Linux Software (DM35425)

Generated by Doxygen 1.8.17

1 Module Index **1**

1.1 Modules 1

2 Data Structure Index **3**

2.1 Data Structures 3

3 File Index **5**

3.1 File List 5

4 Module Documentation **7**

4.1 DM35425 ADC Library Constants 7

4.1.1 Detailed Description 10

4.1.2 Enumeration Type Documentation 10

4.1.2.1 DM35425_Adc_Clock_Events 10

4.1.2.2 DM35425_Channel_Delay 11

4.1.2.3 DM35425_Gains 11

4.1.2.4 DM35425_Input_Mode 12

4.1.2.5 DM35425_Input_Ranges 12

4.2 DM35425 ADC Public Library Functions 14

4.2.1 Detailed Description 16

4.2.2 Function Documentation 16

4.2.2.1 DM35425_Adc_Channel_Find_Interrupt() 17

4.2.2.2 DM35425_Adc_Channel_Get_Filter() 17

4.2.2.3 DM35425_Adc_Channel_Get_Front_End_Config() 18

4.2.2.4 DM35425_Adc_Channel_Get_Last_Sample() 18

4.2.2.5 DM35425_Adc_Channel_Get_Thresholds() 19

4.2.2.6 DM35425_Adc_Channel_Interrupt_Clear_Status() 20

4.2.2.7 DM35425_Adc_Channel_Interrupt_Get_Config() 20

4.2.2.8 DM35425_Adc_Channel_Interrupt_Get_Status() 21

4.2.2.9 DM35425_Adc_Channel_Interrupt_Set_Config() 21

4.2.2.10 DM35425_Adc_Channel_Reset() 22

4.2.2.11 DM35425_Adc_Channel_Set_Filter() 22

4.2.2.12 DM35425_Adc_Channel_Set_High_Threshold() 23

4.2.2.13 DM35425_Adc_Channel_Set_Low_Threshold() 24

4.2.2.14 DM35425_Adc_Channel_Setup() 24

4.2.2.15 DM35425_Adc_Fifo_Channel_Read() 25

4.2.2.16 DM35425_Adc_Get_Clock_Source_Global() 26

4.2.2.17 DM35425_Adc_Get_Clock_Src() 26

4.2.2.18 DM35425_Adc_Get_Mode_Status() 27

4.2.2.19 DM35425_Adc_Get_Post_Stop_Samples() 27

4.2.2.20 DM35425_Adc_Get_Pre_Trigger_Samples() 28

4.2.2.21 DM35425_Adc_Get_Sample_Count() 28

4.2.2.22 DM35425_Adc_Get_Start_Trigger() 29

4.2.2.23 DM35425_Adc_Get_Stop_Trigger() 29
4.2.2.24 DM35425_Adc_Initialize() 30
4.2.2.25 DM35425_Adc_Interrupt_Clear_Status() 31
4.2.2.26 DM35425_Adc_Interrupt_Get_Config() 31
4.2.2.27 DM35425_Adc_Interrupt_Get_Status() 32
4.2.2.28 DM35425_Adc_Interrupt_Set_Config() 32
4.2.2.29 DM35425_Adc_Open() 33
4.2.2.30 DM35425_Adc_Pause() 33
4.2.2.31 DM35425_Adc_Reset() 34
4.2.2.32 DM35425_Adc_Sample_To_Volts() 34
4.2.2.33 DM35425_Adc_Set_Clk_Divider() 35
4.2.2.34 DM35425_Adc_Set_Clock_Source_Global() 35
4.2.2.35 DM35425_Adc_Set_Clock_Src() 36
4.2.2.36 DM35425_Adc_Set_Post_Stop_Samples() 36
4.2.2.37 DM35425_Adc_Set_Pre_Trigger_Samples() 37
4.2.2.38 DM35425_Adc_Set_Sample_Rate() 38
4.2.2.39 DM35425_Adc_Set_Start_Trigger() 38
4.2.2.40 DM35425_Adc_Set_Stop_Trigger() 39
4.2.2.41 DM35425_Adc_Start() 39
4.2.2.42 DM35425_Adc_Start_Rearm() 40
4.2.2.43 DM35425_Adc_Uninitialize() 40
4.2.2.44 DM35425_Adc_Volts_To_Sample() 41
4.3 DM35425 ADIO Public Library Enums 42
4.3.1 Detailed Description 42
4.3.2 Enumeration Type Documentation 42
4.3.2.1 DM35425_Adv_Interrupt_Mode 42
4.4 DM35425 ADIO Public Library Functions 43
4.4.1 Detailed Description 45
4.4.2 Function Documentation 45
4.4.2.1 DM35425_Adio_Fifo_Channel_Read() 45
4.4.2.2 DM35425_Adio_Fifo_Channel_Write() 46
4.4.2.3 DM35425_Adio_Get_Adv_Int_Capt() 46
4.4.2.4 DM35425_Adio_Get_Adv_Int_Comp() 47
4.4.2.5 DM35425_Adio_Get_Adv_Int_Mask() 47
4.4.2.6 DM35425_Adio_Get_Adv_Int_Mode() 48
4.4.2.7 DM35425_Adio_Get_Clk_Div_Counter() 48
4.4.2.8 DM35425_Adio_Get_Clock_Source_Global() 48
4.4.2.9 DM35425_Adio_Get_Direction() 49
4.4.2.10 DM35425_Adio_Get_Input_Value() 50
4.4.2.11 DM35425_Adio_Get_Mode_Status() 50
4.4.2.12 DM35425_Adio_Get_Output_Value() 51
4.4.2.13 DM35425_Adio_Get_P_Bus_Enable() 51

4.4.2.14 DM35425_Adio_Get_P_Bus_Ready_Enable() 52
4.4.2.15 DM35425_Adio_Get_Sample_Count() 52
4.4.2.16 DM35425_Adio_Interrupt_Clear_Status() 53
4.4.2.17 DM35425_Adio_Interrupt_Get_Config() 53
4.4.2.18 DM35425_Adio_Interrupt_Get_Status() 54
4.4.2.19 DM35425_Adio_Interrupt_Set_Config() 54
4.4.2.20 DM35425_Adio_Open() 55
4.4.2.21 DM35425_Adio_Pause() 55
4.4.2.22 DM35425_Adio_Reset() 56
4.4.2.23 DM35425_Adio_Set_Adv_Int_Capt() 56
4.4.2.24 DM35425_Adio_Set_Adv_Int_Comp() 56
4.4.2.25 DM35425_Adio_Set_Adv_Int_Mask() 57
4.4.2.26 DM35425_Adio_Set_Adv_Int_Mode() 57
4.4.2.27 DM35425_Adio_Set_Clk_Divider() 58
4.4.2.28 DM35425_Adio_Set_Clock_Source_Global() 58
4.4.2.29 DM35425_Adio_Set_Clock_Src() 59
4.4.2.30 DM35425_Adio_Set_Direction() 59
4.4.2.31 DM35425_Adio_Set_Output_Value() 60
4.4.2.32 DM35425_Adio_Set_P_Bus_Enable() 61
4.4.2.33 DM35425_Adio_Set_P_Bus_Ready_Enable() 61
4.4.2.34 DM35425_Adio_Set_Pacer_Clk_Rate() 62
4.4.2.35 DM35425_Adio_Set_Post_Stop_Samples() 62
4.4.2.36 DM35425_Adio_Set_Pre_Trigger_Samples() 63
4.4.2.37 DM35425_Adio_Set_Start_Trigger() 63
4.4.2.38 DM35425_Adio_Set_Stop_Trigger() 64
4.4.2.39 DM35425_Adio_Start() 65
4.4.2.40 DM35425_Adio_Start_Rearm() 65
4.4.2.41 DM35425_Adio_Uninitialize() 65
4.5 DM35425 Board Access Structures 67
4.5.1 Detailed Description 67
4.5.2 Function Documentation 67
4.5.2.1 DM35425_Board_Close() 67
4.5.2.2 DM35425_Board_Open() 68
4.5.2.3 DM35425_Dma() 68
4.5.2.4 DM35425_Modify() 69
4.5.2.5 DM35425_Read() 69
4.5.2.6 DM35425_Write() 70
4.6 DM35425 PCI Region Structures 71
4.6.1 Detailed Description 71
4.6.2 Enumeration Type Documentation 71
4.6.2.1 DM35425_DMA_FUNCTIONS 71
4.6.2.2 dm35425_pci_region_access_size 72

4.6.2.3 dm35425_pci_region_num 72
4.7 DM35425 DAC Library Constants 73
4.7.1 Detailed Description 74
4.7.2 Enumeration Type Documentation 74
4.7.2.1 DM35425_Dac_Clock_Events 74
4.7.2.2 DM35425_Output_Ranges 75
4.8 DM35425 DAC Library Public Functions 76
4.8.1 Detailed Description 78
4.8.2 Function Documentation 78
4.8.2.1 DM35425_Dac_Channel_Clear_Marker_Status() 78
4.8.2.2 DM35425_Dac_Channel_Get_Marker_Config() 78
4.8.2.3 DM35425_Dac_Channel_Get_Marker_Status() 79
4.8.2.4 DM35425_Dac_Channel_Reset() 79
4.8.2.5 DM35425_Dac_Channel_Set_Marker_Config() 80
4.8.2.6 DM35425_Dac_Channel_Setup() 80
4.8.2.7 DM35425_Dac_Conv_To_Volts() 81
4.8.2.8 DM35425_Dac_Fifo_Channel_Write() 81
4.8.2.9 DM35425_Dac_Get_Clock_Div() 82
4.8.2.10 DM35425_Dac_Get_Clock_Src() 82
4.8.2.11 DM35425_Dac_Get_Conversion_Count() 83
4.8.2.12 DM35425_Dac_Get_Last_Conversion() 83
4.8.2.13 DM35425_Dac_Get_Mode_Status() 84
4.8.2.14 DM35425_Dac_Get_Post_Stop_Conversion_Count() 84
4.8.2.15 DM35425_Dac_Get_Start_Trigger() 85
4.8.2.16 DM35425_Dac_Get_Stop_Trigger() 85
4.8.2.17 DM35425_Dac_Interrupt_Clear_Status() 86
4.8.2.18 DM35425_Dac_Interrupt_Get_Config() 86
4.8.2.19 DM35425_Dac_Interrupt_Get_Status() 87
4.8.2.20 DM35425_Dac_Interrupt_Set_Config() 87
4.8.2.21 DM35425_Dac_Open() 88
4.8.2.22 DM35425_Dac_Pause() 88
4.8.2.23 DM35425_Dac_Reset() 89
4.8.2.24 DM35425_Dac_Set_Clock_Div() 89
4.8.2.25 DM35425_Dac_Set_Clock_Source_Global() 90
4.8.2.26 DM35425_Dac_Set_Clock_Src() 90
4.8.2.27 DM35425_Dac_Set_Conversion_Rate() 91
4.8.2.28 DM35425_Dac_Set_Last_Conversion() 91
4.8.2.29 DM35425_Dac_Set_Post_Stop_Conversion_Count() 92
4.8.2.30 DM35425_Dac_Set_Start_Trigger() 93
4.8.2.31 DM35425_Dac_Set_Stop_Trigger() 93
4.8.2.32 DM35425_Dac_Start() 94
4.8.2.33 DM35425_Dac_Volts_To_Conv() 94

4.9 DM35425 DMA Public Library Constants 95
4.9.1 Detailed Description 96
4.9.2 Enumeration Type Documentation 96
4.9.2.1 DM35425_Fifo_States 96
4.10 DM35425 DMA Public Library Functions 97
4.10.1 Detailed Description 98
4.10.2 Function Documentation 98
4.10.2.1 DM35425_Dma_Buffer_Setup() 98
4.10.2.2 DM35425_Dma_Buffer_Status() 99
4.10.2.3 DM35425_Dma_Check_Buffer_Used() 99
4.10.2.4 DM35425_Dma_Check_For_Error() 100
4.10.2.5 DM35425_Dma_Clear() 101
4.10.2.6 DM35425_Dma_Clear_Interrupt() 101
4.10.2.7 DM35425_Dma_Configure_Interrupts() 102
4.10.2.8 DM35425_Dma_Find_Interrupt() 103
4.10.2.9 DM35425_Dma_Get_Current_Buffer_Count() 104
4.10.2.10 DM35425_Dma_Get_Errors() 104
4.10.2.11 DM35425_Dma_Get_Fifo_Counts() 105
4.10.2.12 DM35425_Dma_Get_Fifo_State() 105
4.10.2.13 DM35425_Dma_Get_Interrupt_Configuration() 106
4.10.2.14 DM35425_Dma_Pause() 107
4.10.2.15 DM35425_Dma_Reset_Buffer() 107
4.10.2.16 DM35425_Dma_Setup() 108
4.10.2.17 DM35425_Dma_Setup_Set_Direction() 109
4.10.2.18 DM35425_Dma_Setup_Set_Used() 109
4.10.2.19 DM35425_Dma_Start() 110
4.10.2.20 DM35425_Dma_Status() 110
4.10.2.21 DM35425_Dma_Stop() 111
4.11 DM35425 Driver Constants 113
4.11.1 Detailed Description 113
4.12 DM35425 Driver Enumerations 114
4.12.1 Detailed Description 114
4.12.2 Enumeration Type Documentation 114
4.12.2.1 dm35425_pci_region_access_dir 114
4.13 DM35425 Driver Structures 115
4.13.1 Detailed Description 115
4.14 DM35425 Example Programs Constants 116
4.14.1 Detailed Description 118
4.14.2 Enumeration Type Documentation 118
4.14.2.1 Help_Options 118
4.15 DM35425 Global Clocking Library Constants 120
4.15.1 Detailed Description 120

4.16 DM35425 Global Clocking Library Public Functions 121
4.16.1 Detailed Description 121
4.16.2 Function Documentation 121
4.16.2.1 DM35425_Ext_Clocking_Get_Dir() 122
4.16.2.2 DM35425_Ext_Clocking_Get_Edge() 122
4.16.2.3 DM35425_Ext_Clocking_Get_Gate_In() 122
4.16.2.4 DM35425_Ext_Clocking_Get_In() 123
4.16.2.5 DM35425_Ext_Clocking_Get_Method() 123
4.16.2.6 DM35425_Ext_Clocking_Get_Pulse_Width() 124
4.16.2.7 DM35425_Ext_Clocking_Open() 124
4.16.2.8 DM35425_Ext_Clocking_Set_Dir() 125
4.16.2.9 DM35425_Ext_Clocking_Set_Edge() 125
4.16.2.10 DM35425_Ext_Clocking_Set_Method() 126
4.16.2.11 DM35425_Ext_Clocking_Set_Pulse_Width() 126
4.17 DM35425 Board Macros 128
4.17.1 Detailed Description 128
4.17.2 Macro Definition Documentation 128
4.17.2.1 CLK_40MHZ 128
4.18 DM35425 Board Library Public Functions 129
4.18.1 Detailed Description 129
4.18.2 Function Documentation 129
4.18.2.1 DM35425_Function_Block_Open() 129
4.18.2.2 DM35425_Function_Block_Open_Module() 130
4.18.2.3 DM35425_Gbc_Ack_Interrupt() 130
4.18.2.4 DM35425_Gbc_Board_Reset() 132
4.18.2.5 DM35425_Gbc_Get_Format() 132
4.18.2.6 DM35425_Gbc_Get_Fpga_Build() 133
4.18.2.7 DM35425_Gbc_Get_Pdp_Number() 133
4.18.2.8 DM35425_Gbc_Get_Revision() 134
4.18.2.9 DM35425_Gbc_Get_Sys_Clock_Freq() 134
4.19 DM35425 Ioctl macros 135
4.19.1 Detailed Description 135
4.20 DM35425 Board Access Public Library Functions 136
4.20.1 Detailed Description 136
4.20.2 Function Documentation 136
4.20.2.1 DM35425_Dma_Initialize() 136
4.20.2.2 DM35425_Dma_Read() 137
4.20.2.3 DM35425_Dma_Write() 138
4.20.2.4 DM35425_General_InstallISR() 138
4.20.2.5 DM35425_General_RemoveISR() 139
4.20.2.6 DM35425_General_SetISRPriority() 139
4.20.2.7 DM35425_General_WaitForInterrupt() 140

4.21 DM35425 Register Offsets 141
4.21.1 Detailed Description 146
4.21.2 Macro Definition Documentation 146
4.21.2.1 DM35425_OFFSET_DAC_STOP_TRIG 146
4.21.2.2 DM35425_OFFSET_FB_ADC_FIFO 146
4.22 DM35425 Board Types 147
4.22.1 Detailed Description 148
4.23 DM35425 Utility Library Functions 149
4.23.1 Detailed Description 149
4.23.2 Enumeration Type Documentation 149
4.23.2.1 DM35425_Waveforms 149
4.23.3 Function Documentation 149
4.23.3.1 check_result() 150
4.23.3.2 DM35425_Generate_Signal_Data() 150
4.23.3.3 DM35425_Get_Maskable() 151
4.23.3.4 DM35425_Get_Time_Diff() 151
4.23.3.5 DM35425_Micro_Sleep() 152

5 Data Structure Documentation **153**
5.1 DM35425_Board_Descriptor Struct Reference 153
5.1.1 Detailed Description 153
5.1.2 Field Documentation 153
5.1.2.1 file_descriptor 153
5.1.2.2 isr 154
5.1.2.3 pid 154
5.2 dm35425_device_descriptor Struct Reference 154
5.2.1 Detailed Description 154
5.2.2 Field Documentation 155
5.2.2.1 device_lock 155
5.2.2.2 dma_descr_list 155
5.2.2.3 dma_wait_queue 155
5.2.2.4 int_queue_count 155
5.2.2.5 int_queue_in_marker 155
5.2.2.6 int_queue_missed 156
5.2.2.7 int_queue_out_marker 156
5.2.2.8 int_wait_queue 156
5.2.2.9 interrupt_fb 156
5.2.2.10 irq_number 156
5.2.2.11 name 157
5.2.2.12 pci 157
5.2.2.13 reference_count 157
5.2.2.14 remove_isr_flag 157

5.3 DM35425_DMA_Descriptor Struct Reference 157
5.3.1 Detailed Description 158
5.3.2 Field Documentation 158
5.3.2.1 buffer_start_offset 158
5.3.2.2 control_offset 158
5.3.2.3 num_buffers 158
5.4 dm35425_dma_descriptor Struct Reference 159
5.4.1 Detailed Description 159
5.4.2 Field Documentation 159
5.4.2.1 buffer 159
5.4.2.2 buffer_size 159
5.4.2.3 bus_addr 160
5.4.2.4 channel 160
5.4.2.5 fb_num 160
5.4.2.6 list 160
5.4.2.7 virt_addr 160
5.5 DM35425_Function_Block Struct Reference 161
5.5.1 Detailed Description 161
5.5.2 Field Documentation 161
5.5.2.1 control_offset 161
5.5.2.2 dma_channel 161
5.5.2.3 dma_offset 162
5.5.2.4 fb_num 162
5.5.2.5 fb_offset 162
5.5.2.6 num_dma_buffers 162
5.5.2.7 num_dma_channels 162
5.5.2.8 ordinal_fb_type_num 163
5.5.2.9 sub_type 163
5.5.2.10 type 163
5.5.2.11 type_revision 163
5.6 dm35425_ioctl_argument Union Reference 163
5.6.1 Detailed Description 164
5.6.2 Field Documentation 164
5.6.2.1 dma 164
5.6.2.2 interrupt 164
5.6.2.3 modify 164
5.6.2.4 readwrite 165
5.7 dm35425_ioctl_dma Struct Reference 165
5.7.1 Detailed Description 165
5.7.2 Field Documentation 165
5.7.2.1 buffer 165
5.7.2.2 buffer_ptr 166

5.7.2.3 buffer_size 166
5.7.2.4 channel 166
5.7.2.5 fb_num 166
5.7.2.6 function 166
5.7.2.7 num_buffers 167
5.7.2.8 pci 167
5.8 dm35425_ioctl_interrupt_info_request Struct Reference 167
5.8.1 Detailed Description 167
5.8.2 Field Documentation 167
5.8.2.1 error_occurred 168
5.8.2.2 interrupt_fb 168
5.8.2.3 interrupts_remaining 168
5.8.2.4 valid_interrupt 168
5.9 dm35425_ioctl_region_modify Struct Reference 169
5.9.1 Detailed Description 169
5.9.2 Field Documentation 169
5.9.2.1 access 169
5.9.2.2 mask 169
5.9.2.3 mask16 170
5.9.2.4 mask32 170
5.9.2.5 mask8 170
5.10 dm35425_ioctl_region_readwrite Struct Reference 170
5.10.1 Detailed Description 170
5.10.2 Field Documentation 171
5.10.2.1 access 171
5.11 dm35425_pci_access_request Struct Reference 171
5.11.1 Detailed Description 171
5.11.2 Field Documentation 171
5.11.2.1 data 172
5.11.2.2 data16 172
5.11.2.3 data32 172
5.11.2.4 data8 172
5.11.2.5 offset 172
5.11.2.6 region 173
5.11.2.7 size 173
5.12 dm35425_pci_region Struct Reference 173
5.12.1 Detailed Description 173
5.12.2 Field Documentation 173
5.12.2.1 allocated 174
5.12.2.2 io_addr 174
5.12.2.3 length 174
5.12.2.4 phys_addr 174

5.12.2.5 virt_addr 174

6 File Documentation 175

6.1 examples/_non_public/dm35425_dac_fifo.c File Reference 175

6.1.1 Detailed Description 176

6.1.2 Macro Definition Documentation 176

6.1.2.1 DEFAULT_CHANNEL 177

6.1.2.2 DEFAULT_DAC_RATE 177

6.1.2.3 DEFAULT_RANGE 177

6.1.3 Function Documentation 177

6.1.3.1 main() 177

6.1.3.2 print_fifo_status() 178

6.1.3.3 sigint_handler() 178

6.1.4 Variable Documentation 179

6.1.4.1 exit_program 179

6.1.4.2 program_name 179

6.2 examples/dm35425_adc.c File Reference 180

6.2.1 Detailed Description 181

6.2.2 Macro Definition Documentation 181

6.2.2.1 DEFAULT_CHANNEL 181

6.2.2.2 DEFAULT_MODE 181

6.2.2.3 DEFAULT_RANGE 182

6.2.2.4 DEFAULT_RATE 182

6.2.3 Function Documentation 182

6.2.3.1 DM35425_Setup_Dacs() 182

6.2.3.2 main() 182

6.2.3.3 sigint_handler() 183

6.2.4 Variable Documentation 184

6.2.4.1 exit_program 184

6.2.4.2 interrupt_count 184

6.2.4.3 program_name 184

6.3 examples/dm35425_adc_all_dma.c File Reference 184

6.3.1 Detailed Description 185

6.3.2 Macro Definition Documentation 186

6.3.2.1 DAT_FILE_NAME_PREFIX 186

6.3.2.2 DAT_FILE_NAME_SUFFIX 186

6.3.2.3 DEFAULT_RANGE 186

6.3.2.4 DEFAULT_RATE 186

6.3.3 Function Documentation 186

6.3.3.1 ISR() 186

6.3.3.2 main() 187

6.3.3.3 output_channel_status() 188

6.3.3.4 sigint_handler() 188
6.3.4 Variable Documentation 189
6.3.4.1 board 189
6.3.4.2 buffer_count 189
6.3.4.3 buffer_size_bytes 189
6.3.4.4 dma_has_error 190
6.3.4.5 exit_program 190
6.3.4.6 local_buffer 190
6.3.4.7 my_adc 190
6.3.4.8 next_buffer 191
6.3.4.9 program_name 191
6.4 examples/dm35425_adc_continuous_dma.c File Reference 191
6.4.1 Detailed Description 193
6.4.2 Macro Definition Documentation 193
6.4.2.1 ASCII_FILE_NAME 193
6.4.2.2 BIN_FILE_NAME 193
6.4.2.3 DAC_BUFFER_SIZE_BYTES 194
6.4.2.4 DAC_BUFFER_SIZE_SAMPLES 194
6.4.2.5 DAC_RATE 194
6.4.2.6 DEFAULT_CHANNEL 194
6.4.2.7 DEFAULT_RANGE 194
6.4.2.8 DEFAULT_RATE 195
6.4.3 Function Documentation 195
6.4.3.1 convert_bin_to_txt() 195
6.4.3.2 ISR() 195
6.4.3.3 main() 196
6.4.3.4 output_channel_status() 197
6.4.3.5 setup_dacs() 197
6.4.3.6 sigint_handler() 198
6.4.4 Variable Documentation 198
6.4.4.1 board 198
6.4.4.2 buffer_count 198
6.4.4.3 buffer_size_bytes 199
6.4.4.4 channel 199
6.4.4.5 dma_has_error 199
6.4.4.6 exit_program 199
6.4.4.7 local_buffer 200
6.4.4.8 my_adc 200
6.4.4.9 next_buffer 200
6.4.4.10 program_name 200
6.5 examples/dm35425_adc_fifo.c File Reference 201
6.5.1 Detailed Description 202

6.5.2 Macro Definition Documentation 202
6.5.2.1 DEFAULT_CHANNEL 202
6.5.2.2 DEFAULT_MODE 202
6.5.2.3 DEFAULT_RANGE 203
6.5.2.4 DEFAULT_RATE 203
6.5.3 Function Documentation 203
6.5.3.1 main() 203
6.5.3.2 sigint_handler() 204
6.5.4 Variable Documentation 204
6.5.4.1 dma_interrupt_count 204
6.5.4.2 exit_program 205
6.5.4.3 interrupt_count 205
6.5.4.4 my_adc 205
6.5.4.5 non_dma_interrupt_count 205
6.5.4.6 program_name 206
6.5.4.7 unexpected_interrupt_count 206
6.6 examples/dm35425_adio.c File Reference 206
6.6.1 Detailed Description 207
6.6.2 Macro Definition Documentation 207
6.6.2.1 DM35425_ADIO_DIRECTION 207
6.6.3 Function Documentation 208
6.6.3.1 main() 208
6.6.4 Variable Documentation 208
6.6.4.1 board 208
6.6.4.2 my_adio 209
6.6.4.3 program_name 209
6.7 examples/dm35425_adio_adv_int.c File Reference 209
6.7.1 Detailed Description 210
6.7.2 Macro Definition Documentation 211
6.7.2.1 DM35425_ADIO_DIRECTION 211
6.7.2.2 DM35425_ADIO_MATCH1 211
6.7.2.3 DM35425_ADIO_MATCH2 211
6.7.3 Function Documentation 211
6.7.3.1 ISR() 211
6.7.3.2 main() 212
6.7.3.3 sigint_handler() 212
6.7.4 Variable Documentation 213
6.7.4.1 board 213
6.7.4.2 exit_program 213
6.7.4.3 my_adio 213
6.7.4.4 program_name 213
6.7.4.5 total_interrupt_count 214

6.8 examples/dm35425_adio_dma.c File Reference . . . 214
6.8.1 Detailed Description . . . 215
6.8.2 Macro Definition Documentation . . . 215
6.8.2.1 ADIO_RATE . . . 216
6.8.2.2 BUFFER_SIZE_BYTES . . . 216
6.8.2.3 BUFFER_SIZE_SAMPLES . . . 216
6.8.2.4 DM35425_ADIO_DIRECTION1 . . . 216
6.8.2.5 DM35425_ADIO_DIRECTION2 . . . 216
6.8.3 Function Documentation . . . 216
6.8.3.1 ISR() . . . 216
6.8.3.2 main() . . . 217
6.8.3.3 output_channel_status() . . . 218
6.8.3.4 output_dma_buffer_status() . . . 218
6.8.3.5 sigint_handler() . . . 219
6.8.4 Variable Documentation . . . 219
6.8.4.1 board . . . 219
6.8.4.2 buffer_copied . . . 220
6.8.4.3 dma_has_error . . . 220
6.8.4.4 exit_program . . . 220
6.8.4.5 my_adio . . . 220
6.8.4.6 program_name . . . 221
6.9 examples/dm35425_adio_parallel_bus.c File Reference . . . 221
6.9.1 Detailed Description . . . 222
6.9.2 Macro Definition Documentation . . . 222
6.9.2.1 BUFFER_SIZE_BYTES . . . 223
6.9.2.2 BUFFER_SIZE_SAMPLES . . . 223
6.9.2.3 DEFAULT_RATE . . . 223
6.9.2.4 DM35425_ADIO_IN_DIRECTION . . . 223
6.9.2.5 DM35425_ADIO_OUT_DIRECTION . . . 223
6.9.3 Function Documentation . . . 223
6.9.3.1 ISR() . . . 223
6.9.3.2 main() . . . 224
6.9.3.3 output_channel_status() . . . 225
6.9.3.4 sigint_handler() . . . 225
6.9.4 Variable Documentation . . . 226
6.9.4.1 board . . . 226
6.9.4.2 exit_program . . . 226
6.9.4.3 interrupt_count . . . 226
6.9.4.4 is_receiver . . . 227
6.9.4.5 is_sender . . . 227
6.9.4.6 my_adio . . . 227
6.9.4.7 program_name . . . 227

6.10 examples/dm35425_dac.c File Reference 228
6.10.1 Detailed Description 228
6.10.2 Macro Definition Documentation 229
6.10.2.1 DEFAULT_CHANNEL 229
6.10.2.2 DEFAULT_RANGE 229
6.10.3 Function Documentation 229
6.10.3.1 main() 229
6.10.4 Variable Documentation 230
6.10.4.1 program_name 230
6.11 examples/dm35425_dac_dma.c File Reference 230
6.11.1 Detailed Description 231
6.11.2 Macro Definition Documentation 232
6.11.2.1 BUFFER_SIZE_BYTES 232
6.11.2.2 BUFFER_SIZE_SAMPLES 232
6.11.2.3 DEFAULT_CHANNEL 232
6.11.2.4 DEFAULT_RANGE 233
6.11.2.5 DEFAULT_RATE 233
6.11.2.6 NUM_BUFFERS_TO_USE 233
6.11.3 Function Documentation 233
6.11.3.1 main() 233
6.11.3.2 sigint_handler() 234
6.11.4 Variable Documentation 234
6.11.4.1 exit_program 235
6.11.4.2 program_name 235
6.12 examples/dm35425_ext_clocking.c File Reference 235
6.12.1 Detailed Description 236
6.12.2 Macro Definition Documentation 237
6.12.2.1 BUFFER_SIZE_BYTES 237
6.12.2.2 BUFFER_SIZE_SAMPLES 237
6.12.2.3 DEFAULT_RATE 237
6.12.2.4 DM35425_ADIO_DIR_OUTPUT 238
6.12.2.5 DM35425_EXT_CLK_DIR_INPUT 238
6.12.2.6 DM35425_EXT_CLK_EDGE_RISING 238
6.12.2.7 NUM_BUFFERS_TO_USE 238
6.12.3 Function Documentation 238
6.12.3.1 main() 238
6.12.3.2 sigint_handler() 239
6.12.4 Variable Documentation 240
6.12.4.1 exit_program 240
6.12.4.2 program_name 240
6.13 examples/dm35425_list_fb.c File Reference 240
6.13.1 Detailed Description 241

6.13.2 Function Documentation . . . 241
6.13.2.1 main() . . . 241
6.13.3 Variable Documentation . . . 242
6.13.3.1 program_name . . . 242
6.14 include/dm35425.h File Reference . . . 242
6.14.1 Detailed Description . . . 243
6.15 include/dm35425_adc_library.h File Reference . . . 243
6.15.1 Detailed Description . . . 249
6.16 include/dm35425_adio_library.h File Reference . . . 249
6.16.1 Detailed Description . . . 254
6.17 include/dm35425_board_access.h File Reference . . . 254
6.17.1 Detailed Description . . . 255
6.17.2 Macro Definition Documentation . . . 255
6.17.2.1 DM35425LIB_API . . . 255
6.18 include/dm35425_board_access_structs.h File Reference . . . 255
6.18.1 Detailed Description . . . 256
6.19 include/dm35425_dac_library.h File Reference . . . 256
6.19.1 Detailed Description . . . 260
6.20 include/dm35425_dma_library.h File Reference . . . 260
6.20.1 Detailed Description . . . 263
6.21 include/dm35425_driver.h File Reference . . . 263
6.21.1 Detailed Description . . . 264
6.22 include/dm35425_examples.h File Reference . . . 264
6.22.1 Detailed Description . . . 266
6.23 include/dm35425_ext_clocking_library.h File Reference . . . 266
6.23.1 Detailed Description . . . 267
6.24 include/dm35425_gbc_library.h File Reference . . . 268
6.24.1 Detailed Description . . . 268
6.25 include/dm35425_ioctl.h File Reference . . . 269
6.25.1 Detailed Description . . . 269
6.26 include/dm35425_os.h File Reference . . . 269
6.26.1 Detailed Description . . . 270
6.27 include/dm35425_registers.h File Reference . . . 270
6.27.1 Detailed Description . . . 276
6.28 include/dm35425_types.h File Reference . . . 276
6.28.1 Detailed Description . . . 278
6.28.2 Enumeration Type Documentation . . . 278
6.28.2.1 DM35425_Clock_Buses . . . 278
6.28.2.2 DM35425_Clock_Sources . . . 278
6.29 include/dm35425_util_library.h File Reference . . . 279
6.29.1 Detailed Description . . . 280

Index 281

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

DM35425 ADC Library Constants 7
DM35425 ADC Public Library Functions 14
DM35425 ADIO Public Library Enums 42
DM35425 ADIO Public Library Functions 43
DM35425 Board Access Structures 67
DM35425 PCI Region Structures 71
DM35425 DAC Library Constants 73
DM35425 DAC Library Public Functions 76
DM35425 DMA Public Library Constants 95
DM35425 DMA Public Library Functions 97
DM35425 Driver Constants 113
DM35425 Driver Enumerations 114
DM35425 Driver Structures 115
DM35425 Example Programs Constants 116
DM35425 Global Clocking Library Constants 120
DM35425 Global Clocking Library Public Functions 121
DM35425 Board Macros 128
DM35425 Board Library Public Functions 129
DM35425 Ioctl macros 135
DM35425 Board Access Public Library Functions 136
DM35425 Register Offsets 141
DM35425 Board Types 147
DM35425 Utility Library Functions 149

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

DM35425_Board_Descriptor
DM35425 board descriptor. This structure holds information about the board as a whole. It holds the file descriptor and ISR callback function, if applicable 153
dm35425_device_descriptor
DM35425 Device Descriptor. The identifying info for this particular board 154
DM35425_DMA_Descriptor
Descriptor for the DMA on this board 157
dm35425_dma_descriptor
DM35425 DMA descriptor. This structure holds information about a single DMA buffer 159
DM35425_Function_Block
DM35425 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 161
dm35425_ioctl_argument
Ioctl() request structure encapsulating all possible requests. This is what gets passed into the kernel from user space on the ioctl() call 163
dm35425_ioctl_dma
Ioctl() request structure for DMA 165
dm35425_ioctl_interrupt_info_request
Ioctl() request structure for interrupt 167
dm35425_ioctl_region_modify
Ioctl() request structure for PCI region read/modify/write 169
dm35425_ioctl_region_readwrite
Ioctl() request structure for read from or write to PCI region 170
dm35425_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 171
dm35425_pci_region
DM35425 PCI region descriptor. This structure holds information about one of a device's PCI memory regions 173

Chapter 3

File Index

3.1 File List

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

examples/dm35425_adc.c
Example program which demonstrates the use of the ADC, setting and responding to interrupts 180
examples/dm35425_adc_all_dma.c
Example program which demonstrates the use of the ADC and DMA, using all ADC channels at the same time 184
examples/dm35425_adc_continuous_dma.c
Example program which demonstrates the use of the ADC and DMA 191
examples/dm35425_adc_fifo.c
Example program which demonstrates the use of the ADC FIFO and its interrupts 201
examples/dm35425_adio.c
Example program which demonstrates the use of the ADIO 206
examples/dm35425_adio_adv_int.c
Example program which demonstrates the use of the ADIO advanced interrupts 209
examples/dm35425_adio_dma.c
Example program which demonstrates the use of the ADIO and DMA 214
examples/dm35425_adio_parallel_bus.c
Example program which demonstrates the use of the ADIO acting as a parallel bus 221
examples/dm35425_dac.c
Example program which demonstrates the use of the DAC 228
examples/dm35425_dac_dma.c
Example program which demonstrates the use of the DAC and DMA 230
examples/dm35425_ext_clocking.c
Example program which demonstrates the use of the external clocking function block 235
examples/dm35425_list_fb.c
Example program which demonstrates use of the library to open a function block for use 240
examples/_non_public/dm35425_dac_fifo.c
Example program which demonstrates the use of the DAC FIFO 175
include/dm35425.h
Defines for the DM35425 (Device-specific values) 242
include/dm35425_adc_library.h
Definitions for the DM35425 ADC Library 243
include/dm35425_adio_library.h
Definitions for the DM35425 ADIO Library 249
include/dm35425_board_access.h
Structures for the DM35425 Board Access Library 254

include/dm35425_board_access_structs.h
Structures for the DM35425 Board Access Library 255
include/dm35425_dac_library.h
Definitions for the DM35425 DAC Library 256
include/dm35425_dma_library.h
Definitions for the DM35425 DMA Library 260
include/dm35425_driver.h
Structures and defines for the DM35425 driver module 263
include/dm35425_examples.h
Defines for the DM35425 Example programs. Commonly used constants for the example programs included with the software package 264
include/dm35425_ext_clocking_library.h
Definitions for the DM35425 External Clocking Library 266
include/dm35425_gbc_library.h
Definitions for the DM35425 Board Library, a library for accessing the board registers 268
include/dm35425_ioctl.h
DM35425 Low level ioctl() request descriptor structure and request code definitions 269
include/dm35425_os.h
Function declarations for the DM35425 that are Linux specific 269
include/dm35425_registers.h
Defines for the DM35425 Registers (Offsets) 270
include/dm35425_types.h
Defines for the DM35425. Values for the general board, not specific to a particular function block 276
include/dm35425_util_library.h
Definitions for the DM35425 Utilities library, various helper functions 279

Chapter 4

Module Documentation

4.1 DM35425 ADC Library Constants

Macros

- #define DM35425_ADC_MODE_RESET 0x00

  Register value for ADC Mode Reset.
- #define DM35425_ADC_MODE_PAUSE 0x01

  Register value for ADC Mode Pause.
- #define DM35425_ADC_MODE_GO_SINGLE_SHOT 0x02

  Register value for ADC Mode Go (Single Shot)
- #define DM35425_ADC_MODE_GO_REARM 0x03

  Register value for ADC Mode Go (Rearm after Stop)
- #define DM35425_ADC_MODE_UNINITIALIZED 0x04

  Register value for ADC Mode Uninitialized.
- #define DM35425_ADC_STAT_STOPPED 0x00

  Register value for ADC Status - Stopped.
- #define DM35425_ADC_STAT_FILLING_PRE_TRIG_BUFF 0x01

  Register value for ADC Status - Filling Pre-Start Buffer.
- #define DM35425_ADC_STAT_WAITING_START_TRIG 0x02

  Register value for ADC Status - Waiting for Start Trigger.
- #define DM35425_ADC_STAT_SAMPLING 0x03

  Register value for ADC Status - Sampling Data.
- #define DM35425_ADC_STAT_FILLING_POST_TRIG_BUFF 0x04

  Register value for ADC Status - Filling Post-Stop Buffer.
- #define DM35425_ADC_STAT_WAIT_REARM 0x05

  Register value for ADC Status - Wait for Rearm.
- #define DM35425_ADC_STAT_DONE 0x07

  Register value for ADC Status - Done.
- #define DM35425_ADC_STAT_UNINITIALIZED 0x08

  Register value for ADC Status - Uninitialized.
- #define DM35425_ADC_STAT_INITIALIZING 0x09

  Register value for ADC Status - Initializing.
- #define DM35425_ADC_INT_SAMPLE_TAKEN_MASK 0x01

  Register value for Interrupt Mask - Sample Taken.
- #define DM35425_ADC_INT_CHAN_THRESHOLD_MASK 0x02

Register value for Interrupt Mask - Channel Threshold Exceeded.

- #define DM35425_ADC_INT_PRE_BUFF_FULL_MASK 0x04

  Register value for Interrupt Mask - Pre-Start Buffer Filled.
- #define DM35425_ADC_INT_START_TRIG_MASK 0x08

  Register value for Interrupt Mask - Start Trigger Occurred.
- #define DM35425_ADC_INT_STOP_TRIG_MASK 0x10

  Register value for Interrupt Mask - Stop Trigger Occurred.
- #define DM35425_ADC_INT_POST_BUFF_FULL_MASK 0x20

  Register value for Interrupt Mask - Post-Stop Buffer Filled.
- #define DM35425_ADC_INT_SAMP_COMPL_MASK 0x40

  Register value for Interrupt Mask - Sampling Complete.
- #define DM35425_ADC_INT_PACER_TICK_MASK 0x80

  Register value for Interrupt Mask - Pacer Clock Tick Occurred.
- #define DM35425_ADC_INT_ALL_MASK 0xFF

  Register value for Interrupt Mask - All Bits.
- #define DM35425_ADC_CHAN_INTR_LOW_THRESHOLD_MASK 0x01

  Register value for Channel Low Threshold Interrupt.
- #define DM35425_ADC_CHAN_INTR_HIGH_THRESHOLD_MASK 0x02

  Register value for Channel High Threshold Interrupt.
- #define DM35425_ADC_CHAN_FILTER_ORDER0 0x0

  Register value for Channel Filter Order 0.
- #define DM35425_ADC_CHAN_FILTER_ORDER1 0x1

  Register value for Channel Filter Order 1.
- #define DM35425_ADC_CHAN_FILTER_ORDER2 0x2

  Register value for Channel Filter Order 2.
- #define DM35425_ADC_CHAN_FILTER_ORDER3 0x3

  Register value for Channel Filter Order 3.
- #define DM35425_ADC_CHAN_FILTER_ORDER4 0x4

  Register value for Channel Filter Order 4.
- #define DM35425_ADC_CHAN_FILTER_ORDER5 0x5

  Register value for Channel Filter Order 5.
- #define DM35425_ADC_CHAN_FILTER_ORDER6 0x6

  Register value for Channel Filter Order 6.
- #define DM35425_ADC_CHAN_FILTER_ORDER7 0x7

  Register value for Channel Filter Order 7.
- #define DM35425_ADC_FE_CONFIG_GAIN_05 0x10

  Register value for setting Half-Gain.
- #define DM35425_ADC_FE_CONFIG_GAIN_1 0x00

  Register value for setting a Gain of 1.
- #define DM35425_ADC_FE_CONFIG_GAIN_2 0x04

  Register value for setting a Gain of 2.
- #define DM35425_ADC_FE_CONFIG_GAIN_4 0x08

  Register value for setting a Gain of 4.
- #define DM35425_ADC_FE_CONFIG_GAIN_8 0x0C

  Register value for setting a Gain of 8.
- #define DM35425_ADC_FE_CONFIG_GAIN_MASK 0x1C

  Register mask for setting gain bits.
- #define DM35425_ADC_FE_CONFIG_BIPOLAR 0x00

  Register value for setting input to Bi-Polar.
- #define DM35425_ADC_FE_CONFIG_UNIPOLAR 0x02

  Register value for setting input to Uni-Polar.

- #define DM35425_ADC_FE_CONFIG_POLARITY_MASK 0x02

  Register mask for setting polarity bits.
- #define DM35425_ADC_FE_CONFIG_SINGLE_ENDED 0x00

  Register value for setting input to Single-Ended.
- #define DM35425_ADC_FE_CONFIG_DIFFERENTIAL 0x01

  Register value for setting input to Differential.
- #define DM35425_ADC_FE_CONFIG_MODE_MASK 0x01

  Register mask for setting input mode.
- #define DM35425_ADC_FE_CONFIG_NO_DELAY 0x00

  Register value for configuring no sample delay.
- #define DM35425_ADC_FE_CONFIG_HALF_SAMPL_DELAY 0x40

  Register value for configuring a half sample delay.
- #define DM35425_ADC_FE_CONFIG_FULL_SAMPL_DELAY 0x80

  Register value for configuring a full sample delay.
- #define DM35425_ADC_FE_CONFIG_2_FULL_SAMPL_DELAY 0xC0

  Register value for configuring 2 full sample delay.
- #define DM35425_ADC_FE_CONFIG_DELAY_MASK 0xC0

  Register mask for setting delay value.
- #define DM35425_ADC_FE_CONFIG_ENABLED 0x20

  Register value for enabling ADC channel.
- #define DM35425_ADC_FE_CONFIG_DISABLED 0x00

  Register value for disabling ADC channel.
- #define DM35425_ADC_FE_CONFIG_ENABLE_MASK 0x20

  Register mask for setting ADC channel enable.
- #define DM35425_ADC_MAX_RATE 1250000

  Max allowable rate for the ADC (Hz)
- #define DM35425_ADC_THRESHOLD_MAX 4095L

  Maximum allowable value to write to the threshold register.
- #define DM35425_ADC_THRESHOLD_MIN 0L

  Minimum allowable value to write to the threshold register.
- #define DM35425_ADC_BIT_WIDTH_MAX 4096L

  The maximum value represented by the bit width of the ADC.
- #define DM35425_ADC_BIT_WIDTH_MAX_FLT ((float) DM35425_ADC_BIT_WIDTH_MAX)

  The max value of the ADC as a float.
- #define DM35425_ADC_RNG_1_25_LSB 0.00030517578125

  What each bit is worth at a range of 1.25.
- #define DM35425_ADC_RNG_2_5_LSB 0.0006103515625

  What each bit is worth at a range of 2.5.
- #define DM35425_ADC_RNG_5_LSB 0.001220703125

  What each bit is worth at a range of 5.
- #define DM35425_ADC_RNG_10_LSB 0.00244140625

  What each bit is worth at a range of 10.
- #define DM35425_ADC_RNG_20_LSB 0.0048828125

  What each bit is worth at a range of 20.
- #define DM35425_ADC_UNIPOLAR_MAX 4095L

  Max possible ADC value when in Unipolar.
- #define DM35425_ADC_UNIPOLAR_MIN 0

  Min possible ADC value when in Unipolar.
- #define DM35425_ADC_BIPOLAR_MAX 2047L

  Max possible ADC value when in Bipolar.
- #define DM35425_ADC_BIPOLAR_MIN -2048L

  Min possible ADC value when in Bipolar.

Enumerations

- enum DM35425_Adc_Clock_Events {
  DM35425_ADC_CLK_BUS_SRC_DISABLE = 0x00, DM35425_ADC_CLK_BUS_SRC_SAMPLE_TAKEN = 0x80, DM35425_ADC_CLK_BUS_SRC_CHAN_THRESH = 0x81, DM35425_ADC_CLK_BUS_SRC_PRE_START_BUFF_FUL
  = 0x82,
  DM35425_ADC_CLK_BUS_SRC_START_TRIG = 0x83, DM35425_ADC_CLK_BUS_SRC_STOP_TRIG = 0x84, DM35425_ADC_CLK_BUS_SRC_POST_STOP_BUFF_FULL = 0x85, DM35425_ADC_CLK_BUS_SRC_SAMPLING_C
  = 0x86,
  DM35425_ADC_CLK_BUS_SRC_PACER_TICK = 0x87 }

  Clock events for the global source clocks.
- enum DM35425_Input_Ranges {
  DM35425_ADC_RNG_BIPOLAR_10V, DM35425_ADC_RNG_BIPOLAR_5V, DM35425_ADC_RNG_BIPOLAR_2_5V, DM35425_ADC_RNG_BIPOLAR_1_25V,
  DM35425_ADC_RNG_BIPOLAR_625mV, DM35425_ADC_RNG_UNIPOLAR_5V, DM35425_ADC_RNG_UNIPOLAR_10V, DM35425_ADC_RNG_UNIPOLAR_2_5V,
  DM35425_ADC_RNG_UNIPOLAR_1_25V }

  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 DM35425_Input_Mode { DM35425_ADC_INPUT_SINGLE_ENDED, DM35425_ADC_INPUT_DIFFERENTIAL }

  Input mode of the ADC pin.
- enum DM35425_Gains {
  DM35425_ADC_GAIN_05, DM35425_ADC_GAIN_1, DM35425_ADC_GAIN_2, DM35425_ADC_GAIN_4,
  DM35425_ADC_GAIN_8, DM35425_ADC_GAIN_16, DM35425_ADC_GAIN_32, DM35425_ADC_GAIN_64,
  DM35425_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 DM35425_Channel_Delay { DM35425_ADC_NO_DELAY, DM35425_ADC_HALF_SAMPLE_DELAY, DM35425_ADC_FULL_SAMPLE_DELAY, DM35425_ADC_2_FULL_SAMPLE_DELAY }

  Channel to channel delay value.

4.1.1 Detailed Description

4.1.2 Enumeration Type Documentation

4.1.2.1 DM35425_Adc_Clock_Events

```
enum DM35425_Adc_Clock_Events
```

Clock events for the global source clocks.

Enumerator

DM35425_ADC_CLK_BUS_SRC_DISABLE	Register value for Clock Event - Disabled.
DM35425_ADC_CLK_BUS_SRC_SAMPLE_TAKEN	Register value for Clock Event - Sample Taken.
DM35425_ADC_CLK_BUS_SRC_CHAN_THRESH	Register value for Clock Event - Channel Threshold Exceeded.
DM35425_ADC_CLK_BUS_SRC_PRE_START_B↩ UFF_FULL	Register value for Clock Event - Pre-Start Buffer Full.

Enumerator

DM35425_ADC_CLK_BUS_SRC_START_TRIG	Register value for Clock Event - Start Trigger Occurred.
DM35425_ADC_CLK_BUS_SRC_STOP_TRIG	Register value for Clock Event - Stop Trigger Occurred.
DM35425_ADC_CLK_BUS_SRC_POST_STOP_B↩UFF_FULL	Register value for Clock Event - Post-Stop Buffer Full.
DM35425_ADC_CLK_BUS_SRC_SAMPLING_CO↩MPLETE	Register value for Clock Event - Sampling Complete.
DM35425_ADC_CLK_BUS_SRC_PACER_TICK	Register value for Clock Event - Pacer Tick Occurred.

Definition at line 455 of file dm35425_adc_library.h.

4.1.2.2 DM35425_Channel_Delay

```
enum DM35425_Channel_Delay
```

Channel to channel delay value.

Enumerator

DM35425_ADC_NO_DELAY	No Channel to Channel delay
DM35425_ADC_HALF_SAMPLE_DELAY	Half Sample Clock Channel to Channel Delay
DM35425_ADC_FULL_SAMPLE_DELAY	Full Sample Clock Channel to Channel Delay
DM35425_ADC_2_FULL_SAMPLE_DELAY	2 Full Sample Clock Channel to Channel Delay

Definition at line 661 of file dm35425_adc_library.h.

