilyCode, [43](#)
  - GetFormFactorCode, [43](#)
  - handle, [44](#)
  - internal\_config, [44](#)
  - logfile, [44](#)
  - PrintBoardInfo, [43](#)
  - ret, [44](#)
  - ret\_error, [44](#)
  - set\_board\_info, [44](#)
- DigitizerObject.cpp, [159](#)
- DigitizerObject.h, [165](#)
- DigitizerObjectClose
  - DigitizerObject, [36](#)
- DigitizerObjectGeneric, [45](#)
  - DigitizerObjectGeneric, [46](#)
  - DigitizerObjectGenericGetDRS4SamplingFrequency, [47](#)
  - DigitizerObjectGenericReadRegister, [47](#)
  - DigitizerObjectGenericSetAcquisitionMode, [47](#)
  - DigitizerObjectGenericSetAllInformations, [47](#)
  - DigitizerObjectGenericSetChannelDCOffset, [47](#)
  - DigitizerObjectGenericSetChannelSelfTrigger-Threshold, [47](#)
  - DigitizerObjectGenericSetDCOffset, [47](#)



- DigitizerObjectGenericSetDRS4SamplingFrequency, [DigitizerObjectGenericSetSelfTrigger](#)  
[48](#)
- DigitizerObjectGenericSetDecimationFactor, [48](#)
- DigitizerObjectGenericSetDesMode, [48](#)
- DigitizerObjectGenericSetEnableMask, [48](#)
- DigitizerObjectGenericSetExtTriggerInputMode, [48](#)
- DigitizerObjectGenericSetFastTriggerDigitizing, [49](#)
- DigitizerObjectGenericSetGroupDCOffset, [49](#)
- DigitizerObjectGenericSetIOLevel, [49](#)
- DigitizerObjectGenericSetMaxNumEventsBLT, [49](#)
- DigitizerObjectGenericSetPostTriggerSize, [49](#)
- DigitizerObjectGenericSetRecordLength, [50](#)
- DigitizerObjectGenericSetSWTriggerMode, [50](#)
- DigitizerObjectGenericSetSelfTrigger, [50](#)
- DigitizerObjectGenericSetTestPattern, [50](#)
- DigitizerObjectGenericSetTriggerPolarity, [50](#)
- DigitizerObjectGenericWriteRegister, [50](#)
- DigitizerObjectGeneric, [46](#)
- DigitizerObjectGeneric.cpp, [167](#)
- DigitizerObjectGeneric.h, [174](#)
- DigitizerObjectGenericGetDRS4SamplingFrequency  
DigitizerObjectGeneric, [47](#)
- DigitizerObjectGenericReadRegister  
DigitizerObjectGeneric, [47](#)
- DigitizerObjectGenericSetAcquisitionMode  
DigitizerObjectGeneric, [47](#)
- DigitizerObjectGenericSetAllInformations  
DigitizerObjectGeneric, [47](#)
- DigitizerObjectGenericSetChannelDCOffset  
DigitizerObjectGeneric, [47](#)
- DigitizerObjectGenericSetChannelSelfTriggerThreshold  
DigitizerObjectGeneric, [47](#)
- DigitizerObjectGenericSetDCOffset  
DigitizerObjectGeneric, [47](#)
- DigitizerObjectGenericSetDRS4SamplingFrequency  
DigitizerObjectGeneric, [48](#)
- DigitizerObjectGenericSetDecimationFactor  
DigitizerObjectGeneric, [48](#)
- DigitizerObjectGenericSetDesMode  
DigitizerObjectGeneric, [48](#)
- DigitizerObjectGenericSetEnableMask  
DigitizerObjectGeneric, [48](#)
- DigitizerObjectGenericSetExtTriggerInputMode  
DigitizerObjectGeneric, [48](#)
- DigitizerObjectGenericSetFastTriggerDigitizing  
DigitizerObjectGeneric, [49](#)
- DigitizerObjectGenericSetGroupDCOffset  
DigitizerObjectGeneric, [49](#)
- DigitizerObjectGenericSetIOLevel  
DigitizerObjectGeneric, [49](#)
- DigitizerObjectGenericSetMaxNumEventsBLT  
DigitizerObjectGeneric, [49](#)
- DigitizerObjectGenericSetPostTriggerSize  
DigitizerObjectGeneric, [49](#)
