

MAGResWCHDLL

Generated by Doxygen 1.13.2

1 File Index	1
1.1 File List	1
2 File Documentation	3
2.1 MAGResWCH.h File Reference	3
2.1.1 Detailed Description	7
2.1.2 Enumeration Type Documentation	7
2.1.2.1 dio_errors_e	7
2.1.2.2 dio_ext_trigger_modes_e	7
2.1.2.3 dio_oled_write_mode_e	8
2.1.3 Function Documentation	8
2.1.3.1 DioArm()	8
2.1.3.2 DioClearCtrlMemory()	8
2.1.3.3 DioClearOLEDBuff()	8
2.1.3.4 DIOClearOLEDImm()	9
2.1.3.5 DioClearOutMemory()	9
2.1.3.6 DioClose()	9
2.1.3.7 DioCloseBMP()	9
2.1.3.8 DioGetErrorStr()	10
2.1.3.9 DioHalt()	10
2.1.3.10 DioInitialize()	10
2.1.3.11 DioOpenBMP()	11
2.1.3.12 DioPause()	11
2.1.3.13 DioReadCtrlMemory()	11
2.1.3.14 DioReadCurrentIO()	12
2.1.3.15 DioReadMaxDataLines()	12
2.1.3.16 DioReadOutMemory()	12
2.1.3.17 DioReadProgramCounter()	13
2.1.3.18 DioReadSPI()	13
2.1.3.19 DioReadStatusRegister()	14
2.1.3.20 DioReset()	14
2.1.3.21 DioResetDDS()	14
2.1.3.22 DioSendOLEDBuff()	15
2.1.3.23 DioSetupOLED()	15
2.1.3.24 DioSetupTriggerMode()	15
2.1.3.25 DioStep()	16
2.1.3.26 DioStreamSPI()	16
2.1.3.27 DioTrig()	16
2.1.3.28 DioWriteCtrlMemory()	17
2.1.3.29 DioWriteCurrentIO()	17
2.1.3.30 DioWriteOLED2DUnpackedArrImm()	17
2.1.3.31 DioWriteOLEDAreaArrBuff()	18

2.1.3.32 DioWriteOLEDAreaArrImm()	18
2.1.3.33 DioWriteOLEDAreaBMPBuff()	19
2.1.3.34 DioWriteOLEDAreaBMPImm()	19
2.1.3.35 DioWriteOLEDArrBuff()	20
2.1.3.36 DioWriteOLEDArrImm()	20
2.1.3.37 DioWriteOLEDCharBuff()	21
2.1.3.38 DioWriteOLEDCharImm()	21
2.1.3.39 DioWriteOLEDLineBuff()	21
2.1.3.40 DioWriteOLEDLineImm()	22
2.1.3.41 DioWriteOLEDPixelBuff()	22
2.1.3.42 DioWriteOLEDPixelImm()	23
2.1.3.43 DioWriteOLEDPixelPairBuff()	23
2.1.3.44 DioWriteOLEDPixelPairImm()	24
2.1.3.45 DioWriteOLEDRowArrBuff()	24
2.1.3.46 DioWriteOLEDRowArrImm()	24
2.1.3.47 DioWriteOLEDRowValImm()	25
2.1.3.48 DioWriteOLEDScreenArrBuff()	25
2.1.3.49 DioWriteOLEDScreenArrImm()	25
2.1.3.50 DioWriteOLEDStrBuff()	26
2.1.3.51 DioWriteOLEDStrImm()	26
2.1.3.52 DioWriteOutMemory()	26
2.1.3.53 DioWriteProgramCounter()	27
2.1.3.54 DioWriteSPI()	27
2.2 MAGResWCH.h	28

Index

33

Chapter 1

File Index

1.1 File List

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

MAGResWCH.h	
MAGResWCH DLL API for controlling the MAGRes board	3

Chapter 2

File Documentation

2.1 MAGResWCH.h File Reference

MAGResWCH DLL API for controlling the MAGRes board.

```
#include <stdint.h>
```

Macros

- #define **MAGRESWCH_API** __declspec(dllimport)

Typedefs

- typedef struct dio_handle_s * **dio_handle_h**
The board handle.
- typedef struct dio_bmp_s * **dio_bmp_h**
The handle for a bmp file.