4.1.2.3 DM35425_Gains

```
enum DM35425_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

DM35425_ADC_GAIN_05	Input Half-Gain
DM35425_ADC_GAIN_1	Input Gain of 1
DM35425_ADC_GAIN_2	Input Gain of 2

Enumerator

DM35425_ADC_GAIN_4	Input Gain of 4
DM35425_ADC_GAIN_8	Input Gain of 8
DM35425_ADC_GAIN_16	Input Gain of 16
DM35425_ADC_GAIN_32	Input Gain of 32
DM35425_ADC_GAIN_64	Input Gain of 64
DM35425_ADC_GAIN_128	Input Gain of 128

Definition at line 607 of file dm35425_adc_library.h.

4.1.2.4 DM35425_Input_Mode

`enum DM35425_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

DM35425_ADC_INPUT_SINGLE_ENDED	Single-Ended Operation
DM35425_ADC_INPUT_DIFFERENTIAL	Differential Operation

Definition at line 582 of file dm35425_adc_library.h.

4.1.2.5 DM35425_Input_Ranges

`enum DM35425_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

DM35425_ADC_RNG_BIPOLAR_10V	Bipolar Mode, -10 to 10 V
DM35425_ADC_RNG_BIPOLAR_5V	Bipolar Mode, -5 to 5 V

Enumerator

DM35425_ADC_RNG_BIPOLAR_2_5V	Bipolar Mode, -2.5 to 2.5 V
DM35425_ADC_RNG_BIPOLAR_1_25V	Bipolar Mode, -1.25 to 1.25 V
DM35425_ADC_RNG_BIPOLAR_625mV	Bipolar Mode, -625 mV to 625 mV
DM35425_ADC_RNG_UNIPOLAR_5V	Unipolar Mode, 0 to 5 V
DM35425_ADC_RNG_UNIPOLAR_10V	Unipolar Mode, 0 to 10 V
DM35425_ADC_RNG_UNIPOLAR_2_5V	Unipolar Mode, 0 to 2.5 V
DM35425_ADC_RNG_UNIPOLAR_1_25V	Unipolar Mode, 0 to 1.25 V

Definition at line 524 of file dm35425_adc_library.h.

4.2 DM35425 ADC Public Library Functions

Functions

- DM35425LIB_API int DM35425_Adc_Open (struct DM35425_Board_Descriptor ∗handle, unsigned int number_of_type, struct DM35425_Function_Block ∗func_block)

  Open the ADC indicated, and determine register locations of control blocks needed to control it.
- DM35425LIB_API int DM35425_Adc_Get_Start_Trigger (struct DM35425_Board_Descriptor ∗handle, struct DM35425_Function_Block ∗func_block, uint8_t ∗start_trigger)

  Get the start trigger for data collection.
- DM35425LIB_API int DM35425_Adc_Set_Start_Trigger (struct DM35425_Board_Descriptor ∗handle, struct DM35425_Function_Block ∗func_block, uint8_t start_trigger)

  Set the start trigger for data collection.
- DM35425LIB_API int DM35425_Adc_Get_Stop_Trigger (struct DM35425_Board_Descriptor ∗handle, struct DM35425_Function_Block ∗func_block, uint8_t ∗stop_trigger)

  Get the stop trigger for data collection.
- DM35425LIB_API int DM35425_Adc_Set_Stop_Trigger (struct DM35425_Board_Descriptor ∗handle, struct DM35425_Function_Block ∗func_block, uint8_t stop_trigger)

  Set the stop trigger for data collection.
- DM35425LIB_API int DM35425_Adc_Get_Pre_Trigger_Samples (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t ∗count)

  Get the amount of data to capture prior to start trigger.
- DM35425LIB_API int DM35425_Adc_Set_Pre_Trigger_Samples (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t count)

  Set the amount of data to capture prior to start trigger.
- DM35425LIB_API int DM35425_Adc_Get_Post_Stop_Samples (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t ∗count)

  Get the amount of data to capture after stop trigger.
- DM35425LIB_API int DM35425_Adc_Set_Post_Stop_Samples (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t count)

  Set the amount of data to capture after stop trigger.
- DM35425LIB_API int DM35425_Adc_Get_Clock_Src (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, enum DM35425_Clock_Sources ∗source)

  Get the clock source for the ADC.
- DM35425LIB_API int DM35425_Adc_Set_Clock_Src (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, enum DM35425_Clock_Sources source)

  Set the clock source for the ADC.
- DM35425LIB_API int DM35425_Adc_Initialize (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block)

  Prepare the ADC for actual data collection. Moves the ADC from uninitialized to stopped.
- DM35425LIB_API int DM35425_Adc_Set_Clk_Divider (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t divider)

  Set the Clock Divider for the ADC function block.
- DM35425LIB_API int DM35425_Adc_Set_Sample_Rate (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t rate, uint32_t ∗actual_rate)

  Set the sampling rate for the ADC.
- DM35425LIB_API int DM35425_Adc_Channel_Get_Front_End_Config (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel, uint16_t ∗fe_config)

  Get the front-end config register contents.
- DM35425LIB_API int DM35425_Adc_Interrupt_Set_Config (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint16_t int_source, int enable)

  Configure the interrupts for the ADC.

- DM35425LIB_API int DM35425_Adc_Interrupt_Get_Config (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint16_t ∗interrupt_ena)

  Get the interrupt configuration for the ADC.
- DM35425LIB_API int DM35425_Adc_Start (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block)

  Set the ADC mode to Start.
- DM35425LIB_API int DM35425_Adc_Start_Rearm (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block)

  Set the ADC mode to Start-Rearm.
- DM35425LIB_API int DM35425_Adc_Reset (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block)

  Set the ADC mode to Reset.
- DM35425LIB_API int DM35425_Adc_Pause (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block)

  Set the ADC mode to Pause.
- DM35425LIB_API int DM35425_Adc_Uninitialize (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block)

  Set the ADC mode to Uninitialized.
- DM35425LIB_API int DM35425_Adc_Get_Mode_Status (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint8_t ∗mode_status)

  Get the ADC mode-status value.
- DM35425LIB_API int DM35425_Adc_Channel_Get_Last_Sample (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel, int32_t ∗value)

  Get the last sample taken from the ADC.
- DM35425LIB_API int DM35425_Adc_Get_Sample_Count (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t ∗value)

  Get the count of number of samples taken.
- DM35425LIB_API int DM35425_Adc_Interrupt_Get_Status (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint16_t ∗value)

  Get the interrupt status register.
- DM35425LIB_API int DM35425_Adc_Interrupt_Clear_Status (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint16_t value)

  Clear the interrupt status register.
- DM35425LIB_API int DM35425_Adc_Channel_Setup (struct DM35425_Board_Descriptor ∗handle, struct DM35425_Function_Block ∗func_block, unsigned int channel, enum DM35425_Channel_Delay input_delay, enum DM35425_Input_Ranges input_range, enum DM35425_Input_Mode input_mode)

  Setup the channel input for the ADC.
- DM35425LIB_API int DM35425_Adc_Channel_Reset (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel)

  Reset the channel front-end config.
- DM35425LIB_API int DM35425_Adc_Channel_Interrupt_Set_Config (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel, uint8_t interrupts_to_↩ set, int enable)

  Setup the channel interrupts.
- DM35425LIB_API int DM35425_Adc_Channel_Interrupt_Get_Config (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel, uint8_t ∗chan_intr_↩ enable)

  Get the channel interrupt configuration.
- DM35425LIB_API int DM35425_Adc_Channel_Interrupt_Get_Status (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel, uint8_t ∗chan_intr_↩ status)

  Get the channel interrupt status.
- DM35425LIB_API int DM35425_Adc_Channel_Interrupt_Clear_Status (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel, uint8_t chan_intr_status)

Clear the interrupt status for this channel.

- DM35425LIB_API int DM35425_Adc_Channel_Find_Interrupt (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_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.
- DM35425LIB_API int DM35425_Adc_Channel_Set_Filter (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel, uint8_t chan_filter)

  Set the filter value for the channel.
- DM35425LIB_API int DM35425_Adc_Channel_Get_Filter (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel, uint8_t ∗chan_filter)

  Get the filter value for the channel.
- DM35425LIB_API int DM35425_Adc_Channel_Set_Low_Threshold (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel, int32_t threshold)

  Set the lower threshold for this channel.
- DM35425LIB_API int DM35425_Adc_Channel_Set_High_Threshold (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel, int32_t threshold)

  Set the high threshold for this channel.
- DM35425LIB_API int DM35425_Adc_Channel_Get_Thresholds (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel, int32_t ∗low_threshold, int32_t ∗high_threshold)

  Get both thresholds for this channel.
- DM35425LIB_API int DM35425_Adc_Fifo_Channel_Read (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel, int32_t ∗value)

  Read an ADC sample stored in the onboard FIFO.
- DM35425LIB_API int DM35425_Adc_Set_Clock_Source_Global (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, enum DM35425_Clock_Sources clock_↩ select, enum DM35425_Adc_Clock_Events clock_driver)

  Set the global clock source for the ADC.
- DM35425LIB_API int DM35425_Adc_Get_Clock_Source_Global (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, int clock_select, int ∗clock_source)

  Get the global clock source for the selected clock.
- DM35425LIB_API int DM35425_Adc_Sample_To_Volts (enum DM35425_Input_Ranges input_range, int32_t adc_sample, float ∗volts)

  Convert an ADC sample to a volts value.
- DM35425LIB_API int DM35425_Adc_Volts_To_Sample (enum DM35425_Input_Ranges input_range, float volts, int32_t ∗adc_sample)

  Convert volts to an ADC value.

4.2.1 Detailed Description

DM35425_Adc_Library_Constants

4.2.2 Function Documentation

4.2.2.1 DM35425_Adc_Channel_Find_Interrupt()

```
DM35425LIB_API int DM35425_Adc_Channel_Find_Interrupt (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_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.

4.2.2.2 DM35425_Adc_Channel_Get_Filter()

```
DM35425LIB_API int DM35425_Adc_Channel_Get_Filter (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_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.

Return values

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

4.2.2.3 DM35425_Adc_Channel_Get_Front_End_Config()

```
DM35425LIB_API int DM35425_Adc_Channel_Get_Front_End_Config (
             struct DM35425_Board_Descriptor * handle,
             const struct DM35425_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.

4.2.2.4 DM35425_Adc_Channel_Get_Last_Sample()

```
DM35425LIB_API int DM35425_Adc_Channel_Get_Last_Sample (
             struct DM35425_Board_Descriptor * handle,
             const struct DM35425_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().

4.2.2.5 DM35425_Adc_Channel_Get_Thresholds()

```
DM35425LIB_API int DM35425_Adc_Channel_Get_Thresholds (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_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.

4.2.2.6 DM35425_Adc_Channel_Interrupt_Clear_Status()

```
DM35425LIB_API int DM35425_Adc_Channel_Interrupt_Clear_Status (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_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.

4.2.2.7 DM35425_Adc_Channel_Interrupt_Get_Config()

```
DM35425LIB_API int DM35425_Adc_Channel_Interrupt_Get_Config (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_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.

4.2.2.8 DM35425_Adc_Channel_Interrupt_Get_Status()

```
DM35425LIB_API int DM35425_Adc_Channel_Interrupt_Get_Status (
             struct DM35425_Board_Descriptor * handle,
             const struct DM35425_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.

4.2.2.9 DM35425_Adc_Channel_Interrupt_Set_Config()

```
DM35425LIB_API int DM35425_Adc_Channel_Interrupt_Set_Config (
             struct DM35425_Board_Descriptor * handle,
             const struct DM35425_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.

4.2.2.10 DM35425_Adc_Channel_Reset()

```
DM35425LIB_API int DM35425_Adc_Channel_Reset (
              struct DM35425_Board_Descriptor * handle,
              const struct DM35425_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.

4.2.2.11 DM35425_Adc_Channel_Set_Filter()

```
DM35425LIB_API int DM35425_Adc_Channel_Set_Filter (
              struct DM35425_Board_Descriptor * handle,
              const struct DM35425_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.

4.2.2.12 DM35425_Adc_Channel_Set_High_Threshold()

```
DM35425LIB_API int DM35425_Adc_Channel_Set_High_Threshold (
              struct DM35425_Board_Descriptor * handle,
              const struct DM35425_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.

Return values

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

4.2.2.13 DM35425_Adc_Channel_Set_Low_Threshold()

```
DM35425LIB_API int DM35425_Adc_Channel_Set_Low_Threshold (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_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.

4.2.2.14 DM35425_Adc_Channel_Setup()

```
DM35425LIB_API int DM35425_Adc_Channel_Setup (
            struct DM35425_Board_Descriptor * handle,
            struct DM35425_Function_Block * func_block,
            unsigned int channel,
            enum DM35425_Channel_Delay input_delay,
            enum DM35425_Input_Ranges input_range,
            enum DM35425_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_delay	Channel delay 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().

4.2.2.15 DM35425_Adc_Fifo_Channel_Read()

```
DM35425LIB_API int DM35425_Adc_Fifo_Channel_Read (
                struct DM35425_Board_Descriptor * handle,
                const struct DM35425_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.

Referenced by main().

4.2.2.16 DM35425_Adc_Get_Clock_Source_Global()

```
DM35425LIB_API int DM35425_Adc_Get_Clock_Source_Global (
              struct DM35425_Board_Descriptor * handle,
              const struct DM35425_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..

4.2.2.17 DM35425_Adc_Get_Clock_Src()

```
DM35425LIB_API int DM35425_Adc_Get_Clock_Src (
              struct DM35425_Board_Descriptor * handle,
              const struct DM35425_Function_Block * func_block,
              enum DM35425_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.

4.2.2.18 DM35425_Adc_Get_Mode_Status()

```
DM35425LIB_API int DM35425_Adc_Get_Mode_Status (
              struct DM35425_Board_Descriptor * handle,
              const struct DM35425_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.

4.2.2.19 DM35425_Adc_Get_Post_Stop_Samples()

```
DM35425LIB_API int DM35425_Adc_Get_Post_Stop_Samples (
              struct DM35425_Board_Descriptor * handle,
              const struct DM35425_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.

4.2.2.20 DM35425_Adc_Get_Pre_Trigger_Samples()

```
DM35425LIB_API int DM35425_Adc_Get_Pre_Trigger_Samples (
                struct DM35425_Board_Descriptor ∗ handle,
                const struct DM35425_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.

4.2.2.21 DM35425_Adc_Get_Sample_Count()

```
DM35425LIB_API int DM35425_Adc_Get_Sample_Count (
                struct DM35425_Board_Descriptor ∗ handle,
                const struct DM35425_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().

4.2.2.22 DM35425_Adc_Get_Start_Trigger()

```
DM35425LIB_API int DM35425_Adc_Get_Start_Trigger (
             struct DM35425_Board_Descriptor * handle,
             struct DM35425_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.

4.2.2.23 DM35425_Adc_Get_Stop_Trigger()

```
DM35425LIB_API int DM35425_Adc_Get_Stop_Trigger (
             struct DM35425_Board_Descriptor * handle,
             struct DM35425_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.

4.2.2.24 DM35425_Adc_Initialize()

```
DM35425LIB_API int DM35425_Adc_Initialize (
             struct DM35425_Board_Descriptor * handle,
             const struct DM35425_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().

4.2.2.25 DM35425_Adc_Interrupt_Clear_Status()

```
DM35425LIB_API int DM35425_Adc_Interrupt_Clear_Status (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_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().

4.2.2.26 DM35425_Adc_Interrupt_Get_Config()

```
DM35425LIB_API int DM35425_Adc_Interrupt_Get_Config (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_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().

4.2.2.27 DM35425_Adc_Interrupt_Get_Status()

```
DM35425LIB_API int DM35425_Adc_Interrupt_Get_Status (
             struct DM35425_Board_Descriptor * handle,
             const struct DM35425_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().

4.2.2.28 DM35425_Adc_Interrupt_Set_Config()

```
DM35425LIB_API int DM35425_Adc_Interrupt_Set_Config (
             struct DM35425_Board_Descriptor * handle,
             const struct DM35425_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.

Return values

Non-zero	Failure.

Referenced by main().

4.2.2.29 DM35425_Adc_Open()

```
DM35425LIB_API int DM35425_Adc_Open (
              struct DM35425_Board_Descriptor * handle,
              unsigned int number_of_type,
              struct DM35425_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().

4.2.2.30 DM35425_Adc_Pause()

```
DM35425LIB_API int DM35425_Adc_Pause (
              struct DM35425_Board_Descriptor * handle,
              const struct DM35425_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.

4.2.2.31 DM35425_Adc_Reset()

```
DM35425LIB_API int DM35425_Adc_Reset (
             struct DM35425_Board_Descriptor * handle,
             const struct DM35425_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().

4.2.2.32 DM35425_Adc_Sample_To_Volts()

```
DM35425LIB_API int DM35425_Adc_Sample_To_Volts (
             enum DM35425_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().

4.2.2.33 DM35425_Adc_Set_Clk_Divider()

```
DM35425LIB_API int DM35425_Adc_Set_Clk_Divider (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_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.

Referenced by main().

4.2.2.34 DM35425_Adc_Set_Clock_Source_Global()

```
DM35425LIB_API int DM35425_Adc_Set_Clock_Source_Global (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_Function_Block * func_block,
            enum DM35425_Clock_Sources clock_select,
            enum DM35425_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.

Parameters

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..

4.2.2.35 DM35425_Adc_Set_Clock_Src()

```
DM35425LIB_API int DM35425_Adc_Set_Clock_Src (
             struct DM35425_Board_Descriptor * handle,
             const struct DM35425_Function_Block * func_block,
             enum DM35425_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().

4.2.2.36 DM35425_Adc_Set_Post_Stop_Samples()

```
DM35425LIB_API int DM35425_Adc_Set_Post_Stop_Samples (
             struct DM35425_Board_Descriptor * handle,
             const struct DM35425_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().

4.2.2.37 DM35425_Adc_Set_Pre_Trigger_Samples()

```
DM35425LIB_API int DM35425_Adc_Set_Pre_Trigger_Samples (
                struct DM35425_Board_Descriptor * handle,
                const struct DM35425_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().

4.2.2.38 DM35425_Adc_Set_Sample_Rate()

```
DM35425LIB_API int DM35425_Adc_Set_Sample_Rate (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_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().

4.2.2.39 DM35425_Adc_Set_Start_Trigger()

```
DM35425LIB_API int DM35425_Adc_Set_Start_Trigger (
            struct DM35425_Board_Descriptor * handle,
            struct DM35425_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().

4.2.2.40 DM35425_Adc_Set_Stop_Trigger()

```
DM35425LIB_API int DM35425_Adc_Set_Stop_Trigger (
             struct DM35425_Board_Descriptor * handle,
             struct DM35425_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().

4.2.2.41 DM35425_Adc_Start()

```
DM35425LIB_API int DM35425_Adc_Start (
             struct DM35425_Board_Descriptor * handle,
             const struct DM35425_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().

4.2.2.42 DM35425_Adc_Start_Rearm()

```
DM35425LIB_API int DM35425_Adc_Start_Rearm (
             struct DM35425_Board_Descriptor * handle,
             const struct DM35425_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.

4.2.2.43 DM35425_Adc_Uninitialize()

```
DM35425LIB_API int DM35425_Adc_Uninitialize (
             struct DM35425_Board_Descriptor * handle,
             const struct DM35425_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.

4.2.2.44 DM35425_Adc_Volts_To_Sample()

```
DM35425LIB_API int DM35425_Adc_Volts_To_Sample (
              enum DM35425_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 DM35425 ADIO Public Library Enums

Enumerations

- enum DM35425_Adv_Interrupt_Mode { DM35425_ADV_INT_DISABLED, DM35425_ADV_INT_MATCH, DM35425_ADV_INT_EVENT }

  Advanced Interrupt Mode of the ADIO.

4.3.1 Detailed Description

4.3.2 Enumeration Type Documentation

4.3.2.1 DM35425_Adv_Interrupt_Mode

`enum DM35425_Adv_Interrupt_Mode`

Advanced Interrupt Mode of the ADIO.

Enumerator

DM35425_ADV_INT_DISABLED	Disabled
DM35425_ADV_INT_MATCH	Matching
DM35425_ADV_INT_EVENT	Event

Definition at line 291 of file dm35425_adio_library.h.

4.4 DM35425 ADIO Public Library Functions

Functions

- DM35425LIB_API int DM35425_Adio_Open (struct DM35425_Board_Descriptor ∗handle, unsigned int number_of_type, struct DM35425_Function_Block ∗func_block)

  Open the ADIO indicated, and determine register locations of control blocks needed to control it.
- DM35425LIB_API int DM35425_Adio_Start (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block)

  Set the ADIO mode to Start.
- DM35425LIB_API int DM35425_Adio_Start_Rearm (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block)

  Set the ADIO mode to Start-Rearm.
- DM35425LIB_API int DM35425_Adio_Reset (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block)

  Set the ADIO mode to Reset.
- DM35425LIB_API int DM35425_Adio_Pause (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block)

  Set the ADIO mode to Pause.
- DM35425LIB_API int DM35425_Adio_Uninitialize (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block)

  Set the ADIO mode to Uninitialized.
- DM35425LIB_API int DM35425_Adio_Get_Mode_Status (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint8_t ∗mode_status)

  Get the ADIO mode-status value.
- DM35425LIB_API int DM35425_Adio_Set_Clock_Src (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, enum DM35425_Clock_Sources source)

  Set the clock source for the ADIO.
- DM35425LIB_API int DM35425_Adio_Set_Start_Trigger (struct DM35425_Board_Descriptor ∗handle, struct DM35425_Function_Block ∗func_block, uint8_t start_trigger)

  Set the start trigger for data collection.
- DM35425LIB_API int DM35425_Adio_Set_Stop_Trigger (struct DM35425_Board_Descriptor ∗handle, struct DM35425_Function_Block ∗func_block, uint8_t stop_trigger)

  Set the stop trigger for data collection.
- DM35425LIB_API int DM35425_Adio_Set_Clk_Divider (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t divider)

  Set the Clock Divider for the ADIO function block.
- DM35425LIB_API int DM35425_Adio_Get_Clk_Div_Counter (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t ∗counter)

  Get the Clock Divider Counter for the ADIO function block.
- DM35425LIB_API int DM35425_Adio_Set_Pacer_Clk_Rate (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t requested_rate, uint32_t ∗actual_rate)

  Set the pacer clock rate, the rate at which conversions happen.
- DM35425LIB_API int DM35425_Adio_Set_Pre_Trigger_Samples (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t pre_capture_count)

  Set the amount of data to capture prior to start trigger.
- DM35425LIB_API int DM35425_Adio_Set_Post_Stop_Samples (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t post_capture_count)

  Set the amount of data to capture after stop trigger.
- DM35425LIB_API int DM35425_Adio_Get_Sample_Count (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t ∗value)

  Get the count of number of samples taken.

- DM35425LIB_API int DM35425_Adio_Interrupt_Set_Config (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint16_t interrupt_src, int enable)

  Configure the interrupts for the ADIO.
- DM35425LIB_API int DM35425_Adio_Interrupt_Get_Config (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint16_t ∗interrupt_ena)

  Get the interrupt configuration for the ADIO.
- DM35425LIB_API int DM35425_Adio_Interrupt_Get_Status (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint16_t ∗value)

  Get the interrupt status register.
- DM35425LIB_API int DM35425_Adio_Interrupt_Clear_Status (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint16_t value)

  Clear the interrupt status register.
- DM35425LIB_API int DM35425_Adio_Set_Clock_Source_Global (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, enum DM35425_Clock_Sources clock_←select, int clock_source)

  Set the global clock source for the ADIO.
- DM35425LIB_API int DM35425_Adio_Get_Clock_Source_Global (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, int clock_select, int ∗clock_source)

  Get the global clock source for the selected clock.
- DM35425LIB_API int DM35425_Adio_Get_Input_Value (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t ∗value)

  Get the input value of the ADIO.
- DM35425LIB_API int DM35425_Adio_Get_Output_Value (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t ∗value)

  Get the current value of the output register.
- DM35425LIB_API int DM35425_Adio_Set_Output_Value (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t value)

  Set the value to be put on output pins.
- DM35425LIB_API int DM35425_Adio_Get_Direction (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t ∗direction)

  Get the direction of the ADIO pins.
- DM35425LIB_API int DM35425_Adio_Set_Direction (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t direction)

  Set the direction of the ADIO pins.
- DM35425LIB_API int DM35425_Adio_Get_Adv_Int_Mode (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint8_t ∗adv_int_mode)

  Get the Advanced Interrupt Mode.
- DM35425LIB_API int DM35425_Adio_Set_Adv_Int_Mode (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint8_t adv_int_mode)

  Set the Advanced Interrupt Mode.
- DM35425LIB_API int DM35425_Adio_Get_Adv_Int_Mask (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t ∗adv_int_mask)

  Get the Advanced Interrupt Mask.
- DM35425LIB_API int DM35425_Adio_Set_Adv_Int_Mask (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t adv_int_mask)

  Set the Advanced Interrupt Mask.
- DM35425LIB_API int DM35425_Adio_Get_Adv_Int_Comp (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t ∗adv_int_comp)

  Get the Advanced Interrupt Compare Register.
- DM35425LIB_API int DM35425_Adio_Set_Adv_Int_Comp (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t adv_int_comp)

  Set the Advanced Interrupt Compare Register.

- DM35425LIB_API int DM35425_Adio_Get_Adv_Int_Capt (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t ∗adv_int_capt)

  Get the Advanced Interrupt Capture Register.
- DM35425LIB_API int DM35425_Adio_Set_Adv_Int_Capt (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t adv_int_capt)

  Set the Advanced Interrupt Capture Register.
- DM35425LIB_API int DM35425_Adio_Get_P_Bus_Enable (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, int ∗p_bus_enabled)

  Get the Parallel Bus Enable Register boolean value.
- DM35425LIB_API int DM35425_Adio_Set_P_Bus_Enable (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, int p_bus_enabled)

  Set the Parallel Bus Enable.
- DM35425LIB_API int DM35425_Adio_Get_P_Bus_Ready_Enable (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, int ∗p_bus_ready_enabled)

  Get the Parallel Bus Ready Enable Register boolean value.
- DM35425LIB_API int DM35425_Adio_Set_P_Bus_Ready_Enable (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, int p_bus_ready_enabled)

  Set the Parallel Bus Ready Enable.
- DM35425LIB_API int DM35425_Adio_Fifo_Channel_Read (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel, int32_t ∗value)

  Read an aDIO sample from the FIFO.
- DM35425LIB_API int DM35425_Adio_Fifo_Channel_Write (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel, int32_t value)

  Write an aDIO sample to the FIFO.

4.4.1 Detailed Description

DM35425_Adio_Library_Enums

4.4.2 Function Documentation

4.4.2.1 DM35425_Adio_Fifo_Channel_Read()

```
DM35425LIB_API int DM35425_Adio_Fifo_Channel_Read (
            struct DM35425_Board_Descriptor ∗ handle,
            const struct DM35425_Function_Block ∗ func_block,
            unsigned int channel,
            int32_t ∗ value )
```

Read an aDIO sample from the 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.

4.4.2.2 DM35425_Adio_Fifo_Channel_Write()

```
DM35425LIB_API int DM35425_Adio_Fifo_Channel_Write (
             struct DM35425_Board_Descriptor * handle,
             const struct DM35425_Function_Block * func_block,
             unsigned int channel,
             int32_t value )
```

Write an aDIO sample to the 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 to.
value	sample value to write.

Return values

0	Success.
Non-zero	Failure.

4.4.2.3 DM35425_Adio_Get_Adv_Int_Capt()

```
DM35425LIB_API int DM35425_Adio_Get_Adv_Int_Capt (
             struct DM35425_Board_Descriptor * handle,
             const struct DM35425_Function_Block * func_block,
             uint32_t * adv_int_capt )
```

Get the Advanced Interrupt Capture Register.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block representing the ADIO.
adv_int_capt	Pointer to the returned value of the advanced interrupt capture register.

Return values

0	Success.
Non-Zero	Failure.

4.4.2.4 DM35425_Adio_Get_Adv_Int_Comp()

```
DM35425LIB_API int DM35425_Adio_Get_Adv_Int_Comp (
             struct DM35425_Board_Descriptor * handle,
             const struct DM35425_Function_Block * func_block,
             uint32_t * adv_int_comp )
```

Get the Advanced Interrupt Compare Register.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block representing the ADIO.
adv_int_comp	Pointer to the returned value of the advanced interrupt compare register.

Return values

0	Success.
Non-Zero	Failure.

4.4.2.5 DM35425_Adio_Get_Adv_Int_Mask()

```
DM35425LIB_API int DM35425_Adio_Get_Adv_Int_Mask (
             struct DM35425_Board_Descriptor * handle,
             const struct DM35425_Function_Block * func_block,
             uint32_t * adv_int_mask )
```

Get the Advanced Interrupt Mask.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block representing the ADIO.
adv_int_mask	Pointer to the returned value of the advanced interrupt mask register.

Return values

0	Success.
Non-Zero	Failure.

4.4.2.6 DM35425_Adio_Get_Adv_Int_Mode()

```
DM35425LIB_API int DM35425_Adio_Get_Adv_Int_Mode (
             struct DM35425_Board_Descriptor * handle,
             const struct DM35425_Function_Block * func_block,
             uint8_t * adv_int_mode )
```

Get the Advanced Interrupt Mode.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block representing the ADIO.
adv_int_mode	Pointer to the returned value of the advanced interrupt mode register.

Return values

0	Success.
Non-Zero	Failure.

4.4.2.7 DM35425_Adio_Get_Clk_Div_Counter()

```
DM35425LIB_API int DM35425_Adio_Get_Clk_Div_Counter (
             struct DM35425_Board_Descriptor * handle,
             const struct DM35425_Function_Block * func_block,
             uint32_t * counter )
```

Get the Clock Divider Counter for the ADIO 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.
counter	Pointer where the counter value will be returned.

Return values

0	Success.
-1	Failure.

4.4.2.8 DM35425_Adio_Get_Clock_Source_Global()

```
DM35425LIB_API int DM35425_Adio_Get_Clock_Source_Global (
              struct DM35425_Board_Descriptor * handle,
              const struct DM35425_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..

4.4.2.9 DM35425_Adio_Get_Direction()

```
DM35425LIB_API int DM35425_Adio_Get_Direction (
              struct DM35425_Board_Descriptor * handle,
              const struct DM35425_Function_Block * func_block,
              uint32_t * direction )
```

Get the direction of the ADIO pins.

Parameters

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

Return values

0	Success.
Non-Zero	Failure.

4.4.2.10 DM35425_Adio_Get_Input_Value()

```
DM35425LIB_API int DM35425_Adio_Get_Input_Value (
             struct DM35425_Board_Descriptor * handle,
             const struct DM35425_Function_Block * func_block,
             uint32_t * value )
```

Get the input value of the ADIO.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block representing the ADIO.
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().

4.4.2.11 DM35425_Adio_Get_Mode_Status()

```
DM35425LIB_API int DM35425_Adio_Get_Mode_Status (
             struct DM35425_Board_Descriptor * handle,
             const struct DM35425_Function_Block * func_block,
             uint8_t * mode_status )
```

Get the ADIO 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.

4.4.2.12 DM35425_Adio_Get_Output_Value()

```
DM35425LIB_API int DM35425_Adio_Get_Output_Value (
              struct DM35425_Board_Descriptor * handle,
              const struct DM35425_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 ADIO.
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.

4.4.2.13 DM35425_Adio_Get_P_Bus_Enable()

```
DM35425LIB_API int DM35425_Adio_Get_P_Bus_Enable (
              struct DM35425_Board_Descriptor * handle,
              const struct DM35425_Function_Block * func_block,
              int * p_bus_enabled )
```

Get the Parallel Bus Enable Register boolean value.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block representing the ADIO.
p_bus_enabled	Pointer to the returned value of the parallel bus enable, as a boolean (0 = disabled, non-zero = enabled).

Return values

0	Success.
Non-Zero	Failure.

4.4.2.14 DM35425_Adio_Get_P_Bus_Ready_Enable()

```
DM35425LIB_API int DM35425_Adio_Get_P_Bus_Ready_Enable (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_Function_Block * func_block,
            int * p_bus_ready_enabled )
```

Get the Parallel Bus Ready Enable Register boolean value.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block representing the ADIO.
p_bus_ready_enabled	Pointer to the returned value of the parallel bus ready enable, as a boolean (0 = disabled, non-zero = enabled).

Return values

0	Success.
Non-Zero	Failure.

4.4.2.15 DM35425_Adio_Get_Sample_Count()

```
DM35425LIB_API int DM35425_Adio_Get_Sample_Count (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_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.

4.4.2.16 DM35425_Adio_Interrupt_Clear_Status()

```
DM35425LIB_API int DM35425_Adio_Interrupt_Clear_Status (
              struct DM35425_Board_Descriptor * handle,
              const struct DM35425_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 ISR().

4.4.2.17 DM35425_Adio_Interrupt_Get_Config()

```
DM35425LIB_API int DM35425_Adio_Interrupt_Get_Config (
              struct DM35425_Board_Descriptor * handle,
              const struct DM35425_Function_Block * func_block,
              uint16_t * interrupt_ena )
```

Get the interrupt configuration for the ADIO.

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.

4.4.2.18 DM35425_Adio_Interrupt_Get_Status()

```
DM35425LIB_API int DM35425_Adio_Interrupt_Get_Status (
              struct DM35425_Board_Descriptor * handle,
              const struct DM35425_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.

4.4.2.19 DM35425_Adio_Interrupt_Set_Config()

```
DM35425LIB_API int DM35425_Adio_Interrupt_Set_Config (
              struct DM35425_Board_Descriptor * handle,
              const struct DM35425_Function_Block * func_block,
              uint16_t interrupt_src,
              int enable )
```

Configure the interrupts for the ADIO.

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_src	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.

4.4.2.20 DM35425_Adio_Open()

```
DM35425LIB_API int DM35425_Adio_Open (
            struct DM35425_Board_Descriptor * handle,
            unsigned int number_of_type,
            struct DM35425_Function_Block * func_block )
```

Open the ADIO 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 ADIO to open. The first ADIO 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().

4.4.2.21 DM35425_Adio_Pause()

```
DM35425LIB_API int DM35425_Adio_Pause (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_Function_Block * func_block )
```

Set the ADIO 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.

4.4.2.22 DM35425_Adio_Reset()

```
DM35425LIB_API int DM35425_Adio_Reset (
              struct DM35425_Board_Descriptor * handle,
              const struct DM35425_Function_Block * func_block )
```

Set the ADIO 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.

4.4.2.23 DM35425_Adio_Set_Adv_Int_Capt()

```
DM35425LIB_API int DM35425_Adio_Set_Adv_Int_Capt (
              struct DM35425_Board_Descriptor * handle,
              const struct DM35425_Function_Block * func_block,
              uint32_t adv_int_capt )
```

Set the Advanced Interrupt Capture Register.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block representing the ADIO.
adv_int_capt	Value of the advanced interrupt compare register.

Return values

0	Success.
Non-Zero	Failure.

4.4.2.24 DM35425_Adio_Set_Adv_Int_Comp()

```
DM35425LIB_API int DM35425_Adio_Set_Adv_Int_Comp (
              struct DM35425_Board_Descriptor * handle,
              const struct DM35425_Function_Block * func_block,
              uint32_t adv_int_comp )
```

Set the Advanced Interrupt Compare Register.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block representing the ADIO.
adv_int_comp	Value of the advanced interrupt compare register.

Return values

0	Success.
Non-Zero	Failure.

4.4.2.25 DM35425_Adio_Set_Adv_Int_Mask()

```
DM35425LIB_API int DM35425_Adio_Set_Adv_Int_Mask (
             struct DM35425_Board_Descriptor * handle,
             const struct DM35425_Function_Block * func_block,
             uint32_t adv_int_mask )
```

Set the Advanced Interrupt Mask.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block representing the ADIO.
adv_int_mask	Value of the advanced interrupt mask register.

Return values

0	Success.
Non-Zero	Failure.

4.4.2.26 DM35425_Adio_Set_Adv_Int_Mode()

```
DM35425LIB_API int DM35425_Adio_Set_Adv_Int_Mode (
             struct DM35425_Board_Descriptor * handle,
             const struct DM35425_Function_Block * func_block,
             uint8_t adv_int_mode )
```

Set the Advanced Interrupt Mode.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block representing the ADIO.
adv_int_mode	Pointer to the returned value of the advanced interrupt mode register.

Return values

0	Success.
Non-Zero	Failure.

4.4.2.27 DM35425_Adio_Set_Clk_Divider()

```
DM35425LIB_API int DM35425_Adio_Set_Clk_Divider (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_Function_Block * func_block,
            uint32_t divider )
```

Set the Clock Divider for the ADIO 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.

Referenced by main().

4.4.2.28 DM35425_Adio_Set_Clock_Source_Global()

```
DM35425LIB_API int DM35425_Adio_Set_Clock_Source_Global (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_Function_Block * func_block,
            enum DM35425_Clock_Sources clock_select,
            int clock_source )
```

Set the global clock source for the ADIO.

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_source	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..

4.4.2.29 DM35425_Adio_Set_Clock_Src()

```
DM35425LIB_API int DM35425_Adio_Set_Clock_Src (
             struct DM35425_Board_Descriptor * handle,
             const struct DM35425_Function_Block * func_block,
             enum DM35425_Clock_Sources source )
```

Set the clock source for the ADIO.

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 ADIO. 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().

4.4.2.30 DM35425_Adio_Set_Direction()

```
DM35425LIB_API int DM35425_Adio_Set_Direction (
             struct DM35425_Board_Descriptor * handle,
             const struct DM35425_Function_Block * func_block,
             uint32_t direction )
```

Set the direction of the ADIO pins.

Parameters

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

Return values

0	Success.
Non-Zero	Failure.

Referenced by main().

4.4.2.31 DM35425_Adio_Set_Output_Value()

```
DM35425LIB_API int DM35425_Adio_Set_Output_Value (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_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 ADIO.
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().

4.4.2.32 DM35425_Adio_Set_P_Bus_Enable()

```
DM35425LIB_API int DM35425_Adio_Set_P_Bus_Enable (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_Function_Block * func_block,
            int p_bus_enabled )
```

Set the Parallel Bus Enable.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block representing the ADIO.
p_bus_enabled	Boolean value indicating whether to enable the parallel bus or not. (0 = disabled, non-zero = enabled).

Return values

0	Success.
Non-Zero	Failure.

Referenced by main().

4.4.2.33 DM35425_Adio_Set_P_Bus_Ready_Enable()

```
DM35425LIB_API int DM35425_Adio_Set_P_Bus_Ready_Enable (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_Function_Block * func_block,
            int p_bus_ready_enabled )
```

Set the Parallel Bus Ready Enable.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the function block representing the ADIO.
p_bus_ready_enabled	Boolean value indicating whether to enable the parallel bus ready or not. (0 = disabled, non-zero = enabled).

Return values

0	Success.
Non-Zero	Failure.

Referenced by main().

4.4.2.34 DM35425_Adio_Set_Pacer_Clk_Rate()

```
DM35425LIB_API int DM35425_Adio_Set_Pacer_Clk_Rate (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_Function_Block * func_block,
            uint32_t requested_rate,
            uint32_t * actual_rate )
```

Set the pacer clock rate, the rate at which conversions happen.

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.
requested_rate	The rate being requested (Hz). It is not always possible to provide the exact rate being requested due to the resolution of the divider.
actual_rate	Pointer where the returned value of the actual rate achieved (Hz).

Return values

0	Success.
-1	Failure.

Referenced by main().

4.4.2.35 DM35425_Adio_Set_Post_Stop_Samples()

```
DM35425LIB_API int DM35425_Adio_Set_Post_Stop_Samples (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_Function_Block * func_block,
            uint32_t post_capture_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.
post_capture_count	Number of samples to capture after the stop trigger.

Return values

0	Success.

Return values

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

4.4.2.36 DM35425_Adio_Set_Pre_Trigger_Samples()

```
DM35425LIB_API int DM35425_Adio_Set_Pre_Trigger_Samples (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_Function_Block * func_block,
            uint32_t pre_capture_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.
pre_capture_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.

4.4.2.37 DM35425_Adio_Set_Start_Trigger()

```
DM35425LIB_API int DM35425_Adio_Set_Start_Trigger (
            struct DM35425_Board_Descriptor * handle,
            struct DM35425_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().

4.4.2.38 DM35425_Adio_Set_Stop_Trigger()

```
DM35425LIB_API int DM35425_Adio_Set_Stop_Trigger (
            struct DM35425_Board_Descriptor * handle,
            struct DM35425_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().

4.4.2.39 DM35425_Adio_Start()

```
DM35425LIB_API int DM35425_Adio_Start (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_Function_Block * func_block )
```

Set the ADIO 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().

4.4.2.40 DM35425_Adio_Start_Rearm()

```
DM35425LIB_API int DM35425_Adio_Start_Rearm (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_Function_Block * func_block )
```

Set the ADIO 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.

4.4.2.41 DM35425_Adio_Uninitialize()

```
DM35425LIB_API int DM35425_Adio_Uninitialize (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_Function_Block * func_block )
```

Set the ADIO 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.

4.5 DM35425 Board Access Structures

Data Structures

- struct DM35425_DMA_Descriptor

  Descriptor for the DMA on this board.
- struct DM35425_Function_Block

  DM35425 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

- DM35425LIB_API int DM35425_Board_Open (uint8_t dev_num, struct DM35425_Board_Descriptor **handle)

  Open the board, providing the file descriptor that all future operations will reference. Also allocate memory for the device descriptor.
- DM35425LIB_API int DM35425_Board_Close (struct DM35425_Board_Descriptor *handle)

  Close the board, closing the open handle for the device file, and freeing the memory allocated for the decriptor.
- DM35425LIB_API int DM35425_Read (struct DM35425_Board_Descriptor *handle, union dm35425_ioctl_argument *ioctl_request)

  Read from the board.
- DM35425LIB_API int DM35425_Write (struct DM35425_Board_Descriptor *handle, union dm35425_ioctl_argument *ioctl_request)

  Write to the board.
- DM35425LIB_API int DM35425_Modify (struct DM35425_Board_Descriptor *handle, union dm35425_ioctl_argument *ioctl_request)

  Read/Modify/Write to the board.
- int DM35425_Dma (struct DM35425_Board_Descriptor *handle, union dm35425_ioctl_argument *ioctl_↩ request)

  Perform a DMA operation.

4.5.1 Detailed Description

4.5.2 Function Documentation

4.5.2.1 DM35425_Board_Close()

```
DM35425LIB_API int DM35425_Board_Close (
            struct DM35425_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().

4.5.2.2 DM35425_Board_Open()

```
DM35425LIB_API int DM35425_Board_Open (
             uint8_t dev_num,
             struct DM35425_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().

4.5.2.3 DM35425_Dma()

```
int DM35425_Dma (
             struct DM35425_Board_Descriptor * handle,
             union dm35425_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.

4.5.2.4 DM35425_Modify()

```
DM35425LIB_API int DM35425_Modify (
            struct DM35425_Board_Descriptor * handle,
            union dm35425_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.

4.5.2.5 DM35425_Read()

```
DM35425LIB_API int DM35425_Read (
            struct DM35425_Board_Descriptor * handle,
            union dm35425_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.

4.5.2.6 DM35425_Write()

```
DM35425LIB_API int DM35425_Write (
            struct DM35425_Board_Descriptor * handle,
            union dm35425_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.

4.6 DM35425 PCI Region Structures

Data Structures

- struct dm35425_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 dm35425_ioctl_region_readwrite

  ioctl() request structure for read from or write to PCI region
- struct dm35425_ioctl_region_modify

  ioctl() request structure for PCI region read/modify/write
- struct dm35425_ioctl_interrupt_info_request

  ioctl() request structure for interrupt
- struct dm35425_ioctl_dma

  ioctl() request structure for DMA
- union dm35425_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 dm35425_pci_region_num { DM35425_PCI_REGION_GBC = 0, DM35425_PCI_REGION_GBC2, DM35425_PCI_REGION_FB }

  Standard PCI region number.
- enum dm35425_pci_region_access_size { DM35425_PCI_REGION_ACCESS_8 = 0, DM35425_PCI_REGION_ACCESS_16, DM35425_PCI_REGION_ACCESS_32 }

  Desired size in bits of access to standard PCI region.
- enum DM35425_DMA_FUNCTIONS { DM35425_DMA_INITIALIZE, DM35425_DMA_READ, DM35425_DMA_WRITE }

4.6.1 Detailed Description

4.6.2 Enumeration Type Documentation

4.6.2.1 DM35425_DMA_FUNCTIONS

```
enum DM35425_DMA_FUNCTIONS
```

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

Enumerator

DM35425_DMA_INITIALIZE	Initialize the DMA buffers
DM35425_DMA_READ	Read from the DMA buffers (transfer to user space)
DM35425_DMA_WRITE	Write to the DMA buffers (transfer from user space)

Definition at line 96 of file dm35425_board_access_structs.h.

4.6.2.2 dm35425_pci_region_access_size

```
enum dm35425_pci_region_access_size
```

Desired size in bits of access to standard PCI region.

Enumerator

DM35425_PCI_REGION_ACCESS_8	8-bit access
DM35425_PCI_REGION_ACCESS_16	16-bit access
DM35425_PCI_REGION_ACCESS_32	32-bit access

Definition at line 68 of file dm35425_board_access_structs.h.

4.6.2.3 dm35425_pci_region_num

```
enum dm35425_pci_region_num
```

Standard PCI region number.

Enumerator

DM35425_PCI_REGION_GBC	General Board Control Registers (BAR0)
DM35425_PCI_REGION_GBC2	General Board Control Registers (64-bit) (BAR1)
DM35425_PCI_REGION_FB	Functional Blocks Registers (BAR2)

Definition at line 40 of file dm35425_board_access_structs.h.

4.7 DM35425 DAC Library Constants

Functions.

Macros

- #define DM35425_DAC_INT_CONVERSION_SENT_MASK 0x01

  Register value for Interrupt Mask - Conversion Sent.
- #define DM35425_DAC_INT_CHAN_MARKER_MASK 0x02

  Register value for Interrupt Mask - Channel has enabled marker.
- #define DM35425_DAC_INT_START_TRIG_MASK 0x08

  Register value for Interrupt Mask - Start Trigger Occurred.
- #define DM35425_DAC_INT_STOP_TRIG_MASK 0x10

  Register value for Interrupt Mask - Stop Trigger Occurred.
- #define DM35425_DAC_INT_POST_STOP_DONE_MASK 0x20

  Register value for Interrupt Mask - Post-Stop Conversions Completed.
- #define DM35425_DAC_INT_PACER_TICK_MASK 0x80

  Register value for Interrupt Mask - Pacer Clock Tick.
- #define DM35425_DAC_INT_ALL_MASK 0xBB

  Register value for Interrupt Mask - All Bits.
- #define DM35425_DAC_MODE_RESET 0x00

  Register value for Mode - Reset.
- #define DM35425_DAC_MODE_PAUSE 0x01

  Register value for Mode - Pause.
- #define DM35425_DAC_MODE_GO_SINGLE_SHOT 0x02

  Register value for Mode - Go (Single Shot)
- #define DM35425_DAC_MODE_GO_REARM 0x03

  Register value for Mode - Go (Re-arm)
- #define DM35425_DAC_STATUS_STOPPED 0x00

  Register value for DAC Status - Stopped.
- #define DM35425_DAC_STATUS_WAITING_START_TRIG 0x02

  Register value for DAC Status - Waiting for Start Trigger.
- #define DM35425_DAC_STATUS_CONVERTING 0x03

  Register value for DAC Status - Converting Data.
- #define DM35425_DAC_STATUS_OUTPUT_POST 0x04

  Register value for DAC Status - Outputting Post-Stop conversions.
- #define DM35425_DAC_STATUS_WAITING_REARM 0x05

  Register value for DAC Status - Waiting for Re-Arm.
- #define DM35425_DAC_STATUS_DONE 0x07

  Register value for DAC Status - Done.
- #define DM35425_DAC_FE_CONFIG_OUTPUT_ENABLE 0x04

  Register value for enabling DAC output.
- #define DM35425_DAC_FE_CONFIG_OUTPUT_DISABLE 0x00

  Register value for disabling DAC output.
- #define DM35425_DAC_FE_CONFIG_ENABLE_MASK 0x04

  Register mask for setting DAC enable/disable.
- #define DM35425_DAC_FE_CONFIG_GAIN_1 0x00

  Register value for setting a Gain of 1.
- #define DM35425_DAC_FE_CONFIG_GAIN_2 0x01

Register value for setting a Gain of 2.
- #define DM35425_DAC_FE_CONFIG_UNIPOLAR 0x00

  Register value for setting DAC to Unipolar.
- #define DM35425_DAC_FE_CONFIG_BIPOLAR 0x02

  Register value for setting DAC to Bipolar.
- #define DM35425_DAC_FE_CONFIG_GAIN_MASK 0x01

  Register mask for setting gain.
- #define DM35425_DAC_FE_CONFIG_POLARITY_MASK 0x02

  Register mask for setting polarity.
- #define DM35425_DAC_MIN 0

  DAC Min value.
- #define DM35425_DAC_MAX 4095

  DAC Max value.
- #define DM35425_DAC_BIPOLAR_OFFSET 0x0800

  Offset to add to DAC value when in Bipolar mode.
- #define DM35425_DAC_UNIPOLAR_OFFSET 0x00

  Offset to add to the DAC value when in Unipolar mode.
- #define DM35425_DAC_RNG_5_LSB 0.001220703125f

  What each bit is worth at a range of 5.
- #define DM35425_DAC_RNG_10_LSB 0.00244140625f

  What each bit is worth at a range of 10.
- #define DM35425_DAC_RNG_20_LSB 0.0048828125f

  What each bit is worth at a range of 20.
- #define DM35425_DAC_MAX_RATE 200000

  Max allowable rate for the DAC (Hz)

Enumerations

- enum DM35425_Dac_Clock_Events {
  DM35425_DAC_CLK_BUS_SRC_DISABLE = 0x00, DM35425_DAC_CLK_BUS_SRC_CONVERSION_SENT = 0x80, DM35425_DAC_CLK_BUS_SRC_CHAN_MARKER = 0x81, DM35425_DAC_CLK_BUS_SRC_START_TRIG = 0x83,
  DM35425_DAC_CLK_BUS_SRC_STOP_TRIG = 0x84, DM35425_DAC_CLK_BUS_SRC_CONV_COMPL = 0x85 }

  Clocking events that can be used as the global clock sources.
- enum DM35425_Output_Ranges { DM35425_DAC_RNG_UNIPOLAR_5V, DM35425_DAC_RNG_UNIPOLAR_10V, DM35425_DAC_RNG_BIPOLAR_5V, DM35425_DAC_RNG_BIPOLAR_10V }

  Output range of the DAC pin.

4.7.1 Detailed Description

Functions.

4.7.2 Enumeration Type Documentation

4.7.2.1 DM35425_Dac_Clock_Events

```
enum DM35425_Dac_Clock_Events
```

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

Enumerator

DM35425_DAC_CLK_BUS_SRC_DISABLE	Register value for Clock Event - Disabled.
DM35425_DAC_CLK_BUS_SRC_CONVERSION_↩ SENT	Register value for Clock Event - Conversion Sent.
DM35425_DAC_CLK_BUS_SRC_CHAN_MARKER	Register value for Clock Event - Channel has enabled marker.
DM35425_DAC_CLK_BUS_SRC_START_TRIG	Register value for Clock Event - Start Trigger Occurred.
DM35425_DAC_CLK_BUS_SRC_STOP_TRIG	Register value for Clock Event - Stop Trigger Occurred.
DM35425_DAC_CLK_BUS_SRC_CONV_COMPL	Register value for Clock Event - Conversions Complete.

Definition at line 258 of file dm35425_dac_library.h.

4.7.2.2 DM35425_Output_Ranges

```
enum DM35425_Output_Ranges
```

Output range of the DAC 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 legal values.

Enumerator

DM35425_DAC_RNG_UNIPOLAR_5V	Range 0 to 5 V
DM35425_DAC_RNG_UNIPOLAR_10V	Range 0 to 10 V
DM35425_DAC_RNG_BIPOLAR_5V	Range -5 V to 5 V
DM35425_DAC_RNG_BIPOLAR_10V	Range -10 V to 10 V

Definition at line 309 of file dm35425_dac_library.h.

4.8 DM35425 DAC Library Public Functions

Functions

- DM35425LIB_API int DM35425_Dac_Open (struct DM35425_Board_Descriptor ∗handle, unsigned int number_of_type, struct DM35425_Function_Block ∗func_block)

  Open the DAC indicated, and determine register locations of control blocks needed to control it.
- DM35425LIB_API int DM35425_Dac_Set_Clock_Src (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, enum DM35425_Clock_Sources source)

  Set the clock source of the DAC.
- DM35425LIB_API int DM35425_Dac_Get_Clock_Src (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, enum DM35425_Clock_Sources ∗source)

  Get the clock source of the DAC.
- DM35425LIB_API int DM35425_Dac_Get_Clock_Div (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t ∗divider)

  Get the clock divider value.
- DM35425LIB_API int DM35425_Dac_Set_Clock_Div (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t divider)

  Set the clock divider value.
- DM35425LIB_API int DM35425_Dac_Set_Conversion_Rate (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t requested_rate, uint32_t ∗actual_rate)

  Set the conversion rate of this DAC.
- DM35425LIB_API int DM35425_Dac_Interrupt_Set_Config (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint16_t interrupt_src, int enable)

  Set the interrupt configuration for this DAC.
- DM35425LIB_API int DM35425_Dac_Interrupt_Get_Config (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint16_t ∗interrupt_ena)

  Get the interrupt configuration for this DAC.
- DM35425LIB_API int DM35425_Dac_Set_Start_Trigger (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint8_t trigger_value)

  Set the start trigger.
- DM35425LIB_API int DM35425_Dac_Set_Stop_Trigger (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint8_t trigger_value)

  Set the stop trigger.
- DM35425LIB_API int DM35425_Dac_Get_Start_Trigger (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint8_t ∗trigger_value)

  Get the start trigger.
- DM35425LIB_API int DM35425_Dac_Get_Stop_Trigger (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint8_t ∗trigger_value)

  Get the stop trigger.
- DM35425LIB_API int DM35425_Dac_Start (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block)

  Set the DAC Mode to Start.
- DM35425LIB_API int DM35425_Dac_Reset (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block)

  Set the DAC Mode to Reset.
- DM35425LIB_API int DM35425_Dac_Pause (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block)

  Set the DAC Mode to Pause.
- DM35425LIB_API int DM35425_Dac_Get_Mode_Status (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint8_t ∗mode_status)

  Get the Mode and Status of the DAC.

- DM35425LIB_API int DM35425_Dac_Get_Last_Conversion (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel, uint8_t ∗marker, int16_t ∗value)

    Get the value of the last conversion of the DAC.
- DM35425LIB_API int DM35425_Dac_Set_Last_Conversion (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel, uint8_t marker, int16_t value)

    Set a value to be converted by the DAC immediately.
- DM35425LIB_API int DM35425_Dac_Get_Conversion_Count (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t ∗value)

    Get a count of the number of conversions that DAC has executed.
- DM35425LIB_API int DM35425_Dac_Interrupt_Get_Status (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint16_t ∗value)

    Get a interrupt status register of the DAC.
- DM35425LIB_API int DM35425_Dac_Interrupt_Clear_Status (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint16_t value)

    Clear the interrupt status register of the DAC.
- DM35425LIB_API int DM35425_Dac_Set_Post_Stop_Conversion_Count (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t value)

    Set the number of conversions the DAC will make after a stop trigger.
- DM35425LIB_API int DM35425_Dac_Get_Post_Stop_Conversion_Count (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t ∗value)

    Get the number of conversions the DAC will make after a stop trigger.
- DM35425LIB_API int DM35425_Dac_Set_Clock_Source_Global (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, enum DM35425_Clock_Sources clock, enum DM35425_Dac_Clock_Events clock_driver)

    Set the source that will drive the global clock.
- DM35425LIB_API int DM35425_Dac_Channel_Setup (struct DM35425_Board_Descriptor ∗handle, struct DM35425_Function_Block ∗func_block, unsigned int channel, enum DM35425_Output_Ranges output_←range)

    Setup the selected DAC channel.
- DM35425LIB_API int DM35425_Dac_Channel_Reset (struct DM35425_Board_Descriptor ∗handle, struct DM35425_Function_Block ∗func_block, unsigned int channel)

    Reset the DAC channel, by writing all zeros to the front-end config.
- DM35425LIB_API int DM35425_Dac_Channel_Set_Marker_Config (struct DM35425_Board_Descriptor ∗handle, struct DM35425_Function_Block ∗func_block, unsigned int channel, uint8_t marker_enable)

    Set the configuration of the marker interrupts for this channel.
- DM35425LIB_API int DM35425_Dac_Channel_Get_Marker_Config (struct DM35425_Board_Descriptor ∗handle, struct DM35425_Function_Block ∗func_block, unsigned int channel, uint8_t ∗marker_enable)

    Get the configuration of the marker interrupts for this channel.
- DM35425LIB_API int DM35425_Dac_Channel_Get_Marker_Status (struct DM35425_Board_Descriptor ∗handle, struct DM35425_Function_Block ∗func_block, unsigned int channel, uint8_t ∗marker_status)

    Get the status of the marker interrupts for this channel.
- DM35425LIB_API int DM35425_Dac_Channel_Clear_Marker_Status (struct DM35425_Board_Descriptor ∗handle, struct DM35425_Function_Block ∗func_block, unsigned int channel, uint8_t marker_to_clear)

    Clear the marker interrupts for this channel.
- DM35425LIB_API int DM35425_Dac_Fifo_Channel_Write (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel, int32_t value)

    Write a value to the onboard FIFO.
- DM35425LIB_API int DM35425_Dac_Volts_To_Conv (enum DM35425_Output_Ranges output_range, float volts, int16_t ∗dac_conversion)

    Convert a value in volts to a DAC equivalent signed value.
- DM35425LIB_API int DM35425_Dac_Conv_To_Volts (enum DM35425_Output_Ranges output_range, int16_t conversion, float ∗volts)

    Convert a DAC conversion value to volts.

4.8.1 Detailed Description

DM35425_Dac_Library_Constants

4.8.2 Function Documentation

4.8.2.1 DM35425_Dac_Channel_Clear_Marker_Status()

```
DM35425LIB_API int DM35425_Dac_Channel_Clear_Marker_Status (
             struct DM35425_Board_Descriptor * handle,
             struct DM35425_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	

4.8.2.2 DM35425_Dac_Channel_Get_Marker_Config()

```
DM35425LIB_API int DM35425_Dac_Channel_Get_Marker_Config (
             struct DM35425_Board_Descriptor * handle,
             struct DM35425_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	

4.8.2.3 DM35425_Dac_Channel_Get_Marker_Status()

```
DM35425LIB_API int DM35425_Dac_Channel_Get_Marker_Status (
             struct DM35425_Board_Descriptor * handle,
             struct DM35425_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	

4.8.2.4 DM35425_Dac_Channel_Reset()

```
DM35425LIB_API int DM35425_Dac_Channel_Reset (
             struct DM35425_Board_Descriptor * handle,
             struct DM35425_Function_Block * func_block,
             unsigned int channel )
```

Reset the DAC channel, by writing all zeros to the front-end config.

