

libDaisy

Generated by Doxygen 1.9.1

1 libDaisy	1
1.1 Working with Daisy	1
1.2 Troubleshooting	1
2 Unit testing libDaisy	3
2.1 Why write tests?	3
2.2 How to run tests?	4
2.2.1 From the commandline	4
2.2.2 From Visual Studio Code	4
2.3 Getting started with unit testing	5
2.4 Drawbacks & things to watch out for	5
3 Working With GPIO	7
3.1 The CPP objects used	7
3.2 The Hardware Connections	7
3.3 GPIO Input	8
3.4 Breaking Down the Init function	8
3.4.1 Pin Argument	8
3.4.2 Mode Argument	9
3.4.3 Pull Argument	9
3.4.4 Speed Argument	9
3.4.5 Defaults	9
3.5 GPIO Output	10
3.6 Further Reading	10
4 Todo List	11
5 Module Index	13
5.1 Modules	13
6 Namespace Index	15
6.1 Namespace List	15
7 Hierarchical Index	17
7.1 Class Hierarchy	17
8 Class Index	21
8.1 Class List	21
9 File Index	25
9.1 File List	25
10 Module Documentation	27
10.1 LIBDAISY	27
10.1.1 Detailed Description	27
10.2 HUMAN_INTERFACE	27

10.2.1 Detailed Description	28
10.3 AUDIO	28
10.4 CONTROLS	28
10.4.1 Detailed Description	28
10.5 FEEDBACK	29
10.5.1 Detailed Description	29
10.6 EXTERNAL	29
10.6.1 Detailed Description	30
10.6.2 Macro Definition Documentation	30
10.6.2.1 FLT_FMT	30
10.6.2.2 FLT_FMT3	31
10.6.2.3 FLT_VAR	31
10.6.2.4 FLT_VAR3	31
10.6.2.5 LOGGER_BUFFER	31
10.6.2.6 LOGGER_NEWLINE	31
10.6.2.7 PPCAT	31
10.6.2.8 PPCAT_NX	32
10.6.2.9 STRINGIZE	32
10.6.2.10 STRINGIZE_NX	32
10.6.3 Enumeration Type Documentation	32
10.6.3.1 LoggerConsts	32
10.6.4 Function Documentation	32
10.6.4.1 AppendNewLine()	32
10.6.4.2 Logger()	33
10.6.4.3 NewLineSeqLength()	33
10.6.4.4 Print()	33
10.6.4.5 PrintLine()	33
10.6.4.6 PrintLineV()	33
10.6.4.7 PrintV()	33
10.6.4.8 StartLog()	33
10.6.4.9 TransmitBuf()	34
10.6.4.10 TransmitSync()	34
10.6.5 Variable Documentation	34
10.6.5.1 impl_	34
10.6.5.2 pc_sync_	34
10.6.5.3 tx_buff_	35
10.6.5.4 tx_ptr_	35
10.7 PERIPHERAL	35
10.7.1 Detailed Description	35
10.8 SERIAL	35
10.8.1 Detailed Description	36
10.8.2 Function Documentation	36

10.8.2.1 <code>dsy_spi_global_init()</code>	36
10.9 ANALOG_DIGITAL_CONVERSION	36
10.9.1 Detailed Description	36
10.10 OTHER	36
10.10.1 Detailed Description	37
10.10.2 Enumeration Type Documentation	37
10.10.2.1 <code>dsy_gpio_mode</code>	37
10.10.2.2 <code>dsy_gpio_pull</code>	37
10.10.3 Function Documentation	39
10.10.3.1 <code>dsy_gpio_deinit()</code>	39
10.10.3.2 <code>dsy_gpio_init()</code>	39
10.10.3.3 <code>dsy_gpio_read()</code>	39
10.10.3.4 <code>dsy_gpio_toggle()</code>	40
10.10.3.5 <code>dsy_gpio_write()</code>	40
10.11 SYSTEM	40
10.11.1 Detailed Description	41
10.11.2 Function Documentation	41
10.11.2.1 <code>dsy_dma_clear_cache_for_buffer()</code>	41
10.11.2.2 <code>dsy_dma_deinit()</code>	41
10.11.2.3 <code>dsy_dma_init()</code>	41
10.11.2.4 <code>dsy_dma_invalidate_cache_for_buffer()</code>	41
10.12 DEVICE	42
10.12.1 Detailed Description	42
10.13 SHIFTREGISTER	42
10.13.1 Detailed Description	42
10.14 FLASH	43
10.14.1 Detailed Description	46
10.14.2 Macro Definition Documentation	47
10.14.2.1 <code>BLOCK_ERASE_32K_CMD [1/2]</code>	47
10.14.2.2 <code>BLOCK_ERASE_32K_CMD [2/2]</code>	47
10.14.2.3 <code>BLOCK_ERASE_CMD [1/2]</code>	47
10.14.2.4 <code>BLOCK_ERASE_CMD [2/2]</code>	47
10.14.2.5 <code>CHIP_ERASE_CMD [1/2]</code>	47
10.14.2.6 <code>CHIP_ERASE_CMD [2/2]</code>	47
10.14.2.7 <code>CLEAR_EXT_READ_PARAM_CMD</code>	47
10.14.2.8 <code>DUAL_INOUT_FAST_READ_CMD [1/2]</code>	48
10.14.2.9 <code>DUAL_INOUT_FAST_READ_CMD [2/2]</code>	48
10.14.2.10 <code>DUAL_INOUT_FAST_READ_DTR_CMD [1/2]</code>	48
10.14.2.11 <code>DUAL_INOUT_FAST_READ_DTR_CMD [2/2]</code>	48
10.14.2.12 <code>DUAL_OUT_FAST_READ_CMD [1/2]</code>	48
10.14.2.13 <code>DUAL_OUT_FAST_READ_CMD [2/2]</code>	48
10.14.2.14 <code>ENTER_DEEP_POWER_DOWN [1/2]</code>	48

10.14.2.15 ENTER_DEEP_POWER_DOWN [2/2]	48
10.14.2.16 ENTER_QUAD_CMD [1/2]	49
10.14.2.17 ENTER_QUAD_CMD [2/2]	49
10.14.2.18 EXIT_DEEP_POWER_DOWN [1/2]	49
10.14.2.19 EXIT_DEEP_POWER_DOWN [2/2]	49
10.14.2.20 EXIT_QUAD_CMD [1/2]	49
10.14.2.21 EXIT_QUAD_CMD [2/2]	49
10.14.2.22 EXT_CHIP_ERASE_CMD [1/2]	49
10.14.2.23 EXT_CHIP_ERASE_CMD [2/2]	49
10.14.2.24 EXT_PROG_ERASE_RESUME_CMD [1/2]	50
10.14.2.25 EXT_PROG_ERASE_RESUME_CMD [2/2]	50
10.14.2.26 EXT_PROG_ERASE_SUSPEND_CMD [1/2]	50
10.14.2.27 EXT_PROG_ERASE_SUSPEND_CMD [2/2]	50
10.14.2.28 EXT_QUAD_IN_FAST_PROG_CMD	50
10.14.2.29 EXT_QUAD_IN_PAGE_PROG_CMD [1/2]	50
10.14.2.30 EXT_QUAD_IN_PAGE_PROG_CMD [2/2]	50
10.14.2.31 EXT_WRITE_READ_PARAM_REG_CMD	50
10.14.2.32 FAST_READ_CMD [1/2]	51
10.14.2.33 FAST_READ_CMD [2/2]	51
10.14.2.34 FAST_READ_DTR_CMD [1/2]	51
10.14.2.35 FAST_READ_DTR_CMD [2/2]	51
10.14.2.36 INFO_ROW_ERASE_CMD [1/2]	51
10.14.2.37 INFO_ROW_ERASE_CMD [2/2]	51
10.14.2.38 INFO_ROW_PROGRAM_CMD [1/2]	51
10.14.2.39 INFO_ROW_PROGRAM_CMD [2/2]	51
10.14.2.40 INFO_ROW_READ_CMD [1/2]	52
10.14.2.41 INFO_ROW_READ_CMD [2/2]	52
10.14.2.42 IS25LP064A_EAR_HIGHEST_SE	52
10.14.2.43 IS25LP064A_EAR_LOWEST_SEG	52
10.14.2.44 IS25LP064A_EAR_SECOND_SEG	52
10.14.2.45 IS25LP064A_EAR_THIRD_SEG	52
10.14.2.46 IS25LP064A_EVCR_DTRP	52
10.14.2.47 IS25LP064A_EVCR_DUAL	52
10.14.2.48 IS25LP064A_EVCR_ODS	53
10.14.2.49 IS25LP064A_EVCR_QUAD	53
10.14.2.50 IS25LP064A_EVCR_RH	53
10.14.2.51 IS25LP064A_FSR_ERERR	53
10.14.2.52 IS25LP064A_FSR_ERSUS	53
10.14.2.53 IS25LP064A_FSR_NBADDR	53
10.14.2.54 IS25LP064A_FSR_PGERR	53
10.14.2.55 IS25LP064A_FSR_PGSUS	53
10.14.2.56 IS25LP064A_FSR_PRERR	54

10.14.2.57 IS25LP064A_FSR_READY	54
10.14.2.58 IS25LP064A_NVCR_DTRP	54
10.14.2.59 IS25LP064A_NVCR_DUAL	54
10.14.2.60 IS25LP064A_NVCR_NB_DUMMY	54
10.14.2.61 IS25LP064A_NVCR_NBADDR	54
10.14.2.62 IS25LP064A_NVCR_ODS	54
10.14.2.63 IS25LP064A_NVCR_QUAB	54
10.14.2.64 IS25LP064A_NVCR_RH	55
10.14.2.65 IS25LP064A_NVCR_SEGMENT	55
10.14.2.66 IS25LP064A_NVCR_XIP	55
10.14.2.67 IS25LP064A_SR_QE	55
10.14.2.68 IS25LP064A_SR_SRWREN	55
10.14.2.69 IS25LP064A_SR_WIP	55
10.14.2.70 IS25LP064A_SR_WREN	55
10.14.2.71 IS25LP064A_VCR_NB_DUMMY	56
10.14.2.72 IS25LP064A_VCR_WRAP	56
10.14.2.73 IS25LP064A_VCR_XIP	56
10.14.2.74 IS25LP080D_EAR_HIGHEST_SE	56
10.14.2.75 IS25LP080D_EAR_LOWEST_SEG	56
10.14.2.76 IS25LP080D_EAR_SECOND_SEG	56
10.14.2.77 IS25LP080D_EAR_THIRD_SEG	56
10.14.2.78 IS25LP080D_EVCR_DTRP	56
10.14.2.79 IS25LP080D_EVCR_DUAL	57
10.14.2.80 IS25LP080D_EVCR_ODS	57
10.14.2.81 IS25LP080D_EVCR_QUAD	57
10.14.2.82 IS25LP080D_EVCR_RH	57
10.14.2.83 IS25LP080D_FSR_ERERR	57
10.14.2.84 IS25LP080D_FSR_ERSUS	57
10.14.2.85 IS25LP080D_FSR_NBADDR	57
10.14.2.86 IS25LP080D_FSR_PGERR	57
10.14.2.87 IS25LP080D_FSR_PGSUS	58
10.14.2.88 IS25LP080D_FSR_PRERR	58
10.14.2.89 IS25LP080D_FSR_READY	58
10.14.2.90 IS25LP080D_NVCR_DTRP	58
10.14.2.91 IS25LP080D_NVCR_DUAL	58
10.14.2.92 IS25LP080D_NVCR_NB_DUMMY	58
10.14.2.93 IS25LP080D_NVCR_NBADDR	58
10.14.2.94 IS25LP080D_NVCR_ODS	58
10.14.2.95 IS25LP080D_NVCR_QUAB	59
10.14.2.96 IS25LP080D_NVCR_RH	59
10.14.2.97 IS25LP080D_NVCR_SEGMENT	59
10.14.2.98 IS25LP080D_NVCR_XIP	59

10.14.2.99 IS25LP080D_SR_QE	59
10.14.2.100 IS25LP080D_SR_SRWREN	59
10.14.2.101 IS25LP080D_SR_WIP	59
10.14.2.102 IS25LP080D_SR_WREN	60
10.14.2.103 IS25LP080D_VCR_NB_DUMMY	60
10.14.2.104 IS25LP080D_VCR_WRAP	60
10.14.2.105 IS25LP080D_VCR_XIP	60
10.14.2.106 MULTIPLE_IO_READ_ID_CMD [1/2]	60
10.14.2.107 MULTIPLE_IO_READ_ID_CMD [2/2]	60
10.14.2.108 NO_OP [1/2]	60
10.14.2.109 NO_OP [2/2]	60
10.14.2.110 PAGE_PROG_CMD [1/3]	61
10.14.2.111 PAGE_PROG_CMD [2/3]	61
10.14.2.112 PAGE_PROG_CMD [3/3]	61
10.14.2.113 PROG_ERASE_RESUME_CMD [1/2]	61
10.14.2.114 PROG_ERASE_RESUME_CMD [2/2]	61
10.14.2.115 PROG_ERASE_SUSPEND_CMD [1/2]	61
10.14.2.116 PROG_ERASE_SUSPEND_CMD [2/2]	61
10.14.2.117 QUAD_IN_FAST_PROG_CMD	62
10.14.2.118 QUAD_IN_PAGE_PROG_CMD [1/2]	62
10.14.2.119 QUAD_IN_PAGE_PROG_CMD [2/2]	62
10.14.2.120 QUAD_INOUT_FAST_READ_CMD [1/2]	62
10.14.2.121 QUAD_INOUT_FAST_READ_CMD [2/2]	62
10.14.2.122 QUAD_INOUT_FAST_READ_DTR_CMD [1/2]	62
10.14.2.123 QUAD_INOUT_FAST_READ_DTR_CMD [2/2]	62
10.14.2.124 QUAD_OUT_FAST_READ_CMD [1/2]	62
10.14.2.125 QUAD_OUT_FAST_READ_CMD [2/2]	63
10.14.2.126 READ_CMD [1/2]	63
10.14.2.127 READ_CMD [2/2]	63
10.14.2.128 READ_EXT_READ_PARAM_CMD	63
10.14.2.129 READ_FUNCTION_REGISTER [1/2]	63
10.14.2.130 READ_FUNCTION_REGISTER [2/2]	63
10.14.2.131 READ_ID_CMD [1/2]	63
10.14.2.132 READ_ID_CMD [2/2]	63
10.14.2.133 READ_ID_CMD2 [1/2]	64
10.14.2.134 READ_ID_CMD2 [2/2]	64
10.14.2.135 READ_MANUFACT_AND_ID [1/2]	64
10.14.2.136 READ_MANUFACT_AND_ID [2/2]	64
10.14.2.137 READ_READ_PARAM_REG_CMD	64
10.14.2.138 READ_SERIAL_FLASH_DISCO_PARAM_CMD [1/2]	64
10.14.2.139 READ_SERIAL_FLASH_DISCO_PARAM_CMD [2/2]	64
10.14.2.140 READ_STATUS_REG_CMD [1/2]	64

10.14.2.141 READ_STATUS_REG_CMD [2/2]	65
10.14.2.142 READ_UNIQUE_ID [1/2]	65
10.14.2.143 READ_UNIQUE_ID [2/2]	65
10.14.2.144 RESET_ENABLE_CMD [1/2]	65
10.14.2.145 RESET_ENABLE_CMD [2/2]	65
10.14.2.146 RESET_MEMORY_CMD [1/2]	65
10.14.2.147 RESET_MEMORY_CMD [2/2]	65
10.14.2.148 SECTOR_ERASE_CMD [1/2]	65
10.14.2.149 SECTOR_ERASE_CMD [2/2]	66
10.14.2.150 SECTOR_ERASE_QPI_CMD [1/2]	66
10.14.2.151 SECTOR_ERASE_QPI_CMD [2/2]	66
10.14.2.152 SECTOR_LOCK [1/2]	66
10.14.2.153 SECTOR_LOCK [2/2]	66
10.14.2.154 SECTOR_UNLOCK [1/2]	66
10.14.2.155 SECTOR_UNLOCK [2/2]	66
10.14.2.156 WRITE_DISABLE_CMD [1/2]	66
10.14.2.157 WRITE_DISABLE_CMD [2/2]	67
10.14.2.158 WRITE_ENABLE_CMD [1/2]	67
10.14.2.159 WRITE_ENABLE_CMD [2/2]	67
10.14.2.160 WRITE_EXT_NV_READ_PARAM_REG_CMD	67
10.14.2.161 WRITE_EXT_READ_PARAM_REG_CMD	67
10.14.2.162 WRITE_FUNCTION_REGISTER [1/2]	67
10.14.2.163 WRITE_FUNCTION_REGISTER [2/2]	67
10.14.2.164 WRITE_NV_READ_PARAM_REG_CMD	67
10.14.2.165 WRITE_READ_PARAM_REG_CMD [1/2]	68
10.14.2.166 WRITE_READ_PARAM_REG_CMD [2/2]	68
10.14.2.167 WRITE_STATUS_REG_CMD [1/2]	68
10.14.2.168 WRITE_STATUS_REG_CMD [2/2]	68
10.15 CODEC	68
10.16 LED	69
10.17 SDRAM	69
10.17.1 Detailed Description	69
10.17.2 Macro Definition Documentation	69
10.17.2.1 DSY_SDRAM_BSS	69
10.17.2.2 DSY_SDRAM_DATA	69
10.18 BOARDS	70
10.18.1 Detailed Description	70
10.19 UI	70
10.20 UTILITY	73
10.20.1 Detailed Description	75
10.20.2 Macro Definition Documentation	76
10.20.2.1 BSP_SD_CardInfo	76

10.20.2.2 DMA_BUFFER_MEM_SECTION	76
10.20.2.3 DTCM_MEM_SECTION	76
10.20.2.4 F2S16_SCALE	76
10.20.2.5 F2S24_SCALE	76
10.20.2.6 F2S32_SCALE	76
10.20.2.7 F2S8_SCALE	76
10.20.2.8 F2U8_SCALE	77
10.20.2.9 FBIPMAX	77
10.20.2.10 FBIPMIN	77
10.20.2.11 IN_L	77
10.20.2.12 IN_R	77
10.20.2.13 MSD_ERROR	77
10.20.2.14 MSD_ERROR_SD_NOT_PRESENT	77
10.20.2.15 MSD_OK	78
10.20.2.16 OUT_L	78
10.20.2.17 OUT_R	78
10.20.2.18 S162F_SCALE	78
10.20.2.19 S242F_SCALE	78
10.20.2.20 S24SIGN	78
10.20.2.21 S322F_SCALE	78
10.20.2.22 S82F_SCALE	78
10.20.2.23 SD_DATATIMEOUT	79
10.20.2.24 SD_NOT_PRESENT	79
10.20.2.25 SD_PRESENT	79
10.20.2.26 SD_TRANSFER_BUSY	79
10.20.2.27 SD_TRANSFER_OK	79
10.20.2.28 U82F_SCALE	79
10.20.3 Enumeration Type Documentation	79
10.20.3.1 dsy_gpio_port	79
10.20.4 Function Documentation	80
10.20.4.1 BSP_SD_AbortCallback()	80
10.20.4.2 BSP_SD_Erase()	80
10.20.4.3 BSP_SD_GetCardInfo()	81
10.20.4.4 BSP_SD_GetCardState()	81
10.20.4.5 BSP_SD_Init()	81
10.20.4.6 BSP_SD_IsDetected()	81
10.20.4.7 BSP_SD_ITConfig()	82
10.20.4.8 BSP_SD_ReadBlocks()	82
10.20.4.9 BSP_SD_ReadBlocks_DMA()	82
10.20.4.10 BSP_SD_ReadCpltCallback()	83
10.20.4.11 BSP_SD_WriteBlocks()	83
10.20.4.12 BSP_SD_WriteBlocks_DMA()	83

10.20.4.13 BSP_SD_WriteCpltCallback()	84
10.20.4.14 cube()	84
10.20.4.15 dsy_get_unique_id()	84
10.20.4.16 dsy_hal_map_get_pin()	85
10.20.4.17 dsy_hal_map_get_port()	85
10.20.4.18 dsy_hal_map_gpio_clk_enable()	85
10.20.4.19 dsy_pin()	86
10.20.4.20 dsy_pin_cmp()	86
10.20.4.21 f2s16()	86
10.20.4.22 f2s24()	87
10.20.4.23 f2s32()	87
10.20.4.24 f2s8()	87
10.20.4.25 f2u8()	88
10.20.4.26 s162f()	88
10.20.4.27 s242f()	88
10.20.4.28 s322f()	89
10.20.4.29 s82f()	89
10.20.4.30 u82f()	89
10.20.5 Variable Documentation	89
10.20.5.1 Font_11x18	89
10.20.5.2 Font_16x26	90
10.20.5.3 Font_6x8	90
10.20.5.4 Font_7x10	90
10.21 Lcd	90
10.22 Dac	90
10.22.1 Detailed Description	91
10.22.2 Enumeration Type Documentation	91
10.22.2.1 Pin	91
10.22.2.2 Result [1/2]	91
10.22.2.3 Result [2/2]	92
10.22.2.4 VoltageRange	92
10.22.3 Function Documentation	92
10.22.3.1 Defaults() [1/2]	92
10.22.3.2 Defaults() [2/2]	93
10.22.3.3 Init()	93
10.22.3.4 ReadAnalogPinRaw()	93
10.22.3.5 ReadAnalogPinVolts()	93
10.22.3.6 ReadDigitalPin()	95
10.22.3.7 TwelveBitUintToVolts()	95
10.22.3.8 Update()	96
10.22.3.9 VoltsTo12BitUint()	96
10.22.3.10 WriteAnalogPinRaw()	96

10.22.3.11 WriteAnalogPinVolts()	97
10.22.3.12 WriteDigitalPin()	97
10.22.4 Variable Documentation	97
10.22.4.1 mode	97
10.22.4.2 range	98
10.22.4.3 spi_config	98
10.22.4.4 threshold	98
10.22.4.5 value	98
10.23 Externals	98
11 Namespace Documentation	99
11.1 daisy Namespace Reference	99
11.1.1 Detailed Description	103
11.1.2 Typedef Documentation	103
11.1.2.1 SSD130x4WireSpi128x32Driver	104
11.1.2.2 SSD130x4WireSpi128x64Driver	104
11.1.2.3 SSD130x4WireSpi64x32Driver	104
11.1.2.4 SSD130x4WireSpi64x48Driver	104
11.1.2.5 SSD130x4WireSpi98x16Driver	104
11.1.2.6 SSD130xl2c128x32Driver	104
11.1.2.7 SSD130xl2c128x64Driver	104
11.1.2.8 SSD130xl2c64x32Driver	105
11.1.2.9 SSD130xl2c64x48Driver	105
11.1.2.10 SSD130xl2c98x16Driver	105
11.1.3 Enumeration Type Documentation	105
11.1.3.1 Alignment	105
11.1.3.2 ApplicationTypeDef	105
11.1.3.3 ArrowButtonType	105
11.1.3.4 ChannelModeType	106
11.1.3.5 GPIOPort	106
11.1.3.6 LoggerDestination	106
11.1.3.7 MidiMessageType	107
11.1.3.8 SystemCommonType	107
11.1.3.9 SystemRealTimeType	108
11.1.3.10 WavFileFormatCode	108
11.1.4 Function Documentation	108
11.1.4.1 dsy_i2c_global_init()	108
11.1.5 Variable Documentation	109
11.1.5.1 kWavFileChunkId	109
11.1.5.2 kWavFileSubChunk1Id	109
11.1.5.3 kWavFileSubChunk2Id	109
11.1.5.4 kWavFileWavId	109

11.2 daisy::seed Namespace Reference	109
11.2.1 Detailed Description	110
11.2.2 Variable Documentation	110
11.2.2.1 A0	110
11.2.2.2 A12	111
11.2.2.3 D0	111
11.2.2.4 D31	111
12 Class Documentation	113
12.1 daisy::AbstractMenu Class Reference	113
12.1.1 Member Enumeration Documentation	114
12.1.1.1 ItemType	114
12.1.1.2 Orientation	114
12.1.2 Member Function Documentation	115
12.1.2.1 Init()	115
12.1.2.2 IsFunctionButtonDown()	115
12.1.2.3 OnArrowButton()	115
12.1.2.4 OnCancelButton()	116
12.1.2.5 OnFunctionButton()	116
12.1.2.6 OnMenuEncoderTurned()	117
12.1.2.7 OnOkayButton()	117
12.1.2.8 OnShow()	118
12.1.2.9 OnValueEncoderTurned()	118
12.1.2.10 OnValuePotMoved()	118
12.1.3 Member Data Documentation	119
12.1.3.1 allowEntering_	119
12.1.3.2 isEditing_	119
12.1.3.3 items_	119
12.1.3.4 numItems_	119
12.1.3.5 orientation_	119
12.1.3.6 selectedItemIdx_	119
12.2 daisy::AdcChannelConfig Struct Reference	120
12.2.1 Detailed Description	120
12.2.2 Member Enumeration Documentation	120
12.2.2.1 MuxPin	120
12.2.3 Member Function Documentation	120
12.2.3.1 InitMux()	121
12.2.3.2 InitSingle()	121
12.2.4 Member Data Documentation	121
12.2.4.1 mux_channels_	121
12.2.4.2 mux_pin_	121
12.2.4.3 pin_	122

12.3 daisy::AdcHandle Class Reference	122
12.3.1 Detailed Description	122
12.3.2 Member Enumeration Documentation	122
12.3.2.1 OverSampling	122
12.3.3 Member Function Documentation	123
12.3.3.1 Get()	123
12.3.3.2 GetFloat()	123
12.3.3.3 GetMux()	124
12.3.3.4 GetMuxFloat()	124
12.3.3.5 GetMuxPtr()	124
12.3.3.6 GetPtr()	125
12.3.3.7 Init()	125
12.3.3.8 Start()	125
12.3.3.9 Stop()	126
12.4 daisy::Ak4556 Class Reference	126
12.4.1 Member Function Documentation	126
12.4.1.1 DeInit()	126
12.4.1.2 Init()	126
12.5 daisy::AllNotesOffEvent Struct Reference	126
12.5.1 Detailed Description	127
12.5.2 Member Data Documentation	127
12.5.2.1 channel	127
12.6 daisy::AllSoundOffEvent Struct Reference	127
12.6.1 Detailed Description	127
12.6.2 Member Data Documentation	127
12.6.2.1 channel	128
12.7 daisy::AnalogControl Class Reference	128
12.7.1 Detailed Description	128
12.7.2 Constructor & Destructor Documentation	129
12.7.2.1 AnalogControl()	129
12.7.2.2 ~AnalogControl()	129
12.7.3 Member Function Documentation	129
12.7.3.1 GetRawFloat()	129
12.7.3.2 GetRawValue()	129
12.7.3.3 Init()	129
12.7.3.4 InitBipolarCv()	130
12.7.3.5 Process()	130
12.7.3.6 SetCoeff()	130
12.7.3.7 SetSampleRate()	130
12.7.3.8 Value()	131
12.8 daisy::AudioHandle Class Reference	131
12.8.1 Member Typedef Documentation	132

12.8.1.1 AudioCallback	132
12.8.1.2 InputBuffer	132
12.8.1.3 InterleavingAudioCallback	132
12.8.1.4 InterleavingInputBuffer	132
12.8.1.5 InterleavingOutputBuffer	132
12.8.1.6 OutputBuffer	132
12.8.2 Member Function Documentation	132
12.8.2.1 ChangeCallback() [1/2]	133
12.8.2.2 ChangeCallback() [2/2]	133
12.8.2.3 DelInit()	133
12.8.2.4 GetChannels()	133
12.8.2.5 GetConfig()	133
12.8.2.6 GetSampleRate()	133
12.8.2.7 Init() [1/2]	133
12.8.2.8 Init() [2/2]	134
12.8.2.9 SetBlockSize()	134
12.8.2.10 SetPostGain()	134
12.8.2.11 SetSampleRate()	134
12.8.2.12 Start() [1/2]	134
12.8.2.13 Start() [2/2]	134
12.8.2.14 Stop()	135
12.9 daisy::BlockingSpiTransport Class Reference	135
12.10 daisy::ButtonMonitor< BackendType, numButtons > Class Template Reference	135
12.10.1 Member Function Documentation	135
12.10.1.1 GetBackend()	136
12.10.1.2 GetNumButtonsMonitored()	136
12.10.1.3 Init()	136
12.10.1.4 IsButtonPressed()	136
12.10.1.5 Process()	137
12.11 daisy::ChannelPressureEvent Struct Reference	137
12.11.1 Detailed Description	137
12.11.2 Member Data Documentation	137
12.11.2.1 channel	137
12.11.2.2 pressure	138
12.12 daisy::Color Class Reference	138
12.12.1 Detailed Description	138
12.12.2 Member Enumeration Documentation	138
12.12.2.1 PresetColor	138
12.12.3 Member Function Documentation	139
12.12.3.1 Blue()	139
12.12.3.2 Green()	139
12.12.3.3 Init() [1/2]	139

12.12.3.4 Init() [2/2]	140
12.12.3.5 Red()	140
12.13 daisy::AudioHandle::Config Struct Reference	140
12.13.1 Detailed Description	140
12.14 daisy::BlockingSpiTransport::Config Struct Reference	140
12.14.1 Detailed Description	141
12.15 daisy::DacHandle::Config Struct Reference	141
12.15.1 Detailed Description	141
12.15.2 Member Data Documentation	141
12.15.2.1 target_samplerate	141
12.16 daisy::FatFSInterface::Config Struct Reference	142
12.16.1 Detailed Description	142
12.16.2 Member Enumeration Documentation	142
12.16.2.1 Media	142
12.17 daisy::GPIO::Config Struct Reference	142
12.17.1 Detailed Description	143
12.17.2 Constructor & Destructor Documentation	143
12.17.2.1 Config()	143
12.18 daisy::I2CHandle::Config Struct Reference	143
12.18.1 Detailed Description	144
12.18.2 Member Enumeration Documentation	144
12.18.2.1 Mode	144
12.18.2.2 Peripheral	144
12.18.2.3 Speed	144
12.18.3 Member Data Documentation	145
12.18.3.1 address	145
12.18.3.2 mode	145
12.18.3.3 periph	145
12.18.3.4	145
12.18.3.5 scl	145
12.18.3.6 sda	145
12.18.3.7 speed	145
12.19 daisy::LcdHD44780::Config Struct Reference	146
12.20 daisy::MAX11300Driver< Transport >::Config Struct Reference	146
12.21 daisy::MidiHandler< Transport >::Config Struct Reference	146
12.22 daisy::MidiUartTransport::Config Struct Reference	147
12.23 daisy::MidiUsbTransport::Config Struct Reference	147
12.24 daisy::OledDisplay< DisplayDriver >::Config Struct Reference	147
12.25 daisy::QspiHandle::Config Struct Reference	147
12.25.1 Detailed Description	148
12.25.2 Member Enumeration Documentation	148
12.25.2.1 Device	148

12.25.2.2 Mode	148
12.25.3 Member Data Documentation	150
12.25.3.1 clk	150
12.25.3.2 io0	150
12.25.3.3 io1	150
12.25.3.4 io2	150
12.25.3.5 io3	150
12.25.3.6 ncs	150
12.26 daisy::SaiHandle::Config Struct Reference	151
12.26.1 Detailed Description	151
12.26.2 Member Enumeration Documentation	151
12.26.2.1 BitDepth	151
12.26.2.2 Direction	152
12.26.2.3 Peripheral	152
12.26.2.4 SampleRate	152
12.26.2.5 Sync	152
12.27 daisy::SdmmcHandler::Config Struct Reference	152
12.27.1 Member Function Documentation	152
12.27.1.1 Defaults()	153
12.27.2 Member Data Documentation	153
12.27.2.1 clock_powersave	153
12.28 daisy::ShiftRegister4021< num_daisychained, num_parallel >::Config Struct Reference	153
12.28.1 Detailed Description	153
12.28.2 Member Data Documentation	153
12.28.2.1 clk	154
12.28.2.2 data	154
12.28.2.3 latch	154
12.29 daisy::SpiHandle::Config Struct Reference	154
12.29.1 Member Data Documentation	155
12.29.1.1 miso	155
12.29.1.2 mosi	155
12.29.1.3 nss	155
12.29.1.4 sclk	155
12.30 daisy::SSD130x4WireSpiTransport::Config Struct Reference	156
12.30.1 Member Data Documentation	156
12.30.1.1 dc	156
12.30.1.2 reset	156
12.31 daisy::SSD130xDriver< width, height, Transport >::Config Struct Reference	156
12.32 daisy::SSD130xI2CTransport::Config Struct Reference	157
12.33 daisy::System::Config Struct Reference	157
12.33.1 Detailed Description	157
12.33.2 Member Enumeration Documentation	157

12.33.2.1 SysClkFreq	158
12.33.3 Member Function Documentation	158
12.33.3.1 Boost()	158
12.33.3.2 Defaults()	158
12.34 daisy::TimerHandle::Config Struct Reference	158
12.34.1 Member Enumeration Documentation	158
12.34.1.1 CounterDir	158
12.34.1.2 Peripheral	158
12.35 daisy::UartHandler::Config Struct Reference	159
12.35.1 Member Data Documentation	159
12.35.1.1	160
12.35.1.2 rx	160
12.35.1.3 tx	160
12.36 daisy::USBHostHandle::Config Struct Reference	160
12.36.1 Detailed Description	160
12.37 daisy::WavWriter< transfer_size >::Config Struct Reference	160
12.37.1 Detailed Description	161
12.38 daisy::Wm8731::Config Struct Reference	161
12.38.1 Detailed Description	161
12.38.2 Member Enumeration Documentation	161
12.38.2.1 Format	162
12.38.2.2 WordLength	162
12.38.3 Member Function Documentation	162
12.38.3.1 Defaults()	162
12.38.4 Member Data Documentation	162
12.38.4.1 csb_pin_state	162
12.38.4.2 lr_swap	162
12.38.4.3 mcu_is_master	163
12.39 daisy::ControlChangeEvent Struct Reference	163
12.39.1 Detailed Description	163
12.39.2 Member Data Documentation	163
12.39.2.1 channel	163
12.39.2.2 control_number	163
12.39.2.3 value	164
12.40 daisy::CpuLoadMeter Class Reference	164
12.40.1 Member Function Documentation	164
12.40.1.1 GetAvgCpuLoad()	164
12.40.1.2 GetMaxCpuLoad()	164
12.40.1.3 GetMinCpuLoad()	164
12.40.1.4 Init()	164
12.40.1.5 OnBlockEnd()	165
12.40.1.6 OnBlockStart()	165

12.40.1.7 Reset()	165
12.41 daisy::AbstractMenu::CustomItem Class Reference	165
12.41.1 Detailed Description	166
12.41.2 Member Function Documentation	166
12.41.2.1 CanBeEnteredForEditing()	166
12.41.2.2 Draw()	166
12.41.2.3 ModifyValue() [1/2]	166
12.41.2.4 ModifyValue() [2/2]	167
12.41.2.5 OnOkayButton()	167
12.42 daisy::DacHandle Class Reference	167
12.42.1 Detailed Description	168
12.42.2 Member Typedef Documentation	168
12.42.2.1 DacCallback	168
12.42.3 Member Enumeration Documentation	168
12.42.3.1 BitDepth	168
12.42.3.2 BufferState	168
12.42.3.3 Channel	168
12.42.3.4 Mode	169
12.42.3.5 Result	169
12.42.4 Member Function Documentation	169
12.42.4.1 Init()	169
12.42.4.2 Start() [1/2]	169
12.42.4.3 Start() [2/2]	169
12.42.4.4 Stop()	170
12.42.4.5 WriteValue()	170
12.43 daisy::DaisyField Class Reference	170
12.43.1 Member Enumeration Documentation	171
12.43.1.1 anonymous enum	171
12.43.1.2 anonymous enum	172
12.43.1.3 anonymous enum	172
12.43.1.4 anonymous enum	172
12.43.2 Member Function Documentation	173
12.43.2.1 AudioBlockSize()	173
12.43.2.2 AudioCallbackRate()	173
12.43.2.3 AudioSampleRate()	173
12.43.2.4 ChangeAudioCallback() [1/2]	174
12.43.2.5 ChangeAudioCallback() [2/2]	174
12.43.2.6 DelayMs()	174
12.43.2.7 GetCv()	174
12.43.2.8 GetCvValue()	175
12.43.2.9 GetKnob()	175
12.43.2.10 GetKnobValue()	175

12.43.2.11 GetSwitch()	176
12.43.2.12 Init()	176
12.43.2.13 KeyboardFallingEdge()	176
12.43.2.14 KeyboardRisingEdge()	176
12.43.2.15 KeyboardState()	177
12.43.2.16 ProcessAllControls()	177
12.43.2.17 ProcessAnalogControls()	177
12.43.2.18 ProcessDigitalControls()	177
12.43.2.19 SetAudioBlockSize()	177
12.43.2.20 SetAudioSampleRate()	177
12.43.2.21 SetCvOut1()	178
12.43.2.22 SetCvOut2()	178
12.43.2.23 StartAdc()	178
12.43.2.24 StartAudio() [1/2]	178
12.43.2.25 StartAudio() [2/2]	178
12.43.2.26 StartDac()	178
12.43.2.27 StopAdc()	178
12.43.2.28 StopAudio()	179
12.43.2.29 VegasMode()	179
12.44 daisy::DaisyPatch Class Reference	179
12.44.1 Detailed Description	180
12.44.2 Member Enumeration Documentation	180
12.44.2.1 Ctrl	180
12.44.2.2 GateInput	180
12.44.3 Constructor & Destructor Documentation	180
12.44.3.1 DaisyPatch()	181
12.44.3.2 ~DaisyPatch()	181
12.44.4 Member Function Documentation	181
12.44.4.1 AudioBlockSize()	181
12.44.4.2 AudioCallbackRate()	181
12.44.4.3 AudioSampleRate()	181
12.44.4.4 ChangeAudioCallback()	181
12.44.4.5 DelayMs()	182
12.44.4.6 DisplayControls()	182
12.44.4.7 GetKnobValue()	182
12.44.4.8 Init()	182
12.44.4.9 ProcessAllControls()	182
12.44.4.10 ProcessAnalogControls()	183
12.44.4.11 ProcessDigitalControls()	183
12.44.4.12 SetAudioBlockSize()	183
12.44.4.13 SetAudioSampleRate()	183
12.44.4.14 StartAdc()	183

12.44.4.15 StartAudio()	183
12.44.4.16 StopAdc()	184
12.44.4.17 StopAudio()	184
12.44.5 Member Data Documentation	184
12.44.5.1 controls	184
12.44.5.2 display	184
12.44.5.3 encoder	184
12.44.5.4 gate_input	184
12.44.5.5 gate_output	184
12.44.5.6 midi	185
12.44.5.7 seed	185
12.45 daisy::patch_sm::DaisyPatchSM Class Reference	185
12.45.1 Detailed Description	187
12.45.2 Member Enumeration Documentation	187
12.45.2.1 PinBank	187
12.45.3 Member Function Documentation	187
12.45.3.1 AudioBlockSize()	187
12.45.3.2 AudioCallbackRate()	188
12.45.3.3 AudioSampleRate()	188
12.45.3.4 ChangeAudioCallback() [1/2]	188
12.45.3.5 ChangeAudioCallback() [2/2]	188
12.45.3.6 Delay()	188
12.45.3.7 GetAdcValue()	188
12.45.3.8 GetPin()	188
12.45.3.9 GetRandomFloat()	189
12.45.3.10 GetRandomValue()	189
12.45.3.11 Init()	189
12.45.3.12 Print()	189
12.45.3.13 PrintLine()	189
12.45.3.14 ProcessAllControls()	190
12.45.3.15 ProcessAnalogControls()	190
12.45.3.16 ProcessDigitalControls()	190
12.45.3.17 SetAudioBlockSize()	190
12.45.3.18 SetAudioSampleRate()	190
12.45.3.19 StartAdc()	190
12.45.3.20 StartAudio() [1/2]	190
12.45.3.21 StartAudio() [2/2]	191
12.45.3.22 StartDac()	191
12.45.3.23 StartLog()	191
12.45.3.24 StopAdc()	191
12.45.3.25 StopAudio()	191
12.45.3.26 StopDac()	191

12.45.3.27 ValidateQSPI()	191
12.45.3.28 ValidateSDRAM()	192
12.45.3.29 WriteCvOut()	192
12.45.4 Member Data Documentation	193
12.45.4.1 A1	193
12.45.4.2 system	193
12.45.4.3 user_led	193
12.46 daisy::DaisyPetal Class Reference	193
12.46.1 Detailed Description	194
12.46.2 Member Enumeration Documentation	194
12.46.2.1 FootswitchLed	194
12.46.2.2 Knob	195
12.46.2.3 RingLed	195
12.46.2.4 Sw	196
12.46.3 Constructor & Destructor Documentation	196
12.46.3.1 DaisyPetal()	196
12.46.3.2 ~DaisyPetal()	196
12.46.4 Member Function Documentation	196
12.46.4.1 AudioBlockSize()	196
12.46.4.2 AudioCallbackRate()	197
12.46.4.3 AudioSampleRate()	197
12.46.4.4 ChangeAudioCallback() [1/2]	197
12.46.4.5 ChangeAudioCallback() [2/2]	197
12.46.4.6 ClearLeds()	197
12.46.4.7 DelayMs()	198
12.46.4.8 GetExpression()	199
12.46.4.9 GetKnobValue()	199
12.46.4.10 Init()	199
12.46.4.11 ProcessAllControls()	199
12.46.4.12 ProcessAnalogControls()	199
12.46.4.13 ProcessDigitalControls()	200
12.46.4.14 SetAudioBlockSize()	200
12.46.4.15 SetAudioSampleRate()	200
12.46.4.16 SetFootswitchLed()	200
12.46.4.17 SetRingLed()	200
12.46.4.18 StartAdc()	202
12.46.4.19 StartAudio() [1/2]	202
12.46.4.20 StartAudio() [2/2]	202
12.46.4.21 StopAdc()	202
12.46.4.22 StopAudio()	202
12.46.4.23 UpdateLeds()	202
12.46.5 Member Data Documentation	203

12.46.5.1 encoder	203
12.46.5.2 expression	203
12.46.5.3 footswitch_led	203
12.46.5.4 knob	203
12.46.5.5 ring_led	203
12.46.5.6 seed	203
12.46.5.7 switches	203
12.47 daisy::DaisyPod Class Reference	204
12.47.1 Detailed Description	205
12.47.2 Member Enumeration Documentation	205
12.47.2.1 Knob	205
12.47.2.2 Sw	205
12.47.3 Member Function Documentation	205
12.47.3.1 AudioBlockSize()	206
12.47.3.2 AudioCallbackRate()	206
12.47.3.3 AudioSampleRate()	206
12.47.3.4 ChangeAudioCallback() [1/2]	206
12.47.3.5 ChangeAudioCallback() [2/2]	206
12.47.3.6 ClearLeds()	206
12.47.3.7 DelayMs()	207
12.47.3.8 GetKnobValue()	207
12.47.3.9 Init()	207
12.47.3.10 ProcessAllControls()	207
12.47.3.11 ProcessAnalogControls()	207
12.47.3.12 ProcessDigitalControls()	207
12.47.3.13 SetAudioBlockSize()	208
12.47.3.14 SetAudioSampleRate()	208
12.47.3.15 StartAdc()	208
12.47.3.16 StartAudio() [1/2]	208
12.47.3.17 StartAudio() [2/2]	208
12.47.3.18 StopAdc()	208
12.47.3.19 StopAudio()	208
12.47.3.20 UpdateLeds()	209
12.47.4 Member Data Documentation	209
12.47.4.1 button1	209
12.47.4.2 button2	209
12.47.4.3 buttons	209
12.47.4.4 encoder	209
12.47.4.5 knob1	209
12.47.4.6 knob2	209
12.47.4.7 knobs	210
12.47.4.8 led1	210

12.47.4.9 led2	210
12.47.4.10 seed	210
12.47.5 autotoc_md0	210
12.48 daisy::DaisySeed Class Reference	210
12.48.1 Detailed Description	211
12.48.2 Member Enumeration Documentation	211
12.48.2.1 BoardVersion	212
12.48.3 Member Function Documentation	213
12.48.3.1 AudioBlockSize()	213
12.48.3.2 AudioCallbackRate()	213
12.48.3.3 AudioSampleRate()	213
12.48.3.4 ChangeAudioCallback() [1/2]	213
12.48.3.5 ChangeAudioCallback() [2/2]	213
12.48.3.6 CheckBoardVersion()	214
12.48.3.7 Configure()	214
12.48.3.8 DelInit()	214
12.48.3.9 DelayMs()	214
12.48.3.10 GetPin()	214
12.48.3.11 Init()	214
12.48.3.12 Print()	215
12.48.3.13 PrintLine()	215
12.48.3.14 SetAudioBlockSize()	215
12.48.3.15 SetAudioSampleRate()	215
12.48.3.16 SetLed()	215
12.48.3.17 SetTestPoint()	215
12.48.3.18 StartAudio() [1/2]	216
12.48.3.19 StartAudio() [2/2]	216
12.48.3.20 StartLog()	216
12.48.3.21 StopAudio()	216
12.48.4 Member Data Documentation	216
12.48.4.1 adc	216
12.48.4.2 audio_handle	216
12.48.4.3 sdram_handle	217
12.48.4.4 usb_handle	217
12.49 daisy::DaisyVersio Class Reference	217
12.49.1 Detailed Description	218
12.49.2 Member Function Documentation	218
12.49.2.1 AudioBlockSize()	218
12.49.2.2 AudioCallbackRate()	218
12.49.2.3 AudioSampleRate()	218
12.49.2.4 ChangeAudioCallback() [1/2]	218
12.49.2.5 ChangeAudioCallback() [2/2]	219

12.49.2.6 DelayMs()	219
12.49.2.7 Gate()	219
12.49.2.8 GetKnobValue()	219
12.49.2.9 Init()	220
12.49.2.10 ProcessAllControls()	220
12.49.2.11 ProcessAnalogControls()	220
12.49.2.12 SetAudioBlockSize()	220
12.49.2.13 SetAudioSampleRate()	220
12.49.2.14 SetLed()	220
12.49.2.15 StartAdc()	220
12.49.2.16 StartAudio() [1/2]	221
12.49.2.17 StartAudio() [2/2]	221
12.49.2.18 StopAdc()	221
12.49.2.19 StopAudio()	221
12.49.2.20 SwitchPressed()	221
12.49.2.21 UpdateLeds()	221
12.50 dsy_gpio Struct Reference	221
12.50.1 Detailed Description	222
12.50.2 Member Data Documentation	222
12.50.2.1 mode	222
12.50.2.2 pin	222
12.50.2.3 pull	222
12.51 dsy_gpio_pin Struct Reference	222
12.51.1 Detailed Description	223
12.51.2 Member Data Documentation	223
12.51.2.1 pin	223
12.51.2.2 port	223
12.52 DSY_SD_CardInfoTypeDef Struct Reference	223
12.52.1 Detailed Description	224
12.52.2 Member Data Documentation	224
12.52.2.1 BlockNbr	224
12.52.2.2 BlockSize	224
12.52.2.3 CardSpeed	224
12.52.2.4 CardType	224
12.52.2.5 CardVersion	224
12.52.2.6 Class	224
12.52.2.7 LogBlockNbr	225
12.52.2.8 LogBlockSize	225
12.52.2.9 RelCardAdd	225
12.53 daisy::Encoder Class Reference	225
12.53.1 Detailed Description	225
12.53.2 Member Function Documentation	226

12.53.2.1 Debounce()	226
12.53.2.2 FallingEdge()	226
12.53.2.3 Increment()	226
12.53.2.4 Init()	226
12.53.2.5 Pressed()	226
12.53.2.6 RisingEdge()	226
12.53.2.7 SetUpdateRate()	226
12.53.2.8 TimeHeldMs()	227
12.54 daisy::FatFSInterface Class Reference	227
12.54.1 Detailed Description	228
12.54.2 Member Enumeration Documentation	228
12.54.2.1 Result	228
12.54.3 Member Function Documentation	228
12.54.3.1 Delinit()	228
12.54.3.2 GetConfig()	228
12.54.3.3 GetMutableConfig()	228
12.54.3.4 GetSDFileSystem()	229
12.54.3.5 GetSDPath()	229
12.54.3.6 GetUSBFileSystem()	229
12.54.3.7 GetUSBPath()	229
12.54.3.8 Init() [1/2]	229
12.54.3.9 Init() [2/2]	229
12.55 daisy::FIFO< T, capacity > Class Template Reference	230
12.55.1 Detailed Description	230
12.55.2 Constructor & Destructor Documentation	230
12.55.2.1 FIFO() [1/3]	230
12.55.2.2 FIFO() [2/3]	230
12.55.2.3 FIFO() [3/3]	231
12.55.3 Member Function Documentation	231
12.55.3.1 operator=()	231
12.56 daisy::FIFOBase< T > Class Template Reference	231
12.56.1 Detailed Description	232
12.56.2 Member Function Documentation	232
12.56.2.1 Back() [1/2]	232
12.56.2.2 Back() [2/2]	233
12.56.2.3 Clear()	233
12.56.2.4 Contains()	233
12.56.2.5 CountEqualTo()	233
12.56.2.6 Front() [1/2]	233
12.56.2.7 Front() [2/2]	233
12.56.2.8 GetCapacity()	233
12.56.2.9 GetNumElements()	234

12.56.2.10 Insert()	234
12.56.2.11 IsEmpty()	234
12.56.2.12 IsFull()	234
12.56.2.13 operator=()	234
12.56.2.14 operator[]() [1/2]	234
12.56.2.15 operator[]() [2/2]	235
12.56.2.16 PopFront()	235
12.56.2.17 PushBack() [1/2]	235
12.56.2.18 PushBack() [2/2]	235
12.56.2.19 Remove()	235
12.56.2.20 RemoveAllEqualTo()	235
12.57 daisy::FixedCapStr< capacity, CharType > Class Template Reference	236
12.58 daisy::FixedCapStrBase< CharType > Class Template Reference	236
12.59 FontDef Struct Reference	237
12.59.1 Detailed Description	238
12.59.2 Member Data Documentation	238
12.59.2.1 data	238
12.59.2.2 FontHeight	238
12.59.2.3 FontWidth	238
12.60 daisy::FullScreenMenuItem Class Reference	239
12.60.1 Member Function Documentation	239
12.60.1.1 Draw()	239
12.60.1.2 Init()	239
12.60.1.3 SetOneBitGraphicsDisplayToDrawTo()	240
12.61 daisy::GateIn Class Reference	240
12.61.1 Detailed Description	240
12.61.2 Constructor & Destructor Documentation	241
12.61.2.1 GateIn()	241
12.61.2.2 ~GateIn()	241
12.61.3 Member Function Documentation	241
12.61.3.1 Init()	241
12.61.3.2 State()	241
12.61.3.3 Trig()	241
12.62 daisy::GPIO Class Reference	242
12.62.1 Detailed Description	242
12.62.2 Member Enumeration Documentation	242
12.62.2.1 Mode	242
12.62.2.2 Pull	243
12.62.3 Member Function Documentation	243
12.62.3.1 GetConfig()	243
12.62.3.2 Write()	243
12.63 daisy::I2CHandle Class Reference	244

12.63.1 Detailed Description	244
12.63.2 Member Typedef Documentation	245
12.63.2.1 CallbackFunctionPtr	245
12.63.3 Member Enumeration Documentation	245
12.63.3.1 Direction	245
12.63.3.2 Result	245
12.63.4 Member Function Documentation	245
12.63.4.1 GetConfig()	245
12.63.4.2 Init()	246
12.63.4.3 ReadDataAtAddress()	246
12.63.4.4 ReceiveBlocking()	246
12.63.4.5 ReceiveDma()	247
12.63.4.6 TransmitBlocking()	247
12.63.4.7 TransmitDma()	247
12.63.4.8 WriteDataAtAddress()	248
12.64 daisy::AbstractMenu::ItemConfig Struct Reference	249
12.64.1 Member Data Documentation	249
12.64.1.1	249
12.64.1.2	249
12.64.1.3	249
12.64.1.4	250
12.64.1.5	250
12.64.1.6	250
12.64.1.7 itemObject	250
12.64.1.8 pageToOpen	250
12.64.1.9 text	250
12.64.1.10 type	250
12.64.1.11 valueToModify [1/2]	251
12.64.1.12 valueToModify [2/2]	251
12.65 daisy::LcdHD44780 Class Reference	251
12.65.1 Member Function Documentation	251
12.65.1.1 Clear()	251
12.65.1.2 Init()	251
12.65.1.3 Print()	252
12.65.1.4 PrintInt()	252
12.65.1.5 SetCursor()	252
12.66 daisy::Led Class Reference	253
12.66.1 Detailed Description	253
12.66.2 Member Function Documentation	253
12.66.2.1 Init()	253
12.66.2.2 Set()	254
12.66.2.3 SetSampleRate()	254

12.66.2.4 Update()	254
12.67 daisy::LedDriverPca9685< numDrivers, persistentBufferContents > Class Template Reference	254
12.67.1 Detailed Description	255
12.67.2 Member Typedef Documentation	255
12.67.2.1 DmaBuffer	255
12.67.3 Member Function Documentation	256
12.67.3.1 __attribute__()	256
12.67.3.2 GetNumLeds()	256
12.67.3.3 Init()	256
12.67.3.4 SetAllTo() [1/2]	257
12.67.3.5 SetAllTo() [2/2]	257
12.67.3.6 SetAllToRaw()	257
12.67.3.7 SetLed() [1/2]	257
12.67.3.8 SetLed() [2/2]	257
12.67.3.9 SetLedRaw()	258
12.67.3.10 SwapBuffersAndTransmit()	258
12.68 daisy::LocalControlEvent Struct Reference	258
12.68.1 Detailed Description	258
12.68.2 Member Data Documentation	258
12.68.2.1 channel	258
12.68.2.2 local_control_off	259
12.68.2.3 local_control_on	259
12.69 daisy::Logger< dest > Class Template Reference	259
12.69.1 Detailed Description	260
12.70 daisy::Logger< LOGGER_NONE > Class Reference	260
12.70.1 Detailed Description	260
12.71 daisy::LoggerImpl< dest > Class Template Reference	261
12.71.1 Detailed Description	261
12.71.2 Member Function Documentation	261
12.71.2.1 Init()	261
12.71.2.2 Transmit()	261
12.72 daisy::LoggerImpl< LOGGER_EXTERNAL > Class Reference	262
12.72.1 Detailed Description	262
12.72.2 Member Function Documentation	262
12.72.2.1 Init()	262
12.72.2.2 Transmit()	262
12.72.3 Member Data Documentation	262
12.72.3.1 usb_handle_	263
12.73 daisy::LoggerImpl< LOGGER_INTERNAL > Class Reference	263
12.73.1 Detailed Description	263
12.73.2 Member Function Documentation	263
12.73.2.1 Init()	263

12.73.2.2 <code>Transmit()</code>	263
12.73.3 Member Data Documentation	264
12.73.3.1 <code>usb_handle_</code>	264
12.74 <code>daisy::LoggerImpl< LOGGER_SEMIHOST ></code> Class Reference	264
12.74.1 Detailed Description	264
12.74.2 Member Function Documentation	264
12.74.2.1 <code>Init()</code>	264
12.74.2.2 <code>Transmit()</code>	265
12.75 <code>daisy::MappedFloatValue</code> Class Reference	265
12.75.1 Member Enumeration Documentation	265
12.75.1.1 <code>Mapping</code>	265
12.75.2 Constructor & Destructor Documentation	266
12.75.2.1 <code>MappedFloatValue()</code>	266
12.75.3 Member Function Documentation	266
12.75.3.1 <code>ApppentToString()</code>	266
12.75.3.2 <code>Get()</code>	267
12.75.3.3 <code>GetAs0to1()</code>	267
12.75.3.4 <code>GetPtr()</code>	267
12.75.3.5 <code>operator float()</code>	267
12.75.3.6 <code>operator=()</code>	267
12.75.3.7 <code>ResetToDefault()</code>	268
12.75.3.8 <code>Set()</code>	268
12.75.3.9 <code>SetFrom0to1()</code>	268
12.75.3.10 <code>Step()</code>	268
12.76 <code>daisy::MappedIntValue</code> Class Reference	268
12.76.1 Constructor & Destructor Documentation	269
12.76.1.1 <code>MappedIntValue()</code>	269
12.76.2 Member Function Documentation	269
12.76.2.1 <code>ApppentToString()</code>	269
12.76.2.2 <code>Get()</code>	270
12.76.2.3 <code>GetAs0to1()</code>	270
12.76.2.4 <code>GetPtr()</code>	270
12.76.2.5 <code>operator int()</code>	270
12.76.2.6 <code>operator=()</code>	270
12.76.2.7 <code>ResetToDefault()</code>	271
12.76.2.8 <code>Set()</code>	271
12.76.2.9 <code>SetFrom0to1()</code>	271
12.76.2.10 <code>Step()</code>	271
12.77 <code>daisy::MappedStringValue</code> Class Reference	271
12.77.1 Constructor & Destructor Documentation	272
12.77.1.1 <code>MappedStringValue()</code>	272
12.77.2 Member Function Documentation	272

12.77.2.1 AppentToString()	272
12.77.2.2 GetAs0to1()	273
12.77.2.3 GetIndex()	273
12.77.2.4 GetIndexPtr()	273
12.77.2.5 GetString()	273
12.77.2.6 operator const char *()	273
12.77.2.7 operator int()	273
12.77.2.8 operator=(())	274
12.77.2.9 ResetToDefault()	274
12.77.2.10 SetFrom0to1()	274
12.77.2.11 SetIndex()	274
12.77.2.12 Step()	274
12.78 daisy::MappedValue Class Reference	275
12.78.1 Member Function Documentation	275
12.78.1.1 AppentToString()	275
12.78.1.2 GetAs0to1()	275
12.78.1.3 ResetToDefault()	276
12.78.1.4 SetFrom0to1()	276
12.78.1.5 Step()	276
12.79 daisy::MAX11300Driver< Transport > Class Template Reference	276
12.79.1 Detailed Description	277
12.80 daisy::MidiEvent Struct Reference	278
12.80.1 Detailed Description	278
12.80.2 Member Function Documentation	278
12.80.2.1 AsChannelPressure()	279
12.80.2.2 AsControlChange()	279
12.80.2.3 AsNoteOff()	279
12.80.2.4 AsNoteOn()	279
12.80.2.5 AsPitchBend()	279
12.80.2.6 AsPolyphonicKeyPressure()	279
12.80.2.7 AsProgramChange()	279
12.80.3 Member Data Documentation	279
12.80.3.1 channel	280
12.80.3.2 data	280
12.80.3.3 sysex_data	280
12.80.3.4 type	280
12.81 daisy::MidiHandler< Transport > Class Template Reference	280
12.81.1 Detailed Description	281
12.81.2 Member Function Documentation	281
12.81.2.1 HasEvents()	281
12.81.2.2 Init()	281
12.81.2.3 Listen()	282

12.81.2.4 Parse()	282
12.81.2.5 PopEvent()	282
12.81.2.6 SendMessage()	282
12.81.2.7 StartReceive()	282
12.82 daisy::MidiUartTransport Class Reference	283
12.83 daisy::MidiUsbTransport Class Reference	283
12.84 daisy::MonoModeOnEvent Struct Reference	283
12.84.1 Detailed Description	284
12.84.2 Member Data Documentation	284
12.84.2.1 channel	284
12.84.2.2 num_channels	284
12.85 daisy::MTCQuarterFrameEvent Struct Reference	284
12.85.1 Detailed Description	284
12.85.2 Member Data Documentation	285
12.85.2.1 message_type	285
12.85.2.2 value	285
12.86 daisy::NoteOffEvent Struct Reference	285
12.86.1 Detailed Description	285
12.86.2 Member Data Documentation	285
12.86.2.1 channel	285
12.86.2.2 note	286
12.86.2.3 velocity	286
12.87 daisy::NoteOnEvent Struct Reference	286
12.87.1 Detailed Description	286
12.87.2 Member Data Documentation	286
12.87.2.1 channel	286
12.87.2.2 note	286
12.87.2.3 velocity	287
12.88 daisy::OledDisplay< DisplayDriver > Class Template Reference	287
12.88.1 Detailed Description	287
12.88.2 Member Function Documentation	288
12.88.2.1 DrawPixel()	288
12.88.2.2 Fill()	288
12.88.2.3 Update()	288
12.89 daisy::OmniModeOffEvent Struct Reference	289
12.89.1 Detailed Description	289
12.89.2 Member Data Documentation	289
12.89.2.1 channel	289
12.90 daisy::OmniModeOnEvent Struct Reference	289
12.90.1 Detailed Description	289
12.90.2 Member Data Documentation	289
12.90.2.1 channel	290

12.91 daisy::OneBitGraphicsDisplay Class Reference	290
12.91.1 Detailed Description	291
12.91.2 Member Function Documentation	291
12.91.2.1 DrawArc()	291
12.91.2.2 DrawCircle()	291
12.91.2.3 DrawLine()	292
12.91.2.4 DrawPixel()	292
12.91.2.5 DrawRect() [1/2]	292
12.91.2.6 DrawRect() [2/2]	293
12.91.2.7 Fill()	293
12.91.2.8 SetCursor()	294
12.91.2.9 Update()	294
12.91.2.10 WriteChar()	294
12.91.2.11 WriteString()	295
12.91.2.12 WriteStringAligned()	295
12.92 daisy::OneBitGraphicsDisplayImpl< ChildType > Class Template Reference	296
12.92.1 Detailed Description	296
12.92.2 Member Function Documentation	297
12.92.2.1 DrawArc()	297
12.92.2.2 DrawLine()	297
12.92.2.3 DrawRect()	298
12.92.2.4 WriteChar()	298
12.92.2.5 WriteString()	299
12.92.2.6 WriteStringAligned()	299
12.93 daisy::Parameter Class Reference	300
12.93.1 Detailed Description	300
12.93.2 Member Enumeration Documentation	300
12.93.2.1 Curve	300
12.93.3 Constructor & Destructor Documentation	301
12.93.3.1 Parameter()	301
12.93.3.2 ~Parameter()	301
12.93.4 Member Function Documentation	301
12.93.4.1 Init()	301
12.93.4.2 Process()	301
12.93.4.3 Value()	302
12.94 daisy::Pcm3060 Class Reference	302
12.94.1 Member Function Documentation	302
12.94.1.1 Init()	302
12.95 daisy::PersistentStorage< SettingStruct > Class Template Reference	303
12.95.1 Detailed Description	303
12.95.2 Member Enumeration Documentation	303
12.95.2.1 State	304

12.95.3 Constructor & Destructor Documentation	304
12.95.3.1 PersistentStorage()	304
12.95.4 Member Function Documentation	304
12.95.4.1 GetSettings()	304
12.95.4.2 GetState()	304
12.95.4.3 Init()	305
12.95.4.4 RestoreDefaults()	305
12.95.4.5 Save()	305
12.96 daisy::Pin Struct Reference	305
12.96.1 Detailed Description	306
12.96.2 Constructor & Destructor Documentation	306
12.96.2.1 Pin()	306
12.96.3 Member Function Documentation	306
12.96.3.1 IsValid()	307
12.96.3.2 operator dsy_gpio_pin()	307
12.97 daisy::PitchBendEvent Struct Reference	307
12.97.1 Detailed Description	307
12.97.2 Member Data Documentation	307
12.97.2.1 channel	307
12.97.2.2 value	308
12.98 daisy::PolyModeOnEvent Struct Reference	308
12.98.1 Detailed Description	308
12.98.2 Member Data Documentation	308
12.98.2.1 channel	308
12.99 daisy::PolyphonicKeyPressureEvent Struct Reference	308
12.99.1 Detailed Description	309
12.100 daisy::PotMonitor< BackendType, numPots > Class Template Reference	309
12.100.1 Member Function Documentation	309
12.100.1.1 GetBackend()	309
12.100.1.2 GetCurrentPotValue()	309
12.100.1.3 GetNumPotsMonitored()	310
12.100.1.4 Init()	310
12.100.1.5 IsMoving()	310
12.100.1.6 Process()	311
12.101 daisy::ProgramChangeEvent Struct Reference	311
12.101.1 Detailed Description	311
12.101.2 Member Data Documentation	311
12.101.2.1 channel	311
12.101.2.2 program	312
12.102 daisy::QSPIHandle Class Reference	312
12.102.1 Detailed Description	312
12.102.2 Member Enumeration Documentation	312

12.102.2.1 Status	312
12.102.3 Member Function Documentation	313
12.102.3.1 DelInit()	313
12.102.3.2 Erase()	313
12.102.3.3 EraseSector()	313
12.102.3.4 GetConfig()	314
12.102.3.5 GetData()	314
12.102.3.6 GetStatus()	314
12.102.3.7 Init()	314
12.102.3.8 Write()	315
12.102.3.9 WritePage()	315
12.103 daisy::Random Class Reference	316
12.103.1 Detailed Description	316
12.103.2 Member Function Documentation	316
12.103.2.1 DelInit()	316
12.103.2.2 GetFloat()	316
12.103.2.3 GetValue()	317
12.103.2.4 Init()	317
12.103.2.5 IsReady()	317
12.104 daisy::Rectangle Class Reference	318
12.105 daisy::ResetAllControllersEvent Struct Reference	319
12.105.1 Detailed Description	319
12.105.2 Member Data Documentation	319
12.105.2.1 channel	319
12.105.2.2 value	319
12.106 daisy::RgbLed Class Reference	319
12.106.1 Detailed Description	320
12.106.2 Member Function Documentation	320
12.106.2.1 Init()	320
12.106.2.2 Set()	320
12.106.2.3 SetBlue()	320
12.106.2.4 SetColor()	321
12.106.2.5 SetGreen()	321
12.106.2.6 SetRed()	321
12.106.2.7 Update()	322
12.107 daisy::RingBuffer< T, size > Class Template Reference	322
12.107.1 Detailed Description	322
12.107.2 Member Function Documentation	322
12.107.2.1 Advance()	323
12.107.2.2 capacity()	323
12.107.2.3 Flush()	323
12.107.2.4 GetMutableBuffer()	323

12.107.2.5 ImmediateRead() [1/2]	323
12.107.2.6 ImmediateRead() [2/2]	323
12.107.2.7 Init()	324
12.107.2.8 isEmpty()	324
12.107.2.9 Overwrite() [1/2]	324
12.107.2.10 Overwrite() [2/2]	324
12.107.2.11 Read()	326
12.107.2.12 readable()	326
12.107.2.13 Swallow()	326
12.107.2.14 writable()	326
12.107.2.15 Write()	327
12.108 daisy::RingBuffer< T, 0 > Class Template Reference	327
12.108.1 Detailed Description	327
12.108.2 Member Function Documentation	328
12.108.2.1 capacity()	328
12.108.2.2 Flush()	328
12.108.2.3 ImmediateRead() [1/2]	328
12.108.2.4 ImmediateRead() [2/2]	328
12.108.2.5 Init()	329
12.108.2.6 Overwrite() [1/2]	329
12.108.2.7 Overwrite() [2/2]	329
12.108.2.8 Read()	329
12.108.2.9 readable()	330
12.108.2.10 writable()	330
12.108.2.11 Write()	330
12.109 daisy::SaiHandle Class Reference	330
12.109.1 Detailed Description	331
12.109.2 Member Typedef Documentation	331
12.109.2.1 CallbackFunctionPtr	331
12.109.3 Member Enumeration Documentation	332
12.109.3.1 Result	332
12.109.4 Member Function Documentation	332
12.109.4.1 DelInit()	332
12.109.4.2 GetBlockRate()	332
12.109.4.3 GetBlockSize()	332
12.109.4.4 GetConfig()	332
12.109.4.5 GetOffset()	332
12.109.4.6 GetSampleRate()	333
12.109.4.7 Init()	333
12.109.4.8 StartDma()	333
12.109.4.9 StopDma()	333
12.110 daisy::ScopedIrqBlocker Class Reference	333

12.110.1 Detailed Description	333
12.111 daisy::SdmmcHandler Class Reference	334
12.111.1 Detailed Description	334
12.111.2 Member Enumeration Documentation	334
12.111.2.1 BusWidth	334
12.111.2.2 Result	335
12.111.2.3 Speed	335
12.111.3 Member Function Documentation	335
12.111.3.1 Init()	335
12.112 SdramHandle Class Reference	335
12.112.1 Member Enumeration Documentation	336
12.112.1.1 Result	336
12.112.2 Member Function Documentation	336
12.112.2.1 Init()	336
12.113 daisy::ShiftRegister4021< num_daisychained, num_parallel > Class Template Reference	336
12.113.1 Member Function Documentation	337
12.113.1.1 Init()	337
12.113.1.2 State()	337
12.113.1.3 Update()	337
12.114 ShiftRegister595 Class Reference	337
12.114.1 Detailed Description	338
12.114.2 Member Enumeration Documentation	338
12.114.2.1 Pins	338
12.114.3 Member Function Documentation	338
12.114.3.1 Init()	338
12.114.3.2 Set()	339
12.114.3.3 Write()	339
12.115 daisy::SongPositionPointerEvent Struct Reference	339
12.115.1 Detailed Description	339
12.115.2 Member Data Documentation	340
12.115.2.1 position	340
12.116 daisy::SongSelectEvent Struct Reference	340
12.116.1 Detailed Description	340
12.116.2 Member Data Documentation	340
12.116.2.1 song	340
12.117 daisy::UI::SpecialControlIds Struct Reference	340
12.117.1 Detailed Description	341
12.117.2 Member Data Documentation	341
12.117.2.1 menuEncoderId	341
12.117.2.2 valueEncoderId	341
12.117.2.3 valuePotId	341
12.118 daisy::SpiHandle Class Reference	342

12.118.1 Detailed Description	342
12.118.2 Member Typedef Documentation	342
12.118.2.1 CallbackFunctionPtr	342
12.118.3 Member Enumeration Documentation	342
12.118.3.1 DmaDirection	342
12.118.3.2 Result	343
12.118.4 Member Function Documentation	343
12.118.4.1 BlockingReceive()	343
12.118.4.2 BlockingTransmit()	343
12.118.4.3 BlockingTransmitAndReceive()	344
12.118.4.4 CheckError()	344
12.118.4.5 DmaReceive()	344
12.118.4.6 DmaTransmit()	345
12.118.4.7 GetConfig()	345
12.118.4.8 Init()	345
12.119 daisy::SSD130x4WireSpiTransport Class Reference	346
12.119.1 Detailed Description	346
12.120 daisy::SSD130xDriver< width, height, Transport > Class Template Reference	346
12.120.1 Detailed Description	346
12.120.2 Member Function Documentation	347
12.120.2.1 Update()	347
12.121 daisy::SSD130xI2CTransport Class Reference	347
12.121.1 Detailed Description	347
12.122 daisy::Stack< T, capacity > Class Template Reference	347
12.122.1 Detailed Description	348
12.122.2 Constructor & Destructor Documentation	348
12.122.2.1 Stack() [1/3]	348
12.122.2.2 Stack() [2/3]	348
12.122.2.3 Stack() [3/3]	348
12.122.3 Member Function Documentation	349
12.122.3.1 operator=()	349
12.123 daisy::StackBase< T > Class Template Reference	349
12.123.1 Detailed Description	350
12.123.2 Member Function Documentation	350
12.123.2.1 Clear()	350
12.123.2.2 Contains()	350
12.123.2.3 CountEqualTo()	350
12.123.2.4 GetCapacity()	350
12.123.2.5 GetNumElements()	350
12.123.2.6 Insert()	351
12.123.2.7 IsEmpty()	351
12.123.2.8 IsFull()	351

12.123.2.9 operator=()	351
12.123.2.10 operator[]() [1/2]	351
12.123.2.11 operator[]() [2/2]	351
12.123.2.12 PopBack()	352
12.123.2.13 PushBack() [1/2]	352
12.123.2.14 PushBack() [2/2]	352
12.123.2.15 Remove()	352
12.123.2.16 RemoveAllEqualTo()	352
12.124 daisy::Switch Class Reference	352
12.124.1 Detailed Description	353
12.124.2 Member Enumeration Documentation	353
12.124.2.1 Polarity	353
12.124.2.2 Pull	354
12.124.2.3 Type	354
12.124.3 Member Function Documentation	354
12.124.3.1 Debounce()	354
12.124.3.2 FallingEdge()	354
12.124.3.3 Init() [1/2]	355
12.124.3.4 Init() [2/2]	355
12.124.3.5 Pressed()	355
12.124.3.6 RawState()	356
12.124.3.7 RisingEdge()	356
12.124.3.8 SetUpdateRate()	356
12.124.3.9 TimeHeldMs()	356
12.125 daisy::Switch3 Class Reference	357
12.126 daisy::System Class Reference	357
12.126.1 Member Enumeration Documentation	358
12.126.1.1 MemoryRegion	358
12.126.2 Member Function Documentation	358
12.126.2.1 DelInit()	358
12.126.2.2 Delay()	358
12.126.2.3 DelayTicks()	358
12.126.2.4 DelayUs()	359
12.126.2.5 GetConfig()	359
12.126.2.6 GetHClkFreq()	359
12.126.2.7 GetMemoryRegion()	359
12.126.2.8 GetNow()	360
12.126.2.9 GetPclk1Freq()	360
12.126.2.10 GetPclk2Freq()	360
12.126.2.11 GetProgramMemoryRegion()	360
12.126.2.12 GetSysClkFreq()	360
12.126.2.13 GetTick()	360

12.126.2.14 GetTickFreq()	361
12.126.2.15 GetUs()	361
12.126.2.16 Init() [1/2]	361
12.126.2.17 Init() [2/2]	361
12.126.2.18 JumpToQspi()	361
12.126.2.19 ResetToBootloader()	361
12.126.3 Member Data Documentation	361
12.126.3.1 kQspiBootloaderOffset	362
12.127 daisy::SystemExclusiveEvent Struct Reference	362
12.127.1 Detailed Description	362
12.127.2 Member Data Documentation	362
12.127.2.1 data	362
12.128 daisy::TimerHandle Class Reference	362
12.128.1 Detailed Description	363
12.128.2 Member Enumeration Documentation	364
12.128.2.1 Result	364
12.128.3 Member Function Documentation	364
12.128.3.1 DelInit()	364
12.128.3.2 DelayMs()	364
12.128.3.3 DelayTick()	364
12.128.3.4 DelayUs()	364
12.128.3.5 GetConfig()	364
12.128.3.6 GetFreq()	365
12.128.3.7 GetMs()	365
12.128.3.8 GetTick()	365
12.128.3.9 GetUs()	365
12.128.3.10 Init()	365
12.128.3.11 SetPeriod()	365
12.128.3.12 SetPrescaler()	366
12.128.3.13 Start()	366
12.128.3.14 Stop()	366
12.129 daisy::UartHandler Class Reference	366
12.129.1 Detailed Description	367
12.129.2 Member Enumeration Documentation	367
12.129.2.1 Result	367
12.129.3 Member Function Documentation	367
12.129.3.1 CheckError()	367
12.129.3.2 FlushRx()	367
12.129.3.3 GetConfig()	368
12.129.3.4 Init()	368
12.129.3.5 PollReceive()	368
12.129.3.6 PollTx()	368

12.129.3.7 PopRx()	369
12.129.3.8 Readable()	369
12.129.3.9 RxActive()	369
12.129.3.10 StartRx()	369
12.130 daisy::Ui Class Reference	370
12.130.1 Member Function Documentation	370
12.130.1.1 ClosePage()	370
12.130.1.2 GetPrimaryOneBitGraphicsDisplayId()	370
12.130.1.3 GetSpecialControlIds()	370
12.130.1.4 Init()	370
12.130.1.5 Mute()	371
12.130.1.6 OpenPage()	371
12.130.1.7 Process()	371
12.130.2 Member Data Documentation	371
12.130.2.1 invalidCanvasId	371
12.131 daisy::UiCanvasDescriptor Struct Reference	372
12.131.1 Member Typedef Documentation	372
12.131.1.1 ClearFuncPtr	372
12.131.1.2 FlushFuncPtr	372
12.131.2 Member Data Documentation	372
12.131.2.1 handle_	372
12.131.2.2 id_	373
12.131.2.3 screenSaverTimeOut	373
12.131.2.4 updateRateMs_	373
12.132 daisy::UiEventQueue Class Reference	373
12.132.1 Member Function Documentation	373
12.132.1.1 __attribute__()	374
12.132.1.2 AddButtonPressed()	375
12.132.1.3 AddButtonReleased()	375
12.132.1.4 AddEncoderActivityChanged()	375
12.132.1.5 AddEncoderTurned()	375
12.132.1.6 AddPotActivityChanged()	375
12.132.1.7 AddPotMoved()	375
12.132.1.8 GetAndRemoveNextEvent()	376
12.132.1.9 IsQueueEmpty()	376
12.132.2 Member Data Documentation	376
12.132.2.1 invalidButtonId	376
12.132.2.2 invalidEncoderId	376
12.132.2.3 invalidPotId	376
12.133 daisy::UiPage Class Reference	376
12.133.1 Member Function Documentation	377
12.133.1.1 Close()	377

12.133.1.2 Draw()	377
12.133.1.3 GetParentUI() [1/2]	378
12.133.1.4 GetParentUI() [2/2]	378
12.133.1.5 IsActive()	378
12.133.1.6 IsOpaque()	378
12.133.1.7 OnArrowButton()	378
12.133.1.8 OnButton()	379
12.133.1.9 OnCancelButton()	379
12.133.1.10 OnEncoderActivityChanged()	379
12.133.1.11 OnEncoderTurned()	381
12.133.1.12 OnFocusGained()	381
12.133.1.13 OnFocusLost()	381
12.133.1.14 OnFunctionButton()	381
12.133.1.15 OnHide()	382
12.133.1.16 OnMenuEncoderActivityChanged()	382
12.133.1.17 OnMenuEncoderTurned()	382
12.133.1.18 OnOkayButton()	383
12.133.1.19 OnPotActivityChanged()	383
12.133.1.20 OnPotMoved()	384
12.133.1.21 OnShow()	384
12.133.1.22 OnValueEncoderActivityChanged()	384
12.133.1.23 OnValueEncoderTurned()	385
12.133.1.24 OnValuePotActivityChanged()	385
12.133.1.25 OnValuePotMoved()	385
12.134 UsbHandle Class Reference	386
12.134.1 Detailed Description	387
12.134.2 Member Typedef Documentation	387
12.134.2.1 ReceiveCallback [1/2]	387
12.134.2.2 ReceiveCallback [2/2]	387
12.134.3 Member Enumeration Documentation	387
12.134.3.1 Result [1/2]	387
12.134.3.2 Result [2/2]	387
12.134.3.3 UsbPeriph [1/2]	387
12.134.3.4 UsbPeriph [2/2]	388
12.134.4 Member Function Documentation	388
12.134.4.1 DelInit() [1/2]	388
12.134.4.2 DelInit() [2/2]	388
12.134.4.3 Init() [1/2]	389
12.134.4.4 Init() [2/2]	389
12.134.4.5 SetReceiveCallback() [1/2]	389
12.134.4.6 SetReceiveCallback() [2/2]	390
12.134.4.7 TransmitExternal() [1/2]	390

12.134.4.8 TransmitExternal() [2/2]	390
12.134.4.9 TransmitInternal() [1/2]	391
12.134.4.10 TransmitInternal() [2/2]	391
12.135 daisy::USBHostHandle Class Reference	391
12.135.1 Detailed Description	392
12.135.2 Member Function Documentation	392
12.135.2.1 Deinit()	392
12.135.2.2 GetPresent()	392
12.135.2.3 GetReady()	392
12.135.2.4 Init()	392
12.135.2.5 Process()	393
12.136 daisy::VoctCalibration Class Reference	393
12.136.1 Detailed Description	393
12.136.2 Member Function Documentation	393
12.136.2.1 GetData()	393
12.136.2.2 ProcessInput()	394
12.136.2.3 Record()	394
12.136.2.4 SetData()	394
12.137 daisy::WAV_FormatTypeDef Struct Reference	395
12.137.1 Detailed Description	395
12.137.2 Member Data Documentation	395
12.137.2.1 AudioFormat	395
12.137.2.2 BitPerSample	395
12.137.2.3 BlockAlign	395
12.137.2.4 ByteRate	396
12.137.2.5 ChunkId	396
12.137.2.6 FileFormat	396
12.137.2.7 FileSize	396
12.137.2.8 NbrChannels	396
12.137.2.9 SampleRate	396
12.137.2.10 SubChunk1ID	396
12.137.2.11 SubChunk1Size	396
12.137.2.12 SubChunk2ID	397
12.137.2.13 SubChunk2Size	397
12.138 daisy::WaveTableLoader Class Reference	397
12.138.1 Detailed Description	397
12.138.2 Member Function Documentation	397
12.138.2.1 GetTable()	398
12.138.2.2 Import()	398
12.138.2.3 Init()	398
12.138.2.4 SetWaveTableInfo()	398
12.139 daisy::WavFileInfo Struct Reference	398

12.139.1 Detailed Description	399
12.139.2 Member Data Documentation	399
12.139.2.1 name	399
12.139.2.2 raw_data	399
12.140 daisy::WavPlayer Class Reference	399
12.140.1 Detailed Description	399
12.140.2 Member Function Documentation	400
12.140.2.1 Close()	400
12.140.2.2 GetCurrentFile()	400
12.140.2.3 GetLooping()	400
12.140.2.4 GetNumberFiles()	400
12.140.2.5 Init()	400
12.140.2.6 Open()	400
12.140.2.7 Prepare()	401
12.140.2.8 Restart()	401
12.140.2.9 SetLooping()	401
12.140.2.10 Stream()	401
12.141 daisy::WavWriter< transfer_size > Class Template Reference	401
12.141.1 Detailed Description	402
12.141.2 Member Enumeration Documentation	403
12.141.2.1 BufferState	403
12.141.2.2 Result	403
12.141.3 Member Function Documentation	403
12.141.3.1 GetLengthSamps()	403
12.141.3.2 GetLengthSeconds()	403
12.141.3.3 Init()	403
12.141.3.4 IsRecording()	404
12.141.3.5 OpenFile()	404
12.141.3.6 Sample()	404
12.141.3.7 SaveFile()	404
12.141.3.8 Write()	404
12.142 daisy::Wm8731 Class Reference	405
12.142.1 Detailed Description	405
12.142.2 Member Enumeration Documentation	405
12.142.2.1 Result	405
12.142.3 Member Function Documentation	406
12.142.3.1 Init()	406
13 File Documentation	407
13.1 src/sys/ffconf.h File Reference	407
13.1.1 Detailed Description	408
13.1.2 Macro Definition Documentation	408

13.1.2.1 _CODE_PAGE	408
13.1.2.2 _FFCONF	408
13.1.2.3 _FS_EXFAT	409
13.1.2.4 _FS_LOCK	409
13.1.2.5 _FS_MINIMIZE	409
13.1.2.6 _FS_NOFSINFO	409
13.1.2.7 _FS_NORTC	409
13.1.2.8 _FS_READONLY	409
13.1.2.9 _FS_REENTRANT	410
13.1.2.10 _FS_RPATH	410
13.1.2.11 _FS_TIMEOUT	410
13.1.2.12 _FS_TINY	410
13.1.2.13 _LFN_UNICODE	410
13.1.2.14 _MAX_LFN	410
13.1.2.15 _MAX_SS	411
13.1.2.16 _MIN_SS	411
13.1.2.17 _MULTI_PARTITION	411
13.1.2.18 _NORTC_MDAY	411
13.1.2.19 _NORTC_MON	411
13.1.2.20 _NORTC_YEAR	411
13.1.2.21 _STR_VOLUME_ID	412
13.1.2.22 _STRFUNC_ENCODE	412
13.1.2.23 _SYNC_t	412
13.1.2.24 _USE_CHMOD	412
13.1.2.25 _USE_EXPAND	412
13.1.2.26 _USE_FASTSEEK	412
13.1.2.27 _USE_FIND	413
13.1.2.28 _USE_FORWARD	413
13.1.2.29 _USE_LABEL	413
13.1.2.30 _USE_LFN	413
13.1.2.31 _USE_MKFS	413
13.1.2.32 _USE_STRFUNC	413
13.1.2.33 _USE_TRIM	413
13.1.2.34 _VOLUME_STRS	414
13.1.2.35 _VOLUMES	414
13.1.2.36 ff_free	414
13.1.2.37 ff_malloc	414
13.2 src/util/usbh_diskio.h File Reference	414
13.2.1 Detailed Description	414
Index	415

Chapter 1

libDaisy

libDaisy is a C++ hardware support library for the electrosmith Daisy platform.

the contents of libDaisy fall into several categories, and levels:

- `boards`: these are user-facing APIs for interfacing directly with the official Daisy hardware
- `hid`: these are classes that a user is most likely to directly interact with (switches, encoders, analog controls, etc.)
- `dev`: these are classes that define an interface to a particular externally connected device (shift registers, LED drivers, etc.)
- `per`: these are classes that wrap APIs for access to the microcontrollers internal peripherals (SPI, UART, I2C, etc.)
- `ui`: these are classes to handle a user interface (Event Queues, Control Monitors, Display Management)
- `sys`: these are classes and interfaces for managing the lower levels of the hardware (initializing clock trees, DMA, linking filesystem drivers, etc.)
- `util`: classes and other structures that can be useful on their own, or used within other classes in libDaisy (CPU Load Profiling, FIFOs, diskio, Persistent Storage, etc.)

1.1 Working with Daisy

- [Setting up the development environment]()
- [Creating a new Project]()
- [Working with GPIO](#)
- [Printing to a serial console]()
- [Reading and Writing files to an SD Card or USB Drive]()
- [Using the external 64MB of SDRAM]()
- [Audio Callbacks explained]()
- [Working with the ADC Inputs]()
- [Working with MIDI]()
- [Running and writing Unit Tests](#)

1.2 Troubleshooting

Report bugs, typos, errors, etc. [[here on Github](#)]()

Chapter 2

Unit testing libDaisy

libDaisy has integrated unit tests.

- Unit tests are small tests that verify that a piece of code works as intended.
- Unit tests are compiled into a small application that runs on your development computer. They're not compiled into libDaisy.
- Unit tests are run as part of the continuous integration (CI) setup. They're build and executed with each pull request, making sure that the changes don't break existing functionality.
- Unit tests in libDaisy are based on the widely used `googletest framework`

2.1 Why write tests?

Writing a test can help you in many ways. Here are some of the many benefits:

- With unit tests in place, we can modify existing code without fear. If there's a test and its green, you can feel safe.
- Unit tests give you a great, simple environment to quickly run and test your code with a debugger. Not only does a test work independently from the hardware, it also gives you a clean, blank slate that's easy to setup so that you can quickly check out a particular scenario. Want to quickly try out that idea you had earlier? No need to setup a hardware testbed! Just add a small test and off you go!
- In writing the tests for new code, you force yourself to really think about what your code is supposed and not supposed to do. This helps you split a new functionality into small, independent parts that serve a well defined purpose. Additionally, the test itself is the best documentation. It clearly states how the code should behave.

2.2 How to run tests?

Update the git submodules so that the googletest code is checked out.

```
> git submodule update --init --recursive
```

Linux:

- Install the `build-essential` package. That's it.

macOS:

- Install Xcode and the Xcode commandline tools.

Windows:

- Install Cygwin with the packages `g++`, `gdb` and `make`. Add them to your PATH environment variable.
This guide may be helpful for you.

2.2.1 From the commandline

All tests live in the `tests/` directory and share a common makefile. To build and run the tests, execute this:

```
> cd tests/  
> make test
```

This generates and runs a small commandline program `tests/libDaisy_gtest`.

2.2.2 From Visual Studio Code

Programmers using Visual Studio Code can install the recommended extensions (you will be prompted to do so) and use the test panel to directly run the tests. Take a look at the screenshot below:

- Build tests by executing the `build-libDaisy-tests` task with `Ctrl+Shift+B` (Win) or `Cmd+Shift+B` (macOS).
- Run and debug individual tests or entire test suites from the test panel. (see screenshot)
- Launch the tests from the launch tab with the `Launch Tests Locally` launch configuration.

2.3 Getting started with unit testing

Adding a new test to libDaisy is easy.

1. Add a new *_gtest.cpp file to the tests/ directory. This is where you will write your tests. It makes sense to create individual files for each component you want to test.
2. In this file, include the googletest header `#include <gtest/gtest.h>`
3. Also include the code you want to test `#include "myHeaderFileFromLibDaisy.h"`
4. If your code also has a *.cpp file, include that in the file tests/libDaisyImpls.cpp like this: `#include "myCppClassFromLibDaisy.cpp"`
5. Add a test to your fresh new *_gtest.cpp file.

This is what a simple test could look like:

```
// tests/myTest/gtest.cpp
#include <gtest/gtest.h>
#include "myHeaderFileFromLibDaisy.h"
TEST(MyTestSuiteName, a_myFirstTest)
{
    EXPECT_EQ(2, 6 - 4);
}
TEST(MyTestSuiteName, b_mySecondTest)
{
    const bool someResult = 2 == 2;
    EXPECT_TRUE(someResult);
}
```

For more examples, check out the existing tests and take a look at the [official googletest primer](#).

2.4 Drawbacks & things to watch out for

Tests are built locally on your development computer. While that enables you to build and test without hardware, it comes with a couple of drawbacks that you should be aware of:

- Any code that uses features specific to the hardware can't be unit tested. This includes
 - Code that uses inline assembly
 - Code that relies on hardware peripherals
- The unit test makefile is setup so that the UNIT_TEST macro is defined whenever code is compiled for unit tests. You can use this to replace actual hardware dependent code with a dummy version.
 - Let's assume you want to write tests for `src/hid/led.h`.
 - This file will include `src/per/gpio.h` which obviously can't work natively on your development computer.
 - To resolve this issue, you could use the preprocessor macro `UNIT_TEST` to selectively replace the problematic gpio code with a dummy version that you can control and observe from your unit tests.

Should you face issues with native code that won't compile in unit tests, feel free to ask on the forums or in Slack! We're there to help!

Chapter 3

Working With GPIO

GPIO stands for General Purpose Input/Output. These are common to microcontrollers of all sizes, and are one of the many ways to work with external components.

GPIO are used as the basis for several interface components including switches, LEDs, encoders, etc.

The Daisy exposes several GPIO on its pinout. Most of these pins can do much more than just be a digital input or output, but we'll keep it simple for now.

Below, we'll discuss the code, hardware, and provide a few examples of how to use the GPIO within the Daisy ecosystem.

3.1 The CPP objects used

We'll be using the `DaisySeed`, `Pin`, and `GPIO` objects in the following sections.

If you're using a different Daisy SOM (i.e. `DaisyPatchSM`, etc.) this still applies, but some of the pin names will be different.

The `DaisySeed` object is a class that manages all of the hardware on the Seed board. All we need to do with it for now is initialize it.

The `Pin` class is used to describe a specific physical pin on the hardware. These objects are used to initialize GPIO, but are also used in the configuration of more complex peripherals and devices (i.e. ADCs, Shift Registers, etc.).

The `GPIO` class is used for the basic reading/writing of a digital signal.

When we talk about "digital" signals, we mean a signal that has only two states, HIGH and LOW.

3.2 The Hardware Connections

For the examples below we'll be using a few components. These can be replaced with a number of similar alternatives.

- `Mini Pushbutton`
- `3mm LED`
- Resistor(s) any low value (1-10K should work okay in the examples discussed below)

3.3 GPIO Input

One of the simplest uses of a GPIO is to read the state of a button.

For this example we'll use a simple `mini pushbutton` to read the state of a button.

The easiest way to wire up a switch like this is to connect one set of legs (pins 1 and 2) to the GPIO input, and connect the other two legs (pins 3 and 4) to GND.

A very short example of a program to do this looks like:

```
#include "daisy_seed.h"
using namespace daisy;
using namespace daisy::seed;
DaisySeed hw;
int main(void) {
    hw.Init();
    GPIO my_button;
    my_button.Init(D0, GPIO::Mode::INPUT, GPIO::Pull::PULLUP);
    while(1) {
        bool button_state = my_button.Read();
    }
}
```

If you haven't followed the guides on [setting up your development environment](#) and [creating a new project](#) now would be a perfect time to catch up, and create a new project to work with the GPIO.

Compiling this example, you might get a warning, for an "unused variable `button_state`", but we'll get to using that shortly!

3.4 Breaking Down the Init function

Let's look at that GPIO init function in a bit more detail, shall we? First, we'll breakdown the full Initialization definition:

```
void Init(Pin p, Mode m=Mode::INPUT, Pull pu=Pull::NOPULL, Speed sp=Speed::LOW);
```

The first thing you may be wondering, if you're not used to C++ is the `::` characters all over the place. These are used for the configuration settings so that the compiler will only allow you to use the correct inputs instead of being able to pass the wrong thing to the input, or even just mis-order the arguments, and end up scratching your head for hours wondering what's not working.

These `::` are also used for namespaces, which we handle in the above example with the `using namespace` lines. Without that we'd have to use `daisy::GPIO` instead of `GPIO`, `daisy::seed::D0` instead of `D0`, etc.

3.4.1 Pin Argument

So, the first argument: `Pin p` describes the daisy pin to assign to our GPIO, this ties our new GPIO object to some of the actual hardware on the Daisy Seed. By using `D0` (or more explicitly, `daisy::seed::D0`), we're saying we want to use that pin for our GPIO.

3.4.2 Mode Argument

The second argument: `Mode m` can be any of the following:

- `Mode::INPUT` - configures the pin as an input
- `Mode::OUTPUT` - configures the pin as an output (in push pull configuration)
- `Mode::OUTPUT_OD` - also an output, but the transistor connection to pull the signal to GND is not connected, this is less commonly used.
- `Mode::ANALOG` - configures the GPIO for connection to the ADC or DAC peripherals within the Microcontroller.

If you only supply the `Pin` argument, the GPIO will default to `Mode::INPUT` configuration.

3.4.3 Pull Argument

The third argument: `Pull pu` is used to select whether the GPIO will use an internal (around 30-50k) resistor as a pull up or pull down resistor.

A pull up resistor will keep the GPIO idling at 3v3 unless something pulls it down, while a pull down resistor will keep a GPIO idling at 0V unless something pulls it up.

The options for this argument are:

- `Pull::NOPULL` - no pull up or pull down resistor is connected.
- `Pull::PULLUP` - pull up resistor is connected to the GPIO line.
- `Pull::PULLDOWN` - pull down resistor is connected to the GPIO line.

This argument will default to `Pull::NOPULL` if you only supply a `Pin` and a `Mode`

3.4.4 Speed Argument

The final argument, `Speed sp` is a special configuration for Output modes that control the slew rate, or the speed at which it can switch from 0V to 3V. We won't go super in depth on that in this, but you'll always want it to be the lowest speed possible for a given application.

3.4.5 Defaults

So in our example above, we can avoid having to connect any external resistors to our button by using the pull up feature.