Enumerations

- enum **dio_errors_e** {
ERR_CANNOT_ALLOCATE_LPNMASTERHANDLE = -1 ,
ERR_INVALID_CH367_HANDLE = -2 ,
ERR_CANNOT_GET_IO_REG_ADDR = -3 ,
ERR_STEP_SIZE_LESS_EQUAL_ZERO = -4 ,
ERR_WRITE_LONGER_THAN_AVAIL_MEM = -5 ,
ERR_WRITE_ZERO_BYTES = -6 ,
ERR_READ_ZERO_BYTES = -7 ,
ERR_SPI_READ_ERR = -8 ,
ERR_SPI_STREAM_ERR = -9 ,
ERR_SPI_WRITE_ERR = -10 ,
ERR_SPI_ADDR_OUT_OF_BOUNDS = -11 ,
ERR_BMP_INVALID_FILE = -12 }
Errors represented by lpnStatus.

- enum `dio_ext_trigger_modes_e` {
`DIO_EXT_TRIG_IGNORE` = 0 ,
`DIO_EXT_TRIG_RISING_EDGE` ,
`DIO_EXT_TRIG_FALLING_EDGE` ,
`DIO_EXT_TRIG_HIGH_HOLD` ,
`DIO_EXT_TRIG_LOW_HOLD` }
- enum `dio_adrsr_pin_state_e` {
`DIO_ADRSR_PIN_LOW` = 0 ,
`DIO_ADRSR_PIN_HIGH` }
- enum `dio_oled_write_mode_e` {
`DIO_OLED_MODE_OVERWRITE` = 0 ,
`DIO_OLED_MODE_OR` ,
`DIO_OLED_MODE_GREATER_THAN` ,
`DIO_OLED_MODE_XOR` ,
`DIO_OLED_MODE_NONZERO` }

External Trigger modes.

Pixel blending modes when using the OLED screen buffer.

Functions

- MAGRESWCH_API void `DioInitialize` (short nMasterNum, `dio_handle_h` *lpnMasterHandle, int16_t *lpnStatus)
Provides a one time initialization of a master board and all slave boards associated with it.
- MAGRESWCH_API void `DioArm` (`dio_handle_h` nMasterHandle, int16_t *lpnStatus)
Changes the board state from Halt to Pause.
- MAGRESWCH_API void `DioHalt` (`dio_handle_h` nMasterHandle, int16_t *lpnStatus)
Changes specified master board state to HALT.
- MAGRESWCH_API void `DioPause` (`dio_handle_h` nMasterHandle, int16_t *lpnStatus)
Changes the specified master board state from a RUN to PAUSE state.
- MAGRESWCH_API void `DioTrig` (`dio_handle_h` nMasterHandle, int16_t *lpnStatus)
Takes the specified master board state from PAUSE to RUN.
- MAGRESWCH_API void `DioReadCtrlMemory` (`dio_handle_h` nHandle, uint32_t *lpdwVector, uint32_t dwStart, uint32_t dwSize, int16_t *lpnStatus)
Reads a block of control memory from the specified board.
- MAGRESWCH_API void `DioReadOutMemory` (`dio_handle_h` nHandle, uint32_t *lpdwVector, uint32_t dwStart, uint32_t dwSize, int16_t *lpnStatus)
Reads a block of output memory from the specified board.
- MAGRESWCH_API void `DioWriteCtrlMemory` (`dio_handle_h` nHandle, uint32_t *lpdwMemory, uint32_t dwStart, uint32_t dwSize, int16_t *lpnStatus)
Writes a block of data to the control memory of the specified board.
- MAGRESWCH_API void `DioWriteOutMemory` (`dio_handle_h` nHandle, uint32_t *lpdwMemory, uint32_t dwStart, uint32_t dwSize, int16_t *lpnStatus)
Writes a block of data to the output memory of the specified board.
- MAGRESWCH_API void `DioReadProgramCounter` (`dio_handle_h` nHandle, int32_t *lpdwCounter, int16_t *lpnStatus)
Returns the specified master board program counter value.
- MAGRESWCH_API void `DioWriteProgramCounter` (`dio_handle_h` nHandle, int32_t dwCounter, int16_t *lpnStatus)
Writes the specified master board program counter value.
- MAGRESWCH_API void `DioReadCurrentIO` (`dio_handle_h` nHandle, int32_t *lpdwCounter, int16_t *lpnStatus)
Returns the specified master board current output value.
- MAGRESWCH_API void `DioWriteCurrentIO` (`dio_handle_h` nHandle, int32_t dwCounter, int16_t *lpnStatus)

