Linux Software (DM35424)

Generated by Doxygen 1.8.5

Tue Jun 29 2021 14:56:05

Contents

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

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

Chapter 4

Module Documentation

4.1 DM35424 ADC Library Constants

Macros

- #define DM35424_ADC_MODE_RESET 0x00

  Register value for ADC Mode Reset.
- #define DM35424_ADC_MODE_PAUSE 0x01

  Register value for ADC Mode Pause.
- #define DM35424_ADC_MODE_GO_SINGLE_SHOT 0x02

  Register value for ADC Mode Go (Single Shot)
- #define DM35424_ADC_MODE_GO_REARM 0x03

  Register value for ADC Mode Go (Rearm after Stop)
- #define DM35424_ADC_MODE_UNINITIALIZED 0x04

  Register value for ADC Mode Uninitialized.
- #define DM35424_ADC_STAT_STOPPED 0x00

  Register value for ADC Status - Stopped.
- #define DM35424_ADC_STAT_FILLING_PRE_TRIG_BUFF 0x01

  Register value for ADC Status - Filling Pre-Start Buffer.
- #define DM35424_ADC_STAT_WAITING_START_TRIG 0x02

  Register value for ADC Status - Waiting for Start Trigger.
- #define DM35424_ADC_STAT_SAMPLING 0x03

  Register value for ADC Status - Sampling Data.
- #define DM35424_ADC_STAT_FILLING_POST_TRIG_BUFF 0x04

  Register value for ADC Status - Filling Post-Stop Buffer.
- #define DM35424_ADC_STAT_WAIT_REARM 0x05

  Register value for ADC Status - Wait for Rearm.
- #define DM35424_ADC_STAT_DONE 0x07

  Register value for ADC Status - Done.
- #define DM35424_ADC_STAT_UNINITIALIZED 0x08

  Register value for ADC Status - Uninitialized.
- #define DM35424_ADC_STAT_INITIALIZING 0x09

  Register value for ADC Status - Initializing.
- #define DM35424_ADC_INT_SAMPLE_TAKEN_MASK 0x01

  Register value for Interrupt Mask - Sample Taken.
- #define DM35424_ADC_INT_CHAN_THRESHOLD_MASK 0x02

  Register value for Interrupt Mask - Channel Threshold Exceeded.

- #define DM35424_ADC_INT_PRE_BUFF_FULL_MASK 0x04

  Register value for Interrupt Mask - Pre-Start Buffer Filled.
- #define DM35424_ADC_INT_START_TRIG_MASK 0x08

  Register value for Interrupt Mask - Start Trigger Occurred.
- #define DM35424_ADC_INT_STOP_TRIG_MASK 0x10

  Register value for Interrupt Mask - Stop Trigger Occurred.
- #define DM35424_ADC_INT_POST_BUFF_FULL_MASK 0x20

  Register value for Interrupt Mask - Post-Stop Buffer Filled.
- #define DM35424_ADC_INT_SAMP_COMPL_MASK 0x40

  Register value for Interrupt Mask - Sampling Complete.
- #define DM35424_ADC_INT_PACER_TICK_MASK 0x80

  Register value for Interrupt Mask - Pacer Clock Tick Occurred.
- #define DM35424_ADC_INT_ALL_MASK 0xFF

  Register value for Interrupt Mask - All Bits.
- #define DM35424_ADC_CHAN_INTR_LOW_THRESHOLD_MASK 0x01

  Register value for Channel Low Threshold Interrupt.
- #define DM35424_ADC_CHAN_INTR_HIGH_THRESHOLD_MASK 0x02

  Register value for Channel High Threshold Interrupt.
- #define DM35424_ADC_CHAN_FILTER_ORDER0 0x0

  Register value for Channel Filter Order 0.
- #define DM35424_ADC_CHAN_FILTER_ORDER1 0x1

  Register value for Channel Filter Order 1.
- #define DM35424_ADC_CHAN_FILTER_ORDER2 0x2

  Register value for Channel Filter Order 2.
- #define DM35424_ADC_CHAN_FILTER_ORDER3 0x3

  Register value for Channel Filter Order 3.
- #define DM35424_ADC_CHAN_FILTER_ORDER4 0x4

  Register value for Channel Filter Order 4.
- #define DM35424_ADC_CHAN_FILTER_ORDER5 0x5

  Register value for Channel Filter Order 5.
- #define DM35424_ADC_CHAN_FILTER_ORDER6 0x6

  Register value for Channel Filter Order 6.
- #define DM35424_ADC_CHAN_FILTER_ORDER7 0x7

  Register value for Channel Filter Order 7.
- #define DM35424_ADC_FE_CONFIG_POWER_ACTIVE 0x80

  Register value for setting channel power to active.
- #define DM35424_ADC_FE_CONFIG_PGA_ACTIVE 0x40

  Register value for setting channel PGA to active.
- #define DM35424_ADC_FE_CONFIG_IN_SWITCH_ENABLED 0x20

  Register value for setting channel input switch to enabled.
- #define DM35424_ADC_FE_CONFIG_DAC_LOOPBACK 0x00

  Register value for measuring between DACx Chy and VREF (2.5V)
- #define DM35424_ADC_FE_CONFIG_SNGL_END_POS 0x08

  Register value for measuring InP - VREF (singled ended)
- #define DM35424_ADC_FE_CONFIG_SNGL_END_NEG 0x10

  Register value for measuring VREF - InN (singled ended)
- #define DM35424_ADC_FE_CONFIG_DIFFERENTIAL 0x18

  Register value for setting channel to In(Positive) - In(Negative) connection. This is the most common.
- #define DM35424_ADC_FE_CONFIG_GAIN_1 0x00

  Register value for setting a Gain of 1.
- #define DM35424_ADC_FE_CONFIG_GAIN_2 0x04

Register value for setting a Gain of 2.

- #define DM35424_ADC_FE_CONFIG_GAIN_4 0x02

  Register value for setting a Gain of 4.
- #define DM35424_ADC_FE_CONFIG_GAIN_8 0x06

  Register value for setting a Gain of 8.
- #define DM35424_ADC_FE_CONFIG_GAIN_16 0x01

  Register value for setting a Gain of 16.
- #define DM35424_ADC_FE_CONFIG_GAIN_32 0x05

  Register value for setting a Gain of 32.
- #define DM35424_ADC_FE_CONFIG_GAIN_64 0x03

  Register value for setting a Gain of 64.
- #define DM35424_ADC_FE_CONFIG_GAIN_128 0x07

  Register value for setting a Gain of 128.
- #define DM35424_ADC_FE_CONFIG_POWER_MASK 0x80

  Bit mask for the channel power bit of the FE Config.
- #define DM35424_ADC_FE_CONFIG_PGA_MASK 0x40

  Bit mask for the channel PGA bit of the FE Config.
- #define DM35424_ADC_FE_CONFIG_INPUT_SW_ENABLE_MASK 0x20

  Bit mask for the channel input switch bit of the FE Config.
- #define DM35424_ADC_FE_CONFIG_INPUT_LINE_MASK 0x18

  Bit mask for the channel input line bits of the FE Config.
- #define DM35424_ADC_FE_CONFIG_GAIN_MASK 0x07

  Bit mask for the channel gain bits of the FE Config.
- #define DM35424_ADC_THRESHOLD_MAX 8388607L

  Maximum allowable value to write to the threshold register.
- #define DM35424_ADC_THRESHOLD_MIN -8388608L

  Minimum allowable value to write to the threshold register.
- #define DM35424_ADC_HIGH_SPD_MIN_DIV 2

  Minimum divider allowed for HIGH SPEED mode.
- #define DM35424_ADC_HIGH_RES_MIN_DIV 2

  Minimum divider allowed for HIGH RES mode.
- #define DM35424_ADC_LOW_POW_MIN_DIV 4

  Minimum divider allowed for LOW POWER mode.
- #define DM35424_ADC_LOW_SPD_MIN_DIV 10

  Minimum divider allowed for LOW SPEED mode.
- #define DM35424_ADC_MAX_RATE 106000

  Max rate of the ADC (Hz)
- #define DM35424_ADC_MAX_VALUE 8388607

  Max possible value for ADC.
- #define DM35424_ADC_MIN_VALUE -8388608

  Min possible value for ADC.

Enumerations

- enum DM35424_Adc_Clock_Events {
  DM35424_ADC_CLK_BUS_SRC_DISABLE = 0x00, DM35424_ADC_CLK_BUS_SRC_SAMPLE_TAKEN = 0x80, DM35424_ADC_CLK_BUS_SRC_CHAN_THRESH = 0x81, DM35424_ADC_CLK_BUS_SRC_PRE_-START_BUFF_FULL = 0x82,
  DM35424_ADC_CLK_BUS_SRC_START_TRIG = 0x83, DM35424_ADC_CLK_BUS_SRC_STOP_TRIG = 0x84, DM35424_ADC_CLK_BUS_SRC_POST_STOP_BUFF_FULL = 0x85, DM35424_ADC_CLK_BUS_S-RC_SAMPLING_COMPLETE = 0x86,
  DM35424_ADC_CLK_BUS_SRC_PACER_TICK = 0x87 }

Clock events for the global source clocks.

- enum DM35424_Input_Ranges { DM35424_ADC_RNG_BIPOLAR_2_5V, DM35424_ADC_RNG_BIPOLAR_1_25V, DM35424_ADC_RNG_BIPOLAR_625mV, DM35424_ADC_RNG_BIPOLAR_312mV, DM35424_ADC_RNG_BIPOLAR_156mV, DM35424_ADC_RNG_BIPOLAR_78mV, DM35424_ADC_RNG_BIPOLAR_39mV, DM35424_ADC_RNG_BIPOLAR_19mV, DM35424_ADC_RNG_UNIPOLAR_5V }

  Input range of the ADC input pin. This combines polarity and gain into a single enumeration, and is the preferred way of setting polarity and gain.

- enum DM35424_Input_Mode { DM35424_ADC_INPUT_DIFFERENTIAL, DM35424_ADC_INPUT_SINGLE_ENDED_POS, DM35424_ADC_INPUT_SINGLE_ENDED_NEG, DM35424_ADC_INPUT_DAC_LOOPBACK }

  Input mode of the ADC pin.

- enum DM35424_Gains { DM35424_ADC_GAIN_05, DM35424_ADC_GAIN_1, DM35424_ADC_GAIN_2, DM35424_ADC_GAIN_4, DM35424_ADC_GAIN_8, DM35424_ADC_GAIN_16, DM35424_ADC_GAIN_32, DM35424_ADC_GAIN_64, DM35424_ADC_GAIN_128 }

  Input gain to apply to the incoming signal. Note that the preferred method of setting the gain is through the input range enumeration.

- enum DM35424_Sampling_Mode { DM35424_ADC_MODE_CONFIG_HIGH_SPEED =0x01, DM35424_ADC_MODE_CONFIG_HIGH_RES =0x03, DM35424_ADC_MODE_CONFIG_LOW_POWER =0x04, DM35424_ADC_MODE_CONFIG_LOW_SPEED =0x06 }

  Sampling mode for the AD Config Register.

4.1.1 Detailed Description

4.1.2 Enumeration Type Documentation

4.1.2.1 enum DM35424_Adc_Clock_Events

Clock events for the global source clocks.

Enumerator

DM35424_ADC_CLK_BUS_SRC_DISABLE Register value for Clock Event - Disabled.

DM35424_ADC_CLK_BUS_SRC_SAMPLE_TAKEN Register value for Clock Event - Sample Taken.

DM35424_ADC_CLK_BUS_SRC_CHAN_THRESH Register value for Clock Event - Channel Threshold Exceeded.

DM35424_ADC_CLK_BUS_SRC_PRE_START_BUFF_FULL Register value for Clock Event - Pre-Start Buffer Full.

DM35424_ADC_CLK_BUS_SRC_START_TRIG Register value for Clock Event - Start Trigger Occurred.

DM35424_ADC_CLK_BUS_SRC_STOP_TRIG Register value for Clock Event - Stop Trigger Occurred.

DM35424_ADC_CLK_BUS_SRC_POST_STOP_BUFF_FULL Register value for Clock Event - Post-Stop Buffer Full.

DM35424_ADC_CLK_BUS_SRC_SAMPLING_COMPLETE Register value for Clock Event - Sampling Complete.

DM35424_ADC_CLK_BUS_SRC_PACER_TICK Register value for Clock Event - Pacer Tick Occurred.

Definition at line 425 of file dm35424_adc_library.h.

4.1.2.2 enum DM35424_Gains

Input gain to apply to the incoming signal. Note that the preferred method of setting the gain is through the input range enumeration.

Note

Not all values in this enumeration may apply to your board,as this is a shared library. Please consult the board manual for legal values.

Enumerator

DM35424_ADC_GAIN_05 Input Half-Gain

DM35424_ADC_GAIN_1 Input Gain of 1

DM35424_ADC_GAIN_2 Input Gain of 2

DM35424_ADC_GAIN_4 Input Gain of 4

DM35424_ADC_GAIN_8 Input Gain of 8

DM35424_ADC_GAIN_16 Input Gain of 16

DM35424_ADC_GAIN_32 Input Gain of 32

DM35424_ADC_GAIN_64 Input Gain of 64

DM35424_ADC_GAIN_128 Input Gain of 128

Definition at line 591 of file dm35424_adc_library.h.

4.1.2.3 enum DM35424_Input_Mode

Input mode of the ADC pin.

Note

Not all values in this enumeration may apply to your board,as this is a shared library. Please consult the board manual for valid values.

Enumerator

DM35424_ADC_INPUT_DIFFERENTIAL Differential Operation

DM35424_ADC_INPUT_SINGLE_ENDED_POS Single-Ended operation, input connected to positive ADC input.

DM35424_ADC_INPUT_SINGLE_ENDED_NEG Single-Ended operation, input connected to negative ADC input.

DM35424_ADC_INPUT_DAC_LOOPBACK DAC Output internally looped-back to ADC input. See hardware manual for more info.

Definition at line 553 of file dm35424_adc_library.h.

4.1.2.4 enum DM35424_Input_Ranges

Input range of the ADC input pin. This combines polarity and gain into a single enumeration, and is the preferred way of setting polarity and gain.

Note

Not all values in this enumeration may apply to your board,as this is a shared library. Please consult the board manual for legal values.

Enumerator

DM35424_ADC_RNG_BIPOLAR_2_5V Bipolar Mode, -2.5 to 2.5 V

DM35424_ADC_RNG_BIPOLAR_1_25V Bipolar Mode, -1.25 to 1.25 V

DM35424_ADC_RNG_BIPOLAR_625mV Bipolar Mode, -625 mV to 625 mV

DM35424_ADC_RNG_BIPOLAR_312mV Bipolar Mode, -312.5 mV to 312.5 mV

DM35424_ADC_RNG_BIPOLAR_156mV Bipolar Mode, -156.25 mV to 156.25 mV

DM35424_ADC_RNG_BIPOLAR_78mV Bipolar Mode, -78.125 mV to 78.125 mV

DM35424_ADC_RNG_BIPOLAR_39mV Bipolar Mode, -39.0626 mV to 39.0626 mV

DM35424_ADC_RNG_BIPOLAR_19mV Bipolar Mode, -19.53125 mV to 19.53125 mV

DM35424_ADC_RNG_UNIPOLAR_5V Unipolar Mode, 0 to 5 V

Definition at line 494 of file dm35424_adc_library.h.

4.1.2.5 enum DM35424_Sampling_Mode

Sampling mode for the AD Config Register.

Enumerator

DM35424_ADC_MODE_CONFIG_HIGH_SPEED Register value for ADC AD Config - High Speed.

DM35424_ADC_MODE_CONFIG_HIGH_RES Register value for ADC AD Config - High Resolution.

DM35424_ADC_MODE_CONFIG_LOW_POWER Register value for ADC AD Config - Low Power.

DM35424_ADC_MODE_CONFIG_LOW_SPEED Register value for ADC AD Config - Low Speed.

Definition at line 645 of file dm35424_adc_library.h.

4.2 DM35424 ADC Public Library Functions

Functions

- DM35424LIB_API int DM35424_Adc_Open (struct DM35424_Board_Descriptor *handle, unsigned int number_of_type, struct DM35424_Function_Block *func_block)

  Open the ADC indicated, and determine register locations of control blocks needed to control it.
- DM35424LIB_API int DM35424_Adc_Get_Start_Trigger (struct DM35424_Board_Descriptor *handle, struct DM35424_Function_Block *func_block, uint8_t *start_trigger)

  Get the start trigger for data collection.
- DM35424LIB_API int DM35424_Adc_Set_Start_Trigger (struct DM35424_Board_Descriptor *handle, struct DM35424_Function_Block *func_block, uint8_t start_trigger)

  Set the start trigger for data collection.
- DM35424LIB_API int DM35424_Adc_Get_Stop_Trigger (struct DM35424_Board_Descriptor *handle, struct DM35424_Function_Block *func_block, uint8_t *stop_trigger)

  Get the stop trigger for data collection.
- DM35424LIB_API int DM35424_Adc_Set_Stop_Trigger (struct DM35424_Board_Descriptor *handle, struct DM35424_Function_Block *func_block, uint8_t stop_trigger)

  Set the stop trigger for data collection.
- DM35424LIB_API int DM35424_Adc_Get_Pre_Trigger_Samples (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint32_t *count)

  Get the amount of data to capture prior to start trigger.
- DM35424LIB_API int DM35424_Adc_Set_Pre_Trigger_Samples (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint32_t count)

  Set the amount of data to capture prior to start trigger.
- DM35424LIB_API int DM35424_Adc_Get_Post_Stop_Samples (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint32_t *count)

  Get the amount of data to capture after stop trigger.
- DM35424LIB_API int DM35424_Adc_Set_Post_Stop_Samples (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint32_t count)

  Set the amount of data to capture after stop trigger.
- DM35424LIB_API int DM35424_Adc_Get_Clock_Src (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, enum DM35424_Clock_Sources *source)

  Get the clock source for the ADC.
- DM35424LIB_API int DM35424_Adc_Set_Clock_Src (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, enum DM35424_Clock_Sources source)

  Set the clock source for the ADC.
- DM35424LIB_API int DM35424_Adc_Initialize (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block)

  Prepare the ADC for actual data collection. Moves the ADC from uninitialized to stopped.
- DM35424LIB_API int DM35424_Adc_Set_Clk_Divider (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint32_t divider)

  Set the Clock Divider for the ADC function block.
- DM35424LIB_API int DM35424_Adc_Set_Sample_Rate (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint32_t rate, uint32_t *actual_rate)

  Set the sampling rate for the ADC.
- DM35424LIB_API int DM35424_Adc_Channel_Get_Front_End_Config (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, uint16_t *fe_config)

  Get the front-end config register contents.
- DM35424LIB_API int DM35424_Adc_Ad_Config_Set_Mode (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, enum DM35424_Sampling_Mode mode)

  Set the Configuration of the AD Mode.

- DM35424LIB_API int DM35424_Adc_Ad_Config_Get_Mode (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint16_t *mode)

  Get AD Config register.
- DM35424LIB_API int DM35424_Adc_Interrupt_Set_Config (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint16_t int_source, int enable)

  Configure the interrupts for the ADC.
- DM35424LIB_API int DM35424_Adc_Interrupt_Get_Config (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint16_t *interrupt_ena)

  Get the interrupt configuration for the ADC.
- DM35424LIB_API int DM35424_Adc_Start (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block)

  Set the ADC mode to Start.
- DM35424LIB_API int DM35424_Adc_Start_Rearm (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block)

  Set the ADC mode to Start-Rearm.
- DM35424LIB_API int DM35424_Adc_Reset (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block)

  Set the ADC mode to Reset.
- DM35424LIB_API int DM35424_Adc_Pause (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block)

  Set the ADC mode to Pause.
- DM35424LIB_API int DM35424_Adc_Uninitialize (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block)

  Set the ADC mode to Uninitialized.
- DM35424LIB_API int DM35424_Adc_Get_Mode_Status (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint8_t *mode_status)

  Get the ADC mode-status value.
- DM35424LIB_API int DM35424_Adc_Channel_Get_Last_Sample (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, int32_t *value)

  Get the last sample taken from the ADC.
- DM35424LIB_API int DM35424_Adc_Get_Sample_Count (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint32_t *value)

  Get the count of number of samples taken.
- DM35424LIB_API int DM35424_Adc_Interrupt_Get_Status (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint16_t *value)

  Get the interrupt status register.
- DM35424LIB_API int DM35424_Adc_Interrupt_Clear_Status (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint16_t value)

  Clear the interrupt status register.
- DM35424LIB_API int DM35424_Adc_Channel_Setup (struct DM35424_Board_Descriptor *handle, struct DM35424_Function_Block *func_block, unsigned int channel, enum DM35424_Input_Ranges input_range, enum DM35424_Input_Mode input_mode)

  Setup the channel input for the ADC.
- DM35424LIB_API int DM35424_Adc_Channel_Reset (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel)

  Reset the channel front-end config.
- DM35424LIB_API int DM35424_Adc_Channel_Interrupt_Set_Config (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, uint8_t interrupts_to_set, int enable)

  Setup the channel interrupts.
- DM35424LIB_API int DM35424_Adc_Channel_Interrupt_Get_Config (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, uint8_t *chan_intr_enable)

Get the channel interrupt configuration.

- DM35424LIB_API int DM35424_Adc_Channel_Interrupt_Get_Status (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, uint8_t *chan_intr_status)

  Get the channel interrupt status.

- DM35424LIB_API int DM35424_Adc_Channel_Interrupt_Clear_Status (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, uint8_t chan_intr_status)

  Clear the interrupt status for this channel.

- DM35424LIB_API int DM35424_Adc_Channel_Find_Interrupt (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int *channel_with_interrupt, int *channel_has_interrupt, uint8_t *channel_intr_status, uint8_t *channel_intr_enable)

  Find the first channel with an interrupt. Note that this is only useful when looking for a threshold interrupt.

- DM35424LIB_API int DM35424_Adc_Channel_Set_Filter (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, uint8_t chan_filter)

  Set the filter value for the channel.

- DM35424LIB_API int DM35424_Adc_Channel_Get_Filter (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, uint8_t *chan_filter)

  Get the filter value for the channel.

- DM35424LIB_API int DM35424_Adc_Channel_Set_Low_Threshold (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, int32_t threshold)

  Set the lower threshold for this channel.

- DM35424LIB_API int DM35424_Adc_Channel_Set_High_Threshold (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, int32_t threshold)

  Set the high threshold for this channel.

- DM35424LIB_API int DM35424_Adc_Channel_Get_Thresholds (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, int32_t *low_threshold, int32_t *high_threshold)

  Get both thresholds for this channel.

- DM35424LIB_API int DM35424_Adc_Fifo_Channel_Read (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, int32_t *value)

  Read an ADC sample stored in the onboard FIFO.

- DM35424LIB_API int DM35424_Adc_Set_Clock_Source_Global (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, enum DM35424_Clock_Sources clock_select, enum DM35424_Adc_Clock_Events clock_driver)

  Set the global clock source for the ADC.

- DM35424LIB_API int DM35424_Adc_Get_Clock_Source_Global (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, int clock_select, int *clock_source)

  Get the global clock source for the selected clock.

- DM35424LIB_API int DM35424_Adc_Sample_To_Volts (enum DM35424_Input_Ranges input_range, int32_t adc_sample, float *volts)

  Convert an ADC sample to a volts value.

- DM35424LIB_API int DM35424_Adc_Volts_To_Sample (enum DM35424_Input_Ranges input_range, float volts, int32_t *adc_sample)

  Convert volts to an ADC value.

4.2.1 Detailed Description

DM35424_Adc_Library_Constants

4.2.2 Function Documentation

4.2.2.1 DM35424LIB_API int DM35424_Adc_Ad_Config_Get_Mode (struct DM35424_Board_Descriptor * *handle*, const struct DM35424_Function_Block * *func_block*, uint16_t * *mode*)

Get AD Config register.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor, which contains the offsets to command sections of the board.
mode	Returned AD_Config register value

Note

This library function will only apply to some ADC subtypes, and may not be applicable to the DM35424.

Return values

0	Success.
Non-zero	Failure. errno may be set as follows: • `EINVAL` Invalid mode requested, or does not apply to this ADC. ENODEV Attempted to set for an invalid ADC

4.2.2.2 DM35424LIB_API int DM35424_Adc_Ad_Config_Set_Mode (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* enum DM35424_Sampling_Mode *mode*)

Set the Configuration of the AD Mode.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor, which contains the offsets to command sections of the board.
mode	Mode to set for the AD Config

Note

This library function will only apply to some ADC subtypes, and may not be applicable to the DM35424.

Return values

0	Success.
Non-zero	Failure. errno may be set as follows: • `EINVAL` Invalid mode requested, or does not apply to this ADC. ENODEV Attempted to set for an invalid ADC

Referenced by main(), and setup_adc().

4.2.2.3 DM35424LIB_API int DM35424_Adc_Channel_Find_Interrupt (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* unsigned int * *channel_with_interrupt,* int * *channel_has_interrupt,* uint8_t * *channel_intr_status,* uint8_t * *channel_intr_enable*)

Find the first channel with an interrupt. Note that this is only useful when looking for a threshold interrupt.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor, which contains the offsets to command sections of the board.
channel_with_interrupt	Pointer to the returned channel that has an interrupt (if any)
channel_has_interrupt	Pointer to boolean indicating whether returned channel has interrupt or not.
channel_intr_status	Pointer to the channel's interrupt status.
channel_intr_enable	Pointer to the channel's interrupt enable register.

Return values

0	Success.
Non-zero	Failure.

4.2.2.4 DM35424LIB_API int DM35424_Adc_Channel_Get_Filter (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* unsigned int *channel,* uint8_t * *chan_filter*)

Get the filter value for the channel.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor, which contains the offsets to command sections of the board.
channel	Channel to get filter of
chan_filter	Pointer to returned channel filter value. Reference the user's manual for valid filter values.

Return values

0	Success.
Non-zero	Failure. errno may be set as follows: • `EINVAL` Invalid channel requested.

4.2.2.5 DM35424LIB_API int DM35424_Adc_Channel_Get_Front_End_Config (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* unsigned int *channel,* uint16_t * *fe_config*)

Get the front-end config register contents.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor, which contains the offsets to command sections of the board.
channel	Channel the FE Config is requested for.
fe_config	Pointer to the returned FE Config register value

Return values

0	Success.
Non-zero	Failure. errno may be set as follows: • EINVAL Invalid channel requested.

4.2.2.6 DM35424LIB_API int DM35424_Adc_Channel_Get_Last_Sample (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* unsigned int *channel,* int32_t * *value*)

Get the last sample taken from the ADC.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor, which contains the offsets to command sections of the board.
channel	Channel to get sample from.
value	Pointer to returned sample value.

Return values

0	Success.
Non-zero	Failure.

Referenced by main(), and run_test_9().

4.2.2.7 DM35424LIB_API int DM35424_Adc_Channel_Get_Thresholds (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* unsigned int *channel,* int32_t * *low_threshold,* int32_t * *high_threshold*)

Get both thresholds for this channel.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor, which contains the offsets to command sections of the board.
channel	Channel to set the high threshold of.
low_threshold	Pointer to signed integer value of threshold.
high_threshold	Pointer to signed integer value of threshold.

Note

The threshold register is only 16-bits. Thus, the threshold value really only represents the top 16-bits of the 24-bit ADC value. For convenience, the threshold parameters are returned as 32-bit integers. After getting the value from the register, it will be left-shifted 16-bits.

Return values

0	Success.
Non-zero	Failure. errno may be set as follows: • EINVAL Invalid channel requested.

4.2.2.8 DM35424LIB_API int DM35424_Adc_Channel_Interrupt_Clear_Status (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* unsigned int *channel,* uint8_t *chan_intr_status*)

Clear the interrupt status for this channel.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor, which contains the offsets to command sections of the board.
channel	Channel to clear.
chan_intr_status	Bit mask indicating which interrupts to clear.

Return values

0	Success.
Non-zero	Failure. errno may be set as follows: • EINVAL Invalid channel requested.

Referenced by run_test_9().

4.2.2.9 DM35424LIB_API int DM35424_Adc_Channel_Interrupt_Get_Config (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* unsigned int *channel,* uint8_t * *chan_intr_enable*)

Get the channel interrupt configuration.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor, which contains the offsets to command sections of the board.
channel	Channel to get configururation.
chan_intr_enable	Pointer to interrupt configuration being returned.

Return values

0	Success.
Non-zero	Failure. errno may be set as follows: • EINVAL Invalid channel requested.

4.2.2.10 DM35424LIB_API int DM35424_Adc_Channel_Interrupt_Get_Status (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* unsigned int *channel,* uint8_t * *chan_intr_status*)

Get the channel interrupt status.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor, which contains the offsets to command sections of the board.

channel	Channel to get configururation.
chan_intr_status	Pointer to interrupt status being returned.

Return values

0	Success.
Non-zero	Failure. errno may be set as follows: • EINVAL Invalid channel requested.

4.2.2.11 DM35424LIB_API int DM35424_Adc_Channel_Interrupt_Set_Config (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* unsigned int *channel,* uint8_t *interrupts_to_set,* int *enable*)

Setup the channel interrupts.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor, which contains the offsets to command sections of the board.
channel	Channel to configure.
interrupts_to_set	A bit mask indicating which interrupts to set
enable	A boolean value indicating if selected interrupts should be enabled or disabled.

Return values

0	Success.
Non-zero	Failure. errno may be set as follows: • EINVAL Invalid channel requested, or requested input mode is not possible on this ADC subtype.

Referenced by run_test_9().

4.2.2.12 DM35424LIB_API int DM35424_Adc_Channel_Reset (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* unsigned int *channel*)

Reset the channel front-end config.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor, which contains the offsets to command sections of the board.
channel	Channel to reset.

Return values

0	Success.
Non-zero	Failure. errno may be set as follows: • EINVAL Invalid channel requested.

4.2.2.13 DM35424LIB_API int DM35424_Adc_Channel_Set_Filter (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* unsigned int *channel,* uint8_t *chan_filter*)

Set the filter value for the channel.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor, which contains the offsets to command sections of the board.
channel	Channel to set filter
chan_filter	Channel filter value. Reference the user's manual for valid filter values.

Return values

0	Success.
Non-zero	Failure. errno may be set as follows: • EINVAL Invalid channel requested.

Referenced by main().

4.2.2.14 DM35424LIB_API int DM35424_Adc_Channel_Set_High_Threshold (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* unsigned int *channel,* int32_t *threshold*)

Set the high threshold for this channel.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor, which contains the offsets to command sections of the board.
channel	Channel to set the high threshold of.
threshold	Signed high threshold value for this channel.

Note

The threshold register is only 16-bits. Thus, the threshold value really only represents the top 16-bits of the 24-bit ADC value. For convenience, the threshold parameter is accepted as a 32-bit integer. Before writing the value, it will be right-shifted 16 bits.

Return values

0	Success.
Non-zero	Failure. errno may be set as follows: • EINVAL Invalid channel requested.

Referenced by run_test_9().

4.2.2.15 DM35424LIB_API int DM35424_Adc_Channel_Set_Low_Threshold (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* unsigned int *channel,* int32_t *threshold*)

Set the lower threshold for this channel.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor, which contains the offsets to command sections of the board.
channel	Channel to set the lower threshold of.
threshold	Signed lower threshold value for this channel.

Note

The threshold register is only 16-bits. Thus, the threshold value really only represents the top 16-bits of the 24-bit ADC value. For convenience, the threshold parameter is accepted as a 32-bit integer. Before writing the value, it will be right-shifted 16 bits.

Return values

0	Success.
Non-zero	Failure. errno may be set as follows: • `EINVAL` Invalid channel requested.

Referenced by run_test_9().

4.2.2.16 DM35424LIB_API int DM35424_Adc_Channel_Setup (struct DM35424_Board_Descriptor * *handle,* struct DM35424_Function_Block * *func_block,* unsigned int *channel,* enum DM35424_Input_Ranges *input_range,* enum DM35424_Input_Mode *input_mode*)

Setup the channel input for the ADC.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor, which contains the offsets to command sections of the board.
channel	Channel to configure.
input_range	An enumerated value representing the input voltage range of the input.
input_mode	An enumerated value representing the mode to set the input line to.

Note

The input line mode and input voltage ranges available for the board is dependent on the ADC subtype on the board. Review the user's guide to see what values the ADC can be set to.

Return values

0	Success.
Non-zero	Failure. errno may be set as follows: • `EINVAL` Invalid channel requested, or requested mode/range is not possible on this ADC subtype.

Referenced by main(), and setup_adc().

4.2.2.17 DM35424LIB_API int DM35424_Adc_Fifo_Channel_Read (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* unsigned int *channel,* int32_t * *value*)

Read an ADC sample stored in the onboard FIFO.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor, which contains the offsets to command sections of the board.
channel	Channel to get sample from.
value	Pointer to returned sample value.

Return values

0	Success.
Non-zero	Failure.

4.2.2.18 DM35424LIB_API int DM35424_Adc_Get_Clock_Source_Global (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* int *clock_select,* int * *clock_source*)

Get the global clock source for the selected clock.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor, which contains the offsets to command sections of the board.
clock_select	Which global clock source to get
clock_source	Pointer to the returned clock source for the selected global clock

Return values

0	Success.
Non-zero	Failure. errno may be set as follows: • `EINVAL` Invalid clock select or source..

4.2.2.19 DM35424LIB_API int DM35424_Adc_Get_Clock_Src (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* enum DM35424_Clock_Sources * *source*)

Get the clock source for the ADC.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor, which contains the offsets to command sections of the board.
source	Pointer to returned clock source.

Return values

0	Success.
Non-zero	Failure.

4.2.2.20 DM35424LIB_API int DM35424_Adc_Get_Mode_Status (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* uint8_t * *mode_status*)

Get the ADC mode-status value.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor, which contains the offsets to command sections of the board.
mode_status	Pointer to the mode_status value to return.

Return values

0	Success.
Non-zero	Failure.

Referenced by run_test_9().

4.2.2.21 DM35424LIB_API int DM35424_Adc_Get_Post_Stop_Samples (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* uint32_t * *count*)

Get the amount of data to capture after stop trigger.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor, which contains the offsets to command sections of the board.
count	Pointer to the returned count.

Return values

0	Success.
Non-zero	Failure.

4.2.2.22 DM35424LIB_API int DM35424_Adc_Get_Pre_Trigger_Samples (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* uint32_t * *count*)

Get the amount of data to capture prior to start trigger.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor, which contains the offsets to command sections of the board.
count	Pointer to the returned capture count

Return values

0	Success.
Non-zero	Failure.

4.2.2.23 DM35424LIB_API int DM35424_Adc_Get_Sample_Count (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* uint32_t * *value*)

Get the count of number of samples taken.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor, which contains the offsets to command sections of the board.
value	Pointer to returned sample count.

Return values

0	Success.
Non-zero	Failure.

Referenced by main(), and run_test_9().

4.2.2.24 DM35424LIB_API int DM35424_Adc_Get_Start_Trigger (struct DM35424_Board_Descriptor * *handle,* struct DM35424_Function_Block * *func_block,* uint8_t * *start_trigger*)

Get the start trigger for data collection.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor, which contains the offsets to command sections of the board.
start_trigger	Pointer to the returned trigger value.

Return values

0	Success.
Non-zero	Failure.

4.2.2.25 DM35424LIB_API int DM35424_Adc_Get_Stop_Trigger (struct DM35424_Board_Descriptor * *handle,* struct DM35424_Function_Block * *func_block,* uint8_t * *stop_trigger*)

Get the stop trigger for data collection.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor, which contains the offsets to command sections of the board.
stop_trigger	Pointer to the returned trigger value

Return values

0	Success.
Non-zero	Failure.

4.2.2.26 DM35424LIB_API int DM35424_Adc_Initialize (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block*)

Prepare the ADC for actual data collection. Moves the ADC from uninitialized to stopped.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor, which contains the offsets to command sections of the board.

Note

In many cases, several other steps have to occur before initialization is attempted, or the device will not initialize correctly or at all. Please review the user's manual for the correct steps to take.

Return values

0	Success.
Non-zero	Failure. errno may be set as follows: • `EPERM` Attempted to initialize an ADC with no active channels. EBUSY Device did not complete initialization in the time expected (timeout).

Referenced by main(), and setup_adc().

4.2.2.27 DM35424LIB_API int DM35424_Adc_Interrupt_Clear_Status (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* uint16_t *value*)

Clear the interrupt status register.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor, which contains the offsets to command sections of the board.
value	Bit mask of which interrupts to clear.

Return values

0	Success.
Non-zero	Failure.

Referenced by main().

4.2.2.28 DM35424LIB_API int DM35424_Adc_Interrupt_Get_Config (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* uint16_t * *interrupt_ena*)

Get the interrupt configuration for the ADC.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor, which contains the offsets to command sections of the board.
interrupt_ena	Pointer to the interrupt configuration register.

Return values

0	Success.
Non-zero	Failure.

Referenced by main().

4.2.2.29 DM35424LIB_API int DM35424_Adc_Interrupt_Get_Status (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* uint16_t * *value*)

Get the interrupt status register.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor, which contains the offsets to command sections of the board.
value	Pointer to returned interrupt status.

Return values

0	Success.
Non-zero	Failure.

Referenced by main().

4.2.2.30 DM35424LIB_API int DM35424_Adc_Interrupt_Set_Config (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* uint16_t *int_source,* int *enable*)

Configure the interrupts for the ADC.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor, which contains the offsets to command sections of the board.
int_source	The interrupts to configure. The bits indicate specific interrupts. Consult the user's manual for a description.
enable	Boolean indicating to enable or disable the selected interrupts.

Return values

0	Success.
Non-zero	Failure.

Referenced by main().

4.2.2.31 DM35424LIB_API int DM35424_Adc_Open (struct DM35424_Board_Descriptor * *handle,* unsigned int *number_of_type,* struct DM35424_Function_Block * *func_block*)

Open the ADC indicated, and determine register locations of control blocks needed to control it.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
number_of_type	Which ADC to open. The first ADC on the board will be 0.
func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.

Return values

0	Success.
-1	Failure.

Referenced by main(), and setup_adc().

4.2.2.32 DM35424LIB_API int DM35424_Adc_Pause (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block*)

Set the ADC mode to Pause.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor, which contains the offsets to command sections of the board.

Return values

0	Success.
Non-zero	Failure.

4.2.2.33 DM35424LIB_API int DM35424_Adc_Reset (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block*)

Set the ADC mode to Reset.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor, which contains the offsets to command sections of the board.

Return values

0	Success.
Non-zero	Failure.

Referenced by main(), and run_test_9().

4.2.2.34 DM35424LIB_API int DM35424_Adc_Sample_To_Volts (enum DM35424_Input_Ranges *input_range,* int32_t *adc_sample,* float * *volts*)

Convert an ADC sample to a volts value.

Parameters

input_range	Enumerated value indicating what range the ADC channel has been set to, or NULL if the ADC does not have selectable input ranges.
adc_sample	Signed value from the ADC that we want to convert to volts.
volts	Pointer to the returned float value in volts.

Return values

0	Success.
Non-zero	Failure. errno may be set as follows: • ENODEV The function block passed in is the wrong subtype

Referenced by main().

4.2.2.35 DM35424LIB_API int DM35424_Adc_Set_Clk_Divider (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* uint32_t *divider*)

Set the Clock Divider for the ADC function block.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor, which contains the offsets to command sections of the board.
divider	The requested clock divider.

Return values

0	Success.
-1	Failure.

4.2.2.36 DM35424LIB_API int DM35424_Adc_Set_Clock_Source_Global (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* enum DM35424_Clock_Sources *clock_select,* enum DM35424_Adc_Clock_Events *clock_driver*)

Set the global clock source for the ADC.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor, which contains the offsets to command sections of the board.
clock_select	Which global clock source to set
clock_driver	Source to set global clock to (what is driving it?)

Return values

0	Success.
Non-zero	Failure. errno may be set as follows: • `EINVAL` Invalid clock select or source..

Referenced by run_test_9().

4.2.2.37 DM35424LIB_API int DM35424_Adc_Set_Clock_Src (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* enum DM35424_Clock_Sources *source*)

Set the clock source for the ADC.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor, which contains the offsets to command sections of the board.
source	Clock source to use for the ADC. Consult the user's manual for the list of available sources.

Return values

0	Success.
Non-zero	Failure. errno may be set as follows: • `EINVAL` The clock source selected is not valid.

Referenced by main(), and setup_adc().

4.2.2.38 DM35424LIB_API int DM35424_Adc_Set_Post_Stop_Samples (struct DM35424_Board_Descriptor * *handle*, const struct DM35424_Function_Block * *func_block*, uint32_t *count*)

Set the amount of data to capture after stop trigger.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor, which contains the offsets to command sections of the board.
count	Number of samples to capture after the stop trigger.

Return values

0	Success.
Non-zero	Failure.

Referenced by main(), and run_test_9().

4.2.2.39 DM35424LIB_API int DM35424_Adc_Set_Pre_Trigger_Samples (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* uint32_t *count*)

Set the amount of data to capture prior to start trigger.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor, which contains the offsets to command sections of the board.
count	Number of samples to capture prior to the start trigger.

Note

The amount of data that can be captured prior to the start trigger is limited by the size of the FIFO. Consult the user's manual for this information.

Return values

0	Success.
Non-zero	Failure. errno may be set as follows: • `EINVAL` The size is not within the valid value range.

Referenced by main(), and run_test_9().