Parameters

handle	Pointer to the board handle
func_block	Pointer to the function block we want to reset.
channel	The channel to reset.

Return values

0	Success.

Return values

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

4.8.2.5 DM35425_Dac_Channel_Set_Marker_Config()

```
DM35425LIB_API int DM35425_Dac_Channel_Set_Marker_Config (
             struct DM35425_Board_Descriptor * handle,
             struct DM35425_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	

4.8.2.6 DM35425_Dac_Channel_Setup()

```
DM35425LIB_API int DM35425_Dac_Channel_Setup (
             struct DM35425_Board_Descriptor * handle,
             struct DM35425_Function_Block * func_block,
             unsigned int channel,
             enum DM35425_Output_Ranges output_range )
```

Setup the selected DAC channel.

Parameters

handle	Pointer to the board handle
func_block	Pointer to the function block we want to change the output range on.
channel	The channel to set the output range of.
output_range	Enumerated value defining desired output range.

Return values

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

Referenced by DM35425_Setup_Dacs(), main(), and setup_dacs().

4.8.2.7 DM35425_Dac_Conv_To_Volts()

```
DM35425LIB_API int DM35425_Dac_Conv_To_Volts (
              enum DM35425_Output_Ranges output_range,
              int16_t conversion,
              float * volts )
```

Convert a DAC conversion value to volts.

Parameters

output_range	The output range the channel was set to.
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().

4.8.2.8 DM35425_Dac_Fifo_Channel_Write()

```
DM35425LIB_API int DM35425_Dac_Fifo_Channel_Write (
              struct DM35425_Board_Descriptor * handle,
              const struct DM35425_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.

Referenced by main().

4.8.2.9 DM35425_Dac_Get_Clock_Div()

```
DM35425LIB_API int DM35425_Dac_Get_Clock_Div (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_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.

4.8.2.10 DM35425_Dac_Get_Clock_Src()

```
DM35425LIB_API int DM35425_Dac_Get_Clock_Src (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_Function_Block * func_block,
            enum DM35425_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.

4.8.2.11 DM35425_Dac_Get_Conversion_Count()

```
DM35425LIB_API int DM35425_Dac_Get_Conversion_Count (
             struct DM35425_Board_Descriptor * handle,
             const struct DM35425_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.

4.8.2.12 DM35425_Dac_Get_Last_Conversion()

```
DM35425LIB_API int DM35425_Dac_Get_Last_Conversion (
             struct DM35425_Board_Descriptor * handle,
             const struct DM35425_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().

4.8.2.13 DM35425_Dac_Get_Mode_Status()

```
DM35425LIB_API int DM35425_Dac_Get_Mode_Status (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_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.

4.8.2.14 DM35425_Dac_Get_Post_Stop_Conversion_Count()

```
DM35425LIB_API int DM35425_Dac_Get_Post_Stop_Conversion_Count (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_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.

4.8.2.15 DM35425_Dac_Get_Start_Trigger()

```
DM35425LIB_API int DM35425_Dac_Get_Start_Trigger (
             struct DM35425_Board_Descriptor * handle,
             const struct DM35425_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.

4.8.2.16 DM35425_Dac_Get_Stop_Trigger()

```
DM35425LIB_API int DM35425_Dac_Get_Stop_Trigger (
             struct DM35425_Board_Descriptor * handle,
             const struct DM35425_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.

Parameters

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.

4.8.2.17 DM35425_Dac_Interrupt_Clear_Status()

```
DM35425LIB_API int DM35425_Dac_Interrupt_Clear_Status (
             struct DM35425_Board_Descriptor * handle,
             const struct DM35425_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.

4.8.2.18 DM35425_Dac_Interrupt_Get_Config()

```
DM35425LIB_API int DM35425_Dac_Interrupt_Get_Config (
             struct DM35425_Board_Descriptor * handle,
             const struct DM35425_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.

4.8.2.19 DM35425_Dac_Interrupt_Get_Status()

```
DM35425LIB_API int DM35425_Dac_Interrupt_Get_Status (
             struct DM35425_Board_Descriptor * handle,
             const struct DM35425_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.

4.8.2.20 DM35425_Dac_Interrupt_Set_Config()

```
DM35425LIB_API int DM35425_Dac_Interrupt_Set_Config (
             struct DM35425_Board_Descriptor * handle,
             const struct DM35425_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.

4.8.2.21 DM35425_Dac_Open()

```
DM35425LIB_API int DM35425_Dac_Open (
             struct DM35425_Board_Descriptor * handle,
             unsigned int number_of_type,
             struct DM35425_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 DM35425_Setup_Dacs(), main(), and setup_dacs().

4.8.2.22 DM35425_Dac_Pause()

```
DM35425LIB_API int DM35425_Dac_Pause (
             struct DM35425_Board_Descriptor * handle,
             const struct DM35425_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.

Return values

Non-Zero	Failure.

4.8.2.23 DM35425_Dac_Reset()

```
DM35425LIB_API int DM35425_Dac_Reset (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_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 DM35425_Setup_Dacs(), and main().

4.8.2.24 DM35425_Dac_Set_Clock_Div()

```
DM35425LIB_API int DM35425_Dac_Set_Clock_Div (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_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.

4.8.2.25 DM35425_Dac_Set_Clock_Source_Global()

```
DM35425LIB_API int DM35425_Dac_Set_Clock_Source_Global (
             struct DM35425_Board_Descriptor * handle,
             const struct DM35425_Function_Block * func_block,
             enum DM35425_Clock_Sources clock,
             enum DM35425_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.

4.8.2.26 DM35425_Dac_Set_Clock_Src()

```
DM35425LIB_API int DM35425_Dac_Set_Clock_Src (
             struct DM35425_Board_Descriptor * handle,
             const struct DM35425_Function_Block * func_block,
             enum DM35425_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(), and setup_dacs().

4.8.2.27 DM35425_Dac_Set_Conversion_Rate()

```
DM35425LIB_API int DM35425_Dac_Set_Conversion_Rate (
             struct DM35425_Board_Descriptor * handle,
             const struct DM35425_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().

4.8.2.28 DM35425_Dac_Set_Last_Conversion()

```
DM35425LIB_API int DM35425_Dac_Set_Last_Conversion (
             struct DM35425_Board_Descriptor * handle,
             const struct DM35425_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 DM35425_Setup_Dacs(), and main().

4.8.2.29 DM35425_Dac_Set_Post_Stop_Conversion_Count()

```
DM35425LIB_API int DM35425_Dac_Set_Post_Stop_Conversion_Count (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_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 main().

4.8.2.30 DM35425_Dac_Set_Start_Trigger()

```
DM35425LIB_API int DM35425_Dac_Set_Start_Trigger (
             struct DM35425_Board_Descriptor * handle,
             const struct DM35425_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(), and setup_dacs().

4.8.2.31 DM35425_Dac_Set_Stop_Trigger()

```
DM35425LIB_API int DM35425_Dac_Set_Stop_Trigger (
             struct DM35425_Board_Descriptor * handle,
             const struct DM35425_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(), and setup_dacs().

4.8.2.32 DM35425_Dac_Start()

```
DM35425LIB_API int DM35425_Dac_Start (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_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(), and setup_dacs().

4.8.2.33 DM35425_Dac_Volts_To_Conv()

```
DM35425LIB_API int DM35425_Dac_Volts_To_Conv (
            enum DM35425_Output_Ranges output_range,
            float volts,
            int16_t * dac_conversion )
```

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

Parameters

output_range	The output range this channel was set to.
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 DM35425_Setup_Dacs(), and main().

4.9 DM35425 DMA Public Library Constants

Macros

- #define DM35425_DMA_ACTION_CLEAR 0x00

  Register value for DMA clear action.
- #define DM35425_DMA_ACTION_GO 0x01

  Register value for DMA go action.
- #define DM35425_DMA_ACTION_PAUSE 0x02

  Register value for DMA pause action.
- #define DM35425_DMA_ACTION_HALT 0x03

  Register value for DMA halt action.
- #define DM35425_DMA_SETUP_DIRECTION_READ 0x04

  Register value to set DMA to READ direction.
- #define DM35425_DMA_SETUP_DIRECTION_WRITE 0x00

  Register value to set DMA to WRITE direction.
- #define DM35425_DMA_SETUP_DIRECTION_MASK 0x04

  Register value to set DMA to READ direction.
- #define DM35425_DMA_SETUP_IGNORE_USED 0x08

  Register value to tell DMA to ignore used buffers.
- #define DM35425_DMA_SETUP_NOT_IGNORE_USED 0x00

  Register value to tell DMA to not ignore used buffers.
- #define DM35425_DMA_SETUP_IGNORE_USED_MASK 0x08

  Bit mask for Ignore Used bit in setup register.
- #define DM35425_DMA_SETUP_INT_ENABLE 0x01

  Register value to enabled interrupts in the setup register.
- #define DM35425_DMA_SETUP_INT_DISABLE 0x00

  Register value to disable interrupts in the setup register.
- #define DM35425_DMA_SETUP_INT_MASK 0x01

  Bit mask for the interrupt bit in the setup register.
- #define DM35425_DMA_SETUP_ERR_INT_ENABLE 0x02

  Register value to enable the error interrupt.
- #define DM35425_DMA_SETUP_ERR_INT_DISABLE 0x00

  Register value to disable the error interrupt.
- #define DM35425_DMA_SETUP_ERR_INT_MASK 0x02

  Bit mask for the error interrupt bit in the setup register.
- #define DM35425_DMA_STATUS_CLEAR 0x00

  Register value to write to status registers to clear them.
- #define DM35425_DMA_CTRL_CLEAR 0x00

  Register value to write to control register to clear it.
- #define DM35425_DMA_BUFFER_STATUS_CLEAR 0x00

  Register value to write to the buffer status register to clear it.
- #define DM35425_DMA_BUFFER_CTRL_CLEAR 0x00

  Register value to write to the buffer control register to clear it.
- #define DM35425_DMA_BUFFER_STATUS_USED_MASK 0x01

  Bit mask for the used buffer bit in the buffer status register.
- #define DM35425_DMA_BUFFER_STATUS_TERM_MASK 0x02

  Bit mask for the terminated buffer bit in the buffer status register.
- #define DM35425_DMA_BUFFER_CTRL_VALID 0x01

  Register value to write to buffer control register to mark it as valid.

- #define DM35425_DMA_BUFFER_CTRL_HALT 0x02

  Register value to write to buffer control register to tell DMA to halt after processing this buffer.
- #define DM35425_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 DM35425_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 DM35425_DMA_BUFFER_CTRL_PAUSE 0x10

  Register value to write to buffer control register to tell DMA to pause after processing this buffer.
- #define DM35425_DMA_CTRL_BLOCK_SIZE 0x10

  Constant value indicating DMA control block size.
- #define DM35425_DMA_BUFFER_CTRL_BLOCK_SIZE 0x10

  Constant value indicating DMA buffer control block size.
- #define DM35425_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 DM35425_Fifo_States { DM35425_FIFO_UNKNOWN, DM35425_FIFO_EMPTY, DM35425_FIFO_FULL, DM35425_FIFO_HAS_DATA }

  Descriptions of the possible states the FIFO might be in.

4.9.1 Detailed Description

4.9.2 Enumeration Type Documentation

4.9.2.1 DM35425_Fifo_States

```
enum DM35425_Fifo_States
```

Descriptions of the possible states the FIFO might be in.

Enumerator

DM35425_FIFO_UNKNOWN	State of FIFO is unknown.
DM35425_FIFO_EMPTY	FIFO is empty.
DM35425_FIFO_FULL	FIFO is full
DM35425_FIFO_HAS_DATA	FIFO is between empty and full

Definition at line 237 of file dm35425_dma_library.h.

4.10 DM35425 DMA Public Library Functions

Functions

- DM35425LIB_API int DM35425_Dma_Start (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel)

  Start the DMA.
- DM35425LIB_API int DM35425_Dma_Stop (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel)

  Stop the DMA.
- DM35425LIB_API int DM35425_Dma_Pause (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel)

  Pause the DMA.
- DM35425LIB_API int DM35425_Dma_Clear (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel)

  Clear the DMA.
- DM35425LIB_API int DM35425_Dma_Get_Fifo_Counts (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel, uint16_t ∗write_count, uint16_t ∗read↩_count)

  Get the Read and Write FIFO count values.
- DM35425LIB_API int DM35425_Dma_Get_Fifo_State (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel, enum DM35425_Fifo_States ∗state)

  Get the state of the FIFO.
- DM35425LIB_API int DM35425_Dma_Configure_Interrupts (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel, int enable, int error_enable)

  Configure the interrupts for the DMA channel.
- DM35425LIB_API int DM35425_Dma_Get_Interrupt_Configuration (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel, int ∗enable, int ∗error↩_enable)

  Get the configuration of the interrupts for the DMA channel.
- DM35425LIB_API int DM35425_Dma_Setup (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_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.
- DM35425LIB_API int DM35425_Dma_Setup_Set_Direction (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel, int direction)

  Set the direction of the DMA, read or write.
- DM35425LIB_API int DM35425_Dma_Setup_Set_Used (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_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.
- DM35425LIB_API int DM35425_Dma_Get_Errors (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_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.
- DM35425LIB_API int DM35425_Dma_Status (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_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.
- DM35425LIB_API int DM35425_Dma_Get_Current_Buffer_Count (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_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.

- DM35425LIB_API int DM35425_Dma_Check_For_Error (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_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.
- DM35425LIB_API int DM35425_Dma_Buffer_Setup (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel, unsigned int buffer, uint8_t ctrl)

    Setup the DMA buffer for use.
- DM35425LIB_API int DM35425_Dma_Buffer_Status (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_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.
- DM35425LIB_API int DM35425_Dma_Check_Buffer_Used (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_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 DM35425_Dma_Find_Interrupt (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_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 DM35425_Dma_Clear_Interrupt (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_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 DM35425_Dma_Reset_Buffer (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel, unsigned int buffer)

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

4.10.1 Detailed Description

DM35425_Dma_Library_Constants

4.10.2 Function Documentation

4.10.2.1 DM35425_Dma_Buffer_Setup()

```
DM35425LIB_API int DM35425_Dma_Buffer_Setup (
            struct DM35425_Board_Descriptor ∗ handle,
            const struct DM35425_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(), and setup_dacs().

4.10.2.2 DM35425_Dma_Buffer_Status()

```
DM35425LIB_API int DM35425_Dma_Buffer_Status (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_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 output_dma_buffer_status().

4.10.2.3 DM35425_Dma_Check_Buffer_Used()

```
DM35425LIB_API int DM35425_Dma_Check_Buffer_Used (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_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.

Referenced by ISR().

4.10.2.4 DM35425_Dma_Check_For_Error()

```
DM35425LIB_API int DM35425_Dma_Check_For_Error (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_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.

Return values

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

Referenced by ISR().

4.10.2.5 DM35425_Dma_Clear()

```
DM35425LIB_API int DM35425_Dma_Clear (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_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().

4.10.2.6 DM35425_Dma_Clear_Interrupt()

```
int DM35425_Dma_Clear_Interrupt (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_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(), and main().

4.10.2.7 DM35425_Dma_Configure_Interrupts()

```
DM35425LIB_API int DM35425_Dma_Configure_Interrupts (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_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().

4.10.2.8 DM35425_Dma_Find_Interrupt()

```
int DM35425_Dma_Find_Interrupt (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_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().

4.10.2.9 DM35425_Dma_Get_Current_Buffer_Count()

```
DM35425LIB_API int DM35425_Dma_Get_Current_Buffer_Count (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_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.

Referenced by main().

4.10.2.10 DM35425_Dma_Get_Errors()

```
DM35425LIB_API int DM35425_Dma_Get_Errors (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_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.

Referenced by main().

4.10.2.11 DM35425_Dma_Get_Fifo_Counts()

```
DM35425LIB_API int DM35425_Dma_Get_Fifo_Counts (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_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.

Referenced by print_fifo_status().

4.10.2.12 DM35425_Dma_Get_Fifo_State()

```
DM35425LIB_API int DM35425_Dma_Get_Fifo_State (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_Function_Block * func_block,
            unsigned int channel,
            enum DM35425_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.

4.10.2.13 DM35425_Dma_Get_Interrupt_Configuration()

```
DM35425LIB_API int DM35425_Dma_Get_Interrupt_Configuration (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_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

Return values

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

4.10.2.14 DM35425_Dma_Pause()

```
DM35425LIB_API int DM35425_Dma_Pause (
             struct DM35425_Board_Descriptor * handle,
             const struct DM35425_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.

Referenced by main().

4.10.2.15 DM35425_Dma_Reset_Buffer()

```
int DM35425_Dma_Reset_Buffer (
             struct DM35425_Board_Descriptor * handle,
             const struct DM35425_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(), and main().

4.10.2.16 DM35425_Dma_Setup()

```
DM35425LIB_API int DM35425_Dma_Setup (
             struct DM35425_Board_Descriptor * handle,
             const struct DM35425_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(), and setup_dacs().

4.10.2.17 DM35425_Dma_Setup_Set_Direction()

```
DM35425LIB_API int DM35425_Dma_Setup_Set_Direction (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_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.

4.10.2.18 DM35425_Dma_Setup_Set_Used()

```
DM35425LIB_API int DM35425_Dma_Setup_Set_Used (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_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.

4.10.2.19 DM35425_Dma_Start()

```
DM35425LIB_API int DM35425_Dma_Start (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_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().

4.10.2.20 DM35425_Dma_Status()

```
DM35425LIB_API int DM35425_Dma_Status (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_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(), output_channel_status(), and print_fifo_status().

4.10.2.21 DM35425_Dma_Stop()

```
DM35425LIB_API int DM35425_Dma_Stop (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_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.

4.11 DM35425 Driver Constants

Macros

- #define DM35425_NAME_LENGTH 200

  DM35425 Max possible board name length.
- #define DM35425_PCI_NUM_REGIONS PCI_ROM_RESOURCE

  Number of standard PCI regions.
- #define DM35425_INT_QUEUE_SIZE 256

  Number of interrupts to hold in a queue for processing.

4.11.1 Detailed Description

4.12 DM35425 Driver Enumerations

Enumerations

- enum dm35425_pci_region_access_dir { DM35425_PCI_REGION_ACCESS_READ = 0, DM35425_PCI_REGION_ACCESS_
  }

  Direction of access to standard PCI region.

4.12.1 Detailed Description

DM35425_Driver_Constants

4.12.2 Enumeration Type Documentation

4.12.2.1 dm35425_pci_region_access_dir

```
enum dm35425_pci_region_access_dir
```

Direction of access to standard PCI region.

Enumerator

DM35425_PCI_REGION_ACCESS_READ	Read from the region
DM35425_PCI_REGION_ACCESS_WRITE	Write to the region

Definition at line 80 of file dm35425_driver.h.

4.13 DM35425 Driver Structures

Data Structures

- struct dm35425_pci_region

  DM35425 PCI region descriptor. This structure holds information about one of a device's PCI memory regions.
- struct dm35425_dma_descriptor

  DM35425 DMA descriptor. This structure holds information about a single DMA buffer.
- struct dm35425_device_descriptor

  DM35425 Device Descriptor. The identifying info for this particular board.

Variables

- static struct file_operations dm35425_file_ops

  Placeholder protoype for file ops struct.

4.13.1 Detailed Description

DM35425_Driver_Enumerations

4.14 DM35425 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.

4.14.1 Detailed Description

4.14.2 Enumeration Type Documentation

4.14.2.1 Help_Options

```
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.

Enumerator

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 dm35425_examples.h.

4.15 DM35425 Global Clocking Library Constants

4.15.1 Detailed Description

4.16 DM35425 Global Clocking Library Public Functions

Functions

- DM35425LIB_API int DM35425_Ext_Clocking_Open (struct DM35425_Board_Descriptor ∗handle, unsigned int number_of_type, struct DM35425_Function_Block ∗func_block)

    Open the Global Clocking functional block, making it available for operations.
- DM35425LIB_API int DM35425_Ext_Clocking_Get_In (struct DM35425_Board_Descriptor ∗handle, struct DM35425_Function_Block ∗func_block, uint8_t ∗clk_curr_val)

    Get the current value on the external clocking pins.
- DM35425LIB_API int DM35425_Ext_Clocking_Get_Gate_In (struct DM35425_Board_Descriptor ∗handle, struct DM35425_Function_Block ∗func_block, uint8_t ∗gate_curr_val)

    Get the current value on the external clocking gate pins.
- DM35425LIB_API int DM35425_Ext_Clocking_Get_Dir (struct DM35425_Board_Descriptor ∗handle, struct DM35425_Function_Block ∗func_block, uint8_t ∗dir)

    Get the current value of the external clocking pin direction.
- DM35425LIB_API int DM35425_Ext_Clocking_Set_Dir (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint8_t dir)

    Set the current value of the external clocking pin direction.
- DM35425LIB_API int DM35425_Ext_Clocking_Get_Edge (struct DM35425_Board_Descriptor ∗handle, struct DM35425_Function_Block ∗func_block, uint8_t ∗edge_detect)

    Get the current value of the external clocking edge detect.
- DM35425LIB_API int DM35425_Ext_Clocking_Set_Edge (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint8_t edge_detect)

    Set the current value of the external clocking edge detect.
- DM35425LIB_API int DM35425_Ext_Clocking_Get_Pulse_Width (struct DM35425_Board_Descriptor ∗handle, struct DM35425_Function_Block ∗func_block, enum DM35425_Clock_Sources clock_src, uint8_t ∗pulse_width)

    Get the pulse width setting for a specific external clock.
- DM35425LIB_API int DM35425_Ext_Clocking_Set_Pulse_Width (struct DM35425_Board_Descriptor ∗handle, struct DM35425_Function_Block ∗func_block, enum DM35425_Clock_Sources clock_src, uint8_t pulse_width)

    Set the pulse width setting for a specific external clock.
- DM35425LIB_API int DM35425_Ext_Clocking_Get_Method (struct DM35425_Board_Descriptor ∗handle, struct DM35425_Function_Block ∗func_block, enum DM35425_Clock_Sources clock_src, enum DM35425↩_Ext_Clocking_Method ∗clocking_method)

    Get the setting for a specific external clock.
- DM35425LIB_API int DM35425_Ext_Clocking_Set_Method (struct DM35425_Board_Descriptor ∗handle, struct DM35425_Function_Block ∗func_block, enum DM35425_Clock_Sources clock_src, enum DM35425↩_Ext_Clocking_Method clocking_method)

    Set the setting for a specific external clock.

4.16.1 Detailed Description

DM35425_Ext_Clocking_Library_Constants

4.16.2 Function Documentation

4.16.2.1 DM35425_Ext_Clocking_Get_Dir()

```
DM35425LIB_API int DM35425_Ext_Clocking_Get_Dir (
            struct DM35425_Board_Descriptor * handle,
            struct DM35425_Function_Block * func_block,
            uint8_t * dir )
```

Get the current value of the external clocking pin direction.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the returned function block descriptor.
dir	Pointer to the returned value of the external clocking pin directions. (0 = input, 1 = output)

Return values

0	Success.
Non-Zero	Failure.

4.16.2.2 DM35425_Ext_Clocking_Get_Edge()

```
DM35425LIB_API int DM35425_Ext_Clocking_Get_Edge (
            struct DM35425_Board_Descriptor * handle,
            struct DM35425_Function_Block * func_block,
            uint8_t * edge_detect )
```

Get the current value of the external clocking edge detect.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the returned function block descriptor.
edge_detect	Pointer to the returned value of the per-clock edge detect settings. (0 = rising edge, 1 = falling edge)

Return values

0	Success.
Non-Zero	Failure.

4.16.2.3 DM35425_Ext_Clocking_Get_Gate_In()

```
DM35425LIB_API int DM35425_Ext_Clocking_Get_Gate_In (
            struct DM35425_Board_Descriptor * handle,
            struct DM35425_Function_Block * func_block,
            uint8_t * gate_curr_val )
```

Get the current value on the external clocking gate pins.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the returned function block descriptor.
gate_curr_val	Pointer to the returned current value of the external clocking gate pins.

Return values

0	Success.
Non-Zero	Failure.

4.16.2.4 DM35425_Ext_Clocking_Get_In()

```
DM35425LIB_API int DM35425_Ext_Clocking_Get_In (
            struct DM35425_Board_Descriptor * handle,
            struct DM35425_Function_Block * func_block,
            uint8_t * clk_curr_val )
```

Get the current value on the external clocking pins.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the returned function block descriptor.
clk_curr_val	Pointer to the returned current value on the external clocking pins.

Return values

0	Success.
Non-Zero	Failure.

4.16.2.5 DM35425_Ext_Clocking_Get_Method()

```
DM35425LIB_API int DM35425_Ext_Clocking_Get_Method (
            struct DM35425_Board_Descriptor * handle,
            struct DM35425_Function_Block * func_block,
            enum DM35425_Clock_Sources clock_src,
            enum DM35425_Ext_Clocking_Method * clocking_method )
```

Get the setting for a specific external clock.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the returned function block descriptor.
clock_src	Which external clock the pulse width is associated with.
clocking_method	Pointer to the returned value for the setting for this external clock.

Return values

0	Success.
Non-Zero	Failure.

4.16.2.6 DM35425_Ext_Clocking_Get_Pulse_Width()

```
DM35425LIB_API int DM35425_Ext_Clocking_Get_Pulse_Width (
             struct DM35425_Board_Descriptor * handle,
             struct DM35425_Function_Block * func_block,
             enum DM35425_Clock_Sources clock_src,
             uint8_t * pulse_width )
```

Get the pulse width setting for a specific external clock.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the returned function block descriptor.
clock_src	Which external clock the pulse width is associated with.
pulse_width	Pointer to the returned value for the pulse width.

Return values

0	Success.
Non-Zero	Failure.

4.16.2.7 DM35425_Ext_Clocking_Open()

```
DM35425LIB_API int DM35425_Ext_Clocking_Open (
             struct DM35425_Board_Descriptor * handle,
             unsigned int number_of_type,
             struct DM35425_Function_Block * func_block )
```

Open the Global Clocking functional block, making it available for operations.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
number_of_type	Ordinal value of external clock function block to open (0th, 1st, etc).
func_block	Pointer to the returned function block descriptor.

Return values

0	Success.
Non-Zero	Failure.

Referenced by main().

4.16.2.8 DM35425_Ext_Clocking_Set_Dir()

```
DM35425LIB_API int DM35425_Ext_Clocking_Set_Dir (
             struct DM35425_Board_Descriptor * handle,
             const struct DM35425_Function_Block * func_block,
             uint8_t dir )
```

Set the current value of the external clocking pin direction.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the returned function block descriptor.
dir	Value of the external clocking pin directions. (0 = input, 1 = output)

Return values

0	Success.
Non-Zero	Failure.

Referenced by main().

4.16.2.9 DM35425_Ext_Clocking_Set_Edge()

```
DM35425LIB_API int DM35425_Ext_Clocking_Set_Edge (
             struct DM35425_Board_Descriptor * handle,
             const struct DM35425_Function_Block * func_block,
             uint8_t edge_detect )
```

Set the current value of the external clocking edge detect.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the returned function block descriptor.
edge_detect	Value of the per-clock edge detect settings. (0 = rising edge, 1 = falling edge)

Return values

0	Success.
Non-Zero	Failure.

Referenced by main().

4.16.2.10 DM35425_Ext_Clocking_Set_Method()

```
DM35425LIB_API int DM35425_Ext_Clocking_Set_Method (
             struct DM35425_Board_Descriptor * handle,
             struct DM35425_Function_Block * func_block,
             enum DM35425_Clock_Sources clock_src,
             enum DM35425_Ext_Clocking_Method clocking_method )
```

Set the setting for a specific external clock.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the returned function block descriptor.
clock_src	Which external clock the pulse width is associated with.
clocking_method	Enumerated value for the setting for this external clock.

Return values

0	Success.
Non-Zero	Failure.

Referenced by main().

4.16.2.11 DM35425_Ext_Clocking_Set_Pulse_Width()

```
DM35425LIB_API int DM35425_Ext_Clocking_Set_Pulse_Width (
             struct DM35425_Board_Descriptor * handle,
             struct DM35425_Function_Block * func_block,
             enum DM35425_Clock_Sources clock_src,
             uint8_t pulse_width )
```

Set the pulse width setting for a specific external clock.

Parameters

handle	Address of the handle pointer, which will contain the device descriptor.
func_block	Pointer to the returned function block descriptor.
clock_src	Which external clock the pulse width is associated with.
pulse_width	Value for the pulse width.

Return values

0	Success.
Non-Zero	Failure.

Referenced by main().

4.17 DM35425 Board Macros

Macros

- #define CLK_40MHZ 40000000

4.17.1 Detailed Description

4.17.2 Macro Definition Documentation

4.17.2.1 CLK_40MHZ

```
#define CLK_40MHZ 40000000
```

This is the standard clock of the DM35x18 boards

Definition at line 52 of file dm35425_gbc_library.h.

4.18 DM35425 Board Library Public Functions

Functions

- DM35425LIB_API int DM35425_Gbc_Board_Reset (struct DM35425_Board_Descriptor ∗handle)

  Write the reset value to the correct register to initiate a board-level reset.
- DM35425LIB_API int DM35425_Gbc_Ack_Interrupt (struct DM35425_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.
- DM35425LIB_API int DM35425_Function_Block_Open (struct DM35425_Board_Descriptor ∗handle, unsigned int number, struct DM35425_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.
- DM35425LIB_API int DM35425_Function_Block_Open_Module (struct DM35425_Board_Descriptor ∗handle, uint32_t fb_type, unsigned int number_of_type, struct DM35425_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.
- DM35425LIB_API int DM35425_Gbc_Get_Format (struct DM35425_Board_Descriptor ∗handle, uint8_↩
t ∗format_id)

  Get the format ID of the board.
- DM35425LIB_API int DM35425_Gbc_Get_Revision (struct DM35425_Board_Descriptor ∗handle, uint8_↩
t ∗rev)

  Get the PDP revision number of the board.
- DM35425LIB_API int DM35425_Gbc_Get_Pdp_Number (struct DM35425_Board_Descriptor ∗handle, uint32_t ∗pdp_num)

  Get PDP Number of the board.
- DM35425LIB_API int DM35425_Gbc_Get_Fpga_Build (struct DM35425_Board_Descriptor ∗handle, uint32_t ∗fpga_build)

  Get the FPGA Build number of the board.
- DM35425LIB_API int DM35425_Gbc_Get_Sys_Clock_Freq (struct DM35425_Board_Descriptor ∗handle, uint32_t ∗clock_freq, int ∗is_std_clk)

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

4.18.1 Detailed Description

DM35425_Board_Macros

4.18.2 Function Documentation

4.18.2.1 DM35425_Function_Block_Open()

```
DM35425LIB_API int DM35425_Function_Block_Open (
              struct DM35425_Board_Descriptor * handle,
              unsigned int number,
              struct DM35425_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().

4.18.2.2 DM35425_Function_Block_Open_Module()

```
DM35425LIB_API int DM35425_Function_Block_Open_Module (
             struct DM35425_Board_Descriptor * handle,
             uint32_t fb_type,
             unsigned int number_of_type,
             struct DM35425_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 dm35425_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.

4.18.2.3 DM35425_Gbc_Ack_Interrupt()

```
DM35425LIB_API int DM35425_Gbc_Ack_Interrupt (
             struct DM35425_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().

4.18.2.4 DM35425_Gbc_Board_Reset()

```
DM35425LIB_API int DM35425_Gbc_Board_Reset (
            struct DM35425_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().

4.18.2.5 DM35425_Gbc_Get_Format()

```
DM35425LIB_API int DM35425_Gbc_Get_Format (
            struct DM35425_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.

4.18.2.6 DM35425_Gbc_Get_Fpga_Build()

```
DM35425LIB_API int DM35425_Gbc_Get_Fpga_Build (
             struct DM35425_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().

4.18.2.7 DM35425_Gbc_Get_Pdp_Number()

```
DM35425LIB_API int DM35425_Gbc_Get_Pdp_Number (
             struct DM35425_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().

4.18.2.8 DM35425_Gbc_Get_Revision()

```
DM35425LIB_API int DM35425_Gbc_Get_Revision (
            struct DM35425_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().

4.18.2.9 DM35425_Gbc_Get_Sys_Clock_Freq()

```
DM35425LIB_API int DM35425_Gbc_Get_Sys_Clock_Freq (
            struct DM35425_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.

4.19 DM35425 Ioctl macros

Macros

- #define DM35425_IOCTL_MAGIC 'D'

  Unique 8-bit value used to generate unique ioctl() request codes.
- #define DM35425_IOCTL_REQUEST_BASE 0x00

  First ioctl() request number.
- #define DM35425_IOCTL_REGION_READ

  ioctl() request code for reading from a PCI region
- #define DM35425_IOCTL_REGION_WRITE

  ioctl() request code for writing to a PCI region
- #define DM35425_IOCTL_REGION_MODIFY

  ioctl() request code for PCI region read/modify/write
- #define DM35425_IOCTL_DMA_FUNCTION

  ioctl() request code for DMA function
- #define DM35425_IOCTL_WAKEUP

  ioctl() request code for User ISR thread wake up
- #define DM35425_IOCTL_INTERRUPT_GET

  ioctl() request code to retrieve interrupt status information

4.19.1 Detailed Description

4.20 DM35425 Board Access Public Library Functions

Data Structures

- struct DM35425_Board_Descriptor

  DM35425 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 DM35425_Dma_Initialize (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_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 DM35425_Dma_Read (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_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 DM35425_Dma_Write (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_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 DM35425_General_RemoveISR (struct DM35425_Board_Descriptor ∗handle)

  Remove the ISR from the system interrupt.
- void ∗ DM35425_General_WaitForInterrupt (void ∗ptr)

  Loop/Poll and wait for an interrupt to happen, then take action.
- int DM35425_General_InstallISR (struct DM35425_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 DM35425_General_SetISRPriority (struct DM35425_Board_Descriptor ∗handle, int priority)

  Set the priority of the user ISR thread.

4.20.1 Detailed Description

4.20.2 Function Documentation

4.20.2.1 DM35425_Dma_Initialize()

```
int DM35425_Dma_Initialize (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_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(), and setup_dacs().

4.20.2.2 DM35425_Dma_Read()

```
int DM35425_Dma_Read (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_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(), and main().

4.20.2.3 DM35425_Dma_Write()

```
int DM35425_Dma_Write (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_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().

4.20.2.4 DM35425_General_InstallISR()

```
int DM35425_General_InstallISR (
            struct DM35425_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().

4.20.2.5 DM35425_General_RemoveISR()

```
int DM35425_General_RemoveISR (
            struct DM35425_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().

4.20.2.6 DM35425_General_SetISRPriority()

```
int DM35425_General_SetISRPriority (
            struct DM35425_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.

4.20.2.7 DM35425_General_WaitForInterrupt()

```
void* DM35425_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.

4.21 DM35425 Register Offsets

Macros

- #define DM35425_OFFSET_GBC_FORMAT 0x00

  Offset to General Board Control (BAR0) Format ID register.
- #define DM35425_OFFSET_GBC_REV 0x01

  Offset to General Board Control (BAR0) Format ID register.
- #define DM35425_OFFSET_GBC_END_INTERRUPT 0x02

  Offset to General Board Control (BAR0) EOI (End of Interrupt) register.
- #define DM35425_OFFSET_GBC_BOARD_RESET 0x03

  Offset to General Board Control (BAR0) Board Reset register.
- #define DM35425_OFFSET_GBC_PDP_NUMBER 0x04

  Offset to General Board Control (BAR0) PDP Number register.
- #define DM35425_OFFSET_GBC_FPGA_BUILD 0x08

  Offset to General Board Control (BAR0) FPGA Build register.
- #define DM35425_OFFSET_GBC_SYS_CLK_FREQ 0x0c

  Offset to General Board Control (BAR0) System Clock register.
- #define DM35425_OFFSET_GBC_IRQ_STATUS 0x10

  Offset to General Board Control (BAR0) IRQ Status register. Each bit corresponds to a function block.
- #define DM35425_OFFSET_GBC_DMA_IRQ_STATUS 0x18

  Offset to General Board Control (BAR0) DMA IRQ Status register. Each bit corresponds to a function block.
- #define DM35425_OFFSET_GBC_FB_START 0x20

  Offset to the beginning of the Function Blocks section of the GBC.
- #define DM35425_GBC_FB_BLK_SIZE 0x10

  Size of the function block entries in the GBC.
- #define DM35425_OFFSET_GBC_FB_ID 0x00

  Offset to Function Block ID, from the start of the function block section.
- #define DM35425_FB_ID_TYPE_MASK 0x0000FFFF

  Bit mask for TYPE portion of FB ID.
- #define DM35425_FB_ID_SUBTYPE_MASK 0x00FF0000

  Bit mask for SUBTYPE portion of FB ID.
- #define DM35425_FB_ID_TYPE_REV_MASK 0xFF000000

  Bit mask for TYPE REV portion of FB ID.
- #define DM35425_OFFSET_GBC_FB_OFFSET 0x04

  Offset to the FB Offset in the GBC, from the start of the FB data block.
- #define DM35425_OFFSET_GBC_FB_DMA_OFFSET 0x08

  Offset to the FB DMA Offset in the GBC, from the start of the FB data block.
- #define DM35425_OFFSET_DMA_ACTION 0x00

  Offset to the DMA Action Register (BAR2)
- #define DM35425_OFFSET_DMA_SETUP 0x01

  Offset to the DMA Setup Register (BAR2)
- #define DM35425_OFFSET_DMA_STAT_OVERFLOW 0x02

  Offset to the DMA Status (Overflow) Register (BAR2)
- #define DM35425_OFFSET_DMA_STAT_UNDERFLOW 0x03

  Offset to the DMA Status (Underflow) Register (BAR2)
- #define DM35425_OFFSET_DMA_CURRENT_COUNT 0x04

  Offset to the DMA Current Count Register (BAR2)
- #define DM35425_OFFSET_DMA_CURRENT_BUFFER 0x07

  Offset to the DMA Current Buffer Register (BAR2)

- #define DM35425_OFFSET_DMA_WR_FIFO_CNT 0x08

  Offset to the DMA Write FIFO Count Register (BAR2)
- #define DM35425_OFFSET_DMA_RD_FIFO_CNT 0x0A

  Offset to the DMA Read FIFO Count Register (BAR2)
- #define DM35425_OFFSET_DMA_STAT_USED 0x0C

  Offset to the DMA Status (Used) Register (BAR2)
- #define DM35425_OFFSET_DMA_STAT_INVALID 0x0D

  Offset to the DMA Status (Invalid) Register (BAR2)
- #define DM35425_OFFSET_DMA_STAT_COMPLETE 0x0E

  Offset to the DMA Status (Complete) Register (BAR2)
- #define DM35425_OFFSET_DMA_LAST_ACTION 0x0F

  Offset to the DMA Last Action Register (BAR2)
- #define DM35425_OFFSET_DMA_BUFF_START 0x10

  Offset to the start of the buffer control section (BAR2)
- #define DM35425_OFFSET_DMA_BUFFER_STAT 0x02

  Offset to the buffer status register, from the start of the buffer control section (BAR2)
- #define DM35425_OFFSET_DMA_BUFFER_CTRL 0x03

  Offset to the buffer control register, from the start of the buffer control section (BAR2)
- #define DM35425_OFFSET_DMA_BUFFER_SIZE 0x04

  Offset to the buffer size register, from the start of the buffer control section (BAR2)
- #define DM35425_OFFSET_DMA_BUFFER_ADDRESS 0x08

  Offset to the buffer address register, from the start of the buffer control section (BAR2)
- #define DM35425_OFFSET_FB_DMA_CHANNELS 0x06

  Offset to the DMA Channels count of the function block (BAR2)
- #define DM35425_OFFSET_FB_DMA_BUFFERS 0x07

  Offset to the DMA buffers count of the function block (BAR2)
- #define DM35425_OFFSET_FB_CTRL_START 0x08

  Offset to the beginning of the Function Block control section in BAR2.
- #define DM35425_OFFSET_ADC_MODE_STATUS 0x00

  Offset to the ADC Mode-Status register, from the start of the ADC control section.
- #define DM35425_OFFSET_ADC_CLK_SRC 0x01

  Offset to the ADC Clock Source register, from the start of the ADC control section.
- #define DM35425_OFFSET_ADC_START_TRIG 0x02

  Offset to the ADC Start Trigger register, from the start of the ADC control section.
- #define DM35425_OFFSET_ADC_STOP_TRIG 0x03

  Offset to the ADC Stop Trigger register, from the start of the ADC control section.
- #define DM35425_OFFSET_ADC_CLK_DIV 0x04

  Offset to the ADC Clock Divider register, from the start of the ADC control section.
- #define DM35425_OFFSET_ADC_CLK_DIV_COUNTER 0x08

  Offset to the ADC Clock Divider Counter register, from the start of the ADC control section.
- #define DM35425_OFFSET_ADC_PRE_CAPT_COUNT 0x0c

  Offset to the ADC Pre-Start Capture Count register, from the start of the ADC control section.
- #define DM35425_OFFSET_ADC_POST_CAPT_COUNT 0x10

  Offset to the ADC Post-Stop Capture Count register, from the start of the ADC control section.
- #define DM35425_OFFSET_ADC_SAMPLE_COUNT 0x14

  Offset to the ADC Sample Count register, from the start of the ADC control section.
- #define DM35425_OFFSET_ADC_INT_ENABLE 0x18

  Offset to the ADC Interrupt Enable register, from the start of the ADC control section.
- #define DM35425_OFFSET_ADC_INT_STAT 0x1e

  Offset to the ADC Interrupt Status register, from the start of the ADC control section.
- #define DM35425_OFFSET_ADC_CLK_BUS2 0x22

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

- #define DM35425_OFFSET_ADC_CLK_BUS3 0x23

  Offset to the ADC Clock Bus 3 register, from the start of the ADC control section.
- #define DM35425_OFFSET_ADC_CLK_BUS4 0x24

  Offset to the ADC Clock Bus 4 register, from the start of the ADC control section.
- #define DM35425_OFFSET_ADC_CLK_BUS5 0x25

  Offset to the ADC Clock Bus 5 register, from the start of the ADC control section.
- #define DM35425_OFFSET_ADC_CLK_BUS6 0x26

  Offset to the ADC Clock Bus 6 register, from the start of the ADC control section.
- #define DM35425_OFFSET_ADC_CLK_BUS7 0x27

  Offset to the ADC Clock Bus 7 register, from the start of the ADC control section.
- #define DM35425_OFFSET_ADC_AD_CONFIG 0x28

  Offset to the ADC AD Config register, from the start of the ADC control section.
- #define DM35425_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 DM35425_ADC_CHAN_CTRL_BLK_SIZE 0x18

  Constant size of ADC channel section in function block.
- #define DM35425_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 DM35425_OFFSET_ADC_CHAN_DATA_COUNT 0x04

  Offset to the Channel FIFO Data count register, from the start of the ADC channel control section.
- #define DM35425_OFFSET_ADC_CHAN_FILTER 0x09

  Offset to the Channel Filter register, from the start of the ADC channel control section.
- #define DM35425_OFFSET_ADC_CHAN_INTR_STAT 0x0a

  Offset to the Channel Interrupt Status register, from the start of the ADC channel control section.
- #define DM35425_OFFSET_ADC_CHAN_INTR_ENABLE 0x0b

  Offset to the Channel Interrupt Enable register, from the start of the ADC channel control section.
- #define DM35425_OFFSET_ADC_CHAN_LOW_THRESHOLD 0x0c

  Offset to the Channel Low Threshold register, from the start of the ADC channel control section.
- #define DM35425_OFFSET_ADC_CHAN_HIGH_THRESHOLD 0x10

  Offset to the Channel High Threshold register, from the start of the ADC channel control section.
- #define DM35425_OFFSET_ADC_CHAN_LAST_SAMPLE 0x14

  Offset to the Channel Last Sample register, from the start of the ADC channel control section.
- #define DM35425_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 DM35425_ADC_FIFO_CTRL_BLK_SIZE 0x4

  Constant size of ADC FIFO section in function block.
- #define DM35425_OFFSET_FB_ADC_FIFO 0x0334

  Offset to the FIFO for non-DMA read and write operations.
- #define DM35425_OFFSET_DAC_MODE_STATUS 0x00

  Offset to the Mode/Status register, from the start of the DAC control section.
- #define DM35425_OFFSET_DAC_CLK_SRC 0x01

  Offset to the Clock Source register, from the start of the DAC control section.
- #define DM35425_OFFSET_DAC_START_TRIG 0x02

  Offset to the Start Trigger register, from the start of the DAC control section.
- #define DM35425_OFFSET_DAC_STOP_TRIG 0x03

  Offset to the Stop Trigger register, from the start of the DAC control section.
- #define DM35425_OFFSET_DAC_CLK_DIV 0x04

  Offset to the Clock Divider register, from the start of the DAC control section.
- #define DM35425_OFFSET_DAC_CLK_DIV_COUNT 0x08

  Offset to the Clock Divider Counter register, from the start of the DAC control section.