- DigitizerObjectGenericSetRecordLength  
DigitizerObjectGeneric, [50](#)
- DigitizerObjectGenericSetSWTriggerMode  
DigitizerObjectGeneric, [50](#)
- DigitizerObjectGenericSetSelfTrigger  
DigitizerObjectGeneric, [50](#)
- DigitizerObjectGenericSetTestPattern  
DigitizerObjectGeneric, [50](#)
- DigitizerObjectGenericSetTriggerPolarity  
DigitizerObjectGeneric, [50](#)
- DigitizerObjectGenericWriteRegister  
DigitizerObjectGeneric, [50](#)
- DigitizerObjectGetDRS4SamplingFrequency  
DigitizerObject, [36](#)
- DigitizerObjectGetInfo  
DigitizerObject, [38](#)
- DigitizerObjectOpen  
DigitizerObject, [38](#)
- DigitizerObjectReadRegister  
DigitizerObject, [38](#)
- DigitizerObjectReset  
DigitizerObject, [38](#)
- DigitizerObjectSWStartAcquisition  
DigitizerObject, [42](#)
- DigitizerObjectSWStopAcquisition  
DigitizerObject, [43](#)
- DigitizerObjectSetAcquisitionMode  
DigitizerObject, [38](#)
- DigitizerObjectSetChannelDCOffset  
DigitizerObject, [39](#)
- DigitizerObjectSetConfigStructureConfObject  
DigitizerObject, [39](#)
- DigitizerObjectSetConfigStructureInit  
DigitizerObject, [39](#)
- DigitizerObjectSetConfigStructureSetup  
DigitizerObject, [39](#)
- DigitizerObjectSetDRS4SamplingFrequency  
DigitizerObject, [40](#)
- DigitizerObjectSetExtTriggerInputMode  
DigitizerObject, [40](#)
- DigitizerObjectSetGroupDCOffset  
DigitizerObject, [40](#)
- DigitizerObjectSetGroupEnableMask  
DigitizerObject, [40](#)
- DigitizerObjectSetMaxNumEventsBLT  
DigitizerObject, [42](#)
- DigitizerObjectSetRecordLength  
DigitizerObject, [42](#)
- DigitizerObjectSetSWTriggerMode  
DigitizerObject, [42](#)
- DigitizerObjectWriteRegister  
DigitizerObject, [43](#)
- DigitizerStateMachine, [51](#)
  - Acquisition\_Cond1, [57](#)
  - Acquisition\_Cond2, [57](#)
  - BlockedConsumerInput, [57](#)
  - BlockedProducerInput, [57](#)
  - circular\_buffer\_preprocessing, [57](#)
  - circular\_buffer\_raw\_data, [58](#)
  - circular\_buffer\_visualization, [58](#)
  - coda\_preprocessing, [58](#)
  - coda\_raw\_data, [58](#)

- coda\_visualization, 58
- Consumatore\_Dispatcher, 54
- consumatore\_thread, 58
- digitizer, 58
- DigitizerStateMachine, 53
- DigitizerStateMachineInit, 54
- DigitizerStateMachinePrintStatus, 54
- DigitizerStateMachineQuit, 54
- DigitizerStateMachineRawDataInit, 54
- DigitizerStateMachineSendSWTrigger, 55
- DigitizerStateMachineSetup, 55
- DigitizerStateMachineStartReading, 55
- DigitizerStateMachineStopReading, 55
- DigitizerStateStartPreprocessing, 55
- DigitizerStateStartRawDataWriting, 56
- DigitizerStateStartVisualization, 56
- DigitizerStateStopPreprocessing, 56
- DigitizerStateStopRawDataWriting, 56
- DigitizerStateStopVisualization, 56
- DigitizerStateMachine, 53
- go\_general, 59
- go\_preprocessing, 59
- go\_raw\_data, 59
- go\_visualization, 59
- imset, 59
- imstarted, 59
- num\_mex\_preprocessing, 59
- num\_mex\_raw\_data, 59
- num\_mex\_visualization, 59
- pre\_cond, 60
- Preprocessing, 56
- preprocessing\_thread, 60
- Produttore, 57
- produttore\_thread, 60
- raw\_cond, 60
- ReservedConsumerDispatcherInputArea, 60