4.2.2.40 DM35424LIB_API int DM35424_Adc_Set_Sample_Rate (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* uint32_t *rate,* uint32_t * *actual_rate*)

Set the sampling rate for the ADC.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor, which contains the offsets to command sections of the board.
rate	The requested sampling rate for the ADC (Hz).
actual_rate	Pointer to the actual rate achieved by the ADC (Hz). Due to divider and clock values, the actual rate will rarely ever be the exact same as the requested rate.

Return values

0	Success.
Non-zero	Failure. errno may be set as follows: • EINVAL Asked for an invalid sampling rate (negative or 0) ERANGE Requested sampling rate is outside of the possible range for this ADC.

Referenced by main(), and setup_adc().

4.2.2.41 DM35424LIB_API int DM35424_Adc_Set_Start_Trigger (struct DM35424_Board_Descriptor * *handle,* struct DM35424_Function_Block * *func_block,* uint8_t *start_trigger*)

Set the start trigger for data collection.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor, which contains the offsets to command sections of the board.
start_trigger	Trigger to start capturing values. See the hardware manual for valid trigger values.

Return values

0	Success.
Non-zero	Failure. errno may be set as follows: • EINVAL An invalid value was passed for a start trigger

Referenced by main(), run_test_9(), and setup_adc().

4.2.2.42 DM35424LIB_API int DM35424_Adc_Set_Stop_Trigger (struct DM35424_Board_Descriptor * *handle,* struct DM35424_Function_Block * *func_block,* uint8_t *stop_trigger*)

Set the stop trigger for data collection.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor, which contains the offsets to command sections of the board.
stop_trigger	Trigger to stop capturing values. See the hardware manual for valid trigger values.

Return values

0	Success.
Non-zero	Failure. errno may be set as follows: • EINVAL An invalid value was passed for a stop trigger

Referenced by main(), run_test_9(), and setup_adc().

4.2.2.43 DM35424LIB_API int DM35424_Adc_Start (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block*)

Set the ADC mode to Start.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor, which contains the offsets to command sections of the board.

Return values

0	Success.
Non-zero	Failure.

Referenced by main(), and run_test_9().

4.2.2.44 DM35424LIB_API int DM35424_Adc_Start_Rearm (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block*)

Set the ADC mode to Start-Rearm.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor, which contains the offsets to command sections of the board.

Return values

0	Success.
Non-zero	Failure.

4.2.2.45 DM35424LIB_API int DM35424_Adc_Uninitialize (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block*)

Set the ADC mode to Uninitialized.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor, which contains the offsets to command sections of the board.

Return values

0	Success.
Non-zero	Failure.

Referenced by main().

4.2.2.46 DM35424LIB_API int DM35424_Adc_Volts_To_Sample (enum DM35424_Input_Ranges *input_range,* float *volts,* int32_t * *adc_sample*)

Convert volts to an ADC value.

Parameters

input_range	Enumerated value indicating what range the ADC channel has been set to
volts	Value to be converted to counts.
adc_sample	Pointer to the returned ADC count value.

Return values

0	Success.
Non-zero	Failure. errno may be set as follows: • `ENODEV` The function block passed in is the wrong subtype

4.3 DM35424 Board Access Structures

Data Structures

- struct DM35424_DMA_Descriptor

  Descriptor for the DMA on this board.
- struct DM35424_Function_Block

  DM35424 function block descriptor. This structure holds information about a function block, including type, number of DMA channels and buffers, descriptors for each DMA channel, and memory offsets to various control locations.

Functions

- DM35424LIB_API int DM35424_Board_Open (uint8_t dev_num, struct DM35424_Board_Descriptor **handle)

  Open the board, providing the file descriptor that all future operations will reference. Also allocate memory for the device descriptor.
- DM35424LIB_API int DM35424_Board_Close (struct DM35424_Board_Descriptor *handle)

  Close the board, closing the open handle for the device file, and freeing the memory allocated for the decriptor.
- DM35424LIB_API int DM35424_Read (struct DM35424_Board_Descriptor *handle, union dm35424_ioctl_argument *ioctl_request)

  Read from the board.
- DM35424LIB_API int DM35424_Write (struct DM35424_Board_Descriptor *handle, union dm35424_ioctl_argument *ioctl_request)

  Write to the board.
- DM35424LIB_API int DM35424_Modify (struct DM35424_Board_Descriptor *handle, union dm35424_ioctl_argument *ioctl_request)

  Read/Modify/Write to the board.
- int DM35424_Dma (struct DM35424_Board_Descriptor *handle, union dm35424_ioctl_argument *ioctl_request)

  Perform a DMA operation.

4.3.1 Detailed Description

4.3.2 Function Documentation

4.3.2.1 DM35424LIB_API int DM35424_Board_Close (struct DM35424_Board_Descriptor * *handle*)

Close the board, closing the open handle for the device file, and freeing the memory allocated for the decriptor.

Parameters

handle	Pointer to the device descriptor, which contains the open file id.

Return values

0	Success.
-1	Failure. errno may be set as follows: • `ENODATA` Device handle is null.

Referenced by main(), and run_test_9().

4.3.2.2 DM35424LIB_API int DM35424_Board_Open (uint8_t *dev_num,* struct DM35424_Board_Descriptor ** *handle*)

Open the board, providing the file descriptor that all future operations will reference. Also allocate memory for the device descriptor.

Parameters

dev_num	The minor number of the device being opened.
handle	Address of the handle pointer, which will contain the device descriptor.

Return values

0	Success.
-1	Failure. errno may be set as follows: • EBUSY Cannot open specified device. ENOMEM Cannot allocate memory for device descriptor.

Referenced by main(), and run_test_9().

4.3.2.3 int DM35424_Dma (struct DM35424_Board_Descriptor * *handle,* union dm35424_ioctl_argument * *ioctl_request*)

Perform a DMA operation.

Parameters

handle	Pointer to the device descriptor, which contains the open file id.
ioctl_request	Structure holding all required information for request to complete, including a DMA descriptor.

Return values

0	Success.
-1	Failure.

Warning

This function is not compatible with the Windows driver package and is therefore not included in the Windows DLL.

4.3.2.4 DM35424LIB_API int DM35424_Modify (struct DM35424_Board_Descriptor * *handle,* union dm35424_ioctl_argument * *ioctl_request*)

Read/Modify/Write to the board.

Parameters

handle	Pointer to the device descriptor, which contains the open file id.
ioctl_request	Structure holding all required information for the modify to complete, including register offset, data size, PCI region number, value mask, and data to write

Return values

0	Success.
-1	Failure.

4.3.2.5 DM35424LIB_API int DM35424_Read (struct DM35424_Board_Descriptor * *handle,* union dm35424_ioctl_argument * *ioctl_request*)

Read from the board.

Parameters

handle	Pointer to the device descriptor, which contains the open file id.
ioctl_request	Structure holding all required information for the read to complete, including register offset, data size, PCI region number, and pointer for returned data.

Return values

0	Success.
-1	Failure.

4.3.2.6 DM35424LIB_API int DM35424_Write (struct DM35424_Board_Descriptor * *handle,* union dm35424_ioctl_argument * *ioctl_request*)

Write to the board.

Parameters

handle	Pointer to the device descriptor, which contains the open file id.
ioctl_request	Structure holding all required information for the write to complete, including register offset, data size, PCI region number, and data to write.

Return values

0	Success.
-1	Failure.

4.4 DM35424 PCI Region Structures

Data Structures

- struct dm35424_pci_access_request

  PCI region access request descriptor. This structure holds information about a request to read data from or write data to one of a device's PCI regions.

- struct dm35424_ioctl_region_readwrite

  ioctl() request structure for read from or write to PCI region

- struct dm35424_ioctl_region_modify

  ioctl() request structure for PCI region read/modify/write

- struct dm35424_ioctl_interrupt_info_request

  ioctl() request structure for interrupt

- struct dm35424_ioctl_dma

  ioctl() request structure for DMA

- union dm35424_ioctl_argument

  ioctl() request structure encapsulating all possible requests. This is what gets passed into the kernel from user space on the ioctl() call.

Enumerations

- enum dm35424_pci_region_num { DM35424_PCI_REGION_GBC = 0, DM35424_PCI_REGION_GBC2, DM35424_PCI_REGION_FB }

  Standard PCI region number.

- enum dm35424_pci_region_access_size { DM35424_PCI_REGION_ACCESS_8 = 0, DM35424_PCI_REGION_ACCESS_16, DM35424_PCI_REGION_ACCESS_32 }

  Desired size in bits of access to standard PCI region.

- enum DM35424_DMA_FUNCTIONS { DM35424_DMA_INITIALIZE, DM35424_DMA_READ, DM35424_DMA_WRITE }

  Enumeration for DMA functions that can be requested for the driver to perform.

4.4.1 Detailed Description

4.4.2 Enumeration Type Documentation

4.4.2.1 enum DM35424_DMA_FUNCTIONS

Enumeration for DMA functions that can be requested for the driver to perform.

Enumerator

DM35424_DMA_INITIALIZE Initialize the DMA buffers

DM35424_DMA_READ Read from the DMA buffers (transfer to user space)

DM35424_DMA_WRITE Write to the DMA buffers (transfer from user space)

Definition at line 96 of file dm35424_board_access_structs.h.

4.4.2.2 enum dm35424_pci_region_access_size

Desired size in bits of access to standard PCI region.

Enumerator

DM35424_PCI_REGION_ACCESS_8 8-bit access

DM35424_PCI_REGION_ACCESS_16 16-bit access

DM35424_PCI_REGION_ACCESS_32 32-bit access

Definition at line 68 of file dm35424_board_access_structs.h.

4.4.2.3 enum dm35424_pci_region_num

Standard PCI region number.

Enumerator

DM35424_PCI_REGION_GBC General Board Control Registers (BAR0)

DM35424_PCI_REGION_GBC2 General Board Control Registers (64-bit) (BAR1)

DM35424_PCI_REGION_FB Functional Blocks Registers (BAR2)

Definition at line 40 of file dm35424_board_access_structs.h.

4.5 DM35424 DAC Library Constants

Macros

- #define DM35424_DAC_INT_CONVERSION_SENT_MASK 0x01

  Register value for Interrupt Mask - Conversion Sent.
- #define DM35424_DAC_INT_CHAN_MARKER_MASK 0x02

  Register value for Interrupt Mask - Channel has enabled marker.
- #define DM35424_DAC_INT_START_TRIG_MASK 0x08

  Register value for Interrupt Mask - Start Trigger Occurred.
- #define DM35424_DAC_INT_STOP_TRIG_MASK 0x10

  Register value for Interrupt Mask - Stop Trigger Occurred.
- #define DM35424_DAC_INT_POST_STOP_DONE_MASK 0x20

  Register value for Interrupt Mask - Post-Stop Conversions Completed.
- #define DM35424_DAC_INT_PACER_TICK_MASK 0x80

  Register value for Interrupt Mask - Pacer Clock Tick.
- #define DM35424_DAC_INT_ALL_MASK 0xBB

  Register value for Interrupt Mask - All Bits.
- #define DM35424_DAC_MODE_RESET 0x00

  Register value for Mode - Reset.
- #define DM35424_DAC_MODE_PAUSE 0x01

  Register value for Mode - Pause.
- #define DM35424_DAC_MODE_GO_SINGLE_SHOT 0x02

  Register value for Mode - Go (Single Shot)
- #define DM35424_DAC_MODE_GO_REARM 0x03

  Register value for Mode - Go (Re-arm)
- #define DM35424_DAC_STATUS_STOPPED 0x00

  Register value for DAC Status - Stopped.
- #define DM35424_DAC_STATUS_WAITING_START_TRIG 0x02

  Register value for DAC Status - Waiting for Start Trigger.
- #define DM35424_DAC_STATUS_CONVERTING 0x03

  Register value for DAC Status - Converting Data.
- #define DM35424_DAC_STATUS_OUTPUT_POST 0x04

  Register value for DAC Status - Outputting Post-Stop conversions.
- #define DM35424_DAC_STATUS_WAITING_REARM 0x05

  Register value for DAC Status - Waiting for Re-Arm.
- #define DM35424_DAC_STATUS_DONE 0x07

  Register value for DAC Status - Done.
- #define DM35424_DAC_MAX_RATE 106000

  Max allowable rate for the DAC (Hz)
- #define DM35424_DAC_MAX_VALUE 32767

  Max value of the DAC.
- #define DM35424_DAC_MIN_VALUE -32768

  Min value of the DAC.
- #define DM35424_DAC_LSB_AT_MIN_RANGE 0.000152587890625f

  DAC LSB (at lowest voltage range)

Enumerations

- enum DM35424_Dac_Clock_Events {
  DM35424_DAC_CLK_BUS_SRC_DISABLE = 0x00, DM35424_DAC_CLK_BUS_SRC_CONVERSION_SENT = 0x80, DM35424_DAC_CLK_BUS_SRC_CHAN_MARKER = 0x81, DM35424_DAC_CLK_BUS_SRC_START_TRIG = 0x83,
  DM35424_DAC_CLK_BUS_SRC_STOP_TRIG = 0x84, DM35424_DAC_CLK_BUS_SRC_CONV_COMPL = 0x85 }

  Clocking events that can be used as the global clock sources.

4.5.1 Detailed Description

Functions

4.5.2 Enumeration Type Documentation

4.5.2.1 enum DM35424_Dac_Clock_Events

Clocking events that can be used as the global clock sources.

Enumerator

DM35424_DAC_CLK_BUS_SRC_DISABLE Register value for Clock Event - Disabled.

DM35424_DAC_CLK_BUS_SRC_CONVERSION_SENT Register value for Clock Event - Conversion Sent.

DM35424_DAC_CLK_BUS_SRC_CHAN_MARKER Register value for Clock Event - Channel has enabled marker.

DM35424_DAC_CLK_BUS_SRC_START_TRIG Register value for Clock Event - Start Trigger Occurred.

DM35424_DAC_CLK_BUS_SRC_STOP_TRIG Register value for Clock Event - Stop Trigger Occurred.

DM35424_DAC_CLK_BUS_SRC_CONV_COMPL Register value for Clock Event - Conversions Complete.

Definition at line 181 of file dm35424_dac_library.h.

4.6 DM35424 DAC Library Public Functions

Functions

- DM35424LIB_API int DM35424_Dac_Open (struct DM35424_Board_Descriptor *handle, unsigned int number_of_type, struct DM35424_Function_Block *func_block)

  Open the DAC indicated, and determine register locations of control blocks needed to control it.
- DM35424LIB_API int DM35424_Dac_Set_Clock_Src (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, enum DM35424_Clock_Sources source)

  Set the clock source of the DAC.
- DM35424LIB_API int DM35424_Dac_Get_Clock_Src (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, enum DM35424_Clock_Sources *source)

  Get the clock source of the DAC.
- DM35424LIB_API int DM35424_Dac_Get_Clock_Div (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint32_t *divider)

  Get the clock divider value.
- DM35424LIB_API int DM35424_Dac_Set_Clock_Div (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint32_t divider)

  Set the clock divider value.
- DM35424LIB_API int DM35424_Dac_Set_Conversion_Rate (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint32_t requested_rate, uint32_t *actual_rate)

  Set the conversion rate of this DAC.
- DM35424LIB_API int DM35424_Dac_Interrupt_Set_Config (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint16_t interrupt_src, int enable)

  Set the interrupt configuration for this DAC.
- DM35424LIB_API int DM35424_Dac_Interrupt_Get_Config (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint16_t *interrupt_ena)

  Get the interrupt configuration for this DAC.
- DM35424LIB_API int DM35424_Dac_Set_Start_Trigger (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint8_t trigger_value)

  Set the start trigger.
- DM35424LIB_API int DM35424_Dac_Set_Stop_Trigger (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint8_t trigger_value)

  Set the stop trigger.
- DM35424LIB_API int DM35424_Dac_Get_Start_Trigger (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint8_t *trigger_value)

  Get the start trigger.
- DM35424LIB_API int DM35424_Dac_Get_Stop_Trigger (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint8_t *trigger_value)

  Get the stop trigger.
- DM35424LIB_API int DM35424_Dac_Start (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block)

  Set the DAC Mode to Start.
- DM35424LIB_API int DM35424_Dac_Reset (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block)

  Set the DAC Mode to Reset.
- DM35424LIB_API int DM35424_Dac_Pause (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block)

  Set the DAC Mode to Pause.
- DM35424LIB_API int DM35424_Dac_Get_Mode_Status (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint8_t *mode_status)

  Get the Mode and Status of the DAC.

- DM35424LIB_API int DM35424_Dac_Get_Last_Conversion (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, uint8_t *marker, int16_t *value)

  Get the value of the last conversion of the DAC.
- DM35424LIB_API int DM35424_Dac_Set_Last_Conversion (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, uint8_t marker, int16_t value)

  Set a value to be converted by the DAC immediately.
- DM35424LIB_API int DM35424_Dac_Get_Conversion_Count (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint32_t *value)

  Get a count of the number of conversions that DAC has executed.
- DM35424LIB_API int DM35424_Dac_Interrupt_Get_Status (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint16_t *value)

  Get a interrupt status register of the DAC.
- DM35424LIB_API int DM35424_Dac_Interrupt_Clear_Status (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint16_t value)

  Clear the interrupt status register of the DAC.
- DM35424LIB_API int DM35424_Dac_Set_Post_Stop_Conversion_Count (struct DM35424_Board_-Descriptor *handle, const struct DM35424_Function_Block *func_block, uint32_t value)

  Set the number of conversions the DAC will make after a stop trigger.
- DM35424LIB_API int DM35424_Dac_Get_Post_Stop_Conversion_Count (struct DM35424_Board_-Descriptor *handle, const struct DM35424_Function_Block *func_block, uint32_t *value)

  Get the number of conversions the DAC will make after a stop trigger.
- DM35424LIB_API int DM35424_Dac_Set_Clock_Source_Global (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, enum DM35424_Clock_Sources clock, enum DM35424_Dac_Clock_Events clock_driver)

  Set the source that will drive the global clock.
- DM35424LIB_API int DM35424_Dac_Channel_Set_Marker_Config (struct DM35424_Board_Descriptor *handle, struct DM35424_Function_Block *func_block, unsigned int channel, uint8_t marker_enable)

  Set the configuration of the marker interrupts for this channel.
- DM35424LIB_API int DM35424_Dac_Channel_Get_Marker_Config (struct DM35424_Board_Descriptor *handle, struct DM35424_Function_Block *func_block, unsigned int channel, uint8_t *marker_enable)

  Get the configuration of the marker interrupts for this channel.
- DM35424LIB_API int DM35424_Dac_Channel_Get_Marker_Status (struct DM35424_Board_Descriptor *handle, struct DM35424_Function_Block *func_block, unsigned int channel, uint8_t *marker_status)

  Get the status of the marker interrupts for this channel.
- DM35424LIB_API int DM35424_Dac_Channel_Clear_Marker_Status (struct DM35424_Board_Descriptor *handle, struct DM35424_Function_Block *func_block, unsigned int channel, uint8_t marker_to_clear)

  Clear the marker interrupts for this channel.
- DM35424LIB_API int DM35424_Dac_Fifo_Channel_Write (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, int32_t value)

  Write a value to the onboard FIFO.
- DM35424LIB_API int DM35424_Dac_Volts_To_Conv (float volts, int16_t *dac_conversion)

  Convert a value in volts to a DAC equivalent signed value.
- DM35424LIB_API int DM35424_Dac_Conv_To_Volts (int16_t conversion, float *volts)

  Convert a DAC conversion value to volts.

4.6.1 Detailed Description

DM35424_Dac_Library_Constants

4.6.2 Function Documentation

4.6.2.1 DM35424LIB_API int DM35424_Dac_Channel_Clear_Marker_Status (struct DM35424_Board_Descriptor * *handle,* struct DM35424_Function_Block * *func_block,* unsigned int *channel,* uint8_t *marker_to_clear*)

Clear the marker interrupts for this channel.

Parameters

handle	Pointer to the board handle
func_block	Pointer to the function block.
channel	The channel to change.
marker_to_clear	Bit values indicating which bits in register to clear.

Return values

0	Success.
Non-zero	

4.6.2.2 DM35424LIB_API int DM35424_Dac_Channel_Get_Marker_Config (struct DM35424_Board_Descriptor * *handle,* struct DM35424_Function_Block * *func_block,* unsigned int *channel,* uint8_t * *marker_enable*)

Get the configuration of the marker interrupts for this channel.

Parameters

handle	Pointer to the board handle
func_block	Pointer to the function block.
channel	The channel to change.
marker_enable	Pointer to returned marker interrupt config.

Return values

0	Success.
Non-zero	

4.6.2.3 DM35424LIB_API int DM35424_Dac_Channel_Get_Marker_Status (struct DM35424_Board_Descriptor * *handle,* struct DM35424_Function_Block * *func_block,* unsigned int *channel,* uint8_t * *marker_status*)

Get the status of the marker interrupts for this channel.

Parameters

handle	Pointer to the board handle
func_block	Pointer to the function block.
channel	The channel to change.
marker_status	Pointer to returned marker status.

Return values

0	Success.
Non-zero	

4.6.2.4 DM35424LIB_API int DM35424_Dac_Channel_Set_Marker_Config (struct DM35424_Board_Descriptor * *handle,* struct DM35424_Function_Block * *func_block,* unsigned int *channel,* uint8_t *marker_enable*)

Set the configuration of the marker interrupts for this channel.

Parameters

handle	Pointer to the board handle

func_block	Pointer to the function block.
channel	The channel to change.
marker_enable	Bit values indicating whether to enable marker interrupts (1) or disable (0).

Return values

0	Success.
Non-zero	

Referenced by run_test_9().

4.6.2.5 DM35424LIB_API int DM35424_Dac_Conv_To_Volts (int16_t *conversion*, float * *volts*)

Convert a DAC conversion value to volts.

Parameters

conversion	DAC converter signed value.
volts	The volts equivalent of the converter value.

Return values

0	Success.
Non-zero	Failure. errno may be set as follows: • EINVAL Function called by an unsupported function block.

Referenced by main().

4.6.2.6 DM35424LIB_API int DM35424_Dac_Fifo_Channel_Write (struct DM35424_Board_Descriptor * *handle*, const struct DM35424_Function_Block * *func_block*, unsigned int *channel*, int32_t *value*)

Write a value to the onboard FIFO.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor, which contains the offsets to command sections of the board.
channel	Channel to write the data to.
value	value to write.

Return values

0	Success.
Non-zero	Failure.

4.6.2.7 DM35424LIB_API int DM35424_Dac_Get_Clock_Div (struct DM35424_Board_Descriptor * *handle*, const struct DM35424_Function_Block * *func_block*, uint32_t * *divider*)

Get the clock divider value.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.
divider	Pointer to the clock divider returned.

Return values

0	Success.
Non-Zero	Failure.

4.6.2.8 DM35424LIB_API int DM35424_Dac_Get_Clock_Src (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* enum DM35424_Clock_Sources * *source*)

Get the clock source of the DAC.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.
source	Pointer to the returned clock set for this DAC.

Return values

0	Success.
Non-Zero	Failure.

4.6.2.9 DM35424LIB_API int DM35424_Dac_Get_Conversion_Count (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* uint32_t * *value*)

Get a count of the number of conversions that DAC has executed.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.
value	Pointer to the returned count of conversions executed.

Return values

0	Success.
Non-Zero	Failure.

Referenced by run_test_9().

4.6.2.10 DM35424LIB_API int DM35424_Dac_Get_Last_Conversion (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* unsigned int *channel,* uint8_t * *marker,* int16_t * *value*)

Get the value of the last conversion of the DAC.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.
channel	DAC channel that we want the last conversion value from.
marker	Pointer to the returned value for the marker byte.
value	Pointer to the returned signed value of the last conversion.

Return values

0	Success.
Non-Zero	Failure.

Referenced by main().

4.6.2.11 DM35424LIB_API int DM35424_Dac_Get_Mode_Status (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* uint8_t * *mode_status*)

Get the Mode and Status of the DAC.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.
mode_status	Pointer to the value of the returned mode_status register.

Return values

0	Success.
Non-Zero	Failure.

Referenced by run_test_9().

4.6.2.12 DM35424LIB_API int DM35424_Dac_Get_Post_Stop_Conversion_Count (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* uint32_t * *value*)

Get the number of conversions the DAC will make after a stop trigger.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.
value	Pointer to the returned number of conversions.

Return values

0	Success.
Non-Zero	Failure.

4.6.2.13 DM35424LIB_API int DM35424_Dac_Get_Start_Trigger (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* uint8_t * *trigger_value*)

Get the start trigger.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.
trigger_value	Pointer to the returned rigger value (event) that will initiate conversions on this DAC.

Return values

0	Success.
Non-Zero	Failure.

4.6.2.14 DM35424LIB_API int DM35424_Dac_Get_Stop_Trigger (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* uint8_t * *trigger_value*)

Get the stop trigger.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.
trigger_value	Pointer to the returned trigger value (event) that will halt conversions on this DAC.

Return values

0	Success.
Non-Zero	Failure.

4.6.2.15 DM35424LIB_API int DM35424_Dac_Interrupt_Clear_Status (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* uint16_t *value*)

Clear the interrupt status register of the DAC.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.
value	Bitmask indicating which interrupts to clear.

Return values

0	Success.
Non-Zero	Failure.

4.6.2.16 DM35424LIB_API int DM35424_Dac_Interrupt_Get_Config (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* uint16_t * *interrupt_ena*)

Get the interrupt configuration for this DAC.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.

func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.
interrupt_ena	Pointer to the returned bitmask indicating which interrupts are set.

Return values

0	Success.
Non-Zero	Failure.

4.6.2.17 DM35424LIB_API int DM35424_Dac_Interrupt_Get_Status (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* uint16_t * *value*)

Get a interrupt status register of the DAC.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.
value	Pointer to the returned interrupt status.

Return values

0	Success.
Non-Zero	Failure.

4.6.2.18 DM35424LIB_API int DM35424_Dac_Interrupt_Set_Config (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* uint16_t *interrupt_src,* int *enable*)

Set the interrupt configuration for this DAC.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.
interrupt_src	Bitmask indicating which interrupts to set.
enable	A boolean value indicating whether selected interrupts are to be enabled or disabled.

Return values

0	Success.
Non-Zero	Failure.

4.6.2.19 DM35424LIB_API int DM35424_Dac_Open (struct DM35424_Board_Descriptor * *handle,* unsigned int *number_of_type,* struct DM35424_Function_Block * *func_block*)

Open the DAC indicated, and determine register locations of control blocks needed to control it.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
number_of_type	Which DAC to open. The first DAC on the board will be 0.

func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.

Return values

0	Success.
Non-Zero	Failure.

Referenced by main(), and setup_dacs_and_start().

4.6.2.20 DM35424LIB_API int DM35424_Dac_Pause (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block*)

Set the DAC Mode to Pause.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.

Return values

0	Success.
Non-Zero	Failure.

4.6.2.21 DM35424LIB_API int DM35424_Dac_Reset (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block*)

Set the DAC Mode to Reset.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.

Return values

0	Success.
Non-Zero	Failure.

Referenced by main(), and run_test_9().

4.6.2.22 DM35424LIB_API int DM35424_Dac_Set_Clock_Div (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* uint32_t *divider*)

Set the clock divider value.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.
divider	Divider value to set this DAC clock to.

Return values

0	Success.
Non-Zero	Failure.

Referenced by run_test_9().

4.6.2.23 DM35424LIB_API int DM35424_Dac_Set_Clock_Source_Global (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* enum DM35424_Clock_Sources *clock,* enum DM35424_Dac_Clock_Events *clock_driver*)

Set the source that will drive the global clock.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.
clock	Which global clock to set.
clock_driver	Source to drive global clock.

Return values

0	Success.
Non-Zero	Failure. errno may be set as follows: • `EINVAL` Invalid clock or source requested.

Referenced by run_test_9().

4.6.2.24 DM35424LIB_API int DM35424_Dac_Set_Clock_Src (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* enum DM35424_Clock_Sources *source*)

Set the clock source of the DAC.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.
source	The clock source that we want to set for this DAC.

Return values

0	Success.
Non-Zero	Failure.

Referenced by main(), run_test_9(), and setup_dacs_and_start().

4.6.2.25 DM35424LIB_API int DM35424_Dac_Set_Conversion_Rate (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* uint32_t *requested_rate,* uint32_t * *actual_rate*)

Set the conversion rate of this DAC.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.
requested_rate	Requested rate of conversion for the DAC (Hz).
actual_rate	Pointer to the returned value of the actual rate achieved (Hz).

Note

The actual obtainable rate depends on many board-specific values and clocks, and so the returned rate will rarely be the exact same as the requested rate.

Return values

0	Success.
Non-Zero	Failure.

errno may be set as follows:

- `EINVAL` Invalid rate requested.

Referenced by main(), and setup_dacs_and_start().

4.6.2.26 DM35424LIB_API int DM35424_Dac_Set_Last_Conversion (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* unsigned int *channel,* uint8_t *marker,* int16_t *value*)

Set a value to be converted by the DAC immediately.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.
channel	DAC channel that we want the last conversion set to.
marker	Value of the marker bits (top 8 bits)
value	Value to be converted by DAC and set on its output pin.

Note

The DAC will set its output value to the last conversion register value only if the DAC is in Reset mode.

Return values

0	Success.
Non-Zero	Failure.

Referenced by main(), and run_test_9().

4.6.2.27 DM35424LIB_API int DM35424_Dac_Set_Post_Stop_Conversion_Count (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* uint32_t *value*)

Set the number of conversions the DAC will make after a stop trigger.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.

func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.
value	Number of conversions.

Return values

0	Success.
Non-Zero	Failure.

Referenced by run_test_9().

4.6.2.28 DM35424LIB_API int DM35424_Dac_Set_Start_Trigger (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* uint8_t *trigger_value*)

Set the start trigger.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.
trigger_value	Trigger value (event) that will initiate conversions on this DAC.

Return values

0	Success.
Non-Zero	Failure.

Referenced by main(), run_test_9(), and setup_dacs_and_start().

4.6.2.29 DM35424LIB_API int DM35424_Dac_Set_Stop_Trigger (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* uint8_t *trigger_value*)

Set the stop trigger.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.
trigger_value	Trigger value (event) that will halt conversions on this DAC.

Return values

0	Success.
Non-Zero	Failure.

Referenced by main(), run_test_9(), and setup_dacs_and_start().

4.6.2.30 DM35424LIB_API int DM35424_Dac_Start (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block*)

Set the DAC Mode to Start.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.

Return values

0	Success.
Non-Zero	Failure.

Referenced by main(), run_test_9(), and setup_dacs_and_start().

4.6.2.31 DM35424LIB_API int DM35424_Dac_Volts_To_Conv (float *volts*, int16_t * *dac_conversion*)

Convert a value in volts to a DAC equivalent signed value.

Parameters

volts	The volts value we want the DAC to output.
dac_conversion	Pointer to signed value representing the equivalent of the volts.

Return values

0	Success.
Non-zero	Failure. errno may be set as follows: • EINVAL Function called by an unsupported function block.

Referenced by main(), and setup_dacs_and_start().

4.7 DM35424 DIO Public

Functions

- DM35424LIB_API int DM35424_Dio_Open (struct DM35424_Board_Descriptor *handle, unsigned int number_of_type, struct DM35424_Function_Block *func_block)

  Open the DIO indicated, and determine register locations of control blocks needed to control it.
- DM35424LIB_API int DM35424_Dio_Set_Direction (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint32_t direction)

  Set the direction of the DIO pins.
- DM35424LIB_API int DM35424_Dio_Get_Direction (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint32_t *direction)

  Get the direction of the DIO pins.
- DM35424LIB_API int DM35424_Dio_Get_Input_Value (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint32_t *value)

  Get the input value of the DIO.
- DM35424LIB_API int DM35424_Dio_Get_Output_Value (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint32_t *value)

  Get the current value of the output register.
- DM35424LIB_API int DM35424_Dio_Set_Output_Value (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint32_t value)

  Set the value to be put on output pins.

4.7.1 Detailed Description

Library Functions

4.7.2 Function Documentation

4.7.2.1 DM35424LIB_API int DM35424_Dio_Get_Direction (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* uint32_t * *direction*)

Get the direction of the DIO pins.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block representing the DIO.
direction	Bitmask representing the directions of the pins. (0 = input, 1 = output)

Return values

0	Success.
Non-Zero	Failure.

4.7.2.2 DM35424LIB_API int DM35424_Dio_Get_Input_Value (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* uint32_t * *value*)

Get the input value of the DIO.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block representing the DIO.
value	Pointer to returned value.

Note

The value of pins that are set to output will be zero.

Return values

0	Success.
Non-Zero	Failure.

Referenced by main().

4.7.2.3 DM35424LIB_API int DM35424_Dio_Get_Output_Value (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* uint32_t * *value*)

Get the current value of the output register.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block representing the DIO.
value	Pointer to returned value.

Note

The value of pins that are set to input will be zero.

Return values

0	Success.
Non-Zero	Failure.

4.7.2.4 DM35424LIB_API int DM35424_Dio_Open (struct DM35424_Board_Descriptor * *handle,* unsigned int *number_of_type,* struct DM35424_Function_Block * *func_block*)

Open the DIO indicated, and determine register locations of control blocks needed to control it.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
number_of_type	Which DIO to open. The first DIO on the board will be 0.
func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.

Return values

0	Success.
Non-Zero	Failure.

Referenced by main().

4.7.2.5 DM35424LIB_API int DM35424_Dio_Set_Direction (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* uint32_t *direction*)

Set the direction of the DIO pins.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block representing the DIO.
direction	Bitmask representing the directions to set the pins. (0 = input, 1 = output)

Return values

0	Success.
Non-Zero	Failure.

Referenced by main().

4.7.2.6 DM35424LIB_API int DM35424_Dio_Set_Output_Value (struct DM35424_Board_Descriptor * *handle*, const struct DM35424_Function_Block * *func_block*, uint32_t *value*)

Set the value to be put on output pins.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block representing the DIO.
value	Value to be written to output pins.

Note

Writing a bit to a pin set to input will have no effect.

Return values

0	Success.
Non-Zero	Failure.

Referenced by main().

4.8 DM35424 DMA Public Library Constants

Macros

- #define DM35424_DMA_ACTION_CLEAR 0x00

  Register value for DMA clear action.
- #define DM35424_DMA_ACTION_GO 0x01

  Register value for DMA go action.
- #define DM35424_DMA_ACTION_PAUSE 0x02

  Register value for DMA pause action.
- #define DM35424_DMA_ACTION_HALT 0x03

  Register value for DMA halt action.
- #define DM35424_DMA_SETUP_DIRECTION_READ 0x04

  Register value to set DMA to READ direction.
- #define DM35424_DMA_SETUP_DIRECTION_WRITE 0x00

  Register value to set DMA to WRITE direction.
- #define DM35424_DMA_SETUP_DIRECTION_MASK 0x04

  Register value to set DMA to READ direction.
- #define DM35424_DMA_SETUP_IGNORE_USED 0x08

  Register value to tell DMA to ignore used buffers.
- #define DM35424_DMA_SETUP_NOT_IGNORE_USED 0x00

  Register value to tell DMA to not ignore used buffers.
- #define DM35424_DMA_SETUP_IGNORE_USED_MASK 0x08

  Bit mask for Ignore Used bit in setup register.
- #define DM35424_DMA_SETUP_INT_ENABLE 0x01

  Register value to enabled interrupts in the setup register.
- #define DM35424_DMA_SETUP_INT_DISABLE 0x00

  Register value to disable interrupts in the setup register.
- #define DM35424_DMA_SETUP_INT_MASK 0x01

  Bit mask for the interrupt bit in the setup register.
- #define DM35424_DMA_SETUP_ERR_INT_ENABLE 0x02

  Register value to enable the error interrupt.
- #define DM35424_DMA_SETUP_ERR_INT_DISABLE 0x00

  Register value to disable the error interrupt.
- #define DM35424_DMA_SETUP_ERR_INT_MASK 0x02

  Bit mask for the error interrupt bit in the setup register.
- #define DM35424_DMA_STATUS_CLEAR 0x00

  Register value to write to status registers to clear them.
- #define DM35424_DMA_CTRL_CLEAR 0x00

  Register value to write to control register to clear it.
- #define DM35424_DMA_BUFFER_STATUS_CLEAR 0x00

  Register value to write to the buffer status register to clear it.
- #define DM35424_DMA_BUFFER_CTRL_CLEAR 0x00

  Register value to write to the buffer control register to clear it.
- #define DM35424_DMA_BUFFER_STATUS_USED_MASK 0x01

  Bit mask for the used buffer bit in the buffer status register.
- #define DM35424_DMA_BUFFER_STATUS_TERM_MASK 0x02

  Bit mask for the terminated buffer bit in the buffer status register.
- #define DM35424_DMA_BUFFER_CTRL_VALID 0x01

  Register value to write to buffer control register to mark it as valid.
- #define DM35424_DMA_BUFFER_CTRL_HALT 0x02

Register value to write to buffer control register to tell DMA to halt after processing this buffer.

- #define DM35424_DMA_BUFFER_CTRL_LOOP 0x04

  Register value to write to buffer control register to tell DMA to loop back to buffer 0 after using this buffer.

- #define DM35424_DMA_BUFFER_CTRL_INTR 0x08

  Register value to write to buffer control register to tell DMA to issue an interrupt after using this buffer.

- #define DM35424_DMA_BUFFER_CTRL_PAUSE 0x10

  Register value to write to buffer control register to tell DMA to pause after processing this buffer.

- #define DM35424_DMA_CTRL_BLOCK_SIZE 0x10

  Constant value indicating DMA control block size.

- #define DM35424_DMA_BUFFER_CTRL_BLOCK_SIZE 0x10

  Constant value indicating DMA buffer control block size.

- #define DM35424_BIT_MASK_DMA_BUFFER_SIZE 0x0FFFFFF

  Bit mask for the DMA buffer size, since it is 24-bits of a 32-bit register.

Enumerations

- enum DM35424_Fifo_States { DM35424_FIFO_UNKNOWN, DM35424_FIFO_EMPTY, DM35424_FIFO_FULL, DM35424_FIFO_HAS_DATA }

  Descriptions of the possible states the FIFO might be in.

4.8.1 Detailed Description

4.8.2 Enumeration Type Documentation

4.8.2.1 enum DM35424_Fifo_States

Descriptions of the possible states the FIFO might be in.

Enumerator

DM35424_FIFO_UNKNOWN State of FIFO is unknown.

DM35424_FIFO_EMPTY FIFO is empty.

DM35424_FIFO_FULL FIFO is full

DM35424_FIFO_HAS_DATA FIFO is between empty and full

Definition at line 237 of file dm35424_dma_library.h.

4.9 DM35424 DMA Public Library Functions

Functions

- DM35424LIB_API int DM35424_Dma_Start (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel)

  Start the DMA.
- DM35424LIB_API int DM35424_Dma_Stop (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel)

  Stop the DMA.
- DM35424LIB_API int DM35424_Dma_Pause (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel)

  Pause the DMA.
- DM35424LIB_API int DM35424_Dma_Clear (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel)

  Clear the DMA.
- DM35424LIB_API int DM35424_Dma_Get_Fifo_Counts (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, uint16_t *write_count, uint16_t *read_count)

  Get the Read and Write FIFO count values.
- DM35424LIB_API int DM35424_Dma_Get_Fifo_State (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, enum DM35424_Fifo_States *state)

  Get the state of the FIFO.
- DM35424LIB_API int DM35424_Dma_Configure_Interrupts (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, int enable, int error_enable)

  Configure the interrupts for the DMA channel.
- DM35424LIB_API int DM35424_Dma_Get_Interrupt_Configuration (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, int *enable, int *error_enable)

  Get the configuration of the interrupts for the DMA channel.
- DM35424LIB_API int DM35424_Dma_Setup (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, int direction, int ignore_used)

  Setup the DMA channel, specifically the direction and if used buffers are ignored.
- DM35424LIB_API int DM35424_Dma_Setup_Set_Direction (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, int direction)

  Set the direction of the DMA, read or write.
- DM35424LIB_API int DM35424_Dma_Setup_Set_Used (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, int ignore_used)

  Set the DMA channel to ignore or not ignore a used buffer. Ignoring used buffers is mostly useful when outputting a repeating data cycle.
- DM35424LIB_API int DM35424_Dma_Get_Errors (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, int *stat_overflow, int *stat_underflow, int *stat_used, int *stat_invalid)

  Get the current value of the DMA channel error registers.
- DM35424LIB_API int DM35424_Dma_Status (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, uint32_t *current_buffer, uint32_t *current_count, int *current_action, int *stat_overflow, int *stat_underflow, int *stat_used, int *stat_invalid, int *stat_complete)

  Get the current status of the DMA channel. Determine which buffer it is using, what its current action is, and the state of all error conditions and normal interrupt conditions.
- DM35424LIB_API int DM35424_Dma_Get_Current_Buffer_Count (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, uint32_t *current_buffer, uint32_t *current_count)

  Get the current buffer and buffer count in use by the DMA.

