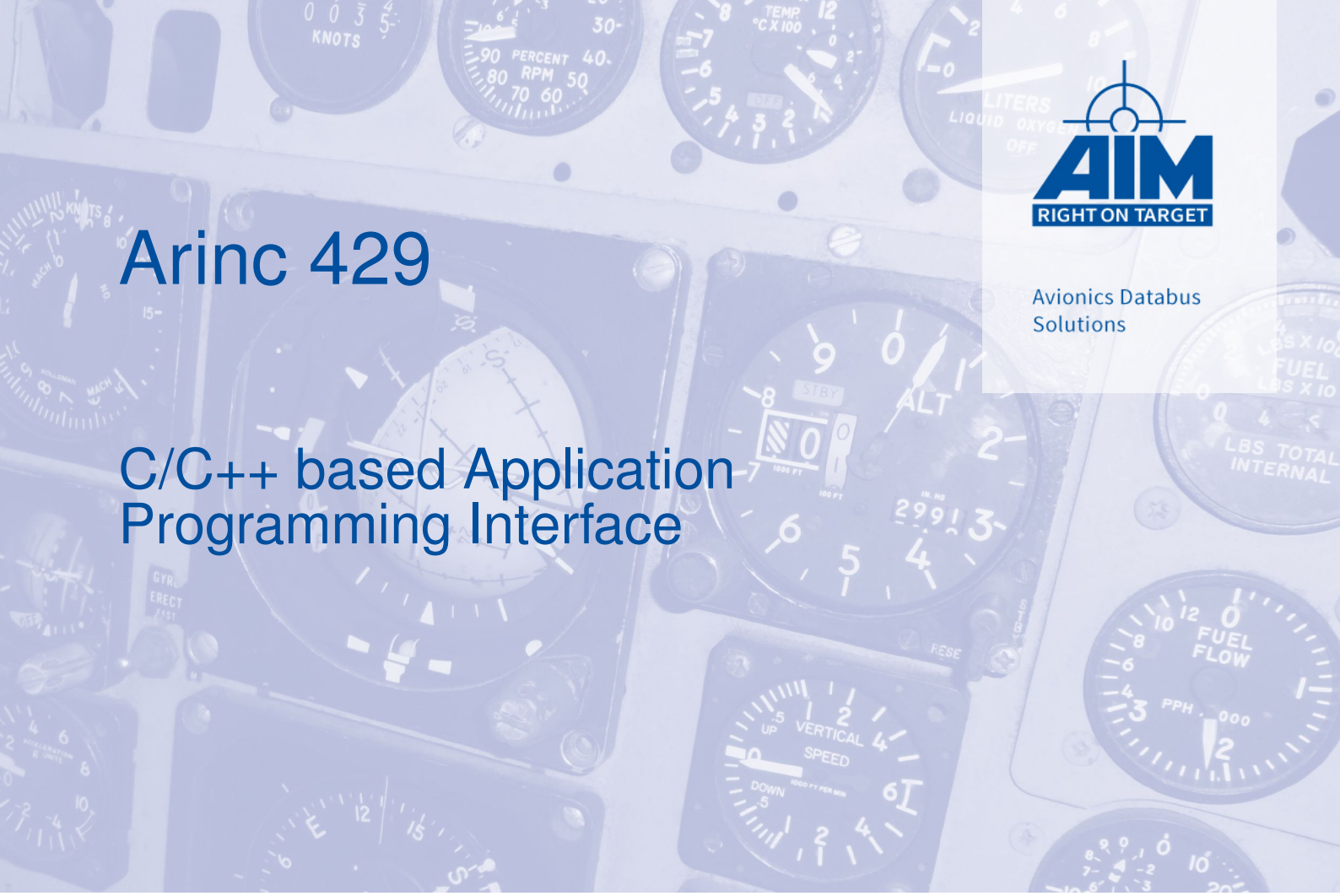

AIM
RIGHT ON TARGET

Avionics Databus Solutions

Arinc 429

C/C++ based Application Programming Interface

Reference Manual

Version 9.8.0
June, 2022

www.aim-online.com

Arinc 429

C/C++ based Application Programming Interface

Reference Manual

Version 9.8.0
June, 2022

AIM NO. 60-12900-36-9.8.0

AIM – Gesellschaft für angewandte Informatik und Mikroelektronik mbH

AIM GmbH
Sasbacher Str. 2
D-79111 Freiburg / Germany
Phone +49 (0)761 4 52 29-0
Fax +49 (0)761 4 52 29-33
sales@aim-online.com

AIM GmbH – Munich Sales Office
Terofalstr. 23a
D-80689 München / Germany
Phone +49 (0)89 70 92 92-92
Fax +49 (0)89 70 92 92-94
salesgermany@aim-online.com

AIM UK Office
Cressex Enterprise Centre, Lincoln Rd.
High Wycombe, Bucks. HP12 3RB / UK
Phone +44 (0)1494-446844
Fax +44 (0)1494-449324
salesuk@aim-online.com

AIM USA LLC
Seven Neshaminy Interplex
Suite 211 Trevose, PA 19053
Phone 267-982-2600
Fax 215-645-1580
salesusa@aim-online.com

TABLE OF CONTENTS

1 Module Index **1**
1.1 Modules ... 2

2 Data Structure Index **3**
2.1 Data Structures ... 4

3 Module Documentation **7**
3.1 Board Related Functions ... 8
3.1.1 Detailed Description ... 15
3.1.2 Macro Definition Documentation ... 15
3.1.3 Typedef Documentation ... 21
3.1.4 Enumeration Type Documentation ... 23
3.1.5 Function Documentation ... 33
3.2 Raw Board Memory Access ... 44
3.2.1 Detailed Description ... 45
3.2.2 Typedef Documentation ... 45
3.2.3 Enumeration Type Documentation ... 45
3.2.4 Function Documentation ... 46
3.3 Channel Handling ... 51
3.3.1 Detailed Description ... 53
3.3.2 Macro Definition Documentation ... 53
3.3.3 Typedef Documentation ... 61
3.3.4 Enumeration Type Documentation ... 63
3.3.5 Function Documentation ... 64
3.4 Discrete Handling ... 68
3.4.1 Detailed Description ... 68
3.4.2 Typedef Documentation ... 68
3.4.3 Function Documentation ... 69
3.5 Library Administration Functions ... 73
3.5.1 Detailed Description ... 73
3.5.2 Typedef Documentation ... 73
3.5.3 Enumeration Type Documentation ... 74
3.5.4 Function Documentation ... 74
3.6 Data Replay ... 79
3.6.1 Detailed Description ... 79
3.6.2 Typedef Documentation ... 79
3.6.3 Function Documentation ... 80
3.7 Data Monitoring ... 83
3.7.1 Detailed Description ... 86

3.7.2 Macro Definition Documentation 86
3.7.3 Typedef Documentation 87
3.7.4 Enumeration Type Documentation 89
3.7.5 Function Documentation 93
3.8 Data Receiver 103
3.8.1 Detailed Description 104
3.8.2 Typedef Documentation 104
3.8.3 Function Documentation 105
3.9 Data Transmitting 115
3.9.1 Detailed Description 117
3.9.2 Typedef Documentation 118
3.9.3 Enumeration Type Documentation 120
3.9.4 Function Documentation 122
3.10 FIFO Based Transmitting 139
3.10.1 Detailed Description 140
3.10.2 Typedef Documentation 140
3.10.3 Function Documentation 140
3.11 Version Handling 148
3.11.1 Detailed Description 148
3.11.2 Typedef Documentation 148
3.11.3 Function Documentation 149
3.12 VxWorks Related Functions 152
3.12.1 Detailed Description 152
3.12.2 Function Documentation 152

4 Data Structure Documentation **155**
4.1 _FMListDetails Struct Reference 156
4.1.1 Detailed Description 156
4.1.2 Field Documentation 156
4.2 api429_board_info Struct Reference 157
4.2.1 Detailed Description 157
4.2.2 Field Documentation 157
4.3 api429_brw Union Reference 160
4.3.1 Detailed Description 161
4.3.2 Field Documentation 161
4.4 api429_channel_info Struct Reference 162
4.4.1 Detailed Description 162
4.4.2 Field Documentation 162
4.5 api429_discr_info Struct Reference 164
4.5.1 Detailed Description 164
4.5.2 Field Documentation 164
4.6 api429_discrete_pps_config Struct Reference 166
4.6.1 Detailed Description 166
4.6.2 Field Documentation 166
4.7 api429_intr_loglist_entry Struct Reference 167
4.7.1 Detailed Description 167
4.7.2 Field Documentation 167
4.8 api429_loglist_entry Union Reference 170
4.8.1 Detailed Description 170

4.9 api429_lxfer_dyntag Struct Reference 171
4.9.1 Detailed Description 171
4.9.2 Field Documentation 171
4.10 api429_mframe_in Struct Reference 173
4.10.1 Detailed Description 173
4.10.2 Field Documentation 173
4.11 api429_mframe_out Struct Reference 175
4.11.1 Detailed Description 175
4.11.2 Field Documentation 175
4.12 api429_rcv_pb_cmd Struct Reference 176
4.12.1 Detailed Description 176
4.12.2 Field Documentation 176
4.13 api429_rcv_stack_entry Struct Reference 179
4.13.1 Detailed Description 179
4.13.2 Field Documentation 179
4.14 api429_rcv_stack_entry_ex Struct Reference 181
4.14.1 Detailed Description 181
4.14.2 Field Documentation 181
4.15 api429_remote_server Struct Reference 183
4.15.1 Detailed Description 183
4.15.2 Field Documentation 183
4.16 api429_replay_status Struct Reference 185
4.16.1 Detailed Description 185
4.16.2 Field Documentation 185
4.17 api429_rm_activity_trigger_def Struct Reference 188
4.17.1 Detailed Description 188
4.17.2 Field Documentation 188
4.18 api429_rm_function_block Struct Reference 190
4.18.1 Detailed Description 190
4.18.2 Field Documentation 190
4.19 api429_rm_label_config Struct Reference 195
4.19.1 Detailed Description 195
4.19.2 Field Documentation 195
4.20 api429_rm_setup Struct Reference 197
4.20.1 Detailed Description 197
4.20.2 Field Documentation 197
4.21 api429_rm_trigger_setup Struct Reference 199
4.21.1 Detailed Description 199
4.21.2 Field Documentation 199
4.22 api429_rx_activity Struct Reference 201
4.22.1 Detailed Description 201
4.22.2 Field Documentation 201
4.23 api429_rx_buf_ctl Struct Reference 202
4.23.1 Detailed Description 202
4.23.2 Field Documentation 202
4.24 api429_rx_buf_entry Struct Reference 204
4.24.1 Detailed Description 204
4.24.2 Field Documentation 204
4.25 api429_rx_frame_response_setup Struct Reference 205

4.25.1 Detailed Description 205
4.25.2 Field Documentation 205
4.26 api429_rx_label_setup Struct Reference 207
4.26.1 Detailed Description 207
4.26.2 Field Documentation 207
4.27 api429_time Struct Reference 209
4.27.1 Detailed Description 209
4.27.2 Field Documentation 209
4.28 api429_tm_tag Union Reference 211
4.28.1 Detailed Description 211
4.29 api429_trg_ctl_cmd Struct Reference 212
4.29.1 Detailed Description 212
4.29.2 Field Documentation 212
4.30 api429_tx_fifo_entry Struct Reference 214
4.30.1 Detailed Description 214
4.30.2 Field Documentation 214
4.31 api429_tx_fifo_setup Struct Reference 215
4.31.1 Detailed Description 215
4.31.2 Field Documentation 215
4.32 api429_tx_fifo_status Struct Reference 217
4.32.1 Detailed Description 217
4.32.2 Field Documentation 217
4.33 api429_version Struct Reference 218
4.33.1 Detailed Description 218
4.33.2 Field Documentation 218
4.34 api429_version_info Struct Reference 220
4.34.1 Detailed Description 220
4.34.2 Field Documentation 220
4.35 api429_xfer Struct Reference 224
4.35.1 Detailed Description 224
4.35.2 Field Documentation 224
4.36 api429_xfer_data Struct Reference 227
4.36.1 Detailed Description 227
4.36.2 Field Documentation 227
4.37 api429_xfer_data_read_input Struct Reference 229
4.37.1 Detailed Description 229
4.37.2 Field Documentation 229
4.38 api429_xfer_info Struct Reference 231
4.38.1 Detailed Description 231
4.38.2 Field Documentation 231
4.39 api429_xfer_out Struct Reference 233
4.39.1 Detailed Description 233
4.39.2 Field Documentation 233
4.40 cmplx128 Struct Reference 235
4.40.1 Detailed Description 235
4.41 cmplx64 Struct Reference 236
4.41.1 Detailed Description 236
4.42 cmplxExt Struct Reference 237
4.42.1 Detailed Description 237

4.43 CPStr Struct Reference 238
4.43.1 Detailed Description 238
4.44 DateRec Struct Reference 239
4.44.1 Detailed Description 239
4.45 FInfoRec Struct Reference 240
4.45.1 Detailed Description 240
4.45.2 Field Documentation 240
4.46 FInfoRec64 Struct Reference 244
4.46.1 Detailed Description 244
4.46.2 Field Documentation 244
4.47 LStr Struct Reference 248
4.47.1 Detailed Description 248
4.48 MemStatRec Struct Reference 249
4.48.1 Detailed Description 249
4.49 VInfoRec Struct Reference 250
4.49.1 Detailed Description 250
4.49.2 Field Documentation 250

Index I

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Board Related Functions 8
Raw Board Memory Access 44
Channel Handling 51
Discrete Handling 68
Library Administration Functions 73
Data Replay 79
Data Monitoring 83
Data Receiver 103
Data Transmitting 115
FIFO Based Transmitting 139
Version Handling 148
VxWorks Related Functions 152

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

_FMListDetails
Data used with FListDir to describe the files listed in a directory 156
api429_board_info 157
api429_brw 160
api429_channel_info 162
api429_discr_info 164
api429_discrete_pps_config 166
api429_intr_loglist_entry 167
api429_loglist_entry 170
api429_lxfer_dyntag 171
api429_mframe_in 173
api429_mframe_out 175
api429_rcv_pb_cmd 176
api429_rcv_stack_entry 179
api429_rcv_stack_entry_ex 181
api429_remote_server 183
api429_replay_status 185
api429_rm_activity_trigger_def 188
api429_rm_function_block 190
api429_rm_label_config 195
api429_rm_setup 197
api429_rm_trigger_setup 199
api429_rx_activity 201
api429_rx_buf_ctl 202
api429_rx_buf_entry 204
api429_rx_frame_response_setup 205
api429_rx_label_setup 207
api429_time 209
api429_tm_tag 211
api429_trg_ctl_cmd 212
api429_tx_fifo_entry 214
api429_tx_fifo_setup 215
api429_tx_fifo_status 217
api429_version 218
api429_version_info 220
api429_xfer 224
api429_xfer_data 227
api429_xfer_data_read_input 229
api429_xfer_info 231
api429_xfer_out 233
cmplx128
Complex double-precision floating point number 235
cmplx64
Complex single-precision floating point number 236

cmplxExt
Complex extended-precision floating point number 237
CPStr
Concatenated Pascal string types 238
DateRec
Date/time record 239
FInfoRec
Descriptive information about a file 240
FInfoRec64
Descriptive information about a file, with large file support 244
LStr
Long Pascal-style string types 248
MemStatRec
Describes memory statistics 249
VInfoRec
Data structure describing a disk volume 250

Chapter 3

Module Documentation

3.1 Board Related Functions

Data Structures

- struct api429_time
- struct api429_board_info
- struct api429_remote_server

Macros

- #define API429_MAX_BOARDNAME_LEN 32
- #define API429_MAX_URL_LEN 128
- #define API429_MAX_OS_INFO_LEN 128
- #define API429_BOARD_CAP_PXI_TRG_FLAG (1 << 0)
- #define API429_BOARD_TYPE_EMBEDDED_FLAG (1 << 4)
- #define API429_BOARD_TX_INHIBIT_FLAG (1 << 12)
- #define MAX_API429_BIU 2
- #define MAX_API429_BIU_CHN 16
- #define API429_MAX_CHANNELS (MAX_API429_BIU*MAX_API429_BIU_CHN)
- #define API429_BOARD_HAS_PXI_TRIGGER(board_info) (board_info)->capability_flags & API429_BOARD_CAP_PXI_TRG_FLAG ? AiTrue : AiFalse
- #define API429_BOARD_TYPE_IS_EMBEDDED(board_info) (board_info)->capability_flags & API429_BOARD_TYPE_EMBEDDED_FLAG ? AiTrue : AiFalse
- #define API429_BOARD_TX_INHIBIT_IS_ACTIVATED(board_info) (board_info)->capability_flags & API429_BOARD_TX_INHIBIT_FLAG ? AiTrue : AiFalse
- #define API429_LS_GRADE_DEFAULT API429_LS_GRADE_12_50
- #define API429_MAX_BOARDNAME_LEN 32
- #define API429_MAX_URL_LEN 128
- #define API429_MAX_OS_INFO_LEN 128
- #define API429_BOARD_CAP_PXI_TRG_FLAG (1 << 0)
- #define API429_BOARD_TYPE_EMBEDDED_FLAG (1 << 4)
- #define API429_BOARD_TX_INHIBIT_FLAG (1 << 12)
- #define MAX_API429_BIU 2
- #define MAX_API429_BIU_CHN 16
- #define API429_MAX_CHANNELS (MAX_API429_BIU*MAX_API429_BIU_CHN)
- #define API429_BOARD_HAS_PXI_TRIGGER(board_info) (board_info)->capability_flags & API429_BOARD_CAP_PXI_TRG_FLAG ? AiTrue : AiFalse
- #define API429_BOARD_TYPE_IS_EMBEDDED(board_info) (board_info)->capability_flags & API429_BOARD_TYPE_EMBEDDED_FLAG ? AiTrue : AiFalse
- #define API429_BOARD_TX_INHIBIT_IS_ACTIVATED(board_info) (board_info)->capability_flags & API429_BOARD_TX_INHIBIT_FLAG ? AiTrue : AiFalse
- #define API429_LS_GRADE_DEFAULT API429_LS_GRADE_12_50

Typedefs

- typedef struct api429_time TY_API429_TIME
- typedef struct api429_board_info TY_API429_BOARD_INFO
- typedef enum api429_low_speed_grade TY_E_API429_LOW_SPEED_GRADE
- typedef enum api429_speed_modifier TY_E_API429_SPEED_MODIFIER
- typedef enum api429_irig_source TY_E_API429_IRIG_SOURCE
- typedef enum api429_channel_coupling TY_E_API429_CHANNEL_COUPLING
- typedef struct api429_time **TY_API429_TIME**
- typedef struct api429_board_info **TY_API429_BOARD_INFO**
- typedef enum api429_low_speed_grade **TY_E_API429_LOW_SPEED_GRADE**
- typedef enum api429_speed_modifier **TY_E_API429_SPEED_MODIFIER**
- typedef enum api429_irig_source **TY_E_API429_IRIG_SOURCE**
- typedef enum api429_channel_coupling **TY_E_API429_CHANNEL_COUPLING**

Enumerations

- enum api429_board_id {
  API429_BOARD_1 = 0,
  API429_BOARD_2,
  API429_BOARD_3,
  API429_BOARD_4,
  API429_BOARD_5,
  API429_BOARD_6,
  API429_BOARD_7,
  API429_BOARD_8,
  API429_BOARD_9,
  API429_BOARD_10,
  API429_BOARD_11,
  API429_BOARD_12,
  API429_BOARD_13,
  API429_BOARD_14,
  API429_BOARD_15,
  API429_BOARD_16,
  API429_BOARD_17,
  API429_BOARD_18,
  API429_BOARD_19,
  API429_BOARD_20,
  API429_BOARD_21,
  API429_BOARD_22,
  API429_BOARD_23,
  API429_BOARD_24,
  API429_BOARD_25,
  API429_BOARD_26,
  API429_BOARD_27,
  API429_BOARD_28,
  API429_BOARD_29,

API429_BOARD_30,
API429_BOARD_31,
API429_BOARD_32,
API429_BOARD_1 = 0,
API429_BOARD_2,
API429_BOARD_3,
API429_BOARD_4,
API429_BOARD_5,
API429_BOARD_6,
API429_BOARD_7,
API429_BOARD_8,
API429_BOARD_9,
API429_BOARD_10,
API429_BOARD_11,
API429_BOARD_12,
API429_BOARD_13,
API429_BOARD_14,
API429_BOARD_15,
API429_BOARD_16,
API429_BOARD_17,
API429_BOARD_18,
API429_BOARD_19,
API429_BOARD_20,
API429_BOARD_21,
API429_BOARD_22,
API429_BOARD_23,
API429_BOARD_24,
API429_BOARD_25,
API429_BOARD_26,
API429_BOARD_27,
API429_BOARD_28,
API429_BOARD_29,
API429_BOARD_30,
API429_BOARD_31,
API429_BOARD_32 }

- enum api429_pxi_trigger_line {
API429_TRG_PXI0 = 0,
API429_TRG_PXI1,
API429_TRG_PXI2,
API429_TRG_PXI3,
API429_TRG_PXI4,
API429_TRG_PXI5,
API429_TRG_PXI6,
API429_TRG_PXI7,
API429_TRG_PBI429,
API429_TRG_PXI0 = 0,
API429_TRG_PXI1,
API429_TRG_PXI2,
API429_TRG_PXI3,

API429_TRG_PXI4,
API429_TRG_PXI5,
API429_TRG_PXI6,
API429_TRG_PXI7,
API429_TRG_PBI429 }

- enum api429_pxi_mode {
  API429_PXI_SET_TRG = 0,
  API429_PXI_CLR_TRG,
  API429_PXI_SET_TTSRC_BACKPLANE,
  API429_PXI_SET_TTSRC_FRONT,
  API429_PXI_SET_TRG = 0,
  API429_PXI_CLR_TRG,
  API429_PXI_SET_TTSRC_BACKPLANE,
  API429_PXI_SET_TTSRC_FRONT }
- enum api429_low_speed_grade {
  API429_LS_GRADE_12_50 = 0,
  API429_LS_GRADE_13_33,
  API429_LS_GRADE_12_50 = 0,
  API429_LS_GRADE_13_33 }
- enum api429_speed_modifier {
  API429_SPEED_MOD_NORMAL = 0,
  API429_SPEED_MOD_INC_9_1 = 1,
  API429_SPEED_MOD_DEC_7_7 = 2,
  API429_SPEED_MOD_INC_20 = 3,
  API429_SPEED_MOD_NORMAL = 0,
  API429_SPEED_MOD_INC_9_1 = 1,
  API429_SPEED_MOD_DEC_7_7 = 2,
  API429_SPEED_MOD_INC_20 = 3 }
- enum api429_irig_source {
  API429_IRIG_ONBOARD = 0,
  API429_IRIG_EXTERN = 1,
  API429_IRIG_ONBOARD = 0,
  API429_IRIG_EXTERN = 1 }
- enum api429_channel_coupling {
  API429_CHANNEL_COUPLING_EXT = 0,
  API429_CHANNEL_COUPLING_DUAL,
  API429_CHANNEL_COUPLING_INT,
  API429_CHANNEL_COUPLING_EXT = 0,
  API429_CHANNEL_COUPLING_DUAL,
  API429_CHANNEL_COUPLING_INT }
- enum api429_board_id {
  API429_BOARD_1 = 0,
  API429_BOARD_2,
  API429_BOARD_3,
  API429_BOARD_4,
  API429_BOARD_5,
  API429_BOARD_6,
  API429_BOARD_7,
  API429_BOARD_8,

API429_BOARD_9,
API429_BOARD_10,
API429_BOARD_11,
API429_BOARD_12,
API429_BOARD_13,
API429_BOARD_14,
API429_BOARD_15,
API429_BOARD_16,
API429_BOARD_17,
API429_BOARD_18,
API429_BOARD_19,
API429_BOARD_20,
API429_BOARD_21,
API429_BOARD_22,
API429_BOARD_23,
API429_BOARD_24,
API429_BOARD_25,
API429_BOARD_26,
API429_BOARD_27,
API429_BOARD_28,
API429_BOARD_29,
API429_BOARD_30,
API429_BOARD_31,
API429_BOARD_32,
API429_BOARD_1 = 0,
API429_BOARD_2,
API429_BOARD_3,
API429_BOARD_4,
API429_BOARD_5,
API429_BOARD_6,
API429_BOARD_7,
API429_BOARD_8,
API429_BOARD_9,
API429_BOARD_10,
API429_BOARD_11,
API429_BOARD_12,
API429_BOARD_13,
API429_BOARD_14,
API429_BOARD_15,
API429_BOARD_16,
API429_BOARD_17,
API429_BOARD_18,
API429_BOARD_19,
API429_BOARD_20,
API429_BOARD_21,
API429_BOARD_22,
API429_BOARD_23,
API429_BOARD_24,
API429_BOARD_25,

API429_BOARD_26,
API429_BOARD_27,
API429_BOARD_28,
API429_BOARD_29,
API429_BOARD_30,
API429_BOARD_31,
API429_BOARD_32 }

- enum api429_pxi_trigger_line {
API429_TRG_PXI0 = 0,
API429_TRG_PXI1,
API429_TRG_PXI2,
API429_TRG_PXI3,
API429_TRG_PXI4,
API429_TRG_PXI5,
API429_TRG_PXI6,
API429_TRG_PXI7,
API429_TRG_PBI429,
API429_TRG_PXI0 = 0,
API429_TRG_PXI1,
API429_TRG_PXI2,
API429_TRG_PXI3,
API429_TRG_PXI4,
API429_TRG_PXI5,
API429_TRG_PXI6,
API429_TRG_PXI7,
API429_TRG_PBI429 }
- enum api429_pxi_mode {
API429_PXI_SET_TRG = 0,
API429_PXI_CLR_TRG,
API429_PXI_SET_TTSRC_BACKPLANE,
API429_PXI_SET_TTSRC_FRONT,
API429_PXI_SET_TRG = 0,
API429_PXI_CLR_TRG,
API429_PXI_SET_TTSRC_BACKPLANE,
API429_PXI_SET_TTSRC_FRONT }
- enum api429_low_speed_grade {
API429_LS_GRADE_12_50 = 0,
API429_LS_GRADE_13_33,
API429_LS_GRADE_12_50 = 0,
API429_LS_GRADE_13_33 }
- enum api429_speed_modifier {
API429_SPEED_MOD_NORMAL = 0,
API429_SPEED_MOD_INC_9_1 = 1,
API429_SPEED_MOD_DEC_7_7 = 2,
API429_SPEED_MOD_INC_20 = 3,
API429_SPEED_MOD_NORMAL = 0,
API429_SPEED_MOD_INC_9_1 = 1,
API429_SPEED_MOD_DEC_7_7 = 2,
API429_SPEED_MOD_INC_20 = 3 }

- enum api429_irig_source {
  API429_IRIG_ONBOARD = 0,
  API429_IRIG_EXTERN = 1,
  API429_IRIG_ONBOARD = 0,
  API429_IRIG_EXTERN = 1 }
- enum api429_channel_coupling {
  API429_CHANNEL_COUPLING_EXT = 0,
  API429_CHANNEL_COUPLING_DUAL,
  API429_CHANNEL_COUPLING_INT,
  API429_CHANNEL_COUPLING_EXT = 0,
  API429_CHANNEL_COUPLING_DUAL,
  API429_CHANNEL_COUPLING_INT }

Functions

- AiReturn Api429BoardInfoGet (AiUInt8 board_handle, struct api429_board_info *board_info)

  Get properties of a specific device.
- AiReturn Api429BoardLsGradeSet (AiUInt8 board_handle, enum api429_low_speed_grade low_speed_grade)

  Set Arinc 429 low speed graduation for a specific board.
- AiReturn Api429BoardSpeedModifierSet (AiUInt8 board_handle, enum api429_speed_modifier speed_modifier)

  Set Arinc 429 speed modification value for a specific boards.
- AiReturn Api429BoardTimeSet (AiUInt8 board_handle, struct api429_time *irig_time)

  Set onboard IRIG time.
- AiReturn Api429BoardTimeGet (AiUInt8 board_handle, struct api429_time *irig_time)

  Get IRIG time.
- AiReturn Api429BoardTimeSourceSet (AiUInt8 board_handle, enum api429_irig_source new_source)

  Set IRIG time source.
- AiReturn Api429BoardTimeSourceGet (AiUInt8 board_handle, enum api429_irig_source *source, AiBoolean *in_sync)

  Get IRIG time source.
- const char * Api429BoardNameGet (AiUInt8 board_handle)

  Get name of specific board.
- AiReturn Api429BoardReset (AiUInt8 board_handle)

  Resets whole board to power-up initial state.
- AiReturn Api429BoardOpen (enum api429_board_id board_id, const char *ac_SrvName, AiUInt8 *handle)

  Open access to a specific board.
- AiReturn Api429BoardClose (AiUInt8 board_handle)

  Close access to device.
- AiReturn Api429BoardSelftest (AiUInt8 board_handle, AiUInt8 b_Func, AiUInt8 *pb_Status, AiUInt8 *pb_Echo)

  Perform board self-test.

- AiReturn Api429BoardChannelCouplingSet (AiUInt8 board_handle, enum api429_channel_coupling coupling)

  Function to control the channel coupling.
- AiReturn Api429BoardChannelCouplingInfo (AiUInt8 board_handle, AiUInt8 auc_Connection[API429_MAX_CHANNELS+1])

  Function to get information about the fixed internal channel coupling.
- AiReturn Api429BoardChannelCouplingIsSupported (AiUInt8 board_handle, AiBoolean *is_supported)

  Function to get information if internal coupling is supported.
- AiReturn Api429BoardPXITriggerConfigure (AiUInt8 board_handle, enum api429_pxi_mode ul_Mode, enum api429_pxi_trigger_line ul_TrgSource, enum api429_pxi_trigger_line ul_TrgDest, AiBoolean ul_TTClear)

  Configure PXI trigger.
- AiReturn Api429BoardPXIGeographicalAddressGet (AiUInt8 boardHandle, AiUInt32 *pxiSlotGeographicalAddress)

  Request PXI geographical address.
- AiReturn Api429BoardServerInfoGet (AiUInt8 board_handle, struct api429_remote_server *server_info)

  Request extended information from a host.

3.1.1 Detailed Description

This module contains functionality related to AIM Arinc 429 board handling

3.1.2 Macro Definition Documentation

API429_BOARD_CAP_PXI_TRG_FLAG [1/2]

```
#define API429_BOARD_CAP_PXI_TRG_FLAG (1 << 0)
```

Flag used in api429_board_info capability flags
which indicates if a board supports PXI trigger functionality

Definition at line 53 of file Api429Board.h.

API429_BOARD_CAP_PXI_TRG_FLAG [2/2]

```
#define API429_BOARD_CAP_PXI_TRG_FLAG (1 << 0)
```

Flag used in api429_board_info capability flags
which indicates if a board supports PXI trigger functionality

Definition at line 53 of file Api429Board - LV.h.

API429_BOARD_HAS_PXI_TRIGGER [1/2]

```
#define API429_BOARD_HAS_PXI_TRIGGER(
                board_info ) (board_info)->capability_flags & API429_BOARD_CAP_PXI_TRG_FLAG ? AiTrue :  Ai↩
False
```

Takes a pointer to api429_board_info and extracts the PXI trigger capability from it.
Will return AiTrue if board has PXI triggers, AiFalse if not

Definition at line 94 of file Api429Board.h.

API429_BOARD_HAS_PXI_TRIGGER [2/2]

```
#define API429_BOARD_HAS_PXI_TRIGGER(
                board_info ) (board_info)->capability_flags & API429_BOARD_CAP_PXI_TRG_FLAG ? AiTrue :  Ai↩
False
```

Takes a pointer to api429_board_info and extracts the PXI trigger capability from it.
Will return AiTrue if board has PXI triggers, AiFalse if not

Definition at line 94 of file Api429Board - LV.h.

API429_BOARD_TX_INHIBIT_FLAG [1/2]

```
#define API429_BOARD_TX_INHIBIT_FLAG (1 << 12)
```

Flag used in api429_board_info capability flags
which indicates if board tx inhibit is activated

Definition at line 67 of file Api429Board - LV.h.

API429_BOARD_TX_INHIBIT_FLAG [2/2]

```
#define API429_BOARD_TX_INHIBIT_FLAG (1 << 12)
```

Flag used in api429_board_info capability flags
which indicates if board tx inhibit is activated

Definition at line 67 of file Api429Board.h.

API429_BOARD_TX_INHIBIT_IS_ACTIVATED [1/2]

```
#define API429_BOARD_TX_INHIBIT_IS_ACTIVATED(
                board_info ) (board_info)->capability_flags & API429_BOARD_TX_INHIBIT_FLAG ? AiTrue :  AiFalse
```

Takes a pointer to api429_board_info and extracts the TX inhibit capability from it.
Will return AiTrue if TX inhibit is activated, AiFalse if not

Definition at line 110 of file Api429Board.h.

API429_BOARD_TX_INHIBIT_IS_ACTIVATED [2/2]

```
#define API429_BOARD_TX_INHIBIT_IS_ACTIVATED(
                board_info ) (board_info)->capability_flags & API429_BOARD_TX_INHIBIT_FLAG ? AiTrue :  AiFalse
```

Takes a pointer to api429_board_info and extracts the TX inhibit capability from it.
Will return AiTrue if TX inhibit is activated, AiFalse if not

Definition at line 110 of file Api429Board - LV.h.

API429_BOARD_TYPE_EMBEDDED_FLAG [1/2]

```
#define API429_BOARD_TYPE_EMBEDDED_FLAG (1 << 4)
```

Flag used in api429_board_info capability flags
which indicates if board application type is embedded

Definition at line 60 of file Api429Board - LV.h.

API429_BOARD_TYPE_EMBEDDED_FLAG [2/2]

```
#define API429_BOARD_TYPE_EMBEDDED_FLAG (1 << 4)
```

Flag used in api429_board_info capability flags
which indicates if board application type is embedded

Definition at line 60 of file Api429Board.h.

API429_BOARD_TYPE_IS_EMBEDDED [1/2]

```
#define API429_BOARD_TYPE_IS_EMBEDDED(
                board_info ) (board_info)->capability_flags & API429_BOARD_TYPE_EMBEDDED_FLAG ? AiTrue :  Ai↩
False
```

Takes a pointer to api429_board_info and extracts the board embedded type from it.
Will return AiTrue if board application type is embedded, AiFalse if not

Definition at line 102 of file Api429Board.h.

API429_BOARD_TYPE_IS_EMBEDDED [2/2]

```
#define API429_BOARD_TYPE_IS_EMBEDDED(
                board_info ) (board_info)->capability_flags & API429_BOARD_TYPE_EMBEDDED_FLAG ? AiTrue :  Ai↩
False
```

Takes a pointer to api429_board_info and extracts the board embedded type from it.
Will return AiTrue if board application type is embedded, AiFalse if not

Definition at line 102 of file Api429Board - LV.h.

API429_LS_GRADE_DEFAULT [1/2]

```
#define API429_LS_GRADE_DEFAULT API429_LS_GRADE_12_50
```

Default low-speed graduation of a board

Definition at line 286 of file Api429Board - LV.h.

API429_LS_GRADE_DEFAULT [2/2]

```
#define API429_LS_GRADE_DEFAULT API429_LS_GRADE_12_50
```

Default low-speed graduation of a board

Definition at line 286 of file Api429Board.h.

API429_MAX_BOARDNAME_LEN [1/2]

```
#define API429_MAX_BOARDNAME_LEN 32
```

Maximum length of board names in number of single-byte characters

Definition at line 31 of file Api429Board.h.

API429_MAX_BOARDNAME_LEN [2/2]

```
#define API429_MAX_BOARDNAME_LEN 32
```

Maximum length of board names in number of single-byte characters

Definition at line 31 of file Api429Board - LV.h.

API429_MAX_CHANNELS [1/2]

```
#define API429_MAX_CHANNELS (MAX_API429_BIU*MAX_API429_BIU_CHN)
```

Maximum number of channels for an Arinc 429 board

Definition at line 86 of file Api429Board.h.

API429_MAX_CHANNELS [2/2]

```
#define API429_MAX_CHANNELS (MAX_API429_BIU*MAX_API429_BIU_CHN)
```

Maximum number of channels for an Arinc 429 board

Definition at line 86 of file Api429Board - LV.h.

API429_MAX_OS_INFO_LEN [1/2]

```
#define API429_MAX_OS_INFO_LEN 128
```

Maximum length of Operating System information in number of single byte characters

Definition at line 45 of file Api429Board.h.

API429_MAX_OS_INFO_LEN [2/2]

```
#define API429_MAX_OS_INFO_LEN 128
```

Maximum length of Operating System information in number of single byte characters

Definition at line 45 of file Api429Board - LV.h.

API429_MAX_URL_LEN [1/2]

```
#define API429_MAX_URL_LEN 128
```

Maximum length of server URLs in number of single byte characters

Definition at line 38 of file Api429Board.h.

API429_MAX_URL_LEN [2/2]

```
#define API429_MAX_URL_LEN 128
```

Maximum length of server URLs in number of single byte characters

Definition at line 38 of file Api429Board - LV.h.

MAX_API429_BIU [1/2]

```
#define MAX_API429_BIU 2
```

maximum number of BIU's on PCI board

Definition at line 73 of file Api429Board.h.

MAX_API429_BIU [2/2]

```
#define MAX_API429_BIU 2
```

maximum number of BIU's on PCI board

Definition at line 73 of file Api429Board - LV.h.

MAX_API429_BIU_CHN [1/2]

```
#define MAX_API429_BIU_CHN 16
```

maximum number of channels on BIU's on PCI board

Definition at line 79 of file Api429Board.h.

MAX_API429_BIU_CHN [2/2]

```
#define MAX_API429_BIU_CHN 16
```

maximum number of channels on BIU's on PCI board

Definition at line 79 of file Api429Board - LV.h.

3.1.3 Typedef Documentation

TY_API429_BOARD_INFO

```
TY_API429_BOARD_INFO
```

Convenience typedef for api429_board_info

Definition at line 243 of file Api429Board.h.