If we wanted to use some external resistors we could pull one side high with a resistor to 5V, and the other with a resistor to GND, and connect one side to the GPIO input, and if we didn't need to use the GPIO pull up resistor we could initialize the pin using the default settings:

```
my_button.Init(D0)
```

By specifying the input mode, and pullup configuration we can do the same thing with just a button, and no external resistors by adding the Mode and Pull arguments:

```
my_button.Init(D0, GPIO::Mode::INPUT, GPIO::Pull::PULLUP);
```

We will always have to Init the GPIO with a Pin, otherwise the program won't know what pin on the hardware we actually want to use, but the other inputs have the following defaults:

- Mode: Input
- Pull: No Pull Up
- Speed: Low

3.5 GPIO Output

So you've got your button reading code, and your board wired up, and you're probably thinking, "Great! So how can I tell if I've done this correctly!?".

Well, let's set up a second pin as an output to light up an LED when we push our button. That way we can tell everything we've done so far is working.

Setting up the GPIO output isn't much more work.

On your hardware you'll want to take the second pin (right above the square D0 pin), and connect a resistor, and an LED in series. You'll want the **cathode** of the LED connected to GND, with the **anode** connected to the resistor.

Once you've got that set up we can add a few lines to our example.

In the Initialization section:

```
GPIO my_led;
my_led.Init(D1, GPIO::Mode::OUTPUT);
```

And then we can use the `button_state` variable we were reading from the button earlier to control the state of the LED:

```
while(1)
{
    bool button_state = my_button.Read();
    my_led.Write(button_state);
}
```

And that's it! Easy, right?

So you get that all set up, and you might be thinking, "Well, this is great, but the LED turns off when I push the button instead of turning it on, what the heck?"

That's because of the way we set up the button in the first step. Because we used the internal pull up resistor, to avoiding adding extra parts to our board, the GPIO is returning true while the button is not pressed instead of what we might expect. Well, C++ has a very easy way of flipping that over.

```
bool button_state = my_button.Read();
bool button_pressed = !button_state;
```

So all together, our short little light switch program will look something like:

```
#include "daisy_seed.h"
using namespace daisy;
using namespace daisy::seed;
DaisySeed hw;
int main(void) {
    hw.Init();
    GPIO my_button;
    GPIO my_led;
    my_button.Init(D0, GPIO::Mode::INPUT, GPIO::Pull::PULLUP);
    my_led.Init(D1, GPIO::Mode::OUTPUT);
    while(1) {
        bool button_state = my_button.Read();
        my_led.Write(!button_state);
    }
}
```

3.6 Further Reading

Topics coming soon:

- Working with ADCs
- Serial Printing over USB
- Using the Switch and LED classes instead of GPIO, and Why?

Chapter 4

Todo List

Member [daisy::patch_sm::DaisyPatchSM::WriteCvOut](#) (const int channel, float voltage)

Add Calibration to CV Outputs

Class [daisy::PersistentStorage< SettingStruct >](#)

- Make [Save\(\)](#) non-blocking
- Add wear leveling

Member [daisy::VoctCalibration::Record](#) (float val1V, float val3V)

Add some sort of range validation. Originally we had a check for a valid range on the input, but given that the input circuit or the [AnalogControl](#) configuration can have a drastic effect on input, that could cause unintentional failure to calibrate, it was removed.

Chapter 5

Module Index

5.1 Modules

Here is a list of all modules:

LIBDAISY	27
HUMAN_INTERFACE	27
AUDIO	28
CONTROLS	28
FEEDBACK	29
EXTERNAL	29
PERIPHERAL	35
SERIAL	35
ANALOG_DIGITAL_CONVERSION	36
OTHER	36
SYSTEM	40
DEVICE	42
SHIFTREGISTER	42
FLASH	43
CODEC	68
LED	69
SDRAM	69
BOARDS	70
UI	70
UTILITY	73
Lcd	90
Dac	90
Externals	98

Chapter 6

Namespace Index

6.1 Namespace List

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

daisy	Hardware defines and helpers for daisy field platform	99
daisy::seed	109

Chapter 7

Hierarchical Index

7.1 Class Hierarchy

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

daisy::AdcChannelConfig	120
daisy::AdcHandle	122
daisy::Ak4556	126
daisy::AllNotesOffEvent	126
daisy::AllSoundOffEvent	127
daisy::AnalogControl	128
daisy::AudioHandle	131
daisy::BlockingSpiTransport	135
daisy::ButtonMonitor< BackendType, numButtons >	135
daisy::ChannelPressureEvent	137
daisy::Color	138
daisy::AudioHandle::Config	140
daisy::BlockingSpiTransport::Config	140
daisy::DacHandle::Config	141
daisy::FatFSInterface::Config	142
daisy::GPIO::Config	142
daisy::I2CHandle::Config	143
daisy::LcdHD44780::Config	146
daisy::MAX11300Driver< Transport >::Config	146
daisy::MidiHandler< Transport >::Config	146
daisy::MidiUartTransport::Config	147
daisy::MidiUsbTransport::Config	147
daisy::OledDisplay< DisplayDriver >::Config	147
daisy::QSPIHandle::Config	147
daisy::SaiHandle::Config	151
daisy::SdmmcHandler::Config	152
daisy::ShiftRegister4021< num_daisychained, num_parallel >::Config	153
daisy::SpiHandle::Config	154
daisy::SSD130x4WireSpiTransport::Config	156
daisy::SSD130xDriver< width, height, Transport >::Config	156
daisy::SSD130xl2CTransport::Config	157
daisy::System::Config	157
daisy::TimerHandle::Config	158
daisy::UartHandler::Config	159
daisy::USBHostHandle::Config	160

daisy::WavWriter< transfer_size >::Config	160
daisy::Wm8731::Config	161
daisy::ControlChangeEvent	163
daisy::CpuLoadMeter	164
daisy::AbstractMenu::CustomItem	165
daisy::DacHandle	167
daisy::DaisyField	170
daisy::DaisyPatch	179
daisy::patch_sm::DaisyPatchSM	185
daisy::DaisyPetal	193
daisy::DaisyPod	204
daisy::DaisySeed	210
daisy::DaisyVersio	217
dsy_gpio	221
dsy_gpio_pin	222
DSY_SD_CardInfoTypeDef	223
daisy::Encoder	225
daisy::FatFSInterface	227
daisy::FIFOBase< T >	231
daisy::FIFO< Event, 256 >	230
daisy::FIFO< T, capacity >	230
daisy::FixedCapStrBase< CharType >	236
daisy::FixedCapStrBase< char >	236
daisy::FixedCapStr< capacity, CharType >	236
FontDef	237
daisy::GateIn	240
daisy::GPIO	242
daisy::I2CHandle	244
daisy::AbstractMenu::ItemConfig	249
daisy::LcdHD44780	251
daisy::Led	253
daisy::LedDriverPca9685< numDrivers, persistentBufferContents >	254
daisy::LedDriverPca9685< 2, true >	254
daisy::LocalControlEvent	258
daisy::Logger< dest >	259
daisy::Logger< LOGGER_NONE >	260
daisy::LoggerImpl< dest >	261
daisy::LoggerImpl< LOGGER_EXTERNAL >	262
daisy::LoggerImpl< LOGGER_INTERNAL >	263
daisy::LoggerImpl< LOGGER_SEMIHOST >	264
daisy::MappedValue	275
daisy::MappedFloatValue	265
daisy::MappedIntValue	268
daisy::MappedStringListValue	271
daisy::MAX11300Driver< Transport >	276
daisy::MidiEvent	278
daisy::MidiHandler< Transport >	280
daisy::MidiHandler< MidiUartTransport >	280
daisy::MidiUartTransport	283
daisy::MidiUsbTransport	283
daisy::MonoModeOnEvent	283
daisy::MTCQuarterFrameEvent	284
daisy::NoteOffEvent	285
daisy::NoteOnEvent	286
daisy::OmniModeOffEvent	289
daisy::OmniModeOnEvent	289
daisy::OneBitGraphicsDisplay	290

daisy::OneBitGraphicsDisplayImpl< OledDisplay< DisplayDriver > >	296
daisy::OledDisplay< DisplayDriver >	287
daisy::OneBitGraphicsDisplayImpl< OledDisplay< daisy::SSD130xDriver > >	296
daisy::OledDisplay< daisy::SSD130xDriver >	287
daisy::OneBitGraphicsDisplayImpl< ChildType >	296
daisy::Parameter	300
daisy::Pcm3060	302
daisy::PersistentStorage< SettingStruct >	303
daisy::Pin	305
daisy::PitchBendEvent	307
daisy::PolyModeOnEvent	308
daisy::PolyphonicKeyPressureEvent	308
daisy::PotMonitor< BackendType, numPots >	309
daisy::ProgramChangeEvent	311
daisy::QSPIHandle	312
daisy::Random	316
daisy::Rectangle	318
daisy::ResetAllControllersEvent	319
daisy::RgbLed	319
daisy::RingBuffer< T, size >	322
daisy::RingBuffer< daisy::MidiEvent, 256 >	322
daisy::RingBuffer< T, 0 >	327
daisy::SaiHandle	330
daisy::ScopdedIrqBlocker	333
daisy::SdmmcHandler	334
SdramHandle	335
daisy::ShiftRegister4021< num_daisychained, num_parallel >	336
daisy::ShiftRegister4021< 2 >	336
ShiftRegister595	337
daisy::SongPositionPointerEvent	339
daisy::SongSelectEvent	340
daisy::UI::SpecialControlIds	340
daisy::SpiHandle	342
daisy::SSD130x4WireSpiTransport	346
daisy::SSD130xDriver< width, height, Transport >	346
daisy::SSD130xl2CTransport	347
daisy::StackBase< T >	349
daisy::Stack< daisy::UiPage *, kMaxNumPages >	347
daisy::Stack< daisy::UiCanvasDescriptor, kMaxNumCanvases >	347
daisy::Stack< T, capacity >	347
daisy::Switch	352
daisy::Switch3	357
daisy::System	357
daisy::SystemExclusiveEvent	362
daisy::TimerHandle	362
daisy::UartHandler	366
daisy::UI	370
daisy::UiCanvasDescriptor	372
daisy::UiEventQueue	373
daisy::UiPage	376
daisy::AbstractMenu	113
daisy::FullScreenItemMenu	239
UsbHandle	386
daisy::USBHostHandle	391
daisy::VocCalibration	393
daisy::WAV_FormatTypeDef	395
daisy::WaveTableLoader	397

daisy::WavFileInfo	398
daisy::WavPlayer	399
daisy::WavWriter< transfer_size >	401
daisy::Wm8731	405

Chapter 8

Class Index

8.1 Class List

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

daisy::AbstractMenu	113
daisy::AdcChannelConfig	120
daisy::AdcHandle	122
daisy::Ak4556	126
daisy::AllNotesOffEvent	126
daisy::AllSoundOffEvent	127
daisy::AnalogControl Hardware Interface for control inputs Primarily designed for ADC input controls such as potentiometers, and control voltage. 128	
daisy::AudioHandle	131
daisy::BlockingSpiTransport	135
daisy::ButtonMonitor< BackendType, numButtons >	135
daisy::ChannelPressureEvent	137
daisy::Color	138
daisy::AudioHandle::Config	140
daisy::BlockingSpiTransport::Config	140
daisy::DacHandle::Config	141
daisy::FatFSInterface::Config	142
daisy::GPIO::Config Configuration for a given GPIO	142
daisy::I2CHandle::Config	143
daisy::LcdHD44780::Config	146
daisy::MAX11300Driver< Transport >::Config	146
daisy::MidiHandler< Transport >::Config	146
daisy::MidiUartTransport::Config	147
daisy::MidiUsbTransport::Config	147
daisy::OledDisplay< DisplayDriver >::Config	147
daisy::QSPIHandle::Config	147
daisy::SaiHandle::Config	151
daisy::SdmmcHandler::Config	152
daisy::ShiftRegister4021< num_daisychained, num_parallel >::Config	153
daisy::SpiHandle::Config	154
daisy::SSD130x4WireSpiTransport::Config	156

daisy::SSD130xDriver< width, height, Transport >::Config	156
daisy::SSD130xI2CTransport::Config	157
daisy::System::Config	157
daisy::TimerHandle::Config	158
daisy::UartHandler::Config	159
daisy::USBHostHandle::Config	160
daisy::WavWriter< transfer_size >::Config	160
daisy::Wm8731::Config	161
daisy::ControlChangeEvent	163
daisy::CpuLoadMeter	164
daisy::AbstractMenu::CustomItem	165
daisy::DacHandle	167
daisy::DaisyField	170
daisy::DaisyPatch	
Class that handles initializing all of the hardware specific to the Daisy Patch Board.	
Helper funtions are also in place to provide easy access to built-in controls and peripherals	179
daisy::patch_sm::DaisyPatchSM	
Board support file for DaisyPatchSM hardware	185
daisy::DaisyPetal	
Helpers and hardware definitions for daisy petal	193
daisy::DaisyPod	
Class that handles initializing all of the hardware specific to the Daisy Patch Board.	
Helper funtions are also in place to provide easy access to built-in controls and peripherals	204
daisy::DaisySeed	
This is the higher-level interface for the Daisy board.	
All basic peripheral configuration/initialization is setup here.	
210	
daisy::DaisyVersio	
Class that handles initializing all of the hardware specific to the Desmodus Versio hardware.	
Helper funtions are also in place to provide easy access to built-in controls and peripherals	217
dsy_gpio	221
dsy_gpio_pin	222
DSY_SD_CardInfoTypeDef	223
daisy::Encoder	
Generic Class for handling Quadrature Encoders	
Inspired/influenced by Mutable Instruments (pichenettes) Encoder classes	225
daisy::FatFSInterface	
Daisy FatFS Driver Interface	227
daisy::FIFO< T, capacity >	230
daisy::FIFOBase< T >	231
daisy::FixedCapStr< capacity, CharType >	236
daisy::FixedCapStrBase< CharType >	236
FontDef	237
daisy::FullScreenItemMenu	239
daisy::GateIn	
Generic Class for handling gate inputs through GPIO	240
daisy::GPIO	
General Purpose I/O control	242
daisy::I2CHandle	244
daisy::AbstractMenu::ItemConfig	249
daisy::LcdHD44780	251
daisy::Led	
LED Class providing simple Software PWM ability, etc	
Eventually this will work with hardware PWM, and external LED Driver devices as well	253
daisy::LedDriverPca9685< numDrivers, persistentBufferContents >	254
daisy::LocalControlEvent	258
daisy::Logger< dest >	
Interface for simple USB logging	259

daisy::Logger< LOGGER_NONE >	260
daisy::LoggerImpl< dest >	
Logging I/O underlying implementation	261
daisy::LoggerImpl< LOGGER_EXTERNAL >	
Specialization for external USB port	262
daisy::LoggerImpl< LOGGER_INTERNAL >	
Specialization for internal USB port	263
daisy::LoggerImpl< LOGGER_SEMIHOST >	
Specialization for semihosting (stdout)	264
daisy::MappedFloatValue	265
daisy::MappedIntValue	268
daisy::MappedStringListValue	271
daisy::MappedValue	275
daisy::MAX11300Driver< Transport >	
Device Driver for the MAX11300 20 port ADC/DAC/GPIO device	276
daisy::MidiEvent	278
daisy::MidiHandler< Transport >	
Simple MIDI Handler	
Parses bytes from an input into valid MidiEvents.	
The MidiEvents fill a FIFO queue that the user can pop messages from	280
daisy::MidiUartTransport	283
daisy::MidiUsbTransport	283
daisy::MonoModeOnEvent	283
daisy::MTCQuarterFrameEvent	284
daisy::NoteOffEvent	285
daisy::NoteOnEvent	286
daisy::OledDisplay< DisplayDriver >	287
daisy::OmniModeOffEvent	289
daisy::OmniModeOnEvent	289
daisy::OneBitGraphicsDisplay	290
daisy::OneBitGraphicsDisplayImpl< ChildType >	296
daisy::Parameter	300
daisy::Pcm3060	302
daisy::PersistentStorage< SettingStruct >	
Non Volatile storage class for persistent settings on an external flash device	303
daisy::Pin	
Representation of hardware port/pin combination	305
daisy::PitchBendEvent	307
daisy::PolyModeOnEvent	308
daisy::PolyphonicKeyPressureEvent	308
daisy::PotMonitor< BackendType, numPots >	309
daisy::ProgramChangeEvent	311
daisy::QSPIHandle	312
daisy::Random	
True Random Number Generator access	316
daisy::Rectangle	318
daisy::ResetAllControllersEvent	319
daisy::RgbLed	319
daisy::RingBuffer< T, size >	322
daisy::RingBuffer< T, 0 >	327
daisy::SaiHandle	330
daisy::ScopedIrqBlocker	333
daisy::SdmmcHandler	334
SdramHandle	335
daisy::ShiftRegister4021< num_daisychained, num_parallel >	336
ShiftRegister595	
Device Driver for 8-bit shift register.	
CD74HC595 - 8-bit serial to parallel output shift	337

daisy::SongPositionPointerEvent	339
daisy::SongSelectEvent	340
daisy::UI::SpecialControlIds	340
daisy::SpiHandle	342
daisy::SSD130x4WireSpiTransport	346
daisy::SSD130xDriver< width, height, Transport >	346
daisy::SSD130xI2CTransport	347
daisy::Stack< T, capacity >	347
daisy::StackBase< T >	349
daisy::Switch	352
daisy::Switch3	357
daisy::System	357
daisy::SystemExclusiveEvent	362
daisy::TimerHandle	362
daisy::UartHandler	366
daisy::UI	370
daisy::UiCanvasDescriptor	372
daisy::UiEventQueue	373
daisy::UiPage	376
UsbHandle Interface for initializing and using the USB Peripherals on the daisy	386
daisy::USBHostHandle Presents a USB Mass Storage Device host interface	391
daisy::VoctCalibration Helper class for calibrating an input to 1V/oct response	393
daisy::WAV_FormatTypeDef	395
daisy::WaveTableLoader	397
daisy::WavFileInfo	398
daisy::WavPlayer	399
daisy::WavWriter< transfer_size >	401
daisy::Wm8731	405

Chapter 9

File Index

9.1 File List

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

src/ daisy.h	??
src/ daisy_core.h	??
src/ daisy_field.h	??
src/ daisy_patch.h	??
src/ daisy_patch_sm.h	??
src/ daisy_petal.h	??
src/ daisy_pod.h	??
src/ daisy_seed.h	??
src/ daisy_versio.h	??
src/ version.h	??
src/dev/ codec_ak4556.h	??
src/dev/ codec_pcm3060.h	??
src/dev/ codec_wm8731.h	??
src/dev/ flash_IS25LP064A.h	??
src/dev/ flash_IS25LP080D.h	??
src/dev/ lcd_hd44780.h	??
src/dev/ leddriver.h	??
src/dev/ max11300.h	??
src/dev/ oled_ssdl30x.h	??
src/dev/ sdram.h	??
src/dev/ sr_4021.h	??
src/dev/ sr_595.h	??
src/hid/ audio.h	??
src/hid/ ctrl.h	??
src/hid/ encoder.h	??
src/hid/ gatein.h	??
src/hid/ led.h	??
src/hid/ logger.h	??
src/hid/ logger_impl.h	??
src/hid/ midi.h	??
src/hid/ MidiEvent.h	??
src/hid/ parameter.h	??
src/hid/ rgb_led.h	??
src/hid/ switch.h	??
src/hid/ switch3.h	??

src/hid/ usb.h	??
src/hid/ usb_host.h	??
src/hid/ usb_midi.h	??
src/hid/ wavplayer.h	??
src/hid/disp/ display.h	??
src/hid/disp/ graphics_common.h	??
src/hid/disp/ oled_display.h	??
src/per/ adc.h	??
src/per/ dac.h	??
src/per/ gpio.h	??
src/per/ i2c.h	??
src/per/ qspi.h	??
src/per/ rng.h	??
src/per/ sai.h	??
src/per/ sdmmc.h	??
src/per/ spi.h	??
src/per/ tim.h	??
src/per/ uart.h	??
src/sys/ dma.h	??
src/sys/ fatfs.h	??
src/sys/ ffconf.h	407
src/sys/ stm32h7xx_hal_conf.h	??
src/sys/ system.h	??
src/ui/ AbstractMenu.h	??
src/ui/ ButtonMonitor.h	??
src/ui/ FullScreenItemMenu.h	??
src/ui/ PotMonitor.h	??
src/ui/ UI.h	??
src/ui/ UiEventQueue.h	??
src/util/ bsp_sd_diskio.h	??
src/util/ color.h	??
src/util/ CpuLoadMeter.h	??
src/util/ FIFO.h	??
src/util/ FixedCapStr.h	??
src/util/ hal_map.h	??
src/util/ MappedValue.h	??
src/util/ oled_fonts.h	??
src/util/ PersistentStorage.h	??
src/util/ ringbuffer.h	??
src/util/ scopedirqblocker.h	??
src/util/ sd_diskio.h	??
src/util/ Stack.h	??
src/util/ unique_id.h	??
src/util/ usbh_diskio.h	414
Header for usbh_diskio.c module	414
src/util/ VoctCalibration.h	??
src/util/ wav_format.h	??
src/util/ WaveTableLoader.h	??
src/util/ WavWriter.h	??

Chapter 10

Module Documentation

10.1 LIBDAISY

The daisy library.

Modules

- [HUMAN_INTERFACE](#)
Interface with the world.
- [PERIPHERAL](#)
Peripheral devices, not meant for human interaction.
- [SYSTEM](#)
Deals with system. DMA, clocks, etc.
- [DEVICE](#)
Low level devices. Led drivers, codecs, etc.
- [BOARDS](#)
Daisy devices. Pod, seed, etc.
- [UI](#)
UI system. UI event queue, event readers, etc.
- [UTILITY](#)
General utilities. Ringbuffers, LED colors, OLED stuff, etc.

10.1.1 Detailed Description

The daisy library.

10.2 HUMAN_INTERFACE

Interface with the world.

Modules

- [AUDIO](#)
Embedded Audio Engine.
- [CONTROLS](#)
Hardware Controls.
- [FEEDBACK](#)
Screens, leds, etc.
- [EXTERNAL](#)
External interface devices.

10.2.1 Detailed Description

Interface with the world.

10.3 AUDIO

Embedded Audio Engine.

Embedded Audio Engine.

10.4 CONTROLS

Hardware Controls.

Classes

- class [daisy::AnalogControl](#)
*Hardware Interface for control inputs
Primarily designed for ADC input controls such as
potentiometers, and control voltage.*
- class [daisy::Encoder](#)
*Generic Class for handling Quadrature Encoders
Inspired/influenced by Mutable Instruments (*pichenettes*) [Encoder](#) classes.*
- class [daisy::GateIn](#)
Generic Class for handling gate inputs through [GPIO](#).
- class [daisy::Parameter](#)
- class [daisy::Switch](#)

10.4.1 Detailed Description

Hardware Controls.

10.5 FEEDBACK

Screens, leds, etc.

Classes

- class [daisy::Led](#)
LED Class providing simple Software PWM ability, etc
Eventually this will work with hardware PWM, and external LED Driver devices as well.
- class [daisy::RgbLed](#)

10.5.1 Detailed Description

Screens, leds, etc.

10.6 EXTERNAL

External interface devices.

Classes

- class [daisy::Logger< dest >](#)
Interface for simple USB logging.
- class [daisy::Logger< LOGGER_NONE >](#)
- class [daisy::MidiUartTransport](#)
- class [daisy::MidiHandler< Transport >](#)
Simple MIDI Handler
Parses bytes from an input into valid MidiEvents.
The MidiEvents fill a [FIFO](#) queue that the user can pop messages from.

Macros

- #define [LOGGER_NEWLINE](#) "\r\n"
- #define [LOGGER_BUFFER](#) 128

Enumerations

- enum [daisy::Logger< dest >::LoggerConsts](#) { [LOGGER_SYNC_OUT](#) = 0 , [daisy::Logger< dest >::LOGGER_SYNC_IN](#) = 2 }

Functions

- `daisy::Logger< dest >::Logger ()`
- static void `daisy::Logger< dest >::Print (const char *format,...)`
- static void `daisy::Logger< dest >::PrintLine (const char *format,...)`
- static void `daisy::Logger< dest >::StartLog (bool wait_for_pc=false)`
- static void `daisy::Logger< dest >::PrintV (const char *format, va_list va)`
- static void `daisy::Logger< dest >::PrintLineV (const char *format, va_list va)`
- static void `daisy::Logger< dest >::TransmitSync (const void *buffer, size_t bytes)`
- static void `daisy::Logger< dest >::TransmitBuf ()`
- static void `daisy::Logger< dest >::AppendNewLine ()`
- static constexpr size_t `daisy::Logger< dest >::NewLineSeqLength ()`
- static void `daisy::Logger< LOGGER_NONE >::Print (const char *format,...)`
- static void `daisy::Logger< LOGGER_NONE >::PrintLine (const char *format,...)`
- static void `daisy::Logger< LOGGER_NONE >::StartLog (bool wait_for_pc=false)`
- static void `daisy::Logger< LOGGER_NONE >::PrintV (const char *format, va_list va)`
- static void `daisy::Logger< LOGGER_NONE >::PrintLineV (const char *format, va_list va)`

- static char `daisy::Logger< dest >::tx_buff_ [128]`
- static size_t `daisy::Logger< dest >::tx_ptr_ = 0`
- static size_t `daisy::Logger< dest >::pc_sync_ = LOGGER_SYNC_OUT`
- static `LoggerImpl< dest > daisy::Logger< dest >::impl_`
- #define `PPCAT_NX(A, B) A##B`
- #define `PPCAT(A, B) PPCAT_NX(A, B)`
- #define `STRINGIZE_NX(A) #A`
- #define `STRINGIZE(A) STRINGIZE_NX(A)`
- #define `FLT_FMT(_n) STRINGIZE(PPCAT(PPCAT(%c%d.%0, _n), d))`
- #define `FLT_VAR(_n, _x)`
- #define `FLT_FMT3 FLT_FMT(3)`
- #define `FLT_VAR3(_x) FLT_VAR(3, _x)`

10.6.1 Detailed Description

External interface devices.

10.6.2 Macro Definition Documentation

10.6.2.1 FLT_FMT

```
#define FLT_FMT(
    _n ) STRINGIZE(PPCAT(PPCAT(%c%d.%0, _n), d))
```

Floating point output formatting string. Include in your printf-style format string example: `printf("float value = "%
FLT_FMT(3) " continue like that", FLT_VAR(3, x));`

10.6.2.2 FLT_FMT3

```
#define FLT_FMT3 FLT_FMT(3)
```

Shorthand for 10^{-3} fraction, output equivalent to %.3f

10.6.2.3 FLT_VAR

```
#define FLT_VAR(
    _n,
    _x )
```

Value:

```
(_x < 0 ? '-' : '+'), (int)(abs(_x)), \
(int)((abs(_x)) - (int)(abs(_x))) * pow(10, (_n)))
```

Floating point output variable preprocessing Note: uses truncation instead of rounding -> the last digit may be off

10.6.2.4 FLT_VAR3

```
#define FLT_VAR3 (
    _x ) FLT_VAR(3, _x)
```

Shorthand for 10^{-3} fraction

10.6.2.5 LOGGER_BUFFER

```
#define LOGGER_BUFFER 128
```

size in bytes

10.6.2.6 LOGGER_NEWLINE

```
#define LOGGER_NEWLINE "\r\n"
```

Logger configuration custom newline character sequence

10.6.2.7 PPCAT

```
#define PPCAT(
    A,
    B ) PPCAT_NX(A, B)
```

concatenate tokens

10.6.2.8 PPCAT_NX

```
#define PPCAT_NX(
    A,
    B ) A##B
```

Helper macros for string concatenation and macro expansion non-expanding concatenation

10.6.2.9 STRINGIZE

```
#define STRINGIZE(
    A ) STRINGIZE_NX(A)
```

make a string

10.6.2.10 STRINGIZE_NX

```
#define STRINGIZE_NX(
    A ) #A
```

non-expanding stringize

10.6.3 Enumeration Type Documentation

10.6.3.1 LoggerConsts

```
template<LoggerDestination dest = LOGGER_INTERNAL>
enum daisy::Logger::LoggerConsts [protected]
```

Internal constants

Enumerator

LOGGER_SYNC_IN	successfully transmit this many packets before switching to blocking transfers
----------------	--

10.6.4 Function Documentation

10.6.4.1 AppendNewLine()

```
template<LoggerDestination dest = LOGGER_INTERNAL>
static void daisy::Logger< dest >::AppendNewLine( ) [static], [protected]
```

Trim control characters and append clean newline sequence, if there's room in the buffer

10.6.4.2 `Logger()`

```
template<LoggerDestination dest = LOGGER_INTERNAL>
daisy::Logger< dest >::Logger ( ) [inline]
```

Object constructor

10.6.4.3 `NewLineSeqLength()`

```
template<LoggerDestination dest = LOGGER_INTERNAL>
static constexpr size_t daisy::Logger< dest >::NewLineSeqLength ( ) [inline], [static], [constexpr], [protected]
```

Constexpr function equivalent of `strlen(LOGGER_NEWLINE) < custom newline character sequence`

10.6.4.4 `Print()`

```
template<LoggerDestination dest = LOGGER_INTERNAL>
static void daisy::Logger< dest >::Print (
    const char * format,
    ... ) [static]
```

Print formatted string

10.6.4.5 `PrintLine()`

```
template<LoggerDestination dest = LOGGER_INTERNAL>
static void daisy::Logger< dest >::PrintLine (
    const char * format,
    ... ) [static]
```

Print formatted string appending line termination sequence

10.6.4.6 `PrintLineV()`

```
template<LoggerDestination dest = LOGGER_INTERNAL>
static void daisy::Logger< dest >::PrintLineV (
    const char * format,
    va_list va ) [static]
```

Variadic argument variant of `PrintLine()`

10.6.4.7 `PrintV()`

```
template<LoggerDestination dest = LOGGER_INTERNAL>
static void daisy::Logger< dest >::PrintV (
    const char * format,
    va_list va ) [static]
```

Variadic argument variant of `Print()`

10.6.4.8 `StartLog()`

```
template<LoggerDestination dest = LOGGER_INTERNAL>
static void daisy::Logger< dest >::StartLog (
    bool wait_for_pc = false ) [static]
```

Start the logging session.

Parameters

<code>wait_for_pc</code>	block until remote terminal is ready
--------------------------	--------------------------------------

10.6.4.9 `TransmitBuf()`

```
template<LoggerDestination dest = LOGGER_INTERNAL>
static void daisy::Logger< dest >::TransmitBuf ( ) [static], [protected]
```

Transfer accumulated data

10.6.4.10 `TransmitSync()`

```
template<LoggerDestination dest = LOGGER_INTERNAL>
static void daisy::Logger< dest >::TransmitSync (
    const void * buffer,
    size_t bytes ) [inline], [static], [protected]
```

Blocking wrapper for `Transmit()`

10.6.5 Variable Documentation**10.6.5.1 `impl_`**

```
template<LoggerDestination dest>
LoggerImpl< dest > daisy::Logger< dest >::impl_ [static], [protected]
```

underlying trasnfer implementation

10.6.5.2 `pc_sync_`

```
template<LoggerDestination dest>
size_t daisy::Logger< dest >::pc_sync_ = LOGGER_SYNC_OUT [static], [protected]
```

terminal synchronization state

start with non-blocking transfers to support startup-time printouts

10.6.5.3 tx_buff_

```
template<LoggerDestination dest>
char daisy::Logger< dest >::tx_buff_ [static], [protected]
```

member variables buffer for log data

member variable definition (could switch to inline statics in C++17) this needs to remain in SRAM to support startup-time printouts

10.6.5.4 tx_ptr_

```
template<LoggerDestination dest>
size_t daisy::Logger< dest >::tx_ptr_ = 0 [static], [protected]
```

current position in the buffer

10.7 PERIPHERAL

Peripheral devices, not meant for human interaction.

Modules

- [SERIAL](#)
Serial Communications.
- [ANALOG_DIGITAL_CONVERSION](#)
Convert from digital to analog, or vice-versa.
- [OTHER](#)
GPIO, timers, and SDMMC.

10.7.1 Detailed Description

Peripheral devices, not meant for human interaction.

10.8 SERIAL

Serial Communications.

Classes

- class [daisy::QSPIHandle](#)
- class [daisy::SpiHandle](#)
- class [daisy::UartHandler](#)

Functions

- void [daisy::dsy_spi_global_init \(\)](#)

10.8.1 Detailed Description

Serial Communications.

10.8.2 Function Documentation

10.8.2.1 [dsy_spi_global_init\(\)](#)

```
void daisy::dsy_spi_global_init ( )
```

internal. Used for global init.

10.9 ANALOG_DIGITAL_CONVERSION

Convert from digital to analog, or vice-versa.

Classes

- struct [daisy::AdcChannelConfig](#)
- class [daisy::AdcHandle](#)

10.9.1 Detailed Description

Convert from digital to analog, or vice-versa.

10.10 OTHER

[GPIO](#), timers, and SDMMC.

Classes

- struct [dsy_gpio](#)
- class [daisy::SdmmcHandler](#)

Enumerations

- enum `dsy_gpio_mode` {
 `DSY_GPIO_MODE_INPUT` , `DSY_GPIO_MODE_OUTPUT_PP` , `DSY_GPIO_MODE_OUTPUT_OD` ,
 `DSY_GPIO_MODE_ANALOG` ,
 `DSY_GPIO_MODE_LAST` }
- enum `dsy_gpio_pull` { `DSY_GPIO_NOPULL` , `DSY_GPIO_PULLUP` , `DSY_GPIO_PULLDOWN` }

Functions

- void `dsy_gpio_init` (const `dsy_gpio` *p)
- void `dsy_gpio_deinit` (const `dsy_gpio` *p)
- `uint8_t dsy_gpio_read` (const `dsy_gpio` *p)
- void `dsy_gpio_write` (const `dsy_gpio` *p, `uint8_t state`)
- void `dsy_gpio_toggle` (const `dsy_gpio` *p)

10.10.1 Detailed Description

`GPIO`, timers, and SDMMC.

Old Style C API for `GPIO` is staying in place for a few versions to support backwards compatibility.

This should not be used for anything new. General Purpose IO driver

10.10.2 Enumeration Type Documentation

10.10.2.1 `dsy_gpio_mode`

enum `dsy_gpio_mode`

Sets the mode of the GPIO

Enumerator

<code>DSY_GPIO_MODE_INPUT</code>	&
<code>DSY_GPIO_MODE_OUTPUT_PP</code>	Push-Pull
<code>DSY_GPIO_MODE_OUTPUT_OD</code>	Open-Drain
<code>DSY_GPIO_MODE_ANALOG</code>	&
<code>DSY_GPIO_MODE_LAST</code>	&

10.10.2.2 `dsy_gpio_pull`

enum `dsy_gpio_pull`

Configures whether an internal Pull up or Pull down resistor is used

Enumerator

DSY_GPIO_NOPULL	&
DSY_GPIO_PULLUP	&
DSY_GPIO_PULLDOWN	&

10.10.3 Function Documentation

10.10.3.1 dsy_gpio_deinit()

```
void dsy_gpio_deinit (
    const dsy_gpio * p )
```

Deinitializes the gpio pin

Parameters

*p	Pin pointer
----	-------------

10.10.3.2 dsy_gpio_init()

```
void dsy_gpio_init (
    const dsy_gpio * p )
```

Initializes the gpio with the settings configured.

Parameters

*p	Pin pointer
----	-------------

10.10.3.3 dsy_gpio_read()

```
uint8_t dsy_gpio_read (
    const dsy_gpio * p )
```

Reads the state of the gpio pin

Parameters

*p	Pin pointer
----	-------------

Returns

1 if the pin is HIGH, and 0 if the pin is LOW

10.10.3.4 `dsy_gpio_toggle()`

```
void dsy_gpio_toggle (
    const dsy_gpio * p )
```

Toggles the state of the pin so that it is not at the same state as it was previously.

Parameters

<code>*p</code>	Pin pointer
-----------------	-------------

10.10.3.5 `dsy_gpio_write()`

```
void dsy_gpio_write (
    const dsy_gpio * p,
    uint8_t state )
```

Writes the state to the gpio pin Pin will be set to 3v3 when state is 1, and 0V when state is 0

Parameters

<code>*p</code>	Pin pointer
<code>state</code>	State to write

10.11 SYSTEM

Deals with system. DMA, clocks, etc.

Functions

- void `dsy_dma_init` (void)
- void `dsy_dma_deinit` (void)
- void `dsy_dma_clear_cache_for_buffer` (uint8_t *buffer, size_t size)
- void `dsy_dma_invalidate_cache_for_buffer` (uint8_t *buffer, size_t size)

10.11.1 Detailed Description

Deals with system. DMA, clocks, etc.

A handle for interacting with the Core [System](#). This includes the Clock tree, MPU, global DMA initialization, cache handling, and any other necessary global initialization

Author

shensley

10.11.2 Function Documentation

10.11.2.1 `dsy_dma_clear_cache_for_buffer()`

```
void dsy_dma_clear_cache_for_buffer (
    uint8_t * buffer,
    size_t size )
```

DMA transfers require the buffers to be excluded from the cache because the DMA reads / writes directly to the SRAM whereas the processor itself accesses the cache. Otherwise the DMA will access whatever is in the SRAM at the time which may not be in sync with the data in the cache - resulting in wrong data transmitted / received. You can place buffers in the D2 memory domain, in a section that has the cache disabled like this: `uint8_t DMA←_BUFFER_MEM_SECTION my_buffer[100];` If this is not possible for some reason, call this function to clear the cache (write cache contents to SRAM if required) before starting to transmit data via the DMA.

10.11.2.2 `dsy_dma_deinit()`

```
void dsy_dma_deinit (
    void )
```

Deinitializes the DMA (specifically for the modules used within the library)

10.11.2.3 `dsy_dma_init()`

```
void dsy_dma_init (
    void )
```

Initializes the Direct Memory Access Peripheral used by many internal elements of libdaisy. Initializes the DMA (specifically for the modules used within the library)

10.11.2.4 `dsy_dma_invalidate_cache_for_buffer()`

```
void dsy_dma_invalidate_cache_for_buffer (
    uint8_t * buffer,
    size_t size )
```

DMA transfers require the buffers to be excluded from the cache because the DMA reads / writes directly to the SRAM whereas the processor itself accesses the cache. Otherwise the DMA will access whatever is in the SRAM at the time which may not be in sync with the data in the cache - resulting in wrong data transmitted / received. You can place buffers in the D2 memory domain, in a section that has the cache disabled like this: `uint8_t DMA←_BUFFER_MEM_SECTION my_buffer[100];` If this is not possible for some reason, call this function to invalidate the cache (read SRAM contents to cache if required) after reading data from peripherals via the DMA.

10.12 DEVICE

Low level devices. Led drivers, codecs, etc.

Modules

- [SHIFTREGISTER](#)
Digital shift registers.
- [FLASH](#)
Flash memory.
- [CODEC](#)
Audio codecs.
- [LED](#)
LED driver devices.
- [SDRAM](#)
SDRAM devices.

10.12.1 Detailed Description

Low level devices. Led drivers, codecs, etc.

10.13 SHIFTREGISTER

Digital shift registers.

Classes

- class [ShiftRegister595](#)
*Device Driver for 8-bit shift register.
CD74HC595 - 8-bit serial to parallel output shift.*

10.13.1 Detailed Description

Digital shift registers.

Device Driver for CD4021 shift register.

Author

shensley

CD4021B-Q1: CMOS 8-STAGE STATIC SHIFT REGISTER

Supply Voltage: 3V to 18V Clock Freq: 3MHz at 5V (less at 3v3) -> 8.5MHz at 15V [Pin Descriptions](#):

- Parallel Data[1-8] - 7, 6, 5, 4, 13, 14, 115, 1
- Serial Data - 11
- Clock - 10
- P/S - 9
- Q[6-8] - 2, 12, 3

Driver has support for daisy chaining and running up to 2 same-sized chains in parallel from a single set of clk/latch pins to reduce pin/code overhead when using multiple devices.

When dealing with multiple parallel/daisy-chained devices the states of all inputs will be filled in the following order (example uses two chained and two parallel): data[chain0,parallel0], data[chain1,parallel0], data[chain0,parallel1], data[chain1,parallel1];

When combining multiple daisy chained and parallel devices the number of devices chained should match for each parallel device chain.

10.14 FLASH

Flash memory.

Macros

- #define ENTER_DEEP_POWER_DOWN 0XB9
- #define EXIT_DEEP_POWER_DOWN 0XB9
- #define RESET_ENABLE_CMD 0x66
- #define RESET_MEMORY_CMD 0x99
- #define READ_ID_CMD 0xAB
- #define READ_ID_CMD2 0x9F
- #define MULTIPLE_IO_READ_ID_CMD 0xAF
- #define READ_SERIAL_FLASH_DISCO_PARAM_CMD 0x5A
- #define READ_MANUFACT_AND_ID 0x90
- #define READ_UNIQUE_ID 0x4B
- #define NO_OP 0x00
- #define SECTOR_UNLOCK 0x26
- #define SECTOR_LOCK 0x24
- #define INFO_ROW_ERASE_CMD 0x64
- #define INFO_ROW_PROGRAM_CMD 0x62
- #define INFO_ROW_READ_CMD 0x68
- #define READ_CMD 0x03
- #define FAST_READ_CMD 0x0B
- #define FAST_READ_DTR_CMD 0x0D

- #define DUAL_OUT_FAST_READ_CMD 0x3B
- #define DUAL_INOUT_FAST_READ_CMD 0xBB
- #define DUAL_INOUT_FAST_READ_DTR_CMD 0xBD
- #define QUAD_OUT_FAST_READ_CMD 0x6B
- #define QUAD_INOUT_FAST_READ_CMD 0xEB
- #define QUAD_INOUT_FAST_READ_DTR_CMD 0xED
- #define WRITE_ENABLE_CMD 0x06
- #define WRITE_DISABLE_CMD 0x04
- #define READ_STATUS_REG_CMD 0x05
- #define WRITE_STATUS_REG_CMD 0x01
- #define READ_FUNCTION_REGISTER 0X48
- #define WRITE_FUNCTION_REGISTER 0x42
- #define WRITE_READ_PARAM_REG_CMD 0xC0
- #define PAGE_PROG_CMD 0x02
- #define QUAD_IN_PAGE_PROG_CMD 0x32
- #define EXT_QUAD_IN_PAGE_PROG_CMD 0x38
- #define SECTOR_ERASE_CMD 0xd7
- #define SECTOR_ERASE_QPI_CMD 0x20
- #define BLOCK_ERASE_CMD 0xD8
- #define BLOCK_ERASE_32K_CMD 0x52
- #define CHIP_ERASE_CMD 0xC7
- #define EXT_CHIP_ERASE_CMD 0x60
- #define PROG_ERASE_RESUME_CMD 0x7A
- #define EXT_PROG_ERASE_RESUME_CMD 0x30
- #define PROG_ERASE_SUSPEND_CMD 0x75
- #define EXT_PROG_ERASE_SUSPEND_CMD 0xB0
- #define ENTER_QUAD_CMD 0x35
- #define EXIT_QUAD_CMD 0xF5
- #define IS25LP064A_SR_WIP ((uint8_t)0x01)

IS25LP08D Registers

- #define IS25LP064A_SR_WREN ((uint8_t)0x02)
- #define IS25LP064A_SR_SRWREN ((uint8_t)0x80)
- #define IS25LP064A_SR_QE ((uint8_t)0x40)
- #define IS25LP064A_NVCR_NBADDR ((uint16_t)0x0001)
- #define IS25LP064A_NVCR_SEGMENT ((uint16_t)0x0002)
- #define IS25LP064A_NVCR_DUAL ((uint16_t)0x0004)
- #define IS25LP064A_NVCR_QUAB ((uint16_t)0x0008)
- #define IS25LP064A_NVCR_RH ((uint16_t)0x0010)
- #define IS25LP064A_NVCR_DTRP ((uint16_t)0x0020)
- #define IS25LP064A_NVCR_ODS ((uint16_t)0x01C0)
- #define IS25LP064A_NVCR_XIP ((uint16_t)0xE00)
- #define IS25LP064A_NVCR_NB_DUMMY ((uint16_t)0xF000)
- #define IS25LP064A_VCR_WRAP ((uint8_t)0x03)
- #define IS25LP064A_VCR_XIP ((uint8_t)0x08)
- #define IS25LP064A_VCR_NB_DUMMY ((uint8_t)0xF0)
- #define IS25LP064A_EAR_HIGHEST_SE ((uint8_t)0x03)
- #define IS25LP064A_EAR_THIRD_SEG ((uint8_t)0x02)
- #define IS25LP064A_EAR_SECOND_SEG ((uint8_t)0x01)
- #define IS25LP064A_EAR_LOWEST_SEG ((uint8_t)0x00)
- #define IS25LP064A_EVCR_ODS ((uint8_t)0x07)
- #define IS25LP064A_EVCR_RH ((uint8_t)0x10)
- #define IS25LP064A_EVCR_DTRP ((uint8_t)0x20)
- #define IS25LP064A_EVCR_DUAL ((uint8_t)0x40)

- #define IS25LP064A_EVCR_QUAD ((uint8_t)0x80)
- #define IS25LP064A_FSR_NBADDR ((uint8_t)0x01)
- #define IS25LP064A_FSR_PRERR ((uint8_t)0x02)
- #define IS25LP064A_FSR_PGSUS ((uint8_t)0x04)
- #define IS25LP064A_FSR_PGERR ((uint8_t)0x10)
- #define IS25LP064A_FSR_ERERR ((uint8_t)0x20)
- #define IS25LP064A_FSR_ERSUS ((uint8_t)0x40)
- #define IS25LP064A_FSR_READY ((uint8_t)0x80)
- #define ENTER_DEEP_POWER_DOWN 0XB9
- #define EXIT_DEEP_POWER_DOWN 0XB9
- #define RESET_ENABLE_CMD 0x66
- #define RESET_MEMORY_CMD 0x99
- #define READ_ID_CMD 0xAB
- #define READ_ID_CMD2 0x9F
- #define MULTIPLE_IO_READ_ID_CMD 0xAF
- #define READ_SERIAL_FLASH_DISCO_PARAM_CMD 0x5A
- #define READ_MANUFACT_AND_ID 0x90
- #define READ_UNIQUE_ID 0x4B
- #define NO_OP 0x00
- #define SECTOR_UNLOCK 0x26
- #define SECTOR_LOCK 0x24
- #define INFO_ROW_ERASE_CMD 0x64
- #define INFO_ROW_PROGRAM_CMD 0x62
- #define INFO_ROW_READ_CMD 0x68
- #define PAGE_PROG_CMD 0x02
- #define PAGE_PROG_CMD 0x02
- #define QUAD_IN_PAGE_PROG_CMD 0x32
- #define EXT_QUAD_IN_PAGE_PROG_CMD 0x38
- #define READ_CMD 0x03
- #define FAST_READ_CMD 0x0B
- #define FAST_READ_DTR_CMD 0x0D
- #define DUAL_OUT_FAST_READ_CMD 0x3B
- #define DUAL_INOUT_FAST_READ_CMD 0xBB
- #define DUAL_INOUT_FAST_READ_DTR_CMD 0xBD
- #define QUAD_OUT_FAST_READ_CMD 0x6B
- #define QUAD_INOUT_FAST_READ_CMD 0xEB
- #define QUAD_INOUT_FAST_READ_DTR_CMD 0xED
- #define WRITE_ENABLE_CMD 0x06
- #define WRITE_DISABLE_CMD 0x04
- #define READ_STATUS_REG_CMD 0x05
- #define WRITE_STATUS_REG_CMD 0x01
- #define READ_FUNCTION_REGISTER 0X48
- #define WRITE_FUNCTION_REGISTER 0x42
- #define READ_READ_PARAM_REG_CMD 0x61
- #define READ_EXT_READ_PARAM_CMD 0x81
- #define CLEAR_EXT_READ_PARAM_CMD 0x82
- #define WRITE_READ_PARAM_REG_CMD 0xC0
- #define WRITE_NV_READ_PARAM_REG_CMD 0x65
- #define EXT_WRITE_READ_PARAM_REG_CMD 0x63
- #define WRITE_EXT_READ_PARAM_REG_CMD 0x83
- #define WRITE_EXT_NV_READ_PARAM_REG_CMD 0x85
- #define QUAD_IN_FAST_PROG_CMD 0x32
- #define EXT_QUAD_IN_FAST_PROG_CMD 0x38
- #define SECTOR_ERASE_CMD 0xd7
- #define SECTOR_ERASE_QPI_CMD 0x20

- #define BLOCK_ERASE_CMD 0xD8
- #define BLOCK_ERASE_32K_CMD 0x52
- #define CHIP_ERASE_CMD 0xC7
- #define EXT_CHIP_ERASE_CMD 0x60
- #define PROG_ERASE_RESUME_CMD 0x7A
- #define EXT_PROG_ERASE_RESUME_CMD 0x30
- #define PROG_ERASE_SUSPEND_CMD 0x75
- #define EXT_PROG_ERASE_SUSPEND_CMD 0xB0
- #define ENTER_QUAD_CMD 0x35
- #define EXIT_QUAD_CMD 0xF5
- #define IS25LP080D_SR_WIP ((uint8_t)0x01)

IS25LP080D Registers

- #define IS25LP080D_SR_WREN ((uint8_t)0x02)
- #define IS25LP080D_SR_SRWREN ((uint8_t)0x80)
- #define IS25LP080D_SR_QE ((uint8_t)0x40)
- #define IS25LP080D_NVCR_NBADDR ((uint16_t)0x0001)
- #define IS25LP080D_NVCR_SEGMENT ((uint16_t)0x0002)
- #define IS25LP080D_NVCR_DUAL ((uint16_t)0x0004)
- #define IS25LP080D_NVCR_QUAB ((uint16_t)0x0008)
- #define IS25LP080D_NVCR_RH ((uint16_t)0x0010)
- #define IS25LP080D_NVCR_DTRP ((uint16_t)0x0020)
- #define IS25LP080D_NVCR_ODS ((uint16_t)0x01C0)
- #define IS25LP080D_NVCR_XIP ((uint16_t)0xE00)
- #define IS25LP080D_NVCR_NB_DUMMY ((uint16_t)0xF000)
- #define IS25LP080D_VCR_WRAP ((uint8_t)0x03)
- #define IS25LP080D_VCR_XIP ((uint8_t)0x08)
- #define IS25LP080D_VCR_NB_DUMMY ((uint8_t)0xF0)
- #define IS25LP080D_EAR_HIGHEST_SE ((uint8_t)0x03)
- #define IS25LP080D_EAR_THIRD_SEG ((uint8_t)0x02)
- #define IS25LP080D_EAR_SECOND_SEG ((uint8_t)0x01)
- #define IS25LP080D_EAR_LOWEST_SEG ((uint8_t)0x00)
- #define IS25LP080D_EVCR_ODS ((uint8_t)0x07)
- #define IS25LP080D_EVCR_RH ((uint8_t)0x10)
- #define IS25LP080D_EVCR_DTRP ((uint8_t)0x20)
- #define IS25LP080D_EVCR_DUAL ((uint8_t)0x40)
- #define IS25LP080D_EVCR_QUAD ((uint8_t)0x80)
- #define IS25LP080D_FSR_NBADDR ((uint8_t)0x01)
- #define IS25LP080D_FSR_PRERR ((uint8_t)0x02)
- #define IS25LP080D_FSR_PGSUS ((uint8_t)0x04)
- #define IS25LP080D_FSR_PGERR ((uint8_t)0x10)
- #define IS25LP080D_FSR_ERERR ((uint8_t)0x20)
- #define IS25LP080D_FSR_ERSUS ((uint8_t)0x40)
- #define IS25LP080D_FSR_READY ((uint8_t)0x80)

10.14.1 Detailed Description

Flash memory.

IS25LP080D Commands

10.14.2 Macro Definition Documentation

10.14.2.1 BLOCK_ERASE_32K_CMD [1/2]

```
#define BLOCK_ERASE_32K_CMD 0x52  
&
```

10.14.2.2 BLOCK_ERASE_32K_CMD [2/2]

```
#define BLOCK_ERASE_32K_CMD 0x52  
&
```

10.14.2.3 BLOCK_ERASE_CMD [1/2]

```
#define BLOCK_ERASE_CMD 0xD8  
&
```

10.14.2.4 BLOCK_ERASE_CMD [2/2]

```
#define BLOCK_ERASE_CMD 0xD8  
&
```

10.14.2.5 CHIP_ERASE_CMD [1/2]

```
#define CHIP_ERASE_CMD 0xC7  
&
```

10.14.2.6 CHIP_ERASE_CMD [2/2]

```
#define CHIP_ERASE_CMD 0xC7  
&
```

10.14.2.7 CLEAR_EXT_READ_PARAM_CMD

```
#define CLEAR_EXT_READ_PARAM_CMD 0x82  
&
```

10.14.2.8 DUAL_INOUT_FAST_READ_CMD [1/2]

```
#define DUAL_INOUT_FAST_READ_CMD 0xBB  
&
```

10.14.2.9 DUAL_INOUT_FAST_READ_CMD [2/2]

```
#define DUAL_INOUT_FAST_READ_CMD 0xBB  
&
```

10.14.2.10 DUAL_INOUT_FAST_READ_DTR_CMD [1/2]

```
#define DUAL_INOUT_FAST_READ_DTR_CMD 0xBD  
&
```

10.14.2.11 DUAL_INOUT_FAST_READ_DTR_CMD [2/2]

```
#define DUAL_INOUT_FAST_READ_DTR_CMD 0xBD  
&
```

10.14.2.12 DUAL_OUT_FAST_READ_CMD [1/2]

```
#define DUAL_OUT_FAST_READ_CMD 0x3B  
&
```

10.14.2.13 DUAL_OUT_FAST_READ_CMD [2/2]

```
#define DUAL_OUT_FAST_READ_CMD 0x3B  
&
```

10.14.2.14 ENTER_DEEP_POWER_DOWN [1/2]

```
#define ENTER_DEEP_POWER_DOWN 0XB9  
Low Power Modes &
```

10.14.2.15 ENTER_DEEP_POWER_DOWN [2/2]

```
#define ENTER_DEEP_POWER_DOWN 0XB9  
Low Power Modes &
```

10.14.2.16 ENTER_QUAD_CMD [1/2]

```
#define ENTER_QUAD_CMD 0x35
```

Quad Operations

10.14.2.17 ENTER_QUAD_CMD [2/2]

```
#define ENTER_QUAD_CMD 0x35
```

Quad Operations

10.14.2.18 EXIT_DEEP_POWER_DOWN [1/2]

```
#define EXIT_DEEP_POWER_DOWN 0XB9
```

&

10.14.2.19 EXIT_DEEP_POWER_DOWN [2/2]

```
#define EXIT_DEEP_POWER_DOWN 0XB9
```

&

10.14.2.20 EXIT_QUAD_CMD [1/2]

```
#define EXIT_QUAD_CMD 0xF5
```

&

10.14.2.21 EXIT_QUAD_CMD [2/2]

```
#define EXIT_QUAD_CMD 0xF5
```

&

10.14.2.22 EXT_CHIP_ERASE_CMD [1/2]

```
#define EXT_CHIP_ERASE_CMD 0x60
```

&

10.14.2.23 EXT_CHIP_ERASE_CMD [2/2]

```
#define EXT_CHIP_ERASE_CMD 0x60
```

&

10.14.2.24 EXT_PROG_ERASE_RESUME_CMD [1/2]

```
#define EXT_PROG_ERASE_RESUME_CMD 0x30  
&
```

10.14.2.25 EXT_PROG_ERASE_RESUME_CMD [2/2]

```
#define EXT_PROG_ERASE_RESUME_CMD 0x30  
&
```

10.14.2.26 EXT_PROG_ERASE_SUSPEND_CMD [1/2]

```
#define EXT_PROG_ERASE_SUSPEND_CMD 0xB0  
&
```

10.14.2.27 EXT_PROG_ERASE_SUSPEND_CMD [2/2]

```
#define EXT_PROG_ERASE_SUSPEND_CMD 0xB0  
&
```

10.14.2.28 EXT_QUAD_IN_FAST_PROG_CMD

```
#define EXT_QUAD_IN_FAST_PROG_CMD 0x38  
&
```

10.14.2.29 EXT_QUAD_IN_PAGE_PROG_CMD [1/2]

```
#define EXT_QUAD_IN_PAGE_PROG_CMD 0x38  
&
```

10.14.2.30 EXT_QUAD_IN_PAGE_PROG_CMD [2/2]

```
#define EXT_QUAD_IN_PAGE_PROG_CMD 0x38  
&
```

10.14.2.31 EXT_WRITE_READ_PARAM_REG_CMD

```
#define EXT_WRITE_READ_PARAM_REG_CMD 0x63  
volatile
```

10.14.2.32 FAST_READ_CMD [1/2]

```
#define FAST_READ_CMD 0x0B
```

&

10.14.2.33 FAST_READ_CMD [2/2]

```
#define FAST_READ_CMD 0x0B
```

&

10.14.2.34 FAST_READ_DTR_CMD [1/2]

```
#define FAST_READ_DTR_CMD 0x0D
```

&

10.14.2.35 FAST_READ_DTR_CMD [2/2]

```
#define FAST_READ_DTR_CMD 0x0D
```

&

10.14.2.36 INFO_ROW_ERASE_CMD [1/2]

```
#define INFO_ROW_ERASE_CMD 0x64
```

Security Information Row &

10.14.2.37 INFO_ROW_ERASE_CMD [2/2]

```
#define INFO_ROW_ERASE_CMD 0x64
```

Security Information Row &

10.14.2.38 INFO_ROW_PROGRAM_CMD [1/2]

```
#define INFO_ROW_PROGRAM_CMD 0x62
```

&

10.14.2.39 INFO_ROW_PROGRAM_CMD [2/2]

```
#define INFO_ROW_PROGRAM_CMD 0x62
```

&

10.14.2.40 INFO_ROW_READ_CMD [1/2]

```
#define INFO_ROW_READ_CMD 0x68  
&
```

10.14.2.41 INFO_ROW_READ_CMD [2/2]

```
#define INFO_ROW_READ_CMD 0x68  
&
```

10.14.2.42 IS25LP064A_EAR_HIGHEST_SE

```
#define IS25LP064A_EAR_HIGHEST_SE ((uint8_t)0x03)
```

Select the Highest 128Mb segment

10.14.2.43 IS25LP064A_EAR_LOWEST_SEG

```
#define IS25LP064A_EAR_LOWEST_SEG ((uint8_t)0x00)
```

Select the Lowest 128Mb segment (default)

10.14.2.44 IS25LP064A_EAR_SECOND_SEG

```
#define IS25LP064A_EAR_SECOND_SEG ((uint8_t)0x01)
```

Select the Second 128Mb segment

10.14.2.45 IS25LP064A_EAR_THIRD_SEG

```
#define IS25LP064A_EAR_THIRD_SEG ((uint8_t)0x02)
```

Select the Third 128Mb segment

10.14.2.46 IS25LP064A_EVCR_DTRP

```
#define IS25LP064A_EVCR_DTRP ((uint8_t)0x20)
```

Double transfer rate protocol

10.14.2.47 IS25LP064A_EVCR_DUAL

```
#define IS25LP064A_EVCR_DUAL ((uint8_t)0x40)
```

Dual I/O protocol

10.14.2.48 IS25LP064A_EVCR_ODS

```
#define IS25LP064A_EVCR_ODS ((uint8_t)0x07)
```

Output driver strength

10.14.2.49 IS25LP064A_EVCR_QUAD

```
#define IS25LP064A_EVCR_QUAD ((uint8_t)0x80)
```

Quad I/O protocol

10.14.2.50 IS25LP064A_EVCR_RH

```
#define IS25LP064A_EVCR_RH ((uint8_t)0x10)
```

Reset/hold

10.14.2.51 IS25LP064A_FSR_ERERR

```
#define IS25LP064A_FSR_ERERR ((uint8_t)0x20)
```

Erase error

10.14.2.52 IS25LP064A_FSR_ERSUS

```
#define IS25LP064A_FSR_ERSUS ((uint8_t)0x40)
```

Erase operation suspended

10.14.2.53 IS25LP064A_FSR_NBADDR

```
#define IS25LP064A_FSR_NBADDR ((uint8_t)0x01)
```

3-bytes or 4-bytes addressing

10.14.2.54 IS25LP064A_FSR_PGERR

```
#define IS25LP064A_FSR_PGERR ((uint8_t)0x10)
```

Program error

10.14.2.55 IS25LP064A_FSR_PGSUS

```
#define IS25LP064A_FSR_PGSUS ((uint8_t)0x04)
```

Program operation suspended

10.14.2.56 IS25LP064A_FSR_PRERR

```
#define IS25LP064A_FSR_PRERR ((uint8_t)0x02)
```

Protection error

10.14.2.57 IS25LP064A_FSR_READY

```
#define IS25LP064A_FSR_READY ((uint8_t)0x80)
```

Ready or command in progress

10.14.2.58 IS25LP064A_NVCR_DTRP

```
#define IS25LP064A_NVCR_DTRP ((uint16_t)0x0020)
```

Double transfer rate protocol

10.14.2.59 IS25LP064A_NVCR_DUAL

```
#define IS25LP064A_NVCR_DUAL ((uint16_t)0x0004)
```

Dual I/O protocol

10.14.2.60 IS25LP064A_NVCR_NB_DUMMY

```
#define IS25LP064A_NVCR_NB_DUMMY ((uint16_t)0xF000)
```

Number of dummy clock cycles

10.14.2.61 IS25LP064A_NVCR_NBADDR

```
#define IS25LP064A_NVCR_NBADDR ((uint16_t)0x0001)
```

3-bytes or 4-bytes addressing

10.14.2.62 IS25LP064A_NVCR_ODS

```
#define IS25LP064A_NVCR_ODS ((uint16_t)0x01C0)
```

Output driver strength

10.14.2.63 IS25LP064A_NVCR_QUAB

```
#define IS25LP064A_NVCR_QUAB ((uint16_t)0x0008)
```

Quad I/O protocol

10.14.2.64 IS25LP064A_NVCR_RH

```
#define IS25LP064A_NVCR_RH ((uint16_t)0x0010)
```

Reset/hold

10.14.2.65 IS25LP064A_NVCR_SEGMENT

```
#define IS25LP064A_NVCR_SEGMENT ((uint16_t)0x0002)
```

Upper or lower 128Mb segment selected by default

10.14.2.66 IS25LP064A_NVCR_XIP

```
#define IS25LP064A_NVCR_XIP ((uint16_t)0x0E00)
```

XIP mode at power-on reset

10.14.2.67 IS25LP064A_SR_QE

```
#define IS25LP064A_SR_QE ((uint8_t)0x40)
```

&

10.14.2.68 IS25LP064A_SR_SRWREN

```
#define IS25LP064A_SR_SRWREN ((uint8_t)0x80)
```

Status register write enable/disable

10.14.2.69 IS25LP064A_SR_WIP

```
#define IS25LP064A_SR_WIP ((uint8_t)0x01)
```

IS25LP08D Registers

Write in progress

10.14.2.70 IS25LP064A_SR_WREN

```
#define IS25LP064A_SR_WREN ((uint8_t)0x02)
```

Write enable latch

10.14.2.71 IS25LP064A_VCR_NB_DUMMY

```
#define IS25LP064A_VCR_NB_DUMMY ((uint8_t)0xF0)
```

Number of dummy clock cycles

10.14.2.72 IS25LP064A_VCR_WRAP

```
#define IS25LP064A_VCR_WRAP ((uint8_t)0x03)
```

Wrap

10.14.2.73 IS25LP064A_VCR_XIP

```
#define IS25LP064A_VCR_XIP ((uint8_t)0x08)
```

XIP

10.14.2.74 IS25LP080D_EAR_HIGHEST_SE

```
#define IS25LP080D_EAR_HIGHEST_SE ((uint8_t)0x03)
```

Select the Highest 128Mb segment

10.14.2.75 IS25LP080D_EAR_LOWEST_SEG

```
#define IS25LP080D_EAR_LOWEST_SEG ((uint8_t)0x00)
```

Select the Lowest 128Mb segment (default)

10.14.2.76 IS25LP080D_EAR_SECOND_SEG

```
#define IS25LP080D_EAR_SECOND_SEG ((uint8_t)0x01)
```

Select the Second 128Mb segment

10.14.2.77 IS25LP080D_EAR_THIRD_SEG

```
#define IS25LP080D_EAR_THIRD_SEG ((uint8_t)0x02)
```

Select the Third 128Mb segment

10.14.2.78 IS25LP080D_EVCR_DTRP

```
#define IS25LP080D_EVCR_DTRP ((uint8_t)0x20)
```

Double transfer rate protocol

10.14.2.79 IS25LP080D_EVCR_DUAL

```
#define IS25LP080D_EVCR_DUAL ((uint8_t)0x40)
```

Dual I/O protocol

10.14.2.80 IS25LP080D_EVCR_ODS

```
#define IS25LP080D_EVCR_ODS ((uint8_t)0x07)
```

Output driver strength

10.14.2.81 IS25LP080D_EVCR_QUAD

```
#define IS25LP080D_EVCR_QUAD ((uint8_t)0x80)
```

Quad I/O protocol

10.14.2.82 IS25LP080D_EVCR_RH

```
#define IS25LP080D_EVCR_RH ((uint8_t)0x10)
```

Reset/hold

10.14.2.83 IS25LP080D_FSR_ERERR

```
#define IS25LP080D_FSR_ERERR ((uint8_t)0x20)
```

Erase error

10.14.2.84 IS25LP080D_FSR_ERSUS

```
#define IS25LP080D_FSR_ERSUS ((uint8_t)0x40)
```

Erase operation suspended

10.14.2.85 IS25LP080D_FSR_NBADDR

```
#define IS25LP080D_FSR_NBADDR ((uint8_t)0x01)
```

3-bytes or 4-bytes addressing

10.14.2.86 IS25LP080D_FSR_PGERR

```
#define IS25LP080D_FSR_PGERR ((uint8_t)0x10)
```

Program error

10.14.2.87 IS25LP080D_FSR_PGSUS

```
#define IS25LP080D_FSR_PGSUS ((uint8_t)0x04)
```

Program operation suspended

10.14.2.88 IS25LP080D_FSR_PRERR

```
#define IS25LP080D_FSR_PRERR ((uint8_t)0x02)
```

Protection error

10.14.2.89 IS25LP080D_FSR_READY

```
#define IS25LP080D_FSR_READY ((uint8_t)0x80)
```

Ready or command in progress

10.14.2.90 IS25LP080D_NVCR_DTRP

```
#define IS25LP080D_NVCR_DTRP ((uint16_t)0x0020)
```

Double transfer rate protocol

10.14.2.91 IS25LP080D_NVCR_DUAL

```
#define IS25LP080D_NVCR_DUAL ((uint16_t)0x0004)
```

Dual I/O protocol

10.14.2.92 IS25LP080D_NVCR_NB_DUMMY

```
#define IS25LP080D_NVCR_NB_DUMMY ((uint16_t)0xF000)
```

Number of dummy clock cycles

10.14.2.93 IS25LP080D_NVCR_NBADDR

```
#define IS25LP080D_NVCR_NBADDR ((uint16_t)0x0001)
```

3-bytes or 4-bytes addressing

10.14.2.94 IS25LP080D_NVCR_ODS

```
#define IS25LP080D_NVCR_ODS ((uint16_t)0x01C0)
```

Output driver strength

10.14.2.95 IS25LP080D_NVCR_QUAB

```
#define IS25LP080D_NVCR_QUAB ((uint16_t)0x0008)
```

Quad I/O protocol

10.14.2.96 IS25LP080D_NVCR_RH

```
#define IS25LP080D_NVCR_RH ((uint16_t)0x0010)
```

Reset/hold

10.14.2.97 IS25LP080D_NVCR_SEGMENT

```
#define IS25LP080D_NVCR_SEGMENT ((uint16_t)0x0002)
```

Upper or lower 128Mb segment selected by default

10.14.2.98 IS25LP080D_NVCR_XIP

```
#define IS25LP080D_NVCR_XIP ((uint16_t)0x0E00)
```

XIP mode at power-on reset

10.14.2.99 IS25LP080D_SR_QE

```
#define IS25LP080D_SR_QE ((uint8_t)0x40)
```

&

10.14.2.100 IS25LP080D_SR_SRWREN

```
#define IS25LP080D_SR_SRWREN ((uint8_t)0x80)
```

Status register write enable/disable

10.14.2.101 IS25LP080D_SR_WIP

```
#define IS25LP080D_SR_WIP ((uint8_t)0x01)
```

IS25LP08D Registers

Status Register Write in progress

10.14.2.102 IS25LP080D_SR_WREN

```
#define IS25LP080D_SR_WREN ((uint8_t)0x02)
```

Write enable latch

10.14.2.103 IS25LP080D_VCR_NB_DUMMY

```
#define IS25LP080D_VCR_NB_DUMMY ((uint8_t)0xF0)
```

Number of dummy clock cycles

10.14.2.104 IS25LP080D_VCR_WRAP

```
#define IS25LP080D_VCR_WRAP ((uint8_t)0x03)
```

Wrap

10.14.2.105 IS25LP080D_VCR_XIP

```
#define IS25LP080D_VCR_XIP ((uint8_t)0x08)
```

XIP

10.14.2.106 MULTIPLE_IO_READ_ID_CMD [1/2]

```
#define MULTIPLE_IO_READ_ID_CMD 0xAF
```

&

10.14.2.107 MULTIPLE_IO_READ_ID_CMD [2/2]

```
#define MULTIPLE_IO_READ_ID_CMD 0xAF
```

&

10.14.2.108 NO_OP [1/2]

```
#define NO_OP 0x00
```

Cancels Reset Enable

10.14.2.109 NO_OP [2/2]

```
#define NO_OP 0x00
```

Cancels Reset Enable

10.14.2.110 PAGE_PROG_CMD [1/3]

```
#define PAGE_PROG_CMD 0x02
```

Page Program Operations

10.14.2.111 PAGE_PROG_CMD [2/3]

```
#define PAGE_PROG_CMD 0x02
```

Page Operations

Program Operations

10.14.2.112 PAGE_PROG_CMD [3/3]

```
#define PAGE_PROG_CMD 0x02
```

Page Operations

Program Operations

10.14.2.113 PROG_ERASE_RESUME_CMD [1/2]

```
#define PROG_ERASE_RESUME_CMD 0x7A
```

&

10.14.2.114 PROG_ERASE_RESUME_CMD [2/2]

```
#define PROG_ERASE_RESUME_CMD 0x7A
```

&

10.14.2.115 PROG_ERASE_SUSPEND_CMD [1/2]

```
#define PROG_ERASE_SUSPEND_CMD 0x75
```

&

10.14.2.116 PROG_ERASE_SUSPEND_CMD [2/2]

```
#define PROG_ERASE_SUSPEND_CMD 0x75
```

&

10.14.2.117 QUAD_IN_FAST_PROG_CMD

```
#define QUAD_IN_FAST_PROG_CMD 0x32  
&
```

10.14.2.118 QUAD_IN_PAGE_PROG_CMD [1/2]

```
#define QUAD_IN_PAGE_PROG_CMD 0x32  
&
```

10.14.2.119 QUAD_IN_PAGE_PROG_CMD [2/2]

```
#define QUAD_IN_PAGE_PROG_CMD 0x32  
&
```

10.14.2.120 QUAD_INOUT_FAST_READ_CMD [1/2]

```
#define QUAD_INOUT_FAST_READ_CMD 0xEB  
&
```

10.14.2.121 QUAD_INOUT_FAST_READ_CMD [2/2]

```
#define QUAD_INOUT_FAST_READ_CMD 0xEB  
&
```

10.14.2.122 QUAD_INOUT_FAST_READ_DTR_CMD [1/2]

```
#define QUAD_INOUT_FAST_READ_DTR_CMD 0xED  
&
```

10.14.2.123 QUAD_INOUT_FAST_READ_DTR_CMD [2/2]

```
#define QUAD_INOUT_FAST_READ_DTR_CMD 0xED  
&
```

10.14.2.124 QUAD_OUT_FAST_READ_CMD [1/2]

```
#define QUAD_OUT_FAST_READ_CMD 0x6B  
&
```

10.14.2.125 QUAD_OUT_FAST_READ_CMD [2/2]

```
#define QUAD_OUT_FAST_READ_CMD 0x6B  
&
```

10.14.2.126 READ_CMD [1/2]

```
#define READ_CMD 0x03
```

Read Operations

10.14.2.127 READ_CMD [2/2]

```
#define READ_CMD 0x03
```

Read Operations

10.14.2.128 READ_EXT_READ_PARAM_CMD

```
#define READ_EXT_READ_PARAM_CMD 0x81
```

&

10.14.2.129 READ_FUNCTION_REGISTER [1/2]

```
#define READ_FUNCTION_REGISTER 0X48
```

&

10.14.2.130 READ_FUNCTION_REGISTER [2/2]

```
#define READ_FUNCTION_REGISTER 0X48
```

&

10.14.2.131 READ_ID_CMD [1/2]

```
#define READ_ID_CMD 0xAB
```

Identification Operations

10.14.2.132 READ_ID_CMD [2/2]

```
#define READ_ID_CMD 0xAB
```

Identification Operations

10.14.2.133 READ_ID_CMD2 [1/2]

```
#define READ_ID_CMD2 0x9F
```

&

10.14.2.134 READ_ID_CMD2 [2/2]

```
#define READ_ID_CMD2 0x9F
```

&

10.14.2.135 READ_MANUFACT_AND_ID [1/2]

```
#define READ_MANUFACT_AND_ID 0x90
```

&

10.14.2.136 READ_MANUFACT_AND_ID [2/2]

```
#define READ_MANUFACT_AND_ID 0x90
```

&

10.14.2.137 READ_READ_PARAM_REG_CMD

```
#define READ_READ_PARAM_REG_CMD 0x61
```

&

10.14.2.138 READ_SERIAL_FLASH_DISCO_PARAM_CMD [1/2]

```
#define READ_SERIAL_FLASH_DISCO_PARAM_CMD 0x5A
```

&

10.14.2.139 READ_SERIAL_FLASH_DISCO_PARAM_CMD [2/2]

```
#define READ_SERIAL_FLASH_DISCO_PARAM_CMD 0x5A
```

&

10.14.2.140 READ_STATUS_REG_CMD [1/2]

```
#define READ_STATUS_REG_CMD 0x05
```

Register Operations

10.14.2.141 READ_STATUS_REG_CMD [2/2]

```
#define READ_STATUS_REG_CMD 0x05
```

Register Operations

10.14.2.142 READ_UNIQUE_ID [1/2]

```
#define READ_UNIQUE_ID 0x4B
```

&

10.14.2.143 READ_UNIQUE_ID [2/2]

```
#define READ_UNIQUE_ID 0x4B
```

&

10.14.2.144 RESET_ENABLE_CMD [1/2]

```
#define RESET_ENABLE_CMD 0x66
```

Reset Operations

10.14.2.145 RESET_ENABLE_CMD [2/2]

```
#define RESET_ENABLE_CMD 0x66
```

Reset Operations

10.14.2.146 RESET_MEMORY_CMD [1/2]

```
#define RESET_MEMORY_CMD 0x99
```

&

10.14.2.147 RESET_MEMORY_CMD [2/2]

```
#define RESET_MEMORY_CMD 0x99
```

&

10.14.2.148 SECTOR_ERASE_CMD [1/2]

```
#define SECTOR_ERASE_CMD 0xd7
```

Erase Operations

10.14.2.149 SECTOR_ERASE_CMD [2/2]

```
#define SECTOR_ERASE_CMD 0xd7
```

Erase Operations

10.14.2.150 SECTOR_ERASE_QPI_CMD [1/2]

```
#define SECTOR_ERASE_QPI_CMD 0x20
```

&

10.14.2.151 SECTOR_ERASE_QPI_CMD [2/2]

```
#define SECTOR_ERASE_QPI_CMD 0x20
```

&

10.14.2.152 SECTOR_LOCK [1/2]

```
#define SECTOR_LOCK 0x24
```

&

10.14.2.153 SECTOR_LOCK [2/2]

```
#define SECTOR_LOCK 0x24
```

&

10.14.2.154 SECTOR_UNLOCK [1/2]

```
#define SECTOR_UNLOCK 0x26
```

&

10.14.2.155 SECTOR_UNLOCK [2/2]

```
#define SECTOR_UNLOCK 0x26
```

&

10.14.2.156 WRITE_DISABLE_CMD [1/2]

```
#define WRITE_DISABLE_CMD 0x04
```

&

10.14.2.157 WRITE_DISABLE_CMD [2/2]

```
#define WRITE_DISABLE_CMD 0x04  
&
```

10.14.2.158 WRITE_ENABLE_CMD [1/2]

```
#define WRITE_ENABLE_CMD 0x06  
Write Operations
```

10.14.2.159 WRITE_ENABLE_CMD [2/2]

```
#define WRITE_ENABLE_CMD 0x06  
Write Operations
```

10.14.2.160 WRITE_EXT_NV_READ_PARAM_REG_CMD

```
#define WRITE_EXT_NV_READ_PARAM_REG_CMD 0x85  
non-volatile
```

10.14.2.161 WRITE_EXT_READ_PARAM_REG_CMD

```
#define WRITE_EXT_READ_PARAM_REG_CMD 0x83  
volatile
```

10.14.2.162 WRITE_FUNCTION_REGISTER [1/2]

```
#define WRITE_FUNCTION_REGISTER 0x42  
&
```

10.14.2.163 WRITE_FUNCTION_REGISTER [2/2]

```
#define WRITE_FUNCTION_REGISTER 0x42  
&
```

10.14.2.164 WRITE_NV_READ_PARAM_REG_CMD

```
#define WRITE_NV_READ_PARAM_REG_CMD 0x65  
non-volatile
```

10.14.2.165 WRITE_READ_PARAM_REG_CMD [1/2]

```
#define WRITE_READ_PARAM_REG_CMD 0xC0  
&
```

10.14.2.166 WRITE_READ_PARAM_REG_CMD [2/2]

```
#define WRITE_READ_PARAM_REG_CMD 0xC0  
volatile
```

10.14.2.167 WRITE_STATUS_REG_CMD [1/2]

```
#define WRITE_STATUS_REG_CMD 0x01  
&
```

10.14.2.168 WRITE_STATUS_REG_CMD [2/2]

```
#define WRITE_STATUS_REG_CMD 0x01  
&
```

10.15 CODEC

Audio codecs.

Audio codecs.

Driver for the TI PCM3060 Audio Codec.

[Ak4556](#) Codec support.

Author

shensley

I don't see any real reason to have this be more than a function, but in case we want to add other functions down the road I wrapped the function in a class.

For now this is a limited interface that uses I2C to communicate with the PCM3060. The device can also be accessed with SPI, which is not yet supported.

For now all registers are set to their defaults, and the Init function will perform a MRST and SRST before setting the format to 24bit LJ, and disabling power save for both the ADC and DAC.

10.16 LED

LED driver devices.

LED driver devices.

10.17 SDRAM

SDRAM devices.

Classes

- class [SdramHandle](#)

Macros

- `#define DSY_SDRAM_DATA __attribute__((section(".sdram_data")))`
- `#define DSY_SDRAM_BSS __attribute__((section(".sdram_bss")))`

10.17.1 Detailed Description

SDRAM devices.

10.17.2 Macro Definition Documentation

10.17.2.1 DSY_SDRAM_BSS

```
#define DSY_SDRAM_BSS __attribute__((section(".sdram_bss")))
```

Variables placed here will not be initialized.

Usage

E.g. `int DSY_SDRAM_BSS uninitialized_var;`

10.17.2.2 DSY_SDRAM_DATA

```
#define DSY_SDRAM_DATA __attribute__((section(".sdram_data")))
```

Usage:

E.g. `int DSY_SDRAM_DATA initialized_var = 1;`

10.18 BOARDS

Daisy devices. Pod, seed, etc.

Classes

- class [daisy::DaisyField](#)
*Class that handles initializing all of the hardware specific to the Daisy Patch Board.
Helper funtions are also in place to provide easy access to built-in controls and peripherals.*
- class [daisy::patch_sm::DaisyPatchSM](#)
Board support file for [DaisyPatchSM](#) hardware.
- class [daisy::DaisyPetal](#)
Helpers and hardware definitions for daisy petal.
- class [daisy::DaisyPod](#)
*Class that handles initializing all of the hardware specific to the Daisy Patch Board.
Helper funtions are also in place to provide easy access to built-in controls and peripherals.*
- class [daisy::DaisySeed](#)
*This is the higher-level interface for the Daisy board.
All basic peripheral configuration/initialization is setup here.*
- class [daisy::DaisyVersio](#)
*Class that handles initializing all of the hardware specific to the Desmodus Versio hardware.
Helper funtions are also in place to provide easy access to built-in controls and peripherals.*

10.18.1 Detailed Description

Daisy devices. Pod, seed, etc.

10.19 UI

[UI](#) system. [UI](#) event queue, event readers, etc.

[UI](#) system. [UI](#) event queue, event readers, etc.

A queue that holds user input events in the [UI](#) system.

A generic [UI](#) system.

The base class for a page in the [UI](#) system.

A descriptor for a canvas in the [UI](#) system.

The type of arrow button in the [UI](#) system.

A potentiometer monitor that generates events in a [UiEventQueue](#).

A menu page for small screens.

A button monitor that generates events in a [UiEventQueue](#).

Base class for complex menus.

Author

jelliesen

This is the base class for any form of [UiPage](#) that displays a menu with multiple items. It handles all the logic behind a menu (selecting items, editing items, etc.) but doesn't implement any form of drawing. Implement your own drawing routines by overriding [UiPage::Draw\(\)](#) or use [FullScreenItemMenu](#)

Various types of items can be added to the menu, e.g.

- Checkbox items to toggle something on/off,
- Generic action items that call a function when activated,
- Value editing items for editing int/float/enum values,
- Close items that close the menu when activated (useful when no cancel button is available),
- Custom items that do whatever you want them to do, by providing a `CustomItem` object that handles the item-specific functionality.

The Abstract Menu can work with a wide variety of physical controls, here are a couple of combinations that are possible:

- 3 buttons: Left/Right to select and edit items, Ok to activate or enter/leave editing mode
- 5 buttons: Up/Down to select, Ok to activate/enter/leave, Cancel to close menu, function button to use coarse step size when editing
- 6 buttons: Left/Right to select, Up/Down to quickly edit selected items, Ok/Cancel to enter/leave
- 1 encoder with pushbutton
- 1 encoder for selecting items, another one for editing their values
- 1 encoder for selecting items and a value slider/potentiometer for editing
- ... any other combination of the above

These are the controls that the [AbstractMenu](#) will react to and their associated function:

- Left/Right buttons: Select items when `Orientation::leftRightSelectUpDownModify`, directly edit value of the selected item when `Orientation::upDownSelectLeftRightModify`.
- Up/Down buttons: Select items when `Orientation::upDownSelectLeftRightModify`, directly edit value of the selected item when `Orientation::leftRightSelectUpDownModify`.
- Ok button: Activate the action of the selected item (if it has an action, e.g. `ItemType::closeMenuItem`) otherwise enter/leave editing mode where the arrow buttons used for selection will now edit the value instead (only possible if `allowEntering` is set to `true`).
- Cancel button: Closes the menu page or leaves editing mode
- Menu encoder: Selects items; edits item value when in editing mode
- Value [Encoder](#): Edits value of selected item
- Value potentiometer/slider: Edits value of selected item
- Function button: Uses an alternate step size when modifying the value with encoders or buttons while pressed

Author

jelliesen

This class monitors a number of buttons and detects changes in their state. When a change is detected, an event is added to a [UiEventQueue](#). If required, software debouncing can be applied.

This class can monitor an arbitrary number of buttons or switches, as configured by its template argument `numButtons`. Each of the buttons is identified by an ID number from `0 .. numButtons - 1`. This number will also be used when events are posted to the [UiEventQueue](#). It's suggested to define an enum in your project like this:

```
enum ButtonId { btnOkay = 0, btnCancel = 1, btnStart = 2 };
```

In different projects, different ways of reading the button states will be used. That's why this class uses a generic backend that you'll have to write. The `BackendType` class will provide the source data for each button or switch. An instance of this backend must be supplied via the constructor. It must implement the following public function via which the [ButtonMonitor](#) will request the current state of the button:

```
bool IsButtonPressed(uint16_t buttonId);
```

Template Parameters

<code>BackendType</code>	The class type of the backend that will supply button states.
<code>numButtons</code>	The number of buttons to monitor.

Author

jelliesen

This class builds upon the menu logic of [AbstractMenu](#) and adds drawing routines that are suitable for small screens.

By default, it will paint to the canvas returned by [UI::GetPrimaryOneBitGraphicsDisplayId\(\)](#). It can also be configured to paint to a different canvas.

Each item will occupy the entire display. [FullScreenMenuItem](#) uses the LookAndFeel system to draw the items. This means that you can create your own graphics design by creating your own LookAndFeel based on the `OneBitGraphicsLookAndFeel` class and apply that either globally (`UI::SetOneBitGraphicsLookAndFeel()`) or to this page only (`UiPage::SetOneBitGraphicsLookAndFeel()`).

Author

jelliesen

This class monitors a number of potentiometers and detects pot movements. When a movement is detected, an event is added to a [UiEventQueue](#). Pots can be either "idle" or "moving" in which case different dead bands are applied to them. The current state and value of a pot can be requested at any time.

This class can monitor an arbitrary number of potentiometers, as configured by its template argument `numPots`. Each of the pots is identified by an ID number from `0 .. numPots - 1`. This number will also be used when events are posted to the [UiEventQueue](#). It's suggested to define an enum in your project like this:

```
enum PotId { potA = 0, potB = 1, potC = 2 };
```

In different projects, different ways of reading the potentiometer positions will be used. That's why this class uses a generic backend that you'll have to write. The `BackendType` class will provide the source data for each potentiometer. An instance of this backend must be supplied via the constructor. It must implement the following public function via which the [PotMonitor](#) will request the current value of the potentiometer in the range `0 .. 1`:

```
float GetPotValue(uint16_t potId);
```

Template Parameters

<i>BackendType</i>	The class type of the backend that will supply pot values.
<i>numPots</i>	The number of pots to monitor.

Author

jelliesen
jelliesen

A descriptor for a generic display / led / output device that's used in the [UI](#) system.

Author

jelliesen

This system allows you to create complex and dynamic user interfaces with menus, pages and dialogs. It holds a stack of pages. Each page can react to user input on buttons, potentiometers, and encoders while drawing to one or multiple displays, leds or other output devices.

User input is consumed from a [UiEventQueue](#) and distributed to the pages from the top down. If a page doesn't handle an event, it will be forwarded to the next page below.

Pages are drawn from the bottom up. Multiple abstract canvases can be used for the drawing, where each canvas could be a graphics display, LEDs, alphanumeric displays, etc. The [UI](#) system makes sure that drawing is executed with a constant refresh rate that can be individually specified for each canvas.

Author

jelliesen

A queue that holds user interface events such as button presses or encoder turns. The queue can be filled from hardware drivers and read from a [UI](#) object. Access to the queue is protected by a [ScopedIrqBlocker](#) - that means it's safe to add events from interrupt handlers.

10.20 UTILITY

General utilities. Ringbuffers, LED colors, OLED stuff, etc.

Classes

- struct [dsy_gpio_pin](#)
- class [daisy::Random](#)
 - True Random Number Generator access.*
- struct [DSY_SD_CardInfoTypeDef](#)
- class [daisy::Color](#)
- struct [FontDef](#)
- class [daisy::RingBuffer< T, size >](#)
- class [daisy::RingBuffer< T, 0 >](#)

Macros

- #define DMA_BUFFER_MEM_SECTION __attribute__((section(".sram1_bss"))))
- #define DTCM_MEM_SECTION __attribute__((section(".dtcmram_bss"))))
- #define FBIPMAX 0.999985f
- #define FBIPMIN (-FBIPMAX)
- #define U82F_SCALE 0.0078740f
- #define F2U8_SCALE 127.0f
- #define S82F_SCALE 0.0078125f
- #define F2S8_SCALE 127.0f
- #define S162F_SCALE 3.0517578125e-05f
- #define F2S16_SCALE 32767.0f
- #define F2S24_SCALE 8388608.0f
- #define S242F_SCALE 1.192092896e-07f
- #define S24SIGN 0x800000
- #define S322F_SCALE 4.6566129e-10f
- #define F2S32_SCALE 2147483647.f
- #define OUT_L out[0]
- #define OUT_R out[1]
- #define IN_L in[0]
- #define IN_R in[1]
- #define BSP_SD_CardInfo DSY_SD_CardInfoTypeDef
- #define MSD_OK ((uint8_t)0x00)
- #define MSD_ERROR ((uint8_t)0x01)
- #define MSD_ERROR_SD_NOT_PRESENT ((uint8_t)0x02)
- #define SD_TRANSFER_OK ((uint8_t)0x00)
- #define SD_TRANSFER_BUSY ((uint8_t)0x01)
- #define SD_PRESENT ((uint8_t)0x01)
- #define SD_NOT_PRESENT ((uint8_t)0x00)
- #define SD_DATATIMEOUT ((uint32_t)100000000)

Enumerations

- enum dsy_gpio_port {
 DSY_GPIOA , DSY_GPIOB , DSY_GPIOC , DSY_GPIOD ,
 DSY_GPIOE , DSY_GPIOF , DSY_GPIOG , DSY_GPIOH ,
 DSY_GPIOI , DSY_GPIOJ , DSY_GPIOK , DSY_GPIOX ,
 DSY_GPIO_LAST }

Functions

- FORCE_INLINE float cube (float x)
- FORCE_INLINE float u82f (uint8_t x)
- FORCE_INLINE uint8_t f2u8 (float x)
- FORCE_INLINE float s82f (int8_t x)
- FORCE_INLINE int8_t f2s8 (float x)
- FORCE_INLINE float s162f (int16_t x)
- FORCE_INLINE int16_t f2s16 (float x)
- FORCE_INLINE float s242f (int32_t x)
- FORCE_INLINE int32_t f2s24 (float x)
- FORCE_INLINE float s322f (int32_t x)
- FORCE_INLINE int32_t f2s32 (float x)
- FORCE_INLINE dsy_gpio_pin dsy_pin (dsy_gpio_port port, uint8_t pin)

- FORCE_INLINE uint8_t `dsy_pin_cmp` (`dsy_gpio_pin` *`a`, `dsy_gpio_pin` *`b`)
- uint8_t `BSP_SD_Init` (void)
- uint8_t `BSP_SD_ITConfig` (void)
- uint8_t `BSP_SD_ReadBlocks` (uint32_t *`pData`, uint32_t `ReadAddr`, uint32_t `NumOfBlocks`, uint32_t `Time-out`)
- uint8_t `BSP_SD_WriteBlocks` (uint32_t *`pData`, uint32_t `WriteAddr`, uint32_t `NumOfBlocks`, uint32_t `Timeout`)
- uint8_t `BSP_SD_ReadBlocks_DMA` (uint32_t *`pData`, uint32_t `ReadAddr`, uint32_t `NumOfBlocks`)
- uint8_t `BSP_SD_WriteBlocks_DMA` (uint32_t *`pData`, uint32_t `WriteAddr`, uint32_t `NumOfBlocks`)
- uint8_t `BSP_SD_Erase` (uint32_t `StartAddr`, uint32_t `EndAddr`)
- uint8_t `BSP_SD_GetCardState` (void)
- void `BSP_SD_GetCardInfo` (`DSY_SD_CardInfoTypeDef` *`CardInfo`)
- uint8_t `BSP_SD_IsDetected` (void)
- void `BSP_SD_AbortCallback` (void)
- void `BSP_SD_WriteCpltCallback` (void)
- void `BSP_SD_ReadCpltCallback` (void)
- GPIO_TypeDef * `dsy_hal_map_get_port` (const `dsy_gpio_pin` *`p`)
- uint16_t `dsy_hal_map_get_pin` (const `dsy_gpio_pin` *`p`)
- void `dsy_hal_map_gpio_clk_enable` (`dsy_gpio_port` `port`)
- void `dsy_get_unique_id` (uint32_t *`w0`, uint32_t *`w1`, uint32_t *`w2`)

Variables

- `FontDef Font_6x8`
- `FontDef Font_7x10`
- `FontDef Font_11x18`
- `FontDef Font_16x26`

10.20.1 Detailed Description

General utilities. Ringbuffers, LED colors, OLED stuff, etc.

A `MappedValue` that maps an list of strings linearly.

A `MappedValue` that maps an int value linearly.

A `MappedValue` that maps a float value using various mapping functions.

Abstract base class for a value that is mapped to a 0..1 normalized range.

A safe and convenient statically allocated string with `constexpr` powers.

CPU load metering.

Author

jelliesen

To measure the CPU load of your audio processing, create a `CpuLoadMeter` and initialize it with your block size and sample rate. Then at the beginning of the audio callback, call `OnBlockStart()`, and at the end of the audio callback, call `OnBlockEnd()`. You can then read out the minimum, maximum and average CPU load.

Author

jelliesen

This string class is statically allocated. All of its functions can be evaluated at compile time through the power of `constexpr`.

Author

jelliesen

10.20.2 Macro Definition Documentation

10.20.2.1 BSP_SD_CardInfo

```
#define BSP_SD_CardInfo DSY_SD_CardInfoTypeDef
```

&

10.20.2.2 DMA_BUFFER_MEM_SECTION

```
#define DMA_BUFFER_MEM_SECTION __attribute__((section(".sram1_bss")))
```

Macro for area of memory that is configured as cacheless. This should be used primarily for DMA buffers, and the like.

10.20.2.3 DTCM_MEM_SECTION

```
#define DTCM_MEM_SECTION __attribute__((section(".dtcmram_bss")))
```

THE DTCM RAM section is also non-cached. However, is not suitable for DMA transfers. Performance is on par with internal SRAM w/ cache enabled.