- ReservedPreprocessingInputArea, 60
- ReservedVisualizationInputArea, 60
- testa\_preprocessing, 60
- testa\_raw\_data, 60
- testa\_visualization, 61
- vis\_cond, 61
- Visualization, 57
- visualization\_thread, 61
- DigitizerStateMachine.cpp, 175
- DigitizerStateMachine.h, 183
- DigitizerStateMachineInit
  - DigitizerStateMachine, 54
- DigitizerStateMachinePrintStatus
  - DigitizerStateMachine, 54
- DigitizerStateMachineQuit
  - DigitizerStateMachine, 54
- DigitizerStateMachineRawDataInit
  - DigitizerStateMachine, 54
- DigitizerStateMachineSendSWTrigger
  - DigitizerStateMachine, 55
- DigitizerStateMachineSetup
  - DigitizerStateMachine, 55
- DigitizerStateMachineStartReading
  - DigitizerStateMachine, 55
- DigitizerStateMachineStopReading
  - DigitizerStateMachine, 55
- DigitizerStateStartPreprocessing
  - DigitizerStateMachine, 55
- DigitizerStateStartRawDataWriting
  - DigitizerStateMachine, 56
- DigitizerStateStartVisualization
  - DigitizerStateMachine, 56
- DigitizerStateStopPreprocessing
  - DigitizerStateMachine, 56
- DigitizerStateStopRawDataWriting
  - DigitizerStateMachine, 56
- DigitizerStateStopVisualization
  - DigitizerStateMachine, 56
- EVENT\_HEADER\_SIZE
  - X742DecodeRoutines.h, 209
- enable\_fast\_trigger\_digitizing
  - ConfObject, 25
- enable\_input
  - ChannelObject, 14
  - ConfObject, 25
  - GroupObject, 64
- eventInfo
  - RawData, 77
- external\_trigger\_acquisition\_mode
  - ConfObject, 26
- fast\_trigger\_acquisition\_mode
  - ConfObject, 26
- FastInformation
  - AnalizzatoreUtils.c, 87
  - AnalizzatoreUtils.h, 99
- FastObject, 61
  - dc\_offset, 62
  - FastObject, 62
  - FastObject, 62
  - numFast, 62
  - PrintFast, 62
  - set, 62
  - trigger\_threshold, 62
- fasts
  - ConfObject, 26
- FetchInputMode
  - ApplicationSetup, 11
- file
  - RawData, 77
- FindIntegerValue
  - AnalizzatoreUtils.c, 87
  - AnalizzatoreUtils.h, 99
- FindPointer
  - AnalizzatoreUtils.c, 87
  - AnalizzatoreUtils.h, 100
  - ClientApplication.c, 109
- Finish
  - CommunicationObject, 16
  - Input, 66

- first\_parameter
  - TcpUser, 79
- GetAcquisitionMode
  - AnalizzatoreUtils.c, 88
  - AnalizzatoreUtils.h, 100
- GetBoardInfo
  - DigitizerObject, 43
- GetCommand
  - CommunicationObject, 17
- GetEventPtr
  - X742DecodeRoutines.c, 204
  - X742DecodeRoutines.h, 209
- GetFamilyCode
  - DigitizerObject, 43
- GetFormFactorCode
  - DigitizerObject, 43
- GetInput
  - Input, 66
- GetNumEvents
  - X742DecodeRoutines.c, 204
  - X742DecodeRoutines.h, 209
- GetOpenInformation
  - AnalizzatoreUtils.c, 88
  - AnalizzatoreUtils.h, 100
- GetWriteRegisterInformation
  - AnalizzatoreUtils.c, 88
  - AnalizzatoreUtils.h, 100
- gnuplot
  - ConfObject, 26
- go
  - ClientApplication.c, 111
  - CommunicationObject, 18
  - Input, 66
- go\_general
  - DigitizerStateMachine, 59
- go\_preprocessing
  - DigitizerStateMachine, 59
- go\_raw\_data
  - DigitizerStateMachine, 59
- go\_visualization
  - DigitizerStateMachine, 59
- group\_enable\_mask
  - ConfObject, 26
- group\_trg\_enable\_mask
  - GroupObject, 64
- GroupEnableMask