TY_API429_TIME

TY_API429_TIME

Convenience typedef for api429_time

Definition at line 203 of file Api429Board.h.

TY_E_API429_CHANNEL_COUPLING

TY_E_API429_CHANNEL_COUPLING

Convenience macro for api429_channel_coupling

Definition at line 320 of file Api429Board.h.

TY_E_API429_IRIG_SOURCE

TY_E_API429_IRIG_SOURCE

Convenience macro for api429_speed_modifier

Definition at line 303 of file Api429Board.h.

TY_E_API429_LOW_SPEED_GRADE

TY_E_API429_LOW_SPEED_GRADE

Convenience typedef for api429_low_speed_grade

Definition at line 260 of file Api429Board.h.

TY_E_API429_SPEED_MODIFIER

TY_E_API429_SPEED_MODIFIER

Convenience macro for api429_speed_modifier

Definition at line 279 of file Api429Board.h.

3.1.4 Enumeration Type Documentation

api429_board_id [1/2]

enum api429_board_id

Enumeration of all board IDs that are supported by the API.

Enumerator

API429_BOARD_1	First board detected by API
API429_BOARD_2	Second board detected by API
API429_BOARD_3	Third board detected by API
API429_BOARD_4	4th board detected by API
API429_BOARD_5	5th board detected by API
API429_BOARD_6	6th board detected by API
API429_BOARD_7	7th board detected by API
API429_BOARD_8	8th board detected by API
API429_BOARD_9	9th board detected by API
API429_BOARD_10	10th board detected by API
API429_BOARD_11	11th board detected by API
API429_BOARD_12	12th board detected by API
API429_BOARD_13	13th board detected by API
API429_BOARD_14	14th board detected by API
API429_BOARD_15	15th board detected by API
API429_BOARD_16	16th board detected by API
API429_BOARD_17	17th board detected by API
API429_BOARD_18	18th board detected by API
API429_BOARD_19	19th board detected by API
API429_BOARD_20	20th board detected by API
API429_BOARD_21	21st board detected by API
API429_BOARD_22	22nd board detected by API
API429_BOARD_23	23rd board detected by API
API429_BOARD_24	24th board detected by API
API429_BOARD_25	25th board detected by API
API429_BOARD_26	26th board detected by API
API429_BOARD_27	27th board detected by API
API429_BOARD_28	28th board detected by API
API429_BOARD_29	29th board detected by API
API429_BOARD_30	30th board detected by API
API429_BOARD_31	31st board detected by API

Enumerator

API429_BOARD_32	32nd board detected by API
API429_BOARD_1	First board detected by API
API429_BOARD_2	Second board detected by API
API429_BOARD_3	Third board detected by API
API429_BOARD_4	4th board detected by API
API429_BOARD_5	5th board detected by API
API429_BOARD_6	6th board detected by API
API429_BOARD_7	7th board detected by API
API429_BOARD_8	8th board detected by API
API429_BOARD_9	9th board detected by API
API429_BOARD_10	10th board detected by API
API429_BOARD_11	11th board detected by API
API429_BOARD_12	12th board detected by API
API429_BOARD_13	13th board detected by API
API429_BOARD_14	14th board detected by API
API429_BOARD_15	15th board detected by API
API429_BOARD_16	16th board detected by API
API429_BOARD_17	17th board detected by API
API429_BOARD_18	18th board detected by API
API429_BOARD_19	19th board detected by API
API429_BOARD_20	20th board detected by API
API429_BOARD_21	21st board detected by API
API429_BOARD_22	22nd board detected by API
API429_BOARD_23	23rd board detected by API
API429_BOARD_24	24th board detected by API
API429_BOARD_25	25th board detected by API
API429_BOARD_26	26th board detected by API
API429_BOARD_27	27th board detected by API
API429_BOARD_28	28th board detected by API
API429_BOARD_29	29th board detected by API
API429_BOARD_30	30th board detected by API
API429_BOARD_31	31st board detected by API
API429_BOARD_32	32nd board detected by API

Definition at line 117 of file Api429Board.h.

api429_board_id [2/2]

```
enum api429_board_id
```

Enumerator

API429_BOARD_1	First board detected by API
API429_BOARD_2	Second board detected by API
API429_BOARD_3	Third board detected by API
API429_BOARD_4	4th board detected by API
API429_BOARD_5	5th board detected by API
API429_BOARD_6	6th board detected by API
API429_BOARD_7	7th board detected by API
API429_BOARD_8	8th board detected by API
API429_BOARD_9	9th board detected by API
API429_BOARD_10	10th board detected by API
API429_BOARD_11	11th board detected by API
API429_BOARD_12	12th board detected by API
API429_BOARD_13	13th board detected by API
API429_BOARD_14	14th board detected by API
API429_BOARD_15	15th board detected by API
API429_BOARD_16	16th board detected by API
API429_BOARD_17	17th board detected by API
API429_BOARD_18	18th board detected by API
API429_BOARD_19	19th board detected by API
API429_BOARD_20	20th board detected by API
API429_BOARD_21	21st board detected by API
API429_BOARD_22	22nd board detected by API
API429_BOARD_23	23rd board detected by API
API429_BOARD_24	24th board detected by API
API429_BOARD_25	25th board detected by API
API429_BOARD_26	26th board detected by API
API429_BOARD_27	27th board detected by API
API429_BOARD_28	28th board detected by API
API429_BOARD_29	29th board detected by API
API429_BOARD_30	30th board detected by API
API429_BOARD_31	31st board detected by API
API429_BOARD_32	32nd board detected by API
API429_BOARD_1	First board detected by API
API429_BOARD_2	Second board detected by API
API429_BOARD_3	Third board detected by API
API429_BOARD_4	4th board detected by API
API429_BOARD_5	5th board detected by API

Enumerator

API429_BOARD_6	6th board detected by API
API429_BOARD_7	7th board detected by API
API429_BOARD_8	8th board detected by API
API429_BOARD_9	9th board detected by API
API429_BOARD_10	10th board detected by API
API429_BOARD_11	11th board detected by API
API429_BOARD_12	12th board detected by API
API429_BOARD_13	13th board detected by API
API429_BOARD_14	14th board detected by API
API429_BOARD_15	15th board detected by API
API429_BOARD_16	16th board detected by API
API429_BOARD_17	17th board detected by API
API429_BOARD_18	18th board detected by API
API429_BOARD_19	19th board detected by API
API429_BOARD_20	20th board detected by API
API429_BOARD_21	21st board detected by API
API429_BOARD_22	22nd board detected by API
API429_BOARD_23	23rd board detected by API
API429_BOARD_24	24th board detected by API
API429_BOARD_25	25th board detected by API
API429_BOARD_26	26th board detected by API
API429_BOARD_27	27th board detected by API
API429_BOARD_28	28th board detected by API
API429_BOARD_29	29th board detected by API
API429_BOARD_30	30th board detected by API
API429_BOARD_31	31st board detected by API
API429_BOARD_32	32nd board detected by API

Definition at line 117 of file Api429Board - LV.h.

api429_channel_coupling [1/2]

`enum api429_channel_coupling`

Enumeration of channel coupling modes

Enumerator

API429_CHANNEL_COUPLING_EXT	All channels will send/receive physically to/from bus.

Enumerator

API429_CHANNEL_COUPLING_DUAL	All channels will send/receive physically to/from bus and the send data is also internally looped to its peer channel.
API429_CHANNEL_COUPLING_INT	All channels are physically disconnected from bus. Sent data is internally looped to the peer channel.
API429_CHANNEL_COUPLING_EXT	All channels will send/receive physically to/from bus.
API429_CHANNEL_COUPLING_DUAL	All channels will send/receive physically to/from bus and the send data is also internally looped to its peer channel.
API429_CHANNEL_COUPLING_INT	All channels are physically disconnected from bus. Sent data is internally looped to the peer channel.

Definition at line 309 of file Api429Board.h.

api429_channel_coupling [2/2]

enum api429_channel_coupling

Enumerator

API429_CHANNEL_COUPLING_EXT	All channels will send/receive physically to/from bus.
API429_CHANNEL_COUPLING_DUAL	All channels will send/receive physically to/from bus and the send data is also internally looped to its peer channel.
API429_CHANNEL_COUPLING_INT	All channels are physically disconnected from bus. Sent data is internally looped to the peer channel.
API429_CHANNEL_COUPLING_EXT	All channels will send/receive physically to/from bus.
API429_CHANNEL_COUPLING_DUAL	All channels will send/receive physically to/from bus and the send data is also internally looped to its peer channel.
API429_CHANNEL_COUPLING_INT	All channels are physically disconnected from bus. Sent data is internally looped to the peer channel.

Definition at line 309 of file Api429Board - LV.h.

api429_irig_source [1/2]

enum api429_irig_source

Enumeration of all supported irig source values Used in Api429BoardTimeSourceGet and Api429↩ BoardTimeSourceSet.

Enumerator

API429_IRIG_ONBOARD	use the IRIG time that was generated onboard
API429_IRIG_EXTERN	use the IRIG signal of an external IRIG time generator. The outgoing signal IRIG OUT of this board must not be used in this mode.
API429_IRIG_ONBOARD	use the IRIG time that was generated onboard
API429_IRIG_EXTERN	use the IRIG signal of an external IRIG time generator. The outgoing signal IRIG OUT of this board must not be used in this mode.

Definition at line 294 of file Api429Board.h.

api429_irig_source [2/2]

`enum api429_irig_source`

Enumerator

API429_IRIG_ONBOARD	use the IRIG time that was generated onboard
API429_IRIG_EXTERN	use the IRIG signal of an external IRIG time generator. The outgoing signal IRIG OUT of this board must not be used in this mode.
API429_IRIG_ONBOARD	use the IRIG time that was generated onboard
API429_IRIG_EXTERN	use the IRIG signal of an external IRIG time generator. The outgoing signal IRIG OUT of this board must not be used in this mode.

Definition at line 294 of file Api429Board - LV.h.

api429_low_speed_grade [1/2]

`enum api429_low_speed_grade`

Enumeration of all possible Arinc 429 low-speed gradations. Used as input parameter for Api429Board↩ LsGradeSet

Enumerator

API429_LS_GRADE_12_50	Default Arinc 429 Low-speed grade of 12,5 kBit/s.
API429_LS_GRADE_13_33	Intermediate Arinc 429 Low-Speed grade of 13,33 kBit/s.
API429_LS_GRADE_12_50	Default Arinc 429 Low-speed grade of 12,5 kBit/s.

Enumerator

API429_LS_GRADE_13_33	Intermediate Arinc 429 Low-Speed grade of 13,33 kBit/s.

Definition at line 251 of file Api429Board.h.

api429_low_speed_grade [2/2]

```
enum api429_low_speed_grade
```

Enumerator

API429_LS_GRADE_12_50	Default Arinc 429 Low-speed grade of 12,5 kBit/s.
API429_LS_GRADE_13_33	Intermediate Arinc 429 Low-Speed grade of 13,33 kBit/s.
API429_LS_GRADE_12_50	Default Arinc 429 Low-speed grade of 12,5 kBit/s.
API429_LS_GRADE_13_33	Intermediate Arinc 429 Low-Speed grade of 13,33 kBit/s.

Definition at line 251 of file Api429Board - LV.h.

api429_pxi_mode [1/2]

```
enum api429_pxi_mode
```

Enumerator

API429_PXI_SET_TRG	Define a new Trigger combination using trigger source and destination given in parameters 'ul_TrgSource' and 'ul_TrgDest'.
API429_PXI_CLR_TRG	Clear all previously defined trigger combinations.
API429_PXI_SET_TTSRC_BACKPLANE	The external 10MHz PXI-Reference Clock from the PXI-Rack Backplane is used for Time Tagging.
API429_PXI_SET_TTSRC_FRONT	The external 1KHz analog IRIG-B input signal via Frontpanel-I/O is used for Time Tagging.
API429_PXI_SET_TRG	Define a new Trigger combination using trigger source and destination given in parameters 'ul_TrgSource' and 'ul_TrgDest'.
API429_PXI_CLR_TRG	Clear all previously defined trigger combinations.
API429_PXI_SET_TTSRC_BACKPLANE	The external 10MHz PXI-Reference Clock from the PXI-Rack Backplane is used for Time Tagging.

Enumerator

API429_PXI_SET_TTSRC_FRONT	The external 1KHz analog IRIG-B input signal via Frontpanel-I/O is used for Time Tagging.

Definition at line 176 of file Api429Board - LV.h.

api429_pxi_mode [2/2]

enum api429_pxi_mode

Enumeration of all possible Arinc 429 pxi trigger sources

Enumerator

API429_PXI_SET_TRG	Define a new Trigger combination using trigger source and destination given in parameters 'ul_TrgSource' and 'ul_TrgDest'.
API429_PXI_CLR_TRG	Clear all previously defined trigger combinations.
API429_PXI_SET_TTSRC_BACKPLANE	The external 10MHz PXI-Reference Clock from the PXI-Rack Backplane is used for Time Tagging.
API429_PXI_SET_TTSRC_FRONT	The external 1KHz analog IRIG-B input signal via Frontpanel-I/O is used for Time Tagging.
API429_PXI_SET_TRG	Define a new Trigger combination using trigger source and destination given in parameters 'ul_TrgSource' and 'ul_TrgDest'.
API429_PXI_CLR_TRG	Clear all previously defined trigger combinations.
API429_PXI_SET_TTSRC_BACKPLANE	The external 10MHz PXI-Reference Clock from the PXI-Rack Backplane is used for Time Tagging.
API429_PXI_SET_TTSRC_FRONT	The external 1KHz analog IRIG-B input signal via Frontpanel-I/O is used for Time Tagging.

Definition at line 176 of file Api429Board.h.

api429_pxi_trigger_line [1/2]

enum api429_pxi_trigger_line

Enumerator

API429_TRG_PXI0	PXI Trigger Line 0.

Enumerator

API429_TRG_PXI1	PXI Trigger Line 1.
API429_TRG_PXI2	PXI Trigger Line 2.
API429_TRG_PXI3	PXI Trigger Line 3.
API429_TRG_PXI4	PXI Trigger Line 4.
API429_TRG_PXI5	PXI Trigger Line 5.
API429_TRG_PXI6	PXI Trigger Line 6.
API429_TRG_PXI7	PXI Trigger Line 7.
API429_TRG_PBI429	Onboard Trigger.
API429_TRG_PXI0	PXI Trigger Line 0.
API429_TRG_PXI1	PXI Trigger Line 1.
API429_TRG_PXI2	PXI Trigger Line 2.
API429_TRG_PXI3	PXI Trigger Line 3.
API429_TRG_PXI4	PXI Trigger Line 4.
API429_TRG_PXI5	PXI Trigger Line 5.
API429_TRG_PXI6	PXI Trigger Line 6.
API429_TRG_PXI7	PXI Trigger Line 7.
API429_TRG_PBI429	Onboard Trigger.

Definition at line 158 of file Api429Board - LV.h.

api429_pxi_trigger_line [2/2]

```
enum api429_pxi_trigger_line
```

Enumeration of all possible Arinc 429 pxi trigger line (as source or destination)

Enumerator

API429_TRG_PXI0	PXI Trigger Line 0.
API429_TRG_PXI1	PXI Trigger Line 1.
API429_TRG_PXI2	PXI Trigger Line 2.
API429_TRG_PXI3	PXI Trigger Line 3.
API429_TRG_PXI4	PXI Trigger Line 4.
API429_TRG_PXI5	PXI Trigger Line 5.
API429_TRG_PXI6	PXI Trigger Line 6.
API429_TRG_PXI7	PXI Trigger Line 7.
API429_TRG_PBI429	Onboard Trigger.
API429_TRG_PXI0	PXI Trigger Line 0.
API429_TRG_PXI1	PXI Trigger Line 1.

Enumerator

API429_TRG_PXI2	PXI Trigger Line 2.
API429_TRG_PXI3	PXI Trigger Line 3.
API429_TRG_PXI4	PXI Trigger Line 4.
API429_TRG_PXI5	PXI Trigger Line 5.
API429_TRG_PXI6	PXI Trigger Line 6.
API429_TRG_PXI7	PXI Trigger Line 7.
API429_TRG_PBI429	Onboard Trigger.

Definition at line 158 of file Api429Board.h.

api429_speed_modifier [1/2]

```
enum api429_speed_modifier
```

Enumerator

API429_SPEED_MOD_NORMAL	Normal Arinc 429 bus speed.
API429_SPEED_MOD_INC_9_1	Increases bus speed by 9.1%.
API429_SPEED_MOD_DEC_7↩_7	Decreases bus speed by 7.7%.
API429_SPEED_MOD_INC_20	Increases bus speed by 20%.
API429_SPEED_MOD_NORMAL	Normal Arinc 429 bus speed.
API429_SPEED_MOD_INC_9_1	Increases bus speed by 9.1%.
API429_SPEED_MOD_DEC_7↩_7	Decreases bus speed by 7.7%.
API429_SPEED_MOD_INC_20	Increases bus speed by 20%.

Definition at line 268 of file Api429Board - LV.h.

api429_speed_modifier [2/2]

```
enum api429_speed_modifier
```

Enumeration of all supported Arinc 429 bus speed modifications. Used as input parameter in Api429↩BoardSpeedModifierSet.

Enumerator

API429_SPEED_MOD_NORMAL	Normal Arinc 429 bus speed.
API429_SPEED_MOD_INC_9_1	Increases bus speed by 9.1%.
API429_SPEED_MOD_DEC_7↩_7	Decreases bus speed by 7.7%.
API429_SPEED_MOD_INC_20	Increases bus speed by 20%.
API429_SPEED_MOD_NORMAL	Normal Arinc 429 bus speed.
API429_SPEED_MOD_INC_9_1	Increases bus speed by 9.1%.
API429_SPEED_MOD_DEC_7↩_7	Decreases bus speed by 7.7%.
API429_SPEED_MOD_INC_20	Increases bus speed by 20%.

Definition at line 268 of file Api429Board.h.

3.1.5 Function Documentation

Api429BoardChannelCouplingInfo()

```
AiReturn Api429BoardChannelCouplingInfo (
                AiUInt8 board_handle,
                AiUInt8 auc_Connection[API429_MAX_CHANNELS+1] )
```

Function to get information about the fixed internal channel coupling.

Function to learn how the channels are connected with each other, if the internal channel coupling is used. This functionality is known as channel loop in the firmware. This command was named channel wrap, to avoid confusion with the loop/pollution mode.

Parameters

in	*board_handle*	Handle to board
out	*auc_Connection*	- array that tells the destination of a channel wrap. A value of zero means that this channel is not connected. auc_Connection[0] is not used for a channel. Example: if channel 1 is connected to channel 5, then auc_Connection[1] would be 5 and auc_Connection[5] would be 1.

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429BoardChannelCouplingIsSupported()

```
AiReturn Api429BoardChannelCouplingIsSupported (
                AiUInt8 board_handle,
                AiBoolean * is_supported )
```

Function to get information if internal coupling is supported.

Function to learn if internal channel coupling is supported.

Parameters

in	*board_handle*	Handle to board
out	*is_supported*	- AiTrue if channel coupling is supported by the hardware, else "0"

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429BoardChannelCouplingSet()

```
AiReturn Api429BoardChannelCouplingSet (
                AiUInt8 board_handle,
                enum api429_channel_coupling coupling )
```

Function to control the channel coupling.

Function to enable or disable the internal channel coupling. It is also possible to send data to the front connector and also internally couple the channel at the same time. This functionality is known as channel loop in the firmware. This command was named channel coupling, to avoid confusion with the loop/pollution mode.

Parameters

in	*board_handle*	Handle to board
in	*coupling*	coupling mode to set

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429BoardClose()

```
AiReturn Api429BoardClose (
                AiUInt8 board_handle )
```

Close access to device.

This function is used to close a connection to a device.

Parameters

in	*board_handle*	handle to board to close

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429BoardInfoGet()

```
AiReturn Api429BoardInfoGet (
                AiUInt8 board_handle,
                struct api429_board_info * board_info )
```

Get properties of a specific device.

This function can be used to get some static properties of a specific device

Parameters

in	*board_handle*	handle of the board to get properties of
out	*board_info*	pointer to api429_board_info where properties will be stored

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429BoardLsGradeSet()

```
AiReturn Api429BoardLsGradeSet (
                AiUInt8 board_handle,
                enum api429_low_speed_grade low_speed_grade )
```

Set Arinc 429 low speed graduation for a specific board.

This function can be used to switch a board between the Arinc429 low speed graduations of 12,5 kBit/s and 13,33 kBit/s.
This setting will only have an effect with channels that are configured for low-speed transmission or reception. This function affects the whole board and should not be used when a channel is active

Parameters

in	*board_handle*	handle to the board to set low speed grade for
in	*low_speed_grade*	the Arinc 429 low speed grade to set. See api429_low_speed_grade

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429BoardNameGet()

```
const char * Api429BoardNameGet (
                AiUInt8 board_handle )
```

Get name of specific board.

Returns a human readable name for a specific hardware module.

Parameters

in	*board_handle*	handle to board to get name of

Returns

- pointer to zero-terminated ASCII string that contains board name
- NULL on error

Api429BoardOpen()

```
AiReturn Api429BoardOpen (
                enum api429_board_id board_id,
                const char * ac_SrvName,
                AiUInt8 * handle )
```

Open access to a specific board.

This function is used to open a specific board.
A board must be opened before it can be used hence this function must be called first to get a valid device handle that can be used in subsequent calls to this device.

Parameters

in	*board_id*	ID of the device to open. See api429_board_id
in	*ac_SrvName*	zero-terminated ASCII string of the server the device is hosted on. Must be 'local' when using devices on local systems
out	*handle*	handle to the opened device is stored here on success. Must be used for all subsequent calls to this device.

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429BoardPXIGeographicalAddressGet()

```
AiReturn Api429BoardPXIGeographicalAddressGet (
                AiUInt8 boardHandle,
                AiUInt32 * pxiSlotGeographicalAddress )
```

Request PXI geographical address.

This function will request the geographical address of the PXI slot where the 429 PXI board is plugged in.

Parameters

in	*board_handle*	handle to board to request PXI geographical address for
out	*geographical*	address of used PXI slot

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429BoardPXITriggerConfigure()

```
AiReturn Api429BoardPXITriggerConfigure (
                AiUInt8 board_handle,
                enum api429_pxi_mode ul_Mode,
                enum api429_pxi_trigger_line ul_TrgSource,
                enum api429_pxi_trigger_line ul_TrgDest,
                AiBoolean ul_TTClear )
```

Configure PXI trigger.

This function is used to combine PXI specific trigger lines with the trigger lines of AIM boards.
It also provides support to switch the IRIG TT source.

Parameters

in	*board_handle*	handle to board to configure PXI triggers on
in	*ul_Mode*	(see api429_pxi_mode)
in	*ul_TrgSource*	trigger source (see api429_pxi_trigger_line)
in	*ul_TrgDest*	trigger destination (see api429_pxi_trigger_line)
in	*ul_TTClear*	Allow to clear time tag with external pulse on PXI TRIGGER INPUT 0 or PXI STAR TRIGGER. Note: Only applicable for mode values 2 (API429_PXI_SET_TTSRC_BACKPLANE) and 3 (API429_PXI_SET_TTSRC_FRONT). Note: If enabled the PXI Trigger Line 0 cannot be used as trigger source or destination.

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429BoardReset()

```
AiReturn Api429BoardReset (
              AiUInt8 board_handle )
```

Resets whole board to power-up initial state.

This function can be used to reset a specific device to its

Parameters

in	*board_handle*	handle to device to reset

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429BoardSelftest()

```
AiReturn Api429BoardSelftest (
              AiUInt8 board_handle,
              AiUInt8 b_Func,
              AiUInt8 * pb_Status,
              AiUInt8 * pb_Echo )
```

Perform board self-test.

Parameters

in	*board_handle*	Handle to board to perform self-test on
in	*b_Func*	Can be used to select specific functions to test • 0 Execute all test • 1 Board Enable Test • 2 Internal Firmware Selftest • 3 Interrupt Test • 4 API Global RAM Test
out	*pb_Status*	status variable is stored here
out	*pb_Echo*	echo variable is stored here

Status	Echo	Description
0	0	Selftest passed
1		Board Enable Test Error
2		Internal Firmware Selftest Error
	1	Disable Board Failed
	2	Enable Board Failed
	3	Selftest Start Failed
	4	RAM Test Failed
	5	Encoder / Decoder Failure
	6	Processor Failure
	7	Other Failure

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429BoardServerInfoGet()

```
AiReturn Api429BoardServerInfoGet (
              AiUInt8 board_handle,
              struct api429_remote_server * server_info )
```

Request extended information from a host.

This function will request extended information about the host system a specific board is situated on

Parameters

in	*board_handle*	handle to board to request host information for
out	*server_info*	server information is stored here. See api429_remote_server

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429BoardSpeedModifierSet()

```
AiReturn Api429BoardSpeedModifierSet (
```

```
                AiUInt8 board_handle,
                enum api429_speed_modifier speed_modifier )
```

Set Arinc 429 speed modification value for a specific boards.

This function allows to adjust the bus speed on all transmit or receive channels of a board by a specific factor.
This function affects the whole board and should not be used when a channel is active The bus speed is no more Arinc 429 compliant after calling this function, unless setting it to API429_SPEED_MOD_↩ NORMAL.

Parameters

in	*board_handle*	handle to board to modify bus speed for
in	*speed_modifier*	Arinc 429 bus speed modifier to set. See api429_speed_modifier

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429BoardTimeGet()

```
AiReturn Api429BoardTimeGet (
                AiUInt8 board_handle,
                struct api429_time * irig_time )
```

Get IRIG time.

This function allows to read the time of the onboard IRIG timecode decoder.

Parameters

in	*board_handle*	handle to board to modify bus speed for
out	*irig_time*	current point of time, as provide by the IRIG timecode decoder. See api429_time

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429BoardTimeSet()

```
AiReturn Api429BoardTimeSet (
                AiUInt8 board_handle,
                struct api429_time * irig_time )
```

Set onboard IRIG time.

This function allows to set the time of the onboard IRIG timecode encoder. The IRIG timecode encoder needs up to three seconds before changing the time

Parameters

in	*board_handle*	handle to board to modify bus speed for
in	*irig_time*	the time the IRIG timecode encoder shall be set to. See api429_time

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429BoardTimeSourceGet()

```
AiReturn Api429BoardTimeSourceGet (
                AiUInt8 board_handle,
                enum api429_irig_source * source,
                AiBoolean * in_sync )
```

Get IRIG time source.

This function provides the source of the IRIG signal and some status information

Parameters

in	*board_handle*	handle to board to modify bus speed for
out	*source*	the source the IRIG timecode decoder receives its signal from. See api429_irig_source
	in_sync	information whether or not an IRIG signal is detected. This value is AiFalse if no valid IRIG signal can be read.

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429BoardTimeSourceSet()

```
AiReturn Api429BoardTimeSourceSet (
                AiUInt8 board_handle,
                enum api429_irig_source new_source )
```

Set IRIG time source.

This function allows to set the source of the IRIG signal.

Parameters

in	*board_handle*	handle to board to modify bus speed for
in	*new_source*	the source the IRIG timecode decoder shall receive its signal from. See api429_irig_source

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

3.2 Raw Board Memory Access

Macros

- #define **API429_MEM_DATA_WIDTH_8BIT** 1
- #define **API429_MEM_DATA_WIDTH_16BIT** 2
- #define **API429_MEM_DATA_WIDTH_32BIT** 4

Typedefs

- typedef enum api429_memory_object TY_E_API429_MEMORY_OBJECT

Enumerations

- enum api429_memory_object {
  API429_MEM_OBJ_XFER = 1,
  API429_MEM_OBJ_XFER_BUF,
  API429_MEM_OBJ_MINFRM,
  API429_MEM_OBJ_MAJFRM,
  API429_MEM_OBJ_LABDESC,
  API429_MEM_OBJ_RXBUF,
  API429_MEM_OBJ_TXCOUNT,
  API429_MEM_OBJ_CHN_DESC,
  API429_MEM_OBJ_RM_BUF,
  API429_MEM_OBJ_MAX }

Functions

- AiReturn Api429BoardMemLocationGet (AiUInt8 board_handle, AiUInt8 uc_Chn, enum api429_memory_object object_type, AiUInt32 id, enum ty_e_mem_type *pul_MemType, AiUInt32 *pul_Offset)

  Get offset of specific on-board data types.
- AiReturn Api429BoardMemTimerAddrGet (AiUInt8 board_handle, AiUInt32 *pul_TimerOffset, enum ty_e_mem_type *pul_MemType)

  Get location of Irig time registers.
- AiReturn Api429BoardMemRead (AiUInt8 board_handle, enum ty_e_mem_type memtype, AiUInt32 offset, AiUInt8 width, void *data_p)

  Read board memory.
- AiReturn Api429BoardMemWrite (AiUInt8 board_handle, enum ty_e_mem_type memtype, AiUInt32 offset, AiUInt8 width, void *data_p)

  Write board memory.
- AiReturn Api429BoardMemBlockRead (AiUInt8 board_handle, enum ty_e_mem_type memtype, AiUInt32 offset, AiUInt8 width, void *data_p, AiUInt32 size, AiUInt32 *pul_BytesRead)

  Read block of board memory.

- AiReturn Api429BoardMemBlockWrite (AiUInt8 board_handle, enum ty_e_mem_type memtype, AiUInt32 offset, AiUInt8 width, void *data_p, AiUInt32 size, AiUInt32 *pul_BytesWritten)

  Write block of board memory.
- AiReturn **Api429BoardMemAlloc** (AiUInt8 board_handle, enum ty_e_mem_type mem_type, AiUInt32 ulSize, AiUInt32 ulTag, AiUInt32 *pulOffset)
- AiReturn **Api429BoardMemFree** (AiUInt8 board_handle, enum ty_e_mem_type mem_type, AiUInt32 ulOffset)
- AiReturn **Api429BoardMemSizeGet** (AiUInt8 board_handle, enum ty_e_mem_type mem_type, AiSize *size)

3.2.1 Detailed Description

This module contains functionality related to raw access to AIM Arinc 429 board memory

3.2.2 Typedef Documentation

TY_E_API429_MEMORY_OBJECT

`TY_E_API429_MEMORY_OBJECT`

Convenience macro for api429_memory_object

Definition at line 50 of file Api429BoardMem.h.

3.2.3 Enumeration Type Documentation

api429_memory_object

`enum api429_memory_object`

Enumeration of all possible memory object types

Enumerator

API429_MEM_OBJ_XFER	Transfer object.
API429_MEM_OBJ_XFER_BUF	Transfer buffer object.
API429_MEM_OBJ_MINFRM	Minor Frame object.
API429_MEM_OBJ_MAJFRM	Major frame object.

Enumerator

API429_MEM_OBJ_LABDESC	Label descriptor object.
API429_MEM_OBJ_RXBUF	Receive buffer object.
API429_MEM_OBJ_TXCOUNT	Transfer counter object.
API429_MEM_OBJ_CHN_DESC	Channel descriptor.
API429_MEM_OBJ_RM_BUF	RM buffer.

Definition at line 31 of file Api429BoardMem.h.

3.2.4 Function Documentation

Api429BoardMemBlockRead()

```
AiReturn Api429BoardMemBlockRead (
                AiUInt8 board_handle,
                enum ty_e_mem_type memtype,
                AiUInt32 offset,
                AiUInt8 width,
                void * data_p,
                AiUInt32 size,
                AiUInt32 * pul_BytesRead )
```

Read block of board memory.

This function is used to read a data block from the AIM board memory in avoidance of AIM board command and acknowledge interface access. This is necessary to access the AIM board memory in case of interrupt. The function does a direct access to the AIM board memory with the specified offset address.

Parameters

in	*board_handle*	handle to the board to read from
in	*memtype*	Memory Type to be accessed
in	*offset*	Byte offset address relative to the start of a specific onboard memory described in parameter 'memtype'.
in	*width*	Data width of access. This is important if data is read from a source with different endianess e.g. if reading at a MVME5100 from the global memory of an AMCX429 board)
out	*data_p*	Data to read to. This pointer should match to the data size given in the parameter 'width'.
in	*size*	Amount of data block elements.
out	*pul_BytesRead*	Amount of bytes actually read.

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429BoardMemBlockWrite()

```
AiReturn Api429BoardMemBlockWrite (
                AiUInt8 board_handle,
                enum ty_e_mem_type memtype,
                AiUInt32 offset,
                AiUInt8 width,
                void * data_p,
                AiUInt32 size,
                AiUInt32 * pul_BytesWritten )
```

Write block of board memory.

This function is used to write a data block to the AIM board memory in avoidance of AIM board command and acknowledge interface access. This is necessary to access the AIM board memory in case of interrupt. The function does a direct access to the AIM board memory with the specified offset address.

Parameters

in	*board_handle*	handle to board to write to
in	*memtype*	Memory Type to be accessed
in	*offset*	Byte offset address relative to the start of a specific onboard memory described in parameter 'memtype'.
in	*width*	Data width of access. This is important if data is read from a source with different endianess e.g. if reading at a MVME5100 from the global memory of an AMCX429 board)
in	*data_p*	Data to write from. This pointer should match to the data size given in the parameter 'width'.
in	*size*	Amount of data block elements.
out	*pul_BytesWritten*	

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429BoardMemLocationGet()

```
AiReturn Api429BoardMemLocationGet (
                AiUInt8 board_handle,
                AiUInt8 uc_Chn,
                enum api429_memory_object object_type,
                AiUInt32 id,
                enum ty_e_mem_type * pul_MemType,
                AiUInt32 * pul_Offset )
```

Get offset of specific on-board data types.

Parameters

in	*board_handle*	handle to board
in	*uc_Chn*	ID of channel the memory object is assigned to
in	*object_type*	memory object type to get location of
in	*id*	The ID of the Xfer/Minor frame/Label/etc. (Not used for all object_types)
out	*pul_MemType*	memory type of requested object is stored here
out	*pul_Offset*	offset in bytes of requested memory object is stored here

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429BoardMemRead()

```
AiReturn Api429BoardMemRead (
                AiUInt8 board_handle,
                enum ty_e_mem_type memtype,
                AiUInt32 offset,
                AiUInt8 width,
                void * data_p )
```

Read board memory.

This function is used to read a byte/word/longword from the AIM board memory in avoidance of AIM board command and acknowledge interface access. This is necessary to access the AIM board memory in case of interrupt. The function does a direct access to the AIM board memory with the specified offset address.

Parameters

in	*board_handle*	handle to board to read from
in	*memtype*	Memory Type to be accessed
in	*offset*	Byte offset address relative to the start of a specific onboard memory described in parameter 'memtype'.
in	*width*	Data width of access. This is important if data is read from a source with different endianess e.g. if reading at a MVME5100 from the global memory of an AMCX429 board)
out	*data_p*	Data to read to. This pointer should match to the data size given in the parameter 'width'.

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429BoardMemTimerAddrGet()

```
AiReturn Api429BoardMemTimerAddrGet (
                AiUInt8 board_handle,
                AiUInt32 * pul_TimerOffset,
                enum ty_e_mem_type * pul_MemType )
```

Get location of Irig time registers.

Parameters

in	*board_handle*	handle of board
out	*pul_TimerOffset*	offset of time registers is stored here
out	*pul_MemType*	memory type where irig time registers are located

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429BoardMemWrite()

```
AiReturn Api429BoardMemWrite (
                AiUInt8 board_handle,
                enum ty_e_mem_type memtype,
                AiUInt32 offset,
                AiUInt8 width,
                void * data_p )
```

Write board memory.

This function is used to write a byte/word/longword to the AIM board memory in avoidance of AIM board command and acknowledge interface access. This is necessary to access the AIM board memory in case of interrupt. The function does a direct access to the AIM board memory with the specified offset address.

Parameters

in	*board_handle*	handle to the board to write to
in	*memtype*	Memory Type to be accessed
in	*offset*	Byte offset address relative to the start of a specific onboard memory described in parameter 'memtype'.
in	*width*	Data width of access. This is important if data is read from a source with different endianess e.g. if reading at a MVME5100 from the global memory of an AMCX429 board)
in	*data_p*	Data to write from. This pointer should match to the data size given in the parameter 'width'.