10.20.2.4 F2S16_SCALE

```
#define F2S16_SCALE 32767.0f
```

$(2^{**} 15) - 1$

10.20.2.5 F2S24_SCALE

```
#define F2S24_SCALE 8388608.0f
```

$2^{**} 23$

10.20.2.6 F2S32_SCALE

```
#define F2S32_SCALE 2147483647.f
```

$(2^{**} 31) - 1$

10.20.2.7 F2S8_SCALE

```
#define F2S8_SCALE 127.0f
```

$(2^{**} 7) - 1$

10.20.2.8 F2U8_SCALE

```
#define F2U8_SCALE 127.0f
```

128 - 1

10.20.2.9 FBIPMAX

```
#define FBIPMAX 0.999985f
```

close to 1.0f-LSB at 16 bit

10.20.2.10 FBIPMIN

```
#define FBIPMIN (-FBIPMAX)
```

- (1 - LSB)

10.20.2.11 IN_L

```
#define IN_L in[0]
```

shorthand macro for simplifying the reading of the left channel of a non-interleaved input buffer named in

10.20.2.12 IN_R

```
#define IN_R in[1]
```

shorthand macro for simplifying the reading of the right channel of a non-interleaved input buffer named in

10.20.2.13 MSD_ERROR

```
#define MSD_ERROR ((uint8_t)0x01)
```

&

10.20.2.14 MSD_ERROR_SD_NOT_PRESENT

```
#define MSD_ERROR_SD_NOT_PRESENT ((uint8_t)0x02)
```

&

10.20.2.15 MSD_OK

```
#define MSD_OK ((uint8_t)0x00)  
&
```

10.20.2.16 OUT_L

```
#define OUT_L out[0]
```

shorthand macro for simplifying the reading of the left channel of a non-interleaved output buffer named out

10.20.2.17 OUT_R

```
#define OUT_R out[1]
```

shorthand macro for simplifying the reading of the right channel of a non-interleaved output buffer named out

10.20.2.18 S162F_SCALE

```
#define S162F_SCALE 3.0517578125e-05f  
1 / (2** 15)
```

10.20.2.19 S242F_SCALE

```
#define S242F_SCALE 1.192092896e-07f  
1 / (2 ** 23)
```

10.20.2.20 S24SIGN

```
#define S24SIGN 0x800000  
2 ** 23
```

10.20.2.21 S322F_SCALE

```
#define S322F_SCALE 4.6566129e-10f  
1 / (2** 31)
```

10.20.2.22 S82F_SCALE

```
#define S82F_SCALE 0.0078125f  
1 / (2**7)
```

10.20.2.23 SD_DATATIMEOUT

```
#define SD_DATATIMEOUT ((uint32_t)100000000)  
&
```

10.20.2.24 SD_NOT_PRESENT

```
#define SD_NOT_PRESENT ((uint8_t)0x00)  
&
```

10.20.2.25 SD_PRESENT

```
#define SD_PRESENT ((uint8_t)0x01)  
&
```

10.20.2.26 SD_TRANSFER_BUSY

```
#define SD_TRANSFER_BUSY ((uint8_t)0x01)  
&
```

10.20.2.27 SD_TRANSFER_OK

```
#define SD_TRANSFER_OK ((uint8_t)0x00)  
&
```

10.20.2.28 U82F_SCALE

```
#define U82F_SCALE 0.0078740f  
1 / 127
```

10.20.3 Enumeration Type Documentation**10.20.3.1 dsy_gpio_port**

```
enum dsy_gpio_port
```

Enums and a simple struct for defining a hardware pin on the MCU These correlate with the stm32 datasheet, and are used to configure the hardware.

This along with the [dsy_gpio_pin](#) class should no longer be used. They are available for backwards compatibility.

Please use GPIOPort enum and the Pin struct instead.

Enumerator

DSY_GPIOA	&
DSY_GPIOB	&
DSY_GPIOC	&
DSY_GPIOD	&
DSY_GPIOE	&
DSY_GPIOF	&
DSY_GPIOG	&
DSY_GPIOH	&
DSY_GPIOI	&
DSY_GPIOJ	&
DSY_GPIOK	&
DSY_GPIO_LAST	This is a non-existent port for unsupported bits of hardware.

10.20.4 Function Documentation**10.20.4.1 BSP_SD_AbortCallback()**

```
void BSP_SD_AbortCallback (
    void )
```

These functions can be modified in case the current settings (e.g. DMA stream) need to be changed for specific application needs /n

Abort the callback

10.20.4.2 BSP_SD_Erase()

```
uint8_t BSP_SD_Erase (
    uint32_t StartAddr,
    uint32_t EndAddr )
```

Erase a section of memory

Parameters

<i>StartAddr</i>	Address to start erasing at
<i>EndAddr</i>	Address to stop erasing at

Returns

card state, ERROR, etc.

10.20.4.3 BSP_SD_GetCardInfo()

```
void BSP_SD_GetCardInfo (
    DSY_SD_CardInfoTypeDef * CardInfo )
```

Parameters

* <i>CardInfo</i>	Pointer to write card info to
-------------------	-------------------------------

Parameters

<i>CardInfo</i>	&
-----------------	---

10.20.4.4 BSP_SD_GetCardState()

```
uint8_t BSP_SD_GetCardState (
    void )
```

Returns

card state, ERROR, etc.

10.20.4.5 BSP_SD_Init()

```
uint8_t BSP_SD_Init (
    void )
```

Returns

card state, ERROR, etc.

10.20.4.6 BSP_SD_IsDetected()

```
uint8_t BSP_SD_IsDetected (
    void )
```

Returns

Is card detected

10.20.4.7 BSP_SD_ITConfig()

```
uint8_t BSP_SD_ITConfig (
    void )
```

Returns

card state, ERROR, etc.

10.20.4.8 BSP_SD_ReadBlocks()

```
uint8_t BSP_SD_ReadBlocks (
    uint32_t * pData,
    uint32_t ReadAddr,
    uint32_t NumOfBlocks,
    uint32_t Timeout )
```

Parameters

<i>*pData</i>	&
<i>ReadAddr</i>	Address to read from
<i>NumOfBlocks</i>	Number of blocks to be read
<i>Timeout</i>	Timeout len in ms

Returns

OK ERROR, etc.

10.20.4.9 BSP_SD_ReadBlocks_DMA()

```
uint8_t BSP_SD_ReadBlocks_DMA (
    uint32_t * pData,
    uint32_t ReadAddr,
    uint32_t NumOfBlocks )
```

No timeout

Parameters

<i>*pData</i>	&
<i>ReadAddr</i>	Address to read from
<i>NumOfBlocks</i>	Number of blocks to be read

Returns

card state, ERROR, etc.

10.20.4.10 BSP_SD_ReadCpltCallback()

```
void BSP_SD_ReadCpltCallback (
    void )
```

Write complete callback

10.20.4.11 BSP_SD_WriteBlocks()

```
uint8_t BSP_SD_WriteBlocks (
    uint32_t * pData,
    uint32_t WriteAddr,
    uint32_t NumOfBlocks,
    uint32_t Timeout )
```

Parameters

<i>*pData</i>	&
<i>WriteAddr</i>	Address to write to
<i>NumOfBlocks</i>	Number of blocks to be written
<i>Timeout</i>	Timeout len in ms

Returns

card state, ERROR, etc.

10.20.4.12 BSP_SD_WriteBlocks_DMA()

```
uint8_t BSP_SD_WriteBlocks_DMA (
    uint32_t * pData,
    uint32_t WriteAddr,
    uint32_t NumOfBlocks )
```

No timeout

Parameters

<i>*pData</i>	&
<i>WriteAddr</i>	Address to write to
<i>NumOfBlocks</i>	Number of blocks to be read

Returns

card state, ERROR, etc.

10.20.4.13 BSP_SD_WriteCpltCallback()

```
void BSP_SD_WriteCpltCallback (
    void )
```

Read complete callback

10.20.4.14 cube()

```
FORCE_INLINE float cube (
    float x )
```

Computes cube.

Parameters

x	Number to be cubed
---	--------------------

Returns

x^3

10.20.4.15 dsy_get_unique_id()

```
void dsy_get_unique_id (
    uint32_t * w0,
    uint32_t * w1,
    uint32_t * w2 )
```

Returns 96-bit Unique ID of the MCU

Author

shensley

Date

May 2020 fills the three pointer arguments with the unique ID of the MCU.

Parameters

<code>*w0</code>	First pointer
<code>*w1</code>	Second pointer
<code>*w2</code>	Third pointer

10.20.4.16 dsy_hal_map_get_pin()

```
uint16_t dsy_hal_map_get_pin (
    const dsy\_gpio\_pin * p )
```

Parameters

<code>*p</code>	Pin pin to get
-----------------	----------------

Returns

HAL GPIO Pin as used in the HAL from a [dsy_gpio_pin](#) input.

10.20.4.17 dsy_hal_map_get_port()

```
GPIO_TypeDef* dsy_hal_map_get_port (
    const dsy\_gpio\_pin * p )
```

global structs, and helper functions for interfacing with the stm32 HAL library while it remains a dependency. This file should only be included from source files (c/cpp) Including it from a header within libdaisy would expose the entire HAL to the users. This should be an option for users, but should not be required.

Parameters

<code>*p</code>	Pin pin to get
-----------------	----------------

Returns

HAL GPIO_TypeDef as used in the HAL from a [dsy_gpio_pin](#) input.

10.20.4.18 dsy_hal_map_gpio_clk_enable()

```
void dsy_hal_map_gpio_clk_enable (
    dsy\_gpio\_port port )
```

Parameters

<i>port</i>	port clock to enable
-------------	----------------------

10.20.4.19 dsy_pin()

```
FORCE_INLINE dsy_gpio_pin dsy_pin (
    dsy_gpio_port port,
    uint8_t pin )
```

Helper for creating pins from port/pin combos easily

The [dsy_gpio_pin](#) struct should no longer be used, and is only available for backwards compatibility.

Please use Pin struct instead.

10.20.4.20 dsy_pin_cmp()

```
FORCE_INLINE uint8_t dsy_pin_cmp (
    dsy_gpio_pin * a,
    dsy_gpio_pin * b )
```

Helper for testing sameness of two [dsy_gpio_pins](#)

Returns

1 if same, 0 if different

The [dsy_gpio_pin](#) struct should no longer be used, and is only available for backwards compatibility.

Please use Pin struct instead.

10.20.4.21 f2s16()

```
FORCE_INLINE int16_t f2s16 (
    float x )
```

Converts float to Signed 16-bit < close to 1.0f-LSB at 16 bit

< - (1 - LSB)

< close to 1.0f-LSB at 16 bit

< - (1 - LSB)

< close to 1.0f-LSB at 16 bit

< close to 1.0f-LSB at 16 bit

< (2 ** 15) - 1

10.20.4.22 f2s24()

```
FORCE_INLINE int32_t f2s24 (
    float x )
```

Converts float to Signed 24-bit < close to 1.0f-LSB at 16 bit

< - (1 - LSB)

< close to 1.0f-LSB at 16 bit

< - (1 - LSB)

< close to 1.0f-LSB at 16 bit

< close to 1.0f-LSB at 16 bit

< 2 ** 23

10.20.4.23 f2s32()

```
FORCE_INLINE int32_t f2s32 (
    float x )
```

Converts float to Signed 24-bit < close to 1.0f-LSB at 16 bit

< - (1 - LSB)

< close to 1.0f-LSB at 16 bit

< - (1 - LSB)

< close to 1.0f-LSB at 16 bit

< close to 1.0f-LSB at 16 bit

< (2 ** 31) - 1

10.20.4.24 f2s8()

```
FORCE_INLINE int8_t f2s8 (
    float x )
```

Converts float to Signed 8-bit < close to 1.0f-LSB at 16 bit

< - (1 - LSB)

< close to 1.0f-LSB at 16 bit

< - (1 - LSB)

< close to 1.0f-LSB at 16 bit

< close to 1.0f-LSB at 16 bit

< (2 ** 7) - 1

10.20.4.25 f2u8()

```
FORCE_INLINE uint8_t f2u8 (
    float x )
```

Converts float to unsigned 8-bit < close to 1.0f-LSB at 16 bit

< - (1 - LSB)

< close to 1.0f-LSB at 16 bit

< - (1 - LSB)

< close to 1.0f-LSB at 16 bit

< close to 1.0f-LSB at 16 bit

< 128 - 1

< 128 - 1

10.20.4.26 s162f()

```
FORCE_INLINE float s162f (
    int16_t x )
```

Converts Signed 16-bit to float

Parameters

x	Number to be scaled.
---	----------------------

Returns

Scaled number.

< 1 / (2** 15)

10.20.4.27 s242f()

```
FORCE_INLINE float s242f (
    int32_t x )
```

Converts Signed 24-bit to float < 2 ** 23

< 2 ** 23

< 1 / (2 ** 23)

10.20.4.28 s322f()

```
FORCE_INLINE float s322f (
    int32_t x )
```

Converts Signed 32-bit to float $< 1 / (2^{**} 31)$

10.20.4.29 s82f()

```
FORCE_INLINE float s82f (
    int8_t x )
```

Converts Signed 8-bit to float

Parameters

x	Number to be scaled.
---	----------------------

Returns

Scaled number.

$< 1 / (2^{**} 7)$

10.20.4.30 u82f()

```
FORCE_INLINE float u82f (
    uint8_t x )
```

Converts unsigned 8-bit to float

Parameters

x	Number to be scaled.
---	----------------------

Returns

Scaled number.

$< 1 / 127$

10.20.5 Variable Documentation**10.20.5.1 Font_11x18**

`FontDef` `Font_11x18` [extern]

&

10.20.5.2 Font_16x26

```
FontDef Font_16x26 [extern]
```

&

10.20.5.3 Font_6x8

```
FontDef Font_6x8 [extern]
```

These are the different sizes of fonts (width x height in pixels per character)

10.20.5.4 Font_7x10

```
FontDef Font_7x10 [extern]
```

&

10.21 Lcd

10.22 Dac

Classes

- class [daisy::BlockingSpiTransport](#)
- struct [daisy::BlockingSpiTransport::Config](#)
- class [daisy::MAX11300Driver< Transport >](#)
Device Driver for the MAX11300 20 port ADC/DAC/GPIO device.
- struct [daisy::MAX11300Driver< Transport >::Config](#)

Typedefs

- using [daisy::MAX11300 = daisy::MAX11300Driver< BlockingSpiTransport >](#)

Enumerations

- enum class [daisy::BlockingSpiTransport::Result](#) { [daisy::BlockingSpiTransport::OK](#) , [daisy::BlockingSpiTransport::ERR](#) }
- enum [daisy::MAX11300Driver< Transport >::Pin](#) {
[PIN_0](#) , [PIN_1](#) , [PIN_2](#) , [PIN_3](#) ,
[PIN_4](#) , [PIN_5](#) , [PIN_6](#) , [PIN_7](#) ,
[PIN_8](#) , [PIN_9](#) , [PIN_10](#) , [PIN_11](#) ,
[PIN_12](#) , [PIN_13](#) , [PIN_14](#) , [PIN_15](#) ,
[PIN_16](#) , [PIN_17](#) , [PIN_18](#) , [PIN_19](#) }
- enum class [daisy::MAX11300Driver< Transport >::VoltageRange](#) { [ZERO_TO_10](#) = 0x0100 , [NEGATIVE_TO_5](#) = 0x0200 , [NEGATIVE_TO_10_TO_0](#) = 0x0300 }
- enum class [daisy::MAX11300Driver< Transport >::Result](#) { [daisy::MAX11300Driver< Transport >::OK](#) , [daisy::MAX11300Driver< Transport >::ERR](#) }

Functions

- void `daisy::BlockingSpiTransport::Config::Defaults ()`
- void `daisy::BlockingSpiTransport::Init (Config config)`
- bool `daisy::BlockingSpiTransport::Ready ()`
- `BlockingSpiTransport::Result daisy::BlockingSpiTransport::Transmit (uint8_t *buff, size_t size, uint32_t wait_us)`
- `BlockingSpiTransport::Result daisy::BlockingSpiTransport::TransmitAndReceive (uint8_t *tx_buff, uint8_t *rx_buff, size_t size)`
- void `daisy::MAX11300Driver< Transport >::Config::Defaults ()`
- `Result daisy::MAX11300Driver< Transport >::Init (Config config)`
- `Result daisy::MAX11300Driver< Transport >::ConfigurePinAsDigitalRead (Pin pin, float threshold_↔ voltage)`
- `Result daisy::MAX11300Driver< Transport >::ConfigurePinAsDigitalWrite (Pin pin, float output_↔ voltage)`
- `Result daisy::MAX11300Driver< Transport >::ConfigurePinAsAnalogRead (Pin pin, VoltageRange range)`
- `Result daisy::MAX11300Driver< Transport >::ConfigurePinAsAnalogWrite (Pin pin, VoltageRange range)`
- `Result daisy::MAX11300Driver< Transport >::DisablePin (Pin pin)`
- `uint16_t daisy::MAX11300Driver< Transport >::ReadAnalogPinRaw (Pin pin)`
- `float daisy::MAX11300Driver< Transport >::ReadAnalogPinVolts (Pin pin)`
- `void daisy::MAX11300Driver< Transport >::WriteAnalogPinRaw (Pin pin, uint16_t raw_value)`
- `void daisy::MAX11300Driver< Transport >::WriteAnalogPinVolts (Pin pin, float voltage)`
- `bool daisy::MAX11300Driver< Transport >::ReadDigitalPin (Pin pin)`
- `void daisy::MAX11300Driver< Transport >::WriteDigitalPin (Pin pin, bool value)`
- `Result daisy::MAX11300Driver< Transport >::Update ()`
- static `uint16_t daisy::MAX11300Driver< Transport >::VoltsTo12BitUint (float volts, VoltageRange range)`
- static `float daisy::MAX11300Driver< Transport >::TwelveBitUintToVolts (uint16_t value, VoltageRange range)`
- void `daisy::MAX11300Driver< Transport >::PinConfig::Defaults ()`

Variables

- `SpiHandle::Config daisy::BlockingSpiTransport::Config::spi_config`
- `Transport::Config daisy::MAX11300Driver< Transport >::Config::transport_config`
- `PinMode daisy::MAX11300Driver< Transport >::PinConfig::mode`
- `VoltageRange daisy::MAX11300Driver< Transport >::PinConfig::range`
- `float daisy::MAX11300Driver< Transport >::PinConfig::threshold`
- `uint16_t * daisy::MAX11300Driver< Transport >::PinConfig::value`

10.22.1 Detailed Description

10.22.2 Enumeration Type Documentation

10.22.2.1 Pin

```
template<typename Transport >
enum daisy::MAX11300Driver::Pin
```

Represents a pin/port on the MAX11300, of which there are 20.

10.22.2.2 Result [1/2]

```
enum daisy::BlockingSpiTransport::Result [strong]
```

Enumerator

OK	&
ERR	&

10.22.2.3 Result [2/2]

```
template<typename Transport >
enum daisy::MAX11300Driver::Result  [strong]
```

Indicates the success or failure of an operation within this class

Enumerator

OK	&
ERR	&

10.22.2.4 VoltageRange

```
template<typename Transport >
enum daisy::MAX11300Driver::VoltageRange  [strong]
```

Pins of the MAX11300 configured for AnalogRead/Write may be defined to operate within several pre-defined voltage ranges (assuming the power supply requirements for the range is met).

Pins configuired for DigitalRead/Write are 0-5V only, and do not tolerate or produce negative voltages.

WARNING, when a pin is configured as DigitalRead and a voltage lower than -250mV is applied, The codes read from ALL other pins configuired as AnalogRead will become unusably corrupted.

10.22.3 Function Documentation

10.22.3.1 Defaults() [1/2]

```
void daisy::BlockingSpiTransport::Config::Defaults ( )  [inline]
```

Default configuration for the MAX11300 using the SPI_1 peripheral and its default pinout.

10.22.3.2 Defaults() [2/2]

```
template<typename Transport >
void daisy::MAX11300Driver< Transport >::PinConfig::Defaults ( ) [inline]
```

Default pin settings - Disabled (High-Z mode)

10.22.3.3 Init()

```
template<typename Transport >
Result daisy::MAX11300Driver< Transport >::Init (
    Config config ) [inline]
```

Initialize the MAX11300

This method verifies SPI connectivity, configures the chip to operate within the scope of this implementation, and initializes all pins by default to High-Z mode.

Parameters

<i>config</i>	- The MAX11300 configuration
---------------	------------------------------

10.22.3.4 ReadAnalogPinRaw()

```
template<typename Transport >
uint16_t daisy::MAX11300Driver< Transport >::ReadAnalogPinRaw (
    Pin pin ) [inline]
```

Read the raw 12 bit (0-4095) value of a given ANALOG_IN (ADC) pin.

*note this read is local, call [MAX11300::Update\(\)](#) to sync with the MAX11300

Parameters

<i>pin</i>	- The pin of which to read the value
------------	--------------------------------------

Returns

- The raw, 12 bit value of the given ANALOG_IN (ADC) pin.

10.22.3.5 ReadAnalogPinVolts()

```
template<typename Transport >
float daisy::MAX11300Driver< Transport >::ReadAnalogPinVolts (
    Pin pin ) [inline]
```

Read the value of a given ADC pin in volts.

*note this read is local, call [MAX11300::Update\(\)](#) to sync with the MAX11300

Parameters

<i>pin</i>	- The pin of which to read the voltage
------------	--

Returns

- The value of the given ANALOG_IN (ADC) pin in volts

10.22.3.6 ReadDigitalPin()

```
template<typename Transport >
bool daisy::MAX11300Driver< Transport >::ReadDigitalPin (
    Pin pin ) [inline]
```

Read the state of a GPI pin

*note this read is local, call [MAX11300::Update\(\)](#) to sync with the MAX11300

Parameters

<i>pin</i>	- The pin of which to read the value
------------	--------------------------------------

Returns

- The boolean state of the pin

10.22.3.7 TwelveBitUintToVolts()

```
template<typename Transport >
static float daisy::MAX11300Driver< Transport >::TwelveBitUintToVolts (
    uint16_t value,
    VoltageRange range ) [inline], [static]
```

A utility function for converting the first 12 bits (0-4095) of an unsigned 16 bit integer value, to a voltage (float) value. The voltage value is scaled and bound to the given voltage range.

Parameters

<i>value</i>	the 12 bit value to convert
<i>range</i>	the MAX11300::VoltageRange to constrain to

Returns

the value as a float voltage constrained to the given voltage range

10.22.3.8 Update()

```
template<typename Transport >
Result daisy::MAX11300Driver< Transport >::Update ( ) [inline]
```

Update and synchronize the MAX11300 - This method does the following:

- Write all current ANALOG_OUT (DAC) values to the MAX11300
- Read all current ANALOG_IN (ADC) values to memory
- Write all GPO states to the MAX11300
- Read all GPI states to memory

TODO - Provide more info on usage location and the side-effects of blocking...

10.22.3.9 VoltsTo12BitUint()

```
template<typename Transport >
static uint16_t daisy::MAX11300Driver< Transport >::VoltsTo12BitUint (
    float volts,
    VoltageRange range ) [inline], [static]
```

A utility function for converting a voltage (float) value, bound to a given voltage range, to the first 12 bits (0-4095) of an unsigned 16 bit integer value.

Parameters

<i>volts</i>	the voltage to convert
<i>range</i>	the MAX11300::VoltageRange to constrain to

Returns

the voltage as 12 bit unsigned integer

10.22.3.10 WriteAnalogPinRaw()

```
template<typename Transport >
void daisy::MAX11300Driver< Transport >::WriteAnalogPinRaw (
    Pin pin,
    uint16_t raw_value ) [inline]
```

Write a raw 12 bit (0-4095) value to a given ANALOG_OUT (DAC) pin

*note this write is local, call [MAX11300::Update\(\)](#) to sync with the MAX11300

Parameters

<i>pin</i>	- The pin of which to write the value
------------	---------------------------------------

10.22.3.11 WriteAnalogPinVolts()

```
template<typename Transport >
void daisy::MAX11300Driver< Transport >::WriteAnalogPinVolts (
    Pin pin,
    float voltage ) [inline]
```

Write a voltage value, within the bounds of the configured volatge range, to a given ANALOG_OUT (DAC) pin.

*note this write is local, call [MAX11300::Update\(\)](#) to sync with the MAX11300

Parameters

<i>pin</i>	- The pin of which to write the voltage
------------	---

10.22.3.12 WriteDigitalPin()

```
template<typename Transport >
void daisy::MAX11300Driver< Transport >::WriteDigitalPin (
    Pin pin,
    bool value ) [inline]
```

Write a digital state to the given GPO pin

*note this write is local, call [MAX11300::Update\(\)](#) to sync with the MAX11300

Parameters

<i>pin</i>	- The pin of which to write the value
<i>value</i>	- the boolean state to write

10.22.4 Variable Documentation**10.22.4.1 mode**

```
template<typename Transport >
PinMode daisy::MAX11300Driver< Transport >::PinConfig::mode
&
```

10.22.4.2 range

```
template<typename Transport >
VoltageRange daisy::MAX11300Driver< Transport >::PinConfig::range
&
```

10.22.4.3 spi_config

```
SpiHandle::Config daisy::BlockingSpiTransport::Config::spi_config
&
```

10.22.4.4 threshold

```
template<typename Transport >
float daisy::MAX11300Driver< Transport >::PinConfig::threshold
```

This is a voltage value used as follows:

GPI - Defines what input voltage constitutes a logical 1 GPO - The output voltage of the pin at logical 1

10.22.4.5 value

```
template<typename Transport >
uint16_t* daisy::MAX11300Driver< Transport >::PinConfig::value
```

In the case of ANALOG_IN or ANALOG_OUT modes, this points to the current 12 bit value of the pin.

10.23 Externals

Chapter 11

Namespace Documentation

11.1 daisy Namespace Reference

Hardware defines and helpers for daisy field platform.

Namespaces

- [seed](#)

Classes

- struct [Pin](#)
representation of hardware port/pin combination
- class [DaisyField](#)
*Class that handles initializing all of the hardware specific to the Daisy Patch Board.
Helper funtions are also in place to provide easy access to built-in controls and peripherals.*
- class [DaisyPatch](#)
Helpers and hardware definitions for daisy petal.
- class [DaisyPod](#)
*Class that handles initializing all of the hardware specific to the Daisy Patch Board.
Helper funtions are also in place to provide easy access to built-in controls and peripherals.*
- class [DaisySeed](#)
*This is the higher-level interface for the Daisy board.
All basic peripheral configuration/initialization is setup here.*
- class [DaisyVersio](#)
*Class that handles initializing all of the hardware specific to the Desmodus Versio hardware.
Helper funtions are also in place to provide easy access to built-in controls and peripherals.*
- class [Ak4556](#)
- class [Pcm3060](#)
- class [Wm8731](#)
- class [LcdHD44780](#)
- class [LedDriverPca9685](#)
- class [BlockingSpiTransport](#)
- class [MAX11300Driver](#)

Device Driver for the MAX11300 20 port ADC/DAC/GPIO device.

- class [SSD130xI2CTransport](#)
- class [SSD130x4WireSpiTransport](#)
- class [SSD130xDriver](#)
- class [ShiftRegister4021](#)
- class [AudioHandle](#)
- class [AnalogControl](#)

Hardware Interface for control inputs

*Primarily designed for ADC input controls such as
potentiometers, and control voltage.*

- class [OneBitGraphicsDisplay](#)
- class [OneBitGraphicsDisplayImpl](#)
- class [Rectangle](#)
- class [OledDisplay](#)
- class [Encoder](#)

Generic Class for handling Quadrature Encoders

Inspired/influenced by Mutable Instruments (pichenettes) [Encoder](#) classes.

- class [GateIn](#)
- Generic Class for handling gate inputs through [GPIO](#).*

- class [Led](#)

LED Class providing simple Software PWM ability, etc

Eventually this will work with hardware PWM, and external LED Driver devices as well.

- class [Logger](#)
- Interface for simple USB logging.*

- class [Logger< LOGGER_NONE >](#)
- class [LoggerImpl](#)

Logging I/O underlying implementation.

- class [LoggerImpl< LOGGER_INTERNAL >](#)

Specialization for internal USB port.

- class [LoggerImpl< LOGGER_EXTERNAL >](#)

Specialization for external USB port.

- class [LoggerImpl< LOGGER_SEMIHOST >](#)

Specialization for semihosting (stdout)

- class [MidiUartTransport](#)

- class [MidiHandler](#)

Simple MIDI Handler

Parses bytes from an input into valid MidiEvents.

The MidiEvents fill a [FIFO](#) queue that the user can pop messages from.

- struct [NoteOffEvent](#)
- struct [NoteOnEvent](#)
- struct [PolyphonicKeyPressureEvent](#)
- struct [ControlChangeEvent](#)
- struct [ProgramChangeEvent](#)
- struct [ChannelPressureEvent](#)
- struct [PitchBendEvent](#)
- struct [SystemExclusiveEvent](#)
- struct [MTCQuarterFrameEvent](#)
- struct [SongPositionPointerEvent](#)
- struct [SongSelectEvent](#)
- struct [AllSoundOffEvent](#)
- struct [ResetAllControllersEvent](#)
- struct [LocalControlEvent](#)
- struct [AllNotesOffEvent](#)

- struct [OmniModeOffEvent](#)
- struct [OmniModeOnEvent](#)
- struct [MonoModeOnEvent](#)
- struct [PolyModeOnEvent](#)
- struct [MidiEvent](#)
- class [Parameter](#)
- class [RgbLed](#)
- class [Switch](#)
- class [Switch3](#)
- class [USBHostHandle](#)

Presents a USB Mass Storage Device host interface.

- class [MidiUsbTransport](#)
- struct [WavFileInfo](#)
- class [WavPlayer](#)
- struct [AdcChannelConfig](#)
- class [AdcHandle](#)
- class [DacHandle](#)
- class [GPIO](#)

General Purpose I/O control.

- class [I2CHandle](#)
- class [QSPIHandle](#)
- class [Random](#)

True Random Number Generator access.

- class [SaiHandle](#)
- class [SdmmcHandler](#)
- class [SpiHandle](#)
- class [TimerHandle](#)
- class [UartHandler](#)
- class [FatFSInterface](#)

Daisy FatFS Driver Interface.

- class [System](#)
- class [AbstractMenu](#)
- class [ButtonMonitor](#)
- class [FullScreenItemMenu](#)
- class [PotMonitor](#)
- struct [UiCanvasDescriptor](#)
- class [UiPage](#)
- class [UI](#)
- class [UiEventQueue](#)
- class [Color](#)
- class [CpuLoadMeter](#)
- class [FIFOBase](#)
- class [FIFO](#)
- class [FixedCapStrBase](#)
- class [FixedCapStr](#)
- class [MappedValue](#)
- class [MappedFloatValue](#)
- class [MappedIntValue](#)
- class [MappedStringListValue](#)
- class [PersistentStorage](#)

Non Volatile storage class for persistent settings on an external flash device.

- class [RingBuffer](#)
- class [RingBuffer< T, 0 >](#)
- class [ScopedIRQBlocker](#)

- class [StackBase](#)
- class [Stack](#)
- class [VoctCalibration](#)
Helper class for calibrating an input to 1V/oct response.
- struct [WAV_FormatTypeDef](#)
- class [WaveTableLoader](#)
- class [WavWriter](#)

Typedefs

- using [MAX11300](#) = [daisy::MAX11300Driver< BlockingSpiTransport >](#)
- using [SSD130x4WireSpi128x64Driver](#) = [daisy::SSD130xDriver< 128, 64, SSD130x4WireSpiTransport >](#)
- using [SSD130x4WireSpi128x32Driver](#) = [daisy::SSD130xDriver< 128, 32, SSD130x4WireSpiTransport >](#)
- using [SSD130x4WireSpi98x16Driver](#) = [daisy::SSD130xDriver< 98, 16, SSD130x4WireSpiTransport >](#)
- using [SSD130x4WireSpi64x48Driver](#) = [daisy::SSD130xDriver< 64, 48, SSD130x4WireSpiTransport >](#)
- using [SSD130x4WireSpi64x32Driver](#) = [daisy::SSD130xDriver< 64, 32, SSD130x4WireSpiTransport >](#)
- using [SSD130xl2c128x64Driver](#) = [daisy::SSD130xDriver< 128, 64, SSD130xl2CTransport >](#)
- using [SSD130xl2c128x32Driver](#) = [daisy::SSD130xDriver< 128, 32, SSD130xl2CTransport >](#)
- using [SSD130xl2c98x16Driver](#) = [daisy::SSD130xDriver< 98, 16, SSD130xl2CTransport >](#)
- using [SSD130xl2c64x48Driver](#) = [daisy::SSD130xDriver< 64, 48, SSD130xl2CTransport >](#)
- using [SSD130xl2c64x32Driver](#) = [daisy::SSD130xDriver< 64, 32, SSD130xl2CTransport >](#)
- using [MidiUartHandler](#) = [MidiHandler< MidiUartTransport >](#)
- using [MidiUsbHandler](#) = [MidiHandler< MidiUsbTransport >](#)

Enumerations

- enum [GPIOPort](#) {
 [PORTA](#) , [PORTB](#) , [PORTC](#) , [PORTD](#) ,
 [PORTE](#) , [PORTF](#) , [PORTG](#) , [PORTH](#) ,
 [PORTI](#) , [PORTJ](#) , [PORTK](#) , [PORTX](#) }
- GPIO Port names.*
- enum class [Alignment](#) {
 [centered](#) , [topLeft](#) , [topCentered](#) , [topRight](#) ,
 [bottomLeft](#) , [bottomCentered](#) , [bottomRight](#) , [centeredLeft](#) ,
 [centeredRight](#) }
- enum [LoggerDestination](#) { [LOGGER_NONE](#) , [LOGGER_INTERNAL](#) , [LOGGER_EXTERNAL](#) , [LOGGER_SEMIHOST](#) }
- enum [MidiMessageType](#) {
 [NoteOff](#) , [NoteOn](#) , [PolyphonicKeyPressure](#) , [ControlChange](#) ,
 [ProgramChange](#) , [ChannelPressure](#) , [PitchBend](#) , [SystemCommon](#) ,
 [SystemRealTime](#) , [ChannelMode](#) , [MessageLast](#) }
- enum [SystemCommonType](#) {
 [SystemExclusive](#) , [MTCQuarterFrame](#) , [SongPositionPointer](#) , [SongSelect](#) ,
 [SCUndefined0](#) , [SCUndefined1](#) , [TuneRequest](#) , [SysExEnd](#) ,
 [SystemCommonLast](#) }
- enum [SystemRealTimeType](#) {
 [TimingClock](#) , [SRTUndefined0](#) , [Start](#) , [Continue](#) ,
 [Stop](#) , [SRTUndefined1](#) , [ActiveSensing](#) , [Reset](#) ,
 [SystemRealTimeLast](#) }
- enum [ChannelModeType](#) {
 [AllSoundOff](#) , [ResetAllControllers](#) , [LocalControl](#) , [AllNotesOff](#) ,
 [OmniModeOff](#) , [OmniModeOn](#) , [MonoModeOn](#) , [PolyModeOn](#) ,
 [ChannelModeLast](#) }

- enum `ApplicationTypeDef` { `APPLICATION_IDLE` = 0 , `APPLICATION_START` , `APPLICATION_READY` , `APPLICATION_DISCONNECT` }
- enum class `ArrowButtonType` { `left` = 0 , `right` , `up` , `down` }
- enum `WavFileFormatCode` {
`WAVE_FORMAT_PCM` = 0x0001 , `WAVE_FORMAT_IEEE_FLOAT` = 0x0003 , `WAVE_FORMAT_ALAW` = 0x0006 , `WAVE_FORMAT_ULAW` = 0x0007 ,
`WAVE_FORMAT_EXTENSIBLE` = 0xFFFF }

Functions

- void `dsy_i2c_global_init` ()
- void `dsy_spi_global_init` ()
- template<class CharType , std::size_t capacity>
constexpr void `Swap` (const `FixedCapStr`< capacity, CharType > &lhs, const `FixedCapStr`< capacity, CharType > &rhs) noexcept

Variables

- const uint32_t `kWavFileChunkId` = 0x46464952
- const uint32_t `kWavFileWavId` = 0x45564157
- const uint32_t `kWavFileSubChunk1Id` = 0x20746d66
- const uint32_t `kWavFileSubChunk2Id` = 0x61746164

11.1.1 Detailed Description

Hardware defines and helpers for daisy field platform.

Device Driver for 16x2 LCD panel.

HD44780 with 4 data lines.

Example product: <https://www.adafruit.com/product/181>.

Author

StaffanMelin

Date

March 2021

New C++ `GPIO`

Copyright (C) Johannes Elliesen, 2021

11.1.2 Typedef Documentation

11.1.2.1 SSD130x4WireSpi128x32Driver

```
using daisy::SSD130x4WireSpi128x32Driver = typedef daisy::SSD130xDriver<128, 32, SSD130x4WireSpiTransport>
```

A driver for the SSD1306/SSD1309 128x32 OLED displays connected via 4 wire SPI

11.1.2.2 SSD130x4WireSpi128x64Driver

```
using daisy::SSD130x4WireSpi128x64Driver = typedef daisy::SSD130xDriver<128, 64, SSD130x4WireSpiTransport>
```

A driver for the SSD1306/SSD1309 128x64 OLED displays connected via 4 wire SPI

11.1.2.3 SSD130x4WireSpi64x32Driver

```
using daisy::SSD130x4WireSpi64x32Driver = typedef daisy::SSD130xDriver<64, 32, SSD130x4WireSpiTransport>
```

A driver for the SSD1306/SSD1309 64x32 OLED displays connected via 4 wire SPI

11.1.2.4 SSD130x4WireSpi64x48Driver

```
using daisy::SSD130x4WireSpi64x48Driver = typedef daisy::SSD130xDriver<64, 48, SSD130x4WireSpiTransport>
```

A driver for the SSD1306/SSD1309 64x48 OLED displays connected via 4 wire SPI

11.1.2.5 SSD130x4WireSpi98x16Driver

```
using daisy::SSD130x4WireSpi98x16Driver = typedef daisy::SSD130xDriver<98, 16, SSD130x4WireSpiTransport>
```

A driver for the SSD1306/SSD1309 98x16 OLED displays connected via 4 wire SPI

11.1.2.6 SSD130xI2c128x32Driver

```
using daisy::SSD130xI2c128x32Driver = typedef daisy::SSD130xDriver<128, 32, SSD130xI2CTransport>
```

A driver for the SSD1306/SSD1309 128x32 OLED displays connected via I2C

11.1.2.7 SSD130xI2c128x64Driver

```
using daisy::SSD130xI2c128x64Driver = typedef daisy::SSD130xDriver<128, 64, SSD130xI2CTransport>
```

A driver for the SSD1306/SSD1309 128x64 OLED displays connected via I2C

11.1.2.8 SSD130xI2c64x32Driver

```
using daisy::SSD130xI2c64x32Driver = typedef daisy::SSD130xDriver<64, 32, SSD130xI2CTransport>
```

A driver for the SSD1306/SSD1309 64x32 OLED displays connected via I2C

11.1.2.9 SSD130xI2c64x48Driver

```
using daisy::SSD130xI2c64x48Driver = typedef daisy::SSD130xDriver<64, 48, SSD130xI2CTransport>
```

A driver for the SSD1306/SSD1309 64x48 OLED displays connected via I2C

11.1.2.10 SSD130xI2c98x16Driver

```
using daisy::SSD130xI2c98x16Driver = typedef daisy::SSD130xDriver<98, 16, SSD130xI2CTransport>
```

A driver for the SSD1306/SSD1309 98x16 OLED displays connected via I2C

11.1.3 Enumeration Type Documentation

11.1.3.1 Alignment

```
enum daisy::Alignment [strong]
```

Justifications

11.1.3.2 ApplicationTypeDef

```
enum daisy::ApplicationTypeDef
```

Status of USB Host application

11.1.3.3 ArrowButtonType

```
enum daisy::ArrowButtonType [strong]
```

Enumerator

left	The left arrow button.
right	The right arrow button.
up	The up arrow button.
down	The down arrow button.

11.1.3.4 ChannelModeType

```
enum daisy::ChannelModeType
```

Enumerator

AllSoundOff	&
ResetAllControllers	&
LocalControl	&
AllNotesOff	&
OmniModeOff	&
OmniModeOn	&
MonoModeOn	&
PolyModeOn	&
ChannelModeLast	&

11.1.3.5 GPIOPort

```
enum daisy::GPIOPort
```

GPIO Port names.

Enumerator

PORTA	Port A
PORTB	Port B
PORTC	Port C
PORTD	Port D
PORTE	Port E
PORTF	Port F
PORTG	Port G
PORTH	Port H
PORTI	Port I
PORTJ	Port J
PORTK	Port K
PORTX	Used as a dummy port to signal an invalid pin.

11.1.3.6 LoggerDestination

```
enum daisy::LoggerDestination
```

Enumeration of destination ports for debug logging

Enumerator

LOGGER_NONE	mute logging
LOGGER_INTERNAL	internal USB port
LOGGER_EXTERNAL	external USB port
LOGGER_SEMIHOST	stdout

11.1.3.7 MidiMessageType

```
enum daisy::MidiMessageType
```

Parsed from the Status Byte, these are the common Midi Messages that can be handled.
At this time only 3-byte messages are correctly parsed into MidiEvents.

Enumerator

NoteOff	&
NoteOn	&
PolyphonicKeyPressure	&
ControlChange	&
ProgramChange	&
ChannelPressure	&
PitchBend	&
SystemCommon	&
SystemRealTime	&
ChannelMode	&
MessageLast	&

11.1.3.8 SystemCommonType

```
enum daisy::SystemCommonType
```

Enumerator

SystemExclusive	&
MTCQuarterFrame	&
SongPositionPointer	&
SongSelect	&
SCUndefined0	&
SCUndefined1	&
TuneRequest	&
SysExEnd	&
SystemCommonLast	&

11.1.3.9 SystemRealTimeType

```
enum daisy::SystemRealTimeType
```

Enumerator

TimingClock	&
SRTUndefined0	&
Start	&
Continue	&
Stop	&
SRTUndefined1	&
ActiveSensing	&
Reset	&
SystemRealTimeLast	&

11.1.3.10 WavFileFormatCode

```
enum daisy::WavFileFormatCode
```

Standard Format codes for the waveform data.

According to spec, extensible should be used whenever:

- PCM data has more than 16 bits/sample
- The number of channels is more than 2
- The actual number of bits/sample is not equal to the container size
- The mapping from channels to speakers needs to be specified.

11.1.4 Function Documentation

11.1.4.1 dsy_i2c_global_init()

```
void daisy::dsy_i2c_global_init( )
```

internal. Used for global init.

11.1.5 Variable Documentation

11.1.5.1 kWavFileChunkId

```
const uint32_t daisy::kWavFileChunkId = 0x46464952
```

Constants for In-Header IDs "RIFF"

11.1.5.2 kWavFileSubChunk1Id

```
const uint32_t daisy::kWavFileSubChunk1Id = 0x20746d66  
"fmt "
```

11.1.5.3 kWavFileSubChunk2Id

```
const uint32_t daisy::kWavFileSubChunk2Id = 0x61746164  
"data"
```

11.1.5.4 kWavFileWaveId

```
const uint32_t daisy::kWavFileWaveId = 0x45564157  
"WAVE"
```

11.2 daisy::seed Namespace Reference

Variables

- `constexpr Pin D0 = Pin(PORTB, 12)`
- `constexpr Pin D1 = Pin(PORTC, 11)`
- `constexpr Pin D2 = Pin(PORTC, 10)`
- `constexpr Pin D3 = Pin(PORTC, 9)`
- `constexpr Pin D4 = Pin(PORTC, 8)`
- `constexpr Pin D5 = Pin(PORTD, 2)`
- `constexpr Pin D6 = Pin(PORTC, 12)`
- `constexpr Pin D7 = Pin(PORTG, 10)`
- `constexpr Pin D8 = Pin(PORTG, 11)`
- `constexpr Pin D9 = Pin(PORTB, 4)`
- `constexpr Pin D10 = Pin(PORTB, 5)`
- `constexpr Pin D11 = Pin(PORTB, 8)`
- `constexpr Pin D12 = Pin(PORTB, 9)`
- `constexpr Pin D13 = Pin(PORTB, 6)`
- `constexpr Pin D14 = Pin(PORTB, 7)`

- `constexpr Pin D15 = Pin(PORTC, 0)`
- `constexpr Pin D16 = Pin(PORTA, 3)`
- `constexpr Pin D17 = Pin(PORTB, 1)`
- `constexpr Pin D18 = Pin(PORTA, 7)`
- `constexpr Pin D19 = Pin(PORTA, 6)`
- `constexpr Pin D20 = Pin(PORTC, 1)`
- `constexpr Pin D21 = Pin(PORTC, 4)`
- `constexpr Pin D22 = Pin(PORTA, 5)`
- `constexpr Pin D23 = Pin(PORTA, 4)`
- `constexpr Pin D24 = Pin(PORTA, 1)`
- `constexpr Pin D25 = Pin(PORTA, 0)`
- `constexpr Pin D26 = Pin(PORTD, 11)`
- `constexpr Pin D27 = Pin(PORTG, 9)`
- `constexpr Pin D28 = Pin(PORTA, 2)`
- `constexpr Pin D29 = Pin(PORTB, 14)`
- `constexpr Pin D30 = Pin(PORTB, 15)`
- `constexpr Pin A0 = D15`
- `constexpr Pin A1 = D16`
- `constexpr Pin A2 = D17`
- `constexpr Pin A3 = D18`
- `constexpr Pin A4 = D19`
- `constexpr Pin A5 = D20`
- `constexpr Pin A6 = D21`
- `constexpr Pin A7 = D22`
- `constexpr Pin A8 = D23`
- `constexpr Pin A9 = D24`
- `constexpr Pin A10 = D25`
- `constexpr Pin A11 = D28`
- `constexpr Pin D31 = Pin(PORTC, 2)`
- `constexpr Pin D32 = Pin(PORTC, 3)`
- `constexpr Pin A12 = D31`
- `constexpr Pin A13 = D32`

11.2.1 Detailed Description

`seed` namespace contains pinout constants for addressing the pins on the Daisy Seed SOM.

11.2.2 Variable Documentation

11.2.2.1 A0

```
constexpr Pin daisy::seed::A0 = D15 [constexpr]
```

Analog pins share same pins as digital pins

11.2.2.2 A12

```
constexpr Pin daisy::seed::A12 = D31 [constexpr]
```

Analog Pin alias

11.2.2.3 D0

```
constexpr Pin daisy::seed::D0 = Pin(PORTB, 12) [constexpr]
```

Constant Pinout consts

11.2.2.4 D31

```
constexpr Pin daisy::seed::D31 = Pin(PORTC, 2) [constexpr]
```

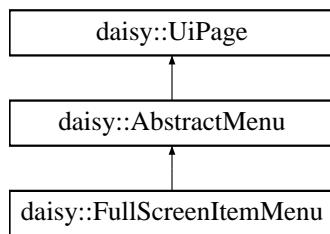
Pins unique to Daisy Seed 2 DFM

Chapter 12

Class Documentation

12.1 daisy::AbstractMenu Class Reference

Inheritance diagram for daisy::AbstractMenu:



Classes

- class [CustomItem](#)
- struct [ItemConfig](#)

Public Types

- enum class [Orientation](#) { [leftRightSelectUpDownModify](#) , [upDownSelectLeftRightModify](#) }
- enum class [ItemType](#) { [callbackFunctionItem](#) , [checkboxItem](#) , [valueItem](#) , [openUiPageItem](#) , [closeMenuItem](#) , [customItem](#) }

Public Member Functions

- uint16_t [GetNumItems](#) () const
- const [ItemConfig](#) & [GetItem](#) (uint16_t itemIdx) const
- void [SelectItem](#) (uint16_t itemIdx)
- int16_t [GetSelectedItemIdx](#) () const
- bool [OnOkayButton](#) (uint8_t number_of_presses, bool is_retriggering) override
- bool [OnCancelButton](#) (uint8_t number_of_presses, bool is_retriggering) override
- bool [OnArrowButton](#) ([ArrowButtonType](#) arrow_type, uint8_t number_of_presses, bool is_retriggering) override
- bool [OnFunctionButton](#) (uint8_t number_of_presses, bool is_retriggering) override
- bool [OnMenuEncoderTurned](#) (int16_t turns, uint16_t steps_per_revolution) override
- bool [OnValueEncoderTurned](#) (int16_t turns, uint16_t steps_per_revolution) override
- bool [OnValuePotMoved](#) (float new_position) override
- void [OnShow](#) () override

Protected Member Functions

- void [Init](#) (const [ItemConfig](#) *items, uint16_t numItems, [Orientation](#) orientation, bool allowEntering)
- bool [IsFunctionButtonDown](#) () const

Protected Attributes

- [Orientation orientation_](#) = [Orientation::upDownSelectLeftRightModify](#)
- const [ItemConfig](#) * [items_](#) = nullptr
- uint16_t [numItems_](#) = 0
- int16_t [selectedItemIdx_](#) = -1
- bool [allowEntering_](#) = true
- bool [isEditing_](#) = false

12.1.1 Member Enumeration Documentation

12.1.1.1 ItemType

```
enum daisy::AbstractMenu::ItemType [strong]
```

The types of entries that can be added to the menu.

Enumerator

callbackFunctionItem	Displays a text and calls a callback function when activated with the enter button
checkboxItem	Displays a name and a checkbox. When selected, the modify keys will allow to change the value directly. Pressing the enter button toggles the value.
valueItem	Displays a name and a value (with unit) from a MappedValue . When selected, the modify keys will allow to change the value directly. Pressing the enter button allows to change the value with the selection buttons as well.
openUiPageItem	Displays a name and opens another UiPage when selected.
closeMenuItem	Displays a text and closes the menu page when selected. This is useful when no cancel button is available to close a menu and return to the page below.
customItem	A custom item. See also CustomItem

12.1.1.2 Orientation

```
enum daisy::AbstractMenu::Orientation [strong]
```

Controls which buttons are used to navigate back and forth between the menu items (selection buttons) and which buttons can be used to modify their value directly without pressing the enter button first (modify buttons; these don't have to be available).

See also

[AbstractMenuPage](#)

Enumerator

leftRightSelectUpDownModify	left/right buttons => selection buttons, up/down => value buttons
upDownSelectLeftRightModify	up/down buttons => selection buttons, left/right => value buttons

12.1.2 Member Function Documentation**12.1.2.1 Init()**

```
void daisy::AbstractMenu::Init (
    const ItemConfig * items,
    uint16_t numItems,
    Orientation orientation,
    bool allowEntering ) [protected]
```

Call this from your child class to initialize the menu. It's okay to re-initialize an AbstractMene multiple times, even while it's displayed on the [UI](#).

Parameters

<i>items</i>	An array of ItemConfig that determine which items are available in the menu.
<i>numItems</i>	The number of items in the <i>items</i> array.
<i>orientation</i>	Controls which pair of arrow buttons are used for selection / editing
<i>allowEntering</i>	Globally controls if the Ok button can enter items for editing. If you have a physical controls that can edit selected items directly (value slider, a second arrow button pair, value encoder) you can set this to false, otherwise you set it to true so that the controls used for selecting items can now also be used to edit the values.

12.1.2.2 IsFunctionButtonDown()

```
bool daisy::AbstractMenu::IsFunctionButtonDown () const [inline], [protected]
```

Returns the state of the function button.

12.1.2.3 OnArrowButton()

```
bool daisy::AbstractMenu::OnArrowButton (
    ArrowButtonType arrowType,
    uint8_t numberOfPresses,
    bool isRetriggering ) [override], [virtual]
```

Called when an arrow button is pressed or released.

Parameters

<i>arrowType</i>	The arrow button affected.
<i>numberOfPresses</i>	Holds the number of successive button presses. It will be 1 on the first call and increasing by 1 with each successive call. A button down event is signaled by <i>numberOfButtonPresses</i> == 0.
<i>isRetriggering</i>	True if the button is auto-retriggering (due to being held down)

Returns

false, if you want the event to be passed on to the page below.

Reimplemented from [daisy::UiPage](#).

12.1.2.4 OnCancelButton()

```
bool daisy::AbstractMenu::OnCancelButton (
    uint8_t numberOfPresses,
    bool isRetriggering ) [override], [virtual]
```

Called when the cancel button is pressed or released.

Parameters

<i>numberOfPresses</i>	Holds the number of successive button presses. It will be 1 on the first call and increasing by 1 with each successive call. A button down event is signaled by <i>numberOfButtonPresses</i> == 0.
<i>isRetriggering</i>	True if the button is auto-retriggering (due to being held down)

Returns

false, if you want the event to be passed on to the page below.

Reimplemented from [daisy::UiPage](#).

12.1.2.5 OnFunctionButton()

```
bool daisy::AbstractMenu::OnFunctionButton (
    uint8_t numberOfPresses,
    bool isRetriggering ) [override], [virtual]
```

Called when the function button is pressed or released.

Parameters

<i>numberOfPresses</i>	Holds the number of successive button presses. It will be 1 on the first call and increasing by 1 with each successive call. A button down event is signaled by <i>numberOfButtonPresses</i> == 0.
<i>isRetriggering</i>	True if the button is auto-retriggering (due to being held down)

Returns

false, if you want the event to be passed on to the page below.

Reimplemented from [daisy::UiPage](#).

12.1.2.6 OnMenuEncoderTurned()

```
bool daisy::AbstractMenu::OnMenuEncoderTurned (
    int16_t turns,
    uint16_t stepsPerRevolution ) [override], [virtual]
```

Called when the menu encoder is turned.

Parameters

<i>turns</i>	The number of increments, positive is clockwise.
<i>stepsPerRevolution</i>	The total number of increments per revolution on this encoder.

Returns

false, if you want the event to be passed on to the page below.

Reimplemented from [daisy::UiPage](#).

12.1.2.7 OnOkayButton()

```
bool daisy::AbstractMenu::OnOkayButton (
    uint8_t numberOfPresses,
    bool isRetriggering ) [override], [virtual]
```

Called when the okay button is pressed or released.

Parameters

<i>numberOfPresses</i>	Holds the number of successive button presses. It will be 1 on the first call and increasing by 1 with each successive call. A button down event is signaled by <i>numberOfButtonPresses</i> == 0.
<i>isRetriggering</i>	True if the button is auto-retriggering (due to being held down)

Returns

false, if you want the event to be passed on to the page below.

Reimplemented from [daisy::UiPage](#).

12.1.2.8 OnShow()

```
void daisy::AbstractMenu::OnShow ( ) [override], [virtual]
```

Called when the page is added to the [UI](#).

Reimplemented from [daisy::UiPage](#).

12.1.2.9 OnValueEncoderTurned()

```
bool daisy::AbstractMenu::OnValueEncoderTurned (
    int16_t turns,
    uint16_t stepsPerRevolution ) [override], [virtual]
```

Called when the menu encoder is turned.

Parameters

<i>turns</i>	The number of increments, positive is clockwise.
<i>stepsPerRevolution</i>	The total number of increments per revolution on this encoder.

Returns

false, if you want the event to be passed on to the page below.

Reimplemented from [daisy::UiPage](#).

12.1.2.10 OnValuePotMoved()

```
bool daisy::AbstractMenu::OnValuePotMoved (
    float newPosition ) [override], [virtual]
```

Called when the value potentiometer is turned.

Parameters

<i>newPosition</i>	The new position in the range 0 .. 1
--------------------	--------------------------------------

Returns

false, if you want the event to be passed on to the page below.

Reimplemented from [daisy::UiPage](#).

12.1.3 Member Data Documentation

12.1.3.1 `allowEntering_`

```
bool daisy::AbstractMenu::allowEntering_ = true [protected]
```

If true, the menu allows "entering" an item to modify its value with the encoder / selection buttons.

12.1.3.2 `isEditing_`

```
bool daisy::AbstractMenu::isEditing_ = false [protected]
```

If true, the currently selected item index is "entered" so that it can be edited with the encoder/ selection buttons.

12.1.3.3 `items_`

```
const ItemConfig* daisy::AbstractMenu::items_ = nullptr [protected]
```

A list of items to include in the menu.

12.1.3.4 `numItems_`

```
uint16_t daisy::AbstractMenu::numItems_ = 0 [protected]
```

The number of items in `items_`

12.1.3.5 `orientation_`

```
Orientation daisy::AbstractMenu::orientation_ = Orientation::upDownSelectLeftRightModify [protected]
```

The orientation of the menu. This is used to determine which function the arrow keys will be assigned to.

12.1.3.6 `selectedItemIdx_`

```
int16_t daisy::AbstractMenu::selectedItemIdx_ = -1 [protected]
```

The currently selected item index

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

- src/ui/AbstractMenu.h

12.2 daisy::AdcChannelConfig Struct Reference

```
#include <adc.h>
```

Public Types

- enum `MuxPin` { `MUX_SEL_0` , `MUX_SEL_1` , `MUX_SEL_2` , `MUX_SEL_LAST` }

Public Member Functions

- void `InitSingle` (`dsy_gpio_pin` pin)
- void `InitMux` (`dsy_gpio_pin` adc_pin, `size_t` mux_channels, `dsy_gpio_pin` mux_0, `dsy_gpio_pin` mux_1={DSY_GPIOX, 0}, `dsy_gpio_pin` mux_2={DSY_GPIOX, 0})

Public Attributes

- `dsy_gpio_pin`
- `dsy_gpio mux_pin_[MUX_SEL_LAST]`
- `uint8_t mux_channels_`

12.2.1 Detailed Description

Configuration Structure for a given channel

12.2.2 Member Enumeration Documentation

12.2.2.1 MuxPin

```
enum daisy::AdcChannelConfig::MuxPin
```

Which pin to use for multiplexing

Enumerator

<code>MUX_SEL_0</code>	&
<code>MUX_SEL_1</code>	&
<code>MUX_SEL_2</code>	&
<code>MUX_SEL_LAST</code>	&

12.2.3 Member Function Documentation

12.2.3.1 InitMux()

```
void daisy::AdcChannelConfig::InitMux (
    dsy_gpio_pin adc_pin,
    size_t mux_channels,
    dsy_gpio_pin mux_0,
    dsy_gpio_pin mux_1 = {DSY_GPIOX, 0},
    dsy_gpio_pin mux_2 = {DSY_GPIOX, 0} )
```

Initializes a single ADC pin as a Multiplexed ADC. Requires a CD405X Multiplexer connected to the pin. You only need to supply the mux pins that are required, e.g. a 4052 mux would only require mux_0 and mux_1. Internal Callbacks handle the pin addressing.

Parameters

<i>mux_channels</i>	must be 1-8
<i>mux_0</i>	First mux pin
<i>mux_1</i>	Second mux pin
<i>mux_2</i>	Third mux pin
<i>adc_pin</i>	&

12.2.3.2 InitSingle()

```
void daisy::AdcChannelConfig::InitSingle (
    dsy_gpio_pin pin )
```

Initializes a single ADC pin as an ADC.

Parameters

<i>pin</i>	Pin to init.
------------	--------------

12.2.4 Member Data Documentation

12.2.4.1 mux_channels_

```
uint8_t daisy::AdcChannelConfig::mux_channels_
&
```

12.2.4.2 mux_pin_

```
dsy_gpio daisy::AdcChannelConfig::mux_pin_[MUX_SEL_LAST]
&
```

12.2.4.3 pin_

```
dsy_gpio daisy::AdcChannelConfig::pin_
```

&

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

- src/per/adc.h

12.3 daisy::AdcHandle Class Reference

```
#include <adc.h>
```

Public Types

- enum OverSampling {
 OVS_NONE , OVS_4 , OVS_8 , OVS_16 ,
 OVS_32 , OVS_64 , OVS_128 , OVS_256 ,
 OVS_512 , OVS_1024 , OVS_LAST }

Public Member Functions

- void **Init** (AdcChannelConfig *cfg, size_t num_channels, OverSampling ovs=OVS_32)
- void **Start** ()
- void **Stop** ()
- uint16_t **Get** (uint8_t chn) const
- uint16_t * **GetPtr** (uint8_t chn) const
- float **GetFloat** (uint8_t chn) const
- uint16_t **GetMux** (uint8_t chn, uint8_t idx) const
- uint16_t * **GetMuxPtr** (uint8_t chn, uint8_t idx) const
- float **GetMuxFloat** (uint8_t chn, uint8_t idx) const

12.3.1 Detailed Description

Handler for analog to digital conversion

12.3.2 Member Enumeration Documentation

12.3.2.1 OverSampling

```
enum daisy::AdcHandle::OverSampling
```

Supported oversampling amounts

Enumerator

OVS_NONE	&
OVS_4	&
OVS_8	&
OVS_16	&
OVS_32	&
OVS_64	&
OVS_128	&
OVS_256	&
OVS_512	&
OVS_1024	&
OVS_LAST	&

12.3.3 Member Function Documentation**12.3.3.1 Get()**

```
uint16_t daisy::AdcHandle::Get (
    uint8_t chn ) const
```

Single channel getter

Parameters

<i>chn</i>	channel to get
------------	----------------

Returns

Converted value

12.3.3.2 GetFloat()

```
float daisy::AdcHandle::GetFloat (
    uint8_t chn ) const
```

Get floating point from single channel

Parameters

<i>chn</i>	Channel to get from
------------	---------------------

Returns

Floating point converted value

12.3.3.3 GetMux()

```
uint16_t daisy::AdcHandle::GetMux (
    uint8_t chn,
    uint8_t idx ) const
```

Getters for multiplexed inputs on a single channel (up to 8 per ADC input).

Parameters

<i>chn</i>	Channel to get from
<i>idx</i>	&

Returns

data

12.3.3.4 GetMuxFloat()

```
float daisy::AdcHandle::GetMuxFloat (
    uint8_t chn,
    uint8_t idx ) const
```

Getters for multiplexed inputs on a single channel (up to 8 per ADC input).

Parameters

<i>chn</i>	Channel to get from
<i>idx</i>	&

Returns

Floating point data

12.3.3.5 GetMuxPtr()

```
uint16_t* daisy::AdcHandle::GetMuxPtr (
    uint8_t chn,
    uint8_t idx ) const
```

Getters for multiplexed inputs on a single channel. (Max 8 per chan)

Parameters

<i>chn</i>	Channel to get from
<i>idx</i>	&

Returns

Pointer to data

12.3.3.6 GetPtr()

```
uint16_t* daisy::AdcHandle::GetPtr (
    uint8_t chn ) const
```

Get pointer to a value from a single channel

Parameters

<i>chn</i>	
------------	--

Returns

Pointer to converted value

12.3.3.7 Init()

```
void daisy::AdcHandle::Init (
    AdcChannelConfig * cfg,
    size_t num_channels,
    OverSampling ovs = OVS_32 )
```

Initializes the ADC with the pins passed in.

Parameters

<i>*cfg</i>	an array of AdcChannelConfig of the desired channel
<i>num_channels</i>	number of ADC channels to initialize
<i>ovs</i>	Oversampling amount - Defaults to OVS_32

12.3.3.8 Start()

```
void daisy::AdcHandle::Start ( )
```

Starts reading from the ADC

12.3.3.9 Stop()

```
void daisy::AdcHandle::Stop ( )
```

Stops reading from the ADC

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

- src/per/adc.h

12.4 daisy::Ak4556 Class Reference

Static Public Member Functions

- static void [Init](#) ([dsy_gpio_pin](#) reset_pin)
- static void [DeInit](#) ([dsy_gpio_pin](#) reset_pin)

12.4.1 Member Function Documentation

12.4.1.1 DeInit()

```
static void daisy::Ak4556::DeInit (
    dsy\_gpio\_pin reset_pin ) [static]
```

Deinitialization function for [Ak4556](#) Can be called statically: Ak4556::DeInit(pin);

12.4.1.2 Init()

```
static void daisy::Ak4556::Init (
    dsy\_gpio\_pin reset_pin ) [static]
```

Initialization function for [Ak4556](#) Can be called statically: Ak4556::Init(pin);

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

- src/dev/codec_ak4556.h

12.5 daisy::AllNotesOffEvent Struct Reference

```
#include <MidiEvent.h>
```

Public Attributes

- int [channel](#)

12.5.1 Detailed Description

Struct containing [AllNotesOffEvent](#) data. Can be made from [MidiEvent](#)

12.5.2 Member Data Documentation

12.5.2.1 channel

```
int daisy::AllNotesOffEvent::channel
```

&

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

- [src/hid/MidiEvent.h](#)

12.6 daisy::AllSoundOffEvent Struct Reference

```
#include <MidiEvent.h>
```

Public Attributes

- int [channel](#)

12.6.1 Detailed Description

Struct containing sound off data. Can be made from [MidiEvent](#)

12.6.2 Member Data Documentation

12.6.2.1 channel

```
int daisy::AllSoundOffEvent::channel
```

&

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

- src/hid/MidiEvent.h

12.7 daisy::AnalogControl Class Reference

Hardware Interface for control inputs

Primarily designed for ADC input controls such as
potentiometers, and control voltage.

```
#include <ctrl.h>
```

Public Member Functions

- [AnalogControl \(\)](#)
- [~AnalogControl \(\)](#)
- void [Init \(uint16_t *adcptr, float sr, bool flip=false, bool invert=false, float slew_seconds=0.002f\)](#)
- void [InitBipolarCv \(uint16_t *adcptr, float sr\)](#)
- float [Process \(\)](#)
- float [Value \(\) const](#)
- void [SetCoeff \(float val\)](#)
- uint16_t [GetRawValue \(\)](#)
- float [GetRawFloat \(\)](#)
- void [SetSampleRate \(float sample_rate\)](#)

12.7.1 Detailed Description

Hardware Interface for control inputs

Primarily designed for ADC input controls such as
potentiometers, and control voltage.

Author

Stephen Hensley

Date

November 2019

12.7.2 Constructor & Destructor Documentation

12.7.2.1 AnalogControl()

```
daisy::AnalogControl::AnalogControl () [inline]
```

Constructor

12.7.2.2 ~AnalogControl()

```
daisy::AnalogControl::~AnalogControl () [inline]
```

destructor

12.7.3 Member Function Documentation

12.7.3.1 GetRawFloat()

```
float daisy::AnalogControl::GetRawFloat () [inline]
```

Returns a normalized float value representing the current ADC value.

12.7.3.2 GetRawValue()

```
uint16_t daisy::AnalogControl::GetRawValue () [inline]
```

Returns the raw unsigned 16-bit value from the ADC

12.7.3.3 Init()

```
void daisy::AnalogControl::Init (
    uint16_t * adcptr,
    float sr,
    bool flip = false,
    bool invert = false,
    float slew_seconds = 0.002f )
```

Initializes the control

Parameters

<i>*adcptr</i>	is a pointer to the raw adc read value – This can be acquired with <code>dsy_adc_get_rawptr()</code> , or <code>dsy_adc_get_mux_rawptr()</code>
<i>sr</i>	is the samplerate in Hz that the Process function will be called at.
<i>flip</i>	determines whether the input is flipped (i.e. $1.f - \text{input}$) or not before being processed.
<i>invert</i>	determines whether the input is inverted (i.e. $-1.f * \text{input}$) or not before being processed.
<i>slew_seconds</i>	is the slew time in seconds that it takes for the control to change to a new value.

12.7.3.4 InitBipolarCv()

```
void daisy::AnalogControl::InitBipolarCv (
    uint16_t * adcptr,
    float sr )
```

This initializes the [AnalogControl](#) for a -5V to 5V inverted input All of the Init details are the same otherwise

Parameters

<i>*adcptr</i>	Pointer to analog digital converter
<i>sr</i>	Audio engine sample rate

12.7.3.5 Process()

```
float daisy::AnalogControl::Process ( )
```

Filters, and transforms a raw ADC read into a normalized range. this should be called at the rate of specified by samplerate at Init time.

Default Initializations will return 0.0 -> 1.0 Bi-polar CV inputs will return -1.0 -> 1.0

12.7.3.6 SetCoeff()

```
void daisy::AnalogControl::SetCoeff (
    float val ) [inline]
```

Directly set the Coefficient of the one pole smoothing filter.

12.7.3.7 SetSampleRate()

```
void daisy::AnalogControl::SetSampleRate (
    float sample_rate )
```

Set a new sample rate after the ctrl has been initialized

Parameters

<code>sample_rate</code>	New update rate for the switch in hz
--------------------------	--------------------------------------

12.7.3.8 Value()

```
float daisy::AnalogControl::Value ( ) const [inline]
```

Returns the current stored value, without reprocessing

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

- `src/hid/ctrl.h`

12.8 daisy::AudioHandle Class Reference

Classes

- struct [Config](#)

Public Types

- enum class [Result](#) { [OK](#) , [ERR](#) }
- typedef const float *const * [InputBuffer](#)
- typedef float ** [OutputBuffer](#)
- typedef void(* [AudioCallback](#)) ([InputBuffer](#) in, [OutputBuffer](#) out, `size_t` size)
- typedef const float * [InterleavingInputBuffer](#)
- typedef float * [InterleavingOutputBuffer](#)
- typedef void(* [InterleavingAudioCallback](#)) ([InterleavingInputBuffer](#) in, [InterleavingOutputBuffer](#) out, `size_t` size)

Public Member Functions

- [AudioHandle](#) (const [AudioHandle](#) &other)=default
- [AudioHandle](#) & [operator=](#) (const [AudioHandle](#) &other)=default
- [Result](#) [Init](#) (const [Config](#) &config, [SaiHandle](#) sai)
- [Result](#) [Init](#) (const [Config](#) &config, [SaiHandle](#) sai1, [SaiHandle](#) sai2)
- [Result](#) [Delinit](#) ()
- const [Config](#) & [GetConfig](#) () const
- `size_t` [GetChannels](#) () const
- float [GetSampleRate](#) ()
- [Result](#) [SetSampleRate](#) ([SaiHandle::Config::SampleRate](#) samplerate)
- [Result](#) [SetBlockSize](#) (`size_t` size)
- [Result](#) [SetPostGain](#) (float val)
- [Result](#) [Start](#) ([AudioCallback](#) callback)
- [Result](#) [Start](#) ([InterleavingAudioCallback](#) callback)
- [Result](#) [Stop](#) ()
- [Result](#) [ChangeCallback](#) ([AudioCallback](#) callback)
- [Result](#) [ChangeCallback](#) ([InterleavingAudioCallback](#) callback)

12.8.1 Member Typedef Documentation

12.8.1.1 AudioCallback

```
typedef void(* daisy::AudioHandle::AudioCallback) (InputBuffer in, OutputBuffer out, size_t size)
```

Type for a Non-Interleaving audio callback Non-Interleaving audio callbacks in daisy will be of this type

12.8.1.2 InputBuffer

```
typedef const float* const* daisy::AudioHandle::InputBuffer
```

Non-Interleaving input buffer Buffer arranged by float[chn][sample] const so that the user can't modify the input

12.8.1.3 InterleavingAudioCallback

```
typedef void(* daisy::AudioHandle::InterleavingAudioCallback) (InterleavingInputBuffer in, InterleavingOutputBuffer out, size_t size)
```

Interleaving Audio Callback Interleaving audio callbacks in daisy must be of this type

12.8.1.4 InterleavingInputBuffer

```
typedef const float* daisy::AudioHandle::InterleavingInputBuffer
```

Interleaving Input buffer audio is prepared as { L0, R0, L1, R1, . . . LN, RN } this is const, as the user shouldn't modify it

12.8.1.5 InterleavingOutputBuffer

```
typedef float* daisy::AudioHandle::InterleavingOutputBuffer
```

Interleaving Output buffer audio is prepared as { L0, R0, L1, R1, . . . LN, RN }

12.8.1.6 OutputBuffer

```
typedef float** daisy::AudioHandle::OutputBuffer
```

Non-Interleaving output buffer Arranged by float[chn][sample]

12.8.2 Member Function Documentation

12.8.2.1 ChangeCallback() [1/2]

```
Result daisy::AudioHandle::ChangeCallback (
    AudioCallback callback )
```

Immediatley changes the audio callback to the non-interleaving callback passed in.

12.8.2.2 ChangeCallback() [2/2]

```
Result daisy::AudioHandle::ChangeCallback (
    InterleavingAudioCallback callback )
```

Immediatley changes the audio callback to the interleaving callback passed in.

12.8.2.3 DeInit()

```
Result daisy::AudioHandle::DeInit ( )
```

Stops and deinitializes audio.

12.8.2.4 GetChannels()

```
size_t daisy::AudioHandle::GetChannels ( ) const
```

Returns the number of channels of audio.

When using a single SAI this returns 2, when using two SAI it returns 4 If no SAI is initialized this returns 0

Eventually when we add non-standard I2S for each SAI this will be work differently

12.8.2.5 GetConfig()

```
const Config& daisy::AudioHandle::GetConfig ( ) const
```

Returns the Global Configuration struct for the Audio

12.8.2.6 GetSampleRate()

```
float daisy::AudioHandle::GetSampleRate ( )
```

Returns the Samplerate as a float

12.8.2.7 Init() [1/2]

```
Result daisy::AudioHandle::Init (
    const Config & config,
    SaiHandle sai )
```

Initializes audio to run using a single SAI configured in Stereo I2S mode.

12.8.2.8 `Init()` [2/2]

```
Result daisy::AudioHandle::Init (
    const Config & config,
    SaiHandle sail,
    SaiHandle sai2 )
```

Initializes audio to run using two SAI, each configured in Stereo I2S mode.

12.8.2.9 `SetBlockSize()`

```
Result daisy::AudioHandle::SetBlockSize (
    size_t size )
```

Sets the block size after initialization, and updates the internal configuration struct. Get BlockSize and other details via the GetConfig

12.8.2.10 `SetPostGain()`

```
Result daisy::AudioHandle::SetPostGain (
    float val )
```

Sets the amount of gain adjustment to perform before and after callback. useful if the hardware has additional headroom, and the nominal value shouldn't be 1.0

Parameters

<code>val</code>	Gain adjustment amount. The hardware will clip at the reciprocal of this value.
------------------	---

12.8.2.11 `SetSampleRate()`

```
Result daisy::AudioHandle::SetSampleRate (
    SaiHandle::Config::SampleRate samplerate )
```

Sets the samplerate, and reinitializes the sai as needed.

12.8.2.12 `Start()` [1/2]

```
Result daisy::AudioHandle::Start (
    AudioCallback callback )
```

Starts the Audio using the non-interleaving callback.

12.8.2.13 `Start()` [2/2]

```
Result daisy::AudioHandle::Start (
    InterleavingAudioCallback callback )
```

Starts the Audio using the interleaving callback. For now only two channels are supported via this method.

12.8.2.14 Stop()

```
Result daisy::AudioHandle::Stop ( )
```

Stop the Audio

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

- src/hid/audio.h

12.9 daisy::BlockingSpiTransport Class Reference

Classes

- struct [Config](#)

Public Types

- enum class [Result { OK , ERR }](#)

Public Member Functions

- void [Init \(Config config\)](#)
- bool [Ready \(\)](#)
- [BlockingSpiTransport::Result Transmit \(uint8_t *buff, size_t size, uint32_t wait_us\)](#)
- [BlockingSpiTransport::Result TransmitAndReceive \(uint8_t *tx_buff, uint8_t *rx_buff, size_t size\)](#)

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

- src/dev/max11300.h

12.10 daisy::ButtonMonitor< BackendType, numButtons > Class Template Reference

Public Member Functions

- void [Init \(UiEventQueue &queueToAddEventsTo, BackendType &backend, uint16_t debounceTimeoutMs=50, uint32_t doubleClickTimeoutMs=500, uint32_t retriggerTimeoutMs=2000, uint32_t retriggerPeriodMs=50\)](#)
- void [Process \(\)](#)
- bool [IsButtonPressed \(uint16_t buttonId\) const](#)
- BackendType & [GetBackend \(\)](#)
- uint16_t [GetNumButtonsMonitored \(\) const](#)

12.10.1 Member Function Documentation

12.10.1.1 GetBackend()

```
template<typename BackendType , uint32_t numButtons>
BackendType& daisy::ButtonMonitor< BackendType, numButtons >::GetBackend ( ) [inline]
```

Returns the BackendType that is used by the monitor.

12.10.1.2 GetNumButtonsMonitored()

```
template<typename BackendType , uint32_t numButtons>
uint16_t daisy::ButtonMonitor< BackendType, numButtons >::GetNumButtonsMonitored ( ) const
[inline]
```

Returns the number of buttons that are monitored by this class.

12.10.1.3 Init()

```
template<typename BackendType , uint32_t numButtons>
void daisy::ButtonMonitor< BackendType, numButtons >::Init (
    UiEventQueue & queueToAddEventsTo,
    BackendType & backend,
    uint16_t debounceTimeoutMs = 50,
    uint32_t doubleClickTimeoutMs = 500,
    uint32_t retriggerTimeoutMs = 2000,
    uint32_t retriggerPeriodMs = 50 ) [inline]
```

Initialises the [ButtonMonitor](#).

Parameters

<i>queueToAddEventsTo</i>	The UiEventQueue to which events should be posted.
<i>backend</i>	The backend that supplies the current state of each button.
<i>debounceTimeoutMs</i>	A event is posted to the queue if the button state doesn't change for <i>debounceTimeoutMs</i> . Can be 0 to disable debouncing.
<i>doubleClickTimeoutMs</i>	The timeout for detecting double clicks.
<i>retriggerTimeoutMs</i>	The timeout after which a button will be retrigged when held down. 0 to disable retrigging.
<i>retriggerPeriodMs</i>	The speed with which a button will be retrigged when held down.

12.10.1.4 IsButtonPressed()

```
template<typename BackendType , uint32_t numButtons>
bool daisy::ButtonMonitor< BackendType, numButtons >::IsButtonPressed (
    uint16_t buttonId ) const [inline]
```

Returns true, if the given button is currently pressed.

Parameters

<i>buttonId</i>	The unique ID of the button (< numButtons)
-----------------	--

12.10.1.5 Process()

```
template<typename BackendType , uint32_t numButtons>
void daisy::ButtonMonitor< BackendType, numButtons >::Process ( ) [inline]
```

Checks the value of each button and generates messages for the UIEventQueue. Call this at regular intervals, ideally from your main() idle loop.

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

- src/ui/ButtonMonitor.h

12.11 daisy::ChannelPressureEvent Struct Reference

```
#include <MidiEvent.h>
```

Public Attributes

- int [channel](#)
- uint8_t [pressure](#)

12.11.1 Detailed Description

Struct containing pressure (aftertouch), for a given channel. Can be made from [MidiEvent](#)

12.11.2 Member Data Documentation**12.11.2.1 channel**

```
int daisy::ChannelPressureEvent::channel  
&
```

12.11.2.2 pressure

```
uint8_t daisy::ChannelPressureEvent::pressure
```

&

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

- src/hid/MidiEvent.h

12.12 daisy::Color Class Reference

```
#include <color.h>
```

Public Types

- enum PresetColor {
 RED , GREEN , BLUE , WHITE ,
 PURPLE , CYAN , GOLD , OFF ,
 LAST }

Public Member Functions

- void Init (PresetColor c)
- void Init (float red, float green, float blue)
- float Red () const
- float Green () const
- float Blue () const

12.12.1 Detailed Description

Class for handling simple colors

12.12.2 Member Enumeration Documentation

12.12.2.1 PresetColor

```
enum daisy::Color::PresetColor
```

List of colors that have a preset RGB value

Enumerator

RED	&
GREEN	&
BLUE	&
WHITE	&
PURPLE	&
CYAN	&
GOLD	&
OFF	&
LAST	&

12.12.3 Member Function Documentation**12.12.3.1 Blue()**

```
float daisy::Color::Blue () const [inline]
```

Returns the 0-1 value for Blue

12.12.3.2 Green()

```
float daisy::Color::Green () const [inline]
```

Returns the 0-1 value for Green

12.12.3.3 Init() [1/2]

```
void daisy::Color::Init (
    float red,
    float green,
    float blue )
```

Initializes the [Color](#) with a specific RGB value red, green, and blue should be floats between 0 and 1

Parameters

<i>red</i>	Red value
<i>green</i>	Green value
<i>blue</i>	Blue value

12.12.3.4 `Init()` [2/2]

```
void daisy::Color::Init (
    PresetColor c )
```

Initializes the [Color](#) with a given preset.

Parameters

<code>c</code>	Color to init to
----------------	----------------------------------

12.12.3.5 `Red()`

```
float daisy::Color::Red ( ) const [inline]
```

Returns the 0-1 value for Red

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

- src/util/color.h

12.13 `daisy::AudioHandle::Config` Struct Reference

```
#include <audio.h>
```

Public Attributes

- `size_t blocksize`
- [SaiHandle::Config::SampleRate](#) `samplerate`
- `float postgain`

12.13.1 Detailed Description

Manually configurable details about the Audio Engine TODO: Figure out how to get samplerate in here.

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

- src/hid/audio.h

12.14 `daisy::BlockingSpiTransport::Config` Struct Reference

```
#include <max11300.h>
```

Public Member Functions

- void [Defaults \(\)](#)

Public Attributes

- [SpiHandle::Config spi_config](#)

12.14.1 Detailed Description

Transport configuration struct for the MAX11300

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

- [src/dev/max11300.h](#)

12.15 daisy::DacHandle::Config Struct Reference

```
#include <dac.h>
```

Public Attributes

- `uint32_t target_samplerate`
- [Channel chn](#)
- [Mode mode](#)
- [BitDepth bitdepth](#)
- [BufferState buff_state](#)

12.15.1 Detailed Description

Configuration structure for initializing the DAC structure.

12.15.2 Member Data Documentation

12.15.2.1 target_samplerate

```
uint32_t daisy::DacHandle::Config::target_samplerate
```

Target Samplerate in Hz used to configure the internal timebase for DMA mode. Does nothing in POLLING mode. If the value is 0 at Init time this will default to 48000Hz otherwise the driver will attempt meet the target.

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

- [src/per/dac.h](#)

12.16 daisy::FatFSInterface::Config Struct Reference

```
#include <fatfs.h>
```

Public Types

- enum [Media](#) : uint8_t { **MEDIA_SD** = 0x01 , **MEDIA_USB** = 0x02 }

Public Attributes

- uint8_t **media**

12.16.1 Detailed Description

[Config](#) structure for configuring Daisy to FatFS

12.16.2 Member Enumeration Documentation

12.16.2.1 Media

```
enum daisy::FatFSInterface::Config::Media : uint8_t
```

Selected Media that will be linked to FatFS These values can be OR'd together when multiple volumes are desired i.e. config.media = Media::SD | Media::USBH;

When mounting multiple volumes, [ffconf.h](#) must have _VOLUMES set to an appropriate value.

FatFS will register multiple volumes in the order of the enum, the first registered class will mount at "0:/", the second registered class will mount at "1:/", and so on

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

- src/sys/fatfs.h

12.17 daisy::GPIO::Config Struct Reference

Configuration for a given [GPIO](#).

```
#include <gpio.h>
```

Public Member Functions

- [Config \(\)](#)

Public Attributes

- [Pin pin](#)
- [Mode mode](#)
- [Pull pull](#)
- [Speed speed](#)

12.17.1 Detailed Description

Configuration for a given [GPIO](#).

12.17.2 Constructor & Destructor Documentation

12.17.2.1 Config()

```
daisy::GPIO::Config::Config ( ) [inline]
```

Constructor with no arguments will prepare an invalid [GPIO](#) set as an input, with no pullup.

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

- [src/per/gpio.h](#)

12.18 daisy::I2CHandle::Config Struct Reference

```
#include <i2c.h>
```

Public Types

- enum class [Mode](#) { [I2C_MASTER](#) , [I2C_SLAVE](#) }
- enum class [Peripheral](#) { [I2C_1](#) = 0 , [I2C_2](#) , [I2C_3](#) , [I2C_4](#) }
- enum class [Speed](#) { [I2C_100KHZ](#) , [I2C_400KHZ](#) , [I2C_1MHZ](#) }

Public Attributes

- [Peripheral periph](#)
- struct {
 [dsy_gpio_pin scl](#)
 [dsy_gpio_pin sda](#)
} [pin_config](#)
- [Speed speed](#)
- [Mode mode](#)
- [uint8_t address](#) = 0x10

12.18.1 Detailed Description

Contains settings for initialising an I2C interface.

12.18.2 Member Enumeration Documentation

12.18.2.1 Mode

```
enum daisy::I2CHandle::Config::Mode [strong]
```

Specifies whether the interface will operate in master or slave mode.

12.18.2.2 Peripheral

```
enum daisy::I2CHandle::Config::Peripheral [strong]
```

Specifies the internal peripheral to use (these are mapped to different pins on the hardware).

Enumerator

I2C \leftarrow _1	&
I2C \leftarrow _2	&
I2C \leftarrow _3	&
I2C \leftarrow _4	&

12.18.2.3 Speed

```
enum daisy::I2CHandle::Config::Speed [strong]
```

Rate at which the clock/data will be sent/received. The device being used will have maximum speeds. 1MHZ Mode is currently 886kHz

Enumerator

I2C_100KHZ	&
I2C_400KHZ	&
I2C_1MHZ	&

12.18.3 Member Data Documentation

12.18.3.1 address

```
uint8_t daisy::I2CHandle::Config::address = 0x10  
&
```

12.18.3.2 mode

```
Mode daisy::I2CHandle::Config::mode  
&
```

12.18.3.3 periph

```
Peripheral daisy::I2CHandle::Config::periph  
&
```

12.18.3.4

```
struct { ... } daisy::I2CHandle::Config::pin_config  
&
```

12.18.3.5 scl

```
dsy_gpio_pin daisy::I2CHandle::Config::scl  
&
```

12.18.3.6 sda

```
dsy_gpio_pin daisy::I2CHandle::Config::sda  
&
```

12.18.3.7 speed

```
Speed daisy::I2CHandle::Config::speed  
&
```

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

- src/per/i2c.h

12.19 daisy::LcdHD44780::Config Struct Reference

Public Attributes

- bool **cursor_on**
- bool **cursor_blink**
- **dsy_gpio_pin rs**
- **dsy_gpio_pin en**
- **dsy_gpio_pin d4**
- **dsy_gpio_pin d5**
- **dsy_gpio_pin d6**
- **dsy_gpio_pin d7**

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

- src/dev/lcd_hd44780.h

12.20 daisy::MAX11300Driver< Transport >::Config Struct Reference

Public Member Functions

- void **Defaults ()**

Public Attributes

- Transport::Config **transport_config**

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

- src/dev/max11300.h

12.21 daisy::MidiHandler< Transport >::Config Struct Reference

Public Attributes

- Transport::Config **transport_config**

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

- src/hid/midi.h

12.22 daisy::MidiUartTransport::Config Struct Reference

Public Attributes

- UartHandler::Config::Peripheral **periph**
- [dsy_gpio_pin rx](#)
- [dsy_gpio_pin tx](#)

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

- src/hid/midi.h

12.23 daisy::MidiUsbTransport::Config Struct Reference

Public Types

- enum **Periph** { **INTERNAL** = 0 , **EXTERNAL** }

Public Attributes

- Periph **periph**

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

- src/hid/usb_midi.h

12.24 daisy::OledDisplay< DisplayDriver >::Config Struct Reference

Public Attributes

- DisplayDriver::Config **driver_config**

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

- src/hid/disp/oled_display.h

12.25 daisy::QSPIHandle::Config Struct Reference

```
#include <qspi.h>
```

Public Types

- enum `Device` { `IS25LP080D` , `IS25LP064A` , `DEVICE_LAST` }
- enum `Mode` { `MEMORY_MAPPED` , `INDIRECT_POLLING` , `MODE_LAST` }

Public Attributes

- ```
struct {
 dsy_gpio_pin io0
 dsy_gpio_pin io1
 dsy_gpio_pin io2
 dsy_gpio_pin io3
 dsy_gpio_pin clk
 dsy_gpio_pin ncs
} pin_config
```

  - `Device device`
  - `Mode mode`

### 12.25.1 Detailed Description

Configuration structure for interfacing with QSPI Driver

### 12.25.2 Member Enumeration Documentation

#### 12.25.2.1 Device

```
enum daisy::QSPIHandle::Config::Device
```

Flash Devices supported. (Both of these are more-or-less the same, just different sizes).

Enumerator

|                          |   |
|--------------------------|---|
| <code>IS25LP080D</code>  | & |
| <code>IS25LP064A</code>  | & |
| <code>DEVICE_LAST</code> | & |

#### 12.25.2.2 Mode

```
enum daisy::QSPIHandle::Config::Mode
```

Modes of operation. Memory Mapped mode: QSPI configured so that the QSPI can be read from starting address 0x90000000. Writing is not possible in this mode.

Indirect Polling mode: Device driver enabled.

Enumerator

|                  |   |
|------------------|---|
| MEMORY_MAPPED    | & |
| INDIRECT_POLLING | & |

### 12.25.3 Member Data Documentation

#### 12.25.3.1 clk

```
dsy_gpio_pin daisy::QSPIHandle::Config::clk
&
```

#### 12.25.3.2 io0

```
dsy_gpio_pin daisy::QSPIHandle::Config::io0
&
```

#### 12.25.3.3 io1

```
dsy_gpio_pin daisy::QSPIHandle::Config::io1
&
```

#### 12.25.3.4 io2

```
dsy_gpio_pin daisy::QSPIHandle::Config::io2
&
```

#### 12.25.3.5 io3

```
dsy_gpio_pin daisy::QSPIHandle::Config::io3
&
```

#### 12.25.3.6 ncs

```
dsy_gpio_pin daisy::QSPIHandle::Config::ncs
&
```

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

- src/per/qspi.h

## 12.26 daisy::SaiHandle::Config Struct Reference

```
#include <sai.h>
```

### Public Types

- enum class **Peripheral** { **SAI\_1** , **SAI\_2** }
- enum class **SampleRate** {  
    **SAI\_8KHZ** , **SAI\_16KHZ** , **SAI\_32KHZ** , **SAI\_48KHZ** ,  
    **SAI\_96KHZ** }
- enum class **BitDepth** { **SAI\_16BIT** , **SAI\_24BIT** , **SAI\_32BIT** }
- enum class **Sync** { **MASTER** , **SLAVE** }
- enum class **Direction** { **TRANSMIT** , **RECEIVE** }

### Public Attributes

- **Peripheral periph**
- 
- struct {  
    dsy\_gpio\_pin **mclk**  
    dsy\_gpio\_pin **fs**  
    dsy\_gpio\_pin **sck**  
    dsy\_gpio\_pin **sa**  
    dsy\_gpio\_pin **sb**  
} **pin\_config**
- **SampleRate sr**
- **BitDepth bit\_depth**
- **Sync a\_sync**
- **Sync b\_sync**
- **Direction a\_dir**
- **Direction b\_dir**

### 12.26.1 Detailed Description

Contains settings for initialising an SAI Interface

### 12.26.2 Member Enumeration Documentation

#### 12.26.2.1 BitDepth

```
enum daisy::SaiHandle::Config::BitDepth [strong]
```

Bit Depth that the hardware expects to be transferred to/from the device.

### 12.26.2.2 Direction

```
enum daisy::SaiHandle::Config::Direction [strong]
```

Specifies the direction for each peripheral block.

### 12.26.2.3 Peripheral

```
enum daisy::SaiHandle::Config::Peripheral [strong]
```

Specifies the internal peripheral to use (mapped to different hardware pins)

### 12.26.2.4 SampleRate

```
enum daisy::SaiHandle::Config::SampleRate [strong]
```

Rate at which samples will be streaming to/from the device.

### 12.26.2.5 Sync

```
enum daisy::SaiHandle::Config::Sync [strong]
```

Specifies whether a particular block is the master or the slave. If both are set to slave, no MCLK signal will be used, and it is expected that the codec will have its own xtal.

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

- src/per/sai.h

## 12.27 daisy::SdmmcHandler::Config Struct Reference

### Public Member Functions

- void [Defaults \(\)](#)

### Public Attributes

- [Speed speed](#)
- [BusWidth width](#)
- bool [clock\\_powersave](#)

### 12.27.1 Member Function Documentation

### 12.27.1.1 Defaults()

```
void daisy::SdmmcHandler::Config::Defaults () [inline]
```

Configures settings to their default settings

## 12.27.2 Member Data Documentation

### 12.27.2.1 clock\_powersave

```
bool daisy::SdmmcHandler::Config::clock_powersave
```

When true, the clock will stop between transfers to save power.

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

- src/per/sdmmc.h

## 12.28 daisy::ShiftRegister4021< num\_daisychained, num\_parallel >::Config Struct Reference

```
#include <sr_4021.h>
```

### Public Attributes

- [dsy\\_gpio\\_pin clk](#)
- [dsy\\_gpio\\_pin latch](#)
- [dsy\\_gpio\\_pin data \[num\\_parallel\]](#)

### 12.28.1 Detailed Description

```
template<size_t num_daisychained = 1, size_t num_parallel = 1>
struct daisy::ShiftRegister4021< num_daisychained, num_parallel >::Config
```

Configuration Structure for handling the pin setting of the device

## 12.28.2 Member Data Documentation

### 12.28.2.1 clk

```
template<size_t num_daisychained = 1, size_t num_parallel = 1>
dsy_gpio_pin daisy::ShiftRegister4021< num_daisychained, num_parallel >::Config::clk
```

Clock pin to attach to pin 10 of device(s)

### 12.28.2.2 data

```
template<size_t num_daisychained = 1, size_t num_parallel = 1>
dsy_gpio_pin daisy::ShiftRegister4021< num_daisychained, num_parallel >::Config::data[num_←
parallel]
```

Data Pin(s)

### 12.28.2.3 latch

```
template<size_t num_daisychained = 1, size_t num_parallel = 1>
dsy_gpio_pin daisy::ShiftRegister4021< num_daisychained, num_parallel >::Config::latch
```

Latch pin to attach to pin 9 of device(s)

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

- src/dev/sr\_4021.h

## 12.29 daisy::SpiHandle::Config Struct Reference

### Public Types

- enum class **Peripheral** {
 SPI\_1 , SPI\_2 , SPI\_3 , SPI\_4 ,
 SPI\_5 , SPI\_6 }
- enum class **Mode** { **MASTER** , **SLAVE** }
- enum class **Direction** { TWO\_LINES , TWO\_LINES\_TX\_ONLY , TWO\_LINES\_RX\_ONLY , ONE\_LINE }
- enum class **ClockPolarity** { LOW , HIGH }
- enum class **ClockPhase** { ONE\_EDGE , TWO\_EDGE }
- enum class **NSS** { SOFT , HARD\_INPUT , HARD\_OUTPUT }
- enum class **BaudPrescaler** {
 PS\_2 , PS\_4 , PS\_8 , PS\_16 ,
 PS\_32 , PS\_64 , PS\_128 , PS\_256 }

## Public Attributes

- struct {  
    dsy\_gpio\_pin sclk  
    dsy\_gpio\_pin miso  
    dsy\_gpio\_pin mosi  
    dsy\_gpio\_pin nss  
} pin\_config
- Peripheral **periph**
- Mode **mode**
- Direction **direction**
- unsigned long **datasize**
- ClockPolarity **clock\_polarity**
- ClockPhase **clock\_phase**
- NSS **nss**
- BaudPrescaler **baud\_prescaler**

### 12.29.1 Member Data Documentation

#### 12.29.1.1 miso

```
dsy_gpio_pin daisy::SpiHandle::Config::miso
&
```

#### 12.29.1.2 mosi

```
dsy_gpio_pin daisy::SpiHandle::Config::mosi
&
```

#### 12.29.1.3 nss

```
dsy_gpio_pin daisy::SpiHandle::Config::nss
&
```

#### 12.29.1.4 sclk

```
dsy_gpio_pin daisy::SpiHandle::Config::sclk
&
```

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

- src/per/spi.h

## 12.30 daisy::SSD130x4WireSpiTransport::Config Struct Reference

### Public Member Functions

- void **Defaults** ()

### Public Attributes

- struct {  
 dsy\_gpio\_pin dc  
 dsy\_gpio\_pin reset  
} pin\_config

#### 12.30.1 Member Data Documentation

##### 12.30.1.1 dc

```
dsy_gpio_pin daisy::SSD130x4WireSpiTransport::Config::dc
&
```

##### 12.30.1.2 reset

```
dsy_gpio_pin daisy::SSD130x4WireSpiTransport::Config::reset
&
```

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

- src/dev/oled\_ssdl30x.h

## 12.31 daisy::SSD130xDriver< width, height, Transport >::Config Struct Reference

### Public Attributes

- Transport::Config **transport\_config**

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

- src/dev/oled\_ssdl30x.h

## 12.32 daisy::SSD130xI2CTransport::Config Struct Reference

### Public Member Functions

- void **Defaults** ()

### Public Attributes

- I2CHandle::Config **i2c\_config**
- uint8\_t **i2c\_address**

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

- src/dev/oled\_ssdl30x.h

## 12.33 daisy::System::Config Struct Reference

```
#include <system.h>
```

### Public Types

- enum class **SysClkFreq** { **FREQ\_400MHZ** , **FREQ\_480MHZ** }

### Public Member Functions

- void **Defaults** ()
- void **Boost** ()

### Public Attributes

- **SysClkFreq cpu\_freq**
- bool **use\_dcache**
- bool **use\_icache**
- bool **skip\_clocks**

### 12.33.1 Detailed Description

Contains settings for initializing the [System](#)

### 12.33.2 Member Enumeration Documentation

### 12.33.2.1 SysClkFreq

```
enum daisy::System::Config::SysClkFreq [strong]
```

Specifies the system clock frequency that feeds APB/AHB clocks, etc.