- DM35424LIB_API int DM35424_Dma_Check_For_Error (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, int *has_error)

  Check the DMA channel for any error conditions. This just returns a simple boolean as quickly as possible. If there is an error condition, you will have to query the DMA again to determine what the error is.
- DM35424LIB_API int DM35424_Dma_Buffer_Setup (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, unsigned int buffer, uint8_t ctrl)

  Setup the DMA buffer for use.
- DM35424LIB_API int DM35424_Dma_Buffer_Status (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, unsigned int buffer, uint8_t *status, uint8_t *control, uint32_t *size)

  Get the status of the buffer. This gets the status, control, and size registers.
- DM35424LIB_API int DM35424_Dma_Check_Buffer_Used (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, unsigned int buffer_num, int *is_used)

  Check if the indicated buffer has the "Used" flag set.
- int DM35424_Dma_Find_Interrupt (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int *channel, int *channel_complete, int *channel_error)

  Find which DMA channel has an interrupt condition, whether from using a buffer with interrupt set, or from an error. DMA channels are evaluated starting at Channel 0.
- int DM35424_Dma_Clear_Interrupt (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, int clear_overflow, int clear_underflow, int clear_used, int clear_invalid, int clear_complete)

  Clear the interrupt flag from a DMA channel. Clearing the flags will allow another interrupt of the same type to occur again, and is the normal operation after handling the interrupt itself.
- int DM35424_Dma_Reset_Buffer (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, unsigned int buffer)

  Reset the DMA buffer, preparing it to be used again by the DMA engine.

4.9.1 Detailed Description

DM35424_Dma_Library_Constants

4.9.2 Function Documentation

4.9.2.1 DM35424LIB_API int DM35424_Dma_Buffer_Setup (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* unsigned int *channel,* unsigned int *buffer,* uint8_t *ctrl*)

Setup the DMA buffer for use.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.
channel	DMA Channel to set.
buffer	DMA buffer to set.
ctrl	Unsigned short containing control bits. Will be written to control register.

Return values

0	Success.

Non-Zero	Failure. errno may be set as follows: • EINVAL Invalid channel or buffer requested.

Referenced by main(), setup_adc(), and setup_dacs_and_start().

4.9.2.2 DM35424LIB_API int DM35424_Dma_Buffer_Status (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* unsigned int *channel,* unsigned int *buffer,* uint8_t * *status,* uint8_t * *control,* uint32_t * *size*)

Get the status of the buffer. This gets the status, control, and size registers.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.
channel	DMA Channel to get.
buffer	DMA buffer to get.
status	Pointer to returned DMA buffer status register value.
control	Pointer to returned DMA buffer control register value.
size	Pointer to returned DMA buffer size (in bytes).

Return values

0	Success.
Non-Zero	Failure. errno may be set as follows: • EINVAL Invalid channel or buffer requested.

Referenced by main(), and setup_adc().

4.9.2.3 DM35424LIB_API int DM35424_Dma_Check_Buffer_Used (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* unsigned int *channel,* unsigned int *buffer_num,* int * *is_used*)

Check if the indicated buffer has the "Used" flag set.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.
channel	DMA Channel to get.
buffer_num	Which buffer in the DMA channel to check.
is_used	Pointer to returned boolean indicating if the buffer has the Used flag set.

Return values

0	Success.
Non-Zero	Failure. errno may be set as follows: • EINVAL Invalid channel requested.

4.9.2.4 DM35424LIB_API int DM35424_Dma_Check_For_Error (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* unsigned int *channel,* int * *has_error*)

Check the DMA channel for any error conditions. This just returns a simple boolean as quickly as possible. If there is an error condition, you will have to query the DMA again to determine what the error is.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.
channel	DMA Channel to set.
has_error	Pointer to returned boolean value indicating if the DMA channel has an error condition.

Return values

0	Success.
Non-Zero	Failure. errno may be set as follows: • `EINVAL` Invalid channel requested.

4.9.2.5 DM35424LIB_API int DM35424_Dma_Clear (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* unsigned int *channel*)

Clear the DMA.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.
channel	DMA Channel to clear.

Return values

0	Success.
-1	Failure. errno may be set as follows: • `EBUSY` The action was not executed before timeout. EINVAL Invalid channel requested.

Referenced by main().

4.9.2.6 int DM35424_Dma_Clear_Interrupt (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* unsigned int *channel,* int *clear_overflow,* int *clear_underflow,* int *clear_used,* int *clear_invalid,* int *clear_complete*)

Clear the interrupt flag from a DMA channel. Clearing the flags will allow another interrupt of the same type to occur again, and is the normal operation after handling the interrupt itself.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.
channel	DMA channel to clear.
clear_overflow	Boolean indicating whether or not to clear the overflow interrupt status.
clear_underflow	Boolean indicating whether or not to clear the underflow interrupt status.
clear_used	Boolean indicating whether or not to clear the used interrupt status.

clear_invalid	Boolean indicating whether or not to clear the invalid buffer interrupt status.
clear_complete	Boolean indicating whether or not to clear the buffer completed interrupt status.

Return values

0	Success.
Non-Zero	Failure. errno may be set as follows: • `EINVAL` Invalid channel requested.

Warning

This function is not compatible with the Windows driver package and is therefore not included in the Windows DLL.

Referenced by ISR().

4.9.2.7 DM35424LIB_API int DM35424_Dma_Configure_Interrupts (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* unsigned int *channel,* int *enable,* int *error_enable*)

Configure the interrupts for the DMA channel.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.
channel	DMA Channel to configure interrupts for.
enable	Boolean value indicating if interrupt is to be enabled or disabled.
error_enable	Boolean value indicating if interrupts for error conditions are to be enabled or disabled.

Return values

0	Success.
-1	Failure. errno may be set as follows: • `EINVAL` Invalid channel requested.

Referenced by main(), and setup_adc().

4.9.2.8 int DM35424_Dma_Find_Interrupt (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* unsigned int * *channel,* int * *channel_complete,* int * *channel_error*)

Find which DMA channel has an interrupt condition, whether from using a buffer with interrupt set, or from an error. DMA channels are evaluated starting at Channel 0.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.
channel	Pointer to returned DMA channel with interrupt, if any.
channel_complete	Pointer to returned boolean indicating that the interrupt on this channel was from using a buffer with the interrupt bit set.
channel_error	Pointer to returned boolean indicating that the interrupt on this channel was from an error condition.

Return values

0	Success.
Non-Zero	Failure.

Warning

This function is not compatible with the Windows driver package and is therefore not included in the Windows DLL.

Referenced by ISR().

4.9.2.9 DM35424LIB_API int DM35424_Dma_Get_Current_Buffer_Count (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* unsigned int *channel,* uint32_t * *current_buffer,* uint32_t * *current_count*)

Get the current buffer and buffer count in use by the DMA.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.
channel	Channel to get buffer info from.
current_buffer	Pointer to the returned current buffer the DMA is using.
current_count	Pointer to the returned count for the current buffer. This indicates how far into the buffer the DMA is.

Return values

0	Success.
Non-Zero	Failure.

4.9.2.10 DM35424LIB_API int DM35424_Dma_Get_Errors (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* unsigned int *channel,* int * *stat_overflow,* int * *stat_underflow,* int * *stat_used,* int * *stat_invalid*)

Get the current value of the DMA channel error registers.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.
channel	DMA Channel to configure interrupts for.
stat_overflow	Pointer to the returned boolean indicating if overflow has occurred.
stat_underflow	Pointer to the returned boolean indicating if underflow has occurred.
stat_used	Pointer to the returned boolean indicating if the DMA attempted to use an already used buffer.
stat_invalid	Pointer to the returned boolean indicating if the DMA attempted to use an invalid buffer.

Return values

0	Success.
Non-Zero	Failure. errno may be set as follows: • `EINVAL` Invalid channel requested.

4.9.2.11 DM35424LIB_API int DM35424_Dma_Get_Fifo_Counts (struct DM35424_Board_Descriptor * *handle*, const struct DM35424_Function_Block * *func_block*, unsigned int *channel*, uint16_t * *write_count*, uint16_t * *read_count*)

Get the Read and Write FIFO count values.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.
channel	DMA Channel to get counts for.
write_count	Pointer to the returned number of bytes of space available in the FIFO.
read_count	Pointer to the returned number of bytes of data available in the FIFO.

Note

These counts are valid regardless of the direction of the DMA.

Return values

0	Success.
-1	Failure. errno may be set as follows: • `EINVAL` Invalid channel requested.

4.9.2.12 DM35424LIB_API int DM35424_Dma_Get_Fifo_State (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* unsigned int *channel,* enum DM35424_Fifo_States * *state*)

Get the state of the FIFO.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.
channel	DMA Channel to get state for.
state	Pointer to the returned FIFO state enumeration.

Note

This is just a convenience function that infers a FIFO state from the FIFO counts.

Return values

0	Success.
-1	Failure. errno may be set as follows: • `EINVAL` Invalid channel requested.

4.9.2.13 DM35424LIB_API int DM35424_Dma_Get_Interrupt_Configuration (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* unsigned int *channel,* int * *enable,* int * *error_enable*)

Get the configuration of the interrupts for the DMA channel.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.
channel	DMA Channel to configure interrupts for.
enable	Pointer to returned value indicating if interrupt is enabled or disabled.
error_enable	Pointer to returned boolean value indicating if interrupts for error conditions are enabled or disabled.

Return values

0	Success.
-1	Failure. errno may be set as follows: • EINVAL Invalid channel requested.

4.9.2.14 DM35424LIB_API int DM35424_Dma_Pause (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* unsigned int *channel*)

Pause the DMA.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.
channel	DMA Channel to pause.

Return values

0	Success.
-1	Failure. errno may be set as follows: • EBUSY The action was not executed before timeout. EINVAL Invalid channel requested.

4.9.2.15 int DM35424_Dma_Reset_Buffer (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* unsigned int *channel,* unsigned int *buffer*)

Reset the DMA buffer, preparing it to be used again by the DMA engine.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.
channel	DMA channel containing the buffer.
buffer	Buffer in channel to clear.

Return values

0	Success.
Non-Zero	Failure. errno may be set as follows: • EINVAL Invalid channel or buffer requested.

Warning

This function is not compatible with the Windows driver package and is therefore not included in the Windows DLL.

Referenced by ISR().

4.9.2.16 DM35424LIB_API int DM35424_Dma_Setup (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* unsigned int *channel,* int *direction,* int *ignore_used*)

Setup the DMA channel, specifically the direction and if used buffers are ignored.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.
channel	DMA Channel to set.
direction	Direction for DMA. See the DMA constants for possible values.
ignore_used	Boolean value indicating if used buffers should be ignored.

Note

This is a convenience function that accomplishes two of the setup steps in one function. This function is identical to calling the direction and ignore used library functions separately.

Return values

0	Success.
-1	Failure. errno may be set as follows: • EINVAL Invalid channel requested, or wrong direction requested for function block type.

Referenced by main(), setup_adc(), and setup_dacs_and_start().

4.9.2.17 DM35424LIB_API int DM35424_Dma_Setup_Set_Direction (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* unsigned int *channel,* int *direction*)

Set the direction of the DMA, read or write.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.
channel	DMA Channel to set.
direction	Direction for DMA. See the DMA constants for possible values.

Return values

0	Success.
-1	Failure. errno may be set as follows: • EINVAL Invalid channel requested, or wrong direction requested for function block type.

4.9.2.18 DM35424LIB_API int DM35424_Dma_Setup_Set_Used (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* unsigned int *channel,* int *ignore_used*)

Set the DMA channel to ignore or not ignore a used buffer. Ignoring used buffers is mostly useful when outputting a repeating data cycle.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.
channel	DMA Channel to set.
ignore_used	Boolean value indicating if used buffers should be ignored.

Return values

0	Success.
-1	Failure. errno may be set as follows: • EINVAL Invalid channel requested.

4.9.2.19 DM35424LIB_API int DM35424_Dma_Start (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* unsigned int *channel*)

Start the DMA.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.
channel	DMA Channel to start.

Return values

0	Success.
-1	Failure. errno may be set as follows: • EBUSY The action was not executed before timeout. EINVAL Invalid channel requested.

Referenced by main(), and setup_dacs_and_start().

4.9.2.20 DM35424LIB_API int DM35424_Dma_Status (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* unsigned int *channel,* uint32_t * *current_buffer,* uint32_t * *current_count,* int * *current_action,* int * *stat_overflow,* int * *stat_underflow,* int * *stat_used,* int * *stat_invalid,* int * *stat_complete*)

Get the current status of the DMA channel. Determine which buffer it is using, what its current action is, and the state of all error conditions and normal interrupt conditions.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.
channel	DMA Channel to configure interrupts for.
current_buffer	Pointer to the returned current buffer the DMA is using.
current_count	Pointer to the returned count for the current buffer. This indicates how far into the buffer the DMA is.

current_action	Pointer to the returned action the DMA is currently taking.
stat_overflow	Pointer to the returned boolean indicating if overflow has occurred.
stat_underflow	Pointer to the returned boolean indicating if underflow has occurred.
stat_used	Pointer to the returned boolean indicating if the DMA attempted to use an already used buffer.
stat_invalid	Pointer to the returned boolean indicating if the DMA attempted to use an invalid buffer.
stat_complete	Pointer to the returned boolean indicating if the DMA has completed using a buffer that had an interrupt set.

Return values

0	Success.
Non-Zero	Failure. errno may be set as follows: • `EINVAL` Invalid channel requested.

Referenced by main(), and output_channel_status().

4.9.2.21 DM35424LIB_API int DM35424_Dma_Stop (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* unsigned int *channel*)

Stop the DMA.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.
channel	DMA Channel to stop.

Return values

0	Success.
-1	Failure. errno may be set as follows: • `EBUSY` The action was not executed before timeout. EINVAL Invalid channel requested.

4.10 DM35424 Driver Constants

Macros

- #define DM35424_NAME_LENGTH 200

  DM35424 Max possible board name length.
- #define DM35424_PCI_NUM_REGIONS PCI_ROM_RESOURCE

  Number of standard PCI regions.
- #define DM35424_INT_QUEUE_SIZE 256

  Number of interrupts to hold in a queue for processing.

4.10.1 Detailed Description

4.11 DM35424 Driver Enumerations

Enumerations

- enum dm35424_pci_region_access_dir { DM35424_PCI_REGION_ACCESS_READ = 0, DM35424_PCI_REGION_ACCESS_WRITE }

  Direction of access to standard PCI region.

4.11.1 Detailed Description

DM35424_Driver_Constants

4.11.2 Enumeration Type Documentation

4.11.2.1 enum dm35424_pci_region_access_dir

Direction of access to standard PCI region.

Enumerator

DM35424_PCI_REGION_ACCESS_READ Read from the region

DM35424_PCI_REGION_ACCESS_WRITE Write to the region

Definition at line 80 of file dm35424_driver.h.

4.12 DM35424 Driver Structures

Data Structures

- struct dm35424_pci_region

  DM35424 PCI region descriptor. This structure holds information about one of a device's PCI memory regions.
- struct dm35424_dma_descriptor

  DM35424 DMA descriptor. This structure holds information about a single DMA buffer.
- struct dm35424_device_descriptor

  DM35424 Device Descriptor. The identifying info for this particular board.

Variables

- static struct file_operations dm35424_file_ops

  Placeholder protoype for file ops struct.

4.12.1 Detailed Description

DM35424_Driver_Enumerations

4.13 DM35424 Example Programs Constants

Macros

- #define BUFFER_VALID 1

  Boolean indicating buffer valid.
- #define BUFFER_NO_VALID 0

  Boolean indicating buffer not valid.
- #define BUFFER_HALT 1

  Boolean indicating buffer halt set.
- #define BUFFER_NO_HALT 0

  Boolean indicating buffer halt not set.
- #define BUFFER_LOOP 1

  Boolean indicating buffer loop set.
- #define BUFFER_NO_LOOP 0

  Boolean indicating buffer loop not set.
- #define BUFFER_INTERRUPT 1

  Boolean indicating buffer interrupt.
- #define BUFFER_NO_INTERRUPT 0

  Boolean indicating no buffer interrupt.
- #define BUFFER_PAUSE 1

  Boolean indicating buffer should pause when filled.
- #define BUFFER_NO_PAUSE 0

  Boolean indicating buffer should not pause when filled.
- #define IGNORE_USED 1

  Boolean indicating ignore used buffers.
- #define NOT_IGNORE_USED 0

  Boolean indicating not ignore used buffers.
- #define CLEAR_INTERRUPT 1

  Boolean indicating to clear an interrupt.
- #define NO_CLEAR_INTERRUPT 0

  Boolean indicating to not clear an interrupt.
- #define INTERRUPT_ENABLE 1

  Boolean indicating interrupt enable.
- #define INTERRUPT_DISABLE 0

  Boolean indicating interrupt disable.
- #define ERROR_INTR_ENABLE 1

  Boolean indicating error interrupt enable.
- #define ERROR_INTR_DISABLE 0

  Boolean indicating error interrupt disable.
- #define SYNCBUS_NONE 0

  Value indicating no Syncbus option was chosen.
- #define SYNCBUS_MASTER 1

  Value indicating Syncbus Master was chosen.
- #define SYNCBUS_SLAVE 2

  Value indicating Syncbus Slave was chosen.
- #define CHANNEL_0 0

  Constant for selecting Channel 0.
- #define CHANNEL_1 1

  Constant for selecting Channel 1.
- #define CHANNEL_2 2

Constant for selecting Channel 2.

- #define CHANNEL_3 3

  Constant for selecting Channel 3.

- #define BUFFER_0 0

  Constant for selecting Buffer 0.

- #define BUFFER_1 1

  Constant for selecting Buffer 1.

- #define ADC_0 0

  Constant for selecting ADC 0.

- #define ADC_1 1

  Constant for selecting ADC 1.

- #define DAC_0 0

  Constant for selecting DAC 0.

- #define DAC_1 1

  Constant for selecting DAC 1.

- #define DAC_2 2

  Constant for selecting DAC 2.

- #define DAC_3 3

  Constant for selecting DAC 3.

- #define REF_0 0

  Constant for selecting REF 0.

- #define REF_1 1

  Constant for selecting REF 1.

- #define DIO_0 0

  Constant for selecting DIO 0.

- #define ADIO_0 0

  Constant for selecting ADIO 0.

- #define ENABLED 1

  Constant to indicate an Enabled value.

- #define DISABLED 0

  Constant to indicate a Disabled value.

Enumerations

- enum Help_Options {
  HELP_OPTION = 1, MINOR_OPTION, RATE_OPTION, CHANNELS_OPTION,
  FILE_OPTION, START_OPTION, WAVE_OPTION, TEST_OPTION,
  NOSTOP_OPTION, SYNCBUS_OPTION, DUMP_OPTION, HOURS_OPTION,
  OUTPUT_RMS_OPTION, OUTPUT_ADC_OPTION, ADC_NUM_OPTION, DAC_NUM_OPTION,
  ADC_OPTION, DAC_OPTION, PATTERN_OPTION, SAMPLES_OPTION,
  MODE_OPTION, AD_MODE_OPTION, REF_NUM_OPTION, BINARY_OPTION,
  SENDER_OPTION, RECEIVER_OPTION, RANGE_OPTION, REFILL_FIFO_OPTION,
  LOW_THRESHOLD_OPTION, PORT_OPTION, BAUD_OPTION, EXTERNAL_OPTION,
  SIZE_OPTION, VERBOSE_OPTION, USER_ID_OPTION, COUNT_OPTION,
  NUM_OPTION, SYNC_TERM_OPTION, BIN2TXT_OPTION, STORE_OPTION,
  TERM_OPTION, REFCLK_OPTION, OFILE_OPTION, PACKED_OPTION,
  MASTER_OPTION, SLAVE_OPTION, SYNC_CONN_OPTION }

  Constants used for parsing command line parameters of example programs.

4.13.1 Detailed Description

4.13.2 Enumeration Type Documentation

4.13.2.1 enum Help_Options

Constants used for parsing command line parameters of example programs.

Note

This value won't be seen by the user (except in code), so the value can be used for any desired option.

Enumerator

HELP_OPTION Command line parameter –help.

MINOR_OPTION Command line parameter –minor.

RATE_OPTION Command line parameter –rate.

CHANNELS_OPTION Command line parameter –chan.

FILE_OPTION Command line parameter for including a file.

START_OPTION Command line parameter –start.

WAVE_OPTION Command line parameter –wave.

TEST_OPTION Command line parameter –test.

NOSTOP_OPTION Command line parameter –nostop.

SYNCBUS_OPTION Command line parameter –syncbus.

DUMP_OPTION Command line parameter –dump.

HOURS_OPTION Command line parameter –hours.

OUTPUT_RMS_OPTION Command line parameter –output_rms.

OUTPUT_ADC_OPTION Command line parameter –output_adc.

ADC_NUM_OPTION Command line parameter –num_adc.

DAC_NUM_OPTION Command line parameter –num_dac.

ADC_OPTION Command line parameter –adc.

DAC_OPTION Command line parameter –dac.

PATTERN_OPTION Command line parameter –pattern.

SAMPLES_OPTION Command line parameter –samples.

MODE_OPTION Command line parameter –mode.

AD_MODE_OPTION Command line parameter –ad_mode.

REF_NUM_OPTION Command line parameter –ref.

BINARY_OPTION Command line parameter –binary.

SENDER_OPTION Command line parameter –sender.

RECEIVER_OPTION Command line parameter –receiver.

RANGE_OPTION Command line parameter –range.

REFILL_FIFO_OPTION Command line parameter –refill.

LOW_THRESHOLD_OPTION Command line parameter –low.

PORT_OPTION Command line parameter –port.

BAUD_OPTION Command line parameter –baud.

EXTERNAL_OPTION Command line parameter –external.

SIZE_OPTION Command line parameter –size.

VERBOSE_OPTION Command line parameter –verbose.

USER_ID_OPTION Command line parameter –userid.

COUNT_OPTION Command line parameter –count.

NUM_OPTION Command line parameter –num.

SYNC_TERM_OPTION Command line parameter –syncterm.

BIN2TXT_OPTION Command line parameter –bin2txt.

STORE_OPTION Command line parameter –store.

TERM_OPTION Command line parameter –term (Termination)

REFCLK_OPTION Command line parameter –refclk (Reference clock selection)

OFILE_OPTION Command line parameter –ofile (Output file selection)

PACKED_OPTION Command line parameter –packed (Packed, 16-bit samples selection)

MASTER_OPTION Master minor option for synchronization.

SLAVE_OPTION Slave minor option for synchronization.

SYNC_CONN_OPTION Syncbus connector option.

Definition at line 43 of file dm35424_examples.h.

4.14 DM35424 Board Macros

Macros

- #define CLK_40MHZ 40000000

4.14.1 Detailed Description

4.14.2 Macro Definition Documentation

4.14.2.1 #define CLK_40MHZ 40000000

This is the standard clock of the DM35x18 boards

Definition at line 52 of file dm35424_gbc_library.h.

4.15 DM35424 Board Library Public Functions

Functions

- DM35424LIB_API int DM35424_Gbc_Board_Reset (struct DM35424_Board_Descriptor ∗handle)

  Write the reset value to the correct register to initiate a board-level reset.
- DM35424LIB_API int DM35424_Gbc_Ack_Interrupt (struct DM35424_Board_Descriptor ∗handle)

  Send an End-Of-Interrupt acknowledgement to the board. This will cause any pending interrupts to re-issue. This is a protection against missing interrupts while in the interrupt handler.
- DM35424LIB_API int DM35424_Function_Block_Open (struct DM35424_Board_Descriptor ∗handle, unsigned int number, struct DM35424_Function_Block ∗func_block)

  Open a specific function block. Nothing is opened in a file sense, but the memory location for the function block is read and certain important values are read. A function block descriptor is allocated to hold the data that will be used every time this function block is accessed.
- DM35424LIB_API int DM35424_Function_Block_Open_Module (struct DM35424_Board_Descriptor ∗handle, uint32_t fb_type, unsigned int number_of_type, struct DM35424_Function_Block ∗func_block)

  Open a specific function block module. This is the same as opening a function block, except we are looking for a function block with a specific type. This is the method you would use to open the 2nd ADC, for example.
- DM35424LIB_API int DM35424_Gbc_Get_Format (struct DM35424_Board_Descriptor ∗handle, uint8_t ∗format_id)

  Get the format ID of the board.
- DM35424LIB_API int DM35424_Gbc_Get_Revision (struct DM35424_Board_Descriptor ∗handle, uint8_t ∗rev)

  Get the PDP revision number of the board.
- DM35424LIB_API int DM35424_Gbc_Get_Pdp_Number (struct DM35424_Board_Descriptor ∗handle, uint32_t ∗pdp_num)

  Get PDP Number of the board.
- DM35424LIB_API int DM35424_Gbc_Get_Fpga_Build (struct DM35424_Board_Descriptor ∗handle, uint32_t ∗fpga_build)

  Get the FPGA Build number of the board.
- DM35424LIB_API int DM35424_Gbc_Get_Sys_Clock_Freq (struct DM35424_Board_Descriptor ∗handle, uint32_t ∗clock_freq, int ∗is_std_clk)

  Get the measured frequency of the system clock of the board.

4.15.1 Detailed Description

DM35424_Board_Macros

4.15.2 Function Documentation

4.15.2.1 DM35424LIB_API int DM35424_Function_Block_Open (struct DM35424_Board_Descriptor ∗ *handle,* unsigned int *number,* struct DM35424_Function_Block ∗ *func_block*)

Open a specific function block. Nothing is opened in a file sense, but the memory location for the function block is read and certain important values are read. A function block descriptor is allocated to hold the data that will be used every time this function block is accessed.

Parameters

handle	Pointer to the device descriptor, which contains the open file id.

number	Which function block to open. The first function block on the board is at number 0.
func_block	Pointer to the function block descriptor. When the function block info is successfully read from the device, then this descriptor will be allocated to hold the data.

Return values

0	Success.
Non-Zero	Failure.

Referenced by main().

4.15.2.2 DM35424LIB_API int DM35424_Function_Block_Open_Module (struct DM35424_Board_Descriptor * *handle,* uint32_t *fb_type,* unsigned int *number_of_type,* struct DM35424_Function_Block * *func_block*)

Open a specific function block module. This is the same as opening a function block, except we are looking for a function block with a specific type. This is the method you would use to open the 2nd ADC, for example.

Parameters

handle	Pointer to the device descriptor, which contains the open file id.
fb_type	Type of function block you want to open. ADC, DAC, DIO, etc. The constant values are in the dm35424_types.h file.
number_of_type	Ordinal number of that particular type of function block that you wish to access. The first instance of that type is 0th.
func_block	Pointer to the function block descriptor. When the function block info is successfully read from the device, then this descriptor will be allocated to hold the data.

Return values

0	Success.
Non-Zero	Failure.

4.15.2.3 DM35424LIB_API int DM35424_Gbc_Ack_Interrupt (struct DM35424_Board_Descriptor * *handle*)

Send an End-Of-Interrupt acknowledgement to the board. This will cause any pending interrupts to re-issue. This is a protection against missing interrupts while in the interrupt handler.

Parameters

handle	Pointer to the device descriptor, which contains the open file id.

Return values

0	Success.
Non-Zero	Failure.

Referenced by ISR().

4.15.2.4 DM35424LIB_API int DM35424_Gbc_Board_Reset (struct DM35424_Board_Descriptor * *handle*)

Write the reset value to the correct register to initiate a board-level reset.

Parameters

handle	Pointer to the device descriptor, which contains the open file id.

Return values

0	Success.
Non-Zero	Failure.

Referenced by main(), and run_test_9().

4.15.2.5 DM35424LIB_API int DM35424_Gbc_Get_Format (struct DM35424_Board_Descriptor * *handle,* uint8_t * *format_id*)

Get the format ID of the board.

Parameters

handle	Pointer to the device descriptor, which contains the open file id.
format_id	Pointer to the returned format ID value.

Return values

0	Success.
Non-Zero	Failure.

4.15.2.6 DM35424LIB_API int DM35424_Gbc_Get_Fpga_Build (struct DM35424_Board_Descriptor * *handle,* uint32_t * *fpga_build*)

Get the FPGA Build number of the board.

Parameters

handle	Pointer to the device descriptor, which contains the open file id.
fpga_build	Pointer to the returned FPGA Build number.

Return values

0	Success.
Non-Zero	Failure.

Referenced by main().

4.15.2.7 DM35424LIB_API int DM35424_Gbc_Get_Pdp_Number (struct DM35424_Board_Descriptor * *handle,* uint32_t * *pdp_num*)

Get PDP Number of the board.

Parameters

handle	Pointer to the device descriptor, which contains the open file id.
pdp_num	Pointer to the returned PDP Number.

Return values

0	Success.
Non-Zero	Failure.

Referenced by main().

4.15.2.8 DM35424LIB_API int DM35424_Gbc_Get_Revision (struct DM35424_Board_Descriptor * *handle,* uint8_t * *rev*)

Get the PDP revision number of the board.

Parameters

handle	Pointer to the device descriptor, which contains the open file id.
rev	Pointer to the returned revision value.

Return values

0	Success.
Non-Zero	Failure.

Referenced by main().

4.15.2.9 DM35424LIB_API int DM35424_Gbc_Get_Sys_Clock_Freq (struct DM35424_Board_Descriptor * *handle,* uint32_t * *clock_freq,* int * *is_std_clk*)

Get the measured frequency of the system clock of the board.

Parameters

handle	Pointer to the device descriptor, which contains the open file id.
clock_freq	Pointer to the returned system clock frequency (in Hz)
is_std_clk	Boolean value indicating if the clock read is a standard value. If true, then this function will always return the same value upon every call. If false, then this function will return the clock frequency actually read from the register. Note: If the value read from the GBC register is not a standard clock, then the clock frequency returned can change from read to read by slight variations.

Return values

0	Success.
Non-Zero	Failure.

4.16 DM35424 Ioctl macros

Macros

- #define DM35424_IOCTL_MAGIC 'D'

  Unique 8-bit value used to generate unique ioctl() request codes.
- #define DM35424_IOCTL_REQUEST_BASE 0x00

  First ioctl() request number.
- #define DM35424_IOCTL_REGION_READ

  ioctl() request code for reading from a PCI region
- #define DM35424_IOCTL_REGION_WRITE

  ioctl() request code for writing to a PCI region
- #define DM35424_IOCTL_REGION_MODIFY

  ioctl() request code for PCI region read/modify/write
- #define DM35424_IOCTL_DMA_FUNCTION

  ioctl() request code for DMA function
- #define DM35424_IOCTL_WAKEUP

  ioctl() request code for User ISR thread wake up
- #define DM35424_IOCTL_INTERRUPT_GET

  ioctl() request code to retrieve interrupt status information

4.16.1 Detailed Description

4.17 DM35424 Board Access Public Library Functions

Data Structures

- struct DM35424_Board_Descriptor

  DM35424 board descriptor. This structure holds information about the board as a whole. It holds the file descriptor and ISR callback function, if applicable.

Functions

- int DM35424_Dma_Initialize (struct DM35424_Board_Descriptor ∗handle, const struct DM35424_Function_Block ∗func_block, unsigned int channel, unsigned int num_buffers, uint32_t buffer_size)

  Initialize the DMA channel and prepare it for data. Interrupts are disabled, error conditions are cleared, buffers are allocated in kernel space and their status and controls are cleared.
- int DM35424_Dma_Read (struct DM35424_Board_Descriptor ∗handle, const struct DM35424_Function_Block ∗func_block, unsigned int channel, unsigned int buffer_to_read_from, uint32_t buffer_size, void ∗local_buffer_ptr)

  Read data from the DMA buffer. Data is copied from kernel buffers to local user-space buffers.
- int DM35424_Dma_Write (struct DM35424_Board_Descriptor ∗handle, const struct DM35424_Function_Block ∗func_block, unsigned int channel, unsigned int buffer_to_write_to, uint32_t buffer_size, void ∗local_buffer_ptr)

  Write data to the DMA buffer. Data is copied from local user buffers to kernel buffers.
- int DM35424_General_RemoveISR (struct DM35424_Board_Descriptor ∗handle)

  Remove the ISR from the system interrupt.
- void ∗ DM35424_General_WaitForInterrupt (void ∗ptr)

  Loop/Poll and wait for an interrupt to happen, then take action.
- int DM35424_General_InstallISR (struct DM35424_Board_Descriptor ∗handle, void(∗isr_fnct))

  Start a thread that will sit and wait for an interrupt from the board, and call the user ISR when it happens.
- int DM35424_General_SetISRPriority (struct DM35424_Board_Descriptor ∗handle, int priority)

  Set the priority of the user ISR thread.

4.17.1 Detailed Description

4.17.2 Function Documentation

4.17.2.1 int DM35424_Dma_Initialize (struct DM35424_Board_Descriptor ∗ *handle,* const struct DM35424_Function_Block ∗ *func_block,* unsigned int *channel,* unsigned int *num_buffers,* uint32_t *buffer_size*)

Initialize the DMA channel and prepare it for data. Interrupts are disabled, error conditions are cleared, buffers are allocated in kernel space and their status and controls are cleared.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.
channel	DMA Channel to get state for.
num_buffers	Number of DMA buffers to allocate and initialize.
buffer_size	The size in bytes to allocate for each buffer.

Return values

0	Success.
-1	Failure. errno may be set as follows: • EINVAL Invalid channel or buffer requested. ENOMEM Memory could not be allocated for the DMA buffers.

Referenced by main(), setup_adc(), and setup_dacs_and_start().

4.17.2.2 int DM35424_Dma_Read (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* unsigned int *channel,* unsigned int *buffer_to_read_from,* uint32_t *buffer_size,* void * *local_buffer_ptr*)

Read data from the DMA buffer. Data is copied from kernel buffers to local user-space buffers.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.
channel	DMA channel containing the buffer to be read.
buffer_to_read_from	Buffer in channel to read.
buffer_size	Number of bytes to read from the DMA buffer. In most cases, this should be equal to the allocated size of the buffer.
local_buffer_ptr	Pointer to local memory buffer already allocated that data will be copied into.

Return values

0	Success.
Non-Zero	Failure. errno may be set as follows: • EINVAL Invalid channel or buffer requested.

Referenced by ISR().

4.17.2.3 int DM35424_Dma_Write (struct DM35424_Board_Descriptor * *handle,* const struct DM35424_Function_Block * *func_block,* unsigned int *channel,* unsigned int *buffer_to_write_to,* uint32_t *buffer_size,* void * *local_buffer_ptr*)

Write data to the DMA buffer. Data is copied from local user buffers to kernel buffers.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block descriptor. The descriptor holds the information about the function block, including offsets.
channel	DMA channel containing the buffer to be written to.
buffer_to_write_to	Buffer in channel to write to.
buffer_size	Number of bytes to write to the DMA buffer. In most cases, this should be equal to the allocated size of the buffer.

local_buffer_ptr	Pointer to local memory buffer already allocated where data will come from.

Return values

0	Success.
Non-Zero	Failure. errno may be set as follows: • `EINVAL` Invalid channel or buffer requested.

Referenced by main(), and setup_dacs_and_start().

4.17.2.4 int DM35424_General_InstallISR (struct DM35424_Board_Descriptor * *handle,* void * *isr_fnct*)

Start a thread that will sit and wait for an interrupt from the board, and call the user ISR when it happens.

Parameters

handle	Pointer to the device descriptor, which contains the open file id.
isr_fnct	Pointer to the user ISR function that will be executed when an interrupt happens.

Return values

0	Success.
-1	Failure. errno may be set as follows: • `EFAULT` Could not create thread.

Referenced by main().

4.17.2.5 int DM35424_General_RemoveISR (struct DM35424_Board_Descriptor * *handle*)

Remove the ISR from the system interrupt.

Parameters

handle	Pointer to the device descriptor, which contains the open file id.

Return values

0	Success.
-1	Failure. errno may be set as follows: • `EFAULT` User ISR was already removed.

Referenced by main().

4.17.2.6 int DM35424_General_SetISRPriority (struct DM35424_Board_Descriptor * *handle,* int *priority*)

Set the priority of the user ISR thread.

Parameters

handle	Pointer to the device descriptor, which contains the open file id.
priority	Attempt to set the priority of the user ISR thread.

Note

This may require root priviledges

Return values

0	Success.
-1	Failure. errno may be set as follows: • `EFAULT` User ISR did not exist.

4.17.2.7 void* DM35424_General_WaitForInterrupt (void * *ptr*)

Loop/Poll and wait for an interrupt to happen, then take action.

Parameters

ptr	A void pointer for the board descriptor.

Return values

0	Success.
-1	Failure. errno may be set as follows: • `ENODATA` File descriptor is missing or unreadable. EIO There was no interrupt allocated to this device.

4.18 DM35424 Reference Adjustment Library Constants

Macros

- #define DM35424_REF_ADJUST_SPI_BUSY 0x00

  Register value for SPI Interface is busy.
- #define DM35424_REF_ADJUST_SPI_READY 0x01

  Register value for SPI Interface is ready.
- #define DM35424_REF_ADJUST_START_TRANS 0x01

  Register value for starting the SPI transaction.
- #define DM35424_REF_ADJUST_WRITE_ADC_VOLATILE 0x0100

  Register value for writing to the ADC Volatile memory.
- #define DM35424_REF_ADJUST_WRITE_DAC_VOLATILE 0x0200

  Register value for writing to the DAC Volatile memory.
- #define DM35424_REF_ADJUST_WRITE_ADC_NON_VOLATILE 0x1100

  Register value for writing to the ADC Non-Volatile memory.
- #define DM35424_REF_ADJUST_WRITE_DAC_NON_VOLATILE 0x1200

  Register value for writing to the DAC Non-Volatile memory.
- #define DM35424_REF_ADJUST_COPY_ADC_VOL_TO_NON 0x2100

  Register value for copying ADC data from Volatile to Non-Volatile.
- #define DM35424_REF_ADJUST_COPY_DAC_VOL_TO_NON 0x2200

  Register value for copying DAC data from Volatile to Non-Volatile.
- #define DM35424_REF_ADJUST_COPY_BOTH_VOL_TO_NON 0x2300

  Register value for copying ADC and DAC data from Volatile to Non-Volatile.
- #define DM35424_REF_ADJUST_COPY_ADC_NON_TO_VOL 0x3100

  Register value for copying ADC data from Non-Volatile to Volatile.
- #define DM35424_REF_ADJUST_COPY_DAC_NON_TO_VOL 0x3200

  Register value for copying DAC data from Non-Volatile to Volatile.
- #define DM35424_REF_ADJUST_COPY_BOTH_NON_TO_VOL 0x3300

  Register value for copying ADC and DAC data from Non-Volatile to Volatile.

Enumerations

- enum DM35424_Copy_Directions {
  DM35424_ADC_VOL_TO_NON_VOL, DM35424_DAC_VOL_TO_NON_VOL, DM35424_BOTH_VOL_TO_-
  NON_VOL, DM35424_ADC_NON_VOL_TO_VOL,
  DM35424_DAC_NON_VOL_TO_VOL, DM35424_BOTH_NON_VOL_TO_VOL }

  Direction of Reference Adjustment data copy action.

4.18.1 Detailed Description

4.18.2 Enumeration Type Documentation

4.18.2.1 enum DM35424_Copy_Directions

Direction of Reference Adjustment data copy action.

Enumerator

DM35424_ADC_VOL_TO_NON_VOL ADC Volatile to Non-Volatile

DM35424_DAC_VOL_TO_NON_VOL DAC Volatile to Non-Volatile

DM35424_BOTH_VOL_TO_NON_VOL ADC and DAC Volatile to Non-Volatile

DM35424_ADC_NON_VOL_TO_VOL ADC Non-Volatile to Volatile

DM35424_DAC_NON_VOL_TO_VOL DAC Non-Volatile to Volatile

DM35424_BOTH_NON_VOL_TO_VOL ADC and DAC Non-Volatile to Volatile

Definition at line 129 of file dm35424_ref_adjust_library.h.

4.19 DM35424 Reference Adjustment Public Library Functions

Functions

- DM35424LIB_API int DM35424_Ref_Adjust_Open (struct DM35424_Board_Descriptor *handle, unsigned int ordinal_to_open, struct DM35424_Function_Block *fb_temp)

  Open the reference adjustment function block, getting address values that will be used later by other library functions.
- DM35424LIB_API int DM35424_Ref_Adjust_Write_Adc_To_Volatile (struct DM35424_Board_Descriptor *handle, struct DM35424_Function_Block *fb, uint8_t adjustment)

  Write the ADC Reference Adjustment value to volatile memory.
- DM35424LIB_API int DM35424_Ref_Adjust_Write_Adc_To_NonVolatile (struct DM35424_Board_Descriptor *handle, struct DM35424_Function_Block *fb, uint8_t adjustment)

  Write the ADC Reference Adjustment value to non-volatile memory.
- DM35424LIB_API int DM35424_Ref_Adjust_Write_Dac_To_Volatile (struct DM35424_Board_Descriptor *handle, struct DM35424_Function_Block *fb, uint8_t adjustment)

  Write the DAC Reference Adjustment value to volatile memory.
- DM35424LIB_API int DM35424_Ref_Adjust_Write_Dac_To_NonVolatile (struct DM35424_Board_Descriptor *handle, struct DM35424_Function_Block *fb, uint8_t adjustment)

  Write the DAC Reference Adjustment value to non-volatile memory.
- DM35424LIB_API int DM35424_Ref_Adjust_Copy_Data (struct DM35424_Board_Descriptor *handle, struct DM35424_Function_Block *fb, enum DM35424_Copy_Directions direction)

  Copy the reference adjustment data from volatile to non-volatile, or vice versa.