  - AnalizzatoreUtils.c, 88
  - AnalizzatoreUtils.h, 101
- GroupInformation
  - AnalizzatoreUtils.c, 89
  - AnalizzatoreUtils.h, 101
- GroupObject, 63
  - dc\_offset, 64
  - enable\_input, 64
  - group\_trg\_enable\_mask, 64
  - GroupObject, 64
  - GroupObject, 64
  - numGroup, 64
  - PrintGroup, 64
  - set, 64
  - trigger\_threshold, 64
- groups
  - ConfObject, 26
- handle
  - DigitizerObject, 44
  - RawData, 77
- header\_yes\_no
  - ConfObject, 26
- Help
  - ClientApplication.c, 109
  - DigitizerFlowControl, 31
- i
  - ClientApplication.c, 111
- INIT
  - DefineCommands.h, 134
- imset
  - ApplicationSetup, 13
  - DigitizerStateMachine, 59
  - RawData, 78
- imstarted
  - DigitizerStateMachine, 59
- Input, 65
  - Finish, 66
  - GetInput, 66
  - go, 66
  - Input, 65
  - input\_buffer, 66
  - mutex1, 66
  - num\_mex, 66
  - Producer, 66
  - producer\_thread, 67
- Input.cpp, 185
- Input.h, 186
- input\_buffer
  - Input, 66
- input\_flow\_cond
  - DigitizerFlowControl, 33
- input\_flow\_mutex
  - DigitizerFlowControl, 33
- input\_mode
  - ApplicationSetup, 13
- inputline
  - ClientApplication.c, 111
- Instance
  - ApplicationSetup, 11
  - LogFile, 68
  - OutputModule, 70
- internal\_config
  - DigitizerObject, 44
- LinkNumber
  - ConfObject, 26
- LinkType
  - ConfObject, 27
- log\_file\_path

- LogFile, 69
- LogFile, 67
  - Instance, 68
  - log\_file\_path, 69
  - LogFileMessageOff, 68
  - LogFileMessageOn, 68
  - LogFileRead, 68
  - LogFileSet, 68
  - LogFileWrite, 68
  - LogFileWriteString, 69
- LogFile.cpp, 187
- LogFile.h, 189
- LogFileMessageOff
  - LogFile, 68
- LogFileMessageOn
  - LogFile, 68
- LogFileRead
  - LogFile, 68
- LogFileSet
  - LogFile, 68
- LogFileWrite
  - LogFile, 68
- LogFileWriteString
  - LogFile, 69
- logfile
  - DigitizerObject, 44
- MAXCHANNELOBJECT
  - DefineGeneral.h, 137
- MAXCOMMAND
  - DefineGeneral.h, 137
- MAXFASTOBJECT
  - DefineGeneral.h, 137
- MAXGROUPOBJECT
  - DefineGeneral.h, 137
- MORE
  - DefineCommands.h, 134
- Main
  - CommunicationObject, 17
- main
  - ClientApplication.c, 109
  - Main.c, 191
- Main.c, 190
  - main, 191
  - ONLY\_TCP, 191
- main\_thread
  - CommunicationObject, 18
- Mask\_register
  - ConfObject, 27
- max\_num\_events\_BLT
  - ConfObject, 27
- mioTCP
  - DigitizerFlowControl, 33
- mioinput
  - DigitizerFlowControl, 33
- Mode
  - RawData, 78
- mutex1
  - Input, 66

- n
  - ClientApplication.c, 111
- nim\_ttl
  - ConfObject, 27
- no\_new\_line
  - ClientApplication.c, 111
- num\_mex
  - CommunicationObject, 19
  - Input, 66
- num\_mex\_preprocessing
  - DigitizerStateMachine, 59
- num\_mex\_raw\_data
  - DigitizerStateMachine, 59
- num\_mex\_visualization
  - DigitizerStateMachine, 59
- numChannel
  - ChannelObject, 15
- numFast
  - FastObject, 62
- numGroup
  - GroupObject, 64
- ONLY\_TCP
  - Main.c, 191
- ONLYTCP
  - DefineGeneral.h, 137
- ONLYUSER
  - DefineGeneral.h, 137
- operator=
  - RawData, 74
- Output
  - OutputModule, 70
- output\_file\_format
  - ConfObject, 27
- output\_module\_application\_setup
  - OutputModule, 72
- OutputFileFormat
  - AnalizzatoreUtils.c, 89