## 12.33.3 Member Function Documentation

### 12.33.3.1 Boost()

```
void daisy::System::Config::Boost () [inline]
```

Method to call on the struct to set to boost mode: CPU Freq set to 480MHz Cache Enabled

### 12.33.3.2 Defaults()

```
void daisy::System::Config::Defaults () [inline]
```

Method to call on the struct to set to defaults CPU Freq set to 400MHz Cache Enabled

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

- src/sys/system.h

## 12.34 daisy::TimerHandle::Config Struct Reference

### Public Types

- enum class **Peripheral** { **TIM\_2** = 0 , **TIM\_3** , **TIM\_4** , **TIM\_5** }
- enum class **CounterDir** { **UP** = 0 , **DOWN** }

### Public Attributes

- Peripheral** **periph**
- CounterDir** **dir**

## 12.34.1 Member Enumeration Documentation

### 12.34.1.1 CounterDir

```
enum daisy::TimerHandle::Config::CounterDir [strong]
```

Direction of the auto-reload counter. TODO: Add support for the various versions of Up/Down counters.

### 12.34.1.2 Peripheral

```
enum daisy::TimerHandle::Config::Peripheral [strong]
```

Hardwaare Timer to configure, and use.

## Enumerator

|                  |                |
|------------------|----------------|
| TIM <sub>2</sub> | 32-bit counter |
| TIM <sub>3</sub> | 16-bit counter |
| TIM <sub>4</sub> | 16-bit counter |
| TIM <sub>5</sub> | 32-bit counter |

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

- src/per/tim.h

## 12.35 daisy::UartHandler::Config Struct Reference

### Public Types

- enum class **Peripheral** {
 USART\_1, USART\_2, USART\_3, UART\_4,
 UART\_5, USART\_6, UART\_7, UART\_8,
 LPUART\_1
 }
- enum class **StopBits** { BITS\_0\_5, BITS\_1, BITS\_1\_5, BITS\_2 }
- enum class **Parity** { NONE, EVEN, ODD }
- enum class **Mode** { RX, TX, TX\_RX }
- enum class **WordLength** { BITS\_7, BITS\_8, BITS\_9 }

### Public Attributes

- struct {
 [dsy\\_gpio\\_pin tx](#)
[dsy\\_gpio\\_pin rx](#)
} [pin\\_config](#)
- Peripheral [periph](#)
- StopBits [stopbits](#)
- Parity [parity](#)
- Mode [mode](#)
- WordLength [wordlength](#)
- uint32\_t [baudrate](#)

### 12.35.1 Member Data Documentation

### 12.35.1.1

```
struct { ... } daisy::UartHandler::Config::pin_config
&
```

### 12.35.1.2 rx

```
dsy_gpio_pin daisy::UartHandler::Config::rx
&
```

### 12.35.1.3 tx

```
dsy_gpio_pin daisy::UartHandler::Config::tx
&
```

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

- src/per/uart.h

## 12.36 daisy::USBHostHandle::Config Struct Reference

```
#include <usb_host.h>
```

### 12.36.1 Detailed Description

Configuration structure for interfacing with MSD Driver

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

- src/hid/usb\_host.h

## 12.37 daisy::WavWriter< transfer\_size >::Config Struct Reference

```
#include <WavWriter.h>
```

### Public Attributes

- float **samplerate**
- int32\_t **channels**
- int32\_t **bitspersample**

### 12.37.1 Detailed Description

```
template<size_t transfer_size>
struct daisy::WavWriter< transfer_size >::Config
```

Configuration structure for the wave writer.

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

- src/util/WavWriter.h

## 12.38 daisy::Wm8731::Config Struct Reference

```
#include <codec_wm8731.h>
```

### Public Types

- enum class [Format](#) { **MSB\_FIRST\_RJ** = 0x00 , **MSB\_FIRST\_LJ** = 0x01 , **I2S** = 0x02 , **DSP** = 0x03 }
- enum class [WordLength](#) { **BITS\_16** = (0x00 << 2) , **BITS\_20** = (0x01 << 2) , **BITS\_24** = (0x02 << 2) ,  
**BITS\_32** = (0x03 << 2) }

### Public Member Functions

- void [Defaults](#) ()

### Public Attributes

- bool [mcu\\_is\\_master](#)
- bool [lr\\_swap](#)
- bool [csb\\_pin\\_state](#)
- [Format](#) [fmt](#)
- [WordLength](#) [wl](#)

### 12.38.1 Detailed Description

Configuration struct for use in initializing the device. For now, only 48kHz is supported. USB Mode is also not yet supported.

### 12.38.2 Member Enumeration Documentation

### 12.38.2.1 Format

```
enum daisy::Wm8731::Config::Format [strong]
```

Sets the communication format used

### 12.38.2.2 WordLength

```
enum daisy::Wm8731::Config::WordLength [strong]
```

Defines the size of a sample in bits This is for communication only, the device processes audio at 24-bits, and the strips/pads bits to send to the processor.

## 12.38.3 Member Function Documentation

### 12.38.3.1 Defaults()

```
void daisy::Wm8731::Config::Defaults () [inline]
```

Sets the following config: MCU is master = true L/R Swap = false CSB Pin state = false Format = MSB First LJ WordLength = 24-bit

## 12.38.4 Member Data Documentation

### 12.38.4.1 csb\_pin\_state

```
bool daisy::Wm8731::Config::csb_pin_state
```

Set true if tied high, and false if tied low. determines the I2C address for communicating with the device

### 12.38.4.2 lr\_swap

```
bool daisy::Wm8731::Config::lr_swap
```

Sets whether the left/right channels are swapped or not.

### 12.38.4.3 mcu\_is\_master

```
bool daisy::Wm8731::Config::mcu_is_master
```

Sets the device to slave mode if true, and master mode if false.

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

- src/dev/codec\_wm8731.h

## 12.39 daisy::ControlChangeEvent Struct Reference

```
#include <MidiEvent.h>
```

### Public Attributes

- int [channel](#)
- uint8\_t [control\\_number](#)
- uint8\_t [value](#)

### 12.39.1 Detailed Description

Struct containing control number, and value for a given channel. Can be made from [MidiEvent](#)

### 12.39.2 Member Data Documentation

#### 12.39.2.1 channel

```
int daisy::ControlChangeEvent::channel
```

&

#### 12.39.2.2 control\_number

```
uint8_t daisy::ControlChangeEvent::control_number
```

&

### 12.39.2.3 value

```
uint8_t daisy::ControlChangeEvent::value
```

&

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

- src/hid/MidiEvent.h

## 12.40 daisy::CpuLoadMeter Class Reference

### Public Member Functions

- void [Init](#) (float sampleRateInHz, int blockSizeInSamples, float smoothingFilterCutoffHz=1.0f)
- void [OnBlockStart](#) ()
- void [OnBlockEnd](#) ()
- float [GetAvgCpuLoad](#) () const
- float [GetMinCpuLoad](#) () const
- float [GetMaxCpuLoad](#) () const
- void [Reset](#) ()

### 12.40.1 Member Function Documentation

#### 12.40.1.1 GetAvgCpuLoad()

```
float daisy::CpuLoadMeter::GetAvgCpuLoad () const [inline]
```

Returns the smoothed average CPU load in the range 0..1

#### 12.40.1.2 GetMaxCpuLoad()

```
float daisy::CpuLoadMeter::GetMaxCpuLoad () const [inline]
```

Returns the maximum CPU load observed since the last call to [Reset\(\)](#).

#### 12.40.1.3 GetMinCpuLoad()

```
float daisy::CpuLoadMeter::GetMinCpuLoad () const [inline]
```

Returns the minimum CPU load observed since the last call to [Reset\(\)](#).

#### 12.40.1.4 Init()

```
void daisy::CpuLoadMeter::Init (
 float sampleRateInHz,
 int blockSizeInSamples,
 float smoothingFilterCutoffHz = 1.0f) [inline]
```

Initializes the [CpuLoadMeter](#) for a particular sample rate and block size.

**Parameters**

|                                |                                                                                                  |
|--------------------------------|--------------------------------------------------------------------------------------------------|
| <i>sampleRateInHz</i>          | The sample rate in Hz                                                                            |
| <i>blockSizeInSamples</i>      | The block size in samples                                                                        |
| <i>smoothingFilterCutoffHz</i> | The cutoff frequency of the smoothing filter that's used to create the average CPU load reading. |

**12.40.1.5 OnBlockEnd()**

```
void daisy::CpuLoadMeter::OnBlockEnd () [inline]
```

Call this at the end of your audio callback

**12.40.1.6 OnBlockStart()**

```
void daisy::CpuLoadMeter::OnBlockStart () [inline]
```

Call this at the beginning of your audio callback

**12.40.1.7 Reset()**

```
void daisy::CpuLoadMeter::Reset () [inline]
```

Resets the minimum, maximum and average load readings.

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

- src/util/CpuLoadMeter.h

**12.41 daisy::AbstractMenu::CustomItem Class Reference**

```
#include <AbstractMenu.h>
```

**Public Member Functions**

- virtual void [Draw](#) ([OneBitGraphicsDisplay](#) &display, int currentIndex, int numItemsTotal, [Rectangle](#) bounds← ToDrawIn, bool isEditing)=0
- virtual bool [CanBeEnteredForEditing](#) () const
- virtual void [ModifyValue](#) (int16\_t increments, uint16\_t stepsPerRevolution, bool isFunctionButtonPressed)
- virtual void [ModifyValue](#) (float valueSliderPosition0To1, bool isFunctionButtonPressed)
- virtual void [OnOkayButton](#) ()

### 12.41.1 Detailed Description

Base class for a custom menu item

### 12.41.2 Member Function Documentation

#### 12.41.2.1 CanBeEnteredForEditing()

```
virtual bool daisy::AbstractMenu::CustomItem::CanBeEnteredForEditing () const [inline],
[virtual]
```

Returns true, if this item can be entered for direct editing of the value.

#### 12.41.2.2 Draw()

```
virtual void daisy::AbstractMenu::CustomItem::Draw (
 OneBitGraphicsDisplay & display,
 int currentIndex,
 int numItemsTotal,
 Rectangle boundsToDrawIn,
 bool isEditing) [pure virtual]
```

Draws the item to a [OneBitGraphicsDisplay](#).

##### Parameters

|                       |                                                                              |
|-----------------------|------------------------------------------------------------------------------|
| <i>display</i>        | The display to draw to                                                       |
| <i>currentIndex</i>   | The index in the menu                                                        |
| <i>numItemsTotal</i>  | The total number of items in the menu                                        |
| <i>boundsToDrawIn</i> | The <a href="#">Rectangle</a> to draw the item into                          |
| <i>isEditing</i>      | True if the enter button was pressed and the value is being edited directly. |

#### 12.41.2.3 ModifyValue() [1/2]

```
virtual void daisy::AbstractMenu::CustomItem::ModifyValue (
 float valueSliderPosition0To1,
 bool isFunctionButtonPressed) [inline], [virtual]
```

Called when the value slider is used to modify the value.

### 12.41.2.4 ModifyValue() [2/2]

```
virtual void daisy::AbstractMenu::CustomItem::ModifyValue (
 int16_t increments,
 uint16_t stepsPerRevolution,
 bool isFunctionButtonPressed) [inline], [virtual]
```

Called when the encoder of the buttons are used to modify the value.

### 12.41.2.5 OnOkayButton()

```
virtual void daisy::AbstractMenu::CustomItem::OnOkayButton () [inline], [virtual]
```

Called when the okay button is pressed (and [CanBeEnteredForEditing\(\)](#) returns false).

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

- src/ui/AbstractMenu.h

## 12.42 daisy::DacHandle Class Reference

```
#include <dac.h>
```

### Classes

- struct [Config](#)

### Public Types

- enum class [Result](#) { **OK** , **ERR** }
- enum class [Channel](#) { **ONE** , **TWO** , **BOTH** }
- enum class [Mode](#) { **POLLING** , **DMA** }
- enum class [BitDepth](#) { **BITS\_8** , **BITS\_12** }
- enum class [BufferState](#) { **ENABLED** , **DISABLED** }
- typedef void(\* [DacCallback](#)) (uint16\_t \*\*out, size\_t size)

### Public Member Functions

- [DacHandle](#) (const [DacHandle](#) &other)=default
- [DacHandle](#) & [operator=](#) (const [DacHandle](#) &other)=default
- [Result](#) [Init](#) (const [Config](#) &config)
- const [Config](#) & [GetConfig](#) () const
- [Result](#) [Start](#) (uint16\_t \*buffer, size\_t size, [DacCallback](#) cb)
- [Result](#) [Start](#) (uint16\_t \*buffer\_1, uint16\_t \*buffer\_2, size\_t size, [DacCallback](#) cb)
- [Result](#) [Stop](#) ()
- [Result](#) [WriteValue](#) ([Channel](#) chn, uint16\_t val)

### 12.42.1 Detailed Description

DAC handle for Built-in DAC Peripheral

For now only Normal Mode is supported, Sample and hold mode provides reduced power consumption, but requires a bit more setup.

For now connecting the DAC through other internal peripherals is also not supported.

Since the DAC channels have dedicated pins we don't need to pass in a pin config like with other modules. However, it is still important to not try to use the DAC pins for anything else. DAC Channel 1 is on PA4, and DAC Channel 2 is on PA5

### 12.42.2 Member Typedef Documentation

#### 12.42.2.1 DacCallback

```
typedef void(* daisy::DacHandle::DacCallback) (uint16_t **out, size_t size)
```

Callback for DMA transfers. This is called every time half of the samples of the buffer are transmitted, and the buffer is ready to be filled again.

The data is organized in arrays per channel, for example if both channels are in use: { {ch1-0, ch1-1, ch1-2 . . . ch1-N}, {ch2-0, ch2-1, ch2-2 . . . ch2-N} }

### 12.42.3 Member Enumeration Documentation

#### 12.42.3.1 BitDepth

```
enum daisy::DacHandle::BitDepth [strong]
```

Sets the number of bits per sample transmitted out of the DAC. The output range will be: 0V - VDDA The resolution will be roughly: bitdepth / (VDDA - 0V)

#### 12.42.3.2 BufferState

```
enum daisy::DacHandle::BufferState [strong]
```

Sets whether the DAC output is buffered for higher drive ability.

#### 12.42.3.3 Channel

```
enum daisy::DacHandle::Channel [strong]
```

Selects which channel(s) will be configured for use.

### 12.42.3.4 Mode

```
enum daisy::DacHandle::Mode [strong]
```

Sets the Mode for the DAC channels.

Polling mode uses the blocking mode to transmit a single value at a time.

DMA mode uses a buffer, and periodically transmits it triggering a callback to fill the buffer when it is ready for more samples.

### 12.42.3.5 Result

```
enum daisy::DacHandle::Result [strong]
```

Return Values for the [DacHandle](#) class

## 12.42.4 Member Function Documentation

### 12.42.4.1 Init()

```
Result daisy::DacHandle::Init (
 const Config & config)
```

Initialize the DAC Peripheral

### 12.42.4.2 Start() [1/2]

```
Result daisy::DacHandle::Start (
 uint16_t * buffer,
 size_t size,
 DacCallback cb)
```

Starts the DAC conversion on the DMA calling the user callback whenever new samples are ready to be filled.

This will return Result::ERR if used when configured to BOTH channels.

### 12.42.4.3 Start() [2/2]

```
Result daisy::DacHandle::Start (
 uint16_t * buffer_1,
 uint16_t * buffer_2,
 size_t size,
 DacCallback cb)
```

If using both channels, use this function to start the DMA transfer for both. The callback will provide an array per-channel to fill.

#### 12.42.4.4 Stop()

```
Result daisy::DacHandle::Stop ()
```

Stops the DAC channel(s).

#### 12.42.4.5 WriteValue()

```
Result daisy::DacHandle::WriteValue (
 Channel chn,
 uint16_t val)
```

Sets and Writes value in Polling Mode Has no effect in DMA mode.

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

- src/per/dac.h

## 12.43 daisy::DaisyField Class Reference

### Public Types

- enum { SW\_1 , SW\_2 , SW\_LAST }
- enum {
 KNOB\_1 , KNOB\_2 , KNOB\_3 , KNOB\_4 ,
 KNOB\_5 , KNOB\_6 , KNOB\_7 , KNOB\_8 ,
 KNOB\_LAST }
- enum {
 CV\_1 , CV\_2 , CV\_3 , CV\_4 ,
 CV\_LAST }
- enum {
 LED\_KEY\_B1 , LED\_KEY\_B2 , LED\_KEY\_B3 , LED\_KEY\_B4 ,
 LED\_KEY\_B5 , LED\_KEY\_B6 , LED\_KEY\_B7 , LED\_KEY\_B8 ,
 LED\_KEY\_A8 , LED\_KEY\_A7 , LED\_KEY\_A6 , LED\_KEY\_A5 ,
 LED\_KEY\_A4 , LED\_KEY\_A3 , LED\_KEY\_A2 , LED\_KEY\_A1 ,
 LED\_KNOB\_1 , LED\_KNOB\_2 , LED\_KNOB\_3 , LED\_KNOB\_4 ,
 LED\_KNOB\_5 , LED\_KNOB\_6 , LED\_KNOB\_7 , LED\_KNOB\_8 ,
 LED\_SW\_1 , LED\_SW\_2 , LED\_LAST }

### Public Member Functions

- void `Init` (bool boost=false)
- void `DelayMs` (size\_t del)
- void `StartAudio` (AudioHandle::InterleavingAudioCallback cb)
- void `StartAudio` (AudioHandle::AudioCallback cb)
- void `StopAudio` ()
- void `ChangeAudioCallback` (AudioHandle::InterleavingAudioCallback cb)
- void `ChangeAudioCallback` (AudioHandle::AudioCallback cb)
- void `SetAudioSampleRate` (SaiHandle::Config::SampleRate samplerate)
- float `AudioSampleRate` ()
- void `SetAudioBlockSize` (size\_t blocksize)

- size\_t `AudioBlockSize ()`
- float `AudioCallbackRate ()`
- void `StartAdc ()`
- void `StopAdc ()`
- void `StartDac ()`
- void `ProcessAnalogControls ()`
- void `ProcessDigitalControls ()`
- void `ProcessAllControls ()`
- void `SetCvOut1 (uint16_t val)`
- void `SetCvOut2 (uint16_t val)`
- bool `KeyboardState (size_t idx) const`
- bool `KeyboardRisingEdge (size_t idx) const`
- bool `KeyboardFallingEdge (size_t idx) const`
- float `GetKnobValue (size_t idx) const`
- float `GetCvValue (size_t idx) const`
- `Switch * GetSwitch (size_t idx)`
- `AnalogControl * GetKnob (size_t idx)`
- `AnalogControl * GetCv (size_t idx)`
- void `VegasMode ()`

## Public Attributes

- `DaisySeed seed`
- `OledDisplay< SSD130x4WireSpi128x64Driver > display`
- `dsy_gpio gate_out`
- `GateIn gate_in`
- `LedDriverPca9685< 2, true > led_driver`
- `Switch sw [SW_LAST]`
- `AnalogControl knob [KNOB_LAST]`
- `AnalogControl cv [CV_LAST]`
- `MidiUartHandler midi`

### 12.43.1 Member Enumeration Documentation

#### 12.43.1.1 anonymous enum

anonymous enum

enums for controls, etc.

Enumerator

|                      |                |
|----------------------|----------------|
| <code>SW_1</code>    | tactile switch |
| <code>SW_2</code>    | tactile switch |
| <code>SW_LAST</code> | &              |

### 12.43.1.2 anonymous enum

anonymous enum

All knobs connect to Daisy Seed's ADC1 pin via CD4051 mux Knobs are in order that they are laid out on hardware.

Enumerator

|           |   |
|-----------|---|
| KNOB_1    | & |
| KNOB_2    | & |
| KNOB_3    | & |
| KNOB_4    | & |
| KNOB_5    | & |
| KNOB_6    | & |
| KNOB_7    | & |
| KNOB_8    | & |
| KNOB_LAST | & |

### 12.43.1.3 anonymous enum

anonymous enum

Enumerator

|         |                           |
|---------|---------------------------|
| CV_2    | Connected to ADC1_INP17   |
| CV_3    | Connected to ADC1_INP15   |
| CV_4    | Connected to ADC1_INP4    |
| CV_LAST | Connected to ADC1_INP11 & |

### 12.43.1.4 anonymous enum

anonymous enum

Enumerator

|            |   |
|------------|---|
| LED_KEY_B1 | & |
| LED_KEY_B2 | & |
| LED_KEY_B3 | & |
| LED_KEY_B4 | & |
| LED_KEY_B5 | & |
| LED_KEY_B6 | & |
| LED_KEY_B7 | & |
| LED_KEY_B8 | & |
| LED_KEY_A8 | & |
| LED_KEY_A7 | & |

## Enumerator

|                       |   |
|-----------------------|---|
| LED_KEY_A6            | & |
| LED_KEY_A5            | & |
| LED_KEY_A4            | & |
| LED_KEY_A3            | & |
| LED_KEY_A2            | & |
| LED_KEY_A1            | & |
| LED_KNOB <sub>1</sub> | & |
| LED_KNOB <sub>2</sub> | & |
| LED_KNOB <sub>3</sub> | & |
| LED_KNOB <sub>4</sub> | & |
| LED_KNOB <sub>5</sub> | & |
| LED_KNOB <sub>6</sub> | & |
| LED_KNOB <sub>7</sub> | & |
| LED_KNOB <sub>8</sub> | & |
| LED_SW_1              | & |
| LED_SW_2              | & |
| LED_LAST              | & |

**12.43.2 Member Function Documentation****12.43.2.1 AudioBlockSize()**

```
size_t daisy::DaisyField::AudioBlockSize ()
```

Returns the number of samples per channel in a block of audio.

**12.43.2.2 AudioCallbackRate()**

```
float daisy::DaisyField::AudioCallbackRate ()
```

Returns the rate in Hz that the Audio callback is called

**12.43.2.3 AudioSampleRate()**

```
float daisy::DaisyField::AudioSampleRate ()
```

Returns the audio sample rate in Hz as a floating point number.

#### 12.43.2.4 ChangeAudioCallback() [1/2]

```
void daisy::DaisyField::ChangeAudioCallback (
 AudioHandle::AudioCallback cb)
```

Switch callback functions

Parameters

|           |                                     |
|-----------|-------------------------------------|
| <i>cb</i> | New multichannel callback function. |
|-----------|-------------------------------------|

#### 12.43.2.5 ChangeAudioCallback() [2/2]

```
void daisy::DaisyField::ChangeAudioCallback (
 AudioHandle::InterleavingAudioCallback cb)
```

Switch callback functions

Parameters

|           |                                    |
|-----------|------------------------------------|
| <i>cb</i> | New interleaved callback function. |
|-----------|------------------------------------|

#### 12.43.2.6 DelayMs()

```
void daisy::DaisyField::DelayMs (
 size_t del)
```

Wait some ms before going on.

Parameters

|            |                   |
|------------|-------------------|
| <i>del</i> | Delay time in ms. |
|------------|-------------------|

#### 12.43.2.7 GetCv()

```
AnalogControl* daisy::DaisyField::GetCv (
 size_t idx)
```

Getter for CV objects.

**Parameters**

|            |                           |
|------------|---------------------------|
| <i>idx</i> | The CV input of interest. |
|------------|---------------------------|

**12.43.2.8 GetCvValue()**

```
float daisy::DaisyField::GetCvValue (
 size_t idx) const
```

Returns the CV input's value

**Parameters**

|            |                           |
|------------|---------------------------|
| <i>idx</i> | The CV input of interest. |
|------------|---------------------------|

**12.43.2.9 GetKnob()**

```
AnalogControl* daisy::DaisyField::GetKnob (
 size_t idx)
```

Getter for knob objects

**Parameters**

|            |                             |
|------------|-----------------------------|
| <i>idx</i> | The knob input of interest. |
|------------|-----------------------------|

**12.43.2.10 GetKnobValue()**

```
float daisy::DaisyField::GetKnobValue (
 size_t idx) const
```

Returns the knob's value

**Parameters**

|            |                       |
|------------|-----------------------|
| <i>idx</i> | The knob of interest. |
|------------|-----------------------|

### 12.43.2.11 GetSwitch()

```
Switch* daisy::DaisyField::GetSwitch (
 size_t idx)
```

Getter for switch objects

#### Parameters

|            |                         |
|------------|-------------------------|
| <i>idx</i> | The switch of interest. |
|------------|-------------------------|

### 12.43.2.12 Init()

```
void daisy::DaisyField::Init (
 bool boost = false)
```

Initializes the Daisy Field, and all of its hardware.

### 12.43.2.13 KeyboardFallingEdge()

```
bool daisy::DaisyField::KeyboardFallingEdge (
 size_t idx) const
```

Returns true if the key has just been released

#### Parameters

|            |                     |
|------------|---------------------|
| <i>idx</i> | the key of interest |
|------------|---------------------|

### 12.43.2.14 KeyboardRisingEdge()

```
bool daisy::DaisyField::KeyboardRisingEdge (
 size_t idx) const
```

Returns true if the key has just been pressed

#### Parameters

|            |                     |
|------------|---------------------|
| <i>idx</i> | the key of interest |
|------------|---------------------|

**12.43.2.15 KeyboardState()**

```
bool daisy::DaisyField::KeyboardState (
 size_t idx) const
```

Returns true if the key has not been pressed recently

Parameters

|            |                     |
|------------|---------------------|
| <i>idx</i> | the key of interest |
|------------|---------------------|

**12.43.2.16 ProcessAllControls()**

```
void daisy::DaisyField::ProcessAllControls () [inline]
```

Process Analog and Digital Controls

**12.43.2.17 ProcessAnalogControls()**

```
void daisy::DaisyField::ProcessAnalogControls ()
```

Processes the ADC inputs, updating their values

**12.43.2.18 ProcessDigitalControls()**

```
void daisy::DaisyField::ProcessDigitalControls ()
```

Process tactile switches and keyboard states

**12.43.2.19 SetAudioBlockSize()**

```
void daisy::DaisyField::SetAudioBlockSize (
 size_t blocksize)
```

Sets the number of samples processed per channel by the audio callback.

**12.43.2.20 SetAudioSampleRate()**

```
void daisy::DaisyField::SetAudioSampleRate (
 SaiHandle::Config::SampleRate samplerate)
```

Updates the Audio Sample Rate, and reinitializes. Audio must be stopped for this to work.

**12.43.2.21 SetCvOut1()**

```
void daisy::DaisyField::SetCvOut1 (
 uint16_t val)
```

Sets the output of CV out 1 to a value between 0-4095 that corresponds to 0-5V

**12.43.2.22 SetCvOut2()**

```
void daisy::DaisyField::SetCvOut2 (
 uint16_t val)
```

Sets the output of CV out 2 to a value between 0-4095 that corresponds to 0-5V

**12.43.2.23 StartAdc()**

```
void daisy::DaisyField::StartAdc ()
```

Starts Transferring data from the ADC

**12.43.2.24 StartAudio() [1/2]**

```
void daisy::DaisyField::StartAudio (
 AudioHandle::AudioCallback cb)
```

Starts the callback \cb multichannel callback function

**12.43.2.25 StartAudio() [2/2]**

```
void daisy::DaisyField::StartAudio (
 AudioHandle::InterleavingAudioCallback cb)
```

Starts the callback \cb Interleaved callback function

**12.43.2.26 StartDac()**

```
void daisy::DaisyField::StartDac ()
```

Turns on the built-in 12-bit DAC on the Daisy Seed **This is now deprecated and does nothing.** The polling use of the DACs now handles starting the transmission.

**12.43.2.27 StopAdc()**

```
void daisy::DaisyField::StopAdc ()
```

Stops Transferring data from the ADC

### 12.43.2.28 StopAudio()

```
void daisy::DaisyField::StopAudio ()
```

Stops the audio if it is running.

### 12.43.2.29 VegasMode()

```
void daisy::DaisyField::VegasMode ()
```

Light show, cycling through all LEDs, and OLED

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

- src/daisy\_field.h

## 12.44 daisy::DaisyPatch Class Reference

Class that handles initializing all of the hardware specific to the Daisy Patch Board.  
Helper funtions are also in place to provide easy access to built-in controls and peripherals.

```
#include <daisy_patch.h>
```

### Public Types

- enum `Ctrl` {  
  `CTRL_1` , `CTRL_2` , `CTRL_3` , `CTRL_4` ,  
  `CTRL_LAST` }
- enum `GateInput` { `GATE_IN_1` , `GATE_IN_2` , `GATE_IN_LAST` }

### Public Member Functions

- `DaisyPatch ()`
- `~DaisyPatch ()`
- `void Init (bool boost=false)`
- `void DelayMs (size_t del)`
- `void StartAudio (AudioHandle::AudioCallback cb)`
- `void ChangeAudioCallback (AudioHandle::AudioCallback cb)`
- `void StopAudio ()`
- `void SetAudioSampleRate (SaiHandle::Config::SampleRate samplerate)`
- `float AudioSampleRate ()`
- `void SetAudioBlockSize (size_t size)`
- `size_t AudioBlockSize ()`
- `float AudioCallbackRate ()`
- `void StartAdc ()`
- `void StopAdc ()`
- `void ProcessAnalogControls ()`
- `void ProcessAllControls ()`
- `float GetKnobValue (Ctrl k)`
- `void ProcessDigitalControls ()`
- `void DisplayControls (bool invert=true)`

## Public Attributes

- `DaisySeed seed`
- `Encoder encoder`
- `AnalogControl controls [CTRL_LAST]`
- `GateIn gate_input [GATE_IN_LAST]`
- `MidiUartHandler midi`
- `OledDisplay< SSD130x4WireSpi128x64Driver > display`
- `dsy_gpio gate_output`

### 12.44.1 Detailed Description

Class that handles initializing all of the hardware specific to the Daisy Patch Board.  
Helper funtions are also in place to provide easy access to built-in controls and peripherals.

#### Author

Stephen Hensley

#### Date

November 2019

### 12.44.2 Member Enumeration Documentation

#### 12.44.2.1 Ctrl

```
enum daisy::DaisyPatch::Ctrl
```

Enum of Ctrl's to represent the four CV/Knob combos on the Patch

#### 12.44.2.2 GateInput

```
enum daisy::DaisyPatch::GateInput
```

Daisy patch gate inputs

#### Enumerator

|              |   |
|--------------|---|
| GATE_IN_LAST | < |
|--------------|---|

### 12.44.3 Constructor & Destructor Documentation

#### 12.44.3.1 DaisyPatch()

```
daisy::DaisyPatch::DaisyPatch () [inline]
```

Constructor

#### 12.44.3.2 ~DaisyPatch()

```
daisy::DaisyPatch::~DaisyPatch () [inline]
```

Destructor

### 12.44.4 Member Function Documentation

#### 12.44.4.1 AudioBlockSize()

```
size_t daisy::DaisyPatch::AudioBlockSize ()
```

Returns the number of samples per channel in a block of audio.

#### 12.44.4.2 AudioCallbackRate()

```
float daisy::DaisyPatch::AudioCallbackRate ()
```

Returns the rate in Hz that the Audio callback is called

#### 12.44.4.3 AudioSampleRate()

```
float daisy::DaisyPatch::AudioSampleRate ()
```

Get sample rate

#### 12.44.4.4 ChangeAudioCallback()

```
void daisy::DaisyPatch::ChangeAudioCallback (
 AudioHandle::AudioCallback cb)
```

[Switch](#) callback functions

Parameters

|                 |                                     |
|-----------------|-------------------------------------|
| <code>cb</code> | New multichannel callback function. |
|-----------------|-------------------------------------|

#### 12.44.4.5 DelayMs()

```
void daisy::DaisyPatch::DelayMs (
 size_t del)
```

Wait some ms before going on.

##### Parameters

|            |                   |
|------------|-------------------|
| <i>del</i> | Delay time in ms. |
|------------|-------------------|

#### 12.44.4.6 DisplayControls()

```
void daisy::DaisyPatch::DisplayControls (
 bool invert = true)
```

Control the display

#### 12.44.4.7 GetKnobValue()

```
float daisy::DaisyPatch::GetKnobValue (
 Ctrl k)
```

Get value for a particular control

##### Parameters

|          |                      |
|----------|----------------------|
| <i>k</i> | Which control to get |
|----------|----------------------|

#### 12.44.4.8 Init()

```
void daisy::DaisyPatch::Init (
 bool boost = false)
```

Initializes the daisy seed, and patch hardware.

#### 12.44.4.9 ProcessAllControls()

```
void daisy::DaisyPatch::ProcessAllControls () [inline]
```

Process Analog and Digital Controls

**12.44.4.10 ProcessAnalogControls()**

```
void daisy::DaisyPatch::ProcessAnalogControls ()
```

Call at same rate as reading controls for good reads.

**12.44.4.11 ProcessDigitalControls()**

```
void daisy::DaisyPatch::ProcessDigitalControls ()
```

Process the digital controls

**12.44.4.12 SetAudioBlockSize()**

```
void daisy::DaisyPatch::SetAudioBlockSize (size_t size)
```

Audio Block size defaults to 48. Change it using this function before StartingAudio

**Parameters**

|                   |                   |
|-------------------|-------------------|
| <code>size</code> | Audio block size. |
|-------------------|-------------------|

**12.44.4.13 SetAudioSampleRate()**

```
void daisy::DaisyPatch::SetAudioSampleRate (SaiHandle::Config::SampleRate samplerate)
```

Set the sample rate for the audio

**12.44.4.14 StartAdc()**

```
void daisy::DaisyPatch::StartAdc ()
```

Start analog to digital conversion.

**12.44.4.15 StartAudio()**

```
void daisy::DaisyPatch::StartAudio (AudioHandle::AudioCallback cb)
```

Starts the callback \cb multichannel callback function

#### 12.44.4.16 StopAdc()

```
void daisy::DaisyPatch::StopAdc ()
```

Stops Transferring data from the ADC

#### 12.44.4.17 StopAudio()

```
void daisy::DaisyPatch::StopAudio ()
```

Stops the audio

### 12.44.5 Member Data Documentation

#### 12.44.5.1 controls

```
AnalogControl daisy::DaisyPatch::controls[CTRL_LAST]
```

Array of controls

#### 12.44.5.2 display

```
OledDisplay<SSD130x4WireSpi128x64Driver> daisy::DaisyPatch::display
```

&

#### 12.44.5.3 encoder

```
Encoder daisy::DaisyPatch::encoder
```

Encoder object

#### 12.44.5.4 gate\_input

```
GateIn daisy::DaisyPatch::gate_input[GATE_IN_LAST]
```

Gate inputs

#### 12.44.5.5 gate\_output

```
dsy_gpio daisy::DaisyPatch::gate_output
```

&

### 12.44.5.6 midi

`MidiUartHandler` `daisy::DaisyPatch::midi`

Handles midi

### 12.44.5.7 seed

`DaisySeed` `daisy::DaisyPatch::seed`

Seed object

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

- `src/daisy_patch.h`

## 12.45 daisy::patch\_sm::DaisyPatchSM Class Reference

Board support file for [DaisyPatchSM](#) hardware.

```
#include <daisy_patch_sm.h>
```

### Public Types

- enum class `PinBank` { **A** , **B** , **C** , **D** }

### Public Member Functions

- void `Init` ()
- void `StartAudio` (`AudioHandle::AudioCallback` cb)
- void `StartAudio` (`AudioHandle::InterleavingAudioCallback` cb)
- void `ChangeAudioCallback` (`AudioHandle::AudioCallback` cb)
- void `ChangeAudioCallback` (`AudioHandle::InterleavingAudioCallback` cb)
- void `StopAudio` ()
- void `SetAudioBlockSize` (size\_t size)
- void `SetAudioSampleRate` (float sr)
- void `SetAudioSampleRate` (`SaiHandle::Config::SampleRate` sample\_rate)
- size\_t `AudioBlockSize` ()
- float `AudioSampleRate` ()
- float `AudioCallbackRate` ()
- void `StartAdc` ()
- void `StopAdc` ()
- void `ProcessAnalogControls` ()
- void `ProcessDigitalControls` ()
- void `ProcessAllControls` ()
- float `GetAdcValue` (int idx)
- `dsy_gpio_pin` `GetPin` (const `PinBank` bank, const int idx)
- void `StartDac` (`DacHandle::DacCallback` callback=nullptr)
- void `StopDac` ()
- void `WriteCvOut` (const int channel, float voltage)
- void `Delay` (uint32\_t milliseconds)
- uint32\_t `GetRandomValue` ()
- float `GetRandomFloat` (float min=0.f, float max=1.f)
- void `SetLed` (bool state)
- bool `ValidateSDRAM` ()

*Tests entirety of SDRAM for validity This will wipe contents of SDRAM when testing.*

- bool `ValidateQSPI` (bool quick=true)

*Tests the QSPI for validity This will wipe contents of QSPI when testing.*

## Static Public Member Functions

- template<typename... VA>  
  static void **Print** (const char \*format, VA... va)
- template<typename... VA>  
  static void **PrintLine** (const char \*format, VA... va)
- static void **StartLog** (bool wait\_for\_pc=false)

## Public Attributes

- **System system**
- **SdramHandle sdram**
- **QSPIHandle qspi**
- **AudioHandle audio**
- **AdcHandle adc**
- **UsbHandle usb**
- **Pcm3060 codec**
- **DacHandle dac**
- **dsy\_gpio user\_led**
- **AnalogControl controls** [ADC\_LAST]
- **GateIn gate\_in\_1**
- **GateIn gate\_in\_2**
- **dsy\_gpio gate\_out\_1**
- **dsy\_gpio gate\_out\_2**

## Static Public Attributes

- static const **dsy\_gpio\_pin A1**
- static const **dsy\_gpio\_pin A2**
- static const **dsy\_gpio\_pin A3**
- static const **dsy\_gpio\_pin A4**
- static const **dsy\_gpio\_pin A5**
- static const **dsy\_gpio\_pin A6**
- static const **dsy\_gpio\_pin A7**
- static const **dsy\_gpio\_pin A8**
- static const **dsy\_gpio\_pin A9**
- static const **dsy\_gpio\_pin A10**
- static const **dsy\_gpio\_pin B1**
- static const **dsy\_gpio\_pin B2**
- static const **dsy\_gpio\_pin B3**
- static const **dsy\_gpio\_pin B4**
- static const **dsy\_gpio\_pin B5**
- static const **dsy\_gpio\_pin B6**
- static const **dsy\_gpio\_pin B7**
- static const **dsy\_gpio\_pin B8**
- static const **dsy\_gpio\_pin B9**
- static const **dsy\_gpio\_pin B10**
- static const **dsy\_gpio\_pin C1**
- static const **dsy\_gpio\_pin C2**
- static const **dsy\_gpio\_pin C3**
- static const **dsy\_gpio\_pin C4**
- static const **dsy\_gpio\_pin C5**
- static const **dsy\_gpio\_pin C6**

- static const `dsy_gpio_pin C7`
- static const `dsy_gpio_pin C8`
- static const `dsy_gpio_pin C9`
- static const `dsy_gpio_pin C10`
- static const `dsy_gpio_pin D1`
- static const `dsy_gpio_pin D2`
- static const `dsy_gpio_pin D3`
- static const `dsy_gpio_pin D4`
- static const `dsy_gpio_pin D5`
- static const `dsy_gpio_pin D6`
- static const `dsy_gpio_pin D7`
- static const `dsy_gpio_pin D8`
- static const `dsy_gpio_pin D9`
- static const `dsy_gpio_pin D10`

### 12.45.1 Detailed Description

Board support file for [DaisyPatchSM](#) hardware.

#### Author

shensley

Daisy Patch SM is a complete Eurorack module DSP engine. Based on the Daisy Seed, with circuits added for interfacing directly with eurorack modular synthesizers.

### 12.45.2 Member Enumeration Documentation

#### 12.45.2.1 PinBank

```
enum daisy::patch_sm::DaisyPatchSM::PinBank [strong]
```

Helper for mapping pins, and accessing them using the GetPin function

### 12.45.3 Member Function Documentation

#### 12.45.3.1 AudioBlockSize()

```
size_t daisy::patch_sm::DaisyPatchSM::AudioBlockSize ()
```

Returns the number of samples processed in an audio callback

### 12.45.3.2 AudioCallbackRate()

```
float daisy::patch_sm::DaisyPatchSM::AudioCallbackRate ()
```

Returns the rate at which the audio callback will be called in Hz

### 12.45.3.3 AudioSampleRate()

```
float daisy::patch_sm::DaisyPatchSM::AudioSampleRate ()
```

Returns the audio engine's samplerate in Hz

### 12.45.3.4 ChangeAudioCallback() [1/2]

```
void daisy::patch_sm::DaisyPatchSM::ChangeAudioCallback (
 AudioHandle::AudioCallback cb)
```

Changes the callback that is executing. This may cause clicks if done while audio is processing.

### 12.45.3.5 ChangeAudioCallback() [2/2]

```
void daisy::patch_sm::DaisyPatchSM::ChangeAudioCallback (
 AudioHandle::InterleavingAudioCallback cb)
```

Changes the callback that is executing. This may cause clicks if done while audio is processing.

### 12.45.3.6 Delay()

```
void daisy::patch_sm::DaisyPatchSM::Delay (
 uint32_t milliseconds) [inline]
```

Here are some wrappers around libDaisy Static functions to provide simpler syntax to those who prefer it. Delays for a specified number of milliseconds

### 12.45.3.7 GetAdcValue()

```
float daisy::patch_sm::DaisyPatchSM::GetAdcValue (
 int idx)
```

Returns the current value for one of the ADCs

### 12.45.3.8 GetPin()

```
dsy_gpio_pin daisy::patch_sm::DaisyPatchSM::GetPin (
 const PinBank bank,
 const int idx)
```

Returns the STM32 port/pin combo for the desired pin (or an invalid pin for HW only pins)

Macros at top of file can be used in place of separate arguments (i.e. GetPin(A4), etc.)

**Parameters**

|             |                                                                  |
|-------------|------------------------------------------------------------------|
| <i>bank</i> | should be one of the PinBank options above                       |
| <i>idx</i>  | pin number between 1 and 10 for each of the pins on each header. |

**12.45.3.9 GetRandomFloat()**

```
float daisy::patch_sm::DaisyPatchSM::GetRandomFloat (
 float min = 0.f,
 float max = 1.f) [inline]
```

Gets a random floating point value between the specified minimum, and maximum

**12.45.3.10 GetRandomValue()**

```
uint32_t daisy::patch_sm::DaisyPatchSM::GetRandomValue () [inline]
```

Gets a random 32-bit value

**12.45.3.11 Init()**

```
void daisy::patch_sm::DaisyPatchSM::Init ()
```

Initializes the memories, and core peripherals for the Daisy Patch SM

**12.45.3.12 Print()**

```
template<typename... VA>
static void daisy::patch_sm::DaisyPatchSM::Print (
 const char * format,
 VA... va) [inline], [static]
```

Print formatted debug log message

**12.45.3.13 PrintLine()**

```
template<typename... VA>
static void daisy::patch_sm::DaisyPatchSM::PrintLine (
 const char * format,
 VA... va) [inline], [static]
```

Print formatted debug log message with automatic line termination

**12.45.3.14 ProcessAllControls()**

```
void daisy::patch_sm::DaisyPatchSM::ProcessAllControls () [inline]
```

Does both of the above

**12.45.3.15 ProcessAnalogControls()**

```
void daisy::patch_sm::DaisyPatchSM::ProcessAnalogControls ()
```

Reads and filters all of the analog control inputs

**12.45.3.16 ProcessDigitalControls()**

```
void daisy::patch_sm::DaisyPatchSM::ProcessDigitalControls ()
```

Reads and debounces any of the digital control inputs This does nothing on this board at this time.

**12.45.3.17 SetAudioBlockSize()**

```
void daisy::patch_sm::DaisyPatchSM::SetAudioBlockSize (size_t size)
```

Sets the number of samples processed in an audio callback. This will only take effect on the next invocation of StartAudio

**12.45.3.18 SetAudioSampleRate()**

```
void daisy::patch_sm::DaisyPatchSM::SetAudioSampleRate (float sr)
```

Sets the samplerate for the audio engine This will set it to the closest valid samplerate. Options being: 8kHz, 16kHz, 32kHz, 48kHz, and 96kHz

**12.45.3.19 StartAdc()**

```
void daisy::patch_sm::DaisyPatchSM::StartAdc ()
```

Starts the Control ADCs

This is started automatically when [Init\(\)](#) is called.

**12.45.3.20 StartAudio() [1/2]**

```
void daisy::patch_sm::DaisyPatchSM::StartAudio (AudioHandle::AudioCallback cb)
```

Starts a non-interleaving audio callback

**12.45.3.21 StartAudio() [2/2]**

```
void daisy::patch_sm::DaisyPatchSM::StartAudio (
 AudioHandle::InterleavingAudioCallback cb)
```

Starts an interleaving audio callback

**12.45.3.22 StartDac()**

```
void daisy::patch_sm::DaisyPatchSM::StartDac (
 DacHandle::DacCallback callback = nullptr)
```

Starts the DAC for the CV Outputs

By default this starts by running the internal callback at 48kHz, which will update the values based on the SetCvOut function.

This is started automatically when [Init\(\)](#) is called.

**12.45.3.23 StartLog()**

```
static void daisy::patch_sm::DaisyPatchSM::StartLog (
 bool wait_for_pc = false) [inline], [static]
```

Start the logging session. Optionally wait for terminal connection before proceeding.

**12.45.3.24 StopAdc()**

```
void daisy::patch_sm::DaisyPatchSM::StopAdc ()
```

Stops the Control ADCs

**12.45.3.25 StopAudio()**

```
void daisy::patch_sm::DaisyPatchSM::StopAudio ()
```

Stops the transmission of audio.

**12.45.3.26 StopDac()**

```
void daisy::patch_sm::DaisyPatchSM::StopDac ()
```

Stop the DAC from updating. This will suspend the CV Outputs from changing

**12.45.3.27 ValidateQSPI()**

```
bool daisy::patch_sm::DaisyPatchSM::ValidateQSPI (
 bool quick = true)
```

Tests the QSPI for validity. This will wipe contents of QSPI when testing.

**Note**

If called with quick = false, this will erase all memory the "quick" test starts 0x400000 bytes into the memory and test 16kB of data

**Parameters**

|              |                                                                                                                           |
|--------------|---------------------------------------------------------------------------------------------------------------------------|
| <i>quick</i> | if this is true the test will only test a small piece of the QSPI checking the entire 8MB can take roughly over a minute. |
|--------------|---------------------------------------------------------------------------------------------------------------------------|

**Return values**

|                |                                        |
|----------------|----------------------------------------|
| <i>returns</i> | true if SDRAM is okay, otherwise false |
|----------------|----------------------------------------|

**12.45.3.28 ValidateSDRAM()**

```
bool daisy::patch_sm::DaisyPatchSM::ValidateSDRAM ()
```

Tests entirety of SDRAM for validity This will wipe contents of SDRAM when testing.

**Note**

If using the SDRAM for the default bss, or heap, and using constructors as initializers do not call this function. Otherwise, it could overwrite changes performed by constructors.

**Return values**

|                |                                        |
|----------------|----------------------------------------|
| <i>returns</i> | true if SDRAM is okay, otherwise false |
|----------------|----------------------------------------|

**12.45.3.29 WriteCvOut()**

```
void daisy::patch_sm::DaisyPatchSM::WriteCvOut (
 const int channel,
 float voltage)
```

Sets specified DAC channel to the target voltage. This may not be 100% accurate without calibration.

**Todo** Add Calibration to CV Outputs

**Parameters**

|                |                                                                              |
|----------------|------------------------------------------------------------------------------|
| <i>channel</i> | desired channel to update. 0 is both, otherwise 1 or 2 are valid.            |
| <i>voltage</i> | value in Volts that you'd like to write to the DAC. The valid range is 0-5V. |

## 12.45.4 Member Data Documentation

### 12.45.4.1 A1

```
const dsy_gpio_pin daisy::patch_sm::DaisyPatchSM::A1 [static]
```

Pin Accessors for the [DaisyPatchSM](#) hardware Used for initializing various [GPIO](#), etc.

### 12.45.4.2 system

```
System daisy::patch_sm::DaisyPatchSM::system
```

Direct Access Structs/Classes

### 12.45.4.3 user\_led

```
dsy_gpio daisy::patch_sm::DaisyPatchSM::user_led
```

Dedicated Function Pins

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

- src/daisy\_patch\_sm.h

## 12.46 daisy::DaisyPetal Class Reference

Helpers and hardware definitions for daisy petal.

```
#include <daisy_petal.h>
```

### Public Types

- enum [Sw](#){  
    [SW\\_1](#) , [SW\\_2](#) , [SW\\_3](#) , [SW\\_4](#) ,  
    [SW\\_5](#) , [SW\\_6](#) , [SW\\_7](#) , [SW\\_LAST](#) }
- enum [Knob](#){  
    [KNOB\\_1](#) , [KNOB\\_2](#) , [KNOB\\_3](#) , [KNOB\\_4](#) ,  
    [KNOB\\_5](#) , [KNOB\\_6](#) , [KNOB\\_LAST](#) }
- enum [RingLed](#){  
    [RING\\_LED\\_1](#) , [RING\\_LED\\_2](#) , [RING\\_LED\\_3](#) , [RING\\_LED\\_4](#) ,  
    [RING\\_LED\\_5](#) , [RING\\_LED\\_6](#) , [RING\\_LED\\_7](#) , [RING\\_LED\\_8](#) ,  
    [RING\\_LED\\_LAST](#) }
- enum [FootswitchLed](#){  
    [FOOTSWITCH\\_LED\\_1](#) , [FOOTSWITCH\\_LED\\_2](#) , [FOOTSWITCH\\_LED\\_3](#) , [FOOTSWITCH\\_LED\\_4](#) ,  
    [FOOTSWITCH\\_LED\\_LAST](#) }

## Public Member Functions

- `DaisyPetal ()`
- `~DaisyPetal ()`
- `void Init (bool boost=false)`
- `void DelayMs (size_t del)`
- `void StartAudio (AudioHandle::InterleavingAudioCallback cb)`
- `void StartAudio (AudioHandle::AudioCallback cb)`
- `void ChangeAudioCallback (AudioHandle::InterleavingAudioCallback cb)`
- `void ChangeAudioCallback (AudioHandle::AudioCallback cb)`
- `void StopAudio ()`
- `void SetAudioSampleRate (SaiHandle::Config::SampleRate samplerate)`
- `float AudioSampleRate ()`
- `void SetAudioBlockSize (size_t size)`
- `size_t AudioBlockSize ()`
- `float AudioCallbackRate ()`
- `void StartAdc ()`
- `void StopAdc ()`
- `void ProcessAnalogControls ()`
- `void ProcessAllControls ()`
- `float GetKnobValue (Knob k)`
- `float GetExpression ()`
- `void ProcessDigitalControls ()`
- `void ClearLeds ()`
- `void UpdateLeds ()`
- `void SetRingLed (RingLed idx, float r, float g, float b)`
- `void SetFootswitchLed (FootswitchLed idx, float bright)`

## Public Attributes

- `DaisySeed seed`
- `Encoder encoder`
- `AnalogControl knob [KNOB_LAST]`
- `AnalogControl expression`
- `Switch switches [SW_LAST]`
- `RgbLed ring_led [8]`
- `Led footswitch_led [4]`

### 12.46.1 Detailed Description

Helpers and hardware definitions for daisy petal.

### 12.46.2 Member Enumeration Documentation

#### 12.46.2.1 FootswitchLed

```
enum daisy::DaisyPetal::FootswitchLed
```

footswitch leds

**Enumerator**

|                     |   |
|---------------------|---|
| FOOTSWITCH_LED_1    | & |
| FOOTSWITCH_LED_2    | & |
| FOOTSWITCH_LED_3    | & |
| FOOTSWITCH_LED_4    | & |
| FOOTSWITCH_LED_LAST | & |

**12.46.2.2 Knob**

```
enum daisy::DaisyPetal::Knob
```

**Knobs****Enumerator**

|           |   |
|-----------|---|
| KNOB_1    | & |
| KNOB_2    | & |
| KNOB_3    | & |
| KNOB_4    | & |
| KNOB_5    | & |
| KNOB_6    | & |
| KNOB_LAST | & |

**12.46.2.3 RingLed**

```
enum daisy::DaisyPetal::RingLed
```

**Leds in ringled****Enumerator**

|               |   |
|---------------|---|
| RING_LED_1    | & |
| RING_LED_2    | & |
| RING_LED_3    | & |
| RING_LED_4    | & |
| RING_LED_5    | & |
| RING_LED_6    | & |
| RING_LED_7    | & |
| RING_LED_8    | & |
| RING_LED_LAST | & |

#### 12.46.2.4 Sw

```
enum daisy::DaisyPetal::Sw
```

Switches

Enumerator

|         |                |
|---------|----------------|
| SW_1    | Footswitch     |
| SW_2    | Footswitch     |
| SW_3    | Footswitch     |
| SW_4    | Footswitch     |
| SW_5    | Toggle         |
| SW_6    | Toggle         |
| SW_7    | Toggle         |
| SW_LAST | Last enum item |

### 12.46.3 Constructor & Destructor Documentation

#### 12.46.3.1 DaisyPetal()

```
daisy::DaisyPetal::DaisyPetal () [inline]
```

Constructor

#### 12.46.3.2 ~DaisyPetal()

```
daisy::DaisyPetal::~DaisyPetal () [inline]
```

Destructor

### 12.46.4 Member Function Documentation

#### 12.46.4.1 AudioBlockSize()

```
size_t daisy::DaisyPetal::AudioBlockSize ()
```

Returns the number of samples per channel in a block of audio.

**12.46.4.2 AudioCallbackRate()**

```
float daisy::DaisyPetal::AudioCallbackRate ()
```

Returns the rate in Hz that the Audio callback is called

**12.46.4.3 AudioSampleRate()**

```
float daisy::DaisyPetal::AudioSampleRate ()
```

Returns the audio sample rate in Hz as a floating point number.

**12.46.4.4 ChangeAudioCallback() [1/2]**

```
void daisy::DaisyPetal::ChangeAudioCallback (
 AudioHandle::AudioCallback cb)
```

[Switch](#) callback functions

**Parameters**

|           |                                     |
|-----------|-------------------------------------|
| <i>cb</i> | New multichannel callback function. |
|-----------|-------------------------------------|

**12.46.4.5 ChangeAudioCallback() [2/2]**

```
void daisy::DaisyPetal::ChangeAudioCallback (
 AudioHandle::InterleavingAudioCallback cb)
```

[Switch](#) callback functions

**Parameters**

|           |                                    |
|-----------|------------------------------------|
| <i>cb</i> | New interleaved callback function. |
|-----------|------------------------------------|

**12.46.4.6 ClearLeds()**

```
void daisy::DaisyPetal::ClearLeds ()
```

Turn all leds off

#### 12.46.4.7 DelayMs()

```
void daisy::DaisyPetal::DelayMs (
 size_t del)
```

Wait before moving on.

**Parameters**

|            |                   |
|------------|-------------------|
| <i>del</i> | Delay time in ms. |
|------------|-------------------|

**12.46.4.8 GetExpression()**

```
float daisy::DaisyPetal::GetExpression ()
&
```

**12.46.4.9 GetKnobValue()**

```
float daisy::DaisyPetal::GetKnobValue (
 Knob k)
```

Get value per knob.

**Parameters**

|          |                   |
|----------|-------------------|
| <i>k</i> | Which knob to get |
|----------|-------------------|

**Returns**

Floating point knob position.

**12.46.4.10 Init()**

```
void daisy::DaisyPetal::Init (
 bool boost = false)
```

Initialize daisy petal

**12.46.4.11 ProcessAllControls()**

```
void daisy::DaisyPetal::ProcessAllControls () [inline]
```

Process Analog and Digital Controls

**12.46.4.12 ProcessAnalogControls()**

```
void daisy::DaisyPetal::ProcessAnalogControls ()
```

Call at the same frequency as controls are read for stable readings.

#### 12.46.4.13 ProcessDigitalControls()

```
void daisy::DaisyPetal::ProcessDigitalControls ()
```

Process digital controls

#### 12.46.4.14 SetAudioBlockSize()

```
void daisy::DaisyPetal::SetAudioBlockSize (
 size_t size)
```

Sets the number of samples processed per channel by the audio callback.

Parameters

|             |                  |
|-------------|------------------|
| <i>size</i> | Audio block size |
|-------------|------------------|

#### 12.46.4.15 SetAudioSampleRate()

```
void daisy::DaisyPetal::SetAudioSampleRate (
 SaiHandle::Config::SampleRate samplerate)
```

Updates the Audio Sample Rate, and reinitializes. Audio must be stopped for this to work.

#### 12.46.4.16 SetFootswitchLed()

```
void daisy::DaisyPetal::SetFootswitchLed (
 FootswitchLed idx,
 float bright)
```

Set footswitch LED

Parameters

|               |            |
|---------------|------------|
| <i>idx</i>    | Led Index  |
| <i>bright</i> | Brightness |

#### 12.46.4.17 SetRingLed()

```
void daisy::DaisyPetal::SetRingLed (
 RingLed idx,
 float r,
```

```
 float g,
 float b)
```

Set ring LED colors

**Parameters**

|            |              |
|------------|--------------|
| <i>idx</i> | Index to set |
| <i>r</i>   | Red value    |
| <i>g</i>   | Green value  |
| <i>b</i>   | Blue value   |

**12.46.4.18 StartAdc()**

```
void daisy::DaisyPetal::StartAdc ()
```

Start analog to digital conversion.

**12.46.4.19 StartAudio() [1/2]**

```
void daisy::DaisyPetal::StartAudio (
 AudioHandle::AudioCallback cb)
```

Starts the callback \cb multichannel callback function

**12.46.4.20 StartAudio() [2/2]**

```
void daisy::DaisyPetal::StartAudio (
 AudioHandle::InterleavingAudioCallback cb)
```

Starts the callback \cb Interleaved callback function

**12.46.4.21 StopAdc()**

```
void daisy::DaisyPetal::StopAdc ()
```

Stops Transferring data from the ADC

**12.46.4.22 StopAudio()**

```
void daisy::DaisyPetal::StopAudio ()
```

Stops the audio if it is running.

**12.46.4.23 UpdateLeds()**

```
void daisy::DaisyPetal::UpdateLeds ()
```

Update Leds to values you had set.

## 12.46.5 Member Data Documentation

### 12.46.5.1 encoder

```
Encoder daisy::DaisyPetal::encoder
&
```

### 12.46.5.2 expression

```
AnalogControl daisy::DaisyPetal::expression
&
```

### 12.46.5.3 footswitch\_led

```
Led daisy::DaisyPetal::footswitch_led[4]
&
```

### 12.46.5.4 knob

```
AnalogControl daisy::DaisyPetal::knob[KNOB_LAST]
&
```

### 12.46.5.5 ring\_led

```
RgbLed daisy::DaisyPetal::ring_led[8]
&
```

### 12.46.5.6 seed

```
DaisySeed daisy::DaisyPetal::seed
&
```

### 12.46.5.7 switches

```
Switch daisy::DaisyPetal::switches[SW_LAST]
< &
```

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

- src/daisy\_petal.h

## 12.47 daisy::DaisyPod Class Reference

Class that handles initializing all of the hardware specific to the Daisy Patch Board.  
Helper funtions are also in place to provide easy access to built-in controls and peripherals.

```
#include <daisy_pod.h>
```

### Public Types

- enum `Sw` { `BUTTON_1` , `BUTTON_2` , `BUTTON_LAST` }
- enum `Knob` { `KNOB_1` , `KNOB_2` , `KNOB_LAST` }

### Public Member Functions

- void `Init` (bool boost=false)
- void `DelayMs` (size\_t del)
- void `StartAudio` (AudioHandle::InterleavingAudioCallback cb)
- void `StartAudio` (AudioHandle::AudioCallback cb)
- void `ChangeAudioCallback` (AudioHandle::InterleavingAudioCallback cb)
- void `ChangeAudioCallback` (AudioHandle::AudioCallback cb)
- void `StopAudio` ()
- void `SetAudioSampleRate` (SaiHandle::Config::SampleRate samplerate)
- float `AudioSampleRate` ()
- void `SetAudioBlockSize` (size\_t blocksize)
- size\_t `AudioBlockSize` ()
- float `AudioCallbackRate` ()
- void `StartAdc` ()
- void `StopAdc` ()
- void `ProcessAnalogControls` ()
- void `ProcessAllControls` ()
- float `GetKnobValue` (Knob k)
- void `ProcessDigitalControls` ()
- void `ClearLeds` ()
- void `UpdateLeds` ()

### Public Attributes

- `DaisySeed` seed
- `Encoder` encoder
- `AnalogControl` knob1
- `AnalogControl` knob2
- `AnalogControl` \* knobs [KNOB\_LAST]
- `Switch` button1
- `Switch` button2
- `Switch` \* buttons [BUTTON\_LAST]
- `RgbLed` led1
- `RgbLed` led2
- `MidiUartHandler` midi

### 12.47.1 Detailed Description

Class that handles initializing all of the hardware specific to the Daisy Patch Board.  
Helper functions are also in place to provide easy access to built-in controls and peripherals.

#### Author

Stephen Hensley

#### Date

November 2019

### 12.47.2 Member Enumeration Documentation

#### 12.47.2.1 Knob

```
enum daisy::DaisyPod::Knob
```

Knobs

##### Enumerator

|           |   |
|-----------|---|
| KNOB_2    | & |
| KNOB_LAST | & |

#### 12.47.2.2 Sw

```
enum daisy::DaisyPod::Sw
```

Switches

##### Enumerator

|             |   |
|-------------|---|
| BUTTON_2    | & |
| BUTTON_LAST | & |

### 12.47.3 Member Function Documentation

### 12.47.3.1 AudioBlockSize()

```
size_t daisy::DaisyPod::AudioBlockSize ()
```

Returns the number of samples per channel in a block of audio.

### 12.47.3.2 AudioCallbackRate()

```
float daisy::DaisyPod::AudioCallbackRate ()
```

Returns the rate in Hz that the Audio callback is called

### 12.47.3.3 AudioSampleRate()

```
float daisy::DaisyPod::AudioSampleRate ()
```

Returns the audio sample rate in Hz as a floating point number.

### 12.47.3.4 ChangeAudioCallback() [1/2]

```
void daisy::DaisyPod::ChangeAudioCallback (
 AudioHandle::AudioCallback cb)
```

**Switch** callback functions

#### Parameters

|           |                                     |
|-----------|-------------------------------------|
| <i>cb</i> | New multichannel callback function. |
|-----------|-------------------------------------|

### 12.47.3.5 ChangeAudioCallback() [2/2]

```
void daisy::DaisyPod::ChangeAudioCallback (
 AudioHandle::InterleavingAudioCallback cb)
```

**Switch** callback functions

#### Parameters

|           |                                    |
|-----------|------------------------------------|
| <i>cb</i> | New interleaved callback function. |
|-----------|------------------------------------|

### 12.47.3.6 ClearLeds()

```
void daisy::DaisyPod::ClearLeds ()
```

Reset Leds

#### 12.47.3.7 DelayMs()

```
void daisy::DaisyPod::DelayMs (
 size_t del)
```

Wait for a bit

##### Parameters

|            |                     |
|------------|---------------------|
| <i>del</i> | Time to wait in ms. |
|------------|---------------------|

#### 12.47.3.8 GetKnobValue()

```
float daisy::DaisyPod::GetKnobValue (
 Knob k)
```

&

#### 12.47.3.9 Init()

```
void daisy::DaisyPod::Init (
 bool boost = false)
```

Init related stuff.

#### 12.47.3.10 ProcessAllControls()

```
void daisy::DaisyPod::ProcessAllControls () [inline]
```

Process Analog and Digital Controls

#### 12.47.3.11 ProcessAnalogControls()

```
void daisy::DaisyPod::ProcessAnalogControls ()
```

Call at same rate as analog reads for smooth reading.

#### 12.47.3.12 ProcessDigitalControls()

```
void daisy::DaisyPod::ProcessDigitalControls ()
```

Process digital controls

**12.47.3.13 SetAudioBlockSize()**

```
void daisy::DaisyPod::SetAudioBlockSize (
 size_t blocksize)
```

Sets the number of samples processed per channel by the audio callback.

**12.47.3.14 SetAudioSampleRate()**

```
void daisy::DaisyPod::SetAudioSampleRate (
 SaiHandle::Config::SampleRate samplerate)
```

Updates the Audio Sample Rate, and reinitializes. Audio must be stopped for this to work.

**12.47.3.15 StartAdc()**

```
void daisy::DaisyPod::StartAdc ()
```

Start analog to digital conversion.

**12.47.3.16 StartAudio() [1/2]**

```
void daisy::DaisyPod::StartAudio (
 AudioHandle::AudioCallback cb)
```

Starts the callback \cb multichannel callback function

**12.47.3.17 StartAudio() [2/2]**

```
void daisy::DaisyPod::StartAudio (
 AudioHandle::InterleavingAudioCallback cb)
```

Starts the callback \cb Interleaved callback function

**12.47.3.18 StopAdc()**

```
void daisy::DaisyPod::StopAdc ()
```

Stops Transferring data from the ADC

**12.47.3.19 StopAudio()**

```
void daisy::DaisyPod::StopAudio ()
```

Stops the audio if it is running.

**12.47.3.20 UpdateLeds()**

```
void daisy::DaisyPod::UpdateLeds ()
```

Update Leds to set colors

**12.47.4 Member Data Documentation****12.47.4.1 button1**

```
Switch daisy::DaisyPod::button1
&
```

**12.47.4.2 button2**

```
Switch daisy::DaisyPod::button2
&
```

**12.47.4.3 buttons**

```
Switch * daisy::DaisyPod::buttons[BUTTON_LAST]
&
```

**12.47.4.4 encoder**

```
Encoder daisy::DaisyPod::encoder
&
```

**12.47.4.5 knob1**

```
AnalogControl daisy::DaisyPod::knob1
&
```

**12.47.4.6 knob2**

```
AnalogControl daisy::DaisyPod::knob2
&
```

#### 12.47.4.7 knobs

```
AnalogControl * daisy::DaisyPod::knobs [KNOB_LAST]
```

&

#### 12.47.4.8 led1

```
RgbLed daisy::DaisyPod::led1
```

&

#### 12.47.4.9 led2

```
RgbLed daisy::DaisyPod::led2
```

&

#### 12.47.4.10 seed

```
DaisySeed daisy::DaisyPod::seed
```

Public Members

### 12.47.5 autotoc\_md0

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

- src/daisy\_pod.h

## 12.48 daisy::DaisySeed Class Reference

This is the higher-level interface for the Daisy board.

All basic peripheral configuration/initialization is setup here.

```
#include <daisy_seed.h>
```

### Public Types

- enum class `BoardVersion` { `DAISY_SEED` , `DAISY_SEED_1_1` }

## Public Member Functions

- void `Configure ()`
- void `Init (bool boost=false)`
- void `DeInit ()`
- void `DelayMs (size_t del)`
- void `StartAudio (AudioHandle::InterleavingAudioCallback cb)`
- void `StartAudio (AudioHandle::AudioCallback cb)`
- void `ChangeAudioCallback (AudioHandle::InterleavingAudioCallback cb)`
- void `ChangeAudioCallback (AudioHandle::AudioCallback cb)`
- void `StopAudio ()`
- void `SetAudioSampleRate (SaiHandle::Config::SampleRate samplerate)`
- float `AudioSampleRate ()`
- void `SetAudioBlockSize (size_t blocksize)`
- size\_t `AudioBlockSize () const`
- float `AudioCallbackRate () const`
- void `SetLed (bool state)`
- void `SetTestPoint (bool state)`
- BoardVersion `CheckBoardVersion ()`

## Static Public Member Functions

- static `dsy_gpio_pin GetPin (uint8_t pin_idx)`
- template<typename... VA>  
  static void `Print (const char *format, VA... va)`
- template<typename... VA>  
  static void `PrintLine (const char *format, VA... va)`
- static void `StartLog (bool wait_for_pc=false)`

## Public Attributes

- `QSPIHandle qspi`
- `QSPIHandle::Config qspi_config`
- `SdramHandle sram_handle`
- `AudioHandle audio_handle`
- `AdcHandle adc`
- `DacHandle dac`
- `UsbHandle usb_handle`
- `dsy_gpio led`
- `dsy_gpio testpoint`
- `System system`

### 12.48.1 Detailed Description

This is the higher-level interface for the Daisy board.  
All basic peripheral configuration/initialization is setup here.

### 12.48.2 Member Enumeration Documentation

### 12.48.2.1 BoardVersion

```
enum daisy::DaisySeed::BoardVersion [strong]
```

Internal indices for DaisySeed-equivalent devices This shouldn't have any effect on user-facing code, and only needs to be checked to properly initialize the onboard-circuits.

## Enumerator

|                     |                                                                                                                                          |
|---------------------|------------------------------------------------------------------------------------------------------------------------------------------|
| DAISY_SEED          | Daisy Seed Rev4 This is the original Daisy Seed                                                                                          |
| DAISY_SEED_1←<br>_1 | Daisy Seed 1.1 (aka Daisy Seed Rev5) This is a pin-compatible version of the Daisy Seed that uses the WM8731 codec instead of the AK4430 |

**12.48.3 Member Function Documentation****12.48.3.1 AudioBlockSize()**

```
size_t daisy::DaisySeed::AudioBlockSize ()
```

Returns the number of samples per channel in a block of audio.

**12.48.3.2 AudioCallbackRate()**

```
float daisy::DaisySeed::AudioCallbackRate () const
```

Returns the rate in Hz that the Audio callback is called

**12.48.3.3 AudioSampleRate()**

```
float daisy::DaisySeed::AudioSampleRate ()
```

Returns the audio sample rate in Hz as a floating point number.

**12.48.3.4 ChangeAudioCallback() [1/2]**

```
void daisy::DaisySeed::ChangeAudioCallback (
 AudioHandle::AudioCallback cb)
```

Changes to a new multichannel callback

**12.48.3.5 ChangeAudioCallback() [2/2]**

```
void daisy::DaisySeed::ChangeAudioCallback (
 AudioHandle::InterleavingAudioCallback cb)
```

Changes to a new interleaved callback

### 12.48.3.6 CheckBoardVersion()

```
BoardVersion daisy::DaisySeed::CheckBoardVersion ()
```

Returns the BoardVersion detected during initialization

### 12.48.3.7 Configure()

```
void daisy::DaisySeed::Configure ()
```

This function used to provide a pre-initialization configuration it has since been deprecated, and does nothing.

### 12.48.3.8 DeInit()

```
void daisy::DaisySeed::DeInit ()
```

Deinitializes all peripherals automatically handled by Init.

### 12.48.3.9 DelayMs()

```
void daisy::DaisySeed::DelayMs (
 size_t del)
```

Wait some ms before going on.

#### Parameters

|            |                   |
|------------|-------------------|
| <i>del</i> | Delay time in ms. |
|------------|-------------------|

### 12.48.3.10 GetPin()

```
static dsy_gpio_pin daisy::DaisySeed::GetPin (
 uint8_t pin_idx) [static]
```

Returns the gpio\_pin corresponding to the index 0-31. For the given [GPIO](#) on the Daisy Seed (labeled 1-32 in docs).

### 12.48.3.11 Init()

```
void daisy::DaisySeed::Init (
 bool boost = false)
```

Initializes the Daisy Seed and the following peripherals: SDRAM, QSPI, 24-bit 48kHz Audio via AK4556, Internal USB, as well as the built-in LED and Testpoint.

ADCs, DACs, and other special peripherals (such as I2C, SPI, etc.) can be initialized using their specific initializers within libdaisy for a specific application.

**12.48.3.12 Print()**

```
template<typename... VA>
static void daisy::DaisySeed::Print (
 const char * format,
 VA... va) [inline], [static]
```

Print formatted debug log message

**12.48.3.13 PrintLine()**

```
template<typename... VA>
static void daisy::DaisySeed::PrintLine (
 const char * format,
 VA... va) [inline], [static]
```

Print formatted debug log message with automatic line termination

**12.48.3.14 SetAudioBlockSize()**

```
void daisy::DaisySeed::SetAudioBlockSize (
 size_t blocksize)
```

Sets the number of samples processed per channel by the audio callback.

**12.48.3.15 SetAudioSampleRate()**

```
void daisy::DaisySeed::SetAudioSampleRate (
 SaiHandle::Config::SampleRate samplerate)
```

Updates the Audio Sample Rate, and reinitializes. Audio must be stopped for this to work.

**12.48.3.16 SetLed()**

```
void daisy::DaisySeed::SetLed (
 bool state)
```

Sets the state of the built in LED

**12.48.3.17 SetTestPoint()**

```
void daisy::DaisySeed::SetTestPoint (
 bool state)
```

Sets the state of the test point near pin 10

**12.48.3.18 StartAudio() [1/2]**

```
void daisy::DaisySeed::StartAudio (
 AudioHandle::AudioCallback cb)
```

Begins the audio for the seeds builtin audio. the specified callback will get called whenever new data is ready to be prepared. This will use the newer non-interleaved callback.

**12.48.3.19 StartAudio() [2/2]**

```
void daisy::DaisySeed::StartAudio (
 AudioHandle::InterleavingAudioCallback cb)
```

Begins the audio for the seeds builtin audio. the specified callback will get called whenever new data is ready to be prepared.

**12.48.3.20 StartLog()**

```
static void daisy::DaisySeed::StartLog (
 bool wait_for_pc = false) [inline], [static]
```

Start the logging session. Optionally wait for terminal connection before proceeding.

**12.48.3.21 StopAudio()**

```
void daisy::DaisySeed::StopAudio ()
```

Stops the audio if it is running.

**12.48.4 Member Data Documentation****12.48.4.1 adc**

```
AdcHandle daisy::DaisySeed::adc
```

&

**12.48.4.2 audio\_handle**

```
AudioHandle daisy::DaisySeed::audio_handle
```

&

#### 12.48.4.3 s dram \_ handle

```
SdramHandle daisy::DaisySeed::sdram_handle
&
```

#### 12.48.4.4 u sb \_ handle

```
UsbHandle daisy::DaisySeed::usb_handle
&
```

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

- src/daisy\_seed.h

## 12.49 daisy::DaisyVersio Class Reference

Class that handles initializing all of the hardware specific to the Desmodus Versio hardware. Helper funtions are also in place to provide easy access to built-in controls and peripherals.

```
#include <daisy_versio.h>
```

### Public Types

- enum AV\_LEDS {
 LED\_0 , LED\_1 , LED\_2 , LED\_3 ,
 LED\_LAST }
- enum AV\_KNOBS {
 KNOB\_0 , KNOB\_1 , KNOB\_2 , KNOB\_3 ,
 KNOB\_4 , KNOB\_5 , KNOB\_6 , KNOB\_LAST }
- enum AV\_TOGGLE3 { SW\_0 , SW\_1 , SW\_LAST }

### Public Member Functions

- void **Init** (bool boost=false)
- void **DelayMs** (size\_t del)
- void **StartAudio** (AudioHandle::InterleavingAudioCallback cb)
- void **StartAudio** (AudioHandle::AudioCallback cb)
- void **ChangeAudioCallback** (AudioHandle::InterleavingAudioCallback cb)
- void **ChangeAudioCallback** (AudioHandle::AudioCallback cb)
- void **StopAudio** ()
- void **SetAudioBlockSize** (size\_t size)
- size\_t **AudioBlockSize** ()
- void **SetAudioSampleRate** (SaiHandle::Config::SampleRate samplerate)
- float **AudioSampleRate** ()
- float **AudioCallbackRate** ()
- void **StartAdc** ()
- void **StopAdc** ()
- void **ProcessAnalogControls** ()
- void **ProcessAllControls** ()
- bool **SwitchPressed** ()
- bool **Gate** ()
- void **SetLed** (size\_t idx, float red, float green, float blue)
- float **GetKnobValue** (int idx)
- void **UpdateLeds** ()
- void **UpdateExample** ()

## Public Attributes

- `DaisySeed seed`
- `RgbLed leds [LED_LAST]`
- `AnalogControl knobs [KNOB_LAST]`
- `Switch tap`
- `GateIn gate`
- `Switch3 sw [SW_LAST]`

### 12.49.1 Detailed Description

Class that handles initializing all of the hardware specific to the Desmodus Versio hardware. Helper funtions are also in place to provide easy access to built-in controls and peripherals.

#### Author

Ankoor Apte, Noise Engineering

#### Date

October 2020

### 12.49.2 Member Function Documentation

#### 12.49.2.1 AudioBlockSize()

```
size_t daisy::DaisyVersio::AudioBlockSize ()
```

Returns the number of samples per channel in a block of audio.

#### 12.49.2.2 AudioCallbackRate()

```
float daisy::DaisyVersio::AudioCallbackRate ()
```

Returns the rate in Hz that the Audio callback is called

#### 12.49.2.3 AudioSampleRate()

```
float daisy::DaisyVersio::AudioSampleRate ()
```

Returns the audio sample rate in Hz as a floating point number.

#### 12.49.2.4 ChangeAudioCallback() [1/2]

```
void daisy::DaisyVersio::ChangeAudioCallback (
 AudioHandle::AudioCallback cb)
```

`Switch` callback functions

**Parameters**

|           |                                        |
|-----------|----------------------------------------|
| <i>cb</i> | New non-interleaved callback function. |
|-----------|----------------------------------------|

**12.49.2.5 ChangeAudioCallback() [2/2]**

```
void daisy::DaisyVersio::ChangeAudioCallback (
 AudioHandle::InterleavingAudioCallback cb)
```

[Switch](#) callback functions

**Parameters**

|           |                                    |
|-----------|------------------------------------|
| <i>cb</i> | New interleaved callback function. |
|-----------|------------------------------------|

**12.49.2.6 DelayMs()**

```
void daisy::DaisyVersio::DelayMs (
 size_t del)
```

Wait some ms before going on.

**Parameters**

|            |                   |
|------------|-------------------|
| <i>del</i> | Delay time in ms. |
|------------|-------------------|

**12.49.2.7 Gate()**

```
bool daisy::DaisyVersio::Gate ()
```

Returns true if gate in is HIGH

**12.49.2.8 GetKnobValue()**

```
float daisy::DaisyVersio::GetKnobValue (
 int idx)
```

Get Knob Value, float from 0.0f to 1.0f

**12.49.2.9 Init()**

```
void daisy::DaisyVersio::Init (
 bool boost = false)
```

Initializes the Versio, and all of its hardware.

**12.49.2.10 ProcessAllControls()**

```
void daisy::DaisyVersio::ProcessAllControls () [inline]
```

Does what it says

**12.49.2.11 ProcessAnalogControls()**

```
void daisy::DaisyVersio::ProcessAnalogControls ()
```

Normalize ADC CV input. Call this once per main loop update to normalize CV input to range (0.0f, 1.0f)

**12.49.2.12 SetAudioBlockSize()**

```
void daisy::DaisyVersio::SetAudioBlockSize (
 size_t size)
```

Sets the number of samples processed per channel by the audio callback.

**12.49.2.13 SetAudioSampleRate()**

```
void daisy::DaisyVersio::SetAudioSampleRate (
 SaiHandle::Config::SampleRate samplerate)
```

Updates the Audio Sample Rate, and reinitializes. Audio must be stopped for this to work.

**12.49.2.14 SetLed()**

```
void daisy::DaisyVersio::SetLed (
 size_t idx,
 float red,
 float green,
 float blue)
```

Set an LED (idx < 4) to a color

**12.49.2.15 StartAdc()**

```
void daisy::DaisyVersio::StartAdc ()
```

Start analog to digital conversion.

**12.49.2.16 StartAudio() [1/2]**

```
void daisy::DaisyVersio::StartAudio (
 AudioHandle::AudioCallback cb)
```

Starts the callback \cb Non-interleaved callback function

**12.49.2.17 StartAudio() [2/2]**

```
void daisy::DaisyVersio::StartAudio (
 AudioHandle::InterleavingAudioCallback cb)
```

Starts the callback \cb Interleaved callback function

**12.49.2.18 StopAdc()**

```
void daisy::DaisyVersio::StopAdc ()
```

Stop converting ADCs

**12.49.2.19 StopAudio()**

```
void daisy::DaisyVersio::StopAudio ()
```

Stops the audio if it is running.

**12.49.2.20 SwitchPressed()**

```
bool daisy::DaisyVersio::SwitchPressed ()
```

Returns true if momentary switch is pressed

**12.49.2.21 UpdateLeds()**

```
void daisy::DaisyVersio::UpdateLeds ()
```

Update LED PWM state. Call this once per main loop update to correctly display led colors

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

- src/daisy\_versio.h

## 12.50 dsy\_gpio Struct Reference

```
#include <gpio.h>
```

## Public Attributes

- `dsy_gpio_pin` pin
- `dsy_gpio_mode` mode
- `dsy_gpio_pull` pull

### 12.50.1 Detailed Description

Struct for holding the pin, and configuration

### 12.50.2 Member Data Documentation

#### 12.50.2.1 mode

`dsy_gpio_mode` `dsy_gpio::mode`

&

#### 12.50.2.2 pin

`dsy_gpio_pin` `dsy_gpio::pin`

&

#### 12.50.2.3 pull

`dsy_gpio_pull` `dsy_gpio::pull`

&

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

- `src/per/gpio.h`

## 12.51 `dsy_gpio_pin` Struct Reference

```
#include <daisy_core.h>
```

## Public Attributes

- `dsy_gpio_port` port
- `uint8_t` pin

### 12.51.1 Detailed Description

Hardware define pins

The [dsy\\_gpio\\_pin](#) struct should no longer be used, and is only available for backwards compatibility.

Please use Pin struct instead.

### 12.51.2 Member Data Documentation

#### 12.51.2.1 pin

`uint8_t dsy_gpio_pin::pin`

number 0-15

#### 12.51.2.2 port

`dsy_gpio_port dsy_gpio_pin::port`

&

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

- `src/daisy_core.h`

## 12.52 DSY\_SD\_CardInfoTypeDef Struct Reference

```
#include <bsp_sd_diskio.h>
```

### Public Attributes

- `uint32_t CardType`
- `uint32_t CardVersion`
- `uint32_t Class`
- `uint32_t RelCardAdd`
- `uint32_t BlockNbr`
- `uint32_t BlockSize`
- `uint32_t LogBlockNbr`
- `uint32_t LogBlockSize`
- `uint32_t CardSpeed`

### 12.52.1 Detailed Description

Functions for handling DiskIO via SDMMC These are usually configured through the FatFS driver/interface, and won't need to be accessed directly often.

### 12.52.2 Member Data Documentation

#### 12.52.2.1 BlockNbr

```
uint32_t DSY_SD_CardInfoTypeDef::BlockNbr
```

Specifies the Card Capacity in blocks

#### 12.52.2.2 BlockSize

```
uint32_t DSY_SD_CardInfoTypeDef::BlockSize
```

Specifies one block size in bytes

#### 12.52.2.3 CardSpeed

```
uint32_t DSY_SD_CardInfoTypeDef::CardSpeed
```

Specifies the card Speed

#### 12.52.2.4 CardType

```
uint32_t DSY_SD_CardInfoTypeDef::CardType
```

Specifies the card Type

#### 12.52.2.5 CardVersion

```
uint32_t DSY_SD_CardInfoTypeDef::CardVersion
```

Specifies the card version

#### 12.52.2.6 Class

```
uint32_t DSY_SD_CardInfoTypeDef::Class
```

Specifies the class of the card class

### 12.52.2.7 LogBlockNbr

```
uint32_t DSY_SD_CardInfoTypeDef::LogBlockNbr
```

Specifies the Card logical Capacity in blocks

### 12.52.2.8 LogBlockSize

```
uint32_t DSY_SD_CardInfoTypeDef::LogBlockSize
```

Specifies logical block size in bytes

### 12.52.2.9 RelCardAdd

```
uint32_t DSY_SD_CardInfoTypeDef::RelCardAdd
```

Specifies the Relative Card Address

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

- src/util/bsp\_sd\_diskio.h

## 12.53 daisy::Encoder Class Reference

Generic Class for handling Quadrature Encoders

Inspired/influenced by Mutable Instruments (pichenettes) [Encoder](#) classes.

```
#include <encoder.h>
```

### Public Member Functions

- void [Init](#) ([dsy\\_gpio\\_pin](#) a, [dsy\\_gpio\\_pin](#) b, [dsy\\_gpio\\_pin](#) click, float update\_rate=0.f)
- void [Debounce](#) ()
- int32\_t [Increment](#) () const
- bool [RisingEdge](#) () const
- bool [FallingEdge](#) () const
- bool [Pressed](#) () const
- float [TimeHeldMs](#) () const
- void [SetUpdateRate](#) (float update\_rate)

### 12.53.1 Detailed Description

Generic Class for handling Quadrature Encoders

Inspired/influenced by Mutable Instruments (pichenettes) [Encoder](#) classes.

#### Author

Stephen Hensley

#### Date

December 2019

## 12.53.2 Member Function Documentation

### 12.53.2.1 Debounce()

```
void daisy::Encoder::Debounce ()
```

Called at update\_rate to debounce and handle timing for the switch. In order for events not to be missed, its important that the Edge/Pressed checks be made at the same rate as the debounce function is being called.

### 12.53.2.2 FallingEdge()

```
bool daisy::Encoder::FallingEdge () const [inline]
```

Returns true if the encoder was just released.

### 12.53.2.3 Increment()

```
int32_t daisy::Encoder::Increment () const [inline]
```

Returns +1 if the encoder was turned clockwise, -1 if it was turned counter-clockwise, or 0 if it was not just turned.

### 12.53.2.4 Init()

```
void daisy::Encoder::Init (
 dsy_gpio_pin a,
 dsy_gpio_pin b,
 dsy_gpio_pin click,
 float update_rate = 0.f)
```

Initializes the encoder with the specified hardware pins. Update rate is to be deprecated in a future release

### 12.53.2.5 Pressed()

```
bool daisy::Encoder::Pressed () const [inline]
```

Returns true while the encoder is held down.

### 12.53.2.6 RisingEdge()

```
bool daisy::Encoder::RisingEdge () const [inline]
```

Returns true if the encoder was just pressed.

### 12.53.2.7 SetUpdateRate()

```
void daisy::Encoder::SetUpdateRate (
 float update_rate) [inline]
```

To be removed in breaking update

**Parameters**

|                    |              |
|--------------------|--------------|
| <i>update_rate</i> | Does nothing |
|--------------------|--------------|

**12.53.2.8 TimeHeldMs()**

```
float daisy::Encoder::TimeHeldMs () const [inline]
```

Returns the time in milliseconds that the encoder has been held down.

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

- src/hid/encoder.h

## 12.54 daisy::FatFSInterface Class Reference

Daisy FatFS Driver Interface.

```
#include <fatfs.h>
```

### Classes

- struct [Config](#)

### Public Types

- enum [Result](#) { [OK](#) , [ERR\\_TOO\\_MANY\\_VOLUMES](#) , [ERR\\_NO\\_MEDIA\\_SELECTED](#) , [ERR\\_GENERIC](#) }

### Public Member Functions

- [Result Init](#) (const [Config](#) &cfg)
- [Result Init](#) (const uint8\_t media)
- [Result Delinit](#) ()
- bool [Initialized](#) () const
- const [Config](#) & [GetConfig](#) () const
- [Config](#) & [GetMutableConfig](#) ()
- const char \* [GetSDPath](#) () const
- const char \* [GetUSBPath](#) () const
- [FATFS](#) & [GetSDFileSystem](#) ()
- [FATFS](#) & [GetUSBFileSystem](#) ()

### 12.54.1 Detailed Description

Daisy FatFS Driver Interface.

Specifies the desired media (SD Card, USB, etc.) to be mountable with FatFS within a given application. Once initialization is called, the standard FatFS API will be usable with the media mounted in the order they are listed in the Media config struct. For example, when only using an SD Card, it will mount at "0:/", when only using USB, it will mount at "0:/". However, when mounting with SD card and USB, the SD card will mount at "0:/", and the USB will mount at "1:/". The relevant hardware peripheral (SDMMC, or MSD) needs to be initialized separately by the application before using FatFS.

This object has some memory limitations due to the media connected to it. The SDMMC1 peripheral can only communicate with the AXI SRAM, and the DTCMRAM cannot communicate with the DMA. So the [FatFSInterface](#) object should always be located in the AXI SRAM. This is the default location for all data/bss memory using the standard build. However, applications using the electrosmith bootloader will need special consideration when using this object AND an SD Card.

### 12.54.2 Member Enumeration Documentation

#### 12.54.2.1 Result

```
enum daisy::FatFSInterface::Result
```

Return values specifying specific errors for linking Daisy to FatFS

### 12.54.3 Member Function Documentation

#### 12.54.3.1 DeInit()

```
Result daisy::FatFSInterface::DeInit ()
```

Unlinks FatFS from the configured media

#### 12.54.3.2 GetConfig()

```
const Config& daisy::FatFSInterface::GetConfig () const [inline]
```

Return the current configuration

#### 12.54.3.3 GetMutableConfig()

```
Config& daisy::FatFSInterface::GetMutableConfig () [inline]
```

Return a mutable reference to the Configuration

#### 12.54.3.4 GetSDFileSystem()

```
FATFS& daisy::FatFSInterface::GetSDFileSystem () [inline]
```

Returns reference to filesystem object for the SD volume.

#### 12.54.3.5 GetSDPath()

```
const char* daisy::FatFSInterface::GetSDPath () const [inline]
```

Returns the path to an SD Card volume to use with f\_mount

#### 12.54.3.6 GetUSBFileSystem()

```
FATFS& daisy::FatFSInterface::GetUSBFileSystem () [inline]
```

Returns reference to filesystem object for the USB volume.

#### 12.54.3.7 GetUSBPath()

```
const char* daisy::FatFSInterface::GetUSBPath () const [inline]
```

Returns the path to a USB Device volume to use with f\_mount

#### 12.54.3.8 Init() [1/2]

```
Result daisy::FatFSInterface::Init (
 const Config & cfg)
```

Link the desired hardware specified via [Config::Media](#)

#### 12.54.3.9 Init() [2/2]

```
Result daisy::FatFSInterface::Init (
 const uint8_t media)
```

Link the desired hardware specified via [Config::Media](#)

Alternate, explicit initialization provided for simplified syntax.

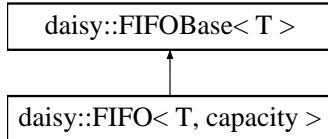
The documentation for this class was generated from the following file:

- src/sys/fatfs.h

## 12.55 daisy::FIFO< T, capacity > Class Template Reference

```
#include <FIFO.h>
```

Inheritance diagram for daisy::FIFO< T, capacity >:



### Public Member Functions

- [FIFO \(\)](#)
- [FIFO \(std::initializer\\_list< T > valuesToAdd\)](#)
- template<size\_t otherCapacity>  
[FIFO \(const FIFO< T, otherCapacity > &other\)](#)
- template<size\_t otherCapacity>  
[FIFO< T, capacity > & operator= \(const FIFO< T, otherCapacity > &other\)](#)

### Additional Inherited Members

#### 12.55.1 Detailed Description

```
template<typename T, size_t capacity>
class daisy::FIFO< T, capacity >
```

A simple [FIFO](#) ring buffer with a fixed size.

#### 12.55.2 Constructor & Destructor Documentation

##### 12.55.2.1 FIFO() [1/3]

```
template<typename T , size_t capacity>
daisy::FIFO< T, capacity >::FIFO () [inline]
```

Creates an empty [FIFO](#)

##### 12.55.2.2 FIFO() [2/3]

```
template<typename T , size_t capacity>
daisy::FIFO< T, capacity >::FIFO (
 std::initializer_list< T > valuesToAdd) [inline], [explicit]
```

Creates a [FIFO](#) and adds a list of values

### 12.55.2.3 FIFO() [3/3]

```
template<typename T , size_t capacity>
template<size_t otherCapacity>
daisy::FIFO< T, capacity >::FIFO (
 const FIFO< T, otherCapacity > & other) [inline]
```

Creates a [FIFO](#) and copies all values from another [FIFO](#)

### 12.55.3 Member Function Documentation

#### 12.55.3.1 operator=(\*)

```
template<typename T , size_t capacity>
template<size_t otherCapacity>
FIFO<T, capacity>& daisy::FIFO< T, capacity >::operator= (
 const FIFO< T, otherCapacity > & other) [inline]
```

Copies all values from another [FIFO](#)

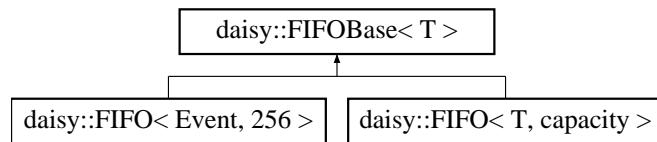
The documentation for this class was generated from the following file:

- src/util/FIFO.h

## 12.56 daisy::FIFOBase< T > Class Template Reference

```
#include <FIFO.h>
```

Inheritance diagram for daisy::FIFOBase< T >:



## Public Member Functions

- `FIFOBase< T > & operator= (const FIFOBase< T > &other)`
- `void Clear ()`
- `bool PushBack (const T &elementToAdd)`
- `int PushBack (std::initializer_list< T > valuesToAdd)`
- `T & Back ()`
- `const T & Back () const`
- `T PopFront ()`
- `T & Front ()`
- `const T & Front () const`
- `bool Contains (const T &element)`
- `size_t CountEqualTo (const T &element)`
- `bool IsEmpty () const`
- `bool IsFull () const`
- `size_t GetNumElements () const`
- `bool Insert (size_t idx, const T &element)`
- `bool Remove (size_t idx)`
- `size_t RemoveAllEqualTo (const T &element)`
- `T & operator[] (size_t idx)`
- `const T & operator[] (size_t idx) const`
- `size_t GetCapacity () const`

## Protected Member Functions

- `FIFOBase (T *buffer, size_t bufferSize)`
- `FIFOBase (T *buffer, size_t bufferSize, std::initializer_list< T > valuesToAdd)`

### 12.56.1 Detailed Description

```
template<typename T>
class daisy::FIFOBase< T >
```

Capacity-independent base class for [FIFO](#). Use [FIFO](#) instead.