4.19.1 Detailed Description

DM35424_Ref_Adjust_Library_Constants

4.19.2 Function Documentation

4.19.2.1 DM35424LIB_API int DM35424_Ref_Adjust_Copy_Data (struct DM35424_Board_Descriptor * *handle,* struct DM35424_Function_Block * *fb,* enum DM35424_Copy_Directions *direction*)

Copy the reference adjustment data from volatile to non-volatile, or vice versa.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
fb	Address of the function block that contains the reference adjustment we're using.
direction	Direction of the copy, including whether it is ADC, DAC or both.

Return values

0	Success.
Non-Zero	Failure.

4.19.2.2 DM35424LIB_API int DM35424_Ref_Adjust_Open (struct DM35424_Board_Descriptor * *handle,* unsigned int *ordinal_to_open,* struct DM35424_Function_Block * *fb_temp*)

Open the reference adjustment function block, getting address values that will be used later by other library functions.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
ordinal_to_open	Which function block on the board to open (0th, 1st, 2nd, etc)
fb_temp	Pointer to function block structure that will hold register offset values.

Return values

0	Success.
Non-Zero	Failure.

Referenced by main().

4.19.2.3 DM35424LIB_API int DM35424_Ref_Adjust_Write_Adc_To_NonVolatile (struct DM35424_Board_Descriptor * *handle,* struct DM35424_Function_Block * *fb,* uint8_t *adjustment*)

Write the ADC Reference Adjustment value to non-volatile memory.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
fb	Address of the function block that contains the reference adjustment we're using.
adjustment	Reference adjustment value.

Return values

0	Success.
Non-Zero	Failure.

Referenced by main().

4.19.2.4 DM35424LIB_API int DM35424_Ref_Adjust_Write_Adc_To_Volatile (struct DM35424_Board_Descriptor * *handle,* struct DM35424_Function_Block * *fb,* uint8_t *adjustment*)

Write the ADC Reference Adjustment value to volatile memory.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
fb	Address of the function block that contains the reference adjustment we're using.
adjustment	Reference adjustment value.

Return values

0	Success.
Non-Zero	Failure.

Referenced by main().

4.19.2.5 DM35424LIB_API int DM35424_Ref_Adjust_Write_Dac_To_NonVolatile (struct DM35424_Board_Descriptor * *handle,* struct DM35424_Function_Block * *fb,* uint8_t *adjustment*)

Write the DAC Reference Adjustment value to non-volatile memory.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.

fb	Address of the function block that contains the reference adjustment we're using.
adjustment	Reference adjustment value.

Return values

0	Success.
Non-Zero	Failure.

Referenced by main().

4.19.2.6 DM35424LIB_API int DM35424_Ref_Adjust_Write_Dac_To_Volatile (struct DM35424_Board_Descriptor * *handle,* struct DM35424_Function_Block * *fb,* uint8_t *adjustment*)

Write the DAC Reference Adjustment value to volatile memory.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
fb	Address of the function block that contains the reference adjustment we're using.
adjustment	Reference adjustment value.

Return values

0	Success.
Non-Zero	Failure.

Referenced by main().

4.20 DM35424 Register Offsets

Macros

- #define DM35424_OFFSET_GBC_FORMAT 0x00

  Offset to General Board Control (BAR0) Format ID register.
- #define DM35424_OFFSET_GBC_REV 0x01

  Offset to General Board Control (BAR0) Format ID register.
- #define DM35424_OFFSET_GBC_END_INTERRUPT 0x02

  Offset to General Board Control (BAR0) EOI (End of Interrupt) register.
- #define DM35424_OFFSET_GBC_BOARD_RESET 0x03

  Offset to General Board Control (BAR0) Board Reset register.
- #define DM35424_OFFSET_GBC_PDP_NUMBER 0x04

  Offset to General Board Control (BAR0) PDP Number register.
- #define DM35424_OFFSET_GBC_FPGA_BUILD 0x08

  Offset to General Board Control (BAR0) FPGA Build register.
- #define DM35424_OFFSET_GBC_SYS_CLK_FREQ 0x0c

  Offset to General Board Control (BAR0) System Clock register.
- #define DM35424_OFFSET_GBC_IRQ_STATUS 0x10

  Offset to General Board Control (BAR0) IRQ Status register. Each bit corresponds to a function block.
- #define DM35424_OFFSET_GBC_DMA_IRQ_STATUS 0x18

  Offset to General Board Control (BAR0) DMA IRQ Status register. Each bit corresponds to a function block.
- #define DM35424_OFFSET_GBC_FB_START 0x20

  Offset to the beginning of the Function Blocks section of the GBC.
- #define DM35424_GBC_FB_BLK_SIZE 0x10

  Size of the function block entries in the GBC.
- #define DM35424_OFFSET_GBC_FB_ID 0x00

  Offset to Function Block ID, from the start of the function block section.
- #define DM35424_FB_ID_TYPE_MASK 0x0000FFFF

  Bit mask for TYPE portion of FB ID.
- #define DM35424_FB_ID_SUBTYPE_MASK 0x00FF0000

  Bit mask for SUBTYPE portion of FB ID.
- #define DM35424_FB_ID_TYPE_REV_MASK 0xFF000000

  Bit mask for TYPE REV portion of FB ID.
- #define DM35424_OFFSET_GBC_FB_OFFSET 0x04

  Offset to the FB Offset in the GBC, from the start of the FB data block.
- #define DM35424_OFFSET_GBC_FB_DMA_OFFSET 0x08

  Offset to the FB DMA Offset in the GBC, from the start of the FB data block.
- #define DM35424_OFFSET_DMA_ACTION 0x00

  Offset to the DMA Action Register (BAR2)
- #define DM35424_OFFSET_DMA_SETUP 0x01

  Offset to the DMA Setup Register (BAR2)
- #define DM35424_OFFSET_DMA_STAT_OVERFLOW 0x02

  Offset to the DMA Status (Overflow) Register (BAR2)
- #define DM35424_OFFSET_DMA_STAT_UNDERFLOW 0x03

  Offset to the DMA Status (Underflow) Register (BAR2)
- #define DM35424_OFFSET_DMA_CURRENT_COUNT 0x04

  Offset to the DMA Current Count Register (BAR2)
- #define DM35424_OFFSET_DMA_CURRENT_BUFFER 0x07

  Offset to the DMA Current Buffer Register (BAR2)
- #define DM35424_OFFSET_DMA_WR_FIFO_CNT 0x08

Offset to the DMA Write FIFO Count Register (BAR2)

- #define DM35424_OFFSET_DMA_RD_FIFO_CNT 0x0A

  Offset to the DMA Read FIFO Count Register (BAR2)
- #define DM35424_OFFSET_DMA_STAT_USED 0x0C

  Offset to the DMA Status (Used) Register (BAR2)
- #define DM35424_OFFSET_DMA_STAT_INVALID 0x0D

  Offset to the DMA Status (Invalid) Register (BAR2)
- #define DM35424_OFFSET_DMA_STAT_COMPLETE 0x0E

  Offset to the DMA Status (Complete) Register (BAR2)
- #define DM35424_OFFSET_DMA_LAST_ACTION 0x0F

  Offset to the DMA Last Action Register (BAR2)
- #define DM35424_OFFSET_DMA_BUFF_START 0x10

  Offset to the start of the buffer control section (BAR2)
- #define DM35424_OFFSET_DMA_BUFFER_STAT 0x02

  Offset to the buffer status register, from the start of the buffer control section (BAR2)
- #define DM35424_OFFSET_DMA_BUFFER_CTRL 0x03

  Offset to the buffer control register, from the start of the buffer control section (BAR2)
- #define DM35424_OFFSET_DMA_BUFFER_SIZE 0x04

  Offset to the buffer size register, from the start of the buffer control section (BAR2)
- #define DM35424_OFFSET_DMA_BUFFER_ADDRESS 0x08

  Offset to the buffer address register, from the start of the buffer control section (BAR2)
- #define DM35424_OFFSET_FB_DMA_CHANNELS 0x06

  Offset to the DMA Channels count of the function block (BAR2)
- #define DM35424_OFFSET_FB_DMA_BUFFERS 0x07

  Offset to the DMA buffers count of the function block (BAR2)
- #define DM35424_OFFSET_FB_CTRL_START 0x08

  Offset to the beginning of the Function Block control section in BAR2.
- #define DM35424_OFFSET_ADC_MODE_STATUS 0x00

  Offset to the ADC Mode-Status register, from the start of the ADC control section.
- #define DM35424_OFFSET_ADC_CLK_SRC 0x01

  Offset to the ADC Clock Source register, from the start of the ADC control section.
- #define DM35424_OFFSET_ADC_START_TRIG 0x02

  Offset to the ADC Start Trigger register, from the start of the ADC control section.
- #define DM35424_OFFSET_ADC_STOP_TRIG 0x03

  Offset to the ADC Stop Trigger register, from the start of the ADC control section.
- #define DM35424_OFFSET_ADC_CLK_DIV 0x04

  Offset to the ADC Clock Divider register, from the start of the ADC control section.
- #define DM35424_OFFSET_ADC_CLK_DIV_COUNTER 0x08

  Offset to the ADC Clock Divider Counter register, from the start of the ADC control section.
- #define DM35424_OFFSET_ADC_PRE_CAPT_COUNT 0x0c

  Offset to the ADC Pre-Start Capture Count register, from the start of the ADC control section.
- #define DM35424_OFFSET_ADC_POST_CAPT_COUNT 0x10

  Offset to the ADC Post-Stop Capture Count register, from the start of the ADC control section.
- #define DM35424_OFFSET_ADC_SAMPLE_COUNT 0x14

  Offset to the ADC Sample Count register, from the start of the ADC control section.
- #define DM35424_OFFSET_ADC_INT_ENABLE 0x18

  Offset to the ADC Interrupt Enable register, from the start of the ADC control section.
- #define DM35424_OFFSET_ADC_INT_STAT 0x1e

  Offset to the ADC Interrupt Status register, from the start of the ADC control section.
- #define DM35424_OFFSET_ADC_CLK_BUS2 0x22

  Offset to the ADC Clock Bus 2, from the start of the ADC control section.

- #define DM35424_OFFSET_ADC_CLK_BUS3 0x23

  Offset to the ADC Clock Bus 3 register, from the start of the ADC control section.
- #define DM35424_OFFSET_ADC_CLK_BUS4 0x24

  Offset to the ADC Clock Bus 4 register, from the start of the ADC control section.
- #define DM35424_OFFSET_ADC_CLK_BUS5 0x25

  Offset to the ADC Clock Bus 5 register, from the start of the ADC control section.
- #define DM35424_OFFSET_ADC_CLK_BUS6 0x26

  Offset to the ADC Clock Bus 6 register, from the start of the ADC control section.
- #define DM35424_OFFSET_ADC_CLK_BUS7 0x27

  Offset to the ADC Clock Bus 7 register, from the start of the ADC control section.
- #define DM35424_OFFSET_ADC_AD_CONFIG 0x28

  Offset to the ADC AD Config register, from the start of the ADC control section.
- #define DM35424_OFFSET_ADC_CHAN_CTRL_BLK_START 0x2c

  Offset to the start of the Channel Control Section, from the start of the ADC control section.
- #define DM35424_ADC_CHAN_CTRL_BLK_SIZE 0x18

  Constant size of ADC channel section in function block.
- #define DM35424_OFFSET_ADC_CHAN_FRONT_END_CONFIG 0x00

  Offset to the Channel Front End Config register, from the start of the ADC channel control section.
- #define DM35424_OFFSET_ADC_CHAN_DATA_COUNT 0x04

  Offset to the Channel FIFO Data count register, from the start of the ADC channel control section.
- #define DM35424_OFFSET_ADC_CHAN_FILTER 0x09

  Offset to the Channel Filter register, from the start of the ADC channel control section.
- #define DM35424_OFFSET_ADC_CHAN_INTR_STAT 0x0a

  Offset to the Channel Interrupt Status register, from the start of the ADC channel control section.
- #define DM35424_OFFSET_ADC_CHAN_INTR_ENABLE 0x0b

  Offset to the Channel Interrupt Enable register, from the start of the ADC channel control section.
- #define DM35424_OFFSET_ADC_CHAN_LOW_THRESHOLD 0x0c

  Offset to the Channel Low Threshold register, from the start of the ADC channel control section.
- #define DM35424_OFFSET_ADC_CHAN_HIGH_THRESHOLD 0x10

  Offset to the Channel High Threshold register, from the start of the ADC channel control section.
- #define DM35424_OFFSET_ADC_CHAN_LAST_SAMPLE 0x14

  Offset to the Channel Last Sample register, from the start of the ADC channel control section.
- #define DM35424_OFFSET_ADC_FIFO_CTRL_BLK_START 0x334

  Offset to the start of the FIFO Control Section, from the start of the ADC control section.
- #define DM35424_ADC_FIFO_CTRL_BLK_SIZE 0x4

  Constant size of ADC FIFO section in function block.
- #define DM35424_OFFSET_FB_ADC_FIFO 0x0334

  Offset to the FIFO for non-DMA read and write operations.
- #define DM35424_OFFSET_DAC_MODE_STATUS 0x00

  Offset to the Mode/Status register, from the start of the DAC control section.
- #define DM35424_OFFSET_DAC_CLK_SRC 0x01

  Offset to the Clock Source register, from the start of the DAC control section.
- #define DM35424_OFFSET_DAC_START_TRIG 0x02

  Offset to the Start Trigger register, from the start of the DAC control section.
- #define DM35424_OFFSET_DAC_STOP_TRIG 0x03

  Offset to the Stop Trigger register, from the start of the DAC control section.
- #define DM35424_OFFSET_DAC_CLK_DIV 0x04

  Offset to the Clock Divider register, from the start of the DAC control section.
- #define DM35424_OFFSET_DAC_CLK_DIV_COUNT 0x08

  Offset to the Clock Divider Counter register, from the start of the DAC control section.
- #define DM35424_OFFSET_DAC_POST_STOP_CONV 0x10

Offset to the Post-Stop Conversion Count register, from the start of the DAC control section.

- #define DM35424_OFFSET_DAC_CONV_COUNT 0x14

  Offset to the Conversion Count register, from the start of the DAC control section.
- #define DM35424_OFFSET_DAC_INT_ENABLE 0x18

  Offset to the Interrupt Enable register, from the start of the DAC control section.
- #define DM35424_OFFSET_DAC_INT_STAT 0x1e

  Offset to the Interrupt Status register, from the start of the DAC control section.
- #define DM35424_OFFSET_DAC_CLK_BUS2 0x22

  Offset to the Clock Bus 2 register, from the start of the DAC control section.
- #define DM35424_OFFSET_DAC_CLK_BUS3 0x23

  Offset to the Clock Bus 3 register, from the start of the DAC control section.
- #define DM35424_OFFSET_DAC_CLK_BUS4 0x24

  Offset to the Clock Bus 4 register, from the start of the DAC control section.
- #define DM35424_OFFSET_DAC_CLK_BUS5 0x25

  Offset to the Clock Bus 5 register, from the start of the DAC control section.
- #define DM35424_OFFSET_DAC_CLK_BUS6 0x26

  Offset to the Clock Bus 6 register, from the start of the DAC control section.
- #define DM35424_OFFSET_DAC_CLK_BUS7 0x27

  Offset to the Clock Bus 7 register, from the start of the DAC control section.
- #define DM35424_OFFSET_DAC_DA_CONFIG 0x28

  Offset to the DA Config register, from the start of the DAC control section.
- #define DM35424_OFFSET_DAC_CHAN_CTRL_BLK_START 0x2c

  Offset to the start of the DAC channel control section, from the start of the DAC control section.
- #define DM35424_DAC_CHAN_CTRL_BLK_SIZE 0x14

  Constant size of channel control section in function block.
- #define DM35424_OFFSET_DAC_CHAN_FRONT_END_CONFIG 0x00

  Offset to the Front-End Config register, from the start of the DAC channel control section.
- #define DM35424_OFFSET_DAC_CHAN_MARKER_STATUS 0x0a

  Offset to the Channel marker Interrupt Status register, from the start of the DAC channel control section.
- #define DM35424_OFFSET_DAC_CHAN_MARKER_ENABLE 0x0b

  Offset to the Channel marker Interrupt Enable register, from the start of the DAC channel control section.
- #define DM35424_OFFSET_DAC_CHAN_LAST_CONVERSION 0x10

  Offset to the Channel Last Conversion register, from the start of the DAC channel control section.
- #define DM35424_OFFSET_DAC_FIFO_CTRL_BLK_START 0x84

  Offset to the start of the DAC FIFO control section, from the start of the DAC control section.
- #define DM35424_OFFSET_DAC_FIFO_CTRL_BLK_SIZE 0x4

  Constant size of FIFO control section in function block.
- #define DM35424_OFFSET_DIO_INPUT_VAL 0x00

  Offset to the Input Value register, from the start of the DIO control section.
- #define DM35424_OFFSET_DIO_OUTPUT_VAL 0x04

  Offset to the Output Value register, from the start of the DIO control section.
- #define DM35424_OFFSET_DIO_DIRECTION 0x08

  Offset to the Direction register, from the start of the DIO control section.
- #define DM35424_OFFSET_TEMPERATURE 0x00

  Offset to the Temperature register, from the start of the Temperature control section.
- #define DM35424_OFFSET_REF_ADJUST_GO_BUSY 0x00

  Offset to the Go/Busy register, from the start of the Reference Adjustment control section.
- #define DM35424_OFFSET_REF_OUTPUT_LATCH 0x04

  Offset to the output latch register, from the start of the Reference Adjustment control section.

4.20.1 Detailed Description

4.20.2 Macro Definition Documentation

4.20.2.1 #define DM35424_OFFSET_FB_ADC_FIFO 0x0334

Offset to the FIFO for non-DMA read and write operations.

Note

This value should be used directly. It is used in conjunction with a channel number.

Definition at line 495 of file dm35424_registers.h.

4.21 DM35424 Temperature

Functions

- DM35424LIB_API int DM35424_Temperature_Open (struct DM35424_Board_Descriptor *handle, unsigned int ordinal_to_open, struct DM35424_Function_Block *fb_temp)

  Open the temperature function block, getting address values that will be used later by other library functions.
- DM35424LIB_API int DM35424_Temperature_Read (struct DM35424_Board_Descriptor *handle, struct DM35424_Function_Block *temp_fb, float *temperature)

  Read the temperature of the board, in Celsius degrees.

4.21.1 Detailed Description

Public Library Functions

4.21.2 Function Documentation

4.21.2.1 DM35424LIB_API int DM35424_Temperature_Open (struct DM35424_Board_Descriptor * *handle,* unsigned int *ordinal_to_open,* struct DM35424_Function_Block * *fb_temp*)

Open the temperature function block, getting address values that will be used later by other library functions.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
ordinal_to_open	Which function block on the board to open (0th, 1st, 2nd, etc)
fb_temp	Pointer to function block structure that will hold register offset values.

Return values

0	Success.
Non-Zero	Failure.

Referenced by main().

4.21.2.2 DM35424LIB_API int DM35424_Temperature_Read (struct DM35424_Board_Descriptor * *handle,* struct DM35424_Function_Block * *temp_fb,* float * *temperature*)

Read the temperature of the board, in Celsius degrees.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
temp_fb	Pointer to function block we want to read.
temperature	Pointer to a preallocated float. On output holds the temperature in Celsius degree.

Return values

0	Success.
Non-Zero	Failure.

Referenced by main().

4.22 DM35424 Board Types

Macros

- #define DM35424_SUBTYPE_00 0

  Constant for FB subtype 0.
- #define DM35424_SUBTYPE_01 1

  Constant for FB subtype 1.
- #define DM35424_SUBTYPE_02 2

  Constant for FB subtype 2.
- #define DM35424_SUBTYPE_03 3

  Constant for FB subtype 3.
- #define DM35424_SUBTYPE_INVALID 0xFF

  Constant value indicating an invalid subtype.
- #define DM35424_FUNC_BLOCK_INVALID 0x0000

  Constant value indicating an invalid function block.
- #define DM35424_FUNC_BLOCK_INVALID2 0xFFFF

  Constant value indicating an invalid function block.
- #define DM35424_FUNC_BLOCK_SYNCBUS 0x0001

  Function Block Constant for SyncBus.
- #define DM35424_FUNC_BLOCK_EXT_CLOCKING 0x0002

  Function Block Constant for Global Clocking.
- #define DM35424_FUNC_BLOCK_CLK0003 0x0003

  Function Block Constant for External Clocking (0003)
- #define DM35424_FUNC_BLOCK_CAPTWIN 0x0005

  Function Block Constant for Capture Window.
- #define DM35424_FUNC_BLOCK_ADC 0x1000

  Function Block Constant for ADC.
- #define DM35424_FUNC_BLOCK_ADC1001 0x1001

  Function Block Constant for 10 MHz ADC (1001)
- #define DM35424_FUNC_BLOCK_DAC 0x2000

  Function Block Constant for DAC.
- #define DM35424_FUNC_BLOCK_DAC2001 0x2001

  Function Block Constant for High Speed DAC (2001)
- #define DM35424_FUNC_BLOCK_DIO 0x3000

  Function Block Constant for DIO.
- #define DM35424_FUNC_BLOCK_ADIO 0x3001

  Function Block Constant for ADIO.
- #define DM35424_FUNC_BLOCK_ADIO3010 0x3010

  Function Block Constant for ADIO3010.
- #define DM35424_FUNC_BLOCK_USART 0x4000

  Function Block Constant for Synchronous/Asynchronous Serial Port.
- #define DM35424_FUNC_BLOCK_REF_ADJUST 0xF000

  Function Block Constant for Reference Adjustment.
- #define DM35424_FUNC_BLOCK_TEMPERATURE_SENSOR 0xF001

  Function Block Constant for Temperature Sensor.
- #define DM35424_FUNC_BLOCK_FLASH_PROGRAMMER 0xF002

  Function Block Constant for Flash Programmer.
- #define DM35424_FUNC_BLOCK_CLK_GEN 0xF003

  Function Block Constant for Clock Generator.
- #define DM35424_FUNC_BLOCK_DIN3011 0x3011

Function Block Constant for Digital Input (3011)

- #define DM35424_FUNC_BLOCK_DOT3012 0x3012

  Function Block Constant for Digital Output (3012)
- #define DM35424_FUNC_BLOCK_INC3200 0x3200

  Function Block Constant for Incremental Encoder (3200)
- #define DM35424_FUNC_BLOCK_PWM3100 0x3100

  Function Block Constant for PWM (3100)
- #define DM35424_FUNC_BLOCK_CLK0004 0x0004

  Function Block Constant for Programmable Clock (0004)
- #define DM35424_MAX_FB 62

  Maximum possible number of function blocks on a board.
- #define MAX_DMA_BUFFERS 16

  Maximum possible number of DMA buffers for any function block.
- #define MAX_DMA_CHANNELS 32

  Maximum possible number of DMA channels for any function block.
- #define DM35424_DMA_MAX_BUFFER_SIZE 0xFFFFFC

  Maximum possible DMA buffer size.
- #define DM35424_BOARD_ACK_INTERRUPT 0x1

  Value to write to the EOI register to acknowledge interrupts.
- #define DM35424_BOARD_RESET_VALUE 0xAA

  Value to write to the Reset register in order to reset the board.
- #define DM35424_FIFO_ACCESS_FB_REVISION 0x01

  Minimum function block revision that supports direct FIFO read/write access.

4.22.1 Detailed Description

4.23 DM35424 Utility Library Functions

Enumerations

- enum DM35424_Waveforms { DM35424_SINE_WAVE, DM35424_SQUARE_WAVE, DM35424_SAWTOOTH_WAVE }

  List of possible waveforms that can be generated for DAC purposes.

Functions

- uint32_t DM35424_Get_Maskable (uint16_t data, uint16_t mask)

  Return a 32-bit maskable register value from the data and mask.
- void DM35424_Micro_Sleep (unsigned long microsecs)

  Sleep for a specified number of microseconds.
- long DM35424_Get_Time_Diff (struct timeval last, struct timeval first)

  Calculate the time difference between the two timeval structs, in microseconds.
- int DM35424_Generate_Signal_Data (enum DM35424_Waveforms waveform, int32_t *data, uint32_t data_count, int32_t max, int32_t minimum, int32_t offset, uint32_t mask)

  Generate data with a specific wave pattern. This is useful for producing recognizeable waves for DAC output.
- void check_result (int return_val, char *message)

  Check the result of an operation, usually a library call. If the result is non-zero, then it is an error and output the passed message.

4.23.1 Detailed Description

4.23.2 Enumeration Type Documentation

4.23.2.1 enum DM35424_Waveforms

List of possible waveforms that can be generated for DAC purposes.

Enumerator

DM35424_SINE_WAVE A simple sine wave.

DM35424_SQUARE_WAVE A square wave starting at max value

DM35424_SAWTOOTH_WAVE A sawtooth wave going for min to max

Definition at line 46 of file dm35424_util_library.h.

4.23.3 Function Documentation

4.23.3.1 void check_result (int *return_val*, char * *message*)

Check the result of an operation, usually a library call. If the result is non-zero, then it is an error and output the passed message.

Parameters

return_val	Value to be evaluated. Non-zero values will be considered an error.

message	Pointer to string that will be output if an error condition exists.

Return values

None	

Referenced by ISR(), main(), output_channel_status(), setup_adc(), and setup_dacs_and_start().

4.23.3.2 int DM35424_Generate_Signal_Data (enum DM35424_Waveforms *waveform,* int32_t * *data,* uint32_t *data_count,* int32_t *max,* int32_t *minimum,* int32_t *offset,* uint32_t *mask*)

Generate data with a specific wave pattern. This is useful for producing recognizeable waves for DAC output.

Parameters

waveform	Enumerated value indicating what waveform to produce.
data	Pointer to pre-allocated memory to hold resulting data values.
data_count	Number of data samples to produce
max	The maximum value in this generated data.
minimum	The minimum value in this generated data.
offset	Offset from 0 that will be the median value of this wave.
mask	Bitmask that is applied to every calculated value. This allows for handling of generated data that is less than 32 bits. To use all 32-bits, the mask would be 0xFFFFFFFF.

Note

No matter the data count, the returned data will only contain 1 period of the waveform. A higher data count will result in a "finer" set of data.

Return values

0	Success
Non-Zero	Failure

Referenced by main(), and setup_dacs_and_start().

4.23.3.3 uint32_t DM35424_Get_Maskable (uint16_t *data,* uint16_t *mask*)

Return a 32-bit maskable register value from the data and mask.

Parameters

data	Data portion (upper 16-bits) of the maskable.
mask	Mask portion (lower 16-bits) of the maskable

Return values

maskable	Maskable register value.

4.23.3.4 long DM35424_Get_Time_Diff (struct timeval *last,* struct timeval *first*)

Calculate the time difference between the two timeval structs, in microseconds.

Parameters

last	The last (most recent) timeval to compare.
first	The first (least recent) timeval to compare.

Return values

difference	The difference between the two timevals, in microseconds.

Referenced by main().

4.23.3.5 void DM35424_Micro_Sleep (unsigned long *microsecs*)

Sleep for a specified number of microseconds.

Parameters

microsecs	Length of sleep (microseconds)

Return values

None	

Referenced by main(), and run_test_9().

Chapter 5

Data Structure Documentation

5.1 DM35424_Board_Descriptor Struct Reference

DM35424 board descriptor. This structure holds information about the board as a whole. It holds the file descriptor and ISR callback function, if applicable.

`#include <dm35424_os.h>`

Data Fields

- int file_descriptor
- void(∗ isr)()
- pthread_t pid

5.1.1 Detailed Description

DM35424 board descriptor. This structure holds information about the board as a whole. It holds the file descriptor and ISR callback function, if applicable.

Definition at line 50 of file dm35424_os.h.

5.1.2 Field Documentation

5.1.2.1 int file_descriptor

File descriptor for device returned from open()

Definition at line 55 of file dm35424_os.h.

Referenced by main().

5.1.2.2 void(∗ isr)()

Function pointer to the user ISR callback function.

Definition at line 60 of file dm35424_os.h.

5.1.2.3 pthread_t pid

Process ID of the child process which will monitor DMA done interrupts.

Definition at line 65 of file dm35424_os.h.

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

- include/dm35424_os.h

5.2 dm35424_device_descriptor Struct Reference

DM35424 Device Descriptor. The identifying info for this particular board.

`#include <dm35424_driver.h>`

Data Fields

- char name [DM35424_NAME_LENGTH]
- struct dm35424_pci_region pci [PCI_ROM_RESOURCE]
- spinlock_t device_lock
- uint8_t reference_count
- unsigned int irq_number
- uint8_t remove_isr_flag
- wait_queue_head_t int_wait_queue
- wait_queue_head_t dma_wait_queue
- int interrupt_fb [DM35424_INT_QUEUE_SIZE]
- unsigned int int_queue_missed
- unsigned int int_queue_count
- unsigned int int_queue_in_marker
- unsigned int int_queue_out_marker
- struct list_head dma_descr_list

5.2.1 Detailed Description

DM35424 Device Descriptor. The identifying info for this particular board.

Definition at line 210 of file dm35424_driver.h.

5.2.2 Field Documentation

5.2.2.1 spinlock_t device_lock

Concurrency control

Definition at line 229 of file dm35424_driver.h.

5.2.2.2 struct list_head dma_descr_list

A list of all allocated DMA buffers

Definition at line 298 of file dm35424_driver.h.

5.2.2.3 wait_queue_head_t dma_wait_queue

Queue of processes waiting to be woken up when an interrupt occurs

Definition at line 261 of file dm35424_driver.h.

5.2.2.4 unsigned int int_queue_count

Number of interrupts currently in the queue

Definition at line 280 of file dm35424_driver.h.

5.2.2.5 unsigned int int_queue_in_marker

Where in the queue new entries are put

Definition at line 286 of file dm35424_driver.h.

5.2.2.6 unsigned int int_queue_missed

Number of interrupts missed because of a full queue

Definition at line 274 of file dm35424_driver.h.

5.2.2.7 unsigned int int_queue_out_marker

Where in the queue entries are pulled from

Definition at line 292 of file dm35424_driver.h.

5.2.2.8 wait_queue_head_t int_wait_queue

Queue of processes waiting to be woken up when an interrupt occurs

Definition at line 255 of file dm35424_driver.h.

5.2.2.9 int interrupt_fb[DM35424_INT_QUEUE_SIZE]

Interrupt queue containing which functional blocks caused interrupts

Definition at line 267 of file dm35424_driver.h.

5.2.2.10 unsigned int irq_number

IRQ line number

Definition at line 242 of file dm35424_driver.h.

5.2.2.11 char name[DM35424_NAME_LENGTH]

Device name used when requesting resources; a NUL terminated string of the form rtd-dm35424-x where x is the device minor number.

Definition at line 217 of file dm35424_driver.h.

5.2.2.12 struct dm35424_pci_region pci[PCI_ROM_RESOURCE]

Information about each of the standard PCI regions

Definition at line 223 of file dm35424_driver.h.

5.2.2.13 uint8_t reference_count

Number of entities which have the device file open. Used to enforce single open semantics.

Definition at line 236 of file dm35424_driver.h.

5.2.2.14 uint8_t remove_isr_flag

Used to assist poll in shutting down the thread waiting for interrupts

Definition at line 249 of file dm35424_driver.h.

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

- include/dm35424_driver.h

5.3 DM35424_DMA_Descriptor Struct Reference

Descriptor for the DMA on this board.

`#include <dm35424_board_access.h>`

Data Fields

- uint32_t control_offset
- uint8_t num_buffers
- uint32_t buffer_start_offset [MAX_DMA_BUFFERS]

5.3.1 Detailed Description

Descriptor for the DMA on this board.

Definition at line 65 of file dm35424_board_access.h.

5.3.2 Field Documentation

5.3.2.1 uint32_t buffer_start_offset[MAX_DMA_BUFFERS]

Offset to the beginning of the buffer control section.

Definition at line 80 of file dm35424_board_access.h.

5.3.2.2 uint32_t control_offset

Offset to the DMA control register section

Definition at line 70 of file dm35424_board_access.h.

5.3.2.3 uint8_t num_buffers

Number of buffers for this DMA channel.

Definition at line 75 of file dm35424_board_access.h.

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

- include/dm35424_board_access.h

5.4 dm35424_dma_descriptor Struct Reference

DM35424 DMA descriptor. This structure holds information about a single DMA buffer.

`#include <dm35424_driver.h>`

Data Fields

- uint32_t fb_num
- int channel
- int buffer
- void * virt_addr
- dma_addr_t bus_addr
- unsigned int buffer_size
- struct list_head list

5.4.1 Detailed Description

DM35424 DMA descriptor. This structure holds information about a single DMA buffer.

Definition at line 160 of file dm35424_driver.h.

5.4.2 Field Documentation

5.4.2.1 int buffer

DMA buffer number this descriptor represents.

Definition at line 175 of file dm35424_driver.h.

5.4.2.2 unsigned int buffer_size

Size of this allocated buffer

Definition at line 192 of file dm35424_driver.h.

5.4.2.3 dma_addr_t bus_addr

Bus memory address for buffer.

Definition at line 186 of file dm35424_driver.h.

5.4.2.4 int channel

DMA channel this buffer is in.

Definition at line 170 of file dm35424_driver.h.

5.4.2.5 uint32_t fb_num

Function block number this DMA is associated with.

Definition at line 165 of file dm35424_driver.h.

5.4.2.6 struct list_head list

List head so that descriptors can be kept in a linked list.

Definition at line 198 of file dm35424_driver.h.

5.4.2.7 void* virt_addr

System memory address for buffer

Definition at line 180 of file dm35424_driver.h.

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

- include/dm35424_driver.h

5.5 DM35424_Function_Block Struct Reference

DM35424 function block descriptor. This structure holds information about a function block, including type, number of DMA channels and buffers, descriptors for each DMA channel, and memory offsets to various control locations.

`#include <dm35424_board_access.h>`

Data Fields

- uint16_t type
- uint16_t sub_type
- uint16_t type_revision
- uint32_t fb_offset
- uint32_t dma_offset
- int fb_num
- int ordinal_fb_type_num
- uint8_t num_dma_buffers
- uint8_t num_dma_channels
- uint32_t control_offset
- struct DM35424_DMA_Descriptor dma_channel [MAX_DMA_CHANNELS]

5.5.1 Detailed Description

DM35424 function block descriptor. This structure holds information about a function block, including type, number of DMA channels and buffers, descriptors for each DMA channel, and memory offsets to various control locations.

Definition at line 93 of file dm35424_board_access.h.

5.5.2 Field Documentation

5.5.2.1 uint32_t control_offset

Offset to the beginning of the control registers for this function block

Definition at line 145 of file dm35424_board_access.h.

5.5.2.2 struct DM35424_DMA_Descriptor dma_channel[MAX_DMA_CHANNELS]

Array of descriptors for each DMA channel

Definition at line 154 of file dm35424_board_access.h.

5.5.2.3 uint32_t dma_offset

Offset to the beginning of the DMA registers for this function block

Definition at line 119 of file dm35424_board_access.h.

5.5.2.4 int fb_num

Function block num (as identified in GBC)

Definition at line 124 of file dm35424_board_access.h.

Referenced by ISR(), main(), and output_channel_status().

5.5.2.5 uint32_t fb_offset

Offset to the beginning of the function block registers

Definition at line 114 of file dm35424_board_access.h.

5.5.2.6 uint8_t num_dma_buffers

Number of DMA buffers in this function block

Definition at line 135 of file dm35424_board_access.h.

Referenced by main(), setup_adc(), and setup_dacs_and_start().

5.5.2.7 uint8_t num_dma_channels

Number of DMA channels in this function block

Definition at line 140 of file dm35424_board_access.h.

Referenced by main(), run_test_9(), setup_adc(), and setup_dacs_and_start().

5.5.2.8 int ordinal_fb_type_num

The ordinal number of this particular function block type (0th, 1st, etc)

Definition at line 130 of file dm35424_board_access.h.

5.5.2.9 uint16_t sub_type

Type of specific function block (ADC1, ADC2, ADC3, etc)

Definition at line 103 of file dm35424_board_access.h.

Referenced by main().

5.5.2.10 uint16_t type

Type of function block (ADC, DAC, DIO, etc)

Definition at line 98 of file dm35424_board_access.h.

Referenced by main().

5.5.2.11 uint16_t type_revision

Revision of subtype (internal use only)

Definition at line 109 of file dm35424_board_access.h.

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

- include/dm35424_board_access.h

5.6 dm35424_ioctl_argument Union Reference

ioctl() request structure encapsulating all possible requests. This is what gets passed into the kernel from user space on the ioctl() call.

`#include <dm35424_board_access_structs.h>`

Data Fields

- struct dm35424_ioctl_region_readwrite readwrite
- struct dm35424_ioctl_region_modify modify
- struct dm35424_ioctl_interrupt_info_request interrupt
- struct dm35424_ioctl_dma dma

5.6.1 Detailed Description

ioctl() request structure encapsulating all possible requests. This is what gets passed into the kernel from user space on the ioctl() call.

Definition at line 320 of file dm35424_board_access_structs.h.

5.6.2 Field Documentation

5.6.2.1 struct dm35424_ioctl_dma dma

DMA Configuration and Control

Definition at line 343 of file dm35424_board_access_structs.h.

5.6.2.2 struct dm35424_ioctl_interrupt_info_request interrupt

Interrupt request structure

Definition at line 338 of file dm35424_board_access_structs.h.

5.6.2.3 struct dm35424_ioctl_region_modify modify

PCI region read/modify/write

Definition at line 332 of file dm35424_board_access_structs.h.

5.6.2.4 struct dm35424_ioctl_region_readwrite readwrite

PCI region read and write

Definition at line 326 of file dm35424_board_access_structs.h.

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

- include/dm35424_board_access_structs.h

5.7 dm35424_ioctl_dma Struct Reference

ioctl() request structure for DMA

`#include <dm35424_board_access_structs.h>`

Data Fields

- enum DM35424_DMA_FUNCTIONS function
- int num_buffers
- uint32_t buffer_size
- uint32_t fb_num
- int channel
- int buffer
- struct dm35424_pci_access_request pci
- void * buffer_ptr

5.7.1 Detailed Description

ioctl() request structure for DMA

Definition at line 269 of file dm35424_board_access_structs.h.

5.7.2 Field Documentation

5.7.2.1 int buffer

Buffer in DMA channel that DMA is meant for.

Definition at line 299 of file dm35424_board_access_structs.h.

5.7.2.2 void* buffer_ptr

Pointer to user-space buffer for read or write.

Definition at line 309 of file dm35424_board_access_structs.h.

5.7.2.3 uint32_t buffer_size

Size (in bytes) to allocate for buffers

Definition at line 284 of file dm35424_board_access_structs.h.

5.7.2.4 int channel

Channel in function with DMA operation is for.

Definition at line 294 of file dm35424_board_access_structs.h.

5.7.2.5 uint32_t fb_num

Function Block DMA is for.

Definition at line 289 of file dm35424_board_access_structs.h.

5.7.2.6 enum DM35424_DMA_FUNCTIONS function

Requested DMA function to perform.

Definition at line 274 of file dm35424_board_access_structs.h.

5.7.2.7 int num_buffers

Number of buffers to initialize for DMA

Definition at line 279 of file dm35424_board_access_structs.h.

5.7.2.8 struct dm35424_pci_access_request pci

PCI Address of DMA registers for this operation

Definition at line 304 of file dm35424_board_access_structs.h.

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

- include/dm35424_board_access_structs.h

5.8 dm35424_ioctl_interrupt_info_request Struct Reference

ioctl() request structure for interrupt

`#include <dm35424_board_access_structs.h>`

Data Fields

- int interrupts_remaining
- int valid_interrupt
- int error_occurred
- int interrupt_fb

5.8.1 Detailed Description

ioctl() request structure for interrupt

Definition at line 238 of file dm35424_board_access_structs.h.

5.8.2 Field Documentation

5.8.2.1 int error_occurred

Boolean if error occurred during interrupt

Definition at line 254 of file dm35424_board_access_structs.h.

Referenced by ISR().

5.8.2.2 int interrupt_fb

Function block that had interrupt. The MSB indicates if this was a DMA interrupt or not. (0 = Not DMA, 1 = DMA)

Definition at line 260 of file dm35424_board_access_structs.h.

Referenced by ISR().

5.8.2.3 int interrupts_remaining

Count of interrupts remaining in the driver queue.

Definition at line 244 of file dm35424_board_access_structs.h.

5.8.2.4 int valid_interrupt

Boolean of if interrupt is valid or not.

Definition at line 249 of file dm35424_board_access_structs.h.

Referenced by ISR().

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

- include/dm35424_board_access_structs.h

5.9 dm35424_ioctl_region_modify Struct Reference

ioctl() request structure for PCI region read/modify/write

```
#include <dm35424_board_access_structs.h>
```

Data Fields

- struct dm35424_pci_access_request access
- union {
    uint8_t mask8
    uint16_t mask16
    uint32_t mask32
  } mask

5.9.1 Detailed Description

ioctl() request structure for PCI region read/modify/write

Definition at line 189 of file dm35424_board_access_structs.h.

5.9.2 Field Documentation

5.9.2.1 struct dm35424_pci_access_request access

PCI region access request

Definition at line 194 of file dm35424_board_access_structs.h.

5.9.2.2 union { ... } mask

Bit mask that controls which bits can be modified. A zero in a bit position means that the corresponding register bit should not be modified. A one in a bit position means that the corresponding register bit should be modified.

Note that it's possible to set bits outside of the mask depending upon the register value before modification. When processing the associated request code, the driver will silently prevent this from happening but will not return an indication that the mask or new value was incorrect.