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

3.3 Channel Handling

Data Structures

- struct api429_channel_info
- union api429_loglist_entry
- struct api429_intr_loglist_entry

Macros

- #define API429_CHN_CFG_UNCONFIGURED 0
- #define API429_CHN_CFG_RX_LABEL_FLAG (1 << 0)
- #define API429_CHN_CFG_RX_POLLUTION_FLAG (1 << 1)
- #define API429_CHN_CFG_RM_LOCAL_FLAG (1 << 2)
- #define API429_CHN_CFG_RM_GLOBAL_FLAG (1 << 3)
- #define API429_CHN_CFG_TX_FRAMING_FLAG (1 << 4)
- #define API429_CHN_CFG_TX_POLLUTION_FLAG (1 << 5)
- #define API429_CHN_CFG_PHYS_REPLAY_FLAG (1 << 6)
- #define API429_CHN_CFG_TX_DYNTAG_FRAMING_FLAG (1 << 7)
- #define API429_CHN_CFG_TX_FIFO_FLAG (1 << 8)
- #define API429_CHN_CFG_TX_RATE_CONTROLLED_FLAG (1 << 9)
- #define API429_CHN_CFG_SDI_ENABLED_FLAG (1 << 10)
- #define API429_CHN_CFG_PARITY_ENABLED_FLAG (1 << 11)
- #define API429_CHANNEL_IN_RX_NORMAL_MODE(channel) ((channel)->config_flags & API429_CHN_CFG_RX_LABEL_FLAG)
- #define API429_CHANNEL_IN_RX_LP_MODE(channel) ((channel)->config_flags & API429_CHN_CFG_RX_POLLUTION_FLAG)
- #define API429_CHANNEL_IN_RX_TX_MIXING_MODE(channel) ((channel)->config_flags & API429_CHN_CFG_RX_MIXING_FLAG)
- #define API429_CHANNEL_IN_LOCAL_MONITORING_MODE(channel) ((channel)->config_flags & API429_CHN_CFG_RM_LOCAL_FLAG)
- #define API429_CHANNEL_IN_GLOBAL_MONITORING_MODE(channel) ((channel)->config_flags & API429_CHN_CFG_RM_GLOBAL_FLAG)
- #define API429_CHANNEL_IN_TX_FRAMING_MODE(channel) ((channel)->config_flags & API429_CHN_CFG_TX_FRAMING_FLAG)
- #define API429_CHANNEL_IN_TX_LP_MODE(channel) ((channel)->config_flags & API429_CHN_CFG_TX_POLLUTION_FLAG)
- #define API429_CHANNEL_IN_PHYS_REPLAY_MODE(channel) ((channel)->config_flags & API429_CHN_CFG_PHYS_REPLAY_FLAG)
- #define API429_CHANNEL_IN_TX_DYNTAG_MODE(channel) ((channel)->config_flags & API429_CHN_CFG_TX_DYNTAG_FRAMING_FLAG)
- #define API429_CHANNEL_IN_TX_FIFO_MODE(channel) ((channel)->config_flags & API429_CHN_CFG_TX_FIFO_FLAG)
- #define API429_CHANNEL_IN_TX_RATE_CONTROLLED_MODE(channel) ((channel)->config_flags & API429_CHN_CFG_TX_RATE_CONTROLLED_FLAG)

- #define API429_CHANNEL_IN_TX_MODE(channel)
- #define API429_CHANNEL_IN_REPLAY_MODE(channel) API429_CHANNEL_IN_BUFFER_REPLAY_MODE(channel) | API429_CHANNEL_IN_PHYS_REPLAY_MODE(channel)
- #define API429_CHANNEL_IN_RX_MODE(channel) API429_CHANNEL_IN_RX_NORMAL_MODE(channel) | API429_CHANNEL_IN_RX_LP_MODE(channel)
- #define API429_CHANNEL_IN_MONITORING_MODE(channel) API429_CHANNEL_IN_LOCAL_MONITORING_MODE(channel) | API429_CHANNEL_IN_GLOBAL_MONITORING_MODE(channel)
- #define API429_CHN_CAP_RX_FLAG (1 << 0)
- #define API429_CHN_CAP_TX_FLAG (1 << 1)
- #define API429_CHN_CAP_VAR_AMP_FLAG (1 << 2)
- #define API429_CHANNEL_CAN_TRANSMIT(channel) ((channel).capability_flags & API429_CHN_CAP_TX_FLAG)
- #define API429_CHANNEL_CAN_RECEIVE(channel) ((channel).capability_flags & API429_CHN_CAP_RX_FLAG)
- #define API429_CHANNEL_HAS_VARIABLE_AMPLITUDE(channel) ((channel).capability_flags & API429_CHN_CAP_VAR_AMP_FLAG)
- #define **API429_CHANNEL_EVENT_TAG**(channel_event) (((channel_event)->ul_Lla >> 20) & 0xFF)

Typedefs

- typedef enum api429_speed TY_E_API429_SPEED
- typedef struct api429_channel_info TY_API429_CHANNEL_INFO
- typedef union api429_loglist_entry TY_API429_LOGLIST_TYPE_ENTRY
- typedef struct api429_intr_loglist_entry TY_API429_INTR_LOGLIST_ENTRY
- typedef void(* API429_CHANNEL_CALLBACK) (AiUInt8 module, AiUInt8 channel, enum api429_event_type type, struct api429_intr_loglist_entry *info)
- typedef API429_CHANNEL_CALLBACK **TY_INT429_FUNC_PTR**

Enumerations

- enum api429_speed {
  API429_LO_SPEED = 0,
  API429_HI_SPEED }
- enum api429_event_type {
  API429_EVENT_UNDEFINED = 0,
  API429_EVENT_TX_HALT,
  API429_EVENT_TX_SKIP,
  API429_EVENT_TX_LABEL,
  API429_EVENT_TX_INDEX,
  API429_EVENT_RX_ANY_LABEL,
  API429_EVENT_RX_INDEX,
  API429_EVENT_RX_ERROR,
  API429_EVENT_FUNC_BLOCK,
  API429_EVENT_RM_TRIGGER,
  API429_EVENT_RM_BUFFER_FULL,

API429_EVENT_RM_BUFFER_HALF_FULL,
API429_EVENT_REPLAY_HALF_BUFFER,
API429_EVENT_REPLAY_STOP,
API429_EVENT_TX_FIFO }

Functions

- AiReturn Api429ChannelInfoGet (AiUInt8 board_handle, AiUInt8 channel_id, struct api429_channel_info *pxInfo)

  Get information about a specific channel.
- AiReturn Api429ChannelClear (AiUInt8 board_handle, AiUInt8 channel_id)

  Clear configuration of a channel.
- AiReturn Api429ChannelStart (AiUInt8 board_handle, AiUInt8 channel_id)

  Start a channel.
- AiReturn Api429ChannelHalt (AiUInt8 board_handle, AiUInt8 channel_id)

  Halt a channel.
- AiReturn Api429ChannelSpeedSet (AiUInt8 board_handle, AiUInt8 b_Chn, enum api429_speed speed)

  Set channel speed.
- AiReturn Api429ChannelCallbackRegister (AiUInt8 board_handle, AiUInt8 uc_Chn, AiUInt8 uc_Type, API429_CHANNEL_CALLBACK pf_IntFunc)

  Registers an event notification handler.
- AiReturn Api429ChannelCallbackUnregister (AiUInt8 board_handle, AiUInt8 uc_Chn, AiUInt8 uc_Type)

  Unregisters an event notification handler.

3.3.1 Detailed Description

This module contains functionality related to generic channel handling independent from receiving or transmitting data

3.3.2 Macro Definition Documentation

API429_CHANNEL_CAN_RECEIVE

```
#define API429_CHANNEL_CAN_RECEIVE(
                channel ) ((channel).capability_flags & API429_CHN_CAP_RX_FLAG)
```

Test macro that indicates if channel is able to receive data

Definition at line 279 of file Api429Channel.h.

API429_CHANNEL_CAN_TRANSMIT

```
#define API429_CHANNEL_CAN_TRANSMIT(
                channel ) ((channel).capability_flags & API429_CHN_CAP_TX_FLAG)
```

Test macro that indicates if channel is able to transmit data

Definition at line 273 of file Api429Channel.h.

API429_CHANNEL_HAS_VARIABLE_AMPLITUDE

```
#define API429_CHANNEL_HAS_VARIABLE_AMPLITUDE(
                channel ) ((channel).capability_flags & API429_CHN_CAP_VAR_AMP_FLAG)
```

Test macro that indicates if channel is able to transmit with variable amplitude

Definition at line 285 of file Api429Channel.h.

API429_CHANNEL_IN_GLOBAL_MONITORING_MODE

```
#define API429_CHANNEL_IN_GLOBAL_MONITORING_MODE(
                channel ) ((channel)->config_flags & API429_CHN_CFG_RM_GLOBAL_FLAG)
```

Checks if channel is configured for global monitoring

Definition at line 172 of file Api429Channel.h.

API429_CHANNEL_IN_LOCAL_MONITORING_MODE

```
#define API429_CHANNEL_IN_LOCAL_MONITORING_MODE(
                channel ) ((channel)->config_flags & API429_CHN_CFG_RM_LOCAL_FLAG)
```

Checks if channel is configured for local monitoring

Definition at line 166 of file Api429Channel.h.

API429_CHANNEL_IN_MONITORING_MODE

```
#define API429_CHANNEL_IN_MONITORING_MODE(
                channel ) API429_CHANNEL_IN_LOCAL_MONITORING_MODE(channel) | API429_CHANNEL_IN_GLOBAL_MONIT↩
ORING_MODE(channel)
```

Checks if channel is in any of the supported monitoring modes

Definition at line 238 of file Api429Channel.h.

API429_CHANNEL_IN_PHYS_REPLAY_MODE

```
#define API429_CHANNEL_IN_PHYS_REPLAY_MODE(
                channel ) ((channel)->config_flags & API429_CHN_CFG_PHYS_REPLAY_FLAG)
```

Checks if channel is configured for physical replay mode

Definition at line 190 of file Api429Channel.h.

API429_CHANNEL_IN_REPLAY_MODE

```
#define API429_CHANNEL_IN_REPLAY_MODE(
                channel ) API429_CHANNEL_IN_BUFFER_REPLAY_MODE(channel) | API429_CHANNEL_IN_PHYS_REPLAY_MODE(channel)
```

Checks if a channel is in any supported replay mode

Definition at line 225 of file Api429Channel.h.

API429_CHANNEL_IN_RX_LP_MODE

```
#define API429_CHANNEL_IN_RX_LP_MODE(
                channel ) ((channel)->config_flags & API429_CHN_CFG_RX_POLLUTION_FLAG)
```

Checks if channel is configured as loop/pollution receiver

Definition at line 154 of file Api429Channel.h.

API429_CHANNEL_IN_RX_MODE

```
#define API429_CHANNEL_IN_RX_MODE(
                channel ) API429_CHANNEL_IN_RX_NORMAL_MODE(channel) | API429_CHANNEL_IN_RX_LP_MODE(channel)
```

Checks if a channel is in any mode that is able to receive data

Definition at line 232 of file Api429Channel.h.

API429_CHANNEL_IN_RX_NORMAL_MODE

```
#define API429_CHANNEL_IN_RX_NORMAL_MODE(
                channel ) ((channel)->config_flags & API429_CHN_CFG_RX_LABEL_FLAG)
```

Checks if channel if configured for normal receive mode

Definition at line 148 of file Api429Channel.h.

API429_CHANNEL_IN_RX_TX_MIXING_MODE

```
#define API429_CHANNEL_IN_RX_TX_MIXING_MODE(
                channel ) ((channel)->config_flags & API429_CHN_CFG_RX_MIXING_FLAG)
```

Checks if channel is configured in RX/TX mixing mode

Definition at line 160 of file Api429Channel.h.

API429_CHANNEL_IN_TX_DYNTAG_MODE

```
#define API429_CHANNEL_IN_TX_DYNTAG_MODE(
                channel ) ((channel)->config_flags & API429_CHN_CFG_TX_DYNTAG_FRAMING_FLAG)
```

Checks if channel is in TX framing mode with dyntags enabled

Definition at line 196 of file Api429Channel.h.

API429_CHANNEL_IN_TX_FIFO_MODE

```
#define API429_CHANNEL_IN_TX_FIFO_MODE(
```

```
                channel ) ((channel)->config_flags & API429_CHN_CFG_TX_FIFO_FLAG)
```

Checks if channel is configured as TX FIFO

Definition at line 202 of file Api429Channel.h.

API429_CHANNEL_IN_TX_FRAMING_MODE

```
#define API429_CHANNEL_IN_TX_FRAMING_MODE(
                channel ) ((channel)->config_flags & API429_CHN_CFG_TX_FRAMING_FLAG)
```

Checks if channel is configured for TX framing

Definition at line 178 of file Api429Channel.h.

API429_CHANNEL_IN_TX_LP_MODE

```
#define API429_CHANNEL_IN_TX_LP_MODE(
                channel ) ((channel)->config_flags & API429_CHN_CFG_TX_POLLUTION_FLAG)
```

Checks if channel is configured as loop/pollution transmitter

Definition at line 184 of file Api429Channel.h.

API429_CHANNEL_IN_TX_MODE

```
#define API429_CHANNEL_IN_TX_MODE(
                channel )
```

Value:

```
API429_CHANNEL_IN_TX_FRAMING_MODE(channel) |
      API429_CHANNEL_IN_TX_LP_MODE(channel)  | \
    API429_CHANNEL_IN_PHYS_REPLAY_MODE(channel) |
      API429_CHANNEL_IN_TX_DYNTAG_MODE(channel) | \
    API429_CHANNEL_IN_TX_FIFO_MODE(channel) |
      API429_CHANNEL_IN_TX_RATE_CONTROLLED_MODE(channel)
```

Checks if a channel is in any mode that is able to transmit data

Definition at line 216 of file Api429Channel.h.

API429_CHANNEL_IN_TX_RATE_CONTROLLED_MODE

```
#define API429_CHANNEL_IN_TX_RATE_CONTROLLED_MODE(
                channel ) ((channel)->config_flags & API429_CHN_CFG_TX_RATE_CONTROLLED_FLAG)
```

Checks if channel is in rate controlled TX mode

Definition at line 209 of file Api429Channel.h.

API429_CHN_CAP_RX_FLAG

```
#define API429_CHN_CAP_RX_FLAG (1 << 0)
```

Flag that indicates a channel is able to receive data

Definition at line 252 of file Api429Channel.h.

API429_CHN_CAP_TX_FLAG

```
#define API429_CHN_CAP_TX_FLAG (1 << 1)
```

Flag that indicates a channel is able to transmit data

Definition at line 257 of file Api429Channel.h.

API429_CHN_CAP_VAR_AMP_FLAG

```
#define API429_CHN_CAP_VAR_AMP_FLAG (1 << 2)
```

Flag that indicates a channel is able to transmit with variable amplitude

Definition at line 262 of file Api429Channel.h.

API429_CHN_CFG_PARITY_ENABLED_FLAG

```
#define API429_CHN_CFG_PARITY_ENABLED_FLAG (1 << 11)
```

Flag that indicates a channel was configured with parity handling enabled

Definition at line 135 of file Api429Channel.h.

API429_CHN_CFG_PHYS_REPLAY_FLAG

```
#define API429_CHN_CFG_PHYS_REPLAY_FLAG (1 << 6)
```

Flag that indicates a channel was configured for Physical Replay

Definition at line 109 of file Api429Channel.h.

API429_CHN_CFG_RM_GLOBAL_FLAG

```
#define API429_CHN_CFG_RM_GLOBAL_FLAG (1 << 3)
```

Flag that indicates a channel was configured for global monitoring mode

Definition at line 94 of file Api429Channel.h.

API429_CHN_CFG_RM_LOCAL_FLAG

```
#define API429_CHN_CFG_RM_LOCAL_FLAG (1 << 2)
```

Flag that indicates a channel was configured for local monitoring mode

Definition at line 89 of file Api429Channel.h.

API429_CHN_CFG_RX_LABEL_FLAG

```
#define API429_CHN_CFG_RX_LABEL_FLAG (1 << 0)
```

Flag that indicates a channel was configured for normal receive mode

Definition at line 79 of file Api429Channel.h.

API429_CHN_CFG_RX_POLLUTION_FLAG

```
#define API429_CHN_CFG_RX_POLLUTION_FLAG (1 << 1)
```

Flag that indicates a channel was configured as Loop/Pollution receiver

Definition at line 84 of file Api429Channel.h.

API429_CHN_CFG_SDI_ENABLED_FLAG

```
#define API429_CHN_CFG_SDI_ENABLED_FLAG (1 << 10)
```

Flag that indicates a channel was configured with SDI sorting enabled

Definition at line 130 of file Api429Channel.h.

API429_CHN_CFG_TX_DYNTAG_FRAMING_FLAG

```
#define API429_CHN_CFG_TX_DYNTAG_FRAMING_FLAG (1 << 7)
```

Flag that indicates a channel was configured for dynamic framing

Definition at line 114 of file Api429Channel.h.

API429_CHN_CFG_TX_FIFO_FLAG

```
#define API429_CHN_CFG_TX_FIFO_FLAG (1 << 8)
```

Flag that indicates a channel was configured as transmitter FIFO

Definition at line 119 of file Api429Channel.h.

API429_CHN_CFG_TX_FRAMING_FLAG

```
#define API429_CHN_CFG_TX_FRAMING_FLAG (1 << 4)
```

Flag that indicates a channel was configured for framing transmit mode

Definition at line 99 of file Api429Channel.h.

API429_CHN_CFG_TX_POLLUTION_FLAG

```
#define API429_CHN_CFG_TX_POLLUTION_FLAG (1 << 5)
```

Flag that indicates a channel was configured as Loop/Pollution transmitter

Definition at line 104 of file Api429Channel.h.

API429_CHN_CFG_TX_RATE_CONTROLLED_FLAG

```
#define API429_CHN_CFG_TX_RATE_CONTROLLED_FLAG (1 << 9)
```

Flag that indicates a channel was configured for automatic (Rate-Oriented) framing

Definition at line 124 of file Api429Channel.h.

API429_CHN_CFG_UNCONFIGURED

```
#define API429_CHN_CFG_UNCONFIGURED 0
```

Value for 'config_flags' member of api429_channel_info that indicates a channel is not configured

Definition at line 74 of file Api429Channel.h.

3.3.3 Typedef Documentation

API429_CHANNEL_CALLBACK

```
API429_CHANNEL_CALLBACK
```

prototype of the function that should be used for the event callback

Parameters

module	handle to the device
channel	ID of the channel to initialize
type	type of the event. See api429_event_type

Parameters

info	additional information about the source of this event. See api429_intr_loglist_entry **Note:** This pointer is only valid during event callback processing. It must not be accessed after callback finished.

Definition at line 417 of file Api429Channel.h.

TY_API429_CHANNEL_INFO

`TY_API429_CHANNEL_INFO`

Convenience typedef for api429_channel_info

Definition at line 58 of file Api429Channel.h.

TY_API429_INTR_LOGLIST_ENTRY

`TY_API429_INTR_LOGLIST_ENTRY`

Convenience typedef for api429_intr_loglist_entry

Definition at line 382 of file Api429Channel.h.

TY_API429_LOGLIST_TYPE_ENTRY

`TY_API429_LOGLIST_TYPE_ENTRY`

Convenience typedef for api429_loglist_entry

Definition at line 351 of file Api429Channel.h.

TY_E_API429_SPEED

`TY_E_API429_SPEED`

Convenience typedef for api429_speed

Definition at line 38 of file Api429Channel.h.

3.3.4 Enumeration Type Documentation

api429_event_type

`enum api429_event_type`

Enumeration of all event sources

Enumerator

API429_EVENT_UNDEFINED	event is of no known type
API429_EVENT_TX_HALT	end of major frame reached
API429_EVENT_TX_SKIP	event raised by a TX skip instruction
API429_EVENT_TX_LABEL	event raised by a TX label transfer instruction
API429_EVENT_TX_INDEX	tx buffer interrupt index reached
API429_EVENT_RX_ANY_LABEL	rx label buffer event
API429_EVENT_RX_INDEX	rx buffer interrupt index reached
API429_EVENT_RX_ERROR	rx received an error
API429_EVENT_FUNC_BLOCK	event raised by a function block
API429_EVENT_RM_TRIGGER	rm trigger event
API429_EVENT_RM_BUFFER_FULL	rm buffer end reached
API429_EVENT_RM_BUFFER_HALF_FULL	rm buffer middle reached
API429_EVENT_REPLAY_HALF_BUFFER	replay buffer middle or end reached
API429_EVENT_REPLAY_STOP	replay stopped
API429_EVENT_TX_FIFO	event raised by tx fifo

Definition at line 389 of file Api429Channel.h.

api429_speed

`enum api429_speed`

Enumeration of all possible Arinc429 speeds

Enumerator

API429_LO_SPEED	low speed value
API429_HI_SPEED	high speed value

Definition at line 30 of file Api429Channel.h.

3.3.5 Function Documentation

Api429ChannelCallbackRegister()

```
AiReturn Api429ChannelCallbackRegister (
                AiUInt8 board_handle,
                AiUInt8 uc_Chn,
                AiUInt8 uc_Type,
                API429_CHANNEL_CALLBACK pf_IntFunc )
```

Registers an event notification handler.

This function is used to register an event notification handler for a specific channel.
If an event handler function that handles several channels is needed, it is necessary to call this function for all required channels, each with the same given handler function.

Parameters

in	*board_handle*	handle to board the channel belongs to
in	*uc_Chn*	ID of channel to install event handler for
in	*uc_Type*	Reserved
in	*pf_IntFunc*	function pointer to handler

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429ChannelCallbackUnregister()

```
AiReturn Api429ChannelCallbackUnregister (
                AiUInt8 board_handle,
                AiUInt8 uc_Chn,
                AiUInt8 uc_Type )
```

Unregisters an event notification handler.

This function will unregister an event notification handler that has previously been installed using Api429ChannelCallbackRegister.

Parameters

in	*board_handle*	handle to board the channel belongs to
in	*uc_Chn*	ID of channel to unregister notification from
in	*uc_Type*	Reserved

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429ChannelClear()

```
AiReturn Api429ChannelClear (
              AiUInt8 board_handle,
              AiUInt8 channel_id )
```

Clear configuration of a channel.

This function will release a specific channel.
The channel will be halted and all of its used resources freed.
After successful return of this function the channel is stopped and in an uninitialized state.
The channel must be initialized before it can be started again using either Api429RxInit or Api429TxInit.

Parameters

in	*board_handle*	handle of the module the channel belongs to
in	*channel_id*	ID of the channel to clear

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429ChannelHalt()

```
AiReturn Api429ChannelHalt (
              AiUInt8 board_handle,
              AiUInt8 channel_id )
```

Halt a channel.

This function will halt a specific channel. In will stop sending or receiving any data.

Parameters

in	*board_handle*	handle of the module the channel belongs to
in	*channel_id*	ID of the channel to be halted

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429ChannelInfoGet()

```
AiReturn Api429ChannelInfoGet (
                AiUInt8 board_handle,
                AiUInt8 channel_id,
                struct api429_channel_info * pxInfo )
```

Get information about a specific channel.

This function will return information about a specific channel of a specific device.
It contains the some dynamic properties as current configuration etc. and also some static capabilities the channel offers e.g. if it is able to transmit or receive data

Parameters

in	*board_handle*	module handle of the device the channel belongs to
in	*channel_id*	ID of channel to query for information
out	*pxInfo*	pointer to api429_channel_info where information will be stored

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429ChannelSpeedSet()

```
AiReturn Api429ChannelSpeedSet (
```

```
                AiUInt8 board_handle,
                AiUInt8 b_Chn,
                enum api429_speed speed )
```

Set channel speed.

This function can be used to set the transmit/receive speed of a specific channel to either Arinc 429 High Speed or Low Speed.
Channel must be initialized for transmission or receiving with either Api429TxInit or Api429RxInit before calling this function.

Parameters

in	*board_handle*	handle to board the channel belongs to
in	*b_Chn*	ID of channel to set speed for
in	*speed*	Speed to set. See api429_speed

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429ChannelStart()

```
AiReturn Api429ChannelStart (
                AiUInt8 board_handle,
                AiUInt8 channel_id )
```

Start a channel.

This function will start a specific channel, if it was initialized before using either Api429RxInit or Api429↩ TxInit.

Parameters

in	*board_handle*	handle of the module the channel belongs to
in	*channel_id*	ID of the channel to be started

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

3.4 Discrete Handling

Data Structures

- struct api429_discr_info
- struct api429_discrete_pps_config

Typedefs

- typedef struct api429_discr_info TY_API429_DISCR_INFO
- typedef struct api429_discrete_pps_config TY_API429_DISCRETE_PPS_CONFIG

Functions

- AiReturn Api429DiscretesRead (AiUInt8 board_handle, AiUInt32 *pul_Value)

  Read current value from dicrete channels.
- AiReturn Api429DiscretesWrite (AiUInt8 board_handle, AiUInt32 ul_Mask, AiUInt32 ul_Value)

  Write to discrete output channels.
- AiReturn Api429DiscretesConfigSet (AiUInt8 board_handle, AiUInt32 ul_DiscreteSetup)

  Configures discrete I/O channel.
- AiReturn Api429DiscretesConfigGet (AiUInt8 board_handle, AiUInt32 *pul_DiscreteSetup)

  Get current configuration of discrete I/O channels.
- AiReturn Api429DiscretesInfoGet (AiUInt8 board_handle, struct api429_discr_info *px_DiscrInfo)

  Get current discrete configuration.
- AiReturn Api429DiscretesPPSSet (AiUInt8 board_handle, struct api429_discrete_pps_config *px↩ _Pps)

  On AyX429 generate an PPS signal on a discrete channel.

3.4.1 Detailed Description

This module contains functionality related to discrete I/O channel handling of AIM Arinc 429 boards

3.4.2 Typedef Documentation

TY_API429_DISCR_INFO

TY_API429_DISCR_INFO

Convenience macro for struct api429_discr_info

Definition at line 42 of file Api429Discretes.h.

TY_API429_DISCRETE_PPS_CONFIG

TY_API429_DISCRETE_PPS_CONFIG

Convenience macro for struct api429_discrete_pps_config

Definition at line 58 of file Api429Discretes.h.

3.4.3 Function Documentation

Api429DiscretesConfigGet()

```
AiReturn Api429DiscretesConfigGet (
              AiUInt8 board_handle,
              AiUInt32 * pul_DiscreteSetup )
```

Get current configuration of discrete I/O channels.

Parameters

in	*board_handle*	handle to board to get discrete I/O configuration from
out	*pul_DiscreteSetup*	Each bit indicates if corresponding discrete I/O channel is programmed for Output or Input where 1 means Output and 0 means Input

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429DiscretesConfigSet()

```
AiReturn Api429DiscretesConfigSet (
              AiUInt8 board_handle,
              AiUInt32 ul_DiscreteSetup )
```

Configures discrete I/O channel.

This command is used to configure the discretes. APX429, ACX429-3U and APU429 boards provide 8 discretes where each of the discretes can be initialized as Input or Output.

Parameters

in	*board_handle*	handle to board to configure discrete channels on
in	*ul_DiscreteSetup*	Each of the discretes can be programmed to Input or Output where 1 means Output and 0 means Input

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429DiscretesInfoGet()

```
AiReturn Api429DiscretesInfoGet (
                AiUInt8 board_handle,
                struct api429_discr_info * px_DiscrInfo )
```

Get current discrete configuration.

This command is used to read the configuration of the discrete channels.

Parameters

in	*board_handle*	handle to board to get discrete configuration of
out	*px_DiscrInfo*	discrete configuration will be stored here

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429DiscretesPPSSet()

```
AiReturn Api429DiscretesPPSSet (
                AiUInt8 board_handle,
                struct api429_discrete_pps_config * px_Pps )
```

On AyX429 generate an PPS signal on a discrete channel.

This command is used to generate a PPS signal on a given discrete channel

Parameters

in	*board_handle*	handle to the device
in	*px_Pps*	setup information for the PPS signal

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429DiscretesRead()

```
AiReturn Api429DiscretesRead (
              AiUInt8 board_handle,
              AiUInt32 * pul_Value )
```

Read current value from dicrete channels.

This command is used to read the current values from the discrete I/O channels

Parameters

in	*board_handle*	handle to board to read discrete channels of
out	*pul_Value*	This bitfield shows the values of the corresponding discrete channels. BIT0 corresponds to the first channel, BIT7 to the eighth.

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429DiscretesWrite()

```
AiReturn Api429DiscretesWrite (
              AiUInt8 board_handle,
              AiUInt32 ul_Mask,
              AiUInt32 ul_Value )
```

Write to discrete output channels.

This command is used to write to the discrete outputs.

Parameters

in	*board_handle*	handle to board to write discrete channels on
in	*ul_Mask*	The mask is a bitfield used to define the discrete output channels that are modified. A logical “1” in a bit marks the discrete output to be modified.
in	*ul_Value*	This bitfield shows the values of the corresponding discrete channels. BIT0 corresponds to the first channel, BIT7 to the eighth.

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

3.5 Library Administration Functions

Typedefs

- typedef enum api429_pnp_event TY_E_API429_PNP_EVENT
- typedef void(∗ TY_429PNP_CALLBACK_FUNC_PTR) (AiUInt32 module_id, enum api429_pnp↩ _event event_type)

Enumerations

- enum api429_pnp_event {
  API429_DEVICE_CONNECTED = 0,
  API429_DEVICE_DISCONNECTED }

Functions

- const char ∗ Api429LibErrorDescGet (AiReturn ul_ErrorCode)

  Convert API error code to human readable description.
- AiInt Api429LibInit (void)

  Initialization function for library.
- AiReturn Api429LibExit (void)

  Clean-up function for library This function cleans up the application interface initialization done in Api429↩ Init and should be called in an application program after any other function has finished.
- AiReturn Api429LibDebugLevelSet (AiUInt32 ul_DllDbgLevel)

  Set library debug level.
- AiReturn Api429LibPnpCallbackSet (TY_429PNP_CALLBACK_FUNC_PTR PnPCallbackFunction)

  Register handler for Plug N' Play device events.
- AiReturn Api429LibServerConnect (const char ∗pszNetworkAdress, AiInt16 ∗w_ServerBoards)

  Establish connection to remote server.
- AiReturn Api429LibServerDisconnect (const char ∗pszNetworkAdress)

  Disconnect from server.

3.5.1 Detailed Description

This module contains functionality related to library handling

3.5.2 Typedef Documentation

TY_429PNP_CALLBACK_FUNC_PTR

`TY_429PNP_CALLBACK_FUNC_PTR`

Callback function that handles library Plug 'n' Play device events.

Parameters

`out`	*module_id*	ID of module that caused the event
`out`	*event_type*	Event that caused callback invocation. See api429_pnp_event

Definition at line 50 of file Api429Lib.h.

TY_E_API429_PNP_EVENT

`TY_E_API429_PNP_EVENT`

Convenience typedef for api429_pnp_event

Definition at line 41 of file Api429Lib.h.

3.5.3 Enumeration Type Documentation

api429_pnp_event

`enum api429_pnp_event`

Enumeration for all possible events related to Plug 'n' Play device events

Enumerator

API429_DEVICE_CONNECTED	Plug 'n' Play device connected.
API429_DEVICE_DISCONNECTED	Plug 'n' Play device disconnected.

Definition at line 31 of file Api429Lib.h.

3.5.4 Function Documentation

Api429LibDebugLevelSet()

```
AiReturn Api429LibDebugLevelSet (
              AiUInt32 ul_DllDbgLevel )
```

Set library debug level.

This function is for internal use (debugging purposes).

Parameters

in	*ul_DllDbgLevel*	level to set

Debug Levels:

Constant	Description
API429_DBG_NONE	Disable additional debug information (default)
API429_DBG_TRACE	No longer supported

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429LibErrorDescGet()

```
const char* Api429LibErrorDescGet (
              AiReturn ul_ErrorCode )
```

Convert API error code to human readable description.

Parameters

in	*ul_ErrorCode*	code to convert

Returns

- pointer to zero-terminated ASCII string that contains error description on success
- NULL on failure

Api429LibExit()

```
AiReturn Api429LibExit (
              void  )
```

Clean-up function for library This function cleans up the application interface initialization done in Api429↩ Init and should be called in an application program after any other function has finished.

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429LibInit()

```
AiInt Api429LibInit (
              void  )
```

Initialization function for library.

This function initializes the entire application interface and must be called first in an application program before any other function is applied.

Returns

- Number of local AIM Arinc 429 boards

Api429LibPnpCallbackSet()

```
AiReturn Api429LibPnpCallbackSet (
              TY_429PNP_CALLBACK_FUNC_PTR PnPCallbackFunction )
```

Register handler for Plug N' Play device events.

This function provides mechanism to notify PnP device changes. For example adding or removing an APU429 at runtime.

Parameters

in	*PnPCallbackFunction*	pointer to a handler function that shall be called when a device was added to or removed from the system. See TY_429PNP_CALLBACK_FUNC_PTR

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429LibServerConnect()

```
AiReturn Api429LibServerConnect (
              const char * pszNetworkAdress,
              AiInt16 * w_ServerBoards )
```

Establish connection to remote server.

Establishes a network connection to a specified server PC where the ANS software (AIM Network Server) or an Ethernet based Arinc 429 device (e.g. ANET429) is running.

Parameters

in	*pszNetworkAdress*	Name of the PC, where the ANS Server is running. Name of the PC, where the ANS Server (AIM Network Server) is running (e.g. “SW-PC-06” or “192.168.0.119”)
out	*w_ServerBoards*	Number of AIM boards installed on server PC

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429LibServerDisconnect()

```
AiReturn Api429LibServerDisconnect (
              const char * pszNetworkAdress )
```

Disconnect from server.

Disconnects a previously established network connection from a specified server PC where the ANS software (AIM Network Server) is running.

Parameters

in	*pszNetworkAdress*	Name of the PC, where the ANS Server (AIM Network Server) is running (e.g. “SW-PC-06” or “192.168.0.119”)

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

3.6 Data Replay

Data Structures

- struct api429_replay_status

Typedefs

- typedef struct api429_replay_status TY_API429_REPLAY_STATUS

Functions

- AiReturn Api429ReplayInit (AiUInt8 board_handle, AiUInt8 uc_Chn, AiUInt8 uc_ClrEntryBit, AiUInt8 uc_NoRepCnt, AiUInt8 uc_CycOpr, AiUInt8 uc_RepErrors, AiUInt8 uc_RepIntMode, AiUInt8 uc_AbsLongTTag, AiUInt16 uw_DayOfYear, AiInt32 l_Min, AiInt32 l_MSec, AiUInt32 ul_FileSize)

  Initialize Replay on a specific channel.
- AiReturn Api429ReplayStatusGet (AiUInt8 board_handle, AiUInt8 uc_Chn, struct api429_replay_status *px_RepStatus)

  Get Replay Status of a specific channel.
- AiReturn Api429ReplayDataWrite (AiUInt8 board_handle, AiUInt8 uc_Chn, struct api429_replay_status *px_RepStatus, void *pv_Buf, AiUInt32 *pul_BytesWritten)

  Write Replay data for a specific channel.

3.6.1 Detailed Description

This module contains functionality related to replaying received Arinc 429 data.

3.6.2 Typedef Documentation

TY_API429_REPLAY_STATUS

TY_API429_REPLAY_STATUS

Convenience typedef for api429_replay_status

Definition at line 50 of file Api429Replay.h.

3.6.3 Function Documentation

Api429ReplayDataWrite()

```
AiReturn Api429ReplayDataWrite (
                AiUInt8 board_handle,
                AiUInt8 uc_Chn,
                struct api429_replay_status * px_RepStatus,
                void * pv_Buf,
                AiUInt32 * pul_BytesWritten )
```

Write Replay data for a specific channel.

This function writes and copies replay data and allows an easy implementation of an ARINC replay task.

Parameters

in	*board_handle*	handle to board the channel belongs to
in	*uc_Chn*	ID of channel to write replay data for
in	*px_RepStatus*	Replay Status
in	*pv_Buf*	Pointer to application buffer area (replay data)
out	*pul_BytesWritten*	Amount of bytes written (may be NULL)

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429ReplayInit()

```
AiReturn Api429ReplayInit (
                AiUInt8 board_handle,
                AiUInt8 uc_Chn,
                AiUInt8 uc_ClrEntryBit,
                AiUInt8 uc_NoRepCnt,
                AiUInt8 uc_CycOpr,
                AiUInt8 uc_RepErrors,
                AiUInt8 uc_RepIntMode,
                AiUInt8 uc_AbsLongTTag,
                AiUInt16 uw_DayOfYear,
                AiInt32 l_Min,
```

```
                AiInt32 l_MSec,
                AiUInt32 ul_FileSize )
```

Initialize Replay on a specific channel.

This function is used to initialize the physical replay mode of the AIM board. This functionality is not supported on embedded boards (like AXE429)

Parameters

in	*board_handle*	handle to the board the channel belongs to
in	*uc_Chn*	ID of channel to initialize Replay on
in	*uc_ClrEntryBit*	Clear Entry Bit
in	*uc_NoRepCnt*	No Replay Count
in	*uc_CycOpr*	Cyclic Operation
in	*uc_RepErrors*	Replay Errors
in	*uc_RepIntMode*	Replay Interrupt Control
in	*uc_AbsLongTTag*	Absolute Long Time Tag Replay
in	*uw_DayOfYear*	Specifies the original start day of the recorded buffer. This is necessary because the time tag information in the recording file specifies only the hour of a day.
in	*l_Min*	Absolute minute offset. Only required if “uc_AbsLongTTag” is set to one. Specify the offset between the recorded replay time and the actual IRIG time of the system at which the replay process shall start. The value is specified as a two's complement integer value with a possible range of one year in minutes (±525600, if no leap year)
in	*l_MSec*	Absolute microsecond offset. Only required if “uc_AbsLongTTag” is set to one. Specify the offset between the recorded replay time and the actual IRIG time of the system at which the replay process shall start. The value is specified as a two's complement integer value with a possible range of one minute in microseconds (±59 999 999).
in	*ul_FileSize*	Replay file size in bytes

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429ReplayStatusGet()

```
AiReturn Api429ReplayStatusGet (
                AiUInt8 board_handle,
                AiUInt8 uc_Chn,
                struct api429_replay_status * px_RepStatus )
```

Get Replay Status of a specific channel.

This function is used to read the replay status information. The initial value of parameter 'ul_RpiCnt' indicates the amount of half buffer transmitted interrupts required for writing the requested replay file size (refer to parameter 'ul_FileSize' of library function Api429ReplayInit).