- Writes the specified master board current output value.*
- MAGRESWCH_API void [DioReadStatusRegister](#) ([dio_handle_h](#) nHandle, int16_t *lpwData, int16_t *lpnStatus)
- Reads the status register from the specified board.*
- MAGRESWCH_API void [DioReset](#) ([dio_handle_h](#) nHandle, int16_t *lpnStatus)
- Changes state of specified master board to HALT.*
- MAGRESWCH_API void [DioStep](#) ([dio_handle_h](#) nMasterHandle, int32_t dwSteps, int16_t *lpnStatus)
- Executes the specified master board program for a number of steps.*
- MAGRESWCH_API void [DioReadMaxDataLines](#) ([dio_handle_h](#) nHandle, int32_t *lpdwCounter, int16_t *lpnStatus)
- Returns the specified master board max data lines available to program.*
- MAGRESWCH_API void [DioClearCtrlMemory](#) ([dio_handle_h](#) nMasterHandle, int16_t *lpnStatus)
- Clears the control memory on the board.*
- MAGRESWCH_API void [DioClearOutMemory](#) ([dio_handle_h](#) nMasterHandle, int16_t *lpnStatus)
- Clears the output memory on the board.*
- MAGRESWCH_API void [DioStreamSPI](#) ([dio_handle_h](#) nMasterHandle, uint8_t addr, uint8_t *write_buff, uint8_t *read_buff, size_t len, int16_t *lpnStatus)
- Streams the SPI.*
- MAGRESWCH_API void [DioWriteSPI](#) ([dio_handle_h](#) nMasterHandle, uint8_t addr, uint8_t *write_buff, size_t len, int16_t *lpnStatus)
- Writes to the SPI.*
- MAGRESWCH_API void [DioReadSPI](#) ([dio_handle_h](#) nMasterHandle, uint8_t addr, uint8_t *read_buff, size_t len, int16_t *lpnStatus)
- Reads the SPI.*
- MAGRESWCH_API void [DioResetDDS](#) ([dio_handle_h](#) nMasterHandle, int16_t *lpnStatus)
- Resets and initializes the DDS. Can only be sent during HALT.*
- MAGRESWCH_API void [DioSetupTriggerMode](#) ([dio_handle_h](#) nMasterHandle, uint8_t nPauseMode, uint8_t nRunMode, int16_t *lpnstatus)
- Sets the external trigger mode during the PAUSE and RUN states.*
- MAGRESWCH_API void [DioClose](#) ([dio_handle_h](#) nMasterHandle, int16_t *lpnStatus)
- Closes the connection to the board, and releases the handle from being used.*
- MAGRESWCH_API void [DioSetADRSR_Low](#) ([dio_handle_h](#) nMasterHandle, uint8_t io_byte)
- MAGRESWCH_API void [DioSetADRSR_High](#) ([dio_handle_h](#) nMasterHandle, uint8_t io_byte)
- MAGRESWCH_API void [DioSetupOLED](#) ([dio_handle_h](#) nMasterHandle, int16_t *lpnStatus)
- Sets up and turns on the OLED.*
- MAGRESWCH_API void [DioWriteOLEDPixelPairImm](#) ([dio_handle_h](#) nMasterHandle, uint8_t column, uint8_t y, uint8_t val)
- Writes a value to a pair of pixels immediately.*
- MAGRESWCH_API void [DioWriteOLEDPixelPairBuff](#) ([dio_handle_h](#) nMasterHandle, [dio_oled_write_mode_e](#) mode, uint8_t column, uint8_t y, uint8_t val)
- Writes a value to a pair of pixels in the local screen buffer.*
- MAGRESWCH_API void [DioWriteOLEDPixelImm](#) ([dio_handle_h](#) nMasterHandle, uint8_t row, uint8_t column, uint8_t val)
- Writes a pixel to the screen at the designated coordinate immediately.*
- MAGRESWCH_API void [DioWriteOLEDPixelBuff](#) ([dio_handle_h](#) nMasterHandle, [dio_oled_write_mode_e](#) mode, uint8_t x, uint8_t y, uint8_t val)
- Writes a pixel to the screen at the designated coordinate in the local screen buffer.*
- MAGRESWCH_API void [DioWriteOLEDLinImm](#) ([dio_handle_h](#) nMasterHandle, uint8_t x1, uint8_t y1, uint8_t x2, uint8_t y2, uint8_t val1, uint8_t val2)
- Immediately writes a line to the screen that begins at (x1, y1), with value val1, to (x2, y2) with value val2.*
- MAGRESWCH_API void [DioWriteOLEDLinBuff](#) ([dio_handle_h](#) nMasterHandle, [dio_oled_write_mode_e](#) mode, uint8_t x1, uint8_t y1, uint8_t x2, uint8_t y2, uint8_t val1, uint8_t val2)