5.9.2.3 uint16_t mask16

Mask for 16-bit operations

Definition at line 220 of file dm35424_board_access_structs.h.

5.9.2.4 uint32_t mask32

Mask for 32-bit operations

Definition at line 226 of file dm35424_board_access_structs.h.

5.9.2.5 uint8_t mask8

Mask for 8-bit operations

Definition at line 214 of file dm35424_board_access_structs.h.

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

- include/dm35424_board_access_structs.h

5.10 dm35424_ioctl_region_readwrite Struct Reference

ioctl() request structure for read from or write to PCI region

`#include <dm35424_board_access_structs.h>`

Data Fields

- struct dm35424_pci_access_request access

5.10.1 Detailed Description

ioctl() request structure for read from or write to PCI region

Definition at line 174 of file dm35424_board_access_structs.h.

5.10.2 Field Documentation

5.10.2.1 struct dm35424_pci_access_request access

PCI region access request

Definition at line 180 of file dm35424_board_access_structs.h.

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

- include/dm35424_board_access_structs.h

5.11 dm35424_pci_access_request Struct Reference

PCI region access request descriptor. This structure holds information about a request to read data from or write data to one of a device's PCI regions.

`#include <dm35424_board_access_structs.h>`

Data Fields

- enum dm35424_pci_region_access_size size
- enum dm35424_pci_region_num region
- uint16_t offset
- union {
  uint8_t data8
  uint16_t data16
  uint32_t data32
  } data

5.11.1 Detailed Description

PCI region access request descriptor. This structure holds information about a request to read data from or write data to one of a device's PCI regions.

Definition at line 122 of file dm35424_board_access_structs.h.

5.11.2 Field Documentation

5.11.2.1 union { ... } data

Data to write or the data read

5.11.2.2 uint16_t data16

16-bit value

Definition at line 158 of file dm35424_board_access_structs.h.

5.11.2.3 uint32_t data32

32-bit value

Definition at line 164 of file dm35424_board_access_structs.h.

5.11.2.4 uint8_t data8

8-bit value

Definition at line 152 of file dm35424_board_access_structs.h.

5.11.2.5 uint16_t offset

Offset within region to access

Definition at line 140 of file dm35424_board_access_structs.h.

5.11.2.6 enum dm35424_pci_region_num region

The PCI region to access

Definition at line 134 of file dm35424_board_access_structs.h.

5.11.2.7 enum dm35424_pci_region_access_size size

Size of access in bits

Definition at line 128 of file dm35424_board_access_structs.h.

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

- include/dm35424_board_access_structs.h

5.12 dm35424_pci_region Struct Reference

DM35424 PCI region descriptor. This structure holds information about one of a device's PCI memory regions.

`#include <dm35424_driver.h>`

Data Fields

- unsigned long io_addr
- unsigned long length
- unsigned long phys_addr
- void * virt_addr
- uint8_t allocated

5.12.1 Detailed Description

DM35424 PCI region descriptor. This structure holds information about one of a device's PCI memory regions.

Definition at line 117 of file dm35424_driver.h.

5.12.2 Field Documentation

5.12.2.1 uint8_t allocated

Flag indicating whether or not the I/O-mapped memory ranged was allocated. A value of zero means the memory range was not allocated. Any other value means the memory range was allocated.

Definition at line 151 of file dm35424_driver.h.

5.12.2.2 unsigned long io_addr

I/O port number if I/O mapped

Definition at line 123 of file dm35424_driver.h.

5.12.2.3 unsigned long length

Length of region in bytes

Definition at line 129 of file dm35424_driver.h.

5.12.2.4 unsigned long phys_addr

Region's physical address if memory mapped or I/O port number if I/O mapped

Definition at line 136 of file dm35424_driver.h.

5.12.2.5 void* virt_addr

Address at which region is mapped in kernel virtual address space if memory mapped

Definition at line 143 of file dm35424_driver.h.

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

- include/dm35424_driver.h

Chapter 6

File Documentation

6.1 examples/_non_public/dm35424_adc_test.c File Reference

Example program which demonstrates the use of the ADC and DMA.

```
#include <stdio.h>
#include <stddef.h>
#include <stdlib.h>
#include <errno.h>
#include <error.h>
#include <fcntl.h>
#include <unistd.h>
#include <signal.h>
#include <limits.h>
#include <getopt.h>
#include <string.h>
#include "dm35424_gbc_library.h"
#include "dm35424_dac_library.h"
#include "dm35424_adc_library.h"
#include "dm35424_ioctl.h"
#include "dm35424_examples.h"
#include "dm35424_dma_library.h"
#include "dm35424.h"
#include "dm35424_util_library.h"
```

Macros

- #define DEFAULT_RATE 10000
- #define BUFFER_SIZE_SAMPLES 100
- #define BUFFER_SIZE_BYTES (BUFFER_SIZE_SAMPLES ∗ sizeof(int))
- #define DAT_FILE_NAME_PREFIX "./adc"
- #define DAT_FILE_NAME_SUFFIX "_dma_data.dat"

Functions

- static void usage (void)

  Print information on stderr about how the program is to be used. After doing so, the program is exited.
- static void sigint_handler (int signal_number)

  Signal handler for SIGINT Control-C keyboard interrupt.

- void output_channel_status (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel)

  Output the status of a DMA channel. This is a helper function to determine the cause of an error when it occurs.

- void ISR (struct dm35424_ioctl_interrupt_info_request int_info)

  The interrupt subroutine that will execute when a DMA interrupt occurs. This function will read from the DMA, copying data from the kernel buffers to the user buffers so that we can access the data.

- int main (int argument_count, char **arguments)

  The main program.

Variables

- static char * program_name
- static int dma_has_error = 0
- static struct DM35424_Board_Descriptor * board
- static struct DM35424_Function_Block my_adc [DM35424_NUM_ADC_ON_BOARD]
- static unsigned long buffer_count [DM35424_NUM_ADC_ON_BOARD][DM35424_NUM_ADC_DMA_CHANNELS]
- static int ** local_buffer [DM35424_NUM_ADC_ON_BOARD][DM35424_NUM_ADC_DMA_CHANNELS]
- static volatile int exit_program = 0
- static unsigned long buffer_size_bytes = 0

6.1.1 Detailed Description

Example program which demonstrates the use of the ADC and DMA.

```
    This example program will collect data from the ADC(s)
    specified by the user, at the rate specified by the user, and will
    write the data to a file.  It will do this continuously until the
    user hits CTRL-C (or the filesystem becomes full).

    This is a very intensive operation for the PC, working CPU, memory,
    and file I/O fairly hard.  Thus, there is no way to pre-determine
    for sure what the highest sustainable rate of collecting data is.

    Maximum sustainable throughput is HIGHLY system dependent. Higher
    sample rates might be achievable through better buffer size
    selection or use of an operating system with realtime features.


-------------------------------------------------------------------------
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc.  All Rights Reserved.

This software is licensed as described in the RTD End-User Software License
Agreement.  For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
-------------------------------------------------------------------------
```

Id:

dm35424_adc_test.c 98432 2016-03-30 19:53:41Z rgroner

Definition in file dm35424_adc_test.c.

6.1.2 Macro Definition Documentation

6.1.2.1 #define BUFFER_SIZE_BYTES (BUFFER_SIZE_SAMPLES ∗ sizeof(int))

Size of DAC DMA buffer, in bytes

Definition at line 75 of file dm35424_adc_test.c.

Referenced by main().

6.1.2.2 #define BUFFER_SIZE_SAMPLES 100

Number of samples in the DAC buffer (to form the wave pattern)

Definition at line 70 of file dm35424_adc_test.c.

Referenced by main().

6.1.2.3 #define DAT_FILE_NAME_PREFIX "./adc"

Prefix for files that will be output during example

Definition at line 80 of file dm35424_adc_test.c.

Referenced by main().

6.1.2.4 #define DAT_FILE_NAME_SUFFIX "_dma_data.dat"

Suffix for files that will be output during example

Definition at line 85 of file dm35424_adc_test.c.

Referenced by main().

6.1.2.5 #define DEFAULT_RATE 10000

Default rate to use, if user does not enter one.

Definition at line 65 of file dm35424_adc_test.c.

Referenced by main().

6.1.3 Function Documentation

6.1.3.1 void ISR (struct dm35424_ioctl_interrupt_info_request *int_info*)

The interrupt subroutine that will execute when a DMA interrupt occurs. This function will read from the DMA, copying data from the kernel buffers to the user buffers so that we can access the data.

Parameters

int_info	A structure containing information about the interrupt.

Return values

None.	

Definition at line 266 of file dm35424_adc_test.c.

References buffer_count, buffer_size_bytes, check_result(), CLEAR_INTERRUPT, DM35424_Dma_Clear_-Interrupt(), DM35424_Dma_Find_Interrupt(), DM35424_Dma_Read(), DM35424_Dma_Reset_Buffer(), DM35424-

_Gbc_Ack_Interrupt(), DM35424_NUM_ADC_ON_BOARD, dma_has_error, dm35424_ioctl_interrupt_info_request::error_occurred, exit_program, DM35424_Function_Block::fb_num, dm35424_ioctl_interrupt_info_request::interrupt_fb, local_buffer, my_adc, NO_CLEAR_INTERRUPT, and dm35424_ioctl_interrupt_info_request::valid_interrupt.

Referenced by main().

6.1.3.2 int main (int *argument_count,* char ** *arguments*)

The main program.

Parameters

argument_count	Number of args passed on the command line, including the executable name
arguments	Pointer to array of character strings, which are the args themselves.

Return values

0	Success
Non-Zero	Failure.

First, setup the DACS. They will produce a sine wave that needs to be looped back to the ADC inputs. This will cause the ADC to see what looks like a max-value sine wave.

We load the even channels with the wave pattern, and the odd channels with the same pattern, but offset by half its length. Doing this gives us an opposing pattern between the even and odd channels, which helps when using DAC for ADC input.

Check to see if any channel has not yet been copied from DMA.

Definition at line 427 of file dm35424_adc_test.c.

References AD_MODE_OPTION, ADC_OPTION, BUFFER_0, buffer_count, BUFFER_INTERRUPT, BUFFER_LOOP, BUFFER_NO_HALT, BUFFER_NO_INTERRUPT, BUFFER_NO_LOOP, BUFFER_SIZE_BYTES, buffer_size_bytes, BUFFER_SIZE_SAMPLES, BUFFER_VALID, check_result(), DAT_FILE_NAME_PREFIX, DAT_FILE_NAME_SUFFIX, DEFAULT_RATE, DM35424_Adc_Ad_Config_Set_Mode(), DM35424_Adc_Channel_Setup(), DM35424_Adc_Initialize(), DM35424_ADC_INPUT_DAC_LOOPBACK, DM35424_ADC_INPUT_DIFFERENTIAL, DM35424_ADC_INPUT_SINGLE_ENDED_NEG, DM35424_ADC_INPUT_SINGLE_ENDED_POS, DM35424_ADC_MODE_CONFIG_HIGH_RES, DM35424_ADC_MODE_CONFIG_HIGH_SPEED, DM35424_ADC_MODE_CONFIG_LOW_POWER, DM35424_ADC_MODE_CONFIG_LOW_SPEED, DM35424_Adc_Open(), DM35424_ADC_RNG_BIPOLAR_19mV, DM35424_Adc_Set_Clock_Src(), DM35424_Adc_Set_Sample_Rate(), DM35424_Adc_Set_Start_Trigger(), DM35424_Adc_Set_Stop_Trigger(), DM35424_Adc_Start(), DM35424_Board_Close(), DM35424_Board_Open(), DM35424_CLK_SRC_IMMEDIATE, DM35424_CLK_SRC_NEVER, DM35424_Dac_Open(), DM35424_Dac_Set_Clock_Src(), DM35424_Dac_Set_Conversion_Rate(), DM35424_Dac_Set_Start_Trigger(), DM35424_Dac_Set_Stop_Trigger(), DM35424_Dac_Start(), DM35424_Dac_Volts_To_Conv(), DM35424_Dma_Buffer_Setup(), DM35424_Dma_Buffer_Status(), DM35424_Dma_Configure_Interrupts(), DM35424_Dma_Initialize(), DM35424_Dma_Setup(), DM35424_DMA_SETUP_DIRECTION_READ, DM35424_DMA_SETUP_DIRECTION_WRITE, DM35424_Dma_Start(), DM35424_Dma_Write(), DM35424_Gbc_Board_Reset(), DM35424_General_InstallISR(), DM35424_General_RemoveISR(), DM35424_Generate_Signal_Data(), DM35424_Micro_Sleep(), DM35424_NUM_ADC_DMA_BUFFERS, DM35424_NUM_ADC_DMA_CHANNELS, DM35424_NUM_ADC_ON_BOARD, DM35424_NUM_DAC_ON_BOARD, DM35424_SINE_WAVE, dma_has_error, ERROR_INTR_DISABLE, ERROR_INTR_ENABLE, exit_program, HELP_OPTION, IGNORE_USED, INTERRUPT_DISABLE, INTERRUPT_ENABLE, ISR(), local_buffer, MINOR_OPTION, MODE_OPTION, my_adc, NOT_IGNORE_USED, DM35424_Function_Block::num_dma_buffers, DM35424_Function_Block::num_dma_channels, output_channel_status(), program_name, RATE_OPTION, SAMPLES_OPTION, sigint_handler(), and usage().

6.1.3.3 void output_channel_status (struct **DM35424_Board_Descriptor** * *handle,* const struct **DM35424_Function_Block** * *func_block,* unsigned int *channel*)

Output the status of a DMA channel. This is a helper function to determine the cause of an error when it occurs.

Parameters

handle	Pointer to the board handle.
func_block	Pointer to the function block containing the DMA channel
channel	The DMA channel we want the status of.

Return values

None	

Definition at line 215 of file dm35424_adc_test.c.

References check_result(), DM35424_Dma_Status(), and DM35424_Function_Block::fb_num.

Referenced by main().

6.1.3.4 static void sigint_handler (int *signal_number*) [static]

Signal handler for SIGINT Control-C keyboard interrupt.

Parameters

signal_number	Signal number passed in from the kernel.

Warning

One must be extremely careful about what functions are called from a signal handler.

Definition at line 184 of file dm35424_adc_test.c.

References exit_program.

Referenced by main().

6.1.4 Variable Documentation

6.1.4.1 struct DM35424_Board_Descriptor* board [static]

Pointer to board descriptor

Definition at line 101 of file dm35424_adc_test.c.

6.1.4.2 unsigned long buffer_count[DM35424_NUM_ADC_ON_BOARD][DM35424_NUM_ADC_DMA_CHANNELS] [static]

Array of buffer counts, used to track progress of each ADC as data is copied.

Definition at line 112 of file dm35424_adc_test.c.

Referenced by ISR(), and main().

6.1.4.3 unsigned long buffer_size_bytes = 0 [static]

Size of the buffer allocated, in bytes.

Definition at line 128 of file dm35424_adc_test.c.

Referenced by ISR(), and main().

6.1.4.4 int dma_has_error = 0 [static]

Boolean flag indicating if there was a DMA error.

Definition at line 96 of file dm35424_adc_test.c.

Referenced by ISR(), and main().

6.1.4.5 volatile int exit_program = 0 [static]

Boolean indicating the program should exit.

Definition at line 123 of file dm35424_adc_test.c.

Referenced by ISR(), main(), run_test_9(), and sigint_handler().

6.1.4.6 int** local_buffer[DM35424_NUM_ADC_ON_BOARD][DM35424_NUM_ADC_DMA_CHANNELS] [static]

Pointer to local memory buffer where data is copied from the kernel buffers when a DMA buffer becomes full.

Definition at line 118 of file dm35424_adc_test.c.

Referenced by ISR(), and main().

6.1.4.7 struct DM35424_Function_Block my_adc[DM35424_NUM_ADC_ON_BOARD] [static]

Pointer to array of function blocks that will hold the ADC descriptors

Definition at line 106 of file dm35424_adc_test.c.

Referenced by ISR(), main(), and run_test_9().

6.1.4.8 char* program_name [static]

Name of the program as invoked on the command line

Definition at line 91 of file dm35424_adc_test.c.

Referenced by main(), and usage().

6.2 examples/_non_public/dm35424_board_checkout.c File Reference

Example program which checks out the functionality of all of the basic parts of the board.

```
#include <stdio.h>
#include <stddef.h>
#include <stdlib.h>
#include <errno.h>
#include <error.h>
#include <unistd.h>
#include <limits.h>
#include <getopt.h>
#include <signal.h>
#include <string.h>
#include <time.h>
#include <sys/time.h>
#include "dm35424_gbc_library.h"
#include "dm35424_dac_library.h"
#include "dm35424_adc_library.h"
#include "dm35424_board_access_struct.h"
#include "dm35424_examples.h"
#include "dm35424.h"
#include "dm35424_dma_library.h"
#include "dm35424_util_library.h"
```

Functions

- static void usage (void)

  Print information on stderr about how the program is to be used. After doing so, the program is exited.
- static void sigint_handler (int signal_number)

  Signal handler for SIGINT Control-C keyboard interrupt.
- void run_test_9 (unsigned int minor)

Variables

- static char * program_name

6.2.1 Detailed Description

Example program which checks out the functionality of all of the basic parts of the board.

```
    This example program is meant to be a basic checkout of board
    functionality.  It will make use of a loop-back connector so that
    we can use the DACs and ADCs together.

    The program is meant to have limited user input during execution.

--------------------------------------------------------------------------
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc.  All Rights Reserved.

This software is licensed as described in the RTD End-User Software License
Agreement.  For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
--------------------------------------------------------------------------
```

Id:

dm35424_board_checkout.c 98432 2016-03-30 19:53:41Z rgroner

Definition in file dm35424_board_checkout.c.

6.2.2 Function Documentation

6.2.2.1 void run_test_9 (unsigned int *minor*)

We let the thresh-inv clock run first, then the thresh clock. This is important, as it lets the inv and normal clocks be close to the same. It does NOT work if we put in the low threshold here, and then switch to the high threshold. It works this way for reasons that only Andy understands.

For better or worse, we're not going to investigate why the counts are sometimes off by 1 here, when in theory they should be identical. So, we'll allow a difference of 1 either way between the two threshold clocks and the system clock.

Setting threshold to a value it should be crossed at.

Check that the ADC is still running

Setting threshold to a value it should be crossed at.

Check that the ADC has stopped

Definition at line 5375 of file dm35424_board_checkout.c.

References CHANNEL_0, DM35424_ADC_CHAN_INTR_HIGH_THRESHOLD_MASK, DM35424_ADC_CHAN_INTR_LOW_THRESHOLD_MASK, DM35424_Adc_Channel_Get_Last_Sample(), DM35424_Adc_Channel_Interrupt_Clear_Status(), DM35424_Adc_Channel_Interrupt_Set_Config(), DM35424_Adc_Channel_Set_High_Threshold(), DM35424_Adc_Channel_Set_Low_Threshold(), DM35424_Adc_Get_Mode_Status(), DM35424_Adc_Get_Sample_Count(), DM35424_ADC_MODE_GO_SINGLE_SHOT, DM35424_Adc_Reset(), DM35424_Adc_Set_Clock_Source_Global(), DM35424_Adc_Set_Post_Stop_Samples(), DM35424_Adc_Set_Pre_Trigger_Samples(), DM35424_Adc_Set_Start_Trigger(), DM35424_Adc_Set_Stop_Trigger(), DM35424_Adc_Start(), DM35424_ADC_STAT_DONE, DM35424_ADC_STAT_SAMPLING, DM35424_ADC_STAT_WAITING_START_TRIG, DM35424_Board_Close(), DM35424_Board_Open(), DM35424_CLK_SRC_CHAN_THRESH, DM35424_CLK_SRC_CHAN_THRESH_INV, DM35424_CLK_SRC_IMMEDIATE, DM35424_CLK_SRC_NEVER, DM35424_Dac_Channel_Set_Marker_Config(), DM35424_Dac_Get_Conversion_Count(), DM35424_Dac_Get_Mode_Status(), DM35424_Dac_Reset(), DM35424_Dac_Set_Clock_Div(), DM35424_Dac_Set_Clock_Source_Global(), DM35424_Dac_Set_Clock_Src(), DM35424_Dac_Set_Last_Conversion(), DM35424_Dac_Set_Post_Stop_Conversion_Count(), DM35424_Dac_Set_Start_Trigger(), DM35424_Dac_Set_Stop_Trigger(), DM35424_Dac_Start(), DM35424_DMA_SETUP_DIRECTION_READ, DM35424_Gbc_Board_Reset(), DM35424_Micro_Sleep(), DM35424_NUM_ADC_ON_BOARD, DM35424_NUM_DAC_ON_BOARD, exit_program, INTERRUPT_DISABLE, INTERRUPT_ENABLE, my_adc, and DM35424_Function_Block::num_dma_channels.

6.2.2.2 static void sigint_handler (int *signal_number*) [static]

Signal handler for SIGINT Control-C keyboard interrupt.

Parameters

signal_number	Signal number passed in from the kernel.

Warning

One must be extremely careful about what functions are called from a signal handler.

Definition at line 226 of file dm35424_board_checkout.c.

References exit_program.

6.2.3 Variable Documentation

6.2.3.1 char* program_name [static]

Name of the program as invoked on the command line

Definition at line 166 of file dm35424_board_checkout.c.

Referenced by usage().

6.3 examples/_non_public/dm35424_calibrate.c File Reference

Example program which demonstrates the reference voltage adjustment function blocks.

```
#include <stdio.h>
#include <stddef.h>
#include <stdlib.h>
#include <errno.h>
#include <error.h>
#include <unistd.h>
#include <limits.h>
#include <signal.h>
#include <string.h>
#include <getopt.h>
#include <termios.h>
#include <time.h>
#include <sys/time.h>
#include "dm35424_gbc_library.h"
#include "dm35424_adc_library.h"
#include "dm35424_dac_library.h"
#include "dm35424_dma_library.h"
#include "dm35424_ioctl.h"
#include "dm35424_examples.h"
#include "dm35424_util_library.h"
#include "dm35424_board_access.h"
#include "dm35424_types.h"
#include "dm35424.h"
#include "dm35424_ref_adjust_library.h"
```

Functions

- static void usage (void)

  Print information on stderr about how the program is to be used. After doing so, the program is exited.
- static void sigint_handler (int signal_number)

  Signal handler for SIGINT Control-C keyboard interrupt.
- int main (int argument_count, char **arguments)

  The main program.

Variables

- static char * program_name
- volatile int exit_program = 0

6.3.1 Detailed Description

Example program which demonstrates the reference voltage adjustment function blocks.

```
        This example program demonstrates using the reference adjustment
        value to adjust the value reported by the ADC or sent by the DAC.
        The example allows for setting an adjustment value, and then
```

```
        writing that value to the appropriate memory location.

        The reference voltage source should be attached to the differential
        inputs of ADC0 or ADC1 (depending on which reference adjustment you
        are using).  The voltage reading from the ADC will be displayed on
        the screen, allowing you to view the effect reference value changes has
        on the measured voltage.

        For purposes of running the example, and for convenience, the
        onboard DACs will be set to supply a value of 2 volts.  You can then
        loopback the DAC outputs into the ADC inputs, as described below.

        Note that the supplied DAC outputs should not be considered a
        reference voltage source for the purposes of calibrating the
        board.

        WARNING: This example will allow you to change the preset
        calibration values on the board.  Do not do so unless you
        understand the risks of permanently changing that value.

        Setup: Attach a reference voltage source to the ADC Channel 0-
        and Channel 0+ pins.  If using the onboard DACs, connect DAC
        Channel 0 to ADC Channel 0+ and DAC Channel 1 to ADC Channel 0-.


-------------------------------------------------------------------------
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc.  All Rights Reserved.

This software is licensed as described in the RTD End-User Software License
Agreement.  For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
-------------------------------------------------------------------------
```

Id:

dm35424_calibrate.c 78173 2014-04-10 19:51:34Z rgroner

Definition in file dm35424_calibrate.c.

6.3.2 Function Documentation

6.3.2.1 int main (int *argument_count*, char ** *arguments*)

The main program.

Parameters

argument_count	Number of args passed on the command line, including the executable name
arguments	Pointer to array of character strings, which are the args themselves.

Return values

0	Success
Non-Zero	Failure.

Definition at line 179 of file dm35424_calibrate.c.

References ADC_0, ADC_1, board, CHANNEL_0, CHANNEL_1, check_result(), DAC_0, DAC_2, DEFAULT_RATE, DM35424_Adc_Ad_Config_Set_Mode(), DM35424_Adc_Channel_Get_Last_Sample(), DM35424_Adc_Channel_Setup(), DM35424_Adc_Initialize(), DM35424_ADC_INPUT_DIFFERENTIAL, DM35424_ADC_MODE_CONFIG_HIGH_SPEED, DM35424_Adc_Open(), DM35424_Adc_Reset(), DM35424_ADC_RNG_BIPOLAR_2_5V, DM35424_Adc_Sample_To_Volts(), DM35424_Adc_Set_Clock_Src(), DM35424_Adc_Set_Post_Stop_Samples(), DM35424_Adc_Set_Pre_Trigger_Samples(), DM35424_Adc_Set_Sample_Rate(), DM35424_Adc_Set_Start_Trigger(), DM35424_Adc_Set_Stop_Trigger(), DM35424_Adc_Start(), DM35424_Adc_Uninitialize(),

DM35424_Board_Close(), DM35424_Board_Open(), DM35424_CLK_SRC_IMMEDIATE, DM35424_CLK_SRC_NEVER, DM35424_Dac_Open(), DM35424_Dac_Reset(), DM35424_Dac_Set_Last_Conversion(), DM35424_Dac_Volts_To_Conv(), DM35424_Gbc_Board_Reset(), DM35424_Micro_Sleep(), DM35424_Ref_Adjust_Open(), DM35424_Ref_Adjust_Write_Adc_To_NonVolatile(), DM35424_Ref_Adjust_Write_Adc_To_Volatile(), DM35424_Ref_Adjust_Write_Dac_To_NonVolatile(), DM35424_Ref_Adjust_Write_Dac_To_Volatile(), exit_program, DM35424_Board_Descriptor::file_descriptor, HELP_OPTION, MAX_REF_ADJUST, MINOR_OPTION, DM35424_Function_Block::num_dma_buffers, DM35424_Function_Block::num_dma_channels, program_name, REF_0, REF_1, REF_NUM_OPTION, sigint_handler(), and usage().

6.3.2.2 static void sigint_handler (int *signal_number*) [static]

Signal handler for SIGINT Control-C keyboard interrupt.

Parameters

signal_number	Signal number passed in from the kernel.

Warning

One must be extremely careful about what functions are called from a signal handler.

Definition at line 146 of file dm35424_calibrate.c.

References exit_program.

Referenced by main().

6.3.3 Variable Documentation

6.3.3.1 volatile int exit_program = 0

Boolean indicating whether or not to exit the program.

Definition at line 95 of file dm35424_calibrate.c.

6.3.3.2 char* program_name [static]

Name of the program as invoked on the command line

Definition at line 89 of file dm35424_calibrate.c.

Referenced by main(), and usage().

6.4 examples/_non_public/dm35424_oven_test.c File Reference

```
#include <stdio.h>
```

```
#include <stddef.h>
#include <stdlib.h>
#include <errno.h>
#include <error.h>
#include <unistd.h>
#include <limits.h>
#include <getopt.h>
#include <signal.h>
#include <string.h>
#include <math.h>
#include <sys/time.h>
#include <time.h>
#include "dm35424_gbc_library.h"
#include "dm35424_dac_library.h"
#include "dm35424_adc_library.h"
#include "dm35424_ioctl.h"
#include "dm35424_examples.h"
#include "dm35424_dma_library.h"
#include "dm35424.h"
#include "dm35424_util_library.h"
```

Functions

- static void usage (void)

  Print information on stderr about how the program is to be used. After doing so, the program is exited.
- static void sigint_handler (int signal_number)

  Signal handler for SIGINT Control-C keyboard interrupt.

Variables

- static char * program_name

6.4.1 Detailed Description

```
--------------------------------------------------------------------------
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc.  All Rights Reserved.

This software is licensed as described in the RTD End-User Software License
Agreement.  For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
--------------------------------------------------------------------------
```

Id:

dm35424_oven_test.c 68203 2013-03-19 19:00:51Z rgroner

Definition in file dm35424_oven_test.c.

6.4.2 Function Documentation

6.4.2.1 static void sigint_handler (int *signal_number*) [static]

Signal handler for SIGINT Control-C keyboard interrupt.

Parameters

signal_number	Signal number passed in from the kernel.

Warning

One must be extremely careful about what functions are called from a signal handler.

Definition at line 195 of file dm35424_oven_test.c.

References exit_program.

6.4.3 Variable Documentation

6.4.3.1 char* program_name [static]

Name of the program as invoked on the command line

Definition at line 90 of file dm35424_oven_test.c.

Referenced by usage().

6.5 examples/_non_public/dm35424_ref_adjust_test.c File Reference

Example program which demonstrates the use of the digital potentiometer, to adjust the reference voltage of the DACs and ADCs on the board.

```
#include <stdio.h>
#include <stddef.h>
#include <stdlib.h>
#include <errno.h>
#include <error.h>
#include <unistd.h>
#include <limits.h>
#include <signal.h>
#include <string.h>
#include <getopt.h>
#include <termios.h>
#include <time.h>
#include <sys/time.h>
#include "dm35424_gbc_library.h"
#include "dm35424_adc_library.h"
#include "dm35424_dac_library.h"
#include "dm35424_dma_library.h"
#include "dm35424_ioctl.h"
#include "dm35424_examples.h"
#include "dm35424_util_library.h"
#include "dm35424_board_access.h"
#include "dm35424_types.h"
#include "dm35424.h"
#include "dm35424_ref_adjust_library.h"
```

Functions

- static void usage (void)

  Print information on stderr about how the program is to be used. After doing so, the program is exited.

- static void sigint_handler (int signal_number)

  Signal handler for SIGINT Control-C keyboard interrupt.
- int main (int argument_count, char **arguments)

  The main program.

Variables

- static char * program_name
- volatile int exit_program = 0

6.5.1 Detailed Description

Example program which demonstrates the use of the digital potentiometer, to adjust the reference voltage of the DACs and ADCs on the board.

```
    The program will continue to run until CTRL-C is pressed.


------------------------------------------------------------------------------
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc.  All Rights Reserved.

This software is licensed as described in the RTD End-User Software License
Agreement.  For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
------------------------------------------------------------------------------
```

Id:

dm35424_ref_adjust_test.c 78173 2014-04-10 19:51:34Z rgroner

Definition in file dm35424_ref_adjust_test.c.

6.5.2 Function Documentation

6.5.2.1 int main (int *argument_count*, char ** *arguments*)

The main program.

Parameters

argument_count	Number of args passed on the command line, including the executable name
arguments	Pointer to array of character strings, which are the args themselves.

Return values

0	Success
Non-Zero	Failure.

Definition at line 158 of file dm35424_ref_adjust_test.c.

References ADC_0, ADC_1, board, CHANNEL_0, CHANNEL_1, check_result(), DAC_0, DAC_2, DEFAULT_RATE, DM35424_Adc_Ad_Config_Set_Mode(), DM35424_Adc_Channel_Get_Last_Sample(), DM35424_Adc_Channel_Setup(), DM35424_Adc_Initialize(), DM35424_ADC_INPUT_DIFFERENTIAL, DM35424_ADC_MODE_CONFIG_HIGH_SPEED, DM35424_Adc_Open(), DM35424_Adc_Reset(), DM35424_ADC_RNG_BIPOLAR_2_5V, DM35424_Adc_Sample_To_Volts(), DM35424_Adc_Set_Clock_Src(), DM35424_Adc_Set_Post_Stop_Samples(), DM35424_Adc_Set_Pre_Trigger_Samples(), DM35424_Adc_Set_Sample_Rate(), DM35424_Adc_Set_Start_Trigger(), DM35424_Adc_Set_Stop_Trigger(), DM35424_Adc_Start(), DM35424_Adc_Uninitialize(),

DM35424_Board_Close(), DM35424_Board_Open(), DM35424_CLK_SRC_IMMEDIATE, DM35424_CLK_SRC_NEVER, DM35424_Dac_Open(), DM35424_Dac_Reset(), DM35424_Dac_Set_Last_Conversion(), DM35424_Dac_Volts_To_Conv(), DM35424_Gbc_Board_Reset(), DM35424_Micro_Sleep(), DM35424_Ref_Adjust_Open(), DM35424_Ref_Adjust_Write_Adc_To_NonVolatile(), DM35424_Ref_Adjust_Write_Adc_To_Volatile(), exit_program, HELP_OPTION, MAX_REF_ADJUST, MINOR_OPTION, DM35424_Function_Block::num_dma_buffers, DM35424_Function_Block::num_dma_channels, program_name, REF_0, REF_1, REF_NUM_OPTION, sigint_handler(), and usage().

6.5.2.2 static void sigint_handler (int *signal_number*) [static]

Signal handler for SIGINT Control-C keyboard interrupt.

Parameters

signal_number	Signal number passed in from the kernel.

Warning

One must be extremely careful about what functions are called from a signal handler.

Definition at line 125 of file dm35424_ref_adjust_test.c.

References exit_program.

Referenced by main().

6.5.3 Variable Documentation

6.5.3.1 volatile int exit_program = 0

Boolean indicating whether or not to exit the program.

Definition at line 76 of file dm35424_ref_adjust_test.c.

6.5.3.2 char* program_name [static]

Name of the program as invoked on the command line

Definition at line 70 of file dm35424_ref_adjust_test.c.

Referenced by main(), and usage().

6.6 examples/dm35424_adc.c File Reference

Example program which demonstrates the use of the ADC, setting and responding to interrupts.

```
#include <stdio.h>
#include <stddef.h>
#include <stdlib.h>
#include <errno.h>
#include <error.h>
#include <unistd.h>
#include <limits.h>
#include <signal.h>
#include <getopt.h>
#include "dm35424_gbc_library.h"
#include "dm35424_adc_library.h"
#include "dm35424_dac_library.h"
#include "dm35424_dma_library.h"
#include "dm35424_ioctl.h"
#include "dm35424_examples.h"
#include "dm35424_util_library.h"
#include "dm35424_board_access.h"
#include "dm35424_types.h"
#include "dm35424.h"
```

Macros

- #define DAC_RATE 25
- #define DEFAULT_RATE 300
- #define BUFFER_SIZE_SAMPLES 100
- #define BUFFER_SIZE_BYTES (BUFFER_SIZE_SAMPLES ∗ sizeof(int))

Functions

- static void usage (void)

  Print information on stderr about how the program is to be used. After doing so, the program is exited.
- void ISR (struct dm35424_ioctl_interrupt_info_request int_info)

  The interrupt subroutine that will execute when an interrupt occurs. It will simply increment a count, which the main program will then trigger from.
- static void sigint_handler (int signal_number)

  Signal handler for SIGINT Control-C keyboard interrupt.
- int main (int argument_count, char ∗∗arguments)

  The main program.

Variables

- static char ∗ program_name
- volatile int interrupt_count = 0
- volatile int exit_program = 0

6.6.1 Detailed Description

Example program which demonstrates the use of the ADC, setting and responding to interrupts.

```
This example program uses an ADC to collect data.  An interrupt
is generated every time data is collected by the ADC.
After acknowledging the interrupt, the program queries the value
last taken by the ADC, and the sample counter, and prints them
to the screen.
```

```
    You can put any differential signal you want on the ADC input pins.
    However, for convenience, this example sets up the DACs to provide
    a signal for the ADC to measure.  In order for that to work, you
    must loopback the DAC outputs to the ADC inputs.  Since there are
    twice as many ADC inputs as DAC outputs, each DAC output must go
    to 2 different ADC inputs. DAC Channel 0 would go to ADC Channel
    0+ and Channel 1-.  DAC Channel 1 would go to ADC Channel 0- and
    Channel 1+, etc.  In this way, all ADC even channels will have the
    same signal, and all odd channels will have the opposite.

    The program will demonstrate all input modes of the ADC: DIFFERENTIAL,
    SINGLE-ENDED POS, SINGLE-ENDED NEG, and DAC INTERNAL LOOPBACK.

    The program will also demonstrate the use of filters.  They will be
    applied to the odd-numbered channels after differential mode.

    The program will continue to run until CTRL-C is pressed.

-------------------------------------------------------------------------
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc.  All Rights Reserved.

This software is licensed as described in the RTD End-User Software License
Agreement.  For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
-------------------------------------------------------------------------
```

Id:

dm35424_adc.c 108358 2017-04-26 17:42:29Z rgroner

Definition in file dm35424_adc.c.

6.6.2 Macro Definition Documentation

6.6.2.1 #define BUFFER_SIZE_BYTES (BUFFER_SIZE_SAMPLES * sizeof(int))

Number of bytes in DAC sample buffer

Definition at line 91 of file dm35424_adc.c.

Referenced by main().

6.6.2.2 #define BUFFER_SIZE_SAMPLES 100

Number of samples to play out DAC pins

Definition at line 86 of file dm35424_adc.c.

Referenced by main().

6.6.2.3 #define DAC_RATE 25

DAC rate to use.

Definition at line 76 of file dm35424_adc.c.

Referenced by main().

6.6.2.4 #define DEFAULT_RATE 300

Rate to run at, if the user does not provide one. (Hz)

Definition at line 81 of file dm35424_adc.c.

Referenced by main(), and usage().

6.6.3 Function Documentation

6.6.3.1 int main (int *argument_count,* char ** *arguments*)

The main program.

Parameters

argument_count	Number of args passed on the command line, including the executable name
arguments	Pointer to array of character strings, which are the args themselves.

Return values

0	Success
Non-Zero	Failure.

First, setup the DACS. They will produce a sine wave that needs to be looped back to the ADC inputs. This will cause the ADC to see what looks like a max-value sine wave.

We load the even channels with the wave pattern, and the odd channels with the same pattern, but offset by half its length. Doing this gives us an opposing pattern between the even and odd channels, which helps when using DAC for ADC input.

Definition at line 213 of file dm35424_adc.c.

References ADC_0, board, BUFFER_0, BUFFER_SIZE_BYTES, BUFFER_SIZE_SAMPLES, check_result(), DAC_RATE, DEFAULT_RATE, DM35424_Adc_Ad_Config_Set_Mode(), DM35424_ADC_CHAN_FILTER_ORDER0, DM35424_ADC_CHAN_FILTER_ORDER7, DM35424_Adc_Channel_Get_Last_Sample(), DM35424_Adc_Channel_Set_Filter(), DM35424_Adc_Channel_Setup(), DM35424_Adc_Get_Sample_Count(), DM35424_Adc_Initialize(), DM35424_ADC_INPUT_DAC_LOOPBACK, DM35424_ADC_INPUT_DIFFERENTIAL, DM35424_ADC_INPUT_SINGLE_ENDED_NEG, DM35424_ADC_INPUT_SINGLE_ENDED_POS, DM35424_ADC_INT_ALL_MASK, DM35424_ADC_INT_SAMPLE_TAKEN_MASK, DM35424_Adc_Interrupt_Clear_Status(), DM35424_Adc_Interrupt_Get_Config(), DM35424_Adc_Interrupt_Get_Status(), DM35424_Adc_Interrupt_Set_Config(), DM35424_ADC_MODE_CONFIG_HIGH_SPEED, DM35424_Adc_Open(), DM35424_Adc_Reset(), DM35424_ADC_RNG_BIPOLAR_2_5V, DM35424_ADC_RNG_UNIPOLAR_5V, DM35424_Adc_Sample_To_Volts(), DM35424_Adc_Set_Clock_Src(), DM35424_Adc_Set_Post_Stop_Samples(), DM35424_Adc_Set_Pre_Trigger_Samples(), DM35424_Adc_Set_Sample_Rate(), DM35424_Adc_Set_Start_Trigger(), DM35424_Adc_Set_Stop_Trigger(), DM35424_Adc_Start(), DM35424_Adc_Uninitialize(), DM35424_Board_Close(), DM35424_Board_Open(), DM35424_CLK_SRC_IMMEDIATE, DM35424_CLK_SRC_NEVER, DM35424_Dac_Open(), DM35424_Dac_Reset(), DM35424_Dac_Set_Clock_Src(), DM35424_Dac_Set_Conversion_Rate(), DM35424_Dac_Set_Start_Trigger(), DM35424_Dac_Set_Stop_Trigger(), DM35424_Dac_Start(), DM35424_Dac_Volts_To_Conv(), DM35424_DMA_BUFFER_CTRL_LOOP, DM35424_DMA_BUFFER_CTRL_VALID, DM35424_Dma_Buffer_Setup(), DM35424_Dma_Clear(), DM35424_Dma_Initialize(), DM35424_Dma_Setup(), DM35424_DMA_SETUP_DIRECTION_WRITE, DM35424_Dma_Start(), DM35424_Dma_Write(), DM35424_Gbc_Board_Reset(), DM35424_General_InstallISR(), DM35424_General_RemoveISR(), DM35424_Generate_Signal_Data(), DM35424_Micro_Sleep(), DM35424_NUM_DAC_ON_BOARD, DM35424_SINE_WAVE, exit_program, HELP_OPTION, IGNORE_USED, interrupt_count, INTERRUPT_DISABLE, INTERRUPT_ENABLE, ISR(), MINOR_OPTION, DM35424_Function_Block::num_dma_buffers, DM35424_Function_Block::num_dma_channels, program_name, RATE_OPTION, sigint_handler(), and usage().