Parameters

`in`	*board_handle*	handle to board the channel belongs to
`in`	*uc_Chn*	ID of channel to get replay status of
`out`	*px_RepStatus*	Replay status structure

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

3.7 Data Monitoring

Data Structures

- struct api429_rm_setup
- struct api429_rm_trigger_setup
- struct api429_rm_label_config
- struct api429_rm_function_block
- struct api429_rm_activity_trigger_def
- struct api429_trg_ctl_cmd
- union api429_tm_tag
- union api429_brw
- struct api429_rcv_stack_entry
- struct api429_rcv_stack_entry_ex

Macros

- #define API429_RM_MON_DEFAULT_SIZE 4096
- #define API429_RM_CONTINUOUS_CAPTURE 0
- #define API429_RM_MON_DEFAULT_SIZE 4096
- #define API429_RM_CONTINUOUS_CAPTURE 0

Typedefs

- typedef enum api429_rm_mode TY_E_API429_RM_MODE
- typedef enum api429_rm_interrupt_mode TY_E_API429_RM_INTERRUPT_MODE
- typedef enum api429_rm_trigger_mode TY_E_API429_RM_TRIGGER_MODE
- typedef struct api429_rm_setup TY_API429_RM_SETUP
- typedef struct api429_rm_trigger_setup TY_API429_RM_TRIGGER_SETUP
- typedef struct api429_rm_label_config TY_API429_RM_LABEL_CONFIG
- typedef struct api429_rm_function_block TY_API429_RM_FUNCTION_BLOCK
- typedef struct api429_rm_activity_trigger_def TY_API429_RM_ACTIVITY_TRIGGER_DEF
- typedef struct api429_trg_ctl_cmd TY_API429_TRG_CTL_CMD
- typedef struct api429_rcv_stack_entry TY_API429_RCV_STACK_ENTRY
- typedef struct api429_rcv_stack_entry_ex TY_API429_RCV_STACK_ENTRY_EX
- typedef enum api429_rm_mode **TY_E_API429_RM_MODE**
- typedef enum api429_rm_interrupt_mode **TY_E_API429_RM_INTERRUPT_MODE**
- typedef enum api429_rm_trigger_mode **TY_E_API429_RM_TRIGGER_MODE**
- typedef struct api429_rm_setup **TY_API429_RM_SETUP**
- typedef struct api429_rm_trigger_setup **TY_API429_RM_TRIGGER_SETUP**
- typedef struct api429_rm_label_config **TY_API429_RM_LABEL_CONFIG**
- typedef struct api429_rm_function_block **TY_API429_RM_FUNCTION_BLOCK**
- typedef struct api429_rm_activity_trigger_def **TY_API429_RM_ACTIVITY_TRIGGER_DEF**
- typedef struct api429_trg_ctl_cmd **TY_API429_TRG_CTL_CMD**
- typedef struct api429_rcv_stack_entry **TY_API429_RCV_STACK_ENTRY**
- typedef struct api429_rcv_stack_entry_ex **TY_API429_RCV_STACK_ENTRY_EX**

Enumerations

- enum api429_rm_mode {
  API429_RM_MODE_UNDEFINED = -1,
  API429_RM_MODE_LOC = 0,
  API429_RM_MODE_GLB = 1,
  API429_RM_MODE_UNDEFINED = -1,
  API429_RM_MODE_LOC = 0,
  API429_RM_MODE_GLB = 1 }
- enum api429_rm_interrupt_mode {
  API429_RM_IR_UNDEFINED = -1,
  API429_RM_IR_DIS = 0,
  API429_RM_IR_START,
  API429_RM_IR_STOP,
  API429_RM_IR_BFI,
  API429_RM_IR_HFI,
  API429_RM_IR_START_HFI_BFI,
  API429_RM_IR_UNDEFINED = -1,
  API429_RM_IR_DIS = 0,
  API429_RM_IR_START,
  API429_RM_IR_STOP,
  API429_RM_IR_BFI,
  API429_RM_IR_HFI,
  API429_RM_IR_START_HFI_BFI }
- enum api429_rm_trigger_mode {
  API429_TRG_START = 0,
  API429_TRG_ERR,
  API429_TRG_EXT,
  API429_TRG_ANY,
  API429_TRG_START = 0,
  API429_TRG_ERR,
  API429_TRG_EXT,
  API429_TRG_ANY }
- enum api429_rm_mode {
  API429_RM_MODE_UNDEFINED = -1,
  API429_RM_MODE_LOC = 0,
  API429_RM_MODE_GLB = 1,
  API429_RM_MODE_UNDEFINED = -1,
  API429_RM_MODE_LOC = 0,
  API429_RM_MODE_GLB = 1 }
- enum api429_rm_interrupt_mode {
  API429_RM_IR_UNDEFINED = -1,
  API429_RM_IR_DIS = 0,
  API429_RM_IR_START,
  API429_RM_IR_STOP,
  API429_RM_IR_BFI,
  API429_RM_IR_HFI,
  API429_RM_IR_START_HFI_BFI,
  API429_RM_IR_UNDEFINED = -1,

API429_RM_IR_DIS = 0,
API429_RM_IR_START,
API429_RM_IR_STOP,
API429_RM_IR_BFI,
API429_RM_IR_HFI,
API429_RM_IR_START_HFI_BFI }

- enum api429_rm_trigger_mode {
API429_TRG_START = 0,
API429_TRG_ERR,
API429_TRG_EXT,
API429_TRG_ANY,
API429_TRG_START = 0,
API429_TRG_ERR,
API429_TRG_EXT,
API429_TRG_ANY }

Functions

- AiReturn Api429RmCreate (AiUInt8 board_handle, AiUInt8 channel_id, const struct api429_rm_setup *setup)

  Initializes receive monitoring for a channel.
- AiReturn Api429RmInfoGet (AiUInt8 board_handle, AiUInt8 channel_id, struct api429_rm_setup *monitor_setup)

  Get information about a receive monitor instance on a specific Arinc4 429 receive channel.
- AiReturn Api429RmTriggerConfigSet (AiUInt8 board_handle, AiUInt8 channel, const struct api429_rm_trigger_setup *trigger_setup)

  Defines general trigger set-up for receive monitor.
- AiReturn Api429RmLabelConfigure (AiUInt8 board_handle, AiUInt8 b_Chn, const struct api429_rm_label_config *label_config)

  Configures monitoring of Labels.
- AiReturn Api429RmMultiLabelConfigure (AiUInt8 board_handle, AiUInt8 b_Chn, AiUInt32 ulCount, struct api429_rm_label_config *pxSetup)

  Configures monitoring of multiple Labels.
- AiReturn Api429RmFuncBlockConfigure (AiUInt8 board_handle, AiUInt8 b_Chn, AiUInt8 b_Label, AiUInt8 b_Sdi, AiUInt8 b_Con, struct api429_rm_function_block *px_FuncBlk)

  Configures a monitor function block.
- AiReturn Api429RmTriggerPatternSet (AiUInt8 board_handle, AiUInt8 b_Chn, struct api429_rm_activity_trigger_def *px_Para)

  This function is utilized to set the start/stop trigger pattern.
- AiReturn Api429RmStartTriggerSet (AiUInt8 board_handle, AiUInt8 b_Chn, AiUInt8 b_Label, AiUInt8 b_Sdi, AiUInt8 b_Con, struct api429_trg_ctl_cmd *px_Ctl)

  Set a monitor start trigger.
- AiReturn Api429RmStopTriggerSet (AiUInt8 board_handle, AiUInt8 b_Chn, AiUInt8 b_Label, AiUInt8 b_Sdi, AiUInt8 b_Con, struct api429_trg_ctl_cmd *px_Ctl)

  Set a monitor stop trigger.

- AiReturn Api429RmResume (AiUInt8 board_handle, AiUInt8 channel)

  Resume monitoring on a channel.
- AiReturn Api429RmSuspend (AiUInt8 board_handle, AiUInt8 channel)

  Suspend monitoring on a channel.
- AiReturn Api429RmStatusGet (AiUInt8 board_handle, AiUInt8 channel_id, AiUInt8 *monitoring_activity, AiUInt16 *status)

  Get a channel's current data monitoring status.
- AiReturn Api429RmStackPointersGet (AiUInt8 board_handle, AiUInt8 b_Chn, AiUInt32 *pl_Stp, AiUInt32 *pl_Ctp, AiUInt32 *pl_Etp)

  Get current position of monitor stack pointers.
- AiReturn Api429RmDataRead (AiUInt8 board_handle, AiUInt8 b_Chn, AiUInt16 uw_EntriesToRead, AiUInt16 *puw_Count, struct api429_rcv_stack_entry *px_SData)

  Read Monitor data.
- AiReturn Api429RmDataReadWithDayOfYear (AiUInt8 board_handle, AiUInt8 b_Chn, AiUInt16 uw_EntriesToRead, AiUInt16 *puw_Count, struct api429_rcv_stack_entry_ex *px_SData)

  Read monitor data and add the day of year.

3.7.1 Detailed Description

This module contains functionality related to monitoring of received Arinc 429 bus traffic.

3.7.2 Macro Definition Documentation

API429_RM_CONTINUOUS_CAPTURE [1/2]

```
#define API429_RM_CONTINUOUS_CAPTURE 0
```

Can be used as parameter 'tat_count' for Api429RmCreate for a continuous monitoring of data without stopping the monitor after a specific number of entries has been received.

Definition at line 40 of file Api429Rm.h.

API429_RM_CONTINUOUS_CAPTURE [2/2]

```
#define API429_RM_CONTINUOUS_CAPTURE 0
```

Can be used as parameter 'tat_count' for Api429RmCreate for a continuous monitoring of data without stopping the monitor after a specific number of entries has been received.

Definition at line 40 of file Api429Rm - LV.h.

API429_RM_MON_DEFAULT_SIZE [1/2]

```
#define API429_RM_MON_DEFAULT_SIZE 4096
```

Can be used with Api429RmCreate to create a monitor buffer with default size

Definition at line 32 of file Api429Rm - LV.h.

API429_RM_MON_DEFAULT_SIZE [2/2]

```
#define API429_RM_MON_DEFAULT_SIZE 4096
```

Can be used with Api429RmCreate to create a monitor buffer with default size

Definition at line 32 of file Api429Rm.h.

3.7.3 Typedef Documentation

TY_API429_RCV_STACK_ENTRY

```
TY_API429_RCV_STACK_ENTRY
```

Convenience typedef for api429_rcv_stack_entry

Definition at line 335 of file Api429Rm.h.

TY_API429_RCV_STACK_ENTRY_EX

```
TY_API429_RCV_STACK_ENTRY_EX
```

Convenience typedef for api429_rcv_stack_entry_ex

Definition at line 356 of file Api429Rm.h.

TY_API429_RM_ACTIVITY_TRIGGER_DEF

TY_API429_RM_ACTIVITY_TRIGGER_DEF

Convenience typedef for api429_rm_activity_trigger_def

Definition at line 215 of file Api429Rm.h.

TY_API429_RM_FUNCTION_BLOCK

TY_API429_RM_FUNCTION_BLOCK

Convenience typedef for api429_rm_function_block

Definition at line 198 of file Api429Rm.h.

TY_API429_RM_LABEL_CONFIG

TY_API429_RM_LABEL_CONFIG

Convenience typedef for api429_rm_label_config

Definition at line 161 of file Api429Rm.h.

TY_API429_RM_SETUP

TY_API429_RM_SETUP

Convenience typedef for api429_rm_setup

Definition at line 121 of file Api429Rm.h.

TY_API429_RM_TRIGGER_SETUP

TY_API429_RM_TRIGGER_SETUP

Convenience typedef for api429_rm_trigger_setup

Definition at line 143 of file Api429Rm.h.

TY_API429_TRG_CTL_CMD

TY_API429_TRG_CTL_CMD

Convenience typedef for api429_trg_ctl_cmd

Definition at line 234 of file Api429Rm.h.

TY_E_API429_RM_INTERRUPT_MODE

TY_E_API429_RM_INTERRUPT_MODE

Convenience typedef for api429_rm_interrupt_mode

Definition at line 78 of file Api429Rm.h.

TY_E_API429_RM_MODE

TY_E_API429_RM_MODE

Convenience typedef for api429_rm_mode

Definition at line 58 of file Api429Rm.h.

TY_E_API429_RM_TRIGGER_MODE

TY_E_API429_RM_TRIGGER_MODE

Convenience typedef for api429_rm_trigger_mode

Definition at line 96 of file Api429Rm.h.

3.7.4 Enumeration Type Documentation

api429_rm_interrupt_mode [1/2]

enum api429_rm_interrupt_mode

Enumeration of all possible monitor interrupt modes

Enumerator

API429_RM_IR_UNDEFINED	Undefined
API429_RM_IR_DIS	No interrupts are issued
API429_RM_IR_START	Interrupt is issued when monitor starts due to trigger
API429_RM_IR_STOP	Interrupt is issued when monitor stops
API429_RM_IR_BFI	Interrupt is issued when monitor buffer is full
API429_RM_IR_HFI	Interrupt is issued when monitor buffer is half full
API429_RM_IR_START_HFI_BFI	Interrupt is issued when monitor starts, buffer is half full or full
API429_RM_IR_UNDEFINED	Undefined
API429_RM_IR_DIS	No interrupts are issued
API429_RM_IR_START	Interrupt is issued when monitor starts due to trigger
API429_RM_IR_STOP	Interrupt is issued when monitor stops
API429_RM_IR_BFI	Interrupt is issued when monitor buffer is full
API429_RM_IR_HFI	Interrupt is issued when monitor buffer is half full
API429_RM_IR_START_HFI_BFI	Interrupt is issued when monitor starts, buffer is half full or full

Definition at line 64 of file Api429Rm.h.

api429_rm_interrupt_mode [2/2]

enum api429_rm_interrupt_mode

Enumerator

API429_RM_IR_UNDEFINED	Undefined
API429_RM_IR_DIS	No interrupts are issued
API429_RM_IR_START	Interrupt is issued when monitor starts due to trigger
API429_RM_IR_STOP	Interrupt is issued when monitor stops
API429_RM_IR_BFI	Interrupt is issued when monitor buffer is full
API429_RM_IR_HFI	Interrupt is issued when monitor buffer is half full
API429_RM_IR_START_HFI_BFI	Interrupt is issued when monitor starts, buffer is half full or full
API429_RM_IR_UNDEFINED	Undefined
API429_RM_IR_DIS	No interrupts are issued
API429_RM_IR_START	Interrupt is issued when monitor starts due to trigger
API429_RM_IR_STOP	Interrupt is issued when monitor stops

Enumerator

API429_RM_IR_BFI	Interrupt is issued when monitor buffer is full
API429_RM_IR_HFI	Interrupt is issued when monitor buffer is half full
API429_RM_IR_START_HFI_BFI	Interrupt is issued when monitor starts, buffer is half full or full

Definition at line 64 of file Api429Rm - LV.h.

api429_rm_mode [1/2]

`enum api429_rm_mode`

Enumeration of all supported monitoring modes

Enumerator

API429_RM_MODE_UNDEFINED	API429_RM_MODE_UNDEFINED.
API429_RM_MODE_LOC	Can be used for channel specific monitoring. Channel uses its own monitor buffer.
API429_RM_MODE_GLB	Can be used for multiplexing received data of several channels into one monitor buffer.
API429_RM_MODE_UNDEFINED	API429_RM_MODE_UNDEFINED.
API429_RM_MODE_LOC	Can be used for channel specific monitoring. Channel uses its own monitor buffer.
API429_RM_MODE_GLB	Can be used for multiplexing received data of several channels into one monitor buffer.

Definition at line 47 of file Api429Rm.h.

api429_rm_mode [2/2]

`enum api429_rm_mode`

Enumerator

API429_RM_MODE_UNDEFINED	API429_RM_MODE_UNDEFINED.
API429_RM_MODE_LOC	Can be used for channel specific monitoring. Channel uses its own monitor buffer.
API429_RM_MODE_GLB	Can be used for multiplexing received data of several channels into one monitor buffer.
API429_RM_MODE_UNDEFINED	API429_RM_MODE_UNDEFINED.

Enumerator

API429_RM_MODE_LOC	Can be used for channel specific monitoring. Channel uses its own monitor buffer.
API429_RM_MODE_GLB	Can be used for multiplexing received data of several channels into one monitor buffer.

Definition at line 47 of file Api429Rm - LV.h.

api429_rm_trigger_mode [1/2]

`enum api429_rm_trigger_mode`

Enumeration of all possible trigger modes for a receive monitor

Enumerator

API429_TRG_START	trigger on start and stop function trigger
API429_TRG_ERR	trigger on any error
API429_TRG_EXT	trigger on event on external input trigger line
API429_TRG_ANY	trigger on first enabled label received on channel
API429_TRG_START	trigger on start and stop function trigger
API429_TRG_ERR	trigger on any error
API429_TRG_EXT	trigger on event on external input trigger line
API429_TRG_ANY	trigger on first enabled label received on channel

Definition at line 85 of file Api429Rm.h.

api429_rm_trigger_mode [2/2]

`enum api429_rm_trigger_mode`

Enumerator

API429_TRG_START	trigger on start and stop function trigger
API429_TRG_ERR	trigger on any error
API429_TRG_EXT	trigger on event on external input trigger line
API429_TRG_ANY	trigger on first enabled label received on channel
API429_TRG_START	trigger on start and stop function trigger
API429_TRG_ERR	trigger on any error

Enumerator

API429_TRG_EXT	trigger on event on external input trigger line
API429_TRG_ANY	trigger on first enabled label received on channel

Definition at line 85 of file Api429Rm - LV.h.

3.7.5 Function Documentation

Api429RmCreate()

```
AiReturn Api429RmCreate (
                AiUInt8 board_handle,
                AiUInt8 channel_id,
                const struct api429_rm_setup * setup )
```

Initializes receive monitoring for a channel.

This function can be used to enable receive monitoring for a channel.
Data received on a channel can be monitored to a channel specific buffer which is called 'local' monitoring with API429_RM_MODE_LOC

The function also supports multiplexing of data streams received on several channels into a single buffer which is called 'global' monitoring.
The last call to this function with API429_RM_MODE_GLB will determine the properties of the monitoring buffer
used for global monitoring. The settings of all previous calls with API429_RM_MODE_GLB will be overwritten.
Calling this function will overwrite all settings of a previously created monitor for the channel, hence it will fail when channel is currently active.

Parameters

in	*board_handle*	the handle of the module the channel belongs to
in	*channel_id*	ID of channel to initialize receive monitor on
in	*setup*	Monitor setup properties. See api429_rm_setup

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429RmDataRead()

```
AiReturn Api429RmDataRead (
                AiUInt8 board_handle,
                AiUInt8 b_Chn,
                AiUInt16 uw_EntriesToRead,
                AiUInt16 * puw_Count,
                struct api429_rcv_stack_entry * px_SData )
```

Read Monitor data.

This function is utilized to read the new stack entries from the monitor stack of the selected Arinc 429 receiver channel since the last call of the function. If more entries than specified in uw_EntriesToRead are on the stack, the oldest entries will be read. A stack entry consists of three 32-bit words.

Parameters

in	*board_handle*	handle to board the channel belongs to
in	*b_Chn*	ID of monitor channel
in	*uw_EntriesToRead*	Maximum number of entries to be read.
out	*puw_Count*	The amount of entries really read. If ∗puw_count is as high as uw_EntriesToRead, there might be some entries left in the monitor stack. In this case it's recommended to call this function again to read the rest of data.
out	*px_SData*	Array of pointer to Stack Entry Data structure

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429RmDataReadWithDayOfYear()

```
AiReturn Api429RmDataReadWithDayOfYear (
                AiUInt8 board_handle,
                AiUInt8 b_Chn,
                AiUInt16 uw_EntriesToRead,
                AiUInt16 * puw_Count,
                struct api429_rcv_stack_entry_ex * px_SData )
```

Read monitor data and add the day of year.

This function is utilized to read the new stack entries from the monitor stack of the selected Arinc 429 receiver channel since the last call of the function. If more entries than specified in uw_EntriesToRead are on the stack, the oldest entries will be read. This function internally calls Api429RmDataRead to get

the three 32-bit words that are part of the receiver monitor buffer entry and adds the day of year that is read from the IRIG source.
This command may show incorrect data if called less than once per 24 hours.
The day of year information is added when the data is read from the receiver monitor buffer and not when the data is received.
Changing the onboard time might cause the next call of this function to fail
It is better to use Api429RmDataRead instead.

Parameters

in	*board_handle*	handle to board the channel belongs to
in	*b_Chn*	ID of monitor channel
in	*uw_EntriesToRead*	Maximum number of entries to be read.
out	*puw_Count*	The amount of entries really read. If ∗puw_count is as high as uw_EntriesToRead, there might be some entries left in the monitor stack. In this case it's recommended to call this function again to read the rest of data.
out	*px_SData*	See api429_rcv_stack_entry_ex

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429RmFuncBlockConfigure()

```
AiReturn Api429RmFuncBlockConfigure (
                AiUInt8 board_handle,
                AiUInt8 b_Chn,
                AiUInt8 b_Label,
                AiUInt8 b_Sdi,
                AiUInt8 b_Con,
                struct api429_rm_function_block * px_FuncBlk )
```

Configures a monitor function block.

Parameters

in	*board_handle*	handle to board the monitor channel belongs to
in	*b_Chn*	ID of monitoring channel
in	*b_Label*	ID of label
in	*b_Sdi*	SDI extension of label
in	*b_Con*	enable/disable function block
in	*px_FuncBlk*	pointer to function block definition

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429RmInfoGet()

```
AiReturn Api429RmInfoGet (
                AiUInt8 board_handle,
                AiUInt8 channel_id,
                struct api429_rm_setup * monitor_setup )
```

Get information about a receive monitor instance on a specific Arinc4 429 receive channel.

Will retrieve information about the general monitor set-up on a specific channel.
The channel must be configured for receiving and monitoring before calling this function, otherwise an error will be returned.

Parameters

in	*board_handle*	the handle of the module the channel belongs to
in	*channel_id*	ID of the channel to get monitor info of
out	*monitor_setup*	pointer to api429_rm_setup where info will be stored

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429RmLabelConfigure()

```
AiReturn Api429RmLabelConfigure (
                AiUInt8 board_handle,
                AiUInt8 b_Chn,
                const struct api429_rm_label_config * label_config )
```

Configures monitoring of Labels.

This function can be used to either enable or disable specific label/SDI combinations for monitoring.

Parameters

in	*board_handle*	handle to board the receive channel belongs to

Parameters

in	*b_Chn*	receive channel that is enabled for monitoring
in	*label_config*	See api429_rm_label_config

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429RmMultiLabelConfigure()

```
AiReturn Api429RmMultiLabelConfigure (
                AiUInt8 board_handle,
                AiUInt8 b_Chn,
                AiUInt32 ulCount,
                struct api429_rm_label_config * pxSetup )
```

Configures monitoring of multiple Labels.

This function can be used to either enable or disable multiple label/SDI combinations for monitoring.

Parameters

in	*board_handle*	handle to board the receive channel belongs to
in	*b_Chn*	receive channel that is enabled for monitoring
in	*ulCount*	number of labels to configure
in	*pxSetup*	pointer to array of api429_rm_label_config instances. Must hold at least ulCount entries.

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429RmResume()

```
AiReturn Api429RmResume (
                AiUInt8 board_handle,
                AiUInt8 channel )
```

Resume monitoring on a channel.

This function will resume monitoring on a channel that has been previously suspended using Api429↩ RmSuspend.
Will return an error when monitoring is not initialized for the given channel.
When calling this function on an active monitoring, the function will have not effect and return successful execution.

Parameters

in	*board_handle*	handle to the module the channel belongs to
in	*channel*	ID of the channel to resume monitoring on

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429RmStackPointersGet()

```
AiReturn Api429RmStackPointersGet (
                AiUInt8 board_handle,
                AiUInt8 b_Chn,
                AiUInt32 * pl_Stp,
                AiUInt32 * pl_Ctp,
                AiUInt32 * pl_Etp )
```

Get current position of monitor stack pointers.

This function is utilized to read the monitor stack pointers on the selected Arinc 429 receiver channel. Using these pointers the monitored data can be read directly from the board. The start of the receiver monitor buffer can be found with Api429BoardMemLocationGet Please have a look into the Programmer's Guide for details. For reading the monitor data, please consider using Api429RmDataRead

Parameters

in	*board_handle*	handle to board the channel belongs to
in	*b_Chn*	ID of monitor channel
out	*pl_Stp*	Capture Start Pointer. Shows the offset of the first received message, relative to the start global memory. Only valid after monitoring has started.
out	*pl_Ctp*	Trigger Pointer. Shows the offset of the received message that triggered the monitoring, relative to the start global memory. Only valid after monitoring has started.
out	*pl_Etp*	Capture End Pointer. Shows where the firmware will write the next monitoring entry to (Monitor Buffer Fill Pointer).

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429RmStartTriggerSet()

```
AiReturn Api429RmStartTriggerSet (
                AiUInt8 board_handle,
                AiUInt8 b_Chn,
                AiUInt8 b_Label,
                AiUInt8 b_Sdi,
                AiUInt8 b_Con,
                struct api429_trg_ctl_cmd * px_Ctl )
```

Set a monitor start trigger.

Parameters

in	*board_handle*	handle to boar the monitor channel belongs to
in	*b_Chn*	ID of monitor channel
in	*b_Label*	ID of label to set trigger for
in	*b_Sdi*	SDI extension for label
in	*b_Con*	enable/disable start trigger
in	*px_Ctl*	See api429_trg_ctl_cmd

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429RmStatusGet()

```
AiReturn Api429RmStatusGet (
                AiUInt8 board_handle,
                AiUInt8 channel_id,
                AiUInt8 * monitoring_activity,
                AiUInt16 * status )
```

Get a channel's current data monitoring status.

This function is utilized to retrieve information about the data monitoring status on the specified Arinc 429 receiver channel.
Several monitor properties are encoded in the *status* output parameter of this function:

status bit	Description
ERT (bit 5)	error trigger detected
CT (bit 6)	complex function trigger detected
MS (bit 7)	monitor has stopped
MT (bit 8)	monitor has triggered
MODE (bit 10..12)	0: no monitoring 1: local monitoring 2: global monitoring

Parameters

`in`	*board_handle*	handle to board the channel belongs to
`in`	*channel_id*	ID of channel
`out`	*monitoring_activity*	General state of data monitoring on the specified channel. • API429_HALT if channel was not started or monitoring has been suspended • API429_BUSY if data monitoring is active.
`out`	*status*	Detailed information about data monitoring status of the specified channel.

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429RmStopTriggerSet()

```
AiReturn Api429RmStopTriggerSet (
                AiUInt8 board_handle,
                AiUInt8 b_Chn,
                AiUInt8 b_Label,
                AiUInt8 b_Sdi,
                AiUInt8 b_Con,
                struct api429_trg_ctl_cmd * px_Ctl )
```

Set a monitor stop trigger.

This function defines the monitor stop trigger condition.

Parameters

in	*board_handle*	handle to boar the monitor channel belongs to
in	*b_Chn*	ID of monitor channel
in	*b_Label*	ID of label to set trigger for
in	*b_Sdi*	SDI extension for label
in	*b_Con*	enable/disable stop trigger
in	*px_Ctl*	See api429_trg_ctl_cmd

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429RmSuspend()

```
AiReturn Api429RmSuspend (
                AiUInt8 board_handle,
                AiUInt8 channel )
```

Suspend monitoring on a channel.

This function will suspend monitoring on a channel. No more receive data will be monitored after calling this function.
Monitoring can be resumed afterwards by calling Api429RmResume.
When calling this function before activating the channel using Api429ChannelStart, the monitor will be in a suspended state.
Will return an error when monitoring is not initialized for the given channel.
When calling this function on an already suspended monitoring, the function will have not effect and return successful execution.

Parameters

in	*board_handle*	handle to the module the channel belongs to
in	*channel*	ID of the channel to suspend monitoring on

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429RmTriggerConfigSet()

```
AiReturn Api429RmTriggerConfigSet (
                AiUInt8 board_handle,
                AiUInt8 channel,
                const struct api429_rm_trigger_setup * trigger_setup )
```

Defines general trigger set-up for receive monitor.

This function is used to define general trigger set-up of a monitor on a specific receive channel.
It is used to configure internal, external as well as software defined monitor start and stop triggers.
For setting up software defined trigger conditions, additional configuration is necessary using the functions Api429RmTriggerPatternSet, Api429RmStartTriggerSet or Api429RmStopTriggerSet.

Parameters

in	*board_handle*	handle of the module the receive channel belongs to
in	*channel*	ID of receive channel to set-up monitor triggers for
in	*trigger_setup*	pointer to trigger set-up structure

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429RmTriggerPatternSet()

```
AiReturn Api429RmTriggerPatternSet (
                AiUInt8 board_handle,
                AiUInt8 b_Chn,
                struct api429_rm_activity_trigger_def * px_Para )
```

This function is utilized to set the start/stop trigger pattern.

Parameters

in	*board_handle*	
in	*b_Chn*	
in	*px_Para*	

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

3.8 Data Receiver

Data Structures

- struct api429_rx_frame_response_setup
- struct api429_rx_activity
- struct api429_rcv_pb_cmd
- struct api429_rx_label_setup
- struct api429_rx_buf_entry
- struct api429_rx_buf_ctl

Typedefs

- typedef struct api429_rx_frame_response_setup TY_API429_RX_FRAME_RESPONSE_SETUP
- typedef struct api429_rx_activity TY_API429_RX_ACTIVITY
- typedef struct api429_rcv_pb_cmd TY_API429_RCV_PB_CMD
- typedef struct api429_rx_label_setup TY_API429_RX_LABEL_SETUP
- typedef struct api429_rx_buf_entry TY_API429_RX_BUF_ENTRY
- typedef struct api429_rx_buf_ctl TY_API429_RX_BUF_CTL

Functions

- AiReturn Api429RxInit (AiUInt8 board_handle, AiUInt8 b_Chn, AiBoolean sdi_enabled, AiBoolean parity_check_enabled)

  Basic initialization of a channel for receiving data.
- AiReturn Api429RxDataLoopAssign (AiUInt8 board_handle, AiUInt8 channel, AiUInt8 transmitter_channel)

  Set up a data forwarding for a receive channel.
- AiReturn Api429RxFrameResponseAssign (AiUInt8 board_handle, AiUInt8 receive_channel, AiUInt8 label, AiUInt8 sdi, struct api429_rx_frame_response_setup *p_Setup)

  Assign an automatic frame response on reception of a label.
- AiReturn Api429RxFrameResponseRelease (AiUInt8 board_handle, AiUInt8 receive_channel, AiUInt8 label, AiUInt8 sdi)

  Release an automatic frame response on reception of a label.
- AiReturn Api429RxActivityGet (AiUInt8 board_handle, struct api429_rx_activity *pxActivity)

  Get overview of received label activity.
- AiReturn Api429RxPollutionConfigure (AiUInt8 board_handle, AiUInt8 b_Chn, AiUInt8 b_Label, AiUInt8 b_Sdi, AiUInt8 b_Con, struct api429_rcv_pb_cmd *px_PollBlk)

  Used to define data modifications in a receiver/transmitter data loop.
- AiReturn Api429RxLabelConfigure (AiUInt8 board_handle, AiUInt8 b_Chn, struct api429_rx_label_setup *label_setup, AiUInt8 *puc_Status, AiUInt32 *pul_FreeMem)

  Configure reception of specific labels.
- AiReturn Api429RxMultiLabelConfigure (AiUInt8 board_handle, AiUInt8 b_Chn, AiUInt32 ulCount, struct api429_rx_label_setup *pxSetup)

Configure reception of multiple labels.

- AiReturn Api429RxLabelBufferWrite (AiUInt8 board_handle, AiUInt8 b_Chn, AiUInt8 b_Label, AiUInt8 b_Sdi, AiUInt16 w_BufSize, struct api429_rx_buf_entry *px_LData, AiUInt8 b_Clear)

  Writes data to a receive label buffer.
- AiReturn Api429RxStatusGet (AiUInt8 board_handle, AiUInt8 b_Chn, AiUInt8 *b_RxStatus, AiUInt32 *px_MsgCnt, AiUInt32 *px_ErrCnt)

  Get status of channel in receive mode.
- AiReturn Api429RxLabelStatusGet (AiUInt8 board_handle, AiUInt8 b_Chn, AiUInt8 b_Label, AiUInt8 b_Sdi, AiUInt8 b_LabCntIni, AiUInt16 *pw_LabIx, AiUInt32 *px_LabCnt, AiUInt32 *px_LabErr)

  Get receive status of a specific label on a specific channel.
- AiReturn Api429RxLabelBufferRead (AiUInt8 board_handle, AiUInt8 b_Chn, AiUInt8 b_Label, AiUInt8 b_Sdi, AiUInt16 w_BufSize, struct api429_rx_buf_ctl *px_Info, struct api429_rx_buf_entry *px_LData)

  Read data from a specific label's receive buffer.
- AiReturn Api429RxLabelBufferOffsetGet (AiUInt8 board_handle, AiUInt8 b_Chn, AiUInt8 b_Label, AiUInt8 b_Sdi, AiUInt32 *offset, AiUInt16 *w_Size)

  Get offset of specific label receive buffer.

3.8.1 Detailed Description

This module contains functionality related to receiving of Arinc 429 bus traffic.

3.8.2 Typedef Documentation

TY_API429_RCV_PB_CMD

TY_API429_RCV_PB_CMD

Convenience typedef for api429_rcv_pb_cmd

Definition at line 90 of file Api429Rx.h.

TY_API429_RX_ACTIVITY

TY_API429_RX_ACTIVITY

Convenience macro for api429_rx_activity

Definition at line 59 of file Api429Rx.h.

TY_API429_RX_BUF_CTL

TY_API429_RX_BUF_CTL

Convenience typedef for api429_rx_buf_ctl

Definition at line 156 of file Api429Rx.h.

TY_API429_RX_BUF_ENTRY

TY_API429_RX_BUF_ENTRY

Convenience macro for api429_rx_buf_entry

Definition at line 137 of file Api429Rx.h.

TY_API429_RX_FRAME_RESPONSE_SETUP

TY_API429_RX_FRAME_RESPONSE_SETUP

Convenience macro for api429_rx_frame_response_setup

Definition at line 43 of file Api429Rx.h.

TY_API429_RX_LABEL_SETUP

TY_API429_RX_LABEL_SETUP

Convenience typedef for api429_rx_label_setup

Definition at line 122 of file Api429Rx.h.

3.8.3 Function Documentation

Api429RxActivityGet()

```
AiReturn Api429RxActivityGet (
                AiUInt8 board_handle,
                struct api429_rx_activity * pxActivity )
```

Get overview of received label activity.

This function reads the activity information of the RX labels for each channel. Each time a label is received the corresponding bit of the activity bit field is set. The activity bit field is reset, when stopping and restarting the RX channel.

Parameters

in	*board_handle*	handle to board to read receive activity from
out	*pxActivity*	receive activity will be stored here

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429RxDataLoopAssign()

```
AiReturn Api429RxDataLoopAssign (
                AiUInt8 board_handle,
                AiUInt8 channel,
                AiUInt8 transmitter_channel )
```

Set up a data forwarding for a receive channel.

This function can be applied to receive channels that have been initialized with Api429RxInit.
After calling this function, all data received for enabled labels will be forwarded to the specified transmitter channel and sent on this channel at once.
Attention: The specified transmitter channel is not checked for being correctly configured.
The transmitter must be configured in loop/pollution mode and should match the speed of the receiver channel. Looping a high speed channel to a low speed channel probably results in data loss.
Receiver and Transmitter channel must be both either located on the lower 16 or on the upper 16 channels of the channel set the device provides in order to get a valid data loop.

This functionality is not supported on embedded boards (like AXE429)

Parameters

in	*board_handle*	handle to the device the channels belong to
in	*channel*	the receive channel that shall forward the data
in	*transmitter_channel*	the transmitter channel that will send the forwarded data

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429RxFrameResponseAssign()

```
AiReturn Api429RxFrameResponseAssign (
                AiUInt8 board_handle,
                AiUInt8 receive_channel,
                AiUInt8 label,
                AiUInt8 sdi,
                struct api429_rx_frame_response_setup * p_Setup )
```

Assign an automatic frame response on reception of a label.

This function can be used to initiate sending of a response when a specific label is received.
The channel on which the response frame shall be transmitted has to be initialized in a frame based mode before with Api429TxInit.
Also the frame to be responded must be set-up with Api429TxAcycFrameCreate before calling this function with the specified frame ID.

Parameters

in	*board_handle*	handle to the module to use
in	*receive_channel*	the receive channel to react on reception
in	*label*	the label on which reception a response shall be initiated
in	*sdi*	the SDI that may extend the specified label. Only valid if SDI sorting has been enabled using Api429RxInit
in	*p_Setup*	pointer to response set-up data. See api429_rx_frame_response_setup

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429RxFrameResponseRelease()

```
AiReturn Api429RxFrameResponseRelease (
                AiUInt8 board_handle,
                AiUInt8 receive_channel,
```

```
                AiUInt8 label,
                AiUInt8 sdi )
```

Release an automatic frame response on reception of a label.

This function can be used to release a frame response previously set-up with Api429RxFrameResponseAssign.

Parameters

in	*board_handle*	handle to the module to use
in	*receive_channel*	the receive channel to release response from
in	*label*	the label to release response from
in	*sdi*	the SDI that may extend the specified label. Only valid if SDI sorting has been enabled using Api429RxInit

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429RxInit()

```
AiReturn Api429RxInit (
                AiUInt8 board_handle,
                AiUInt8 b_Chn,
                AiBoolean sdi_enabled,
                AiBoolean parity_check_enabled )
```

Basic initialization of a channel for receiving data.

This function will initialize an Arinc429 channel for data reception.
Any previous configuration of the specified channel is cleared.
After running this command, the channel will be in a halted state with reception of any labels disabled and no monitor assigned.

Parameters

in	*board_handle*	handle to the device the channel belongs to
in	*b_Chn*	ID of the channel to initialize
in	*sdi_enabled*	if set to AiTrue, the SDI field is taken into account to determine received labels
in	*parity_check_enabled*	if set to AiTrue, the parity bit of received labels is used to indicate parity errors if set to AiFalse, the parity bit will be untouched and returned as received on bus

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429RxLabelBufferOffsetGet()

```
AiReturn Api429RxLabelBufferOffsetGet (
                AiUInt8 board_handle,
                AiUInt8 b_Chn,
                AiUInt8 b_Label,
                AiUInt8 b_Sdi,
                AiUInt32 * offset,
                AiUInt16 * w_Size )
```

Get offset of specific label receive buffer.

This function is utilized to return the offset of a specified label receiver buffer in global memory.

Parameters

in	*board_handle*	handle to board the receive channel belongs to
in	*b_Chn*	ID of receive channel
in	*b_Label*	ID of label to get receive buffer offset of
in	*b_Sdi*	SDI extension for label
out	*offset*	It is returned as a relative offset based on the start of the Global Memory. The offset points to the data buffer header (refer to the Hardware Manual of the Data Buffer Structure, if this command is used).
out	*w_Size*	Label Receiver Buffer Size

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429RxLabelBufferRead()

```
AiReturn Api429RxLabelBufferRead (
                AiUInt8 board_handle,
                AiUInt8 b_Chn,
                AiUInt8 b_Label,
```

```
                AiUInt8 b_Sdi,
                AiUInt16 w_BufSize,
                struct api429_rx_buf_ctl * px_Info,
                struct api429_rx_buf_entry * px_LData )
```

Read data from a specific label's receive buffer.