  - AnalizzatoreUtils.h, 101
- OutputFlex
  - OutputModule, 71
- OutputModule, 69
  - buffer, 72
  - Instance, 70
  - Output, 70
  - output\_module\_application\_setup, 72
  - OutputFlex, 71
  - OutputModuleSockidOff, 71
  - OutputModuleSockidOn, 71
  - OutputModuleStdoutOff, 71
  - OutputModuleStdoutOn, 71
  - sockid\_array, 72
  - StdOutInsert, 71
  - StdOutInsertLex, 71
  - StdOutPrint, 71
  - TcpUserArrayDelete, 72
  - TcpUserArrayInsert, 72
  - TcpUserArraySendStdOut, 72
- OutputModule.cpp, 192

- OutputModule.h, [194](#)
- OutputModuleSockidOff
  - OutputModule, [71](#)
- OutputModuleSockidOn
  - OutputModule, [71](#)
- OutputModuleStdoutOff
  - OutputModule, [71](#)
- OutputModuleStdoutOn
  - OutputModule, [71](#)
- OutputNIMTTL
  - AnalizzatoreUtils.c, [89](#)
  - AnalizzatoreUtils.h, [103](#)
- OutputRisingFalling
  - AnalizzatoreUtils.c, [90](#)
  - AnalizzatoreUtils.h, [103](#)
- PREPROCESSINGQUEUE
  - DefineGeneral.h, [137](#)
- PRESTART
  - DefineCommands.h, [134](#)
- PRESTOP
  - DefineCommands.h, [134](#)
- PRINT
  - DefineCommands.h, [134](#)
- PRINTFILES
  - DefineCommands.h, [134](#)
- ParseCommand
  - DigitizerFlowControl, [31](#)
- post\_trigger
  - ConfObject, [27](#)
- pre\_cond
  - DigitizerStateMachine, [60](#)
- Preprocessing
  - DigitizerStateMachine, [56](#)
- preprocessing\_thread
  - DigitizerStateMachine, [60](#)
- PrintAll
  - ConfObject, [24](#)
- PrintAllHuman
  - ConfObject, [24](#)
- PrintBoardInfo
  - DigitizerObject, [43](#)
- PrintChannel
  - ChannelObject, [14](#)
- PrintFast
  - FastObject, [62](#)
- PrintGroup
  - GroupObject, [64](#)
- Producer
  - Input, [66](#)
- producer\_thread
  - Input, [67](#)
- Produttore
  - DigitizerStateMachine, [57](#)
- produttore\_thread
  - DigitizerStateMachine, [60](#)
- QUIT
  - DefineCommands.h, [134](#)
- RAWDATAQUEUE
  - DefineGeneral.h, [137](#)
- RAWSTART
  - DefineCommands.h, [134](#)
- RAWSTOP
  - DefineCommands.h, [135](#)
- READREGISTER
  - DefineCommands.h, [135](#)
- raw\_cond
  - DigitizerStateMachine, [60](#)
- RawData, [72](#)
  - bsize, [77](#)
  - buffer, [77](#)
  - digitizer, [77](#)
  - eventInfo, [77](#)
  - file, [77](#)
  - handle, [77](#)
  - imset, [78](#)
  - Mode, [78](#)
  - operator=, [74](#)
  - RawData, [74](#)
  - RawDataDecode, [74](#)
  - RawDataDel, [74](#)
  - RawDataPlot, [74](#)
  - RawDataRead, [75](#)
  - RawDataSet, [75](#)
  - RawDataWriteDecodeEventOnPlotFile, [75](#)
  - RawDataWriteOnFile, [75, 77](#)
  - RawData, [74](#)
  - ret, [78](#)
  - ret\_error, [78](#)
  - size, [78](#)
- RawData.cpp, [195](#)
- RawData.h, [201](#)
- RawDataDecode
  - RawData, [74](#)
- RawDataDel
  - RawData, [74](#)
- RawDataPlot
  - RawData, [74](#)
- RawDataRead
  - RawData, [75](#)
- RawDataSet
  - RawData, [75](#)
- RawDataWriteDecodeEventOnPlotFile
  - RawData, [75](#)
- RawDataWriteOnFile
  - RawData, [75, 77](#)
- receiveline
  - ClientApplication.c, [111](#)
- record\_length
  - ConfObject, [27](#)
- reg\_matches
  - AnalizzatoreUtils.c, [90](#)
  - ClientApplication.c, [111](#)