- Writes a line to the local screen buffer that begins at (x1, y1), with value val1, to (x2, y2) with value val2.*
- MAGRESWCH_API void [DioWriteOLEDArrImm](#) ([dio_handle_h](#) nMasterHandle, uint8_t x, uint8_t y, uint8_t *arr, size_t len)
Writes an array of length len to screen immediately.
 - MAGRESWCH_API void [DioWriteOLEDArrBuff](#) ([dio_handle_h](#) nMasterHandle, [dio_oled_write_mode_e](#) mode, uint8_t x, uint8_t y, uint8_t *arr, size_t len)
Writes an array of length len to the local screen buffer.
 - MAGRESWCH_API void [DioWriteOLEDScreenArrImm](#) ([dio_handle_h](#) nMasterHandle, uint8_t *arr)
Writes 128x64 bytes to the screen at once, given a single dimensional array of that size.
 - MAGRESWCH_API void [DioWriteOLEDScreenArrBuff](#) ([dio_handle_h](#) nMasterHandle, [dio_oled_write_mode_e](#) mode, uint8_t *arr)
Writes 128x64 bytes to the local screen buffer, given a single dimensional array of that size.
 - MAGRESWCH_API void [DioWriteOLEDRowArrImm](#) ([dio_handle_h](#) nMasterHandle, uint8_t y, uint8_t *arr)
Writes an array of length 128 to a row on screen immediately.
 - MAGRESWCH_API void [DioWriteOLEDRowArrBuff](#) ([dio_handle_h](#) nMasterHandle, [dio_oled_write_mode_e](#) mode, uint8_t y, uint8_t *arr)
Writes an array of length 128 to a row in the local screen buffer.
 - MAGRESWCH_API void [DioWriteOLEDRowVallImm](#) ([dio_handle_h](#) nMasterHandle, uint8_t y, uint8_t val)
Writes singular value to a row on screen immediately.
 - MAGRESWCH_API void [DioWriteOLEDAreaArrImm](#) ([dio_handle_h](#) nMasterHandle, uint8_t x, uint8_t y, uint8_t col_width, uint8_t height, uint8_t *arr)
*Writes an array of length col_width*height to the screen immediately.*
 - MAGRESWCH_API void [DioWriteOLEDAreaArrBuff](#) ([dio_handle_h](#) nMasterHandle, [dio_oled_write_mode_e](#) mode, uint8_t x, uint8_t y, uint8_t col_width, uint8_t height, uint8_t *arr)
*Writes an array of length col_width*height to the local screen buffer.*
 - MAGRESWCH_API void [DioWriteOLED2DUnpackedArrImm](#) ([dio_handle_h](#) nMasterHandle, uint8_t **arr)
Writes 128x64 bytes to the screen at once, given a single dimensional array of that size.
 - MAGRESWCH_API void [DioSendOLEDBuff](#) ([dio_handle_h](#) nMasterHandle)
Sends the current local screen buffer to the OLED screen.
 - MAGRESWCH_API void [DioClearOLEDBuff](#) ([dio_handle_h](#) nMasterHandle)
Clears the current local screen buffer.
 - MAGRESWCH_API void [DIOClearOLEDImm](#) ([dio_handle_h](#) nMasterHandle)
Clears the screen immediately.
 - MAGRESWCH_API [dio_bmp_h](#) [DioOpenBMP](#) ([dio_handle_h](#) nMasterHandle, char *path, int16_t *lpn← Status)
Opens and allocates memory for a bitmap file.
 - MAGRESWCH_API void [DioCloseBMP](#) ([dio_handle_h](#) nMasterHandle, [dio_bmp_h](#) *bmp_handle)
Closes and deallocates memory for a bitmap file.
 - MAGRESWCH_API void [DioWriteOLEDAreaBMPImm](#) ([dio_handle_h](#) nMasterHandle, uint8_t x, uint8_t y, [dio_bmp_h](#) bmp_handle)
Writes a BMP to the screen immediately at an x and y offset.
 - MAGRESWCH_API void [DioWriteOLEDAreaBMPBuff](#) ([dio_handle_h](#) nMasterHandle, [dio_oled_write_mode_e](#) mode, uint8_t x, uint8_t y, [dio_bmp_h](#) bmp_handle)
Writes a BMP to the local screen buffer at an x and y offset.
 - MAGRESWCH_API void [DioWriteOLEDCharImm](#) ([dio_handle_h](#) nMasterHandle, uint8_t x, uint8_t y, char letter)
Writes a single character immediately to the screen.
 - MAGRESWCH_API void [DioWriteOLEDCharBuff](#) ([dio_handle_h](#) nMasterHandle, [dio_oled_write_mode_e](#) mode, uint8_t x, uint8_t y, char letter)
Writes a single character to the local screen buffer.
 - MAGRESWCH_API void [DioWriteOLEDStrImm](#) ([dio_handle_h](#) nMasterHandle, uint8_t x, uint8_t y, char *str)
Writes a string immediately to the screen.