6.6.3.2 static void sigint_handler (int *signal_number*) [static]

Signal handler for SIGINT Control-C keyboard interrupt.

Parameters

signal_number	Signal number passed in from the kernel.

Warning

One must be extremely careful about what functions are called from a signal handler.

Definition at line 179 of file dm35424_adc.c.

References exit_program.

Referenced by main().

6.6.4 Variable Documentation

6.6.4.1 volatile int exit_program = 0

Boolean indicating whether or not to exit the program.

Definition at line 106 of file dm35424_adc.c.

6.6.4.2 volatile int interrupt_count = 0

Count of interrupts that have happened.

Definition at line 101 of file dm35424_adc.c.

Referenced by ISR(), and main().

6.6.4.3 char* program_name [static]

Name of the program as invoked on the command line

Definition at line 96 of file dm35424_adc.c.

Referenced by main(), and usage().

6.7 examples/dm35424_adc_continuous_dma.c File Reference

Example program which demonstrates the use of the ADC and DMA.

```
#include <stdio.h>
#include <stddef.h>
#include <stdlib.h>
#include <errno.h>
#include <error.h>
#include <fcntl.h>
#include <unistd.h>
#include <signal.h>
#include <limits.h>
#include <getopt.h>
#include "dm35424_gbc_library.h"
#include "dm35424_dac_library.h"
#include "dm35424_adc_library.h"
#include "dm35424_ioctl.h"
#include "dm35424_examples.h"
#include "dm35424_dma_library.h"
#include "dm35424.h"
#include "dm35424_util_library.h"
```

Macros

- #define DEFAULT_RATE 10000
- #define BUFFER_SIZE_SAMPLES 1000
- #define BUFFER_SIZE_BYTES (BUFFER_SIZE_SAMPLES ∗ sizeof(int))
- #define ASCII_FILE_NAME "./adc_dma.txt"
- #define BIN_FILE_NAME "./adc_dma.bin"

Functions

- static void usage (void)

  Print information on stderr about how the program is to be used. After doing so, the program is exited.
- static void sigint_handler (int signal_number)

  Signal handler for SIGINT Control-C keyboard interrupt.
- void output_channel_status (struct DM35424_Board_Descriptor ∗handle, const struct DM35424_Function_Block ∗func_block, unsigned int channel)

  Output the status of a DMA channel. This is a helper function to determine the cause of an error when it occurs.
- void ISR (struct dm35424_ioctl_interrupt_info_request int_info)

  The interrupt subroutine that will execute when a DMA interrupt occurs. This function will read from the DMA, copying data from the kernel buffers to the user buffers so that we can access the data.
- void setup_dacs_and_start (struct DM35424_Function_Block ∗my_dac)

  Setup the DACs to produce a sine wave as a signal to sample, then start them.
- void setup_adc (uint32_t rate)

  Setup the ADCs to sample.
- void setup_ctrlc_handler ()

  Handler to detect when user hits Ctrl-C.
- void convert_bin_to_txt (unsigned int samples_in_buff)

  Convert a binary data file to ASCII values. The format will be the same as the data file produced without the –binary argument. The example program will exit after finishing.
- int main (int argument_count, char ∗∗arguments)

  The main program.

Variables

- static char * program_name
- static int dma_has_error = 0
- static struct DM35424_Board_Descriptor * board
- static struct DM35424_Function_Block my_adc
- static unsigned long buffer_count [DM35424_NUM_ADC_DMA_CHANNELS]
- static int ** local_buffer [DM35424_NUM_ADC_DMA_CHANNELS]
- static volatile int exit_program = 0
- static unsigned long buffer_size_bytes = 0
- static unsigned int next_buffer [DM35424_NUM_ADC_DMA_CHANNELS]

6.7.1 Detailed Description

Example program which demonstrates the use of the ADC and DMA.

```
This example program will collect data from the ADC(s)
specified by the user, at the rate specified by the user, and will
write the data to a file.  It will do this continuously until the
user hits CTRL-C (or the filesystem becomes full).

You can put any signal you want on the ADC input pins.
However, for convenience, this example sets up the DACs to provide
a signal for the ADC to measure.  In order for that to work, you
must loopback the DAC outputs to the ADC differential inputs.  Connect
DAC0 Channel 0 to ADC_0 Channel 0+ and ADC_0 Channel 1-, and connect
DAC0 Channel 1 to ADC_0 Channel 0- and ADC_0 Channel 1+, etc.

Maximum sustainable throughput is HIGHLY system dependent. Higher
sample rates might be achievable through better buffer size
selection or use of an operating system with realtime features.


--------------------------------------------------------------------
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc.  All Rights Reserved.

This software is licensed as described in the RTD End-User Software License
Agreement.  For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
--------------------------------------------------------------------
```

Id:

dm35424_adc_continuous_dma.c 108389 2017-04-27 13:27:05Z rgroner

Definition in file dm35424_adc_continuous_dma.c.

6.7.2 Macro Definition Documentation

6.7.2.1 #define ASCII_FILE_NAME "./adc_dma.txt"

Name of file when saving as ASCII

Definition at line 82 of file dm35424_adc_continuous_dma.c.

Referenced by convert_bin_to_txt(), main(), and usage().

6.7.2.2 #define BIN_FILE_NAME "./adc_dma.bin"

Name of file when saving as binary

Definition at line 87 of file dm35424_adc_continuous_dma.c.

Referenced by convert_bin_to_txt(), main(), and usage().

6.7.2.3 #define BUFFER_SIZE_BYTES (BUFFER_SIZE_SAMPLES * sizeof(int))

Size of DAC DMA buffer, in bytes

Definition at line 77 of file dm35424_adc_continuous_dma.c.

Referenced by setup_dacs_and_start().

6.7.2.4 #define BUFFER_SIZE_SAMPLES 1000

Number of samples in the DAC buffer (to form the wave pattern)

Definition at line 72 of file dm35424_adc_continuous_dma.c.

Referenced by setup_dacs_and_start().

6.7.2.5 #define DEFAULT_RATE 10000

Default rate to use, if user does not enter one. (Hz)

Definition at line 67 of file dm35424_adc_continuous_dma.c.

Referenced by main(), and usage().

6.7.3 Function Documentation

6.7.3.1 void convert_bin_to_txt (unsigned int *samples_in_buff*)

Convert a binary data file to ASCII values. The format will be the same as the data file produced without the –binary argument. The example program will exit after finishing.

Return values

None.	

Definition at line 716 of file dm35424_adc_continuous_dma.c.

References ASCII_FILE_NAME, BIN_FILE_NAME, and DM35424_NUM_ADC_DMA_CHANNELS.

Referenced by main().

6.7.3.2 void ISR (struct dm35424_ioctl_interrupt_info_request *int_info*)

The interrupt subroutine that will execute when a DMA interrupt occurs. This function will read from the DMA, copying data from the kernel buffers to the user buffers so that we can access the data.

Parameters

int_info	A structure containing information about the interrupt.

Return values

None.	

Definition at line 279 of file dm35424_adc_continuous_dma.c.

References buffer_count, buffer_size_bytes, CHANNEL_0, check_result(), CLEAR_INTERRUPT, DM35424_Dma_Clear_Interrupt(), DM35424_Dma_Find_Interrupt(), DM35424_Dma_Read(), DM35424_Dma_Reset_Buffer(), DM35424_Gbc_Ack_Interrupt(), DM35424_NUM_ADC_DMA_BUFFERS, DM35424_NUM_ADC_DMA_CHANNELS, dma_has_error, exit_program, DM35424_Function_Block::fb_num, dm35424_ioctl_interrupt_info_request::interrupt_fb, local_buffer, my_adc, next_buffer, NO_CLEAR_INTERRUPT, and dm35424_ioctl_interrupt_info_request::valid_interrupt.

6.7.3.3 int main (int *argument_count,* char ∗∗ *arguments*)

The main program.

Parameters

argument_count	Number of args passed on the command line, including the executable name
arguments	Pointer to array of character strings, which are the args themselves.

Return values

0	Success
Non-Zero	Failure.

First, setup the DACS. They will produce a sine wave that needs to be looped back to the ADC inputs. This will cause the ADC to see what looks like a max-value sine wave.

Check to see if any channel has not yet been copied from DMA.

Definition at line 818 of file dm35424_adc_continuous_dma.c.

References ASCII_FILE_NAME, BIN2TXT_OPTION, BIN_FILE_NAME, BINARY_OPTION, buffer_count, buffer_size_bytes, check_result(), convert_bin_to_txt(), DEFAULT_RATE, DM35424_Adc_Start(), DM35424_Board_Close(), DM35424_Board_Open(), DM35424_Dma_Configure_Interrupts(), DM35424_Dma_Start(), DM35424_Gbc_Board_Reset(), DM35424_General_InstallISR(), DM35424_General_RemoveISR(), DM35424_Micro_Sleep(), DM35424_NUM_ADC_DMA_CHANNELS, dma_has_error, ERROR_INTR_DISABLE, exit_program, HELP_OPTION, INTERRUPT_DISABLE, ISR(), local_buffer, MINOR_OPTION, my_adc, DM35424_Function_Block::num_dma_buffers, DM35424_Function_Block::num_dma_channels, output_channel_status(), program_name, RATE_OPTION, SAMPLES_OPTION, setup_adc(), setup_ctrlc_handler(), setup_dacs_and_start(), and usage().

6.7.3.4 void output_channel_status (struct **DM35424_Board_Descriptor** ∗ *handle,* const struct **DM35424_Function_Block** ∗ *func_block,* unsigned int *channel*)

Output the status of a DMA channel. This is a helper function to determine the cause of an error when it occurs.

Parameters

handle	Pointer to the board handle.
func_block	Pointer to the function block containing the DMA channel
channel	The DMA channel we want the status of.

Return values

None	

Definition at line 227 of file dm35424_adc_continuous_dma.c.

References check_result(), DM35424_Dma_Status(), and DM35424_Function_Block::fb_num.

6.7.3.5 void setup_adc (uint32_t *rate*)

Setup the ADCs to sample.

Parameters

rate	The sampling rate to set the ADC to.

Return values

None.	

Definition at line 544 of file dm35424_adc_continuous_dma.c.

References ADC_0, buffer_size_bytes, CHANNEL_0, check_result(), DM35424_Adc_Ad_Config_Set_Mode(), DM35424_Adc_Channel_Setup(), DM35424_Adc_Initialize(), DM35424_ADC_INPUT_DIFFERENTIAL, DM35424_ADC_MODE_CONFIG_HIGH_SPEED, DM35424_Adc_Open(), DM35424_ADC_RNG_BIPOLAR_2_5V, DM35424_Adc_Set_Clock_Src(), DM35424_Adc_Set_Sample_Rate(), DM35424_Adc_Set_Start_Trigger(), DM35424_Adc_Set_Stop_Trigger(), DM35424_CLK_SRC_IMMEDIATE, DM35424_CLK_SRC_NEVER, DM35424_DMA_BUFFER_CTRL_INTR, DM35424_DMA_BUFFER_CTRL_LOOP, DM35424_DMA_BUFFER_CTRL_VALID, DM35424_Dma_Buffer_Setup(), DM35424_Dma_Buffer_Status(), DM35424_Dma_Configure_Interrupts(), DM35424_Dma_Initialize(), DM35424_Dma_Setup(), DM35424_DMA_SETUP_DIRECTION_READ, DM35424_NUM_ADC_DMA_BUFFERS, ERROR_INTR_ENABLE, INTERRUPT_ENABLE, my_adc, next_buffer, NOT_IGNORE_USED, DM35424_Function_Block::num_dma_buffers, and DM35424_Function_Block::num_dma_channels.

Referenced by main().

6.7.3.6 void setup_ctrlc_handler ( )

Handler to detect when user hits Ctrl-C.

Return values

None.	

Definition at line 689 of file dm35424_adc_continuous_dma.c.

References sigint_handler().

Referenced by main().

6.7.3.7 void setup_dacs_and_start (struct DM35424_Function_Block * *my_dac*)

Setup the DACs to produce a sine wave as a signal to sample, then start them.

Parameters

my_dac	Pointer to the DAC function block structure.

Return values

None.	

We load the even channels with the wave pattern, and the odd channels with the same pattern, but offset by half its length. Doing this gives us an opposing pattern between the even and odd channels, which helps when using DAC for ADC input.

Definition at line 375 of file dm35424_adc_continuous_dma.c.

References BUFFER_0, BUFFER_SIZE_BYTES, BUFFER_SIZE_SAMPLES, check_result(), DM35424_CLK_SRC_IMMEDIATE, DM35424_CLK_SRC_NEVER, DM35424_Dac_Open(), DM35424_Dac_Set_Clock_Src(), DM35424_Dac_Set_Conversion_Rate(), DM35424_Dac_Set_Start_Trigger(), DM35424_Dac_Set_Stop_Trigger(), DM35424_Dac_Start(), DM35424_Dac_Volts_To_Conv(), DM35424_DMA_BUFFER_CTRL_LOOP, DM35424_DMA_BUFFER_CTRL_VALID, DM35424_Dma_Buffer_Setup(), DM35424_Dma_Initialize(), DM35424_Dma_Setup(), DM35424_DMA_SETUP_DIRECTION_WRITE, DM35424_Dma_Start(), DM35424_Dma_Write(), DM35424_Generate_Signal_Data(), DM35424_NUM_DAC_ON_BOARD, DM35424_SINE_WAVE, IGNORE_USED, DM35424_Function_Block::num_dma_buffers, and DM35424_Function_Block::num_dma_channels.

Referenced by main().

6.7.3.8 static void sigint_handler (int *signal_number*) [static]

Signal handler for SIGINT Control-C keyboard interrupt.

Parameters

signal_number	Signal number passed in from the kernel.

Warning

One must be extremely careful about what functions are called from a signal handler.

Definition at line 196 of file dm35424_adc_continuous_dma.c.

References exit_program.

Referenced by setup_ctrlc_handler().

6.7.4 Variable Documentation

6.7.4.1 struct DM35424_Board_Descriptor* board [static]

Pointer to board descriptor

Definition at line 103 of file dm35424_adc_continuous_dma.c.

6.7.4.2 unsigned long buffer_count[DM35424_NUM_ADC_DMA_CHANNELS] [static]

Array of buffer counts, used to track progress of each ADC as data is copied.

Definition at line 114 of file dm35424_adc_continuous_dma.c.

Referenced by ISR(), and main().

6.7.4.3 unsigned long buffer_size_bytes = 0 [static]

Size of the buffer allocated, in bytes.

Definition at line 130 of file dm35424_adc_continuous_dma.c.

Referenced by ISR(), main(), and setup_adc().

6.7.4.4 int dma_has_error = 0 [static]

Boolean flag indicating if there was a DMA error.

Definition at line 98 of file dm35424_adc_continuous_dma.c.

Referenced by ISR(), and main().

6.7.4.5 volatile int exit_program = 0 [static]

Boolean indicating the program should exit.

Definition at line 125 of file dm35424_adc_continuous_dma.c.

6.7.4.6 int** local_buffer[DM35424_NUM_ADC_DMA_CHANNELS] [static]

Pointer to local memory buffer where data is copied from the kernel buffers when a DMA buffer becomes full.

Definition at line 120 of file dm35424_adc_continuous_dma.c.

Referenced by ISR(), and main().

6.7.4.7 struct DM35424_Function_Block my_adc [static]

Pointer to array of function blocks that will hold the ADC descriptors

Definition at line 108 of file dm35424_adc_continuous_dma.c.

Referenced by ISR(), main(), and setup_adc().

6.7.4.8 unsigned int next_buffer[DM35424_NUM_ADC_DMA_CHANNELS] [static]

Which buffer is next to be copied from DMA

Definition at line 135 of file dm35424_adc_continuous_dma.c.

Referenced by ISR(), and setup_adc().

6.7.4.9 char* program_name [static]

Name of the program as invoked on the command line

Definition at line 93 of file dm35424_adc_continuous_dma.c.

Referenced by main(), and usage().

6.8 examples/dm35424_dac.c File Reference

Example program which demonstrates the use of the DAC.

```
#include <stdio.h>
#include <stddef.h>
#include <stdlib.h>
#include <errno.h>
#include <error.h>
#include <unistd.h>
#include <limits.h>
#include <getopt.h>
#include <termios.h>
#include <time.h>
#include <sys/time.h>
#include "dm35424_gbc_library.h"
#include "dm35424_dac_library.h"
#include "dm35424_ioctl.h"
#include "dm35424_examples.h"
#include "dm35424.h"
#include "dm35424_util_library.h"
```

Macros

- #define DEFAULT_DAC_NUM 0

Functions

- static void usage (void)

  Print information on stderr about how the program is to be used. After doing so, the program is exited.
- int main (int argument_count, char **arguments)

  The main program.

Variables

- static char * program_name

6.8.1 Detailed Description

Example program which demonstrates the use of the DAC.

```
    This example program sends data to the DAC for instant conversion.
    To see the output data, connect an oscilloscope to the DACx Channel 0
    through Channel 3 pins, or appropriate DACx pin if you change the DAC
    number.

    The user can control what value goes out the DAC by using keys to
    increase or decrease the desired voltage, up to 5 V and down to -5 V.
    Follow the on-screen instructions for adjusting the voltage.

    Press 'q' to quit the program.

-------------------------------------------------------------------------
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc.  All Rights Reserved.

This software is licensed as described in the RTD End-User Software License
Agreement.  For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
-------------------------------------------------------------------------
```

Id:

dm35424_dac.c 107191 2017-03-17 17:32:43Z rgroner

Definition in file dm35424_dac.c.

6.8.2 Macro Definition Documentation

6.8.2.1 #define DEFAULT_DAC_NUM 0

DAC to use, if user does not choose one.

Definition at line 59 of file dm35424_dac.c.

Referenced by main(), and usage().

6.8.3 Function Documentation

6.8.3.1 int main (int *argument_count,* char ** *arguments*)

The main program.

Parameters

argument_count	Number of args passed on the command line, including the executable name
arguments	Pointer to array of character strings, which are the args themselves.

Return values

0	Success
Non-Zero	Failure.

The DAC cannot achieve +5.0 volts, but for purposes of the example, we allow them to select 5 as a value. However, we'll have to request the actual max value from the library function.

Definition at line 120 of file dm35424_dac.c.

References board, check_result(), DAC_NUM_OPTION, DEFAULT_DAC_NUM, DM35424_Board_Close(), DM35424_Board_Open(), DM35424_Dac_Conv_To_Volts(), DM35424_Dac_Get_Last_Conversion(), DM35424_Dac_Open(), DM35424_Dac_Reset(), DM35424_Dac_Set_Last_Conversion(), DM35424_Dac_Volts_To_Conv(), DM35424_Gbc_Board_Reset(), DM35424_Get_Time_Diff(), HELP_OPTION, MINOR_OPTION, DM35424_Function_Block::num_dma_buffers, DM35424_Function_Block::num_dma_channels, program_name, and usage().

6.8.4 Variable Documentation

6.8.4.1 char* program_name [static]

Name of the program as invoked on the command line

Definition at line 64 of file dm35424_dac.c.

Referenced by main(), and usage().

6.9 examples/dm35424_dac_dma.c File Reference

Example program which demonstrates the use of the DAC and DMA.

```
#include <stdio.h>
#include <stddef.h>
#include <stdlib.h>
#include <errno.h>
#include <error.h>
#include <unistd.h>
#include <limits.h>
#include <getopt.h>
#include <signal.h>
#include <string.h>
#include "dm35424_gbc_library.h"
#include "dm35424_dac_library.h"
#include "dm35424_ioctl.h"
#include "dm35424_examples.h"
#include "dm35424_dma_library.h"
#include "dm35424_util_library.h"
#include "dm35424.h"
```

Macros

- #define NUM_BUFFERS_TO_USE 1
- #define DEFAULT_DAC_TO_USE 0
- #define DEFAULT_RATE 100

- #define BUFFER_SIZE_SAMPLES 100
- #define BUFFER_SIZE_BYTES (BUFFER_SIZE_SAMPLES * sizeof(int))

Functions

- static void usage (void)

  Print information on stderr about how the program is to be used. After doing so, the program is exited.
- static void sigint_handler (int signal_number)

  Signal handler for SIGINT Control-C keyboard interrupt.
- int main (int argument_count, char **arguments)

  The main program.

Variables

- static char * program_name
- volatile int exit_program = 0

6.9.1 Detailed Description

Example program which demonstrates the use of the DAC and DMA.

```
    This example program generates wave form data and "plays" it out
    all channels for the specified DAC.  To see the output data,
    connect an oscilloscope to the DAC0 Channel 0 - 3 pins, or appropriate
    DAC/Channel pins if you change the DAC number.

    The odd-numbered channels will be delayed in their starting so that
    they are a half-period out of phase.

    After the program is running, you can alter the rate of DAC output
    by entering a new frequency and hitting Enter.  Note that the frequency
    of the waveform seen on an oscilloscope will be different than the frequency
    of the DAC, depending on the number of samples used in creating the wave.

    Use the --help command line option to see all possible input values.

    Hit Ctrl-C to exit the example.


-------------------------------------------------------------------------
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc.  All Rights Reserved.

This software is licensed as described in the RTD End-User Software License
Agreement.  For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
-------------------------------------------------------------------------
```

Id:

dm35424_dac_dma.c 108397 2017-04-27 14:15:37Z rgroner

Definition in file dm35424_dac_dma.c.

6.9.2 Macro Definition Documentation

6.9.2.1 #define BUFFER_SIZE_BYTES (BUFFER_SIZE_SAMPLES * sizeof(int))

Buffer size to allocate in bytes

Definition at line 86 of file dm35424_dac_dma.c.

Referenced by main().

6.9.2.2 #define BUFFER_SIZE_SAMPLES 100

Number of samples to create. Increase this number for a "finer" waveform.

Definition at line 81 of file dm35424_dac_dma.c.

Referenced by main().

6.9.2.3 #define DEFAULT_DAC_TO_USE 0

DAC to use if user does not enter one on the command line

Definition at line 70 of file dm35424_dac_dma.c.

Referenced by main(), and usage().

6.9.2.4 #define DEFAULT_RATE 100

Rate to use if user does not enter one on the command line (Hz)

Definition at line 75 of file dm35424_dac_dma.c.

Referenced by main(), and usage().

6.9.2.5 #define NUM_BUFFERS_TO_USE 1

We will only use one buffer in this example, and loop it

Definition at line 65 of file dm35424_dac_dma.c.

Referenced by main().

6.9.3 Function Documentation

6.9.3.1 int main (int *argument_count,* char ** *arguments*)

The main program.

Parameters

argument_count	Number of args passed on the command line, including the executable name
arguments	Pointer to array of character strings, which are the args themselves.

Return values

0	Success
Non-Zero	Failure.

We load the even channels with the wave pattern, and the odd channels with the same pattern, but offset by half its length. Doing this gives us an opposing pattern between the even and odd channels, which helps when using DAC for ADC input, or just viewing on a scope.

Definition at line 186 of file dm35424_dac_dma.c.

References board, BUFFER_0, BUFFER_SIZE_BYTES, BUFFER_SIZE_SAMPLES, check_result(), DAC_OPTION, DEFAULT_DAC_TO_USE, DEFAULT_RATE, DM35424_Board_Close(), DM35424_Board_Open(), DM35424_CLK_SRC_IMMEDIATE, DM35424_CLK_SRC_NEVER, DM35424_Dac_Open(), DM35424_Dac_Set_Clock_Src(), DM35424_Dac_Set_Conversion_Rate(), DM35424_Dac_Set_Start_Trigger(), DM35424_Dac_Set_-

Stop_Trigger(), DM35424_Dac_Start(), DM35424_Dac_Volts_To_Conv(), DM35424_DMA_BUFFER_CTRL_LOOP, DM35424_DMA_BUFFER_CTRL_VALID, DM35424_Dma_Buffer_Setup(), DM35424_Dma_Buffer_Status(), DM35424_Dma_Initialize(), DM35424_Dma_Setup(), DM35424_DMA_SETUP_DIRECTION_WRITE, DM35424_Dma_Start(), DM35424_Dma_Status(), DM35424_Dma_Write(), DM35424_Gbc_Board_Reset(), DM35424_Generate_Signal_Data(), DM35424_NUM_DAC_ON_BOARD, DM35424_SAWTOOTH_WAVE, DM35424_SINE_WAVE, DM35424_SQUARE_WAVE, exit_program, HELP_OPTION, IGNORE_USED, MINOR_OPTION, NUM_BUFFERS_TO_USE, DM35424_Function_Block::num_dma_buffers, DM35424_Function_Block::num_dma_channels, PATTERN_OPTION, program_name, RATE_OPTION, sigint_handler(), usage(), and WAVE_OPTION.

6.9.3.2 static void sigint_handler (int *signal_number*) [static]

Signal handler for SIGINT Control-C keyboard interrupt.

Parameters

signal_number	Signal number passed in from the kernel.

Warning

One must be extremely careful about what functions are called from a signal handler.

Definition at line 154 of file dm35424_dac_dma.c.

References exit_program.

Referenced by main().

6.9.4 Variable Documentation

6.9.4.1 volatile int exit_program = 0

Boolean indicating the program should be exited.

Definition at line 97 of file dm35424_dac_dma.c.

6.9.4.2 char* program_name [static]

Name of the program as invoked on the command line

Definition at line 92 of file dm35424_dac_dma.c.

Referenced by main(), and usage().

6.10 examples/dm35424_dio.c File Reference

Example program which demonstrates the use of the DIO.

```
#include <stdio.h>
#include <stddef.h>
#include <stdlib.h>
#include <errno.h>
#include <error.h>
#include <unistd.h>
#include <limits.h>
#include <getopt.h>
#include "dm35424_gbc_library.h"
#include "dm35424_dio_library.h"
#include "dm35424_ioctl.h"
#include "dm35424_util_library.h"
#include "dm35424_examples.h"
```

Macros

- #define DM35424_DIO_DIRECTION 0x0000007F

Functions

- static void usage (void)

  Print information on stderr about how the program is to be used. After doing so, the program is exited.
- int main (int argument_count, char **arguments)

  The main program.

Variables

- static char * program_name
- struct DM35424_Board_Descriptor * board
- struct DM35424_Function_Block my_dio

6.10.1 Detailed Description

Example program which demonstrates the use of the DIO.

```
    This example program sets the lower 7-bits of DIO to
    output, and the upper 7-bits to input.  We'll then
    write every possible 7-bit value to the output
    and verify the same value on the input pins.

    This example requires a loopback from DIO Port 0:0 to
    Port 0:7, Port 0:1 to Port 0:8, etc.

    Port 0, Pins 0-6: Output
    Port 0, Pins 7-13: Input

--------------------------------------------------------------------------
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc.  All Rights Reserved.

This software is licensed as described in the RTD End-User Software License
Agreement.  For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
--------------------------------------------------------------------------
```

Id:

dm35424_dio.c 107191 2017-03-17 17:32:43Z rgroner

Definition in file dm35424_dio.c.

6.10.2 Macro Definition Documentation

6.10.2.1 #define DM35424_DIO_DIRECTION 0x0000007F

Setting for DIO pin direction

Definition at line 55 of file dm35424_dio.c.

Referenced by main().

6.10.3 Function Documentation

6.10.3.1 int main (int *argument_count,* char ** *arguments*)

The main program.

Parameters

argument_count	Number of args passed on the command line, including the executable name
arguments	Pointer to array of character strings, which are the args themselves.

Return values

0	Success
Non-Zero	Failure.

Definition at line 124 of file dm35424_dio.c.

References check_result(), DM35424_Board_Close(), DM35424_Board_Open(), DM35424_DIO_DIRECTION, DM35424_Dio_Get_Input_Value(), DM35424_Dio_Open(), DM35424_Dio_Set_Direction(), DM35424_Dio_Set_Output_Value(), DM35424_Gbc_Board_Reset(), HELP_OPTION, MINOR_OPTION, my_dio, program_name, and usage().

6.10.4 Variable Documentation

6.10.4.1 struct DM35424_Board_Descriptor* board

Descriptor for board

Definition at line 65 of file dm35424_dio.c.

Referenced by main().

6.10.4.2 struct DM35424_Function_Block my_dio

Descriptor for DIO function block

Definition at line 70 of file dm35424_dio.c.

Referenced by main().

6.10.4.3 char* program_name [static]

Name of the program as invoked on the command line

Definition at line 60 of file dm35424_dio.c.

Referenced by main(), and usage().

6.11 examples/dm35424_list_fb.c File Reference

Example program which demonstrates use of the library to open a function block for use.

```
#include <stdio.h>
#include <stddef.h>
#include <stdlib.h>
#include <errno.h>
#include <error.h>
#include <unistd.h>
#include <limits.h>
#include <getopt.h>
#include "dm35424_gbc_library.h"
#include "dm35424_ioctl.h"
#include "dm35424_examples.h"
#include "dm35424_util_library.h"
```

Functions

- static void usage (void)

  Print information on stderr about how the program is to be used. After doing so, the program is exited.

- int main (int argument_count, char **arguments)

  The main program.

Variables

- static char * program_name

6.11.1 Detailed Description

Example program which demonstrates use of the library to open a function block for use.

```
    This example program uses the board library to query all
    function blocks on the board.  When a function block is opened
    that has a valid function type, then the number of DMA channels
    and buffers is printed to the screen.  In this way, the example
    program shows an inventory of the function blocks on a given
    board.


--------------------------------------------------------------------------
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc.  All Rights Reserved.

This software is licensed as described in the RTD End-User Software License
Agreement.  For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
--------------------------------------------------------------------------
```

Id:

dm35424_list_fb.c 108353 2017-04-26 15:53:48Z rgroner

Definition in file dm35424_list_fb.c.

6.11.2 Function Documentation

6.11.2.1 int main (int *argument_count,* char ** *arguments*)

The main program.

Parameters

argument_count	Number of args passed on the command line, including the executable name
arguments	Pointer to array of character strings, which are the args themselves.

Return values

0	Success
Non-Zero	Failure.

Definition at line 105 of file dm35424_list_fb.c.

References board, check_result(), DM35424_Board_Close(), DM35424_Board_Open(), DM35424_FUNC_BLOCK_ADC, DM35424_FUNC_BLOCK_DAC, DM35424_FUNC_BLOCK_DIO, DM35424_FUNC_BLOCK_INVALID, DM35424_FUNC_BLOCK_REF_ADJUST, DM35424_FUNC_BLOCK_TEMPERATURE_SENSOR, DM35424_Function_Block_Open(), DM35424_Gbc_Board_Reset(), DM35424_Gbc_Get_Fpga_Build(), DM35424_Gbc_Get_Pdp_Number(), DM35424_Gbc_Get_Revision(), DM35424_MAX_FB, DM35424_Function_Block::fb_num, HELP_OPTION, MINOR_OPTION, DM35424_Function_Block::num_dma_buffers, DM35424_Function_Block::num_dma_channels, program_name, DM35424_Function_Block::sub_type, DM35424_Function_Block::type, and usage().

6.11.3 Variable Documentation

6.11.3.1 char* program_name [static]

Name of the program as invoked on the command line

Definition at line 51 of file dm35424_list_fb.c.

Referenced by main(), and usage().

6.12 examples/dm35424_ref_adjust.c File Reference

Example program which demonstrates the reference voltage adjustment function blocks.

```
#include <stdio.h>
#include <stddef.h>
#include <stdlib.h>
#include <errno.h>
#include <error.h>
#include <unistd.h>
#include <limits.h>
#include <signal.h>
#include <string.h>
#include <getopt.h>
#include <termios.h>
#include <time.h>
#include <sys/time.h>
#include "dm35424_gbc_library.h"
#include "dm35424_adc_library.h"
#include "dm35424_dac_library.h"
#include "dm35424_dma_library.h"
#include "dm35424_ioctl.h"
#include "dm35424_examples.h"
#include "dm35424_util_library.h"
#include "dm35424_board_access.h"
#include "dm35424_types.h"
#include "dm35424.h"
#include "dm35424_ref_adjust_library.h"
```

Macros

- #define DEFAULT_RATE 80000
- #define MAX_REF_ADJUST 0xFF

Functions

- static void usage (void)

  Print information on stderr about how the program is to be used. After doing so, the program is exited.
- int keyboard_hit ()

  Detect a keypress by selecting on STDIO with 0 wait time.
- int getch ()

  Get the key that was pressed.
- static void sigint_handler (int signal_number)

  Signal handler for SIGINT Control-C keyboard interrupt.
- int main (int argument_count, char **arguments)

  The main program.

Variables

- static char * program_name
- volatile int exit_program = 0

6.12.1 Detailed Description

Example program which demonstrates the reference voltage adjustment function blocks.

```
        This example program demonstrates using the reference adjustment
        value to adjust the value reported by the ADC or sent by the DAC.
        The example allows for setting an adjustment value, and then
        writing that value to the appropriate memory location.

        The reference voltage source should be attached to the differential
        inputs of ADC0 or ADC1 (depending on which reference adjustment you
        are using).  The voltage reading from the ADC will be displayed on
        the screen, allowing you to view the effect reference value changes has
        on the measured voltage.

        For purposes of running the example, and for convenience, the
        onboard DACs will be set to supply a value of 2 volts.  You can then
        loopback the DAC outputs into the ADC inputs, as described below.

        Note that the supplied DAC outputs should not be considered a
        reference voltage source for the purposes of calibrating the
        board.

        WARNING: This example will allow you to change the preset
        calibration values on the board.  Do not do so unless you
        understand the risks of permanently changing that value.

        Setup: Attach a reference voltage source to the ADC Channel 0-
        and Channel 0+ pins.  If using the onboard DACs, connect DAC
        Channel 0 to ADC Channel 0+ and DAC Channel 1 to ADC Channel 0-.

-------------------------------------------------------------------------
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc.  All Rights Reserved.
```

```
This software is licensed as described in the RTD End-User Software License
Agreement.  For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
-------------------------------------------------------------------------
```

Id:

dm35424_ref_adjust.c 130693 2021-06-24 19:07:24Z zcovone

Definition in file dm35424_ref_adjust.c.

6.12.2 Macro Definition Documentation

6.12.2.1 #define DEFAULT_RATE 80000

Sampling rate to use (Hz)

Definition at line 85 of file dm35424_ref_adjust.c.

Referenced by main().

6.12.2.2 #define MAX_REF_ADJUST 0xFF

Maximum REF adjust to allow

Definition at line 90 of file dm35424_ref_adjust.c.

Referenced by main().

6.12.3 Function Documentation

6.12.3.1 int getch ( )

Get the key that was pressed.

Return values

None.	

Definition at line 158 of file dm35424_ref_adjust.c.

Referenced by main().

6.12.3.2 int keyboard_hit ( )

Detect a keypress by selecting on STDIO with 0 wait time.

Return values

None.	

Definition at line 136 of file dm35424_ref_adjust.c.

Referenced by main().

6.12.3.3 int main (int *argument_count*, char ** *arguments*)

The main program.

Parameters

argument_count	Number of args passed on the command line, including the executable name
arguments	Pointer to array of character strings, which are the args themselves.

Return values

0	Success
Non-Zero	Failure.

Definition at line 219 of file dm35424_ref_adjust.c.

References ADC_0, board, CHANNEL_0, CHANNEL_1, check_result(), DAC_0, DEFAULT_RATE, DM35424_Adc_Ad_Config_Set_Mode(), DM35424_Adc_Channel_Get_Last_Sample(), DM35424_Adc_Channel_Setup(), DM35424_Adc_Initialize(), DM35424_ADC_INPUT_DIFFERENTIAL, DM35424_ADC_MODE_CONFIG_HIGH_SPEED, DM35424_Adc_Open(), DM35424_Adc_Reset(), DM35424_ADC_RNG_BIPOLAR_2_5V, DM35424_Adc_Sample_To_Volts(), DM35424_Adc_Set_Clock_Src(), DM35424_Adc_Set_Post_Stop_Samples(), DM35424_Adc_Set_Pre_Trigger_Samples(), DM35424_Adc_Set_Sample_Rate(), DM35424_Adc_Set_Start_Trigger(), DM35424_Adc_Set_Stop_Trigger(), DM35424_Adc_Start(), DM35424_Adc_Uninitialize(), DM35424_Board_Close(), DM35424_Board_Open(), DM35424_CLK_SRC_IMMEDIATE, DM35424_CLK_SRC_NEVER, DM35424_Dac_Open(), DM35424_Dac_Reset(), DM35424_Dac_Set_Last_Conversion(), DM35424_Dac_Volts_To_Conv(), DM35424_Gbc_Board_Reset(), DM35424_Micro_Sleep(), DM35424_Ref_Adjust_Open(), DM35424_Ref_Adjust_Write_Adc_To_NonVolatile(), DM35424_Ref_Adjust_Write_Adc_To_Volatile(), DM35424_Ref_Adjust_Write_Dac_To_NonVolatile(), DM35424_Ref_Adjust_Write_Dac_To_Volatile(), exit_program, getch(), HELP_OPTION, keyboard_hit(), MAX_REF_ADJUST, MINOR_OPTION, DM35424_Function_Block::num_dma_buffers, DM35424_Function_Block::num_dma_channels, program_name, REF_0, sigint_handler(), and usage().

6.12.3.4 static void sigint_handler (int *signal_number*) [static]

Signal handler for SIGINT Control-C keyboard interrupt.

Parameters

signal_number	Signal number passed in from the kernel.

Warning

One must be extremely careful about what functions are called from a signal handler.

Definition at line 186 of file dm35424_ref_adjust.c.

References exit_program.

Referenced by main().

6.12.4 Variable Documentation

6.12.4.1 volatile int exit_program = 0

Boolean indicating whether or not to exit the program.

Definition at line 100 of file dm35424_ref_adjust.c.

6.12.4.2 char* program_name [static]

Name of the program as invoked on the command line

Definition at line 95 of file dm35424_ref_adjust.c.

Referenced by main(), and usage().

6.13 examples/dm35424_temperature.c File Reference

Example program for read the temperature sensor.

```
#include <stdio.h>
#include <stddef.h>
#include <stdlib.h>
#include <errno.h>
#include <error.h>
#include <unistd.h>
#include <limits.h>
#include <signal.h>
#include <getopt.h>
#include <string.h>
#include "dm35424_gbc_library.h"
#include "dm35424_temperature_library.h"
#include "dm35424_ioctl.h"
#include "dm35424_examples.h"
#include "dm35424_util_library.h"
```

Macros

- #define TEMP_FB_TO_OPEN 0

Functions

- static void usage (void)

  Print information on stderr about how the program is to be used. After doing so, the program is exited.
- static void sigint_handler (int signal_number)

  Signal handler for SIGINT Control-C keyboard interrupt.
- int main (int argument_count, char **arguments)

  The main program.

Variables

- static char * program_name
- volatile int exit_program = 0

6.13.1 Detailed Description

Example program for read the temperature sensor.

```
    The program will read the temperature value from the temperature
    function block and print it to the screen.  It will also print the
    maximum and minimum temperature received from the board for the time
    that the example program has been running.

    The program will continue to run until CTRL-C is pressed.

------------------------------------------------------------------------
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc.  All Rights Reserved.

This software is licensed as described in the RTD End-User Software License
Agreement.  For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
```

```
Technologies, Inc.
------------------------------------------------------------------------
```

Id:

dm35424_temperature.c 107191 2017-03-17 17:32:43Z rgroner

Definition in file dm35424_temperature.c.

6.13.2 Macro Definition Documentation

6.13.2.1 #define TEMP_FB_TO_OPEN 0

Which temperature function block to open on this board (there is only one).

Definition at line 57 of file dm35424_temperature.c.

Referenced by main().

6.13.3 Function Documentation

6.13.3.1 int main (int *argument_count,* char ** *arguments*)

The main program.

Parameters

argument_count	Number of args passed on the command line, including the executable name
arguments	Pointer to array of character strings, which are the args themselves.

Return values

0	Success
Non-Zero	Failure.

Definition at line 140 of file dm35424_temperature.c.

References board, check_result(), DM35424_Board_Close(), DM35424_Board_Open(), DM35424_Gbc_Board_Reset(), DM35424_Micro_Sleep(), DM35424_Temperature_Open(), DM35424_Temperature_Read(), exit_program, HELP_OPTION, MINOR_OPTION, program_name, sigint_handler(), TEMP_FB_TO_OPEN, and usage().

6.13.3.2 static void sigint_handler (int *signal_number*) [static]

Signal handler for SIGINT Control-C keyboard interrupt.

Parameters

signal_number	Signal number passed in from the kernel.

Warning

One must be extremely careful about what functions are called from a signal handler.

Definition at line 107 of file dm35424_temperature.c.

References exit_program.

Referenced by main().

6.13.4 Variable Documentation

6.13.4.1 volatile int exit_program = 0

Boolean indicating whether or not to exit the program.

Definition at line 67 of file dm35424_temperature.c.

6.13.4.2 char* program_name [static]

Name of the program as invoked on the command line

Definition at line 62 of file dm35424_temperature.c.

Referenced by main(), and usage().