- #define DM35425_OFFSET_DAC_POST_STOP_CONV 0x10

  Offset to the Post-Stop Conversion Count register, from the start of the DAC control section.
- #define DM35425_OFFSET_DAC_CONV_COUNT 0x14

  Offset to the Conversion Count register, from the start of the DAC control section.
- #define DM35425_OFFSET_DAC_INT_ENABLE 0x18

  Offset to the Interrupt Enable register, from the start of the DAC control section.
- #define DM35425_OFFSET_DAC_INT_STAT 0x1e

  Offset to the Interrupt Status register, from the start of the DAC control section.
- #define DM35425_OFFSET_DAC_CLK_BUS2 0x22

  Offset to the Clock Bus 2 register, from the start of the DAC control section.
- #define DM35425_OFFSET_DAC_CLK_BUS3 0x23

  Offset to the Clock Bus 3 register, from the start of the DAC control section.
- #define DM35425_OFFSET_DAC_CLK_BUS4 0x24

  Offset to the Clock Bus 4 register, from the start of the DAC control section.
- #define DM35425_OFFSET_DAC_CLK_BUS5 0x25

  Offset to the Clock Bus 5 register, from the start of the DAC control section.
- #define DM35425_OFFSET_DAC_CLK_BUS6 0x26

  Offset to the Clock Bus 6 register, from the start of the DAC control section.
- #define DM35425_OFFSET_DAC_CLK_BUS7 0x27

  Offset to the Clock Bus 7 register, from the start of the DAC control section.
- #define DM35425_OFFSET_DAC_DA_CONFIG 0x28

  Offset to the DA Config register, from the start of the DAC control section.
- #define DM35425_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 DM35425_DAC_CHAN_CTRL_BLK_SIZE 0x14

  Constant size of channel control section in function block.
- #define DM35425_OFFSET_DAC_CHAN_FRONT_END_CONFIG 0x00

  Offset to the Front-End Config register, from the start of the DAC channel control section.
- #define DM35425_OFFSET_DAC_CHAN_MARKER_STATUS 0x0a

  Offset to the Channel marker Interrupt Status register, from the start of the DAC channel control section.
- #define DM35425_OFFSET_DAC_CHAN_MARKER_ENABLE 0x0b

  Offset to the Channel marker Interrupt Enable register, from the start of the DAC channel control section.
- #define DM35425_OFFSET_DAC_CHAN_LAST_CONVERSION 0x10

  Offset to the Channel Last Conversion register, from the start of the DAC channel control section.
- #define DM35425_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 DM35425_OFFSET_DAC_FIFO_CTRL_BLK_SIZE 0x4

  Constant size of FIFO control section in function block.
- #define DM35425_OFFSET_ADIO_MODE_STATUS 0x00

  Offset to the ADIO Mode-Status register, from the start of the ADIO control section.
- #define DM35425_OFFSET_ADIO_CLK_SRC 0x01

  Offset to the ADIO Clock Source register, from the start of the ADIO control section.
- #define DM35425_OFFSET_ADIO_START_TRIG 0x02

  Offset to the ADIO Start Trigger register, from the start of the ADIO control section.
- #define DM35425_OFFSET_ADIO_STOP_TRIG 0x03

  Offset to the ADIO Stop Trigger register, from the start of the ADIO control section.
- #define DM35425_OFFSET_ADIO_CLK_DIV 0x04

  Offset to the ADIO Clock Divider register, from the start of the ADIO control section.
- #define DM35425_OFFSET_ADIO_CLK_DIV_COUNTER 0x08

  Offset to the ADIO Clock Divider Counter register, from the start of the ADIO control section.
- #define DM35425_OFFSET_ADIO_PRE_CAPT_COUNT 0x0c

Offset to the ADIO Pre-Start Capture Count register, from the start of the ADIO control section.
- #define DM35425_OFFSET_ADIO_POST_CAPT_COUNT 0x10

  Offset to the ADIO Post-Stop Capture Count register, from the start of the ADIO control section.
- #define DM35425_OFFSET_ADIO_SAMPLE_COUNT 0x14

  Offset to the ADIO Sample Count register, from the start of the ADIO control section.
- #define DM35425_OFFSET_ADIO_INT_ENABLE 0x18

  Offset to the ADIO Interrupt Enable register, from the start of the ADIO control section.
- #define DM35425_OFFSET_ADIO_INT_STAT 0x1e

  Offset to the ADIO Interrupt Status register, from the start of the ADIO control section.
- #define DM35425_OFFSET_ADIO_CLK_BUS2 0x22

  Offset to the ADIO Clock Bus 2, from the start of the ADIO control section.
- #define DM35425_OFFSET_ADIO_CLK_BUS3 0x23

  Offset to the ADIO Clock Bus 3 register, from the start of the ADIO control section.
- #define DM35425_OFFSET_ADIO_CLK_BUS4 0x24

  Offset to the ADIO Clock Bus 4 register, from the start of the ADIO control section.
- #define DM35425_OFFSET_ADIO_CLK_BUS5 0x25

  Offset to the ADIO Clock Bus 5 register, from the start of the ADIO control section.
- #define DM35425_OFFSET_ADIO_CLK_BUS6 0x26

  Offset to the ADIO Clock Bus 6 register, from the start of the ADIO control section.
- #define DM35425_OFFSET_ADIO_CLK_BUS7 0x27

  Offset to the ADIO Clock Bus 7 register, from the start of the ADIO control section.
- #define DM35425_OFFSET_ADIO_CHAN_START 0x28

  Offset to the beginning of the channels control section from the start of the control section.
- #define DM35425_OFFSET_ADIO_INPUT_VAL 0x00

  Offset to the Input Value register, from the start of the ADIO channel control section.
- #define DM35425_OFFSET_ADIO_OUTPUT_VAL 0x04

  Offset to the Output Value register, from the start of the ADIO channel control section.
- #define DM35425_OFFSET_ADIO_DIRECTION 0x08

  Offset to the Direction register, from the start of the ADIO channel control section.
- #define DM35425_OFFSET_ADIO_ADV_INT_MODE 0x0C

  Offset to the Advanced Interrupt mode register, from the start of the ADIO channel control section.
- #define DM35425_OFFSET_ADIO_ADV_INT_MASK 0x10

  Offset to the Advanced Interrupt mask register, from the start of the ADIO channel control section.
- #define DM35425_OFFSET_ADIO_ADV_INT_COMP 0x14

  Offset to the Advanced Interrupt compare register, from the start of the ADIO channel control section.
- #define DM35425_OFFSET_ADIO_ADV_INT_CAPT 0x18

  Offset to the Advanced Interrupt capture register, from the start of the ADIO channel control section.
- #define DM35425_OFFSET_ADIO_P_BUS_ENABLE 0x1C

  Offset to the Advanced Interrupt parallel bus enable register, from the start of the ADIO channel control section.
- #define DM35425_OFFSET_ADIO_P_BUS_READY_ENABLE 0x1D

  Offset to the Advanced Interrupt parallel bus ready register, from the start of the ADIO channel control section.
- #define DM35425_OFFSET_ADIO_FIFO_CTRL_BLK_START 0x50

  Offset to the start of the ADIO FIFO control section, from the start of the ADIO function block.
- #define DM35425_OFFSET_ADIO_FIFO_CTRL_BLK_SIZE 0x4

  Constant size of FIFO control section in function block.
- #define DM35425_OFFSET_EXT_CLOCKING_IN 0x00

  Offset to the pin value register, from the start of the External Clocking control section.
- #define DM35425_OFFSET_EXT_CLOCKING_GATE_IN 0x01

  Offset to the gate pin value register, from the start of the External Clocking control section.
- #define DM35425_OFFSET_EXT_CLOCKING_DIR 0x02

  Offset to the direction register, from the start of the External Clocking control section.

- #define DM35425_OFFSET_EXT_CLOCKING_EDGE 0x03

  Offset to the edge detect register, from the start of the External Clocking control section.
- #define DM35425_OFFSET_EXT_CLOCKING_PW2 0x04

  Offset to the pulse width (CLK2) register, from the start of the External Clocking control section.
- #define DM35425_OFFSET_EXT_CLOCKING_PW3 0x05

  Offset to the pulse width (CLK3) register, from the start of the External Clocking control section.
- #define DM35425_OFFSET_EXT_CLOCKING_PW4 0x06

  Offset to the pulse width (CLK4) register, from the start of the External Clocking control section.
- #define DM35425_OFFSET_EXT_CLOCKING_PW5 0x07

  Offset to the pulse width (CLK5) register, from the start of the External Clocking control section.
- #define DM35425_OFFSET_EXT_CLOCKING_PW6 0x08

  Offset to the pulse width (CLK6) register, from the start of the External Clocking control section.
- #define DM35425_OFFSET_EXT_CLOCKING_PW7 0x09

  Offset to the pulse width (CLK7) register, from the start of the External Clocking control section.
- #define DM35425_OFFSET_EXT_CLOCKING_SETUP_GBL2 0x0A

  Offset to the clocking method (CLK2) register, from the start of the External Clocking control section.
- #define DM35425_OFFSET_EXT_CLOCKING_SETUP_GBL3 0x0B

  Offset to the clocking method (CLK3) register, from the start of the External Clocking control section.
- #define DM35425_OFFSET_EXT_CLOCKING_SETUP_GBL4 0x0C

  Offset to the clocking method (CLK4) register, from the start of the External Clocking control section.
- #define DM35425_OFFSET_EXT_CLOCKING_SETUP_GBL5 0x0D

  Offset to the clocking method (CLK5) register, from the start of the External Clocking control section.
- #define DM35425_OFFSET_EXT_CLOCKING_SETUP_GBL6 0x0E

  Offset to the clocking method (CLK6) register, from the start of the External Clocking control section.
- #define DM35425_OFFSET_EXT_CLOCKING_SETUP_GBL7 0x0F

  Offset to the clocking method (CLK7) register, from the start of the External Clocking control section.

4.21.1 Detailed Description

4.21.2 Macro Definition Documentation

4.21.2.1 DM35425_OFFSET_DAC_STOP_TRIG

```
#define DM35425_OFFSET_DAC_STOP_TRIG 0x03
```

Offset to the Stop Trigger register, from the start of the DAC control section.

Definition at line 530 of file dm35425_registers.h.

4.21.2.2 DM35425_OFFSET_FB_ADC_FIFO

```
#define DM35425_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 dm35425_registers.h.

4.22 DM35425 Board Types

Macros

- #define DM35425_SUBTYPE_00 0

  Constant for FB subtype 0.
- #define DM35425_SUBTYPE_01 1

  Constant for FB subtype 1.
- #define DM35425_SUBTYPE_02 2

  Constant for FB subtype 2.
- #define DM35425_SUBTYPE_03 3

  Constant for FB subtype 3.
- #define DM35425_SUBTYPE_INVALID 0xFF

  Constant value indicating an invalid subtype.
- #define DM35425_FUNC_BLOCK_INVALID 0x0000

  Constant value indicating an invalid function block.
- #define DM35425_FUNC_BLOCK_INVALID2 0xFFFF

  Constant value indicating an invalid function block.
- #define DM35425_FUNC_BLOCK_SYNCBUS 0x0001

  Function Block Constant for SyncBus.
- #define DM35425_FUNC_BLOCK_EXT_CLOCKING 0x0002

  Function Block Constant for Global Clocking.
- #define DM35425_FUNC_BLOCK_CLK0003 0x0003

  Function Block Constant for External Clocking (0003)
- #define DM35425_FUNC_BLOCK_CAPTWIN 0x0005

  Function Block Constant for Capture Window.
- #define DM35425_FUNC_BLOCK_ADC 0x1000

  Function Block Constant for ADC.
- #define DM35425_FUNC_BLOCK_ADC1001 0x1001

  Function Block Constant for 10 MHz ADC (1001)
- #define DM35425_FUNC_BLOCK_DAC 0x2000

  Function Block Constant for DAC.
- #define DM35425_FUNC_BLOCK_DAC2001 0x2001

  Function Block Constant for High Speed DAC (2001)
- #define DM35425_FUNC_BLOCK_DIO 0x3000

  Function Block Constant for DIO.
- #define DM35425_FUNC_BLOCK_ADIO 0x3001

  Function Block Constant for ADIO.
- #define DM35425_FUNC_BLOCK_ADIO3010 0x3010

  Function Block Constant for ADIO3010.
- #define DM35425_FUNC_BLOCK_USART 0x4000

  Function Block Constant for Synchronous/Asynchronous Serial Port.
- #define DM35425_FUNC_BLOCK_REF_ADJUST 0xF000

  Function Block Constant for Reference Adjustment.
- #define DM35425_FUNC_BLOCK_TEMPERATURE_SENSOR 0xF001

  Function Block Constant for Temperature Sensor.
- #define DM35425_FUNC_BLOCK_FLASH_PROGRAMMER 0xF002

  Function Block Constant for Flash Programmer.
- #define DM35425_FUNC_BLOCK_CLK_GEN 0xF003

  Function Block Constant for Clock Generator.

- #define DM35425_FUNC_BLOCK_DIN3011 0x3011

  Function Block Constant for Digital Input (3011)
- #define DM35425_FUNC_BLOCK_DOT3012 0x3012

  Function Block Constant for Digital Output (3012)
- #define DM35425_FUNC_BLOCK_INC3200 0x3200

  Function Block Constant for Incremental Encoder (3200)
- #define DM35425_FUNC_BLOCK_PWM3100 0x3100

  Function Block Constant for PWM (3100)
- #define DM35425_FUNC_BLOCK_CLK0004 0x0004

  Function Block Constant for Programmable Clock (0004)
- #define DM35425_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 DM35425_DMA_MAX_BUFFER_SIZE 0xFFFFFC

  Maximum possible DMA buffer size.
- #define DM35425_BOARD_ACK_INTERRUPT 0x1

  Value to write to the EOI register to acknowledge interrupts.
- #define DM35425_BOARD_RESET_VALUE 0xAA

  Value to write to the Reset register in order to reset the board.
- #define DM35425_FIFO_ACCESS_FB_REVISION 0x01

  Minimum function block revision that supports direct FIFO read/write access.

4.22.1 Detailed Description

4.23 DM35425 Utility Library Functions

Enumerations

- enum DM35425_Waveforms { DM35425_SINE_WAVE, DM35425_SQUARE_WAVE, DM35425_SAWTOOTH_WAVE }

  List of possible waveforms that can be generated for DAC purposes.

Functions

- uint32_t DM35425_Get_Maskable (uint16_t data, uint16_t mask)

  Return a 32-bit maskable register value from the data and mask.
- void DM35425_Micro_Sleep (unsigned long microsecs)

  Sleep for a specified number of microseconds.
- long DM35425_Get_Time_Diff (struct timeval last, struct timeval first)

  Calculate the time difference between the two timeval structs, in microseconds.
- int DM35425_Generate_Signal_Data (enum DM35425_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 DM35425_Waveforms

```
enum DM35425_Waveforms
```

List of possible waveforms that can be generated for DAC purposes.

Enumerator

DM35425_SINE_WAVE	A simple sine wave.
DM35425_SQUARE_WAVE	A square wave starting at max value
DM35425_SAWTOOTH_WAVE	A sawtooth wave going for min to max

Definition at line 46 of file dm35425_util_library.h.

4.23.3 Function Documentation

4.23.3.1 check_result()

```
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 DM35425_Setup_Dacs(), ISR(), main(), output_channel_status(), output_dma_buffer_status(), print_fifo_status(), and setup_dacs().

4.23.3.2 DM35425_Generate_Signal_Data()

```
int DM35425_Generate_Signal_Data (
             enum DM35425_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().

4.23.3.3 DM35425_Get_Maskable()

```
uint32_t DM35425_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 DM35425_Get_Time_Diff()

```
long DM35425_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 DM35425_Micro_Sleep()

```
void DM35425_Micro_Sleep (
             unsigned long microsecs )
```

Sleep for a specified number of microseconds.

Parameters

microsecs	Length of sleep (microseconds)

Return values

None	

Referenced by main().

Chapter 5

Data Structure Documentation

5.1 DM35425_Board_Descriptor Struct Reference

DM35425 board descriptor. This structure holds information about the board as a whole. It holds the file descriptor and ISR callback function, if applicable.

```
#include <dm35425_os.h>
```

Data Fields

- int file_descriptor
- void(∗ isr)()
- pthread_t pid

5.1.1 Detailed Description

DM35425 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 dm35425_os.h.

5.1.2 Field Documentation

5.1.2.1 file_descriptor

```
int file_descriptor
```

File descriptor for device returned from open()

Definition at line 55 of file dm35425_os.h.

5.1.2.2 isr

```
void(* isr()
```

Function pointer to the user ISR callback function.

Definition at line 60 of file dm35425_os.h.

5.1.2.3 pid

```
pthread_t pid
```

Process ID of the child process which will monitor DMA done interrupts.

Definition at line 65 of file dm35425_os.h.

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

- include/dm35425_os.h

5.2 dm35425_device_descriptor Struct Reference

DM35425 Device Descriptor. The identifying info for this particular board.

```
#include <dm35425_driver.h>
```

Data Fields

- char name [DM35425_NAME_LENGTH]
- struct dm35425_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 [DM35425_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

DM35425 Device Descriptor. The identifying info for this particular board.

Definition at line 210 of file dm35425_driver.h.

5.2.2 Field Documentation

5.2.2.1 device_lock

```
spinlock_t device_lock
```

Concurrency control

Definition at line 229 of file dm35425_driver.h.

5.2.2.2 dma_descr_list

```
struct list_head dma_descr_list
```

A list of all allocated DMA buffers

Definition at line 298 of file dm35425_driver.h.

5.2.2.3 dma_wait_queue

```
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 dm35425_driver.h.

5.2.2.4 int_queue_count

```
unsigned int int_queue_count
```

Number of interrupts currently in the queue

Definition at line 280 of file dm35425_driver.h.

5.2.2.5 int_queue_in_marker

```
unsigned int int_queue_in_marker
```

Where in the queue new entries are put

Definition at line 286 of file dm35425_driver.h.

5.2.2.6 int_queue_missed

```
unsigned int int_queue_missed
```

Number of interrupts missed because of a full queue

Definition at line 274 of file dm35425_driver.h.

5.2.2.7 int_queue_out_marker

```
unsigned int int_queue_out_marker
```

Where in the queue entries are pulled from

Definition at line 292 of file dm35425_driver.h.

5.2.2.8 int_wait_queue

```
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 dm35425_driver.h.

5.2.2.9 interrupt_fb

```
int interrupt_fb[DM35425_INT_QUEUE_SIZE]
```

Interrupt queue containing which functional blocks caused interrupts

Definition at line 267 of file dm35425_driver.h.

5.2.2.10 irq_number

```
unsigned int irq_number
```

IRQ line number

Definition at line 242 of file dm35425_driver.h.

5.2.2.11 name

```
char name[DM35425_NAME_LENGTH]
```

Device name used when requesting resources; a NUL terminated string of the form rtd-dm35425-x where x is the device minor number.

Definition at line 217 of file dm35425_driver.h.

5.2.2.12 pci

```
struct dm35425_pci_region pci[PCI_ROM_RESOURCE]
```

Information about each of the standard PCI regions

Definition at line 223 of file dm35425_driver.h.

5.2.2.13 reference_count

```
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 dm35425_driver.h.

5.2.2.14 remove_isr_flag

```
uint8_t remove_isr_flag
```

Used to assist poll in shutting down the thread waiting for interrupts

Definition at line 249 of file dm35425_driver.h.

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

- include/dm35425_driver.h

5.3 DM35425_DMA_Descriptor Struct Reference

Descriptor for the DMA on this board.

```
#include <dm35425_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 dm35425_board_access.h.

5.3.2 Field Documentation

5.3.2.1 buffer_start_offset

```
uint32_t buffer_start_offset[MAX_DMA_BUFFERS]
```

Offset to the beginning of the buffer control section.

Definition at line 80 of file dm35425_board_access.h.

5.3.2.2 control_offset

```
uint32_t control_offset
```

Offset to the DMA control register section

Definition at line 70 of file dm35425_board_access.h.

5.3.2.3 num_buffers

```
uint8_t num_buffers
```

Number of buffers for this DMA channel.

Definition at line 75 of file dm35425_board_access.h.

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

- include/dm35425_board_access.h

5.4 dm35425_dma_descriptor Struct Reference

DM35425 DMA descriptor. This structure holds information about a single DMA buffer.

```
#include <dm35425_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

DM35425 DMA descriptor. This structure holds information about a single DMA buffer.

Definition at line 160 of file dm35425_driver.h.

5.4.2 Field Documentation

5.4.2.1 buffer

```
int buffer
```

DMA buffer number this descriptor represents.

Definition at line 175 of file dm35425_driver.h.

5.4.2.2 buffer_size

```
unsigned int buffer_size
```

Size of this allocated buffer

Definition at line 192 of file dm35425_driver.h.

5.4.2.3 bus_addr

```
dma_addr_t bus_addr
```

Bus memory address for buffer.

Definition at line 186 of file dm35425_driver.h.

5.4.2.4 channel

```
int channel
```

DMA channel this buffer is in.

Definition at line 170 of file dm35425_driver.h.

5.4.2.5 fb_num

```
uint32_t fb_num
```

Function block number this DMA is associated with.

Definition at line 165 of file dm35425_driver.h.

5.4.2.6 list

```
struct list_head list
```

List head so that descriptors can be kept in a linked list.

Definition at line 198 of file dm35425_driver.h.

5.4.2.7 virt_addr

```
void* virt_addr
```

System memory address for buffer

Definition at line 180 of file dm35425_driver.h.

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

- include/dm35425_driver.h

5.5 DM35425_Function_Block Struct Reference

DM35425 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 <dm35425_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 DM35425_DMA_Descriptor dma_channel [MAX_DMA_CHANNELS]

5.5.1 Detailed Description

DM35425 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 dm35425_board_access.h.

5.5.2 Field Documentation

5.5.2.1 control_offset

```
uint32_t control_offset
```

Offset to the beginning of the control registers for this function block

Definition at line 145 of file dm35425_board_access.h.

5.5.2.2 dma_channel

```
struct DM35425_DMA_Descriptor dma_channel[MAX_DMA_CHANNELS]
```

Array of descriptors for each DMA channel

Definition at line 154 of file dm35425_board_access.h.

5.5.2.3 dma_offset

```
uint32_t dma_offset
```

Offset to the beginning of the DMA registers for this function block

Definition at line 119 of file dm35425_board_access.h.

5.5.2.4 fb_num

```
int fb_num
```

Function block num (as identified in GBC)

Definition at line 124 of file dm35425_board_access.h.

Referenced by ISR(), main(), output_channel_status(), and print_fifo_status().

5.5.2.5 fb_offset

```
uint32_t fb_offset
```

Offset to the beginning of the function block registers

Definition at line 114 of file dm35425_board_access.h.

5.5.2.6 num_dma_buffers

```
uint8_t num_dma_buffers
```

Number of DMA buffers in this function block

Definition at line 135 of file dm35425_board_access.h.

Referenced by ISR(), and main().

5.5.2.7 num_dma_channels

```
uint8_t num_dma_channels
```

Number of DMA channels in this function block

Definition at line 140 of file dm35425_board_access.h.

Referenced by main().

5.5.2.8 ordinal_fb_type_num

```
int ordinal_fb_type_num
```

The ordinal number of this particular function block type (0th, 1st, etc)

Definition at line 130 of file dm35425_board_access.h.

5.5.2.9 sub_type

```
uint16_t sub_type
```

Type of specific function block (ADC1, ADC2, ADC3, etc)

Definition at line 103 of file dm35425_board_access.h.

Referenced by main().

5.5.2.10 type

```
uint16_t type
```

Type of function block (ADC, DAC, DIO, etc)

Definition at line 98 of file dm35425_board_access.h.

Referenced by main().

5.5.2.11 type_revision

```
uint16_t type_revision
```

Revision of subtype (internal use only)

Definition at line 109 of file dm35425_board_access.h.

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

- include/dm35425_board_access.h

5.6 dm35425_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 <dm35425_board_access_structs.h>
```

Data Fields

- struct dm35425_ioctl_region_readwrite readwrite
- struct dm35425_ioctl_region_modify modify
- struct dm35425_ioctl_interrupt_info_request interrupt
- struct dm35425_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 dm35425_board_access_structs.h.

5.6.2 Field Documentation

5.6.2.1 dma

```
struct dm35425_ioctl_dma dma
```

DMA Configuration and Control

Definition at line 343 of file dm35425_board_access_structs.h.

5.6.2.2 interrupt

```
struct dm35425_ioctl_interrupt_info_request interrupt
```

Interrupt request structure

Definition at line 338 of file dm35425_board_access_structs.h.

5.6.2.3 modify

```
struct dm35425_ioctl_region_modify modify
```

PCI region read/modify/write

Definition at line 332 of file dm35425_board_access_structs.h.

5.6.2.4 readwrite

```
struct dm35425_ioctl_region_readwrite readwrite
```

PCI region read and write

Definition at line 326 of file dm35425_board_access_structs.h.

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

- include/dm35425_board_access_structs.h

5.7 dm35425_ioctl_dma Struct Reference

ioctl() request structure for DMA

```
#include <dm35425_board_access_structs.h>
```

Data Fields

- enum DM35425_DMA_FUNCTIONS function
- int num_buffers
- uint32_t buffer_size
- uint32_t fb_num
- int channel
- int buffer
- struct dm35425_pci_access_request pci
- void ∗ buffer_ptr

5.7.1 Detailed Description

ioctl() request structure for DMA

Definition at line 269 of file dm35425_board_access_structs.h.

5.7.2 Field Documentation

5.7.2.1 buffer

```
int buffer
```

Buffer in DMA channel that DMA is meant for.

Definition at line 299 of file dm35425_board_access_structs.h.

5.7.2.2 buffer_ptr

```
void* buffer_ptr
```

Pointer to user-space buffer for read or write.

Definition at line 309 of file dm35425_board_access_structs.h.

5.7.2.3 buffer_size

```
uint32_t buffer_size
```

Size (in bytes) to allocate for buffers

Definition at line 284 of file dm35425_board_access_structs.h.

5.7.2.4 channel

```
int channel
```

Channel in function with DMA operation is for.

Definition at line 294 of file dm35425_board_access_structs.h.

5.7.2.5 fb_num

```
uint32_t fb_num
```

Function Block DMA is for.

Definition at line 289 of file dm35425_board_access_structs.h.

5.7.2.6 function

```
enum DM35425_DMA_FUNCTIONS function
```

Requested DMA function to perform.

Definition at line 274 of file dm35425_board_access_structs.h.

5.7.2.7 num_buffers

```
int num_buffers
```

Number of buffers to initialize for DMA

Definition at line 279 of file dm35425_board_access_structs.h.

5.7.2.8 pci

```
struct dm35425_pci_access_request pci
```

PCI Address of DMA registers for this operation

Definition at line 304 of file dm35425_board_access_structs.h.

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

- include/dm35425_board_access_structs.h

5.8 dm35425_ioctl_interrupt_info_request Struct Reference

ioctl() request structure for interrupt

```
#include <dm35425_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 dm35425_board_access_structs.h.

5.8.2 Field Documentation

5.8.2.1 error_occurred

```
int error_occurred
```

Boolean if error occurred during interrupt

Definition at line 254 of file dm35425_board_access_structs.h.

Referenced by ISR().

5.8.2.2 interrupt_fb

```
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 dm35425_board_access_structs.h.

Referenced by ISR().

5.8.2.3 interrupts_remaining

```
int interrupts_remaining
```

Count of interrupts remaining in the driver queue.

Definition at line 244 of file dm35425_board_access_structs.h.

5.8.2.4 valid_interrupt

```
int valid_interrupt
```

Boolean of if interrupt is valid or not.

Definition at line 249 of file dm35425_board_access_structs.h.

Referenced by ISR().

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

- include/dm35425_board_access_structs.h

5.9 dm35425_ioctl_region_modify Struct Reference

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

```
#include <dm35425_board_access_structs.h>
```

Data Fields

- struct dm35425_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 dm35425_board_access_structs.h.

5.9.2 Field Documentation

5.9.2.1 access

```
struct dm35425_pci_access_request access
```

PCI region access request

Definition at line 194 of file dm35425_board_access_structs.h.

5.9.2.2 mask

```
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 mask16

```
uint16_t mask16
```

Mask for 16-bit operations

Definition at line 220 of file dm35425_board_access_structs.h.

5.9.2.4 mask32

```
uint32_t mask32
```

Mask for 32-bit operations

Definition at line 226 of file dm35425_board_access_structs.h.

5.9.2.5 mask8

```
uint8_t mask8
```

Mask for 8-bit operations

Definition at line 214 of file dm35425_board_access_structs.h.

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

- include/dm35425_board_access_structs.h

5.10 dm35425_ioctl_region_readwrite Struct Reference

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

```
#include <dm35425_board_access_structs.h>
```

Data Fields

- struct dm35425_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 dm35425_board_access_structs.h.

5.10.2 Field Documentation

5.10.2.1 access

```
struct dm35425_pci_access_request access
```

PCI region access request

Definition at line 180 of file dm35425_board_access_structs.h.

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

- include/dm35425_board_access_structs.h

5.11 dm35425_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 <dm35425_board_access_structs.h>
```

Data Fields

- enum dm35425_pci_region_access_size size
- enum dm35425_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 dm35425_board_access_structs.h.

5.11.2 Field Documentation

5.11.2.1 data

```
union { ... } data
```

Data to write or the data read

5.11.2.2 data16

```
uint16_t data16
```

16-bit value

Definition at line 158 of file dm35425_board_access_structs.h.

5.11.2.3 data32

```
uint32_t data32
```

32-bit value

Definition at line 164 of file dm35425_board_access_structs.h.

5.11.2.4 data8

```
uint8_t data8
```

8-bit value

Definition at line 152 of file dm35425_board_access_structs.h.

5.11.2.5 offset

```
uint16_t offset
```

Offset within region to access

Definition at line 140 of file dm35425_board_access_structs.h.

5.11.2.6 region

```
enum dm35425_pci_region_num region
```

The PCI region to access

Definition at line 134 of file dm35425_board_access_structs.h.

5.11.2.7 size

```
enum dm35425_pci_region_access_size size
```

Size of access in bits

Definition at line 128 of file dm35425_board_access_structs.h.

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

- include/dm35425_board_access_structs.h

5.12 dm35425_pci_region Struct Reference

DM35425 PCI region descriptor. This structure holds information about one of a device's PCI memory regions.

```
#include <dm35425_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

DM35425 PCI region descriptor. This structure holds information about one of a device's PCI memory regions.

Definition at line 117 of file dm35425_driver.h.

5.12.2 Field Documentation

5.12.2.1 allocated

```
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 dm35425_driver.h.

5.12.2.2 io_addr

```
unsigned long io_addr
```

I/O port number if I/O mapped

Definition at line 123 of file dm35425_driver.h.

5.12.2.3 length

```
unsigned long length
```

Length of region in bytes

Definition at line 129 of file dm35425_driver.h.

5.12.2.4 phys_addr

```
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 dm35425_driver.h.

5.12.2.5 virt_addr

```
void* virt_addr
```

Address at which region is mapped in kernel virtual address space if memory mapped

Definition at line 143 of file dm35425_driver.h.

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

- include/dm35425_driver.h

Chapter 6

File Documentation

6.1 examples/_non_public/dm35425_dac_fifo.c File Reference

Example program which demonstrates the use of the DAC FIFO.

```
#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 <string.h>
#include <signal.h>
#include "dm35425_gbc_library.h"
#include "dm35425_dac_library.h"
#include "dm35425_dma_library.h"
#include "dm35425_ioctl.h"
#include "dm35425_examples.h"
#include "dm35425.h"
#include "dm35425_util_library.h"
```

Macros

- #define DEFAULT_DAC_RATE 100
- #define DEFAULT_RANGE DM35425_DAC_RNG_BIPOLAR_5V
- #define DEFAULT_CHANNEL 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.
- void print_fifo_status (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel)

  Print the full status of the FIFO to the screen.
- void ISR (struct dm35425_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 act on.
- int main (int argument_count, char ∗∗arguments)

  The main program.

Variables

- static char ∗ program_name
- volatile int exit_program = 0

6.1.1 Detailed Description

Example program which demonstrates the use of the DAC FIFO.

```
    This example program sends data to the DAC for instant conversion.
    To see the output data, connect an oscilloscope to the AOUT0
    pin (CN3 Pin 17) and AGND (CN3 Pin 18).

    The user can control what value goes out the DAC by using keys to
    increase or decrease the desired voltage
    .
    Follow the on-screen instructions for adjusting the voltage.

    Press 'q' to quit the program.
```

Id

dm35425_dac_fifo.c 98533 2016-04-04 14:49:44Z rgroner

6.1.2 Macro Definition Documentation

6.1.2.1 DEFAULT_CHANNEL

```
#define DEFAULT_CHANNEL 0
```

Define a channel to use, if the user does not provide one.

Definition at line 74 of file dm35425_dac_fifo.c.

6.1.2.2 DEFAULT_DAC_RATE

```
#define DEFAULT_DAC_RATE 100
```

Default DAC rate, if user does not choose one.

Definition at line 62 of file dm35425_dac_fifo.c.

6.1.2.3 DEFAULT_RANGE

```
#define DEFAULT_RANGE DM35425_DAC_RNG_BIPOLAR_5V
```

Define a default range to use, if the user does not provide one.

Definition at line 68 of file dm35425_dac_fifo.c.

6.1.3 Function Documentation

6.1.3.1 main()

```
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

Return values

Non-Zero	Failure.

Definition at line 268 of file dm35425_dac_fifo.c.

References board, channel, CHANNELS_OPTION, check_result(), DAC_0, DEFAULT_CHANNEL, DEFAULT_↩
DAC_RATE, DEFAULT_RANGE, DM35425_Board_Open(), DM35425_CLK_SRC_IMMEDIATE, DM35425_CL↩
K_SRC_NEVER, DM35425_Dac_Channel_Setup(), DM35425_Dac_Fifo_Channel_Write(), DM35425_DAC_MAX, DM35425_DAC_MIN, DM35425_Dac_Open(), DM35425_DAC_RNG_BIPOLAR_10V, DM35425_DAC_RNG_↩
BIPOLAR_5V, DM35425_Dac_Set_Clock_Src(), DM35425_Dac_Set_Conversion_Rate(), DM35425_Dac_Set↩
_Post_Stop_Conversion_Count(), DM35425_Dac_Set_Start_Trigger(), DM35425_Dac_Set_Stop_Trigger(), D↩
M35425_Dac_Start(), DM35425_Dma_Clear(), DM35425_Dma_Configure_Interrupts(), DM35425_Dma_Pause(), DM35425_FIFO_SAMPLE_SIZE, DM35425_Gbc_Board_Reset(), DM35425_General_InstallISR(), DM35425_↩
Generate_Signal_Data(), DM35425_NUM_DAC_DMA_CHANNELS, DM35425_SINE_WAVE, ERROR_INTR_E↩
NABLE, exit_program, HELP_OPTION, INTERRUPT_DISABLE, ISR(), MINOR_OPTION, DM35425_Function↩
_Block::num_dma_buffers, DM35425_Function_Block::num_dma_channels, program_name, RANGE_OPTION, REFILL_FIFO_OPTION, sigint_handler(), and usage().

6.1.3.2 print_fifo_status()

```
void print_fifo_status (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_Function_Block * func_block,
            unsigned int channel )
```

Print the full status of the FIFO to the screen.

Parameters

handle	Pointer to the handle for the board descriptor
func_block	Pointer to the DAC function block
channel	Which channel of the DAC to output

Return values

None	

Definition at line 167 of file dm35425_dac_fifo.c.

References channel, check_result(), DM35425_Dma_Get_Fifo_Counts(), DM35425_Dma_Status(), and D↩
M35425_Function_Block::fb_num.

6.1.3.3 sigint_handler()

```
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 136 of file dm35425_dac_fifo.c.

References exit_program.

Referenced by main().

6.1.4 Variable Documentation

6.1.4.1 exit_program

```
volatile int exit_program = 0
```

Boolean indicating whether or not to exit the program.

Definition at line 84 of file dm35425_dac_fifo.c.

Referenced by main(), and sigint_handler().

6.1.4.2 program_name

```
char* program_name  [static]
```

Name of the program as invoked on the command line

Definition at line 79 of file dm35425_dac_fifo.c.

Referenced by main(), and usage().

6.2 examples/dm35425_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 <string.h>
#include "dm35425_gbc_library.h"
#include "dm35425_adc_library.h"
#include "dm35425_dac_library.h"
#include "dm35425_ioctl.h"
#include "dm35425_examples.h"
#include "dm35425_util_library.h"
#include "dm35425_board_access.h"
#include "dm35425_types.h"
#include "dm35425.h"
#include "dm35425_os.h"
```

Macros

- #define DEFAULT_RATE 1000
- #define DEFAULT_CHANNEL 0
- #define DEFAULT_RANGE DM35425_ADC_RNG_BIPOLAR_5V
- #define DEFAULT_MODE DM35425_ADC_INPUT_SINGLE_ENDED

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 dm35425_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.
- static void sigint_handler (int signal_number)

  Signal handler for SIGINT Control-C keyboard interrupt.
- static void DM35425_Setup_Dacs (struct DM35425_Board_Descriptor ∗board)

  Prepare the DACs for use.
- 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

6.2.1 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.

        Connect the signal of interest to AIN0 (CN3 Pin 1) and AGND
    (CN3 Pin 21), or pins corresponding to selected channel.

    For convenience in testing the ADC, especially differential voltages,
    the DAC is setup to output these specific voltages:

    AOUT0: -6V
    AOUT1: -3V
    AOUT2:  4V
    AOUT3:  8V

    The program will continue to run until CTRL-C is pressed.
```

Id

dm35425_adc.c 108025 2017-04-14 15:09:34Z rgroner

6.2.2 Macro Definition Documentation

6.2.2.1 DEFAULT_CHANNEL

```
#define DEFAULT_CHANNEL 0
```

Define a channel to use, if the user does not provide one.

Definition at line 78 of file dm35425_adc.c.

6.2.2.2 DEFAULT_MODE

```
#define DEFAULT_MODE DM35425_ADC_INPUT_SINGLE_ENDED
```

Define a default mode to use, if the user does not provide one

Definition at line 89 of file dm35425_adc.c.

6.2.2.3 DEFAULT_RANGE

```
#define DEFAULT_RANGE DM35425_ADC_RNG_BIPOLAR_5V
```

Define a default range to use, if the user does not provide one.

Definition at line 84 of file dm35425_adc.c.

6.2.2.4 DEFAULT_RATE

```
#define DEFAULT_RATE 1000
```

Rate to run at, if the user does not provide one.

Definition at line 72 of file dm35425_adc.c.

6.2.3 Function Documentation

6.2.3.1 DM35425_Setup_Dacs()

```
static void DM35425_Setup_Dacs (
             struct DM35425_Board_Descriptor * board )  [static]
```

Prepare the DACs for use.

Parameters

board	Pointer to the board descriptor

Definition at line 214 of file dm35425_adc.c.

References board, CHANNEL_0, CHANNEL_1, CHANNEL_2, CHANNEL_3, check_result(), DAC_0, DM35425←
_Dac_Channel_Setup(), DM35425_Dac_Open(), DM35425_Dac_Reset(), DM35425_DAC_RNG_BIPOLAR_10V, DM35425_Dac_Set_Last_Conversion(), and DM35425_Dac_Volts_To_Conv().

Referenced by main().

6.2.3.2 main()

```
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 354 of file dm35425_adc.c.

References ADC_0, board, channel, CHANNELS_OPTION, check_result(), DEFAULT_CHANNEL, DEFAU↵LT_MODE, DEFAULT_RANGE, DEFAULT_RATE, DM35425_Adc_Channel_Get_Last_Sample(), DM35425↵_Adc_Channel_Setup(), DM35425_Adc_Get_Sample_Count(), DM35425_Adc_Initialize(), DM35425_ADC_↵INPUT_DIFFERENTIAL, DM35425_ADC_INPUT_SINGLE_ENDED, DM35425_ADC_INT_SAMPLE_TAKE↵N_MASK, DM35425_Adc_Interrupt_Clear_Status(), DM35425_Adc_Interrupt_Get_Config(), DM35425_Adc_↵Interrupt_Get_Status(), DM35425_Adc_Interrupt_Set_Config(), DM35425_ADC_NO_DELAY, DM35425_Adc_↵Open(), DM35425_Adc_Reset(), DM35425_ADC_RNG_BIPOLAR_10V, DM35425_ADC_RNG_BIPOLAR_1_25V, DM35425_ADC_RNG_BIPOLAR_2_5V, DM35425_ADC_RNG_BIPOLAR_5V, DM35425_ADC_RNG_BIPOLAR↵_625mV, DM35425_ADC_RNG_UNIPOLAR_10V, DM35425_ADC_RNG_UNIPOLAR_1_25V, DM35425_ADC_↵RNG_UNIPOLAR_2_5V, DM35425_ADC_RNG_UNIPOLAR_5V, DM35425_Adc_Sample_To_Volts(), DM35425↵_Adc_Set_Clock_Src(), DM35425_Adc_Set_Post_Stop_Samples(), DM35425_Adc_Set_Pre_Trigger_Samples(), DM35425_Adc_Set_Sample_Rate(), DM35425_Adc_Set_Start_Trigger(), DM35425_Adc_Set_Stop_Trigger(), D↵M35425_Adc_Start(), DM35425_Board_Close(), DM35425_Board_Open(), DM35425_CLK_SRC_IMMEDIATE, DM35425_CLK_SRC_NEVER, DM35425_Gbc_Board_Reset(), DM35425_General_InstallISR(), DM35425_↵General_RemoveISR(), DM35425_Micro_Sleep(), DM35425_NUM_ADC_DMA_CHANNELS, DM35425_Setup↵_Dacs(), exit_program, HELP_OPTION, interrupt_count, INTERRUPT_DISABLE, INTERRUPT_ENABLE, ISR(), MINOR_OPTION, MODE_OPTION, my_adc, DM35425_Function_Block::num_dma_buffers, DM35425_Function↵_Block::num_dma_channels, program_name, RANGE_OPTION, sigint_handler(), and usage().

6.2.3.3 sigint_handler()

```
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 197 of file dm35425_adc.c.

References exit_program.

Referenced by main().

6.2.4 Variable Documentation

6.2.4.1 exit_program

```
volatile int exit_program = 0
```

Boolean indicating whether or not to exit the program.

Definition at line 104 of file dm35425_adc.c.

Referenced by main(), and sigint_handler().

6.2.4.2 interrupt_count

```
volatile int interrupt_count = 0
```

Count of interrupts that have happened.

Definition at line 99 of file dm35425_adc.c.

Referenced by ISR(), and main().

6.2.4.3 program_name

```
char* program_name  [static]
```

Name of the program as invoked on the command line

Definition at line 94 of file dm35425_adc.c.

Referenced by main(), and usage().

6.3 examples/dm35425_adc_all_dma.c File Reference

Example program which demonstrates the use of the ADC and DMA, using all ADC channels at the same time.

```
#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 "dm35425_gbc_library.h"
#include "dm35425_adc_library.h"
#include "dm35425_ioctl.h"
#include "dm35425_examples.h"
#include "dm35425_dma_library.h"
#include "dm35425.h"
#include "dm35425_util_library.h"
#include "dm35425_os.h"
```

Macros

- #define DEFAULT_RATE 10
- #define DEFAULT_RANGE DM35425_ADC_RNG_BIPOLAR_5V
- #define DAT_FILE_NAME_PREFIX "./adc_dma_data_ch"
- #define DAT_FILE_NAME_SUFFIX ".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 DM35425_Board_Descriptor ∗handle, const struct DM35425_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.
- static void ISR (struct dm35425_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 [DM35425_NUM_ADC_DMA_CHANNELS]
- static struct DM35425_Board_Descriptor ∗ board
- static struct DM35425_Function_Block my_adc
- static unsigned long buffer_count
- static int ∗∗ local_buffer [DM35425_NUM_ADC_DMA_CHANNELS]
- static volatile int exit_program = 0
- static unsigned long buffer_size_bytes = 0
- static unsigned int next_buffer

6.3.1 Detailed Description

Example program which demonstrates the use of the ADC and DMA, using all ADC channels at the same time.

```
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 files.  It will do this continuously until the
user hits CTRL-C (or the filesystem becomes full).

Connect the signals of interest to the appropriate ADC Input pins.

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

dm35425_adc_all_dma.c 125638 2020-05-07 20:41:01Z lfrankenfield

6.3.2 Macro Definition Documentation

6.3.2.1 DAT_FILE_NAME_PREFIX

```
#define DAT_FILE_NAME_PREFIX "./adc_dma_data_ch"
```

Prefix for files that will be output during example

Definition at line 73 of file dm35425_adc_all_dma.c.

6.3.2.2 DAT_FILE_NAME_SUFFIX

```
#define DAT_FILE_NAME_SUFFIX ".dat"
```

Prefix for files that will be output during example

Definition at line 78 of file dm35425_adc_all_dma.c.

6.3.2.3 DEFAULT_RANGE

```
#define DEFAULT_RANGE DM35425_ADC_RNG_BIPOLAR_5V
```

Define a default range to use, if the user does not provide one.

Definition at line 68 of file dm35425_adc_all_dma.c.

6.3.2.4 DEFAULT_RATE

```
#define DEFAULT_RATE 10
```

Default rate to use, if user does not enter one. (Hz)

Definition at line 62 of file dm35425_adc_all_dma.c.

6.3.3 Function Documentation

6.3.3.1 ISR()

```
static void ISR (
             struct dm35425_ioctl_interrupt_info_request int_info )  [static]
```

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 275 of file dm35425_adc_all_dma.c.

References board, buffer_count, buffer_size_bytes, channel, check_result(), CLEAR_INTERRUPT, DM35425_↩ Dma_Check_Buffer_Used(), DM35425_Dma_Check_For_Error(), DM35425_Dma_Clear_Interrupt(), DM35425_↩ Dma_Read(), DM35425_Dma_Reset_Buffer(), DM35425_Gbc_Ack_Interrupt(), DM35425_NUM_ADC_DMA_C↩ HANNELS, dma_has_error, exit_program, dm35425_ioctl_interrupt_info_request::interrupt_fb, local_buffer, my_↩ adc, next_buffer, NO_CLEAR_INTERRUPT, DM35425_Function_Block::num_dma_buffers, and dm35425_ioctl_↩ interrupt_info_request::valid_interrupt.

Referenced by main().

6.3.3.2 main()

```
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 393 of file dm35425_adc_all_dma.c.

References ADC_0, board, buffer_count, buffer_size_bytes, channel, CHANNEL_0, CHANNELS_OPTION, check_result(), DAT_FILE_NAME_PREFIX, DAT_FILE_NAME_SUFFIX, DEFAULT_RANGE, DEFAULT_RA↩ TE, DM35425_ADC_2_FULL_SAMPLE_DELAY, DM35425_Adc_Channel_Setup(), DM35425_Adc_Initialize(), DM35425_ADC_INPUT_SINGLE_ENDED, DM35425_ADC_MAX_RATE, DM35425_Adc_Open(), DM35425↩ _ADC_RNG_BIPOLAR_10V, DM35425_ADC_RNG_BIPOLAR_1_25V, DM35425_ADC_RNG_BIPOLAR_2_5V, DM35425_ADC_RNG_BIPOLAR_5V, DM35425_ADC_RNG_BIPOLAR_625mV, DM35425_ADC_RNG_UNIPO↩ LAR_10V, DM35425_ADC_RNG_UNIPOLAR_1_25V, DM35425_ADC_RNG_UNIPOLAR_2_5V, DM35425_AD↩ C_RNG_UNIPOLAR_5V, DM35425_Adc_Sample_To_Volts(), DM35425_Adc_Set_Clock_Src(), DM35425_Adc↩ _Set_Sample_Rate(), DM35425_Adc_Set_Start_Trigger(), DM35425_Adc_Set_Stop_Trigger(), DM35425_Adc_↩ Start(), DM35425_Board_Close(), DM35425_Board_Open(), DM35425_CLK_SRC_IMMEDIATE, DM35425_CL↩

K_SRC_NEVER, DM35425_DMA_BUFFER_CTRL_INTR, DM35425_DMA_BUFFER_CTRL_LOOP, DM35425↵_DMA_BUFFER_CTRL_VALID, DM35425_Dma_Buffer_Setup(), DM35425_Dma_Buffer_Status(), DM35425_↵Dma_Configure_Interrupts(), DM35425_Dma_Initialize(), DM35425_Dma_Setup(), DM35425_DMA_SETUP_D↵IRECTION_READ, DM35425_Dma_Start(), DM35425_Gbc_Board_Reset(), DM35425_General_InstallISR(), D↵M35425_General_RemoveISR(), DM35425_Micro_Sleep(), DM35425_NUM_ADC_DMA_BUFFERS, DM35425↵_NUM_ADC_DMA_CHANNELS, dma_has_error, ERROR_INTR_DISABLE, ERROR_INTR_ENABLE, exit_↵program, HELP_OPTION, INTERRUPT_DISABLE, INTERRUPT_ENABLE, ISR(), local_buffer, MINOR_OPTION, my_adc, NOT_IGNORE_USED, DM35425_Function_Block::num_dma_buffers, DM35425_Function_Block::num↵_dma_channels, output_channel_status(), program_name, RANGE_OPTION, RATE_OPTION, SAMPLES_OPT↵ION, sigint_handler(), and usage().

6.3.3.3 output_channel_status()

```
void output_channel_status (
             struct DM35425_Board_Descriptor * handle,
             const struct DM35425_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 224 of file dm35425_adc_all_dma.c.

References channel, check_result(), DM35425_Dma_Status(), and DM35425_Function_Block::fb_num.

Referenced by main().

6.3.3.4 sigint_handler()

```
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 193 of file dm35425_adc_all_dma.c.

References exit_program.

Referenced by main().

6.3.4 Variable Documentation

6.3.4.1 board

```
struct DM35425_Board_Descriptor* board  [static]
```

Pointer to board descriptor

Definition at line 93 of file dm35425_adc_all_dma.c.

Referenced by DM35425_Setup_Dacs(), ISR(), and main().

6.3.4.2 buffer_count

```
unsigned long buffer_count  [static]
```

Buffer count, used to tell the main loop that a buffer has been copied.

Definition at line 103 of file dm35425_adc_all_dma.c.

Referenced by ISR(), and main().

6.3.4.3 buffer_size_bytes

```
unsigned long buffer_size_bytes = 0  [static]
```

Size of the buffer allocated, in bytes.

Definition at line 119 of file dm35425_adc_all_dma.c.

Referenced by ISR(), and main().

6.3.4.4 dma_has_error

```
int dma_has_error[DM35425_NUM_ADC_DMA_CHANNELS]  [static]
```

Boolean flag indicating if there was a DMA error.

Definition at line 88 of file dm35425_adc_all_dma.c.

Referenced by ISR(), and main().

6.3.4.5 exit_program

```
volatile int exit_program = 0  [static]
```

Boolean indicating the program should exit.

Definition at line 114 of file dm35425_adc_all_dma.c.

Referenced by ISR(), main(), and sigint_handler().

6.3.4.6 local_buffer

```
int** local_buffer[DM35425_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 109 of file dm35425_adc_all_dma.c.

Referenced by ISR(), and main().

6.3.4.7 my_adc

```
struct DM35425_Function_Block my_adc  [static]
```

Pointer to array of function blocks that will hold the ADC descriptors

Definition at line 98 of file dm35425_adc_all_dma.c.

Referenced by ISR(), and main().

6.3.4.8 next_buffer

```
unsigned int next_buffer [static]
```

What buffer is next to be copied from DMA

Definition at line 124 of file dm35425_adc_all_dma.c.

Referenced by ISR().

6.3.4.9 program_name

```
char* program_name [static]
```

Name of the program as invoked on the command line

Definition at line 83 of file dm35425_adc_all_dma.c.

Referenced by main(), and usage().

6.4 examples/dm35425_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 <string.h>
#include "dm35425_gbc_library.h"
#include "dm35425_adc_library.h"
#include "dm35425_dac_library.h"
#include "dm35425_ioctl.h"
#include "dm35425_examples.h"
#include "dm35425_dma_library.h"
#include "dm35425.h"
#include "dm35425_util_library.h"
#include "dm35425_os.h"
```

Macros

- #define DEFAULT_RATE 1000
- #define DAC_RATE 10000
- #define DEFAULT_CHANNEL 0
- #define DEFAULT_RANGE DM35425_ADC_RNG_BIPOLAR_5V
- #define DAC_BUFFER_SIZE_SAMPLES 10000
- #define DAC_BUFFER_SIZE_BYTES (DAC_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 DM35425_Board_Descriptor ∗handle, const struct DM35425_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.
- static void ISR (struct dm35425_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.
- static void setup_dacs ()

  This function will setup the DAC to provide an output sine wave that can be used for the testing of ADC DMA.
- void convert_bin_to_txt (unsigned int samples_in_buffer)

  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 DM35425_Board_Descriptor ∗ board
- static struct DM35425_Function_Block my_adc
- static unsigned long buffer_count
- static int ∗∗ local_buffer
- static volatile int exit_program = 0
- static unsigned long buffer_size_bytes = 0
- static unsigned int channel = DEFAULT_CHANNEL
- static unsigned int next_buffer

6.4.1 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).

    Connect the signal of interest to AIN0 (pin 1 of CN3) and AGND
    (pin 21 of CN3).

    This is a very intensive operation for the PC, working CPU, memory,
    and file I/O fairly hard.  Thus, there is no way to 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.
```

Id

dm35425_adc_continuous_dma.c 108578 2017-05-04 20:38:56Z rgroner

6.4.2 Macro Definition Documentation

6.4.2.1 ASCII_FILE_NAME

```
#define ASCII_FILE_NAME "./adc_dma.txt"
```

Name of file when saving as ASCII

Definition at line 100 of file dm35425_adc_continuous_dma.c.

6.4.2.2 BIN_FILE_NAME

```
#define BIN_FILE_NAME "./adc_dma.bin"
```

Name of file when saving as binary

Definition at line 105 of file dm35425_adc_continuous_dma.c.

6.4.2.3 DAC_BUFFER_SIZE_BYTES

```
#define DAC_BUFFER_SIZE_BYTES (DAC_BUFFER_SIZE_SAMPLES * sizeof(int))
```

Size of the DMA buffer for the DAC, in bytes

Definition at line 95 of file dm35425_adc_continuous_dma.c.

6.4.2.4 DAC_BUFFER_SIZE_SAMPLES

```
#define DAC_BUFFER_SIZE_SAMPLES 10000
```

Size of the DMA buffer to be used by the DAC

Definition at line 90 of file dm35425_adc_continuous_dma.c.

6.4.2.5 DAC_RATE

```
#define DAC_RATE 10000
```

Default rate to use for the DAC

Definition at line 73 of file dm35425_adc_continuous_dma.c.

6.4.2.6 DEFAULT_CHANNEL

```
#define DEFAULT_CHANNEL 0
```

Define a channel to use, if the user does not provide one.

Definition at line 79 of file dm35425_adc_continuous_dma.c.

6.4.2.7 DEFAULT_RANGE

```
#define DEFAULT_RANGE DM35425_ADC_RNG_BIPOLAR_5V
```

Define a default range to use, if the user does not provide one.

Definition at line 85 of file dm35425_adc_continuous_dma.c.

6.4.2.8 DEFAULT_RATE

```
#define DEFAULT_RATE 1000
```

Default rate to use, if user does not enter one. (Hz)

Definition at line 68 of file dm35425_adc_continuous_dma.c.

6.4.3 Function Documentation

6.4.3.1 convert_bin_to_txt()

```
void convert_bin_to_txt (
            unsigned int samples_in_buffer )
```

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 558 of file dm35425_adc_continuous_dma.c.

References ASCII_FILE_NAME, and BIN_FILE_NAME.

Referenced by main().

6.4.3.2 ISR()

```
static void ISR (
            struct dm35425_ioctl_interrupt_info_request int_info )  [static]
```

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 328 of file dm35425_adc_continuous_dma.c.

References board, buffer_count, buffer_size_bytes, channel, check_result(), CLEAR_INTERRUPT, DM35425_↩
Dma_Check_Buffer_Used(), DM35425_Dma_Check_For_Error(), DM35425_Dma_Clear_Interrupt(), DM35425_↩
Dma_Find_Interrupt(), DM35425_Dma_Read(), DM35425_Dma_Reset_Buffer(), DM35425_Gbc_Ack_Interrupt(), dma_has_error, exit_program, dm35425_ioctl_interrupt_info_request::interrupt_fb, local_buffer, my_adc, next_↩
buffer, NO_CLEAR_INTERRUPT, DM35425_Function_Block::num_dma_buffers, and dm35425_ioctl_interrupt_↩
info_request::valid_interrupt.

Referenced by main().

6.4.3.3 main()

```
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 642 of file dm35425_adc_continuous_dma.c.