### 12.56.2 Member Function Documentation

#### 12.56.2.1 Back() [1/2]

```
template<typename T >
T& daisy::FIFOBase< T >::Back () [inline]
```

returns a reference to the last element

### 12.56.2.2 Back() [2/2]

```
template<typename T >
const T& daisy::FIFOBase< T >::Back () const [inline]
```

returns a reference to the last element

### 12.56.2.3 Clear()

```
template<typename T >
void daisy::FIFOBase< T >::Clear () [inline]
```

Removes all elements from the FIFO

### 12.56.2.4 Contains()

```
template<typename T >
bool daisy::FIFOBase< T >::Contains (
 const T & element) [inline]
```

Returns true if the buffer contains an element equal to the provided value

### 12.56.2.5 CountEqualTo()

```
template<typename T >
size_t daisy::FIFOBase< T >::CountEqualTo (
 const T & element) [inline]
```

Returns the number of elements in the buffer that are equal to the provided value

### 12.56.2.6 Front() [1/2]

```
template<typename T >
T& daisy::FIFOBase< T >::Front () [inline]
```

returns a copy of the first element

### 12.56.2.7 Front() [2/2]

```
template<typename T >
const T& daisy::FIFOBase< T >::Front () const [inline]
```

returns a reference to the first element

### 12.56.2.8 GetCapacity()

```
template<typename T >
size_t daisy::FIFOBase< T >::GetCapacity () const [inline]
```

returns the total capacity

### 12.56.2.9 GetNumElements()

```
template<typename T >
size_t daisy::FIFOBase< T >::GetNumElements () const [inline]
```

returns the number of elements in the buffer

### 12.56.2.10 Insert()

```
template<typename T >
bool daisy::FIFOBase< T >::Insert (
 size_t idx,
 const T & element) [inline]
```

inserts an element "idx" positions behind the first element and returns true if successful

### 12.56.2.11 IsEmpty()

```
template<typename T >
bool daisy::FIFOBase< T >::IsEmpty () const [inline]
```

returns true, if the buffer is empty

### 12.56.2.12 IsFull()

```
template<typename T >
bool daisy::FIFOBase< T >::IsFull () const [inline]
```

returns true, if the buffer is Full

### 12.56.2.13 operator=( )

```
template<typename T >
FIFOBase<T>& daisy::FIFOBase< T >::operator= (
 const FIFOBase< T > & other) [inline]
```

Copies all elements from another FIFO

### 12.56.2.14 operator[]( ) [1/2]

```
template<typename T >
T& daisy::FIFOBase< T >::operator[] (
 size_t idx) [inline]
```

returns the element "idx" positions behind the first element

**12.56.2.15 operator[]( ) [2/2]**

```
template<typename T >
const T& daisy::FIFOBase< T >::operator[] (
 size_t idx) const [inline]
```

returns the element "idx" positions behind the first element

**12.56.2.16 PopFront()**

```
template<typename T >
T daisy::FIFOBase< T >::PopFront () [inline]
```

removes and returns an element from the front of the buffer

**12.56.2.17 PushBack() [1/2]**

```
template<typename T >
bool daisy::FIFOBase< T >::PushBack (
 const T & elementToAdd) [inline]
```

Adds an element to the back of the buffer, returning true on success

**12.56.2.18 PushBack() [2/2]**

```
template<typename T >
int daisy::FIFOBase< T >::PushBack (
 std::initializer_list< T > valuesToAdd) [inline]
```

Adds multiple elements and returns the number of elements that were added

**12.56.2.19 Remove()**

```
template<typename T >
bool daisy::FIFOBase< T >::Remove (
 size_t idx) [inline]
```

removes the element "idx" positions behind the first element and returns true if successful

**12.56.2.20 RemoveAllEqualTo()**

```
template<typename T >
size_t daisy::FIFOBase< T >::RemoveAllEqualTo (
 const T & element) [inline]
```

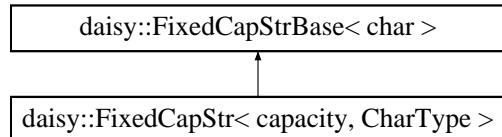
removes all elements from the buffer for which (buffer(index) == element) returns true and returns the number of elements that were removed.

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

- src/util/FIFO.h

## 12.57 daisy::FixedCapStr< capacity, CharType > Class Template Reference

Inheritance diagram for daisy::FixedCapStr< capacity, CharType >:



### Public Member Functions

- template<size\_t otherSize>  
constexpr **FixedCapStr** (const **FixedCapStr**< otherSize > &str) noexcept
- constexpr **FixedCapStr** (const CharType \*str) noexcept
- constexpr **FixedCapStr** (const CharType \*str, std::size\_t length) noexcept

### Additional Inherited Members

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

- src/util/FixedCapStr.h

## 12.58 daisy::FixedCapStrBase< CharType > Class Template Reference

### Public Member Functions

- constexpr **FixedCapStrBase** (CharType \*buffer, size\_t capacity)
- constexpr **FixedCapStrBase** & **operator=** (const **FixedCapStrBase** &str)
- constexpr **FixedCapStrBase** & **operator=** (const CharType \*str)
- constexpr **operator const CharType \* () const** noexcept
- constexpr const CharType \* **Cstr** () const noexcept
- constexpr const CharType \* **Data** () const noexcept
- constexpr CharType \* **Data** () noexcept
- constexpr auto **Size** () const noexcept
- constexpr auto **UpdateSize** () noexcept
- constexpr auto **Capacity** () const noexcept
- constexpr auto **Empty** () const noexcept
- constexpr void **Clear** () noexcept
- constexpr void **Reset** (const CharType \*str)
- constexpr void **Reset** (const CharType \*str, std::size\_t length)
- constexpr void **ResetAt** (const CharType \*str, std::size\_t writePosition)
- constexpr void **Append** (const CharType singleChar)
- constexpr void **Append** (const CharType \*str)
- constexpr void **Append** (const CharType \*str, std::size\_t length)
- template<typename IntType >  
constexpr void **AppendInt** (IntType value, bool alwaysIncludeSign=false)

- `constexpr void AppendFloat` (float value, int maxNumDigits=2, bool omitTrailingZeros=false, bool always←IncludeSign=false)
- `constexpr bool StartsWith` (const CharType \*pattern) const noexcept
- `constexpr bool StartsWithIgnoringCase` (const CharType \*pattern) const noexcept
- `constexpr bool EndsWith` (const CharType \*pattern) const noexcept
- `constexpr bool EndsWithIgnoringCase` (const CharType \*pattern) const noexcept
- `constexpr void RemovePrefix` (std::size\_t length)
- `constexpr void RemoveSuffix` (std::size\_t length) noexcept
- `constexpr void ReverseSection` (std::size\_t firstIdx, std::size\_t lastIdx)
- `constexpr bool operator==` (const CharType \*rhs) const
- `constexpr bool operator!=` (const CharType \*rhs) const
- `constexpr bool operator<` (const CharType \*other) const
- `constexpr bool operator<=` (const CharType \*other) const
- `constexpr bool operator>` (const CharType \*other) const
- `constexpr bool operator>=` (const CharType \*other) const
- `constexpr void Swap` ([FixedCapStrBase](#) &rhs) noexcept

## Protected Member Functions

- `constexpr void Reset_` (const CharType \*str, std::size\_t length)
- `constexpr void ResetAt_` (const CharType \*str, std::size\_t strLen, std::size\_t writePosition)
- `constexpr void Append_` (const CharType \*str, std::size\_t to\_copy)
- `std::size_t clamp` (std::size\_t val, std::size\_t min, std::size\_t max)

## Static Protected Member Functions

- `static constexpr std::size_t strlen` (const CharType \*string)
- `static constexpr void Copy_` (const CharType \*src, const CharType \*srcEnd, CharType \*dest)
- `static constexpr void Swap_` (CharType \*a, CharType \*b, size\_t length)
- `static constexpr char ToUpper_` (char c) noexcept

## Protected Attributes

- `std::size_t size_ {0}`
- `const size_t capacity_`
- `CharType * buffer_`

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

- `src/util/FixedCapStr.h`

## 12.59 FontDef Struct Reference

```
#include <oled_fonts.h>
```

## Public Attributes

- const uint8\_t `FontWidth`
- uint8\_t `FontHeight`
- const uint16\_t \* `data`

### 12.59.1 Detailed Description

Utility for displaying fonts on OLED displays  
Migrated to work with libdaisy from stm32-ssd1306

#### Author

afiskon on github. Font struct

### 12.59.2 Member Data Documentation

#### 12.59.2.1 `data`

const uint16\_t\* `FontDef::data`

Pointer to data font data array

#### 12.59.2.2 `FontHeight`

uint8\_t `FontDef::FontHeight`

Font height in pixels

#### 12.59.2.3 `FontWidth`

const uint8\_t `FontDef::FontWidth`

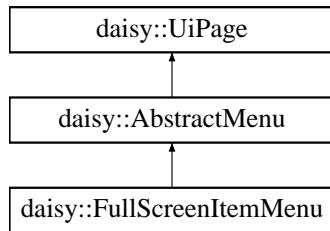
Font width in pixels

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

- src/util/oled\_fonts.h

## 12.60 daisy::FullScreenItemMenu Class Reference

Inheritance diagram for daisy::FullScreenItemMenu:



### Public Member Functions

- void `Init` (const `AbstractMenu::ItemConfig` \*items, `uint16_t` numItems, `AbstractMenu::Orientation` orientation=`AbstractMenu::Orientation::leftRightSelectUpDownModify`, bool allowEntering=true)
- void `SetOneBitGraphicsDisplayToDrawTo` (`uint16_t` canvasId)
- void `Draw` (const `UiCanvasDescriptor` &canvas) override

### Additional Inherited Members

#### 12.60.1 Member Function Documentation

##### 12.60.1.1 Draw()

```
void daisy::FullScreenItemMenu::Draw (
 const UiCanvasDescriptor & canvas) [override], [virtual]
```

Called to make the UIPage repaint everything on a canvas. Check the ID to determine which display this corresponds to. Cast the handle to the corresponding type and do your draw operations on it.

Implements [daisy::UiPage](#).

##### 12.60.1.2 Init()

```
void daisy::FullScreenItemMenu::Init (
 const AbstractMenu::ItemConfig * items,
 uint16_t numItems,
 AbstractMenu::Orientation orientation = AbstractMenu::Orientation::leftRightSelectUpDownModify,
 bool allowEntering = true)
```

Call this to initialize the menu. It's okay to re-initialize a `FullScreenItemMenu` multiple times, even while it's displayed on the [UI](#).

**Parameters**

|                      |                                                                                                                                                                                                                                                                                                                                                |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>items</i>         | An array of ItemConfig that determine which items are available in the menu.                                                                                                                                                                                                                                                                   |
| <i>numItems</i>      | The number of items in the <i>items</i> array.                                                                                                                                                                                                                                                                                                 |
| <i>orientation</i>   | Controls which pair of arrow buttons are used for selection / editing                                                                                                                                                                                                                                                                          |
| <i>allowEntering</i> | Globally controls if the Ok button can enter items for editing. If you have a physical controls that can edit selected items directly (value slider, a second arrow button pair, value encoder) you can set this to false, otherwise you set it to true so that the controls used for selecting items can now also be used to edit the values. |

**12.60.1.3 SetOneBitGraphicsDisplayToDrawTo()**

```
void daisy::FullScreenItemMenu::SetOneBitGraphicsDisplayToDrawTo (
 uint16_t canvasId)
```

Call this to change which canvas this menu will draw to. The canvas must be a [OneBitGraphicsDisplay](#), e.g. the [OledDisplay](#) class. If `canvasId == UI::invalidCanvasId` then this menu will draw to the canvas returned by [UI::GetPrimaryOneBitGraphicsDisplayId\(\)](#). This is also the default behaviour.

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

- src/ui/FullScreenItemMenu.h

**12.61 daisy::GateIn Class Reference**

Generic Class for handling gate inputs through [GPIO](#).

```
#include <gatein.h>
```

**Public Member Functions**

- [GateIn \(\)](#)
- [~GateIn \(\)](#)
- void [Init \(dsy\\_gpio\\_pin \\*pin\\_cfg\)](#)
- bool [Trig \(\)](#)
- bool [State \(\)](#)

**12.61.1 Detailed Description**

Generic Class for handling gate inputs through [GPIO](#).

**Author**

Stephen Hensley

**Date**

March 2020

## 12.61.2 Constructor & Destructor Documentation

### 12.61.2.1 GateIn()

```
daisy::GateIn::GateIn () [inline]
```

GateIn Constructor

### 12.61.2.2 ~GateIn()

```
daisy::GateIn::~GateIn () [inline]
```

GateIn~ Destructor

## 12.61.3 Member Function Documentation

### 12.61.3.1 Init()

```
void daisy::GateIn::Init (
 dsy_gpio_pin * pin_cfg)
```

Init Initializes the gate input with specified hardware pin

### 12.61.3.2 State()

```
bool daisy::GateIn::State () [inline]
```

State Checks current state of gate input (no state required)

read function is inverted because of suggested BJT input circuit

### 12.61.3.3 Trig()

```
bool daisy::GateIn::Trig ()
```

Trig Checks current state of gate input.

#### Returns

True if the [GPIO](#) just transitioned.

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

- src/hid/gatein.h

## 12.62 daisy::GPIO Class Reference

General Purpose I/O control.

```
#include <gpio.h>
```

### Classes

- struct [Config](#)  
*Configuration for a given [GPIO](#).*

### Public Types

- enum class [Mode](#) { [INPUT](#), [OUTPUT](#), [OUTPUT\\_OD](#), [ANALOG](#) }  
*Mode of operation for the specified [GPIO](#).*
- enum class [Pull](#) { [NOPULL](#), [PULLUP](#), [PULLDOWN](#) }  
*Configures whether an internal Pull up or Pull down resistor is used.*
- enum class [Speed](#) { [LOW](#), [MEDIUM](#), [HIGH](#), [VERY\\_HIGH](#) }  
*Output speed controls the drive strength, and slew rate of the pin.*

### Public Member Functions

- void [Init](#) (const [Config](#) &cfg)  
*Initialize the [GPIO](#) from a Configuration struct.*
- void [Init](#) ([Pin](#) p, const [Config](#) &cfg)  
*Initialize the [GPIO](#) with a Configuration struct, and explicit pin.*
- void [Init](#) ([Pin](#) p, [Mode](#) m=[Mode::INPUT](#), [Pull](#) pu=[Pull::NOPULL](#), [Speed](#) sp=[Speed::LOW](#))  
*Explicitly initialize all configuration for the [GPIO](#).*
- void [Delinit](#) ()  
*Deinitializes the [GPIO](#) pin.*
- bool [Read](#) ()  
*Reads the state of the [GPIO](#). If a [GPIO](#) is configured in [Config::Mode::ANALOG](#), this function will always return false.*
- void [Write](#) (bool state)  
*Changes the state of the [GPIO](#) hardware when configured as an [OUTPUT](#).*
- void [Toggle](#) ()  
*flips the current state of the [GPIO](#). If it was [HIGH](#), it will go [LOW](#), and vice versa.*
- [Config](#) & [GetConfig](#) ()

### 12.62.1 Detailed Description

General Purpose I/O control.

### 12.62.2 Member Enumeration Documentation

#### 12.62.2.1 Mode

```
enum daisy::GPIO::Mode [strong]
```

Mode of operation for the specified [GPIO](#).

**Enumerator**

|           |                                                |
|-----------|------------------------------------------------|
| INPUT     | Input for reading state of pin                 |
| OUTPUT    | Output w/ push-pull configuration              |
| OUTPUT_OD | Output w/ open-drain configuration             |
| ANALOG    | Analog for connection to ADC or DAC peripheral |

**12.62.2.2 Pull**

```
enum daisy::GPIO::Pull [strong]
```

Configures whether an internal Pull up or Pull down resistor is used.

Internal Pull up/down resistors are typically 40k ohms, and will be between 30k and 50k

When the [Pin](#) is configured in Analog mode, the pull up/down resistors are disabled by hardware.

**Enumerator**

|          |                            |
|----------|----------------------------|
| NOPULL   | No pull up resistor        |
| PULLUP   | Internal pull up enabled   |
| PULLDOWN | Internal pull down enabled |

**12.62.3 Member Function Documentation****12.62.3.1 GetConfig()**

```
Config& daisy::GPIO::GetConfig() [inline]
```

Return a reference to the internal [Config](#) struct

**12.62.3.2 Write()**

```
void daisy::GPIO::Write(bool state)
```

Changes the state of the [GPIO](#) hardware when configured as an OUTPUT.

**Parameters**

|                    |                                                                               |
|--------------------|-------------------------------------------------------------------------------|
| <code>state</code> | setting true writes an output HIGH, while setting false writes an output LOW. |
|--------------------|-------------------------------------------------------------------------------|

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

- src/per/gpio.h

## 12.63 daisy::I2CHandle Class Reference

```
#include <i2c.h>
```

### Classes

- struct [Config](#)

### Public Types

- enum class [Result](#) { [OK](#) , [ERR](#) }
- enum class [Direction](#) { [TRANSMIT](#) , [RECEIVE](#) }
- typedef void(\* [CallbackFunctionPtr](#)) (void \*context, [Result](#) result)

### Public Member Functions

- [I2CHandle](#) (const [I2CHandle](#) &other)=default
- [I2CHandle](#) & [operator=](#) (const [I2CHandle](#) &other)=default
- [Result](#) [Init](#) (const [Config](#) &config)
- const [Config](#) & [GetConfig](#) () const
- [Result](#) [TransmitBlocking](#) (uint16\_t address, uint8\_t \*data, uint16\_t size, uint32\_t timeout)
- [Result](#) [ReceiveBlocking](#) (uint16\_t address, uint8\_t \*data, uint16\_t size, uint32\_t timeout)
- [Result](#) [TransmitDma](#) (uint16\_t address, uint8\_t \*data, uint16\_t size, [CallbackFunctionPtr](#) callback, void \*callback\_context)
- [Result](#) [ReceiveDma](#) (uint16\_t address, uint8\_t \*data, uint16\_t size, [CallbackFunctionPtr](#) callback, void \*callback\_context)
- [Result](#) [ReadDataAtAddress](#) (uint16\_t address, uint16\_t mem\_address, uint16\_t mem\_address\_size, uint8\_t \*data, uint16\_t data\_size, uint32\_t timeout)
- [Result](#) [WriteDataAtAddress](#) (uint16\_t address, uint16\_t mem\_address, uint16\_t mem\_address\_size, uint8\_t \*data, uint16\_t data\_size, uint32\_t timeout)

### 12.63.1 Detailed Description

A handle for interacting with an I2C peripheral. This is a dumb gateway that internally points to one of the four I2C peripherals after it was initialised. It can then be copied and passed around. Use an [I2CHandle](#) like this:

```
// setup the configuration
I2CHandle::Config i2c_conf;
i2c_conf.periph = I2CHandle::Config::Peripheral::I2C_1;
i2c_conf.speed = I2CHandle::Config::Speed::I2C_400KHZ;
i2c_conf.mode = I2CHandle::Config::Mode::Master;
i2c_conf.pin_config.scl = {DSY_GPIOB, 8};
i2c_conf.pin_config.sda = {DSY_GPIOB, 9};
// initialise the peripheral
I2CHandle i2c;
i2c.Init(i2c_conf);
// now i2c points to the corresponding peripheral and can be used.
i2c.TransmitBlocking(...);
```

## 12.63.2 Member Typedef Documentation

### 12.63.2.1 CallbackFunctionPtr

```
typedef void(* daisy::I2CHandle::CallbackFunctionPtr) (void *context, Result result)
```

A callback to be executed when a dma transfer is complete.

## 12.63.3 Member Enumeration Documentation

### 12.63.3.1 Direction

```
enum daisy::I2CHandle::Direction [strong]
```

Enumerator

|          |   |
|----------|---|
| TRANSMIT | & |
| RECEIVE  | & |

### 12.63.3.2 Result

```
enum daisy::I2CHandle::Result [strong]
```

Return values for I2C functions.

Enumerator

|     |   |
|-----|---|
| OK  | & |
| ERR | & |

## 12.63.4 Member Function Documentation

### 12.63.4.1 GetConfig()

```
const Config& daisy::I2CHandle::GetConfig () const
```

Returns the current config.

#### 12.63.4.2 Init()

```
Result daisy::I2CHandle::Init (
 const Config & config)
```

Initializes an I2C peripheral.

#### 12.63.4.3 ReadDataAtAddress()

```
Result daisy::I2CHandle::ReadDataAtAddress (
 uint16_t address,
 uint16_t mem_address,
 uint16_t mem_address_size,
 uint8_t * data,
 uint16_t data_size,
 uint32_t timeout)
```

Reads an amount of data from a specific memory address. This method will return an error if the I2C peripheral is in slave mode.

##### Parameters

|                         |                                                                           |
|-------------------------|---------------------------------------------------------------------------|
| <i>address</i>          | The slave device address.                                                 |
| <i>mem_address</i>      | Pointer to data containing the address to read from device.               |
| <i>mem_address_size</i> | Size of the memory address in bytes.                                      |
| <i>data</i>             | Pointer to buffer that will be filled with contents at <i>mem_address</i> |
| <i>data_size</i>        | Size of the data to be read in bytes.                                     |

#### 12.63.4.4 ReceiveBlocking()

```
Result daisy::I2CHandle::ReceiveBlocking (
 uint16_t address,
 uint8_t * data,
 uint16_t size,
 uint32_t timeout)
```

Receives data and blocks until the reception is complete. Use this for smaller transmissions of a few bytes.

##### Parameters

|                |                                                 |
|----------------|-------------------------------------------------|
| <i>address</i> | The slave device address. Unused in slave mode. |
| <i>data</i>    | A pointer to the data to be received.           |
| <i>size</i>    | The size of the data to be received, in bytes.  |
| <i>timeout</i> | A timeout.                                      |

#### 12.63.4.5 ReceiveDma()

```
Result daisy::I2CHandle::ReceiveDma (
 uint16_t address,
 uint8_t * data,
 uint16_t size,
 CallbackFunctionPtr callback,
 void * callback_context)
```

Receives data with a DMA and returns immediately. Use this for larger transmissions. The pointer to data must be located in the D2 memory domain by adding the DMA\_BUFFER\_MEM\_SECTION attribute like this: `uint8_t DMA↔_BUFFER_MEM_SECTION my_buffer[100];` If that is not possible for some reason, you MUST clear the cachelines spanning the size of the buffer, before initiating the dma transfer by calling `dsy_dma_clear_cache_for↔buffer(buffer, size);`

A single DMA is shared across I2C, I2C2 and I2C3. I2C4 has no DMA support (yet). If the DMA is busy with another transfer, the job will be queued and executed later. If there is a job waiting to be executed for this I2C peripheral, this function will block until the queue is free and the job can be queued.

##### Parameters

|                               |                                                            |
|-------------------------------|------------------------------------------------------------|
| <code>address</code>          | The slave device address. Unused in slave mode.            |
| <code>data</code>             | A pointer to the data buffer.                              |
| <code>size</code>             | The size of the data to be received, in bytes.             |
| <code>callback</code>         | A callback to execute when the transfer finishes, or NULL. |
| <code>callback_context</code> | A pointer that will be passed back to you in the callback. |

#### 12.63.4.6 TransmitBlocking()

```
Result daisy::I2CHandle::TransmitBlocking (
 uint16_t address,
 uint8_t * data,
 uint16_t size,
 uint32_t timeout)
```

Transmits data and blocks until the transmission is complete. Use this for smaller transmissions of a few bytes.

##### Parameters

|                      |                                                 |
|----------------------|-------------------------------------------------|
| <code>address</code> | The slave device address. Unused in slave mode. |
| <code>data</code>    | A pointer to the data to be sent.               |
| <code>size</code>    | The size of the data to be sent, in bytes.      |
| <code>timeout</code> | A timeout.                                      |

#### 12.63.4.7 TransmitDma()

```
Result daisy::I2CHandle::TransmitDma (
```

```
uint16_t address,
uint8_t * data,
uint16_t size,
CallbackFunctionPtr callback,
void * callback_context)
```

Transmits data with a DMA and returns immediately. Use this for larger transmissions. The pointer to data must be located in the D2 memory domain by adding the DMA\_BUFFER\_MEM\_SECTION attribute like this: uint8\_t DMA↔\_BUFFER\_MEM\_SECTION my\_buffer[100]; If that is not possible for some reason, you MUST clear the cachelines spanning the size of the buffer, before initiating the dma transfer by calling dsy\_dma\_clear\_cache\_for←buffer(buffer, size);

A single DMA is shared across I2C1, I2C2 and I2C3. I2C4 has no DMA support (yet). If the DMA is busy with another transfer, the job will be queued and executed later. If there is a job waiting to be executed for this I2C peripheral, this function will block until the queue is free and the job can be queued.

#### Parameters

|                         |                                                            |
|-------------------------|------------------------------------------------------------|
| <i>address</i>          | The slave device address. Unused in slave mode.            |
| <i>data</i>             | A pointer to the data to be sent.                          |
| <i>size</i>             | The size of the data to be sent, in bytes.                 |
| <i>callback</i>         | A callback to execute when the transfer finishes, or NULL. |
| <i>callback_context</i> | A pointer that will be passed back to you in the callback. |

#### 12.63.4.8 WriteDataAtAddress()

```
Result daisy::I2CHandle::WriteDataAtAddress (
 uint16_t address,
 uint16_t mem_address,
 uint16_t mem_address_size,
 uint8_t * data,
 uint16_t data_size,
 uint32_t timeout)
```

Writes an amount of data from a specific memory address. This method will return an error if the I2C peripheral is in slave mode.

#### Parameters

|                         |                                                            |
|-------------------------|------------------------------------------------------------|
| <i>address</i>          | The slave device address.                                  |
| <i>mem_address</i>      | Pointer to data containing the address to write to device. |
| <i>mem_address_size</i> | Size of the memory address in bytes.                       |
| <i>data</i>             | Pointer to buffer that will be written to the mem_address  |
| <i>data_size</i>        | Size of the data to be written in bytes.                   |

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

- src/per/i2c.h

## 12.64 daisy::AbstractMenu::ItemConfig Struct Reference

### Public Attributes

- `ItemType type = ItemType::closeMenuItem`
- `const char * text = ""`
- `union {`
- `struct {`
- `void(* callbackFunction )(void *context)`
- `void * context`
- `} asCallbackFunctionItem`
- `struct {`
- `bool * valueToModify`
- `} asCheckboxItem`
- `struct {`
- `MappedValue * valueToModify`
- `} asMappedValueItem`
- `struct {`
- `UiPage * pageToOpen`
- `} asOpenUiPageItem`
- `struct {`
- `CustomItem * itemObject`
- `} asCustomItem`
- `};`

### 12.64.1 Member Data Documentation

#### 12.64.1.1

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

additional properties that depend on the value of `type`

#### 12.64.1.2

```
struct { ... } daisy::AbstractMenu::ItemConfig::asCallbackFunctionItem
```

Properties for type == `ItemType::callbackFunctionItem`

#### 12.64.1.3

```
struct { ... } daisy::AbstractMenu::ItemConfig::asCheckboxItem
```

Properties for type == `ItemType::checkboxItem`

#### 12.64.1.4

```
struct { ... } daisy::AbstractMenu::ItemConfig::asCustomItem
```

Properties for type == [ItemType::customItem](#)

#### 12.64.1.5

```
struct { ... } daisy::AbstractMenu::ItemConfig::asMappedValueItem
```

Properties for type == [ItemType::valueItem](#)

#### 12.64.1.6

```
struct { ... } daisy::AbstractMenu::ItemConfig::asOpenUiPageItem
```

Properties for type == [ItemType::openUiPageItem](#)

#### 12.64.1.7 itemObject

```
CustomItem* daisy::AbstractMenu::ItemConfig::itemObject
```

The [CustomItem](#) to display. The object provided here must stay alive longer than the MenuPage, e.g. as a global variable.

#### 12.64.1.8 pageToOpen

```
UiPage* daisy::AbstractMenu::ItemConfig::pageToOpen
```

The [UiPage](#) to open when the okay button is pressed. The object must stay alive longer than the MenuPage, e.g. as a global variable.

#### 12.64.1.9 text

```
const char* daisy::AbstractMenu::ItemConfig::text = ""
```

The name/text to display

#### 12.64.1.10 type

```
ItemType daisy::AbstractMenu::ItemConfig::type = ItemType::closeMenuItem
```

The type of item

### 12.64.1.11 valueToModify [1/2]

```
bool* daisy::AbstractMenu::ItemConfig::valueToModify
```

The variable to modify.

### 12.64.1.12 valueToModify [2/2]

```
MappedValue* daisy::AbstractMenu::ItemConfig::valueToModify
```

The variable to modify.

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

- src/ui/AbstractMenu.h

## 12.65 daisy::LcdHD44780 Class Reference

### Classes

- struct [Config](#)

### Public Member Functions

- void [Init](#) (const [Config](#) &config)
- void [Print](#) (const char \*string)
- void [PrintInt](#) (int number)
- void [SetCursor](#) (uint8\_t row, uint8\_t col)
- void [Clear](#) ()

### 12.65.1 Member Function Documentation

#### 12.65.1.1 Clear()

```
void daisy::LcdHD44780::Clear ()
```

Clears the contents of the LCD.

#### 12.65.1.2 Init()

```
void daisy::LcdHD44780::Init (
 const Config & config)
```

Initializes the LCD.

**Parameters**

|               |                                                                                                                            |
|---------------|----------------------------------------------------------------------------------------------------------------------------|
| <i>config</i> | is a struct that sets cursor on/off, cursor blink on/off and the <a href="#">dsy_gpio_pin</a> 's that connects to the LCD. |
|---------------|----------------------------------------------------------------------------------------------------------------------------|

**12.65.1.3 Print()**

```
void daisy::LcdHD44780::Print (
 const char * string)
```

Prints a string on the LCD.

**Parameters**

|               |                                   |
|---------------|-----------------------------------|
| <i>string</i> | is a C-formatted string to print. |
|---------------|-----------------------------------|

**12.65.1.4 PrintInt()**

```
void daisy::LcdHD44780::PrintInt (
 int number)
```

Prints an integer value on the LCD.

**Parameters**

|               |                         |
|---------------|-------------------------|
| <i>number</i> | is an integer to print. |
|---------------|-------------------------|

**12.65.1.5 SetCursor()**

```
void daisy::LcdHD44780::SetCursor (
 uint8_t row,
 uint8_t col)
```

Moves the cursor of the LCD (the place to print the next value).

**Parameters**

|            |                                 |
|------------|---------------------------------|
| <i>row</i> | is the row number (0 or 1).     |
| <i>col</i> | is the column number (0 to 15). |

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

- src/dev/lcd\_hd44780.h

## 12.66 daisy::Led Class Reference

LED Class providing simple Software PWM ability, etc

Eventually this will work with hardware PWM, and external LED Driver devices as well.

```
#include <led.h>
```

### Public Member Functions

- void [Init \(dsy\\_gpio\\_pin pin, bool invert, float samplerate=1000.0f\)](#)
- void [Set \(float val\)](#)
- void [Update \(\)](#)
- void [SetSampleRate \(float sample\\_rate\)](#)

### 12.66.1 Detailed Description

LED Class providing simple Software PWM ability, etc

Eventually this will work with hardware PWM, and external LED Driver devices as well.

#### Author

shensley

#### Date

March 2020

### 12.66.2 Member Function Documentation

#### 12.66.2.1 Init()

```
void daisy::Led::Init (
 dsy_gpio_pin pin,
 bool invert,
 float samplerate = 1000.0f)
```

Initializes an LED using the specified hardware pin.

#### Parameters

|                   |                                                                                            |
|-------------------|--------------------------------------------------------------------------------------------|
| <i>pin</i>        | chooses LED pin                                                                            |
| <i>invert</i>     | will set whether to internally invert the brightness due to hardware config.               |
| <i>samplerate</i> | sets the rate at which ' <a href="#">Update()</a> ' will be called (used for software PWM) |

### 12.66.2.2 Set()

```
void daisy::Led::Set (
 float val)
```

Sets the brightness of the [Led](#).

#### Parameters

|            |                                                                                                                                                                                              |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>val</i> | will be cubed for gamma correction, and then quantized to 8-bit values for Software PWM 8-bit is for more flexible update rate options, as 12-bit or more would require faster update rates. |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

### 12.66.2.3 SetSampleRate()

```
void daisy::Led::SetSampleRate (
 float sample_rate) [inline]
```

Set the rate at which you'll update the leds without reiniting the led

#### Parameters

|                    |                        |
|--------------------|------------------------|
| <i>sample_rate</i> | New update rate in hz. |
|--------------------|------------------------|

### 12.66.2.4 Update()

```
void daisy::Led::Update ()
```

This processes the pwm of the LED sets the hardware accordingly.

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

- src/hid/led.h

## 12.67 daisy::LedDriverPca9685< numDrivers, persistentBufferContents > Class Template Reference

```
#include <leddriver.h>
```

## Public Types

- using `DmaBuffer` = PCA9685TransmitBuffer[numDrivers]

## Public Member Functions

- struct `__attribute__((packed))` PCA9685TransmitBuffer
- void `Init(I2CHandle i2c, const uint8_t(&addresses)[numDrivers], DmaBuffer dma_buffer_a, DmaBuffer dma_buffer_b, dsy_gpio_pin oe_pin={DSY_GPIOX, 0})`
- constexpr int `GetNumLeds ()` const
- void `SetAllTo (float brightness)`
- void `SetAllTo (uint8_t brightness)`
- void `SetAllToRaw (uint16_t rawBrightness)`
- void `SetLed (int ledIndex, float brightness)`
- void `SetLed (int ledIndex, uint8_t brightness)`
- void `SetLedRaw (int ledIndex, uint16_t rawBrightness)`
- void `SwapBuffersAndTransmit ()`

### 12.67.1 Detailed Description

```
template<int numDrivers, bool persistentBufferContents = true>
class daisy::LedDriverPca9685< numDrivers, persistentBufferContents >
```

LED driver for one or multiple PCA9685 12bit PWM chips connected to a single I2C peripheral. It includes gamma correction from 8bit brightness values but it can also be supplied with raw 12bit values. This driver uses two buffers - one for drawing, one for transmitting. Multiple `LedDriverPca9685` instances can be used at the same time.

#### Parameters

|                                       |                                                                                                                                                                                                                                                                                                                                        |
|---------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>numDrivers</code>               | The number of PCA9685 driver attached to the I2C peripheral.                                                                                                                                                                                                                                                                           |
| <code>persistentBufferContents</code> | If set to true, the current draw buffer contents will be copied to the next draw buffer during <code>SwapBuffersAndTransmit()</code> . Use this, if you plan to write single leds at a time. If you will always update all leds before calling <code>SwapBuffersAndTransmit()</code> , you can set this to false and save some cycles. |

### 12.67.2 Member Typedef Documentation

#### 12.67.2.1 DmaBuffer

```
template<int numDrivers, bool persistentBufferContents = true>
using daisy::LedDriverPca9685< numDrivers, persistentBufferContents >::DmaBuffer = PCA9685< TransmitBuffer[numDrivers]
```

Buffer type for the entire DMA buffer.

## 12.67.3 Member Function Documentation

### 12.67.3.1 `__attribute__()`

```
template<int numDrivers, bool persistentBufferContents = true>
struct daisy::LedDriverPca9685< numDrivers, persistentBufferContents >::__attribute__ (
 (packed)) [inline]
```

Buffer Type for a single PCA9685 driver chip. register address

cycle at which to switch on the led

cycle at which to switch off the led

full size in bytes

### 12.67.3.2 `GetNumLeds()`

```
template<int numDrivers, bool persistentBufferContents = true>
constexpr int daisy::LedDriverPca9685< numDrivers, persistentBufferContents >::GetNumLeds ()
const [inline], [constexpr]
```

Returns the number of leds available from this driver.

### 12.67.3.3 `Init()`

```
template<int numDrivers, bool persistentBufferContents = true>
void daisy::LedDriverPca9685< numDrivers, persistentBufferContents >::Init (
 I2CHandle i2c,
 const uint8_t (&) addresses[numDrivers],
 DmaBuffer dma_buffer_a,
 DmaBuffer dma_buffer_b,
 dsy_gpio_pin oe_pin = {DSY_GPIOX, 0}) [inline]
```

Initialises the driver.

#### Parameters

|                           |                                                                                                                                                                                                                |
|---------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>i2c</code>          | The I2C peripheral to use.                                                                                                                                                                                     |
| <code>addresses</code>    | An array of addresses for each of the driver chips.                                                                                                                                                            |
| <code>dma_buffer_a</code> | The first buffer for the DMA. This must be placed in D2 memory by adding the DMA_BUFFER_MEM_SECTION attribute like this:<br><code>LedDriverPca9685&lt;2&gt;::DmaBuffer DMA_BUFFER_MEM_SECTION bufferA;</code>  |
| <code>dma_buffer_b</code> | The second buffer for the DMA. This must be placed in D2 memory by adding the DMA_BUFFER_MEM_SECTION attribute like this:<br><code>LedDriverPca9685&lt;2&gt;::DmaBuffer DMA_BUFFER_MEM_SECTION bufferB;</code> |
| <code>oe_pin</code>       | If the output enable pin is used, supply its configuration here. It will automatically be pulled low by the driver.                                                                                            |

#### 12.67.3.4 SetAllTo() [1/2]

```
template<int numDrivers, bool persistentBufferContents = true>
void daisy::LedDriverPca9685< numDrivers, persistentBufferContents >::SetAllTo (
 float brightness) [inline]
```

Sets all leds to a gamma corrected brightness between 0.0f and 1.0f.

#### 12.67.3.5 SetAllTo() [2/2]

```
template<int numDrivers, bool persistentBufferContents = true>
void daisy::LedDriverPca9685< numDrivers, persistentBufferContents >::SetAllTo (
 uint8_t brightness) [inline]
```

Sets all leds to a gamma corrected brightness between 0 and 255.

#### 12.67.3.6 SetAllToRaw()

```
template<int numDrivers, bool persistentBufferContents = true>
void daisy::LedDriverPca9685< numDrivers, persistentBufferContents >::SetAllToRaw (
 uint16_t rawBrightness) [inline]
```

Sets all leds to a raw 12bit brightness between 0 and 4095.

#### 12.67.3.7 SetLed() [1/2]

```
template<int numDrivers, bool persistentBufferContents = true>
void daisy::LedDriverPca9685< numDrivers, persistentBufferContents >::SetLed (
 int ledIndex,
 float brightness) [inline]
```

Sets a single led to a gamma corrected brightness between 0.0f and 1.0f.

#### 12.67.3.8 SetLed() [2/2]

```
template<int numDrivers, bool persistentBufferContents = true>
void daisy::LedDriverPca9685< numDrivers, persistentBufferContents >::SetLed (
 int ledIndex,
 uint8_t brightness) [inline]
```

Sets a single led to a gamma corrected brightness between 0 and 255.

### 12.67.3.9 SetLedRaw()

```
template<int numDrivers, bool persistentBufferContents = true>
void daisy::LedDriverPca9685< numDrivers, persistentBufferContents >::SetLedRaw (
 int ledIndex,
 uint16_t rawBrightness) [inline]
```

Sets a single led to a raw 12bit brightness between 0 and 4095.

### 12.67.3.10 SwapBuffersAndTransmit()

```
template<int numDrivers, bool persistentBufferContents = true>
void daisy::LedDriverPca9685< numDrivers, persistentBufferContents >::SwapBuffersAndTransmit (
) [inline]
```

Swaps the current draw buffer and the current transmit buffer and starts transmitting the values to all chips.

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

- src/dev/leddriver.h

## 12.68 daisy::LocalControlEvent Struct Reference

```
#include <MidiEvent.h>
```

### Public Attributes

- int [channel](#)
- bool [local\\_control\\_off](#)
- bool [local\\_control\\_on](#)

### 12.68.1 Detailed Description

Struct containing [LocalControlEvent](#) data. Can be made from [MidiEvent](#)

### 12.68.2 Member Data Documentation

#### 12.68.2.1 channel

```
int daisy::LocalControlEvent::channel
```

&

### 12.68.2.2 local\_control\_off

```
bool daisy::LocalControlEvent::local_control_off
&
```

### 12.68.2.3 local\_control\_on

```
bool daisy::LocalControlEvent::local_control_on
&
```

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

- src/hid/MidiEvent.h

## 12.69 daisy::Logger< dest > Class Template Reference

Interface for simple USB logging.

```
#include <logger.h>
```

### Public Member Functions

- [Logger \(\)](#)

### Static Public Member Functions

- static void [Print](#) (const char \*format,...)
- static void [PrintLine](#) (const char \*format,...)
- static void [StartLog](#) (bool wait\_for\_pc=false)
- static void [PrintV](#) (const char \*format, va\_list va)
- static void [PrintLineV](#) (const char \*format, va\_list va)

### Protected Types

- enum [LoggerConsts](#) { [LOGGER\\_SYNC\\_OUT](#) = 0 , [LOGGER\\_SYNC\\_IN](#) = 2 }

### Static Protected Member Functions

- static void [TransmitSync](#) (const void \*buffer, size\_t bytes)
- static void [TransmitBuf](#) ()
- static void [AppendNewLine](#) ()
- static constexpr size\_t [NewLineSeqLength](#) ()

## Static Protected Attributes

- static char `tx_buff_` [128]
- static size\_t `tx_ptr_` = 0
- static size\_t `pc_sync_` = LOGGER\_SYNC\_OUT
- static `LoggerImpl< dest > impl_`

### 12.69.1 Detailed Description

```
template<LoggerDestination dest = LOGGER_INTERNAL>
class daisy::Logger< dest >
```

Interface for simple USB logging.

#### Author

Alexander Petrov-Savchenko ( [axp@soft-amp.com](mailto:axp@soft-amp.com))

#### Date

November 2020

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

- src/hid/logger.h

## 12.70 daisy::Logger< LOGGER\_NONE > Class Reference

```
#include <logger.h>
```

### Static Public Member Functions

- static void **Print** (const char \*format,...)
- static void **PrintLine** (const char \*format,...)
- static void **StartLog** (bool wait\_for\_pc=false)
- static void **PrintV** (const char \*format, va\_list va)
- static void **PrintLineV** (const char \*format, va\_list va)

### 12.70.1 Detailed Description

Specialization for a muted log

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

- src/hid/logger.h

## 12.71 daisy::LoggerImpl< dest > Class Template Reference

Logging I/O underlying implementation.

```
#include <logger_impl.h>
```

### Static Public Member Functions

- static void [Init \(\)](#)
- static bool [Transmit \(const void \\*buffer, size\\_t bytes\)](#)

#### 12.71.1 Detailed Description

```
template<LoggerDestination dest>
class daisy::LoggerImpl< dest >
```

Logging I/O underlying implementation.

#### Author

Alexander Petrov-Savchenko ( [axp@soft-amp.com](mailto:axp@soft-amp.com))

#### Date

November 2020

### 12.71.2 Member Function Documentation

#### 12.71.2.1 Init()

```
template<LoggerDestination dest>
static void daisy::LoggerImpl< dest >::Init () [inline], [static]
```

Initialize logging destination

#### 12.71.2.2 Transmit()

```
template<LoggerDestination dest>
static bool daisy::LoggerImpl< dest >::Transmit (
 const void * buffer,
 size_t bytes) [inline], [static]
```

Transmit a block of data

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

- src/hid/logger\_impl.h

## 12.72 daisy::LoggerImpl< LOGGER\_EXTERNAL > Class Reference

Specialization for external USB port.

```
#include <logger_impl.h>
```

### Static Public Member Functions

- static void [Init \(\)](#)
- static bool [Transmit \(const void \\*buffer, size\\_t bytes\)](#)

### Static Protected Attributes

- static [UsbHandle usb\\_handle\\_](#)

#### 12.72.1 Detailed Description

Specialization for external USB port.

#### 12.72.2 Member Function Documentation

##### 12.72.2.1 Init()

```
static void daisy::LoggerImpl< LOGGER_EXTERNAL >::Init () [inline], [static]
```

Initialize logging destination this implementation relies on the fact that [UsbHandle](#) class has no member variables and can be shared. assert this statement:

##### 12.72.2.2 Transmit()

```
static bool daisy::LoggerImpl< LOGGER_EXTERNAL >::Transmit (
 const void * buffer,
 size_t bytes) [inline], [static]
```

Transmit a block of data

#### 12.72.3 Member Data Documentation

### 12.72.3.1 usb\_handle\_

UsbHandle daisy::LoggerImpl< LOGGER\_EXTERNAL >::usb\_handle\_ [static], [protected]

USB Handle for CDC transfers

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

- src/hid/logger\_impl.h

## 12.73 daisy::LoggerImpl< LOGGER\_INTERNAL > Class Reference

Specialization for internal USB port.

```
#include <logger_impl.h>
```

### Static Public Member Functions

- static void [Init\(\)](#)
- static bool [Transmit](#)(const void \*buffer, size\_t bytes)

### Static Protected Attributes

- static UsbHandle [usb\\_handle\\_](#)

### 12.73.1 Detailed Description

Specialization for internal USB port.

### 12.73.2 Member Function Documentation

#### 12.73.2.1 Init()

```
static void daisy::LoggerImpl< LOGGER_INTERNAL >::Init () [inline], [static]
```

Initialize logging destination this implementation relies on the fact that [UsbHandle](#) class has no member variables and can be shared assert this statement:

#### 12.73.2.2 Transmit()

```
static bool daisy::LoggerImpl< LOGGER_INTERNAL >::Transmit (
 const void * buffer,
 size_t bytes) [inline], [static]
```

Transmit a block of data

### 12.73.3 Member Data Documentation

#### 12.73.3.1 `usb_handle_`

```
UsbHandle daisy::LoggerImpl< LOGGER_INTERNAL >::usb_handle_ [static], [protected]
```

USB Handle for CDC transfers

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

- src/hid/logger\_impl.h

## 12.74 `daisy::LoggerImpl< LOGGER_SEMIHOST >` Class Reference

Specialization for semihosting (stdout)

```
#include <logger_impl.h>
```

### Static Public Member Functions

- static void `Init` ()
- static bool `Transmit` (const void \*buffer, size\_t bytes)

#### 12.74.1 Detailed Description

Specialization for semihosting (stdout)

#### 12.74.2 Member Function Documentation

##### 12.74.2.1 `Init()`

```
static void daisy::LoggerImpl< LOGGER_SEMIHOST >::Init () [inline], [static]
```

Initialize logging destination

### 12.74.2.2 Transmit()

```
static bool daisy::LoggerImpl< LOGGER_SEMIHOST >::Transmit (
 const void * buffer,
 size_t bytes) [inline], [static]
```

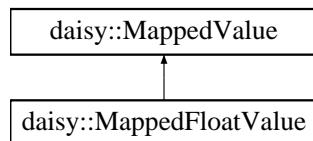
Transmit a block of data

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

- src/hid/logger\_impl.h

## 12.75 daisy::MappedFloatValue Class Reference

Inheritance diagram for daisy::MappedFloatValue:



### Public Types

- enum class [Mapping](#) { [lin](#) , [log](#) , [pow2](#) }

### Public Member Functions

- [MappedFloatValue](#) (float min, float max, float defaultValue, [Mapping](#) mapping=[Mapping::lin](#), const char \*unitStr="", uint8\_t numDecimals=1, bool forceSign=false)
- float [Get](#) () const
- const float \* [GetPtr](#) () const
- void [Set](#) (float newValue)
- [operator float](#) () const
- [MappedFloatValue & operator=](#) (float val)
- void [AppendToString](#) ([FixedCapStrBase< char >](#) &string) const override
- void [ResetToDefault](#) () override
- float [GetAs0to1](#) () const override
- void [SetFrom0to1](#) (float normalizedValue0to1) override
- void [Step](#) (int16\_t numStepsUp, bool useCoarseStepSize) override

### 12.75.1 Member Enumeration Documentation

#### 12.75.1.1 Mapping

```
enum daisy::MappedFloatValue::Mapping [strong]
```

The available mapping functions

## Enumerator

|      |                                                                                                                         |
|------|-------------------------------------------------------------------------------------------------------------------------|
| lin  | The value is mapped linearly between min and max.                                                                       |
| log  | The value is mapped logarithmically. Note that the valid values must be strictly larger than zero, so: min > 0, max > 0 |
| pow2 | The value is mapped with a square law                                                                                   |

**12.75.2 Constructor & Destructor Documentation****12.75.2.1 MappedFloatValue()**

```
daisy::MappedFloatValue::MappedFloatValue (
 float min,
 float max,
 float defaultValue,
 Mapping mapping = Mapping::lin,
 const char * unitStr = "",
 uint8_t numDecimals = 1,
 bool forceSign = false)
```

Creates a [MappedFloatValue](#).

**Parameters**

|                     |                                                                                                       |
|---------------------|-------------------------------------------------------------------------------------------------------|
| <i>min</i>          | The lower end of the range of possible values                                                         |
| <i>max</i>          | The upper end of the range of possible values                                                         |
| <i>defaultValue</i> | The default value                                                                                     |
| <i>mapping</i>      | The Mapping to use. Note that for <a href="#">Mapping::log</a> min, max and `defaultValue must be > 0 |
| <i>unitStr</i>      | A string for the unit, e.g. "Hz"                                                                      |
| <i>numDecimals</i>  | Controls the number of decimals in <a href="#">AppendToString()</a>                                   |
| <i>forceSign</i>    | Controls whether <a href="#">AppendToString()</a> always prints the sign, even for positive numbers   |

**12.75.3 Member Function Documentation****12.75.3.1 AppentToString()**

```
void daisy::MappedFloatValue::AppentToString (
 FixedCapStrBase< char > & string) const [override], [virtual]
```

Generates a string representation and adds it to an existing string.

**Parameters**

|               |                      |
|---------------|----------------------|
| <i>string</i> | The string to add to |
|---------------|----------------------|

Implements [daisy::MappedValue](#).

**12.75.3.2 Get()**

```
float daisy::MappedFloatValue::Get () const [inline]
```

Returns the current value.

**12.75.3.3 GetAs0to1()**

```
float daisy::MappedFloatValue::GetAs0to1 () const [override], [virtual]
```

Returns the 0..1 normalized representation of the value, e.g. to display a slider/knob on a [UI](#).

Implements [daisy::MappedValue](#).

**12.75.3.4 GetPtr()**

```
const float* daisy::MappedFloatValue::GetPtr () const [inline]
```

Returns a const pointer to the current value.

**12.75.3.5 operator float()**

```
daisy::MappedFloatValue::operator float () const [inline]
```

Returns the current value.

**12.75.3.6 operator=()**

```
MappedFloatValue& daisy::MappedFloatValue::operator= (
 float val) [inline]
```

Sets the value, clamping it to the valid range.

### 12.75.3.7 ResetToDefault()

```
void daisy::MappedFloatValue::ResetToDefault () [override], [virtual]
```

Resets the value to its default.

Implements [daisy::MappedValue](#).

### 12.75.3.8 Set()

```
void daisy::MappedFloatValue::Set (
 float newValue)
```

Sets the value, clamping it to the valid range.

### 12.75.3.9 SetFrom0to1()

```
void daisy::MappedFloatValue::SetFrom0to1 (
 float normalizedValue0to1) [override], [virtual]
```

Sets the value so that [GetAs0to1 \(\)](#) returns normalizedValue0to1.

Implements [daisy::MappedValue](#).

### 12.75.3.10 Step()

```
void daisy::MappedFloatValue::Step (
 int16_t numStepsUp,
 bool useCoarseStepSize) [override], [virtual]
```

Steps the 0..1 normalized representation of the value up or down in 1% or 5% steps.

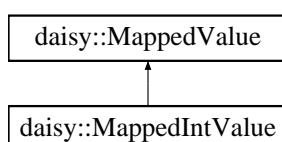
Implements [daisy::MappedValue](#).

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

- src/util/MappedValue.h

## 12.76 daisy::MappedIntValue Class Reference

Inheritance diagram for daisy::MappedIntValue:



## Public Member Functions

- [MappedIntValue](#) (int min, int max, int defaultValue, int stepSizeFine, int stepSizeCoarse, const char \*unitStr="", bool forceSign=false)
- int [Get \(\)](#) const
- const int \* [GetPtr \(\)](#) const
- void [Set](#) (int newValue)
- [operator int \(\)](#) const
- [MappedIntValue & operator=](#) (int val)
- void [AppendToString](#) ([FixedCapStrBase< char >](#) &string) const override
- void [ResetToDefault \(\)](#) override
- float [GetAs0to1 \(\)](#) const override
- void [SetFrom0to1](#) (float normalizedValue0to1) override
- void [Step](#) (int16\_t numStepsUp, bool useCoarseStepSize) override

### 12.76.1 Constructor & Destructor Documentation

#### 12.76.1.1 [MappedIntValue\(\)](#)

```
daisy::MappedIntValue::MappedIntValue (
 int min,
 int max,
 int defaultValue,
 int stepSizeFine,
 int stepSizeCoarse,
 const char * unitStr = "",
 bool forceSign = false)
```

Creates a [MappedIntValue](#).

#### Parameters

|                       |                                                                                                      |
|-----------------------|------------------------------------------------------------------------------------------------------|
| <i>min</i>            | The lower end of the range of possible values                                                        |
| <i>max</i>            | The upper end of the range of possible values                                                        |
| <i>defaultValue</i>   | The default value                                                                                    |
| <i>stepSizeFine</i>   | A fine step size to use in the <a href="#">Step ()</a> function                                      |
| <i>stepSizeCoarse</i> | A coarse step size to use in the <a href="#">Step ()</a> function                                    |
| <i>unitStr</i>        | A string for the unit, e.g. "Hz"                                                                     |
| <i>forceSign</i>      | Controls whether <a href="#">AppendToString ()</a> always prints the sign, even for positive numbers |

### 12.76.2 Member Function Documentation

#### 12.76.2.1 [AppendToString\(\)](#)

```
void daisy::MappedIntValue::AppendToString (
 FixedCapStrBase< char > & string) const [override], [virtual]
```

Generates a string representation and adds it to an existing string.

#### Parameters

|               |                      |
|---------------|----------------------|
| <i>string</i> | The string to add to |
|---------------|----------------------|

Implements [daisy::MappedValue](#).

#### 12.76.2.2 Get()

```
int daisy::MappedIntValue::Get () const [inline]
```

Returns the current value.

#### 12.76.2.3 GetAs0to1()

```
float daisy::MappedIntValue::GetAs0to1 () const [override], [virtual]
```

Returns the 0..1 normalized representation of the value, e.g. to display a slider/knob on a [UI](#).

Implements [daisy::MappedValue](#).

#### 12.76.2.4 GetPtr()

```
const int* daisy::MappedIntValue::GetPtr () const [inline]
```

Returns a const pointer to the current value.

#### 12.76.2.5 operator int()

```
daisy::MappedIntValue::operator int () const [inline]
```

Returns the current value.

#### 12.76.2.6 operator=( )

```
MappedIntValue& daisy::MappedIntValue::operator= (
 int val) [inline]
```

Sets the value, clamping it to the valid range.

### 12.76.2.7 ResetToDefault()

```
void daisy::MappedIntValue::ResetToDefault () [override], [virtual]
```

Resets the value to its default.

Implements [daisy::MappedValue](#).

### 12.76.2.8 Set()

```
void daisy::MappedIntValue::Set (
 int newValue)
```

Sets the value, clamping it to the valid range.

### 12.76.2.9 SetFrom0to1()

```
void daisy::MappedIntValue::SetFrom0to1 (
 float normalizedValue0to1) [override], [virtual]
```

Sets the value so that [GetAs0to1 \(\)](#) returns normalizedValue0to1.

Implements [daisy::MappedValue](#).

### 12.76.2.10 Step()

```
void daisy::MappedIntValue::Step (
 int16_t numStepsUp,
 bool useCoarseStepSize) [override], [virtual]
```

Steps the value up or down using the step sizes specified in the constructor.

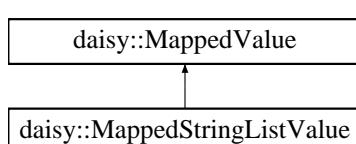
Implements [daisy::MappedValue](#).

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

- src/util/MappedValue.h

## 12.77 daisy::MappedStringValue Class Reference

Inheritance diagram for daisy::MappedStringValue:



## Public Member Functions

- `MappedStringListValue` (const char \*\**itemStrings*, uint16\_t *numItems*, uint32\_t *defaultIndex*)
- int `GetIndex` () const
- const char \* `GetString` () const
- const uint32\_t \* `GetIndexPtr` () const
- void `SetIndex` (uint32\_t *index*)
- `operator int` () const
- `operator const char *` () const
- `MappedStringListValue & operator=` (int *index*)
- void `AppentToString` (`FixedCapStrBase< char >` &*string*) const override
- void `ResetToDefault` () override
- float `GetAs0to1` () const override
- void `SetFrom0to1` (float *normalizedValue0to1*) override
- void `Step` (int16\_t *numStepsUp*, bool *useCoarseStepSize*) override

### 12.77.1 Constructor & Destructor Documentation

#### 12.77.1.1 `MappedStringListValue()`

```
daisy::MappedStringListValue::MappedStringListValue (
 const char ** itemStrings,
 uint16_t numItems,
 uint32_t defaultIndex)
```

Creates a `MappedStringListValue`.

#### Parameters

|                     |                                                  |
|---------------------|--------------------------------------------------|
| <i>itemStrings</i>  | An Array of strings, one for each possible item. |
| <i>numItems</i>     | The number of possible items                     |
| <i>defaultIndex</i> | The default index                                |

### 12.77.2 Member Function Documentation

#### 12.77.2.1 `AppentToString()`

```
void daisy::MappedStringListValue::AppentToString (
 FixedCapStrBase< char > & string) const [override], [virtual]
```

Generates a string representation and adds it to an existing string.

**Parameters**

|               |                      |
|---------------|----------------------|
| <i>string</i> | The string to add to |
|---------------|----------------------|

Implements [daisy::MappedValue](#).

**12.77.2.2 GetAs0to1()**

```
float daisy::MappedStringValue::GetAs0to1 () const [override], [virtual]
```

Returns the 0..1 normalized representation of the value, e.g. to display a slider/knob on a [UI](#).

Implements [daisy::MappedValue](#).

**12.77.2.3 GetIndex()**

```
int daisy::MappedStringValue::GetIndex () const [inline]
```

Returns the current item index.

**12.77.2.4 GetIndexPtr()**

```
const uint32_t* daisy::MappedStringValue::GetIndexPtr () const [inline]
```

Returns a pointer to the current item index.

**12.77.2.5 GetString()**

```
const char* daisy::MappedStringValue::GetString () const [inline]
```

Returns the current item string.

**12.77.2.6 operator const char \*()**

```
daisy::MappedStringValue::operator const char * () const [inline]
```

Returns the current item string.

**12.77.2.7 operator int()**

```
daisy::MappedStringValue::operator int () const [inline]
```

Returns the current item index.

### 12.77.2.8 operator=(*int index*)

```
MappedStringListValue& daisy::MappedStringListValue::operator= (
 int index) [inline]
```

Sets the current item index, clamping it to a valid item index.

### 12.77.2.9 ResetToDefault()

```
void daisy::MappedStringListValue::ResetToDefault () [override], [virtual]
```

Resets the value to its default.

Implements [daisy::MappedValue](#).

### 12.77.2.10 SetFrom0to1()

```
void daisy::MappedStringListValue::SetFrom0to1 (
 float normalizedValue0to1) [override], [virtual]
```

Sets the value so that [GetAs0to1\(\)](#) returns normalizedValue0to1.

Implements [daisy::MappedValue](#).

### 12.77.2.11 SetIndex()

```
void daisy::MappedStringListValue::SetIndex (
 uint32_t index)
```

Sets the current item index, clamping it to a valid item index.

### 12.77.2.12 Step()

```
void daisy::MappedStringListValue::Step (
 int16_t numStepsUp,
 bool useCoarseStepSize) [override], [virtual]
```

Steps through the items up or down. If the coarse step size is used, the value will jump to the first or last item.

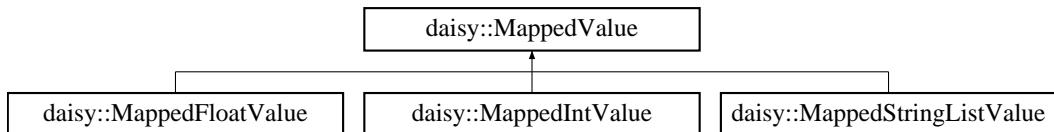
Implements [daisy::MappedValue](#).

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

- src/util/MappedValue.h

## 12.78 daisy::MappedValue Class Reference

Inheritance diagram for daisy::MappedValue:



### Public Member Functions

- virtual void [AppentToString \(FixedCapStrBase< char > &string\) const =0](#)
- virtual void [ResetToDefault \(\)=0](#)
- virtual float [GetAs0to1 \(\) const =0](#)
- virtual void [SetFrom0to1 \(float normalizedValue0to1\)=0](#)
- virtual void [Step \(int16\\_t numStepsUp, bool useCoarseStepSize\)=0](#)

#### 12.78.1 Member Function Documentation

##### 12.78.1.1 AppentToString()

```
virtual void daisy::MappedValue::AppentToString (
 FixedCapStrBase< char > & string) const [pure virtual]
```

Generates a string representation and adds it to an existing string.

###### Parameters

|               |                      |
|---------------|----------------------|
| <i>string</i> | The string to add to |
|---------------|----------------------|

Implemented in [daisy::MappedStringListValue](#), [daisy::MappedIntValue](#), and [daisy::MappedFloatValue](#).

##### 12.78.1.2 GetAs0to1()

```
virtual float daisy::MappedValue::GetAs0to1 () const [pure virtual]
```

Returns the 0..1 normalized representation of the value, e.g. to display a slider/knob on a [UI](#).

Implemented in [daisy::MappedStringListValue](#), [daisy::MappedIntValue](#), and [daisy::MappedFloatValue](#).

### 12.78.1.3 ResetToDefault()

```
virtual void daisy::MappedValue::ResetToDefault () [pure virtual]
```

Resets the value to its default.

Implemented in [daisy::MappedStringValue](#), [daisy::MappedIntValue](#), and [daisy::MappedFloatValue](#).

### 12.78.1.4 SetFrom0to1()

```
virtual void daisy::MappedValue::SetFrom0to1 (
 float normalizedValue0to1) [pure virtual]
```

Sets the value so that [GetAs0to1 \(\)](#) returns normalizedValue0to1.

Implemented in [daisy::MappedStringValue](#), [daisy::MappedIntValue](#), and [daisy::MappedFloatValue](#).

### 12.78.1.5 Step()

```
virtual void daisy::MappedValue::Step (
 int16_t numStepsUp,
 bool useCoarseStepSize) [pure virtual]
```

Steps the value up by whatever is appropriate. This function can be used to increment/decrement the value with buttons/encoders while making use of the specific mapping.

Implemented in [daisy::MappedStringValue](#), [daisy::MappedIntValue](#), and [daisy::MappedFloatValue](#).

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

- src/util/MappedValue.h

## 12.79 daisy::MAX11300Driver< Transport > Class Template Reference

Device Driver for the MAX11300 20 port ADC/DAC/GPIO device.

```
#include <max11300.h>
```

### Classes

- struct [Config](#)

## Public Types

- enum `Pin` {
 `PIN_0` , `PIN_1` , `PIN_2` , `PIN_3` ,
 `PIN_4` , `PIN_5` , `PIN_6` , `PIN_7` ,
 `PIN_8` , `PIN_9` , `PIN_10` , `PIN_11` ,
 `PIN_12` , `PIN_13` , `PIN_14` , `PIN_15` ,
 `PIN_16` , `PIN_17` , `PIN_18` , `PIN_19` }
- enum class `VoltageRange` { `ZERO_TO_10` = `0x0100` , `NEGATIVE_5_TO_5` = `0x0200` , `NEGATIVE_10_TO_0` = `0x0300` }
- enum class `Result` { `OK` , `ERR` }

## Public Member Functions

- `Result Init (Config config)`
- `Result ConfigurePinAsDigitalRead (Pin pin, float threshold_voltage)`
- `Result ConfigurePinAsDigitalWrite (Pin pin, float output_voltage)`
- `Result ConfigurePinAsAnalogRead (Pin pin, VoltageRange range)`
- `Result ConfigurePinAsAnalogWrite (Pin pin, VoltageRange range)`
- `Result DisablePin (Pin pin)`
- `uint16_t ReadAnalogPinRaw (Pin pin)`
- `float ReadAnalogPinVolts (Pin pin)`
- `void WriteAnalogPinRaw (Pin pin, uint16_t raw_value)`
- `void WriteAnalogPinVolts (Pin pin, float voltage)`
- `bool ReadDigitalPin (Pin pin)`
- `void WriteDigitalPin (Pin pin, bool value)`
- `Result Update ()`

## Static Public Member Functions

- static `uint16_t VoltsTo12BitUint (float volts, VoltageRange range)`
- static `float TwelveBitUintToVolts (uint16_t value, VoltageRange range)`

### 12.79.1 Detailed Description

```
template<typename Transport>
class daisy::MAX11300Driver < Transport >
```

Device Driver for the MAX11300 20 port ADC/DAC/GPIO device.

**Author**

sam.braam

**Date**

Oct. 2021

This is a highly opinionated driver implementation for the MAX11300 DAC/ADC/GPIO device.

This implementation has been designed for use in the context of Eurorack modular systems. There are a number of features the MAX11300 offers which are not exposed, as well as a number of configuration decisions that were made in order to simplify usage and improve ergonomics, even at the cost of flexibility.

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

- `src/dev/max11300.h`

## 12.80 daisy::MidiEvent Struct Reference

```
#include <MidiEvent.h>
```

### Public Member Functions

- [NoteOffEvent AsNoteOff \(\)](#)
- [NoteOnEvent AsNoteOn \(\)](#)
- [PolyphonicKeyPressureEvent AsPolyphonicKeyPressure \(\)](#)
- [ControlChangeEvent AsControlChange \(\)](#)
- [ProgramChangeEvent AsProgramChange \(\)](#)
- [ChannelPressureEvent AsChannelPressure \(\)](#)
- [PitchBendEvent AsPitchBend \(\)](#)
- [SystemExclusiveEvent AsSystemExclusive \(\)](#)
- [MTCQuarterFrameEvent AsMTCQuarterFrame \(\)](#)
- [SongPositionPointerEvent AsSongPositionPointer \(\)](#)
- [SongSelectEvent AsSongSelect \(\)](#)
- [AllSoundOffEvent AsAllSoundOff \(\)](#)
- [ResetAllControllersEvent AsResetAllControllers \(\)](#)
- [LocalControlEvent AsLocalControl \(\)](#)
- [AllNotesOffEvent AsAllNotesOff \(\)](#)
- [OmniModeOffEvent AsOmniModeOff \(\)](#)
- [OmniModeOnEvent AsOmniModeOn \(\)](#)
- [MonoModeOnEvent AsMonoModeOn \(\)](#)
- [PolyModeOnEvent AsPolyModeOn \(\)](#)

### Public Attributes

- [MidiMessageType type](#)
- [int channel](#)
- [uint8\\_t data \[2\]](#)
- [uint8\\_t sysex\\_data \[SYSEX\\_BUFFER\\_LEN\]](#)
- [uint8\\_t sysex\\_message\\_len](#)
- [SystemCommonType sc\\_type](#)
- [SystemRealTimeType srt\\_type](#)
- [ChannelModeType cm\\_type](#)

#### 12.80.1 Detailed Description

Simple [MidiEvent](#) with message type, channel, and data[2] members.

#### 12.80.2 Member Function Documentation

### 12.80.2.1 AsChannelPressure()

```
ChannelPressureEvent daisy::MidiEvent::AsChannelPressure () [inline]
```

Returns the data within the [MidiEvent](#) as a [ProgramChangeEvent](#) struct.

### 12.80.2.2 AsControlChange()

```
ControlChangeEvent daisy::MidiEvent::AsControlChange () [inline]
```

Returns the data within the [MidiEvent](#) as a [ControlChangeEvent](#) struct.

### 12.80.2.3 AsNoteOff()

```
NoteOffEvent daisy::MidiEvent::AsNoteOff () [inline]
```

Returns the data within the [MidiEvent](#) as a [NoteOffEvent](#) struct

### 12.80.2.4 AsNoteOn()

```
NoteOnEvent daisy::MidiEvent::AsNoteOn () [inline]
```

Returns the data within the [MidiEvent](#) as a [NoteOnEvent](#) struct

### 12.80.2.5 AsPitchBend()

```
PitchBendEvent daisy::MidiEvent::AsPitchBend () [inline]
```

Returns the data within the [MidiEvent](#) as a [PitchBendEvent](#) struct.

### 12.80.2.6 AsPolyphonicKeyPressure()

```
PolyphonicKeyPressureEvent daisy::MidiEvent::AsPolyphonicKeyPressure () [inline]
```

Returns the data within the [MidiEvent](#) as a [PolyphonicKeyPressureEvent](#) struct

### 12.80.2.7 AsProgramChange()

```
ProgramChangeEvent daisy::MidiEvent::AsProgramChange () [inline]
```

Returns the data within the [MidiEvent](#) as a [ProgramChangeEvent](#) struct.

## 12.80.3 Member Data Documentation

### 12.80.3.1 channel

```
int daisy::MidiEvent::channel
&
```

### 12.80.3.2 data

```
uint8_t daisy::MidiEvent::data[2]
&
```

### 12.80.3.3 sysex\_data

```
uint8_t daisy::MidiEvent::sysex_data[SYSEX_BUFFER_LEN]
&
```

### 12.80.3.4 type

```
MidiMessageType daisy::MidiEvent::type
&
```

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

- src/hid/MidiEvent.h

## 12.81 daisy::MidiHandler< Transport > Class Template Reference

Simple MIDI Handler

Parses bytes from an input into valid MidiEvents.

The MidiEvents fill a [FIFO](#) queue that the user can pop messages from.

```
#include <midi.h>
```

### Classes

- struct [Config](#)

### Public Member Functions

- void [Init \(Config config\)](#)
- void [StartReceive \(\)](#)
- void [Listen \(\)](#)
- bool [HasEvents \(\) const](#)
- [MidiEvent PopEvent \(\)](#)
- void [SendMessage \(uint8\\_t \\*bytes, size\\_t size\)](#)
- void [Parse \(uint8\\_t byte\)](#)

### 12.81.1 Detailed Description

```
template<typename Transport>
class daisy::MidiHandler< Transport >
```

Simple MIDI Handler

Parses bytes from an input into valid MidiEvents.

The MidiEvents fill a [FIFO](#) queue that the user can pop messages from.

#### Author

shensley

#### Date

March 2020

### 12.81.2 Member Function Documentation

#### 12.81.2.1 HasEvents()

```
template<typename Transport >
bool daisy::MidiHandler< Transport >::HasEvents () const [inline]
```

Checks if there are unhandled messages in the queue

#### Returns

True if there are events to be handled, else false.

#### 12.81.2.2 Init()

```
template<typename Transport >
void daisy::MidiHandler< Transport >::Init (
 Config config) [inline]
```

Initializes the [MidiHandler](#)

#### Parameters

|                 |             |
|-----------------|-------------|
| <i>in_mode</i>  | Input mode  |
| <i>out_mode</i> | Output mode |

### 12.81.2.3 Listen()

```
template<typename Transport >
void daisy::MidiHandler< Transport >::Listen () [inline]
```

Start listening

### 12.81.2.4 Parse()

```
template<typename Transport >
void daisy::MidiHandler< Transport >::Parse (
 uint8_t byte) [inline]
```

Feed in bytes to state machine from a queue. Populates internal [FIFO](#) queue with MIDI Messages For example with uart: midi.Parse(uart.PopRx());

#### Parameters

|             |   |
|-------------|---|
| <i>byte</i> | & |
|-------------|---|

### 12.81.2.5 PopEvent()

```
template<typename Transport >
MidiEvent daisy::MidiHandler< Transport >::PopEvent () [inline]
```

Pops the oldest unhandled [MidiEvent](#) from the internal queue

#### Returns

The event to be handled

### 12.81.2.6 SendMessage()

```
template<typename Transport >
void daisy::MidiHandler< Transport >::SendMessage (
 uint8_t * bytes,
 size_t size) [inline]
```

SendMessage Send raw bytes as message

### 12.81.2.7 StartReceive()

```
template<typename Transport >
void daisy::MidiHandler< Transport >::StartReceive () [inline]
```

Starts listening on the selected input mode(s). [MidiEvent](#) Queue will begin to fill, and can be checked with [HasEvents\(\)](#)

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

- src/hid/midi.h

## 12.82 daisy::MidiUartTransport Class Reference

### Classes

- struct [Config](#)

### Public Member Functions

- void **Init** ([Config](#) config)
- void **StartRx** ()
- size\_t **Readable** ()
- uint8\_t **Rx** ()
- bool **RxActive** ()
- void **FlushRx** ()
- void **Tx** (uint8\_t \*buff, size\_t size)

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

- src/hid/midi.h

## 12.83 daisy::MidiUsbTransport Class Reference

### Classes

- struct [Config](#)

### Public Member Functions

- void **Init** ([Config](#) config)
- void **StartRx** ()
- size\_t **Readable** ()
- uint8\_t **Rx** ()
- bool **RxActive** ()
- void **FlushRx** ()
- void **Tx** (uint8\_t \*buffer, size\_t size)
- **MidiUsbTransport** (const [MidiUsbTransport](#) &other)=default
- [MidiUsbTransport](#) & **operator=** (const [MidiUsbTransport](#) &other)=default

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

- src/hid/usb\_midi.h

## 12.84 daisy::MonoModeOnEvent Struct Reference

```
#include <MidiEvent.h>
```

## Public Attributes

- int `channel`
- uint8\_t `num_channels`

### 12.84.1 Detailed Description

Struct containing `MonoModeOnEvent` data. Can be made from `MidiEvent`

### 12.84.2 Member Data Documentation

#### 12.84.2.1 `channel`

```
int daisy::MonoModeOnEvent::channel
&
```

#### 12.84.2.2 `num_channels`

```
uint8_t daisy::MonoModeOnEvent::num_channels
&
```

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

- `src/hid/MidiEvent.h`

## 12.85 `daisy::MTCQuarterFrameEvent` Struct Reference

```
#include <MidiEvent.h>
```

## Public Attributes

- uint8\_t `message_type`
- uint8\_t `value`

### 12.85.1 Detailed Description

Struct containing QuarterFrame data. Can be made from `MidiEvent`

## 12.85.2 Member Data Documentation

### 12.85.2.1 message\_type

```
uint8_t daisy::MTCQuarterFrameEvent::message_type
```

&

### 12.85.2.2 value

```
uint8_t daisy::MTCQuarterFrameEvent::value
```

&

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

- src/hid/MidiEvent.h

## 12.86 daisy::NoteOffEvent Struct Reference

```
#include <MidiEvent.h>
```

### Public Attributes

- int [channel](#)
- uint8\_t [note](#)
- uint8\_t [velocity](#)

### 12.86.1 Detailed Description

Struct containing note, and velocity data for a given channel. Can be made from [MidiEvent](#)

## 12.86.2 Member Data Documentation

### 12.86.2.1 channel

```
int daisy::NoteOffEvent::channel
```

&

### 12.86.2.2 note

```
uint8_t daisy::NoteOffEvent::note
&
```

### 12.86.2.3 velocity

```
uint8_t daisy::NoteOffEvent::velocity
&
```

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

- src/hid/MidiEvent.h

## 12.87 daisy::NoteOnEvent Struct Reference

```
#include <MidiEvent.h>
```

### Public Attributes

- int [channel](#)
- uint8\_t [note](#)
- uint8\_t [velocity](#)

### 12.87.1 Detailed Description

Struct containing note, and velocity data for a given channel. Can be made from [MidiEvent](#)

### 12.87.2 Member Data Documentation

#### 12.87.2.1 channel

```
int daisy::NoteOnEvent::channel
&
```

#### 12.87.2.2 note

```
uint8_t daisy::NoteOnEvent::note
&
```

### 12.87.2.3 velocity

```
uint8_t daisy::NoteOnEvent::velocity
```

&

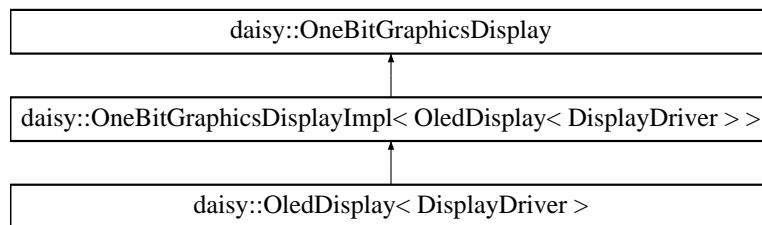
The documentation for this struct was generated from the following file:

- src/hid/MidiEvent.h

## 12.88 daisy::OledDisplay< DisplayDriver > Class Template Reference

```
#include <oled_display.h>
```

Inheritance diagram for daisy::OledDisplay< DisplayDriver >:



## Classes

- struct [Config](#)

## Public Member Functions

- void [Init](#) ([Config](#) config)
- uint16\_t [Height](#) () const override
- uint16\_t [Width](#) () const override
- void [Fill](#) (bool on) override
- void [DrawPixel](#) (uint\_fast8\_t x, uint\_fast8\_t y, bool on) override
- void [Update](#) () override

## Additional Inherited Members

### 12.88.1 Detailed Description

```
template<typename DisplayDriver>
class daisy::OledDisplay< DisplayDriver >
```

This class is for drawing to a monochrome OLED display.

## 12.88.2 Member Function Documentation

### 12.88.2.1 DrawPixel()

```
template<typename DisplayDriver >
void daisy::OledDisplay< DisplayDriver >::DrawPixel (
 uint_fast8_t x,
 uint_fast8_t y,
 bool on) [inline], [override], [virtual]
```

Sets the pixel at the specified coordinate to be on/off.

#### Parameters

|    |              |
|----|--------------|
| x  | x Coordinate |
| y  | y coordinate |
| on | on or off    |

Implements [daisy::OneBitGraphicsDisplay](#).

### 12.88.2.2 Fill()

```
template<typename DisplayDriver >
void daisy::OledDisplay< DisplayDriver >::Fill (
 bool on) [inline], [override], [virtual]
```

Fills the entire display with either on/off.

#### Parameters

|    |                 |
|----|-----------------|
| on | Sets on or off. |
|----|-----------------|

Implements [daisy::OneBitGraphicsDisplay](#).

### 12.88.2.3 Update()

```
template<typename DisplayDriver >
void daisy::OledDisplay< DisplayDriver >::Update () [inline], [override], [virtual]
```

Writes the current display buffer to the OLED device using SPI or I2C depending on how the object was initialized.

Implements [daisy::OneBitGraphicsDisplay](#).

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

- src/hid/disp/oled\_display.h

## 12.89 daisy::OmniModeOffEvent Struct Reference

```
#include <MidiEvent.h>
```

### Public Attributes

- int [channel](#)

#### 12.89.1 Detailed Description

Struct containing [OmniModeOffEvent](#) data. Can be made from [MidiEvent](#)

#### 12.89.2 Member Data Documentation

##### 12.89.2.1 channel

```
int daisy::OmniModeOffEvent::channel
```

&

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

- [src/hid/MidiEvent.h](#)

## 12.90 daisy::OmniModeOnEvent Struct Reference

```
#include <MidiEvent.h>
```

### Public Attributes

- int [channel](#)

#### 12.90.1 Detailed Description

Struct containing [OmniModeOnEvent](#) data. Can be made from [MidiEvent](#)

#### 12.90.2 Member Data Documentation

### 12.90.2.1 channel

```
int daisy::OmniModeOnEvent::channel
```

&

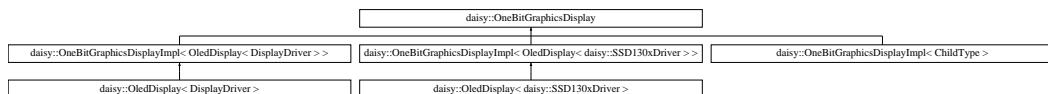
The documentation for this struct was generated from the following file:

- src/hid/MidiEvent.h

## 12.91 daisy::OneBitGraphicsDisplay Class Reference

```
#include <display.h>
```

Inheritance diagram for daisy::OneBitGraphicsDisplay:



### Public Member Functions

- virtual uint16\_t **Height** () const =0
- virtual uint16\_t **Width** () const =0
- Rectangle GetBounds** () const
- size\_t **CurrentX** ()
- size\_t **CurrentY** ()
- virtual void **Fill** (bool on)=0
- virtual void **DrawPixel** (uint\_fast8\_t x, uint\_fast8\_t y, bool on)=0
- virtual void **DrawLine** (uint\_fast8\_t x1, uint\_fast8\_t y1, uint\_fast8\_t x2, uint\_fast8\_t y2, bool on)=0
- virtual void **DrawRect** (uint\_fast8\_t x1, uint\_fast8\_t y1, uint\_fast8\_t x2, uint\_fast8\_t y2, bool on, bool fill=false)=0
- void **DrawRect** (const **Rectangle** &rect, bool on, bool fill=false)
- virtual void **DrawArc** (uint\_fast8\_t x, uint\_fast8\_t y, uint\_fast8\_t radius, int\_fast16\_t start\_angle, int\_fast16\_t sweep, bool on)=0
- void **DrawCircle** (uint\_fast8\_t x, uint\_fast8\_t y, uint\_fast8\_t radius, bool on)
- virtual char **WriteChar** (char ch, **FontDef** font, bool on)=0
- virtual char **WriteString** (const char \*str, **FontDef** font, bool on)=0
- virtual **Rectangle WriteStringAligned** (const char \*str, const **FontDef** &font, **Rectangle** boundingBox, **Alignment** alignment, bool on)=0
- void **SetCursor** (uint16\_t x, uint16\_t y)
- virtual void **Update** ()=0

### Protected Attributes

- uint16\_t **currentX\_**
- uint16\_t **currentY\_**

### 12.91.1 Detailed Description

This interface is used as a base class for all types of 1bit-per-pixel graphics displays.

### 12.91.2 Member Function Documentation

#### 12.91.2.1 DrawArc()

```
virtual void daisy::OneBitGraphicsDisplay::DrawArc (
 uint_fast8_t x,
 uint_fast8_t y,
 uint_fast8_t radius,
 int_fast16_t start_angle,
 int_fast16_t sweep,
 bool on) [pure virtual]
```

Draws an arc around the specified coordinate

##### Parameters

|                    |                                       |
|--------------------|---------------------------------------|
| <i>x</i>           | x Coordinate of the center of the arc |
| <i>y</i>           | y Coordinate of the center of the arc |
| <i>radius</i>      | radius of the arc                     |
| <i>start_angle</i> | angle where to start the arc          |
| <i>sweep</i>       | total angle of the arc                |
| <i>on</i>          | on or off                             |

Implemented in [daisy::OneBitGraphicsDisplayImpl< ChildType >](#), [daisy::OneBitGraphicsDisplayImpl< OledDisplay< DisplayDriver > >](#) and [daisy::OneBitGraphicsDisplayImpl< OledDisplay< daisy::SSD130xDriver > >](#).

#### 12.91.2.2 DrawCircle()

```
void daisy::OneBitGraphicsDisplay::DrawCircle (
 uint_fast8_t x,
 uint_fast8_t y,
 uint_fast8_t radius,
 bool on) [inline]
```

Draws a circle around the specified coordinate

##### Parameters

|               |                                          |
|---------------|------------------------------------------|
| <i>x</i>      | x Coordinate of the center of the circle |
| <i>y</i>      | y Coordinate of the center of the circle |
| <i>radius</i> | radius of the circle                     |
| <i>on</i>     | on or off                                |

### 12.91.2.3 DrawLine()

```
virtual void daisy::OneBitGraphicsDisplay::DrawLine (
 uint_fast8_t x1,
 uint_fast8_t y1,
 uint_fast8_t x2,
 uint_fast8_t y2,
 bool on) [pure virtual]
```

Draws a line from (x1, y1) to (y1, y2)

#### Parameters

|           |                                    |
|-----------|------------------------------------|
| <i>x1</i> | x Coordinate of the starting point |
| <i>y1</i> | y Coordinate of the starting point |
| <i>x2</i> | x Coordinate of the ending point   |
| <i>y2</i> | y Coordinate of the ending point   |
| <i>on</i> | on or off                          |

Implemented in [daisy::OneBitGraphicsDisplayImpl< ChildType >](#), [daisy::OneBitGraphicsDisplayImpl< OledDisplay< DisplayDriver > >](#) and [daisy::OneBitGraphicsDisplayImpl< OledDisplay< daisy::SSD130xDriver > >](#).

### 12.91.2.4 DrawPixel()

```
virtual void daisy::OneBitGraphicsDisplay::DrawPixel (
 uint_fast8_t x,
 uint_fast8_t y,
 bool on) [pure virtual]
```

Sets the pixel at the specified coordinate to be on/off.

#### Parameters

|           |              |
|-----------|--------------|
| <i>x</i>  | x Coordinate |
| <i>y</i>  | y coordinate |
| <i>on</i> | on or off    |

Implemented in [daisy::OledDisplay< DisplayDriver >](#), and [daisy::OledDisplay< daisy::SSD130xDriver >](#).

### 12.91.2.5 DrawRect() [1/2]

```
void daisy::OneBitGraphicsDisplay::DrawRect (
 const Rectangle & rect,
```

```
 bool on,
 bool fill = false) [inline]
```

Draws a rectangle.

#### Parameters

|             |                                             |
|-------------|---------------------------------------------|
| <i>rect</i> | the rectangle                               |
| <i>on</i>   | on or off                                   |
| <i>fill</i> | fill the rectangle or draw only the outline |

#### 12.91.2.6 DrawRect() [2/2]

```
virtual void daisy::OneBitGraphicsDisplay::DrawRect (
 uint_fast8_t x1,
 uint_fast8_t y1,
 uint_fast8_t x2,
 uint_fast8_t y2,
 bool on,
 bool fill = false) [pure virtual]
```

Draws a rectangle based on two coordinates.

#### Parameters

|             |                                             |
|-------------|---------------------------------------------|
| <i>x1</i>   | x Coordinate of the first point             |
| <i>y1</i>   | y Coordinate of the first point             |
| <i>x2</i>   | x Coordinate of the second point            |
| <i>y2</i>   | y Coordinate of the second point            |
| <i>on</i>   | on or off                                   |
| <i>fill</i> | fill the rectangle or draw only the outline |

Implemented in [daisy::OneBitGraphicsDisplayImpl< ChildType >](#), [daisy::OneBitGraphicsDisplayImpl< OledDisplay< DisplayDriver > >](#) and [daisy::OneBitGraphicsDisplayImpl< OledDisplay< daisy::SSD130xDriver > >](#).

#### 12.91.2.7 Fill()

```
virtual void daisy::OneBitGraphicsDisplay::Fill (
 bool on) [pure virtual]
```

Fills the entire display with either on/off.

#### Parameters

|           |                 |
|-----------|-----------------|
| <i>on</i> | Sets on or off. |
|-----------|-----------------|

Implemented in [daisy::OledDisplay< DisplayDriver >](#), and [daisy::OledDisplay< daisy::SSD130xDriver >](#).

### 12.91.2.8 SetCursor()

```
void daisy::OneBitGraphicsDisplay::SetCursor (
 uint16_t x,
 uint16_t y) [inline]
```

Moves the 'Cursor' position used for WriteChar, and WriteStr to the specified coordinate.

#### Parameters

|   |       |
|---|-------|
| x | x pos |
| y | y pos |

### 12.91.2.9 Update()

```
virtual void daisy::OneBitGraphicsDisplay::Update () [pure virtual]
```

Writes the current display buffer to the OLED device using SPI or I2C depending on how the object was initialized.

Implemented in [daisy::OledDisplay< DisplayDriver >](#), and [daisy::OledDisplay< daisy::SSD130xDriver >](#).

### 12.91.2.10 WriteChar()

```
virtual char daisy::OneBitGraphicsDisplay::WriteChar (
 char ch,
 FontDef font,
 bool on) [pure virtual]
```

Writes the character with the specific [FontDef](#) to the display buffer at the current Cursor position.

#### Parameters

|      |                         |
|------|-------------------------|
| ch   | character to be written |
| font | font to be written in   |
| on   | on or off               |

#### Returns

&

Implemented in [daisy::OneBitGraphicsDisplayImpl< ChildType >](#), [daisy::OneBitGraphicsDisplayImpl< OledDisplay< DisplayDriver > >](#) and [daisy::OneBitGraphicsDisplayImpl< OledDisplay< daisy::SSD130xDriver > >](#).

### 12.91.2.11 WriteString()

```
virtual char daisy::OneBitGraphicsDisplay::WriteString (
 const char * str,
 FontDef font,
 bool on) [pure virtual]
```

Similar to WriteChar, except it will handle an entire String. Wrapping does not happen automatically, so the width of the string must be kept within the dimensions of the screen.

#### Parameters

|             |                      |
|-------------|----------------------|
| <i>str</i>  | string to be written |
| <i>font</i> | font to use          |
| <i>on</i>   | on or off            |

#### Returns

&

Implemented in [daisy::OneBitGraphicsDisplayImpl< ChildType >](#), [daisy::OneBitGraphicsDisplayImpl< OledDisplay< DisplayDriver > >](#) and [daisy::OneBitGraphicsDisplayImpl< OledDisplay< daisy::SSD130xDriver > >](#).

### 12.91.2.12 WriteStringAligned()

```
virtual Rectangle daisy::OneBitGraphicsDisplay::WriteStringAligned (
 const char * str,
 const FontDef & font,
 Rectangle boundingBox,
 Alignment alignment,
 bool on) [pure virtual]
```

Similar to WriteString but justified within a bounding box.

#### Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <i>str</i>         | string to be written                 |
| <i>font</i>        | font to use                          |
| <i>boundingBox</i> | the bounding box to draw the text in |
| <i>alignment</i>   | the alignment to use                 |
| <i>on</i>          | on or off                            |

#### Returns

The rectangle that was drawn to

Implemented in `daisy::OneBitGraphicsDisplayImpl< ChildType >`, `daisy::OneBitGraphicsDisplayImpl< OledDisplay< DisplayDriver > >` and `daisy::OneBitGraphicsDisplayImpl< OledDisplay< daisy::SSD130xDriver > >`.

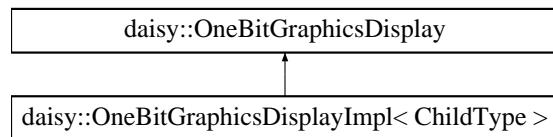
The documentation for this class was generated from the following file:

- `src/hid/disp/display.h`

## 12.92 `daisy::OneBitGraphicsDisplayImpl< ChildType >` Class Template Reference

```
#include <display.h>
```

Inheritance diagram for `daisy::OneBitGraphicsDisplayImpl< ChildType >`:



### Public Member Functions

- void `DrawLine` (`uint_fast8_t x1, uint_fast8_t y1, uint_fast8_t x2, uint_fast8_t y2, bool on`) override
- void `DrawRect` (`uint_fast8_t x1, uint_fast8_t y1, uint_fast8_t x2, uint_fast8_t y2, bool on, bool fill=false`) override
- void `DrawArc` (`uint_fast8_t x, uint_fast8_t y, uint_fast8_t radius, int_fast16_t start_angle, int_fast16_t sweep, bool on`) override
- char `WriteChar` (`char ch, FontDef font, bool on`) override
- char `WriteString` (`const char *str, FontDef font, bool on`) override
- `Rectangle WriteStringAligned` (`const char *str, const FontDef &font, Rectangle boundingBox, Alignment alignment, bool on`) override

### Additional Inherited Members

#### 12.92.1 Detailed Description

```
template<class ChildType>
class daisy::OneBitGraphicsDisplayImpl< ChildType >
```

This class is intended as a intermediary class for your actual implementation of the `OneBitGraphicsDisplay` interface. It uses the CRTP design pattern where the template argument is the child class. It provides implementations for most of the functions, except `DrawPixel()`, `Update()` and `Fill()`, which you'll have to provide in your child class. The main goal of this class is to provide common drawing functions without relying on massive amounts of virtual function calls that would result in a performance loss. To achieve this, any drawing function that is implemented here and internally calls other drawing functions (e.g. `DrawRect()` which internally calls `DrawPixel()` and `DrawLine()`) makes these calls via the qualified name of these functions to explicitly suppress the virtual dispatch mechanism like this:

```
ChildType::DrawPixel(...); // no virtual function call; direct call into the child class function
```

To create a custom `OneBitGraphicsDisplay` implementation, you can A) inherit from `OneBitGraphicsDisplay` directly and provide all the drawing functions yourself B) Inherit from `OneBitGraphicsDisplayImpl` and only provide `DrawPixel()`, `Fill()` and `Update()` like this:

```
class MyDisplayClass : public OneBitGraphicsDisplayImpl<MyDisplayClass> { public: void Fill() override { ... }; void DrawPixel(uint_fast8_t x, uint_fast8_t y, bool on) override { ... }; void Update() override { ... } };
```

## 12.92.2 Member Function Documentation

### 12.92.2.1 DrawArc()

```
template<class ChildType >
void daisy::OneBitGraphicsDisplayImpl< ChildType >::DrawArc (
 uint_fast8_t x,
 uint_fast8_t y,
 uint_fast8_t radius,
 int_fast16_t start_angle,
 int_fast16_t sweep,
 bool on) [inline], [override], [virtual]
```

Draws an arc around the specified coordinate

#### Parameters

|                    |                                       |
|--------------------|---------------------------------------|
| <i>x</i>           | x Coordinate of the center of the arc |
| <i>y</i>           | y Coordinate of the center of the arc |
| <i>radius</i>      | radius of the arc                     |
| <i>start_angle</i> | angle where to start the arc          |
| <i>sweep</i>       | total angle of the arc                |
| <i>on</i>          | on or off                             |

Implements [daisy::OneBitGraphicsDisplay](#).

### 12.92.2.2 DrawLine()

```
template<class ChildType >
void daisy::OneBitGraphicsDisplayImpl< ChildType >::DrawLine (
 uint_fast8_t x1,
 uint_fast8_t y1,
 uint_fast8_t x2,
 uint_fast8_t y2,
 bool on) [inline], [override], [virtual]
```

Draws a line from (x1, y1) to (y1, y2)

#### Parameters

|           |                                    |
|-----------|------------------------------------|
| <i>x1</i> | x Coordinate of the starting point |
| <i>y1</i> | y Coordinate of the starting point |
| <i>x2</i> | x Coordinate of the ending point   |
| <i>y2</i> | y Coordinate of the ending point   |
| <i>on</i> | on or off                          |

Implements [daisy::OneBitGraphicsDisplay](#).

### 12.92.2.3 DrawRect()

```
template<class ChildType >
void daisy::OneBitGraphicsDisplayImpl< ChildType >::DrawRect (
 uint_fast8_t x1,
 uint_fast8_t y1,
 uint_fast8_t x2,
 uint_fast8_t y2,
 bool on,
 bool fill = false) [inline], [override], [virtual]
```

Draws a rectangle based on two coordinates.