  - DigitizerFlowControl, [33](#)
- register\_address
  - TcpUser, [79](#)

- register\_data
  - TcpUser, 79
- ReservedConsumerDispatcherInputArea
  - DigitizerStateMachine, 60
- ReservedKeyBoardInputArea
  - CommunicationObject, 19
- ReservedPreprocessingInputArea
  - DigitizerStateMachine, 60
- ReservedVisualizationInputArea
  - DigitizerStateMachine, 60
- ret
  - DigitizerObject, 44
  - RawData, 78
- ret\_error
  - DigitizerObject, 44
  - RawData, 78
- ricevitore\_function
  - ClientApplication.c, 111
- rising\_falling
  - ConfObject, 27
- SEND
  - DefineCommands.h, 135
- SETUP
  - DefineCommands.h, 135
- STANDARDBUFFERLIMIT
  - DefineGeneral.h, 137
- START
  - DefineCommands.h, 135
- STATUS
  - DefineCommands.h, 135
- STOP
  - DefineCommands.h, 135
- second\_parameter
  - TcpUser, 80
- self\_trigger\_enable\_mask
  - ConfObject, 28
- self\_trigger\_enable\_mask\_mode
  - ConfObject, 28
- sendline
  - ClientApplication.c, 111
- set
  - ChannelObject, 15
  - FastObject, 62
  - GroupObject, 64
- set\_board\_info
  - DigitizerObject, 44
- size
  - RawData, 78
- sockfd
  - ClientApplication.c, 112
- sockid\_array
  - OutputModule, 72
- StdOutInsert
  - OutputModule, 71
- StdOutInsertLex
  - OutputModule, 71
- StdOutPrint
  - OutputModule, 71
- TcpUser, 78
  - command\_sent\_by\_user, 79
  - first\_parameter, 79
  - register\_address, 79
  - register\_data, 79
  - second\_parameter, 80
  - TcpUser, 79
  - TcpUser, 79
  - user\_sockid, 80
- TcpUser.cpp, 202
- TcpUser.h, 203
- TcpUserArrayDelete
  - OutputModule, 72
- TcpUserArrayInsert
  - OutputModule, 72
- TcpUserArraySendStdOut
  - OutputModule, 72
- test\_pattern
  - ConfObject, 28
- testa
  - CommunicationObject, 19
- testa\_preprocessing
  - DigitizerStateMachine, 60
- testa\_raw\_data
  - DigitizerStateMachine, 60
- testa\_visualization
  - DigitizerStateMachine, 61
- trigger\_threshold
  - ChannelObject, 15
  - ConfObject, 28
  - FastObject, 62
  - GroupObject, 64
- use\_interrupt
  - ConfObject, 28
- user\_sockid
  - TcpUser, 80
- VISTART
  - DefineCommands.h, 135
- VISTOP
  - DefineCommands.h, 135
- VISUALIZATIONQUEUE
  - DefineGeneral.h, 137
- VMEBaseAddress
  - ConfObject, 28
- vis\_cond
  - DigitizerStateMachine, 61
- Visualization
  - DigitizerStateMachine, 57
- visualization\_thread
  - DigitizerStateMachine, 61
- WRITEREGISTER
  - DefineCommands.h, 135
- Worker
  - CommunicationObject, 17
- worker\_thread
  - CommunicationObject, 19

X742\_DecodeEvent  
    X742DecodeRoutines.c, [204](#)  
    X742DecodeRoutines.h, [209](#)  
X742\_FIXED\_SIZE  
    X742DecodeRoutines.h, [209](#)  
X742\_MAX\_GROUPS  
    X742DecodeRoutines.h, [209](#)  
X742DecodeRoutines.c, [203](#)  
    GetEventPtr, [204](#)  
    GetNumEvents, [204](#)  
    X742\_DecodeEvent, [204](#)  
X742DecodeRoutines.h, [209](#)  
    EVENT\_HEADER\_SIZE, [209](#)  
    GetEventPtr, [209](#)  
    GetNumEvents, [209](#)  
    X742\_DecodeEvent, [209](#)  
    X742\_FIXED\_SIZE, [209](#)  
    X742\_MAX\_GROUPS, [209](#)  
  
YesNoAnswer  
    AnalizzatoreUtils.c, [90](#)  
    AnalizzatoreUtils.h, [103](#)