References ADC_0, ASCII_FILE_NAME, BIN2TXT_OPTION, BIN_FILE_NAME, BINARY_OPTION, board, buffer_count, buffer_size_bytes, channel, CHANNELS_OPTION, check_result(), convert_bin_to_txt(), DEFA↩
ULT_RANGE, DEFAULT_RATE, DM35425_Adc_Channel_Setup(), DM35425_Adc_Initialize(), DM35425_ADC↩
_INPUT_SINGLE_ENDED, DM35425_ADC_MAX_RATE, DM35425_ADC_NO_DELAY, DM35425_Adc_Open(), DM35425_ADC_RNG_BIPOLAR_10V, DM35425_ADC_RNG_BIPOLAR_1_25V, DM35425_ADC_RNG_BIPO↩
LAR_2_5V, DM35425_ADC_RNG_BIPOLAR_5V, DM35425_ADC_RNG_BIPOLAR_625mV, DM35425_ADC↩
_RNG_UNIPOLAR_10V, DM35425_ADC_RNG_UNIPOLAR_1_25V, DM35425_ADC_RNG_UNIPOLAR_2_5V, DM35425_ADC_RNG_UNIPOLAR_5V, DM35425_Adc_Set_Clock_Src(), DM35425_Adc_Set_Sample_Rate(), DM35425_Adc_Set_Start_Trigger(), DM35425_Adc_Set_Stop_Trigger(), DM35425_Adc_Start(), DM35425_↩
Board_Close(), DM35425_Board_Open(), DM35425_CLK_SRC_IMMEDIATE, DM35425_CLK_SRC_NEVER, DM35425_DMA_BUFFER_CTRL_INTR, DM35425_DMA_BUFFER_CTRL_LOOP, DM35425_DMA_BUFFER_↩
CTRL_VALID, DM35425_Dma_Buffer_Setup(), DM35425_Dma_Buffer_Status(), DM35425_Dma_Configure_↩
Interrupts(), DM35425_Dma_Initialize(), DM35425_Dma_Setup(), DM35425_DMA_SETUP_DIRECTION_READ, DM35425_Dma_Start(), DM35425_Gbc_Board_Reset(), DM35425_General_InstallISR(), DM35425_General_↩
RemoveISR(), DM35425_Micro_Sleep(), DM35425_NUM_ADC_DMA_BUFFERS, DM35425_NUM_ADC_DM↩
A_CHANNELS, dma_has_error, ERROR_INTR_DISABLE, ERROR_INTR_ENABLE, exit_program, HELP_OP↩
TION, INTERRUPT_DISABLE, INTERRUPT_ENABLE, ISR(), local_buffer, MINOR_OPTION, my_adc, NOT_I↩
GNORE_USED, DM35425_Function_Block::num_dma_buffers, DM35425_Function_Block::num_dma_channels, output_channel_status(), program_name, RANGE_OPTION, RATE_OPTION, SAMPLES_OPTION, setup_dacs(), sigint_handler(), and usage().

6.4.3.4 output_channel_status()

```
void output_channel_status (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_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 277 of file dm35425_adc_continuous_dma.c.

References channel, check_result(), DM35425_Dma_Status(), and DM35425_Function_Block::fb_num.

Referenced by main().

6.4.3.5 setup_dacs()

```
static void setup_dacs ( )  [static]
```

This function will setup the DAC to provide an output sine wave that can be used for the testing of ADC DMA.

It will setup DAC Channel 0 only.

Return values

None.	

Definition at line 443 of file dm35425_adc_continuous_dma.c.

References board, BUFFER_0, CHANNEL_0, check_result(), DAC_0, DAC_BUFFER_SIZE_BYTES, DAC↩
_BUFFER_SIZE_SAMPLES, DAC_RATE, DM35425_CLK_SRC_IMMEDIATE, DM35425_CLK_SRC_NEVER, DM35425_Dac_Channel_Setup(), DM35425_DAC_MAX, DM35425_DAC_MIN, DM35425_Dac_Open(), D↩
M35425_DAC_RNG_BIPOLAR_5V, DM35425_Dac_Set_Clock_Src(), DM35425_Dac_Set_Conversion_Rate(), DM35425_Dac_Set_Start_Trigger(), DM35425_Dac_Set_Stop_Trigger(), DM35425_Dac_Start(), DM35425_↩
DMA_BUFFER_CTRL_LOOP, DM35425_DMA_BUFFER_CTRL_VALID, DM35425_Dma_Buffer_Setup(), D↩
M35425_Dma_Initialize(), DM35425_Dma_Setup(), DM35425_DMA_SETUP_DIRECTION_WRITE, DM35425_↩
Dma_Start(), DM35425_Dma_Write(), DM35425_Generate_Signal_Data(), DM35425_SINE_WAVE, and IGNO↩
RE_USED.

Referenced by main().

6.4.3.6 sigint_handler()

```
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 246 of file dm35425_adc_continuous_dma.c.

References exit_program.

Referenced by main().

6.4.4 Variable Documentation

6.4.4.1 board

```
struct DM35425_Board_Descriptor* board  [static]
```

Pointer to board descriptor

Definition at line 120 of file dm35425_adc_continuous_dma.c.

Referenced by ISR(), main(), and setup_dacs().

6.4.4.2 buffer_count

```
unsigned long buffer_count  [static]
```

Array of buffer counts, used to track progress of each ADC as data is copied.

Definition at line 131 of file dm35425_adc_continuous_dma.c.

Referenced by ISR(), and main().

6.4.4.3 buffer_size_bytes

```
unsigned long buffer_size_bytes = 0  [static]
```

Size of the buffer allocated, in bytes.

Definition at line 147 of file dm35425_adc_continuous_dma.c.

Referenced by ISR(), and main().

6.4.4.4 channel

```
unsigned int channel = DEFAULT_CHANNEL  [static]
```

ADC Channel to use

Definition at line 152 of file dm35425_adc_continuous_dma.c.

Referenced by ISR(), main(), output_channel_status(), output_dma_buffer_status(), and print_fifo_status().

6.4.4.5 dma_has_error

```
int dma_has_error = 0  [static]
```

Boolean flag indicating if there was a DMA error.

Definition at line 115 of file dm35425_adc_continuous_dma.c.

Referenced by ISR(), and main().

6.4.4.6 exit_program

```
volatile int exit_program = 0  [static]
```

Boolean indicating the program should exit.

Definition at line 142 of file dm35425_adc_continuous_dma.c.

Referenced by ISR(), main(), and sigint_handler().

6.4.4.7 local_buffer

```
int** local_buffer  [static]
```

Pointer to local memory buffer where data is copied from the kernel buffers when a DMA buffer becomes full.

Definition at line 137 of file dm35425_adc_continuous_dma.c.

Referenced by ISR(), and main().

6.4.4.8 my_adc

```
struct DM35425_Function_Block my_adc  [static]
```

Pointer to array of function blocks that will hold the ADC descriptors

Definition at line 125 of file dm35425_adc_continuous_dma.c.

Referenced by ISR(), and main().

6.4.4.9 next_buffer

```
unsigned int next_buffer  [static]
```

Which buffer is next to be copied from DMA

Definition at line 157 of file dm35425_adc_continuous_dma.c.

Referenced by ISR().

6.4.4.10 program_name

```
char* program_name  [static]
```

Name of the program as invoked on the command line

Definition at line 110 of file dm35425_adc_continuous_dma.c.

Referenced by main(), and usage().

6.5 examples/dm35425_adc_fifo.c File Reference

Example program which demonstrates the use of the ADC FIFO and its 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 <string.h>
#include "dm35425_gbc_library.h"
#include "dm35425_adc_library.h"
#include "dm35425_dac_library.h"
#include "dm35425_dma_library.h"
#include "dm35425_ioctl.h"
#include "dm35425_examples.h"
#include "dm35425_util_library.h"
#include "dm35425_board_access.h"
#include "dm35425_types.h"
#include "dm35425.h"
#include "dm35425_os.h"
```

Macros

- #define DEFAULT_RATE 500
- #define DEFAULT_CHANNEL 0
- #define DEFAULT_RANGE DM35425_ADC_RNG_BIPOLAR_5V
- #define DEFAULT_MODE DM35425_ADC_INPUT_SINGLE_ENDED

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 dm35425_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 act on.
- 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 non_dma_interrupt_count = 0
- volatile int dma_interrupt_count = 0
- volatile int unexpected_interrupt_count = 0
- volatile int interrupt_count = 0
- volatile int exit_program = 0
- struct DM35425_Function_Block my_adc

6.5.1 Detailed Description

Example program which demonstrates the use of the ADC FIFO and its interrupts.

```
This example program uses an ADC to collect data.  The main point
of the example is to demonstrate copying data out of the FIFO and
showing the correct interrupts occurring throughout the process.

Connect the signal of interest to AIN0 (CN3 Pin 1) and AGND
(CN3 Pin 21), or pins corresponding to desired channel.

For convenience in testing the ADC, especially differential voltages,
the DAC is setup to output these specific voltages:

AOUT0: -6V
AOUT1: -3V
AOUT2:  4V
AOUT3:  8V

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

dm35425_adc_fifo.c 108025 2017-04-14 15:09:34Z rgroner

6.5.2 Macro Definition Documentation

6.5.2.1 DEFAULT_CHANNEL

```
#define DEFAULT_CHANNEL 0
```

Define a channel to use, if the user does not provide one.

Definition at line 77 of file dm35425_adc_fifo.c.

6.5.2.2 DEFAULT_MODE

```
#define DEFAULT_MODE DM35425_ADC_INPUT_SINGLE_ENDED
```

Define a default mode to use, if the user does not provide one

Definition at line 88 of file dm35425_adc_fifo.c.

6.5.2.3 DEFAULT_RANGE

```
#define DEFAULT_RANGE DM35425_ADC_RNG_BIPOLAR_5V
```

Define a default range to use, if the user does not provide one.

Definition at line 83 of file dm35425_adc_fifo.c.

6.5.2.4 DEFAULT_RATE

```
#define DEFAULT_RATE 500
```

Rate to run at, if the user does not provide one. (Hz)

Definition at line 71 of file dm35425_adc_fifo.c.

6.5.3 Function Documentation

6.5.3.1 main()

```
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 376 of file dm35425_adc_fifo.c.

References ADC_0, board, channel, CHANNELS_OPTION, check_result(), CLEAR_INTERRUPT, DEFAU↩LT_CHANNEL, DEFAULT_MODE, DEFAULT_RANGE, DEFAULT_RATE, DM35425_Adc_Channel_Setup(), DM35425_Adc_Fifo_Channel_Read(), DM35425_Adc_Get_Sample_Count(), DM35425_Adc_Initialize(), D↩M35425_ADC_INPUT_DIFFERENTIAL, DM35425_ADC_INPUT_SINGLE_ENDED, DM35425_ADC_INT_P↩OST_BUFF_FULL_MASK, DM35425_ADC_INT_SAMP_COMPL_MASK, DM35425_ADC_INT_SAMPLE_T↩AKEN_MASK, DM35425_Adc_Interrupt_Clear_Status(), DM35425_Adc_Interrupt_Get_Status(), DM35425_↩

Adc_Interrupt_Set_Config(), DM35425_ADC_NO_DELAY, DM35425_Adc_Open(), DM35425_Adc_Reset(), D←
M35425_ADC_RNG_BIPOLAR_10V, DM35425_ADC_RNG_BIPOLAR_1_25V, DM35425_ADC_RNG_BIPO←
LAR_2_5V, DM35425_ADC_RNG_BIPOLAR_5V, DM35425_ADC_RNG_BIPOLAR_625mV, DM35425_ADC←
_RNG_UNIPOLAR_10V, DM35425_ADC_RNG_UNIPOLAR_1_25V, DM35425_ADC_RNG_UNIPOLAR_2_5V, DM35425_ADC_RNG_UNIPOLAR_5V, DM35425_Adc_Sample_To_Volts(), DM35425_Adc_Set_Clock_Src(), D←
M35425_Adc_Set_Post_Stop_Samples(), DM35425_Adc_Set_Pre_Trigger_Samples(), DM35425_Adc_Set_←
Sample_Rate(), DM35425_Adc_Set_Start_Trigger(), DM35425_Adc_Set_Stop_Trigger(), DM35425_Adc_Start(), DM35425_Board_Close(), DM35425_Board_Open(), DM35425_CLK_SRC_IMMEDIATE, DM35425_CLK_SR←
C_NEVER, DM35425_Dma_Clear(), DM35425_Dma_Clear_Interrupt(), DM35425_Dma_Configure_Interrupts(), DM35425_Dma_Get_Errors(), DM35425_Dma_Pause(), DM35425_FIFO_SAMPLE_SIZE, DM35425_Gbc_←
Board_Reset(), DM35425_General_InstallISR(), DM35425_General_RemoveISR(), DM35425_Micro_Sleep(), DM35425_NUM_ADC_DMA_CHANNELS, DM35425_Setup_Dacs(), dma_interrupt_count, ERROR_INTR_DIS←
ABLE, ERROR_INTR_ENABLE, exit_program, HELP_OPTION, interrupt_count, INTERRUPT_DISABLE, INT←
ERRUPT_ENABLE, ISR(), MINOR_OPTION, MODE_OPTION, my_adc, NO_CLEAR_INTERRUPT, non_dma←
_interrupt_count, DM35425_Function_Block::num_dma_buffers, DM35425_Function_Block::num_dma_channels, program_name, RANGE_OPTION, sigint_handler(), unexpected_interrupt_count, and usage().

6.5.3.2 sigint_handler()

```
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 230 of file dm35425_adc_fifo.c.

References exit_program.

Referenced by main().

6.5.4 Variable Documentation

6.5.4.1 dma_interrupt_count

```
volatile int dma_interrupt_count = 0
```

Count of DMA interrupts.

Definition at line 103 of file dm35425_adc_fifo.c.

Referenced by ISR(), and main().

6.5.4.2 exit_program

```
volatile int exit_program = 0
```

Boolean indicating whether or not to exit the program.

Definition at line 118 of file dm35425_adc_fifo.c.

Referenced by main(), and sigint_handler().

6.5.4.3 interrupt_count

```
volatile int interrupt_count = 0
```

Count of expected interrupts.

Definition at line 113 of file dm35425_adc_fifo.c.

Referenced by ISR(), and main().

6.5.4.4 my_adc

```
struct DM35425_Function_Block my_adc
```

Function block descriptor

Definition at line 123 of file dm35425_adc_fifo.c.

Referenced by ISR(), and main().

6.5.4.5 non_dma_interrupt_count

```
volatile int non_dma_interrupt_count = 0
```

Count of non-DMA interrupts.

Definition at line 98 of file dm35425_adc_fifo.c.

Referenced by ISR(), and main().

6.5.4.6 program_name

```
char* program_name  [static]
```

Name of the program as invoked on the command line

Definition at line 93 of file dm35425_adc_fifo.c.

Referenced by main(), and usage().

6.5.4.7 unexpected_interrupt_count

```
volatile int unexpected_interrupt_count = 0
```

Count of unexpected interrupts.

Definition at line 108 of file dm35425_adc_fifo.c.

Referenced by ISR(), and main().

6.6 examples/dm35425_adio.c File Reference

Example program which demonstrates the use of the ADIO.

```
#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 "dm35425_gbc_library.h"
#include "dm35425_adio_library.h"
#include "dm35425_ioctl.h"
#include "dm35425_util_library.h"
#include "dm35425_examples.h"
```

Macros

- #define DM35425_ADIO_DIRECTION 0x00FF00FF

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 DM35425_Board_Descriptor ∗ board
- struct DM35425_Function_Block my_adio

6.6.1 Detailed Description

Example program which demonstrates the use of the ADIO.

```
    This example program sets 16-bits of DIO to output, and
    16-bits to input.  We'll connect the output to the input and
    then write every possible 16-bit value to the output
    and verify the same value on the input pins.

    This example requires a loopback of DIO0-DIO7 to DIO8-DIO15
    and DIO16-DIO23 to DIO24-DIO31.  This can most easily be accomplished
    using standard sized jumpers and placing them across the
    following pins:

    CN3 and CN4:
    Pin23 to Pin24
    Pin25 to Pin26
    Pin27 to Pin28
    Pin29 to Pin30
    Pin31 to Pin32
    Pin33 to Pin34
    Pin35 to Pin36
    Pin37 to Pin38
```

Id

dm35425_adio.c 108025 2017-04-14 15:09:34Z rgroner

6.6.2 Macro Definition Documentation

6.6.2.1 DM35425_ADIO_DIRECTION

```
#define DM35425_ADIO_DIRECTION 0x00FF00FF
```

Constant defining the input and output direction of the ADIO pins

Definition at line 65 of file dm35425_adio.c.

6.6.3 Function Documentation

6.6.3.1 main()

```
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 134 of file dm35425_adio.c.

References board, check_result(), DM35425_ADIO_DIRECTION, DM35425_Adio_Get_Input_Value(), D↩M35425_Adio_Open(), DM35425_Adio_Set_Direction(), DM35425_Adio_Set_Output_Value(), DM35425_Board↩_Close(), DM35425_Board_Open(), DM35425_Gbc_Board_Reset(), HELP_OPTION, MINOR_OPTION, my_adio, program_name, and usage().

6.6.4 Variable Documentation

6.6.4.1 board

```
struct DM35425_Board_Descriptor* board
```

Pointer to the board descriptor

Definition at line 75 of file dm35425_adio.c.

Referenced by main().

6.6.4.2 my_adio

```
struct DM35425_Function_Block my_adio
```

Function block for the ADIO

Definition at line 80 of file dm35425_adio.c.

Referenced by main().

6.6.4.3 program_name

```
char* program_name  [static]
```

Name of the program as invoked on the command line

Definition at line 70 of file dm35425_adio.c.

Referenced by main(), and usage().

6.7 examples/dm35425_adio_adv_int.c File Reference

Example program which demonstrates the use of the ADIO advanced interrupts.

```
#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 "dm35425_gbc_library.h"
#include "dm35425_adio_library.h"
#include "dm35425_ioctl.h"
#include "dm35425_util_library.h"
#include "dm35425_examples.h"
#include "dm35425_os.h"
```

Macros

- #define DM35425_ADIO_DIRECTION 0x00FF00FF
- #define DM35425_ADIO_MATCH1 0x0000AA00
- #define DM35425_ADIO_MATCH2 0xAA000000

Functions

- void ISR (struct dm35425_ioctl_interrupt_info_request int_info)

  The interrupt subroutine that will execute when an ADIO interrupt occurs. This function will increment the count and clear the interrupt.
- static void sigint_handler (int signal_number)

  Signal handler for SIGINT Control-C keyboard interrupt.
- 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 DM35425_Board_Descriptor ∗ board
- struct DM35425_Function_Block my_adio
- static volatile unsigned int total_interrupt_count = 0
- volatile int exit_program = 0

6.7.1 Detailed Description

Example program which demonstrates the use of the ADIO advanced interrupts.

```
    This example requires a loopback of DIO0-DIO7 to DIO8-DIO15
    and DIO16-DIO23 to DIO24-DIO31.  This can most easily be accomplished
    using standard sized jumpers and placing them across the
    following pins of CN3 and CN4:

    Pin23 to Pin24
    Pin25 to Pin26
    Pin27 to Pin28
    Pin29 to Pin30
    Pin31 to Pin32
    Pin33 to Pin34
    Pin35 to Pin36
    Pin37 to Pin38

    The program demonstrates the DIO match and event interrupts.
    It will match on the value 0xAA, and when that value passes
    through the input pins, it will throw an interrupt.  It matches
    0xAA on the upper and lower 16-bit input values.

    The example will then wait for an event interrupt, which will occur
    if any input bit changes to zero.  This is accomplished by changing the
    digital outputs from 1s to 0s upon hitting Enter.  Because the outputs
    are tied to the inputs, the input will sense the bit change and
    generate an interrupt.  In this way, event interrupts can be
    tested without changing the loopback configuration.
```

Id

dm35425_adio_adv_int.c 108025 2017-04-14 15:09:34Z rgroner

6.7.2 Macro Definition Documentation

6.7.2.1 DM35425_ADIO_DIRECTION

```
#define DM35425_ADIO_DIRECTION 0x00FF00FF
```

Constant defining the input and output direction of the ADIO pins

Definition at line 74 of file dm35425_adio_adv_int.c.

6.7.2.2 DM35425_ADIO_MATCH1

```
#define DM35425_ADIO_MATCH1 0x0000AA00
```

Value to match on for the lower 8-bits.

Definition at line 79 of file dm35425_adio_adv_int.c.

6.7.2.3 DM35425_ADIO_MATCH2

```
#define DM35425_ADIO_MATCH2 0xAA000000
```

Value to match on for the upper 8-bits

Definition at line 84 of file dm35425_adio_adv_int.c.

6.7.3 Function Documentation

6.7.3.1 ISR()

```
void ISR (
            struct dm35425_ioctl_interrupt_info_request int_info )
```

The interrupt subroutine that will execute when an ADIO interrupt occurs. This function will increment the count and clear the interrupt.

Parameters

int_info	A structure containing information about the interrupt.

Return values

None.	

Definition at line 126 of file dm35425_adio_adv_int.c.

References board, check_result(), DM35425_ADIO_INT_ADV_INT_MASK, DM35425_Adio_Interrupt_Clear_↩ Status(), DM35425_Gbc_Ack_Interrupt(), dm35425_ioctl_interrupt_info_request::error_occurred, my_adio, total↩ _interrupt_count, and dm35425_ioctl_interrupt_info_request::valid_interrupt.

6.7.3.2 main()

```
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 224 of file dm35425_adio_adv_int.c.

6.7.3.3 sigint_handler()

```
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 165 of file dm35425_adio_adv_int.c.

References exit_program.

6.7.4 Variable Documentation

6.7.4.1 board

```
struct DM35425_Board_Descriptor* board
```

Pointer to the board descriptor

Definition at line 94 of file dm35425_adio_adv_int.c.

Referenced by ISR().

6.7.4.2 exit_program

```
volatile int exit_program = 0
```

Boolean indicating whether or not to exit the program.

Definition at line 109 of file dm35425_adio_adv_int.c.

Referenced by sigint_handler().

6.7.4.3 my_adio

```
struct DM35425_Function_Block my_adio
```

Function block for the ADIO

Definition at line 99 of file dm35425_adio_adv_int.c.

Referenced by ISR().

6.7.4.4 program_name

```
char* program_name  [static]
```

Name of the program as invoked on the command line

Definition at line 89 of file dm35425_adio_adv_int.c.

Referenced by usage().

6.7.4.5 total_interrupt_count

```
volatile unsigned int total_interrupt_count = 0  [static]
```

Total count of interrupts

Definition at line 104 of file dm35425_adio_adv_int.c.

Referenced by ISR().

6.8 examples/dm35425_adio_dma.c File Reference

Example program which demonstrates the use of the ADIO 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 "dm35425_gbc_library.h"
#include "dm35425_adio_library.h"
#include "dm35425_ioctl.h"
#include "dm35425_util_library.h"
#include "dm35425_examples.h"
#include "dm35425_dma_library.h"
```

Macros

- #define DM35425_ADIO_DIRECTION1 0x00FF00FF
- #define DM35425_ADIO_DIRECTION2 0xFF00FF00
- #define BUFFER_SIZE_SAMPLES 10000
- #define BUFFER_SIZE_BYTES (BUFFER_SIZE_SAMPLES ∗ 4)
- #define ADIO_RATE 10000

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 ISR (struct dm35425_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 output_channel_status (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_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 output_dma_buffer_status (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel, unsigned int buffer)

  Output the status of a DMA buffer. This is a helper function to determine the cause of an error when it occurs.
- int main (int argument_count, char ∗∗arguments)

  The main program.

Variables

- static char ∗ program_name
- volatile unsigned int buffer_copied = 0
- int dma_has_error = 0
- volatile int exit_program = 0
- struct DM35425_Board_Descriptor ∗ board
- struct DM35425_Function_Block my_adio

6.8.1 Detailed Description

Example program which demonstrates the use of the ADIO and DMA.

```
    The example will make use of 3 DMA buffers for each
    of the three DMA channels (ADIO In, ADIO Out, and ADIO
    Direction).  Data will "play out" of the ADIO Out and
    Direction channels, and be stored in the ADIO In DMA
    buffer.  Doing this, we'll receive a pattern in the DMA
    In buffers that is the result of the output and changing
    bit direction values.

    At the end, we'll compare what is stored in the ADIO In
    DMA buffers to what should have been the result and make
    sure it is correct.

    This example requires a loopback of DIO0-DIO7 to DIO8-DIO15
    and DIO16-DIO23 to DIO24-DIO31.  This can most easily be accomplished
    using standard sized jumpers and placing them across the
    following pins:

    CN3 and CN4:
    Pin23 to Pin24
    Pin25 to Pin26
    Pin27 to Pin28
    Pin29 to Pin30
    Pin31 to Pin32
    Pin33 to Pin34
    Pin35 to Pin36
    Pin37 to Pin38
```

Id

dm35425_adio_dma.c 108025 2017-04-14 15:09:34Z rgroner

6.8.2 Macro Definition Documentation

6.8.2.1 ADIO_RATE

```
#define ADIO_RATE 10000
```

Rate the ADIO is running

Definition at line 96 of file dm35425_adio_dma.c.

6.8.2.2 BUFFER_SIZE_BYTES

```
#define BUFFER_SIZE_BYTES (BUFFER_SIZE_SAMPLES * 4)
```

Size of the DMA buffer, in bytes.

Definition at line 91 of file dm35425_adio_dma.c.

6.8.2.3 BUFFER_SIZE_SAMPLES

```
#define BUFFER_SIZE_SAMPLES 10000
```

Size of the DMA buffer, in samples

Definition at line 86 of file dm35425_adio_dma.c.

6.8.2.4 DM35425_ADIO_DIRECTION1

```
#define DM35425_ADIO_DIRECTION1 0x00FF00FF
```

Constant defining the input and output direction of the ADIO pins

Definition at line 75 of file dm35425_adio_dma.c.

6.8.2.5 DM35425_ADIO_DIRECTION2

```
#define DM35425_ADIO_DIRECTION2 0xFF00FF00
```

Constant defining the input and output direction of the ADIO pins

Definition at line 81 of file dm35425_adio_dma.c.

6.8.3 Function Documentation

6.8.3.1 ISR()

```
void ISR (
            struct dm35425_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 190 of file dm35425_adio_dma.c.

References board, buffer_copied, check_result(), CLEAR_INTERRUPT, DM35425_ADIO_IN_DMA_CHANN←EL, DM35425_Dma_Clear_Interrupt(), DM35425_Gbc_Ack_Interrupt(), dm35425_ioctl_interrupt_info_request←::error_occurred, dm35425_ioctl_interrupt_info_request::interrupt_fb, my_adio, NO_CLEAR_INTERRUPT, and dm35425_ioctl_interrupt_info_request::valid_interrupt.

Referenced by main().

6.8.3.2 main()

```
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.

Set the DMA to halt once the 3rd buffer is finished

Definition at line 367 of file dm35425_adio_dma.c.

References ADIO_RATE, board, buffer_copied, BUFFER_SIZE_BYTES, BUFFER_SIZE_SAMPLES, check_←result(), DM35425_ADIO_DIR_DMA_CHANNEL, DM35425_ADIO_DIRECTION1, DM35425_ADIO_DIRECTI←ON2, DM35425_ADIO_IN_DMA_CHANNEL, DM35425_Adio_Open(), DM35425_ADIO_OUT_DMA_CHANNEL, DM35425_Adio_Set_Clock_Src(), DM35425_Adio_Set_Pacer_Clk_Rate(), DM35425_Adio_Set_Start_Trigger(), DM35425_Adio_Set_Stop_Trigger(), DM35425_Adio_Start(), DM35425_Board_Close(), DM35425_Board_Open(), DM35425_CLK_SRC_IMMEDIATE, DM35425_CLK_SRC_NEVER, DM35425_DMA_BUFFER_CTRL_HALT, D←M35425_DMA_BUFFER_CTRL_INTR, DM35425_DMA_BUFFER_CTRL_VALID, DM35425_Dma_Buffer_Setup(), DM35425_Dma_Configure_Interrupts(), DM35425_Dma_Initialize(), DM35425_Dma_Read(), DM35425_Dma_←Reset_Buffer(), DM35425_Dma_Setup(), DM35425_DMA_SETUP_DIRECTION_READ, DM35425_DMA_SET←UP_DIRECTION_WRITE, DM35425_Dma_Start(), DM35425_Dma_Write(), DM35425_Gbc_Board_Reset(), D←M35425_General_InstallISR(), DM35425_Micro_Sleep(), dma_has_error, ERROR_INTR_ENABLE, exit_program,

HELP_OPTION, IGNORE_USED, INTERRUPT_ENABLE, ISR(), MINOR_OPTION, my_adio, output_channel_↩
status(), output_dma_buffer_status(), program_name, sigint_handler(), and usage().

6.8.3.3 output_channel_status()

```
void output_channel_status (
             struct DM35425_Board_Descriptor * handle,
             const struct DM35425_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 252 of file dm35425_adio_dma.c.

References channel, check_result(), DM35425_Dma_Status(), and DM35425_Function_Block::fb_num.

Referenced by main().

6.8.3.4 output_dma_buffer_status()

```
void output_dma_buffer_status (
             struct DM35425_Board_Descriptor * handle,
             const struct DM35425_Function_Block * func_block,
             unsigned int channel,
             unsigned int buffer )
```

Output the status of a DMA buffer. 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.
buffer	The DMA channel buffer we want the status of.

Return values

None	

Definition at line 317 of file dm35425_adio_dma.c.

References channel, check_result(), and DM35425_Dma_Buffer_Status().

Referenced by main().

6.8.3.5 sigint_handler()

```
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 169 of file dm35425_adio_dma.c.

References exit_program.

Referenced by main().

6.8.4 Variable Documentation

6.8.4.1 board

```
struct DM35425_Board_Descriptor* board
```

Pointer to board descriptor

Definition at line 121 of file dm35425_adio_dma.c.

Referenced by ISR(), and main().

6.8.4.2 buffer_copied

```
volatile unsigned int buffer_copied = 0
```

A count of buffers copied

Definition at line 106 of file dm35425_adio_dma.c.

Referenced by ISR(), and main().

6.8.4.3 dma_has_error

```
int dma_has_error = 0
```

Flag indicating a DMA error occured

Definition at line 111 of file dm35425_adio_dma.c.

Referenced by main().

6.8.4.4 exit_program

```
volatile int exit_program = 0
```

Boolean indicating the program should be exited.

Definition at line 116 of file dm35425_adio_dma.c.

Referenced by main(), and sigint_handler().

6.8.4.5 my_adio

```
struct DM35425_Function_Block my_adio
```

ADIO function block descriptor

Definition at line 126 of file dm35425_adio_dma.c.

Referenced by ISR(), and main().

6.8.4.6 program_name

```
char* program_name  [static]
```

Name of the program as invoked on the command line

Definition at line 101 of file dm35425_adio_dma.c.

Referenced by main(), and usage().

6.9 examples/dm35425_adio_parallel_bus.c File Reference

Example program which demonstrates the use of the ADIO acting as a parallel bus.

```
#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 "dm35425_gbc_library.h"
#include "dm35425_dac_library.h"
#include "dm35425_ioctl.h"
#include "dm35425_examples.h"
#include "dm35425_dma_library.h"
#include "dm35425_util_library.h"
#include "dm35425_adio_library.h"
#include "dm35425.h"
```

Macros

- #define DM35425_ADIO_OUT_DIRECTION 0xBFFFFFFF
- #define DM35425_ADIO_IN_DIRECTION 0x40000000
- #define DEFAULT_RATE 100000
- #define BUFFER_SIZE_SAMPLES 0x400
- #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 output_channel_status (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_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 dm35425_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.
- 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
- unsigned interrupt_count = 0
- int is_sender = 0
- int is_receiver = 0
- struct DM35425_Function_Block my_adio
- struct DM35425_Board_Descriptor ∗ board

6.9.1 Detailed Description

Example program which demonstrates the use of the ADIO acting as a parallel bus.

```
    The ADIO may be used as a parallel bus to transfer data from 1 board
    to another.  In this mode, 3 ADIO signals are used for control, and
    the remaining 29 bits are used for passing data. This board uses DMA
    and the parallel bus mode to transfer data from 1 board to another.

    Two DM35425 boards are required for this example. Both boards will run
    the same example program, but one will be designated as the "sender",
    and one the "receiver".  Both examples should be executed and allowed
    to complete their setup before the data transfer is begun.

    In this example, only the ADIO bits on CN3 will be used for passing
    data.  All ADIO pins on CN3 (Pins 23-38) must be connected from 1 board
    to CN3 (Pins 23-38) of the 2nd board.

    The three control lines on CN4 must also be connected between the
    boards:

    CN4 Pin 24
    CN4 Pin 26
    CN4 Pin 28

    Use the --help command line option to see all possible options.
```

Id

dm35425_adio_parallel_bus.c 108025 2017-04-14 15:09:34Z rgroner

6.9.2 Macro Definition Documentation

6.9.2.1 BUFFER_SIZE_BYTES

```
#define BUFFER_SIZE_BYTES (BUFFER_SIZE_SAMPLES * sizeof(int))
```

Buffer size to allocate in bytes

Definition at line 93 of file dm35425_adio_parallel_bus.c.

6.9.2.2 BUFFER_SIZE_SAMPLES

```
#define BUFFER_SIZE_SAMPLES 0x400
```

Number of samples to create.

Definition at line 88 of file dm35425_adio_parallel_bus.c.

6.9.2.3 DEFAULT_RATE

```
#define DEFAULT_RATE 100000
```

Rate of passing data (Hz)

Definition at line 83 of file dm35425_adio_parallel_bus.c.

6.9.2.4 DM35425_ADIO_IN_DIRECTION

```
#define DM35425_ADIO_IN_DIRECTION 0x40000000
```

Constant giving direction for receiver board

Definition at line 78 of file dm35425_adio_parallel_bus.c.

6.9.2.5 DM35425_ADIO_OUT_DIRECTION

```
#define DM35425_ADIO_OUT_DIRECTION 0xBFFFFFFF
```

Constant giving direction for sender board

Definition at line 73 of file dm35425_adio_parallel_bus.c.

6.9.3 Function Documentation

6.9.3.1 ISR()

```
void ISR (
            struct dm35425_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 243 of file dm35425_adio_parallel_bus.c.

References board, check_result(), CLEAR_INTERRUPT, DM35425_ADIO_IN_DMA_CHANNEL, DM35425_ADI↩O_OUT_DMA_CHANNEL, DM35425_Dma_Clear_Interrupt(), DM35425_Gbc_Ack_Interrupt(), dm35425_ioctl_↩interrupt_info_request::error_occurred, interrupt_count, dm35425_ioctl_interrupt_info_request::interrupt_fb, is_↩sender, my_adio, and NO_CLEAR_INTERRUPT.

Referenced by main().

6.9.3.2 main()

```
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 335 of file dm35425_adio_parallel_bus.c.

References ADIO_0, board, BUFFER_SIZE_BYTES, BUFFER_SIZE_SAMPLES, check_result(), DEFAULT_↩RATE, DM35425_ADIO_IN_DIRECTION, DM35425_ADIO_IN_DMA_CHANNEL, DM35425_Adio_Open(), D↩M35425_ADIO_OUT_DIRECTION, DM35425_ADIO_OUT_DMA_CHANNEL, DM35425_Adio_Set_Clock_Src(), DM35425_Adio_Set_Direction(), DM35425_Adio_Set_P_Bus_Enable(), DM35425_Adio_Set_P_Bus_Ready_↩Enable(), DM35425_Adio_Set_Pacer_Clk_Rate(), DM35425_Adio_Set_Start_Trigger(), DM35425_Adio_Set_↩Stop_Trigger(), DM35425_Adio_Start(), DM35425_Board_Close(), DM35425_Board_Open(), DM35425_CLK↩_SRC_IMMEDIATE, DM35425_CLK_SRC_NEVER, DM35425_DMA_BUFFER_CTRL_HALT, DM35425_DMA↩_BUFFER_CTRL_INTR, DM35425_DMA_BUFFER_CTRL_VALID, DM35425_Dma_Buffer_Setup(), DM35425↩_Dma_Buffer_Status(), DM35425_Dma_Configure_Interrupts(), DM35425_Dma_Get_Current_Buffer_Count(), DM35425_Dma_Initialize(), DM35425_Dma_Read(), DM35425_Dma_Reset_Buffer(), DM35425_Dma_Setup(), DM35425_DMA_SETUP_DIRECTION_READ, DM35425_DMA_SETUP_DIRECTION_WRITE, DM35425_Dma↩_Start(), DM35425_Dma_Status(), DM35425_Dma_Write(), DM35425_FIFO_SAMPLE_SIZE, DM35425_Gbc↩

_Board_Reset(), DM35425_General_InstallISR(), DM35425_Micro_Sleep(), ENABLED, ERROR_INTR_ENAB↩ LE, exit_program, HELP_OPTION, IGNORE_USED, interrupt_count, INTERRUPT_ENABLE, is_receiver, is_↩ sender, ISR(), MINOR_OPTION, my_adio, DM35425_Function_Block::num_dma_buffers, DM35425_Function_↩ Block::num_dma_channels, output_channel_status(), program_name, RECEIVER_OPTION, SENDER_OPTION, sigint_handler(), and usage().

6.9.3.3 output_channel_status()

```
void output_channel_status (
            struct DM35425_Board_Descriptor * handle,
            const struct DM35425_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 192 of file dm35425_adio_parallel_bus.c.

References channel, check_result(), DM35425_Dma_Status(), and DM35425_Function_Block::fb_num.

Referenced by main().

6.9.3.4 sigint_handler()

```
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 303 of file dm35425_adio_parallel_bus.c.

References exit_program.

Referenced by main().

6.9.4 Variable Documentation

6.9.4.1 board

```
struct DM35425_Board_Descriptor* board
```

Board descriptor

Definition at line 128 of file dm35425_adio_parallel_bus.c.

Referenced by ISR(), and main().

6.9.4.2 exit_program

```
volatile int exit_program = 0
```

Boolean indicating the program should be exited.

Definition at line 103 of file dm35425_adio_parallel_bus.c.

Referenced by main(), and sigint_handler().

6.9.4.3 interrupt_count

```
unsigned interrupt_count = 0
```

Keep a count of how many interrupts have happened

Definition at line 108 of file dm35425_adio_parallel_bus.c.

Referenced by ISR(), and main().

6.9.4.4 is_receiver

```
int is_receiver = 0
```

Boolean indicating that this example is being run by the receiver.

Definition at line 118 of file dm35425_adio_parallel_bus.c.

Referenced by main().

6.9.4.5 is_sender

```
int is_sender = 0
```

Boolean indicating that this example is being run by the sender

Definition at line 113 of file dm35425_adio_parallel_bus.c.

Referenced by ISR(), and main().

6.9.4.6 my_adio

```
struct DM35425_Function_Block my_adio
```

Function block for ADIO

Definition at line 123 of file dm35425_adio_parallel_bus.c.

Referenced by ISR(), and main().

6.9.4.7 program_name

```
char* program_name  [static]
```

Name of the program as invoked on the command line

Definition at line 98 of file dm35425_adio_parallel_bus.c.

Referenced by main(), and usage().

6.10 examples/dm35425_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 <string.h>
#include "dm35425_gbc_library.h"
#include "dm35425_dac_library.h"
#include "dm35425_ioctl.h"
#include "dm35425_examples.h"
#include "dm35425.h"
#include "dm35425_util_library.h"
```

Macros

- #define DEFAULT_RANGE DM35425_DAC_RNG_BIPOLAR_5V
- #define DEFAULT_CHANNEL 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

6.10.1 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 AOUT0
pin (CN3 Pin 17) and AGND (CN3 Pin 18).

The user can control what value goes out the DAC by using keys to
increase or decrease the desired voltage.

Follow the on-screen instructions for adjusting the voltage.

Press 'q' to quit the program.
```

Id

dm35425_dac.c 108025 2017-04-14 15:09:34Z rgroner

6.10.2 Macro Definition Documentation

6.10.2.1 DEFAULT_CHANNEL

```
#define DEFAULT_CHANNEL 0
```

Define a channel to use, if the user does not provide one.

Definition at line 67 of file dm35425_dac.c.

6.10.2.2 DEFAULT_RANGE

```
#define DEFAULT_RANGE DM35425_DAC_RNG_BIPOLAR_5V
```

Define a default range to use, if the user does not provide one.

Definition at line 61 of file dm35425_dac.c.

6.10.3 Function Documentation

6.10.3.1 main()

```
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 136 of file dm35425_dac.c.

References board, channel, CHANNELS_OPTION, check_result(), DAC_0, DEFAULT_CHANNEL, DEFAULT↩_RANGE, DM35425_Board_Close(), DM35425_Board_Open(), DM35425_Dac_Channel_Setup(), DM35425_↩Dac_Conv_To_Volts(), DM35425_Dac_Get_Last_Conversion(), DM35425_DAC_MAX, DM35425_DAC_MIN, D↩M35425_Dac_Open(), DM35425_Dac_Reset(), DM35425_DAC_RNG_BIPOLAR_10V, DM35425_DAC_RNG_B↩IPOLAR_5V, DM35425_Dac_Set_Last_Conversion(), DM35425_Dac_Volts_To_Conv(), DM35425_Gbc_Board_↩Reset(), DM35425_Get_Time_Diff(), DM35425_NUM_DAC_DMA_CHANNELS, HELP_OPTION, MINOR_OPT↩ION, DM35425_Function_Block::num_dma_buffers, DM35425_Function_Block::num_dma_channels, program_↩name, RANGE_OPTION, and usage().

6.10.4 Variable Documentation

6.10.4.1 program_name

```
char* program_name  [static]
```

Name of the program as invoked on the command line

Definition at line 72 of file dm35425_dac.c.

Referenced by main(), and usage().

6.11 examples/dm35425_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 "dm35425_gbc_library.h"
#include "dm35425_dac_library.h"
#include "dm35425_ioctl.h"
#include "dm35425_examples.h"
#include "dm35425_dma_library.h"
#include "dm35425_util_library.h"
#include "dm35425.h"
```

Macros

- #define NUM_BUFFERS_TO_USE 1
- #define DEFAULT_RATE 100
- #define DEFAULT_RANGE DM35425_DAC_RNG_BIPOLAR_5V
- #define DEFAULT_CHANNEL 0
- #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

6.11.1 Detailed Description

Example program which demonstrates the use of the DAC and DMA.

```
This example program generates wave form data and "plays" it out
the specified DAC channel.  To see the output data, connect an
oscilloscope to the AOUT0 pin.

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.
```

Id

dm35425_dac_dma.c 108025 2017-04-14 15:09:34Z rgroner

6.11.2 Macro Definition Documentation

6.11.2.1 BUFFER_SIZE_BYTES

```
#define BUFFER_SIZE_BYTES (BUFFER_SIZE_SAMPLES * sizeof(int))
```

Buffer size to allocate in bytes

Definition at line 89 of file dm35425_dac_dma.c.

6.11.2.2 BUFFER_SIZE_SAMPLES

```
#define BUFFER_SIZE_SAMPLES 100
```

Number of samples to create. Increase this number for a "finer" waveform.

Definition at line 84 of file dm35425_dac_dma.c.

6.11.2.3 DEFAULT_CHANNEL

```
#define DEFAULT_CHANNEL 0
```

Define a channel to use, if the user does not provide one.

Definition at line 78 of file dm35425_dac_dma.c.

6.11.2.4 DEFAULT_RANGE

```
#define DEFAULT_RANGE DM35425_DAC_RNG_BIPOLAR_5V
```

Define a default range to use, if the user does not provide one.

Definition at line 72 of file dm35425_dac_dma.c.

6.11.2.5 DEFAULT_RATE

```
#define DEFAULT_RATE 100
```

Rate to use if user does not enter one on the command line (Hz)

Definition at line 66 of file dm35425_dac_dma.c.

6.11.2.6 NUM_BUFFERS_TO_USE

```
#define NUM_BUFFERS_TO_USE 1
```

We will only use one buffer in this example, and loop it

Definition at line 61 of file dm35425_dac_dma.c.

6.11.3 Function Documentation

6.11.3.1 main()

```
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

Return values

Non-Zero	Failure.

Definition at line 198 of file dm35425_dac_dma.c.

References board, BUFFER_0, BUFFER_SIZE_BYTES, BUFFER_SIZE_SAMPLES, channel, CHANNEL_↩
0, CHANNELS_OPTION, check_result(), DAC_0, DEFAULT_CHANNEL, DEFAULT_RANGE, DEFAULT_RATE, DM35425_Board_Close(), DM35425_Board_Open(), DM35425_CLK_SRC_IMMEDIATE, DM35425_CLK_SRC↩
_NEVER, DM35425_Dac_Channel_Setup(), DM35425_DAC_MAX, DM35425_DAC_MIN, DM35425_Dac_Open(), DM35425_DAC_RNG_BIPOLAR_10V, DM35425_DAC_RNG_BIPOLAR_5V, DM35425_Dac_Set_Clock_Src(), DM35425_Dac_Set_Conversion_Rate(), DM35425_Dac_Set_Start_Trigger(), DM35425_Dac_Set_Stop_Trigger(), DM35425_Dac_Start(), DM35425_DMA_BUFFER_CTRL_LOOP, DM35425_DMA_BUFFER_CTRL_VALID, D↩
M35425_Dma_Buffer_Setup(), DM35425_Dma_Buffer_Status(), DM35425_Dma_Initialize(), DM35425_Dma_↩
Setup(), DM35425_DMA_SETUP_DIRECTION_WRITE, DM35425_Dma_Start(), DM35425_Dma_Status(), D↩
M35425_Dma_Write(), DM35425_Gbc_Board_Reset(), DM35425_Generate_Signal_Data(), DM35425_NUM_↩
DAC_DMA_CHANNELS, DM35425_SAWTOOTH_WAVE, DM35425_SINE_WAVE, DM35425_SQUARE_WAVE, exit_program, HELP_OPTION, IGNORE_USED, MINOR_OPTION, NUM_BUFFERS_TO_USE, DM35425_↩
Function_Block::num_dma_buffers, DM35425_Function_Block::num_dma_channels, program_name, RANGE_↩
OPTION, RATE_OPTION, sigint_handler(), usage(), and WAVE_OPTION.

6.11.3.2 sigint_handler()

```
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 166 of file dm35425_dac_dma.c.

References exit_program.

Referenced by main().

6.11.4 Variable Documentation

6.11.4.1 exit_program

```
volatile int exit_program = 0
```

Boolean indicating the program should be exited.

Definition at line 100 of file dm35425_dac_dma.c.

Referenced by main(), and sigint_handler().

6.11.4.2 program_name

```
char* program_name  [static]
```

Name of the program as invoked on the command line

Definition at line 95 of file dm35425_dac_dma.c.

Referenced by main(), and usage().

6.12 examples/dm35425_ext_clocking.c File Reference

Example program which demonstrates the use of the external clocking function block.

```
#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 "dm35425_gbc_library.h"
#include "dm35425_dac_library.h"
#include "dm35425_adc_library.h"
#include "dm35425_adio_library.h"
#include "dm35425_ext_clocking_library.h"
#include "dm35425_ioctl.h"
#include "dm35425_examples.h"
#include "dm35425_dma_library.h"
#include "dm35425_util_library.h"
#include "dm35425.h"
#include "dm35425_os.h"
```

Macros

- #define DM35425_ADIO_DIR_OUTPUT 0xFFFFFFFF
- #define DM35425_EXT_CLK_DIR_INPUT 0x00
- #define DM35425_EXT_CLK_EDGE_RISING 0x00
- #define NUM_BUFFERS_TO_USE 1
- #define DEFAULT_RATE 20
- #define BUFFER_SIZE_SAMPLES 2
- #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

6.12.1 Detailed Description

Example program which demonstrates the use of the external clocking function block.

```
This example program uses function blocks to create signals which
are looped back into external clock inputs.  Each signal generated
produces the equivalent of a square wave.

Make connections as follows:

CN3: Pin 17 to Pin 39
CN3: Pin 37 to Pin 41

The DAC uses DMA data to output a square wave (all 0's then all 1's).
One of the DAC pins is then looped to the first external clock
input pin.

The ADIO will use the external clock signal to clock its own data
out, data which consists of a square wave (all 0's then all 1's).
One of the ADIO pins is then looped to the second external clock
input pin.

The ADC will use that external clock signal to control sampling
of data.

The sample/clock counter of each function block can then be polled
to verify the correct functioning.  The ADIO should run at half the
rate of the DAC, and the ADC should run at half the rate of the ADIO.

Hit Ctrl-C to exit the example.
```

Id

dm35425_ext_clocking.c 108025 2017-04-14 15:09:34Z rgroner

6.12.2 Macro Definition Documentation

6.12.2.1 BUFFER_SIZE_BYTES

```
#define BUFFER_SIZE_BYTES (BUFFER_SIZE_SAMPLES * sizeof(int))
```

Buffer size to allocate in bytes

Definition at line 110 of file dm35425_ext_clocking.c.

6.12.2.2 BUFFER_SIZE_SAMPLES

```
#define BUFFER_SIZE_SAMPLES 2
```

Number of samples to create.

Definition at line 105 of file dm35425_ext_clocking.c.

6.12.2.3 DEFAULT_RATE

```
#define DEFAULT_RATE 20
```

Rate to use if user does not enter one on the command line (Hz)

Definition at line 100 of file dm35425_ext_clocking.c.

6.12.2.4 DM35425_ADIO_DIR_OUTPUT

```
#define DM35425_ADIO_DIR_OUTPUT 0xFFFFFFFF
```

Set the direction of the ADIO to output

Definition at line 80 of file dm35425_ext_clocking.c.

6.12.2.5 DM35425_EXT_CLK_DIR_INPUT

```
#define DM35425_EXT_CLK_DIR_INPUT 0x00
```

Set the direction of the external clocking to input

Definition at line 85 of file dm35425_ext_clocking.c.

6.12.2.6 DM35425_EXT_CLK_EDGE_RISING

```
#define DM35425_EXT_CLK_EDGE_RISING 0x00
```

Set the edge detect to be on the rising edge for all clocks

Definition at line 90 of file dm35425_ext_clocking.c.

6.12.2.7 NUM_BUFFERS_TO_USE

```
#define NUM_BUFFERS_TO_USE 1
```

We will only use one buffer in this example, and loop it

Definition at line 95 of file dm35425_ext_clocking.c.

6.12.3 Function Documentation

6.12.3.1 main()

```
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 197 of file dm35425_ext_clocking.c.

References ADC_0, board, BUFFER_0, BUFFER_SIZE_BYTES, CHANNEL_0, check_result(), DAC_0, DEFA←ULT_RATE, DM35425_Adc_Channel_Setup(), DM35425_Adc_Initialize(), DM35425_ADC_INPUT_SINGLE_E←NDED, DM35425_ADC_NO_DELAY, DM35425_Adc_Open(), DM35425_ADC_RNG_BIPOLAR_5V, DM35425←_Adc_Set_Clk_Divider(), DM35425_Adc_Set_Clock_Src(), DM35425_Adc_Set_Start_Trigger(), DM35425_←Adc_Set_Stop_Trigger(), DM35425_ADIO_DIR_OUTPUT, DM35425_Adio_Open(), DM35425_ADIO_OUT_D←MA_CHANNEL, DM35425_Adio_Set_Clk_Divider(), DM35425_Adio_Set_Clock_Src(), DM35425_Adio_Set_←Direction(), DM35425_Adio_Set_Start_Trigger(), DM35425_Adio_Set_Stop_Trigger(), DM35425_Board_Open(), DM35425_CLK_BUS2, DM35425_CLK_BUS3, DM35425_CLK_SRC_IMMEDIATE, DM35425_CLK_SRC_N←EVER, DM35425_Dac_Channel_Setup(), DM35425_DAC_MAX, DM35425_DAC_MIN, DM35425_Dac_Open(), DM35425_DAC_RNG_UNIPOLAR_5V, DM35425_Dac_Set_Clock_Src(), DM35425_Dac_Set_Conversion_Rate(), DM35425_Dac_Set_Start_Trigger(), DM35425_Dac_Set_Stop_Trigger(), DM35425_DMA_BUFFER_CTRL_LO←OP, DM35425_DMA_BUFFER_CTRL_VALID, DM35425_Dma_Buffer_Setup(), DM35425_Dma_Buffer_Status(), DM35425_Dma_Configure_Interrupts(), DM35425_Dma_Initialize(), DM35425_Dma_Setup(), DM35425_DM←A_SETUP_DIRECTION_READ, DM35425_DMA_SETUP_DIRECTION_WRITE, DM35425_Dma_Start(), D←M35425_Dma_Status(), DM35425_Dma_Write(), DM35425_EXT_CLK_DIR_INPUT, DM35425_EXT_CLK_ED←GE_RISING, DM35425_Ext_Clocking_Open(), DM35425_Ext_Clocking_Set_Dir(), DM35425_Ext_Clocking_Set←_Edge(), DM35425_Ext_Clocking_Set_Method(), DM35425_Ext_Clocking_Set_Pulse_Width(), DM35425_Gbc_←Board_Reset(), ERROR_INTR_DISABLE, HELP_OPTION, IGNORE_USED, INTERRUPT_DISABLE, MINOR_←OPTION, my_adc, my_adio, NOT_IGNORE_USED, DM35425_Function_Block::num_dma_buffers, DM35425_←Function_Block::num_dma_channels, program_name, RATE_OPTION, sigint_handler(), and usage().

6.12.3.2 sigint_handler()

```
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 165 of file dm35425_ext_clocking.c.