#### Parameters

|             |                                             |
|-------------|---------------------------------------------|
| <i>x1</i>   | x Coordinate of the first point             |
| <i>y1</i>   | y Coordinate of the first point             |
| <i>x2</i>   | x Coordinate of the second point            |
| <i>y2</i>   | y Coordinate of the second point            |
| <i>on</i>   | on or off                                   |
| <i>fill</i> | fill the rectangle or draw only the outline |

Implements [daisy::OneBitGraphicsDisplay](#).

### 12.92.2.4 WriteChar()

```
template<class ChildType >
char daisy::OneBitGraphicsDisplayImpl< ChildType >::WriteChar (
 char ch,
 FontDef font,
 bool on) [inline], [override], [virtual]
```

Writes the character with the specific [FontDef](#) to the display buffer at the current Cursor position.

#### Parameters

|             |                         |
|-------------|-------------------------|
| <i>ch</i>   | character to be written |
| <i>font</i> | font to be written in   |
| <i>on</i>   | on or off               |

#### Returns

&

Implements [daisy::OneBitGraphicsDisplay](#).

### 12.92.2.5 WriteString()

```
template<class ChildType >
char daisy::OneBitGraphicsDisplayImpl< ChildType >::WriteString (
 const char * str,
 FontDef font,
 bool on) [inline], [override], [virtual]
```

Similar to WriteChar, except it will handle an entire String. Wrapping does not happen automatically, so the width of the string must be kept within the dimensions of the screen.

#### Parameters

|             |                      |
|-------------|----------------------|
| <i>str</i>  | string to be written |
| <i>font</i> | font to use          |
| <i>on</i>   | on or off            |

#### Returns

&

Implements [daisy::OneBitGraphicsDisplay](#).

### 12.92.2.6 WriteStringAligned()

```
template<class ChildType >
Rectangle daisy::OneBitGraphicsDisplayImpl< ChildType >::WriteStringAligned (
 const char * str,
 const FontDef & font,
 Rectangle boundingBox,
 Alignment alignment,
 bool on) [inline], [override], [virtual]
```

Similar to WriteString but justified within a bounding box.

#### Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <i>str</i>         | string to be written                 |
| <i>font</i>        | font to use                          |
| <i>boundingBox</i> | the bounding box to draw the text in |
| <i>alignment</i>   | the alignment to use                 |
| <i>on</i>          | on or off                            |

#### Returns

The rectangle that was drawn to

Implements [daisy::OneBitGraphicsDisplay](#).

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

- [src/hid/disp/display.h](#)

## 12.93 daisy::Parameter Class Reference

```
#include <parameter.h>
```

### Public Types

- enum [Curve](#) {  
  [LINEAR](#) , [EXPONENTIAL](#) , [LOGARITHMIC](#) , [CUBE](#) ,  
  [LAST](#) }

### Public Member Functions

- [Parameter \(\)](#)
- [~Parameter \(\)](#)
- void [Init \(AnalogControl input, float min, float max, Curve curve\)](#)
- float [Process \(\)](#)
- float [Value \(\)](#)

#### 12.93.1 Detailed Description

Simple parameter mapping tool that takes a 0-1 input from an hid\_ctrl.

#### 12.93.2 Member Enumeration Documentation

##### 12.93.2.1 Curve

```
enum daisy::Parameter::Curve
```

Curves are applied to the output signal

Enumerator

|                             |                     |
|-----------------------------|---------------------|
| <a href="#">LINEAR</a>      | Linear curve        |
| <a href="#">EXPONENTIAL</a> | Exponential curve   |
| <a href="#">LOGARITHMIC</a> | Logarithmic curve   |
| <a href="#">CUBE</a>        | Cubic curve         |
| <a href="#">LAST</a>        | Final enum element. |

### 12.93.3 Constructor & Destructor Documentation

#### 12.93.3.1 Parameter()

```
daisy::Parameter::Parameter () [inline]
```

Constructor

#### 12.93.3.2 ~Parameter()

```
daisy::Parameter::~Parameter () [inline]
```

Destructor

### 12.93.4 Member Function Documentation

#### 12.93.4.1 Init()

```
void daisy::Parameter::Init (
 AnalogControl input,
 float min,
 float max,
 Curve curve)
```

initialize a parameter using an hid\_ctrl object.

##### Parameters

|              |                                                                   |
|--------------|-------------------------------------------------------------------|
| <i>input</i> | - object containing the direct link to a hardware control source. |
| <i>min</i>   | - bottom of range. (when input is 0.0)                            |
| <i>max</i>   | - top of range (when input is 1.0)                                |
| <i>curve</i> | - the scaling curve for the input->output transformation.         |

#### 12.93.4.2 Process()

```
float daisy::Parameter::Process ()
```

processes the input signal, this should be called at the samplerate of the hid\_ctrl passed in.

**Returns**

a float with the specified transformation applied.

#### 12.93.4.3 Value()

```
float daisy::Parameter::Value () [inline]
```

**Returns**

the current value from the parameter without processing another sample. this is useful if you need to use the value multiple times, and don't store

the output of process in a local variable.

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

- src/hid/parameter.h

## 12.94 daisy::Pcm3060 Class Reference

### Public Types

- enum class **Result** { **OK** , **ERR** }

### Public Member Functions

- Result [Init \(I2CHandle i2c\)](#)

#### 12.94.1 Member Function Documentation

##### 12.94.1.1 Init()

```
Result daisy::Pcm3060::Init (
 I2CHandle i2c)
```

Initializes the PCM3060 in 24-bit MSB aligned I2S mode, and disables powersave

**Parameters**

|            |                                                                    |
|------------|--------------------------------------------------------------------|
| <i>i2c</i> | Initialized <a href="#">I2CHandle</a> configured at 400kHz or less |
|------------|--------------------------------------------------------------------|

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

- src/dev/codec\_pcm3060.h

## 12.95 daisy::PersistentStorage< SettingStruct > Class Template Reference

Non Volatile storage class for persistent settings on an external flash device.

```
#include <PersistentStorage.h>
```

### Public Types

- enum class **State** { **UNKNOWN** = 0 , **FACTORY** = 1 , **USER** = 2 }

### Public Member Functions

- [PersistentStorage \(QSPIHandle &qspi\)](#)
- void [Init \(const SettingStruct &defaults, uint32\\_t address\\_offset=0\)](#)
- [State GetState \(\) const](#)
- [SettingStruct & GetSettings \(\)](#)
- void [Save \(\)](#)
- void [RestoreDefaults \(\)](#)

#### 12.95.1 Detailed Description

```
template<typename SettingStruct>
class daisy::PersistentStorage< SettingStruct >
```

Non Volatile storage class for persistent settings on an external flash device.

#### Author

shensley

Storage occupied by the struct will be one word larger than the SettingStruct used. The extra word is used to store the state of the data, and whether it's been overwritten or not.

**Todo**

- Make [Save\(\)](#) non-blocking
- Add wear leveling

#### 12.95.2 Member Enumeration Documentation

### 12.95.2.1 State

```
template<typename SettingStruct >
enum daisy::PersistentStorage::State [strong]
```

State of the storage. When created, prior to initialiation, the state will be Unknown

During initialization, the state will be changed to either FACTORY, or USER.

If this is the first time these settings are being written to the target address, the defaults will be written to that location, and the state will be set to FACTORY.

Once the first user-trigger save has been made, the state will be updated to USER to indicate that the defaults have overwritten.

## 12.95.3 Constructor & Destructor Documentation

### 12.95.3.1 PersistentStorage()

```
template<typename SettingStruct >
daisy::PersistentStorage< SettingStruct >::PersistentStorage (
 QSPIHandle & qspi) [inline]
```

Constructor for storage class

#### Parameters

|             |                                            |
|-------------|--------------------------------------------|
| <i>qspi</i> | reference to the hardware qspi peripheral. |
|-------------|--------------------------------------------|

## 12.95.4 Member Function Documentation

### 12.95.4.1 GetSettings()

```
template<typename SettingStruct >
SettingStruct& daisy::PersistentStorage< SettingStruct >::GetSettings () [inline]
```

Returns a reference to the setting struct

### 12.95.4.2 GetState()

```
template<typename SettingStruct >
State daisy::PersistentStorage< SettingStruct >::GetState () const [inline]
```

Returns the state of the Persistent Data

### 12.95.4.3 Init()

```
template<typename SettingStruct >
void daisy::PersistentStorage< SettingStruct >::Init (
 const SettingStruct & defaults,
 uint32_t address_offset = 0) [inline]
```

Initialize Storage class

The values in this class will be stored as the default for restoration to 'factory' settings.

#### Parameters

|                       |                                                                                                                                                                            |
|-----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>defaults</i>       | should be a setting structure containing the default values. this will be updated to contain the stored data.                                                              |
| <i>address_offset</i> | offset for location on the QSPI chip (offset to base address of device). This defaults to the first address on the chip, and will be masked to the nearest multiple of 256 |

### 12.95.4.4 RestoreDefaults()

```
template<typename SettingStruct >
void daisy::PersistentStorage< SettingStruct >::RestoreDefaults () [inline]
```

Restores the settings stored in the QSPI

### 12.95.4.5 Save()

```
template<typename SettingStruct >
void daisy::PersistentStorage< SettingStruct >::Save () [inline]
```

Performs the save operation, storing the storage

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

- src/util/PersistentStorage.h

## 12.96 daisy::Pin Struct Reference

representation of hardware port/pin combination

```
#include <daisy_core.h>
```

## Public Member Functions

- `constexpr Pin (const GPIOPort pt, const uint8_t pn)`  
*Constructor creates a valid pin.*
- `constexpr Pin ()`  
*Basic Constructor creates an invalid Pin object.*
- `constexpr bool IsValid () const`  
*checks validity of a Pin*
- `constexpr bool operator==(const Pin &rhs) const`  
*comparison operator for checking equality between Pin objects*
- `constexpr bool operator!=(const Pin &rhs) const`  
*comparison operator for checking inequality between Pin objects*
- `operator dsy_gpio_pin () const`  
*conversion operation for converting to the old-style representation of a pin.*

## Public Attributes

- `GPIOPort port`
- `uint8_t pin`

### 12.96.1 Detailed Description

representation of hardware port/pin combination

### 12.96.2 Constructor & Destructor Documentation

#### 12.96.2.1 Pin()

```
constexpr daisy::Pin::Pin (
 const GPIOPort pt,
 const uint8_t pn) [inline], [constexpr]
```

Constructor creates a valid pin.

#### Parameters

|                 |                                                          |
|-----------------|----------------------------------------------------------|
| <code>pt</code> | GPIOPort between PA, and PK corresponding to STM32 Port. |
| <code>pn</code> | pin number in range of 0-15                              |

### 12.96.3 Member Function Documentation

### 12.96.3.1 IsValid()

```
constexpr bool daisy::Pin::IsValid () const [inline], [constexpr]
```

checks validity of a [Pin](#)

Return values

|                |                                                            |
|----------------|------------------------------------------------------------|
| <i>returns</i> | true if the port is a valid hardware pin, otherwise false. |
|----------------|------------------------------------------------------------|

### 12.96.3.2 operator dsy\_gpio\_pin()

```
daisy::Pin::operator dsy_gpio_pin () const [inline]
```

conversion operation for converting to the old-style representation of a pin.

This allows the new [Pin](#) type to be used in place of the older, [dsy\\_gpio\\_pin](#) type.

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

- src/daisy\_core.h

## 12.97 daisy::PitchBendEvent Struct Reference

```
#include <MidiEvent.h>
```

### Public Attributes

- int [channel](#)
- int16\_t [value](#)

### 12.97.1 Detailed Description

Struct containing pitch bend value for a given channel. Can be made from [MidiEvent](#)

### 12.97.2 Member Data Documentation

#### 12.97.2.1 channel

```
int daisy::PitchBendEvent::channel
```

&

### 12.97.2.2 value

```
int16_t daisy::PitchBendEvent::value
```

&

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

- src/hid/MidiEvent.h

## 12.98 daisy::PolyModeOnEvent Struct Reference

```
#include <MidiEvent.h>
```

### Public Attributes

- int [channel](#)

### 12.98.1 Detailed Description

Struct containing [PolyModeOnEvent](#) data. Can be made from [MidiEvent](#)

### 12.98.2 Member Data Documentation

#### 12.98.2.1 channel

```
int daisy::PolyModeOnEvent::channel
```

&

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

- src/hid/MidiEvent.h

## 12.99 daisy::PolyphonicKeyPressureEvent Struct Reference

```
#include <MidiEvent.h>
```

## Public Attributes

- int **channel**
- uint8\_t **note**
- uint8\_t **pressure**

### 12.99.1 Detailed Description

Struct containing note, and pressure data for a given channel. Can be made from [MidiEvent](#)

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

- src/hid/MidiEvent.h

## 12.100 daisy::PotMonitor< BackendType, numPots > Class Template Reference

### Public Member Functions

- void [Init](#) ([UiEventQueue](#) &queueToAddEventsTo, BackendType &backend, uint16\_t idleTimeoutMs=500, float deadBandIdle=1.0/(1<< 10), float deadBand=1.0/(1<< 12))
- void [Process](#) ()
- bool [IsMoving](#) (uint16\_t potId) const
- float [GetCurrentPotValue](#) (uint16\_t potId) const
- BackendType & [GetBackend](#) ()
- uint16\_t [GetNumPotsMonitored](#) () const

### 12.100.1 Member Function Documentation

#### 12.100.1.1 GetBackend()

```
template<typename BackendType , uint32_t numPots>
BackendType& daisy::PotMonitor< BackendType, numPots >::GetBackend () [inline]
```

Returns the BackendType that is used by the monitor.

#### 12.100.1.2 GetCurrentPotValue()

```
template<typename BackendType , uint32_t numPots>
float daisy::PotMonitor< BackendType, numPots >::GetCurrentPotValue (
 uint16_t potId) const [inline]
```

For a given potentiometer, this will return the last value that was posted to the [UiEventQueue](#).

**Parameters**

|              |                                                |
|--------------|------------------------------------------------|
| <i>potId</i> | The unique ID of the potentiometer (< numPots) |
|--------------|------------------------------------------------|

**12.100.1.3 GetNumPotsMonitored()**

```
template<typename BackendType , uint32_t numPots>
uint16_t daisy::PotMonitor< BackendType, numPots >::GetNumPotsMonitored () const [inline]
```

Returns the number of pots that are monitored by this class.

**12.100.1.4 Init()**

```
template<typename BackendType , uint32_t numPots>
void daisy::PotMonitor< BackendType, numPots >::Init (
 UiEventQueue & queueToAddEventsTo,
 BackendType & backend,
 uint16_t idleTimeoutMs = 500,
 float deadBandIdle = 1.0 / (1 << 10),
 float deadBand = 1.0 / (1 << 12)) [inline]
```

Initialises the PotMonitor.

**Parameters**

|                           |                                                                                                                  |
|---------------------------|------------------------------------------------------------------------------------------------------------------|
| <i>queueToAddEventsTo</i> | The <a href="#">UiEventQueue</a> to which events should be posted.                                               |
| <i>backend</i>            | The backend that supplies the current value of each potentiometer.                                               |
| <i>idleTimeoutMs</i>      | When the pot is currently moving, but no event is generated over "idleTimeoutMs", the pot enters the idle state. |
| <i>deadBandIdle</i>       | The dead band that must be exceeded before a movement is detected when the pot is currently idle.                |
| <i>deadBand</i>           | The dead band that must be exceeded before a movement is detected when the pot is currently moving.              |

**12.100.1.5 IsMoving()**

```
template<typename BackendType , uint32_t numPots>
bool daisy::PotMonitor< BackendType, numPots >::IsMoving (
 uint16_t potId) const [inline]
```

Returns true, if the requested pot is currently being moved.

**Parameters**

|                           |                                                |
|---------------------------|------------------------------------------------|
| <i>pot</i> ↪<br><i>Id</i> | The unique ID of the potentiometer (< numPots) |
|---------------------------|------------------------------------------------|

**12.100.1.6 Process()**

```
template<typename BackendType , uint32_t numPots>
void daisy::PotMonitor< BackendType, numPots >::Process () [inline]
```

Checks the value of each pot and generates messages for the UIEventQueue. Call this at regular intervals, ideally from your main() idle loop.

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

- src/ui/PotMonitor.h

**12.101 daisy::ProgramChangeEvent Struct Reference**

```
#include <MidiEvent.h>
```

**Public Attributes**

- int [channel](#)
- uint8\_t [program](#)

**12.101.1 Detailed Description**

Struct containing new program number, for a given channel. Can be made from [MidiEvent](#)

**12.101.2 Member Data Documentation****12.101.2.1 channel**

```
int daisy::ProgramChangeEvent::channel
```

&

### 12.101.2.2 program

```
uint8_t daisy::ProgramChangeEvent::program
&
```

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

- src/hid/MidiEvent.h

## 12.102 daisy::QSPIHandle Class Reference

```
#include <qspi.h>
```

### Classes

- struct [Config](#)

### Public Types

- enum [Result](#) { **OK** = 0 , **ERR** }
- enum [Status](#) { **GOOD** = 0 , **E\_HAL\_ERROR** , **E\_SWITCHING\_MODES** , **E\_INVALID\_MODE** }

### Public Member Functions

- [Result Init](#) (const [Config](#) &config)
- const [Config](#) & [GetConfig](#) () const
- [Result DelInit](#) ()
- [Result WritePage](#) (uint32\_t address, uint32\_t size, uint8\_t \*buffer)
- [Result Write](#) (uint32\_t address, uint32\_t size, uint8\_t \*buffer)
- [Result Erase](#) (uint32\_t start\_addr, uint32\_t end\_addr)
- [Result EraseSector](#) (uint32\_t address)
- [Status GetStatus](#) ()
- void \* [GetData](#) (uint32\_t offset=0)
- [QSPIHandle](#) (const [QSPIHandle](#) &other)=default
- [QSPIHandle](#) & [operator=](#) (const [QSPIHandle](#) &other)=default

### 12.102.1 Detailed Description

Driver for QSPI peripheral to interface with external flash memory.

Currently supported QSPI Devices:

IS25LP080D

### 12.102.2 Member Enumeration Documentation

#### 12.102.2.1 Status

```
enum daisy::QSPIHandle::Status
```

Indicates the current status of the module. Warnings are indicated by a leading W. Errors are indicated by a leading E and cause an immediate exit.

**Parameters**

|                          |                                                                         |
|--------------------------|-------------------------------------------------------------------------|
| <i>GOOD</i>              | - No errors have been reported.                                         |
| <i>E_HAL_ERROR</i>       | - HAL code did not return HAL_OK.                                       |
| <i>E_SWITCHING_MODES</i> | - An error was encountered while switching QSPI peripheral mode.        |
| <i>E_INVALID_MODE</i>    | - QSPI should not be written to while the program is executing from it. |

**12.102.3 Member Function Documentation****12.102.3.1 DeInit()**

```
Result daisy::QSPIHandle::DeInit ()
```

Deinitializes the peripheral This should be called before reinitializing QSPI in a different mode.

**Returns**

Result::OK or Result::ERR

**12.102.3.2 Erase()**

```
Result daisy::QSPIHandle::Erase (
 uint32_t start_addr,
 uint32_t end_addr)
```

Erases the area specified on the chip. Erasures will happen by 4K, 32K or 64K increments. Smallest erase possible is 4kB at a time. (on IS25LP\*)

**Parameters**

|                   |                               |
|-------------------|-------------------------------|
| <i>start_addr</i> | Address to begin erasing from |
| <i>end_addr</i>   | Address to stop erasing at    |

**Returns**

Result::OK or Result::ERR

**12.102.3.3 EraseSector()**

```
Result daisy::QSPIHandle::EraseSector (
 uint32_t address)
```

Erases a single sector of the chip.  
TODO: Document the size of this function.

#### Parameters

|                   |                            |
|-------------------|----------------------------|
| <code>addr</code> | Address of sector to erase |
|-------------------|----------------------------|

#### Returns

`Result::OK` or `Result::ERR`

### 12.102.3.4 GetConfig()

```
const Config& daisy::QSPIHandle::GetConfig () const
```

Returns the current config.

### 12.102.3.5 GetData()

```
void* daisy::QSPIHandle::GetData (
 uint32_t offset = 0)
```

Returns a pointer to the actual memory used. The memory at this address is read-only to write to it use the Write function.

#### Parameters

|                     |                                                              |
|---------------------|--------------------------------------------------------------|
| <code>offset</code> | returns the pointer starting this many bytes into the memory |
|---------------------|--------------------------------------------------------------|

### 12.102.3.6 GetStatus()

```
Status daisy::QSPIHandle::GetStatus ()
```

Returns the current class status. Useful for debugging.

#### Returns

`Status`

### 12.102.3.7 Init()

```
Result daisy::QSPIHandle::Init (
 const Config & config)
```

Initializes QSPI peripheral, and Resets, and prepares memory for access.

**Parameters**

|               |                                                                                        |
|---------------|----------------------------------------------------------------------------------------|
| <i>config</i> | should be populated with the mode, device and pin_config before calling this function. |
|---------------|----------------------------------------------------------------------------------------|

**Returns**

Result::OK or Result::ERR

**12.102.3.8 Write()**

```
Result daisy::QSPIHandle::Write (
 uint32_t address,
 uint32_t size,
 uint8_t * buffer)
```

Writes data in buffer to to the QSPI. Starting at address to address+size

**Parameters**

|                |                     |
|----------------|---------------------|
| <i>address</i> | Address to write to |
| <i>size</i>    | Buffer size         |
| <i>buffer</i>  | Buffer to write     |

**Returns**

Result::OK or Result::ERR

**12.102.3.9 WritePage()**

```
Result daisy::QSPIHandle::WritePage (
 uint32_t address,
 uint32_t size,
 uint8_t * buffer)
```

Writes a single page to to the specified address on the QSPI chip. For IS25LP\*, page size is 256 bytes.

**Parameters**

|                |                     |
|----------------|---------------------|
| <i>address</i> | Address to write to |
| <i>size</i>    | Buffer size         |
| <i>buffer</i>  | Buffer to write     |

**Returns**

Result::OK or Result::ERR

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

- src/per/qspi.h

## 12.103 daisy::Random Class Reference

True [Random Number Generator](#) access.

```
#include <rng.h>
```

### Static Public Member Functions

- static void [Init](#) ()
- static void [DeInit](#) ()
- static uint32\_t [GetValue](#) ()
- static float [GetFloat](#) (float min=0.f, float max=1.f)
- static bool [IsReady](#) ()

### 12.103.1 Detailed Description

True [Random Number Generator](#) access.

**Author**

shensley

Provides static access to the built-in True [Random Number Generator](#)

### 12.103.2 Member Function Documentation

#### 12.103.2.1 DeInit()

```
static void daisy::Random::DeInit () [static]
```

Deinitializes the Peripheral

#### 12.103.2.2 GetFloat()

```
static float daisy::Random::GetFloat (
 float min = 0.f,
 float max = 1.f) [static]
```

Returns a floating point value between the specified minimum and maximum. Calls [GetValue\(\)](#) internally.

**Parameters**

|            |                                              |
|------------|----------------------------------------------|
| <i>min</i> | the minimum value to return, defaults to 0.f |
| <i>max</i> | the maximum value to return, defaults to 1.f |

**12.103.2.3 GetValue()**

```
static uint32_t daisy::Random::GetValue () [static]
```

Returns a randomly generated 32-bit number This is done by polling the peripheral, and can block for up to 100ms.

To avoid blocking issues, the IsReady function can be used to check if a value is ready before calling this function.

If there is an issue with the peripheral, or a timeout occurs the return value will be 0.

**Returns**

a 32-bit random number

**12.103.2.4 Init()**

```
static void daisy::Random::Init () [static]
```

Initializes the Peripheral

This is called from [System::Init](#), so the GetValue, and GetFloat functions can be used without the application needing to manually initialize the RNG.

**12.103.2.5 IsReady()**

```
static bool daisy::Random::IsReady () [static]
```

Checks the peripheral to see if a new value is ready

**Returns**

true if conditioning has finished and a new value can be read

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

- src/per/rng.h

## 12.104 daisy::Rectangle Class Reference

### Public Member Functions

- `Rectangle (int16_t width, int16_t height)`
- `Rectangle (int16_t x, int16_t y, int16_t width, int16_t height)`
- `Rectangle (const Rectangle &other)`
- `Rectangle & operator= (const Rectangle &other)`
- `bool operator== (const Rectangle &other) const`
- `bool operator!= (const Rectangle &other) const`
- `bool IsEmpty () const`
- `int16_t GetX () const`
- `int16_t GetY () const`
- `int16_t GetWidth () const`
- `int16_t GetHeight () const`
- `int16_t GetRight () const`
- `int16_t GetBottom () const`
- `int16_t GetCenterX () const`
- `int16_t GetCenterY () const`
- `Rectangle WithX (int16_t x) const`
- `Rectangle WithY (int16_t y) const`
- `Rectangle WithWidth (int16_t width) const`
- `Rectangle WithHeight (int16_t height) const`
- `Rectangle WithSize (int16_t width, int16_t height) const`
- `Rectangle WithWidthKeepingCenter (int16_t width) const`
- `Rectangle WithHeightKeepingCenter (int16_t height) const`
- `Rectangle WithSizeKeepingCenter (int16_t width, int16_t height) const`
- `Rectangle Reduced (int16_t sizeToReduce) const`
- `Rectangle Reduced (int16_t xToReduce, int16_t yToReduce) const`
- `Rectangle Translated (int16_t x, int16_t y) const`
- `Rectangle WithLeft (int16_t left) const`
- `Rectangle WithRight (int16_t right) const`
- `Rectangle WithTop (int16_t top) const`
- `Rectangle WithBottom (int16_t bottom) const`
- `Rectangle WithTrimmedLeft (int16_t pxToTrim) const`
- `Rectangle WithTrimmedRight (int16_t pxToTrim) const`
- `Rectangle WithTrimmedTop (int16_t pxToTrim) const`
- `Rectangle WithTrimmedBottom (int16_t pxToTrim) const`
- `Rectangle WithCenterX (int16_t centerX) const`
- `Rectangle WithCenterY (int16_t centerY) const`
- `Rectangle WithCenter (int16_t centerX, int16_t centerY) const`
- `Rectangle RemoveFromLeft (int16_t pxToRemove)`
- `Rectangle RemoveFromRight (int16_t pxToRemove)`
- `Rectangle RemoveFromTop (int16_t pxToRemove)`
- `Rectangle RemoveFromBottom (int16_t pxToRemove)`
- `Rectangle AlignedWithin (const Rectangle &other, Alignment alignment) const`

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

- `src/hid/Disp/Graphics_common.h`

## 12.105 daisy::ResetAllControllersEvent Struct Reference

```
#include <MidiEvent.h>
```

### Public Attributes

- int [channel](#)
- uint8\_t [value](#)

#### 12.105.1 Detailed Description

Struct containing [ResetAllControllersEvent](#) data. Can be made from [MidiEvent](#)

#### 12.105.2 Member Data Documentation

##### 12.105.2.1 channel

```
int daisy::ResetAllControllersEvent::channel
&
```

##### 12.105.2.2 value

```
uint8_t daisy::ResetAllControllersEvent::value
&
```

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

- src/hid/MidiEvent.h

## 12.106 daisy::RgbLed Class Reference

```
#include <rgb_led.h>
```

### Public Member Functions

- void [Init \(dsy\\_gpio\\_pin red, dsy\\_gpio\\_pin green, dsy\\_gpio\\_pin blue, bool invert\)](#)
- void [Set \(float r, float g, float b\)](#)
- void [SetRed \(float val\)](#)
- void [SetGreen \(float val\)](#)
- void [SetBlue \(float val\)](#)
- void [SetColor \(Color c\)](#)
- void [Update \(\)](#)

### 12.106.1 Detailed Description

3x LEDs configured as an RGB for ease of use.

### 12.106.2 Member Function Documentation

#### 12.106.2.1 Init()

```
void daisy::RgbLed::Init (
 dsy_gpio_pin red,
 dsy_gpio_pin green,
 dsy_gpio_pin blue,
 bool invert)
```

Initializes 3x [GPIO](#) Pins as red, green, and blue elements of an RGB LED

##### Parameters

|               |                    |
|---------------|--------------------|
| <i>red</i>    | Red element        |
| <i>green</i>  | Green element      |
| <i>blue</i>   | Blue element       |
| <i>invert</i> | Flips led polarity |

#### 12.106.2.2 Set()

```
void daisy::RgbLed::Set (
 float r,
 float g,
 float b)
```

Sets each element of the LED with a floating point number 0-1

##### Parameters

|          |               |
|----------|---------------|
| <i>r</i> | Red element   |
| <i>g</i> | Green element |
| <i>b</i> | Blue element  |

#### 12.106.2.3 SetBlue()

```
void daisy::RgbLed::SetBlue (
 float val)
```

Sets the blue channel of the LED with a floating point number 0-1

**Parameters**

|            |                                |
|------------|--------------------------------|
| <i>val</i> | brightness of the blue channel |
|------------|--------------------------------|

#### 12.106.2.4 SetColor()

```
void daisy::RgbLed::SetColor (
 Color c)
```

Sets the RGB using a [Color](#) object.

**Parameters**

|          |                      |
|----------|----------------------|
| <i>c</i> | Color object to set. |
|----------|----------------------|

#### 12.106.2.5 SetGreen()

```
void daisy::RgbLed::SetGreen (
 float val)
```

Sets the green channel of the LED with a floating point number 0-1

**Parameters**

|            |                                 |
|------------|---------------------------------|
| <i>val</i> | brightness of the green channel |
|------------|---------------------------------|

#### 12.106.2.6 SetRed()

```
void daisy::RgbLed::SetRed (
 float val)
```

Sets the red channel of the LED with a floating point number 0-1

**Parameters**

|            |                               |
|------------|-------------------------------|
| <i>val</i> | brightness of the red channel |
|------------|-------------------------------|

### 12.106.2.7 Update()

```
void daisy::RgbLed::Update ()
```

Updates the PWM of the LED based on the current values. Should be called at a regular interval. (i.e. 1kHz/1ms)

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

- src/hid/rgb\_led.h

## 12.107 daisy::RingBuffer< T, size > Class Template Reference

```
#include <ringbuffer.h>
```

### Public Member Functions

- void `Init ()`
- `size_t capacity () const`
- `size_t writable () const`
- `size_t readable () const`
- `bool isEmpty () const`
- `void Write (T v)`
- `void Overwrite (T v)`
- `T Read ()`
- `T ImmediateRead ()`
- `void Flush ()`
- `void Swallow (size_t n)`
- `void ImmediateRead (T *destination, size_t num_elements)`
- `void Overwrite (const T *source, size_t num_elements)`
- `void Advance (size_t num_elements)`
- `T * GetMutableBuffer ()`

### 12.107.1 Detailed Description

```
template<typename T, size_t size>
class daisy::RingBuffer< T, size >
```

Utility Ring Buffer  
imported from pichenettes/stmlib

### 12.107.2 Member Function Documentation

**12.107.2.1 Advance()**

```
template<typename T , size_t size>
void daisy::RingBuffer< T, size >::Advance (
 size_t num_elements) [inline]
```

Advances the write pointer, for when a peripheral is writing to the buffer.

**12.107.2.2 capacity()**

```
template<typename T , size_t size>
size_t daisy::RingBuffer< T, size >::capacity () const [inline]
```

**Returns**

The total size of the ring buffer

**12.107.2.3 Flush()**

```
template<typename T , size_t size>
void daisy::RingBuffer< T, size >::Flush () [inline]
```

Flushes unread elements from the ring buffer

**12.107.2.4 GetMutableBuffer()**

```
template<typename T , size_t size>
T* daisy::RingBuffer< T, size >::GetMutableBuffer () [inline]
```

Returns a pointer to the actual Ring Buffer Useful for when a peripheral needs direct access to the buffer.

**12.107.2.5 ImmediateRead() [1/2]**

```
template<typename T , size_t size>
T daisy::RingBuffer< T, size >::ImmediateRead () [inline]
```

Reads next element from ring buffer immediately

**Returns**

read value

**12.107.2.6 ImmediateRead() [2/2]**

```
template<typename T , size_t size>
void daisy::RingBuffer< T, size >::ImmediateRead (
 T * destination,
 size_t num_elements) [inline]
```

Reads a number of elements into a buffer immediately

**Parameters**

|                     |                              |
|---------------------|------------------------------|
| <i>destination</i>  | buffer to write to           |
| <i>num_elements</i> | number of elements in buffer |

**12.107.2.7 Init()**

```
template<typename T , size_t size>
void daisy::RingBuffer< T, size >::Init () [inline]
```

Initializes the Ring Buffer

**12.107.2.8 isEmpty()**

```
template<typename T , size_t size>
bool daisy::RingBuffer< T, size >::isEmpty () const [inline]
```

**Returns**

True, if the buffer is empty.

**12.107.2.9 Overwrite() [1/2]**

```
template<typename T , size_t size>
void daisy::RingBuffer< T, size >::Overwrite (
 const T * source,
 size_t num_elements) [inline]
```

Overwrites a number of elements using the source buffer as input.

**Parameters**

|                     |                              |
|---------------------|------------------------------|
| <i>source</i>       | Input buffer                 |
| <i>num_elements</i> | Number of elements in source |

**12.107.2.10 Overwrite() [2/2]**

```
template<typename T , size_t size>
void daisy::RingBuffer< T, size >::Overwrite (
 T v) [inline]
```

Writes the new element to the ring buffer, overwriting unread data if necessary.

**Parameters**

|          |                    |
|----------|--------------------|
| <i>v</i> | Value to overwrite |
|----------|--------------------|

**12.107.2.11 Read()**

```
template<typename T , size_t size>
T daisy::RingBuffer< T, size >::Read () [inline]
```

Reads the first available element from the ring buffer

**Returns**

read value

**12.107.2.12 readable()**

```
template<typename T , size_t size>
size_t daisy::RingBuffer< T, size >::readable () const [inline]
```

**Returns**

number of unread elements in ring buffer

**12.107.2.13 Swallow()**

```
template<typename T , size_t size>
void daisy::RingBuffer< T, size >::Swallow (
 size_t n) [inline]
```

Read enough samples to make it possible to read 1 sample.

**Parameters**

|          |             |
|----------|-------------|
| <i>n</i> | Size of T ? |
|----------|-------------|

**12.107.2.14 writable()**

```
template<typename T , size_t size>
size_t daisy::RingBuffer< T, size >::writable () const [inline]
```

**Returns**

the number of samples that can be written to ring buffer without overwriting unread data.

**12.107.2.15 Write()**

```
template<typename T , size_t size>
void daisy::RingBuffer< T, size >::Write (
 T v) [inline]
```

Writes the value to the next available position in the ring buffer

**Parameters**

|                |                |
|----------------|----------------|
| <code>v</code> | Value to write |
|----------------|----------------|

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

- src/util/ringbuffer.h

**12.108 daisy::RingBuffer< T, 0 > Class Template Reference**

```
#include <ringbuffer.h>
```

**Public Member Functions**

- void `Init ()`
- `size_t capacity () const`
- `size_t writable () const`
- `size_t readable () const`
- void `Write (T v)`
- void `Overwrite (T v)`
- `T Read ()`
- `T ImmediateRead ()`
- void `Flush ()`
- void `ImmediateRead (T *destination, size_t num_elements)`
- void `Overwrite (const T *source, size_t num_elements)`

**12.108.1 Detailed Description**

```
template<typename T>
class daisy::RingBuffer< T, 0 >
```

Utility Ring Buffer imported from pichenettes/stmlib

## 12.108.2 Member Function Documentation

### 12.108.2.1 capacity()

```
template<typename T >
size_t daisy::RingBuffer< T, 0 >::capacity () const [inline]
```

#### Returns

0

### 12.108.2.2 Flush()

```
template<typename T >
void daisy::RingBuffer< T, 0 >::Flush () [inline]
```

Flush the buffer

### 12.108.2.3 ImmediateRead() [1/2]

```
template<typename T >
T daisy::RingBuffer< T, 0 >::ImmediateRead () [inline]
```

#### Returns

Read value

### 12.108.2.4 ImmediateRead() [2/2]

```
template<typename T >
void daisy::RingBuffer< T, 0 >::ImmediateRead (
 T * destination,
 size_t num_elements) [inline]
```

#### Parameters

|                     |   |
|---------------------|---|
| <i>destination</i>  | & |
| <i>num_elements</i> | & |

### 12.108.2.5 Init()

```
template<typename T >
void daisy::RingBuffer< T, 0 >::Init () [inline]
```

Initialize ringbuffer

### 12.108.2.6 Overwrite() [1/2]

```
template<typename T >
void daisy::RingBuffer< T, 0 >::Overwrite (
 const T * source,
 size_t num_elements) [inline]
```

Parameters

|              |   |
|--------------|---|
| source       | 3 |
| num_elements | & |

### 12.108.2.7 Overwrite() [2/2]

```
template<typename T >
void daisy::RingBuffer< T, 0 >::Overwrite (
 T v) [inline]
```

Parameters

|   |                    |
|---|--------------------|
| v | Value to overwrite |
|---|--------------------|

### 12.108.2.8 Read()

```
template<typename T >
T daisy::RingBuffer< T, 0 >::Read () [inline]
```

Returns

Read value

### 12.108.2.9 readable()

```
template<typename T >
size_t daisy::RingBuffer< T, 0 >::readable () const [inline]
```

Returns

0

### 12.108.2.10 writable()

```
template<typename T >
size_t daisy::RingBuffer< T, 0 >::writable () const [inline]
```

Returns

0

### 12.108.2.11 Write()

```
template<typename T >
void daisy::RingBuffer< T, 0 >::Write (
 T v) [inline]
```

Parameters

|   |                |
|---|----------------|
| v | Value to write |
|---|----------------|

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

- src/util/ringbuffer.h

## 12.109 daisy::SaiHandle Class Reference

```
#include <sai.h>
```

### Classes

- struct [Config](#)

## Public Types

- enum class **Result** { **OK** , **ERR** }
- typedef void(\* **CallbackFunctionPtr**) (int32\_t \*in, int32\_t \*out, size\_t size)

## Public Member Functions

- SaiHandle** (const **SaiHandle** &other)=default
- SaiHandle** & **operator=** (const **SaiHandle** &other)=default
- Result Init** (const **Config** &config)
- Result DelInit** ()
- const **Config** & **GetConfig** () const
- Result StartDma** (int32\_t \*buffer\_rx, int32\_t \*buffer\_tx, size\_t size, **CallbackFunctionPtr** callback)
- Result StopDma** ()
- float **GetSampleRate** ()
- size\_t **GetBlockSize** ()
- float **GetBlockRate** ()
- size\_t **GetOffset** () const
- bool **IsInitialized** () const

### 12.109.1 Detailed Description

Support for I2S Audio Protocol with different bit-depth, samplerate options Allows for master or slave, as well as freedom of selecting direction, and other behavior for each peripheral, and block.

DMA Transfer commands must use buffers located within non-cached memory or use cache maintenance To declare an uninitialized global element in the DMA memory section: int32\_t DSY\_DMA\_BUFFER\_SECTOR my\_buffer[96];

Callback functions will be called once per half of the buffer. In the above example, the callback function would be called once for every 48 samples.

Use SAI Handle like this:

```
SaiHandle::Config sai_config; sai_config.periph = SaiHandle::Config::Peripheral::SAI_1; sai_config.sr = SaiHandle::Config::SampleRate::SAI_48KHZ; sai_config.bit_depth = SaiHandle::Config::BitDepth::SAI_24BIT; sai_config.a_sync = SaiHandle::Config::Sync::MASTER; sai_config.b_sync = SaiHandle::Config::Sync::SLAVE; sai_config.a_dir = SaiHandle::Config::Direction::RECEIVE; sai_config.b_dir = SaiHandle::Config::Direction::TRANSMIT; sai_config.pin_config.fs = {DSY_GPIOE, 4}; sai_config.pin_config.mclk = {DSY_GPIOE, 2}; sai_config.pin_config.sck = {DSY_GPIOE, 5}; sai_config.pin_config.sa = {DSY_GPIOE, 6}; sai_config.pin_config.sb = {DSY_GPIOE, 3}; // Then Initialize SaiHandle sai; sai.Init(sai_config); // Now you can use it: sai.StartDma(. . .);
```

### 12.109.2 Member Typedef Documentation

#### 12.109.2.1 CallbackFunctionPtr

```
typedef void(* daisy::SaiHandle::CallbackFunctionPtr) (int32_t *in, int32_t *out, size_t size)
```

Callback Function to be called when DMA transfer is complete and half complete. This callback is prepared however the data is transmitted/received from the device. For example, using an AK4556 the data will be interleaved 24bit MSB Justified

The hid/audio class will be allow for type conversions, de-interleaving, etc.

### 12.109.3 Member Enumeration Documentation

#### 12.109.3.1 Result

```
enum daisy::SaiHandle::Result [strong]
```

Return values for SAI functions

### 12.109.4 Member Function Documentation

#### 12.109.4.1 DeInit()

```
Result daisy::SaiHandle::DeInit ()
```

Deinitializes an SAI peripheral

#### 12.109.4.2 GetBlockRate()

```
float daisy::SaiHandle::GetBlockRate ()
```

Returns the Block Rate of the current stream based on the size of the buffer passed in, and the current samplerate.

#### 12.109.4.3 GetBlockSize()

```
size_t daisy::SaiHandle::GetBlockSize ()
```

Returns the number of samples per audio block Calculated as Buffer Size / 2 / number of channels

#### 12.109.4.4 GetConfig()

```
const Config& daisy::SaiHandle::GetConfig () const
```

Returns the current configuration

#### 12.109.4.5 GetOffset()

```
size_t daisy::SaiHandle::GetOffset () const
```

Returns the current offset within the SAI buffer, will be either 0 or size/2

**12.109.4.6 GetSampleRate()**

```
float daisy::SaiHandle::GetSampleRate ()
```

Returns the samplerate based on the current configuration

**12.109.4.7 Init()**

```
Result daisy::SaiHandle::Init (
 const Config & config)
```

Initializes an SAI peripheral

**12.109.4.8 StartDma()**

```
Result daisy::SaiHandle::StartDma (
 int32_t * buffer_rx,
 int32_t * buffer_tx,
 size_t size,
 CallbackFunctionPtr callback)
```

Starts Rx and Tx in Circular Buffer Mode The callback will be called when half of the buffer is ready, and will handle size/2 samples per callback.

**12.109.4.9 StopDma()**

```
Result daisy::SaiHandle::StopDma ()
```

Stops the DMA stream for the SAI blocks in use.

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

- src/per/sai.h

## 12.110 daisy::ScopedIrqBlocker Class Reference

```
#include <scopedirqblocker.h>
```

### 12.110.1 Detailed Description

Tempoaraily disables IRQ handlers with RAII techniques.

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

- src/util/scopedirqblocker.h

## 12.111 daisy::SdmmcHandler Class Reference

```
#include <sdmmc.h>
```

### Classes

- struct [Config](#)

### Public Types

- enum class [Result](#) { **OK** , **ERROR** }
- enum class [BusWidth](#) { **BITS\_1** , **BITS\_4** }
- enum class [Speed](#) {
 **SLOW** , **MEDIUM\_SLOW** , **STANDARD** , **FAST** ,
 **VERY\_FAST** }

### Public Member Functions

- [Result Init](#) (const [Config](#) &cfg)

#### 12.111.1 Detailed Description

Configuration for interfacing with SD cards. Currently only supports operation using FatFS filesystem

Only SDMMC1 is supported at this time.

Pins are fixed to the following: PC12 - SDMMC1 CK PD2 - SDMMC1 CMD PC8 - SDMMC1 D0 PC9 - SDMMC1 D1 (optional) PC10 - SDMMC1 D2 (optional) PC11 - SDMMC1 D3 (optional)

#### 12.111.2 Member Enumeration Documentation

##### 12.111.2.1 BusWidth

```
enum daisy::SdmmcHandler::BusWidth [strong]
```

Sets whether 4-bit mode or 1-bit mode is used for the SDMMC

###### Enumerator

|                     |                                              |
|---------------------|----------------------------------------------|
| <b>BITS←<br/>_1</b> | Only 1 bit of data per clock is transferred  |
| <b>BITS←<br/>_4</b> | 4-bits of parallel data for each clock pulse |

### 12.111.2.2 Result

```
enum daisy::SdmmcHandler::Result [strong]
```

Return values for the [SdmmcHandler](#) class

### 12.111.2.3 Speed

```
enum daisy::SdmmcHandler::Speed [strong]
```

Sets the desired clock speed of the SD card bus.

Initialization is always done at or below 400kHz, and then the user speed is set.

Enumerator

|             |                                                                            |
|-------------|----------------------------------------------------------------------------|
| SLOW        | 400kHz, initialization performed at this rate, before moving to user value |
| MEDIUM_SLOW | 12.5MHz - half of standard rate                                            |
| STANDARD    | 25MHz - DS (Default Speed for SDMMC)                                       |
| FAST        | 50MHz - HS (High Speed signaling)                                          |
| VERY_FAST   | 100MHz - SDR50 Overclocked rate for maximum transfer rates                 |

## 12.111.3 Member Function Documentation

### 12.111.3.1 Init()

```
Result daisy::SdmmcHandler::Init (
 const Config & cfg)
```

Configures the SDMMC Peripheral with the user defined settings. Initialization does not happen immediately and will be called by the filesystem (i.e. FatFS).

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

- src/per/sdmmc.h

## 12.112 SdramHandle Class Reference

### Public Types

- enum class [Result](#) { [OK](#) , [ERR](#) }

## Public Member Functions

- [Result Init \(\)](#)
- [Result DelInit \(\)](#)

### 12.112.1 Member Enumeration Documentation

#### 12.112.1.1 Result

```
enum SdramHandle::Result [strong]
```

Enumerator

|     |   |
|-----|---|
| OK  | & |
| ERR | & |

### 12.112.2 Member Function Documentation

#### 12.112.2.1 Init()

```
Result SdramHandle::Init ()
```

Initializes the SDRAM peripheral

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

- src/dev/sdram.h

## 12.113 daisy::ShiftRegister4021< num\_daisychained, num\_parallel > Class Template Reference

### Classes

- struct [Config](#)

## Public Member Functions

- void [Init \(const Config &cfg\)](#)
- void [Update \(\)](#)
- bool [State \(int index\) const](#)
- const [Config & GetConfig \(\) const](#)

## 12.113.1 Member Function Documentation

### 12.113.1.1 Init()

```
template<size_t num_daisychained = 1, size_t num_parallel = 1>
void daisy::ShiftRegister4021< num_daisychained, num_parallel >::Init (
 const Config & cfg) [inline]
```

Initializes the Device(s)

### 12.113.1.2 State()

```
template<size_t num_daisychained = 1, size_t num_parallel = 1>
bool daisy::ShiftRegister4021< num_daisychained, num_parallel >::State (
 int index) const [inline]
```

returns the last read state of the input at the index. true indicates the pin is held HIGH.

See above for the layout of data when using multiple devices in series or parallel.

### 12.113.1.3 Update()

```
template<size_t num_daisychained = 1, size_t num_parallel = 1>
void daisy::ShiftRegister4021< num_daisychained, num_parallel >::Update () [inline]
```

Reads the states of all pins on the connected device(s)

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

- src/dev/sr\_4021.h

## 12.114 ShiftRegister595 Class Reference

Device Driver for 8-bit shift register.  
CD74HC595 - 8-bit serial to parallel output shift.

```
#include <sr_595.h>
```

### Public Types

- enum Pins { PIN\_LATCH , PIN\_CLK , PIN\_DATA , NUM\_PINS }

## Public Member Functions

- void `Init (dsy_gpio_pin *pin_cfg, size_t num_daisy_chained=1)`
- void `Set (uint8_t idx, bool state)`
- void `Write ()`

### 12.114.1 Detailed Description

Device Driver for 8-bit shift register.  
CD74HC595 - 8-bit serial to parallel output shift.

#### Author

shensley

#### Date

May 2020

### 12.114.2 Member Enumeration Documentation

#### 12.114.2.1 Pins

enum `ShiftRegister595::Pins`

The following pins correspond to the hardware connections to the 595.

#### Enumerator

|          |                                    |
|----------|------------------------------------|
| PIN_CLK  | LATCH corresponds to Pin 12 "RCLK" |
| PIN_DATA | CLK corresponds to Pin 11 "SRCLK"  |
| NUM_PINS | DATA corresponds to Pin 14 "SER"   |

### 12.114.3 Member Function Documentation

#### 12.114.3.1 Init()

```
void ShiftRegister595::Init (
 dsy_gpio_pin * pin_cfg,
 size_t num_daisy_chained = 1)
```

Initializes the GPIO, and data for the ShiftRegister

**Parameters**

|                          |                                                                                   |
|--------------------------|-----------------------------------------------------------------------------------|
| <i>pin_cfg</i>           | is an array of <a href="#">dsy_gpio_pin</a> corresponding to the Pins enum above. |
| <i>num_daisy_chained</i> | (default = 1) is the number of 595 devices daisy chained together.                |

**12.114.3.2 Set()**

```
void ShiftRegister595::Set (
 uint8_t idx,
 bool state)
```

Sets the state of the specified output.

**Parameters**

|              |                                                                                     |
|--------------|-------------------------------------------------------------------------------------|
| <i>idx</i>   | The index starts with QA on the first device and ends with QH on the last device.   |
| <i>state</i> | A true state will set the output HIGH, while a false state will set the output LOW. |

**12.114.3.3 Write()**

```
void ShiftRegister595::Write ()
```

Writes the states of shift register out to the connected devices.

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

- src/dev/sr\_595.h

**12.115 daisy::SongPositionPointerEvent Struct Reference**

```
#include <MidiEvent.h>
```

**Public Attributes**

- `uint16_t position`

**12.115.1 Detailed Description**

Struct containing song position data. Can be made from [MidiEvent](#)

## 12.115.2 Member Data Documentation

### 12.115.2.1 position

```
uint16_t daisy::SongPositionPointerEvent::position
```

&

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

- src/hid/MidiEvent.h

## 12.116 daisy::SongSelectEvent Struct Reference

```
#include <MidiEvent.h>
```

### Public Attributes

- uint8\_t song

### 12.116.1 Detailed Description

Struct containing song select data. Can be made from [MidiEvent](#)

## 12.116.2 Member Data Documentation

### 12.116.2.1 song

```
uint8_t daisy::SongSelectEvent::song
```

&

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

- src/hid/MidiEvent.h

## 12.117 daisy::UI::SpecialControlIds Struct Reference

```
#include <UI.h>
```

## Public Attributes

- `uint16_t funcBttnId = UiEventQueue::invalidButtonId`
- `uint16_t okBttnId = UiEventQueue::invalidButtonId`
- `uint16_t cancelBttnId = UiEventQueue::invalidButtonId`
- `uint16_t upBttnId = UiEventQueue::invalidButtonId`
- `uint16_t downBttnId = UiEventQueue::invalidButtonId`
- `uint16_t leftBttnId = UiEventQueue::invalidButtonId`
- `uint16_t rightBttnId = UiEventQueue::invalidButtonId`
- `uint16_t menuEncoderId = UiEventQueue::invalidEncoderId`
- `uint16_t valueEncoderId = UiEventQueue::invalidEncoderId`
- `uint16_t valuePotId = UiEventQueue::invalidPotId`

### 12.117.1 Detailed Description

Contains information about the control IDs used for special functions such as arrow buttons, okay/cancel, function buttons, value sliders, etc. If such a control is available, set the corresponding variable to the control ID that's used when events are pushed to the `UiEventQueue`. If such a control is not available, use `UiEventQueue::invalidButtonId`, `UiEventQueue::invalidEncoderId` or `UiEventQueue::invalidPotId`.

### 12.117.2 Member Data Documentation

#### 12.117.2.1 menuEncoderId

```
uint16_t daisy::UI::SpecialControlIds::menuEncoderId = UiEventQueue::invalidEncoderId
```

navigates through menu selections

#### 12.117.2.2 valueEncoderId

```
uint16_t daisy::UI::SpecialControlIds::valueEncoderId = UiEventQueue::invalidEncoderId
```

changes the value of selected menu items

#### 12.117.2.3 valuePotId

```
uint16_t daisy::UI::SpecialControlIds::valuePotId = UiEventQueue::invalidPotId
```

changes the value of selected menu items (= old school "value slider")

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

- `src/ui/UI.h`

## 12.118 daisy::SpiHandle Class Reference

```
#include <spi.h>
```

### Classes

- struct [Config](#)

### Public Types

- enum class [Result](#) { [OK](#) , [ERR](#) }
- enum class [DmaDirection](#) { [RX](#) , [TX](#) }
- typedef void(\* [CallbackFunctionPtr](#)) (void \*context, [Result](#) result)

### Public Member Functions

- [SpiHandle](#) (const [SpiHandle](#) &other)=default
- [SpiHandle](#) & [operator=](#) (const [SpiHandle](#) &other)=default
- [Result](#) [Init](#) (const [Config](#) &config)
- const [Config](#) & [GetConfig](#) () const
- [Result](#) [BlockingTransmit](#) (uint8\_t \*buff, size\_t size, uint32\_t timeout=100)
- [Result](#) [BlockingTransmitAndReceive](#) (uint8\_t \*tx\_buff, uint8\_t \*rx\_buff, size\_t size, uint32\_t timeout=100)
- [Result](#) [BlockingReceive](#) (uint8\_t \*buffer, uint16\_t size, uint32\_t timeout)
- [Result](#) [DmaTransmit](#) (uint8\_t \*buff, size\_t size, [CallbackFunctionPtr](#) callback, void \*callback\_context)
- [Result](#) [DmaReceive](#) (uint8\_t \*buff, size\_t size, [SpiHandle::CallbackFunctionPtr](#) callback, void \*callback\_context)
- int [CheckError](#) ()

### 12.118.1 Detailed Description

Handler for serial peripheral interface

### 12.118.2 Member Typedef Documentation

#### 12.118.2.1 [CallbackFunctionPtr](#)

```
typedef void(* daisy::SpiHandle::CallbackFunctionPtr) (void *context, Result result)
```

A callback to be executed when a dma transfer is complete.

### 12.118.3 Member Enumeration Documentation

#### 12.118.3.1 [DmaDirection](#)

```
enum daisy::SpiHandle::DmaDirection [strong]
```

Enumerator

|    |   |
|----|---|
| RX | & |
| TX | & |

### 12.118.3.2 Result

```
enum daisy::SpiHandle::Result [strong]
```

Return values for Spi functions.

Enumerator

|     |   |
|-----|---|
| OK  | & |
| ERR | & |

## 12.118.4 Member Function Documentation

### 12.118.4.1 BlockingReceive()

```
Result daisy::SpiHandle::BlockingReceive (
 uint8_t * buffer,
 uint16_t size,
 uint32_t timeout)
```

Polling Receive

Parameters

|                |                         |
|----------------|-------------------------|
| <i>*buff</i>   | input buffer            |
| <i>size</i>    | buffer size             |
| <i>timeout</i> | How long to timeout for |

Returns

Whether the receive was successful or not

### 12.118.4.2 BlockingTransmit()

```
Result daisy::SpiHandle::BlockingTransmit (
 uint8_t * buff,
```

```
size_t size,
uint32_t timeout = 100)
```

Blocking transmit

#### Parameters

|              |              |
|--------------|--------------|
| <i>*buff</i> | input buffer |
| <i>size</i>  | buffer size  |

### 12.118.4.3 BlockingTransmitAndReceive()

```
Result daisy::SpiHandle::BlockingTransmitAndReceive (
 uint8_t * tx_buff,
 uint8_t * rx_buff,
 size_t size,
 uint32_t timeout = 100)
```

Blocking transmit and receive

#### Parameters

|                 |                               |
|-----------------|-------------------------------|
| <i>*tx_buff</i> | the transmit buffer           |
| <i>*rx_buff</i> | the receive buffer            |
| <i>size</i>     | the length of the transaction |

### 12.118.4.4 CheckError()

```
int daisy::SpiHandle::CheckError ()
```

#### Returns

the result of HAL\_SPI\_GetError() to the user.

### 12.118.4.5 DmaReceive()

```
Result daisy::SpiHandle::DmaReceive (
 uint8_t * buff,
 size_t size,
 SpiHandle::CallbackFunctionPtr callback,
 void * callback_context)
```

DMA-based receive

**Parameters**

|                         |                                                            |
|-------------------------|------------------------------------------------------------|
| <i>*buff</i>            | input buffer                                               |
| <i>size</i>             | buffer size                                                |
| <i>callback</i>         | A callback to execute when the transfer finishes, or NULL. |
| <i>callback_context</i> | A pointer that will be passed back to you in the callback. |

**Returns**

Whether the receive was successful or not

**12.118.4.6 DmaTransmit()**

```
Result daisy::SpiHandle::DmaTransmit (
 uint8_t * buff,
 size_t size,
 CallbackFunctionPtr callback,
 void * callback_context)
```

DMA-based transmit

**Parameters**

|                         |                                                            |
|-------------------------|------------------------------------------------------------|
| <i>*buff</i>            | input buffer                                               |
| <i>size</i>             | buffer size                                                |
| <i>callback</i>         | A callback to execute when the transfer finishes, or NULL. |
| <i>callback_context</i> | A pointer that will be passed back to you in the callback. |

**Returns**

Whether the transmit was successful or not

**12.118.4.7 GetConfig()**

```
const Config& daisy::SpiHandle::GetConfig () const
```

Returns the current config.

**12.118.4.8 Init()**

```
Result daisy::SpiHandle::Init (
 const Config & config)
```

Initializes handler

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

- src/per/spi.h

## 12.119 daisy::SSD130x4WireSpiTransport Class Reference

```
#include <oled_ssdl30x.h>
```

### Classes

- struct [Config](#)

### Public Member Functions

- void [Init](#) (const [Config](#) &config)
- void [SendCommand](#) (uint8\_t cmd)
- void [SendData](#) (uint8\_t \*buff, size\_t size)

#### 12.119.1 Detailed Description

4 Wire SPI Transport for SSD1306 / SSD1309 OLED display devices

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

- src/dev/oled\_ssdl30x.h

## 12.120 daisy::SSD130xDriver< width, height, Transport > Class Template Reference

```
#include <oled_ssdl30x.h>
```

### Classes

- struct [Config](#)

### Public Member Functions

- void [Init](#) ([Config](#) config)
- size\_t [Width](#) () const
- size\_t [Height](#) () const
- void [DrawPixel](#) (uint\_fast8\_t x, uint\_fast8\_t y, bool on)
- void [Fill](#) (bool on)
- void [Update](#) ()

#### 12.120.1 Detailed Description

```
template<size_t width, size_t height, typename Transport>
class daisy::SSD130xDriver< width, height, Transport >
```

A driver implementation for the SSD1306/SSD1309

## 12.120.2 Member Function Documentation

### 12.120.2.1 Update()

```
template<size_t width, size_t height, typename Transport >
void daisy::SSD130xDriver< width, height, Transport >::Update () [inline]
```

Update the display

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

- src/dev/oled\_ssdl30x.h

## 12.121 daisy::SSD130xI2CTransport Class Reference

```
#include <oled_ssdl30x.h>
```

### Classes

- struct [Config](#)

### Public Member Functions

- void [Init](#) (const [Config](#) &config)
- void [SendCommand](#) (uint8\_t cmd)
- void [SendData](#) (uint8\_t \*buff, size\_t size)

### 12.121.1 Detailed Description

I2C Transport for SSD1306 / SSD1309 OLED display devices

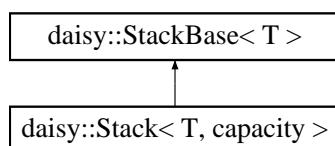
The documentation for this class was generated from the following file:

- src/dev/oled\_ssdl30x.h

## 12.122 daisy::Stack< T, capacity > Class Template Reference

```
#include <Stack.h>
```

Inheritance diagram for daisy::Stack< T, capacity >:



## Public Member Functions

- `Stack ()`
- `Stack (std::initializer_list< T > valuesToAdd)`
- `template<size_t otherCapacity>`  
`Stack (const Stack< T, otherCapacity > &other)`
- `template<size_t otherCapacity>`  
`Stack< T, capacity > & operator= (const Stack< T, otherCapacity > &other)`

## Additional Inherited Members

### 12.122.1 Detailed Description

```
template<typename T, size_t capacity>
class daisy::Stack< T, capacity >
```

A simple FILO (stack) buffer with a fixed size (usefull when allocation on the heap is not an option)

### 12.122.2 Constructor & Destructor Documentation

#### 12.122.2.1 `Stack()` [1/3]

```
template<typename T , size_t capacity>
daisy::Stack< T, capacity >::Stack () [inline]
```

Creates an empty `Stack`

#### 12.122.2.2 `Stack()` [2/3]

```
template<typename T , size_t capacity>
daisy::Stack< T, capacity >::Stack (
 std::initializer_list< T > valuesToAdd) [inline], [explicit]
```

Creates a `Stack` and adds a list of values

#### 12.122.2.3 `Stack()` [3/3]

```
template<typename T , size_t capacity>
template<size_t otherCapacity>
daisy::Stack< T, capacity >::Stack (
 const Stack< T, otherCapacity > & other) [inline]
```

Creates a `Stack` and copies all values from another `Stack`

### 12.122.3 Member Function Documentation

#### 12.122.3.1 operator=()

```
template<typename T , size_t capacity>
template<size_t otherCapacity>
Stack<T, capacity>& daisy::Stack< T, capacity >::operator= (
 const Stack< T, otherCapacity > & other) [inline]
```

Copies all values from another [Stack](#)

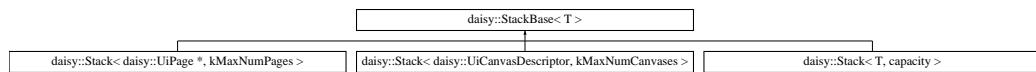
The documentation for this class was generated from the following file:

- src/util/Stack.h

## 12.123 daisy::StackBase< T > Class Template Reference

```
#include <Stack.h>
```

Inheritance diagram for daisy::StackBase< T >:



### Public Member Functions

- [StackBase< T > & operator= \(const \[StackBase< T >\]\(#\) &other\)](#)
- [bool PushBack \(const T &elementToAdd\)](#)
- [int PushBack \(std::initializer\\_list< T > valuesToAdd\)](#)
- [T PopBack \(\)](#)
- [void Clear \(\)](#)
- [T & operator\[\] \(uint32\\_t idx\)](#)
- [const T & operator\[\] \(uint32\\_t idx\) const](#)
- [bool Remove \(uint32\\_t idx\)](#)
- [int RemoveAllEqualTo \(const T &element\)](#)
- [bool Insert \(uint32\\_t idx, const T &item\)](#)
- [bool Contains \(const T &element\)](#)
- [size\\_t CountEqualTo \(const T &element\)](#)
- [bool IsEmpty \(\) const](#)
- [bool IsFull \(\) const](#)
- [size\\_t GetNumElements \(\) const](#)
- [size\\_t GetCapacity \(\) const](#)

### Protected Member Functions

- [StackBase \(T \\*buffer, size\\_t bufferSize\)](#)
- [StackBase \(T \\*buffer, size\\_t bufferSize, std::initializer\\_list< T > valuesToAdd\)](#)

### 12.123.1 Detailed Description

```
template<typename T>
class daisy::StackBase< T >
```

Capacity-independent base class for [Stack](#). Use [Stack](#) instead.

### 12.123.2 Member Function Documentation

#### 12.123.2.1 Clear()

```
template<typename T >
void daisy::StackBase< T >::Clear () [inline]
```

clears the buffer

#### 12.123.2.2 Contains()

```
template<typename T >
bool daisy::StackBase< T >::Contains (
 const T & element) [inline]
```

Returns true if the buffer contains an element equal to the provided value

#### 12.123.2.3 CountEqualTo()

```
template<typename T >
size_t daisy::StackBase< T >::CountEqualTo (
 const T & element) [inline]
```

Returns the number of elements in the buffer that are equal to the provided value

#### 12.123.2.4 GetCapacity()

```
template<typename T >
size_t daisy::StackBase< T >::GetCapacity () const [inline]
```

returns the total capacity

#### 12.123.2.5 GetNumElements()

```
template<typename T >
size_t daisy::StackBase< T >::GetNumElements () const [inline]
```

returns the number of elements in the buffer

**12.123.2.6 Insert()**

```
template<typename T >
bool daisy::StackBase< T >::Insert (
 uint32_t idx,
 const T & item) [inline]
```

adds a single element to the buffer and returns true if successfull

**12.123.2.7 IsEmpty()**

```
template<typename T >
bool daisy::StackBase< T >::IsEmpty () const [inline]
```

returns true, if the buffer is empty

**12.123.2.8 IsFull()**

```
template<typename T >
bool daisy::StackBase< T >::IsFull () const [inline]
```

returns true, if the buffer is Full

**12.123.2.9 operator=( )**

```
template<typename T >
StackBase<T>& daisy::StackBase< T >::operator= (
 const StackBase< T > & other) [inline]
```

Copies all elements from another Stack

**12.123.2.10 operator[]( ) [1/2]**

```
template<typename T >
T& daisy::StackBase< T >::operator[] (
 uint32_t idx) [inline]
```

returns an element at the given index without checking for the index to be within range.

**12.123.2.11 operator[]( ) [2/2]**

```
template<typename T >
const T& daisy::StackBase< T >::operator[] (
 uint32_t idx) const [inline]
```

returns an element at the given index without checking for the index to be within range.

### 12.123.2.12 PopBack()

```
template<typename T >
T daisy::StackBase< T >::PopBack () [inline]
```

removes and returns an element from the back of the buffer

### 12.123.2.13 PushBack() [1/2]

```
template<typename T >
bool daisy::StackBase< T >::PushBack (
 const T & elementToAdd) [inline]
```

Adds an element to the back of the buffer, returning true on success

### 12.123.2.14 PushBack() [2/2]

```
template<typename T >
int daisy::StackBase< T >::PushBack (
 std::initializer_list< T > valuesToAdd) [inline]
```

Adds multiple elements and returns the number of elements that were added

### 12.123.2.15 Remove()

```
template<typename T >
bool daisy::StackBase< T >::Remove (
 uint32_t idx) [inline]
```

removes a single element from the buffer and returns true if successful

### 12.123.2.16 RemoveAllEqualTo()

```
template<typename T >
int daisy::StackBase< T >::RemoveAllEqualTo (
 const T & element) [inline]
```

removes all elements from the buffer for which (buffer(index) == element) returns true and returns the number of elements that were removed.

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

- src/util/Stack.h

## 12.124 daisy::Switch Class Reference

```
#include <switch.h>
```

## Public Types

- enum `Type` { `TYPE_TOGGLE` , `TYPE_MOMENTARY` }
- enum `Polarity` { `POLARITY_NORMAL` , `POLARITY_INVERTED` }
- enum `Pull` { `PULL_UP` , `PULL_DOWN` , `PULL_NONE` }

## Public Member Functions

- void `Init (dsy_gpio_pin` pin, float update\_rate, `Type` t, `Polarity` pol, `Pull` pu)
- void `Init (dsy_gpio_pin` pin, float update\_rate=0.f)
- void `Debounce ()`
- bool `RisingEdge () const`
- bool `FallingEdge () const`
- bool `Pressed () const`
- bool `RawState ()`
- float `TimeHeldMs () const`
- void `SetUpdateRate (float update_rate)`

### 12.124.1 Detailed Description

Generic Class for handling momentary/latching switches

Inspired/influenced by Mutable Instruments (pichenettes) `Switch` classes

#### Author

Stephen Hensley

#### Date

December 2019

### 12.124.2 Member Enumeration Documentation

#### 12.124.2.1 Polarity

enum `daisy::Switch::Polarity`

Specifies whether the pressed is HIGH or LOW.

##### Enumerator

|                                |   |
|--------------------------------|---|
| <code>POLARITY_NORMAL</code>   | & |
| <code>POLARITY_INVERTED</code> | & |

### 12.124.2.2 Pull

```
enum daisy::Switch::Pull
```

Specifies whether to use built-in Pull Up/Down resistors to hold button at a given state when not engaged.

Enumerator

|           |   |
|-----------|---|
| PULL_UP   | & |
| PULL_DOWN | & |
| PULL_NONE | & |

### 12.124.2.3 Type

```
enum daisy::Switch::Type
```

Specifies the expected behavior of the switch

Enumerator

|                |   |
|----------------|---|
| TYPE_TOGGLE    | & |
| TYPE_MOMENTARY | & |

## 12.124.3 Member Function Documentation

### 12.124.3.1 Debounce()

```
void daisy::Switch::Debounce ()
```

Called at update\_rate to debounce and handle timing for the switch. In order for events not to be missed, its important that the Edge/Pressed checks be made at the same rate as the debounce function is being called.

### 12.124.3.2 FallingEdge()

```
bool daisy::Switch::FallingEdge () const [inline]
```

Returns

true if the button was just released

**12.124.3.3 Init() [1/2]**

```
void daisy::Switch::Init (
 dsy_gpio_pin pin,
 float update_rate,
 Type t,
 Polarity pol,
 Pull pu)
```

Initializes the switch object with a given port/pin combo.

**Parameters**

|                    |                                                                   |
|--------------------|-------------------------------------------------------------------|
| <i>pin</i>         | port/pin object to tell the switch which hardware pin to use.     |
| <i>update_rate</i> | Does nothing. Backwards compatibility until next breaking update. |
| <i>t</i>           | switch type – Default: TYPE_MOMENTARY                             |
| <i>pol</i>         | switch polarity – Default: POLARITY_INVERTED                      |
| <i>pu</i>          | switch pull up/down – Default: PULL_UP                            |

**12.124.3.4 Init() [2/2]**

```
void daisy::Switch::Init (
 dsy_gpio_pin pin,
 float update_rate = 0.f)
```

Simplified Init.

**Parameters**

|                    |                                                               |
|--------------------|---------------------------------------------------------------|
| <i>pin</i>         | port/pin object to tell the switch which hardware pin to use. |
| <i>update_rate</i> | Left for backwards compatibility until next breaking change.  |

**12.124.3.5 Pressed()**

```
bool daisy::Switch::Pressed () const [inline]
```

**Returns**

true if the button is held down (or if the toggle is on)

### 12.124.3.6 RawState()

```
bool daisy::Switch::RawState () [inline]
```

#### Returns

true if the button is held down, without debouncing

### 12.124.3.7 RisingEdge()

```
bool daisy::Switch::RisingEdge () const [inline]
```

#### Returns

true if a button was just pressed.

### 12.124.3.8 SetUpdateRate()

```
void daisy::Switch::SetUpdateRate (float update_rate) [inline]
```

Left for backwards compatibility until next breaking change

#### Parameters

|                    |                     |
|--------------------|---------------------|
| <i>update_rate</i> | Doesn't do anything |
|--------------------|---------------------|

### 12.124.3.9 TimeHeldMs()

```
float daisy::Switch::TimeHeldMs () const [inline]
```

#### Returns

the time in milliseconds that the button has been held (or toggle has been on)

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

- src/hid/switch.h

## 12.125 daisy::Switch3 Class Reference

### Public Types

- enum {
 **POS\_CENTER** = 0 , **POS\_LEFT** = 1 , **POS\_UP** = 1 , **POS\_RIGHT** = 2 ,
 **POS\_DOWN** = 2 }

### Public Member Functions

- void **Init** ([dsy\\_gpio\\_pin](#) pina, [dsy\\_gpio\\_pin](#) pinb)
- int **Read** ()

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

- src/hid/switch3.h

## 12.126 daisy::System Class Reference

### Classes

- struct [Config](#)

### Public Types

- enum [MemoryRegion](#) {
 **INTERNAL\_FLASH** = 0 , **ITCMRAM** , **DTCMRAM** , **SRAM\_D1** ,
 **SRAM\_D2** , **SRAM\_D3** , **SDRAM** , **QSPI** ,
 **INVALID\_ADDRESS** }

### Public Member Functions

- void **Init** ()
- void **Init** (const [Config](#) &config)
- void **Delinit** ()
- void **JumpToQspi** ()
- const [Config](#) & **GetConfig** () const

### Static Public Member Functions

- static uint32\_t **GetNow** ()
- static uint32\_t **GetUs** ()
- static uint32\_t **GetTick** ()
- static void **Delay** (uint32\_t delay\_ms)
- static void **DelayUs** (uint32\_t delay\_us)
- static void **DelayTicks** (uint32\_t delay\_ticks)
- static void **ResetToBootloader** ()
- static uint32\_t **GetTickFreq** ()
- static uint32\_t **GetSysClkFreq** ()
- static uint32\_t **GetHClkFreq** ()
- static uint32\_t **GetPClk1Freq** ()
- static uint32\_t **GetPClk2Freq** ()
- static [MemoryRegion](#) **GetProgramMemoryRegion** ()
- static [MemoryRegion](#) **GetMemoryRegion** (uint32\_t address)

## Static Public Attributes

- static constexpr uint32\_t `kQspiBootloaderOffset` = 0x40000U

### 12.126.1 Member Enumeration Documentation

#### 12.126.1.1 MemoryRegion

```
enum daisy::System::MemoryRegion
```

Describes the different regions of memory available to the Daisy

### 12.126.2 Member Function Documentation

#### 12.126.2.1 DeInit()

```
void daisy::System::DeInit ()
```

Deinitializer Deinitializes all modules and peripherals set up with `Init`.

#### 12.126.2.2 Delay()

```
static void daisy::System::Delay (
 uint32_t delay_ms) [static]
```

Blocking Delay that uses the SysTick (1ms callback) to wait.

##### Parameters

|                       |                     |
|-----------------------|---------------------|
| <code>delay_ms</code> | Time to delay in ms |
|-----------------------|---------------------|

#### 12.126.2.3 DelayTicks()

```
static void daisy::System::DelayTicks (
 uint32_t delay_ticks) [static]
```

Blocking Delay using internal timer to wait

**Parameters**

|                          |                                |
|--------------------------|--------------------------------|
| <code>delay_ticks</code> | Time to ddelay in microseconds |
|--------------------------|--------------------------------|

**12.126.2.4 DelayUs()**

```
static void daisy::System::DelayUs (
 uint32_t delay_us) [static]
```

Blocking Delay using internal timer to wait

**Parameters**

|                       |                                |
|-----------------------|--------------------------------|
| <code>delay_us</code> | Time to ddelay in microseconds |
|-----------------------|--------------------------------|

**12.126.2.5 GetConfig()**

```
const Config& daisy::System::GetConfig () const [inline]
```

Returns a const reference to the Systems Configuration struct.

**12.126.2.6 GetHClkFreq()**

```
static uint32_t daisy::System::GetHClkFreq () [static]
```

Returns the frequency of the HCLK (AHB) clock. This is derived from the [System](#) clock, and used to clock the CPU, memory, and peripherals mapped on the AHB, and APB Bus.

**12.126.2.7 GetMemoryRegion()**

```
static MemoryRegion daisy::System::GetMemoryRegion (
 uint32_t address) [static]
```

Returns an enum representing the the memory region that the given address belongs to.

**Parameters**

|                      |                           |
|----------------------|---------------------------|
| <code>address</code> | The address to be checked |
|----------------------|---------------------------|

### 12.126.2.8 GetNow()

```
static uint32_t daisy::System::GetNow () [static]
```

#### Returns

a uint32\_t value of milliseconds since the SysTick started

### 12.126.2.9 GetPCLK1Freq()

```
static uint32_t daisy::System::GetPCLK1Freq () [static]
```

Returns the frequency of the PCLK1 (APB1) clock This is used to clock various peripherals, and timers.

It's important to note that many timers run on a clock twice as fast as the peripheral clock for the timer.

### 12.126.2.10 GetPCLK2Freq()

```
static uint32_t daisy::System::GetPCLK2Freq () [static]
```

Returns the frequency of the PCLK2 (APB2) clock This is used to clock various peripherals, and timers.

It's important to note that many timers run on a clock twice as fast as the peripheral clock for the timer.

### 12.126.2.11 GetProgramMemoryRegion()

```
static MemoryRegion daisy::System::GetProgramMemoryRegion () [static]
```

Returns an enum representing the current (primary) memory space used for executing the program.

### 12.126.2.12 GetSysClkFreq()

```
static uint32_t daisy::System::GetSysClkFreq () [static]
```

Returns the Frequency of the system clock in Hz This is the primary system clock that is used to generate AXI Peripheral, APB, and AHB clocks.

### 12.126.2.13 GetTick()

```
static uint32_t daisy::System::GetTick () [static]
```

#### Returns

a uint32\_t of ticks at (PCLK1 \* 2)Hz Useful for measuring the number of CPU ticks something is taking.

**12.126.2.14 GetTickFreq()**

```
static uint32_t daisy::System::GetTickFreq () [static]
```

Returns the tick rate in Hz with which [GetTick\(\)](#) is incremented.

**12.126.2.15 GetUs()**

```
static uint32_t daisy::System::GetUs () [static]
```

Returns

a uint32\_t of microseconds within the internal timer.

**12.126.2.16 Init() [1/2]**

```
void daisy::System::Init ()
```

Default Initializer with no input will create an internal config, and set everything to Defaults

**12.126.2.17 Init() [2/2]**

```
void daisy::System::Init (const Config & config)
```

Configurable Initializer Initializes clock tree, DMA initializaiton and any necessary global inits.

**12.126.2.18 JumpToQspi()**

```
void daisy::System::JumpToQspi ()
```

Jumps to the first address of the external flash chip (0x90000000) If there is no code there, the chip will likely fall through to the while() loop TODO: Documentation/Loader for using external flash coming soon.

**12.126.2.19 ResetToBootloader()**

```
static void daisy::System::ResetToBootloader () [static]
```

Triggers a reset of the seed and starts in bootloarder mode to allow firmware update.

**12.126.3 Member Data Documentation**

### 12.126.3.1 kQspiBootloaderOffset

```
constexpr uint32_t daisy::System::kQspiBootloaderOffset = 0x40000U [static], [constexpr]
```

This constant indicates the Daisy bootloader's offset from the beginning of QSPI's address space. Data written within the first 256K will remain untouched by the Daisy bootloader.

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

- src/sys/system.h

## 12.127 daisy::SystemExclusiveEvent Struct Reference

```
#include <MidiEvent.h>
```

### Public Attributes

- int **length**
- uint8\_t **data** [SYSEX\_BUFFER\_LEN]

### 12.127.1 Detailed Description

Struct containing sysex data. Can be made from [MidiEvent](#)

### 12.127.2 Member Data Documentation

#### 12.127.2.1 data

```
uint8_t daisy::SystemExclusiveEvent::data[SYSEX_BUFFER_LEN]
```

&

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

- src/hid/MidiEvent.h

## 12.128 daisy::TimerHandle Class Reference

```
#include <tim.h>
```

## Classes

- struct [Config](#)

## Public Types

- enum class [Result](#) { [OK](#) , [ERR](#) }

## Public Member Functions

- [TimerHandle](#) (const [TimerHandle](#) &other)=default
- [TimerHandle](#) & [operator=](#) (const [TimerHandle](#) &other)=default
- [Result Init](#) (const [Config](#) &config)
- [Result DelInit](#) ()
- const [Config](#) & [GetConfig](#) () const
- [Result SetPeriod](#) (uint32\_t ticks)
- [Result SetPrescaler](#) (uint32\_t val)
- [Result Start](#) ()
- [Result Stop](#) ()
- uint32\_t [GetFreq](#) ()
- uint32\_t [GetTick](#) ()
- uint32\_t [GetMs](#) ()
- uint32\_t [GetUs](#) ()
- void [DelayTick](#) (uint32\_t del)
- void [DelayMs](#) (uint32\_t del)
- void [DelayUs](#) (uint32\_t del)

### 12.128.1 Detailed Description

Hardare timer peripheral support.

Supports general-function TIM peripherals:

- TIM2, TIM3, TIM4, TIM5

[DaisySeed](#), and many internal peripherals utilize TIM2 for timing/delay purposes. It is configured to be at the maximum frequency: typically 200MHz or 240MHz (boost) for measuring/delaying for very short periods.

The GetUs/GetMs functions are available for convenience (and backwards compatibility), but to avoid wrapping errors on math when doing time-delta calculations, using ticks is recommended. The data can be converted to the final time-base after getting the difference in ticks. (Using [GetFreq\(\)](#) can be used for these time-base calculations).

TODO:

- Fix issues with realtime getters, and wrapping of the timer(s).
  - This very noticeable with default settings for the 16-bit counters.
- Dispatch periodic callback(s)
- Other General purpose timers
- Non-internal clock sources
- Use of the four-tim channels per tim
  - PWM, etc.
  - InputCapture/OutputCompare, etc.
- HRTIM
- Advanced timers (TIM1/TIM8)

## 12.128.2 Member Enumeration Documentation

### 12.128.2.1 Result

```
enum daisy::TimerHandle::Result [strong]
```

Return values for TIM funcitons.

## 12.128.3 Member Function Documentation

### 12.128.3.1 DeInit()

```
Result daisy::TimerHandle::DeInit ()
```

Deinitializes the timer

### 12.128.3.2 DelayMs()

```
void daisy::TimerHandle::DelayMs (uint32_t del)
```

Stay within this function for del milliseconds

### 12.128.3.3 DelayTick()

```
void daisy::TimerHandle::DelayTick (uint32_t del)
```

Stay within this function for del ticks

### 12.128.3.4 DelayUs()

```
void daisy::TimerHandle::DelayUs (uint32_t del)
```

Stay within this function for del microseconds

### 12.128.3.5 GetConfig()

```
const Config& daisy::TimerHandle::GetConfig () const
```

Returns a const reference to the [Config](#) struct

**12.128.3.6 GetFreq()**

```
uint32_t daisy::TimerHandle::GetFreq ()
```

Returns the frequency of each tick of the timer in Hz

**12.128.3.7 GetMs()**

```
uint32_t daisy::TimerHandle::GetMs ()
```

Returns the ticks scaled as milliseconds

Use care when using for measurements and ensure that the TIM period can handle the maximum desired measurement.

**12.128.3.8 GetTick()**

```
uint32_t daisy::TimerHandle::GetTick ()
```

Returns the number of counter position. This increments according to [Config::CounterDir](#), and wraps around at the specified period (maxing out at  $2^{16}$  or  $2^{32}$  depending on the chosen TIM peripheral).

**12.128.3.9 GetUs()**

```
uint32_t daisy::TimerHandle::GetUs ()
```

Returns the ticks scaled as microseconds

Use care when using for measurements and ensure that the TIM period can handle the maximum desired measurement.

**12.128.3.10 Init()**

```
Result daisy::TimerHandle::Init (
 const Config & config)
```

Initializes the timer according to the configuration

**12.128.3.11 SetPeriod()**

```
Result daisy::TimerHandle::SetPeriod (
 uint32_t ticks)
```

Sets the period of the Timer. This is the number of ticks it takes before it wraps back around. For self-managed timing, this can be left at the default. (0xffff for 16-bit and 0xffffffff for 32-bit timers). This can be changed "on-the-fly"

### 12.128.3.12 SetPrescaler()

```
Result daisy::TimerHandle::SetPrescaler (
 uint32_t val)
```

Sets the Prescalar applied to the TIM peripheral. This can be any number up to 0xffff. This will adjust the rate of ticks: Calculated as APBN\_Freq / prescalar per tick where APBN is APB1 for Most general purpose timers, and APB2 for HRTIM, and the advanced timers. This can be changed "on-the-fly"

### 12.128.3.13 Start()

```
Result daisy::TimerHandle::Start ()
```

Starts the TIM peripheral specified by [Config](#)

### 12.128.3.14 Stop()

```
Result daisy::TimerHandle::Stop ()
```

Stops the TIM peripheral specified by [Config](#)

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

- src/per/tim.h

## 12.129 daisy::UartHandler Class Reference

```
#include <uart.h>
```

### Classes

- struct [Config](#)

### Public Types

- enum class [Result](#) { [OK](#) , [ERR](#) }

### Public Member Functions

- [UartHandler](#) (const [UartHandler](#) &other)=default
- [UartHandler](#) & [operator=](#) (const [UartHandler](#) &other)=default
- [Result](#) [Init](#) (const [Config](#) &config)
- const [Config](#) & [GetConfig](#) () const
- int [PollReceive](#) (uint8\_t \*buff, size\_t size, uint32\_t timeout)
- [Result](#) [StartRx](#) ()
- bool [RxActive](#) ()
- [Result](#) [FlushRx](#) ()
- [Result](#) [PollTx](#) (uint8\_t \*buff, size\_t size)
- uint8\_t [PopRx](#) ()
- size\_t [Readable](#) ()
- int [CheckError](#) ()

### 12.129.1 Detailed Description

Uart Peripheral

Author

shensley

Date

March 2020

### 12.129.2 Member Enumeration Documentation

#### 12.129.2.1 Result

```
enum daisy::UartHandler::Result [strong]
```

Return values for Uart functions.

Enumerator

|     |   |
|-----|---|
| OK  | & |
| ERR | & |

### 12.129.3 Member Function Documentation

#### 12.129.3.1 CheckError()

```
int daisy::UartHandler::CheckError ()
```

Returns

the result of HAL\_UART\_GetError() to the user.

#### 12.129.3.2 FlushRx()

```
Result daisy::UartHandler::FlushRx ()
```

Flushes the Receive Queue

Returns

OK or ERROR

### 12.129.3.3 GetConfig()

```
const Config& daisy::UartHandler::GetConfig () const
```

Returns the current config.

### 12.129.3.4 Init()

```
Result daisy::UartHandler::Init (
 const Config & config)
```

Initializes the UART Peripheral

### 12.129.3.5 PollReceive()

```
int daisy::UartHandler::PollReceive (
 uint8_t * buff,
 size_t size,
 uint32_t timeout)
```

Reads the amount of bytes in blocking mode with a 10ms timeout.

#### Parameters

|                |                                 |
|----------------|---------------------------------|
| <i>*buff</i>   | Buffer to read to               |
| <i>size</i>    | Buff size                       |
| <i>timeout</i> | How long to timeout for (10ms?) |

#### Returns

Data received

### 12.129.3.6 PollTx()

```
Result daisy::UartHandler::PollTx (
 uint8_t * buff,
 size_t size)
```

Sends an amount of data in blocking mode.

#### Parameters

|              |                        |
|--------------|------------------------|
| <i>*buff</i> | Buffer of data to send |
| <i>size</i>  | Buffer size            |

**Returns**

OK or ERROR

**12.129.3.7 PopRx()**

```
uint8_t daisy::UartHandler::PopRx ()
```

Pops the oldest byte from the [FIFO](#).

**Returns**

Popped byte

**12.129.3.8 Readable()**

```
size_t daisy::UartHandler::Readable ()
```

Checks if there are any unread bytes in the [FIFO](#)

**Returns**

1 or 0 ??

**12.129.3.9 RxActive()**

```
bool daisy::UartHandler::RxActive ()
```

**Returns**

whether Rx DMA is listening or not.

**12.129.3.10 StartRx()**

```
Result daisy::UartHandler::StartRx ()
```

Starts a DMA Receive callback to fill a buffer of specified size. Data is populated into a [FIFO](#) queue, and can be queried with the functions below. Size of the buffer is internally fixed to 256. Variable message lengths are transferred to the [FIFO](#) queue anytime there is 1 byte-period without incoming data

**Returns**

OK or ERROR

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

- src/per/uart.h

## 12.130 daisy::UI Class Reference

### Classes

- struct [SpecialControlIds](#)

### Public Member Functions

- void [Init](#) ([UiEventQueue](#) &inputQueue, const [SpecialControlIds](#) &specialControlIds, std::initializer\_list<[UiCanvasDescriptor](#)> canvases, uint16\_t primaryOneBitGraphicsDisplayId=[invalidCanvasId](#))
- void [Process](#) ()
- void [Mute](#) (bool shouldBeMuted, bool queueEvents=false)
- void [OpenPage](#) ([UiPage](#) &page)
- void [ClosePage](#) ([UiPage](#) &page)
- uint16\_t [GetPrimaryOneBitGraphicsDisplayId](#) () const
- [SpecialControlIds](#) [GetSpecialControlIds](#) () const

### Static Public Attributes

- static constexpr uint16\_t [invalidCanvasId](#) = uint16\_t(-1)

#### 12.130.1 Member Function Documentation

##### 12.130.1.1 [ClosePage\(\)](#)

```
void daisy::UI::ClosePage (
 UiPage & page)
```

Called to close a page.

##### 12.130.1.2 [GetPrimaryOneBitGraphicsDisplayId\(\)](#)

```
uint16_t daisy::UI::GetPrimaryOneBitGraphicsDisplayId () const [inline]
```

If this [UI](#) has a canvas that uses a [OneBitGraphicsDisplay](#) AND this canvas should be used as the main display for menus, etc. then this function returns the canvas ID of this display. If no such canvas exists, this function returns [UI::invalidCanvasId](#).

##### 12.130.1.3 [GetSpecialControlIds\(\)](#)

```
SpecialControlIds daisy::UI::GetSpecialControlIds () const [inline]
```

Returns the button IDs, encoder IDs and pot IDs used for special functions.

##### 12.130.1.4 [Init\(\)](#)

```
void daisy::UI::Init (
 UiEventQueue & inputQueue,
 const SpecialControlIds & specialControlIds,
 std::initializer_list<UiCanvasDescriptor> canvases,
 uint16_t primaryOneBitGraphicsDisplayId = invalidCanvasId)
```

Initializes the [UI](#).

**Parameters**

|                                       |                                                                                                                                                                                                              |
|---------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>inputQueue</i>                     | The <a href="#">UiEventQueue</a> to read user input events from.                                                                                                                                             |
| <i>specialControlIds</i>              | Information about the control IDs used for special buttons/encoders/pots.                                                                                                                                    |
| <i>canvases</i>                       | A list of <a href="#">UiCanvasDescriptor</a> that define which canvases to use.                                                                                                                              |
| <i>primaryOneBitGraphicsDisplayId</i> | The ID of a <a href="#">OneBitGraphicsDisplay</a> canvas that should be used as the main display. Menus will draw to this canvas. If no such display is available, use <a href="#">Ui::invalidCanvasId</a> . |

**12.130.1.5 Mute()**

```
void daisy::UI::Mute (
 bool shouldBeMuted,
 bool queueEvents = false)
```

Call this to temporarily disable processing of user input, e.g. while a project is loading. If queueEvents==true, all user input that happens while muted will be queued up and processed when the mute state is removed. If queueEvents==false, all user input that happens while muted will be discarded.

**12.130.1.6 OpenPage()**

```
void daisy::UI::OpenPage (
 UiPage & page)
```

Adds a new [UiPage](#) on the top of the stack of [UI](#) pages without taking ownership of the object. The new page is set to be visible. The page must stay alive until it was removed from the [UI](#). It's best to have each page statically allocated as a global variable.

**12.130.1.7 Process()**

```
void daisy::UI::Process ()
```

Call this regularly to allow processing user input, redraw canvases and do other "housekeeping" work. This is best done from a low priority context, ideally from your `main()` idle loop.

**12.130.2 Member Data Documentation****12.130.2.1 invalidCanvasId**

```
constexpr uint16_t daisy::UI::invalidCanvasId = uint16_t(-1) [static], [constexpr]
```

Use this to denote a nonexistent / invalid canvas ID

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

- `src/ui/UI.h`

## 12.131 daisy::UiCanvasDescriptor Struct Reference

### Public Types

- using `ClearFuncPtr` = `void(*)(const UiCanvasDescriptor &canvasToClear)`
- using `FlushFuncPtr` = `void(*)(const UiCanvasDescriptor &canvasToFlush)`

### Public Attributes

- `uint8_t id_`
- `void * handle_`
- `uint32_t updateRateMs_`
- `uint32_t screenSaverTimeOut = 0`
- `bool screenSaverOn = false`
- `ClearFuncPtr clearFunction_`
- `FlushFuncPtr flushFunction_`

#### 12.131.1 Member Typedef Documentation

##### 12.131.1.1 ClearFuncPtr

```
using daisy::UiCanvasDescriptor::ClearFuncPtr = void (*) (const UiCanvasDescriptor& canvasToClear)
```

A function to clear the display before the UIPages are drawn.

##### 12.131.1.2 FlushFuncPtr

```
using daisy::UiCanvasDescriptor::FlushFuncPtr = void (*) (const UiCanvasDescriptor& canvasToFlush)
```

A function to call when all UIPages have finished the drawing procedure and the results can be flushed out to the device.

#### 12.131.2 Member Data Documentation

##### 12.131.2.1 handle\_

```
void* daisy::UiCanvasDescriptor::handle_
```

A pointer to some object that allows to draw to the canvas. In your UI pages, you will use the `id_` to identify which canvas this is, and then cast this pointer to whatever object it represents, e.g. `OledDisplay`.

### 12.131.2.2 id\_

```
uint8_t daisy::UiCanvasDescriptor::id_
```

An id number to tell apart various types of canvases that are used concurrently in your system.

### 12.131.2.3 screenSaverTimeOut

```
uint32_t daisy::UiCanvasDescriptor::screenSaverTimeOut = 0
```

The desired timeout in ms before a display will shut off. This defaults to 0, which will keep the display on all the time. Nonzero values are useful for displays that can suffer from burn-in, such as OLEDs.

### 12.131.2.4 updateRateMs\_

```
uint32_t daisy::UiCanvasDescriptor::updateRateMs_
```

The desired update rate in ms

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

- src/ui/UI.h

## 12.132 daisy::UiEventQueue Class Reference

### Public Member Functions

- struct attribute ((packed)) Event
- void [AddButtonPressed](#) (uint16\_t buttonID, uint16\_t numSuccessivePresses, bool isRetriggering=false)
- void [AddButtonReleased](#) (uint16\_t buttonID)
- void [AddEncoderTurned](#) (uint16\_t encoderID, int16\_t increments, uint16\_t stepsPerRev)
- void [AddEncoderActivityChanged](#) (uint16\_t encoderId, bool isActive)
- void [AddPotMoved](#) (uint16\_t potId, float newPosition)
- void [AddPotActivityChanged](#) (uint16\_t potId, bool isActive)
- Event [GetAndRemoveNextEvent](#) ()
- bool [IsQueueEmpty](#) ()

### Static Public Attributes

- static constexpr uint16\_t [invalidButtonId](#) = UINT16\_MAX
- static constexpr uint16\_t [invalidEncoderId](#) = UINT16\_MAX
- static constexpr uint16\_t [invalidPotId](#) = UINT16\_MAX

### 12.132.1 Member Function Documentation

### 12.132.1.1 `__attribute__()`

```
struct daisy::UiEventQueue::__attribute__ (
 (packed)) [inline]
```

An event in the queue The type of event

An invalid event. Returned to indicate that no events are left in the queue.

A button was pressed.

A button was released.

An encoder was turned.

The user has started or stopped turning an encoder.

A potentiometer was moved.

The user has started or stopped moving a potentiometer.

Used to indicate if a control is currently being used.

The control is not in use at the moment.

The control is actively used at the moment.

The type of event that this Event object represents.

The unique ID of the button that was pressed.

The number of successive button presses (e.g. double click).

True if the event was generated because a button was retriggered automatically while being held down.

The unique ID of the button that was released.

The unique ID of the encoder that was turned.

The number of increments detected.

The total number of increments per revolution.

The unique ID of the encoder that is affected.