Parameters

in	*board_handle*	handle to board the receive channel belongs to
in	*b_Chn*	ID of receive channel
in	*b_Label*	ID of label to read receive buffer data of
in	*b_Sdi*	SDI extension for label
in	*w_BufSize*	Label Receive Buffer Size It should correspond to "label receive buffer size" in the Api429RxLabelConfigure function.
out	*px_Info*	Index information about the buffer. See api429_rx_buf_ctl
out	*px_LData*	Array of data. See api429_rx_buf_entry

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429RxLabelBufferWrite()

```
AiReturn Api429RxLabelBufferWrite (
                AiUInt8 board_handle,
                AiUInt8 b_Chn,
                AiUInt8 b_Label,
                AiUInt8 b_Sdi,
                AiUInt16 w_BufSize,
                struct api429_rx_buf_entry * px_LData,
                AiUInt8 b_Clear )
```

Writes data to a receive label buffer.

This function is utilized to fill the specified receiver label data buffer on the selected Arinc 429 receiver channel with default data. This command can be used to clear the data buffer after the receiver was stopped.

Parameters

in	*board_handle*	handle to board the receive channel belongs to
in	*b_Chn*	ID of receive channel
in	*b_Label*	ID of label for which buffer shall be written

Parameters

in	*b_Sdi*	SDI extension of label
in	*w_BufSize*	Amount of receive buffer entries to write to label receive buffer
in	*px_LData*	Array of pointer to Label Receive Buffer Data structure
in	*b_Clear*	if set to AiTrue, all data in the buffer will be cleared instead of using **px_LData**

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429RxLabelConfigure()

```
AiReturn Api429RxLabelConfigure (
                AiUInt8 board_handle,
                AiUInt8 b_Chn,
                struct api429_rx_label_setup * label_setup,
                AiUInt8 * puc_Status,
                AiUInt32 * pul_FreeMem )
```

Configure reception of specific labels.

This function controls the specified receiver label handling on the selected Arinc 429 receiver channel and allocates/releases a receive data buffer. This function clears the Loop/Pollution blocks and the function blocks in the target software.

Parameters

in	*board_handle*	handle to board
in	*b_Chn*	ID of channel to configure label on
in	*label_setup*	See api429_rx_label_setup
out	*puc_Status*	if 1, label buffer could not be allocated due to too little memory (may be NULL)
out	*pul_FreeMem*	Size of the free memory available for label buffer (may be NULL)

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429RxLabelStatusGet()

```
AiReturn Api429RxLabelStatusGet (
                AiUInt8 board_handle,
                AiUInt8 b_Chn,
                AiUInt8 b_Label,
                AiUInt8 b_Sdi,
                AiUInt8 b_LabCntIni,
                AiUInt16 * pw_LabIx,
                AiUInt32 * px_LabCnt,
                AiUInt32 * px_LabErr )
```

Get receive status of a specific label on a specific channel.

This function is utilized to read the execution status of a specific label on the selected Arinc 429 receiver channel.

Parameters

in	*board_handle*	handle to board the receive channel belongs to
in	*b_Chn*	ID of receive channel
in	*b_Label*	ID of label to read status of
in	*b_Sdi*	SDI extension for label
in	*b_LabCntIni*	if set to 1, reset counters to zero after function returns
out	*pw_LabIx*	Receive Buffer Fill Index. (may be NULL)
out	*px_LabCnt*	Number of times the label was received (may be NULL)
out	*px_LabErr*	Number of times the label was erroneously detected (may be NULL)

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429RxMultiLabelConfigure()

```
AiReturn Api429RxMultiLabelConfigure (
                AiUInt8 board_handle,
                AiUInt8 b_Chn,
                AiUInt32 ulCount,
                struct api429_rx_label_setup * pxSetup )
```

Configure reception of multiple labels.

This function controls the specified receiver label handling on the selected Arinc 429 receiver channel and allocates/releases a receive data buffer. This function clears the Loop/Pollution blocks and the

function blocks in the target software.

Parameters

in	*board_handle*	handle to board
in	*b_Chn*	ID of channel to configure label on
in	*ulCount*	number of labels to configure
in	*pxSetup*	Array of api429_rx_label_setup instances. Must hold at least **ulCount** entries

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429RxPollutionConfigure()

```
AiReturn Api429RxPollutionConfigure (
                AiUInt8 board_handle,
                AiUInt8 b_Chn,
                AiUInt8 b_Label,
                AiUInt8 b_Sdi,
                AiUInt8 b_Con,
                struct api429_rcv_pb_cmd * px_PollBlk )
```

Used to define data modifications in a receiver/transmitter data loop.

Parameters

in	*board_handle*	handle to board
in	*b_Chn*	ID of receiver channel in a data loop set-up
in	*b_Label*	ID of label to install modification for
in	*b_Sdi*	SDI extension of label
in	*b_Con*	enable/disable modification
in	*px_PollBlk*	See api429_rcv_pb_cmd

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429RxStatusGet()

```
AiReturn Api429RxStatusGet (
                AiUInt8 board_handle,
                AiUInt8 b_Chn,
                AiUInt8 * b_RxStatus,
                AiUInt32 * px_MsgCnt,
                AiUInt32 * px_ErrCnt )
```

Get status of channel in receive mode.

This function is utilized to get the execution status of the selected Arinc 429 receiver channel and the global receive channel message/error count information.

Parameters

in	*board_handle*	handle to board the channel belongs to
in	*b_Chn*	ID of receive channel to get status of
out	*b_RxStatus*	either API429_HALT or API429_BUSY (may be NULL)
out	*px_MsgCnt*	Total number of received labels on the channel (may be NULL)
out	*px_ErrCnt*	Total number of erroneous labels received on the channel (may be NULL)

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

3.9 Data Transmitting

Data Structures

- struct api429_xfer
- struct api429_xfer_out
- struct api429_lxfer_dyntag
- struct api429_xfer_data
- struct api429_xfer_info
- struct api429_xfer_data_read_input
- struct api429_mframe_in
- struct api429_mframe_out

Typedefs

- typedef enum api429_tx_mode TY_E_API429_TX_MODE
- typedef enum api429_xfer_type TY_E_API429_XFER_TYPE
- typedef enum api429_xfer_error TY_E_API429_XFER_ERROR
- typedef struct api429_xfer TY_API429_XFER
- typedef struct api429_xfer_out TY_API429_XFER_OUT
- typedef struct api429_lxfer_dyntag TY_API429_LXFER_DYNTAG
- typedef struct api429_xfer_data TY_API429_XFER_DATA
- typedef struct api429_xfer_info TY_API429_XFER_INFO
- typedef struct api429_xfer_data_read_input TY_API429_XFER_DATA_READ_INPUT
- typedef struct api429_mframe_in TY_API429_MFRAME_IN
- typedef struct api429_mframe_out TY_API429_MFRAME_OUT

Enumerations

- enum api429_tx_mode {
  API429_TX_MODE_FRAMING = 0,
  API429_TX_MODE_LOOP,
  API429_TX_MODE_PHYS_REPLAY,
  API429_TX_MODE_FRAMING_DYNTAG,
  API429_TX_MODE_FIFO,
  API429_TX_MODE_RATE_CONTROLLED }
- enum api429_xfer_type {
  API429_TX_LAB_XFER = 1,
  API429_TX_NOP_XFER,
  API429_TX_STROBE,
  API429_TX_DELAY,
  API429_TX_LAB_XFER_TT,
  API429_TX_EXT_TRIGGER,
  API429_TX_LAB_XFER_32,
  API429_TX_LAB_XFER_TT_32 }

- enum api429_xfer_error {
  API429_XFER_ERR_DIS = 0,
  API429_XFER_ERR_BCH,
  API429_XFER_ERR_BCL,
  API429_XFER_ERR_COD,
  API429_XFER_ERR_PAR,
  API429_XFER_ERR_GAP,
  API429_XFER_ERR_WAIT }

Functions

- AiReturn Api429TxInit (AiUInt8 board_handle, AiUInt8 channel_id, enum api429_tx_mode mode, AiBoolean auto_parity)

  Initialize specific channel for data transmission.
- AiReturn Api429TxAmplitudeSet (AiUInt8 board_handle, AiUInt8 b_Chn, AiFloat fAmplitude, AiFloat *pfCalcAmpl)

  Set transmission amplitude.
- AiReturn Api429TxStatusGet (AiUInt8 board_handle, AiUInt8 b_Chn, AiUInt8 *pb_TxStatus, AiUInt32 *pl_GlbCnt)

  Get Status of a transmitter channel.
- AiReturn Api429TxStartOnTTag (AiUInt8 board_handle, AiUInt8 channel, struct api429_time *start_ttag)

  This function starts a transmit channel on at a given point of time.
- AiReturn Api429TxStartOnTrigger (AiUInt8 board_handle, AiUInt8 channel, AiUInt32 trigger_line)

  This function starts a transmit channel on at a given point of time.
- AiReturn Api429TxAcycFrameCreate (AiUInt8 board_handle, AiUInt8 ucChannel, AiUInt32 ulXferCnt, AiUInt32 *pulXfers, AiUInt32 *pulFrameId)

  Creates an acyclic frame.
- AiReturn Api429TxAcycFrameDelete (AiUInt8 board_handle, AiUInt8 ucChannel, AiUInt32 ulFrameId)

  Deletes an acyclic frame.
- AiReturn Api429TxAcycFrameSend (AiUInt8 board_handle, AiUInt8 ucChannel, AiUInt32 ulFrameId)

  Sends an acyclic frame.
- AiReturn Api429TxRepetitionCountSet (AiUInt8 board_handle, AiUInt8 channel, AiUInt32 count)

  Defines how many times the major frame shall be sent.
- AiReturn Api429TxXferRateAdd (AiUInt8 board_handle, AiUInt8 ucChannel, AiUInt32 ulXferId, AiUInt32 ulRateInMs)

  This function adds a transfer to the framing.
- AiReturn Api429TxXferRateRemove (AiUInt8 board_handle, AiUInt8 ucChannel, AiUInt32 ulXferId)

  This function removes a transfer from the framing.
- AiReturn Api429TxXferRateShow (AiUInt8 board_handle, AiUInt8 ucChannel, AiUInt32 ulXferId, AiUInt32 *pulRateInMs)

  This function shows the actual rate of a transfer.

- AiReturn Api429TxPrepareFraming (AiUInt8 board_handle, AiUInt8 channel)

  This function calculates the framing.
- AiReturn Api429TxFrameTimeSet (AiUInt8 board_handle, AiUInt8 b_Chn, AiUInt16 w_Ftm)

  Set minor frame time for framing based transmit channel.
- AiReturn Api429TxXferCreate (AiUInt8 board_handle, AiUInt8 b_Chn, struct api429_xfer *px_Xfer, struct api429_xfer_out *px_XferInfo)

  Create a transfer instruction for framing based transmitter channels.
- AiReturn Api429TxXferDyntagAssign (AiUInt8 board_handle, AiUInt8 b_Chn, AiUInt32 ul_XferId, struct api429_lxfer_dyntag *px_Dyntag)

  Assign automatic, dynamic data modification to transfers.
- AiReturn Api429TxXferBufferWrite (AiUInt8 board_handle, AiUInt8 b_Chn, AiUInt32 ul_XferId, AiUInt16 w_BufStart, AiUInt16 w_BufSize, AiUInt32 *pl_LData)

  Write data to a transfer buffer.
- AiReturn Api429TxXferBufferRead (AiUInt8 board_handle, AiUInt8 b_Chn, struct api429_xfer_data_read_input *px_XferDataInput, struct api429_xfer_data *px_XferData)

  Read data from a transfer buffer.
- AiReturn Api429TxMinorFrameCreate (AiUInt8 board_handle, AiUInt8 b_Chn, struct api429_mframe_in *px_MFrame, struct api429_mframe_out *px_MFrameInfo)

  Create a minor frame on framing based transmit channels.
- AiReturn **Api429TxMinorFrameCreateLV** (AiUInt8 module_handle, AiUInt8 channel, AiUInt32 ul_FrmId, AiUInt32 ul_XferCnt, AiUInt32 *pul_Xfers, TY_API429_MFRAME_OUT *px_MFrameInfo)
- AiReturn Api429TxMajorFrameCreate (AiUInt8 board_handle, AiUInt8 b_Chn, AiUInt32 ul_FrmCnt, AiUInt32 *pul_Frames)

  Create a major frame on framing based transmit channel.
- AiReturn Api429TxXferDelete (AiUInt8 board_handle, AiUInt8 b_Chn, AiUInt32 ul_XferId)

  Delete a transfer instruction.
- AiReturn Api429TxXferSkip (AiUInt8 board_handle, AiUInt8 b_Chn, AiBoolean skip, AiUInt32 ul_XferId)

  Allows to skip transmission of specific transfers on-th-fly.
- AiReturn Api429TxMinorFrameDelete (AiUInt8 board_handle, AiUInt8 b_Chn, AiUInt32 ul_FrmId)

  Deletes a minor frame from framing based transmit channel.
- AiReturn Api429TxMajorFrameDelete (AiUInt8 board_handle, AiUInt8 b_Chn)

  Deletes a major frame from framing based transmit channel.
- AiReturn Api429TxXferStatusGet (AiUInt8 board_handle, AiUInt8 b_Chn, AiUInt32 b_XferId, struct api429_xfer_info *px_XferInfo)

  Get transmission status of a specific transfer.
- AiReturn Api429TxXferBufferOffsetGet (AiUInt8 board_handle, AiUInt8 b_Chn, AiUInt32 ul_XferId, AiUInt32 *pl_Addr, AiUInt16 *pw_Size)

  Get base offset of transfer buffer.

3.9.1 Detailed Description

This module contains functionality related to transmitting of Arinc 429 data .

3.9.2 Typedef Documentation

TY_API429_LXFER_DYNTAG

TY_API429_LXFER_DYNTAG

Convenience typedef for api429_lxfer_dyntag

Definition at line 172 of file Api429Tx.h.

TY_API429_MFRAME_IN

TY_API429_MFRAME_IN

Convenience typedef for api429_mframe_in

Definition at line 247 of file Api429Tx.h.

TY_API429_MFRAME_OUT

TY_API429_MFRAME_OUT

Convenience typedef api429_mframe_out

Definition at line 263 of file Api429Tx.h.

TY_API429_XFER

TY_API429_XFER

Convenience typedef for api429_xfer

Definition at line 124 of file Api429Tx.h.

TY_API429_XFER_DATA

TY_API429_XFER_DATA

Convenience typedef for api429_xfer_data

Definition at line 191 of file Api429Tx.h.

TY_API429_XFER_DATA_READ_INPUT

TY_API429_XFER_DATA_READ_INPUT

Convenience typedef for api429_xfer_data_read_input

Definition at line 229 of file Api429Tx.h.

TY_API429_XFER_INFO

TY_API429_XFER_INFO

Convenience typedef for api429_xfer_info

Definition at line 210 of file Api429Tx.h.

TY_API429_XFER_OUT

TY_API429_XFER_OUT

Convenience typedef for struct api429_xfer_out

Definition at line 143 of file Api429Tx.h.

TY_E_API429_TX_MODE

TY_E_API429_TX_MODE

Convenience macro for api429_tx_mode

Definition at line 43 of file Api429Tx.h.

TY_E_API429_XFER_ERROR

TY_E_API429_XFER_ERROR

Convenience macro for api429_xfer_error

Definition at line 89 of file Api429Tx.h.

TY_E_API429_XFER_TYPE

TY_E_API429_XFER_TYPE

Convenience typedef for api429_xfer_type

Definition at line 67 of file Api429Tx.h.

3.9.3 Enumeration Type Documentation

api429_tx_mode

enum api429_tx_mode

Enumeration of all available transmission modes

Enumerator

API429_TX_MODE_FRAMING	Standard framing mode
API429_TX_MODE_LOOP	Channel is bound with a receiver channel forming a data loop. It will re-transmit all data words the receiver gets. This mode is not supported on embedded boards (like AXE429)
API429_TX_MODE_PHYS_REPLAY	Replay mode
API429_TX_MODE_FRAMING_DYNTAG	Standard framing mode with dyntags
API429_TX_MODE_FIFO	Fifo based transmission mode
API429_TX_MODE_RATE_CONTROLLED	Framing mode where data word transmission is controlled by specifying cyclic transmission rates

Definition at line 30 of file Api429Tx.h.

api429_xfer_error

enum api429_xfer_error

Enumeration of all available error types that can be injected for a transfer. The error injection functionality is not supported on embedded boards (like AXE429)

Enumerator

API429_XFER_ERR_DIS	No error
API429_XFER_ERR_BCH	Bit count high error
API429_XFER_ERR_BCL	Bit count low error
API429_XFER_ERR_COD	Coding error
API429_XFER_ERR_PAR	Parity error
API429_XFER_ERR_GAP	Gap error
API429_XFER_ERR_WAIT	Error wait. Don't send frame

Definition at line 75 of file Api429Tx.h.

api429_xfer_type

enum api429_xfer_type

Enumeration of all available transfer types for use with framing/rate-oriented based transmitter channels

Enumerator

API429_TX_LAB_XFER	Label Transfer
API429_TX_NOP_XFER	NOP Transfer
API429_TX_STROBE	Trigger Strobe
API429_TX_DELAY	Delay Instruction
API429_TX_LAB_XFER_TT	Label Transfer with TimeTag
API429_TX_EXT_TRIGGER	Wait for external trigger
API429_TX_LAB_XFER_32	32Bit Label Transfer
API429_TX_LAB_XFER_TT_32	32Bit Label Transfer with TimeTag

Definition at line 51 of file Api429Tx.h.

3.9.4 Function Documentation

Api429TxAcycFrameCreate()

```
AiReturn Api429TxAcycFrameCreate (
                AiUInt8 board_handle,
                AiUInt8 ucChannel,
                AiUInt32 ulXferCnt,
                AiUInt32 * pulXfers,
                AiUInt32 * pulFrameId )
```

Creates an acyclic frame.

This function searches for an unused minor frame ID, returns it and defines the sequence of label transfers identified by their transfer identifiers within a frame that may be used for an acyclic. Several acyclic frames may be used in parallel. When using TX Rate Oriented Mode (see function Api429TxInit), the function Api429TxPrepareFraming must be called before calling this function!

Parameters

in	*board_handle*	handle to the device the channel belongs to
in	*ucChannel*	ID of the channel to initialize
in	*ulXferCnt*	number of transfers that shall be sent with this frame
in	*pulXfers*	array containing a list of transfers that shall be sent with this frame
out	*pulFrameId*	ID of acyclic frame that was created with this command

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429TxAcycFrameDelete()

```
AiReturn Api429TxAcycFrameDelete (
                AiUInt8 board_handle,
                AiUInt8 ucChannel,
                AiUInt32 ulFrameId )
```

Deletes an acyclic frame.

This function removes an acyclic frame that was previously created with the command Api429TxAcyc↩ FrameCreate.

Parameters

in	*board_handle*	handle to the device the channel belongs to
in	*ucChannel*	ID of the channel to initialize
in	*ulFrameId*	ID of acyclic frame

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429TxAcycFrameSend()

```
AiReturn Api429TxAcycFrameSend (
              AiUInt8 board_handle,
              AiUInt8 ucChannel,
              AiUInt32 ulFrameId )
```

Sends an acyclic frame.

This function causes the board to immediately send an acyclic frame (created with the command Api429TxAcycFrameCreate). If an ARINC 429 word is currently being sent, the acyclic frame is sent directly after it. This may interrupt the current minor frame. To ensure the acyclic frames do not disturb the standard framing, please make sure the minor frame time is high enough. This function returns immediately and does not wait for the acyclic frame to finish. It is not recommended to issue this command to the board before the previous acyclic frame was completely sent.

Restriction: Acyclic frames are not supported when the channel is configured in loop mode (see Api429TxInit).

Parameters

in	*board_handle*	handle to the device the channel belongs to
in	*ucChannel*	ID of the channel to initialize
in	*ulFrameId*	ID of acyclic frame

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429TxAmplitudeSet()

```
AiReturn Api429TxAmplitudeSet (
                AiUInt8 board_handle,
                AiUInt8 b_Chn,
                AiFloat fAmplitude,
                AiFloat * pfCalcAmpl )
```

Set transmission amplitude.

This command defines the output amplitude of the selected Arinc 429 transmitter channel via a D/A converter.

The transmitter output amplitude can be changed on the lower 8 channels of an APX, APXX, ACX429, ACXX, ACE and APE429. For other modules (and all modules with 32 channels) this function has no effect! Please have a look into the corresponding data sheet for details.

Note: After a Api429BoardReset command or the Api429TxInit command the related D/A converter is loaded to an output voltage of 9.0 Volt (line to line).

Note: for APX429 and ACX429-3U boards there might be a short delay, before the output amplitude is adjusted

Parameters

in	*board_handle*	handle to board the channel belongs to
in	*b_Chn*	ID of channel to set amplitude for
in	*fAmplitude*	Amplitude Control. The internal D/A converter will be set, so that the output amplitude comes as near as possible to this value. The typical range is 0V .. 11V. Please refer to the Hardware Manual for details.
out	*pfCalcAmpl*	mirrors the input parameter fAmplitude. It was used before to show the digitization error. But it gave a false impression of accuracy, which neglected the variation of the components and the variation of the supply voltage. This parameter is no longer needed, but for compatibility reasons it is still kept.

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429TxFrameTimeSet()

```
AiReturn Api429TxFrameTimeSet (
                AiUInt8 board_handle,
                AiUInt8 b_Chn,
                AiUInt16 w_Ftm )
```

Set minor frame time for framing based transmit channel.

This command defines the minor frame time of the selected Arinc 429 transmitter channel in 1ms steps.

Parameters

in	*board_handle*	handle to board the channel belongs to
in	*b_Chn*	ID of channel to set frame time for
in	*w_Ftm*	Minor Frame Time in 1ms steps. Ranges from 1..65535

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429TxInit()

```
AiReturn Api429TxInit (
                AiUInt8 board_handle,
                AiUInt8 channel_id,
                enum api429_tx_mode mode,
                AiBoolean auto_parity )
```

Initialize specific channel for data transmission.

This command initializes the transmitter mode of the selected Arinc 429 transmitter channel. The transmitter operation is halted, the high and low priority stack pointer are initialized and the channel amplitude and the minor frame time are set to the following default values: -Amplitude : 10 Volts (line to line) -Minor Frame Time : 100ms

Parameters

in	*board_handle*	handle to board the channel belongs to
in	*channel_id*	ID of channel to initialize
in	*mode*	See api429_tx_mode
in	*auto_parity*	If AiTrue, parity bit will automatically be generated for transmitted data word. (Not applicable in **Replay** or **Loop** mode)

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429TxMajorFrameCreate()

```
AiReturn Api429TxMajorFrameCreate (
                AiUInt8 board_handle,
                AiUInt8 b_Chn,
                AiUInt32 ul_FrmCnt,
                AiUInt32 * pul_Frames )
```

Create a major frame on framing based transmit channel.

This function defines the sequence of label transfer minor frames identified by their frame identifiers within the major frame on the selected A429 transmitter channel.

Parameters

in	*board_handle*	handle to board the channel belongs to
in	*b_Chn*	ID of channel to install major frame on
in	*ul_FrmCnt*	Amount of minor frames in major frame. Ranges from 0..2000
in	*pul_Frames*	Minor frame identifiers to add to major frame (up to 2000, see 'ul_FrmCnt')

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429TxMajorFrameDelete()

```
AiReturn Api429TxMajorFrameDelete (
                AiUInt8 board_handle,
                AiUInt8 b_Chn )
```

Deletes a major frame from framing based transmit channel.

This function resets the minor frame sequence within the major frame on the selected Arinc 429 transmitter channel.

Parameters

in	*board_handle*	handle to board the channel belongs to
in	*b_Chn*	ID of transmit channel to delete major frame from

Returns

- API_OK on success

- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429TxMinorFrameCreate()

```
AiReturn Api429TxMinorFrameCreate (
                AiUInt8 board_handle,
                AiUInt8 b_Chn,
                struct api429_mframe_in * px_MFrame,
                struct api429_mframe_out * px_MFrameInfo )
```

Create a minor frame on framing based transmit channels.

This function defines the sequence of label transfers identified by their transfer identifiers within a minor frame specified by a frame identifier on the selected Arinc 429 transmitter channel.

Parameters

in	*board_handle*	handle to board the channel belongs to
in	*b_Chn*	ID of transmit channel
in	*px_MFrame*	See api429_mframe_in
out	*px_MFrameInfo*	See api429_mframe_out (may be NULL)

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429TxMinorFrameDelete()

```
AiReturn Api429TxMinorFrameDelete (
                AiUInt8 board_handle,
                AiUInt8 b_Chn,
                AiUInt32 ul_FrmId )
```

Deletes a minor frame from framing based transmit channel.

This function deletes the transmitter label transfer minor frame specified by its frame identifier on the selected Arinc 429 channel.

Parameters

in	*board_handle*	handle to board the channel belongs to

Parameters

in	*b_Chn*	ID of transmit channel to delete minor frame from
in	*ul_FrmId*	ID of frame to delete

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429TxPrepareFraming()

```
AiReturn Api429TxPrepareFraming (
                AiUInt8 board_handle,
                AiUInt8 channel )
```

This function calculates the framing.

This function calculates a framing with the previously setup transfers and rates, so that the channel can be started later without any delay. This function can only be used when the TX channel is configured for TX Rate Oriented Mode (see Api429TxInit). This function is optional, however, when using also acyclic frames this function must be called before creating an acyclic frame with Api429TxAcycFrameCreate.

Parameters

in	*board_handle*	handle to the device
in	*channel*	channel for which this frame shall be created

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429TxRepetitionCountSet()

```
AiReturn Api429TxRepetitionCountSet (
                AiUInt8 board_handle,
                AiUInt8 channel,
                AiUInt32 count )
```

Defines how many times the major frame shall be sent.

This command defines the number of times the major frame shall be sent. By setting the parameter count to zero, the major frame is sent indefinitely, until explicitely stopped. This is the default value with which the channel is initialized.

Parameters

in	*board_handle*	handle to the device the channel belongs to
in	*channel*	ID of the channel to initialize
in	*count*	Number of times a major frame shall be sent. If set to 0 the major frame is sent until stopped. Valid range is 0..65535

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429TxStartOnTrigger()

```
AiReturn Api429TxStartOnTrigger (
                AiUInt8 board_handle,
                AiUInt8 channel,
                AiUInt32 trigger_line )
```

This function starts a transmit channel on at a given point of time.

This function starts the execution of the label transfers on the selected Arinc 429 transmitter channel when a strobe is detected at the given trigger line

Parameters

in	*board_handle*	handle to the device
in	*channel*	ID of the channel that shall be affected
in	*trigger_line*	ID of the trigger input line (0..3)

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429TxStartOnTTag()

```
AiReturn Api429TxStartOnTTag (
                AiUInt8 board_handle,
                AiUInt8 channel,
                struct api429_time * start_ttag )
```

This function starts a transmit channel on at a given point of time.

This function starts the execution of the label transfers on the selected Arinc 429 transmitter channel at a given point of time. Every 500us the current IRIG Time is compared with the values provided by this command. If it has passed the given time tag, it starts the transmit channel. Note: This command is only supported for AyX429 boards

Parameters

in	*board_handle*	handle to the device
in	*channel*	ID of the channel that shall be affected
in	*start_ttag*	the point of time the transmission shall start.

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429TxStatusGet()

```
AiReturn Api429TxStatusGet (
                AiUInt8 board_handle,
                AiUInt8 b_Chn,
                AiUInt8 * pb_TxStatus,
                AiUInt32 * pl_GlbCnt )
```

Get Status of a transmitter channel.

This function is utilized to read the execution status of the selected Arinc 429 transmitter channel.

Parameters

in	*board_handle*	handle to board the channel belongs to
in	*b_Chn*	ID of channel to get status of
out	*pb_TxStatus*	Either API429_HALT or API429_BUSY (may be NULL)
out	*pl_GlbCnt*	Global transfer count (may be NULL)

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429TxXferBufferOffsetGet()

```
AiReturn Api429TxXferBufferOffsetGet (
                AiUInt8 board_handle,
                AiUInt8 b_Chn,
                AiUInt32 ul_XferId,
                AiUInt32 * pl_Addr,
                AiUInt16 * pw_Size )
```

Get base offset of transfer buffer.

This function is utilized to return the base offset of a specified label transmitter identified buffer and a specified channel.

Parameters

in	*board_handle*	handle to board the channel belongs to
in	*b_Chn*	ID of channel the transfer was set-up on
in	*ul_XferId*	ID of transfer to get buffer offset of
out	*pl_Addr*	Transmit Buffer offset
out	*pw_Size*	Transmit Buffer Size

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429TxXferBufferRead()

```
AiReturn Api429TxXferBufferRead (
                AiUInt8 board_handle,
                AiUInt8 b_Chn,
                struct api429_xfer_data_read_input * px_XferDataInput,
                struct api429_xfer_data * px_XferData )
```

Read data from a transfer buffer.

This function is utilized to read the send label transfer data buffer on the selected Arinc 429 transmitter channel.

Parameters

in	*board_handle*	handle to board the channel belongs to
in	*b_Chn*	ID of channel the transfer was et-up on
in	*px_XferDataInput*	See api429_xfer_data_read_input
out	*px_XferData*	See api429_xfer_data

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429TxXferBufferWrite()

```
AiReturn Api429TxXferBufferWrite (
                AiUInt8 board_handle,
                AiUInt8 b_Chn,
                AiUInt32 ul_XferId,
                AiUInt16 w_BufStart,
                AiUInt16 w_BufSize,
                AiUInt32 * pl_LData )
```

Write data to a transfer buffer.

This function is utilized to fill the specified label transfer data buffer on the selected Arinc 429 transmitter channel with data information.

Parameters

in	*board_handle*	handle to board the channel belongs to
in	*b_Chn*	ID of channel the transfer was created for
in	*ul_XferId*	ID of transfer to which buffer shall be written
in	*w_BufStart*	Buffer index to start writing to. Ranges from 1..1024
in	*w_BufSize*	Amount of 32 bit label data words to write to transmit buffer.
in	*pl_LData*	Array of data words to write. The lowest byte of each data word is the label id. To meet the ARINC 429 specification the label id is automatically transferred in reverse bit order.

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429TxXferCreate()

```
AiReturn Api429TxXferCreate (
                AiUInt8 board_handle,
                AiUInt8 b_Chn,
                struct api429_xfer * px_Xfer,
                struct api429_xfer_out * px_XferInfo )
```

Create a transfer instruction for framing based transmitter channels.

This function defines a complete transmitter label transfer including error injection specification to be executed on the selected Arinc 429 transmitter channel. When executing this instruction a data buffer specified by "buf_size" is allocated within the on-board memory. The data buffer contents may be defined with the Api429TxXferBufferWrite command. Whenever a transfer is called (for example within a minor frame) a single ARINC429 data word of the buffer is being sent. The allocated buffer can be freed using the Api429TxXferDelete command.

Parameters

in	*board_handle*	handle to board the channel belongs to
in	*b_Chn*	ID of transmitter channel to create transfer for
in	*px_Xfer*	See api429_xfer
out	*px_XferInfo*	See api429_xfer_out (may be NULL)

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429TxXferDelete()

```
AiReturn Api429TxXferDelete (
                AiUInt8 board_handle,
                AiUInt8 b_Chn,
                AiUInt32 ul_XferId )
```

Delete a transfer instruction.

This function deletes the transmitter label transfer identified by its transfer identifier on the selected Arinc 429 channel

Parameters

in	*board_handle*	handle to board the channel belongs to
in	*b_Chn*	ID of channel the transfer was created on
in	*ul_XferId*	ID of transfer to delete

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429TxXferDyntagAssign()

```
AiReturn Api429TxXferDyntagAssign (
                AiUInt8 board_handle,
                AiUInt8 b_Chn,
                AiUInt32 ul_XferId,
                struct api429_lxfer_dyntag * px_Dyntag )
```

Assign automatic, dynamic data modification to transfers.

This function defines dynamic data modification for a given label transfer, that was already set up with the Api429TxXferCreate command. This command requires that the channel was set up with Api429TxInit with mode API429_TX_MODE_FRAMING_DYNTAG. After each transmission the buffer entry is modified according to the function specified with the parameter ul_Mode.

Parameters

in	*board_handle*	handle to board the channel belongs to
in	*b_Chn*	ID of channel the transfer was set up for
in	*ul_XferId*	ID of transfer to assign dynamic data modification for
in	*px_Dyntag*	See api429_lxfer_dyntag

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429TxXferRateAdd()

```
AiReturn Api429TxXferRateAdd (
                AiUInt8 board_handle,
                AiUInt8 ucChannel,
                AiUInt32 ulXferId,
                AiUInt32 ulRateInMs )
```

This function adds a transfer to the framing.

This function adds a transfer which was set up with Api429TxXferCreate to a list from which a framing is automatically generated when starting the TX channel with Api429ChannelStart.

Parameters

in	*board_handle*	handle to the device
in	*ucChannel*	channel for which this frame shall be created
in	*ulXferId*	ID of the transfer that shall be added to the framing
in	*ulRateInMs*	rate with which this transfer shall appear in the framing

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429TxXferRateRemove()

```
AiReturn Api429TxXferRateRemove (
                AiUInt8 board_handle,
                AiUInt8 ucChannel,
                AiUInt32 ulXferId )
```

This function removes a transfer from the framing.

This function removes a transfer from the internal list that is used to automatically create the framing when calling Api429ChannelStart.

Parameters

in	*board_handle*	handle to the device
in	*ucChannel*	channel for which this frame shall be created
in	*ulXferId*	ID of the transfer that shall be added to the framing

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429TxXferRateShow()

```
AiReturn Api429TxXferRateShow (
                AiUInt8 board_handle,
                AiUInt8 ucChannel,
                AiUInt32 ulXferId,
                AiUInt32 * pulRateInMs )
```

This function shows the actual rate of a transfer.

This function shows the rate that is really used in the automatically generated framing for a given transfer. If the rate paremeters of some Api429TxXferRateAdd commands are unfavourable, it may result in some rounding deviations.

Parameters

in	*board_handle*	handle to the device
in	*ucChannel*	channel for which this frame shall be created
in	*ulXferId*	ID of the transfer that shall be added to the framing
out	*pulRateInMs*	rate with which this transfer really appears in the framing

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429TxXferSkip()

```
AiReturn Api429TxXferSkip (
                AiUInt8 board_handle,
                AiUInt8 b_Chn,
                AiBoolean skip,
                AiUInt32 ul_XferId )
```

Allows to skip transmission of specific transfers on-th-fly.

This function enables/disables a label transfer identified by its transfer identifier.

Parameters

in	*board_handle*	handle to board the channel belongs to
in	*b_Chn*	ID of channel the transfer was set-up on
in	*skip*	If AiTrue, transfer is skipped, if AiFalse it will sent
in	*ul_XferId*	ID of transfer to skip

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429TxXferStatusGet()

```
AiReturn Api429TxXferStatusGet (
                AiUInt8 board_handle,
```

```
AiUInt8 b_Chn,
AiUInt32 b_XferId,
struct api429_xfer_info * px_XferInfo )
```

Get transmission status of a specific transfer.

This function is utilized to get the status of a specific label transfer on the selected Arinc 429 transmitter channel including time tag information (if setup with Api429TxXferCreate)

Parameters

in	*board_handle*	handle to board the channel belongs to
in	*b_Chn*	ID of channel the transfer was set-up for
in	*b_XferId*	ID of transfer to get status of
out	*px_XferInfo*	See api429_xfer_info

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

3.10 FIFO Based Transmitting

Data Structures

- struct api429_tx_fifo_setup
- struct api429_tx_fifo_status
- struct api429_tx_fifo_entry

Typedefs

- typedef struct api429_tx_fifo_setup TY_API429_TX_FIFO_SETUP
- typedef struct api429_tx_fifo_status TY_API429_TX_FIFO_STATUS
- typedef struct api429_tx_fifo_entry TY_API429_TX_FIFO_ENTRY

Functions

- AiReturn Api429TxFifoReset (AiUInt8 b_Module, AiUInt8 b_Chn)

  This function allows to reset a FIFO.
- AiReturn Api429TxFifoSetup (AiUInt8 b_Module, AiUInt8 b_Chn, struct api429_tx_fifo_setup *px↩ FifoSetup)

  This function allows to set up a TX FIFO with different values than default.
- AiReturn Api429TxFifoSetupGet (AiUInt8 b_Module, AiUInt8 b_Chn, struct api429_tx_fifo_setup *fifo_setup)

  This function returns the current setup of a TX FIFO.
- AiReturn Api429TxFifoStatusGet (AiUInt8 b_Module, AiUInt8 b_Chn, struct api429_tx_fifo_status *pxFifoStatus)
- AiReturn Api429TxFifoDataWordsWrite (AiUInt8 module_handle, AiUInt8 channel_id, AiUInt32 num_words, AiUInt32 *words, AiBoolean block, AiUInt32 *words_written)

  This function is used to write label transfer instructions to the FIFO.
- AiReturn Api429TxFifoWrite (AiUInt8 module_handle, AiUInt8 channel_id, AiUInt32 num_entries, struct api429_tx_fifo_entry *entries, AiBoolean block, AiUInt32 *entries_written)

  This function is used to write instructions to the FIFO.
- AiReturn Api429TxFifoDataWordCreate (struct api429_tx_fifo_entry *entry, AiUInt32 data, AiU↩ Int32 gap, enum api429_xfer_error error)

  Prepares a FIFO entry for sending one Arinc 429 data word.
- AiReturn Api429TxFifoDelayCreate (struct api429_tx_fifo_entry *entry, AiUInt32 time_100_usec)

  Prepares a FIFO entry for delaying the processing of a FIFO.
- AiReturn Api429TxFifoInterruptCreate (struct api429_tx_fifo_entry *entry, AiUInt32 tag)

  Prepares a FIFO entry for generating send notifications.
- AiReturn Api429TxFifoTriggerPulseCreate (struct api429_tx_fifo_entry *entry, AiUInt32 trigger_↩ line)

  Prepares a FIFO entry for generating a pulse on an output trigger line.
- AiReturn Api429TxFifoTriggerWaitCreate (struct api429_tx_fifo_entry *entry, AiUInt32 input_trigger↩ _line)

  Prepares a FIFO entry for suspend FIFO processing until input trigger detected.