- MAGRESWCH_API void [DioWriteOLEDStrBuff](#) ([dio_handle_h](#) nMasterHandle, [dio_oled_write_mode_e](#) mode, uint8_t x, uint8_t y, char *str)
Writes a string to the local screen buffer.
- MAGRESWCH_API const char * [DioGetErrorStr](#) ([dio_handle_h](#) nMasterHandle, int16_t lpnStatus)
Returns a null-terminated string associated with an `lpnStatus` error.

2.1.1 Detailed Description

MAGResWCH DLL API for controlling the MAGRes board.

2.1.2 Enumeration Type Documentation

2.1.2.1 `dio_errors_e`

enum [dio_errors_e](#)

Errors represented by `lpnStatus`.

Enumerator

<code>ERR_CANNOT_ALLOCATE_LPNMASTERHANDLE</code>	Cannot allocate <code>lpnMasterHandle</code> .
<code>ERR_INVALID_CH367_HANDLE</code>	Invalid CH367 handle returned.
<code>ERR_CANNOT_GET_IO_REG_ADDR</code>	Cannot get <code>io_reg</code> address for the CH367.
<code>ERR_STEP_SIZE_LESS_EQUAL_ZERO</code>	Commanded step size is less than or equal to zero.
<code>ERR_WRITE_LONGER_THAN_AVAIL_MEM</code>	Number of bytes to write is larger than available memory.
<code>ERR_WRITE_ZERO_BYTES</code>	Number of bytes to write is zero, needs to be at least 1.
<code>ERR_READ_ZERO_BYTES</code>	Number of bytes to read is specified to be zero, needs to be at least 1.
<code>ERR_SPI_READ_ERR</code>	Cannot read SPI.
<code>ERR_SPI_STREAM_ERR</code>	Cannot stream SPI.
<code>ERR_SPI_WRITE_ERR</code>	Cannot write to the SPI.
<code>ERR_SPI_ADDR_OUT_OF_BOUNDS</code>	Tried to access an invalid SPI address.
<code>ERR_BMP_INVALID_FILE</code>	Tried to open invalid BMP file.