The new activity type.

The unique ID of the pot that was moved.

The new position of the pot.

The unique ID of the pot that is affected.

The new activity type.

### 12.132.1.2 AddButtonPressed()

```
void daisy::UiEventQueue::AddButtonPressed (
 uint16_t buttonID,
 uint16_t numSuccessivePresses,
 bool isRetriggering = false) [inline]
```

Adds a Event::EventType::buttonPressed event to the queue.

### 12.132.1.3 AddButtonReleased()

```
void daisy::UiEventQueue::AddButtonReleased (
 uint16_t buttonID) [inline]
```

Adds a Event::EventType::buttonReleased event to the queue.

### 12.132.1.4 AddEncoderActivityChanged()

```
void daisy::UiEventQueue::AddEncoderActivityChanged (
 uint16_t encoderID,
 bool isActive) [inline]
```

Adds a Event::EventType::encoderActivityChanged event to the queue.

### 12.132.1.5 AddEncoderTurned()

```
void daisy::UiEventQueue::AddEncoderTurned (
 uint16_t encoderID,
 int16_t increments,
 uint16_t stepsPerRev) [inline]
```

Adds a Event::EventType::encoderTurned event to the queue.

### 12.132.1.6 AddPotActivityChanged()

```
void daisy::UiEventQueue::AddPotActivityChanged (
 uint16_t potId,
 bool isActive) [inline]
```

Adds a Event::EventType::potActivityChanged event to the queue.

### 12.132.1.7 AddPotMoved()

```
void daisy::UiEventQueue::AddPotMoved (
 uint16_t potId,
 float newPosition) [inline]
```

Adds a Event::EventType::potMoved event to the queue.

### 12.132.1.8 GetAndRemoveNextEvent()

```
Event daisy::UiEventQueue::GetAndRemoveNextEvent () [inline]
```

Removes and returns an event from the queue.

### 12.132.1.9 IsQueueEmpty()

```
bool daisy::UiEventQueue::IsQueueEmpty () [inline]
```

Returns true, if the queue is empty.

## 12.132.2 Member Data Documentation

### 12.132.2.1 invalidButtonId

```
constexpr uint16_t daisy::UiEventQueue::invalidButtonId = UINT16_MAX [static], [constexpr]
```

A button ID used to indicate an invalid or non existing button.

### 12.132.2.2 invalidEncoderId

```
constexpr uint16_t daisy::UiEventQueue::invalidEncoderId = UINT16_MAX [static], [constexpr]
```

An encoder ID used to indicate an invalid or non existing encoder.

### 12.132.2.3 invalidPotId

```
constexpr uint16_t daisy::UiEventQueue::invalidPotId = UINT16_MAX [static], [constexpr]
```

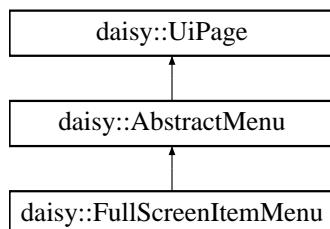
A potentiometer ID used to indicate an invalid or non existing potentiometer.

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

- src/ui/UiEventQueue.h

## 12.133 daisy::UiPage Class Reference

Inheritance diagram for daisy::UiPage:



## Public Member Functions

- virtual bool [IsOpaque](#) (const [UiCanvasDescriptor](#) &display)
- bool [IsActive](#) ()
- void [Close](#) ()
- virtual bool [OnOkayButton](#) (uint8\_t numberOfPresses, bool isRetriggering)
- virtual bool [OnCancelButton](#) (uint8\_t numberOfPresses, bool isRetriggering)
- virtual bool [OnArrowButton](#) ([ArrowButtonType](#) arrowType, uint8\_t numberOfPresses, bool isRetriggering)
- virtual bool [OnFunctionButton](#) (uint8\_t numberOfPresses, bool isRetriggering)
- virtual bool [OnButton](#) (uint16\_t buttonID, uint8\_t numberOfPresses, bool isRetriggering)
- virtual bool [OnMenuEncoderTurned](#) (int16\_t turns, uint16\_t stepsPerRevolution)
- virtual bool [OnValueEncoderTurned](#) (int16\_t turns, uint16\_t stepsPerRevolution)
- virtual bool [OnEncoderTurned](#) (uint16\_t encoderID, int16\_t turns, uint16\_t stepsPerRevolution)
- virtual bool [OnMenuEncoderActivityChanged](#) (bool isCurrentlyActive)
- virtual bool [OnValueEncoderActivityChanged](#) (bool isCurrentlyActive)
- virtual bool [OnEncoderActivityChanged](#) (uint16\_t encoderID, bool isCurrentlyActive)
- virtual bool [OnValuePotMoved](#) (float newPosition)
- virtual bool [OnPotMoved](#) (uint16\_t potID, float newPosition)
- virtual bool [OnValuePotActivityChanged](#) (bool isCurrentlyActive)
- virtual bool [OnPotActivityChanged](#) (uint16\_t potID, bool isCurrentlyActive)
- virtual void [OnShow](#) ()
- virtual void [OnHide](#) ()
- virtual void [OnFocusGained](#) ()
- virtual void [OnFocusLost](#) ()
- virtual void [Draw](#) (const [UiCanvasDescriptor](#) &canvas)=0
- [UI \\* GetParentUI](#) ()
- const [UI \\* GetParentUI](#) () const

## Friends

- class [UI](#)

### 12.133.1 Member Function Documentation

#### 12.133.1.1 Close()

```
void daisy::UiPage::Close ()
```

Closes the current page. This calls the parent [UI](#) and asks it to Remove this page from the page stack.

#### 12.133.1.2 Draw()

```
virtual void daisy::UiPage::Draw (
 const UiCanvasDescriptor & canvas) [pure virtual]
```

Called to make the [UiPage](#) repaint everything on a canvas. Check the ID to determine which display this corresponds to. Cast the handle to the corresponding type and do your draw operations on it.

Implemented in [daisy::FullScreenItemMenu](#).

### 12.133.1.3 GetParentUI() [1/2]

```
UI* daisy::UiPage::GetParentUI () [inline]
```

Returns a reference to the parent [UI](#) object, or nullptr if not added to any [UI](#) at the moment.

### 12.133.1.4 GetParentUI() [2/2]

```
const UI* daisy::UiPage::GetParentUI () const [inline]
```

Returns a reference to the parent [UI](#) object, or nullptr if not added to any [UI](#) at the moment.

### 12.133.1.5 IsActive()

```
bool daisy::UiPage::IsActive () [inline]
```

Returns true if the page is currently active on a [UI](#) - it may not be visible, though.

### 12.133.1.6 IsOpaque()

```
virtual bool daisy::UiPage::IsOpaque (
 const UiCanvasDescriptor & display) [inline], [virtual]
```

Returns true, if the page fills the entire canvas. A canvas can be individual leds, text displays, alphanumeric displays, graphics displays, etc. The [UI](#) class will use this to determine if underlying pages must be drawn before this page.

### 12.133.1.7 OnArrowButton()

```
virtual bool daisy::UiPage::OnArrowButton (
 ArrowButtonType arrowType,
 uint8_t numberOfPresses,
 bool isRetriggering) [inline], [virtual]
```

Called when an arrow button is pressed or released.

#### Parameters

|                              |                                                                                                                                                                                                           |
|------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>arrowType</code>       | The arrow button affected.                                                                                                                                                                                |
| <code>numberOfPresses</code> | Holds the number of successive button presses. It will be 1 on the first call and increasing by 1 with each successive call. A button down event is signaled by <code>numberOfButtonPresses == 0</code> . |
| <code>isRetriggering</code>  | True if the button is auto-retriggering (due to being held down)                                                                                                                                          |

#### Returns

false, if you want the event to be passed on to the page below.

Reimplemented in [daisy::AbstractMenu](#).

**12.133.1.8 OnButton()**

```
virtual bool daisy::UiPage::OnButton (
 uint16_t buttonID,
 uint8_t numberOfPresses,
 bool isRetriggering) [inline], [virtual]
```

Called when any button is pressed or released that is not an arrow button, the function button or the okay / cancel buttons.

**Parameters**

|                        |                                                                                                                                                                                                    |
|------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>buttonID</i>        | The ID of the affected button.                                                                                                                                                                     |
| <i>numberOfPresses</i> | Holds the number of successive button presses. It will be 1 on the first call and increasing by 1 with each successive call. A button down event is signaled by <i>numberOfButtonPresses</i> == 0. |
| <i>isRetriggering</i>  | True if the button is auto-retriggering (due to being held down)                                                                                                                                   |

**Returns**

`false`, if you want the event to be passed on to the page below.

**12.133.1.9 OnCancelButton()**

```
virtual bool daisy::UiPage::OnCancelButton (
 uint8_t numberOfPresses,
 bool isRetriggering) [inline], [virtual]
```

Called when the cancel button is pressed or released.

**Parameters**

|                        |                                                                                                                                                                                                    |
|------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>numberOfPresses</i> | Holds the number of successive button presses. It will be 1 on the first call and increasing by 1 with each successive call. A button down event is signaled by <i>numberOfButtonPresses</i> == 0. |
| <i>isRetriggering</i>  | True if the button is auto-retriggering (due to being held down)                                                                                                                                   |

**Returns**

`false`, if you want the event to be passed on to the page below.

Reimplemented in [daisy::AbstractMenu](#).

**12.133.1.10 OnEncoderActivityChanged()**

```
virtual bool daisy::UiPage::OnEncoderActivityChanged (
 uint16_t encoderID,
 bool isCurrentlyActive) [inline], [virtual]
```

Called when the user starts or stops turning an encoder that is not the menu encoder or the value encoder.

**Parameters**

|                          |                                                 |
|--------------------------|-------------------------------------------------|
| <i>encoderID</i>         | The ID of the affected encoder.                 |
| <i>isCurrentlyActive</i> | True, if the user currently moves this encoder. |

**Returns**

false, if you want the event to be passed on to the page below.

**12.133.1.11 OnEncoderTurned()**

```
virtual bool daisy::UiPage::OnEncoderTurned (
 uint16_t encoderID,
 int16_t turns,
 uint16_t stepsPerRevolution) [inline], [virtual]
```

Called when an encoder is turned that is not the menu encoder or the value encoder.

**Parameters**

|                           |                                                                |
|---------------------------|----------------------------------------------------------------|
| <i>encoderID</i>          | The ID of the affected encoder.                                |
| <i>turns</i>              | The number of increments, positive is clockwise.               |
| <i>stepsPerRevolution</i> | The total number of increments per revolution on this encoder. |

**Returns**

false, if you want the event to be passed on to the page below.

**12.133.1.12 OnFocusGained()**

```
virtual void daisy::UiPage::OnFocusGained () [inline], [virtual]
```

Called when the page becomes the topmost page in the page stack.

**12.133.1.13 OnFocusLost()**

```
virtual void daisy::UiPage::OnFocusLost () [inline], [virtual]
```

Called when the page is no longer the topmost page in the page stack.

**12.133.1.14 OnFunctionButton()**

```
virtual bool daisy::UiPage::OnFunctionButton (
 uint8_t numberOfPresses,
 bool isRetriggering) [inline], [virtual]
```

Called when the function button is pressed or released.

**Parameters**

|                        |                                                                                                                                                                                                    |
|------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>numberOfPresses</i> | Holds the number of successive button presses. It will be 1 on the first call and increasing by 1 with each successive call. A button down event is signaled by <i>numberOfButtonPresses</i> == 0. |
| <i>isRetriggering</i>  | True if the button is auto-retriggering (due to being held down)                                                                                                                                   |

**Returns**

false, if you want the event to be passed on to the page below.

Reimplemented in [daisy::AbstractMenu](#).

**12.133.1.15 OnHide()**

```
virtual void daisy::UiPage::OnHide () [inline], [virtual]
```

Called when the page is removed from the [UI](#).

**12.133.1.16 OnMenuEncoderActivityChanged()**

```
virtual bool daisy::UiPage::OnMenuEncoderActivityChanged (
 bool isCurrentlyActive) [inline], [virtual]
```

Called when the user starts or stops turning the menu encoder.

**Parameters**

|                          |                                                 |
|--------------------------|-------------------------------------------------|
| <i>isCurrentlyActive</i> | True, if the user currently moves this encoder. |
|--------------------------|-------------------------------------------------|

**Returns**

false, if you want the event to be passed on to the page below.

**12.133.1.17 OnMenuEncoderTurned()**

```
virtual bool daisy::UiPage::OnMenuEncoderTurned (
 int16_t turns,
 uint16_t stepsPerRevolution) [inline], [virtual]
```

Called when the menu encoder is turned.

**Parameters**

|                           |                                                                |
|---------------------------|----------------------------------------------------------------|
| <i>turns</i>              | The number of increments, positive is clockwise.               |
| <i>stepsPerRevolution</i> | The total number of increments per revolution on this encoder. |

**Returns**

false, if you want the event to be passed on to the page below.

Reimplemented in [daisy::AbstractMenu](#).

**12.133.1.18 OnOkayButton()**

```
virtual bool daisy::UiPage::OnOkayButton (
 uint8_t numberOfPresses,
 bool isRetriggering) [inline], [virtual]
```

Called when the okay button is pressed or released.

**Parameters**

|                        |                                                                                                                                                                                                    |
|------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>numberOfPresses</i> | Holds the number of successive button presses. It will be 1 on the first call and increasing by 1 with each successive call. A button down event is signaled by <i>numberOfButtonPresses</i> == 0. |
| <i>isRetriggering</i>  | True if the button is auto-retriggering (due to being held down)                                                                                                                                   |

**Returns**

false, if you want the event to be passed on to the page below.

Reimplemented in [daisy::AbstractMenu](#).

**12.133.1.19 OnPotActivityChanged()**

```
virtual bool daisy::UiPage::OnPotActivityChanged (
 uint16_t potID,
 bool isCurrentlyActive) [inline], [virtual]
```

Called when the user starts or stops turning a potentiometer that's not the value potentiometer.

**Parameters**

|                          |                                                       |
|--------------------------|-------------------------------------------------------|
| <i>potID</i>             | The ID of the affected potentiometer.                 |
| <i>isCurrentlyActive</i> | True, if the user currently moves this potentiometer. |

**Returns**

false, if you want the event to be passed on to the page below.

**12.133.1.20 OnPotMoved()**

```
virtual bool daisy::UiPage::OnPotMoved (
 uint16_t potID,
 float newPosition) [inline], [virtual]
```

Called when a potentiometer is turned that's not the value potentiometer.

**Parameters**

|                    |                                       |
|--------------------|---------------------------------------|
| <i>potID</i>       | The ID of the affected potentiometer. |
| <i>newPosition</i> | The new position in the range 0 .. 1  |

**Returns**

false, if you want the event to be passed on to the page below.

**12.133.1.21 OnShow()**

```
virtual void daisy::UiPage::OnShow () [inline], [virtual]
```

Called when the page is added to the [UI](#).

Reimplemented in [daisy::AbstractMenu](#).

**12.133.1.22 OnValueEncoderActivityChanged()**

```
virtual bool daisy::UiPage::OnValueEncoderActivityChanged (
 bool isCurrentlyActive) [inline], [virtual]
```

Called when the user starts or stops turning the value encoder.

**Parameters**

|                          |                                                 |
|--------------------------|-------------------------------------------------|
| <i>isCurrentlyActive</i> | True, if the user currently moves this encoder. |
|--------------------------|-------------------------------------------------|

**Returns**

false, if you want the event to be passed on to the page below.

**12.133.1.23 OnValueEncoderTurned()**

```
virtual bool daisy::UiPage::OnValueEncoderTurned (
 int16_t turns,
 uint16_t stepsPerRevolution) [inline], [virtual]
```

Called when the menu encoder is turned.

**Parameters**

|                           |                                                                |
|---------------------------|----------------------------------------------------------------|
| <i>turns</i>              | The number of increments, positive is clockwise.               |
| <i>stepsPerRevolution</i> | The total number of increments per revolution on this encoder. |

**Returns**

false, if you want the event to be passed on to the page below.

Reimplemented in [daisy::AbstractMenu](#).

**12.133.1.24 OnValuePotActivityChanged()**

```
virtual bool daisy::UiPage::OnValuePotActivityChanged (
 bool isCurrentlyActive) [inline], [virtual]
```

Called when the user starts or stops turning the value potentiometer.

**Parameters**

|                          |                                                       |
|--------------------------|-------------------------------------------------------|
| <i>isCurrentlyActive</i> | True, if the user currently moves this potentiometer. |
|--------------------------|-------------------------------------------------------|

**Returns**

false, if you want the event to be passed on to the page below.

**12.133.1.25 OnValuePotMoved()**

```
virtual bool daisy::UiPage::OnValuePotMoved (
 float newPosition) [inline], [virtual]
```

Called when the value potentiometer is turned.

**Parameters**

|                    |                                      |
|--------------------|--------------------------------------|
| <i>newPosition</i> | The new position in the range 0 .. 1 |
|--------------------|--------------------------------------|

**Returns**

false, if you want the event to be passed on to the page below.

Reimplemented in [daisy::AbstractMenu](#).

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

- src/ui/UI.h

## 12.134 UsbHandle Class Reference

Interface for initializing and using the USB Peripherals on the daisy.

```
#include <usb.h>
```

### Public Types

- enum class [Result](#) { **OK** , **ERR** , **OK** , **ERR** }
- enum [UsbPeriph](#) {
 **FS\_INTERNAL** , **FS\_EXTERNAL** , **FS\_BOTH** , **FS\_INTERNAL** ,
 **FS\_EXTERNAL** , **FS\_BOTH** }
- enum class [Result](#) { **OK** , **ERR** , **OK** , **ERR** }
- enum [UsbPeriph](#) {
 **FS\_INTERNAL** , **FS\_EXTERNAL** , **FS\_BOTH** , **FS\_INTERNAL** ,
 **FS\_EXTERNAL** , **FS\_BOTH** }
- typedef void(\* [ReceiveCallback](#)) (uint8\_t \*buff, uint32\_t \*len)
- typedef void(\* [ReceiveCallback](#)) (uint8\_t \*buff, uint32\_t \*len)

### Public Member Functions

- void [Init](#) ([UsbPeriph](#) dev)
- void [DeInit](#) ([UsbPeriph](#) dev)
- [Result](#) [TransmitInternal](#) (uint8\_t \*buff, size\_t size)
- [Result](#) [TransmitExternal](#) (uint8\_t \*buff, size\_t size)
- void [SetReceiveCallback](#) ([ReceiveCallback](#) cb, [UsbPeriph](#) dev)
- void [Init](#) ([UsbPeriph](#) dev)
- void [DeInit](#) ([UsbPeriph](#) dev)
- [Result](#) [TransmitInternal](#) (uint8\_t \*buff, size\_t size)
- [Result](#) [TransmitExternal](#) (uint8\_t \*buff, size\_t size)
- void [SetReceiveCallback](#) ([ReceiveCallback](#) cb, [UsbPeriph](#) dev)

### 12.134.1 Detailed Description

Interface for initializing and using the USB Peripherals on the daisy.

Author

Stephen Hensley

Date

December 2019

### 12.134.2 Member Typedef Documentation

#### 12.134.2.1 ReceiveCallback [1/2]

```
typedef void(* UsbHandle::ReceiveCallback) (uint8_t *buff, uint32_t *len)
```

Function called upon reception of a buffer

#### 12.134.2.2 ReceiveCallback [2/2]

```
typedef void(* UsbHandle::ReceiveCallback) (uint8_t *buff, uint32_t *len)
```

Function called upon reception of a buffer

### 12.134.3 Member Enumeration Documentation

#### 12.134.3.1 Result [1/2]

```
enum UsbHandle::Result [strong]
```

Return values for USBHandle Functions

#### 12.134.3.2 Result [2/2]

```
enum UsbHandle::Result [strong]
```

Return values for USBHandle Functions

#### 12.134.3.3 UsbPeriph [1/2]

```
enum UsbHandle::UsbPeriph
```

Specified which of the two USB Peripherals to initialize.

## Enumerator

|             |                                                                              |
|-------------|------------------------------------------------------------------------------|
| FS_INTERNAL | Internal pin                                                                 |
| FS_EXTERNAL | FS External D+ pin is Pin 38 (GPIO32). FS External D- pin is Pin 37 (GPIO31) |
| FS_BOTH     | Both                                                                         |
| FS_INTERNAL | Internal pin                                                                 |
| FS_EXTERNAL | FS External D+ pin is Pin 38 (GPIO32). FS External D- pin is Pin 37 (GPIO31) |
| FS_BOTH     | Both                                                                         |

**12.134.3.4 UsbPeriph [2/2]**

```
enum UsbHandle::UsbPeriph
```

Specified which of the two USB Peripherals to initialize.

## Enumerator

|             |                                                                              |
|-------------|------------------------------------------------------------------------------|
| FS_INTERNAL | Internal pin                                                                 |
| FS_EXTERNAL | FS External D+ pin is Pin 38 (GPIO32). FS External D- pin is Pin 37 (GPIO31) |
| FS_BOTH     | Both                                                                         |
| FS_INTERNAL | Internal pin                                                                 |
| FS_EXTERNAL | FS External D+ pin is Pin 38 (GPIO32). FS External D- pin is Pin 37 (GPIO31) |
| FS_BOTH     | Both                                                                         |

**12.134.4 Member Function Documentation****12.134.4.1 DeInit() [1/2]**

```
void UsbHandle::DeInit (
 UsbPeriph dev)
```

Deinitializes the specified peripheral(s)

## Parameters

|            |                        |
|------------|------------------------|
| <i>dev</i> | Device to deinitialize |
|------------|------------------------|

**12.134.4.2 DeInit() [2/2]**

```
void UsbHandle::DeInit (
```

```
 UsbPeriph dev)
```

Deinitializes the specified peripheral(s)

**Parameters**

|            |                        |
|------------|------------------------|
| <i>dev</i> | Device to deinitialize |
|------------|------------------------|

#### 12.134.4.3 Init() [1/2]

```
void UsbHandle::Init (
```

```
 UsbPeriph dev)
```

Initializes the specified peripheral(s) as USB CDC Devices

**Parameters**

|            |                      |
|------------|----------------------|
| <i>dev</i> | Device to initialize |
|------------|----------------------|

#### 12.134.4.4 Init() [2/2]

```
void UsbHandle::Init (
```

```
 UsbPeriph dev)
```

Initializes the specified peripheral(s) as USB CDC Devices

**Parameters**

|            |                      |
|------------|----------------------|
| <i>dev</i> | Device to initialize |
|------------|----------------------|

#### 12.134.4.5 SetReceiveCallback() [1/2]

```
void UsbHandle::SetReceiveCallback (
```

```
 ReceiveCallback cb,
```

```
 UsbPeriph dev)
```

sets the callback to be called upon reception of new data

**Parameters**

|            |                               |
|------------|-------------------------------|
| <i>cb</i>  | Function to serve as callback |
| <i>dev</i> | Device to set callback for    |

#### 12.134.4.6 SetReceiveCallback() [2/2]

```
void UsbHandle::SetReceiveCallback (
 ReceiveCallback cb,
 UsbPeriph dev)
```

sets the callback to be called upon reception of new data

##### Parameters

|            |                               |
|------------|-------------------------------|
| <i>cb</i>  | Function to serve as callback |
| <i>dev</i> | Device to set callback for    |

#### 12.134.4.7 TransmitExternal() [1/2]

```
Result UsbHandle::TransmitExternal (
 uint8_t * buff,
 size_t size)
```

Transmits a buffer of 'size' bytes from a USB port connected to the external USB Pins of the daisy seed.

##### Parameters

|             |                    |
|-------------|--------------------|
| <i>buff</i> | Buffer to transmit |
| <i>size</i> | Buffer size        |

#### 12.134.4.8 TransmitExternal() [2/2]

```
Result UsbHandle::TransmitExternal (
 uint8_t * buff,
 size_t size)
```

Transmits a buffer of 'size' bytes from a USB port connected to the external USB Pins of the daisy seed.

##### Parameters

|             |                    |
|-------------|--------------------|
| <i>buff</i> | Buffer to transmit |
| <i>size</i> | Buffer size        |

#### 12.134.4.9 `TransmitInternal()` [1/2]

```
Result UsbHandle::TransmitInternal (
 uint8_t * buff,
 size_t size)
```

Transmits a buffer of '*size*' bytes from the on board USB FS port.

##### Parameters

|             |                    |
|-------------|--------------------|
| <i>buff</i> | Buffer to transmit |
| <i>size</i> | Buffer size        |

#### 12.134.4.10 `TransmitInternal()` [2/2]

```
Result UsbHandle::TransmitInternal (
 uint8_t * buff,
 size_t size)
```

Transmits a buffer of '*size*' bytes from the on board USB FS port.

##### Parameters

|             |                    |
|-------------|--------------------|
| <i>buff</i> | Buffer to transmit |
| <i>size</i> | Buffer size        |

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

- src/hid/usb.h

## 12.135 daisy::USBHostHandle Class Reference

Presents a USB Mass Storage Device host interface.

```
#include <usb_host.h>
```

### Classes

- struct [Config](#)

### Public Types

- enum [Result](#) { **OK** = 0 , **ERR** }

## Public Member Functions

- Result `Init (Config config)`
- Result `Deinit ()`
- void `Process ()`
- bool `GetReady ()`
- bool `GetPresent ()`
- `USBHostHandle (const USBHostHandle &other)=default`
- `USBHostHandle & operator= (const USBHostHandle &other)=default`

### 12.135.1 Detailed Description

Presents a USB Mass Storage Device host interface.

#### Author

Gabriel Ball

#### Date

September 16, 2021

### 12.135.2 Member Function Documentation

#### 12.135.2.1 Deinit()

```
Result daisy::USBHostHandle::Deinit ()
```

Deinitializes MSD-related peripherals

#### 12.135.2.2 GetPresent()

```
bool daisy::USBHostHandle::GetPresent ()
```

Run after the first `Process` call to detect if a device is present

#### 12.135.2.3 GetReady()

```
bool daisy::USBHostHandle::GetReady ()
```

Returns true if a Mass Storage Device is connected and ready for communication

#### 12.135.2.4 Init()

```
Result daisy::USBHostHandle::Init (Config config)
```

Initializes the USB drivers and starts timeout.

**Parameters**

|                     |                                         |
|---------------------|-----------------------------------------|
| <code>config</code> | Configuration struct for initialization |
|---------------------|-----------------------------------------|

**12.135.2.5 Process()**

```
void daisy::USBHostHandle::Process ()
```

Manages usb host functionality

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

- `src/hid/usb_host.h`

## 12.136 daisy::VoctCalibration Class Reference

Helper class for calibrating an input to 1V/oct response.

```
#include <VoctCalibration.h>
```

### Public Member Functions

- bool `Record` (float val1V, float val3V)
- bool `GetData` (float &scale, float &offset)
- void `SetData` (float scale, float offset)
- float `ProcessInput` (const float inval)

### 12.136.1 Detailed Description

Helper class for calibrating an input to 1V/oct response.

**Author**

shensley

This provides a scale and offset value for converting incoming CV into MIDI note numbers accurately for musical pitch tracking.

To use, record both the 1V and 3V values using the specified functions once calibration is complete you can use the `GetData` function to retrieve the calibration values.

This can also be used for 100mV/Semitone calibration as used by Buchla synthesizer modules. To calibrate for this standard. You would send 1.2V, and 3.6V

### 12.136.2 Member Function Documentation

#### 12.136.2.1 GetData()

```
bool daisy::VoctCalibration::GetData (
 float & scale,
 float & offset) [inline]
```

Get the scale and offset data from the calibration

## Return values

|                |                                         |
|----------------|-----------------------------------------|
| <i>returns</i> | true if calibration has been performed. |
|----------------|-----------------------------------------|

**12.136.2.2 ProcessInput()**

```
float daisy::VoctCalibration::ProcessInput (
 const float inval) [inline]
```

Process a value through the calibrated data to get a MIDI Note number

**12.136.2.3 Record()**

```
bool daisy::VoctCalibration::Record (
 float val1V,
 float val3V) [inline]
```

Uses the values retrieved for 1V and 3V in order to compute a scale and offset value that can be used to convert a CV input signal to a calibrated 1V/oct range.

## Parameters

|              |                         |
|--------------|-------------------------|
| <i>val1V</i> | ADC reading for 1 volt  |
| <i>val3V</i> | ADC reading for 3 volts |

## Return values

|                |                                                             |
|----------------|-------------------------------------------------------------|
| <i>returns</i> | true if the calibraiton is successful - this is always true |
|----------------|-------------------------------------------------------------|

**Todo** Add some sort of range validation. Originally we had a check for a valid range on the input, but given that the input circuit or the [AnalogControl](#) configuration can have a drastic effect on input, that could cause unintentional failure to calibrate, it was removed.

**12.136.2.4 SetData()**

```
void daisy::VoctCalibration::SetData (
 float scale,
 float offset) [inline]
```

Manually set the calibration data and mark internally as "calibrated" This is used to reset the data after a power cycle without having to redo the calibration procedure.

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

- src/util/VoctCalibration.h

## 12.137 daisy::WAV\_FormatTypeDef Struct Reference

```
#include <wav_format.h>
```

### Public Attributes

- `uint32_t ChunkId`
- `uint32_t FileSize`
- `uint32_t FileFormat`
- `uint32_t SubChunk1ID`
- `uint32_t SubChunk1Size`
- `uint16_t AudioFormat`
- `uint16_t NbrChannels`
- `uint32_t SampleRate`
- `uint32_t ByteRate`
- `uint16_t BlockAlign`
- `uint16_t BitPerSample`
- `uint32_t SubChunk2ID`
- `uint32_t SubCHunk2Size`

### 12.137.1 Detailed Description

Helper struct for handling the WAV file format

### 12.137.2 Member Data Documentation

#### 12.137.2.1 AudioFormat

```
uint16_t daisy::WAV_FormatTypeDef::AudioFormat
&
```

#### 12.137.2.2 BitPerSample

```
uint16_t daisy::WAV_FormatTypeDef::BitPerSample
&
```

#### 12.137.2.3 BlockAlign

```
uint16_t daisy::WAV_FormatTypeDef::BlockAlign
&
```

**12.137.2.4 ByteRate**

```
uint32_t daisy::WAV_FormatTypeDef::ByteRate
&
```

**12.137.2.5 ChunkId**

```
uint32_t daisy::WAV_FormatTypeDef::ChunkId
&
```

**12.137.2.6 FileFormat**

```
uint32_t daisy::WAV_FormatTypeDef::FileFormat
&
```

**12.137.2.7 FileSize**

```
uint32_t daisy::WAV_FormatTypeDef::FileSize
&
```

**12.137.2.8 NbrChannels**

```
uint16_t daisy::WAV_FormatTypeDef::NbrChannels
&
```

**12.137.2.9 SampleRate**

```
uint32_t daisy::WAV_FormatTypeDef::SampleRate
&
```

**12.137.2.10 SubChunk1ID**

```
uint32_t daisy::WAV_FormatTypeDef::SubChunk1ID
&
```

**12.137.2.11 SubChunk1Size**

```
uint32_t daisy::WAV_FormatTypeDef::SubChunk1Size
&
```

### 12.137.2.12 SubChunk2ID

```
uint32_t daisy::WAV_FormatTypeDef::SubChunk2ID
&
```

### 12.137.2.13 SubCHunk2Size

```
uint32_t daisy::WAV_FormatTypeDef::SubCHunk2Size
&
```

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

- src/util/wav\_format.h

## 12.138 daisy::WaveTableLoader Class Reference

```
#include <WaveTableLoader.h>
```

### Public Types

- enum class **Result** { **OK** , **ERR\_TABLE\_INFO\_OVERFLOW** , **ERR\_FILE\_READ** , **ERR\_GENERIC** }

### Public Member Functions

- void **Init** (float \*mem, size\_t mem\_size)
- Result **SetWaveTableInfo** (size\_t samps, size\_t count)
- Result **Import** (const char \*filename)
- float \* **GetTable** (size\_t idx)

### 12.138.1 Detailed Description

Loads a bank of wavetables into memory. Pointers to the start of each waveform will be provided, but the user can do whatever they want with the data once it's imported.

A internal 4kB workspace is used for reading from the file, and converting to the correct memory location.

### 12.138.2 Member Function Documentation

### 12.138.2.1 GetTable()

```
float* daisy::WaveTableLoader::GetTable (
 size_t idx)
```

Returns pointer to specific table start or nullptr if invalid idx

### 12.138.2.2 Import()

```
Result daisy::WaveTableLoader::Import (
 const char * filename)
```

Opens and loads the file. The data will be converted from its original type to float. And the wavheader data will be stored internally to the class, but will not be stored in the user-provided buffer.

Currently only 16-bit and 32-bit data is supported. The importer also assumes data is mono so stereo data will be loaded as-is (i.e. interleaved)

### 12.138.2.3 Init()

```
void daisy::WaveTableLoader::Init (
 float * mem,
 size_t mem_size)
```

Initializes the Loader

### 12.138.2.4 SetWaveTableInfo()

```
Result daisy::WaveTableLoader::SetWaveTableInfo (
 size_t samps,
 size_t count)
```

Sets the size of the tables to allow access to the specific waveforms

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

- src/util/WaveTableLoader.h

## 12.139 daisy::WavFileInfo Struct Reference

```
#include <wavplayer.h>
```

### Public Attributes

- [WAV\\_FormatTypeDef raw\\_data](#)
- char [name \[256\]](#)

### 12.139.1 Detailed Description

Struct containing details of Wav File.

### 12.139.2 Member Data Documentation

#### 12.139.2.1 name

```
char daisy::WavFileInfo::name[256]
```

Wav filename

#### 12.139.2.2 raw\_data

```
WAV_FormatTypeDef daisy::WavFileInfo::raw_data
```

Raw wav data

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

- src/hid/wavplayer.h

## 12.140 daisy::WavPlayer Class Reference

```
#include <wavplayer.h>
```

### Public Member Functions

- void [Init](#) (const char \*search\_path)
- int [Open](#) (size\_t sel)
- int [Close](#) ()
- int16\_t [Stream](#) ()
- void [Prepare](#) ()
- void [Restart](#) ()
- void [SetLooping](#) (bool loop)
- bool [GetLooping](#) () const
- size\_t [GetNumberFiles](#) () const
- size\_t [GetCurrentFile](#) () const

### 12.140.1 Detailed Description

Wav Player that will load .wav files from an SD Card, and then provide a method of accessing the samples with double-buffering.

## 12.140.2 Member Function Documentation

### 12.140.2.1 Close()

```
int daisy::WavPlayer::Close ()
```

Closes whatever file is currently open.

Returns

&

### 12.140.2.2 GetCurrentFile()

```
size_t daisy::WavPlayer::GetCurrentFile () const [inline]
```

Returns

currently selected file.

### 12.140.2.3 GetLooping()

```
bool daisy::WavPlayer::GetLooping () const [inline]
```

Returns

Whether the [WavPlayer](#) is looping or not.

### 12.140.2.4 GetNumberFiles()

```
size_t daisy::WavPlayer::GetNumberFiles () const [inline]
```

Returns

The number of files loaded by the [WavPlayer](#)

### 12.140.2.5 Init()

```
void daisy::WavPlayer::Init (const char * search_path)
```

Initializes the [WavPlayer](#), loading up to max\_files of wav files from an SD Card.

### 12.140.2.6 Open()

```
int daisy::WavPlayer::Open (size_t sel)
```

Opens the file at index sel for reading.

**Parameters**

|            |              |
|------------|--------------|
| <i>sel</i> | File to open |
|------------|--------------|

**12.140.2.7 Prepare()**

```
void daisy::WavPlayer::Prepare ()
```

Collects buffer for playback when needed.

**12.140.2.8 Restart()**

```
void daisy::WavPlayer::Restart ()
```

Resets the playback position to the beginning of the file immediately

**12.140.2.9 SetLooping()**

```
void daisy::WavPlayer::SetLooping (bool loop) [inline]
```

Sets whether or not the current file will repeat after completing playback.

**Parameters**

|             |                         |
|-------------|-------------------------|
| <i>loop</i> | To loop or not to loop. |
|-------------|-------------------------|

**12.140.2.10 Stream()**

```
int16_t daisy::WavPlayer::Stream ()
```

**Returns**

The next sample if playing, otherwise returns 0

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

- src/hid/wavplayer.h

**12.141 daisy::WavWriter< transfer\_size > Class Template Reference**

```
#include <WavWriter.h>
```

## Classes

- struct [Config](#)

## Public Types

- enum class [Result](#) { **OK** , **ERROR** }
- enum class [BufferState](#) { **IDLE** , **FLUSH0** , **FLUSH1** }

## Public Member Functions

- void [Init](#) (const [Config](#) &cfg)
- void [Sample](#) (const float \*in)
- void [Write](#) ()
- void [SaveFile](#) ()
- void [OpenFile](#) (const char \*name)
- bool [IsRecording](#) () const
- uint32\_t [GetLengthSamps](#) ()
- float [GetLengthSeconds](#) ()

### 12.141.1 Detailed Description

```
template<size_t transfer_size>
class daisy::WavWriter< transfer_size >
```

Audio Recording Module

Record audio into a working buffer that is gradually written to a WAV file on an SD Card.

Recordings are made with floating point input, and will be converted to the specified bits per sample internally

For now only 16-bit and 32-bit (signed int) formats are supported f32 and s24 formats will be added next

The transfer size determines the amount of internal memory used, and can have an effect on the performance of the streaming behavior of the [WavWriter](#). Memory use can be calculated as:  $(2 * \text{transfer\_size})$  bytes Performance optimal with sizes: 16384, 32768

To use:

1. Create a [WavWriter<size>](#) object (e.g. [WavWriter<32768>](#) writer)
2. Configure the settings as desired by creating a [WavWriter<32768>::Config](#) struct and setting the settings.
3. Initialize the object with the configuration struct.
4. Open a new file for writing with: writer.[OpenFile\("FileName.wav"\)](#)
5. Write to it within your audio callback using: writer.[Sample\(value\)](#)
6. Fill the Wav File on the SD Card with data from your main loop by running: writer.[Write\(\)](#)
7. When finished with the recording finalize, and close the file with: writer.[SaveFile\(\)](#);

## 12.141.2 Member Enumeration Documentation

### 12.141.2.1 BufferState

```
template<size_t transfer_size>
enum daisy::WavWriter::BufferState [strong]
```

State of the internal Writing mechanism. When the buffer is a certain amount full one section will write its contents while the other is still being written to. This is performed circularly so that audio will be uninterrupted during writing.

### 12.141.2.2 Result

```
template<size_t transfer_size>
enum daisy::WavWriter::Result [strong]
```

Return values for write related functions

## 12.141.3 Member Function Documentation

### 12.141.3.1 GetLengthSamps()

```
template<size_t transfer_size>
uint32_t daisy::WavWriter< transfer_size >::GetLengthSamps () [inline]
```

Returns the current length in samples of the recording.

### 12.141.3.2 GetLengthSeconds()

```
template<size_t transfer_size>
float daisy::WavWriter< transfer_size >::GetLengthSeconds () [inline]
```

Returns the current length of the recording in seconds.

### 12.141.3.3 Init()

```
template<size_t transfer_size>
void daisy::WavWriter< transfer_size >::Init (
 const Config & cfg) [inline]
```

Initializes the WavFile header, and prepares the object for recording. "RIFF"

"WAVE"

"fmt "

"data"

Also calcs SubChunk2Size

#### 12.141.3.4 IsRecording()

```
template<size_t transfer_size>
bool daisy::WavWriter< transfer_size >::IsRecording () const [inline]
```

Returns whether recording is currently active or not.

#### 12.141.3.5 OpenFile()

```
template<size_t transfer_size>
void daisy::WavWriter< transfer_size >::OpenFile (
 const char * name) [inline]
```

Opens a file for writing. Writes the initial WAV Header, and gets ready for stream-based recording.

#### 12.141.3.6 Sample()

```
template<size_t transfer_size>
void daisy::WavWriter< transfer_size >::Sample (
 const float * in) [inline]
```

Records the current sample into the working buffer, queues writes to media when necessary.

##### Parameters

|           |                                            |
|-----------|--------------------------------------------|
| <i>in</i> | should be a pointer to an array of samples |
|-----------|--------------------------------------------|

#### 12.141.3.7 SaveFile()

```
template<size_t transfer_size>
void daisy::WavWriter< transfer_size >::SaveFile () [inline]
```

Finalizes the writing of the WAV file. This overwrites the WAV Header with the correct final size, and closes the fptr.

#### 12.141.3.8 Write()

```
template<size_t transfer_size>
void daisy::WavWriter< transfer_size >::Write () [inline]
```

Check buffer state and write

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

- src/util/WavWriter.h

## 12.142 daisy::Wm8731 Class Reference

```
#include <codec_wm8731.h>
```

### Classes

- struct [Config](#)

### Public Types

- enum class [Result](#) { [OK](#) , [ERR](#) }

### Public Member Functions

- [Result Init](#) (const [Config](#) &config, [I2CHandle](#) i2c)

#### 12.142.1 Detailed Description

Device driver for Cirrus (Wolfson) WM8731 Audio Codec

Currently only two-wire (I2C) interface format is supported, and only a limited set of features are configurable:

- Line inputs/outputs
- audio format/word length
- 48kHz

Support for headphones, microphone, and full functionality still needs to be added.

Use the Driver like this (this will be compatible with the Daisy Seed audio/sai config): [I2CHandle::Config](#) i2c\_config; [I2CHandle](#) i2c1\_handle; [Wm8731::Config](#) codec\_cfg; [Wm8731](#) codec; i2c\_config.periph = [I2CHandle::Config::Peripheral::I2C\\_1](#); i2c\_config.speed = [I2CHandle::Config::Speed::I2C\\_400KHZ](#); i2c\_config.pin\_config.scl = {DSY\_GPIOB, 6}; i2c\_config.pin\_config.sda = {DSY\_GPIOB, 9}; i2c1\_handle.Init(i2c\_config); codec\_cfg.Defaults(); // MCU is master, 24-bit, MSB LJ codec.Init(codec\_cfg, i2c1\_handle);

#### 12.142.2 Member Enumeration Documentation

##### 12.142.2.1 Result

```
enum daisy::Wm8731::Result [strong]
```

Return values for WM8731 Functions

### 12.142.3 Member Function Documentation

#### 12.142.3.1 Init()

```
Result daisy::Wm8731::Init (
 const Config & config,
 I2CHandle i2c)
```

Initializes the WM8731 device

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

- src/dev/codec\_wm8731.h

# Chapter 13

## File Documentation

### 13.1 src/sys/ffconf.h File Reference

```
#include "util/bsp_sd_diskio.h"
#include <stdlib.h>
```

#### Macros

- #define \_FFCONF 68300
- #define \_FS\_READONLY 0
- #define \_FS\_MINIMIZE 0
- #define \_USE\_STRFUNC 2
- #define \_USE\_FIND 0
- #define \_USE\_MKFS 1
- #define \_USE\_FASTSEEK 1
- #define \_USE\_EXPAND 0
- #define \_USE\_CHMOD 0
- #define \_USE\_LABEL 0
- #define \_USE\_FORWARD 0
- #define \_CODE\_PAGE 850
- #define \_USE\_LFN 1
- #define \_MAX\_LFN 255
- #define \_LFN\_UNICODE 0
- #define \_STRF\_ENCODE 3
- #define \_FS\_RPATH 0
- #define \_VOLUMES 2
- #define \_STR\_VOLUME\_ID 0
- #define \_VOLUME\_STRS
- #define \_MULTI\_PARTITION 0
- #define \_MIN\_SS 512
- #define \_MAX\_SS 512
- #define \_USE\_TRIM 0
- #define \_FS\_NOFSINFO 0
- #define \_FS\_TINY 0
- #define \_FS\_EXFAT 0
- #define \_FS\_NORTC 0

- #define \_NORTC\_MON 6
- #define \_NORTC\_MDAY 4
- #define \_NORTC\_YEAR 2015
- #define \_FS\_LOCK 0
- #define \_FS\_REENTRANT 0
- #define \_FS\_TIMEOUT 1000
- #define \_SYNC\_t osSemaphoreId
- #define ff\_malloc malloc
- #define ff\_free free

### 13.1.1 Detailed Description

Further fatts support.

### 13.1.2 Macro Definition Documentation

#### 13.1.2.1 \_CODE\_PAGE

```
#define _CODE_PAGE 850
```

This option specifies the OEM code page to be used on the target system. / Incorrect setting of the code page can cause a file open failure. // 1 - ASCII (No extended character. Non-LFN cfg. only) / 437 - U.S. / 720 - Arabic / 737 - Greek / 771 - KBL / 775 - Baltic / 850 - Latin 1 / 852 - Latin 2 / 855 - Cyrillic / 857 - Turkish / 860 - Portuguese / 861 - Icelandic / 862 - Hebrew / 863 - Canadian French / 864 - Arabic / 865 - Nordic / 866 - Russian / 869 - Greek 2 / 932 - Japanese (DBCS) / 936 - Simplified Chinese (DBCS) / 949 - Korean (DBCS) / 950 - Traditional Chinese (DBCS)

#### 13.1.2.2 \_FFCONF

```
#define _FFCONF 68300
```

FatFs - Generic FAT file system module R0.12c (C)ChaN, 2017

Attention

© Copyright (c) 2019 STMicroelectronics. All rights reserved.

This software component is licensed by ST under Ultimate Liberty license SLA0044, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: [www.st.com/SLA0044](http://www.st.com/SLA0044) Revision ID

### 13.1.2.3 \_FS\_EXFAT

```
#define _FS_EXFAT 0
```

This option switches support of exFAT file system. (0:Disable or 1:Enable) / When enable exFAT, also LFN needs to be enabled. (\_USE\_LFN >= 1) / Note that enabling exFAT discards C89 compatibility.

### 13.1.2.4 \_FS\_LOCK

```
#define _FS_LOCK 0
```

0:Disable or >=1:Enable The option \_FS\_LOCK switches file lock function to control duplicated file open / and illegal operation to open objects. This option must be 0 when \_FS\_READONLY / is 1. // 0: Disable file lock function. To avoid volume corruption, application program / should avoid illegal open, remove and rename to the open objects. / >0: Enable file lock function. The value defines how many files/sub-directories / can be opened simultaneously under file lock control. Note that the file / lock control is independent of re-entrancy.

### 13.1.2.5 \_FS\_MINIMIZE

```
#define _FS_MINIMIZE 0
```

0 to 3 This option defines minimization level to remove some basic API functions. // 0: All basic functions are enabled. / 1: f\_stat(), f\_getfree(), f\_unlink(), f\_mkdir(), f\_truncate() and f\_rename() / are removed. / 2: f\_opendir(), f\_readdir() and f\_closedir() are removed in addition to 1. / 3: f\_lseek() function is removed in addition to 2.

### 13.1.2.6 \_FS\_NOFSINFO

```
#define _FS_NOFSINFO 0
```

0,1,2 or 3 If you need to know correct free space on the FAT32 volume, set bit 0 of this / option, and f\_getfree() function at first time after volume mount will force / a full FAT scan. Bit 1 controls the use of last allocated cluster number. // bit0=0: Use free cluster count in the FSINFO if available. / bit0=1: Do not trust free cluster count in the FSINFO. / bit1=0: Use last allocated cluster number in the FSINFO if available. / bit1=1: Do not trust last allocated cluster number in the FSINFO.

### 13.1.2.7 \_FS\_NORTC

```
#define _FS_NORTC 0
```

&

### 13.1.2.8 \_FS\_READONLY

```
#define _FS_READONLY 0
```

0:Read/Write or 1:Read only This option switches read-only configuration. (0:Read/Write or 1:Read-only) / Read-only configuration removes writing API functions, f\_write(), f\_sync(), / f\_unlink(), f\_mkdir(), f\_chmod(), f\_rename(), f\_truncate(), f\_getfree() / and optional writing functions as well.

### 13.1.2.9 \_FS\_REENTRANT

```
#define _FS_REENTRANT 0
```

0:Disable or 1:Enable

### 13.1.2.10 \_FS\_RPATH

```
#define _FS_RPATH 0
```

0 to 2 This option configures support of relative path. // 0: Disable relative path and remove related functions. / 1: Enable relative path. f\_chdir() and f\_chdrive() are available. / 2: f\_getcwd() function is available in addition to 1.

### 13.1.2.11 \_FS\_TIMEOUT

```
#define _FS_TIMEOUT 1000
```

Timeout period in unit of time ticks

### 13.1.2.12 \_FS\_TINY

```
#define _FS_TINY 0
```

0:Normal or 1:Tiny This option switches tiny buffer configuration. (0:Normal or 1:tiny) / At the tiny configuration, size of file object (FIL) is reduced \_MAX\_SS bytes. / Instead of private sector buffer eliminated from the file object, common sector / buffer in the file system object (FATFS) is used for the file data transfer.

### 13.1.2.13 \_LFN\_UNICODE

```
#define _LFN_UNICODE 0
```

0:ANSI/OEM or 1:Unicode This option switches character encoding on the API. (0:ANSI/OEM or 1:UTF-16) / To use Unicode string for the path name, enable LFN and set \_LFN\_UNICODE = 1. / This option also affects behavior of string I/O functions.

### 13.1.2.14 \_MAX\_LFN

```
#define _MAX_LFN 255
```

Maximum LFN length to handle (12 to 255) The \_USE\_LFN switches the support of long file name (LFN). // 0: Disable support of LFN. \_MAX\_LFN has no effect. / 1: Enable LFN with static working buffer on the BSS. Always NOT thread-safe. / 2: Enable LFN with dynamic working buffer on the STACK. / 3: Enable LFN with dynamic working buffer on the HEAP. // To enable the LFN, Unicode handling functions (option/unicode.c) must be added / to the project. The working buffer occupies (\_MAX\_LFN + 1) \* 2 bytes and / additional 608 bytes at exFAT enabled. \_MAX\_LFN can be in range from 12 to 255. / It should be set 255 to support full featured LFN operations. / When use stack for the working buffer, take care on stack overflow. When use heap / memory for the working buffer, memory management functions, ff\_memalloc() and / ff\_memfree(), must be added to the project.

**13.1.2.15 \_MAX\_SS**

```
#define _MAX_SS 512
```

512, 1024, 2048 or 4096 These options configure the range of sector size to be supported. (512, 1024, / 2048 or 4096) Always set both 512 for most systems, all type of memory cards and / harddisk. But a larger value may be required for on-board flash memory and some / type of optical media. When \_MAX\_SS is larger than \_MIN\_SS, FatFs is configured / to variable sector size and GET\_SECTOR\_SIZE command must be implemented to the / disk\_ioctl() function.

**13.1.2.16 \_MIN\_SS**

```
#define _MIN_SS 512
```

512, 1024, 2048 or 4096

**13.1.2.17 \_MULTI\_PARTITION**

```
#define _MULTI_PARTITION 0
```

0:Single partition, 1:Multiple partition This option switches support of multi-partition on a physical drive. / By default (0), each logical drive number is bound to the same physical drive / number and only an FAT volume found on the physical drive will be mounted. / When multi-partition is enabled (1), each logical drive number can be bound to / arbitrary physical drive and partition listed in the VolToPart[]. Also f\_fdisk() / funciton will be available.

**13.1.2.18 \_NORTC\_MDAY**

```
#define _NORTC_MDAY 4
```

&

**13.1.2.19 \_NORTC\_MON**

```
#define _NORTC_MON 6
```

&

**13.1.2.20 \_NORTC\_YEAR**

```
#define _NORTC_YEAR 2015
```

The option \_FS\_NORTC switches timestamp functiton. If the system does not have / any RTC function or valid timestamp is not needed, set \_FS\_NORTC = 1 to disable / the timestamp function. All objects modified by FatFs will have a fixed timestamp / defined by \_NORTC\_MON, \_NORTC\_MDAY and \_NORTC\_YEAR in local time. / To enable timestamp function (\_FS\_NORTC = 0), get\_fattime() function need to be / added to the project to get current time form real-time clock. \_NORTC\_MON, / \_NORTC\_MDAY and \_NORTC\_YEAR have no effect. / These options have no effect at read-only configuration (\_FS\_READONLY = 1).

### 13.1.2.21 \_STR\_VOLUME\_ID

```
#define _STR_VOLUME_ID 0
```

0:Use only 0-9 for drive ID, 1:Use strings for drive ID

### 13.1.2.22 \_STRF\_ENCODE

```
#define _STRF_ENCODE 3
```

When \_LFN\_UNICODE == 1, this option selects the character encoding ON THE FILE to / be read/written via string I/O functions, f\_gets(), f\_putc(), f\_puts and f\_printf(). // 0: ANSI/OEM / 1: UTF-16LE / 2: UTF-16BE / 3: UTF-8 // This option has no effect when \_LFN\_UNICODE == 0.

### 13.1.2.23 \_SYNC\_t

```
#define _SYNC_t osSemaphoreId
```

The option \_FS\_REENTRANT switches the re-entrancy (thread safe) of the FatFs / module itself. Note that regardless of this option, file access to different / volume is always re-entrant and volume control functions, f\_mount(), f\_mkfs() / and f\_fdisk() function, are always not re-entrant. Only file/directory access / to the same volume is under control of this function. // 0: Disable re-entrancy. \_FS\_TIMEOUT and \_SYNC\_t have no effect. / 1: Enable re-entrancy. Also user provided synchronization handlers, / ff\_req\_grant(), ff\_rel\_grant(), ff\_del\_syncobj() and ff\_cre\_syncobj() / function, must be added to the project. Samples are available in / option/syscall.c. // The \_FS←\_TIMEOUT defines timeout period in unit of time tick. / The \_SYNC\_t defines O/S dependent sync object type. e.g. HANDLE, ID, OS\_EVENT\*, / SemaphoreHandle\_t and etc.. A header file for O/S definitions needs to be / included somewhere in the scope of ff.h.

### 13.1.2.24 \_USE\_CHMOD

```
#define _USE_CHMOD 0
```

This option switches attribute manipulation functions, f\_chmod() and f\_utime(). / (0:Disable or 1:Enable) Also \_←\_FS\_READONLY needs to be 0 to enable this option.

### 13.1.2.25 \_USE\_EXPAND

```
#define _USE_EXPAND 0
```

This option switches f\_expand function. (0:Disable or 1:Enable)

### 13.1.2.26 \_USE\_FASTSEEK

```
#define _USE_FASTSEEK 1
```

This option switches fast seek feature. (0:Disable or 1:Enable)

**13.1.2.27 \_USE\_FIND**

```
#define _USE_FIND 0
```

This option switches filtered directory read functions, f\_findfirst() and / f\_findnext(). (0:Disable, 1:Enable 2:Enable with matching altname[] too)

**13.1.2.28 \_USE\_FORWARD**

```
#define _USE_FORWARD 0
```

This option switches f\_forward() function. (0:Disable or 1:Enable)

**13.1.2.29 \_USE\_LABEL**

```
#define _USE_LABEL 0
```

This option switches volume label functions, f\_getlabel() and f\_setlabel(). / (0:Disable or 1:Enable)

**13.1.2.30 \_USE\_LFN**

```
#define _USE_LFN 1
```

0 to 3

**13.1.2.31 \_USE\_MKFS**

```
#define _USE_MKFS 1
```

This option switches f\_mkfs() function. (0:Disable or 1:Enable)

**13.1.2.32 \_USE\_STRFUNC**

```
#define _USE_STRFUNC 2
```

0:Disable or 1-2:Enable This option switches string functions, f\_gets(), f\_putc(), f\_puts() and / f\_printf(). // 0: Disable string functions. / 1: Enable without LF-CRLF conversion. / 2: Enable with LF-CRLF conversion.

**13.1.2.33 \_USE\_TRIM**

```
#define _USE_TRIM 0
```

This option switches support of ATA-TRIM. (0:Disable or 1:Enable) / To enable Trim function, also CTRL\_TRIM command should be implemented to the / disk\_ioctl() function.

### 13.1.2.34 \_VOLUME\_STRS

```
#define _VOLUME_STRS

Value:
 "RAM", "NAND", "CF", "SD1", "SD2", "USB1", "USB2", \
 "USB3"
```

\_STR\_VOLUME\_ID switches string support of volume ID. / When \_STR\_VOLUME\_ID is set to 1, also pre-defined strings can be used as drive / number in the path name. \_VOLUME\_STRS defines the drive ID strings for each / logical drives. Number of items must be equal to \_VOLUMES. Valid characters for / the drive ID strings are: A-Z and 0-9.

### 13.1.2.35 \_VOLUMES

```
#define _VOLUMES 2

Number of volumes (logical drives) to be used.
```

### 13.1.2.36 ff\_free

```
#define ff_free free

define the ff_malloc ff_free macros as standard malloc free
```

### 13.1.2.37 ff\_malloc

```
#define ff_malloc malloc

define the ff_malloc ff_free macros as standard malloc free
```

## 13.2 src/util/usbh\_diskio.h File Reference

Header for usbh\_diskio.c module.

```
#include "usbh_core.h"
#include "usbh_msc.h"
```

### Variables

- const Diskio\_drvTypeDef **USBH\_Driver**

### 13.2.1 Detailed Description

Header for usbh\_diskio.c module.

(based on usbh\_diskio\_dma\_template.h v2.0.2)

**Attention**

© Copyright (c) 2021 STMicroelectronics. All rights reserved.

This software component is licensed by ST under Ultimate Liberty license SLA0044, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: [www.st.com/SLA0044](http://www.st.com/SLA0044)

# Index

\_CODE\_PAGE  
    ffconf.h, 408  
\_FFCONF  
    ffconf.h, 408  
\_FS\_EXFAT  
    ffconf.h, 408  
\_FS\_LOCK  
    ffconf.h, 409  
\_FS\_MINIMIZE  
    ffconf.h, 409  
\_FS\_NOFSINFO  
    ffconf.h, 409  
\_FS\_NORTC  
    ffconf.h, 409  
\_FS\_READONLY  
    ffconf.h, 409  
\_FS\_REENTRANT  
    ffconf.h, 409  
\_FS\_RPATH  
    ffconf.h, 410  
\_FS\_TIMEOUT  
    ffconf.h, 410  
\_FS\_TINY  
    ffconf.h, 410  
\_LFN\_UNICODE  
    ffconf.h, 410  
\_MAX\_LFN  
    ffconf.h, 410  
\_MAX\_SS  
    ffconf.h, 410  
\_MIN\_SS  
    ffconf.h, 411  
\_MULTI\_PARTITION  
    ffconf.h, 411  
\_NORTC\_MDAY  
    ffconf.h, 411  
\_NORTC\_MON  
    ffconf.h, 411  
\_NORTC\_YEAR  
    ffconf.h, 411  
\_STRF\_ENCODE  
    ffconf.h, 412  
\_STR\_VOLUME\_ID  
    ffconf.h, 411  
\_SYNC\_t  
    ffconf.h, 412  
\_USE\_CHMOD  
    ffconf.h, 412  
\_USE\_EXPAND  
    ffconf.h, 412  
  
\_USE\_FASTSEEK  
    ffconf.h, 412  
\_USE\_FIND  
    ffconf.h, 412  
\_USE\_FORWARD  
    ffconf.h, 413  
\_USE\_LABEL  
    ffconf.h, 413  
\_USE\_LFN  
    ffconf.h, 413  
\_USE\_MKFS  
    ffconf.h, 413  
\_USE\_STRFUNC  
    ffconf.h, 413  
\_USE\_TRIM  
    ffconf.h, 413  
\_VOLUMES  
    ffconf.h, 414  
\_VOLUME\_STRS  
    ffconf.h, 413  
\_\_attribute\_\_  
    daisy::LedDriverPca9685< numDrivers, persistent-  
        BufferContents >, 256  
    daisy::UiEventQueue, 373  
~AnalogControl  
    daisy::AnalogControl, 129  
~DaisyPatch  
    daisy::DaisyPatch, 181  
~DaisyPetal  
    daisy::DaisyPetal, 196  
~GateIn  
    daisy::GateIn, 241  
~Parameter  
    daisy::Parameter, 301  
  
A0  
    daisy::seed, 110  
A1  
    daisy::patch\_sm::DaisyPatchSM, 193  
A12  
    daisy::seed, 110  
ActiveSensing  
    daisy, 108  
adc  
    daisy::DaisySeed, 216  
AddButtonPressed  
    daisy::UiEventQueue, 374  
AddButtonReleased  
    daisy::UiEventQueue, 375

AddEncoderActivityChanged  
     daisy::UiEventQueue, 375

AddEncoderTurned  
     daisy::UiEventQueue, 375

AddPotActivityChanged  
     daisy::UiEventQueue, 375

AddPotMoved  
     daisy::UiEventQueue, 375

address  
     daisy::I2CHandle::Config, 145

Advance  
     daisy::RingBuffer< T, size >, 322

Alignment  
     daisy, 105

AllNotesOff  
     daisy, 106

allowEntering\_  
     daisy::AbstractMenu, 119

AllSoundOff  
     daisy, 106

ANALOG  
     daisy::GPIO, 243

ANALOG\_DIGITAL\_CONVERSION, 36

AnalogControl  
     daisy::AnalogControl, 129

AppendNewLine  
     EXTERNAL, 32

AppentToString  
     daisy::MappedFloatValue, 266  
     daisy::MappedIntValue, 269  
     daisy::MappedStringListValue, 272  
     daisy::MappedValue, 275

ApplicationTypeDef  
     daisy, 105

ArrowButtonType  
     daisy, 105

asCallbackFunctionItem  
     daisy::AbstractMenu::ItemConfig, 249

AsChannelPressure  
     daisy::MidiEvent, 278

asCheckboxItem  
     daisy::AbstractMenu::ItemConfig, 249

AsControlChange  
     daisy::MidiEvent, 279

asCustomItem  
     daisy::AbstractMenu::ItemConfig, 249

asMappedValueItem  
     daisy::AbstractMenu::ItemConfig, 250

AsNoteOff  
     daisy::MidiEvent, 279

AsNoteOn  
     daisy::MidiEvent, 279

asOpenUiPageItem  
     daisy::AbstractMenu::ItemConfig, 250

AsPitchBend  
     daisy::MidiEvent, 279

AsPolyphonicKeyPressure  
     daisy::MidiEvent, 279

AsProgramChange  
     daisy::MidiEvent, 279

AUDIO, 28

audio\_handle  
     daisy::DaisySeed, 216

AudioBlockSize  
     daisy::DaisyField, 173  
     daisy::DaisyPatch, 181  
     daisy::DaisyPetal, 196  
     daisy::DaisyPod, 205  
     daisy::DaisySeed, 213  
     daisy::DaisyVersio, 218  
     daisy::patch\_sm::DaisyPatchSM, 187

AudioCallback  
     daisy::AudioHandle, 132

AudioCallbackRate  
     daisy::DaisyField, 173  
     daisy::DaisyPatch, 181  
     daisy::DaisyPetal, 196  
     daisy::DaisyPod, 206  
     daisy::DaisySeed, 213  
     daisy::DaisyVersio, 218  
     daisy::patch\_sm::DaisyPatchSM, 187

AudioFormat  
     daisy::WAV\_FormatTypeDef, 395

AudioSampleRate  
     daisy::DaisyField, 173  
     daisy::DaisyPatch, 181  
     daisy::DaisyPetal, 197  
     daisy::DaisyPod, 206  
     daisy::DaisySeed, 213  
     daisy::DaisyVersio, 218  
     daisy::patch\_sm::DaisyPatchSM, 188

Back  
     daisy::FIFOBase< T >, 232

BitDepth  
     daisy::DacHandle, 168  
     daisy::SaiHandle::Config, 151

BitPerSample  
     daisy::WAV\_FormatTypeDef, 395

BITS\_1  
     daisy::SdmmcHandler, 334

BITS\_4  
     daisy::SdmmcHandler, 334

BLOCK\_ERASE\_32K\_CMD  
     FLASH, 47

BLOCK\_ERASE\_CMD  
     FLASH, 47

BlockAlign  
     daisy::WAV\_FormatTypeDef, 395

BlockingReceive  
     daisy::SpiHandle, 343

BlockingTransmit  
     daisy::SpiHandle, 343

BlockingTransmitAndReceive  
     daisy::SpiHandle, 344

BlockNbr  
     DSY\_SD\_CardInfoTypeDef, 224

BlockSize  
    DSY\_SD\_CardInfoTypeDef, 224

BLUE  
    daisy::Color, 139

Blue  
    daisy::Color, 139

BOARDS, 70

BoardVersion  
    daisy::DaisySeed, 211

Boost  
    daisy::System::Config, 158

BSP\_SD\_AbortCallback  
    UTILITY, 80

BSP\_SD\_CardInfo  
    UTILITY, 76

BSP\_SD\_Erase  
    UTILITY, 80

BSP\_SD\_GetCardInfo  
    UTILITY, 80

BSP\_SD\_GetCardState  
    UTILITY, 81

BSP\_SD\_Init  
    UTILITY, 81

BSP\_SD\_IsDetected  
    UTILITY, 81

BSP\_SD\_ITConfig  
    UTILITY, 81

BSP\_SD\_ReadBlocks  
    UTILITY, 82

BSP\_SD\_ReadBlocks\_DMA  
    UTILITY, 82

BSP\_SD\_ReadCpltCallback  
    UTILITY, 83

BSP\_SD\_WriteBlocks  
    UTILITY, 83

BSP\_SD\_WriteBlocks\_DMA  
    UTILITY, 83

BSP\_SD\_WriteCpltCallback  
    UTILITY, 84

BufferState  
    daisy::DacHandle, 168  
    daisy::WavWriter< transfer\_size >, 403

BusWidth  
    daisy::SdmmcHandler, 334

button1  
    daisy::DaisyPod, 209

button2  
    daisy::DaisyPod, 209

BUTTON\_2  
    daisy::DaisyPod, 205

BUTTON\_LAST  
    daisy::DaisyPod, 205

buttons  
    daisy::DaisyPod, 209

ByteRate  
    daisy::WAV\_FormatTypeDef, 395

callbackFunctionItem  
    daisy::AbstractMenu, 114

CallbackFunctionPtr  
    daisy::I2CHandle, 245  
    daisy::SaiHandle, 331  
    daisy::SpiHandle, 342

CanBeEnteredForEditing  
    daisy::AbstractMenu::CustomItem, 166

capacity  
    daisy::RingBuffer< T, 0 >, 328  
    daisy::RingBuffer< T, size >, 323

CardSpeed  
    DSY\_SD\_CardInfoTypeDef, 224

CardType  
    DSY\_SD\_CardInfoTypeDef, 224

CardVersion  
    DSY\_SD\_CardInfoTypeDef, 224

ChangeAudioCallback  
    daisy::DaisyField, 173, 174  
    daisy::DaisyPatch, 181  
    daisy::DaisyPetal, 197  
    daisy::DaisyPod, 206  
    daisy::DaisySeed, 213  
    daisy::DaisyVersio, 218, 219  
    daisy::patch\_sm::DaisyPatchSM, 188

ChangeCallback  
    daisy::AudioHandle, 132, 133

Channel  
    daisy::DacHandle, 168

channel  
    daisy::AllNotesOffEvent, 127  
    daisy::AllSoundOffEvent, 127  
    daisy::ChannelPressureEvent, 137  
    daisy::ControlChangeEvent, 163  
    daisy::LocalControlEvent, 258  
    daisy::MidiEvent, 279  
    daisy::MonoModeOnEvent, 284  
    daisy::NoteOffEvent, 285  
    daisy::NoteOnEvent, 286  
    daisy::OmniModeOffEvent, 289  
    daisy::OmniModeOnEvent, 289  
    daisy::PitchBendEvent, 307  
    daisy::PolyModeOnEvent, 308  
    daisy::ProgramChangeEvent, 311  
    daisy::ResetAllControllersEvent, 319

ChannelMode  
    daisy, 107

ChannelModeLast  
    daisy, 106

ChannelModeType  
    daisy, 106

ChannelPressure  
    daisy, 107

CheckBoardVersion  
    daisy::DaisySeed, 213

checkboxItem  
    daisy::AbstractMenu, 114

CheckError  
    daisy::SpiHandle, 344  
    daisy::UartHandler, 367

CHIP\_ERASE\_CMD  
     FLASH, 47

ChunkId  
     daisy::WAV\_FormatTypeDef, 396

Class  
     DSY\_SD\_CardInfoTypeDef, 224

Clear  
     daisy::FIFOBase< T >, 233  
     daisy::LcdHD44780, 251  
     daisy::StackBase< T >, 350

CLEAR\_EXT\_READ\_PARAM\_CMD  
     FLASH, 47

ClearFuncPtr  
     daisy::UiCanvasDescriptor, 372

ClearLeds  
     daisy::DaisyPetal, 197  
     daisy::DaisyPod, 206

clk  
     daisy::QSPIHandle::Config, 150  
     daisy::ShiftRegister4021< num\_daisychained,  
         num\_parallel >::Config, 153

clock\_powersave  
     daisy::SdmmcHandler::Config, 153

Close  
     daisy::UiPage, 377  
     daisy::WavPlayer, 400

closeMenuItem  
     daisy::AbstractMenu, 114

ClosePage  
     daisy::UI, 370

CODEC, 68

Config  
     daisy::GPIO::Config, 143

Configure  
     daisy::DaisySeed, 214

Contains  
     daisy::FIFOBase< T >, 233  
     daisy::StackBase< T >, 350

Continue  
     daisy, 108

control\_number  
     daisy::ControlChangeEvent, 163

ControlChange  
     daisy, 107

CONTROLS, 28

controls  
     daisy::DaisyPatch, 184

CountEqualTo  
     daisy::FIFOBase< T >, 233  
     daisy::StackBase< T >, 350

CounterDir  
     daisy::TimerHandle::Config, 158

csb\_pin\_state  
     daisy::Wm8731::Config, 162

Ctrl  
     daisy::DaisyPatch, 180

CUBE  
     daisy::Parameter, 300

cube  
     UTILITY, 84

Curve  
     daisy::Parameter, 300

customItem  
     daisy::AbstractMenu, 114

CV\_2  
     daisy::DaisyField, 172

CV\_3  
     daisy::DaisyField, 172

CV\_4  
     daisy::DaisyField, 172

CV\_LAST  
     daisy::DaisyField, 172

CYAN  
     daisy::Color, 139

D0  
     daisy::seed, 111

D31  
     daisy::seed, 111

Dac, 90  
     Defaults, 92  
     ERR, 92  
     Init, 93  
     mode, 97  
     OK, 92  
     Pin, 91  
     range, 97  
     ReadAnalogPinRaw, 93  
     ReadAnalogPinVolts, 93  
     ReadDigitalPin, 95  
     Result, 91, 92  
     spi\_config, 98  
     threshold, 98  
     TwelveBitUintToVolts, 95  
     Update, 95  
     value, 98  
     VoltageRange, 92  
     VoltsTo12BitUint, 96  
     WriteAnalogPinRaw, 96  
     WriteAnalogPinVolts, 97  
     WriteDigitalPin, 97

DacCallback  
     daisy::DacHandle, 168

daisy, 99  
     ActiveSensing, 108  
     Alignment, 105  
     AllNotesOff, 106  
     AllSoundOff, 106  
     ApplicationTypeDef, 105  
     ArrowButtonType, 105  
     ChannelMode, 107  
     ChannelModeLast, 106  
     ChannelModeType, 106  
     ChannelPressure, 107  
     Continue, 108  
     ControlChange, 107  
     down, 105

dsy\_i2c\_global\_init, 108  
GPIOPort, 106  
kWavFileChunkId, 109  
kWavFileSubChunk1Id, 109  
kWavFileSubChunk2Id, 109  
kWavFileWavId, 109  
left, 105  
LocalControl, 106  
LOGGER\_EXTERNAL, 107  
LOGGER\_INTERNAL, 107  
LOGGER\_NONE, 107  
LOGGER\_SEMIHOST, 107  
LoggerDestination, 106  
MessageLast, 107  
MidiMessageType, 107  
MonoModeOn, 106  
MTCQuarterFrame, 107  
NoteOff, 107  
NoteOn, 107  
OmniModeOff, 106  
OmniModeOn, 106  
PitchBend, 107  
PolyModeOn, 106  
PolyphonicKeyPressure, 107  
PORTA, 106  
PORTB, 106  
PORTC, 106  
PORTD, 106  
PORTE, 106  
PORTF, 106  
PORTG, 106  
PORTH, 106  
PORTI, 106  
PORTJ, 106  
PORTK, 106  
PORTX, 106  
ProgramChange, 107  
Reset, 108  
ResetAllControllers, 106  
right, 105  
SCUndefined0, 107  
SCUndefined1, 107  
SongPositionPointer, 107  
SongSelect, 107  
SRTUndefined0, 108  
SRTUndefined1, 108  
SSD130x4WireSpi128x32Driver, 103  
SSD130x4WireSpi128x64Driver, 104  
SSD130x4WireSpi64x32Driver, 104  
SSD130x4WireSpi64x48Driver, 104  
SSD130x4WireSpi98x16Driver, 104  
SSD130xl2c128x32Driver, 104  
SSD130xl2c128x64Driver, 104  
SSD130xl2c64x32Driver, 104  
SSD130xl2c64x48Driver, 105  
SSD130xl2c98x16Driver, 105  
Start, 108  
Stop, 108  
SysExEnd, 107  
SystemCommon, 107  
SystemCommonLast, 107  
SystemCommonType, 107  
SystemExclusive, 107  
SystemRealTime, 107  
SystemRealTimeLast, 108  
SystemRealTimeType, 108  
TimingClock, 108  
TuneRequest, 107  
up, 105  
WavFormatCode, 108  
daisy::AbstractMenu, 113  
allowEntering\_, 119  
callbackFunctionItem, 114  
checkboxItem, 114  
closeMenuItem, 114  
customItem, 114  
Init, 115  
isEditing\_, 119  
IsFunctionButtonDown, 115  
items\_, 119  
ItemType, 114  
leftRightSelectUpDownModify, 115  
numItems\_, 119  
OnArrowButton, 115  
OnCancelButton, 116  
OnFunctionButton, 116  
OnMenuEncoderTurned, 117  
OnOkayButton, 117  
OnShow, 118  
OnValueEncoderTurned, 118  
OnValuePotMoved, 118  
openUiPageItem, 114  
Orientation, 114  
orientation\_, 119  
selectedItemIdx\_, 119  
upDownSelectLeftRightModify, 115  
valueItem, 114  
daisy::AbstractMenu::CustomItem, 165  
CanBeEnteredForEditing, 166  
Draw, 166  
ModifyValue, 166  
OnOkayButton, 167  
daisy::AbstractMenu::ItemConfig, 249  
asCallbackFunctionItem, 249  
asCheckboxItem, 249  
asCustomItem, 249  
asMappedValueItem, 250  
asOpenUiPageItem, 250  
itemObject, 250  
pageToOpen, 250  
text, 250  
type, 250  
valueToModify, 250, 251  
daisy::AdcChannelConfig, 120  
InitMux, 120  
InitSingle, 121

mux\_channels\_, 121  
 mux\_pin\_, 121  
 MUX\_SEL\_0, 120  
 MUX\_SEL\_1, 120  
 MUX\_SEL\_2, 120  
 MUX\_SEL\_LAST, 120  
 MuxPin, 120  
 pin\_, 121  
 daisy::AdcHandle, 122  
 Get, 123  
 GetFloat, 123  
 GetMux, 124  
 GetMuxFloat, 124  
 GetMuxPtr, 124  
 GetPtr, 125  
 Init, 125  
 OverSampling, 122  
 OVS\_1024, 123  
 OVS\_128, 123  
 OVS\_16, 123  
 OVS\_256, 123  
 OVS\_32, 123  
 OVS\_4, 123  
 OVS\_512, 123  
 OVS\_64, 123  
 OVS\_8, 123  
 OVS\_LAST, 123  
 OVS\_NONE, 123  
 Start, 125  
 Stop, 126  
 daisy::Ak4556, 126  
 DelInit, 126  
 Init, 126  
 daisy::AllNotesOffEvent, 126  
 channel, 127  
 daisy::AllSoundOffEvent, 127  
 channel, 127  
 daisy::AnalogControl, 128  
 ~AnalogControl, 129  
 AnalogControl, 129  
 GetRawFloat, 129  
 GetRawValue, 129  
 Init, 129  
 InitBipolarCv, 130  
 Process, 130  
 SetCoeff, 130  
 SetSampleRate, 130  
 Value, 131  
 daisy::AudioHandle, 131  
 AudioCallback, 132  
 ChangeCallback, 132, 133  
 DelInit, 133  
 GetChannels, 133  
 GetConfig, 133  
 GetSampleRate, 133  
 Init, 133  
 InputBuffer, 132  
 InterleavingAudioCallback, 132  
 InterleavingInputBuffer, 132  
 InterleavingOutputBuffer, 132  
 OutputBuffer, 132  
 SetBlockSize, 134  
 SetPostGain, 134  
 SetSampleRate, 134  
 Start, 134  
 Stop, 134  
 daisy::AudioHandle::Config, 140  
 daisy::BlockingSpiTransport, 135  
 daisy::BlockingSpiTransport::Config, 140  
 daisy::ButtonMonitor< BackendType, numButtons >, 135  
 GetBackend, 135  
 GetNumButtonsMonitored, 136  
 Init, 136  
 IsButtonPressed, 136  
 Process, 137  
 daisy::ChannelPressureEvent, 137  
 channel, 137  
 pressure, 137  
 daisy::Color, 138  
 BLUE, 139  
 Blue, 139  
 CYAN, 139  
 GOLD, 139  
 GREEN, 139  
 Green, 139  
 Init, 139  
 LAST, 139  
 OFF, 139  
 PresetColor, 138  
 PURPLE, 139  
 RED, 139  
 Red, 140  
 WHITE, 139  
 daisy::ControlChangeEvent, 163  
 channel, 163  
 control\_number, 163  
 value, 163  
 daisy::CpuLoadMeter, 164  
 GetAvgCpuLoad, 164  
 GetMaxCpuLoad, 164  
 GetMinCpuLoad, 164  
 Init, 164  
 OnBlockEnd, 165  
 OnBlockStart, 165  
 Reset, 165  
 daisy::DacHandle, 167  
 BitDepth, 168  
 BufferState, 168  
 Channel, 168  
 DacCallback, 168  
 Init, 169  
 Mode, 168  
 Result, 169  
 Start, 169  
 Stop, 169

WriteValue, 170  
daisy::DacHandle::Config, 141  
    target\_samplerate, 141  
daisy::DaisyField, 170  
    AudioBlockSize, 173  
    AudioCallbackRate, 173  
    AudioSampleRate, 173  
    ChangeAudioCallback, 173, 174  
    CV\_2, 172  
    CV\_3, 172  
    CV\_4, 172  
    CV\_LAST, 172  
    DelayMs, 174  
    GetCv, 174  
    GetCvValue, 175  
    GetKnob, 175  
    GetKnobValue, 175  
    GetSwitch, 175  
    Init, 176  
    KeyboardFallingEdge, 176  
    KeyboardRisingEdge, 176  
    KeyboardState, 176  
    KNOB\_1, 172  
    KNOB\_2, 172  
    KNOB\_3, 172  
    KNOB\_4, 172  
    KNOB\_5, 172  
    KNOB\_6, 172  
    KNOB\_7, 172  
    KNOB\_8, 172  
    KNOB\_LAST, 172  
    LED\_KEY\_A1, 173  
    LED\_KEY\_A2, 173  
    LED\_KEY\_A3, 173  
    LED\_KEY\_A4, 173  
    LED\_KEY\_A5, 173  
    LED\_KEY\_A6, 173  
    LED\_KEY\_A7, 172  
    LED\_KEY\_A8, 172  
    LED\_KEY\_B1, 172  
    LED\_KEY\_B2, 172  
    LED\_KEY\_B3, 172  
    LED\_KEY\_B4, 172  
    LED\_KEY\_B5, 172  
    LED\_KEY\_B6, 172  
    LED\_KEY\_B7, 172  
    LED\_KEY\_B8, 172  
    LED\_KNOB\_1, 173  
    LED\_KNOB\_2, 173  
    LED\_KNOB\_3, 173  
    LED\_KNOB\_4, 173  
    LED\_KNOB\_5, 173  
    LED\_KNOB\_6, 173  
    LED\_KNOB\_7, 173  
    LED\_KNOB\_8, 173  
    LED\_LAST, 173  
    LED\_SW\_1, 173  
    LED\_SW\_2, 173  
ProcessAllControls, 177  
ProcessAnalogControls, 177  
ProcessDigitalControls, 177  
SetAudioBlockSize, 177  
SetAudioSampleRate, 177  
SetCvOut1, 177  
SetCvOut2, 178  
StartAdc, 178  
StartAudio, 178  
StartDac, 178  
StopAdc, 178  
StopAudio, 178  
SW\_1, 171  
SW\_2, 171  
SW\_LAST, 171  
VegasMode, 179  
daisy::DaisyPatch, 179  
    ~DaisyPatch, 181  
    AudioBlockSize, 181  
    AudioCallbackRate, 181  
    AudioSampleRate, 181  
    ChangeAudioCallback, 181  
    controls, 184  
    Ctrl, 180  
    DaisyPatch, 180  
    DelayMs, 182  
    display, 184  
    DisplayControls, 182  
    encoder, 184  
    GATE\_IN\_LAST, 180  
    gate\_input, 184  
    gate\_output, 184  
    GateInput, 180  
    GetKnobValue, 182  
    Init, 182  
    midi, 184  
    ProcessAllControls, 182  
    ProcessAnalogControls, 182  
    ProcessDigitalControls, 183  
    seed, 185  
    SetAudioBlockSize, 183  
    SetAudioSampleRate, 183  
    StartAdc, 183  
    StartAudio, 183  
    StopAdc, 183  
    StopAudio, 184  
daisy::DaisyPetal, 193  
    ~DaisyPetal, 196  
    AudioBlockSize, 196  
    AudioCallbackRate, 196  
    AudioSampleRate, 197  
    ChangeAudioCallback, 197  
    ClearLeds, 197  
    DaisyPetal, 196  
    DelayMs, 197  
    encoder, 203  
    expression, 203  
    footswitch\_led, 203

FOOTSWITCH\_LED\_1, 195  
 FOOTSWITCH\_LED\_2, 195  
 FOOTSWITCH\_LED\_3, 195  
 FOOTSWITCH\_LED\_4, 195  
 FOOTSWITCH\_LED\_LAST, 195  
 FootswitchLed, 194  
 GetExpression, 199  
 GetKnobValue, 199  
 Init, 199  
 Knob, 195  
 knob, 203  
 KNOB\_1, 195  
 KNOB\_2, 195  
 KNOB\_3, 195  
 KNOB\_4, 195  
 KNOB\_5, 195  
 KNOB\_6, 195  
 KNOB\_LAST, 195  
 ProcessAllControls, 199  
 ProcessAnalogControls, 199  
 ProcessDigitalControls, 199  
 ring\_led, 203  
 RING\_LED\_1, 195  
 RING\_LED\_2, 195  
 RING\_LED\_3, 195  
 RING\_LED\_4, 195  
 RING\_LED\_5, 195  
 RING\_LED\_6, 195  
 RING\_LED\_7, 195  
 RING\_LED\_8, 195  
 RING\_LED\_LAST, 195  
 RingLed, 195  
 seed, 203  
 SetAudioBlockSize, 200  
 SetAudioSampleRate, 200  
 SetFootswitchLed, 200  
 SetRingLed, 200  
 StartAdc, 202  
 StartAudio, 202  
 StopAdc, 202  
 StopAudio, 202  
 Sw, 195  
 SW\_1, 196  
 SW\_2, 196  
 SW\_3, 196  
 SW\_4, 196  
 SW\_5, 196  
 SW\_6, 196  
 SW\_7, 196  
 SW\_LAST, 196  
 switches, 203  
 UpdateLeds, 202  
 daisy::DaisyPod, 204  
 AudioBlockSize, 205  
 AudioCallbackRate, 206  
 AudioSampleRate, 206  
 button1, 209  
 button2, 209  
 BUTTON\_2, 205  
 BUTTON\_LAST, 205  
 buttons, 209  
 ChangeAudioCallback, 206  
 ClearLeds, 206  
 DelayMs, 207  
 encoder, 209  
 GetKnobValue, 207  
 Init, 207  
 Knob, 205  
 knob1, 209  
 knob2, 209  
 KNOB\_2, 205  
 KNOB\_LAST, 205  
 knobs, 209  
 led1, 210  
 led2, 210  
 ProcessAllControls, 207  
 ProcessAnalogControls, 207  
 ProcessDigitalControls, 207  
 seed, 210  
 SetAudioBlockSize, 207  
 SetAudioSampleRate, 208  
 StartAdc, 208  
 StartAudio, 208  
 StopAdc, 208  
 StopAudio, 208  
 Sw, 205  
 UpdateLeds, 208  
 daisy::DaisySeed, 210  
 adc, 216  
 audio\_handle, 216  
 AudioBlockSize, 213  
 AudioCallbackRate, 213  
 AudioSampleRate, 213  
 BoardVersion, 211  
 ChangeAudioCallback, 213  
 CheckBoardVersion, 213  
 Configure, 214  
 DAISY\_SEED, 213  
 DAISY\_SEED\_1\_1, 213  
 DeInit, 214  
 DelayMs, 214  
 GetPin, 214  
 Init, 214  
 Print, 214  
 PrintLine, 215  
 sdram\_handle, 216  
 SetAudioBlockSize, 215  
 SetAudioSampleRate, 215  
 SetLed, 215  
 SetTestPoint, 215  
 StartAudio, 215, 216  
 StartLog, 216  
 StopAudio, 216  
 usb\_handle, 217  
 daisy::DaisyVersio, 217  
 AudioBlockSize, 218

AudioCallbackRate, 218  
AudioSampleRate, 218  
ChangeAudioCallback, 218, 219  
DelayMs, 219  
Gate, 219  
GetKnobValue, 219  
Init, 219  
ProcessAllControls, 220  
ProcessAnalogControls, 220  
SetAudioBlockSize, 220  
SetAudioSampleRate, 220  
SetLed, 220  
StartAdc, 220  
StartAudio, 220, 221  
StopAdc, 221  
StopAudio, 221  
SwitchPressed, 221  
UpdateLeds, 221  
daisy::Encoder, 225  
  Debounce, 226  
  FallingEdge, 226  
  Increment, 226  
  Init, 226  
  Pressed, 226  
  RisingEdge, 226  
  SetUpdateRate, 226  
  TimeHeldMs, 227  
daisy::FatFSInterface, 227  
  DelInit, 228  
  GetConfig, 228  
  GetMutableConfig, 228  
  GetSDFileSystem, 228  
  GetSDPath, 229  
  GetUSBFileSystem, 229  
  GetUSBPath, 229  
  Init, 229  
  Result, 228  
daisy::FatFSInterface::Config, 142  
  Media, 142  
daisy::FIFO< T, capacity >, 230  
  FIFO, 230  
  operator=, 231  
daisy::FIFOBase< T >, 231  
  Back, 232  
  Clear, 233  
  Contains, 233  
  CountEqualTo, 233  
  Front, 233  
  GetCapacity, 233  
  GetNumElements, 233  
  Insert, 234  
  IsEmpty, 234  
  IsFull, 234  
  operator=, 234  
  operator[], 234  
  PopFront, 235  
  PushBack, 235  
  Remove, 235  
    RemoveAllEqualTo, 235  
daisy::FixedCapStr< capacity, CharType >, 236  
daisy::FixedCapStrBase< CharType >, 236  
daisy::FullScreenItemMenu, 239  
  Draw, 239  
  Init, 239  
  SetOneBitGraphicsDisplayToDrawTo, 240  
daisy::GateIn, 240  
  ~GateIn, 241  
  GateIn, 241  
  Init, 241  
  State, 241  
  Trig, 241  
daisy::GPIO, 242  
  ANALOG, 243  
  GetConfig, 243  
  INPUT, 243  
  Mode, 242  
  NOPULL, 243  
  OUTPUT, 243  
  OUTPUT\_OD, 243  
  Pull, 243  
  PULLDOWN, 243  
  PULLUP, 243  
  Write, 243  
daisy::GPIO::Config, 142  
  Config, 143  
daisy::I2CHandle, 244  
  CallbackFunctionPtr, 245  
  Direction, 245  
  ERR, 245  
  GetConfig, 245  
  Init, 245  
  OK, 245  
  ReadDataAtAddress, 246  
  RECEIVE, 245  
  ReceiveBlocking, 246  
  ReceiveDma, 246  
  Result, 245  
  TRANSMIT, 245  
  TransmitBlocking, 247  
  TransmitDma, 247  
  WriteDataAtAddress, 248  
daisy::I2CHandle::Config, 143  
  address, 145  
  I2C\_1, 144  
  I2C\_100KHZ, 144  
  I2C\_1MHZ, 144  
  I2C\_2, 144  
  I2C\_3, 144  
  I2C\_4, 144  
  I2C\_400KHZ, 144  
  Mode, 144  
  mode, 145  
  periph, 145  
  Peripheral, 144  
  pin\_config, 145  
  scl, 145

sda, 145  
 Speed, 144  
 speed, 145  
**daisy::LcdHD44780**, 251  
 Clear, 251  
 Init, 251  
 Print, 252  
 PrintInt, 252  
 SetCursor, 252  
**daisy::LcdHD44780::Config**, 146  
**daisy::Led**, 253  
 Init, 253  
 Set, 254  
 SetSampleRate, 254  
 Update, 254  
**daisy::LedDriverPca9685**< numDrivers, persistent-  
 BufferContents >, 254  
 \_\_attribute\_\_, 256  
 DmaBuffer, 255  
 GetNumLeds, 256  
 Init, 256  
 SetAllTo, 257  
 SetAllToRaw, 257  
 SetLed, 257  
 SetLedRaw, 257  
 SwapBuffersAndTransmit, 258  
**daisy::LocalControlEvent**, 258  
 channel, 258  
 local\_control\_off, 258  
 local\_control\_on, 259  
**daisy::Logger**< dest >, 259  
**daisy::Logger**< LOGGER\_NONE >, 260  
**daisy::LoggerImpl**< dest >, 261  
 Init, 261  
 Transmit, 261  
**daisy::LoggerImpl**< LOGGER\_EXTERNAL >, 262  
 Init, 262  
 Transmit, 262  
 usb\_handle\_, 262  
**daisy::LoggerImpl**< LOGGER\_INTERNAL >, 263  
 Init, 263  
 Transmit, 263  
 usb\_handle\_, 264  
**daisy::LoggerImpl**< LOGGER\_SEMIHOST >, 264  
 Init, 264  
 Transmit, 264  
**daisy::MappedFloatValue**, 265  
 AppentToString, 266  
 Get, 267  
 GetAs0to1, 267  
 GetPtr, 267  
 lin, 266  
 log, 266  
 MappedFloatValue, 266  
 Mapping, 265  
 operator float, 267  
 operator=, 267  
 pow2, 266  
 ResetToDefault, 267  
 Set, 268  
 SetFrom0to1, 268  
 Step, 268  
**daisy::MappedIntValue**, 268  
 AppentToString, 269  
 Get, 270  
 GetAs0to1, 270  
 GetPtr, 270  
 MappedIntValue, 269  
 operator int, 270  
 operator=, 270  
 ResetToDefault, 270  
 Set, 271  
 SetFrom0to1, 271  
 Step, 271  
**daisy::MappedStringValue**, 271  
 AppentToString, 272  
 GetAs0to1, 273  
 GetIndex, 273  
 GetIndexPtr, 273  
 GetString, 273  
 MappedStringValue, 272  
 operator const char \*, 273  
 operator int, 273  
 operator=, 273  
 ResetToDefault, 274  
 SetFrom0to1, 274  
 SetIndex, 274  
 Step, 274  
**daisy::MappedValue**, 275  
 AppentToString, 275  
 GetAs0to1, 275  
 ResetToDefault, 275  
 SetFrom0to1, 276  
 Step, 276  
**daisy::MAX11300Driver**< Transport >, 276  
**daisy::MAX11300Driver**< Transport >::Config, 146  
**daisy::MidiEvent**, 278  
 AsChannelPressure, 278  
 AsControlChange, 279  
 AsNoteOff, 279  
 AsNoteOn, 279  
 AsPitchBend, 279  
 AsPolyphonicKeyPressure, 279  
 AsProgramChange, 279  
 channel, 279  
 data, 280  
 sysex\_data, 280  
 type, 280  
**daisy::MidiHandler**< Transport >, 280  
 HasEvents, 281  
 Init, 281  
 Listen, 281  
 Parse, 282  
 PopEvent, 282  
 SendMessage, 282  
 StartReceive, 282

daisy::MidiHandler< Transport >::Config, 146  
daisy::MidiUartTransport, 283  
daisy::MidiUartTransport::Config, 147  
daisy::MidiUsbTransport, 283  
daisy::MidiUsbTransport::Config, 147  
daisy::MonoModeOnEvent, 283  
    channel, 284  
    num\_channels, 284  
daisy::MTCQuarterFrameEvent, 284  
    message\_type, 285  
    value, 285  
daisy::NoteOffEvent, 285  
    channel, 285  
    note, 285  
    velocity, 286  
daisy::NoteOnEvent, 286  
    channel, 286  
    note, 286  
    velocity, 286  
daisy::OledDisplay< DisplayDriver >, 287  
    DrawPixel, 288  
    Fill, 288  
    Update, 288  
daisy::OledDisplay< DisplayDriver >::Config, 147  
daisy::OmniModeOffEvent, 289  
    channel, 289  
daisy::OmniModeOnEvent, 289  
    channel, 289  
daisy::OneBitGraphicsDisplay, 290  
    DrawArc, 291  
    DrawCircle, 291  
    DrawLine, 292  
    DrawPixel, 292  
    DrawRect, 292, 293  
    Fill, 293  
    SetCursor, 294  
    Update, 294  
    WriteChar, 294  
    WriteString, 295  
    WriteStringAligned, 295  
daisy::OneBitGraphicsDisplayImpl< ChildType >, 296  
    DrawArc, 297  
    DrawLine, 297  
    DrawRect, 298  
    WriteChar, 298  
    WriteString, 299  
    WriteStringAligned, 299  
daisy::Parameter, 300  
    ~Parameter, 301  
    CUBE, 300  
    Curve, 300  
    EXPONENTIAL, 300  
    Init, 301  
    LAST, 300  
    LINEAR, 300  
    LOGARITHMIC, 300  
    Parameter, 301  
    Process, 301  
        Value, 302  
daisy::patch\_sm::DaisyPatchSM, 185  
    A1, 193  
    AudioBlockSize, 187  
    AudioCallbackRate, 187  
    AudioSampleRate, 188  
    ChangeAudioCallback, 188  
    Delay, 188  
    GetAdcValue, 188  
    GetPin, 188  
    GetRandomFloat, 189  
    GetRandomValue, 189  
    Init, 189  
    PinBank, 187  
    Print, 189  
    PrintLine, 189  
    ProcessAllControls, 189  
    ProcessAnalogControls, 190  
    ProcessDigitalControls, 190  
    SetAudioBlockSize, 190  
    SetAudioSampleRate, 190  
    StartAdc, 190  
    StartAudio, 190  
    StartDac, 191  
    StartLog, 191  
    StopAdc, 191  
    StopAudio, 191  
    StopDac, 191  
    system, 193  
    user\_led, 193  
    ValidateQSPI, 191  
    ValidateSDRAM, 192  
    WriteCvOut, 192  
daisy::Pcm3060, 302  
    Init, 302  
daisy::PersistentStorage< SettingStruct >, 303  
    GetSettings, 304  
    GetState, 304  
    Init, 304  
    PersistentStorage, 304  
    RestoreDefaults, 305  
    Save, 305  
    State, 303  
daisy::Pin, 305  
    IsValid, 306  
    operator dsy\_gpio\_pin, 307  
    Pin, 306  
daisy::PitchBendEvent, 307  
    channel, 307  
    value, 307  
daisy::PolyModeOnEvent, 308  
    channel, 308  
daisy::PolyphonicKeyPressureEvent, 308  
daisy::PotMonitor< BackendType, numPots >, 309  
    GetBackend, 309  
    GetCurrentPotValue, 309  
    GetNumPotsMonitored, 310  
    Init, 310