References exit_program.

Referenced by main().

6.12.4 Variable Documentation

6.12.4.1 exit_program

```
volatile int exit_program = 0
```

Boolean indicating the program should be exited.

Definition at line 121 of file dm35425_ext_clocking.c.

Referenced by sigint_handler().

6.12.4.2 program_name

```
char* program_name  [static]
```

Name of the program as invoked on the command line

Definition at line 116 of file dm35425_ext_clocking.c.

Referenced by main(), and usage().

6.13 examples/dm35425_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 "dm35425_gbc_library.h"
#include "dm35425_ioctl.h"
#include "dm35425_examples.h"
#include "dm35425_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

6.13.1 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.
```

Id

dm35425_list_fb.c 108025 2017-04-14 15:09:34Z rgroner

6.13.2 Function Documentation

6.13.2.1 main()

```
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 107 of file dm35425_list_fb.c.

References board, check_result(), DM35425_Board_Close(), DM35425_Board_Open(), DM35425_FUNC_B←LOCK_ADC, DM35425_FUNC_BLOCK_ADIO, DM35425_FUNC_BLOCK_DAC, DM35425_FUNC_BLOCK_←EXT_CLOCKING, DM35425_FUNC_BLOCK_INVALID, DM35425_Function_Block_Open(), DM35425_Gbc_←Board_Reset(), DM35425_Gbc_Get_Fpga_Build(), DM35425_Gbc_Get_Pdp_Number(), DM35425_Gbc_Get←_Revision(), DM35425_MAX_FB, DM35425_Function_Block::fb_num, HELP_OPTION, MINOR_OPTION, D←M35425_Function_Block::num_dma_buffers, DM35425_Function_Block::num_dma_channels, program_name, DM35425_Function_Block::sub_type, DM35425_Function_Block::type, and usage().

6.13.3 Variable Documentation

6.13.3.1 program_name

```
char* program_name [static]
```

Name of the program as invoked on the command line

Definition at line 51 of file dm35425_list_fb.c.

Referenced by main(), and usage().

6.14 include/dm35425.h File Reference

Defines for the DM35425 (Device-specific values)

Macros

- #define DM35425_PCI_VENDOR_ID 0x1435
  DM35425 PCI vendor ID.
- #define DM35425_PCI_DEVICE_ID 0x5425
  DM35425 PCI device ID.
- #define DM35425_NUM_ADC_ON_BOARD 1
  Number of ADC on the DM35425.
- #define DM35425_NUM_DAC_ON_BOARD 1
  Number of DAC on the DM35425.
- #define DM35425_NUM_ADC_DMA_CHANNELS 32
  Number of channels per ADC.
- #define DM35425_NUM_ADC_DMA_BUFFERS 7
  Number of buffers per ADC DMA channel.
- #define DM35425_NUM_DAC_DMA_CHANNELS 4
  Number of channels per DAC.
- #define DM35425_NUM_DAC_DMA_BUFFERS 7
  Number of buffers per DAC DMA channel.
- #define DM35425_FIFO_SAMPLE_SIZE 511
  Sample size of the FIFO.

6.14.1 Detailed Description

Defines for the DM35425 (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

dm35425.h 80510 2014-07-17 15:46:23Z rgroner

6.15 include/dm35425_adc_library.h File Reference

Definitions for the DM35425 ADC Library.

`#include "dm35425_gbc_library.h"`

Macros

- #define DM35425_ADC_MODE_RESET 0x00

  Register value for ADC Mode Reset.
- #define DM35425_ADC_MODE_PAUSE 0x01

  Register value for ADC Mode Pause.
- #define DM35425_ADC_MODE_GO_SINGLE_SHOT 0x02

  Register value for ADC Mode Go (Single Shot)
- #define DM35425_ADC_MODE_GO_REARM 0x03

  Register value for ADC Mode Go (Rearm after Stop)
- #define DM35425_ADC_MODE_UNINITIALIZED 0x04

  Register value for ADC Mode Uninitialized.
- #define DM35425_ADC_STAT_STOPPED 0x00

  Register value for ADC Status - Stopped.
- #define DM35425_ADC_STAT_FILLING_PRE_TRIG_BUFF 0x01

  Register value for ADC Status - Filling Pre-Start Buffer.
- #define DM35425_ADC_STAT_WAITING_START_TRIG 0x02

  Register value for ADC Status - Waiting for Start Trigger.
- #define DM35425_ADC_STAT_SAMPLING 0x03

  Register value for ADC Status - Sampling Data.
- #define DM35425_ADC_STAT_FILLING_POST_TRIG_BUFF 0x04

  Register value for ADC Status - Filling Post-Stop Buffer.
- #define DM35425_ADC_STAT_WAIT_REARM 0x05

  Register value for ADC Status - Wait for Rearm.
- #define DM35425_ADC_STAT_DONE 0x07

  Register value for ADC Status - Done.

- #define DM35425_ADC_STAT_UNINITIALIZED 0x08

  Register value for ADC Status - Uninitialized.
- #define DM35425_ADC_STAT_INITIALIZING 0x09

  Register value for ADC Status - Initializing.
- #define DM35425_ADC_INT_SAMPLE_TAKEN_MASK 0x01

  Register value for Interrupt Mask - Sample Taken.
- #define DM35425_ADC_INT_CHAN_THRESHOLD_MASK 0x02

  Register value for Interrupt Mask - Channel Threshold Exceeded.
- #define DM35425_ADC_INT_PRE_BUFF_FULL_MASK 0x04

  Register value for Interrupt Mask - Pre-Start Buffer Filled.
- #define DM35425_ADC_INT_START_TRIG_MASK 0x08

  Register value for Interrupt Mask - Start Trigger Occurred.
- #define DM35425_ADC_INT_STOP_TRIG_MASK 0x10

  Register value for Interrupt Mask - Stop Trigger Occurred.
- #define DM35425_ADC_INT_POST_BUFF_FULL_MASK 0x20

  Register value for Interrupt Mask - Post-Stop Buffer Filled.
- #define DM35425_ADC_INT_SAMP_COMPL_MASK 0x40

  Register value for Interrupt Mask - Sampling Complete.
- #define DM35425_ADC_INT_PACER_TICK_MASK 0x80

  Register value for Interrupt Mask - Pacer Clock Tick Occurred.
- #define DM35425_ADC_INT_ALL_MASK 0xFF

  Register value for Interrupt Mask - All Bits.
- #define DM35425_ADC_CHAN_INTR_LOW_THRESHOLD_MASK 0x01

  Register value for Channel Low Threshold Interrupt.
- #define DM35425_ADC_CHAN_INTR_HIGH_THRESHOLD_MASK 0x02

  Register value for Channel High Threshold Interrupt.
- #define DM35425_ADC_CHAN_FILTER_ORDER0 0x0

  Register value for Channel Filter Order 0.
- #define DM35425_ADC_CHAN_FILTER_ORDER1 0x1

  Register value for Channel Filter Order 1.
- #define DM35425_ADC_CHAN_FILTER_ORDER2 0x2

  Register value for Channel Filter Order 2.
- #define DM35425_ADC_CHAN_FILTER_ORDER3 0x3

  Register value for Channel Filter Order 3.
- #define DM35425_ADC_CHAN_FILTER_ORDER4 0x4

  Register value for Channel Filter Order 4.
- #define DM35425_ADC_CHAN_FILTER_ORDER5 0x5

  Register value for Channel Filter Order 5.
- #define DM35425_ADC_CHAN_FILTER_ORDER6 0x6

  Register value for Channel Filter Order 6.
- #define DM35425_ADC_CHAN_FILTER_ORDER7 0x7

  Register value for Channel Filter Order 7.
- #define DM35425_ADC_FE_CONFIG_GAIN_05 0x10

  Register value for setting Half-Gain.
- #define DM35425_ADC_FE_CONFIG_GAIN_1 0x00

  Register value for setting a Gain of 1.
- #define DM35425_ADC_FE_CONFIG_GAIN_2 0x04

  Register value for setting a Gain of 2.
- #define DM35425_ADC_FE_CONFIG_GAIN_4 0x08

  Register value for setting a Gain of 4.
- #define DM35425_ADC_FE_CONFIG_GAIN_8 0x0C

Register value for setting a Gain of 8.
- #define DM35425_ADC_FE_CONFIG_GAIN_MASK 0x1C

  Register mask for setting gain bits.
- #define DM35425_ADC_FE_CONFIG_BIPOLAR 0x00

  Register value for setting input to Bi-Polar.
- #define DM35425_ADC_FE_CONFIG_UNIPOLAR 0x02

  Register value for setting input to Uni-Polar.
- #define DM35425_ADC_FE_CONFIG_POLARITY_MASK 0x02

  Register mask for setting polarity bits.
- #define DM35425_ADC_FE_CONFIG_SINGLE_ENDED 0x00

  Register value for setting input to Single-Ended.
- #define DM35425_ADC_FE_CONFIG_DIFFERENTIAL 0x01

  Register value for setting input to Differential.
- #define DM35425_ADC_FE_CONFIG_MODE_MASK 0x01

  Register mask for setting input mode.
- #define DM35425_ADC_FE_CONFIG_NO_DELAY 0x00

  Register value for configuring no sample delay.
- #define DM35425_ADC_FE_CONFIG_HALF_SAMPL_DELAY 0x40

  Register value for configuring a half sample delay.
- #define DM35425_ADC_FE_CONFIG_FULL_SAMPL_DELAY 0x80

  Register value for configuring a full sample delay.
- #define DM35425_ADC_FE_CONFIG_2_FULL_SAMPL_DELAY 0xC0

  Register value for configuring 2 full sample delay.
- #define DM35425_ADC_FE_CONFIG_DELAY_MASK 0xC0

  Register mask for setting delay value.
- #define DM35425_ADC_FE_CONFIG_ENABLED 0x20

  Register value for enabling ADC channel.
- #define DM35425_ADC_FE_CONFIG_DISABLED 0x00

  Register value for disabling ADC channel.
- #define DM35425_ADC_FE_CONFIG_ENABLE_MASK 0x20

  Register mask for setting ADC channel enable.
- #define DM35425_ADC_MAX_RATE 1250000

  Max allowable rate for the ADC (Hz)
- #define DM35425_ADC_THRESHOLD_MAX 4095L

  Maximum allowable value to write to the threshold register.
- #define DM35425_ADC_THRESHOLD_MIN 0L

  Minimum allowable value to write to the threshold register.
- #define DM35425_ADC_BIT_WIDTH_MAX 4096L

  The maximum value represented by the bit width of the ADC.
- #define DM35425_ADC_BIT_WIDTH_MAX_FLT ((float) DM35425_ADC_BIT_WIDTH_MAX)

  The max value of the ADC as a float.
- #define DM35425_ADC_RNG_1_25_LSB 0.00030517578125

  What each bit is worth at a range of 1.25.
- #define DM35425_ADC_RNG_2_5_LSB 0.0006103515625

  What each bit is worth at a range of 2.5.
- #define DM35425_ADC_RNG_5_LSB 0.001220703125

  What each bit is worth at a range of 5.
- #define DM35425_ADC_RNG_10_LSB 0.00244140625

  What each bit is worth at a range of 10.
- #define DM35425_ADC_RNG_20_LSB 0.0048828125

  What each bit is worth at a range of 20.

- #define DM35425_ADC_UNIPOLAR_MAX 4095L

    Max possible ADC value when in Unipolar.
- #define DM35425_ADC_UNIPOLAR_MIN 0

    Min possible ADC value when in Unipolar.
- #define DM35425_ADC_BIPOLAR_MAX 2047L

    Max possible ADC value when in Bipolar.
- #define DM35425_ADC_BIPOLAR_MIN -2048L

    Min possible ADC value when in Bipolar.

Enumerations

- enum DM35425_Adc_Clock_Events {
  DM35425_ADC_CLK_BUS_SRC_DISABLE = 0x00, DM35425_ADC_CLK_BUS_SRC_SAMPLE_TAKEN = 0x80, DM35425_ADC_CLK_BUS_SRC_CHAN_THRESH = 0x81, DM35425_ADC_CLK_BUS_SRC_PRE_START_BUFF_FUL
  = 0x82,
  DM35425_ADC_CLK_BUS_SRC_START_TRIG = 0x83, DM35425_ADC_CLK_BUS_SRC_STOP_TRIG = 0x84, DM35425_ADC_CLK_BUS_SRC_POST_STOP_BUFF_FULL = 0x85, DM35425_ADC_CLK_BUS_SRC_SAMPLING_C
  = 0x86,
  DM35425_ADC_CLK_BUS_SRC_PACER_TICK = 0x87 }

    Clock events for the global source clocks.
- enum DM35425_Input_Ranges {
  DM35425_ADC_RNG_BIPOLAR_10V, DM35425_ADC_RNG_BIPOLAR_5V, DM35425_ADC_RNG_BIPOLAR_2_5V, DM35425_ADC_RNG_BIPOLAR_1_25V,
  DM35425_ADC_RNG_BIPOLAR_625mV, DM35425_ADC_RNG_UNIPOLAR_5V, DM35425_ADC_RNG_UNIPOLAR_10V, DM35425_ADC_RNG_UNIPOLAR_2_5V,
  DM35425_ADC_RNG_UNIPOLAR_1_25V }

    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 DM35425_Input_Mode { DM35425_ADC_INPUT_SINGLE_ENDED, DM35425_ADC_INPUT_DIFFERENTIAL }

    Input mode of the ADC pin.
- enum DM35425_Gains {
  DM35425_ADC_GAIN_05, DM35425_ADC_GAIN_1, DM35425_ADC_GAIN_2, DM35425_ADC_GAIN_4, DM35425_ADC_GAIN_8, DM35425_ADC_GAIN_16, DM35425_ADC_GAIN_32, DM35425_ADC_GAIN_64, DM35425_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 DM35425_Channel_Delay { DM35425_ADC_NO_DELAY, DM35425_ADC_HALF_SAMPLE_DELAY, DM35425_ADC_FULL_SAMPLE_DELAY, DM35425_ADC_2_FULL_SAMPLE_DELAY }

    Channel to channel delay value.

Functions

- DM35425LIB_API int DM35425_Adc_Open (struct DM35425_Board_Descriptor ∗handle, unsigned int number_of_type, struct DM35425_Function_Block ∗func_block)

    Open the ADC indicated, and determine register locations of control blocks needed to control it.
- DM35425LIB_API int DM35425_Adc_Get_Start_Trigger (struct DM35425_Board_Descriptor ∗handle, struct DM35425_Function_Block ∗func_block, uint8_t ∗start_trigger)

    Get the start trigger for data collection.
- DM35425LIB_API int DM35425_Adc_Set_Start_Trigger (struct DM35425_Board_Descriptor ∗handle, struct DM35425_Function_Block ∗func_block, uint8_t start_trigger)

    Set the start trigger for data collection.

- DM35425LIB_API int DM35425_Adc_Get_Stop_Trigger (struct DM35425_Board_Descriptor ∗handle, struct DM35425_Function_Block ∗func_block, uint8_t ∗stop_trigger)

  Get the stop trigger for data collection.
- DM35425LIB_API int DM35425_Adc_Set_Stop_Trigger (struct DM35425_Board_Descriptor ∗handle, struct DM35425_Function_Block ∗func_block, uint8_t stop_trigger)

  Set the stop trigger for data collection.
- DM35425LIB_API int DM35425_Adc_Get_Pre_Trigger_Samples (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t ∗count)

  Get the amount of data to capture prior to start trigger.
- DM35425LIB_API int DM35425_Adc_Set_Pre_Trigger_Samples (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t count)

  Set the amount of data to capture prior to start trigger.
- DM35425LIB_API int DM35425_Adc_Get_Post_Stop_Samples (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t ∗count)

  Get the amount of data to capture after stop trigger.
- DM35425LIB_API int DM35425_Adc_Set_Post_Stop_Samples (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t count)

  Set the amount of data to capture after stop trigger.
- DM35425LIB_API int DM35425_Adc_Get_Clock_Src (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, enum DM35425_Clock_Sources ∗source)

  Get the clock source for the ADC.
- DM35425LIB_API int DM35425_Adc_Set_Clock_Src (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, enum DM35425_Clock_Sources source)

  Set the clock source for the ADC.
- DM35425LIB_API int DM35425_Adc_Initialize (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block)

  Prepare the ADC for actual data collection. Moves the ADC from uninitialized to stopped.
- DM35425LIB_API int DM35425_Adc_Set_Clk_Divider (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t divider)

  Set the Clock Divider for the ADC function block.
- DM35425LIB_API int DM35425_Adc_Set_Sample_Rate (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t rate, uint32_t ∗actual_rate)

  Set the sampling rate for the ADC.
- DM35425LIB_API int DM35425_Adc_Channel_Get_Front_End_Config (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel, uint16_t ∗fe_config)

  Get the front-end config register contents.
- DM35425LIB_API int DM35425_Adc_Interrupt_Set_Config (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint16_t int_source, int enable)

  Configure the interrupts for the ADC.
- DM35425LIB_API int DM35425_Adc_Interrupt_Get_Config (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint16_t ∗interrupt_ena)

  Get the interrupt configuration for the ADC.
- DM35425LIB_API int DM35425_Adc_Start (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block)

  Set the ADC mode to Start.
- DM35425LIB_API int DM35425_Adc_Start_Rearm (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block)

  Set the ADC mode to Start-Rearm.
- DM35425LIB_API int DM35425_Adc_Reset (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block)

  Set the ADC mode to Reset.
- DM35425LIB_API int DM35425_Adc_Pause (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block)

Set the ADC mode to Pause.

- DM35425LIB_API int DM35425_Adc_Uninitialize (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block)

  Set the ADC mode to Uninitialized.
- DM35425LIB_API int DM35425_Adc_Get_Mode_Status (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint8_t ∗mode_status)

  Get the ADC mode-status value.
- DM35425LIB_API int DM35425_Adc_Channel_Get_Last_Sample (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel, int32_t ∗value)

  Get the last sample taken from the ADC.
- DM35425LIB_API int DM35425_Adc_Get_Sample_Count (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t ∗value)

  Get the count of number of samples taken.
- DM35425LIB_API int DM35425_Adc_Interrupt_Get_Status (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint16_t ∗value)

  Get the interrupt status register.
- DM35425LIB_API int DM35425_Adc_Interrupt_Clear_Status (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint16_t value)

  Clear the interrupt status register.
- DM35425LIB_API int DM35425_Adc_Channel_Setup (struct DM35425_Board_Descriptor ∗handle, struct DM35425_Function_Block ∗func_block, unsigned int channel, enum DM35425_Channel_Delay input_delay, enum DM35425_Input_Ranges input_range, enum DM35425_Input_Mode input_mode)

  Setup the channel input for the ADC.
- DM35425LIB_API int DM35425_Adc_Channel_Reset (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel)

  Reset the channel front-end config.
- DM35425LIB_API int DM35425_Adc_Channel_Interrupt_Set_Config (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel, uint8_t interrupts_to_↩ set, int enable)

  Setup the channel interrupts.
- DM35425LIB_API int DM35425_Adc_Channel_Interrupt_Get_Config (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel, uint8_t ∗chan_intr_↩ enable)

  Get the channel interrupt configuration.
- DM35425LIB_API int DM35425_Adc_Channel_Interrupt_Get_Status (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel, uint8_t ∗chan_intr_↩ status)

  Get the channel interrupt status.
- DM35425LIB_API int DM35425_Adc_Channel_Interrupt_Clear_Status (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel, uint8_t chan_intr_status)

  Clear the interrupt status for this channel.
- DM35425LIB_API int DM35425_Adc_Channel_Find_Interrupt (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_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.
- DM35425LIB_API int DM35425_Adc_Channel_Set_Filter (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel, uint8_t chan_filter)

  Set the filter value for the channel.
- DM35425LIB_API int DM35425_Adc_Channel_Get_Filter (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel, uint8_t ∗chan_filter)

  Get the filter value for the channel.
- DM35425LIB_API int DM35425_Adc_Channel_Set_Low_Threshold (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel, int32_t threshold)

Set the lower threshold for this channel.
- DM35425LIB_API int DM35425_Adc_Channel_Set_High_Threshold (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel, int32_t threshold)

  Set the high threshold for this channel.
- DM35425LIB_API int DM35425_Adc_Channel_Get_Thresholds (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel, int32_t ∗low_threshold, int32_t ∗high_threshold)

  Get both thresholds for this channel.
- DM35425LIB_API int DM35425_Adc_Fifo_Channel_Read (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel, int32_t ∗value)

  Read an ADC sample stored in the onboard FIFO.
- DM35425LIB_API int DM35425_Adc_Set_Clock_Source_Global (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, enum DM35425_Clock_Sources clock_↩ select, enum DM35425_Adc_Clock_Events clock_driver)

  Set the global clock source for the ADC.
- DM35425LIB_API int DM35425_Adc_Get_Clock_Source_Global (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, int clock_select, int ∗clock_source)

  Get the global clock source for the selected clock.
- DM35425LIB_API int DM35425_Adc_Sample_To_Volts (enum DM35425_Input_Ranges input_range, int32_t adc_sample, float ∗volts)

  Convert an ADC sample to a volts value.
- DM35425LIB_API int DM35425_Adc_Volts_To_Sample (enum DM35425_Input_Ranges input_range, float volts, int32_t ∗adc_sample)

  Convert volts to an ADC value.

6.15.1 Detailed Description

Definitions for the DM35425 ADC Library.

Id

dm35425_adc_library.h 106898 2017-03-08 13:44:23Z rgroner

6.16 include/dm35425_adio_library.h File Reference

Definitions for the DM35425 ADIO Library.

```
#include "dm35425_gbc_library.h"
```

Macros

- #define DM35425_ADIO_MODE_RESET 0x00

  Register value for ADIO Mode Reset.
- #define DM35425_ADIO_MODE_PAUSE 0x01

  Register value for ADIO Mode Pause.
- #define DM35425_ADIO_MODE_GO_SINGLE_SHOT 0x02

  Register value for ADIO Mode Go (Single Shot)
- #define DM35425_ADIO_MODE_GO_REARM 0x03

Register value for ADIO Mode Go (Rearm after Stop)
- #define DM35425_ADIO_MODE_UNINITIALIZED 0x04

  Register value for ADIO Mode Uninitialized.
- #define DM35425_ADIO_STAT_STOPPED 0x00

  Register value for ADIO Status - Stopped.
- #define DM35425_ADIO_STAT_WAITING_START_TRIG 0x02

  Register value for ADIO Status - Waiting for Start Trigger.
- #define DM35425_ADIO_STAT_SAMPLING 0x03

  Register value for ADIO Status - Sampling Data.
- #define DM35425_ADIO_STAT_FILLING_POST_TRIG_BUFF 0x04

  Register value for ADIO Status - Filling Post-Stop Buffer.
- #define DM35425_ADIO_STAT_WAIT_REARM 0x05

  Register value for ADIO Status - Wait for Rearm.
- #define DM35425_ADIO_STAT_DONE 0x07

  Register value for ADIO Status - Done.
- #define DM35425_ADIO_STAT_UNINITIALIZED 0x08

  Register value for ADIO Status - Uninitialized.
- #define DM35425_ADIO_STAT_INITIALIZING 0x09

  Register value for ADIO Status - Initializing.
- #define DM35425_ADIO_INT_SAMPLE_TAKEN_MASK 0x0001

  Register value for Interrupt Mask - Sample Taken.
- #define DM35425_ADIO_INT_ADV_INT_MASK 0x0002

  Register value for Interrupt Mask - Advanced Interrupt Occurred.
- #define DM35425_ADIO_INT_PRE_BUFF_FULL_MASK 0x0004

  Register value for Interrupt Mask - Pre-Start Buffer Filled.
- #define DM35425_ADIO_INT_START_TRIG_MASK 0x0008

  Register value for Interrupt Mask - Start Trigger Occurred.
- #define DM35425_ADIO_INT_STOP_TRIG_MASK 0x0010

  Register value for Interrupt Mask - Stop Trigger Occurred.
- #define DM35425_ADIO_INT_POST_BUFF_FULL_MASK 0x0020

  Register value for Interrupt Mask - Post-Stop Buffer Filled.
- #define DM35425_ADIO_INT_SAMP_COMPL_MASK 0x0040

  Register value for Interrupt Mask - Sampling Complete.
- #define DM35425_ADIO_INT_PACER_TICK_MASK 0x0080

  Register value for Interrupt Mask - Pacer Clock Tick Occurred.
- #define DM35425_ADIO_INT_CN3_OVER_CURRENT_MASK 0x0100

  Register value for Interrupt Mask - CN3 5V Over-current.
- #define DM35425_ADIO_INT_CN4_OVER_CURRENT_MASK 0x0200

  Register value for Interrupt Mask - CN4 5V Over-current.
- #define DM35425_ADIO_INT_ALL_MASK 0xFFFF

  Register value for Interrupt Mask - All Bits.
- #define DM35425_ADIO_CLK_BUS_SRC_DISABLE 0x00

  Register value for Clock Event - Disabled.
- #define DM35425_ADIO_CLK_BUS_SRC_SAMPLE_TAKEN 0x80

  Register value for Clock Event - Sample Taken.
- #define DM35425_ADIO_CLK_BUS_SRC_ADV_INT 0x81

  Register value for Clock Event - Advanced Interrupt Occurred.
- #define DM35425_ADIO_CLK_BUS_SRC_PRE_START_BUFF_FULL 0x82

  Register value for Clock Event - Pre-Start Buffer Full.
- #define DM35425_ADIO_CLK_BUS_SRC_START_TRIG 0x83

  Register value for Clock Event - Start Trigger Occurred.

- #define DM35425_ADIO_CLK_BUS_SRC_STOP_TRIG 0x84

    Register value for Clock Event - Stop Trigger Occurred.
- #define DM35425_ADIO_CLK_BUS_SRC_POST_STOP_BUFF_FULL 0x85

    Register value for Clock Event - Post-Stop Buffer Full.
- #define DM35425_ADIO_CLK_BUS_SRC_SAMPLING_COMPLETE 0x86

    Register value for Clock Event - Sampling Complete.
- #define DM35425_ADIO_P_BUS_ENABLED 0x01

    Register value for Parallel Bus Enabled.
- #define DM35425_ADIO_P_BUS_DISABLED 0x00

    Register value for Parallel Bus Disabled.
- #define DM35425_ADIO_P_BUS_READY_ENABLED 0x01

    Register value for Parallel Bus Ready Enabled.
- #define DM35425_ADIO_P_BUS_READY_DISABLED 0x00

    Register value for Parallel Bus Ready Disabled.
- #define DM35425_ADIO_IN_DMA_CHANNEL 0

    DMA Channel number for ADIO IN.
- #define DM35425_ADIO_OUT_DMA_CHANNEL 1

    DMA Channel number for ADIO OUT.
- #define DM35425_ADIO_DIR_DMA_CHANNEL 2

    DMA Channel number for ADIO DIR.
- #define DM35425_ADIO_MAX_FREQ 4000000

    Maximum allowable speed for ADIO.

Enumerations

- enum DM35425_Adv_Interrupt_Mode { DM35425_ADV_INT_DISABLED, DM35425_ADV_INT_MATCH, DM35425_ADV_INT_EVENT }

    Advanced Interrupt Mode of the ADIO.

Functions

- DM35425LIB_API int DM35425_Adio_Open (struct DM35425_Board_Descriptor ∗handle, unsigned int number_of_type, struct DM35425_Function_Block ∗func_block)

    Open the ADIO indicated, and determine register locations of control blocks needed to control it.
- DM35425LIB_API int DM35425_Adio_Start (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block)

    Set the ADIO mode to Start.
- DM35425LIB_API int DM35425_Adio_Start_Rearm (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block)

    Set the ADIO mode to Start-Rearm.
- DM35425LIB_API int DM35425_Adio_Reset (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block)

    Set the ADIO mode to Reset.
- DM35425LIB_API int DM35425_Adio_Pause (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block)

    Set the ADIO mode to Pause.
- DM35425LIB_API int DM35425_Adio_Uninitialize (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block)

    Set the ADIO mode to Uninitialized.

- DM35425LIB_API int DM35425_Adio_Get_Mode_Status (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint8_t ∗mode_status)

  Get the ADIO mode-status value.
- DM35425LIB_API int DM35425_Adio_Set_Clock_Src (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, enum DM35425_Clock_Sources source)

  Set the clock source for the ADIO.
- DM35425LIB_API int DM35425_Adio_Set_Start_Trigger (struct DM35425_Board_Descriptor ∗handle, struct DM35425_Function_Block ∗func_block, uint8_t start_trigger)

  Set the start trigger for data collection.
- DM35425LIB_API int DM35425_Adio_Set_Stop_Trigger (struct DM35425_Board_Descriptor ∗handle, struct DM35425_Function_Block ∗func_block, uint8_t stop_trigger)

  Set the stop trigger for data collection.
- DM35425LIB_API int DM35425_Adio_Set_Clk_Divider (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t divider)

  Set the Clock Divider for the ADIO function block.
- DM35425LIB_API int DM35425_Adio_Get_Clk_Div_Counter (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t ∗counter)

  Get the Clock Divider Counter for the ADIO function block.
- DM35425LIB_API int DM35425_Adio_Set_Pacer_Clk_Rate (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t requested_rate, uint32_t ∗actual_rate)

  Set the pacer clock rate, the rate at which conversions happen.
- DM35425LIB_API int DM35425_Adio_Set_Pre_Trigger_Samples (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t pre_capture_count)

  Set the amount of data to capture prior to start trigger.
- DM35425LIB_API int DM35425_Adio_Set_Post_Stop_Samples (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t post_capture_count)

  Set the amount of data to capture after stop trigger.
- DM35425LIB_API int DM35425_Adio_Get_Sample_Count (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t ∗value)

  Get the count of number of samples taken.
- DM35425LIB_API int DM35425_Adio_Interrupt_Set_Config (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint16_t interrupt_src, int enable)

  Configure the interrupts for the ADIO.
- DM35425LIB_API int DM35425_Adio_Interrupt_Get_Config (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint16_t ∗interrupt_ena)

  Get the interrupt configuration for the ADIO.
- DM35425LIB_API int DM35425_Adio_Interrupt_Get_Status (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint16_t ∗value)

  Get the interrupt status register.
- DM35425LIB_API int DM35425_Adio_Interrupt_Clear_Status (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint16_t value)

  Clear the interrupt status register.
- DM35425LIB_API int DM35425_Adio_Set_Clock_Source_Global (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, enum DM35425_Clock_Sources clock_↩ select, int clock_source)

  Set the global clock source for the ADIO.
- DM35425LIB_API int DM35425_Adio_Get_Clock_Source_Global (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, int clock_select, int ∗clock_source)

  Get the global clock source for the selected clock.
- DM35425LIB_API int DM35425_Adio_Get_Input_Value (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t ∗value)

  Get the input value of the ADIO.

- DM35425LIB_API int DM35425_Adio_Get_Output_Value (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t ∗value)

  Get the current value of the output register.
- DM35425LIB_API int DM35425_Adio_Set_Output_Value (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t value)

  Set the value to be put on output pins.
- DM35425LIB_API int DM35425_Adio_Get_Direction (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t ∗direction)

  Get the direction of the ADIO pins.
- DM35425LIB_API int DM35425_Adio_Set_Direction (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t direction)

  Set the direction of the ADIO pins.
- DM35425LIB_API int DM35425_Adio_Get_Adv_Int_Mode (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint8_t ∗adv_int_mode)

  Get the Advanced Interrupt Mode.
- DM35425LIB_API int DM35425_Adio_Set_Adv_Int_Mode (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint8_t adv_int_mode)

  Set the Advanced Interrupt Mode.
- DM35425LIB_API int DM35425_Adio_Get_Adv_Int_Mask (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t ∗adv_int_mask)

  Get the Advanced Interrupt Mask.
- DM35425LIB_API int DM35425_Adio_Set_Adv_Int_Mask (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t adv_int_mask)

  Set the Advanced Interrupt Mask.
- DM35425LIB_API int DM35425_Adio_Get_Adv_Int_Comp (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t ∗adv_int_comp)

  Get the Advanced Interrupt Compare Register.
- DM35425LIB_API int DM35425_Adio_Set_Adv_Int_Comp (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t adv_int_comp)

  Set the Advanced Interrupt Compare Register.
- DM35425LIB_API int DM35425_Adio_Get_Adv_Int_Capt (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t ∗adv_int_capt)

  Get the Advanced Interrupt Capture Register.
- DM35425LIB_API int DM35425_Adio_Set_Adv_Int_Capt (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t adv_int_capt)

  Set the Advanced Interrupt Capture Register.
- DM35425LIB_API int DM35425_Adio_Get_P_Bus_Enable (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, int ∗p_bus_enabled)

  Get the Parallel Bus Enable Register boolean value.
- DM35425LIB_API int DM35425_Adio_Set_P_Bus_Enable (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, int p_bus_enabled)

  Set the Parallel Bus Enable.
- DM35425LIB_API int DM35425_Adio_Get_P_Bus_Ready_Enable (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, int ∗p_bus_ready_enabled)

  Get the Parallel Bus Ready Enable Register boolean value.
- DM35425LIB_API int DM35425_Adio_Set_P_Bus_Ready_Enable (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, int p_bus_ready_enabled)

  Set the Parallel Bus Ready Enable.
- DM35425LIB_API int DM35425_Adio_Fifo_Channel_Read (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel, int32_t ∗value)

  Read an aDIO sample from the FIFO.
- DM35425LIB_API int DM35425_Adio_Fifo_Channel_Write (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel, int32_t value)

  Write an aDIO sample to the FIFO.

6.16.1 Detailed Description

Definitions for the DM35425 ADIO Library.

Id

dm35425_adio_library.h 106898 2017-03-08 13:44:23Z rgroner

6.17 include/dm35425_board_access.h File Reference

Structures for the DM35425 Board Access Library.

```
#include <stdint.h>
#include "dm35425_board_access_structs.h"
#include "dm35425_types.h"
#include "dm35425_os.h"
```

Data Structures

- struct DM35425_DMA_Descriptor

  Descriptor for the DMA on this board.
- struct DM35425_Function_Block

  DM35425 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 DM35425LIB_API

Functions

- DM35425LIB_API int DM35425_Board_Open (uint8_t dev_num, struct DM35425_Board_Descriptor ∗∗handle)

  Open the board, providing the file descriptor that all future operations will reference. Also allocate memory for the device descriptor.
- DM35425LIB_API int DM35425_Board_Close (struct DM35425_Board_Descriptor ∗handle)

  Close the board, closing the open handle for the device file, and freeing the memory allocated for the decriptor.
- DM35425LIB_API int DM35425_Read (struct DM35425_Board_Descriptor ∗handle, union dm35425_ioctl_argument ∗ioctl_request)

  Read from the board.
- DM35425LIB_API int DM35425_Write (struct DM35425_Board_Descriptor ∗handle, union dm35425_ioctl_argument ∗ioctl_request)

  Write to the board.
- DM35425LIB_API int DM35425_Modify (struct DM35425_Board_Descriptor ∗handle, union dm35425_ioctl_argument ∗ioctl_request)

  Read/Modify/Write to the board.
- int DM35425_Dma (struct DM35425_Board_Descriptor ∗handle, union dm35425_ioctl_argument ∗ioctl_↩ request)

  Perform a DMA operation.

6.17.1 Detailed Description

Structures for the DM35425 Board Access Library.

Id

dm35425_board_access.h 104840 2016-11-30 19:20:54Z rgroner

6.17.2 Macro Definition Documentation

6.17.2.1 DM35425LIB_API

```
#define DM35425LIB_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 dm35425_board_access.h.

6.18 include/dm35425_board_access_structs.h File Reference

Structures for the DM35425 Board Access Library.

Data Structures

- struct dm35425_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 dm35425_ioctl_region_readwrite

  ioctl() request structure for read from or write to PCI region
- struct dm35425_ioctl_region_modify

  ioctl() request structure for PCI region read/modify/write
- struct dm35425_ioctl_interrupt_info_request

  ioctl() request structure for interrupt
- struct dm35425_ioctl_dma

  ioctl() request structure for DMA
- union dm35425_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 dm35425_pci_region_num { DM35425_PCI_REGION_GBC = 0, DM35425_PCI_REGION_GBC2, DM35425_PCI_REGION_FB }

  Standard PCI region number.
- enum dm35425_pci_region_access_size { DM35425_PCI_REGION_ACCESS_8 = 0, DM35425_PCI_REGION_ACCESS_16, DM35425_PCI_REGION_ACCESS_32 }

  Desired size in bits of access to standard PCI region.
- enum DM35425_DMA_FUNCTIONS { DM35425_DMA_INITIALIZE, DM35425_DMA_READ, DM35425_DMA_WRITE }

6.18.1 Detailed Description

Structures for the DM35425 Board Access Library.

Id

  dm35425_board_access_structs.h 80523 2014-07-17 18:38:59Z rgroner

6.19 include/dm35425_dac_library.h File Reference

Definitions for the DM35425 DAC Library.

Macros

- #define DM35425_DAC_INT_CONVERSION_SENT_MASK 0x01

  Register value for Interrupt Mask - Conversion Sent.
- #define DM35425_DAC_INT_CHAN_MARKER_MASK 0x02

  Register value for Interrupt Mask - Channel has enabled marker.
- #define DM35425_DAC_INT_START_TRIG_MASK 0x08

  Register value for Interrupt Mask - Start Trigger Occurred.
- #define DM35425_DAC_INT_STOP_TRIG_MASK 0x10

  Register value for Interrupt Mask - Stop Trigger Occurred.
- #define DM35425_DAC_INT_POST_STOP_DONE_MASK 0x20

  Register value for Interrupt Mask - Post-Stop Conversions Completed.
- #define DM35425_DAC_INT_PACER_TICK_MASK 0x80

  Register value for Interrupt Mask - Pacer Clock Tick.
- #define DM35425_DAC_INT_ALL_MASK 0xBB

  Register value for Interrupt Mask - All Bits.
- #define DM35425_DAC_MODE_RESET 0x00

  Register value for Mode - Reset.
- #define DM35425_DAC_MODE_PAUSE 0x01

  Register value for Mode - Pause.
- #define DM35425_DAC_MODE_GO_SINGLE_SHOT 0x02

  Register value for Mode - Go (Single Shot)
- #define DM35425_DAC_MODE_GO_REARM 0x03

  Register value for Mode - Go (Re-arm)
- #define DM35425_DAC_STATUS_STOPPED 0x00

Register value for DAC Status - Stopped.

- #define DM35425_DAC_STATUS_WAITING_START_TRIG 0x02

  Register value for DAC Status - Waiting for Start Trigger.
- #define DM35425_DAC_STATUS_CONVERTING 0x03

  Register value for DAC Status - Converting Data.
- #define DM35425_DAC_STATUS_OUTPUT_POST 0x04

  Register value for DAC Status - Outputting Post-Stop conversions.
- #define DM35425_DAC_STATUS_WAITING_REARM 0x05

  Register value for DAC Status - Waiting for Re-Arm.
- #define DM35425_DAC_STATUS_DONE 0x07

  Register value for DAC Status - Done.
- #define DM35425_DAC_FE_CONFIG_OUTPUT_ENABLE 0x04

  Register value for enabling DAC output.
- #define DM35425_DAC_FE_CONFIG_OUTPUT_DISABLE 0x00

  Register value for disabling DAC output.
- #define DM35425_DAC_FE_CONFIG_ENABLE_MASK 0x04

  Register mask for setting DAC enable/disable.
- #define DM35425_DAC_FE_CONFIG_GAIN_1 0x00

  Register value for setting a Gain of 1.
- #define DM35425_DAC_FE_CONFIG_GAIN_2 0x01

  Register value for setting a Gain of 2.
- #define DM35425_DAC_FE_CONFIG_UNIPOLAR 0x00

  Register value for setting DAC to Unipolar.
- #define DM35425_DAC_FE_CONFIG_BIPOLAR 0x02

  Register value for setting DAC to Bipolar.
- #define DM35425_DAC_FE_CONFIG_GAIN_MASK 0x01

  Register mask for setting gain.
- #define DM35425_DAC_FE_CONFIG_POLARITY_MASK 0x02

  Register mask for setting polarity.
- #define DM35425_DAC_MIN 0

  DAC Min value.
- #define DM35425_DAC_MAX 4095

  DAC Max value.
- #define DM35425_DAC_BIPOLAR_OFFSET 0x0800

  Offset to add to DAC value when in Bipolar mode.
- #define DM35425_DAC_UNIPOLAR_OFFSET 0x00

  Offset to add to the DAC value when in Unipolar mode.
- #define DM35425_DAC_RNG_5_LSB 0.001220703125f

  What each bit is worth at a range of 5.
- #define DM35425_DAC_RNG_10_LSB 0.00244140625f

  What each bit is worth at a range of 10.
- #define DM35425_DAC_RNG_20_LSB 0.0048828125f

  What each bit is worth at a range of 20.
- #define DM35425_DAC_MAX_RATE 200000

  Max allowable rate for the DAC (Hz)

Enumerations

- enum DM35425_Dac_Clock_Events {
  DM35425_DAC_CLK_BUS_SRC_DISABLE = 0x00, DM35425_DAC_CLK_BUS_SRC_CONVERSION_SENT = 0x80, DM35425_DAC_CLK_BUS_SRC_CHAN_MARKER = 0x81, DM35425_DAC_CLK_BUS_SRC_START_TRIG = 0x83,
  DM35425_DAC_CLK_BUS_SRC_STOP_TRIG = 0x84, DM35425_DAC_CLK_BUS_SRC_CONV_COMPL = 0x85 }

  Clocking events that can be used as the global clock sources.
- enum DM35425_Output_Ranges { DM35425_DAC_RNG_UNIPOLAR_5V, DM35425_DAC_RNG_UNIPOLAR_10V, DM35425_DAC_RNG_BIPOLAR_5V, DM35425_DAC_RNG_BIPOLAR_10V }

  Output range of the DAC pin.

Functions

- DM35425LIB_API int DM35425_Dac_Open (struct DM35425_Board_Descriptor ∗handle, unsigned int number_of_type, struct DM35425_Function_Block ∗func_block)

  Open the DAC indicated, and determine register locations of control blocks needed to control it.
- DM35425LIB_API int DM35425_Dac_Set_Clock_Src (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, enum DM35425_Clock_Sources source)

  Set the clock source of the DAC.
- DM35425LIB_API int DM35425_Dac_Get_Clock_Src (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, enum DM35425_Clock_Sources ∗source)

  Get the clock source of the DAC.
- DM35425LIB_API int DM35425_Dac_Get_Clock_Div (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t ∗divider)

  Get the clock divider value.
- DM35425LIB_API int DM35425_Dac_Set_Clock_Div (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t divider)

  Set the clock divider value.
- DM35425LIB_API int DM35425_Dac_Set_Conversion_Rate (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t requested_rate, uint32_t ∗actual_rate)

  Set the conversion rate of this DAC.
- DM35425LIB_API int DM35425_Dac_Interrupt_Set_Config (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint16_t interrupt_src, int enable)

  Set the interrupt configuration for this DAC.
- DM35425LIB_API int DM35425_Dac_Interrupt_Get_Config (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint16_t ∗interrupt_ena)

  Get the interrupt configuration for this DAC.
- DM35425LIB_API int DM35425_Dac_Set_Start_Trigger (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint8_t trigger_value)

  Set the start trigger.
- DM35425LIB_API int DM35425_Dac_Set_Stop_Trigger (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint8_t trigger_value)

  Set the stop trigger.
- DM35425LIB_API int DM35425_Dac_Get_Start_Trigger (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint8_t ∗trigger_value)

  Get the start trigger.
- DM35425LIB_API int DM35425_Dac_Get_Stop_Trigger (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint8_t ∗trigger_value)

  Get the stop trigger.
- DM35425LIB_API int DM35425_Dac_Start (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block)

Set the DAC Mode to Start.
- DM35425LIB_API int DM35425_Dac_Reset (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block)

    Set the DAC Mode to Reset.
- DM35425LIB_API int DM35425_Dac_Pause (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block)

    Set the DAC Mode to Pause.
- DM35425LIB_API int DM35425_Dac_Get_Mode_Status (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint8_t ∗mode_status)

    Get the Mode and Status of the DAC.
- DM35425LIB_API int DM35425_Dac_Get_Last_Conversion (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel, uint8_t ∗marker, int16_t ∗value)

    Get the value of the last conversion of the DAC.
- DM35425LIB_API int DM35425_Dac_Set_Last_Conversion (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel, uint8_t marker, int16_t value)

    Set a value to be converted by the DAC immediately.
- DM35425LIB_API int DM35425_Dac_Get_Conversion_Count (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t ∗value)

    Get a count of the number of conversions that DAC has executed.
- DM35425LIB_API int DM35425_Dac_Interrupt_Get_Status (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint16_t ∗value)

    Get a interrupt status register of the DAC.
- DM35425LIB_API int DM35425_Dac_Interrupt_Clear_Status (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint16_t value)

    Clear the interrupt status register of the DAC.
- DM35425LIB_API int DM35425_Dac_Set_Post_Stop_Conversion_Count (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t value)

    Set the number of conversions the DAC will make after a stop trigger.
- DM35425LIB_API int DM35425_Dac_Get_Post_Stop_Conversion_Count (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint32_t ∗value)

    Get the number of conversions the DAC will make after a stop trigger.
- DM35425LIB_API int DM35425_Dac_Set_Clock_Source_Global (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, enum DM35425_Clock_Sources clock, enum DM35425_Dac_Clock_Events clock_driver)

    Set the source that will drive the global clock.
- DM35425LIB_API int DM35425_Dac_Channel_Setup (struct DM35425_Board_Descriptor ∗handle, struct DM35425_Function_Block ∗func_block, unsigned int channel, enum DM35425_Output_Ranges output_↩ range)

    Setup the selected DAC channel.
- DM35425LIB_API int DM35425_Dac_Channel_Reset (struct DM35425_Board_Descriptor ∗handle, struct DM35425_Function_Block ∗func_block, unsigned int channel)

    Reset the DAC channel, by writing all zeros to the front-end config.
- DM35425LIB_API int DM35425_Dac_Channel_Set_Marker_Config (struct DM35425_Board_Descriptor ∗handle, struct DM35425_Function_Block ∗func_block, unsigned int channel, uint8_t marker_enable)

    Set the configuration of the marker interrupts for this channel.
- DM35425LIB_API int DM35425_Dac_Channel_Get_Marker_Config (struct DM35425_Board_Descriptor ∗handle, struct DM35425_Function_Block ∗func_block, unsigned int channel, uint8_t ∗marker_enable)

    Get the configuration of the marker interrupts for this channel.
- DM35425LIB_API int DM35425_Dac_Channel_Get_Marker_Status (struct DM35425_Board_Descriptor ∗handle, struct DM35425_Function_Block ∗func_block, unsigned int channel, uint8_t ∗marker_status)

    Get the status of the marker interrupts for this channel.
- DM35425LIB_API int DM35425_Dac_Channel_Clear_Marker_Status (struct DM35425_Board_Descriptor ∗handle, struct DM35425_Function_Block ∗func_block, unsigned int channel, uint8_t marker_to_clear)

Clear the marker interrupts for this channel.
- DM35425LIB_API int DM35425_Dac_Fifo_Channel_Write (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel, int32_t value)

  Write a value to the onboard FIFO.
- DM35425LIB_API int DM35425_Dac_Volts_To_Conv (enum DM35425_Output_Ranges output_range, float volts, int16_t ∗dac_conversion)

  Convert a value in volts to a DAC equivalent signed value.
- DM35425LIB_API int DM35425_Dac_Conv_To_Volts (enum DM35425_Output_Ranges output_range, int16_t conversion, float ∗volts)

  Convert a DAC conversion value to volts.

6.19.1 Detailed Description

Definitions for the DM35425 DAC Library.

Id

dm35425_dac_library.h 106898 2017-03-08 13:44:23Z rgroner

6.20 include/dm35425_dma_library.h File Reference

Definitions for the DM35425 DMA Library.

```
#include <stdint.h>
```

Macros

- #define DM35425_DMA_ACTION_CLEAR 0x00

  Register value for DMA clear action.
- #define DM35425_DMA_ACTION_GO 0x01

  Register value for DMA go action.
- #define DM35425_DMA_ACTION_PAUSE 0x02

  Register value for DMA pause action.
- #define DM35425_DMA_ACTION_HALT 0x03

  Register value for DMA halt action.
- #define DM35425_DMA_SETUP_DIRECTION_READ 0x04

  Register value to set DMA to READ direction.
- #define DM35425_DMA_SETUP_DIRECTION_WRITE 0x00

  Register value to set DMA to WRITE direction.
- #define DM35425_DMA_SETUP_DIRECTION_MASK 0x04

  Register value to set DMA to READ direction.
- #define DM35425_DMA_SETUP_IGNORE_USED 0x08

  Register value to tell DMA to ignore used buffers.
- #define DM35425_DMA_SETUP_NOT_IGNORE_USED 0x00

  Register value to tell DMA to not ignore used buffers.
- #define DM35425_DMA_SETUP_IGNORE_USED_MASK 0x08

  Bit mask for Ignore Used bit in setup register.

- #define DM35425_DMA_SETUP_INT_ENABLE 0x01

  Register value to enabled interrupts in the setup register.
- #define DM35425_DMA_SETUP_INT_DISABLE 0x00

  Register value to disable interrupts in the setup register.
- #define DM35425_DMA_SETUP_INT_MASK 0x01

  Bit mask for the interrupt bit in the setup register.
- #define DM35425_DMA_SETUP_ERR_INT_ENABLE 0x02

  Register value to enable the error interrupt.
- #define DM35425_DMA_SETUP_ERR_INT_DISABLE 0x00

  Register value to disable the error interrupt.
- #define DM35425_DMA_SETUP_ERR_INT_MASK 0x02

  Bit mask for the error interrupt bit in the setup register.
- #define DM35425_DMA_STATUS_CLEAR 0x00

  Register value to write to status registers to clear them.
- #define DM35425_DMA_CTRL_CLEAR 0x00

  Register value to write to control register to clear it.
- #define DM35425_DMA_BUFFER_STATUS_CLEAR 0x00

  Register value to write to the buffer status register to clear it.
- #define DM35425_DMA_BUFFER_CTRL_CLEAR 0x00

  Register value to write to the buffer control register to clear it.
- #define DM35425_DMA_BUFFER_STATUS_USED_MASK 0x01

  Bit mask for the used buffer bit in the buffer status register.
- #define DM35425_DMA_BUFFER_STATUS_TERM_MASK 0x02

  Bit mask for the terminated buffer bit in the buffer status register.
- #define DM35425_DMA_BUFFER_CTRL_VALID 0x01

  Register value to write to buffer control register to mark it as valid.
- #define DM35425_DMA_BUFFER_CTRL_HALT 0x02

  Register value to write to buffer control register to tell DMA to halt after processing this buffer.
- #define DM35425_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 DM35425_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 DM35425_DMA_BUFFER_CTRL_PAUSE 0x10

  Register value to write to buffer control register to tell DMA to pause after processing this buffer.
- #define DM35425_DMA_CTRL_BLOCK_SIZE 0x10

  Constant value indicating DMA control block size.
- #define DM35425_DMA_BUFFER_CTRL_BLOCK_SIZE 0x10

  Constant value indicating DMA buffer control block size.
- #define DM35425_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 DM35425_Fifo_States { DM35425_FIFO_UNKNOWN, DM35425_FIFO_EMPTY, DM35425_FIFO_FULL, DM35425_FIFO_HAS_DATA }

  Descriptions of the possible states the FIFO might be in.

Functions

- DM35425LIB_API int DM35425_Dma_Start (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel)

  Start the DMA.
- DM35425LIB_API int DM35425_Dma_Stop (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel)

  Stop the DMA.
- DM35425LIB_API int DM35425_Dma_Pause (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel)

  Pause the DMA.
- DM35425LIB_API int DM35425_Dma_Clear (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel)

  Clear the DMA.
- DM35425LIB_API int DM35425_Dma_Get_Fifo_Counts (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel, uint16_t ∗write_count, uint16_t ∗read←_count)

  Get the Read and Write FIFO count values.
- DM35425LIB_API int DM35425_Dma_Get_Fifo_State (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel, enum DM35425_Fifo_States ∗state)

  Get the state of the FIFO.
- DM35425LIB_API int DM35425_Dma_Configure_Interrupts (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel, int enable, int error_enable)

  Configure the interrupts for the DMA channel.
- DM35425LIB_API int DM35425_Dma_Get_Interrupt_Configuration (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel, int ∗enable, int ∗error←_enable)

  Get the configuration of the interrupts for the DMA channel.
- DM35425LIB_API int DM35425_Dma_Setup (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_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.
- DM35425LIB_API int DM35425_Dma_Setup_Set_Direction (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel, int direction)

  Set the direction of the DMA, read or write.
- DM35425LIB_API int DM35425_Dma_Setup_Set_Used (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_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.
- DM35425LIB_API int DM35425_Dma_Get_Errors (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_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.
- DM35425LIB_API int DM35425_Dma_Status (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_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.
- DM35425LIB_API int DM35425_Dma_Get_Current_Buffer_Count (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_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.
- DM35425LIB_API int DM35425_Dma_Check_For_Error (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_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.

- DM35425LIB_API int DM35425_Dma_Buffer_Setup (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, unsigned int channel, unsigned int buffer, uint8_t ctrl)

  Setup the DMA buffer for use.
- DM35425LIB_API int DM35425_Dma_Buffer_Status (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_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.
- DM35425LIB_API int DM35425_Dma_Check_Buffer_Used (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_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 DM35425_Dma_Find_Interrupt (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_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 DM35425_Dma_Clear_Interrupt (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_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 DM35425_Dma_Reset_Buffer (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_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 DM35425 DMA Library.

Id

dm35425_dma_library.h 105018 2016-12-07 15:47:31Z rgroner

6.21 include/dm35425_driver.h File Reference

Structures and defines for the DM35425 driver module.