6.14 include/dm35424.h File Reference

Defines for the DM35424 (Device-specific values)

Macros

- #define DM35424_PCI_VENDOR_ID 0x1435

  DM35424 PCI vendor ID.
- #define DM35424_PCI_DEVICE_ID 0x5424

  DM35424 PCI device ID.
- #define DM35424_NUM_ADC_ON_BOARD 2

  Number of ADC on the DM35424.
- #define DM35424_NUM_DAC_ON_BOARD 4

  Number of DAC on the DM35424.
- #define DM35424_NUM_ADC_DMA_CHANNELS 8

  Number of channels per ADC.
- #define DM35424_NUM_ADC_DMA_BUFFERS 7

  Number of buffers per ADC DMA channel.
- #define DM35424_NUM_DAC_DMA_CHANNELS 4

  Number of channels per DAC.
- #define DM35424_NUM_DAC_DMA_BUFFERS 7

  Number of buffers per DAC DMA channel.
- #define DM35424_DAC_FE_CONFIG_GAIN_MASK 0

  Bit Mask for the gain bits of the FE Config.
- #define DM35424_FIFO_SAMPLE_SIZE 511

  Sample size of the FIFO.
- #define DM35424_DAC_MAX_VALUE 32767

  Max value of the DAC.
- #define DM35424_DAC_MIN_VALUE -32768

  Min value of the DAC.
- #define DM35424_DAC_MAX_RATE 106000

  Max rate of the DAC.

6.14.1 Detailed Description

Defines for the DM35424 (Device-specific values)

```
--------------------------------------------------------------------------
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc.  All Rights Reserved.

This software is licensed as described in the RTD End-User Software License
Agreement.  For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
--------------------------------------------------------------------------
```

Id:

dm35424.h 90112 2015-07-15 20:05:32Z rgroner

Definition in file dm35424.h.

6.15 include/dm35424_adc_library.h File Reference

Definitions for the DM35424 ADC Library.

```
#include "dm35424_gbc_library.h"
```

Macros

- #define DM35424_ADC_MODE_RESET 0x00

  Register value for ADC Mode Reset.
- #define DM35424_ADC_MODE_PAUSE 0x01

  Register value for ADC Mode Pause.
- #define DM35424_ADC_MODE_GO_SINGLE_SHOT 0x02

  Register value for ADC Mode Go (Single Shot)
- #define DM35424_ADC_MODE_GO_REARM 0x03

  Register value for ADC Mode Go (Rearm after Stop)
- #define DM35424_ADC_MODE_UNINITIALIZED 0x04

  Register value for ADC Mode Uninitialized.
- #define DM35424_ADC_STAT_STOPPED 0x00

  Register value for ADC Status - Stopped.
- #define DM35424_ADC_STAT_FILLING_PRE_TRIG_BUFF 0x01

  Register value for ADC Status - Filling Pre-Start Buffer.
- #define DM35424_ADC_STAT_WAITING_START_TRIG 0x02

  Register value for ADC Status - Waiting for Start Trigger.
- #define DM35424_ADC_STAT_SAMPLING 0x03

  Register value for ADC Status - Sampling Data.
- #define DM35424_ADC_STAT_FILLING_POST_TRIG_BUFF 0x04

  Register value for ADC Status - Filling Post-Stop Buffer.
- #define DM35424_ADC_STAT_WAIT_REARM 0x05

  Register value for ADC Status - Wait for Rearm.
- #define DM35424_ADC_STAT_DONE 0x07

  Register value for ADC Status - Done.
- #define DM35424_ADC_STAT_UNINITIALIZED 0x08

  Register value for ADC Status - Uninitialized.

- #define DM35424_ADC_STAT_INITIALIZING 0x09

  Register value for ADC Status - Initializing.
- #define DM35424_ADC_INT_SAMPLE_TAKEN_MASK 0x01

  Register value for Interrupt Mask - Sample Taken.
- #define DM35424_ADC_INT_CHAN_THRESHOLD_MASK 0x02

  Register value for Interrupt Mask - Channel Threshold Exceeded.
- #define DM35424_ADC_INT_PRE_BUFF_FULL_MASK 0x04

  Register value for Interrupt Mask - Pre-Start Buffer Filled.
- #define DM35424_ADC_INT_START_TRIG_MASK 0x08

  Register value for Interrupt Mask - Start Trigger Occurred.
- #define DM35424_ADC_INT_STOP_TRIG_MASK 0x10

  Register value for Interrupt Mask - Stop Trigger Occurred.
- #define DM35424_ADC_INT_POST_BUFF_FULL_MASK 0x20

  Register value for Interrupt Mask - Post-Stop Buffer Filled.
- #define DM35424_ADC_INT_SAMP_COMPL_MASK 0x40

  Register value for Interrupt Mask - Sampling Complete.
- #define DM35424_ADC_INT_PACER_TICK_MASK 0x80

  Register value for Interrupt Mask - Pacer Clock Tick Occurred.
- #define DM35424_ADC_INT_ALL_MASK 0xFF

  Register value for Interrupt Mask - All Bits.
- #define DM35424_ADC_CHAN_INTR_LOW_THRESHOLD_MASK 0x01

  Register value for Channel Low Threshold Interrupt.
- #define DM35424_ADC_CHAN_INTR_HIGH_THRESHOLD_MASK 0x02

  Register value for Channel High Threshold Interrupt.
- #define DM35424_ADC_CHAN_FILTER_ORDER0 0x0

  Register value for Channel Filter Order 0.
- #define DM35424_ADC_CHAN_FILTER_ORDER1 0x1

  Register value for Channel Filter Order 1.
- #define DM35424_ADC_CHAN_FILTER_ORDER2 0x2

  Register value for Channel Filter Order 2.
- #define DM35424_ADC_CHAN_FILTER_ORDER3 0x3

  Register value for Channel Filter Order 3.
- #define DM35424_ADC_CHAN_FILTER_ORDER4 0x4

  Register value for Channel Filter Order 4.
- #define DM35424_ADC_CHAN_FILTER_ORDER5 0x5

  Register value for Channel Filter Order 5.
- #define DM35424_ADC_CHAN_FILTER_ORDER6 0x6

  Register value for Channel Filter Order 6.
- #define DM35424_ADC_CHAN_FILTER_ORDER7 0x7

  Register value for Channel Filter Order 7.
- #define DM35424_ADC_FE_CONFIG_POWER_ACTIVE 0x80

  Register value for setting channel power to active.
- #define DM35424_ADC_FE_CONFIG_PGA_ACTIVE 0x40

  Register value for setting channel PGA to active.
- #define DM35424_ADC_FE_CONFIG_IN_SWITCH_ENABLED 0x20

  Register value for setting channel input switch to enabled.
- #define DM35424_ADC_FE_CONFIG_DAC_LOOPBACK 0x00

  Register value for measuring between DACx Chy and VREF (2.5V)
- #define DM35424_ADC_FE_CONFIG_SNGL_END_POS 0x08

  Register value for measuring InP - VREF (singled ended)
- #define DM35424_ADC_FE_CONFIG_SNGL_END_NEG 0x10

Register value for measuring VREF - InN (singled ended)

- #define DM35424_ADC_FE_CONFIG_DIFFERENTIAL 0x18

  Register value for setting channel to In(Positive) - In(Negative) connection. This is the most common.
- #define DM35424_ADC_FE_CONFIG_GAIN_1 0x00

  Register value for setting a Gain of 1.
- #define DM35424_ADC_FE_CONFIG_GAIN_2 0x04

  Register value for setting a Gain of 2.
- #define DM35424_ADC_FE_CONFIG_GAIN_4 0x02

  Register value for setting a Gain of 4.
- #define DM35424_ADC_FE_CONFIG_GAIN_8 0x06

  Register value for setting a Gain of 8.
- #define DM35424_ADC_FE_CONFIG_GAIN_16 0x01

  Register value for setting a Gain of 16.
- #define DM35424_ADC_FE_CONFIG_GAIN_32 0x05

  Register value for setting a Gain of 32.
- #define DM35424_ADC_FE_CONFIG_GAIN_64 0x03

  Register value for setting a Gain of 64.
- #define DM35424_ADC_FE_CONFIG_GAIN_128 0x07

  Register value for setting a Gain of 128.
- #define DM35424_ADC_FE_CONFIG_POWER_MASK 0x80

  Bit mask for the channel power bit of the FE Config.
- #define DM35424_ADC_FE_CONFIG_PGA_MASK 0x40

  Bit mask for the channel PGA bit of the FE Config.
- #define DM35424_ADC_FE_CONFIG_INPUT_SW_ENABLE_MASK 0x20

  Bit mask for the channel input switch bit of the FE Config.
- #define DM35424_ADC_FE_CONFIG_INPUT_LINE_MASK 0x18

  Bit mask for the channel input line bits of the FE Config.
- #define DM35424_ADC_FE_CONFIG_GAIN_MASK 0x07

  Bit mask for the channel gain bits of the FE Config.
- #define DM35424_ADC_THRESHOLD_MAX 8388607L

  Maximum allowable value to write to the threshold register.
- #define DM35424_ADC_THRESHOLD_MIN -8388608L

  Minimum allowable value to write to the threshold register.
- #define DM35424_ADC_HIGH_SPD_MIN_DIV 2

  Minimum divider allowed for HIGH SPEED mode.
- #define DM35424_ADC_HIGH_RES_MIN_DIV 2

  Minimum divider allowed for HIGH RES mode.
- #define DM35424_ADC_LOW_POW_MIN_DIV 4

  Minimum divider allowed for LOW POWER mode.
- #define DM35424_ADC_LOW_SPD_MIN_DIV 10

  Minimum divider allowed for LOW SPEED mode.
- #define DM35424_ADC_MAX_RATE 106000

  Max rate of the ADC (Hz)
- #define DM35424_ADC_MAX_VALUE 8388607

  Max possible value for ADC.
- #define DM35424_ADC_MIN_VALUE -8388608

  Min possible value for ADC.

Enumerations

- enum DM35424_Adc_Clock_Events {
  DM35424_ADC_CLK_BUS_SRC_DISABLE = 0x00, DM35424_ADC_CLK_BUS_SRC_SAMPLE_TAKEN = 0x80, DM35424_ADC_CLK_BUS_SRC_CHAN_THRESH = 0x81, DM35424_ADC_CLK_BUS_SRC_PRE_START_BUFF_FULL = 0x82,
  DM35424_ADC_CLK_BUS_SRC_START_TRIG = 0x83, DM35424_ADC_CLK_BUS_SRC_STOP_TRIG = 0x84, DM35424_ADC_CLK_BUS_SRC_POST_STOP_BUFF_FULL = 0x85, DM35424_ADC_CLK_BUS_SRC_SAMPLING_COMPLETE = 0x86,
  DM35424_ADC_CLK_BUS_SRC_PACER_TICK = 0x87 }

  Clock events for the global source clocks.
- enum DM35424_Input_Ranges {
  DM35424_ADC_RNG_BIPOLAR_2_5V, DM35424_ADC_RNG_BIPOLAR_1_25V, DM35424_ADC_RNG_BIPOLAR_625mV, DM35424_ADC_RNG_BIPOLAR_312mV,
  DM35424_ADC_RNG_BIPOLAR_156mV, DM35424_ADC_RNG_BIPOLAR_78mV, DM35424_ADC_RNG_BIPOLAR_39mV, DM35424_ADC_RNG_BIPOLAR_19mV,
  DM35424_ADC_RNG_UNIPOLAR_5V }

  Input range of the ADC input pin. This combines polarity and gain into a single enumeration, and is the preferred way of setting polarity and gain.
- enum DM35424_Input_Mode { DM35424_ADC_INPUT_DIFFERENTIAL, DM35424_ADC_INPUT_SINGLE_ENDED_POS, DM35424_ADC_INPUT_SINGLE_ENDED_NEG, DM35424_ADC_INPUT_DAC_LOOPBACK }

  Input mode of the ADC pin.
- enum DM35424_Gains {
  DM35424_ADC_GAIN_05, DM35424_ADC_GAIN_1, DM35424_ADC_GAIN_2, DM35424_ADC_GAIN_4,
  DM35424_ADC_GAIN_8, DM35424_ADC_GAIN_16, DM35424_ADC_GAIN_32, DM35424_ADC_GAIN_64,
  DM35424_ADC_GAIN_128 }

  Input gain to apply to the incoming signal. Note that the preferred method of setting the gain is through the input range enumeration.
- enum DM35424_Sampling_Mode { DM35424_ADC_MODE_CONFIG_HIGH_SPEED =0x01, DM35424_ADC_MODE_CONFIG_HIGH_RES =0x03, DM35424_ADC_MODE_CONFIG_LOW_POWER =0x04, DM35424_ADC_MODE_CONFIG_LOW_SPEED =0x06 }

  Sampling mode for the AD Config Register.

Functions

- DM35424LIB_API int DM35424_Adc_Open (struct DM35424_Board_Descriptor *handle, unsigned int number_of_type, struct DM35424_Function_Block *func_block)

  Open the ADC indicated, and determine register locations of control blocks needed to control it.
- DM35424LIB_API int DM35424_Adc_Get_Start_Trigger (struct DM35424_Board_Descriptor *handle, struct DM35424_Function_Block *func_block, uint8_t *start_trigger)

  Get the start trigger for data collection.
- DM35424LIB_API int DM35424_Adc_Set_Start_Trigger (struct DM35424_Board_Descriptor *handle, struct DM35424_Function_Block *func_block, uint8_t start_trigger)

  Set the start trigger for data collection.
- DM35424LIB_API int DM35424_Adc_Get_Stop_Trigger (struct DM35424_Board_Descriptor *handle, struct DM35424_Function_Block *func_block, uint8_t *stop_trigger)

  Get the stop trigger for data collection.
- DM35424LIB_API int DM35424_Adc_Set_Stop_Trigger (struct DM35424_Board_Descriptor *handle, struct DM35424_Function_Block *func_block, uint8_t stop_trigger)

  Set the stop trigger for data collection.
- DM35424LIB_API int DM35424_Adc_Get_Pre_Trigger_Samples (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint32_t *count)

  Get the amount of data to capture prior to start trigger.

- DM35424LIB_API int DM35424_Adc_Set_Pre_Trigger_Samples (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint32_t count)

  Set the amount of data to capture prior to start trigger.
- DM35424LIB_API int DM35424_Adc_Get_Post_Stop_Samples (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint32_t *count)

  Get the amount of data to capture after stop trigger.
- DM35424LIB_API int DM35424_Adc_Set_Post_Stop_Samples (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint32_t count)

  Set the amount of data to capture after stop trigger.
- DM35424LIB_API int DM35424_Adc_Get_Clock_Src (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, enum DM35424_Clock_Sources *source)

  Get the clock source for the ADC.
- DM35424LIB_API int DM35424_Adc_Set_Clock_Src (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, enum DM35424_Clock_Sources source)

  Set the clock source for the ADC.
- DM35424LIB_API int DM35424_Adc_Initialize (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block)

  Prepare the ADC for actual data collection. Moves the ADC from uninitialized to stopped.
- DM35424LIB_API int DM35424_Adc_Set_Clk_Divider (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint32_t divider)

  Set the Clock Divider for the ADC function block.
- DM35424LIB_API int DM35424_Adc_Set_Sample_Rate (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint32_t rate, uint32_t *actual_rate)

  Set the sampling rate for the ADC.
- DM35424LIB_API int DM35424_Adc_Channel_Get_Front_End_Config (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, uint16_t *fe_config)

  Get the front-end config register contents.
- DM35424LIB_API int DM35424_Adc_Ad_Config_Set_Mode (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, enum DM35424_Sampling_Mode mode)

  Set the Configuration of the AD Mode.
- DM35424LIB_API int DM35424_Adc_Ad_Config_Get_Mode (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint16_t *mode)

  Get AD Config register.
- DM35424LIB_API int DM35424_Adc_Interrupt_Set_Config (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint16_t int_source, int enable)

  Configure the interrupts for the ADC.
- DM35424LIB_API int DM35424_Adc_Interrupt_Get_Config (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint16_t *interrupt_ena)

  Get the interrupt configuration for the ADC.
- DM35424LIB_API int DM35424_Adc_Start (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block)

  Set the ADC mode to Start.
- DM35424LIB_API int DM35424_Adc_Start_Rearm (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block)

  Set the ADC mode to Start-Rearm.
- DM35424LIB_API int DM35424_Adc_Reset (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block)

  Set the ADC mode to Reset.
- DM35424LIB_API int DM35424_Adc_Pause (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block)

  Set the ADC mode to Pause.
- DM35424LIB_API int DM35424_Adc_Uninitialize (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block)

Set the ADC mode to Uninitialized.

- DM35424LIB_API int DM35424_Adc_Get_Mode_Status (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint8_t *mode_status)

  Get the ADC mode-status value.

- DM35424LIB_API int DM35424_Adc_Channel_Get_Last_Sample (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, int32_t *value)

  Get the last sample taken from the ADC.

- DM35424LIB_API int DM35424_Adc_Get_Sample_Count (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint32_t *value)

  Get the count of number of samples taken.

- DM35424LIB_API int DM35424_Adc_Interrupt_Get_Status (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint16_t *value)

  Get the interrupt status register.

- DM35424LIB_API int DM35424_Adc_Interrupt_Clear_Status (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint16_t value)

  Clear the interrupt status register.

- DM35424LIB_API int DM35424_Adc_Channel_Setup (struct DM35424_Board_Descriptor *handle, struct DM35424_Function_Block *func_block, unsigned int channel, enum DM35424_Input_Ranges input_range, enum DM35424_Input_Mode input_mode)

  Setup the channel input for the ADC.

- DM35424LIB_API int DM35424_Adc_Channel_Reset (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel)

  Reset the channel front-end config.

- DM35424LIB_API int DM35424_Adc_Channel_Interrupt_Set_Config (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, uint8_t interrupts_to_set, int enable)

  Setup the channel interrupts.

- DM35424LIB_API int DM35424_Adc_Channel_Interrupt_Get_Config (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, uint8_t *chan_intr_enable)

  Get the channel interrupt configuration.

- DM35424LIB_API int DM35424_Adc_Channel_Interrupt_Get_Status (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, uint8_t *chan_intr_status)

  Get the channel interrupt status.

- DM35424LIB_API int DM35424_Adc_Channel_Interrupt_Clear_Status (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, uint8_t chan_intr_status)

  Clear the interrupt status for this channel.

- DM35424LIB_API int DM35424_Adc_Channel_Find_Interrupt (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int *channel_with_interrupt, int *channel_has_interrupt, uint8_t *channel_intr_status, uint8_t *channel_intr_enable)

  Find the first channel with an interrupt. Note that this is only useful when looking for a threshold interrupt.

- DM35424LIB_API int DM35424_Adc_Channel_Set_Filter (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, uint8_t chan_filter)

  Set the filter value for the channel.

- DM35424LIB_API int DM35424_Adc_Channel_Get_Filter (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, uint8_t *chan_filter)

  Get the filter value for the channel.

- DM35424LIB_API int DM35424_Adc_Channel_Set_Low_Threshold (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, int32_t threshold)

  Set the lower threshold for this channel.

- DM35424LIB_API int DM35424_Adc_Channel_Set_High_Threshold (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, int32_t threshold)

Set the high threshold for this channel.

- DM35424LIB_API int DM35424_Adc_Channel_Get_Thresholds (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, int32_t *low_threshold, int32_t *high_threshold)

  Get both thresholds for this channel.
- DM35424LIB_API int DM35424_Adc_Fifo_Channel_Read (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, int32_t *value)

  Read an ADC sample stored in the onboard FIFO.
- DM35424LIB_API int DM35424_Adc_Set_Clock_Source_Global (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, enum DM35424_Clock_Sources clock_select, enum DM35424_Adc_Clock_Events clock_driver)

  Set the global clock source for the ADC.
- DM35424LIB_API int DM35424_Adc_Get_Clock_Source_Global (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, int clock_select, int *clock_source)

  Get the global clock source for the selected clock.
- DM35424LIB_API int DM35424_Adc_Sample_To_Volts (enum DM35424_Input_Ranges input_range, int32_t adc_sample, float *volts)

  Convert an ADC sample to a volts value.
- DM35424LIB_API int DM35424_Adc_Volts_To_Sample (enum DM35424_Input_Ranges input_range, float volts, int32_t *adc_sample)

  Convert volts to an ADC value.

6.15.1 Detailed Description

Definitions for the DM35424 ADC Library.

Id:

dm35424_adc_library.h 106898 2017-03-08 13:44:23Z rgroner

Definition in file dm35424_adc_library.h.

6.16 include/dm35424_board_access.h File Reference

Structures for the DM35424 Board Access Library.

```
#include <stdint.h>
#include "dm35424_board_access_structs.h"
#include "dm35424_types.h"
#include "dm35424_os.h"
```

Data Structures

- struct DM35424_DMA_Descriptor

  Descriptor for the DMA on this board.
- struct DM35424_Function_Block

  DM35424 function block descriptor. This structure holds information about a function block, including type, number of DMA channels and buffers, descriptors for each DMA channel, and memory offsets to various control locations.

Macros

- #define DM35424LIB_API

Functions

- DM35424LIB_API int DM35424_Board_Open (uint8_t dev_num, struct DM35424_Board_Descriptor **handle)

  Open the board, providing the file descriptor that all future operations will reference. Also allocate memory for the device descriptor.
- DM35424LIB_API int DM35424_Board_Close (struct DM35424_Board_Descriptor *handle)

  Close the board, closing the open handle for the device file, and freeing the memory allocated for the decriptor.
- DM35424LIB_API int DM35424_Read (struct DM35424_Board_Descriptor *handle, union dm35424_ioctl_argument *ioctl_request)

  Read from the board.
- DM35424LIB_API int DM35424_Write (struct DM35424_Board_Descriptor *handle, union dm35424_ioctl_argument *ioctl_request)

  Write to the board.
- DM35424LIB_API int DM35424_Modify (struct DM35424_Board_Descriptor *handle, union dm35424_ioctl_argument *ioctl_request)

  Read/Modify/Write to the board.
- int DM35424_Dma (struct DM35424_Board_Descriptor *handle, union dm35424_ioctl_argument *ioctl_request)

  Perform a DMA operation.

6.16.1 Detailed Description

Structures for the DM35424 Board Access Library.

Id:

dm35424_board_access.h 104840 2016-11-30 19:20:54Z rgroner

Definition in file dm35424_board_access.h.

6.16.2 Macro Definition Documentation

6.16.2.1 #define DM35424LIB_API

Conditionally set up the library export symbol for the Windows DLL. This will expand to nothing when compiled for Linux.

Definition at line 47 of file dm35424_board_access.h.

6.17 include/dm35424_board_access_structs.h File Reference

Structures for the DM35424 Board Access Library.

Data Structures

- struct dm35424_pci_access_request

  PCI region access request descriptor. This structure holds information about a request to read data from or write data to one of a device's PCI regions.
- struct dm35424_ioctl_region_readwrite

  ioctl() request structure for read from or write to PCI region
- struct dm35424_ioctl_region_modify

ioctl() request structure for PCI region read/modify/write

- struct dm35424_ioctl_interrupt_info_request

  ioctl() request structure for interrupt

- struct dm35424_ioctl_dma

  ioctl() request structure for DMA

- union dm35424_ioctl_argument

  ioctl() request structure encapsulating all possible requests. This is what gets passed into the kernel from user space on the ioctl() call.

Enumerations

- enum dm35424_pci_region_num { DM35424_PCI_REGION_GBC = 0, DM35424_PCI_REGION_GBC2, DM35424_PCI_REGION_FB }

  Standard PCI region number.

- enum dm35424_pci_region_access_size { DM35424_PCI_REGION_ACCESS_8 = 0, DM35424_PCI_REGION_ACCESS_16, DM35424_PCI_REGION_ACCESS_32 }

  Desired size in bits of access to standard PCI region.

- enum DM35424_DMA_FUNCTIONS { DM35424_DMA_INITIALIZE, DM35424_DMA_READ, DM35424_DMA_WRITE }

  Enumeration for DMA functions that can be requested for the driver to perform.

6.17.1 Detailed Description

Structures for the DM35424 Board Access Library.

Id:

dm35424_board_access_structs.h 80523 2014-07-17 18:38:59Z rgroner

Definition in file dm35424_board_access_structs.h.

6.18 include/dm35424_dac_library.h File Reference

Definitions for the DM35424 DAC Library.

Macros

- #define DM35424_DAC_INT_CONVERSION_SENT_MASK 0x01

  Register value for Interrupt Mask - Conversion Sent.

- #define DM35424_DAC_INT_CHAN_MARKER_MASK 0x02

  Register value for Interrupt Mask - Channel has enabled marker.

- #define DM35424_DAC_INT_START_TRIG_MASK 0x08

  Register value for Interrupt Mask - Start Trigger Occurred.

- #define DM35424_DAC_INT_STOP_TRIG_MASK 0x10

  Register value for Interrupt Mask - Stop Trigger Occurred.

- #define DM35424_DAC_INT_POST_STOP_DONE_MASK 0x20

  Register value for Interrupt Mask - Post-Stop Conversions Completed.

- #define DM35424_DAC_INT_PACER_TICK_MASK 0x80

  Register value for Interrupt Mask - Pacer Clock Tick.

- #define DM35424_DAC_INT_ALL_MASK 0xBB

  Register value for Interrupt Mask - All Bits.

- #define DM35424_DAC_MODE_RESET 0x00

  Register value for Mode - Reset.
- #define DM35424_DAC_MODE_PAUSE 0x01

  Register value for Mode - Pause.
- #define DM35424_DAC_MODE_GO_SINGLE_SHOT 0x02

  Register value for Mode - Go (Single Shot)
- #define DM35424_DAC_MODE_GO_REARM 0x03

  Register value for Mode - Go (Re-arm)
- #define DM35424_DAC_STATUS_STOPPED 0x00

  Register value for DAC Status - Stopped.
- #define DM35424_DAC_STATUS_WAITING_START_TRIG 0x02

  Register value for DAC Status - Waiting for Start Trigger.
- #define DM35424_DAC_STATUS_CONVERTING 0x03

  Register value for DAC Status - Converting Data.
- #define DM35424_DAC_STATUS_OUTPUT_POST 0x04

  Register value for DAC Status - Outputting Post-Stop conversions.
- #define DM35424_DAC_STATUS_WAITING_REARM 0x05

  Register value for DAC Status - Waiting for Re-Arm.
- #define DM35424_DAC_STATUS_DONE 0x07

  Register value for DAC Status - Done.
- #define DM35424_DAC_MAX_RATE 106000

  Max allowable rate for the DAC (Hz)
- #define DM35424_DAC_MAX_VALUE 32767

  Max value of the DAC.
- #define DM35424_DAC_MIN_VALUE -32768

  Min value of the DAC.
- #define DM35424_DAC_LSB_AT_MIN_RANGE 0.000152587890625f

  DAC LSB (at lowest voltage range)

Enumerations

- enum DM35424_Dac_Clock_Events {
  DM35424_DAC_CLK_BUS_SRC_DISABLE = 0x00, DM35424_DAC_CLK_BUS_SRC_CONVERSION_SENT = 0x80, DM35424_DAC_CLK_BUS_SRC_CHAN_MARKER = 0x81, DM35424_DAC_CLK_BUS_SRC_START_TRIG = 0x83,
  DM35424_DAC_CLK_BUS_SRC_STOP_TRIG = 0x84, DM35424_DAC_CLK_BUS_SRC_CONV_COMPL = 0x85 }

  Clocking events that can be used as the global clock sources.

Functions

- DM35424LIB_API int DM35424_Dac_Open (struct DM35424_Board_Descriptor *handle, unsigned int number_of_type, struct DM35424_Function_Block *func_block)

  Open the DAC indicated, and determine register locations of control blocks needed to control it.
- DM35424LIB_API int DM35424_Dac_Set_Clock_Src (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, enum DM35424_Clock_Sources source)

  Set the clock source of the DAC.
- DM35424LIB_API int DM35424_Dac_Get_Clock_Src (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, enum DM35424_Clock_Sources *source)

  Get the clock source of the DAC.
- DM35424LIB_API int DM35424_Dac_Get_Clock_Div (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint32_t *divider)

Get the clock divider value.

- DM35424LIB_API int DM35424_Dac_Set_Clock_Div (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint32_t divider)

  Set the clock divider value.
- DM35424LIB_API int DM35424_Dac_Set_Conversion_Rate (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint32_t requested_rate, uint32_t *actual_rate)

  Set the conversion rate of this DAC.
- DM35424LIB_API int DM35424_Dac_Interrupt_Set_Config (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint16_t interrupt_src, int enable)

  Set the interrupt configuration for this DAC.
- DM35424LIB_API int DM35424_Dac_Interrupt_Get_Config (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint16_t *interrupt_ena)

  Get the interrupt configuration for this DAC.
- DM35424LIB_API int DM35424_Dac_Set_Start_Trigger (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint8_t trigger_value)

  Set the start trigger.
- DM35424LIB_API int DM35424_Dac_Set_Stop_Trigger (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint8_t trigger_value)

  Set the stop trigger.
- DM35424LIB_API int DM35424_Dac_Get_Start_Trigger (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint8_t *trigger_value)

  Get the start trigger.
- DM35424LIB_API int DM35424_Dac_Get_Stop_Trigger (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint8_t *trigger_value)

  Get the stop trigger.
- DM35424LIB_API int DM35424_Dac_Start (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block)

  Set the DAC Mode to Start.
- DM35424LIB_API int DM35424_Dac_Reset (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block)

  Set the DAC Mode to Reset.
- DM35424LIB_API int DM35424_Dac_Pause (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block)

  Set the DAC Mode to Pause.
- DM35424LIB_API int DM35424_Dac_Get_Mode_Status (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint8_t *mode_status)

  Get the Mode and Status of the DAC.
- DM35424LIB_API int DM35424_Dac_Get_Last_Conversion (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, uint8_t *marker, int16_t *value)

  Get the value of the last conversion of the DAC.
- DM35424LIB_API int DM35424_Dac_Set_Last_Conversion (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, uint8_t marker, int16_t value)

  Set a value to be converted by the DAC immediately.
- DM35424LIB_API int DM35424_Dac_Get_Conversion_Count (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint32_t *value)

  Get a count of the number of conversions that DAC has executed.
- DM35424LIB_API int DM35424_Dac_Interrupt_Get_Status (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint16_t *value)

  Get a interrupt status register of the DAC.
- DM35424LIB_API int DM35424_Dac_Interrupt_Clear_Status (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint16_t value)

  Clear the interrupt status register of the DAC.

- DM35424LIB_API int DM35424_Dac_Set_Post_Stop_Conversion_Count (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint32_t value)

  Set the number of conversions the DAC will make after a stop trigger.
- DM35424LIB_API int DM35424_Dac_Get_Post_Stop_Conversion_Count (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint32_t *value)

  Get the number of conversions the DAC will make after a stop trigger.
- DM35424LIB_API int DM35424_Dac_Set_Clock_Source_Global (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, enum DM35424_Clock_Sources clock, enum DM35424_Dac_Clock_Events clock_driver)

  Set the source that will drive the global clock.
- DM35424LIB_API int DM35424_Dac_Channel_Set_Marker_Config (struct DM35424_Board_Descriptor *handle, struct DM35424_Function_Block *func_block, unsigned int channel, uint8_t marker_enable)

  Set the configuration of the marker interrupts for this channel.
- DM35424LIB_API int DM35424_Dac_Channel_Get_Marker_Config (struct DM35424_Board_Descriptor *handle, struct DM35424_Function_Block *func_block, unsigned int channel, uint8_t *marker_enable)

  Get the configuration of the marker interrupts for this channel.
- DM35424LIB_API int DM35424_Dac_Channel_Get_Marker_Status (struct DM35424_Board_Descriptor *handle, struct DM35424_Function_Block *func_block, unsigned int channel, uint8_t *marker_status)

  Get the status of the marker interrupts for this channel.
- DM35424LIB_API int DM35424_Dac_Channel_Clear_Marker_Status (struct DM35424_Board_Descriptor *handle, struct DM35424_Function_Block *func_block, unsigned int channel, uint8_t marker_to_clear)

  Clear the marker interrupts for this channel.
- DM35424LIB_API int DM35424_Dac_Fifo_Channel_Write (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, int32_t value)

  Write a value to the onboard FIFO.
- DM35424LIB_API int DM35424_Dac_Volts_To_Conv (float volts, int16_t *dac_conversion)

  Convert a value in volts to a DAC equivalent signed value.
- DM35424LIB_API int DM35424_Dac_Conv_To_Volts (int16_t conversion, float *volts)

  Convert a DAC conversion value to volts.

6.18.1 Detailed Description

Definitions for the DM35424 DAC Library.

Id:

dm35424_dac_library.h 106898 2017-03-08 13:44:23Z rgroner

Definition in file dm35424_dac_library.h.

6.19 include/dm35424_dio_library.h File Reference

Definitions for the DM35424 DIO Library.

```
#include "dm35424_gbc_library.h"
```

Functions

- DM35424LIB_API int DM35424_Dio_Open (struct DM35424_Board_Descriptor *handle, unsigned int number_of_type, struct DM35424_Function_Block *func_block)

  Open the DIO indicated, and determine register locations of control blocks needed to control it.

- DM35424LIB_API int DM35424_Dio_Set_Direction (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint32_t direction)

  Set the direction of the DIO pins.
- DM35424LIB_API int DM35424_Dio_Get_Direction (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint32_t *direction)

  Get the direction of the DIO pins.
- DM35424LIB_API int DM35424_Dio_Get_Input_Value (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint32_t *value)

  Get the input value of the DIO.
- DM35424LIB_API int DM35424_Dio_Get_Output_Value (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint32_t *value)

  Get the current value of the output register.
- DM35424LIB_API int DM35424_Dio_Set_Output_Value (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, uint32_t value)

  Set the value to be put on output pins.

6.19.1 Detailed Description

Definitions for the DM35424 DIO Library.

Id:

dm35424_dio_library.h 84663 2014-12-19 22:03:35Z mmcintire

Definition in file dm35424_dio_library.h.

6.20 include/dm35424_dma_library.h File Reference

Definitions for the DM35424 DMA Library.

```
#include <stdint.h>
```

Macros

- #define DM35424_DMA_ACTION_CLEAR 0x00

  Register value for DMA clear action.
- #define DM35424_DMA_ACTION_GO 0x01

  Register value for DMA go action.
- #define DM35424_DMA_ACTION_PAUSE 0x02

  Register value for DMA pause action.
- #define DM35424_DMA_ACTION_HALT 0x03

  Register value for DMA halt action.
- #define DM35424_DMA_SETUP_DIRECTION_READ 0x04

  Register value to set DMA to READ direction.
- #define DM35424_DMA_SETUP_DIRECTION_WRITE 0x00

  Register value to set DMA to WRITE direction.
- #define DM35424_DMA_SETUP_DIRECTION_MASK 0x04

  Register value to set DMA to READ direction.
- #define DM35424_DMA_SETUP_IGNORE_USED 0x08

  Register value to tell DMA to ignore used buffers.
- #define DM35424_DMA_SETUP_NOT_IGNORE_USED 0x00

Register value to tell DMA to not ignore used buffers.

- #define DM35424_DMA_SETUP_IGNORE_USED_MASK 0x08

  Bit mask for Ignore Used bit in setup register.

- #define DM35424_DMA_SETUP_INT_ENABLE 0x01

  Register value to enabled interrupts in the setup register.

- #define DM35424_DMA_SETUP_INT_DISABLE 0x00

  Register value to disable interrupts in the setup register.

- #define DM35424_DMA_SETUP_INT_MASK 0x01

  Bit mask for the interrupt bit in the setup register.

- #define DM35424_DMA_SETUP_ERR_INT_ENABLE 0x02

  Register value to enable the error interrupt.

- #define DM35424_DMA_SETUP_ERR_INT_DISABLE 0x00

  Register value to disable the error interrupt.

- #define DM35424_DMA_SETUP_ERR_INT_MASK 0x02

  Bit mask for the error interrupt bit in the setup register.

- #define DM35424_DMA_STATUS_CLEAR 0x00

  Register value to write to status registers to clear them.

- #define DM35424_DMA_CTRL_CLEAR 0x00

  Register value to write to control register to clear it.

- #define DM35424_DMA_BUFFER_STATUS_CLEAR 0x00

  Register value to write to the buffer status register to clear it.

- #define DM35424_DMA_BUFFER_CTRL_CLEAR 0x00

  Register value to write to the buffer control register to clear it.

- #define DM35424_DMA_BUFFER_STATUS_USED_MASK 0x01

  Bit mask for the used buffer bit in the buffer status register.

- #define DM35424_DMA_BUFFER_STATUS_TERM_MASK 0x02

  Bit mask for the terminated buffer bit in the buffer status register.

- #define DM35424_DMA_BUFFER_CTRL_VALID 0x01

  Register value to write to buffer control register to mark it as valid.

- #define DM35424_DMA_BUFFER_CTRL_HALT 0x02

  Register value to write to buffer control register to tell DMA to halt after processing this buffer.

- #define DM35424_DMA_BUFFER_CTRL_LOOP 0x04

  Register value to write to buffer control register to tell DMA to loop back to buffer 0 after using this buffer.

- #define DM35424_DMA_BUFFER_CTRL_INTR 0x08

  Register value to write to buffer control register to tell DMA to issue an interrupt after using this buffer.

- #define DM35424_DMA_BUFFER_CTRL_PAUSE 0x10

  Register value to write to buffer control register to tell DMA to pause after processing this buffer.

- #define DM35424_DMA_CTRL_BLOCK_SIZE 0x10

  Constant value indicating DMA control block size.

- #define DM35424_DMA_BUFFER_CTRL_BLOCK_SIZE 0x10

  Constant value indicating DMA buffer control block size.

- #define DM35424_BIT_MASK_DMA_BUFFER_SIZE 0x0FFFFFF

  Bit mask for the DMA buffer size, since it is 24-bits of a 32-bit register.

Enumerations

- enum DM35424_Fifo_States { DM35424_FIFO_UNKNOWN, DM35424_FIFO_EMPTY, DM35424_FIFO_FULL, DM35424_FIFO_HAS_DATA }

  Descriptions of the possible states the FIFO might be in.

Functions

- DM35424LIB_API int DM35424_Dma_Start (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel)

  Start the DMA.
- DM35424LIB_API int DM35424_Dma_Stop (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel)

  Stop the DMA.
- DM35424LIB_API int DM35424_Dma_Pause (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel)

  Pause the DMA.
- DM35424LIB_API int DM35424_Dma_Clear (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel)

  Clear the DMA.
- DM35424LIB_API int DM35424_Dma_Get_Fifo_Counts (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, uint16_t *write_count, uint16_t *read_count)

  Get the Read and Write FIFO count values.
- DM35424LIB_API int DM35424_Dma_Get_Fifo_State (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, enum DM35424_Fifo_States *state)

  Get the state of the FIFO.
- DM35424LIB_API int DM35424_Dma_Configure_Interrupts (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, int enable, int error_enable)

  Configure the interrupts for the DMA channel.
- DM35424LIB_API int DM35424_Dma_Get_Interrupt_Configuration (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, int *enable, int *error_enable)

  Get the configuration of the interrupts for the DMA channel.
- DM35424LIB_API int DM35424_Dma_Setup (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, int direction, int ignore_used)

  Setup the DMA channel, specifically the direction and if used buffers are ignored.
- DM35424LIB_API int DM35424_Dma_Setup_Set_Direction (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, int direction)

  Set the direction of the DMA, read or write.
- DM35424LIB_API int DM35424_Dma_Setup_Set_Used (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, int ignore_used)

  Set the DMA channel to ignore or not ignore a used buffer. Ignoring used buffers is mostly useful when outputting a repeating data cycle.
- DM35424LIB_API int DM35424_Dma_Get_Errors (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, int *stat_overflow, int *stat_underflow, int *stat_used, int *stat_invalid)

  Get the current value of the DMA channel error registers.
- DM35424LIB_API int DM35424_Dma_Status (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, uint32_t *current_buffer, uint32_t *current_count, int *current_action, int *stat_overflow, int *stat_underflow, int *stat_used, int *stat_invalid, int *stat_complete)

  Get the current status of the DMA channel. Determine which buffer it is using, what its current action is, and the state of all error conditions and normal interrupt conditions.
- DM35424LIB_API int DM35424_Dma_Get_Current_Buffer_Count (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, uint32_t *current_buffer, uint32_t *current_count)

  Get the current buffer and buffer count in use by the DMA.
- DM35424LIB_API int DM35424_Dma_Check_For_Error (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, int *has_error)

Check the DMA channel for any error conditions. This just returns a simple boolean as quickly as possible. If there is an error condition, you will have to query the DMA again to determine what the error is.

- DM35424LIB_API int DM35424_Dma_Buffer_Setup (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, unsigned int buffer, uint8_t ctrl)

  Setup the DMA buffer for use.
- DM35424LIB_API int DM35424_Dma_Buffer_Status (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, unsigned int buffer, uint8_t *status, uint8_t *control, uint32_t *size)

  Get the status of the buffer. This gets the status, control, and size registers.
- DM35424LIB_API int DM35424_Dma_Check_Buffer_Used (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, unsigned int buffer_num, int *is_used)

  Check if the indicated buffer has the "Used" flag set.
- int DM35424_Dma_Find_Interrupt (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int *channel, int *channel_complete, int *channel_error)

  Find which DMA channel has an interrupt condition, whether from using a buffer with interrupt set, or from an error. DMA channels are evaluated starting at Channel 0.
- int DM35424_Dma_Clear_Interrupt (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, int clear_overflow, int clear_underflow, int clear_used, int clear_invalid, int clear_complete)

  Clear the interrupt flag from a DMA channel. Clearing the flags will allow another interrupt of the same type to occur again, and is the normal operation after handling the interrupt itself.
- int DM35424_Dma_Reset_Buffer (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, unsigned int buffer)

  Reset the DMA buffer, preparing it to be used again by the DMA engine.

6.20.1 Detailed Description

Definitions for the DM35424 DMA Library.