2.1.2.2 `dio_ext_trigger_modes_e`

enum [dio_ext_trigger_modes_e](#)

External Trigger modes.

Enumerator

<code>DIO_EXT_TRIG_IGNORE</code>	Ignore the external trigger line.
<code>DIO_EXT_TRIG_RISING_EDGE</code>	Trigger and change state on the rising edge.
<code>DIO_EXT_TRIG_FALLING_EDGE</code>	Trigger and change state on the falling edge.
<code>DIO_EXT_TRIG_HIGH_HOLD</code>	Trigger and change state when the line is held high.
<code>DIO_EXT_TRIG_LOW_HOLD</code>	Trigger and change state when the line is held low.

2.1.2.3 dio_oled_write_mode_e

enum `dio_oled_write_mode_e`

Pixel blending modes when using the OLED screen buffer.

Enumerator

DIO_OLED_MODE_OVERWRITE	Overwrites what's there.
DIO_OLED_MODE_OR	Or current pixel with new pixel value.
DIO_OLED_MODE_GREATER_THAN	Overwrite current pixel if new pixel is greater than existing pixel.
DIO_OLED_MODE_XOR	Xor current pixel with new pixel value.
DIO_OLED_MODE_NONZERO	Overwrite current pixel with new pixel value as long as the new value is nonzero.

2.1.3 Function Documentation

2.1.3.1 DioArm()

```
MAGRESWCH_API void DioArm (
    dio_handle_h nMasterHandle,
    int16_t * lpnStatus)
```

Changes the board state from Halt to Pause.

The board must be in HALT state, otherwise it does nothing.

Parameters

<i>nMasterHandle</i>	master board handle
<i>lpnStatus</i>	return status, 0 on success, <0 on failure

2.1.3.2 DioClearCtrlMemory()

```
MAGRESWCH_API void DioClearCtrlMemory (
    dio_handle_h nMasterHandle,
    int16_t * lpnStatus)
```

Clears the control memory on the board.

Parameters

<i>nMasterHandle</i>	master board handle
<i>lpnStatus</i>	returned status, 0 on success < 0 on failure

2.1.3.3 DioClearOLEDBuff()

```
MAGRESWCH_API void DioClearOLEDBuff (
    dio_handle_h nMasterHandle)
```

Clears the current local screen buffer.

Parameters

<i>nMasterHandle</i>	master board handle
----------------------	---------------------

2.1.3.4 DIOClearOLEDimm()

```
MAGRESWCH_API void DIOClearOLEDimm (  
    dio_handle_h nMasterHandle)
```

Clears the screen immediately.

Parameters

<i>nMasterHandle</i>	master board handle
----------------------	---------------------

2.1.3.5 DioClearOutMemory()

```
MAGRESWCH_API void DioClearOutMemory (  
    dio_handle_h nMasterHandle,  
    int16_t * lpnStatus)
```

Clears the output memory on the board.

Parameters

<i>nMasterHandle</i>	master board handle
<i>lpnStatus</i>	returned status, 0 on success < 0 on failure

2.1.3.6 DioClose()

```
MAGRESWCH_API void DioClose (  
    dio_handle_h nMasterHandle,  
    int16_t * lpnStatus)
```

Closes the connection to the board, and releases the handle from being used.

Passing a handle of 0 or NULL to DioClose will perform no action.

Parameters

<i>nMasterHandle</i>	master board handle
<i>lpnStatus</i>	returned status, 0 on success < 0 on failure

2.1.3.7 DioCloseBMP()

```
MAGRESWCH_API void DioCloseBMP (  
    dio_handle_h nMasterHandle,  
    dio_bmp_h * bmp_handle)
```

Closes and deallocates memory for a bitmap file.