3.10.1 Detailed Description

This module contains functionality related to transmitting of Arinc 429 data via a FIFO based mechanism.

3.10.2 Typedef Documentation

TY_API429_TX_FIFO_ENTRY

`TY_API429_TX_FIFO_ENTRY`

Convenience typedef for api429_tx_fifo_entry

Definition at line 77 of file Api429TxFifo.h.

TY_API429_TX_FIFO_SETUP

`TY_API429_TX_FIFO_SETUP`

Convenience typedef for api429_tx_fifo_setup

Definition at line 43 of file Api429TxFifo.h.

TY_API429_TX_FIFO_STATUS

`TY_API429_TX_FIFO_STATUS`

Convenience typedef for api429_tx_fifo_status

Definition at line 59 of file Api429TxFifo.h.

3.10.3 Function Documentation

Api429TxFifoDataWordCreate()

```
AiReturn Api429TxFifoDataWordCreate (
                struct api429_tx_fifo_entry * entry,
                AiUInt32 data,
                AiUInt32 gap,
                enum api429_xfer_error error )
```

Prepares a FIFO entry for sending one Arinc 429 data word.

This function can be used to set-up a specific FIFO entry for sending of an Arinc429 data word. The entry can be sent with Api429TxFifoWrite afterwards.

Parameters

out	*entry*	pointer to the TX FIFO entry to prepare
in	*data*	the actual data that will be sent on the bus.
in	*gap*	Gap time that will be inserted before sending of next word measured in Arinc 429 bits. The valid range for this parameter is 4..255. Note: values between 0 and 4 will result in a gap time of 4 bits to prevent violation of Arinc 429 specification
in	*error*	specifies an optional error type to inject with the word. See api429_xfer_error

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429TxFifoDataWordsWrite()

```
AiReturn Api429TxFifoDataWordsWrite (
                AiUInt8 module_handle,
                AiUInt8 channel_id,
                AiUInt32 num_words,
                AiUInt32 * words,
                AiBoolean block,
                AiUInt32 * words_written )
```

This function is used to write label transfer instructions to the FIFO.

This function writes label transfer instructions into the FIFO buffer. If the TX channel is started with Api429ChannelStart before or after a call of this function, all FIFO buffer entries will be sent as soon as possible in accordance with the given default interlabel gap. It is recommended to Setup the FIFO and

Start the TX channel only once to arm the FIFO so that data written to the FIFO via this function will be sent with minimum latency.

Parameters

in	*module_handle*	handle to the device
in	*channel_id*	ID of the channel that shall be affected
in	*num_words*	number of data words to send via FIFO
in	*words*	array containing the arinc 429 data words that shall be sent via FIFO
in	*block*	if set to AiTrue, function call will block until all entries were written to FIFO. If set to AiFalse, function will just send the number of words that currently fit into FIFO.
out	*words_written*	number of data words that were really written

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429TxFifoDelayCreate()

```
AiReturn Api429TxFifoDelayCreate (
                struct api429_tx_fifo_entry * entry,
                AiUInt32 time_100_usec )
```

Prepares a FIFO entry for delaying the processing of a FIFO.

This function can be used to prepare a 'wait' FIFO entry.
The entry can be sent with Api429TxFifoWrite afterwards.
When put into a TX FIFO it will delay processing of the FIFO for the specified amount of time.

Parameters

out	*entry*	pointer to FIFO entry to prepare
in	*time_100_usec*	time to delay processing in 100 microsecond steps

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429TxFifoInterruptCreate()

```
AiReturn Api429TxFifoInterruptCreate (
                struct api429_tx_fifo_entry * entry,
                AiUInt32 tag )
```

Prepares a FIFO entry for generating send notifications.

This function can be used to prepare an 'interrupt' FIFO entry.
The entry can be sent with Api429TxFifoWrite afterwards.
When put into a TX FIFO it will issue an interrupt based notification that can be monitored using Api429ChannelCallbackRegister

Parameters

out	*entry*	pointer to FIFO entry to prepare
in	*tag*	user definable identifier for the interrupt. May range from 0 to 255. Will be encoded in api429_intr_loglist_entry

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429TxFifoReset()

```
AiReturn Api429TxFifoReset (
                AiUInt8 b_Module,
                AiUInt8 b_Chn )
```

This function allows to reset a FIFO.

This function clears the FIFO buffer and stops the FIFO handling. The FIFO handling can be started again with the command Api429ChannelStart.

Parameters

in	*b_Module*	handle to the device
in	*b_Chn*	ID of the channel that shall be affected

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429TxFifoSetup()

```
AiReturn Api429TxFifoSetup (
                AiUInt8 b_Module,
                AiUInt8 b_Chn,
                struct api429_tx_fifo_setup * pxFifoSetup )
```

This function allows to set up a TX FIFO with different values than default.

A FIFO is automatically created if Api429TxInit is called with mode API429_TX_MODE_FIFO.
This function allows to set up a FIFO with other parameters than default.
The data that was written into the FIFO with the command Api429TxFifoDataWordsWrite
or Api429TxFifoWrite will be sent after the command Api429ChannelStart
is called or immediately in case the Api429ChannelStart has already been called for
starting the FIFO ("Tx already active").

Parameters

in	*b_Module*	handle to the device
in	*b_Chn*	ID of the channel that shall be affected
in	*pxFifoSetup*	pointer to a struct containg FIFO set up information

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429TxFifoSetupGet()

```
AiReturn Api429TxFifoSetupGet (
                AiUInt8 b_Module,
                AiUInt8 b_Chn,
                struct api429_tx_fifo_setup * fifo_setup )
```

This function returns the current setup of a TX FIFO.

The requested channel must be setup with API429_TX_MODE_FIFO before this function can be used.

Parameters

in	*b_Module*	handle to the device the channel belongs to
in	*b_Chn*	ID of the channel the TX FIFO is used on
out	*fifo_setup*	pointer to api429_tx_fifo_setup where current TX FIFO setup will be stored

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429TxFifoStatusGet()

```
AiReturn Api429TxFifoStatusGet (
                AiUInt8 b_Module,
                AiUInt8 b_Chn,
                struct api429_tx_fifo_status * pxFifoStatus )
```

This function provides some basic information about the FIFO buffer.

Parameters

in	*b_Module*	handle to the device
in	*b_Chn*	channel that shall be affected
out	*pxFifoStatus*	pointer to a struct containg FIFO status information

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429TxFifoTriggerPulseCreate()

```
AiReturn Api429TxFifoTriggerPulseCreate (
                struct api429_tx_fifo_entry * entry,
                AiUInt32 trigger_line )
```

Prepares a FIFO entry for generating a pulse on an output trigger line.

This function can be used to prepare an 'output trigger' FIFO entry.
The entry can be sent with Api429TxFifoWrite afterwards.
When put into a TX FIFO it will issue a pulse on the specified trigger line

Parameters

out	*entry*	pointer to FIFO entry to prepare
in	*trigger_line*	output trigger line (0..3) to send pulse on

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429TxFifoTriggerWaitCreate()

```
AiReturn Api429TxFifoTriggerWaitCreate (
                struct api429_tx_fifo_entry * entry,
                AiUInt32 input_trigger_line )
```

Prepares a FIFO entry for suspend FIFO processing until input trigger detected.

This function can be used to prepare a 'wait for input trigger' FIFO entry.
The entry can be sent with Api429TxFifoWrite afterwards.
When put into a TX FIFO it will suspend FIFO processing until trigger on specified input line is detected

Parameters

out	*entry*	pointer to FIFO entry to prepare
in	*input_trigger_line*	input trigger line (0..3) to wait for trigger event

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429TxFifoWrite()

```
AiReturn Api429TxFifoWrite (
                AiUInt8 module_handle,
                AiUInt8 channel_id,
                AiUInt32 num_entries,
                struct api429_tx_fifo_entry * entries,
                AiBoolean block,
                AiUInt32 * entries_written )
```

This function is used to write instructions to the FIFO.

This function writes instructions into the FIFO buffer. If the TX channel is started with Api429ChannelStart before or after a call of this function, all FIFO buffer entries will be sent as soon as possible in accordance with the

given interlabel gap (see layout of the FIFO entry). It is recommended to Setup the FIFO and Start the TX channel only once to arm the FIFO so that data written to the FIFO via this function will be sent with minimum latency.

Parameters

in	*module_handle*	handle to the device
in	*channel_id*	ID of the channel that shall be affected
in	*num_entries*	number of entries to write to FIFO
in	*entries*	array containing FIFO entries
in	*block*	block if set to AiTrue, function call will block until all entries were written to FIFO. If set to AiFalse, function will just send the number of entries that currently fit into FIFO.
out	*entries_written*	number of entries that were really written

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

3.11 Version Handling

Data Structures

- struct api429_version
- struct api429_version_info

Typedefs

- typedef struct api429_version TY_API429_VERSION
- typedef struct api429_version_info TY_API429_VERSION_INFO

Functions

- AiReturn Api429VersionGetAll (AiUInt8 uc_Handle, AiUInt32 ul_Count, struct ty_ver_info ax_↩ Versions[ ], struct ty_version_out ∗px_VersionInfoOut)

  This function reads all available versions A count of 0 can be used to obtain the number of available versions in px_VersionInfoOut. The array px_VersionInfoInOut has to be pre allocated by the application. The size must be big enought to hold count items. The array shall be cleared with memset before this function is called.
- AiReturn Api429VersionGet (AiUInt8 uc_Handle, TY_E_VERSION_ID eId, struct ty_ver_info ∗px↩ _Version)

  This function reads a specific version This function returns the version in px_Version. If the version does not exist for the given board, it will return AI429_ERR_ID.
- AiReturn Api429ReadBSPVersionEx (AiUInt8 bModule, struct api429_version_info ∗px_Version↩ Info, AiUInt8 ∗puc_BspCompatibility)

  Legacy version read/compatibility function.

3.11.1 Detailed Description

This module contains functionality related to versioning of on-board firmware and host software components.

3.11.2 Typedef Documentation

TY_API429_VERSION

`TY_API429_VERSION`

Convenience macro for struct api429_version

Definition at line 42 of file Api429Versions.h.

TY_API429_VERSION_INFO

TY_API429_VERSION_INFO

Convenience macro for struct api429_version_info

Definition at line 67 of file Api429Versions.h.

3.11.3 Function Documentation

Api429ReadBSPVersionEx()

```
AiReturn Api429ReadBSPVersionEx (
                AiUInt8 bModule,
                struct api429_version_info * px_VersionInfo,
                AiUInt8 * puc_BspCompatibility )
```

Legacy version read/compatibility function.

This function returns the version numbers of the board software package components for the AIM board.

Parameters

in	*bModule*	handle to board to get versions of
out	*px_VersionInfo*	version info is stored here
out	*puc_BspCompatibility*	Compatibility Status of the current BSP components relating to the library

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Api429VersionGet()

```
AiReturn Api429VersionGet (
```

```
                AiUInt8 uc_Handle,
                TY_E_VERSION_ID eId,
                struct ty_ver_info * px_Version )
```

This function reads a specific version This function returns the version in px_Version. If the version does not exist for the given board, it will return AI429_ERR_ID.

Parameters

in	*uc_Handle*	The module handle.
in	*eId*	The version id to read.
out	*px_Version*	Pointer to store the version info.

Returns

- Zero if the version is valid or AI429_ERR_ID if the version does not exist.
- Appropriate other error code, which may be used as input parameter for Api429LibError↩ DescGet

Api429VersionGetAll()

```
AiReturn Api429VersionGetAll (
                AiUInt8 uc_Handle,
                AiUInt32 ul_Count,
                struct ty_ver_info ax_Versions[],
                struct ty_version_out * px_VersionInfoOut )
```

This function reads all available versions A count of 0 can be used to obtain the number of available versions in px_VersionInfoOut. The array px_VersionInfoInOut has to be pre allocated by the application. The size must be big enought to hold count items. The array shall be cleared with memset before this function is called.

Parameters

in	*uc_Handle*	The module handle.
in	*ul_Count*	Defines how many versions to read.
out	*ax_Versions*	The pre allocated version arry (size=count).
out	*px_VersionInfoOut*	Information about the available and returned versions.

Returns

- API_OK on success
- AI429_ERR_INVALID_MODULE_HANDLE if module handle is invalid

- AI429_ERR_NULL_POINTER if px_VersionInfoOut is NULL or ax_Versions is NULL and ul↩ _Count $>$ 0
- Appropriate other error code, which may be used as input parameter for Api429LibError↩ DescGet

3.12 VxWorks Related Functions

Functions

- AiReturn AiVme429MapModule (struct ty_vme_map_module_in ∗in)

  Map a board.
- AiReturn AiVme429UnmapModule (struct ty_vme_map_module_in ∗in)

  Unmap a board.
- AiReturn Ai429CheckModule (struct ty_vme_map_module_in ∗in)

  Verify the board is handled by this driver library.

3.12.1 Detailed Description

This module contains functionality related to AIM Arinc 429 board handling for VME. Also used for PCI bus handling under VxWorks

3.12.2 Function Documentation

Ai429CheckModule()

```
AiReturn Ai429CheckModule (
              struct ty_vme_map_module_in * in )
```

Verify the board is handled by this driver library.

This command verifies the board can be used with AiVme429MapModule

Parameters

in	*in*	struct that was used to map the board

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

AiVme429MapModule()

```
AiReturn AiVme429MapModule (
              struct ty_vme_map_module_in * in )
```

Map a board.

This command maps a board to a given VME A32 address or to a given PCI address

Parameters

in	struct containing all parameters required for mapping the board
in->ul_A32Addr	physical PCI Bus or VME A32 address of the board
in->ul_A32UserAccess	virtual address, with which the board can be accessed
in->ul_Force	force mapping process - even if already mapped
in->ul_cPCI	has to be set to '1' for cPCI bus
in->px_PCI_Info	the pci info (inluding the PCI header) proivded by either AiPciScan or AiVmeExamineSlot

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

AiVme429UnmapModule()

```
AiReturn AiVme429UnmapModule (
              struct ty_vme_map_module_in * in )
```

Unmap a board.

This command clears the mapping done with AiVme429MapModule

Parameters

in	*in*	struct that was used to map the board

Returns

- API_OK on success
- Appropriate error code, which may be used as input parameter for Api429LibErrorDescGet

Chapter 4

Data Structure Documentation

4.1 _FMListDetails Struct Reference

Data used with FListDir to describe the files listed in a directory.

`#include <extcode.h>`

Data Fields

- int32 flags
- FMFileType type

4.1.1 Detailed Description

Data used with FListDir to describe the files listed in a directory.

Definition at line 903 of file extcode.h.

4.1.2 Field Documentation

flags

`int32 _FMListDetails::flags`

descriptive flags for the file (see kIsFile, et. al. above)

Definition at line 905 of file extcode.h.

type

`FMFileType _FMListDetails::type`

type of the file

Definition at line 906 of file extcode.h.

4.2 api429_board_info Struct Reference

`#include <Api429Board.h>`

Data Fields

- AiUInt32 serial
- AiUInt32 num_channels
- AiUInt32 num_trigger_out
- AiUInt32 num_trigger_in
- AiUInt32 capability_flags
- AiChar name [API429_MAX_BOARDNAME_LEN+1]
- AiChar server_url [API429_MAX_URL_LEN+1]

4.2.1 Detailed Description

This struct is used as output parameter in Api429BoardInfoGet

Definition at line 213 of file Api429Board.h.

4.2.2 Field Documentation

capability_flags

`AiUInt32 api429_board_info::capability_flags`

Special board capability flags. Bit0 indicates if a board has PXI trigger lines. Bit4 indicates if board application type is embedded. Bit12 indicates if board TX inhibit is activated.
Also see API429_BOARD_HAS_PXI_TRIGGER, API429_BOARD_TYPE_IS_EMBEDDED and API429_BOARD_TX_INHIBIT_IS_ACTIVATED

Definition at line 219 of file Api429Board.h.

name

`AiChar api429_board_info::name`

Name of the board

Definition at line 222 of file Api429Board.h.

num_channels

`AiUInt32 api429_board_info::num_channels`

number of Arinc429 channels the board offers

Definition at line 216 of file Api429Board.h.

num_trigger_in

`AiUInt32 api429_board_info::num_trigger_in`

number of input trigger lines the board offers

Definition at line 218 of file Api429Board.h.

num_trigger_out

`AiUInt32 api429_board_info::num_trigger_out`

number of output trigger lines the board offers

Definition at line 217 of file Api429Board.h.

serial

`AiUInt32 api429_board_info::serial`

serial number of board

Definition at line 215 of file Api429Board.h.

server_url

```
AiChar api429_board_info::server_url
```

Server address the board is located on. 'local' for local boards

Definition at line 223 of file Api429Board.h.

4.3 api429_brw Union Reference

`#include <Api429Rm.h>`

Data Fields

- AiUInt32 all
  the complete brw
- struct {
  AiUInt32 gap:8
  gap time
  AiUInt32 biu: 2
  set to one, if received on BIU2.
  AiUInt32 e_type: 5
  AiUInt32 udf: 1
  update flag
  AiUInt32 channel: 4
  channel this entry was received from
  AiUInt32 gsm: 2
  global speed modifier
  AiUInt32 lss: 1
  the low speed selector
  AiUInt32 speed: 1
  set to one if channel is set to high speed
  AiUInt32 hours: 8
  hours of time tag
  } b

  struct to access every bit
- struct {
  AiUInt32 gap:8
  gap time
  AiUInt32 biu: 2
  set to one, if received on BIU2.
  AiUInt32 e_type: 5
  AiUInt32 udf: 1
  update flag
  AiUInt32 channel: 4
  channel this entry was received from
  AiUInt32 gsm: 2
  global speed modifier
  AiUInt32 lss: 1
  the low speed selector

AiUInt32 speed: 1
set to one if channel is set to high speed
AiUInt32 hours: 8
hours of time tag
} b

struct to access every bit

4.3.1 Detailed Description

buffer report word of a receiver monitor buffer entry

Definition at line 294 of file Api429Rm.h.

4.3.2 Field Documentation

e_type

`AiUInt32 api429_brw::e_type`

error type

Binary value	Description
00000b	No error detected
00011b	Bitcount high/low error
00101b	Coding error
01001b	Gap error
10001b	Parity error

Definition at line 301 of file Api429Rm.h.

4.4 api429_channel_info Struct Reference

`#include <Api429Channel.h>`

Data Fields

- AiUInt32 config_flags
- AiUInt32 capability_flags
- enum api429_speed speed
- AiBoolean active

4.4.1 Detailed Description

Output parameter of Api429ChannelInfoGet. Holds current properties of a specific channel may be used for Capability and Configuration test macros (see below)

Definition at line 47 of file Api429Channel.h.

4.4.2 Field Documentation

active

`AiBoolean api429_channel_info::active`

Indicates if channel is active. AiTrue if it is, AiFalse if otherwise

Definition at line 52 of file Api429Channel.h.

capability_flags

`AiUInt32 api429_channel_info::capability_flags`

bit field that holds static capabilities of the channel e.g. if it's able to transmit/receive

Definition at line 50 of file Api429Channel.h.

config_flags

```
AiUInt32 api429_channel_info::config_flags
```

bit-field that indicates current configuration of the channel

Definition at line 49 of file Api429Channel.h.

speed

```
enum api429_speed api429_channel_info::speed
```

Current speed setting of channel.

Definition at line 51 of file Api429Channel.h.

4.5 api429_discr_info Struct Reference

`#include <Api429Discretes.h>`

Data Fields

- AiUInt32 channels
- AiUInt32 canIn
- AiUInt32 canOut

4.5.1 Detailed Description

This structure gives information about the discrete channels

Definition at line 32 of file Api429Discretes.h.

4.5.2 Field Documentation

canIn

`AiUInt32 api429_discr_info::canIn`

bit field that tells, which channels can be set to In

Definition at line 35 of file Api429Discretes.h.

canOut

`AiUInt32 api429_discr_info::canOut`

bit field that tells, which channels can be set to Out

Definition at line 36 of file Api429Discretes.h.

channels

```
AiUInt32 api429_discr_info::channels
```

Number of discrete channels

Definition at line 34 of file Api429Discretes.h.

4.6 api429_discrete_pps_config Struct Reference

`#include <Api429Discretes.h>`

Data Fields

- AiUInt32 ul_DiscreteChannelId
- AiUInt32 ul_EnaDis

4.6.1 Detailed Description

This structure allows to control the PPS signal on a given discrete channel (AyX429 only)

Definition at line 48 of file Api429Discretes.h.

4.6.2 Field Documentation

ul_DiscreteChannelId

`AiUInt32 api429_discrete_pps_config::ul_DiscreteChannelId`

Discrete to send the PPS signal to

Definition at line 50 of file Api429Discretes.h.

ul_EnaDis

`AiUInt32 api429_discrete_pps_config::ul_EnaDis`

if set to '1' enable, if set to '0' disable

Definition at line 51 of file Api429Discretes.h.

4.7 api429_intr_loglist_entry Struct Reference

`#include <Api429Channel.h>`

Data Fields

- AiUInt32 ul_Lla
- AiUInt32 ul_Llb
- union api429_loglist_entry x_Llc
- AiUInt32 ul_Lld

4.7.1 Detailed Description

This structure describes an interrupt log list entry

Definition at line 357 of file Api429Channel.h.

4.7.2 Field Documentation

ul_Lla

`AiUInt32 api429_intr_loglist_entry::ul_Lla`

Log list entry A. Internal data that is used to calculate the channel number and the event type

Definition at line 359 of file Api429Channel.h.

ul_Llb

`AiUInt32 api429_intr_loglist_entry::ul_Llb`

Log list entry B. Depending on the event source this is one of the following:

- LDP: receive operation current label descriptor pointer (for receiver interrupts). This was used to calculate the transfer index and the TX data buffer base address in a complicated way. Please use ul_Llc and ul_Lld instead.

- CMBFP: monitor operation current monitor buffer fill pointer (pointing to the next monitor buffer entry used by the monitor, for monitor interrupts).

- RNBP: replay next buffer pointer, points to the next half buffer address that can be refilled. Used for replay interrupts.

- CILP: transmit operation current instruction list pointer (for transmit interrupts). This was used to calculate the transfer index and the TX data buffer base address in a complicated way. Please use ul_Llc and ul_Lld instead.

Definition at line 361 of file Api429Channel.h.

ul_Lld

```
AiUInt32 api429_intr_loglist_entry::ul_Lld
```

Log list entry D.

- For transmit operation based interrupts (of type API429_EVENT_TX_LABEL and API429_EVE↩ NT_TX_INDEX)
  and for receive operation based interrupts (of type API429_EVENT_RX_ANY_LABEL, API429↩ _EVENT_RX_INDEX and API429_EVENT_RX_ERROR), this value shows the data buffer base address

- For interrupts of type API429_EVENT_TX_HALT this value is undefined and should be ignored.

Definition at line 372 of file Api429Channel.h.

x_Llc

```
union api429_loglist_entry api429_intr_loglist_entry::x_Llc
```

Log list entry C.

- For transmit interrupts (of type API429_EVENT_TX_LABEL and API429_EVENT_TX_INDEX), the lower 24 bits of this value show the transfer index

- For receive operation based interrupts (of type API429_EVENT_RX_ANY_LABEL, API429_EV↩ ENT_RX_INDEX and API429_EVENT_RX_ERROR) the lower 24 bits show the label ID.

- For interrupts of type API429_EVENT_TX_HALT this value is undefined and should be ignored.

Definition at line 367 of file Api429Channel.h.

4.8 api429_loglist_entry Union Reference

`#include <Api429Channel.h>`

Data Fields

- AiUInt32 ul_All
  the complete llc
- struct {
  AiUInt32 uc_IntType:8
  interrupt event source information
  AiUInt32 ul_Info:24
  transfer or label index
  } t
  struct where event source is seperated from the payload
- struct {
  AiUInt32 res:3
  reserved
  AiUInt32 uc_Cmd:1
  event source is command completion interrupt
  AiUInt32 uc_Target:1
  event source is target software
  AiUInt32 uc_Dma:1
  event source is DMA
  AiUInt32 uc_Biu2:1
  event source is BIU 2
  AiUInt32 uc_Biu1:1
  event source is BIU 1
  AiUInt32 ul_Info:24
  transfer or label index
  } b
  struct to access every bit

4.8.1 Detailed Description

Log list entry C

Definition at line 326 of file Api429Channel.h.

4.9 api429_lxfer_dyntag Struct Reference

`#include <Api429Tx.h>`

Data Fields

- AiUInt32 ul_Mode
- AiUInt32 ul_LowerLimit
- AiUInt32 ul_UpperLimit
- AiUInt32 ul_StepSize
- AiUInt32 ul_Mask
- AiUInt32 ul_StartValue

4.9.1 Detailed Description

This structure comprises settings for assigning dynamic data modification to a transfer

Definition at line 151 of file Api429Tx.h.

4.9.2 Field Documentation

ul_LowerLimit

`AiUInt32 api429_lxfer_dyntag::ul_LowerLimit`

The lower limit of the function

Definition at line 160 of file Api429Tx.h.

ul_Mask

`AiUInt32 api429_lxfer_dyntag::ul_Mask`

A bit mask that specifies which bits shall be modified by the dynamic modification.

Definition at line 163 of file Api429Tx.h.

ul_Mode

```
AiUInt32 api429_lxfer_dyntag::ul_Mode
```

Specifies the function to be called.

Value	Description
0	No dynamic operation
1	Positive Ramp
2	Negative ramp

Definition at line 153 of file Api429Tx.h.

ul_StartValue

```
AiUInt32 api429_lxfer_dyntag::ul_StartValue
```

The initial value of the buffer. It can be overwritten using direct memory access or using the Api429Tx↩ XferBufferWrite at any time.

Definition at line 164 of file Api429Tx.h.

ul_StepSize

```
AiUInt32 api429_lxfer_dyntag::ul_StepSize
```

The value that is added to the value after each transmission

Definition at line 162 of file Api429Tx.h.

ul_UpperLimit

```
AiUInt32 api429_lxfer_dyntag::ul_UpperLimit
```

The upper limit of the function

Definition at line 161 of file Api429Tx.h.

4.10 api429_mframe_in Struct Reference

`#include <Api429Tx.h>`

Data Fields

- AiUInt32 ul_FrmId
- AiUInt32 ul_XferCnt
- AiUInt32 * pul_Xfers

4.10.1 Detailed Description

Comprises the input parameters for the Api429TxMinorFrameCreate function

Definition at line 236 of file Api429Tx.h.

4.10.2 Field Documentation

pul_Xfers

`AiUInt32* api429_mframe_in::pul_Xfers`

Transfer Identifiers of the Transfers called within the minor frame (up to 2000, see 'ul_XferCnt')

Definition at line 240 of file Api429Tx.h.

ul_FrmId

`AiUInt32 api429_mframe_in::ul_FrmId`

Frame Identifier. Ranges from 0..1023

Definition at line 238 of file Api429Tx.h.

ul_XferCnt

```
AiUInt32 api429_mframe_in::ul_XferCnt
```

Amount of transfers in frame. Ranges from 0..2000

Definition at line 239 of file Api429Tx.h.

4.11 api429_mframe_out Struct Reference

`#include <Api429Tx.h>`

Data Fields

- AiUInt32 ul_MFrameAddr

4.11.1 Detailed Description

Comprises the output parameters for the Api429TxMinorFrameCreate function

Definition at line 254 of file Api429Tx.h.

4.11.2 Field Documentation

ul_MFrameAddr

`AiUInt32 api429_mframe_out::ul_MFrameAddr`

Byte offset of the minor frame in global memory

Definition at line 256 of file Api429Tx.h.

4.12 api429_rcv_pb_cmd Struct Reference

`#include <Api429Rx.h>`

Data Fields

- AiUInt8 pb_id
- AiUInt8 asc
- AiUInt16 padding1
- AiUInt32 and_mask
- AiUInt32 or_mask
- AiUInt32 xor_mask
- AiUInt32 addsub_val
- AiUInt16 start_delay
- AiUInt16 duration

4.12.1 Detailed Description

Defines pollution of data in a receive/transmit loop. In case of pollution, the following operations are applies to the recevied data word in this order, before sending the data.

- &= and_mask
- |= or_mask
- ^= xor_mask
- if(asc == 0) += addsub_val
  else -= addsub_val

Definition at line 73 of file Api429Rx.h.

4.12.2 Field Documentation

addsub_val

`AiUInt32 api429_rcv_pb_cmd::addsub_val`

ADD/SUB Mask

Definition at line 81 of file Api429Rx.h.

and_mask

```
AiUInt32 api429_rcv_pb_cmd::and_mask
```

AND Mask

Definition at line 78 of file Api429Rx.h.

asc

```
AiUInt8 api429_rcv_pb_cmd::asc
```

Addition/ Subtraction Control. 0 = Addition. 1 = Subtraction

Definition at line 76 of file Api429Rx.h.

duration

```
AiUInt16 api429_rcv_pb_cmd::duration
```

Specifies how many entries are looped/polluted.

Definition at line 83 of file Api429Rx.h.

or_mask

```
AiUInt32 api429_rcv_pb_cmd::or_mask
```

OR Mask

Definition at line 79 of file Api429Rx.h.

padding1

```
AiUInt16 api429_rcv_pb_cmd::padding1
```

reserved

Definition at line 77 of file Api429Rx.h.

pb_id

```
AiUInt8 api429_rcv_pb_cmd::pb_id
```

Pollution Block Identifier - each channel allows id 1 .. 24

Definition at line 75 of file Api429Rx.h.

start_delay

```
AiUInt16 api429_rcv_pb_cmd::start_delay
```

Specifies how many entries are skipped before starting loop/pollution.

Definition at line 82 of file Api429Rx.h.

xor_mask

```
AiUInt32 api429_rcv_pb_cmd::xor_mask
```

XOR Mask

Definition at line 80 of file Api429Rx.h.

4.13 api429_rcv_stack_entry Struct Reference

`#include <Api429Rm.h>`

Data Fields

- AiUInt32 ldata
- union api429_tm_tag tm_tag
- union api429_brw brw

4.13.1 Detailed Description

This structure describes a monitor entry

Definition at line 324 of file Api429Rm.h.

4.13.2 Field Documentation

brw

`union api429_brw api429_rcv_stack_entry::brw`

Buffer Report Word

Definition at line 328 of file Api429Rm.h.

ldata

`AiUInt32 api429_rcv_stack_entry::ldata`

Label data

Definition at line 326 of file Api429Rm.h.

tm_tag

```
union api429_tm_tag api429_rcv_stack_entry::tm_tag
```

32 bit Timetag

Definition at line 327 of file Api429Rm.h.

4.14 api429_rcv_stack_entry_ex Struct Reference

`#include <Api429Rm.h>`

Data Fields

- AiUInt32 ldata
- union api429_tm_tag tm_tag
- union api429_brw brw
- AiUInt32 day_of_year

4.14.1 Detailed Description

This structure describes an extended monitor entry that contains day of year.

Definition at line 344 of file Api429Rm.h.

4.14.2 Field Documentation

brw

`union api429_brw api429_rcv_stack_entry_ex::brw`

Buffer Report Word

Definition at line 348 of file Api429Rm.h.

day_of_year

`AiUInt32 api429_rcv_stack_entry_ex::day_of_year`

The day of the year.

Definition at line 349 of file Api429Rm.h.

ldata

```
AiUInt32 api429_rcv_stack_entry_ex::ldata
```

Label data

Definition at line 346 of file Api429Rm.h.

tm_tag

```
union api429_tm_tag api429_rcv_stack_entry_ex::tm_tag
```

32 bit Timetag

Definition at line 347 of file Api429Rm.h.

4.15 api429_remote_server Struct Reference

`#include <Api429Board.h>`

Data Fields

- AiChar host_name [API429_MAX_URL_LEN+1]
- AiChar os_info [API429_MAX_OS_INFO_LEN+1]
- TY_VER_INFO server_version
- AiUInt32 protocol_major
- AiUInt32 protocol_minor

4.15.1 Detailed Description

struct containing information of a remote server for AIM Arinc 429 devices

Definition at line 230 of file Api429Board.h.

4.15.2 Field Documentation

host_name

`AiChar api429_remote_server::host_name`

Name of the remote server

Definition at line 232 of file Api429Board.h.

os_info

`AiChar api429_remote_server::os_info`

Description of the server's operating system

Definition at line 233 of file Api429Board.h.

protocol_major

```
AiUInt32 api429_remote_server::protocol_major
```

Major number of network protocol used for communication between remote server and local host

Definition at line 235 of file Api429Board.h.

protocol_minor

```
AiUInt32 api429_remote_server::protocol_minor
```

Minor number of network protocol used for communication between remote server and local host

Definition at line 236 of file Api429Board.h.

server_version

```
TY_VER_INFO api429_remote_server::server_version
```

Version of the server application.

Definition at line 234 of file Api429Board.h.

4.16 api429_replay_status Struct Reference

`#include <Api429Replay.h>`

Data Fields

- AiUInt8 uc_Status
- AiUInt8 uc_Padding1
- AiUInt16 uw_Padding2
- AiUInt32 ul_RpiCnt
- AiUInt32 ul_EntryCnt
- AiUInt32 ul_StartAddr
- AiUInt32 ul_Size

4.16.1 Detailed Description

This structure holds current status of a replay transmission

Definition at line 30 of file Api429Replay.h.

4.16.2 Field Documentation

uc_Padding1

`AiUInt8 api429_replay_status::uc_Padding1`

reserved

Definition at line 33 of file Api429Replay.h.

uc_Status

`AiUInt8 api429_replay_status::uc_Status`

API429_HALT or API429_BUSY

Definition at line 32 of file Api429Replay.h.

ul_EntryCnt

```
AiUInt32 api429_replay_status::ul_EntryCnt
```

Initial value calculated from the file size (refer to Api429ReplayInit, parameter 'ul_FileSize') and decremented when replay is started.

Definition at line 38 of file Api429Replay.h.

ul_RpiCnt

```
AiUInt32 api429_replay_status::ul_RpiCnt
```

Actual value of the Half Buffer Transmitted Interrupt counter (incremented each time a Half Buffer Transmitted Interrupt occurs) When the 'ul_RpiCnt' value is incremented, new replay data should be reloaded using Api429ReplayDataWrite'.

Definition at line 35 of file Api429Replay.h.

ul_Size

```
AiUInt32 api429_replay_status::ul_Size
```

Size of a half Replay buffer (in bytes).

Definition at line 43 of file Api429Replay.h.

ul_StartAddr

```
AiUInt32 api429_replay_status::ul_StartAddr
```

Start Address of the the next half Replay buffer that can be written to (the half buffer that is currently not being sent) in the Global RAM area to where the user can copy the replay buffer entries from the Shared RAM area.

Definition at line 40 of file Api429Replay.h.

uw_Padding2

```
AiUInt16 api429_replay_status::uw_Padding2
```

reserved

Definition at line 34 of file Api429Replay.h.

4.17 api429_rm_activity_trigger_def Struct Reference

`#include <Api429Rm.h>`

Data Fields

- AiUInt8 start_pat
- AiUInt8 start_mask
- AiUInt8 stop_pat
- AiUInt8 stop_mask

4.17.1 Detailed Description

This structure describes an activity trigger

Definition at line 204 of file Api429Rm.h.

4.17.2 Field Documentation

start_mask

`AiUInt8 api429_rm_activity_trigger_def::start_mask`

start trigger mask

Definition at line 207 of file Api429Rm.h.

start_pat

`AiUInt8 api429_rm_activity_trigger_def::start_pat`

start trigger pattern

Definition at line 206 of file Api429Rm.h.

stop_mask

```
AiUInt8 api429_rm_activity_trigger_def::stop_mask
```

stop trigger mask

Definition at line 209 of file Api429Rm.h.

stop_pat

```
AiUInt8 api429_rm_activity_trigger_def::stop_pat
```

stop trigger pattern

Definition at line 208 of file Api429Rm.h.

4.18 api429_rm_function_block Struct Reference

`#include <Api429Rm.h>`

Data Fields

- AiUInt8 fb_id
- AiUInt8 fbi
- AiUInt8 ir
- AiUInt8 ulc
- AiUInt8 uli
- AiUInt8 llc
- AiUInt8 lli
- AiUInt8 fe
- AiUInt32 trg_set
- AiUInt32 trg_reset
- AiUInt16 pre_cnt
- AiUInt16 pre_rel
- AiUInt32 mask
- AiUInt32 ulim
- AiUInt32 llim
- AiUInt8 ext_trg
- AiUInt8 trg_line

4.18.1 Detailed Description

This structure describes a monitor function block

Definition at line 167 of file Api429Rm.h.

4.18.2 Field Documentation

ext_trg

`AiUInt8 api429_rm_function_block::ext_trg`

Trigger Output disabled/enabled

Definition at line 190 of file Api429Rm.h.

fb_id

```
AiUInt8 api429_rm_function_block::fb_id
```

ID of function block

Definition at line 169 of file Api429Rm.h.

fbi

```
AiUInt8 api429_rm_function_block::fbi
```

Enable/Disable issue of interrupt when function block result is TRUE

Definition at line 170 of file Api429Rm.h.

fe

```
AiUInt8 api429_rm_function_block::fe
```

Filtering disabled/enabled. When 0, Filtering is disabled. When 1 and the result of the function block operation is "TRUE", the label will be stored in the monitor.

Definition at line 176 of file Api429Rm.h.

ir

```
AiUInt8 api429_rm_function_block::ir
```

if 0: Verify masked bits of a label for not in range.
if 1: Verify masked bits of a label for in range

Definition at line 171 of file Api429Rm.h.

llc

```
AiUInt8 api429_rm_function_block::llc
```

Lower Limit Control. See **ulc**

Definition at line 174 of file Api429Rm.h.

lli

`AiUInt8 api429_rm_function_block::lli`

Lower Limit Control Invert If 'lli' is set to one, the 'lower limit check (llc)' will be logically inverted

Definition at line 175 of file Api429Rm.h.

llim

`AiUInt32 api429_rm_function_block::llim`

Lower Limit

Definition at line 189 of file Api429Rm.h.

mask

`AiUInt32 api429_rm_function_block::mask`

Mask of label data

Definition at line 187 of file Api429Rm.h.

pre_cnt

`AiUInt16 api429_rm_function_block::pre_cnt`

Pre Qualify Counter. Each time the examined label matches the condition defined in the function definition word the pre qualify counter will be decremented by one. If the counter reaches zero the function block operation is TRUE. The pre qualify counter should be initialized to 1.