Id:

dm35424_dma_library.h 105018 2016-12-07 15:47:31Z rgroner

Definition in file dm35424_dma_library.h.

6.21 include/dm35424_driver.h File Reference

Structures and defines for the DM35424 driver module.

```
#include <linux/pci.h>
#include <linux/spinlock.h>
#include <linux/types.h>
```

Data Structures

- struct dm35424_pci_region

  DM35424 PCI region descriptor. This structure holds information about one of a device's PCI memory regions.
- struct dm35424_dma_descriptor

  DM35424 DMA descriptor. This structure holds information about a single DMA buffer.
- struct dm35424_device_descriptor

  DM35424 Device Descriptor. The identifying info for this particular board.

Macros

- #define DM35424_NAME_LENGTH 200

  DM35424 Max possible board name length.
- #define DM35424_PCI_NUM_REGIONS PCI_ROM_RESOURCE

  Number of standard PCI regions.
- #define DM35424_INT_QUEUE_SIZE 256

  Number of interrupts to hold in a queue for processing.

Enumerations

- enum dm35424_pci_region_access_dir { DM35424_PCI_REGION_ACCESS_READ = 0, DM35424_PCI_REGION_ACCESS_WRITE }

  Direction of access to standard PCI region.

Variables

- static struct file_operations dm35424_file_ops

  Placeholder protoype for file ops struct.

6.21.1 Detailed Description

Structures and defines for the DM35424 driver module.

Id:

dm35424_driver.h 80523 2014-07-17 18:38:59Z rgroner

Definition in file dm35424_driver.h.

6.22 include/dm35424_examples.h File Reference

Defines for the DM35424 Example programs. Commonly used constants for the example programs included with the software package.

Macros

- #define BUFFER_VALID 1

  Boolean indicating buffer valid.
- #define BUFFER_NO_VALID 0

  Boolean indicating buffer not valid.
- #define BUFFER_HALT 1

  Boolean indicating buffer halt set.
- #define BUFFER_NO_HALT 0

  Boolean indicating buffer halt not set.
- #define BUFFER_LOOP 1

  Boolean indicating buffer loop set.
- #define BUFFER_NO_LOOP 0

  Boolean indicating buffer loop not set.
- #define BUFFER_INTERRUPT 1

  Boolean indicating buffer interrupt.

- #define BUFFER_NO_INTERRUPT 0

  Boolean indicating no buffer interrupt.
- #define BUFFER_PAUSE 1

  Boolean indicating buffer should pause when filled.
- #define BUFFER_NO_PAUSE 0

  Boolean indicating buffer should not pause when filled.
- #define IGNORE_USED 1

  Boolean indicating ignore used buffers.
- #define NOT_IGNORE_USED 0

  Boolean indicating not ignore used buffers.
- #define CLEAR_INTERRUPT 1

  Boolean indicating to clear an interrupt.
- #define NO_CLEAR_INTERRUPT 0

  Boolean indicating to not clear an interrupt.
- #define INTERRUPT_ENABLE 1

  Boolean indicating interrupt enable.
- #define INTERRUPT_DISABLE 0

  Boolean indicating interrupt disable.
- #define ERROR_INTR_ENABLE 1

  Boolean indicating error interrupt enable.
- #define ERROR_INTR_DISABLE 0

  Boolean indicating error interrupt disable.
- #define SYNCBUS_NONE 0

  Value indicating no Syncbus option was chosen.
- #define SYNCBUS_MASTER 1

  Value indicating Syncbus Master was chosen.
- #define SYNCBUS_SLAVE 2

  Value indicating Syncbus Slave was chosen.
- #define CHANNEL_0 0

  Constant for selecting Channel 0.
- #define CHANNEL_1 1

  Constant for selecting Channel 1.
- #define CHANNEL_2 2

  Constant for selecting Channel 2.
- #define CHANNEL_3 3

  Constant for selecting Channel 3.
- #define BUFFER_0 0

  Constant for selecting Buffer 0.
- #define BUFFER_1 1

  Constant for selecting Buffer 1.
- #define ADC_0 0

  Constant for selecting ADC 0.
- #define ADC_1 1

  Constant for selecting ADC 1.
- #define DAC_0 0

  Constant for selecting DAC 0.
- #define DAC_1 1

  Constant for selecting DAC 1.
- #define DAC_2 2

  Constant for selecting DAC 2.
- #define DAC_3 3

Constant for selecting DAC 3.

- #define REF_0 0

  Constant for selecting REF 0.

- #define REF_1 1

  Constant for selecting REF 1.

- #define DIO_0 0

  Constant for selecting DIO 0.

- #define ADIO_0 0

  Constant for selecting ADIO 0.

- #define ENABLED 1

  Constant to indicate an Enabled value.

- #define DISABLED 0

  Constant to indicate a Disabled value.

Enumerations

- enum Help_Options {
  HELP_OPTION = 1, MINOR_OPTION, RATE_OPTION, CHANNELS_OPTION,
  FILE_OPTION, START_OPTION, WAVE_OPTION, TEST_OPTION,
  NOSTOP_OPTION, SYNCBUS_OPTION, DUMP_OPTION, HOURS_OPTION,
  OUTPUT_RMS_OPTION, OUTPUT_ADC_OPTION, ADC_NUM_OPTION, DAC_NUM_OPTION,
  ADC_OPTION, DAC_OPTION, PATTERN_OPTION, SAMPLES_OPTION,
  MODE_OPTION, AD_MODE_OPTION, REF_NUM_OPTION, BINARY_OPTION,
  SENDER_OPTION, RECEIVER_OPTION, RANGE_OPTION, REFILL_FIFO_OPTION,
  LOW_THRESHOLD_OPTION, PORT_OPTION, BAUD_OPTION, EXTERNAL_OPTION,
  SIZE_OPTION, VERBOSE_OPTION, USER_ID_OPTION, COUNT_OPTION,
  NUM_OPTION, SYNC_TERM_OPTION, BIN2TXT_OPTION, STORE_OPTION,
  TERM_OPTION, REFCLK_OPTION, OFILE_OPTION, PACKED_OPTION,
  MASTER_OPTION, SLAVE_OPTION, SYNC_CONN_OPTION }

  Constants used for parsing command line parameters of example programs.

6.22.1 Detailed Description

Defines for the DM35424 Example programs. Commonly used constants for the example programs included with the software package.

Id:

dm35424_examples.h 114740 2018-07-12 14:41:17Z prucz

Definition in file dm35424_examples.h.

6.23 include/dm35424_gbc_library.h File Reference

Definitions for the DM35424 Board Library, a library for accessing the board registers.

```
#include <stdint.h>
#include <time.h>
#include "dm35424_board_access.h"
```

Macros

- #define CLK_40MHZ 40000000

Functions

- DM35424LIB_API int DM35424_Gbc_Board_Reset (struct DM35424_Board_Descriptor *handle)

  Write the reset value to the correct register to initiate a board-level reset.

- DM35424LIB_API int DM35424_Gbc_Ack_Interrupt (struct DM35424_Board_Descriptor *handle)

  Send an End-Of-Interrupt acknowledgement to the board. This will cause any pending interrupts to re-issue. This is a protection against missing interrupts while in the interrupt handler.

- DM35424LIB_API int DM35424_Function_Block_Open (struct DM35424_Board_Descriptor *handle, unsigned int number, struct DM35424_Function_Block *func_block)

  Open a specific function block. Nothing is opened in a file sense, but the memory location for the function block is read and certain important values are read. A function block descriptor is allocated to hold the data that will be used every time this function block is accessed.

- DM35424LIB_API int DM35424_Function_Block_Open_Module (struct DM35424_Board_Descriptor *handle, uint32_t fb_type, unsigned int number_of_type, struct DM35424_Function_Block *func_block)

  Open a specific function block module. This is the same as opening a function block, except we are looking for a function block with a specific type. This is the method you would use to open the 2nd ADC, for example.

- DM35424LIB_API int DM35424_Gbc_Get_Format (struct DM35424_Board_Descriptor *handle, uint8_t *format_id)

  Get the format ID of the board.

- DM35424LIB_API int DM35424_Gbc_Get_Revision (struct DM35424_Board_Descriptor *handle, uint8_t *rev)

  Get the PDP revision number of the board.

- DM35424LIB_API int DM35424_Gbc_Get_Pdp_Number (struct DM35424_Board_Descriptor *handle, uint32_t *pdp_num)

  Get PDP Number of the board.

- DM35424LIB_API int DM35424_Gbc_Get_Fpga_Build (struct DM35424_Board_Descriptor *handle, uint32_t *fpga_build)

  Get the FPGA Build number of the board.

- DM35424LIB_API int DM35424_Gbc_Get_Sys_Clock_Freq (struct DM35424_Board_Descriptor *handle, uint32_t *clock_freq, int *is_std_clk)

  Get the measured frequency of the system clock of the board.

6.23.1 Detailed Description

Definitions for the DM35424 Board Library, a library for accessing the board registers.

Id:

dm35424_gbc_library.h 103741 2016-10-17 20:35:58Z rgroner

Definition in file dm35424_gbc_library.h.

6.24 include/dm35424_ioctl.h File Reference

DM35424 Low level ioctl() request descriptor structure and request code definitions.

```
#include <linux/types.h>
#include <linux/ioctl.h>
```

Macros

- #define DM35424_IOCTL_MAGIC 'D'

  Unique 8-bit value used to generate unique ioctl() request codes.
- #define DM35424_IOCTL_REQUEST_BASE 0x00

  First ioctl() request number.
- #define DM35424_IOCTL_REGION_READ

  ioctl() request code for reading from a PCI region
- #define DM35424_IOCTL_REGION_WRITE

  ioctl() request code for writing to a PCI region
- #define DM35424_IOCTL_REGION_MODIFY

  ioctl() request code for PCI region read/modify/write
- #define DM35424_IOCTL_DMA_FUNCTION

  ioctl() request code for DMA function
- #define DM35424_IOCTL_WAKEUP

  ioctl() request code for User ISR thread wake up
- #define DM35424_IOCTL_INTERRUPT_GET

  ioctl() request code to retrieve interrupt status information

6.24.1 Detailed Description

DM35424 Low level ioctl() request descriptor structure and request code definitions.

Id:

dm35424_ioctl.h 80523 2014-07-17 18:38:59Z rgroner

Definition in file dm35424_ioctl.h.

6.25 include/dm35424_os.h File Reference

Function declarations for the DM35424 that are Linux specific.

```
#include <pthread.h>
```

Data Structures

- struct DM35424_Board_Descriptor

  DM35424 board descriptor. This structure holds information about the board as a whole. It holds the file descriptor and ISR callback function, if applicable.

Functions

- int DM35424_Dma_Initialize (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, unsigned int num_buffers, uint32_t buffer_size)

  Initialize the DMA channel and prepare it for data. Interrupts are disabled, error conditions are cleared, buffers are allocated in kernel space and their status and controls are cleared.
- int DM35424_Dma_Read (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, unsigned int buffer_to_read_from, uint32_t buffer_size, void *local_buffer_ptr)

  Read data from the DMA buffer. Data is copied from kernel buffers to local user-space buffers.

- int DM35424_Dma_Write (struct DM35424_Board_Descriptor *handle, const struct DM35424_Function_Block *func_block, unsigned int channel, unsigned int buffer_to_write_to, uint32_t buffer_size, void *local_buffer_ptr)

  Write data to the DMA buffer. Data is copied from local user buffers to kernel buffers.
- int DM35424_General_RemoveISR (struct DM35424_Board_Descriptor *handle)

  Remove the ISR from the system interrupt.
- void * DM35424_General_WaitForInterrupt (void *ptr)

  Loop/Poll and wait for an interrupt to happen, then take action.
- int DM35424_General_InstallISR (struct DM35424_Board_Descriptor *handle, void(*isr_fnct))

  Start a thread that will sit and wait for an interrupt from the board, and call the user ISR when it happens.
- int DM35424_General_SetISRPriority (struct DM35424_Board_Descriptor *handle, int priority)

  Set the priority of the user ISR thread.

6.25.1 Detailed Description

Function declarations for the DM35424 that are Linux specific.

Id:

dm35424_os.h 109114 2017-06-09 07:13:20Z prucz

Definition in file dm35424_os.h.

6.26 include/dm35424_ref_adjust_library.h File Reference

Definitions for the DM35424 Reference Adjustment Library.

```
#include "dm35424_gbc_library.h"
```

Macros

- #define DM35424_REF_ADJUST_SPI_BUSY 0x00

  Register value for SPI Interface is busy.
- #define DM35424_REF_ADJUST_SPI_READY 0x01

  Register value for SPI Interface is ready.
- #define DM35424_REF_ADJUST_START_TRANS 0x01

  Register value for starting the SPI transaction.
- #define DM35424_REF_ADJUST_WRITE_ADC_VOLATILE 0x0100

  Register value for writing to the ADC Volatile memory.
- #define DM35424_REF_ADJUST_WRITE_DAC_VOLATILE 0x0200

  Register value for writing to the DAC Volatile memory.
- #define DM35424_REF_ADJUST_WRITE_ADC_NON_VOLATILE 0x1100

  Register value for writing to the ADC Non-Volatile memory.
- #define DM35424_REF_ADJUST_WRITE_DAC_NON_VOLATILE 0x1200

  Register value for writing to the DAC Non-Volatile memory.
- #define DM35424_REF_ADJUST_COPY_ADC_VOL_TO_NON 0x2100

  Register value for copying ADC data from Volatile to Non-Volatile.
- #define DM35424_REF_ADJUST_COPY_DAC_VOL_TO_NON 0x2200

  Register value for copying DAC data from Volatile to Non-Volatile.
- #define DM35424_REF_ADJUST_COPY_BOTH_VOL_TO_NON 0x2300

Register value for copying ADC and DAC data from Volatile to Non-Volatile.

- #define DM35424_REF_ADJUST_COPY_ADC_NON_TO_VOL 0x3100

  Register value for copying ADC data from Non-Volatile to Volatile.

- #define DM35424_REF_ADJUST_COPY_DAC_NON_TO_VOL 0x3200

  Register value for copying DAC data from Non-Volatile to Volatile.

- #define DM35424_REF_ADJUST_COPY_BOTH_NON_TO_VOL 0x3300

  Register value for copying ADC and DAC data from Non-Volatile to Volatile.

Enumerations

- enum DM35424_Copy_Directions {
  DM35424_ADC_VOL_TO_NON_VOL, DM35424_DAC_VOL_TO_NON_VOL, DM35424_BOTH_VOL_TO_-
  NON_VOL, DM35424_ADC_NON_VOL_TO_VOL,
  DM35424_DAC_NON_VOL_TO_VOL, DM35424_BOTH_NON_VOL_TO_VOL }

  Direction of Reference Adjustment data copy action.

Functions

- DM35424LIB_API int DM35424_Ref_Adjust_Open (struct DM35424_Board_Descriptor *handle, unsigned int ordinal_to_open, struct DM35424_Function_Block *fb_temp)

  Open the reference adjustment function block, getting address values that will be used later by other library functions.

- DM35424LIB_API int DM35424_Ref_Adjust_Write_Adc_To_Volatile (struct DM35424_Board_Descriptor *handle, struct DM35424_Function_Block *fb, uint8_t adjustment)

  Write the ADC Reference Adjustment value to volatile memory.

- DM35424LIB_API int DM35424_Ref_Adjust_Write_Adc_To_NonVolatile (struct DM35424_Board_Descriptor *handle, struct DM35424_Function_Block *fb, uint8_t adjustment)

  Write the ADC Reference Adjustment value to non-volatile memory.

- DM35424LIB_API int DM35424_Ref_Adjust_Write_Dac_To_Volatile (struct DM35424_Board_Descriptor *handle, struct DM35424_Function_Block *fb, uint8_t adjustment)

  Write the DAC Reference Adjustment value to volatile memory.

- DM35424LIB_API int DM35424_Ref_Adjust_Write_Dac_To_NonVolatile (struct DM35424_Board_Descriptor *handle, struct DM35424_Function_Block *fb, uint8_t adjustment)

  Write the DAC Reference Adjustment value to non-volatile memory.

- DM35424LIB_API int DM35424_Ref_Adjust_Copy_Data (struct DM35424_Board_Descriptor *handle, struct DM35424_Function_Block *fb, enum DM35424_Copy_Directions direction)

  Copy the reference adjustment data from volatile to non-volatile, or vice versa.

6.26.1 Detailed Description

Definitions for the DM35424 Reference Adjustment Library.

Id:

dm35424_ref_adjust_library.h 60276 2012-06-05 16:04:15Z rgroner

Definition in file dm35424_ref_adjust_library.h.

6.27 include/dm35424_registers.h File Reference

Defines for the DM35424 Registers (Offsets)

Macros

- #define DM35424_OFFSET_GBC_FORMAT 0x00

  Offset to General Board Control (BAR0) Format ID register.
- #define DM35424_OFFSET_GBC_REV 0x01

  Offset to General Board Control (BAR0) Format ID register.
- #define DM35424_OFFSET_GBC_END_INTERRUPT 0x02

  Offset to General Board Control (BAR0) EOI (End of Interrupt) register.
- #define DM35424_OFFSET_GBC_BOARD_RESET 0x03

  Offset to General Board Control (BAR0) Board Reset register.
- #define DM35424_OFFSET_GBC_PDP_NUMBER 0x04

  Offset to General Board Control (BAR0) PDP Number register.
- #define DM35424_OFFSET_GBC_FPGA_BUILD 0x08

  Offset to General Board Control (BAR0) FPGA Build register.
- #define DM35424_OFFSET_GBC_SYS_CLK_FREQ 0x0c

  Offset to General Board Control (BAR0) System Clock register.
- #define DM35424_OFFSET_GBC_IRQ_STATUS 0x10

  Offset to General Board Control (BAR0) IRQ Status register. Each bit corresponds to a function block.
- #define DM35424_OFFSET_GBC_DMA_IRQ_STATUS 0x18

  Offset to General Board Control (BAR0) DMA IRQ Status register. Each bit corresponds to a function block.
- #define DM35424_OFFSET_GBC_FB_START 0x20

  Offset to the beginning of the Function Blocks section of the GBC.
- #define DM35424_GBC_FB_BLK_SIZE 0x10

  Size of the function block entries in the GBC.
- #define DM35424_OFFSET_GBC_FB_ID 0x00

  Offset to Function Block ID, from the start of the function block section.
- #define DM35424_FB_ID_TYPE_MASK 0x0000FFFF

  Bit mask for TYPE portion of FB ID.
- #define DM35424_FB_ID_SUBTYPE_MASK 0x00FF0000

  Bit mask for SUBTYPE portion of FB ID.
- #define DM35424_FB_ID_TYPE_REV_MASK 0xFF000000

  Bit mask for TYPE REV portion of FB ID.
- #define DM35424_OFFSET_GBC_FB_OFFSET 0x04

  Offset to the FB Offset in the GBC, from the start of the FB data block.
- #define DM35424_OFFSET_GBC_FB_DMA_OFFSET 0x08

  Offset to the FB DMA Offset in the GBC, from the start of the FB data block.
- #define DM35424_OFFSET_DMA_ACTION 0x00

  Offset to the DMA Action Register (BAR2)
- #define DM35424_OFFSET_DMA_SETUP 0x01

  Offset to the DMA Setup Register (BAR2)
- #define DM35424_OFFSET_DMA_STAT_OVERFLOW 0x02

  Offset to the DMA Status (Overflow) Register (BAR2)
- #define DM35424_OFFSET_DMA_STAT_UNDERFLOW 0x03

  Offset to the DMA Status (Underflow) Register (BAR2)
- #define DM35424_OFFSET_DMA_CURRENT_COUNT 0x04

  Offset to the DMA Current Count Register (BAR2)
- #define DM35424_OFFSET_DMA_CURRENT_BUFFER 0x07

  Offset to the DMA Current Buffer Register (BAR2)
- #define DM35424_OFFSET_DMA_WR_FIFO_CNT 0x08

  Offset to the DMA Write FIFO Count Register (BAR2)
- #define DM35424_OFFSET_DMA_RD_FIFO_CNT 0x0A

Offset to the DMA Read FIFO Count Register (BAR2)

- #define DM35424_OFFSET_DMA_STAT_USED 0x0C

  Offset to the DMA Status (Used) Register (BAR2)
- #define DM35424_OFFSET_DMA_STAT_INVALID 0x0D

  Offset to the DMA Status (Invalid) Register (BAR2)
- #define DM35424_OFFSET_DMA_STAT_COMPLETE 0x0E

  Offset to the DMA Status (Complete) Register (BAR2)
- #define DM35424_OFFSET_DMA_LAST_ACTION 0x0F

  Offset to the DMA Last Action Register (BAR2)
- #define DM35424_OFFSET_DMA_BUFF_START 0x10

  Offset to the start of the buffer control section (BAR2)
- #define DM35424_OFFSET_DMA_BUFFER_STAT 0x02

  Offset to the buffer status register, from the start of the buffer control section (BAR2)
- #define DM35424_OFFSET_DMA_BUFFER_CTRL 0x03

  Offset to the buffer control register, from the start of the buffer control section (BAR2)
- #define DM35424_OFFSET_DMA_BUFFER_SIZE 0x04

  Offset to the buffer size register, from the start of the buffer control section (BAR2)
- #define DM35424_OFFSET_DMA_BUFFER_ADDRESS 0x08

  Offset to the buffer address register, from the start of the buffer control section (BAR2)
- #define DM35424_OFFSET_FB_DMA_CHANNELS 0x06

  Offset to the DMA Channels count of the function block (BAR2)
- #define DM35424_OFFSET_FB_DMA_BUFFERS 0x07

  Offset to the DMA buffers count of the function block (BAR2)
- #define DM35424_OFFSET_FB_CTRL_START 0x08

  Offset to the beginning of the Function Block control section in BAR2.
- #define DM35424_OFFSET_ADC_MODE_STATUS 0x00

  Offset to the ADC Mode-Status register, from the start of the ADC control section.
- #define DM35424_OFFSET_ADC_CLK_SRC 0x01

  Offset to the ADC Clock Source register, from the start of the ADC control section.
- #define DM35424_OFFSET_ADC_START_TRIG 0x02

  Offset to the ADC Start Trigger register, from the start of the ADC control section.
- #define DM35424_OFFSET_ADC_STOP_TRIG 0x03

  Offset to the ADC Stop Trigger register, from the start of the ADC control section.
- #define DM35424_OFFSET_ADC_CLK_DIV 0x04

  Offset to the ADC Clock Divider register, from the start of the ADC control section.
- #define DM35424_OFFSET_ADC_CLK_DIV_COUNTER 0x08

  Offset to the ADC Clock Divider Counter register, from the start of the ADC control section.
- #define DM35424_OFFSET_ADC_PRE_CAPT_COUNT 0x0c

  Offset to the ADC Pre-Start Capture Count register, from the start of the ADC control section.
- #define DM35424_OFFSET_ADC_POST_CAPT_COUNT 0x10

  Offset to the ADC Post-Stop Capture Count register, from the start of the ADC control section.
- #define DM35424_OFFSET_ADC_SAMPLE_COUNT 0x14

  Offset to the ADC Sample Count register, from the start of the ADC control section.
- #define DM35424_OFFSET_ADC_INT_ENABLE 0x18

  Offset to the ADC Interrupt Enable register, from the start of the ADC control section.
- #define DM35424_OFFSET_ADC_INT_STAT 0x1e

  Offset to the ADC Interrupt Status register, from the start of the ADC control section.
- #define DM35424_OFFSET_ADC_CLK_BUS2 0x22

  Offset to the ADC Clock Bus 2, from the start of the ADC control section.
- #define DM35424_OFFSET_ADC_CLK_BUS3 0x23

  Offset to the ADC Clock Bus 3 register, from the start of the ADC control section.

- #define DM35424_OFFSET_ADC_CLK_BUS4 0x24

  Offset to the ADC Clock Bus 4 register, from the start of the ADC control section.
- #define DM35424_OFFSET_ADC_CLK_BUS5 0x25

  Offset to the ADC Clock Bus 5 register, from the start of the ADC control section.
- #define DM35424_OFFSET_ADC_CLK_BUS6 0x26

  Offset to the ADC Clock Bus 6 register, from the start of the ADC control section.
- #define DM35424_OFFSET_ADC_CLK_BUS7 0x27

  Offset to the ADC Clock Bus 7 register, from the start of the ADC control section.
- #define DM35424_OFFSET_ADC_AD_CONFIG 0x28

  Offset to the ADC AD Config register, from the start of the ADC control section.
- #define DM35424_OFFSET_ADC_CHAN_CTRL_BLK_START 0x2c

  Offset to the start of the Channel Control Section, from the start of the ADC control section.
- #define DM35424_ADC_CHAN_CTRL_BLK_SIZE 0x18

  Constant size of ADC channel section in function block.
- #define DM35424_OFFSET_ADC_CHAN_FRONT_END_CONFIG 0x00

  Offset to the Channel Front End Config register, from the start of the ADC channel control section.
- #define DM35424_OFFSET_ADC_CHAN_DATA_COUNT 0x04

  Offset to the Channel FIFO Data count register, from the start of the ADC channel control section.
- #define DM35424_OFFSET_ADC_CHAN_FILTER 0x09

  Offset to the Channel Filter register, from the start of the ADC channel control section.
- #define DM35424_OFFSET_ADC_CHAN_INTR_STAT 0x0a

  Offset to the Channel Interrupt Status register, from the start of the ADC channel control section.
- #define DM35424_OFFSET_ADC_CHAN_INTR_ENABLE 0x0b

  Offset to the Channel Interrupt Enable register, from the start of the ADC channel control section.
- #define DM35424_OFFSET_ADC_CHAN_LOW_THRESHOLD 0x0c

  Offset to the Channel Low Threshold register, from the start of the ADC channel control section.
- #define DM35424_OFFSET_ADC_CHAN_HIGH_THRESHOLD 0x10

  Offset to the Channel High Threshold register, from the start of the ADC channel control section.
- #define DM35424_OFFSET_ADC_CHAN_LAST_SAMPLE 0x14

  Offset to the Channel Last Sample register, from the start of the ADC channel control section.
- #define DM35424_OFFSET_ADC_FIFO_CTRL_BLK_START 0x334

  Offset to the start of the FIFO Control Section, from the start of the ADC control section.
- #define DM35424_ADC_FIFO_CTRL_BLK_SIZE 0x4

  Constant size of ADC FIFO section in function block.
- #define DM35424_OFFSET_FB_ADC_FIFO 0x0334

  Offset to the FIFO for non-DMA read and write operations.
- #define DM35424_OFFSET_DAC_MODE_STATUS 0x00

  Offset to the Mode/Status register, from the start of the DAC control section.
- #define DM35424_OFFSET_DAC_CLK_SRC 0x01

  Offset to the Clock Source register, from the start of the DAC control section.
- #define DM35424_OFFSET_DAC_START_TRIG 0x02

  Offset to the Start Trigger register, from the start of the DAC control section.
- #define DM35424_OFFSET_DAC_STOP_TRIG 0x03

  Offset to the Stop Trigger register, from the start of the DAC control section.
- #define DM35424_OFFSET_DAC_CLK_DIV 0x04

  Offset to the Clock Divider register, from the start of the DAC control section.
- #define DM35424_OFFSET_DAC_CLK_DIV_COUNT 0x08

  Offset to the Clock Divider Counter register, from the start of the DAC control section.
- #define DM35424_OFFSET_DAC_POST_STOP_CONV 0x10

  Offset to the Post-Stop Conversion Count register, from the start of the DAC control section.
- #define DM35424_OFFSET_DAC_CONV_COUNT 0x14

Offset to the Conversion Count register, from the start of the DAC control section.

- #define DM35424_OFFSET_DAC_INT_ENABLE 0x18

  Offset to the Interrupt Enable register, from the start of the DAC control section.

- #define DM35424_OFFSET_DAC_INT_STAT 0x1e

  Offset to the Interrupt Status register, from the start of the DAC control section.

- #define DM35424_OFFSET_DAC_CLK_BUS2 0x22

  Offset to the Clock Bus 2 register, from the start of the DAC control section.

- #define DM35424_OFFSET_DAC_CLK_BUS3 0x23

  Offset to the Clock Bus 3 register, from the start of the DAC control section.

- #define DM35424_OFFSET_DAC_CLK_BUS4 0x24

  Offset to the Clock Bus 4 register, from the start of the DAC control section.

- #define DM35424_OFFSET_DAC_CLK_BUS5 0x25

  Offset to the Clock Bus 5 register, from the start of the DAC control section.

- #define DM35424_OFFSET_DAC_CLK_BUS6 0x26

  Offset to the Clock Bus 6 register, from the start of the DAC control section.

- #define DM35424_OFFSET_DAC_CLK_BUS7 0x27

  Offset to the Clock Bus 7 register, from the start of the DAC control section.

- #define DM35424_OFFSET_DAC_DA_CONFIG 0x28

  Offset to the DA Config register, from the start of the DAC control section.

- #define DM35424_OFFSET_DAC_CHAN_CTRL_BLK_START 0x2c

  Offset to the start of the DAC channel control section, from the start of the DAC control section.

- #define DM35424_DAC_CHAN_CTRL_BLK_SIZE 0x14

  Constant size of channel control section in function block.

- #define DM35424_OFFSET_DAC_CHAN_FRONT_END_CONFIG 0x00

  Offset to the Front-End Config register, from the start of the DAC channel control section.

- #define DM35424_OFFSET_DAC_CHAN_MARKER_STATUS 0x0a

  Offset to the Channel marker Interrupt Status register, from the start of the DAC channel control section.

- #define DM35424_OFFSET_DAC_CHAN_MARKER_ENABLE 0x0b

  Offset to the Channel marker Interrupt Enable register, from the start of the DAC channel control section.

- #define DM35424_OFFSET_DAC_CHAN_LAST_CONVERSION 0x10

  Offset to the Channel Last Conversion register, from the start of the DAC channel control section.

- #define DM35424_OFFSET_DAC_FIFO_CTRL_BLK_START 0x84

  Offset to the start of the DAC FIFO control section, from the start of the DAC control section.

- #define DM35424_OFFSET_DAC_FIFO_CTRL_BLK_SIZE 0x4

  Constant size of FIFO control section in function block.

- #define DM35424_OFFSET_DIO_INPUT_VAL 0x00

  Offset to the Input Value register, from the start of the DIO control section.

- #define DM35424_OFFSET_DIO_OUTPUT_VAL 0x04

  Offset to the Output Value register, from the start of the DIO control section.

- #define DM35424_OFFSET_DIO_DIRECTION 0x08

  Offset to the Direction register, from the start of the DIO control section.

- #define DM35424_OFFSET_TEMPERATURE 0x00

  Offset to the Temperature register, from the start of the Temperature control section.

- #define DM35424_OFFSET_REF_ADJUST_GO_BUSY 0x00

  Offset to the Go/Busy register, from the start of the Reference Adjustment control section.

- #define DM35424_OFFSET_REF_OUTPUT_LATCH 0x04

  Offset to the output latch register, from the start of the Reference Adjustment control section.

6.27.1 Detailed Description

Defines for the DM35424 Registers (Offsets)

Id:

dm35424_registers.h 124951 2020-03-05 16:34:24Z lfrankenfield

Definition in file dm35424_registers.h.

6.28 include/dm35424_temperature_library.h File Reference

Definitions for the DM35424 Temperature Library.

```
#include "dm35424_gbc_library.h"
```

Functions

- DM35424LIB_API int DM35424_Temperature_Open (struct DM35424_Board_Descriptor *handle, unsigned int ordinal_to_open, struct DM35424_Function_Block *fb_temp)

  Open the temperature function block, getting address values that will be used later by other library functions.
- DM35424LIB_API int DM35424_Temperature_Read (struct DM35424_Board_Descriptor *handle, struct DM35424_Function_Block *temp_fb, float *temperature)

  Read the temperature of the board, in Celsius degrees.

6.28.1 Detailed Description

Definitions for the DM35424 Temperature Library.

Id:

dm35424_temperature_library.h 60276 2012-06-05 16:04:15Z rgroner

Definition in file dm35424_temperature_library.h.

6.29 include/dm35424_types.h File Reference

Defines for the DM35424. Values for the general board, not specific to a particular function block.

Macros

- #define DM35424_SUBTYPE_00 0

  Constant for FB subtype 0.
- #define DM35424_SUBTYPE_01 1

  Constant for FB subtype 1.
- #define DM35424_SUBTYPE_02 2

  Constant for FB subtype 2.
- #define DM35424_SUBTYPE_03 3

  Constant for FB subtype 3.
- #define DM35424_SUBTYPE_INVALID 0xFF

  Constant value indicating an invalid subtype.

- #define DM35424_FUNC_BLOCK_INVALID 0x0000

  Constant value indicating an invalid function block.
- #define DM35424_FUNC_BLOCK_INVALID2 0xFFFF

  Constant value indicating an invalid function block.
- #define DM35424_FUNC_BLOCK_SYNCBUS 0x0001

  Function Block Constant for SyncBus.
- #define DM35424_FUNC_BLOCK_EXT_CLOCKING 0x0002

  Function Block Constant for Global Clocking.
- #define DM35424_FUNC_BLOCK_CLK0003 0x0003

  Function Block Constant for External Clocking (0003)
- #define DM35424_FUNC_BLOCK_CAPTWIN 0x0005

  Function Block Constant for Capture Window.
- #define DM35424_FUNC_BLOCK_ADC 0x1000

  Function Block Constant for ADC.
- #define DM35424_FUNC_BLOCK_ADC1001 0x1001

  Function Block Constant for 10 MHz ADC (1001)
- #define DM35424_FUNC_BLOCK_DAC 0x2000

  Function Block Constant for DAC.
- #define DM35424_FUNC_BLOCK_DAC2001 0x2001

  Function Block Constant for High Speed DAC (2001)
- #define DM35424_FUNC_BLOCK_DIO 0x3000

  Function Block Constant for DIO.
- #define DM35424_FUNC_BLOCK_ADIO 0x3001

  Function Block Constant for ADIO.
- #define DM35424_FUNC_BLOCK_ADIO3010 0x3010

  Function Block Constant for ADIO3010.
- #define DM35424_FUNC_BLOCK_USART 0x4000

  Function Block Constant for Synchronous/Asynchronous Serial Port.
- #define DM35424_FUNC_BLOCK_REF_ADJUST 0xF000

  Function Block Constant for Reference Adjustment.
- #define DM35424_FUNC_BLOCK_TEMPERATURE_SENSOR 0xF001

  Function Block Constant for Temperature Sensor.
- #define DM35424_FUNC_BLOCK_FLASH_PROGRAMMER 0xF002

  Function Block Constant for Flash Programmer.
- #define DM35424_FUNC_BLOCK_CLK_GEN 0xF003

  Function Block Constant for Clock Generator.
- #define DM35424_FUNC_BLOCK_DIN3011 0x3011

  Function Block Constant for Digital Input (3011)
- #define DM35424_FUNC_BLOCK_DOT3012 0x3012

  Function Block Constant for Digital Output (3012)
- #define DM35424_FUNC_BLOCK_INC3200 0x3200

  Function Block Constant for Incremental Encoder (3200)
- #define DM35424_FUNC_BLOCK_PWM3100 0x3100

  Function Block Constant for PWM (3100)
- #define DM35424_FUNC_BLOCK_CLK0004 0x0004

  Function Block Constant for Programmable Clock (0004)
- #define DM35424_MAX_FB 62

  Maximum possible number of function blocks on a board.
- #define MAX_DMA_BUFFERS 16

  Maximum possible number of DMA buffers for any function block.
- #define MAX_DMA_CHANNELS 32

Maximum possible number of DMA channels for any function block.

- #define DM35424_DMA_MAX_BUFFER_SIZE 0xFFFFFC

  Maximum possible DMA buffer size.

- #define DM35424_BOARD_ACK_INTERRUPT 0x1

  Value to write to the EOI register to acknowledge interrupts.

- #define DM35424_BOARD_RESET_VALUE 0xAA

  Value to write to the Reset register in order to reset the board.

- #define DM35424_FIFO_ACCESS_FB_REVISION 0x01

  Minimum function block revision that supports direct FIFO read/write access.

Enumerations

- enum DM35424_Clock_Sources {
  DM35424_CLK_SRC_IMMEDIATE, DM35424_CLK_SRC_NEVER, DM35424_CLK_SRC_BUS2, DM35424_CLK_SRC_BUS3,
  DM35424_CLK_SRC_BUS4, DM35424_CLK_SRC_BUS5, DM35424_CLK_SRC_BUS6, DM35424_CLK_SRC_BUS7,
  DM35424_CLK_SRC_CHAN_THRESH = 0x08, DM35424_CLK_SRC_CHAN_THRESH_INV = 0x09, DM35424_CLK_SRC_BUS2_INV = 0x0A, DM35424_CLK_SRC_BUS3_INV,
  DM35424_CLK_SRC_BUS4_INV, DM35424_CLK_SRC_BUS5_INV, DM35424_CLK_SRC_BUS6_INV, DM35424_CLK_SRC_BUS7_INV }

  Possible clock sources used by function blocks. Note that some clock sources may not be available on your particular board. Check the hardware manual to verify which clock sources can be used.

- enum DM35424_Clock_Buses {
  DM35424_CLK_BUS2 = 2, DM35424_CLK_BUS3, DM35424_CLK_BUS4, DM35424_CLK_BUS5,
  DM35424_CLK_BUS6, DM35424_CLK_BUS7 }

  Clock buses available to the function block.

6.29.1 Detailed Description

Defines for the DM35424. Values for the general board, not specific to a particular function block.

Id:

dm35424_types.h 127189 2020-09-16 13:22:33Z lfrankenfield

Definition in file dm35424_types.h.

6.29.2 Enumeration Type Documentation

6.29.2.1 enum DM35424_Clock_Buses

Clock buses available to the function block.

Enumerator

DM35424_CLK_BUS2 Clock Bus 2.

DM35424_CLK_BUS3 Clock Bus 3.

DM35424_CLK_BUS4 Clock Bus 4.

DM35424_CLK_BUS5 Clock Bus 5.

DM35424_CLK_BUS6 Clock Bus 6.

DM35424_CLK_BUS7 Clock Bus 7.

Definition at line 346 of file dm35424_types.h.

6.29.2.2 enum DM35424_Clock_Sources

Possible clock sources used by function blocks. Note that some clock sources may not be available on your particular board. Check the hardware manual to verify which clock sources can be used.

DM35424_General_Definitions

Enumerator

DM35424_CLK_SRC_IMMEDIATE Clock Source - Immediate (0x00)

DM35424_CLK_SRC_NEVER Clock Source - Never (0x01)

DM35424_CLK_SRC_BUS2 Clock Source - Bus 2 (0x02)

DM35424_CLK_SRC_BUS3 Clock Source - Bus 3 (0x03)

DM35424_CLK_SRC_BUS4 Clock Source - Bus 4 (0x04)

DM35424_CLK_SRC_BUS5 Clock Source - Bus 5 (0x05)

DM35424_CLK_SRC_BUS6 Clock Source - Bus 6 (0x06)

DM35424_CLK_SRC_BUS7 Clock Source - Bus 7 (0x07)

DM35424_CLK_SRC_CHAN_THRESH Clock Source - Threshold Exceeded (0x08)

DM35424_CLK_SRC_CHAN_THRESH_INV Clock Source - Threshold Inverse (None Exceeded) (0x09)

DM35424_CLK_SRC_BUS2_INV Clock Source - Bus 2 Inverse (0x0A)

DM35424_CLK_SRC_BUS3_INV Clock Source - Bus 3 Inverse (0x0B)

DM35424_CLK_SRC_BUS4_INV Clock Source - Bus 4 Inverse (0x0C)

DM35424_CLK_SRC_BUS5_INV Clock Source - Bus 5 Inverse (0x0D)

DM35424_CLK_SRC_BUS6_INV Clock Source - Bus 6 Inverse (0x0E)

DM35424_CLK_SRC_BUS7_INV Clock Source - Bus 7 Inverse (0x0F)

Definition at line 258 of file dm35424_types.h.

6.30 include/dm35424_util_library.h File Reference

Definitions for the DM35424 Utilities library, various helper functions.

```
#include <time.h>
#include <sys/time.h>
```

Enumerations

- enum DM35424_Waveforms { DM35424_SINE_WAVE, DM35424_SQUARE_WAVE, DM35424_SAWTOOTH_WAVE }

  List of possible waveforms that can be generated for DAC purposes.

Functions

- uint32_t DM35424_Get_Maskable (uint16_t data, uint16_t mask)

  Return a 32-bit maskable register value from the data and mask.
- void DM35424_Micro_Sleep (unsigned long microsecs)

  Sleep for a specified number of microseconds.
- long DM35424_Get_Time_Diff (struct timeval last, struct timeval first)

  Calculate the time difference between the two timeval structs, in microseconds.

- int DM35424_Generate_Signal_Data (enum DM35424_Waveforms waveform, int32_t *data, uint32_t data_count, int32_t max, int32_t minimum, int32_t offset, uint32_t mask)

  Generate data with a specific wave pattern. This is useful for producing recognizeable waves for DAC output.
- void check_result (int return_val, char *message)

  Check the result of an operation, usually a library call. If the result is non-zero, then it is an error and output the passed message.

6.30.1 Detailed Description

Definitions for the DM35424 Utilities library, various helper functions.

Id:

dm35424_util_library.h 114375 2018-06-20 05:58:38Z prucz

Definition in file dm35424_util_library.h.

Index