```
#include <linux/pci.h>
#include <linux/spinlock.h>
#include <linux/types.h>
```

Data Structures

- struct dm35425_pci_region

  DM35425 PCI region descriptor. This structure holds information about one of a device's PCI memory regions.
- struct dm35425_dma_descriptor

  DM35425 DMA descriptor. This structure holds information about a single DMA buffer.
- struct dm35425_device_descriptor

  DM35425 Device Descriptor. The identifying info for this particular board.

Macros

- #define DM35425_NAME_LENGTH 200

  DM35425 Max possible board name length.
- #define DM35425_PCI_NUM_REGIONS PCI_ROM_RESOURCE

  Number of standard PCI regions.
- #define DM35425_INT_QUEUE_SIZE 256

  Number of interrupts to hold in a queue for processing.

Enumerations

- enum dm35425_pci_region_access_dir { DM35425_PCI_REGION_ACCESS_READ = 0, DM35425_PCI_REGION_ACCESS_
  }

  Direction of access to standard PCI region.

Variables

- static struct file_operations dm35425_file_ops

  Placeholder protoype for file ops struct.

6.21.1 Detailed Description

Structures and defines for the DM35425 driver module.

Id

dm35425_driver.h 80523 2014-07-17 18:38:59Z rgroner

6.22 include/dm35425_examples.h File Reference

Defines for the DM35425 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 DM35425 Example programs. Commonly used constants for the example programs included with the software package.

Id

dm35425_examples.h 114740 2018-07-12 14:41:17Z prucz

6.23 include/dm35425_ext_clocking_library.h File Reference

Definitions for the DM35425 External Clocking Library.

```
#include "dm35425_gbc_library.h"
```

Functions

- DM35425LIB_API int DM35425_Ext_Clocking_Open (struct DM35425_Board_Descriptor ∗handle, unsigned int number_of_type, struct DM35425_Function_Block ∗func_block)

    Open the Global Clocking functional block, making it available for operations.

- DM35425LIB_API int DM35425_Ext_Clocking_Get_In (struct DM35425_Board_Descriptor ∗handle, struct DM35425_Function_Block ∗func_block, uint8_t ∗clk_curr_val)

    Get the current value on the external clocking pins.

- DM35425LIB_API int DM35425_Ext_Clocking_Get_Gate_In (struct DM35425_Board_Descriptor ∗handle, struct DM35425_Function_Block ∗func_block, uint8_t ∗gate_curr_val)

    Get the current value on the external clocking gate pins.

- DM35425LIB_API int DM35425_Ext_Clocking_Get_Dir (struct DM35425_Board_Descriptor ∗handle, struct DM35425_Function_Block ∗func_block, uint8_t ∗dir)

    Get the current value of the external clocking pin direction.

- DM35425LIB_API int DM35425_Ext_Clocking_Set_Dir (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint8_t dir)

    Set the current value of the external clocking pin direction.

- DM35425LIB_API int DM35425_Ext_Clocking_Get_Edge (struct DM35425_Board_Descriptor ∗handle, struct DM35425_Function_Block ∗func_block, uint8_t ∗edge_detect)

    Get the current value of the external clocking edge detect.

- DM35425LIB_API int DM35425_Ext_Clocking_Set_Edge (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_Function_Block ∗func_block, uint8_t edge_detect)

    Set the current value of the external clocking edge detect.

- DM35425LIB_API int DM35425_Ext_Clocking_Get_Pulse_Width (struct DM35425_Board_Descriptor ∗handle, struct DM35425_Function_Block ∗func_block, enum DM35425_Clock_Sources clock_src, uint8_t ∗pulse_width)

    Get the pulse width setting for a specific external clock.

- DM35425LIB_API int DM35425_Ext_Clocking_Set_Pulse_Width (struct DM35425_Board_Descriptor ∗handle, struct DM35425_Function_Block ∗func_block, enum DM35425_Clock_Sources clock_src, uint8_t pulse_width)

    Set the pulse width setting for a specific external clock.

- DM35425LIB_API int DM35425_Ext_Clocking_Get_Method (struct DM35425_Board_Descriptor ∗handle, struct DM35425_Function_Block ∗func_block, enum DM35425_Clock_Sources clock_src, enum DM35425←_Ext_Clocking_Method ∗clocking_method)

    Get the setting for a specific external clock.

- DM35425LIB_API int DM35425_Ext_Clocking_Set_Method (struct DM35425_Board_Descriptor ∗handle, struct DM35425_Function_Block ∗func_block, enum DM35425_Clock_Sources clock_src, enum DM35425←_Ext_Clocking_Method clocking_method)

    Set the setting for a specific external clock.

6.23.1 Detailed Description

Definitions for the DM35425 External Clocking Library.

Id

dm35425_ext_clocking_library.h 60276 2012-06-05 16:04:15Z rgroner

6.24 include/dm35425_gbc_library.h File Reference

Definitions for the DM35425 Board Library, a library for accessing the board registers.

```
#include <stdint.h>
#include <time.h>
#include "dm35425_board_access.h"
```

Macros

- #define CLK_40MHZ 40000000

Functions

- DM35425LIB_API int DM35425_Gbc_Board_Reset (struct DM35425_Board_Descriptor ∗handle)

  Write the reset value to the correct register to initiate a board-level reset.
- DM35425LIB_API int DM35425_Gbc_Ack_Interrupt (struct DM35425_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.
- DM35425LIB_API int DM35425_Function_Block_Open (struct DM35425_Board_Descriptor ∗handle, unsigned int number, struct DM35425_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.
- DM35425LIB_API int DM35425_Function_Block_Open_Module (struct DM35425_Board_Descriptor ∗handle, uint32_t fb_type, unsigned int number_of_type, struct DM35425_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.
- DM35425LIB_API int DM35425_Gbc_Get_Format (struct DM35425_Board_Descriptor ∗handle, uint8_↩
t ∗format_id)

  Get the format ID of the board.
- DM35425LIB_API int DM35425_Gbc_Get_Revision (struct DM35425_Board_Descriptor ∗handle, uint8_↩
t ∗rev)

  Get the PDP revision number of the board.
- DM35425LIB_API int DM35425_Gbc_Get_Pdp_Number (struct DM35425_Board_Descriptor ∗handle, uint32_t ∗pdp_num)

  Get PDP Number of the board.
- DM35425LIB_API int DM35425_Gbc_Get_Fpga_Build (struct DM35425_Board_Descriptor ∗handle, uint32_t ∗fpga_build)

  Get the FPGA Build number of the board.
- DM35425LIB_API int DM35425_Gbc_Get_Sys_Clock_Freq (struct DM35425_Board_Descriptor ∗handle, uint32_t ∗clock_freq, int ∗is_std_clk)

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

6.24.1 Detailed Description

Definitions for the DM35425 Board Library, a library for accessing the board registers.

Id

dm35425_gbc_library.h 103741 2016-10-17 20:35:58Z rgroner

6.25 include/dm35425_ioctl.h File Reference

DM35425 Low level ioctl() request descriptor structure and request code definitions.

```
#include <linux/types.h>
#include <linux/ioctl.h>
```

Macros

- #define DM35425_IOCTL_MAGIC 'D'

  Unique 8-bit value used to generate unique ioctl() request codes.
- #define DM35425_IOCTL_REQUEST_BASE 0x00

  First ioctl() request number.
- #define DM35425_IOCTL_REGION_READ

  ioctl() request code for reading from a PCI region
- #define DM35425_IOCTL_REGION_WRITE

  ioctl() request code for writing to a PCI region
- #define DM35425_IOCTL_REGION_MODIFY

  ioctl() request code for PCI region read/modify/write
- #define DM35425_IOCTL_DMA_FUNCTION

  ioctl() request code for DMA function
- #define DM35425_IOCTL_WAKEUP

  ioctl() request code for User ISR thread wake up
- #define DM35425_IOCTL_INTERRUPT_GET

  ioctl() request code to retrieve interrupt status information

6.25.1 Detailed Description

DM35425 Low level ioctl() request descriptor structure and request code definitions.

Id

dm35425_ioctl.h 80523 2014-07-17 18:38:59Z rgroner

6.26 include/dm35425_os.h File Reference

Function declarations for the DM35425 that are Linux specific.

```
#include <pthread.h>
```

Data Structures

- struct DM35425_Board_Descriptor

  DM35425 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 DM35425_Dma_Initialize (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_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 DM35425_Dma_Read (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_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 DM35425_Dma_Write (struct DM35425_Board_Descriptor ∗handle, const struct DM35425_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 DM35425_General_RemoveISR (struct DM35425_Board_Descriptor ∗handle)

  Remove the ISR from the system interrupt.
- void ∗ DM35425_General_WaitForInterrupt (void ∗ptr)

  Loop/Poll and wait for an interrupt to happen, then take action.
- int DM35425_General_InstallISR (struct DM35425_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 DM35425_General_SetISRPriority (struct DM35425_Board_Descriptor ∗handle, int priority)

  Set the priority of the user ISR thread.

6.26.1 Detailed Description

Function declarations for the DM35425 that are Linux specific.

Id

dm35425_os.h 109114 2017-06-09 07:13:20Z prucz

6.27 include/dm35425_registers.h File Reference

Defines for the DM35425 Registers (Offsets)

Macros

- #define DM35425_OFFSET_GBC_FORMAT 0x00

  Offset to General Board Control (BAR0) Format ID register.
- #define DM35425_OFFSET_GBC_REV 0x01

  Offset to General Board Control (BAR0) Format ID register.
- #define DM35425_OFFSET_GBC_END_INTERRUPT 0x02

  Offset to General Board Control (BAR0) EOI (End of Interrupt) register.
- #define DM35425_OFFSET_GBC_BOARD_RESET 0x03

  Offset to General Board Control (BAR0) Board Reset register.
- #define DM35425_OFFSET_GBC_PDP_NUMBER 0x04

  Offset to General Board Control (BAR0) PDP Number register.
- #define DM35425_OFFSET_GBC_FPGA_BUILD 0x08

  Offset to General Board Control (BAR0) FPGA Build register.

- #define DM35425_OFFSET_GBC_SYS_CLK_FREQ 0x0c

    Offset to General Board Control (BAR0) System Clock register.
- #define DM35425_OFFSET_GBC_IRQ_STATUS 0x10

    Offset to General Board Control (BAR0) IRQ Status register. Each bit corresponds to a function block.
- #define DM35425_OFFSET_GBC_DMA_IRQ_STATUS 0x18

    Offset to General Board Control (BAR0) DMA IRQ Status register. Each bit corresponds to a function block.
- #define DM35425_OFFSET_GBC_FB_START 0x20

    Offset to the beginning of the Function Blocks section of the GBC.
- #define DM35425_GBC_FB_BLK_SIZE 0x10

    Size of the function block entries in the GBC.
- #define DM35425_OFFSET_GBC_FB_ID 0x00

    Offset to Function Block ID, from the start of the function block section.
- #define DM35425_FB_ID_TYPE_MASK 0x0000FFFF

    Bit mask for TYPE portion of FB ID.
- #define DM35425_FB_ID_SUBTYPE_MASK 0x00FF0000

    Bit mask for SUBTYPE portion of FB ID.
- #define DM35425_FB_ID_TYPE_REV_MASK 0xFF000000

    Bit mask for TYPE REV portion of FB ID.
- #define DM35425_OFFSET_GBC_FB_OFFSET 0x04

    Offset to the FB Offset in the GBC, from the start of the FB data block.
- #define DM35425_OFFSET_GBC_FB_DMA_OFFSET 0x08

    Offset to the FB DMA Offset in the GBC, from the start of the FB data block.
- #define DM35425_OFFSET_DMA_ACTION 0x00

    Offset to the DMA Action Register (BAR2)
- #define DM35425_OFFSET_DMA_SETUP 0x01

    Offset to the DMA Setup Register (BAR2)
- #define DM35425_OFFSET_DMA_STAT_OVERFLOW 0x02

    Offset to the DMA Status (Overflow) Register (BAR2)
- #define DM35425_OFFSET_DMA_STAT_UNDERFLOW 0x03

    Offset to the DMA Status (Underflow) Register (BAR2)
- #define DM35425_OFFSET_DMA_CURRENT_COUNT 0x04

    Offset to the DMA Current Count Register (BAR2)
- #define DM35425_OFFSET_DMA_CURRENT_BUFFER 0x07

    Offset to the DMA Current Buffer Register (BAR2)
- #define DM35425_OFFSET_DMA_WR_FIFO_CNT 0x08

    Offset to the DMA Write FIFO Count Register (BAR2)
- #define DM35425_OFFSET_DMA_RD_FIFO_CNT 0x0A

    Offset to the DMA Read FIFO Count Register (BAR2)
- #define DM35425_OFFSET_DMA_STAT_USED 0x0C

    Offset to the DMA Status (Used) Register (BAR2)
- #define DM35425_OFFSET_DMA_STAT_INVALID 0x0D

    Offset to the DMA Status (Invalid) Register (BAR2)
- #define DM35425_OFFSET_DMA_STAT_COMPLETE 0x0E

    Offset to the DMA Status (Complete) Register (BAR2)
- #define DM35425_OFFSET_DMA_LAST_ACTION 0x0F

    Offset to the DMA Last Action Register (BAR2)
- #define DM35425_OFFSET_DMA_BUFF_START 0x10

    Offset to the start of the buffer control section (BAR2)
- #define DM35425_OFFSET_DMA_BUFFER_STAT 0x02

    Offset to the buffer status register, from the start of the buffer control section (BAR2)
- #define DM35425_OFFSET_DMA_BUFFER_CTRL 0x03

Offset to the buffer control register, from the start of the buffer control section (BAR2)
- #define DM35425_OFFSET_DMA_BUFFER_SIZE 0x04

  Offset to the buffer size register, from the start of the buffer control section (BAR2)
- #define DM35425_OFFSET_DMA_BUFFER_ADDRESS 0x08

  Offset to the buffer address register, from the start of the buffer control section (BAR2)
- #define DM35425_OFFSET_FB_DMA_CHANNELS 0x06

  Offset to the DMA Channels count of the function block (BAR2)
- #define DM35425_OFFSET_FB_DMA_BUFFERS 0x07

  Offset to the DMA buffers count of the function block (BAR2)
- #define DM35425_OFFSET_FB_CTRL_START 0x08

  Offset to the beginning of the Function Block control section in BAR2.
- #define DM35425_OFFSET_ADC_MODE_STATUS 0x00

  Offset to the ADC Mode-Status register, from the start of the ADC control section.
- #define DM35425_OFFSET_ADC_CLK_SRC 0x01

  Offset to the ADC Clock Source register, from the start of the ADC control section.
- #define DM35425_OFFSET_ADC_START_TRIG 0x02

  Offset to the ADC Start Trigger register, from the start of the ADC control section.
- #define DM35425_OFFSET_ADC_STOP_TRIG 0x03

  Offset to the ADC Stop Trigger register, from the start of the ADC control section.
- #define DM35425_OFFSET_ADC_CLK_DIV 0x04

  Offset to the ADC Clock Divider register, from the start of the ADC control section.
- #define DM35425_OFFSET_ADC_CLK_DIV_COUNTER 0x08

  Offset to the ADC Clock Divider Counter register, from the start of the ADC control section.
- #define DM35425_OFFSET_ADC_PRE_CAPT_COUNT 0x0c

  Offset to the ADC Pre-Start Capture Count register, from the start of the ADC control section.
- #define DM35425_OFFSET_ADC_POST_CAPT_COUNT 0x10

  Offset to the ADC Post-Stop Capture Count register, from the start of the ADC control section.
- #define DM35425_OFFSET_ADC_SAMPLE_COUNT 0x14

  Offset to the ADC Sample Count register, from the start of the ADC control section.
- #define DM35425_OFFSET_ADC_INT_ENABLE 0x18

  Offset to the ADC Interrupt Enable register, from the start of the ADC control section.
- #define DM35425_OFFSET_ADC_INT_STAT 0x1e

  Offset to the ADC Interrupt Status register, from the start of the ADC control section.
- #define DM35425_OFFSET_ADC_CLK_BUS2 0x22

  Offset to the ADC Clock Bus 2, from the start of the ADC control section.
- #define DM35425_OFFSET_ADC_CLK_BUS3 0x23

  Offset to the ADC Clock Bus 3 register, from the start of the ADC control section.
- #define DM35425_OFFSET_ADC_CLK_BUS4 0x24

  Offset to the ADC Clock Bus 4 register, from the start of the ADC control section.
- #define DM35425_OFFSET_ADC_CLK_BUS5 0x25

  Offset to the ADC Clock Bus 5 register, from the start of the ADC control section.
- #define DM35425_OFFSET_ADC_CLK_BUS6 0x26

  Offset to the ADC Clock Bus 6 register, from the start of the ADC control section.
- #define DM35425_OFFSET_ADC_CLK_BUS7 0x27

  Offset to the ADC Clock Bus 7 register, from the start of the ADC control section.
- #define DM35425_OFFSET_ADC_AD_CONFIG 0x28

  Offset to the ADC AD Config register, from the start of the ADC control section.
- #define DM35425_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 DM35425_ADC_CHAN_CTRL_BLK_SIZE 0x18

  Constant size of ADC channel section in function block.

- #define DM35425_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 DM35425_OFFSET_ADC_CHAN_DATA_COUNT 0x04

  Offset to the Channel FIFO Data count register, from the start of the ADC channel control section.
- #define DM35425_OFFSET_ADC_CHAN_FILTER 0x09

  Offset to the Channel Filter register, from the start of the ADC channel control section.
- #define DM35425_OFFSET_ADC_CHAN_INTR_STAT 0x0a

  Offset to the Channel Interrupt Status register, from the start of the ADC channel control section.
- #define DM35425_OFFSET_ADC_CHAN_INTR_ENABLE 0x0b

  Offset to the Channel Interrupt Enable register, from the start of the ADC channel control section.
- #define DM35425_OFFSET_ADC_CHAN_LOW_THRESHOLD 0x0c

  Offset to the Channel Low Threshold register, from the start of the ADC channel control section.
- #define DM35425_OFFSET_ADC_CHAN_HIGH_THRESHOLD 0x10

  Offset to the Channel High Threshold register, from the start of the ADC channel control section.
- #define DM35425_OFFSET_ADC_CHAN_LAST_SAMPLE 0x14

  Offset to the Channel Last Sample register, from the start of the ADC channel control section.
- #define DM35425_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 DM35425_ADC_FIFO_CTRL_BLK_SIZE 0x4

  Constant size of ADC FIFO section in function block.
- #define DM35425_OFFSET_FB_ADC_FIFO 0x0334

  Offset to the FIFO for non-DMA read and write operations.
- #define DM35425_OFFSET_DAC_MODE_STATUS 0x00

  Offset to the Mode/Status register, from the start of the DAC control section.
- #define DM35425_OFFSET_DAC_CLK_SRC 0x01

  Offset to the Clock Source register, from the start of the DAC control section.
- #define DM35425_OFFSET_DAC_START_TRIG 0x02

  Offset to the Start Trigger register, from the start of the DAC control section.
- #define DM35425_OFFSET_DAC_STOP_TRIG 0x03

  Offset to the Stop Trigger register, from the start of the DAC control section.
- #define DM35425_OFFSET_DAC_CLK_DIV 0x04

  Offset to the Clock Divider register, from the start of the DAC control section.
- #define DM35425_OFFSET_DAC_CLK_DIV_COUNT 0x08

  Offset to the Clock Divider Counter register, from the start of the DAC control section.
- #define DM35425_OFFSET_DAC_POST_STOP_CONV 0x10

  Offset to the Post-Stop Conversion Count register, from the start of the DAC control section.
- #define DM35425_OFFSET_DAC_CONV_COUNT 0x14

  Offset to the Conversion Count register, from the start of the DAC control section.
- #define DM35425_OFFSET_DAC_INT_ENABLE 0x18

  Offset to the Interrupt Enable register, from the start of the DAC control section.
- #define DM35425_OFFSET_DAC_INT_STAT 0x1e

  Offset to the Interrupt Status register, from the start of the DAC control section.
- #define DM35425_OFFSET_DAC_CLK_BUS2 0x22

  Offset to the Clock Bus 2 register, from the start of the DAC control section.
- #define DM35425_OFFSET_DAC_CLK_BUS3 0x23

  Offset to the Clock Bus 3 register, from the start of the DAC control section.
- #define DM35425_OFFSET_DAC_CLK_BUS4 0x24

  Offset to the Clock Bus 4 register, from the start of the DAC control section.
- #define DM35425_OFFSET_DAC_CLK_BUS5 0x25

  Offset to the Clock Bus 5 register, from the start of the DAC control section.
- #define DM35425_OFFSET_DAC_CLK_BUS6 0x26

Offset to the Clock Bus 6 register, from the start of the DAC control section.

- #define DM35425_OFFSET_DAC_CLK_BUS7 0x27

  Offset to the Clock Bus 7 register, from the start of the DAC control section.
- #define DM35425_OFFSET_DAC_DA_CONFIG 0x28

  Offset to the DA Config register, from the start of the DAC control section.
- #define DM35425_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 DM35425_DAC_CHAN_CTRL_BLK_SIZE 0x14

  Constant size of channel control section in function block.
- #define DM35425_OFFSET_DAC_CHAN_FRONT_END_CONFIG 0x00

  Offset to the Front-End Config register, from the start of the DAC channel control section.
- #define DM35425_OFFSET_DAC_CHAN_MARKER_STATUS 0x0a

  Offset to the Channel marker Interrupt Status register, from the start of the DAC channel control section.
- #define DM35425_OFFSET_DAC_CHAN_MARKER_ENABLE 0x0b

  Offset to the Channel marker Interrupt Enable register, from the start of the DAC channel control section.
- #define DM35425_OFFSET_DAC_CHAN_LAST_CONVERSION 0x10

  Offset to the Channel Last Conversion register, from the start of the DAC channel control section.
- #define DM35425_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 DM35425_OFFSET_DAC_FIFO_CTRL_BLK_SIZE 0x4

  Constant size of FIFO control section in function block.
- #define DM35425_OFFSET_ADIO_MODE_STATUS 0x00

  Offset to the ADIO Mode-Status register, from the start of the ADIO control section.
- #define DM35425_OFFSET_ADIO_CLK_SRC 0x01

  Offset to the ADIO Clock Source register, from the start of the ADIO control section.
- #define DM35425_OFFSET_ADIO_START_TRIG 0x02

  Offset to the ADIO Start Trigger register, from the start of the ADIO control section.
- #define DM35425_OFFSET_ADIO_STOP_TRIG 0x03

  Offset to the ADIO Stop Trigger register, from the start of the ADIO control section.
- #define DM35425_OFFSET_ADIO_CLK_DIV 0x04

  Offset to the ADIO Clock Divider register, from the start of the ADIO control section.
- #define DM35425_OFFSET_ADIO_CLK_DIV_COUNTER 0x08

  Offset to the ADIO Clock Divider Counter register, from the start of the ADIO control section.
- #define DM35425_OFFSET_ADIO_PRE_CAPT_COUNT 0x0c

  Offset to the ADIO Pre-Start Capture Count register, from the start of the ADIO control section.
- #define DM35425_OFFSET_ADIO_POST_CAPT_COUNT 0x10

  Offset to the ADIO Post-Stop Capture Count register, from the start of the ADIO control section.
- #define DM35425_OFFSET_ADIO_SAMPLE_COUNT 0x14

  Offset to the ADIO Sample Count register, from the start of the ADIO control section.
- #define DM35425_OFFSET_ADIO_INT_ENABLE 0x18

  Offset to the ADIO Interrupt Enable register, from the start of the ADIO control section.
- #define DM35425_OFFSET_ADIO_INT_STAT 0x1e

  Offset to the ADIO Interrupt Status register, from the start of the ADIO control section.
- #define DM35425_OFFSET_ADIO_CLK_BUS2 0x22

  Offset to the ADIO Clock Bus 2, from the start of the ADIO control section.
- #define DM35425_OFFSET_ADIO_CLK_BUS3 0x23

  Offset to the ADIO Clock Bus 3 register, from the start of the ADIO control section.
- #define DM35425_OFFSET_ADIO_CLK_BUS4 0x24

  Offset to the ADIO Clock Bus 4 register, from the start of the ADIO control section.
- #define DM35425_OFFSET_ADIO_CLK_BUS5 0x25

  Offset to the ADIO Clock Bus 5 register, from the start of the ADIO control section.

- #define DM35425_OFFSET_ADIO_CLK_BUS6 0x26

  Offset to the ADIO Clock Bus 6 register, from the start of the ADIO control section.
- #define DM35425_OFFSET_ADIO_CLK_BUS7 0x27

  Offset to the ADIO Clock Bus 7 register, from the start of the ADIO control section.
- #define DM35425_OFFSET_ADIO_CHAN_START 0x28

  Offset to the beginning of the channels control section from the start of the control section.
- #define DM35425_OFFSET_ADIO_INPUT_VAL 0x00

  Offset to the Input Value register, from the start of the ADIO channel control section.
- #define DM35425_OFFSET_ADIO_OUTPUT_VAL 0x04

  Offset to the Output Value register, from the start of the ADIO channel control section.
- #define DM35425_OFFSET_ADIO_DIRECTION 0x08

  Offset to the Direction register, from the start of the ADIO channel control section.
- #define DM35425_OFFSET_ADIO_ADV_INT_MODE 0x0C

  Offset to the Advanced Interrupt mode register, from the start of the ADIO channel control section.
- #define DM35425_OFFSET_ADIO_ADV_INT_MASK 0x10

  Offset to the Advanced Interrupt mask register, from the start of the ADIO channel control section.
- #define DM35425_OFFSET_ADIO_ADV_INT_COMP 0x14

  Offset to the Advanced Interrupt compare register, from the start of the ADIO channel control section.
- #define DM35425_OFFSET_ADIO_ADV_INT_CAPT 0x18

  Offset to the Advanced Interrupt capture register, from the start of the ADIO channel control section.
- #define DM35425_OFFSET_ADIO_P_BUS_ENABLE 0x1C

  Offset to the Advanced Interrupt parallel bus enable register, from the start of the ADIO channel control section.
- #define DM35425_OFFSET_ADIO_P_BUS_READY_ENABLE 0x1D

  Offset to the Advanced Interrupt parallel bus ready register, from the start of the ADIO channel control section.
- #define DM35425_OFFSET_ADIO_FIFO_CTRL_BLK_START 0x50

  Offset to the start of the ADIO FIFO control section, from the start of the ADIO function block.
- #define DM35425_OFFSET_ADIO_FIFO_CTRL_BLK_SIZE 0x4

  Constant size of FIFO control section in function block.
- #define DM35425_OFFSET_EXT_CLOCKING_IN 0x00

  Offset to the pin value register, from the start of the External Clocking control section.
- #define DM35425_OFFSET_EXT_CLOCKING_GATE_IN 0x01

  Offset to the gate pin value register, from the start of the External Clocking control section.
- #define DM35425_OFFSET_EXT_CLOCKING_DIR 0x02

  Offset to the direction register, from the start of the External Clocking control section.
- #define DM35425_OFFSET_EXT_CLOCKING_EDGE 0x03

  Offset to the edge detect register, from the start of the External Clocking control section.
- #define DM35425_OFFSET_EXT_CLOCKING_PW2 0x04

  Offset to the pulse width (CLK2) register, from the start of the External Clocking control section.
- #define DM35425_OFFSET_EXT_CLOCKING_PW3 0x05

  Offset to the pulse width (CLK3) register, from the start of the External Clocking control section.
- #define DM35425_OFFSET_EXT_CLOCKING_PW4 0x06

  Offset to the pulse width (CLK4) register, from the start of the External Clocking control section.
- #define DM35425_OFFSET_EXT_CLOCKING_PW5 0x07

  Offset to the pulse width (CLK5) register, from the start of the External Clocking control section.
- #define DM35425_OFFSET_EXT_CLOCKING_PW6 0x08

  Offset to the pulse width (CLK6) register, from the start of the External Clocking control section.
- #define DM35425_OFFSET_EXT_CLOCKING_PW7 0x09

  Offset to the pulse width (CLK7) register, from the start of the External Clocking control section.
- #define DM35425_OFFSET_EXT_CLOCKING_SETUP_GBL2 0x0A

  Offset to the clocking method (CLK2) register, from the start of the External Clocking control section.
- #define DM35425_OFFSET_EXT_CLOCKING_SETUP_GBL3 0x0B

Offset to the clocking method (CLK3) register, from the start of the External Clocking control section.
- #define DM35425_OFFSET_EXT_CLOCKING_SETUP_GBL4 0x0C

  Offset to the clocking method (CLK4) register, from the start of the External Clocking control section.
- #define DM35425_OFFSET_EXT_CLOCKING_SETUP_GBL5 0x0D

  Offset to the clocking method (CLK5) register, from the start of the External Clocking control section.
- #define DM35425_OFFSET_EXT_CLOCKING_SETUP_GBL6 0x0E

  Offset to the clocking method (CLK6) register, from the start of the External Clocking control section.
- #define DM35425_OFFSET_EXT_CLOCKING_SETUP_GBL7 0x0F

  Offset to the clocking method (CLK7) register, from the start of the External Clocking control section.

6.27.1 Detailed Description

Defines for the DM35425 Registers (Offsets)

Id

dm35425_registers.h 124951 2020-03-05 16:34:24Z lfrankenfield

6.28 include/dm35425_types.h File Reference

Defines for the DM35425. Values for the general board, not specific to a particular function block.

Macros

- #define DM35425_SUBTYPE_00 0

  Constant for FB subtype 0.
- #define DM35425_SUBTYPE_01 1

  Constant for FB subtype 1.
- #define DM35425_SUBTYPE_02 2

  Constant for FB subtype 2.
- #define DM35425_SUBTYPE_03 3

  Constant for FB subtype 3.
- #define DM35425_SUBTYPE_INVALID 0xFF

  Constant value indicating an invalid subtype.
- #define DM35425_FUNC_BLOCK_INVALID 0x0000

  Constant value indicating an invalid function block.
- #define DM35425_FUNC_BLOCK_INVALID2 0xFFFF

  Constant value indicating an invalid function block.
- #define DM35425_FUNC_BLOCK_SYNCBUS 0x0001

  Function Block Constant for SyncBus.
- #define DM35425_FUNC_BLOCK_EXT_CLOCKING 0x0002

  Function Block Constant for Global Clocking.
- #define DM35425_FUNC_BLOCK_CLK0003 0x0003

  Function Block Constant for External Clocking (0003)
- #define DM35425_FUNC_BLOCK_CAPTWIN 0x0005

  Function Block Constant for Capture Window.
- #define DM35425_FUNC_BLOCK_ADC 0x1000

Function Block Constant for ADC.
- #define DM35425_FUNC_BLOCK_ADC1001 0x1001

  Function Block Constant for 10 MHz ADC (1001)
- #define DM35425_FUNC_BLOCK_DAC 0x2000

  Function Block Constant for DAC.
- #define DM35425_FUNC_BLOCK_DAC2001 0x2001

  Function Block Constant for High Speed DAC (2001)
- #define DM35425_FUNC_BLOCK_DIO 0x3000

  Function Block Constant for DIO.
- #define DM35425_FUNC_BLOCK_ADIO 0x3001

  Function Block Constant for ADIO.
- #define DM35425_FUNC_BLOCK_ADIO3010 0x3010

  Function Block Constant for ADIO3010.
- #define DM35425_FUNC_BLOCK_USART 0x4000

  Function Block Constant for Synchronous/Asynchronous Serial Port.
- #define DM35425_FUNC_BLOCK_REF_ADJUST 0xF000

  Function Block Constant for Reference Adjustment.
- #define DM35425_FUNC_BLOCK_TEMPERATURE_SENSOR 0xF001

  Function Block Constant for Temperature Sensor.
- #define DM35425_FUNC_BLOCK_FLASH_PROGRAMMER 0xF002

  Function Block Constant for Flash Programmer.
- #define DM35425_FUNC_BLOCK_CLK_GEN 0xF003

  Function Block Constant for Clock Generator.
- #define DM35425_FUNC_BLOCK_DIN3011 0x3011

  Function Block Constant for Digital Input (3011)
- #define DM35425_FUNC_BLOCK_DOT3012 0x3012

  Function Block Constant for Digital Output (3012)
- #define DM35425_FUNC_BLOCK_INC3200 0x3200

  Function Block Constant for Incremental Encoder (3200)
- #define DM35425_FUNC_BLOCK_PWM3100 0x3100

  Function Block Constant for PWM (3100)
- #define DM35425_FUNC_BLOCK_CLK0004 0x0004

  Function Block Constant for Programmable Clock (0004)
- #define DM35425_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 DM35425_DMA_MAX_BUFFER_SIZE 0xFFFFFC

  Maximum possible DMA buffer size.
- #define DM35425_BOARD_ACK_INTERRUPT 0x1

  Value to write to the EOI register to acknowledge interrupts.
- #define DM35425_BOARD_RESET_VALUE 0xAA

  Value to write to the Reset register in order to reset the board.
- #define DM35425_FIFO_ACCESS_FB_REVISION 0x01

  Minimum function block revision that supports direct FIFO read/write access.

Enumerations

- enum DM35425_Clock_Sources {
  DM35425_CLK_SRC_IMMEDIATE, DM35425_CLK_SRC_NEVER, DM35425_CLK_SRC_BUS2, DM35425_CLK_SRC_BUS
  DM35425_CLK_SRC_BUS4, DM35425_CLK_SRC_BUS5, DM35425_CLK_SRC_BUS6, DM35425_CLK_SRC_BUS7,
  DM35425_CLK_SRC_CHAN_THRESH = 0x08, DM35425_CLK_SRC_CHAN_THRESH_INV = 0x09,
  DM35425_CLK_SRC_BUS2_INV = 0x0A, DM35425_CLK_SRC_BUS3_INV,
  DM35425_CLK_SRC_BUS4_INV, DM35425_CLK_SRC_BUS5_INV, DM35425_CLK_SRC_BUS6_INV,
  DM35425_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 DM35425_Clock_Buses {
  DM35425_CLK_BUS2 = 2, DM35425_CLK_BUS3, DM35425_CLK_BUS4, DM35425_CLK_BUS5,
  DM35425_CLK_BUS6, DM35425_CLK_BUS7 }

  Clock buses available to the function block.

6.28.1 Detailed Description

Defines for the DM35425. Values for the general board, not specific to a particular function block.

Id

dm35425_types.h 127189 2020-09-16 13:22:33Z lfrankenfield

6.28.2 Enumeration Type Documentation

6.28.2.1 DM35425_Clock_Buses

```
enum DM35425_Clock_Buses
```

Clock buses available to the function block.

Enumerator

DM35425_CLK_BUS2	Clock Bus 2.
DM35425_CLK_BUS3	Clock Bus 3.
DM35425_CLK_BUS4	Clock Bus 4.
DM35425_CLK_BUS5	Clock Bus 5.
DM35425_CLK_BUS6	Clock Bus 6.
DM35425_CLK_BUS7	Clock Bus 7.

Definition at line 346 of file dm35425_types.h.

6.28.2.2 DM35425_Clock_Sources

```
enum DM35425_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.

DM35425_General_Definitions

Enumerator

DM35425_CLK_SRC_IMMEDIATE	Clock Source - Immediate (0x00)
DM35425_CLK_SRC_NEVER	Clock Source - Never (0x01)
DM35425_CLK_SRC_BUS2	Clock Source - Bus 2 (0x02)
DM35425_CLK_SRC_BUS3	Clock Source - Bus 3 (0x03)
DM35425_CLK_SRC_BUS4	Clock Source - Bus 4 (0x04)
DM35425_CLK_SRC_BUS5	Clock Source - Bus 5 (0x05)
DM35425_CLK_SRC_BUS6	Clock Source - Bus 6 (0x06)
DM35425_CLK_SRC_BUS7	Clock Source - Bus 7 (0x07)
DM35425_CLK_SRC_CHAN_THRESH	Clock Source - Threshold Exceeded (0x08)
DM35425_CLK_SRC_CHAN_THRESH_INV	Clock Source - Threshold Inverse (None Exceeded) (0x09)
DM35425_CLK_SRC_BUS2_INV	Clock Source - Bus 2 Inverse (0x0A)
DM35425_CLK_SRC_BUS3_INV	Clock Source - Bus 3 Inverse (0x0B)
DM35425_CLK_SRC_BUS4_INV	Clock Source - Bus 4 Inverse (0x0C)
DM35425_CLK_SRC_BUS5_INV	Clock Source - Bus 5 Inverse (0x0D)
DM35425_CLK_SRC_BUS6_INV	Clock Source - Bus 6 Inverse (0x0E)
DM35425_CLK_SRC_BUS7_INV	Clock Source - Bus 7 Inverse (0x0F)

Definition at line 258 of file dm35425_types.h.

6.29 include/dm35425_util_library.h File Reference

Definitions for the DM35425 Utilities library, various helper functions.

```
#include <time.h>
#include <sys/time.h>
```

Enumerations

- enum DM35425_Waveforms { DM35425_SINE_WAVE, DM35425_SQUARE_WAVE, DM35425_SAWTOOTH_WAVE }

  List of possible waveforms that can be generated for DAC purposes.

Functions

- uint32_t DM35425_Get_Maskable (uint16_t data, uint16_t mask)

  Return a 32-bit maskable register value from the data and mask.
- void DM35425_Micro_Sleep (unsigned long microsecs)

  Sleep for a specified number of microseconds.
- long DM35425_Get_Time_Diff (struct timeval last, struct timeval first)

Calculate the time difference between the two timeval structs, in microseconds.

- int DM35425_Generate_Signal_Data (enum DM35425_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.

6.29.1 Detailed Description

Definitions for the DM35425 Utilities library, various helper functions.

Id

dm35425_util_library.h 114375 2018-06-20 05:58:38Z prucz

Index

access
    dm35425_ioctl_region_modify, 169
    dm35425_ioctl_region_readwrite, 171
AD_MODE_OPTION
    DM35425 Example Programs Constants, 118
ADC_NUM_OPTION
    DM35425 Example Programs Constants, 118
ADC_OPTION
    DM35425 Example Programs Constants, 118
ADIO_RATE
    dm35425_adio_dma.c, 215
allocated
    dm35425_pci_region, 173
ASCII_FILE_NAME
    dm35425_adc_continuous_dma.c, 193

BAUD_OPTION
    DM35425 Example Programs Constants, 119
BIN2TXT_OPTION
    DM35425 Example Programs Constants, 119
BIN_FILE_NAME
    dm35425_adc_continuous_dma.c, 193
BINARY_OPTION
    DM35425 Example Programs Constants, 118
board
    dm35425_adc_all_dma.c, 189
    dm35425_adc_continuous_dma.c, 198
    dm35425_adio.c, 208
    dm35425_adio_adv_int.c, 213
    dm35425_adio_dma.c, 219
    dm35425_adio_parallel_bus.c, 226
buffer
    dm35425_dma_descriptor, 159
    dm35425_ioctl_dma, 165
buffer_copied
    dm35425_adio_dma.c, 219
buffer_count
    dm35425_adc_all_dma.c, 189
    dm35425_adc_continuous_dma.c, 198
buffer_ptr
    dm35425_ioctl_dma, 165
buffer_size
    dm35425_dma_descriptor, 159
    dm35425_ioctl_dma, 166
BUFFER_SIZE_BYTES
    dm35425_adio_dma.c, 216
    dm35425_adio_parallel_bus.c, 222
    dm35425_dac_dma.c, 232
    dm35425_ext_clocking.c, 237
buffer_size_bytes
    dm35425_adc_all_dma.c, 189
    dm35425_adc_continuous_dma.c, 198
BUFFER_SIZE_SAMPLES
    dm35425_adio_dma.c, 216
    dm35425_adio_parallel_bus.c, 223
    dm35425_dac_dma.c, 232
    dm35425_ext_clocking.c, 237
buffer_start_offset
    DM35425_DMA_Descriptor, 158
bus_addr
    dm35425_dma_descriptor, 159

channel
    dm35425_adc_continuous_dma.c, 199
    dm35425_dma_descriptor, 160
    dm35425_ioctl_dma, 166
CHANNELS_OPTION
    DM35425 Example Programs Constants, 118
check_result
    DM35425 Utility Library Functions, 149
CLK_40MHZ
    DM35425 Board Macros, 128
control_offset
    DM35425_DMA_Descriptor, 158
    DM35425_Function_Block, 161
convert_bin_to_txt
    dm35425_adc_continuous_dma.c, 195
COUNT_OPTION
    DM35425 Example Programs Constants, 119

DAC_BUFFER_SIZE_BYTES
    dm35425_adc_continuous_dma.c, 193
DAC_BUFFER_SIZE_SAMPLES
    dm35425_adc_continuous_dma.c, 194
DAC_NUM_OPTION
    DM35425 Example Programs Constants, 118
DAC_OPTION
    DM35425 Example Programs Constants, 118
DAC_RATE
    dm35425_adc_continuous_dma.c, 194
DAT_FILE_NAME_PREFIX
    dm35425_adc_all_dma.c, 186
DAT_FILE_NAME_SUFFIX
    dm35425_adc_all_dma.c, 186
data
    dm35425_pci_access_request, 171
data16
    dm35425_pci_access_request, 172
data32
    dm35425_pci_access_request, 172

data8
dm35425_pci_access_request, 172
DEFAULT_CHANNEL
dm35425_adc.c, 181
dm35425_adc_continuous_dma.c, 194
dm35425_adc_fifo.c, 202
dm35425_dac.c, 229
dm35425_dac_dma.c, 232
dm35425_dac_fifo.c, 176
DEFAULT_DAC_RATE
dm35425_dac_fifo.c, 177
DEFAULT_MODE
dm35425_adc.c, 181
dm35425_adc_fifo.c, 202
DEFAULT_RANGE
dm35425_adc.c, 181
dm35425_adc_all_dma.c, 186
dm35425_adc_continuous_dma.c, 194
dm35425_adc_fifo.c, 202
dm35425_dac.c, 229
dm35425_dac_dma.c, 232
dm35425_dac_fifo.c, 177
DEFAULT_RATE
dm35425_adc.c, 182
dm35425_adc_all_dma.c, 186
dm35425_adc_continuous_dma.c, 194
dm35425_adc_fifo.c, 203
dm35425_adio_parallel_bus.c, 223
dm35425_dac_dma.c, 233
dm35425_ext_clocking.c, 237
device_lock
dm35425_device_descriptor, 155
DM35425 ADC Library Constants, 7
DM35425_ADC_2_FULL_SAMPLE_DELAY, 11
DM35425_ADC_CLK_BUS_SRC_CHAN_THRESH, 10
DM35425_ADC_CLK_BUS_SRC_DISABLE, 10
DM35425_ADC_CLK_BUS_SRC_PACER_TICK, 11
DM35425_ADC_CLK_BUS_SRC_POST_STOP_BUFF_FULL, 11
DM35425_ADC_CLK_BUS_SRC_PRE_START_BUFF_FULL, 10
DM35425_ADC_CLK_BUS_SRC_SAMPLE_TAKEN, 10
DM35425_ADC_CLK_BUS_SRC_SAMPLING_COMPLETE, 11
DM35425_ADC_CLK_BUS_SRC_START_TRIG, 11
DM35425_ADC_CLK_BUS_SRC_STOP_TRIG, 11
DM35425_Adc_Clock_Events, 10
DM35425_ADC_FULL_SAMPLE_DELAY, 11
DM35425_ADC_GAIN_05, 11
DM35425_ADC_GAIN_1, 11
DM35425_ADC_GAIN_128, 12
DM35425_ADC_GAIN_16, 12
DM35425_ADC_GAIN_2, 11
DM35425_ADC_GAIN_32, 12
DM35425_ADC_GAIN_4, 12
DM35425_ADC_GAIN_64, 12
DM35425_ADC_GAIN_8, 12
DM35425_ADC_HALF_SAMPLE_DELAY, 11
DM35425_ADC_INPUT_DIFFERENTIAL, 12
DM35425_ADC_INPUT_SINGLE_ENDED, 12
DM35425_ADC_NO_DELAY, 11
DM35425_ADC_RNG_BIPOLAR_10V, 12
DM35425_ADC_RNG_BIPOLAR_1_25V, 13
DM35425_ADC_RNG_BIPOLAR_2_5V, 13
DM35425_ADC_RNG_BIPOLAR_5V, 12
DM35425_ADC_RNG_BIPOLAR_625mV, 13
DM35425_ADC_RNG_UNIPOLAR_10V, 13
DM35425_ADC_RNG_UNIPOLAR_1_25V, 13
DM35425_ADC_RNG_UNIPOLAR_2_5V, 13
DM35425_ADC_RNG_UNIPOLAR_5V, 13
DM35425_Channel_Delay, 11
DM35425_Gains, 11
DM35425_Input_Mode, 12
DM35425_Input_Ranges, 12
DM35425 ADC Public Library Functions, 14
DM35425_Adc_Channel_Find_Interrupt, 16
DM35425_Adc_Channel_Get_Filter, 17
DM35425_Adc_Channel_Get_Front_End_Config, 18
DM35425_Adc_Channel_Get_Last_Sample, 18
DM35425_Adc_Channel_Get_Thresholds, 19
DM35425_Adc_Channel_Interrupt_Clear_Status, 20
DM35425_Adc_Channel_Interrupt_Get_Config, 20
DM35425_Adc_Channel_Interrupt_Get_Status, 21
DM35425_Adc_Channel_Interrupt_Set_Config, 21
DM35425_Adc_Channel_Reset, 22
DM35425_Adc_Channel_Set_Filter, 22
DM35425_Adc_Channel_Set_High_Threshold, 23
DM35425_Adc_Channel_Set_Low_Threshold, 24
DM35425_Adc_Channel_Setup, 24
DM35425_Adc_Fifo_Channel_Read, 25
DM35425_Adc_Get_Clock_Source_Global, 26
DM35425_Adc_Get_Clock_Src, 26
DM35425_Adc_Get_Mode_Status, 27
DM35425_Adc_Get_Post_Stop_Samples, 27
DM35425_Adc_Get_Pre_Trigger_Samples, 28
DM35425_Adc_Get_Sample_Count, 28
DM35425_Adc_Get_Start_Trigger, 29
DM35425_Adc_Get_Stop_Trigger, 29
DM35425_Adc_Initialize, 30
DM35425_Adc_Interrupt_Clear_Status, 30
DM35425_Adc_Interrupt_Get_Config, 31
DM35425_Adc_Interrupt_Get_Status, 32
DM35425_Adc_Interrupt_Set_Config, 32
DM35425_Adc_Open, 33
DM35425_Adc_Pause, 33
DM35425_Adc_Reset, 34
DM35425_Adc_Sample_To_Volts, 34
DM35425_Adc_Set_Clk_Divider, 35
DM35425_Adc_Set_Clock_Source_Global, 35

DM35425_Adc_Set_Clock_Src, 36
DM35425_Adc_Set_Post_Stop_Samples, 36
DM35425_Adc_Set_Pre_Trigger_Samples, 37
DM35425_Adc_Set_Sample_Rate, 38
DM35425_Adc_Set_Start_Trigger, 38
DM35425_Adc_Set_Stop_Trigger, 39
DM35425_Adc_Start, 39
DM35425_Adc_Start_Rearm, 40
DM35425_Adc_Uninitialize, 40
DM35425_Adc_Volts_To_Sample, 41
DM35425 ADIO Public Library Enums, 42
DM35425_ADV_INT_DISABLED, 42
DM35425_ADV_INT_EVENT, 42
DM35425_ADV_INT_MATCH, 42
DM35425_Adv_Interrupt_Mode, 42
DM35425 ADIO Public Library Functions, 43
DM35425_Adio_Fifo_Channel_Read, 45
DM35425_Adio_Fifo_Channel_Write, 46
DM35425_Adio_Get_Adv_Int_Capt, 46
DM35425_Adio_Get_Adv_Int_Comp, 47
DM35425_Adio_Get_Adv_Int_Mask, 47
DM35425_Adio_Get_Adv_Int_Mode, 48
DM35425_Adio_Get_Clk_Div_Counter, 48
DM35425_Adio_Get_Clock_Source_Global, 48
DM35425_Adio_Get_Direction, 49
DM35425_Adio_Get_Input_Value, 49
DM35425_Adio_Get_Mode_Status, 50
DM35425_Adio_Get_Output_Value, 51
DM35425_Adio_Get_P_Bus_Enable, 51
DM35425_Adio_Get_P_Bus_Ready_Enable, 52
DM35425_Adio_Get_Sample_Count, 52
DM35425_Adio_Interrupt_Clear_Status, 52
DM35425_Adio_Interrupt_Get_Config, 53
DM35425_Adio_Interrupt_Get_Status, 53
DM35425_Adio_Interrupt_Set_Config, 54
DM35425_Adio_Open, 55
DM35425_Adio_Pause, 55
DM35425_Adio_Reset, 55
DM35425_Adio_Set_Adv_Int_Capt, 56
DM35425_Adio_Set_Adv_Int_Comp, 56
DM35425_Adio_Set_Adv_Int_Mask, 57
DM35425_Adio_Set_Adv_Int_Mode, 57
DM35425_Adio_Set_Clk_Divider, 58
DM35425_Adio_Set_Clock_Source_Global, 58
DM35425_Adio_Set_Clock_Src, 59
DM35425_Adio_Set_Direction, 59
DM35425_Adio_Set_Output_Value, 60
DM35425_Adio_Set_P_Bus_Enable, 60
DM35425_Adio_Set_P_Bus_Ready_Enable, 61
DM35425_Adio_Set_Pacer_Clk_Rate, 61
DM35425_Adio_Set_Post_Stop_Samples, 62
DM35425_Adio_Set_Pre_Trigger_Samples, 63
DM35425_Adio_Set_Start_Trigger, 63
DM35425_Adio_Set_Stop_Trigger, 64
DM35425_Adio_Start, 64
DM35425_Adio_Start_Rearm, 65
DM35425_Adio_Uninitialize, 65
DM35425 Board Access Public Library Functions, 136
DM35425_Dma_Initialize, 136
DM35425_Dma_Read, 137
DM35425_Dma_Write, 138
DM35425_General_InstallISR, 138
DM35425_General_RemoveISR, 139
DM35425_General_SetISRPriority, 139
DM35425_General_WaitForInterrupt, 140
DM35425 Board Access Structures, 67
DM35425_Board_Close, 67
DM35425_Board_Open, 68
DM35425_Dma, 68
DM35425_Modify, 69
DM35425_Read, 69
DM35425_Write, 70
DM35425 Board Library Public Functions, 129
DM35425_Function_Block_Open, 129
DM35425_Function_Block_Open_Module, 130
DM35425_Gbc_Ack_Interrupt, 130
DM35425_Gbc_Board_Reset, 132
DM35425_Gbc_Get_Format, 132
DM35425_Gbc_Get_Fpga_Build, 133
DM35425_Gbc_Get_Pdp_Number, 133
DM35425_Gbc_Get_Revision, 134
DM35425_Gbc_Get_Sys_Clock_Freq, 134
DM35425 Board Macros, 128
CLK_40MHZ, 128
DM35425 Board Types, 147
DM35425 DAC Library Constants, 73
DM35425_DAC_CLK_BUS_SRC_CHAN_MARKER, 75
DM35425_DAC_CLK_BUS_SRC_CONV_COMPL, 75
DM35425_DAC_CLK_BUS_SRC_CONVERSION_SENT, 75
DM35425_DAC_CLK_BUS_SRC_DISABLE, 75
DM35425_DAC_CLK_BUS_SRC_START_TRIG, 75
DM35425_DAC_CLK_BUS_SRC_STOP_TRIG, 75
DM35425_Dac_Clock_Events, 74
DM35425_DAC_RNG_BIPOLAR_10V, 75
DM35425_DAC_RNG_BIPOLAR_5V, 75
DM35425_DAC_RNG_UNIPOLAR_10V, 75
DM35425_DAC_RNG_UNIPOLAR_5V, 75
DM35425_Output_Ranges, 75
DM35425 DAC Library Public Functions, 76
DM35425_Dac_Channel_Clear_Marker_Status, 78
DM35425_Dac_Channel_Get_Marker_Config, 78
DM35425_Dac_Channel_Get_Marker_Status, 79
DM35425_Dac_Channel_Reset, 79
DM35425_Dac_Channel_Set_Marker_Config, 80
DM35425_Dac_Channel_Setup, 80
DM35425_Dac_Conv_To_Volts, 81
DM35425_Dac_Fifo_Channel_Write, 81
DM35425_Dac_Get_Clock_Div, 82
DM35425_Dac_Get_Clock_Src, 82
DM35425_Dac_Get_Conversion_Count, 83
DM35425_Dac_Get_Last_Conversion, 83