IsMoving, 310  
 Process, 311  
**daisy::ProgramChangeEvent**, 311  
 channel, 311  
 program, 311  
**daisy::QSPIHandle**, 312  
 Delnit, 313  
 Erase, 313  
 EraseSector, 313  
 GetConfig, 314  
 GetData, 314  
 GetStatus, 314  
 Init, 314  
 Status, 312  
 Write, 315  
 WritePage, 315  
**daisy::QSPIHandle::Config**, 147  
 clk, 150  
 Device, 148  
 DEVICE\_LAST, 148  
 INDIRECT\_POLLING, 150  
 io0, 150  
 io1, 150  
 io2, 150  
 io3, 150  
 IS25LP064A, 148  
 IS25LP080D, 148  
 MEMORY\_MAPPED, 150  
 Mode, 148  
 ncs, 150  
**daisy::Random**, 316  
 Delnit, 316  
 GetFloat, 316  
 GetValue, 317  
 Init, 317  
 IsReady, 317  
**daisy::Rectangle**, 318  
**daisy::ResetAllControllersEvent**, 319  
 channel, 319  
 value, 319  
**daisy::RgbLed**, 319  
 Init, 320  
 Set, 320  
 SetBlue, 320  
 SetColor, 321  
 SetGreen, 321  
 SetRed, 321  
 Update, 321  
**daisy::RingBuffer**< T, 0 >, 327  
 capacity, 328  
 Flush, 328  
 ImmediateRead, 328  
 Init, 328  
 Overwrite, 329  
 Read, 329  
 readable, 329  
 writable, 330  
 Write, 330  
**daisy::RingBuffer**< T, size >, 322  
 Advance, 322  
 capacity, 323  
 Flush, 323  
 GetMutableBuffer, 323  
 ImmediateRead, 323  
 Init, 324  
 isEmpty, 324  
 Overwrite, 324  
 Read, 326  
 readable, 326  
 Swallow, 326  
 writable, 326  
 Write, 327  
**daisy::SaiHandle**, 330  
 CallbackFunctionPtr, 331  
 Delnit, 332  
 GetBlockRate, 332  
 GetBlockSize, 332  
 GetConfig, 332  
 GetOffset, 332  
 GetSampleRate, 332  
 Init, 333  
 Result, 332  
 StartDma, 333  
 StopDma, 333  
**daisy::SaiHandle::Config**, 151  
 BitDepth, 151  
 Direction, 151  
 Peripheral, 152  
 SampleRate, 152  
 Sync, 152  
**daisy::ScopedIrqBlocker**, 333  
**daisy::SdmmcHandler**, 334  
 BITS\_1, 334  
 BITS\_4, 334  
 BusWidth, 334  
 FAST, 335  
 Init, 335  
 MEDIUM\_SLOW, 335  
 Result, 335  
 SLOW, 335  
 Speed, 335  
 STANDARD, 335  
 VERY\_FAST, 335  
**daisy::SdmmcHandler::Config**, 152  
 clock\_powersave, 153  
 Defaults, 152  
**daisy::seed**, 109  
 A0, 110  
 A12, 110  
 D0, 111  
 D31, 111  
**daisy::ShiftRegister4021**< num\_daisychained, num\_parallel >, 336  
 Init, 337  
 State, 337  
 Update, 337

daisy::ShiftRegister4021< num\_daisychained, num\_parallel >::Config, 153  
clk, 153  
data, 154  
latch, 154  
daisy::SongPositionPointerEvent, 339  
position, 340  
daisy::SongSelectEvent, 340  
song, 340  
daisy::SpiHandle, 342  
BlockingReceive, 343  
BlockingTransmit, 343  
BlockingTransmitAndReceive, 344  
CallbackFunctionPtr, 342  
CheckError, 344  
DmaDirection, 342  
DmaReceive, 344  
DmaTransmit, 345  
ERR, 343  
GetConfig, 345  
Init, 345  
OK, 343  
Result, 343  
RX, 343  
TX, 343  
daisy::SpiHandle::Config, 154  
miso, 155  
mosi, 155  
nss, 155  
sclk, 155  
daisy::SSD130x4WireSpiTransport, 346  
daisy::SSD130x4WireSpiTransport::Config, 156  
dc, 156  
reset, 156  
daisy::SSD130xDriver< width, height, Transport >, 346  
Update, 347  
daisy::SSD130xDriver< width, height, Transport >::Config, 156  
daisy::SSD130xl2CTransport, 347  
daisy::SSD130xl2CTransport::Config, 157  
daisy::Stack< T, capacity >, 347  
operator=, 349  
Stack, 348  
daisy::StackBase< T >, 349  
Clear, 350  
Contains, 350  
CountEqualTo, 350  
GetCapacity, 350  
GetNumElements, 350  
Insert, 350  
IsEmpty, 351  
IsFull, 351  
operator=, 351  
operator[], 351  
PopBack, 351  
PushBack, 352  
Remove, 352  
RemoveAllEqualTo, 352  
daisy::Switch, 352  
Debounce, 354  
FallingEdge, 354  
Init, 354, 355  
Polarity, 353  
POLARITY\_INVERTED, 353  
POLARITY\_NORMAL, 353  
Pressed, 355  
Pull, 353  
PULL\_DOWN, 354  
PULL\_NONE, 354  
PULL\_UP, 354  
RawState, 355  
RisingEdge, 356  
SetUpdateRate, 356  
TimeHeldMs, 356  
Type, 354  
TYPE\_MOMENTARY, 354  
TYPE\_TOGGLE, 354  
daisy::Switch3, 357  
daisy::System, 357  
DeInit, 358  
Delay, 358  
DelayTicks, 358  
DelayUs, 359  
GetConfig, 359  
GetHClkFreq, 359  
GetMemoryRegion, 359  
GetNow, 359  
GetPClk1Freq, 360  
GetPClk2Freq, 360  
GetProgramMemoryRegion, 360  
GetSysClkFreq, 360  
GetTick, 360  
GetTickFreq, 360  
GetUs, 361  
Init, 361  
JumpToQspi, 361  
kQspiBootloaderOffset, 361  
MemoryRegion, 358  
ResetToBootloader, 361  
daisy::System::Config, 157  
Boost, 158  
Defaults, 158  
SysClkFreq, 157  
daisy::SystemExclusiveEvent, 362  
data, 362  
daisy::TimerHandle, 362  
DeInit, 364  
DelayMs, 364  
DelayTick, 364  
DelayUs, 364  
GetConfig, 364  
GetFreq, 364  
GetMs, 365  
GetTick, 365  
GetUs, 365  
Init, 365

Result, 364  
 SetPeriod, 365  
 SetPrescaler, 365  
 Start, 366  
 Stop, 366  
**daisy::TimerHandle::Config**, 158  
 CounterDir, 158  
 Peripheral, 158  
 TIM\_2, 159  
 TIM\_3, 159  
 TIM\_4, 159  
 TIM\_5, 159  
**daisy::UartHandler**, 366  
 CheckError, 367  
 ERR, 367  
 FlushRx, 367  
 GetConfig, 367  
 Init, 368  
 OK, 367  
 PollReceive, 368  
 PollTx, 368  
 PopRx, 369  
 Readable, 369  
 Result, 367  
 RxActive, 369  
 StartRx, 369  
**daisy::UartHandler::Config**, 159  
 pin\_config, 159  
 rx, 160  
 tx, 160  
**daisy::UI**, 370  
 ClosePage, 370  
 GetPrimaryOneBitGraphicsDisplayId, 370  
 GetSpecialControlIds, 370  
 Init, 370  
 invalidCanvasId, 371  
 Mute, 371  
 OpenPage, 371  
 Process, 371  
**daisy::UI::SpecialControlIds**, 340  
 menuEncoderId, 341  
 valueEncoderId, 341  
 valuePotId, 341  
**daisy::UiCanvasDescriptor**, 372  
 ClearFuncPtr, 372  
 FlushFuncPtr, 372  
 handle\_, 372  
 id\_, 372  
 screenSaverTimeOut, 373  
 updateRateMs\_, 373  
**daisy::UIEventQueue**, 373  
 \_\_attribute\_\_, 373  
 AddButtonPressed, 374  
 AddButtonReleased, 375  
 AddEncoderActivityChanged, 375  
 AddEncoderTurned, 375  
 AddPotActivityChanged, 375  
 AddPotMoved, 375  
 GetAndRemoveNextEvent, 375  
 invalidButtonId, 376  
 invalidEncoderId, 376  
 invalidPotId, 376  
 IsQueueEmpty, 376  
**daisy::UiPage**, 376  
 Close, 377  
 Draw, 377  
 GetParentUI, 377, 378  
 IsActive, 378  
 IsOpaque, 378  
 OnArrowButton, 378  
 OnButton, 378  
 OnCancelButton, 379  
 OnEncoderActivityChanged, 379  
 OnEncoderTurned, 381  
 OnFocusGained, 381  
 OnFocusLost, 381  
 OnFunctionButton, 381  
 OnHide, 382  
 OnMenuEncoderActivityChanged, 382  
 OnMenuEncoderTurned, 382  
 OnOkayButton, 383  
 OnPotActivityChanged, 383  
 OnPotMoved, 384  
 OnShow, 384  
 OnValueEncoderActivityChanged, 384  
 OnValueEncoderTurned, 385  
 OnValuePotActivityChanged, 385  
 OnValuePotMoved, 385  
**daisy::USBHostHandle**, 391  
 Deinit, 392  
 GetPresent, 392  
 GetReady, 392  
 Init, 392  
 Process, 393  
**daisy::USBHostHandle::Config**, 160  
**daisy::VoctCalibration**, 393  
 GetData, 393  
 ProcessInput, 394  
 Record, 394  
 SetData, 394  
**daisy::WAV\_FormatTypeDef**, 395  
 AudioFormat, 395  
 BitPerSample, 395  
 BlockAlign, 395  
 ByteRate, 395  
 ChunkId, 396  
 FileFormat, 396  
 FileSize, 396  
 NbrChannels, 396  
 SampleRate, 396  
 SubChunk1ID, 396  
 SubChunk1Size, 396  
 SubChunk2ID, 396  
 SubChunk2Size, 397  
**daisy::WaveTableLoader**, 397  
 GetTable, 397

Import, 398  
Init, 398  
SetWaveTableInfo, 398  
**daisy::WavFileInfo**, 398  
name, 399  
raw\_data, 399  
**daisy::WavPlayer**, 399  
Close, 400  
GetCurrentFile, 400  
GetLooping, 400  
GetNumberFiles, 400  
Init, 400  
Open, 400  
Prepare, 401  
Restart, 401  
SetLooping, 401  
Stream, 401  
**daisy::WavWriter< transfer\_size >**, 401  
BufferState, 403  
GetLengthSamps, 403  
GetLengthSeconds, 403  
Init, 403  
IsRecording, 403  
OpenFile, 404  
Result, 403  
Sample, 404  
SaveFile, 404  
Write, 404  
**daisy::WavWriter< transfer\_size >::Config**, 160  
**daisy::Wm8731**, 405  
Init, 406  
Result, 405  
**daisy::Wm8731::Config**, 161  
csb\_pin\_state, 162  
Defaults, 162  
Format, 161  
lr\_swap, 162  
mcu\_is\_master, 162  
WordLength, 162  
**DAISY\_SEED**  
daisy::DaisySeed, 213  
**DAISY\_SEED\_1\_1**  
daisy::DaisySeed, 213  
**DaisyPatch**  
daisy::DaisyPatch, 180  
**DaisyPetal**  
daisy::DaisyPetal, 196  
**data**  
daisy::MidiEvent, 280  
daisy::ShiftRegister4021< num\_daisychained,  
    num\_parallel >::Config, 154  
daisy::SystemExclusiveEvent, 362  
**FontDef**, 238  
**dc**  
daisy::SSD130x4WireSpiTransport::Config, 156  
**Debounce**  
daisy::Encoder, 226  
daisy::Switch, 354  
**Defaults**  
Dac, 92  
daisy::SdmmcHandler::Config, 152  
daisy::System::Config, 158  
daisy::Wm8731::Config, 162  
**DelInit**  
daisy::Ak4556, 126  
daisy::AudioHandle, 133  
daisy::DaisySeed, 214  
daisy::FatFSInterface, 228  
daisy::QSPIHandle, 313  
daisy::Random, 316  
daisy::SaiHandle, 332  
daisy::System, 358  
daisy::TimerHandle, 364  
UsbHandle, 388  
**Deinit**  
daisy::USBHostHandle, 392  
**Delay**  
daisy::patch\_sm::DaisyPatchSM, 188  
daisy::System, 358  
**DelayMs**  
daisy::DaisyField, 174  
daisy::DaisyPatch, 182  
daisy::DaisyPetal, 197  
daisy::DaisyPod, 207  
daisy::DaisySeed, 214  
daisy::DaisyVersio, 219  
daisy::TimerHandle, 364  
**DelayTick**  
daisy::TimerHandle, 364  
**DelayTicks**  
daisy::System, 358  
**DelayUs**  
daisy::System, 359  
daisy::TimerHandle, 364  
**DEVICE**, 42  
**Device**  
daisy::QSPIHandle::Config, 148  
**DEVICE\_LAST**  
daisy::QSPIHandle::Config, 148  
**Direction**  
daisy::I2CHandle, 245  
daisy::SaiHandle::Config, 151  
**display**  
daisy::DaisyPatch, 184  
**DisplayControls**  
daisy::DaisyPatch, 182  
**DMA\_BUFFER\_MEM\_SECTION**  
UTILITY, 76  
**DmaBuffer**  
daisy::LedDriverPca9685< numDrivers, persistent-  
    BufferContents >, 255  
**DmaDirection**  
daisy::SpiHandle, 342  
**DmaReceive**  
daisy::SpiHandle, 344  
**DmaTransmit**

daisy::SpiHandle, 345  
 down  
 daisy, 105  
 Draw  
 daisy::AbstractMenu::CustomItem, 166  
 daisy::FullScreenItemMenu, 239  
 daisy::UiPage, 377  
 DrawArc  
 daisy::OneBitGraphicsDisplay, 291  
 daisy::OneBitGraphicsDisplayImpl< ChildType >, 297  
 DrawCircle  
 daisy::OneBitGraphicsDisplay, 291  
 DrawLine  
 daisy::OneBitGraphicsDisplay, 292  
 daisy::OneBitGraphicsDisplayImpl< ChildType >, 297  
 DrawPixel  
 daisy::OledDisplay< DisplayDriver >, 288  
 daisy::OneBitGraphicsDisplay, 292  
 DrawRect  
 daisy::OneBitGraphicsDisplay, 292, 293  
 daisy::OneBitGraphicsDisplayImpl< ChildType >, 298  
 dsy\_dma\_clear\_cache\_for\_buffer  
 SYSTEM, 41  
 dsy\_dma\_deinit  
 SYSTEM, 41  
 dsy\_dma\_init  
 SYSTEM, 41  
 dsy\_dma\_invalidate\_cache\_for\_buffer  
 SYSTEM, 41  
 dsy\_get\_unique\_id  
 UTILITY, 84  
 dsy\_gpio, 221  
 mode, 222  
 pin, 222  
 pull, 222  
 dsy\_gpio\_deinit  
 OTHER, 39  
 dsy\_gpio\_init  
 OTHER, 39  
 DSY\_GPIO\_LAST  
 UTILITY, 80  
 dsy\_gpio\_mode  
 OTHER, 37  
 DSY\_GPIO\_MODE\_ANALOG  
 OTHER, 37  
 DSY\_GPIO\_MODE\_INPUT  
 OTHER, 37  
 DSY\_GPIO\_MODE\_LAST  
 OTHER, 37  
 DSY\_GPIO\_MODE\_OUTPUT\_OD  
 OTHER, 37  
 DSY\_GPIO\_MODE\_OUTPUT\_PP  
 OTHER, 37  
 DSY\_GPIO\_NOPULL  
 OTHER, 39  
 dsy\_gpio\_pin, 222  
 pin, 223  
 port, 223  
 dsy\_gpio\_port  
 UTILITY, 79  
 dsy\_gpio\_pull  
 OTHER, 37  
 DSY\_GPIO\_PULLDOWN  
 OTHER, 39  
 DSY\_GPIO\_PULLUP  
 OTHER, 39  
 dsy\_gpio\_read  
 OTHER, 39  
 dsy\_gpio\_toggle  
 OTHER, 40  
 dsy\_gpio\_write  
 OTHER, 40  
 DSY\_GPIOA  
 UTILITY, 80  
 DSY\_GPIOB  
 UTILITY, 80  
 DSY\_GPIOC  
 UTILITY, 80  
 DSY\_GPIOD  
 UTILITY, 80  
 DSY\_GPIOE  
 UTILITY, 80  
 DSY\_GPIOF  
 UTILITY, 80  
 DSY\_GPIOG  
 UTILITY, 80  
 DSY\_GPIOH  
 UTILITY, 80  
 DSY\_GPIOI  
 UTILITY, 80  
 DSY\_GPIOJ  
 UTILITY, 80  
 DSY\_GPIOK  
 UTILITY, 80  
 dsy\_hal\_map\_get\_pin  
 UTILITY, 85  
 dsy\_hal\_map\_get\_port  
 UTILITY, 85  
 dsy\_hal\_map\_gpio\_clk\_enable  
 UTILITY, 85  
 dsy\_i2c\_global\_init  
 daisy, 108  
 dsy\_pin  
 UTILITY, 86  
 dsy\_pin\_cmp  
 UTILITY, 86  
 DSY\_SD\_CardInfoTypeDef, 223  
 BlockNbr, 224  
 BlockSize, 224  
 CardSpeed, 224  
 CardType, 224  
 CardVersion, 224  
 Class, 224

LogBlockNbr, 224  
LogBlockSize, 225  
RelCardAdd, 225  
DSY\_SDRAM\_BSS  
    SDRAM, 69  
DSY\_SDRAM\_DATA  
    SDRAM, 69  
dsy\_spi\_global\_init  
    SERIAL, 36  
DTCM\_MEM\_SECTION  
    UTILITY, 76  
DUAL\_INOUT\_FAST\_READ\_CMD  
    FLASH, 47, 48  
DUAL\_INOUT\_FAST\_READ\_DTR\_CMD  
    FLASH, 48  
DUAL\_OUT\_FAST\_READ\_CMD  
    FLASH, 48  
  
encoder  
    daisy::DaisyPatch, 184  
    daisy::DaisyPetal, 203  
    daisy::DaisyPod, 209  
ENTER\_DEEP\_POWER\_DOWN  
    FLASH, 48  
ENTER\_QUAD\_CMD  
    FLASH, 48, 49  
Erase  
    daisy::QSPIHandle, 313  
EraseSector  
    daisy::QSPIHandle, 313  
ERR  
    Dac, 92  
    daisy::I2CHandle, 245  
    daisy::SpiHandle, 343  
    daisy::UartHandler, 367  
    SdramHandle, 336  
EXIT\_DEEP\_POWER\_DOWN  
    FLASH, 49  
EXIT\_QUAD\_CMD  
    FLASH, 49  
EXPONENTIAL  
    daisy::Parameter, 300  
expression  
    daisy::DaisyPetal, 203  
EXT\_CHIP\_ERASE\_CMD  
    FLASH, 49  
EXT\_PROG\_ERASE\_RESUME\_CMD  
    FLASH, 49, 50  
EXT\_PROG\_ERASE\_SUSPEND\_CMD  
    FLASH, 50  
EXT\_QUAD\_IN\_FAST\_PROG\_CMD  
    FLASH, 50  
EXT\_QUAD\_IN\_PAGE\_PROG\_CMD  
    FLASH, 50  
EXT\_WRITE\_READ\_PARAM\_REG\_CMD  
    FLASH, 50  
EXTERNAL, 29  
    AppendNewLine, 32  
FLT\_FMT, 30  
    FLT\_FMT3, 30  
    FLT\_VAR, 31  
    FLT\_VAR3, 31  
    impl\_, 34  
    Logger, 32  
    LOGGER\_BUFFER, 31  
    LOGGER\_NEWLINE, 31  
    LOGGER\_SYNC\_IN, 32  
    LoggerConsts, 32  
    NewLineSeqLength, 33  
    pc\_sync\_, 34  
    PPCAT, 31  
    PPCAT\_NX, 31  
    Print, 33  
    PrintLine, 33  
    PrintLineV, 33  
    PrintV, 33  
    StartLog, 33  
    STRINGIZE, 32  
    STRINGIZE\_NX, 32  
    TransmitBuf, 34  
    TransmitSync, 34  
    tx\_buff\_, 34  
    tx\_ptr\_, 35  
Externals, 98  
  
f2s16  
    UTILITY, 86  
F2S16\_SCALE  
    UTILITY, 76  
f2s24  
    UTILITY, 86  
F2S24\_SCALE  
    UTILITY, 76  
f2s32  
    UTILITY, 87  
F2S32\_SCALE  
    UTILITY, 76  
f2s8  
    UTILITY, 87  
F2S8\_SCALE  
    UTILITY, 76  
f2u8  
    UTILITY, 87  
F2U8\_SCALE  
    UTILITY, 76  
FallingEdge  
    daisy::Encoder, 226  
    daisy::Switch, 354  
FAST  
    daisy::SdmmcHandler, 335  
FAST\_READ\_CMD  
    FLASH, 50, 51  
FAST\_READ\_DTR\_CMD  
    FLASH, 51  
FBIPMAX  
    UTILITY, 77  
FBIPMIN  
    UTILITY, 77

FEEDBACK, 29  
 ff\_free  
     ffconf.h, 414  
 ff\_malloc  
     ffconf.h, 414  
 ffconf.h  
     \_CODE\_PAGE, 408  
     \_FFCNF, 408  
     \_FS\_EXFAT, 408  
     \_FS\_LOCK, 409  
     \_FS\_MINIMIZE, 409  
     \_FS\_NOFSINFO, 409  
     \_FS\_NORTC, 409  
     \_FS\_READONLY, 409  
     \_FS\_REENTRANT, 409  
     \_FS\_RPATH, 410  
     \_FS\_TIMEOUT, 410  
     \_FS\_TINY, 410  
     \_LFN\_UNICODE, 410  
     \_MAX\_LFN, 410  
     \_MAX\_SS, 410  
     \_MIN\_SS, 411  
     \_MULTI\_PARTITION, 411  
     \_NORTC\_MDAY, 411  
     \_NORTC\_MON, 411  
     \_NORTC\_YEAR, 411  
     \_STRF\_ENCODE, 412  
     \_STR\_VOLUME\_ID, 411  
     \_SYNC\_t, 412  
     \_USE\_CHMOD, 412  
     \_USE\_EXPAND, 412  
     \_USE\_FASTSEEK, 412  
     \_USE\_FIND, 412  
     \_USE\_FORWARD, 413  
     \_USE\_LABEL, 413  
     \_USE\_LFN, 413  
     \_USE\_MKFS, 413  
     \_USE\_STRFUNC, 413  
     \_USE\_TRIM, 413  
     \_VOLUMES, 414  
     \_VOLUME\_STRS, 413  
 ff\_free, 414  
 ff\_malloc, 414  
**FIFO**  
     daisy::FIFO< T, capacity >, 230  
**FileFormat**  
     daisy::WAV\_FormatTypeDef, 396  
**FileSize**  
     daisy::WAV\_FormatTypeDef, 396  
**Fill**  
     daisy::OledDisplay< DisplayDriver >, 288  
     daisy::OneBitGraphicsDisplay, 293  
**FLASH**, 43  
     BLOCK\_ERASE\_32K\_CMD, 47  
     BLOCK\_ERASE\_CMD, 47  
     CHIP\_ERASE\_CMD, 47  
     CLEAR\_EXT\_READ\_PARAM\_CMD, 47  
     DUAL\_INOUT\_FAST\_READ\_CMD, 47, 48  
 DUAL\_INOUT\_FAST\_READ\_DTR\_CMD, 48  
 DUAL\_OUT\_FAST\_READ\_CMD, 48  
 ENTER\_DEEP\_POWER\_DOWN, 48  
 ENTER\_QUAD\_CMD, 48, 49  
 EXIT\_DEEP\_POWER\_DOWN, 49  
 EXIT\_QUAD\_CMD, 49  
 EXT\_CHIP\_ERASE\_CMD, 49  
 EXT\_PROG\_ERASE\_RESUME\_CMD, 49, 50  
 EXT\_PROG\_ERASE\_SUSPEND\_CMD, 50  
 EXT\_QUAD\_IN\_FAST\_PROG\_CMD, 50  
 EXT\_QUAD\_IN\_PAGE\_PROG\_CMD, 50  
 EXT\_WRITE\_READ\_PARAM\_REG\_CMD, 50  
 FAST\_READ\_CMD, 50, 51  
 FAST\_READ\_DTR\_CMD, 51  
 INFO\_ROW\_ERASE\_CMD, 51  
 INFO\_ROW\_PROGRAM\_CMD, 51  
 INFO\_ROW\_READ\_CMD, 51, 52  
 IS25LP064A\_EAR\_HIGHEST\_SE, 52  
 IS25LP064A\_EAR\_LOWEST\_SEG, 52  
 IS25LP064A\_EAR\_SECOND\_SEG, 52  
 IS25LP064A\_EAR\_THIRD\_SEG, 52  
 IS25LP064A\_EVCR\_DTRP, 52  
 IS25LP064A\_EVCR\_DUAL, 52  
 IS25LP064A\_EVCR\_ODS, 52  
 IS25LP064A\_EVCR\_QUAD, 53  
 IS25LP064A\_EVCR\_RH, 53  
 IS25LP064A\_FSR\_ERERR, 53  
 IS25LP064A\_FSR\_ERSUS, 53  
 IS25LP064A\_FSR\_NBADDR, 53  
 IS25LP064A\_FSR\_PGERR, 53  
 IS25LP064A\_FSR\_PGSUS, 53  
 IS25LP064A\_FSR\_PRERR, 53  
 IS25LP064A\_FSR\_READY, 54  
 IS25LP064A\_NVCR\_DTRP, 54  
 IS25LP064A\_NVCR\_DUAL, 54  
 IS25LP064A\_NVCR\_NB\_DUMMY, 54  
 IS25LP064A\_NVCR\_NBADDR, 54  
 IS25LP064A\_NVCR\_ODS, 54  
 IS25LP064A\_NVCR\_QUAB, 54  
 IS25LP064A\_NVCR\_RH, 54  
 IS25LP064A\_NVCR\_SEGMENT, 55  
 IS25LP064A\_NVCR\_XIP, 55  
 IS25LP064A\_SR\_QE, 55  
 IS25LP064A\_SR\_SRREN, 55  
 IS25LP064A\_SR\_WIP, 55  
 IS25LP064A\_SR\_WREN, 55  
 IS25LP064A\_VCR\_NB\_DUMMY, 55  
 IS25LP064A\_VCR\_WRAP, 56  
 IS25LP064A\_VCR\_XIP, 56  
 IS25LP080D\_EAR\_HIGHEST\_SE, 56  
 IS25LP080D\_EAR\_LOWEST\_SEG, 56  
 IS25LP080D\_EAR\_SECOND\_SEG, 56  
 IS25LP080D\_EAR\_THIRD\_SEG, 56  
 IS25LP080D\_EVCR\_DTRP, 56  
 IS25LP080D\_EVCR\_DUAL, 56  
 IS25LP080D\_EVCR\_ODS, 57  
 IS25LP080D\_EVCR\_QUAD, 57  
 IS25LP080D\_EVCR\_RH, 57

IS25LP080D\_FSR\_ERERR, 57  
IS25LP080D\_FSR\_ERSUS, 57  
IS25LP080D\_FSR\_NBADDR, 57  
IS25LP080D\_FSR\_PGERR, 57  
IS25LP080D\_FSR\_PGSUS, 57  
IS25LP080D\_FSR\_PRERR, 58  
IS25LP080D\_FSR\_READY, 58  
IS25LP080D\_NVCR\_DTRP, 58  
IS25LP080D\_NVCR\_DUAL, 58  
IS25LP080D\_NVCR\_NB\_DUMMY, 58  
IS25LP080D\_NVCR\_NBADDR, 58  
IS25LP080D\_NVCR\_ODS, 58  
IS25LP080D\_NVCR\_QUAB, 58  
IS25LP080D\_NVCR\_RH, 59  
IS25LP080D\_NVCR\_SEGMENT, 59  
IS25LP080D\_NVCR\_XIP, 59  
IS25LP080D\_SR\_QE, 59  
IS25LP080D\_SR\_SRWREN, 59  
IS25LP080D\_SR\_WIP, 59  
IS25LP080D\_SR\_WREN, 59  
IS25LP080D\_VCR\_NB\_DUMMY, 60  
IS25LP080D\_VCR\_WRAP, 60  
IS25LP080D\_VCR\_XIP, 60  
MULTIPLE\_IO\_READ\_ID\_CMD, 60  
NO\_OP, 60  
PAGE\_PROG\_CMD, 60, 61  
PROG\_ERASE\_RESUME\_CMD, 61  
PROG\_ERASE\_SUSPEND\_CMD, 61  
QUAD\_IN\_FAST\_PROG\_CMD, 61  
QUAD\_IN\_PAGE\_PROG\_CMD, 62  
QUAD\_INOUT\_FAST\_READ\_CMD, 62  
QUAD\_INOUT\_FAST\_READ\_DTR\_CMD, 62  
QUAD\_OUT\_FAST\_READ\_CMD, 62  
READ\_CMD, 63  
READ\_EXT\_READ\_PARAM\_CMD, 63  
READ\_FUNCTION\_REGISTER, 63  
READ\_ID\_CMD, 63  
READ\_ID\_CMD2, 63, 64  
READ\_MANUFACT\_AND\_ID, 64  
READ\_READ\_PARAM\_REG\_CMD, 64  
READ\_SERIAL\_FLASH\_DISCO\_PARAM\_CMD,  
  64  
READ\_STATUS\_REG\_CMD, 64  
READ\_UNIQUE\_ID, 65  
RESET\_ENABLE\_CMD, 65  
RESET\_MEMORY\_CMD, 65  
SECTOR\_ERASE\_CMD, 65  
SECTOR\_ERASE\_QPI\_CMD, 66  
SECTOR\_LOCK, 66  
SECTOR\_UNLOCK, 66  
WRITE\_DISABLE\_CMD, 66  
WRITE\_ENABLE\_CMD, 67  
WRITE\_EXT\_NV\_READ\_PARAM\_REG\_CMD, 67  
WRITE\_EXT\_READ\_PARAM\_REG\_CMD, 67  
WRITE\_FUNCTION\_REGISTER, 67  
WRITE\_NV\_READ\_PARAM\_REG\_CMD, 67  
WRITE\_READ\_PARAM\_REG\_CMD, 67, 68  
WRITE\_STATUS\_REG\_CMD, 68

FLT\_FMT  
  EXTERNAL, 30  
FLT\_FMT3  
  EXTERNAL, 30  
FLT\_VAR  
  EXTERNAL, 31  
FLT\_VAR3  
  EXTERNAL, 31  
Flush  
  daisy::RingBuffer< T, 0 >, 328  
  daisy::RingBuffer< T, size >, 323  
FlushFuncPtr  
  daisy::UiCanvasDescriptor, 372  
FlushRx  
  daisy::UartHandler, 367  
Font\_11x18  
  UTILITY, 89  
Font\_16x26  
  UTILITY, 89  
Font\_6x8  
  UTILITY, 90  
Font\_7x10  
  UTILITY, 90  
FontDef, 237  
  data, 238  
  FontHeight, 238  
  FontWidth, 238  
FontHeight  
  FontDef, 238  
FontWidth  
  FontDef, 238  
footswitch\_led  
  daisy::DaisyPetal, 203  
FOOTSWITCH\_LED\_1  
  daisy::DaisyPetal, 195  
FOOTSWITCH\_LED\_2  
  daisy::DaisyPetal, 195  
FOOTSWITCH\_LED\_3  
  daisy::DaisyPetal, 195  
FOOTSWITCH\_LED\_4  
  daisy::DaisyPetal, 195  
FOOTSWITCH\_LED\_LAST  
  daisy::DaisyPetal, 195  
FootswitchLed  
  daisy::DaisyPetal, 194  
Format  
  daisy::Wm8731::Config, 161  
Front  
  daisy::FIFOBase< T >, 233  
FS\_BOTH  
  UsbHandle, 388  
FS\_EXTERNAL  
  UsbHandle, 388  
FS\_INTERNAL  
  UsbHandle, 388  
Gate  
  daisy::DaisyVersio, 219  
GATE\_IN\_LAST

daisy::DaisyPatch, 180  
gate\_input  
    daisy::DaisyPatch, 184  
gate\_output  
    daisy::DaisyPatch, 184  
GateIn  
    daisy::GateIn, 241  
GateInput  
    daisy::DaisyPatch, 180  
Get  
    daisy::AdcHandle, 123  
    daisy::MappedFloatValue, 267  
    daisy::MappedIntValue, 270  
GetAdcValue  
    daisy::patch\_sm::DaisyPatchSM, 188  
GetAndRemoveNextEvent  
    daisy::UiEventQueue, 375  
GetAs0to1  
    daisy::MappedFloatValue, 267  
    daisy::MappedIntValue, 270  
    daisy::MappedStringValue, 273  
    daisy::MappedValue, 275  
GetAvgCpuLoad  
    daisy::CpuLoadMeter, 164  
GetBackend  
    daisy::ButtonMonitor< BackendType, numButtons >, 135  
    daisy::PotMonitor< BackendType, numPots >, 309  
GetBlockRate  
    daisy::SaiHandle, 332  
GetBlockSize  
    daisy::SaiHandle, 332  
GetCapacity  
    daisy::FIFOBase< T >, 233  
    daisy::StackBase< T >, 350  
GetChannels  
    daisy::AudioHandle, 133  
GetConfig  
    daisy::AudioHandle, 133  
    daisy::FatFSInterface, 228  
    daisy::GPIO, 243  
    daisy::I2CHandle, 245  
    daisy::QSPIHandle, 314  
    daisy::SaiHandle, 332  
    daisy::SpiHandle, 345  
    daisy::System, 359  
    daisy::TimerHandle, 364  
    daisy::UartHandler, 367  
GetCurrentFile  
    daisy::WavPlayer, 400  
GetCurrentPotValue  
    daisy::PotMonitor< BackendType, numPots >, 309  
GetCv  
    daisy::DaisyField, 174  
GetCvValue  
    daisy::DaisyField, 175  
GetData  
    daisy::QSPIHandle, 314  
daisy::VoctCalibration, 393  
GetExpression  
    daisy::DaisyPetal, 199  
GetFloat  
    daisy::AdcHandle, 123  
    daisy::Random, 316  
GetFreq  
    daisy::TimerHandle, 364  
GetHClkFreq  
    daisy::System, 359  
GetIndex  
    daisy::MappedStringValue, 273  
GetIndexPtr  
    daisy::MappedStringValue, 273  
GetKnob  
    daisy::DaisyField, 175  
GetKnobValue  
    daisy::DaisyField, 175  
    daisy::DaisyPatch, 182  
    daisy::DaisyPetal, 199  
    daisy::DaisyPod, 207  
    daisy::DaisyVersio, 219  
GetLengthSamps  
    daisy::WavWriter< transfer\_size >, 403  
GetLengthSeconds  
    daisy::WavWriter< transfer\_size >, 403  
GetLooping  
    daisy::WavPlayer, 400  
GetMaxCpuLoad  
    daisy::CpuLoadMeter, 164  
GetMemoryRegion  
    daisy::System, 359  
GetMinCpuLoad  
    daisy::CpuLoadMeter, 164  
GetMs  
    daisy::TimerHandle, 365  
GetMutableBuffer  
    daisy::RingBuffer< T, size >, 323  
GetMutableConfig  
    daisy::FatFSInterface, 228  
GetMux  
    daisy::AdcHandle, 124  
GetMuxFloat  
    daisy::AdcHandle, 124  
GetMuxPtr  
    daisy::AdcHandle, 124  
GetNow  
    daisy::System, 359  
GetNumberFiles  
    daisy::WavPlayer, 400  
GetNumButtonsMonitored  
    daisy::ButtonMonitor< BackendType, numButtons >, 136  
GetNumElements  
    daisy::FIFOBase< T >, 233  
    daisy::StackBase< T >, 350  
GetNumLeds

daisy::LedDriverPca9685< numDrivers, persistent-BufferContents >, 256  
GetNumPotsMonitored  
daisy::PotMonitor< BackendType, numPots >, 310  
GetOffset  
daisy::SaiHandle, 332  
GetParentUI  
daisy::UiPage, 377, 378  
GetPClk1Freq  
daisy::System, 360  
GetPClk2Freq  
daisy::System, 360  
GetPin  
daisy::DaisySeed, 214  
daisy::patch\_sm::DaisyPatchSM, 188  
GetPresent  
daisy::USBHostHandle, 392  
GetPrimaryOneBitGraphicsDisplayId  
daisy::UI, 370  
GetProgramMemoryRegion  
daisy::System, 360  
GetPtr  
daisy::AdcHandle, 125  
daisy::MappedFloatValue, 267  
daisy::MappedIntValue, 270  
GetRandomFloat  
daisy::patch\_sm::DaisyPatchSM, 189  
GetRandomValue  
daisy::patch\_sm::DaisyPatchSM, 189  
GetRawFloat  
daisy::AnalogControl, 129  
GetRawValue  
daisy::AnalogControl, 129  
GetReady  
daisy::USBHostHandle, 392  
GetSampleRate  
daisy::AudioHandle, 133  
daisy::SaiHandle, 332  
GetSDFileSystem  
daisy::FatFSInterface, 228  
GetSDPath  
daisy::FatFSInterface, 229  
GetSettings  
daisy::PersistentStorage< SettingStruct >, 304  
GetSpecialControlIds  
daisy::UI, 370  
GetState  
daisy::PersistentStorage< SettingStruct >, 304  
GetStatus  
daisy::QSPIHandle, 314  
GetString  
daisy::MappedStringListValue, 273  
GetSwitch  
daisy::DaisyField, 175  
GetSysClkFreq  
daisy::System, 360  
GetTable  
daisy::WaveTableLoader, 397  
GetTick  
daisy::System, 360  
daisy::TimerHandle, 365  
GetTickFreq  
daisy::System, 360  
GetUs  
daisy::System, 361  
daisy::TimerHandle, 365  
GetUSBFileSystem  
daisy::FatFSInterface, 229  
GetUSBPath  
daisy::FatFSInterface, 229  
GetValue  
daisy::Random, 317  
GOLD  
daisy::Color, 139  
GPIOPort  
daisy, 106  
GREEN  
daisy::Color, 139  
Green  
daisy::Color, 139  
handle\_  
daisy::UiCanvasDescriptor, 372  
HasEvents  
daisy::MidiHandler< Transport >, 281  
HUMAN\_INTERFACE, 27  
I2C\_1  
daisy::I2CHandle::Config, 144  
I2C\_100KHZ  
daisy::I2CHandle::Config, 144  
I2C\_1MHZ  
daisy::I2CHandle::Config, 144  
I2C\_2  
daisy::I2CHandle::Config, 144  
I2C\_3  
daisy::I2CHandle::Config, 144  
I2C\_4  
daisy::I2CHandle::Config, 144  
I2C\_400KHZ  
daisy::I2CHandle::Config, 144  
id\_  
daisy::UiCanvasDescriptor, 372  
ImmediateRead  
daisy::RingBuffer< T, 0 >, 328  
daisy::RingBuffer< T, size >, 323  
impl\_  
EXTERNAL, 34  
Import  
daisy::WaveTableLoader, 398  
IN\_L  
UTILITY, 77  
IN\_R  
UTILITY, 77  
Increment  
daisy::Encoder, 226  
INDIRECT\_POLLING

daisy::QSPIHandle::Config, 150  
**INFO\_ROW\_ERASE\_CMD**  
 FLASH, 51  
**INFO\_ROW\_PROGRAM\_CMD**  
 FLASH, 51  
**INFO\_ROW\_READ\_CMD**  
 FLASH, 51, 52  
**Init**  
 Dac, 93  
 daisy::AbstractMenu, 115  
 daisy::AdcHandle, 125  
 daisy::Ak4556, 126  
 daisy::AnalogControl, 129  
 daisy::AudioHandle, 133  
 daisy::ButtonMonitor< BackendType, numButtons >, 136  
 daisy::Color, 139  
 daisy::CpuLoadMeter, 164  
 daisy::DacHandle, 169  
 daisy::DaisyField, 176  
 daisy::DaisyPatch, 182  
 daisy::DaisyPetal, 199  
 daisy::DaisyPod, 207  
 daisy::DaisySeed, 214  
 daisy::DaisyVersio, 219  
 daisy::Encoder, 226  
 daisy::FatFSInterface, 229  
 daisy::FullScreenItemMenu, 239  
 daisy::GateIn, 241  
 daisy::I2CHandle, 245  
 daisy::LcdHD44780, 251  
 daisy::Led, 253  
 daisy::LedDriverPca9685< numDrivers, persistent-BufferContents >, 256  
 daisy::LoggerImpl< dest >, 261  
 daisy::LoggerImpl< LOGGER\_EXTERNAL >, 262  
 daisy::LoggerImpl< LOGGER\_INTERNAL >, 263  
 daisy::LoggerImpl< LOGGER\_SEMIHOST >, 264  
 daisy::MidiHandler< Transport >, 281  
 daisy::Parameter, 301  
 daisy::patch\_sm::DaisyPatchSM, 189  
 daisy::Pcm3060, 302  
 daisy::PersistentStorage< SettingStruct >, 304  
 daisy::PotMonitor< BackendType, numPots >, 310  
 daisy::QSPIHandle, 314  
 daisy::Random, 317  
 daisy::RgbLed, 320  
 daisy::RingBuffer< T, 0 >, 328  
 daisy::RingBuffer< T, size >, 324  
 daisy::SaiHandle, 333  
 daisy::SdmmcHandler, 335  
 daisy::ShiftRegister4021< num\_daisychained, num\_parallel >, 337  
 daisy::SpiHandle, 345  
 daisy::Switch, 354, 355  
 daisy::System, 361  
 daisy::TimerHandle, 365  
 daisy::UartHandler, 368  
 daisy::UI, 370  
 daisy::USBHostHandle, 392  
 daisy::WaveTableLoader, 398  
 daisy::WavPlayer, 400  
 daisy::WavWriter< transfer\_size >, 403  
 daisy::Wm8731, 406  
 SdramHandle, 336  
 ShiftRegister595, 338  
 UsbHandle, 389  
**InitBipolarCv**  
 daisy::AnalogControl, 130  
**InitMux**  
 daisy::AdcChannelConfig, 120  
**InitSingle**  
 daisy::AdcChannelConfig, 121  
**INPUT**  
 daisy::GPIO, 243  
**InputBuffer**  
 daisy::AudioHandle, 132  
**Insert**  
 daisy::FIFOBase< T >, 234  
 daisy::StackBase< T >, 350  
**InterleavingAudioCallback**  
 daisy::AudioHandle, 132  
**InterleavingInputBuffer**  
 daisy::AudioHandle, 132  
**InterleavingOutputBuffer**  
 daisy::AudioHandle, 132  
**invalidButtonId**  
 daisy::UiEventQueue, 376  
**invalidCanvasId**  
 daisy::UI, 371  
**invalidEncoderId**  
 daisy::UiEventQueue, 376  
**invalidPotId**  
 daisy::UiEventQueue, 376  
 io0  
 daisy::QSPIHandle::Config, 150  
 io1  
 daisy::QSPIHandle::Config, 150  
 io2  
 daisy::QSPIHandle::Config, 150  
 io3  
 daisy::QSPIHandle::Config, 150  
**IS25LP064A**  
 daisy::QSPIHandle::Config, 148  
**IS25LP064A\_EAR\_HIGHEST\_SE**  
 FLASH, 52  
**IS25LP064A\_EAR\_LOWEST\_SEG**  
 FLASH, 52  
**IS25LP064A\_EAR\_SECOND\_SEG**  
 FLASH, 52  
**IS25LP064A\_EAR\_THIRD\_SEG**  
 FLASH, 52  
**IS25LP064A\_EVCR\_DTRP**  
 FLASH, 52  
**IS25LP064A\_EVCR\_DUAL**  
 FLASH, 52

IS25LP064A\_EVCR\_ODS  
    FLASH, 52  
IS25LP064A\_EVCR\_QUAD  
    FLASH, 53  
IS25LP064A\_EVCR\_RH  
    FLASH, 53  
IS25LP064A\_FSR\_ERERR  
    FLASH, 53  
IS25LP064A\_FSR\_ERSUS  
    FLASH, 53  
IS25LP064A\_FSR\_NBADDR  
    FLASH, 53  
IS25LP064A\_FSR\_PGERR  
    FLASH, 53  
IS25LP064A\_FSR\_PGSUS  
    FLASH, 53  
IS25LP064A\_FSR\_PRERR  
    FLASH, 53  
IS25LP064A\_FSR\_READY  
    FLASH, 54  
IS25LP064A\_NVCR\_DTRP  
    FLASH, 54  
IS25LP064A\_NVCR\_DUAL  
    FLASH, 54  
IS25LP064A\_NVCR\_NB\_DUMMY  
    FLASH, 54  
IS25LP064A\_NVCR\_NBADDR  
    FLASH, 54  
IS25LP064A\_NVCR\_ODS  
    FLASH, 54  
IS25LP064A\_NVCR\_QUAB  
    FLASH, 54  
IS25LP064A\_NVCR\_RH  
    FLASH, 54  
IS25LP064A\_NVCR\_SEGMENT  
    FLASH, 55  
IS25LP064A\_NVCR\_XIP  
    FLASH, 55  
IS25LP064A\_SR\_QE  
    FLASH, 55  
IS25LP064A\_SR\_SRWREN  
    FLASH, 55  
IS25LP064A\_SR\_WIP  
    FLASH, 55  
IS25LP064A\_SR\_WREN  
    FLASH, 55  
IS25LP064A\_VCR\_NB\_DUMMY  
    FLASH, 55  
IS25LP064A\_VCR\_WRAP  
    FLASH, 56  
IS25LP064A\_VCR\_XIP  
    FLASH, 56  
IS25LP080D  
    daisy::QSPIHandle::Config, 148  
IS25LP080D\_EAR\_HIGHEST\_SEG  
    FLASH, 56  
IS25LP080D\_EAR\_LOWEST\_SEG  
    FLASH, 56  
IS25LP080D\_EAR\_SECOND\_SEG  
    FLASH, 56  
IS25LP080D\_EAR\_THIRD\_SEG  
    FLASH, 56  
IS25LP080D\_EVCR\_DTRP  
    FLASH, 56  
IS25LP080D\_EVCR\_DUAL  
    FLASH, 56  
IS25LP080D\_EVCR\_ODS  
    FLASH, 57  
IS25LP080D\_EVCR\_QUAD  
    FLASH, 57  
IS25LP080D\_EVCR\_RH  
    FLASH, 57  
IS25LP080D\_FSR\_ERERR  
    FLASH, 57  
IS25LP080D\_FSR\_ERSUS  
    FLASH, 57  
IS25LP080D\_FSR\_NBADDR  
    FLASH, 57  
IS25LP080D\_FSR\_PGERR  
    FLASH, 57  
IS25LP080D\_FSR\_PGSUS  
    FLASH, 57  
IS25LP080D\_FSR\_PRERR  
    FLASH, 58  
IS25LP080D\_FSR\_READY  
    FLASH, 58  
IS25LP080D\_NVCR\_DTRP  
    FLASH, 58  
IS25LP080D\_NVCR\_DUAL  
    FLASH, 58  
IS25LP080D\_NVCR\_NB\_DUMMY  
    FLASH, 58  
IS25LP080D\_NVCR\_NBADDR  
    FLASH, 58  
IS25LP080D\_NVCR\_ODS  
    FLASH, 58  
IS25LP080D\_NVCR\_QUAB  
    FLASH, 58  
IS25LP080D\_NVCR\_RH  
    FLASH, 59  
IS25LP080D\_NVCR\_SEGMENT  
    FLASH, 59  
IS25LP080D\_NVCR\_XIP  
    FLASH, 59  
IS25LP080D\_SR\_QE  
    FLASH, 59  
IS25LP080D\_SR\_SRWREN  
    FLASH, 59  
IS25LP080D\_SR\_WIP  
    FLASH, 59  
IS25LP080D\_SR\_WREN  
    FLASH, 59  
IS25LP080D\_VCR\_NB\_DUMMY  
    FLASH, 60  
IS25LP080D\_VCR\_WRAP  
    FLASH, 60

IS25LP080D\_VCR\_XIP  
     FLASH, 60

IsActive  
     daisy::UiPage, 378

IsButtonPressed  
     daisy::ButtonMonitor< BackendType, numButtons >, 136

isEditing\_  
     daisy::AbstractMenu, 119

IsEmpty  
     daisy::FIFOBase< T >, 234  
     daisy::StackBase< T >, 351

isEmpty  
     daisy::RingBuffer< T, size >, 324

IsFull  
     daisy::FIFOBase< T >, 234  
     daisy::StackBase< T >, 351

IsFunctionButtonDown  
     daisy::AbstractMenu, 115

IsMoving  
     daisy::PotMonitor< BackendType, numPots >, 310

IsOpaque  
     daisy::UiPage, 378

IsQueueEmpty  
     daisy::UiEventQueue, 376

IsReady  
     daisy::Random, 317

IsRecording  
     daisy::WavWriter< transfer\_size >, 403

IsValid  
     daisy::Pin, 306

itemObject  
     daisy::AbstractMenu::ItemConfig, 250

items\_  
     daisy::AbstractMenu, 119

ItemType  
     daisy::AbstractMenu, 114

JumpToQspi  
     daisy::System, 361

KeyboardFallingEdge  
     daisy::DaisyField, 176

KeyboardRisingEdge  
     daisy::DaisyField, 176

KeyboardState  
     daisy::DaisyField, 176

Knob  
     daisy::DaisyPetal, 195  
     daisy::DaisyPod, 205

knob  
     daisy::DaisyPetal, 203

knob1  
     daisy::DaisyPod, 209

knob2  
     daisy::DaisyPod, 209

KNOB\_1  
     daisy::DaisyField, 172  
     daisy::DaisyPetal, 195

KNOB\_2  
     daisy::DaisyField, 172  
     daisy::DaisyPetal, 195  
     daisy::DaisyPod, 205

KNOB\_3  
     daisy::DaisyField, 172  
     daisy::DaisyPetal, 195

KNOB\_4  
     daisy::DaisyField, 172  
     daisy::DaisyPetal, 195

KNOB\_5  
     daisy::DaisyField, 172  
     daisy::DaisyPetal, 195

KNOB\_6  
     daisy::DaisyField, 172  
     daisy::DaisyPetal, 195

KNOB\_7  
     daisy::DaisyField, 172

KNOB\_8  
     daisy::DaisyField, 172

KNOB\_LAST  
     daisy::DaisyField, 172  
     daisy::DaisyPetal, 195  
     daisy::DaisyPod, 205

knobs  
     daisy::DaisyPod, 209

kQspiBootloaderOffset  
     daisy::System, 361

kWavFileChunkId  
     daisy, 109

kWavFileSubChunk1Id  
     daisy, 109

kWavFileSubChunk2Id  
     daisy, 109

kWavFileWaveId  
     daisy, 109

LAST  
     daisy::Color, 139  
     daisy::Parameter, 300

latch  
     daisy::ShiftRegister4021< num\_daisychained, num\_parallel >::Config, 154

Lcd, 90

LED, 69

led1  
     daisy::DaisyPod, 210

led2  
     daisy::DaisyPod, 210

LED\_KEY\_A1  
     daisy::DaisyField, 173

LED\_KEY\_A2  
     daisy::DaisyField, 173

LED\_KEY\_A3  
     daisy::DaisyField, 173

LED\_KEY\_A4  
     daisy::DaisyField, 173

LED\_KEY\_A5  
     daisy::DaisyField, 173

LED\_KEY\_A6  
    daisy::DaisyField, 173

LED\_KEY\_A7  
    daisy::DaisyField, 172

LED\_KEY\_A8  
    daisy::DaisyField, 172

LED\_KEY\_B1  
    daisy::DaisyField, 172

LED\_KEY\_B2  
    daisy::DaisyField, 172

LED\_KEY\_B3  
    daisy::DaisyField, 172

LED\_KEY\_B4  
    daisy::DaisyField, 172

LED\_KEY\_B5  
    daisy::DaisyField, 172

LED\_KEY\_B6  
    daisy::DaisyField, 172

LED\_KEY\_B7  
    daisy::DaisyField, 172

LED\_KEY\_B8  
    daisy::DaisyField, 172

LED\_KNOB\_1  
    daisy::DaisyField, 173

LED\_KNOB\_2  
    daisy::DaisyField, 173

LED\_KNOB\_3  
    daisy::DaisyField, 173

LED\_KNOB\_4  
    daisy::DaisyField, 173

LED\_KNOB\_5  
    daisy::DaisyField, 173

LED\_KNOB\_6  
    daisy::DaisyField, 173

LED\_KNOB\_7  
    daisy::DaisyField, 173

LED\_KNOB\_8  
    daisy::DaisyField, 173

LED\_LAST  
    daisy::DaisyField, 173

LED\_SW\_1  
    daisy::DaisyField, 173

LED\_SW\_2  
    daisy::DaisyField, 173

left  
    daisy, 105

leftRightSelectUpDownModify  
    daisy::AbstractMenu, 115

LIBDAISY, 27

lin  
    daisy::MappedFloatValue, 266

LINEAR  
    daisy::Parameter, 300

Listen  
    daisy::MidiHandler< Transport >, 281

local\_control\_off  
    daisy::LocalControlEvent, 258

local\_control\_on

daisy::LocalControlEvent, 259

LocalControl  
    daisy, 106

log  
    daisy::MappedFloatValue, 266

LOGARITHMIC  
    daisy::Parameter, 300

LogBlockNbr  
    DSY\_SD\_CardInfoTypeDef, 224

LogBlockSize  
    DSY\_SD\_CardInfoTypeDef, 225

Logger  
    EXTERNAL, 32

LOGGER\_BUFFER  
    EXTERNAL, 31

LOGGER\_EXTERNAL  
    daisy, 107

LOGGER\_INTERNAL  
    daisy, 107

LOGGER\_NEWLINE  
    EXTERNAL, 31

LOGGER\_NONE  
    daisy, 107

LOGGER\_SEMIHOST  
    daisy, 107

LOGGER\_SYNC\_IN  
    EXTERNAL, 32

LoggerConsts  
    EXTERNAL, 32

LoggerDestination  
    daisy, 106

lr\_swap  
    daisy::Wm8731::Config, 162

MappedFloatValue  
    daisy::MappedFloatValue, 266

MappedIntValue  
    daisy::MappedIntValue, 269

MappedStringListValue  
    daisy::MappedStringListValue, 272

Mapping  
    daisy::MappedFloatValue, 265

mcu\_is\_master  
    daisy::Wm8731::Config, 162

Media  
    daisy::FatFSInterface::Config, 142

MEDIUM\_SLOW  
    daisy::SdmmcHandler, 335

MEMORY\_MAPPED  
    daisy::QSPIHandle::Config, 150

MemoryRegion  
    daisy::System, 358

menuEncoderId  
    daisy::UI::SpecialControlIds, 341

message\_type  
    daisy::MTCQuarterFrameEvent, 285

MessageLast  
    daisy, 107

midi

daisy::DaisyPatch, 184  
 MidiMessageType  
     daisy, 107  
 miso  
     daisy::SpiHandle::Config, 155  
 Mode  
     daisy::DacHandle, 168  
     daisy::GPIO, 242  
     daisy::I2CHandle::Config, 144  
     daisy::QSPIHandle::Config, 148  
 mode  
     Dac, 97  
     daisy::I2CHandle::Config, 145  
     dsy\_gpio, 222  
 ModifyValue  
     daisy::AbstractMenu::CustomItem, 166  
 MonoModeOn  
     daisy, 106  
 mosi  
     daisy::SpiHandle::Config, 155  
 MSD\_ERROR  
     UTILITY, 77  
 MSD\_ERROR\_SD\_NOT\_PRESENT  
     UTILITY, 77  
 MSD\_OK  
     UTILITY, 77  
 MTCQuarterFrame  
     daisy, 107  
 MULTIPLE\_IO\_READ\_ID\_CMD  
     FLASH, 60  
 Mute  
     daisy::UI, 371  
 mux\_channels\_  
     daisy::AdcChannelConfig, 121  
 mux\_pin\_  
     daisy::AdcChannelConfig, 121  
 MUX\_SEL\_0  
     daisy::AdcChannelConfig, 120  
 MUX\_SEL\_1  
     daisy::AdcChannelConfig, 120  
 MUX\_SEL\_2  
     daisy::AdcChannelConfig, 120  
 MUX\_SEL\_LAST  
     daisy::AdcChannelConfig, 120  
 MuxPin  
     daisy::AdcChannelConfig, 120  
 name  
     daisy::WavFileInfo, 399  
 NbrChannels  
     daisy::WAV\_FormatTypeDef, 396  
 ncs  
     daisy::QSPIHandle::Config, 150  
 NewLineSeqLength  
     EXTERNAL, 33  
 NO\_OP  
     FLASH, 60  
 NOPULL  
     daisy::GPIO, 243  
 note  
     daisy::NoteOffEvent, 285  
     daisy::NoteOnEvent, 286  
 NoteOff  
     daisy, 107  
 NoteOn  
     daisy, 107  
 nss  
     daisy::SpiHandle::Config, 155  
 num\_channels  
     daisy::MonoModeOnEvent, 284  
 NUM\_PINS  
     ShiftRegister595, 338  
 numItems\_  
     daisy::AbstractMenu, 119  
 OFF  
     daisy::Color, 139  
 OK  
     Dac, 92  
     daisy::I2CHandle, 245  
     daisy::SpiHandle, 343  
     daisy::UartHandler, 367  
     SdramHandle, 336  
 OmniModeOff  
     daisy, 106  
 OmniModeOn  
     daisy, 106  
 OnArrowButton  
     daisy::AbstractMenu, 115  
     daisy::UiPage, 378  
 OnBlockEnd  
     daisy::CpuLoadMeter, 165  
 OnBlockStart  
     daisy::CpuLoadMeter, 165  
 OnButton  
     daisy::UiPage, 378  
 OnCancelButton  
     daisy::AbstractMenu, 116  
     daisy::UiPage, 379  
 OnEncoderActivityChanged  
     daisy::UiPage, 379  
 OnEncoderTurned  
     daisy::UiPage, 381  
 OnFocusGained  
     daisy::UiPage, 381  
 OnFocusLost  
     daisy::UiPage, 381  
 OnFunctionButton  
     daisy::AbstractMenu, 116  
     daisy::UiPage, 381  
 OnHide  
     daisy::UiPage, 382  
 OnMenuEncoderActivityChanged  
     daisy::UiPage, 382  
 OnMenuEncoderTurned  
     daisy::AbstractMenu, 117  
     daisy::UiPage, 382  
 OnOkayButton

daisy::AbstractMenu, 117  
daisy::AbstractMenu::CustomItem, 167  
daisy::UiPage, 383  
OnPotActivityChanged  
    daisy::UiPage, 383  
OnPotMoved  
    daisy::UiPage, 384  
OnShow  
    daisy::AbstractMenu, 118  
    daisy::UiPage, 384  
OnValueEncoderActivityChanged  
    daisy::UiPage, 384  
OnValueEncoderTurned  
    daisy::AbstractMenu, 118  
    daisy::UiPage, 385  
OnValuePotActivityChanged  
    daisy::UiPage, 385  
OnValuePotMoved  
    daisy::AbstractMenu, 118  
    daisy::UiPage, 385  
Open  
    daisy::WavPlayer, 400  
OpenFile  
    daisy::WavWriter< transfer\_size >, 404  
OpenPage  
    daisy::UI, 371  
openUiPageItem  
    daisy::AbstractMenu, 114  
operator const char \*  
    daisy::MappedStringValue, 273  
operator dsy\_gpio\_pin  
    daisy::Pin, 307  
operator float  
    daisy::MappedFloatValue, 267  
operator int  
    daisy::MappedIntValue, 270  
    daisy::MappedStringValue, 273  
operator=  
    daisy::FIFO< T, capacity >, 231  
    daisy::FIFOBase< T >, 234  
    daisy::MappedFloatValue, 267  
    daisy::MappedIntValue, 270  
    daisy::MappedStringValue, 273  
    daisy::Stack< T, capacity >, 349  
    daisy::StackBase< T >, 351  
operator[]  
    daisy::FIFOBase< T >, 234  
    daisy::StackBase< T >, 351  
Orientation  
    daisy::AbstractMenu, 114  
orientation\_  
    daisy::AbstractMenu, 119  
OTHER, 36  
    dsy\_gpio\_deinit, 39  
    dsy\_gpio\_init, 39  
    dsy\_gpio\_mode, 37  
    DSY\_GPIO\_MODE\_ANALOG, 37  
    DSY\_GPIO\_MODE\_INPUT, 37  
    DSY\_GPIO\_MODE\_LAST, 37  
    DSY\_GPIO\_MODE\_OUTPUT\_OD, 37  
    DSY\_GPIO\_MODE\_OUTPUT\_PP, 37  
    DSY\_GPIO\_NOPULL, 39  
    dsy\_gpio\_pull, 37  
    DSY\_GPIO\_PULLDOWN, 39  
    DSY\_GPIO\_PULLUP, 39  
    dsy\_gpio\_read, 39  
    dsy\_gpio\_toggle, 40  
    dsy\_gpio\_write, 40  
OUT\_L  
    UTILITY, 78  
OUT\_R  
    UTILITY, 78  
OUTPUT  
    daisy::GPIO, 243  
OUTPUT\_OD  
    daisy::GPIO, 243  
OutputBuffer  
    daisy::AudioHandle, 132  
OverSampling  
    daisy::AdcHandle, 122  
Overwrite  
    daisy::RingBuffer< T, 0 >, 329  
    daisy::RingBuffer< T, size >, 324  
OVS\_1024  
    daisy::AdcHandle, 123  
OVS\_128  
    daisy::AdcHandle, 123  
OVS\_16  
    daisy::AdcHandle, 123  
OVS\_256  
    daisy::AdcHandle, 123  
OVS\_32  
    daisy::AdcHandle, 123  
OVS\_4  
    daisy::AdcHandle, 123  
OVS\_512  
    daisy::AdcHandle, 123  
OVS\_64  
    daisy::AdcHandle, 123  
OVS\_8  
    daisy::AdcHandle, 123  
OVS\_LAST  
    daisy::AdcHandle, 123  
OVS\_NONE  
    daisy::AdcHandle, 123  
PAGE\_PROG\_CMD  
    FLASH, 60, 61  
pageToOpen  
    daisy::AbstractMenu::ItemConfig, 250  
Parameter  
    daisy::Parameter, 301  
Parse  
    daisy::MidiHandler< Transport >, 282  
pc\_sync\_  
    EXTERNAL, 34  
periph

daisy::I2CHandle::Config, 145  
**PERIPHERAL**, 35  
 Peripheral  
   daisy::I2CHandle::Config, 144  
   daisy::SaiHandle::Config, 152  
   daisy::TimerHandle::Config, 158  
**PersistentStorage**  
   daisy::PersistentStorage< SettingStruct >, 304  
**Pin**  
   Dac, 91  
   daisy::Pin, 306  
**pin**  
   dsy\_gpio, 222  
   dsy\_gpio\_pin, 223  
**pin\_**  
   daisy::AdcChannelConfig, 121  
**PIN\_CLK**  
   ShiftRegister595, 338  
**pin\_config**  
   daisy::I2CHandle::Config, 145  
   daisy::UartHandler::Config, 159  
**PIN\_DATA**  
   ShiftRegister595, 338  
**PinBank**  
   daisy::patch\_sm::DaisyPatchSM, 187  
**Pins**  
   ShiftRegister595, 338  
**PitchBend**  
   daisy, 107  
**Polarity**  
   daisy::Switch, 353  
**POLARITY\_INVERTED**  
   daisy::Switch, 353  
**POLARITY\_NORMAL**  
   daisy::Switch, 353  
**PollReceive**  
   daisy::UartHandler, 368  
**PollTx**  
   daisy::UartHandler, 368  
**PolyModeOn**  
   daisy, 106  
**PolyphonicKeyPressure**  
   daisy, 107  
**PopBack**  
   daisy::StackBase< T >, 351  
**PopEvent**  
   daisy::MidiHandler< Transport >, 282  
**PopFront**  
   daisy::FIFOBase< T >, 235  
**PopRx**  
   daisy::UartHandler, 369  
**port**  
   dsy\_gpio\_pin, 223  
**PORTA**  
   daisy, 106  
**PORTB**  
   daisy, 106  
**PORTC**  
   daisy, 106  
   daisy, 106  
**PORTD**  
   daisy, 106  
**PORTE**  
   daisy, 106  
**PORTF**  
   daisy, 106  
**PORTG**  
   daisy, 106  
**PORTH**  
   daisy, 106  
**PORTI**  
   daisy, 106  
**PORTJ**  
   daisy, 106  
**PORTK**  
   daisy, 106  
**PORTL**  
   daisy, 106  
**PORTN**  
   daisy, 106  
**position**  
   daisy::SongPositionPointerEvent, 340  
**pow2**  
   daisy::MappedFloatValue, 266  
**PPCAT**  
   EXTERNAL, 31  
**PPCAT\_NX**  
   EXTERNAL, 31  
**Prepare**  
   daisy::WavPlayer, 401  
**PresetColor**  
   daisy::Color, 138  
**Pressed**  
   daisy::Encoder, 226  
   daisy::Switch, 355  
**pressure**  
   daisy::ChannelPressureEvent, 137  
**Print**  
   daisy::DaisySeed, 214  
   daisy::LcdHD44780, 252  
   daisy::patch\_sm::DaisyPatchSM, 189  
   EXTERNAL, 33  
**PrintInt**  
   daisy::LcdHD44780, 252  
**PrintLine**  
   daisy::DaisySeed, 215  
   daisy::patch\_sm::DaisyPatchSM, 189  
   EXTERNAL, 33  
**PrintLineV**  
   EXTERNAL, 33  
**PrintV**  
   EXTERNAL, 33  
**Process**  
   daisy::AnalogControl, 130  
   daisy::ButtonMonitor< BackendType, numButtons >, 137  
   daisy::Parameter, 301  
   daisy::PotMonitor< BackendType, numPots >, 311  
   daisy::Ui, 371

daisy::USBHostHandle, 393  
ProcessAllControls  
    daisy::DaisyField, 177  
    daisy::DaisyPatch, 182  
    daisy::DaisyPetal, 199  
    daisy::DaisyPod, 207  
    daisy::DaisyVersio, 220  
    daisy::patch\_sm::DaisyPatchSM, 189  
ProcessAnalogControls  
    daisy::DaisyField, 177  
    daisy::DaisyPatch, 182  
    daisy::DaisyPetal, 199  
    daisy::DaisyPod, 207  
    daisy::DaisyVersio, 220  
    daisy::patch\_sm::DaisyPatchSM, 190  
ProcessDigitalControls  
    daisy::DaisyField, 177  
    daisy::DaisyPatch, 183  
    daisy::DaisyPetal, 199  
    daisy::DaisyPod, 207  
    daisy::patch\_sm::DaisyPatchSM, 190  
ProcessInput  
    daisy::VocCalibration, 394  
PROG\_ERASE\_RESUME\_CMD  
    FLASH, 61  
PROG\_ERASE\_SUSPEND\_CMD  
    FLASH, 61  
program  
    daisy::ProgramChangeEvent, 311  
ProgramChange  
    daisy, 107  
Pull  
    daisy::GPIO, 243  
    daisy::Switch, 353  
pull  
    dsy\_gpio, 222  
PULL\_DOWN  
    daisy::Switch, 354  
PULL\_NONE  
    daisy::Switch, 354  
PULL\_UP  
    daisy::Switch, 354  
PULLDOWN  
    daisy::GPIO, 243  
PULLUP  
    daisy::GPIO, 243  
PURPLE  
    daisy::Color, 139  
PushBack  
    daisy::FIFOBase< T >, 235  
    daisy::StackBase< T >, 352  
QUAD\_IN\_FAST\_PROG\_CMD  
    FLASH, 61  
QUAD\_IN\_PAGE\_PROG\_CMD  
    FLASH, 62  
QUAD\_INOUT\_FAST\_READ\_CMD  
    FLASH, 62  
QUAD\_INOUT\_FAST\_READ\_DTR\_CMD  
    FLASH, 62  
range  
    Dac, 97  
raw\_data  
    daisy::WavFileInfo, 399  
RawState  
    daisy::Switch, 355  
Read  
    daisy::RingBuffer< T, 0 >, 329  
    daisy::RingBuffer< T, size >, 326  
READ\_CMD  
    FLASH, 63  
READ\_EXT\_READ\_PARAM\_CMD  
    FLASH, 63  
READ\_FUNCTION\_REGISTER  
    FLASH, 63  
READ\_ID\_CMD  
    FLASH, 63  
READ\_ID\_CMD2  
    FLASH, 63, 64  
READ\_MANUFACT\_AND\_ID  
    FLASH, 64  
READ\_READ\_PARAM\_REG\_CMD  
    FLASH, 64  
READ\_SERIAL\_FLASH\_DISCO\_PARAM\_CMD  
    FLASH, 64  
READ\_STATUS\_REG\_CMD  
    FLASH, 64  
READ\_UNIQUE\_ID  
    FLASH, 65  
Readable  
    daisy::UartHandler, 369  
readable  
    daisy::RingBuffer< T, 0 >, 329  
    daisy::RingBuffer< T, size >, 326  
ReadAnalogPinRaw  
    Dac, 93  
ReadAnalogPinVolts  
    Dac, 93  
ReadDataAtAddress  
    daisy::I2CHandle, 246  
ReadDigitalPin  
    Dac, 95  
RECEIVE  
    daisy::I2CHandle, 245  
ReceiveBlocking  
    daisy::I2CHandle, 246  
ReceiveCallback  
    UsbHandle, 387  
ReceiveDma  
    daisy::I2CHandle, 246  
Record  
    daisy::VocCalibration, 394  
RED  
    daisy::Color, 139  
Red

daisy::Color, 140  
**RelCardAdd**  
 DSY\_SD\_CardInfoTypeDef, 225  
**Remove**  
 daisy::FIFOBase< T >, 235  
 daisy::StackBase< T >, 352  
**RemoveAllEqualTo**  
 daisy::FIFOBase< T >, 235  
 daisy::StackBase< T >, 352  
**Reset**  
 daisy, 108  
 daisy::CpuLoadMeter, 165  
**reset**  
 daisy::SSD130x4WireSpiTransport::Config, 156  
**RESET\_ENABLE\_CMD**  
 FLASH, 65  
**RESET\_MEMORY\_CMD**  
 FLASH, 65  
**ResetAllControllers**  
 daisy, 106  
**ResetToBootloader**  
 daisy::System, 361  
**ResetToDefault**  
 daisy::MappedFloatValue, 267  
 daisy::MappedIntValue, 270  
 daisy::MappedStringListValue, 274  
 daisy::MappedValue, 275  
**Restart**  
 daisy::WavPlayer, 401  
**RestoreDefaults**  
 daisy::PersistentStorage< SettingStruct >, 305  
**Result**  
 Dac, 91, 92  
 daisy::DacHandle, 169  
 daisy::FatFSInterface, 228  
 daisy::I2CHandle, 245  
 daisy::SaiHandle, 332  
 daisy::SdmmcHandler, 335  
 daisy::SpiHandle, 343  
 daisy::TimerHandle, 364  
 daisy::UartHandler, 367  
 daisy::WavWriter< transfer\_size >, 403  
 daisy::Wm8731, 405  
 SdramHandle, 336  
 UsbHandle, 387  
**right**  
 daisy, 105  
**ring\_led**  
 daisy::DaisyPetal, 203  
**RING\_LED\_1**  
 daisy::DaisyPetal, 195  
**RING\_LED\_2**  
 daisy::DaisyPetal, 195  
**RING\_LED\_3**  
 daisy::DaisyPetal, 195  
**RING\_LED\_4**  
 daisy::DaisyPetal, 195  
**RING\_LED\_5**  
 daisy::DaisyPetal, 195  
**daisy::DaisyPetal**, 195  
**RING\_LED\_6**  
 daisy::DaisyPetal, 195  
**RING\_LED\_7**  
 daisy::DaisyPetal, 195  
**RING\_LED\_8**  
 daisy::DaisyPetal, 195  
**RING\_LED\_LAST**  
 daisy::DaisyPetal, 195  
**RingLed**  
 daisy::DaisyPetal, 195  
**RisingEdge**  
 daisy::Encoder, 226  
 daisy::Switch, 356  
**RX**  
 daisy::SpiHandle, 343  
**rx**  
 daisy::UartHandler::Config, 160  
**RxActive**  
 daisy::UartHandler, 369  
**s162f**  
 UTILITY, 88  
**S162F\_SCALE**  
 UTILITY, 78  
**s242f**  
 UTILITY, 88  
**S242F\_SCALE**  
 UTILITY, 78  
**S24SIGN**  
 UTILITY, 78  
**s322f**  
 UTILITY, 88  
**S322F\_SCALE**  
 UTILITY, 78  
**s82f**  
 UTILITY, 89  
**S82F\_SCALE**  
 UTILITY, 78  
**Sample**  
 daisy::WavWriter< transfer\_size >, 404  
**SampleRate**  
 daisy::SaiHandle::Config, 152  
 daisy::WAV\_FormatTypeDef, 396  
**Save**  
 daisy::PersistentStorage< SettingStruct >, 305  
**SaveFile**  
 daisy::WavWriter< transfer\_size >, 404  
**scl**  
 daisy::I2CHandle::Config, 145  
**sclk**  
 daisy::SpiHandle::Config, 155  
**screenSaverTimeOut**  
 daisy::UiCanvasDescriptor, 373  
**SCUndefined0**  
 daisy, 107  
**SCUndefined1**  
 daisy, 107  
**SD\_DATATIMEOUT**

UTILITY, 78  
SD\_NOT\_PRESENT  
    UTILITY, 79  
SD\_PRESENT  
    UTILITY, 79  
SD\_TRANSFER\_BUSY  
    UTILITY, 79  
SD\_TRANSFER\_OK  
    UTILITY, 79  
sda  
    daisy::I2CHandle::Config, 145  
SDRAM, 69  
    DSY\_SDRAM\_BSS, 69  
    DSY\_SDRAM\_DATA, 69  
sdram\_handle  
    daisy::DaisySeed, 216  
SdramHandle, 335  
    ERR, 336  
    Init, 336  
    OK, 336  
    Result, 336  
SECTOR\_ERASE\_CMD  
    FLASH, 65  
SECTOR\_ERASE\_QPI\_CMD  
    FLASH, 66  
SECTOR\_LOCK  
    FLASH, 66  
SECTOR\_UNLOCK  
    FLASH, 66  
seed  
    daisy::DaisyPatch, 185  
    daisy::DaisyPetal, 203  
    daisy::DaisyPod, 210  
selectedItemIdx\_  
    daisy::AbstractMenu, 119  
SendMessage  
    daisy::MidiHandler< Transport >, 282  
SERIAL, 35  
    dsy\_spi\_global\_init, 36  
Set  
    daisy::Led, 254  
    daisy::MappedFloatValue, 268  
    daisy::MappedIntValue, 271  
    daisy::RgbLed, 320  
    ShiftRegister595, 339  
SetAllTo  
    daisy::LedDriverPca9685< numDrivers, persistent-  
        BufferContents >, 257  
SetAllToRaw  
    daisy::LedDriverPca9685< numDrivers, persistent-  
        BufferContents >, 257  
SetAudioBlockSize  
    daisy::DaisyField, 177  
    daisy::DaisyPatch, 183  
    daisy::DaisyPetal, 200  
    daisy::DaisyPod, 207  
    daisy::DaisySeed, 215  
    daisy::DaisyVersio, 220  
    daisy::patch\_sm::DaisyPatchSM, 190  
SetAudioSampleRate  
    daisy::DaisyField, 177  
    daisy::DaisyPatch, 183  
    daisy::DaisyPetal, 200  
    daisy::DaisyPod, 208  
    daisy::DaisySeed, 215  
    daisy::DaisyVersio, 220  
    daisy::patch\_sm::DaisyPatchSM, 190  
SetBlockSize  
    daisy::AudioHandle, 134  
SetBlue  
    daisy::RgbLed, 320  
SetCoeff  
    daisy::AnalogControl, 130  
SetColor  
    daisy::RgbLed, 321  
SetCursor  
    daisy::LcdHD44780, 252  
    daisy::OneBitGraphicsDisplay, 294  
SetCvOut1  
    daisy::DaisyField, 177  
SetCvOut2  
    daisy::DaisyField, 178  
SetData  
    daisy::VoctCalibration, 394  
SetFootswitchLed  
    daisy::DaisyPetal, 200  
SetFrom0to1  
    daisy::MappedFloatValue, 268  
    daisy::MappedIntValue, 271  
    daisy::MappedStringListValue, 274  
    daisy::MappedValue, 276  
SetGreen  
    daisy::RgbLed, 321  
SetIndex  
    daisy::MappedStringListValue, 274  
SetLed  
    daisy::DaisySeed, 215  
    daisy::DaisyVersio, 220  
    daisy::LedDriverPca9685< numDrivers, persistent-  
        BufferContents >, 257  
SetLedRaw  
    daisy::LedDriverPca9685< numDrivers, persistent-  
        BufferContents >, 257  
SetLooping  
    daisy::WavPlayer, 401  
SetOneBitGraphicsDisplayToDrawTo  
    daisy::FullScreenItemMenu, 240  
SetPeriod  
    daisy::TimerHandle, 365  
SetPostGain  
    daisy::AudioHandle, 134  
SetPrescaler  
    daisy::TimerHandle, 365  
SetReceiveCallback  
    UsbHandle, 389, 390  
SetRed

daisy::RgbLed, 321  
 SetRingLed  
     daisy::DaisyPetal, 200  
 SetSampleRate  
     daisy::AnalogControl, 130  
     daisy::AudioHandle, 134  
     daisy::Led, 254  
 SetTestPoint  
     daisy::DaisySeed, 215  
 SetUpdateRate  
     daisy::Encoder, 226  
     daisy::Sswitch, 356  
 SetWaveTableInfo  
     daisy::WaveTableLoader, 398  
 SHIFTREGISTER, 42  
 ShiftRegister595, 337  
     Init, 338  
     NUM\_PINS, 338  
     PIN\_CLK, 338  
     PIN\_DATA, 338  
     Pins, 338  
     Set, 339  
     Write, 339  
 SLOW  
     daisy::SdmmcHandler, 335  
 song  
     daisy::SongSelectEvent, 340  
 SongPositionPointer  
     daisy, 107  
 SongSelect  
     daisy, 107  
 Speed  
     daisy::I2CHandle::Config, 144  
     daisy::SdmmcHandler, 335  
 speed  
     daisy::I2CHandle::Config, 145  
 spi\_config  
     Dac, 98  
 src/sys/ffconf.h, 407  
 src/util/usbh\_diskio.h, 414  
 SRTUndefined0  
     daisy, 108  
 SRTUndefined1  
     daisy, 108  
 SSD130x4WireSpi128x32Driver  
     daisy, 103  
 SSD130x4WireSpi128x64Driver  
     daisy, 104  
 SSD130x4WireSpi64x32Driver  
     daisy, 104  
 SSD130x4WireSpi64x48Driver  
     daisy, 104  
 SSD130x4WireSpi98x16Driver  
     daisy, 104  
 SSD130xl2c128x32Driver  
     daisy, 104  
 SSD130xl2c128x64Driver  
     daisy, 104  
 SSD130xl2c64x32Driver  
     daisy, 104  
 SSD130xl2c64x48Driver  
     daisy, 105  
 SSD130xl2c98x16Driver  
     daisy, 105  
 Stack  
     daisy::Stack< T, capacity >, 348  
 STANDARD  
     daisy::SdmmcHandler, 335  
 Start  
     daisy, 108  
     daisy::AdcHandle, 125  
     daisy::AudioHandle, 134  
     daisy::DacHandle, 169  
     daisy::TimerHandle, 366  
 StartAdc  
     daisy::DaisyField, 178  
     daisy::DaisyPatch, 183  
     daisy::DaisyPetal, 202  
     daisy::DaisyPod, 208  
     daisy::DaisyVersio, 220  
     daisy::patch\_sm::DaisyPatchSM, 190  
 StartAudio  
     daisy::DaisyField, 178  
     daisy::DaisyPatch, 183  
     daisy::DaisyPetal, 202  
     daisy::DaisyPod, 208  
     daisy::DaisySeed, 215, 216  
     daisy::DaisyVersio, 220, 221  
     daisy::patch\_sm::DaisyPatchSM, 190  
 StartDac  
     daisy::DaisyField, 178  
     daisy::patch\_sm::DaisyPatchSM, 191  
 StartDma  
     daisy::SaiHandle, 333  
 StartLog  
     daisy::DaisySeed, 216  
     daisy::patch\_sm::DaisyPatchSM, 191  
     EXTERNAL, 33  
 StartReceive  
     daisy::MidiHandler< Transport >, 282  
 StartRx  
     daisy::UartHandler, 369  
 State  
     daisy::GateIn, 241  
     daisy::PersistentStorage< SettingStruct >, 303  
     daisy::ShiftRegister4021< num\_daisychained,  
         num\_parallel >, 337  
 Status  
     daisy::QSPIHandle, 312  
 Step  
     daisy::MappedFloatValue, 268  
     daisy::MappedIntValue, 271  
     daisy::MappedStringListValue, 274  
     daisy::MappedValue, 276  
 Stop  
     daisy, 108

daisy::AdcHandle, 126  
daisy::AudioHandle, 134  
daisy::DacHandle, 169  
daisy::TimerHandle, 366  
StopAdc  
    daisy::DaisyField, 178  
    daisy::DaisyPatch, 183  
    daisy::DaisyPetal, 202  
    daisy::DaisyPod, 208  
    daisy::DaisyVersio, 221  
    daisy::patch\_sm::DaisyPatchSM, 191  
StopAudio  
    daisy::DaisyField, 178  
    daisy::DaisyPatch, 184  
    daisy::DaisyPetal, 202  
    daisy::DaisyPod, 208  
    daisy::DaisySeed, 216  
    daisy::DaisyVersio, 221  
    daisy::patch\_sm::DaisyPatchSM, 191  
StopDac  
    daisy::patch\_sm::DaisyPatchSM, 191  
StopDma  
    daisy::SaiHandle, 333  
Stream  
    daisy::WavPlayer, 401  
STRINGIZE  
    EXTERNAL, 32  
STRINGIZE\_NX  
    EXTERNAL, 32  
SubChunk1ID  
    daisy::WAV\_FormatTypeDef, 396  
SubChunk1Size  
    daisy::WAV\_FormatTypeDef, 396  
SubChunk2ID  
    daisy::WAV\_FormatTypeDef, 396  
SubCHunk2Size  
    daisy::WAV\_FormatTypeDef, 397  
Sw  
    daisy::DaisyPetal, 195  
    daisy::DaisyPod, 205  
SW\_1  
    daisy::DaisyField, 171  
    daisy::DaisyPetal, 196  
SW\_2  
    daisy::DaisyField, 171  
    daisy::DaisyPetal, 196  
SW\_3  
    daisy::DaisyPetal, 196  
SW\_4  
    daisy::DaisyPetal, 196  
SW\_5  
    daisy::DaisyPetal, 196  
SW\_6  
    daisy::DaisyPetal, 196  
SW\_7  
    daisy::DaisyPetal, 196  
SW\_LAST  
    daisy::DaisyField, 171  
                daisy::DaisyPetal, 196  
                daisy::RingBuffer< T, size >, 326  
SwapBuffersAndTransmit  
    daisy::LedDriverPca9685< numDrivers, persistent-  
        BufferContents >, 258  
switches  
    daisy::DaisyPetal, 203  
SwitchPressed  
    daisy::DaisyVersio, 221  
Sync  
    daisy::SaiHandle::Config, 152  
SysClkFreq  
    daisy::System::Config, 157  
sysex\_data  
    daisy::MidiEvent, 280  
SysExEnd  
    daisy, 107  
SYSTEM, 40  
    dsy\_dma\_clear\_cache\_for\_buffer, 41  
    dsy\_dma\_deinit, 41  
    dsy\_dma\_init, 41  
    dsy\_dma\_invalidate\_cache\_for\_buffer, 41  
system  
    daisy::patch\_sm::DaisyPatchSM, 193  
SystemCommon  
    daisy, 107  
SystemCommonLast  
    daisy, 107  
SystemCommonType  
    daisy, 107  
SystemExclusive  
    daisy, 107  
SystemRealTime  
    daisy, 107  
SystemRealTimeLast  
    daisy, 108  
SystemRealTimeType  
    daisy, 108  
target\_samplerate  
    daisy::DacHandle::Config, 141  
text  
    daisy::AbstractMenu::ItemConfig, 250  
threshold  
    Dac, 98  
TIM\_2  
    daisy::TimerHandle::Config, 159  
TIM\_3  
    daisy::TimerHandle::Config, 159  
TIM\_4  
    daisy::TimerHandle::Config, 159  
TIM\_5  
    daisy::TimerHandle::Config, 159  
TimeHeldMs  
    daisy::Encoder, 227  
    daisy::Switch, 356  
TimingClock  
    daisy, 108

TRANSMIT  
 daisy::I2CHandle, 245

Transmit  
 daisy::LoggerImpl< dest >, 261  
 daisy::LoggerImpl< LOGGER\_EXTERNAL >, 262  
 daisy::LoggerImpl< LOGGER\_INTERNAL >, 263  
 daisy::LoggerImpl< LOGGER\_SEMIHOST >, 264

TransmitBlocking  
 daisy::I2CHandle, 247

TransmitBuf  
 EXTERNAL, 34

TransmitDma  
 daisy::I2CHandle, 247

TransmitExternal  
 UsbHandle, 390

TransmitInternal  
 UsbHandle, 390, 391

TransmitSync  
 EXTERNAL, 34

Trig  
 daisy::GateIn, 241

TuneRequest  
 daisy, 107

TwelveBitUintToVolts  
 Dac, 95

TX  
 daisy::SpiHandle, 343

tx  
 daisy::UartHandler::Config, 160

tx\_buff\_  
 EXTERNAL, 34

tx\_ptr\_  
 EXTERNAL, 35

Type  
 daisy::Switch, 354

type  
 daisy::AbstractMenu::ItemConfig, 250  
 daisy::MidiEvent, 280

TYPE\_MOMENTARY  
 daisy::Switch, 354

TYPE\_TOGGLE  
 daisy::Switch, 354

u82f  
 UTILITY, 89

U82F\_SCALE  
 UTILITY, 79

UI, 70

up  
 daisy, 105

Update  
 Dac, 95  
 daisy::Led, 254  
 daisy::OledDisplay< DisplayDriver >, 288  
 daisy::OneBitGraphicsDisplay, 294  
 daisy::RgbLed, 321  
 daisy::ShiftRegister4021< num\_daisychained, num\_parallel >, 337

daisy::SSD130xDriver< width, height, Transport >, 347

UpdateLeds  
 daisy::DaisyPetal, 202  
 daisy::DaisyPod, 208  
 daisy::DaisyVersio, 221

updateRateMs\_  
 daisy::UiCanvasDescriptor, 373

upDownSelectLeftRightModify  
 daisy::AbstractMenu, 115

usb\_handle  
 daisy::DaisySeed, 217

usb\_handle\_  
 daisy::LoggerImpl< LOGGER\_EXTERNAL >, 262  
 daisy::LoggerImpl< LOGGER\_INTERNAL >, 264

UsbHandle, 386  
 DelInit, 388  
 FS\_BOTH, 388  
 FS\_EXTERNAL, 388  
 FS\_INTERNAL, 388  
 Init, 389  
 ReceiveCallback, 387  
 Result, 387  
 SetReceiveCallback, 389, 390  
 TransmitExternal, 390  
 TransmitInternal, 390, 391  
 UsbPeriph, 387, 388

UsbPeriph  
 UsbHandle, 387, 388

user\_led  
 daisy::patch\_sm::DaisyPatchSM, 193

UTILITY, 73  
 BSP\_SD\_AbortCallback, 80  
 BSP\_SD\_CardInfo, 76  
 BSP\_SD\_Erase, 80  
 BSP\_SD\_GetCardInfo, 80  
 BSP\_SD\_GetCardState, 81  
 BSP\_SD\_Init, 81  
 BSP\_SD\_IsDetected, 81  
 BSP\_SD\_ITConfig, 81  
 BSP\_SD\_ReadBlocks, 82  
 BSP\_SD\_ReadBlocks\_DMA, 82  
 BSP\_SD\_ReadCpltCallback, 83  
 BSP\_SD\_WriteBlocks, 83  
 BSP\_SD\_WriteBlocks\_DMA, 83  
 BSP\_SD\_WriteCpltCallback, 84  
 cube, 84  
 DMA\_BUFFER\_MEM\_SECTION, 76  
 dsy\_get\_unique\_id, 84  
 DSY\_GPIO\_LAST, 80  
 dsy\_gpio\_port, 79  
 DSY\_GPIOA, 80  
 DSY\_GPIOB, 80  
 DSY\_GPIOC, 80  
 DSY\_GPIOD, 80  
 DSY\_GPIOE, 80  
 DSY\_GPIOF, 80  
 DSY\_GPIOG, 80

DSY\_GPIOH, 80  
DSY\_GPIOI, 80  
DSY\_GPIOJ, 80  
DSY\_GPIOK, 80  
dsy\_hal\_map\_get\_pin, 85  
dsy\_hal\_map\_get\_port, 85  
dsy\_hal\_map\_gpio\_clk\_enable, 85  
dsy\_pin, 86  
dsy\_pin\_cmp, 86  
DTCM\_MEM\_SECTION, 76  
f2s16, 86  
F2S16\_SCALE, 76  
f2s24, 86  
F2S24\_SCALE, 76  
f2s32, 87  
F2S32\_SCALE, 76  
f2s8, 87  
F2S8\_SCALE, 76  
f2u8, 87  
F2U8\_SCALE, 76  
FBIPMAX, 77  
FBIPMIN, 77  
Font\_11x18, 89  
Font\_16x26, 89  
Font\_6x8, 90  
Font\_7x10, 90  
IN\_L, 77  
IN\_R, 77  
MSD\_ERROR, 77  
MSD\_ERROR\_SD\_NOT\_PRESENT, 77  
MSD\_OK, 77  
OUT\_L, 78  
OUT\_R, 78  
s162f, 88  
S162F\_SCALE, 78  
s242f, 88  
S242F\_SCALE, 78  
S24SIGN, 78  
s322f, 88  
S322F\_SCALE, 78  
s82f, 89  
S82F\_SCALE, 78  
SD\_DATATIMEOUT, 78  
SD\_NOT\_PRESENT, 79  
SD\_PRESENT, 79  
SD\_TRANSFER\_BUSY, 79  
SD\_TRANSFER\_OK, 79  
u82f, 89  
U82F\_SCALE, 79

ValidateQSPI  
    daisy::patch\_sm::DaisyPatchSM, 191

ValidateSDRAM  
    daisy::patch\_sm::DaisyPatchSM, 192

Value  
    daisy::AnalogControl, 131  
    daisy::Parameter, 302

value  
    Dac, 98

daisy::ControlChangeEvent, 163  
daisy::MTCQuarterFrameEvent, 285  
daisy::PitchBendEvent, 307  
daisy::ResetAllControllersEvent, 319

valueEncoderId  
    daisy::UI::SpecialControlIds, 341

valueItem  
    daisy::AbstractMenu, 114

valuePotId  
    daisy::UI::SpecialControlIds, 341

valueToModify  
    daisy::AbstractMenu::ItemConfig, 250, 251

VegasMode  
    daisy::DaisyField, 179

velocity  
    daisy::NoteOffEvent, 286  
    daisy::NoteOnEvent, 286

VERY\_FAST  
    daisy::SdmmcHandler, 335

VoltageRange  
    Dac, 92

VoltsTo12BitUint  
    Dac, 96

WavFileFormatCode  
    daisy, 108

WHITE  
    daisy::Color, 139

WordLength  
    daisy::Wm8731::Config, 162

writable  
    daisy::RingBuffer< T, 0 >, 330  
    daisy::RingBuffer< T, size >, 326

Write  
    daisy::GPIO, 243  
    daisy::QSPIHandle, 315  
    daisy::RingBuffer< T, 0 >, 330  
    daisy::RingBuffer< T, size >, 327  
    daisy::WavWriter< transfer\_size >, 404  
    ShiftRegister595, 339

WRITE\_DISABLE\_CMD  
    FLASH, 66

WRITE\_ENABLE\_CMD  
    FLASH, 67

WRITE\_EXT\_NV\_READ\_PARAM\_REG\_CMD  
    FLASH, 67

WRITE\_EXT\_READ\_PARAM\_REG\_CMD  
    FLASH, 67

WRITE\_FUNCTION\_REGISTER  
    FLASH, 67

WRITE\_NV\_READ\_PARAM\_REG\_CMD  
    FLASH, 67

WRITE\_READ\_PARAM\_REG\_CMD  
    FLASH, 67, 68

WRITE\_STATUS\_REG\_CMD  
    FLASH, 68

WriteAnalogPinRaw  
    Dac, 96

WriteAnalogPinVolts

Dac, [97](#)  
WriteChar  
    daisy::OneBitGraphicsDisplay, [294](#)  
    daisy::OneBitGraphicsDisplayImpl< ChildType >,  
        [298](#)  
WriteCvOut  
    daisy::patch\_sm::DaisyPatchSM, [192](#)  
WriteDataAtAddress  
    daisy::I2CHandle, [248](#)  
WriteDigitalPin  
    Dac, [97](#)  
WritePage  
    daisy::QSPIHandle, [315](#)  
WriteString  
    daisy::OneBitGraphicsDisplay, [295](#)  
    daisy::OneBitGraphicsDisplayImpl< ChildType >,  
        [299](#)  
WriteStringAligned  
    daisy::OneBitGraphicsDisplay, [295](#)  
    daisy::OneBitGraphicsDisplayImpl< ChildType >,  
        [299](#)  
WriteValue  
    daisy::DacHandle, [170](#)