Definition at line 179 of file Api429Rm.h.

pre_rel

```
AiUInt16 api429_rm_function_block::pre_rel
```

Pre Qualify Counter Register Reinitialization. This field sets the pre qualify counter register reinitialization value for the pre qualify counter. This value will be loaded into the pre qualify counter location each time the pre qualify counter has been decremented to zero. The pre qualify counter reg should be initialized to 1.

Definition at line 182 of file Api429Rm.h.

trg_line

```
AiUInt8 api429_rm_function_block::trg_line
```

External Trigger Output Select

Definition at line 191 of file Api429Rm.h.

trg_reset

```
AiUInt32 api429_rm_function_block::trg_reset
```

If the result of the function block operation is "TRUE" these bits indicate which bit positions have to be RESET in the monitor function trigger status register.

Definition at line 178 of file Api429Rm.h.

trg_set

```
AiUInt32 api429_rm_function_block::trg_set
```

If the result of the function block operation is "TRUE" these bits indicate which bit positions have to be SET in the monitor function trigger status register.

Definition at line 177 of file Api429Rm.h.

ulc

```
AiUInt8 api429_rm_function_block::ulc
```

Upper Limit Control. 0 means **Always**, 1 means **Equal**, 2 means **Greater than**, 3 means **Less than**

Definition at line 172 of file Api429Rm.h.

uli

```
AiUInt8 api429_rm_function_block::uli
```

Upper Limit Control Invert. If 'uli' is set to one, the 'upper limit check (ulc)' will be logically inverted.

Definition at line 173 of file Api429Rm.h.

ulim

```
AiUInt32 api429_rm_function_block::ulim
```

Upper Limit

Definition at line 188 of file Api429Rm.h.

4.19 api429_rm_label_config Struct Reference

`#include <Api429Rm.h>`

Data Fields

- AiUInt32 label_id
- AiUInt32 sdi
- AiBoolean enable

4.19.1 Detailed Description

This structure holds a label/SDI combination that can be enabled/disabled with respect to monitoring

Definition at line 151 of file Api429Rm.h.

4.19.2 Field Documentation

enable

`AiBoolean api429_rm_label_config::enable`

AiTrue if Label shall be enabled for monitoring, AiFalse if disabled.

Definition at line 155 of file Api429Rm.h.

label_id

`AiUInt32 api429_rm_label_config::label_id`

Label ID to configure for monitoring

Definition at line 153 of file Api429Rm.h.

sdi

```
AiUInt32 api429_rm_label_config::sdi
```

SDI extension for Label ID. Only valid if SDI sorting is enabled for channel

Definition at line 154 of file Api429Rm.h.

4.20 api429_rm_setup Struct Reference

`#include <Api429Rm.h>`

Data Fields

- enum api429_rm_mode mode
- AiUInt32 size_in_entries
- AiUInt32 tat_count
- enum api429_rm_interrupt_mode interrupt_mode
- AiInt32 mode
- AiInt32 interrupt_mode

4.20.1 Detailed Description

This structure comprises all properties necessary for a general monitor setup.
It is used as input parameter for Api429RmCreate

Definition at line 105 of file Api429Rm.h.

4.20.2 Field Documentation

interrupt_mode [1/2]

`enum api429_rm_interrupt_mode api429_rm_setup::interrupt_mode`

Used for configuring monitor interrupts. See api429_rm_interrupt_mode

Definition at line 114 of file Api429Rm.h.

interrupt_mode [2/2]

`AiInt32 api429_rm_setup::interrupt_mode`

Used for configuring monitor interrupts. See api429_rm_interrupt_mode

Definition at line 114 of file Api429Rm - LV.h.

mode [1/2]

```
AiInt32 api429_rm_setup::mode
```

Monitoring mode. See api429_rm_mode

Definition at line 107 of file Api429Rm - LV.h.

mode [2/2]

```
enum api429_rm_mode api429_rm_setup::mode
```

Monitoring mode. See api429_rm_mode

Definition at line 107 of file Api429Rm.h.

size_in_entries

```
AiUInt32 api429_rm_setup::size_in_entries
```

Size of monitor buffer in entries.
Use API429_RM_MON_DEFAULT_SIZE for a recommended default size.
Attention: This size has to be specified as a multiple of 256 entries.
If size is 0, a default size of API429_RM_MON_DEFAULT_SIZE will be used

Definition at line 108 of file Api429Rm.h.

tat_count

```
AiUInt32 api429_rm_setup::tat_count
```

Trace after trigger count. Number of entries that are monitored after triggering before monitor stops.
Use API429_RM_CONTINUOUS_CAPTURE for continuous monitoring without stopping except on stop trigger signals

Definition at line 112 of file Api429Rm.h.

4.21 api429_rm_trigger_setup Struct Reference

`#include <Api429Rm.h>`

Data Fields

- enum api429_rm_trigger_mode mode
- AiUInt8 input_trigger_line
- AiBoolean generate_output_strobe
- AiUInt8 output_trigger_line

4.21.1 Detailed Description

This structure comprises all properties related to receive monitor trigger set-up.
It is used as input parameter for Api429RmTriggerConfigSet

Definition at line 129 of file Api429Rm.h.

4.21.2 Field Documentation

generate_output_strobe

`AiBoolean api429_rm_trigger_setup::generate_output_strobe`

If set to AiTrue, a strobe is generated on output trigger line defined by 'output_trigger_line'

Definition at line 133 of file Api429Rm.h.

input_trigger_line

`AiUInt8 api429_rm_trigger_setup::input_trigger_line`

ID of input trigger line to react on when using 'mode' API429_TRG_EXT.

Definition at line 132 of file Api429Rm.h.

mode

```
enum api429_rm_trigger_mode api429_rm_trigger_setup::mode
```

trigger mode to use on a specific receive monitor. See api429_rm_trigger_mode

Definition at line 131 of file Api429Rm.h.

output_trigger_line

```
AiUInt8 api429_rm_trigger_setup::output_trigger_line
```

ID of output trigger line that strobe is generated on when 'generate_output_strobe' is set to AiTrue

Definition at line 135 of file Api429Rm.h.

4.22 api429_rx_activity Struct Reference

`#include <Api429Rx.h>`

Data Fields

- AiUInt32 ChannelActivity [32][8]

4.22.1 Detailed Description

Rx Activity structure

Definition at line 51 of file Api429Rx.h.

4.22.2 Field Documentation

ChannelActivity

`AiUInt32 api429_rx_activity::ChannelActivity[32][8]`

Contains eight long words for each channel. The eight long words are a bitfield, that shows for which labels data was received for a specific channel

Definition at line 53 of file Api429Rx.h.

4.23 api429_rx_buf_ctl Struct Reference

`#include <Api429Rx.h>`

Data Fields

- AiUInt16 ixw
- AiUInt16 inr
- AiUInt16 ci

4.23.1 Detailed Description

This structures describes a buffer header for label receive buffers

Definition at line 145 of file Api429Rx.h.

4.23.2 Field Documentation

ci

`AiUInt16 api429_rx_buf_ctl::ci`

Current Index. Changes when a label was received

Definition at line 149 of file Api429Rx.h.

inr

`AiUInt16 api429_rx_buf_ctl::inr`

Index Reload

Definition at line 148 of file Api429Rx.h.

ixw

```
AiUInt16 api429_rx_buf_ctl::ixw
```

Interrupt Index. Only important, if bit 2 of b_IrCon of Api429RxLabelConfigure was set.

Definition at line 147 of file Api429Rx.h.

4.24 api429_rx_buf_entry Struct Reference

`#include <Api429Rx.h>`

Data Fields

- AiUInt32 lab_data

4.24.1 Detailed Description

This structures describes one label receive buffer entry

Definition at line 128 of file Api429Rx.h.

4.24.2 Field Documentation

lab_data

`AiUInt32 api429_rx_buf_entry::lab_data`

Arinc 429 data word. The lowest byte of each data word is the label id. To meet the ARINC 429 specification the label id is automatically transferred in reverse bit order.

Definition at line 130 of file Api429Rx.h.

4.25 api429_rx_frame_response_setup Struct Reference

`#include <Api429Rx.h>`

Data Fields

- AiUInt8 tx_channel
- AiUInt32 tx_acyc_frame_id
- AiUInt32 compare_mask
- AiUInt32 compare_value

4.25.1 Detailed Description

Comprises a setup for a frame based receive response.

Definition at line 32 of file Api429Rx.h.

4.25.2 Field Documentation

compare_mask

`AiUInt32 api429_rx_frame_response_setup::compare_mask`

Received data word will be combined by a logical AND operation with this mask

Definition at line 36 of file Api429Rx.h.

compare_value

`AiUInt32 api429_rx_frame_response_setup::compare_value`

The masked receive data will be compared with this value. If matching, the respond sending will be initiated

Definition at line 37 of file Api429Rx.h.

tx_acyc_frame_id

```
AiUInt32 api429_rx_frame_response_setup::tx_acyc_frame_id
```

ID of acyclic minor frame to respond

Definition at line 35 of file Api429Rx.h.

tx_channel

```
AiUInt8 api429_rx_frame_response_setup::tx_channel
```

ID of transmit channel to send response on

Definition at line 34 of file Api429Rx.h.

4.26 api429_rx_label_setup Struct Reference

`#include <Api429Rx.h>`

Data Fields

- AiUInt32 label
- AiUInt32 sdi
- AiUInt32 con
- AiUInt32 irCon
- AiUInt32 irIndex
- AiUInt32 bufSize

4.26.1 Detailed Description

This structure contains parameters for configuring reception of specific labels

Definition at line 98 of file Api429Rx.h.

4.26.2 Field Documentation

bufSize

`AiUInt32 api429_rx_label_setup::bufSize`

Label Receive Buffer Size. Valid from 1 to 1023

Definition at line 115 of file Api429Rx.h.

con

`AiUInt32 api429_rx_label_setup::con`

0 = disable reception of label. 1 = enable reception of label

Definition at line 104 of file Api429Rx.h.

irCon

```
AiUInt32 api429_rx_label_setup::irCon
```

Label Interrupt Control.
Bit-field that allows to enable/disable generation of interrupts on various label events.
Setting a bit to *1* will enable, setting to *0* disable interrupt generation on the corresponding event.

Bit Position	Description
0	Interrupt when an error is received on a label
1	Interrupt each time a label is received
2	Interrupt when the buffer index passes the Buffer Interrupt Index (irIndex)

Definition at line 105 of file Api429Rx.h.

irIndex

```
AiUInt32 api429_rx_label_setup::irIndex
```

Buffer Interrupt Index. Valid from 1 to 1023, must be a valid buffer index (irIndex <= bufSize)

Definition at line 114 of file Api429Rx.h.

label

```
AiUInt32 api429_rx_label_setup::label
```

Label Number

Definition at line 100 of file Api429Rx.h.

sdi

```
AiUInt32 api429_rx_label_setup::sdi
```

SDI Number. When set to **4**, all received data will be stored in one buffer. The latest received data will be stored in the buffer. This value is ignored if sdi sorting is disabled in Api429RxInit by setting sdi_enabled to AiFalse

Definition at line 101 of file Api429Rx.h.

4.27 api429_time Struct Reference

`#include <Api429Board.h>`

Data Fields

- AiUInt32 day
- AiUInt32 hour
- AiUInt32 minute
- AiUInt32 second
- AiUInt32 millisecond

4.27.1 Detailed Description

This structure describes a specific time point, as it can be handled by Arinc 429 boards

Definition at line 190 of file Api429Board.h.

4.27.2 Field Documentation

day

`AiUInt32 api429_time::day`

Day of year. Starting with 1

Definition at line 192 of file Api429Board.h.

hour

`AiUInt32 api429_time::hour`

Hour

Definition at line 193 of file Api429Board.h.

millisecond

```
AiUInt32 api429_time::millisecond
```

Milliseconds

Definition at line 196 of file Api429Board.h.

minute

```
AiUInt32 api429_time::minute
```

Minute

Definition at line 194 of file Api429Board.h.

second

```
AiUInt32 api429_time::second
```

Second

Definition at line 195 of file Api429Board.h.

4.28 api429_tm_tag Union Reference

`#include <Api429Rm.h>`

Data Fields

- AiUInt32 all
  the complete tm_tag
- struct {
  AiUInt32 minutes: 6
  minutes of time tag
  AiUInt32 seconds: 6
  seconds of time tag
  AiUInt32 microseconds: 20
  microseconds of time tag
  } b
  struct to access every bit
- struct {
  AiUInt32 minutes: 6
  minutes of time tag
  AiUInt32 seconds: 6
  seconds of time tag
  AiUInt32 microseconds: 20
  microseconds of time tag
  } b
  struct to access every bit

4.28.1 Detailed Description

time tag word of a receiver monitor buffer entry

Definition at line 253 of file Api429Rm.h.

4.29 api429_trg_ctl_cmd Struct Reference

`#include <Api429Rm.h>`

Data Fields

- AiUInt8 low_func
- AiUInt8 up_func
- AiUInt16 padding1
- AiUInt32 mask
- AiUInt32 low_limit
- AiUInt32 up_limit

4.29.1 Detailed Description

This structure allows to set the trigger

Definition at line 221 of file Api429Rm.h.

4.29.2 Field Documentation

low_func

`AiUInt8 api429_trg_ctl_cmd::low_func`

reserved for later extension should be set to 0

Definition at line 223 of file Api429Rm.h.

low_limit

`AiUInt32 api429_trg_ctl_cmd::low_limit`

Monitor Trigger Lower Limit

Definition at line 227 of file Api429Rm.h.

mask

```
AiUInt32 api429_trg_ctl_cmd::mask
```

Monitor Trigger Mask

Definition at line 226 of file Api429Rm.h.

padding1

```
AiUInt16 api429_trg_ctl_cmd::padding1
```

reserved

Definition at line 225 of file Api429Rm.h.

up_func

```
AiUInt8 api429_trg_ctl_cmd::up_func
```

Upper Limit Start Trigger Function Mode. 0 means **Equal**, 1 means **Not equal**, 2 means **Less than**, 3 means **Greater than**

Definition at line 224 of file Api429Rm.h.

up_limit

```
AiUInt32 api429_trg_ctl_cmd::up_limit
```

Monitor Trigger Upper Limit

Definition at line 228 of file Api429Rm.h.

4.30 api429_tx_fifo_entry Struct Reference

`#include <Api429TxFifo.h>`

Data Fields

- AiUInt32 ulControl
- AiUInt32 ulData

4.30.1 Detailed Description

This structure describes the generic TX FIFO entry format. it is filled with functions like Api429TxFifo↩ DataWordCreate

Definition at line 67 of file Api429TxFifo.h.

4.30.2 Field Documentation

ulControl

`AiUInt32 api429_tx_fifo_entry::ulControl`

control word

Definition at line 69 of file Api429TxFifo.h.

ulData

`AiUInt32 api429_tx_fifo_entry::ulData`

data word

Definition at line 70 of file Api429TxFifo.h.

4.31 api429_tx_fifo_setup Struct Reference

`#include <Api429TxFifo.h>`

Data Fields

- AiUInt32 ulFifoSize
- AiUInt32 ulFifoControl
- AiUInt32 ulDefaultGapSize

4.31.1 Detailed Description

This structure comprises settings when generating a TX FIFO

Definition at line 32 of file Api429TxFifo.h.

4.31.2 Field Documentation

ulDefaultGapSize

`AiUInt32 api429_tx_fifo_setup::ulDefaultGapSize`

Inter message gap size, with a valid range of 4..255. The default value is 4.

Definition at line 36 of file Api429TxFifo.h.

ulFifoControl

`AiUInt32 api429_tx_fifo_setup::ulFifoControl`

Bitfield defines Interrupt handling. The default value is 0. If bit zero is set a notification interrupt is sent when the FIFO gets empty.

Definition at line 35 of file Api429TxFifo.h.

ulFifoSize

```
AiUInt32 api429_tx_fifo_setup::ulFifoSize
```

Size of FIFO in number of entries. The default value is 1024.

Definition at line 34 of file Api429TxFifo.h.

4.32 api429_tx_fifo_status Struct Reference

`#include <Api429TxFifo.h>`

Data Fields

- AiUInt32 ulEntriesToSend
- AiUInt32 ulEntriesFree

4.32.1 Detailed Description

This structure describes the current state of a TX FIFO

Definition at line 49 of file Api429TxFifo.h.

4.32.2 Field Documentation

ulEntriesFree

`AiUInt32 api429_tx_fifo_status::ulEntriesFree`

number of entries before fifo will be full

Definition at line 52 of file Api429TxFifo.h.

ulEntriesToSend

`AiUInt32 api429_tx_fifo_status::ulEntriesToSend`

number of entries still in fifo

Definition at line 51 of file Api429TxFifo.h.

4.33 api429_version Struct Reference

`#include <Api429Versions.h>`

Data Fields

- AiUInt32 ul_MajorVer
- AiUInt32 ul_MinorVer
- AiUInt32 ul_BuildNr
- AiUInt32 ul_MajorSpecialVer
- AiUInt32 ul_MinorSpecialVer

4.33.1 Detailed Description

This struct contains the version of a given BSP component

Definition at line 31 of file Api429Versions.h.

4.33.2 Field Documentation

ul_BuildNr

`AiUInt32 api429_version::ul_BuildNr`

The build number of the component

Definition at line 35 of file Api429Versions.h.

ul_MajorSpecialVer

`AiUInt32 api429_version::ul_MajorSpecialVer`

Only used in special cases

Definition at line 36 of file Api429Versions.h.

ul_MajorVer

```
AiUInt32 api429_version::ul_MajorVer
```

The major version number of the component

Definition at line 33 of file Api429Versions.h.

ul_MinorSpecialVer

```
AiUInt32 api429_version::ul_MinorSpecialVer
```

Only used in special cases

Definition at line 37 of file Api429Versions.h.

ul_MinorVer

```
AiUInt32 api429_version::ul_MinorVer
```

The minor version number of the component

Definition at line 34 of file Api429Versions.h.

4.34 api429_version_info Struct Reference

`#include <Api429Versions.h>`

Data Fields

- struct api429_version x_TcpVer
- struct api429_version x_PciLcaVer
- struct api429_version x_AslLcaVer
- struct api429_version x_IoLcaBiu1Ver
- struct api429_version x_IoLcaBiu2Ver
- struct api429_version x_FirmwareBiu1Ver
- struct api429_version x_FirmwareBiu2Ver
- struct api429_version x_TargetSWVer
- struct api429_version x_SysDrvVer
- struct api429_version x_DllVer
- struct api429_version x_VmeGeneric
- struct api429_version x_MonitorVer
- AiUInt32 ul_BoardSerialNr

4.34.1 Detailed Description

This structure contains several components of the BSP

Definition at line 48 of file Api429Versions.h.

4.34.2 Field Documentation

ul_BoardSerialNr

`AiUInt32 api429_version_info::ul_BoardSerialNr`

Serial number of the board

Definition at line 62 of file Api429Versions.h.

x_AslLcaVer

`struct api429_version api429_version_info::x_AslLcaVer`

Version of the ASL FPGA (AyX429 only)

Definition at line 52 of file Api429Versions.h.

x_DllVer

`struct api429_version api429_version_info::x_DllVer`

Version of the library

Definition at line 59 of file Api429Versions.h.

x_FirmwareBiu1Ver

`struct api429_version api429_version_info::x_FirmwareBiu1Ver`

Version of the firmware on BIU1

Definition at line 55 of file Api429Versions.h.

x_FirmwareBiu2Ver

`struct api429_version api429_version_info::x_FirmwareBiu2Ver`

Version of the firmware on BIU2

Definition at line 56 of file Api429Versions.h.

x_IoLcaBiu1Ver

`struct api429_version api429_version_info::x_IoLcaBiu1Ver`

Version of the ARINC 429 IO FPGA on BIU1

Definition at line 53 of file Api429Versions.h.

x_IoLcaBiu2Ver

`struct api429_version api429_version_info::x_IoLcaBiu2Ver`

Version of the ARINC 429 IO FPGA on BIU2

Definition at line 54 of file Api429Versions.h.

x_MonitorVer

`struct api429_version api429_version_info::x_MonitorVer`

Version of the included monitor

Definition at line 61 of file Api429Versions.h.

x_PciLcaVer

`struct api429_version api429_version_info::x_PciLcaVer`

Version of the PCI FPGA

Definition at line 51 of file Api429Versions.h.

x_SysDrvVer

`struct api429_version api429_version_info::x_SysDrvVer`

Version of the device driver

Definition at line 58 of file Api429Versions.h.

x_TargetSWVer

`struct api429_version api429_version_info::x_TargetSWVer`

Version of the target software

Definition at line 57 of file Api429Versions.h.

x_TcpVer

`struct api429_version api429_version_info::x_TcpVer`

Version of the time code processor

Definition at line 50 of file Api429Versions.h.

x_VmeGeneric

`struct api429_version api429_version_info::x_VmeGeneric`

Version of the generic part of VME systems

Definition at line 60 of file Api429Versions.h.

4.35 api429_xfer Struct Reference

`#include <Api429Tx.h>`

Data Fields

- AiUInt32 xfer_id
- enum api429_xfer_type xfer_type
- enum api429_xfer_error err_type
- AiUInt16 xfer_gap
- AiUInt16 ir_index
- AiUInt16 buf_size
- AiUInt8 xfer_ir

4.35.1 Detailed Description

This structure describes a transfer set-up for framing/rate-controlled based transmit channels

Definition at line 96 of file Api429Tx.h.

4.35.2 Field Documentation

buf_size

`AiUInt16 api429_xfer::buf_size`

Label data buffer size. Range from 1 to 1023

Definition at line 111 of file Api429Tx.h.

err_type

`enum api429_xfer_error api429_xfer::err_type`

Error type. See api429_xfer_error

Definition at line 100 of file Api429Tx.h.

ir_index

```
AiUInt16 api429_xfer::ir_index
```

Transfer Interrupt Index. Range from 1 to 1023

Definition at line 110 of file Api429Tx.h.

xfer_gap

```
AiUInt16 api429_xfer::xfer_gap
```

Label Transfer Gap

Transfer Type	Range	Description
1,8	4..255	ARINC bits
2	0	Reserved
4	0..3	Output trigger line
7	0..255	Amount of 100us steps to wait
9	0..3	Input Trigger Line

Definition at line 101 of file Api429Tx.h.

xfer_id

```
AiUInt32 api429_xfer::xfer_id
```

Transfer Identifier. Not the label ID!!! Each transfer must be created with an ID unique to this channel, ranging from **1** to **1023**

Definition at line 98 of file Api429Tx.h.

xfer_ir

```
AiUInt8 api429_xfer::xfer_ir
```

Interrupt settings

Bit Position	Description
0	Interrupt on Label transfer or NOP execution
1	Interrupt when transfer interrupt index reached and on Buffer index reload

Definition at line 112 of file Api429Tx.h.

xfer_type

```
enum api429_xfer_type api429_xfer::xfer_type
```

Transfer Type. See api429_xfer_type

Definition at line 99 of file Api429Tx.h.

4.36 api429_xfer_data Struct Reference

`#include <Api429Tx.h>`

Data Fields

- AiUInt32 ul_XferData
- AiUInt32 ul_XferTTHigh
- AiUInt32 ul_XferTTLow

4.36.1 Detailed Description

This structure comprises properties of a specific transfer

Definition at line 180 of file Api429Tx.h.

4.36.2 Field Documentation

ul_XferData

`AiUInt32 api429_xfer_data::ul_XferData`

Transfer data word

Definition at line 182 of file Api429Tx.h.

ul_XferTTHigh

`AiUInt32 api429_xfer_data::ul_XferTTHigh`

Time tag high word of the current sent TX label transfer TODO: Document it

Definition at line 183 of file Api429Tx.h.

ul_XferTTLow

```
AiUInt32 api429_xfer_data::ul_XferTTLow
```

Time tag low word of the current sent TX label transfer TODO: Document it

Definition at line 184 of file Api429Tx.h.

4.37 api429_xfer_data_read_input Struct Reference

`#include <Api429Tx.h>`

Data Fields

- AiUInt32 ul_XferId
- AiUInt32 ul_BufStart
- AiUInt32 ul_BufSize
- AiUInt32 ul_Mode

4.37.1 Detailed Description

This structure comprises the input parameters for Api429TxXferBufferRead

Definition at line 217 of file Api429Tx.h.

4.37.2 Field Documentation

ul_BufSize

`AiUInt32 api429_xfer_data_read_input::ul_BufSize`

Amount of data words to read from label transmit buffer.

Definition at line 221 of file Api429Tx.h.

ul_BufStart

`AiUInt32 api429_xfer_data_read_input::ul_BufStart`

Buffer index to start reading from. Ranges from 0..1023

Definition at line 220 of file Api429Tx.h.

ul_Mode

```
AiUInt32 api429_xfer_data_read_input::ul_Mode
```

If 0: no action. If API_TXRXBUF_TT_UPDATE_CLEAR : Clear update flag in TT High word after reading the data

Definition at line 222 of file Api429Tx.h.

ul_XferId

```
AiUInt32 api429_xfer_data_read_input::ul_XferId
```

ID of transfer to read data from

Definition at line 219 of file Api429Tx.h.

4.38 api429_xfer_info Struct Reference

`#include <Api429Tx.h>`

Data Fields

- AiUInt32 ul_XferIx
- AiUInt32 ul_XferCnt
- AiUInt32 ul_XferData
- AiUInt32 ul_XferTTHigh
- AiUInt32 ul_XferTTLow

4.38.1 Detailed Description

Comprises information about send status of a transfer

Definition at line 197 of file Api429Tx.h.

4.38.2 Field Documentation

ul_XferCnt

`AiUInt32 api429_xfer_info::ul_XferCnt`

Number of times the transfer was sent

Definition at line 200 of file Api429Tx.h.

ul_XferData

`AiUInt32 api429_xfer_info::ul_XferData`

Last sent data word

Definition at line 201 of file Api429Tx.h.

ul_XferIx

```
AiUInt32 api429_xfer_info::ul_XferIx
```

Current Label transmit buffer index

Definition at line 199 of file Api429Tx.h.

ul_XferTTHigh

```
AiUInt32 api429_xfer_info::ul_XferTTHigh
```

Time tag high word of the current sent TX label transfer

Definition at line 202 of file Api429Tx.h.

ul_XferTTLow

```
AiUInt32 api429_xfer_info::ul_XferTTLow
```

Time tag low word of the current sent TX label transfer

Definition at line 203 of file Api429Tx.h.

4.39 api429_xfer_out Struct Reference

`#include <Api429Tx.h>`

Data Fields

- AiUInt32 ul_Status
- AiUInt32 ul_FreeMem
- AiUInt32 ul_XferDescAddr

4.39.1 Detailed Description

This structure comprises output parameters when transfer is created

Definition at line 132 of file Api429Tx.h.

4.39.2 Field Documentation

ul_FreeMem

`AiUInt32 api429_xfer_out::ul_FreeMem`

Size of the free memory available for label buffer

Definition at line 135 of file Api429Tx.h.

ul_Status

`AiUInt32 api429_xfer_out::ul_Status`

If 0: Label buffer has been allocated. If 1: Label buffer could not be allocated due to too little memory

Definition at line 134 of file Api429Tx.h.

ul_XferDescAddr

```
AiUInt32 api429_xfer_out::ul_XferDescAddr
```

Address of the transfer descriptor, relative to the start of global memory.

Definition at line 136 of file Api429Tx.h.

4.40 cmplx128 Struct Reference

Complex double-precision floating point number.

`#include <fundtypes.h>`

Data Fields

- float64 **re**
- float64 **im**

4.40.1 Detailed Description

Complex double-precision floating point number.

Definition at line 526 of file fundtypes.h.

4.41 cmplx64 Struct Reference

Complex single-precision floating point number.

`#include <fundtypes.h>`

Data Fields

- float32 **re**
- float32 **im**

4.41.1 Detailed Description

Complex single-precision floating point number.

Definition at line 517 of file fundtypes.h.

4.42 cmplxExt Struct Reference

Complex extended-precision floating point number.

`#include <fundtypes.h>`

Data Fields

- floatExt **re**
- floatExt **im**

4.42.1 Detailed Description

Complex extended-precision floating point number.

Definition at line 535 of file fundtypes.h.

4.43 CPStr Struct Reference

Concatenated Pascal string types.

`#include <extcode.h>`

Data Fields

- int32 **cnt**
- uChar **str** [1]

4.43.1 Detailed Description

Concatenated Pascal string types.

Definition at line 243 of file extcode.h.

4.44 DateRec Struct Reference

Date/time record.

`#include <extcode.h>`

Data Fields

- int32 **sec**
- int32 **min**
- int32 **hour**
- int32 **mday**
- int32 **mon**
- int32 **year**
- int32 **wday**
- int32 **yday**
- int32 **isdst**

4.44.1 Detailed Description

Date/time record.

Definition at line 757 of file extcode.h.

4.45 FInfoRec Struct Reference

Descriptive information about a file.

`#include <extcode.h>`

Data Fields

- FMFileType type
- FMFileCreator creator
- int32 permissions
- int32 size
- int32 rfSize
- uInt32 cdate
- uInt32 mdate
- Bool32 folder
- Bool32 isInvisible
- struct {
  int16 v
  int16 h
  } location

- Str255 owner
- Str255 group

4.45.1 Detailed Description

Descriptive information about a file.

Definition at line 925 of file extcode.h.

4.45.2 Field Documentation

cdate

`uInt32 FInfoRec::cdate`

creation date

Definition at line 931 of file extcode.h.

creator

```
FMFileCreator FInfoRec::creator
```

system specific file creator– 0 for directories

Definition at line 927 of file extcode.h.

folder

```
Bool32 FInfoRec::folder
```

indicates whether path refers to a folder

Definition at line 933 of file extcode.h.

group

```
Str255 FInfoRec::group
```

group (in pascal string form) of file or folder (Mac, Linux only)

Definition at line 940 of file extcode.h.

h

```
int16 FInfoRec::h
```

horizontal coordinate

Definition at line 937 of file extcode.h.

isInvisible

```
Bool32 FInfoRec::isInvisible
```

indicates whether the file is visible in File Dialog

Definition at line 934 of file extcode.h.

location

```
struct { ... } FInfoRec::location
```

system specific geographical location (on Mac only)

mdate

```
uInt32 FInfoRec::mdate
```

last modification date

Definition at line 932 of file extcode.h.

owner

```
Str255 FInfoRec::owner
```

owner (in pascal string form) of file or folder

Definition at line 939 of file extcode.h.

permissions

```
int32 FInfoRec::permissions
```

system specific file access rights

Definition at line 928 of file extcode.h.

rfSize

```
int32 FInfoRec::rfSize
```

resource fork size (on Mac only)

Definition at line 930 of file extcode.h.

size

```
int32 FInfoRec::size
```

file size in bytes (data fork on Mac) or entries in folder

Definition at line 929 of file extcode.h.

type

```
FMFileType FInfoRec::type
```

system specific file type– 0 for directories

Definition at line 926 of file extcode.h.

v

```
int16 FInfoRec::v
```

vertical coordinate

Definition at line 936 of file extcode.h.

4.46 FInfoRec64 Struct Reference

Descriptive information about a file, with large file support.

`#include <extcode.h>`

Data Fields

- FMFileType type
- FMFileCreator creator
- int32 permissions
- int64 size
- int64 rfSize
- uInt32 cdate
- uInt32 mdate
- Bool32 folder
- Bool32 isInvisible
- struct {
  int16 v
  int16 h
  } location

- Str255 owner
- Str255 group

4.46.1 Detailed Description

Descriptive information about a file, with large file support.

Definition at line 988 of file extcode.h.

4.46.2 Field Documentation

cdate

`uInt32 FInfoRec64::cdate`

creation date

Definition at line 994 of file extcode.h.

creator

```
FMFileCreator FInfoRec64::creator
```

system specific file creator– 0 for directories

Definition at line 990 of file extcode.h.

folder

```
Bool32 FInfoRec64::folder
```

indicates whether path refers to a folder

Definition at line 996 of file extcode.h.

group

```
Str255 FInfoRec64::group
```

group (in pascal string form) of file or folder (Mac, Linux only)

Definition at line 1003 of file extcode.h.

h

```
int16 FInfoRec64::h
```

horizontal coordinate

Definition at line 1000 of file extcode.h.

isInvisible

```
Bool32 FInfoRec64::isInvisible
```

indicates whether the file is visible in File Dialog

Definition at line 997 of file extcode.h.

location

```
struct { ... } FInfoRec64::location
```

system specific geographical location (on Mac only)

mdate

```
uInt32 FInfoRec64::mdate
```

last modification date

Definition at line 995 of file extcode.h.

owner

```
Str255 FInfoRec64::owner
```

owner (in pascal string form) of file or folder

Definition at line 1002 of file extcode.h.

permissions

```
int32 FInfoRec64::permissions
```

system specific file access rights

Definition at line 991 of file extcode.h.

rfSize

```
int64 FInfoRec64::rfSize
```

resource fork size (on Mac only)

Definition at line 993 of file extcode.h.

size

`int64 FInfoRec64::size`

file size in bytes (data fork on Mac) or entries in folder

Definition at line 992 of file extcode.h.

type

`FMFileType FInfoRec64::type`

system specific file type– 0 for directories

Definition at line 989 of file extcode.h.

v

`int16 FInfoRec64::v`

vertical coordinate

Definition at line 999 of file extcode.h.

4.47 LStr Struct Reference

Long Pascal-style string types.

`#include <extcode.h>`

Data Fields

- int32 **cnt**
- uChar **str** [1]

4.47.1 Detailed Description

Long Pascal-style string types.

Definition at line 249 of file extcode.h.

4.48 MemStatRec Struct Reference

Describes memory statistics.

`#include <extcode.h>`

Data Fields

- uInt64 **totFreeSize**
- uInt64 **maxFreeSize**
- int32 **unused1**
- size_t **totAllocSize**
- size_t **unused2**
- int32 **unused3**
- int32 **unused4**
- int32 **unused5**
- uInt64 **totPhysSize**
- int32 **reserved1**
- int32 **reserved2**

4.48.1 Detailed Description

Describes memory statistics.

Definition at line 812 of file extcode.h.

4.49 VInfoRec Struct Reference

Data structure describing a disk volume.

`#include <extcode.h>`

Data Fields

- uInt32 size
- uInt32 used
- uInt32 free

4.49.1 Detailed Description

Data structure describing a disk volume.

Definition at line 1424 of file extcode.h.

4.49.2 Field Documentation

free

`uInt32 VInfoRec::free`

number of bytes available for use on volume

Definition at line 1427 of file extcode.h.

size

`uInt32 VInfoRec::size`

size in bytes of a volume

Definition at line 1425 of file extcode.h.

used

```
uInt32 VInfoRec::used
```

number of bytes used on volume

Definition at line 1426 of file extcode.h.