DM35425_Dac_Get_Mode_Status, 84
DM35425_Dac_Get_Post_Stop_Conversion_Count, 84
DM35425_Dac_Get_Start_Trigger, 85
DM35425_Dac_Get_Stop_Trigger, 85
DM35425_Dac_Interrupt_Clear_Status, 86
DM35425_Dac_Interrupt_Get_Config, 86
DM35425_Dac_Interrupt_Get_Status, 87
DM35425_Dac_Interrupt_Set_Config, 87
DM35425_Dac_Open, 88
DM35425_Dac_Pause, 88
DM35425_Dac_Reset, 89
DM35425_Dac_Set_Clock_Div, 89
DM35425_Dac_Set_Clock_Source_Global, 90
DM35425_Dac_Set_Clock_Src, 90
DM35425_Dac_Set_Conversion_Rate, 91
DM35425_Dac_Set_Last_Conversion, 91
DM35425_Dac_Set_Post_Stop_Conversion_Count, 92
DM35425_Dac_Set_Start_Trigger, 92
DM35425_Dac_Set_Stop_Trigger, 93
DM35425_Dac_Start, 93
DM35425_Dac_Volts_To_Conv, 94
DM35425 DMA Public Library Constants, 95
DM35425_FIFO_EMPTY, 96
DM35425_FIFO_FULL, 96
DM35425_FIFO_HAS_DATA, 96
DM35425_Fifo_States, 96
DM35425_FIFO_UNKNOWN, 96
DM35425 DMA Public Library Functions, 97
DM35425_Dma_Buffer_Setup, 98
DM35425_Dma_Buffer_Status, 99
DM35425_Dma_Check_Buffer_Used, 99
DM35425_Dma_Check_For_Error, 100
DM35425_Dma_Clear, 101
DM35425_Dma_Clear_Interrupt, 101
DM35425_Dma_Configure_Interrupts, 102
DM35425_Dma_Find_Interrupt, 103
DM35425_Dma_Get_Current_Buffer_Count, 103
DM35425_Dma_Get_Errors, 104
DM35425_Dma_Get_Fifo_Counts, 105
DM35425_Dma_Get_Fifo_State, 105
DM35425_Dma_Get_Interrupt_Configuration, 106
DM35425_Dma_Pause, 107
DM35425_Dma_Reset_Buffer, 107
DM35425_Dma_Setup, 108
DM35425_Dma_Setup_Set_Direction, 109
DM35425_Dma_Setup_Set_Used, 109
DM35425_Dma_Start, 110
DM35425_Dma_Status, 110
DM35425_Dma_Stop, 111
DM35425 Driver Constants, 113
DM35425 Driver Enumerations, 114
dm35425_pci_region_access_dir, 114
DM35425_PCI_REGION_ACCESS_READ, 114
DM35425_PCI_REGION_ACCESS_WRITE, 114
DM35425 Driver Structures, 115
DM35425 Example Programs Constants, 116
AD_MODE_OPTION, 118
ADC_NUM_OPTION, 118
ADC_OPTION, 118
BAUD_OPTION, 119
BIN2TXT_OPTION, 119
BINARY_OPTION, 118
CHANNELS_OPTION, 118
COUNT_OPTION, 119
DAC_NUM_OPTION, 118
DAC_OPTION, 118
DUMP_OPTION, 118
EXTERNAL_OPTION, 119
FILE_OPTION, 118
HELP_OPTION, 118
Help_Options, 118
HOURS_OPTION, 118
LOW_THRESHOLD_OPTION, 118
MASTER_OPTION, 119
MINOR_OPTION, 118
MODE_OPTION, 118
NOSTOP_OPTION, 118
NUM_OPTION, 119
OFILE_OPTION, 119
OUTPUT_ADC_OPTION, 118
OUTPUT_RMS_OPTION, 118
PACKED_OPTION, 119
PATTERN_OPTION, 118
PORT_OPTION, 118
RANGE_OPTION, 118
RATE_OPTION, 118
RECEIVER_OPTION, 118
REF_NUM_OPTION, 118
REFCLK_OPTION, 119
REFILL_FIFO_OPTION, 118
SAMPLES_OPTION, 118
SENDER_OPTION, 118
SIZE_OPTION, 119
SLAVE_OPTION, 119
START_OPTION, 118
STORE_OPTION, 119
SYNC_CONN_OPTION, 119
SYNC_TERM_OPTION, 119
SYNCBUS_OPTION, 118
TERM_OPTION, 119
TEST_OPTION, 118
USER_ID_OPTION, 119
VERBOSE_OPTION, 119
WAVE_OPTION, 118
DM35425 Global Clocking Library Constants, 120
DM35425 Global Clocking Library Public Functions, 121
DM35425_Ext_Clocking_Get_Dir, 121
DM35425_Ext_Clocking_Get_Edge, 122
DM35425_Ext_Clocking_Get_Gate_In, 122
DM35425_Ext_Clocking_Get_In, 123
DM35425_Ext_Clocking_Get_Method, 123
DM35425_Ext_Clocking_Get_Pulse_Width, 124
DM35425_Ext_Clocking_Open, 124
DM35425_Ext_Clocking_Set_Dir, 125

DM35425_Ext_Clocking_Set_Edge, 125
DM35425_Ext_Clocking_Set_Method, 126
DM35425_Ext_Clocking_Set_Pulse_Width, 126
DM35425 Ioctl macros, 135
DM35425 PCI Region Structures, 71
DM35425_DMA_FUNCTIONS, 71
DM35425_DMA_INITIALIZE, 71
DM35425_DMA_READ, 71
DM35425_DMA_WRITE, 71
DM35425_PCI_REGION_ACCESS_16, 72
DM35425_PCI_REGION_ACCESS_32, 72
DM35425_PCI_REGION_ACCESS_8, 72
dm35425_pci_region_access_size, 72
DM35425_PCI_REGION_FB, 72
DM35425_PCI_REGION_GBC, 72
DM35425_PCI_REGION_GBC2, 72
dm35425_pci_region_num, 72
DM35425 Register Offsets, 141
DM35425_OFFSET_DAC_STOP_TRIG, 146
DM35425_OFFSET_FB_ADC_FIFO, 146
DM35425 Utility Library Functions, 149
check_result, 149
DM35425_Generate_Signal_Data, 150
DM35425_Get_Maskable, 151
DM35425_Get_Time_Diff, 151
DM35425_Micro_Sleep, 151
DM35425_SAWTOOTH_WAVE, 149
DM35425_SINE_WAVE, 149
DM35425_SQUARE_WAVE, 149
DM35425_Waveforms, 149
dm35425_adc.c
DEFAULT_CHANNEL, 181
DEFAULT_MODE, 181
DEFAULT_RANGE, 181
DEFAULT_RATE, 182
DM35425_Setup_Dacs, 182
exit_program, 184
interrupt_count, 184
main, 182
program_name, 184
sigint_handler, 183
DM35425_ADC_2_FULL_SAMPLE_DELAY
DM35425 ADC Library Constants, 11
dm35425_adc_all_dma.c
board, 189
buffer_count, 189
buffer_size_bytes, 189
DAT_FILE_NAME_PREFIX, 186
DAT_FILE_NAME_SUFFIX, 186
DEFAULT_RANGE, 186
DEFAULT_RATE, 186
dma_has_error, 189
exit_program, 190
ISR, 186
local_buffer, 190
main, 187
my_adc, 190
next_buffer, 190
output_channel_status, 188
program_name, 191
sigint_handler, 188
DM35425_Adc_Channel_Find_Interrupt
DM35425 ADC Public Library Functions, 16
DM35425_Adc_Channel_Get_Filter
DM35425 ADC Public Library Functions, 17
DM35425_Adc_Channel_Get_Front_End_Config
DM35425 ADC Public Library Functions, 18
DM35425_Adc_Channel_Get_Last_Sample
DM35425 ADC Public Library Functions, 18
DM35425_Adc_Channel_Get_Thresholds
DM35425 ADC Public Library Functions, 19
DM35425_Adc_Channel_Interrupt_Clear_Status
DM35425 ADC Public Library Functions, 20
DM35425_Adc_Channel_Interrupt_Get_Config
DM35425 ADC Public Library Functions, 20
DM35425_Adc_Channel_Interrupt_Get_Status
DM35425 ADC Public Library Functions, 21
DM35425_Adc_Channel_Interrupt_Set_Config
DM35425 ADC Public Library Functions, 21
DM35425_Adc_Channel_Reset
DM35425 ADC Public Library Functions, 22
DM35425_Adc_Channel_Set_Filter
DM35425 ADC Public Library Functions, 22
DM35425_Adc_Channel_Set_High_Threshold
DM35425 ADC Public Library Functions, 23
DM35425_Adc_Channel_Set_Low_Threshold
DM35425 ADC Public Library Functions, 24
DM35425_Adc_Channel_Setup
DM35425 ADC Public Library Functions, 24
DM35425_ADC_CLK_BUS_SRC_CHAN_THRESH
DM35425 ADC Library Constants, 10
DM35425_ADC_CLK_BUS_SRC_DISABLE
DM35425 ADC Library Constants, 10
DM35425_ADC_CLK_BUS_SRC_PACER_TICK
DM35425 ADC Library Constants, 11
DM35425_ADC_CLK_BUS_SRC_POST_STOP_BUFF_FULL
DM35425 ADC Library Constants, 11
DM35425_ADC_CLK_BUS_SRC_PRE_START_BUFF_FULL
DM35425 ADC Library Constants, 10
DM35425_ADC_CLK_BUS_SRC_SAMPLE_TAKEN
DM35425 ADC Library Constants, 10
DM35425_ADC_CLK_BUS_SRC_SAMPLING_COMPLETE
DM35425 ADC Library Constants, 11
DM35425_ADC_CLK_BUS_SRC_START_TRIG
DM35425 ADC Library Constants, 11
DM35425_ADC_CLK_BUS_SRC_STOP_TRIG
DM35425 ADC Library Constants, 11
DM35425_Adc_Clock_Events
DM35425 ADC Library Constants, 10
dm35425_adc_continuous_dma.c
ASCII_FILE_NAME, 193
BIN_FILE_NAME, 193
board, 198
buffer_count, 198
buffer_size_bytes, 198
channel, 199

convert_bin_to_txt, 195
DAC_BUFFER_SIZE_BYTES, 193
DAC_BUFFER_SIZE_SAMPLES, 194
DAC_RATE, 194
DEFAULT_CHANNEL, 194
DEFAULT_RANGE, 194
DEFAULT_RATE, 194
dma_has_error, 199
exit_program, 199
ISR, 195
local_buffer, 199
main, 196
my_adc, 200
next_buffer, 200
output_channel_status, 196
program_name, 200
setup_dacs, 197
sigint_handler, 197
dm35425_adc_fifo.c
DEFAULT_CHANNEL, 202
DEFAULT_MODE, 202
DEFAULT_RANGE, 202
DEFAULT_RATE, 203
dma_interrupt_count, 204
exit_program, 204
interrupt_count, 205
main, 203
my_adc, 205
non_dma_interrupt_count, 205
program_name, 205
sigint_handler, 204
unexpected_interrupt_count, 206
DM35425_Adc_Fifo_Channel_Read
DM35425 ADC Public Library Functions, 25
DM35425_ADC_FULL_SAMPLE_DELAY
DM35425 ADC Library Constants, 11
DM35425_ADC_GAIN_05
DM35425 ADC Library Constants, 11
DM35425_ADC_GAIN_1
DM35425 ADC Library Constants, 11
DM35425_ADC_GAIN_128
DM35425 ADC Library Constants, 12
DM35425_ADC_GAIN_16
DM35425 ADC Library Constants, 12
DM35425_ADC_GAIN_2
DM35425 ADC Library Constants, 11
DM35425_ADC_GAIN_32
DM35425 ADC Library Constants, 12
DM35425_ADC_GAIN_4
DM35425 ADC Library Constants, 12
DM35425_ADC_GAIN_64
DM35425 ADC Library Constants, 12
DM35425_ADC_GAIN_8
DM35425 ADC Library Constants, 12
DM35425_Adc_Get_Clock_Source_Global
DM35425 ADC Public Library Functions, 26
DM35425_Adc_Get_Clock_Src
DM35425 ADC Public Library Functions, 26
DM35425_Adc_Get_Mode_Status
DM35425 ADC Public Library Functions, 27
DM35425_Adc_Get_Post_Stop_Samples
DM35425 ADC Public Library Functions, 27
DM35425_Adc_Get_Pre_Trigger_Samples
DM35425 ADC Public Library Functions, 28
DM35425_Adc_Get_Sample_Count
DM35425 ADC Public Library Functions, 28
DM35425_Adc_Get_Start_Trigger
DM35425 ADC Public Library Functions, 29
DM35425_Adc_Get_Stop_Trigger
DM35425 ADC Public Library Functions, 29
DM35425_ADC_HALF_SAMPLE_DELAY
DM35425 ADC Library Constants, 11
DM35425_Adc_Initialize
DM35425 ADC Public Library Functions, 30
DM35425_ADC_INPUT_DIFFERENTIAL
DM35425 ADC Library Constants, 12
DM35425_ADC_INPUT_SINGLE_ENDED
DM35425 ADC Library Constants, 12
DM35425_Adc_Interrupt_Clear_Status
DM35425 ADC Public Library Functions, 30
DM35425_Adc_Interrupt_Get_Config
DM35425 ADC Public Library Functions, 31
DM35425_Adc_Interrupt_Get_Status
DM35425 ADC Public Library Functions, 32
DM35425_Adc_Interrupt_Set_Config
DM35425 ADC Public Library Functions, 32
DM35425_ADC_NO_DELAY
DM35425 ADC Library Constants, 11
DM35425_Adc_Open
DM35425 ADC Public Library Functions, 33
DM35425_Adc_Pause
DM35425 ADC Public Library Functions, 33
DM35425_Adc_Reset
DM35425 ADC Public Library Functions, 34
DM35425_ADC_RNG_BIPOLAR_10V
DM35425 ADC Library Constants, 12
DM35425_ADC_RNG_BIPOLAR_1_25V
DM35425 ADC Library Constants, 13
DM35425_ADC_RNG_BIPOLAR_2_5V
DM35425 ADC Library Constants, 13
DM35425_ADC_RNG_BIPOLAR_5V
DM35425 ADC Library Constants, 12
DM35425_ADC_RNG_BIPOLAR_625mV
DM35425 ADC Library Constants, 13
DM35425_ADC_RNG_UNIPOLAR_10V
DM35425 ADC Library Constants, 13
DM35425_ADC_RNG_UNIPOLAR_1_25V
DM35425 ADC Library Constants, 13
DM35425_ADC_RNG_UNIPOLAR_2_5V
DM35425 ADC Library Constants, 13
DM35425_ADC_RNG_UNIPOLAR_5V
DM35425 ADC Library Constants, 13
DM35425_Adc_Sample_To_Volts
DM35425 ADC Public Library Functions, 34
DM35425_Adc_Set_Clk_Divider
DM35425 ADC Public Library Functions, 35

DM35425_Adc_Set_Clock_Source_Global
  DM35425 ADC Public Library Functions, 35
DM35425_Adc_Set_Clock_Src
  DM35425 ADC Public Library Functions, 36
DM35425_Adc_Set_Post_Stop_Samples
  DM35425 ADC Public Library Functions, 36
DM35425_Adc_Set_Pre_Trigger_Samples
  DM35425 ADC Public Library Functions, 37
DM35425_Adc_Set_Sample_Rate
  DM35425 ADC Public Library Functions, 38
DM35425_Adc_Set_Start_Trigger
  DM35425 ADC Public Library Functions, 38
DM35425_Adc_Set_Stop_Trigger
  DM35425 ADC Public Library Functions, 39
DM35425_Adc_Start
  DM35425 ADC Public Library Functions, 39
DM35425_Adc_Start_Rearm
  DM35425 ADC Public Library Functions, 40
DM35425_Adc_Uninitialize
  DM35425 ADC Public Library Functions, 40
DM35425_Adc_Volts_To_Sample
  DM35425 ADC Public Library Functions, 41
dm35425_adio.c
  board, 208
  DM35425_ADIO_DIRECTION, 207
  main, 208
  my_adio, 208
  program_name, 209
dm35425_adio_adv_int.c
  board, 213
  DM35425_ADIO_DIRECTION, 211
  DM35425_ADIO_MATCH1, 211
  DM35425_ADIO_MATCH2, 211
  exit_program, 213
  ISR, 211
  main, 212
  my_adio, 213
  program_name, 213
  sigint_handler, 212
  total_interrupt_count, 213
DM35425_ADIO_DIR_OUTPUT
  dm35425_ext_clocking.c, 237
DM35425_ADIO_DIRECTION
  dm35425_adio.c, 207
  dm35425_adio_adv_int.c, 211
DM35425_ADIO_DIRECTION1
  dm35425_adio_dma.c, 216
DM35425_ADIO_DIRECTION2
  dm35425_adio_dma.c, 216
dm35425_adio_dma.c
  ADIO_RATE, 215
  board, 219
  buffer_copied, 219
  BUFFER_SIZE_BYTES, 216
  BUFFER_SIZE_SAMPLES, 216
  DM35425_ADIO_DIRECTION1, 216
  DM35425_ADIO_DIRECTION2, 216
  dma_has_error, 220
  exit_program, 220
  ISR, 216
  main, 217
  my_adio, 220
  output_channel_status, 218
  output_dma_buffer_status, 218
  program_name, 220
  sigint_handler, 219
DM35425_Adio_Fifo_Channel_Read
  DM35425 ADIO Public Library Functions, 45
DM35425_Adio_Fifo_Channel_Write
  DM35425 ADIO Public Library Functions, 46
DM35425_Adio_Get_Adv_Int_Capt
  DM35425 ADIO Public Library Functions, 46
DM35425_Adio_Get_Adv_Int_Comp
  DM35425 ADIO Public Library Functions, 47
DM35425_Adio_Get_Adv_Int_Mask
  DM35425 ADIO Public Library Functions, 47
DM35425_Adio_Get_Adv_Int_Mode
  DM35425 ADIO Public Library Functions, 48
DM35425_Adio_Get_Clk_Div_Counter
  DM35425 ADIO Public Library Functions, 48
DM35425_Adio_Get_Clock_Source_Global
  DM35425 ADIO Public Library Functions, 48
DM35425_Adio_Get_Direction
  DM35425 ADIO Public Library Functions, 49
DM35425_Adio_Get_Input_Value
  DM35425 ADIO Public Library Functions, 49
DM35425_Adio_Get_Mode_Status
  DM35425 ADIO Public Library Functions, 50
DM35425_Adio_Get_Output_Value
  DM35425 ADIO Public Library Functions, 51
DM35425_Adio_Get_P_Bus_Enable
  DM35425 ADIO Public Library Functions, 51
DM35425_Adio_Get_P_Bus_Ready_Enable
  DM35425 ADIO Public Library Functions, 52
DM35425_Adio_Get_Sample_Count
  DM35425 ADIO Public Library Functions, 52
DM35425_ADIO_IN_DIRECTION
  dm35425_adio_parallel_bus.c, 223
DM35425_Adio_Interrupt_Clear_Status
  DM35425 ADIO Public Library Functions, 52
DM35425_Adio_Interrupt_Get_Config
  DM35425 ADIO Public Library Functions, 53
DM35425_Adio_Interrupt_Get_Status
  DM35425 ADIO Public Library Functions, 53
DM35425_Adio_Interrupt_Set_Config
  DM35425 ADIO Public Library Functions, 54
DM35425_ADIO_MATCH1
  dm35425_adio_adv_int.c, 211
DM35425_ADIO_MATCH2
  dm35425_adio_adv_int.c, 211
DM35425_Adio_Open
  DM35425 ADIO Public Library Functions, 55
DM35425_ADIO_OUT_DIRECTION
  dm35425_adio_parallel_bus.c, 223
dm35425_adio_parallel_bus.c
  board, 226

BUFFER_SIZE_BYTES, 222
BUFFER_SIZE_SAMPLES, 223
DEFAULT_RATE, 223
DM35425_ADIO_IN_DIRECTION, 223
DM35425_ADIO_OUT_DIRECTION, 223
exit_program, 226
interrupt_count, 226
is_receiver, 226
is_sender, 227
ISR, 223
main, 224
my_adio, 227
output_channel_status, 225
program_name, 227
sigint_handler, 225
DM35425_Adio_Pause
DM35425 ADIO Public Library Functions, 55
DM35425_Adio_Reset
DM35425 ADIO Public Library Functions, 55
DM35425_Adio_Set_Adv_Int_Capt
DM35425 ADIO Public Library Functions, 56
DM35425_Adio_Set_Adv_Int_Comp
DM35425 ADIO Public Library Functions, 56
DM35425_Adio_Set_Adv_Int_Mask
DM35425 ADIO Public Library Functions, 57
DM35425_Adio_Set_Adv_Int_Mode
DM35425 ADIO Public Library Functions, 57
DM35425_Adio_Set_Clk_Divider
DM35425 ADIO Public Library Functions, 58
DM35425_Adio_Set_Clock_Source_Global
DM35425 ADIO Public Library Functions, 58
DM35425_Adio_Set_Clock_Src
DM35425 ADIO Public Library Functions, 59
DM35425_Adio_Set_Direction
DM35425 ADIO Public Library Functions, 59
DM35425_Adio_Set_Output_Value
DM35425 ADIO Public Library Functions, 60
DM35425_Adio_Set_P_Bus_Enable
DM35425 ADIO Public Library Functions, 60
DM35425_Adio_Set_P_Bus_Ready_Enable
DM35425 ADIO Public Library Functions, 61
DM35425_Adio_Set_Pacer_Clk_Rate
DM35425 ADIO Public Library Functions, 61
DM35425_Adio_Set_Post_Stop_Samples
DM35425 ADIO Public Library Functions, 62
DM35425_Adio_Set_Pre_Trigger_Samples
DM35425 ADIO Public Library Functions, 63
DM35425_Adio_Set_Start_Trigger
DM35425 ADIO Public Library Functions, 63
DM35425_Adio_Set_Stop_Trigger
DM35425 ADIO Public Library Functions, 64
DM35425_Adio_Start
DM35425 ADIO Public Library Functions, 64
DM35425_Adio_Start_Rearm
DM35425 ADIO Public Library Functions, 65
DM35425_Adio_Uninitialize
DM35425 ADIO Public Library Functions, 65
DM35425_ADV_INT_DISABLED
DM35425 ADIO Public Library Enums, 42
DM35425_ADV_INT_EVENT
DM35425 ADIO Public Library Enums, 42
DM35425_ADV_INT_MATCH
DM35425 ADIO Public Library Enums, 42
DM35425_Adv_Interrupt_Mode
DM35425 ADIO Public Library Enums, 42
dm35425_board_access.h
DM35425LIB_API, 255
DM35425_Board_Close
DM35425 Board Access Structures, 67
DM35425_Board_Descriptor, 153
file_descriptor, 153
isr, 153
pid, 154
DM35425_Board_Open
DM35425 Board Access Structures, 68
DM35425_Channel_Delay
DM35425 ADC Library Constants, 11
DM35425_CLK_BUS2
dm35425_types.h, 278
DM35425_CLK_BUS3
dm35425_types.h, 278
DM35425_CLK_BUS4
dm35425_types.h, 278
DM35425_CLK_BUS5
dm35425_types.h, 278
DM35425_CLK_BUS6
dm35425_types.h, 278
DM35425_CLK_BUS7
dm35425_types.h, 278
DM35425_CLK_SRC_BUS2
dm35425_types.h, 279
DM35425_CLK_SRC_BUS2_INV
dm35425_types.h, 279
DM35425_CLK_SRC_BUS3
dm35425_types.h, 279
DM35425_CLK_SRC_BUS3_INV
dm35425_types.h, 279
DM35425_CLK_SRC_BUS4
dm35425_types.h, 279
DM35425_CLK_SRC_BUS4_INV
dm35425_types.h, 279
DM35425_CLK_SRC_BUS5
dm35425_types.h, 279
DM35425_CLK_SRC_BUS5_INV
dm35425_types.h, 279
DM35425_CLK_SRC_BUS6
dm35425_types.h, 279
DM35425_CLK_SRC_BUS6_INV
dm35425_types.h, 279
DM35425_CLK_SRC_BUS7
dm35425_types.h, 279
DM35425_CLK_SRC_BUS7_INV
dm35425_types.h, 279
DM35425_CLK_SRC_CHAN_THRESH
dm35425_types.h, 279
DM35425_CLK_SRC_CHAN_THRESH_INV

dm35425_types.h, 279
DM35425_CLK_SRC_IMMEDIATE
dm35425_types.h, 279
DM35425_CLK_SRC_NEVER
dm35425_types.h, 279
DM35425_Clock_Buses
dm35425_types.h, 278
DM35425_Clock_Sources
dm35425_types.h, 278
dm35425_dac.c
DEFAULT_CHANNEL, 229
DEFAULT_RANGE, 229
main, 229
program_name, 230
DM35425_Dac_Channel_Clear_Marker_Status
DM35425 DAC Library Public Functions, 78
DM35425_Dac_Channel_Get_Marker_Config
DM35425 DAC Library Public Functions, 78
DM35425_Dac_Channel_Get_Marker_Status
DM35425 DAC Library Public Functions, 79
DM35425_Dac_Channel_Reset
DM35425 DAC Library Public Functions, 79
DM35425_Dac_Channel_Set_Marker_Config
DM35425 DAC Library Public Functions, 80
DM35425_Dac_Channel_Setup
DM35425 DAC Library Public Functions, 80
DM35425_DAC_CLK_BUS_SRC_CHAN_MARKER
DM35425 DAC Library Constants, 75
DM35425_DAC_CLK_BUS_SRC_CONV_COMPL
DM35425 DAC Library Constants, 75
DM35425_DAC_CLK_BUS_SRC_CONVERSION_SENT
DM35425 DAC Library Constants, 75
DM35425_DAC_CLK_BUS_SRC_DISABLE
DM35425 DAC Library Constants, 75
DM35425_DAC_CLK_BUS_SRC_START_TRIG
DM35425 DAC Library Constants, 75
DM35425_DAC_CLK_BUS_SRC_STOP_TRIG
DM35425 DAC Library Constants, 75
DM35425_Dac_Clock_Events
DM35425 DAC Library Constants, 74
DM35425_Dac_Conv_To_Volts
DM35425 DAC Library Public Functions, 81
dm35425_dac_dma.c
BUFFER_SIZE_BYTES, 232
BUFFER_SIZE_SAMPLES, 232
DEFAULT_CHANNEL, 232
DEFAULT_RANGE, 232
DEFAULT_RATE, 233
exit_program, 234
main, 233
NUM_BUFFERS_TO_USE, 233
program_name, 235
sigint_handler, 234
dm35425_dac_fifo.c
DEFAULT_CHANNEL, 176
DEFAULT_DAC_RATE, 177
DEFAULT_RANGE, 177
exit_program, 179
main, 177
print_fifo_status, 178
program_name, 179
sigint_handler, 178
DM35425_Dac_Fifo_Channel_Write
DM35425 DAC Library Public Functions, 81
DM35425_Dac_Get_Clock_Div
DM35425 DAC Library Public Functions, 82
DM35425_Dac_Get_Clock_Src
DM35425 DAC Library Public Functions, 82
DM35425_Dac_Get_Conversion_Count
DM35425 DAC Library Public Functions, 83
DM35425_Dac_Get_Last_Conversion
DM35425 DAC Library Public Functions, 83
DM35425_Dac_Get_Mode_Status
DM35425 DAC Library Public Functions, 84
DM35425_Dac_Get_Post_Stop_Conversion_Count
DM35425 DAC Library Public Functions, 84
DM35425_Dac_Get_Start_Trigger
DM35425 DAC Library Public Functions, 85
DM35425_Dac_Get_Stop_Trigger
DM35425 DAC Library Public Functions, 85
DM35425_Dac_Interrupt_Clear_Status
DM35425 DAC Library Public Functions, 86
DM35425_Dac_Interrupt_Get_Config
DM35425 DAC Library Public Functions, 86
DM35425_Dac_Interrupt_Get_Status
DM35425 DAC Library Public Functions, 87
DM35425_Dac_Interrupt_Set_Config
DM35425 DAC Library Public Functions, 87
DM35425_Dac_Open
DM35425 DAC Library Public Functions, 88
DM35425_Dac_Pause
DM35425 DAC Library Public Functions, 88
DM35425_Dac_Reset
DM35425 DAC Library Public Functions, 89
DM35425_DAC_RNG_BIPOLAR_10V
DM35425 DAC Library Constants, 75
DM35425_DAC_RNG_BIPOLAR_5V
DM35425 DAC Library Constants, 75
DM35425_DAC_RNG_UNIPOLAR_10V
DM35425 DAC Library Constants, 75
DM35425_DAC_RNG_UNIPOLAR_5V
DM35425 DAC Library Constants, 75
DM35425_Dac_Set_Clock_Div
DM35425 DAC Library Public Functions, 89
DM35425_Dac_Set_Clock_Source_Global
DM35425 DAC Library Public Functions, 90
DM35425_Dac_Set_Clock_Src
DM35425 DAC Library Public Functions, 90
DM35425_Dac_Set_Conversion_Rate
DM35425 DAC Library Public Functions, 91
DM35425_Dac_Set_Last_Conversion
DM35425 DAC Library Public Functions, 91
DM35425_Dac_Set_Post_Stop_Conversion_Count
DM35425 DAC Library Public Functions, 92
DM35425_Dac_Set_Start_Trigger
DM35425 DAC Library Public Functions, 92

DM35425_Dac_Set_Stop_Trigger
DM35425 DAC Library Public Functions, 93
DM35425_Dac_Start
DM35425 DAC Library Public Functions, 93
DM35425_Dac_Volts_To_Conv
DM35425 DAC Library Public Functions, 94
dm35425_device_descriptor, 154
device_lock, 155
dma_descr_list, 155
dma_wait_queue, 155
int_queue_count, 155
int_queue_in_marker, 155
int_queue_missed, 155
int_queue_out_marker, 156
int_wait_queue, 156
interrupt_fb, 156
irq_number, 156
name, 156
pci, 157
reference_count, 157
remove_isr_flag, 157
DM35425_Dma
DM35425 Board Access Structures, 68
DM35425_Dma_Buffer_Setup
DM35425 DMA Public Library Functions, 98
DM35425_Dma_Buffer_Status
DM35425 DMA Public Library Functions, 99
DM35425_Dma_Check_Buffer_Used
DM35425 DMA Public Library Functions, 99
DM35425_Dma_Check_For_Error
DM35425 DMA Public Library Functions, 100
DM35425_Dma_Clear
DM35425 DMA Public Library Functions, 101
DM35425_Dma_Clear_Interrupt
DM35425 DMA Public Library Functions, 101
DM35425_Dma_Configure_Interrupts
DM35425 DMA Public Library Functions, 102
DM35425_DMA_Descriptor, 157
buffer_start_offset, 158
control_offset, 158
num_buffers, 158
dm35425_dma_descriptor, 159
buffer, 159
buffer_size, 159
bus_addr, 159
channel, 160
fb_num, 160
list, 160
virt_addr, 160
DM35425_Dma_Find_Interrupt
DM35425 DMA Public Library Functions, 103
DM35425_DMA_FUNCTIONS
DM35425 PCI Region Structures, 71
DM35425_Dma_Get_Current_Buffer_Count
DM35425 DMA Public Library Functions, 103
DM35425_Dma_Get_Errors
DM35425 DMA Public Library Functions, 104
DM35425_Dma_Get_Fifo_Counts
DM35425 DMA Public Library Functions, 105
DM35425_Dma_Get_Fifo_State
DM35425 DMA Public Library Functions, 105
DM35425_Dma_Get_Interrupt_Configuration
DM35425 DMA Public Library Functions, 106
DM35425_DMA_INITIALIZE
DM35425 PCI Region Structures, 71
DM35425_Dma_Initialize
DM35425 Board Access Public Library Functions, 136
DM35425_Dma_Pause
DM35425 DMA Public Library Functions, 107
DM35425_DMA_READ
DM35425 PCI Region Structures, 71
DM35425_Dma_Read
DM35425 Board Access Public Library Functions, 137
DM35425_Dma_Reset_Buffer
DM35425 DMA Public Library Functions, 107
DM35425_Dma_Setup
DM35425 DMA Public Library Functions, 108
DM35425_Dma_Setup_Set_Direction
DM35425 DMA Public Library Functions, 109
DM35425_Dma_Setup_Set_Used
DM35425 DMA Public Library Functions, 109
DM35425_Dma_Start
DM35425 DMA Public Library Functions, 110
DM35425_Dma_Status
DM35425 DMA Public Library Functions, 110
DM35425_Dma_Stop
DM35425 DMA Public Library Functions, 111
DM35425_DMA_WRITE
DM35425 PCI Region Structures, 71
DM35425_Dma_Write
DM35425 Board Access Public Library Functions, 138
DM35425_EXT_CLK_DIR_INPUT
dm35425_ext_clocking.c, 238
DM35425_EXT_CLK_EDGE_RISING
dm35425_ext_clocking.c, 238
dm35425_ext_clocking.c
BUFFER_SIZE_BYTES, 237
BUFFER_SIZE_SAMPLES, 237
DEFAULT_RATE, 237
DM35425_ADIO_DIR_OUTPUT, 237
DM35425_EXT_CLK_DIR_INPUT, 238
DM35425_EXT_CLK_EDGE_RISING, 238
exit_program, 240
main, 238
NUM_BUFFERS_TO_USE, 238
program_name, 240
sigint_handler, 239
DM35425_Ext_Clocking_Get_Dir
DM35425 Global Clocking Library Public Functions, 121
DM35425_Ext_Clocking_Get_Edge
DM35425 Global Clocking Library Public Functions, 122

DM35425_Ext_Clocking_Get_Gate_In
DM35425 Global Clocking Library Public Functions, 122
DM35425_Ext_Clocking_Get_In
DM35425 Global Clocking Library Public Functions, 123
DM35425_Ext_Clocking_Get_Method
DM35425 Global Clocking Library Public Functions, 123
DM35425_Ext_Clocking_Get_Pulse_Width
DM35425 Global Clocking Library Public Functions, 124
DM35425_Ext_Clocking_Open
DM35425 Global Clocking Library Public Functions, 124
DM35425_Ext_Clocking_Set_Dir
DM35425 Global Clocking Library Public Functions, 125
DM35425_Ext_Clocking_Set_Edge
DM35425 Global Clocking Library Public Functions, 125
DM35425_Ext_Clocking_Set_Method
DM35425 Global Clocking Library Public Functions, 126
DM35425_Ext_Clocking_Set_Pulse_Width
DM35425 Global Clocking Library Public Functions, 126
DM35425_FIFO_EMPTY
DM35425 DMA Public Library Constants, 96
DM35425_FIFO_FULL
DM35425 DMA Public Library Constants, 96
DM35425_FIFO_HAS_DATA
DM35425 DMA Public Library Constants, 96
DM35425_Fifo_States
DM35425 DMA Public Library Constants, 96
DM35425_FIFO_UNKNOWN
DM35425 DMA Public Library Constants, 96
DM35425_Function_Block, 161
control_offset, 161
dma_channel, 161
dma_offset, 161
fb_num, 162
fb_offset, 162
num_dma_buffers, 162
num_dma_channels, 162
ordinal_fb_type_num, 162
sub_type, 163
type, 163
type_revision, 163
DM35425_Function_Block_Open
DM35425 Board Library Public Functions, 129
DM35425_Function_Block_Open_Module
DM35425 Board Library Public Functions, 130
DM35425_Gains
DM35425 ADC Library Constants, 11
DM35425_Gbc_Ack_Interrupt
DM35425 Board Library Public Functions, 130
DM35425_Gbc_Board_Reset
DM35425 Board Library Public Functions, 132
DM35425_Gbc_Get_Format
DM35425 Board Library Public Functions, 132
DM35425_Gbc_Get_Fpga_Build
DM35425 Board Library Public Functions, 133
DM35425_Gbc_Get_Pdp_Number
DM35425 Board Library Public Functions, 133
DM35425_Gbc_Get_Revision
DM35425 Board Library Public Functions, 134
DM35425_Gbc_Get_Sys_Clock_Freq
DM35425 Board Library Public Functions, 134
DM35425_General_InstallISR
DM35425 Board Access Public Library Functions, 138
DM35425_General_RemoveISR
DM35425 Board Access Public Library Functions, 139
DM35425_General_SetISRPriority
DM35425 Board Access Public Library Functions, 139
DM35425_General_WaitForInterrupt
DM35425 Board Access Public Library Functions, 140
DM35425_Generate_Signal_Data
DM35425 Utility Library Functions, 150
DM35425_Get_Maskable
DM35425 Utility Library Functions, 151
DM35425_Get_Time_Diff
DM35425 Utility Library Functions, 151
DM35425_Input_Mode
DM35425 ADC Library Constants, 12
DM35425_Input_Ranges
DM35425 ADC Library Constants, 12
dm35425_ioctl_argument, 163
dma, 164
interrupt, 164
modify, 164
readwrite, 164
dm35425_ioctl_dma, 165
buffer, 165
buffer_ptr, 165
buffer_size, 166
channel, 166
fb_num, 166
function, 166
num_buffers, 166
pci, 167
dm35425_ioctl_interrupt_info_request, 167
error_occurred, 167
interrupt_fb, 168
interrupts_remaining, 168
valid_interrupt, 168
dm35425_ioctl_region_modify, 169
access, 169
mask, 169
mask16, 169
mask32, 170
mask8, 170

dm35425_ioctl_region_readwrite, 170
  access, 171
dm35425_list_fb.c
  main, 241
  program_name, 242
DM35425_Micro_Sleep
  DM35425 Utility Library Functions, 151
DM35425_Modify
  DM35425 Board Access Structures, 69
DM35425_OFFSET_DAC_STOP_TRIG
  DM35425 Register Offsets, 146
DM35425_OFFSET_FB_ADC_FIFO
  DM35425 Register Offsets, 146
DM35425_Output_Ranges
  DM35425 DAC Library Constants, 75
dm35425_pci_access_request, 171
  data, 171
  data16, 172
  data32, 172
  data8, 172
  offset, 172
  region, 172
  size, 173
dm35425_pci_region, 173
  allocated, 173
  io_addr, 174
  length, 174
  phys_addr, 174
  virt_addr, 174
DM35425_PCI_REGION_ACCESS_16
  DM35425 PCI Region Structures, 72
DM35425_PCI_REGION_ACCESS_32
  DM35425 PCI Region Structures, 72
DM35425_PCI_REGION_ACCESS_8
  DM35425 PCI Region Structures, 72
dm35425_pci_region_access_dir
  DM35425 Driver Enumerations, 114
DM35425_PCI_REGION_ACCESS_READ
  DM35425 Driver Enumerations, 114
dm35425_pci_region_access_size
  DM35425 PCI Region Structures, 72
DM35425_PCI_REGION_ACCESS_WRITE
  DM35425 Driver Enumerations, 114
DM35425_PCI_REGION_FB
  DM35425 PCI Region Structures, 72
DM35425_PCI_REGION_GBC
  DM35425 PCI Region Structures, 72
DM35425_PCI_REGION_GBC2
  DM35425 PCI Region Structures, 72
dm35425_pci_region_num
  DM35425 PCI Region Structures, 72
DM35425_Read
  DM35425 Board Access Structures, 69
DM35425_SAWTOOTH_WAVE
  DM35425 Utility Library Functions, 149
DM35425_Setup_Dacs
  dm35425_adc.c, 182
DM35425_SINE_WAVE
  DM35425 Utility Library Functions, 149
DM35425_SQUARE_WAVE
  DM35425 Utility Library Functions, 149
dm35425_types.h
  DM35425_CLK_BUS2, 278
  DM35425_CLK_BUS3, 278
  DM35425_CLK_BUS4, 278
  DM35425_CLK_BUS5, 278
  DM35425_CLK_BUS6, 278
  DM35425_CLK_BUS7, 278
  DM35425_CLK_SRC_BUS2, 279
  DM35425_CLK_SRC_BUS2_INV, 279
  DM35425_CLK_SRC_BUS3, 279
  DM35425_CLK_SRC_BUS3_INV, 279
  DM35425_CLK_SRC_BUS4, 279
  DM35425_CLK_SRC_BUS4_INV, 279
  DM35425_CLK_SRC_BUS5, 279
  DM35425_CLK_SRC_BUS5_INV, 279
  DM35425_CLK_SRC_BUS6, 279
  DM35425_CLK_SRC_BUS6_INV, 279
  DM35425_CLK_SRC_BUS7, 279
  DM35425_CLK_SRC_BUS7_INV, 279
  DM35425_CLK_SRC_CHAN_THRESH, 279
  DM35425_CLK_SRC_CHAN_THRESH_INV, 279
  DM35425_CLK_SRC_IMMEDIATE, 279
  DM35425_CLK_SRC_NEVER, 279
  DM35425_Clock_Buses, 278
  DM35425_Clock_Sources, 278
DM35425_Waveforms
  DM35425 Utility Library Functions, 149
DM35425_Write
  DM35425 Board Access Structures, 70
DM35425LIB_API
  dm35425_board_access.h, 255
dma
  dm35425_ioctl_argument, 164
dma_channel
  DM35425_Function_Block, 161
dma_descr_list
  dm35425_device_descriptor, 155
dma_has_error
  dm35425_adc_all_dma.c, 189
  dm35425_adc_continuous_dma.c, 199
  dm35425_adio_dma.c, 220
dma_interrupt_count
  dm35425_adc_fifo.c, 204
dma_offset
  DM35425_Function_Block, 161
dma_wait_queue
  dm35425_device_descriptor, 155
DUMP_OPTION
  DM35425 Example Programs Constants, 118

error_occurred
  dm35425_ioctl_interrupt_info_request, 167
examples/_non_public/dm35425_dac_fifo.c, 175
examples/dm35425_adc.c, 180
examples/dm35425_adc_all_dma.c, 184
examples/dm35425_adc_continuous_dma.c, 191

examples/dm35425_adc_fifo.c, 201
examples/dm35425_adio.c, 206
examples/dm35425_adio_adv_int.c, 209
examples/dm35425_adio_dma.c, 214
examples/dm35425_adio_parallel_bus.c, 221
examples/dm35425_dac.c, 228
examples/dm35425_dac_dma.c, 230
examples/dm35425_ext_clocking.c, 235
examples/dm35425_list_fb.c, 240
exit_program
    dm35425_adc.c, 184
    dm35425_adc_all_dma.c, 190
    dm35425_adc_continuous_dma.c, 199
    dm35425_adc_fifo.c, 204
    dm35425_adio_adv_int.c, 213
    dm35425_adio_dma.c, 220
    dm35425_adio_parallel_bus.c, 226
    dm35425_dac_dma.c, 234
    dm35425_dac_fifo.c, 179
    dm35425_ext_clocking.c, 240
EXTERNAL_OPTION
    DM35425 Example Programs Constants, 119

fb_num
    dm35425_dma_descriptor, 160
    DM35425_Function_Block, 162
    dm35425_ioctl_dma, 166
fb_offset
    DM35425_Function_Block, 162
file_descriptor
    DM35425_Board_Descriptor, 153
FILE_OPTION
    DM35425 Example Programs Constants, 118
function
    dm35425_ioctl_dma, 166

HELP_OPTION
    DM35425 Example Programs Constants, 118
Help_Options
    DM35425 Example Programs Constants, 118
HOURS_OPTION
    DM35425 Example Programs Constants, 118

include/dm35425.h, 242
include/dm35425_adc_library.h, 243
include/dm35425_adio_library.h, 249
include/dm35425_board_access.h, 254
include/dm35425_board_access_structs.h, 255
include/dm35425_dac_library.h, 256
include/dm35425_dma_library.h, 260
include/dm35425_driver.h, 263
include/dm35425_examples.h, 264
include/dm35425_ext_clocking_library.h, 266
include/dm35425_gbc_library.h, 268
include/dm35425_ioctl.h, 269
include/dm35425_os.h, 269
include/dm35425_registers.h, 270
include/dm35425_types.h, 276
include/dm35425_util_library.h, 279
int_queue_count
    dm35425_device_descriptor, 155
int_queue_in_marker
    dm35425_device_descriptor, 155
int_queue_missed
    dm35425_device_descriptor, 155
int_queue_out_marker
    dm35425_device_descriptor, 156
int_wait_queue
    dm35425_device_descriptor, 156
interrupt
    dm35425_ioctl_argument, 164
interrupt_count
    dm35425_adc.c, 184
    dm35425_adc_fifo.c, 205
    dm35425_adio_parallel_bus.c, 226
interrupt_fb
    dm35425_device_descriptor, 156
    dm35425_ioctl_interrupt_info_request, 168
interrupts_remaining
    dm35425_ioctl_interrupt_info_request, 168
io_addr
    dm35425_pci_region, 174
irq_number
    dm35425_device_descriptor, 156
is_receiver
    dm35425_adio_parallel_bus.c, 226
is_sender
    dm35425_adio_parallel_bus.c, 227
ISR
    dm35425_adc_all_dma.c, 186
    dm35425_adc_continuous_dma.c, 195
    dm35425_adio_adv_int.c, 211
    dm35425_adio_dma.c, 216
    dm35425_adio_parallel_bus.c, 223
isr
    DM35425_Board_Descriptor, 153

length
    dm35425_pci_region, 174
list
    dm35425_dma_descriptor, 160
local_buffer
    dm35425_adc_all_dma.c, 190
    dm35425_adc_continuous_dma.c, 199
LOW_THRESHOLD_OPTION
    DM35425 Example Programs Constants, 118

main
    dm35425_adc.c, 182
    dm35425_adc_all_dma.c, 187
    dm35425_adc_continuous_dma.c, 196
    dm35425_adc_fifo.c, 203
    dm35425_adio.c, 208
    dm35425_adio_adv_int.c, 212
    dm35425_adio_dma.c, 217
    dm35425_adio_parallel_bus.c, 224
    dm35425_dac.c, 229
    dm35425_dac_dma.c, 233

dm35425_dac_fifo.c, 177
dm35425_ext_clocking.c, 238
dm35425_list_fb.c, 241
mask
dm35425_ioctl_region_modify, 169
mask16
dm35425_ioctl_region_modify, 169
mask32
dm35425_ioctl_region_modify, 170
mask8
dm35425_ioctl_region_modify, 170
MASTER_OPTION
DM35425 Example Programs Constants, 119
MINOR_OPTION
DM35425 Example Programs Constants, 118
MODE_OPTION
DM35425 Example Programs Constants, 118
modify
dm35425_ioctl_argument, 164
my_adc
dm35425_adc_all_dma.c, 190
dm35425_adc_continuous_dma.c, 200
dm35425_adc_fifo.c, 205
my_adio
dm35425_adio.c, 208
dm35425_adio_adv_int.c, 213
dm35425_adio_dma.c, 220
dm35425_adio_parallel_bus.c, 227

name
dm35425_device_descriptor, 156
next_buffer
dm35425_adc_all_dma.c, 190
dm35425_adc_continuous_dma.c, 200
non_dma_interrupt_count
dm35425_adc_fifo.c, 205
NOSTOP_OPTION
DM35425 Example Programs Constants, 118
num_buffers
DM35425_DMA_Descriptor, 158
dm35425_ioctl_dma, 166
NUM_BUFFERS_TO_USE
dm35425_dac_dma.c, 233
dm35425_ext_clocking.c, 238
num_dma_buffers
DM35425_Function_Block, 162
num_dma_channels
DM35425_Function_Block, 162
NUM_OPTION
DM35425 Example Programs Constants, 119

offset
dm35425_pci_access_request, 172
OFILE_OPTION
DM35425 Example Programs Constants, 119
ordinal_fb_type_num
DM35425_Function_Block, 162
OUTPUT_ADC_OPTION
DM35425 Example Programs Constants, 118
output_channel_status
dm35425_adc_all_dma.c, 188
dm35425_adc_continuous_dma.c, 196
dm35425_adio_dma.c, 218
dm35425_adio_parallel_bus.c, 225
output_dma_buffer_status
dm35425_adio_dma.c, 218
OUTPUT_RMS_OPTION
DM35425 Example Programs Constants, 118

PACKED_OPTION
DM35425 Example Programs Constants, 119
PATTERN_OPTION
DM35425 Example Programs Constants, 118
pci
dm35425_device_descriptor, 157
dm35425_ioctl_dma, 167
phys_addr
dm35425_pci_region, 174
pid
DM35425_Board_Descriptor, 154
PORT_OPTION
DM35425 Example Programs Constants, 118
print_fifo_status
dm35425_dac_fifo.c, 178
program_name
dm35425_adc.c, 184
dm35425_adc_all_dma.c, 191
dm35425_adc_continuous_dma.c, 200
dm35425_adc_fifo.c, 205
dm35425_adio.c, 209
dm35425_adio_adv_int.c, 213
dm35425_adio_dma.c, 220
dm35425_adio_parallel_bus.c, 227
dm35425_dac.c, 230
dm35425_dac_dma.c, 235
dm35425_dac_fifo.c, 179
dm35425_ext_clocking.c, 240
dm35425_list_fb.c, 242

RANGE_OPTION
DM35425 Example Programs Constants, 118
RATE_OPTION
DM35425 Example Programs Constants, 118
readwrite
dm35425_ioctl_argument, 164
RECEIVER_OPTION
DM35425 Example Programs Constants, 118
REF_NUM_OPTION
DM35425 Example Programs Constants, 118
REFCLK_OPTION
DM35425 Example Programs Constants, 119
reference_count
dm35425_device_descriptor, 157
REFILL_FIFO_OPTION
DM35425 Example Programs Constants, 118
region
dm35425_pci_access_request, 172
remove_isr_flag

dm35425_device_descriptor, 157

SAMPLES_OPTION
DM35425 Example Programs Constants, 118
SENDER_OPTION
DM35425 Example Programs Constants, 118
setup_dacs
dm35425_adc_continuous_dma.c, 197
sigint_handler
dm35425_adc.c, 183
dm35425_adc_all_dma.c, 188
dm35425_adc_continuous_dma.c, 197
dm35425_adc_fifo.c, 204
dm35425_adio_adv_int.c, 212
dm35425_adio_dma.c, 219
dm35425_adio_parallel_bus.c, 225
dm35425_dac_dma.c, 234
dm35425_dac_fifo.c, 178
dm35425_ext_clocking.c, 239
size
dm35425_pci_access_request, 173
SIZE_OPTION
DM35425 Example Programs Constants, 119
SLAVE_OPTION
DM35425 Example Programs Constants, 119
START_OPTION
DM35425 Example Programs Constants, 118
STORE_OPTION
DM35425 Example Programs Constants, 119
sub_type
DM35425_Function_Block, 163
SYNC_CONN_OPTION
DM35425 Example Programs Constants, 119
SYNC_TERM_OPTION
DM35425 Example Programs Constants, 119
SYNCBUS_OPTION
DM35425 Example Programs Constants, 118

TERM_OPTION
DM35425 Example Programs Constants, 119
TEST_OPTION
DM35425 Example Programs Constants, 118
total_interrupt_count
dm35425_adio_adv_int.c, 213
type
DM35425_Function_Block, 163
type_revision
DM35425_Function_Block, 163

unexpected_interrupt_count
dm35425_adc_fifo.c, 206
USER_ID_OPTION
DM35425 Example Programs Constants, 119

valid_interrupt
dm35425_ioctl_interrupt_info_request, 168
VERBOSE_OPTION
DM35425 Example Programs Constants, 119
virt_addr
dm35425_dma_descriptor, 160
dm35425_pci_region, 174

WAVE_OPTION
DM35425 Example Programs Constants, 118