Index

_FMListDetails, 156
    flags, 156
    type, 156

API429_BOARD_CAP_PXI_TRG_FLAG
    Board Related Functions, 15
API429_BOARD_HAS_PXI_TRIGGER
    Board Related Functions, 16
API429_BOARD_TX_INHIBIT_FLAG
    Board Related Functions, 16
API429_BOARD_TX_INHIBIT_IS_ACTIVATED
    Board Related Functions, 17
API429_BOARD_TYPE_EMBEDDED_FLAG
    Board Related Functions, 17
API429_BOARD_TYPE_IS_EMBEDDED
    Board Related Functions, 18
API429_CHANNEL_CALLBACK
    Channel Handling, 61
API429_CHANNEL_CAN_RECEIVE
    Channel Handling, 53
API429_CHANNEL_CAN_TRANSMIT
    Channel Handling, 54
API429_CHANNEL_HAS_VARIABLE_AMPLITU↩
        DE
    Channel Handling, 54
API429_CHANNEL_IN_GLOBAL_MONITORING↩
        _MODE
    Channel Handling, 54
API429_CHANNEL_IN_LOCAL_MONITORING_↩
        MODE
    Channel Handling, 54
API429_CHANNEL_IN_MONITORING_MODE
    Channel Handling, 54
API429_CHANNEL_IN_PHYS_REPLAY_MODE
    Channel Handling, 55
API429_CHANNEL_IN_REPLAY_MODE
    Channel Handling, 55
API429_CHANNEL_IN_RX_LP_MODE
    Channel Handling, 55
API429_CHANNEL_IN_RX_MODE
    Channel Handling, 55
API429_CHANNEL_IN_RX_NORMAL_MODE
    Channel Handling, 56
API429_CHANNEL_IN_RX_TX_MIXING_MODE
    Channel Handling, 56
API429_CHANNEL_IN_TX_DYNTAG_MODE
    Channel Handling, 56
API429_CHANNEL_IN_TX_FIFO_MODE
    Channel Handling, 56
API429_CHANNEL_IN_TX_FRAMING_MODE
    Channel Handling, 57
API429_CHANNEL_IN_TX_LP_MODE
    Channel Handling, 57
API429_CHANNEL_IN_TX_MODE
    Channel Handling, 57
API429_CHANNEL_IN_TX_RATE_CONTROLLE↩
        D_MODE
    Channel Handling, 57
API429_CHN_CAP_RX_FLAG
    Channel Handling, 58
API429_CHN_CAP_TX_FLAG
    Channel Handling, 58
API429_CHN_CAP_VAR_AMP_FLAG
    Channel Handling, 58
API429_CHN_CFG_PARITY_ENABLED_FLAG
    Channel Handling, 58
API429_CHN_CFG_PHYS_REPLAY_FLAG
    Channel Handling, 59
API429_CHN_CFG_RM_GLOBAL_FLAG
    Channel Handling, 59
API429_CHN_CFG_RM_LOCAL_FLAG
    Channel Handling, 59
API429_CHN_CFG_RX_LABEL_FLAG
    Channel Handling, 59
API429_CHN_CFG_RX_POLLUTION_FLAG
    Channel Handling, 59
API429_CHN_CFG_SDI_ENABLED_FLAG
    Channel Handling, 60
API429_CHN_CFG_TX_DYNTAG_FRAMING_F↩
        LAG
    Channel Handling, 60
API429_CHN_CFG_TX_FIFO_FLAG

Channel Handling, 60
API429_CHN_CFG_TX_FRAMING_FLAG
Channel Handling, 60
API429_CHN_CFG_TX_POLLUTION_FLAG
Channel Handling, 61
API429_CHN_CFG_TX_RATE_CONTROLLED_↩ FLAG
Channel Handling, 61
API429_CHN_CFG_UNCONFIGURED
Channel Handling, 61
API429_LS_GRADE_DEFAULT
Board Related Functions, 18
API429_MAX_BOARDNAME_LEN
Board Related Functions, 19
API429_MAX_CHANNELS
Board Related Functions, 19
API429_MAX_OS_INFO_LEN
Board Related Functions, 19, 20
API429_MAX_URL_LEN
Board Related Functions, 20
API429_RM_CONTINUOUS_CAPTURE
Data Monitoring, 86
API429_RM_MON_DEFAULT_SIZE
Data Monitoring, 87
active
api429_channel_info, 162
addsub_val
api429_rcv_pb_cmd, 176
Ai429CheckModule
VxWorks Related Functions, 152
AiVme429MapModule
VxWorks Related Functions, 153
AiVme429UnmapModule
VxWorks Related Functions, 153
and_mask
api429_rcv_pb_cmd, 176
api429_board_id
Board Related Functions, 23, 24
api429_board_info, 157
capability_flags, 157
name, 157
num_channels, 158
num_trigger_in, 158
num_trigger_out, 158
serial, 158
server_url, 158
api429_brw, 160
e_type, 161
api429_channel_coupling
Board Related Functions, 26, 27
api429_channel_info, 162
active, 162
capability_flags, 162
config_flags, 162
speed, 163
api429_discr_info, 164
canIn, 164
canOut, 164
channels, 164
api429_discrete_pps_config, 166
ul_DiscreteChannelId, 166
ul_EnaDis, 166
api429_event_type
Channel Handling, 63
api429_intr_loglist_entry, 167
ul_Lla, 167
ul_Llb, 167
ul_Lld, 168
x_Llc, 168
api429_irig_source
Board Related Functions, 27, 28
api429_loglist_entry, 170
api429_low_speed_grade
Board Related Functions, 28, 29
api429_lxfer_dyntag, 171
ul_LowerLimit, 171
ul_Mask, 171
ul_Mode, 171
ul_StartValue, 172
ul_StepSize, 172
ul_UpperLimit, 172
api429_memory_object
Raw Board Memory Access, 45
api429_mframe_in, 173
pul_Xfers, 173
ul_FrmId, 173
ul_XferCnt, 173
api429_mframe_out, 175
ul_MFrameAddr, 175
api429_pnp_event
Library Administration Functions, 74
api429_pxi_mode
Board Related Functions, 29, 30
api429_pxi_trigger_line
Board Related Functions, 30, 31
api429_rcv_pb_cmd, 176
addsub_val, 176
and_mask, 176

asc, 177
duration, 177
or_mask, 177
padding1, 177
pb_id, 178
start_delay, 178
xor_mask, 178
api429_rcv_stack_entry, 179
brw, 179
ldata, 179
tm_tag, 179
api429_rcv_stack_entry_ex, 181
brw, 181
day_of_year, 181
ldata, 181
tm_tag, 182
api429_remote_server, 183
host_name, 183
os_info, 183
protocol_major, 183
protocol_minor, 184
server_version, 184
api429_replay_status, 185
uc_Padding1, 185
uc_Status, 185
ul_EntryCnt, 185
ul_RpiCnt, 186
ul_Size, 186
ul_StartAddr, 186
uw_Padding2, 186
api429_rm_activity_trigger_def, 188
start_mask, 188
start_pat, 188
stop_mask, 188
stop_pat, 189
api429_rm_function_block, 190
ext_trg, 190
fb_id, 190
fbi, 191
fe, 191
ir, 191
llc, 191
lli, 192
llim, 192
mask, 192
pre_cnt, 192
pre_rel, 192
trg_line, 193
trg_reset, 193
trg_set, 193
ulc, 193
uli, 194
ulim, 194
api429_rm_interrupt_mode
Data Monitoring, 89, 90
api429_rm_label_config, 195
enable, 195
label_id, 195
sdi, 195
api429_rm_mode
Data Monitoring, 91
api429_rm_setup, 197
interrupt_mode, 197
mode, 197, 198
size_in_entries, 198
tat_count, 198
api429_rm_trigger_mode
Data Monitoring, 92
api429_rm_trigger_setup, 199
generate_output_strobe, 199
input_trigger_line, 199
mode, 199
output_trigger_line, 200
api429_rx_activity, 201
ChannelActivity, 201
api429_rx_buf_ctl, 202
ci, 202
inr, 202
ixw, 202
api429_rx_buf_entry, 204
lab_data, 204
api429_rx_frame_response_setup, 205
compare_mask, 205
compare_value, 205
tx_acyc_frame_id, 205
tx_channel, 206
api429_rx_label_setup, 207
bufSize, 207
con, 207
irCon, 207
irIndex, 208
label, 208
sdi, 208
api429_speed
Channel Handling, 63
api429_speed_modifier
Board Related Functions, 32
api429_time, 209

day, 209
hour, 209
millisecond, 209
minute, 210
second, 210
api429_tm_tag, 211
api429_trg_ctl_cmd, 212
low_func, 212
low_limit, 212
mask, 212
padding1, 213
up_func, 213
up_limit, 213
api429_tx_fifo_entry, 214
ulControl, 214
ulData, 214
api429_tx_fifo_setup, 215
ulDefaultGapSize, 215
ulFifoControl, 215
ulFifoSize, 215
api429_tx_fifo_status, 217
ulEntriesFree, 217
ulEntriesToSend, 217
api429_tx_mode
Data Transmitting, 120
api429_version, 218
ul_BuildNr, 218
ul_MajorSpecialVer, 218
ul_MajorVer, 218
ul_MinorSpecialVer, 219
ul_MinorVer, 219
api429_version_info, 220
ul_BoardSerialNr, 220
x_AslLcaVer, 220
x_DllVer, 221
x_FirmwareBiu1Ver, 221
x_FirmwareBiu2Ver, 221
x_IoLcaBiu1Ver, 221
x_IoLcaBiu2Ver, 222
x_MonitorVer, 222
x_PciLcaVer, 222
x_SysDrvVer, 222
x_TargetSWVer, 222
x_TcpVer, 223
x_VmeGeneric, 223
api429_xfer, 224
buf_size, 224
err_type, 224
ir_index, 224
xfer_gap, 225
xfer_id, 225
xfer_ir, 225
xfer_type, 226
api429_xfer_data, 227
ul_XferData, 227
ul_XferTTHigh, 227
ul_XferTTLow, 227
api429_xfer_data_read_input, 229
ul_BufSize, 229
ul_BufStart, 229
ul_Mode, 229
ul_XferId, 230
api429_xfer_error
Data Transmitting, 121
api429_xfer_info, 231
ul_XferCnt, 231
ul_XferData, 231
ul_XferIx, 231
ul_XferTTHigh, 232
ul_XferTTLow, 232
api429_xfer_out, 233
ul_FreeMem, 233
ul_Status, 233
ul_XferDescAddr, 233
api429_xfer_type
Data Transmitting, 121
Api429BoardChannelCouplingInfo
Board Related Functions, 33
Api429BoardChannelCouplingIsSupported
Board Related Functions, 34
Api429BoardChannelCouplingSet
Board Related Functions, 34
Api429BoardClose
Board Related Functions, 35
Api429BoardInfoGet
Board Related Functions, 35
Api429BoardLsGradeSet
Board Related Functions, 36
Api429BoardMemBlockRead
Raw Board Memory Access, 46
Api429BoardMemBlockWrite
Raw Board Memory Access, 47
Api429BoardMemLocationGet
Raw Board Memory Access, 48
Api429BoardMemRead
Raw Board Memory Access, 48
Api429BoardMemTimerAddrGet
Raw Board Memory Access, 49

Api429BoardMemWrite
Raw Board Memory Access, 49
Api429BoardNameGet
Board Related Functions, 36
Api429BoardOpen
Board Related Functions, 37
Api429BoardPXIGeographicalAddressGet
Board Related Functions, 37
Api429BoardPXITriggerConfigure
Board Related Functions, 38
Api429BoardReset
Board Related Functions, 38
Api429BoardSelftest
Board Related Functions, 39
Api429BoardServerInfoGet
Board Related Functions, 40
Api429BoardSpeedModifierSet
Board Related Functions, 40
Api429BoardTimeGet
Board Related Functions, 41
Api429BoardTimeSet
Board Related Functions, 41
Api429BoardTimeSourceGet
Board Related Functions, 42
Api429BoardTimeSourceSet
Board Related Functions, 43
Api429ChannelCallbackRegister
Channel Handling, 64
Api429ChannelCallbackUnregister
Channel Handling, 64
Api429ChannelClear
Channel Handling, 65
Api429ChannelHalt
Channel Handling, 65
Api429ChannelInfoGet
Channel Handling, 66
Api429ChannelSpeedSet
Channel Handling, 66
Api429ChannelStart
Channel Handling, 67
Api429DiscretesConfigGet
Discrete Handling, 69
Api429DiscretesConfigSet
Discrete Handling, 69
Api429DiscretesInfoGet
Discrete Handling, 70
Api429DiscretesPPSSet
Discrete Handling, 70
Api429DiscretesRead
Discrete Handling, 71
Api429DiscretesWrite
Discrete Handling, 71
Api429LibDebugLevelSet
Library Administration Functions, 74
Api429LibErrorDescGet
Library Administration Functions, 75
Api429LibExit
Library Administration Functions, 75
Api429LibInit
Library Administration Functions, 76
Api429LibPnpCallbackSet
Library Administration Functions, 76
Api429LibServerConnect
Library Administration Functions, 77
Api429LibServerDisconnect
Library Administration Functions, 77
Api429ReadBSPVersionEx
Version Handling, 149
Api429ReplayDataWrite
Data Replay, 80
Api429ReplayInit
Data Replay, 80
Api429ReplayStatusGet
Data Replay, 81
Api429RmCreate
Data Monitoring, 93
Api429RmDataRead
Data Monitoring, 93
Api429RmDataReadWithDayOfYear
Data Monitoring, 94
Api429RmFuncBlockConfigure
Data Monitoring, 95
Api429RmInfoGet
Data Monitoring, 96
Api429RmLabelConfigure
Data Monitoring, 96
Api429RmMultiLabelConfigure
Data Monitoring, 97
Api429RmResume
Data Monitoring, 97
Api429RmStackPointersGet
Data Monitoring, 98
Api429RmStartTriggerSet
Data Monitoring, 99
Api429RmStatusGet
Data Monitoring, 99
Api429RmStopTriggerSet
Data Monitoring, 100

Api429RmSuspend
Data Monitoring, 101
Api429RmTriggerConfigSet
Data Monitoring, 101
Api429RmTriggerPatternSet
Data Monitoring, 102
Api429RxActivityGet
Data Receiver, 105
Api429RxDataLoopAssign
Data Receiver, 106
Api429RxFrameResponseAssign
Data Receiver, 107
Api429RxFrameResponseRelease
Data Receiver, 107
Api429RxInit
Data Receiver, 108
Api429RxLabelBufferOffsetGet
Data Receiver, 109
Api429RxLabelBufferRead
Data Receiver, 109
Api429RxLabelBufferWrite
Data Receiver, 110
Api429RxLabelConfigure
Data Receiver, 111
Api429RxLabelStatusGet
Data Receiver, 111
Api429RxMultiLabelConfigure
Data Receiver, 112
Api429RxPollutionConfigure
Data Receiver, 113
Api429RxStatusGet
Data Receiver, 113
Api429TxAcycFrameCreate
Data Transmitting, 122
Api429TxAcycFrameDelete
Data Transmitting, 122
Api429TxAcycFrameSend
Data Transmitting, 123
Api429TxAmplitudeSet
Data Transmitting, 123
Api429TxFifoDataWordCreate
FIFO Based Transmitting, 140
Api429TxFifoDataWordsWrite
FIFO Based Transmitting, 141
Api429TxFifoDelayCreate
FIFO Based Transmitting, 142
Api429TxFifoInterruptCreate
FIFO Based Transmitting, 142
Api429TxFifoReset
FIFO Based Transmitting, 143
Api429TxFifoSetup
FIFO Based Transmitting, 144
Api429TxFifoSetupGet
FIFO Based Transmitting, 144
Api429TxFifoStatusGet
FIFO Based Transmitting, 145
Api429TxFifoTriggerPulseCreate
FIFO Based Transmitting, 145
Api429TxFifoTriggerWaitCreate
FIFO Based Transmitting, 146
Api429TxFifoWrite
FIFO Based Transmitting, 146
Api429TxFrameTimeSet
Data Transmitting, 124
Api429TxInit
Data Transmitting, 125
Api429TxMajorFrameCreate
Data Transmitting, 125
Api429TxMajorFrameDelete
Data Transmitting, 126
Api429TxMinorFrameCreate
Data Transmitting, 127
Api429TxMinorFrameDelete
Data Transmitting, 127
Api429TxPrepareFraming
Data Transmitting, 128
Api429TxRepetitionCountSet
Data Transmitting, 128
Api429TxStartOnTTag
Data Transmitting, 130
Api429TxStartOnTrigger
Data Transmitting, 129
Api429TxStatusGet
Data Transmitting, 130
Api429TxXferBufferOffsetGet
Data Transmitting, 131
Api429TxXferBufferRead
Data Transmitting, 132
Api429TxXferBufferWrite
Data Transmitting, 132
Api429TxXferCreate
Data Transmitting, 133
Api429TxXferDelete
Data Transmitting, 134
Api429TxXferDyntagAssign
Data Transmitting, 134
Api429TxXferRateAdd
Data Transmitting, 135

Api429TxXferRateRemove
Data Transmitting, 136
Api429TxXferRateShow
Data Transmitting, 136
Api429TxXferSkip
Data Transmitting, 137
Api429TxXferStatusGet
Data Transmitting, 137
Api429VersionGet
Version Handling, 149
Api429VersionGetAll
Version Handling, 150
asc
api429_rcv_pb_cmd, 177

Board Related Functions, 8
API429_BOARD_CAP_PXI_TRG_FLAG, 15
API429_BOARD_HAS_PXI_TRIGGER, 16
API429_BOARD_TX_INHIBIT_FLAG, 16
API429_BOARD_TX_INHIBIT_IS_ACTIVAT↩ ED, 17
API429_BOARD_TYPE_EMBEDDED_FLAG, 17
API429_BOARD_TYPE_IS_EMBEDDED, 18
API429_LS_GRADE_DEFAULT, 18
API429_MAX_BOARDNAME_LEN, 19
API429_MAX_CHANNELS, 19
API429_MAX_OS_INFO_LEN, 19, 20
API429_MAX_URL_LEN, 20
api429_board_id, 23, 24
api429_channel_coupling, 26, 27
api429_irig_source, 27, 28
api429_low_speed_grade, 28, 29
api429_pxi_mode, 29, 30
api429_pxi_trigger_line, 30, 31
api429_speed_modifier, 32
Api429BoardChannelCouplingInfo, 33
Api429BoardChannelCouplingIsSupported, 34
Api429BoardChannelCouplingSet, 34
Api429BoardClose, 35
Api429BoardInfoGet, 35
Api429BoardLsGradeSet, 36
Api429BoardNameGet, 36
Api429BoardOpen, 37
Api429BoardPXIGeographicalAddressGet, 37
Api429BoardPXITriggerConfigure, 38
Api429BoardReset, 38
Api429BoardSelftest, 39
Api429BoardServerInfoGet, 40
Api429BoardSpeedModifierSet, 40
Api429BoardTimeGet, 41
Api429BoardTimeSet, 41
Api429BoardTimeSourceGet, 42
Api429BoardTimeSourceSet, 43
MAX_API429_BIU_CHN, 21
MAX_API429_BIU, 20, 21
TY_API429_BOARD_INFO, 21
TY_API429_TIME, 21
TY_E_API429_CHANNEL_COUPLING, 22
TY_E_API429_IRIG_SOURCE, 22
TY_E_API429_LOW_SPEED_GRADE, 22
TY_E_API429_SPEED_MODIFIER, 22
brw
api429_rcv_stack_entry, 179
api429_rcv_stack_entry_ex, 181
buf_size
api429_xfer, 224
bufSize
api429_rx_label_setup, 207

CPStr, 238
canIn
api429_discr_info, 164
canOut
api429_discr_info, 164
capability_flags
api429_board_info, 157
api429_channel_info, 162
cdate
FInfoRec, 240
FInfoRec64, 244
Channel Handling, 51
API429_CHANNEL_CALLBACK, 61
API429_CHANNEL_CAN_RECEIVE, 53
API429_CHANNEL_CAN_TRANSMIT, 54
API429_CHANNEL_HAS_VARIABLE_AMP↩ LITUDE, 54
API429_CHANNEL_IN_GLOBAL_MONITOR↩ ING_MODE, 54
API429_CHANNEL_IN_LOCAL_MONITORI↩ NG_MODE, 54
API429_CHANNEL_IN_MONITORING_MO↩ DE, 54
API429_CHANNEL_IN_PHYS_REPLAY_MO↩ DE, 55
API429_CHANNEL_IN_REPLAY_MODE, 55
API429_CHANNEL_IN_RX_LP_MODE, 55
API429_CHANNEL_IN_RX_MODE, 55

API429_CHANNEL_IN_RX_NORMAL_MODE, 56
API429_CHANNEL_IN_RX_TX_MIXING_MODE, 56
API429_CHANNEL_IN_TX_DYNTAG_MODE, 56
API429_CHANNEL_IN_TX_FIFO_MODE, 56
API429_CHANNEL_IN_TX_FRAMING_MODE, 57
API429_CHANNEL_IN_TX_LP_MODE, 57
API429_CHANNEL_IN_TX_MODE, 57
API429_CHANNEL_IN_TX_RATE_CONTROLLED_MODE, 57
API429_CHN_CAP_RX_FLAG, 58
API429_CHN_CAP_TX_FLAG, 58
API429_CHN_CAP_VAR_AMP_FLAG, 58
API429_CHN_CFG_PARITY_ENABLED_FLAG, 58
API429_CHN_CFG_PHYS_REPLAY_FLAG, 59
API429_CHN_CFG_RM_GLOBAL_FLAG, 59
API429_CHN_CFG_RM_LOCAL_FLAG, 59
API429_CHN_CFG_RX_LABEL_FLAG, 59
API429_CHN_CFG_RX_POLLUTION_FLAG, 59
API429_CHN_CFG_SDI_ENABLED_FLAG, 60
API429_CHN_CFG_TX_DYNTAG_FRAMING_FLAG, 60
API429_CHN_CFG_TX_FIFO_FLAG, 60
API429_CHN_CFG_TX_FRAMING_FLAG, 60
API429_CHN_CFG_TX_POLLUTION_FLAG, 61
API429_CHN_CFG_TX_RATE_CONTROLLED_FLAG, 61
API429_CHN_CFG_UNCONFIGURED, 61
api429_event_type, 63
api429_speed, 63
Api429ChannelCallbackRegister, 64
Api429ChannelCallbackUnregister, 64
Api429ChannelClear, 65
Api429ChannelHalt, 65
Api429ChannelInfoGet, 66
Api429ChannelSpeedSet, 66
Api429ChannelStart, 67
TY_API429_CHANNEL_INFO, 62
TY_API429_INTR_LOGLIST_ENTRY, 62
TY_API429_LOGLIST_TYPE_ENTRY, 62
TY_E_API429_SPEED, 62
ChannelActivity
api429_rx_activity, 201
channels
api429_discr_info, 164
ci
api429_rx_buf_ctl, 202
cmplx128, 235
cmplx64, 236
cmplxExt, 237
compare_mask
api429_rx_frame_response_setup, 205
compare_value
api429_rx_frame_response_setup, 205
con
api429_rx_label_setup, 207
config_flags
api429_channel_info, 162
creator
FInfoRec, 240
FInfoRec64, 244

Data Monitoring, 83
API429_RM_CONTINUOUS_CAPTURE, 86
API429_RM_MON_DEFAULT_SIZE, 87
api429_rm_interrupt_mode, 89, 90
api429_rm_mode, 91
api429_rm_trigger_mode, 92
Api429RmCreate, 93
Api429RmDataRead, 93
Api429RmDataReadWithDayOfYear, 94
Api429RmFuncBlockConfigure, 95
Api429RmInfoGet, 96
Api429RmLabelConfigure, 96
Api429RmMultiLabelConfigure, 97
Api429RmResume, 97
Api429RmStackPointersGet, 98
Api429RmStartTriggerSet, 99
Api429RmStatusGet, 99
Api429RmStopTriggerSet, 100
Api429RmSuspend, 101
Api429RmTriggerConfigSet, 101
Api429RmTriggerPatternSet, 102
TY_API429_RCV_STACK_ENTRY_EX, 87
TY_API429_RCV_STACK_ENTRY, 87
TY_API429_RM_ACTIVITY_TRIGGER_DEF, 87
TY_API429_RM_FUNCTION_BLOCK, 88
TY_API429_RM_LABEL_CONFIG, 88
TY_API429_RM_SETUP, 88
TY_API429_RM_TRIGGER_SETUP, 88
TY_API429_TRG_CTL_CMD, 89
TY_E_API429_RM_INTERRUPT_MODE, 89
TY_E_API429_RM_MODE, 89
TY_E_API429_RM_TRIGGER_MODE, 89

Data Receiver, 103
Api429RxActivityGet, 105
Api429RxDataLoopAssign, 106
Api429RxFrameResponseAssign, 107
Api429RxFrameResponseRelease, 107
Api429RxInit, 108
Api429RxLabelBufferOffsetGet, 109
Api429RxLabelBufferRead, 109
Api429RxLabelBufferWrite, 110
Api429RxLabelConfigure, 111
Api429RxLabelStatusGet, 111
Api429RxMultiLabelConfigure, 112
Api429RxPollutionConfigure, 113
Api429RxStatusGet, 113
TY_API429_RCV_PB_CMD, 104
TY_API429_RX_ACTIVITY, 104
TY_API429_RX_BUF_CTL, 105
TY_API429_RX_BUF_ENTRY, 105
TY_API429_RX_FRAME_RESPONSE_SET↩ UP, 105
TY_API429_RX_LABEL_SETUP, 105
Data Replay, 79
Api429ReplayDataWrite, 80
Api429ReplayInit, 80
Api429ReplayStatusGet, 81
TY_API429_REPLAY_STATUS, 79
Data Transmitting, 115
api429_tx_mode, 120
api429_xfer_error, 121
api429_xfer_type, 121
Api429TxAcycFrameCreate, 122
Api429TxAcycFrameDelete, 122
Api429TxAcycFrameSend, 123
Api429TxAmplitudeSet, 123
Api429TxFrameTimeSet, 124
Api429TxInit, 125
Api429TxMajorFrameCreate, 125
Api429TxMajorFrameDelete, 126
Api429TxMinorFrameCreate, 127
Api429TxMinorFrameDelete, 127
Api429TxPrepareFraming, 128
Api429TxRepetitionCountSet, 128
Api429TxStartOnTTag, 130
Api429TxStartOnTrigger, 129
Api429TxStatusGet, 130
Api429TxXferBufferOffsetGet, 131
Api429TxXferBufferRead, 132
Api429TxXferBufferWrite, 132
Api429TxXferCreate, 133
Api429TxXferDelete, 134
Api429TxXferDyntagAssign, 134
Api429TxXferRateAdd, 135
Api429TxXferRateRemove, 136
Api429TxXferRateShow, 136
Api429TxXferSkip, 137
Api429TxXferStatusGet, 137
TY_API429_LXFER_DYNTAG, 118
TY_API429_MFRAME_IN, 118
TY_API429_MFRAME_OUT, 118
TY_API429_XFER_DATA_READ_INPUT, 119
TY_API429_XFER_DATA, 118
TY_API429_XFER_INFO, 119
TY_API429_XFER_OUT, 119
TY_API429_XFER, 118
TY_E_API429_TX_MODE, 119
TY_E_API429_XFER_ERROR, 120
TY_E_API429_XFER_TYPE, 120
DateRec, 239
day
api429_time, 209
day_of_year
api429_rcv_stack_entry_ex, 181
Discrete Handling, 68
Api429DiscretesConfigGet, 69
Api429DiscretesConfigSet, 69
Api429DiscretesInfoGet, 70
Api429DiscretesPPSSet, 70
Api429DiscretesRead, 71
Api429DiscretesWrite, 71
TY_API429_DISCR_INFO, 68
TY_API429_DISCRETE_PPS_CONFIG, 69
duration
api429_rcv_pb_cmd, 177

e_type
api429_brw, 161
enable
api429_rm_label_config, 195
err_type
api429_xfer, 224
ext_trg
api429_rm_function_block, 190

FIFO Based Transmitting, 139
Api429TxFifoDataWordCreate, 140
Api429TxFifoDataWordsWrite, 141
Api429TxFifoDelayCreate, 142
Api429TxFifoInterruptCreate, 142

Api429TxFifoReset, 143
Api429TxFifoSetup, 144
Api429TxFifoSetupGet, 144
Api429TxFifoStatusGet, 145
Api429TxFifoTriggerPulseCreate, 145
Api429TxFifoTriggerWaitCreate, 146
Api429TxFifoWrite, 146
TY_API429_TX_FIFO_ENTRY, 140
TY_API429_TX_FIFO_SETUP, 140
TY_API429_TX_FIFO_STATUS, 140
FInfoRec, 240
cdate, 240
creator, 240
folder, 241
group, 241
h, 241
isInvisible, 241
location, 242
mdate, 242
owner, 242
permissions, 242
rfSize, 242
size, 243
type, 243
v, 243
FInfoRec64, 244
cdate, 244
creator, 244
folder, 245
group, 245
h, 245
isInvisible, 245
location, 246
mdate, 246
owner, 246
permissions, 246
rfSize, 246
size, 247
type, 247
v, 247
fb_id
api429_rm_function_block, 190
fbi
api429_rm_function_block, 191
fe
api429_rm_function_block, 191
flags
_FMListDetails, 156
folder
FInfoRec, 241
FInfoRec64, 245
free
VInfoRec, 250

generate_output_strobe
api429_rm_trigger_setup, 199
group
FInfoRec, 241
FInfoRec64, 245

h
FInfoRec, 241
FInfoRec64, 245
host_name
api429_remote_server, 183
hour
api429_time, 209

input_trigger_line
api429_rm_trigger_setup, 199
inr
api429_rx_buf_ctl, 202
interrupt_mode
api429_rm_setup, 197
ir
api429_rm_function_block, 191
ir_index
api429_xfer, 224
irCon
api429_rx_label_setup, 207
irIndex
api429_rx_label_setup, 208
isInvisible
FInfoRec, 241
FInfoRec64, 245
ixw
api429_rx_buf_ctl, 202

LStr, 248
lab_data
api429_rx_buf_entry, 204
label
api429_rx_label_setup, 208
label_id
api429_rm_label_config, 195
ldata
api429_rcv_stack_entry, 179
api429_rcv_stack_entry_ex, 181
Library Administration Functions, 73

api429_pnp_event, 74
Api429LibDebugLevelSet, 74
Api429LibErrorDescGet, 75
Api429LibExit, 75
Api429LibInit, 76
Api429LibPnpCallbackSet, 76
Api429LibServerConnect, 77
Api429LibServerDisconnect, 77
TY_429PNP_CALLBACK_FUNC_PTR, 73
TY_E_API429_PNP_EVENT, 74
llc
api429_rm_function_block, 191
lli
api429_rm_function_block, 192
llim
api429_rm_function_block, 192
location
FInfoRec, 242
FInfoRec64, 246
low_func
api429_trg_ctl_cmd, 212
low_limit
api429_trg_ctl_cmd, 212

MAX_API429_BIU_CHN
Board Related Functions, 21
MAX_API429_BIU
Board Related Functions, 20, 21
mask
api429_rm_function_block, 192
api429_trg_ctl_cmd, 212
mdate
FInfoRec, 242
FInfoRec64, 246
MemStatRec, 249
millisecond
api429_time, 209
minute
api429_time, 210
mode
api429_rm_setup, 197, 198
api429_rm_trigger_setup, 199

name
api429_board_info, 157
num_channels
api429_board_info, 158
num_trigger_in
api429_board_info, 158
num_trigger_out
api429_board_info, 158

or_mask
api429_rcv_pb_cmd, 177
os_info
api429_remote_server, 183
output_trigger_line
api429_rm_trigger_setup, 200
owner
FInfoRec, 242
FInfoRec64, 246

padding1
api429_rcv_pb_cmd, 177
api429_trg_ctl_cmd, 213
pb_id
api429_rcv_pb_cmd, 178
permissions
FInfoRec, 242
FInfoRec64, 246
pre_cnt
api429_rm_function_block, 192
pre_rel
api429_rm_function_block, 192
protocol_major
api429_remote_server, 183
protocol_minor
api429_remote_server, 184
pul_Xfers
api429_mframe_in, 173

Raw Board Memory Access, 44
api429_memory_object, 45
Api429BoardMemBlockRead, 46
Api429BoardMemBlockWrite, 47
Api429BoardMemLocationGet, 48
Api429BoardMemRead, 48
Api429BoardMemTimerAddrGet, 49
Api429BoardMemWrite, 49
TY_E_API429_MEMORY_OBJECT, 45
rfSize
FInfoRec, 242
FInfoRec64, 246

sdi
api429_rm_label_config, 195
api429_rx_label_setup, 208
second
api429_time, 210

serial
api429_board_info, 158
server_url
api429_board_info, 158
server_version
api429_remote_server, 184
size
FInfoRec, 243
FInfoRec64, 247
VInfoRec, 250
size_in_entries
api429_rm_setup, 198
speed
api429_channel_info, 163
start_delay
api429_rcv_pb_cmd, 178
start_mask
api429_rm_activity_trigger_def, 188
start_pat
api429_rm_activity_trigger_def, 188
stop_mask
api429_rm_activity_trigger_def, 188
stop_pat
api429_rm_activity_trigger_def, 189

TY_429PNP_CALLBACK_FUNC_PTR
Library Administration Functions, 73
TY_API429_BOARD_INFO
Board Related Functions, 21
TY_API429_CHANNEL_INFO
Channel Handling, 62
TY_API429_DISCR_INFO
Discrete Handling, 68
TY_API429_DISCRETE_PPS_CONFIG
Discrete Handling, 69
TY_API429_INTR_LOGLIST_ENTRY
Channel Handling, 62
TY_API429_LOGLIST_TYPE_ENTRY
Channel Handling, 62
TY_API429_LXFER_DYNTAG
Data Transmitting, 118
TY_API429_MFRAME_IN
Data Transmitting, 118
TY_API429_MFRAME_OUT
Data Transmitting, 118
TY_API429_RCV_PB_CMD
Data Receiver, 104
TY_API429_RCV_STACK_ENTRY_EX
Data Monitoring, 87
TY_API429_RCV_STACK_ENTRY
Data Monitoring, 87
TY_API429_REPLAY_STATUS
Data Replay, 79
TY_API429_RM_ACTIVITY_TRIGGER_DEF
Data Monitoring, 87
TY_API429_RM_FUNCTION_BLOCK
Data Monitoring, 88
TY_API429_RM_LABEL_CONFIG
Data Monitoring, 88
TY_API429_RM_SETUP
Data Monitoring, 88
TY_API429_RM_TRIGGER_SETUP
Data Monitoring, 88
TY_API429_RX_ACTIVITY
Data Receiver, 104
TY_API429_RX_BUF_CTL
Data Receiver, 105
TY_API429_RX_BUF_ENTRY
Data Receiver, 105
TY_API429_RX_FRAME_RESPONSE_SETUP
Data Receiver, 105
TY_API429_RX_LABEL_SETUP
Data Receiver, 105
TY_API429_TIME
Board Related Functions, 21
TY_API429_TRG_CTL_CMD
Data Monitoring, 89
TY_API429_TX_FIFO_ENTRY
FIFO Based Transmitting, 140
TY_API429_TX_FIFO_SETUP
FIFO Based Transmitting, 140
TY_API429_TX_FIFO_STATUS
FIFO Based Transmitting, 140
TY_API429_VERSION_INFO
Version Handling, 149
TY_API429_VERSION
Version Handling, 148
TY_API429_XFER_DATA_READ_INPUT
Data Transmitting, 119
TY_API429_XFER_DATA
Data Transmitting, 118
TY_API429_XFER_INFO
Data Transmitting, 119
TY_API429_XFER_OUT
Data Transmitting, 119
TY_API429_XFER
Data Transmitting, 118
TY_E_API429_CHANNEL_COUPLING

Board Related Functions, 22
TY_E_API429_IRIG_SOURCE
Board Related Functions, 22
TY_E_API429_LOW_SPEED_GRADE
Board Related Functions, 22
TY_E_API429_MEMORY_OBJECT
Raw Board Memory Access, 45
TY_E_API429_PNP_EVENT
Library Administration Functions, 74
TY_E_API429_RM_INTERRUPT_MODE
Data Monitoring, 89
TY_E_API429_RM_MODE
Data Monitoring, 89
TY_E_API429_RM_TRIGGER_MODE
Data Monitoring, 89
TY_E_API429_SPEED_MODIFIER
Board Related Functions, 22
TY_E_API429_SPEED
Channel Handling, 62
TY_E_API429_TX_MODE
Data Transmitting, 119
TY_E_API429_XFER_ERROR
Data Transmitting, 120
TY_E_API429_XFER_TYPE
Data Transmitting, 120
tat_count
api429_rm_setup, 198
tm_tag
api429_rcv_stack_entry, 179
api429_rcv_stack_entry_ex, 182
trg_line
api429_rm_function_block, 193
trg_reset
api429_rm_function_block, 193
trg_set
api429_rm_function_block, 193
tx_acyc_frame_id
api429_rx_frame_response_setup, 205
tx_channel
api429_rx_frame_response_setup, 206
type
_FMListDetails, 156
FInfoRec, 243
FInfoRec64, 247

uc_Padding1
api429_replay_status, 185
uc_Status
api429_replay_status, 185
ul_BoardSerialNr
api429_version_info, 220
ul_BufSize
api429_xfer_data_read_input, 229
ul_BufStart
api429_xfer_data_read_input, 229
ul_BuildNr
api429_version, 218
ul_DiscreteChannelId
api429_discrete_pps_config, 166
ul_EnaDis
api429_discrete_pps_config, 166
ul_EntryCnt
api429_replay_status, 185
ul_FreeMem
api429_xfer_out, 233
ul_FrmId
api429_mframe_in, 173
ul_Lla
api429_intr_loglist_entry, 167
ul_Llb
api429_intr_loglist_entry, 167
ul_Lld
api429_intr_loglist_entry, 168
ul_LowerLimit
api429_lxfer_dyntag, 171
ul_MFrameAddr
api429_mframe_out, 175
ul_MajorSpecialVer
api429_version, 218
ul_MajorVer
api429_version, 218
ul_Mask
api429_lxfer_dyntag, 171
ul_MinorSpecialVer
api429_version, 219
ul_MinorVer
api429_version, 219
ul_Mode
api429_lxfer_dyntag, 171
api429_xfer_data_read_input, 229
ul_RpiCnt
api429_replay_status, 186
ul_Size
api429_replay_status, 186
ul_StartAddr
api429_replay_status, 186
ul_StartValue
api429_lxfer_dyntag, 172

ul_Status
api429_xfer_out, 233
ul_StepSize
api429_lxfer_dyntag, 172
ul_UpperLimit
api429_lxfer_dyntag, 172
ul_XferCnt
api429_mframe_in, 173
api429_xfer_info, 231
ul_XferData
api429_xfer_data, 227
api429_xfer_info, 231
ul_XferDescAddr
api429_xfer_out, 233
ul_XferId
api429_xfer_data_read_input, 230
ul_XferIx
api429_xfer_info, 231
ul_XferTTHigh
api429_xfer_data, 227
api429_xfer_info, 232
ul_XferTTLow
api429_xfer_data, 227
api429_xfer_info, 232
ulControl
api429_tx_fifo_entry, 214
ulData
api429_tx_fifo_entry, 214
ulDefaultGapSize
api429_tx_fifo_setup, 215
ulEntriesFree
api429_tx_fifo_status, 217
ulEntriesToSend
api429_tx_fifo_status, 217
ulFifoControl
api429_tx_fifo_setup, 215
ulFifoSize
api429_tx_fifo_setup, 215
ulc
api429_rm_function_block, 193
uli
api429_rm_function_block, 194
ulim
api429_rm_function_block, 194
up_func
api429_trg_ctl_cmd, 213
up_limit
api429_trg_ctl_cmd, 213
used
VInfoRec, 250
uw_Padding2
api429_replay_status, 186

v
FInfoRec, 243
FInfoRec64, 247
VInfoRec, 250
free, 250
size, 250
used, 250
Version Handling, 148
Api429ReadBSPVersionEx, 149
Api429VersionGet, 149
Api429VersionGetAll, 150
TY_API429_VERSION_INFO, 149
TY_API429_VERSION, 148
VxWorks Related Functions, 152
Ai429CheckModule, 152
AiVme429MapModule, 153
AiVme429UnmapModule, 153

x_AslLcaVer
api429_version_info, 220
x_DllVer
api429_version_info, 221
x_FirmwareBiu1Ver
api429_version_info, 221
x_FirmwareBiu2Ver
api429_version_info, 221
x_IoLcaBiu1Ver
api429_version_info, 221
x_IoLcaBiu2Ver
api429_version_info, 222
x_Llc
api429_intr_loglist_entry, 168
x_MonitorVer
api429_version_info, 222
x_PciLcaVer
api429_version_info, 222
x_SysDrvVer
api429_version_info, 222
x_TargetSWVer
api429_version_info, 222
x_TcpVer
api429_version_info, 223
x_VmeGeneric
api429_version_info, 223
xfer_gap

api429_xfer, 225
xfer_id
api429_xfer, 225
xfer_ir
api429_xfer, 225
xfer_type
api429_xfer, 226
xor_mask
api429_rcv_pb_cmd, 178