Parameters

<i>nMasterHandle</i>	master board handle
<i>bmp_handle</i>	handle of the bmp file

2.1.3.8 DioGetErrorStr()

```
MAGRESWCH_API const char * DioGetErrorStr (  
    dio_handle_h nMasterHandle,  
    int16_t lpnStatus)
```

Returns a null-terminated string associated with an *lpnStatus* error.

Parameters

<i>nMasterHandle</i>	master board handle
<i>lpnStatus</i>	<i>lpnStatus</i> , assumed to be in error

Return values

<i>returns</i>	null terminated error string
----------------	------------------------------

2.1.3.9 DioHalt()

```
MAGRESWCH_API void DioHalt (  
    dio_handle_h nMasterHandle,  
    int16_t * lpnStatus)
```

Changes specified master board state to HALT.

Change board program counter value to zero. Calling Dio Arm then DioTrig after the board is in a halt state will cause the board to run from the beginning.

Parameters

<i>nMasterHandle</i>	master board handle
<i>lpnStatus</i>	returned status, 0 on success < 0 on failure

2.1.3.10 DioInitialize()

```
MAGRESWCH_API void DioInitialize (  
    short nMasterNum,  
    dio_handle_h * lpnMasterHandle,  
    int16_t * lpnStatus)
```

Provides a one time initialization of a master board and all slave boards associated with it.

Returns the handle to the board.

Initialize the DIO system configuration with one command.

Parameters

<i>nMasterNum</i>	number of the master board in the sytem.
<i>lpnMasterHandle</i>	Returned master handle, 0 or NULL on error.
<i>lpnStatus</i>	returned status, 0 on success, < 0 failure.

2.1.3.11 DioOpenBMP()

```
MAGRESWCH_API dio_bmp_h DioOpenBMP (  
    dio_handle_h nMasterHandle,  
    char * path,  
    int16_t * lpnStatus)
```

Opens and allocates memory for a bitmap file.

Parameters

<i>nMasterHandle</i>	master board handle
<i>path</i>	null terminated path to the file
<i>lpnStatus</i>	returned status, 0 on success, < 0 failure.

Return values

<i>returns</i>	a handle to the bmp data
----------------	--------------------------

2.1.3.12 DioPause()

```
MAGRESWCH_API void DioPause (  
    dio_handle_h nMasterHandle,  
    int16_t * lpnStatus)
```

Changes the specified master board state from a RUN to PAUSE state.

Board must be in a RUN state, otherwise the function does nothing.

Parameters

<i>nMasterHandle</i>	master board handle
<i>lpnStatus</i>	returned status, 0 on success, < 0 failure.

2.1.3.13 DioReadCtrlMemory()

```
MAGRESWCH_API void DioReadCtrlMemory (  
    dio_handle_h nHandle,  
    uint32_t * lnpdwVector,  
    uint32_t dwStart,  
    uint32_t dwSize,  
    int16_t * lpnStatus)
```

Reads a block of control memory from the specified board.

Board must be in a HALT state before calling these functions.

Parameters

<i>nHandle</i>	master board handle
<i>lnpdwVector</i>	Array of double words up to 16K elements
<i>dwStart</i>	Starting address to read
<i>dwSize</i>	Number of steps to read
<i>lpnStatus</i>	returned status, 0 on success, < 0 failure, > 0 number of bytes read

2.1.3.14 DioReadCurrentIO()

```
MAGRESWCH_API void DioReadCurrentIO (
    dio_handle_h nHandle,
    int32_t * lpdwCounter,
    int16_t * lpnStatus)
```

Returns the specified master board current output value.

Parameters

<i>nHandle</i>	master board handle
<i>lpdwCounter</i>	returned program counter value
<i>lpnStatus</i>	returned status, 0 on success, < 0 failure, > 0 num bytes read

2.1.3.15 DioReadMaxDataLines()

```
MAGRESWCH_API void DioReadMaxDataLines (
    dio_handle_h nHandle,
    int32_t * lpdwCounter,
    int16_t * lpnStatus)
```

Returns the specified master board max data lines available to program.

Parameters

<i>nHandle</i>	master board handle
<i>lpdwCounter</i>	returned data lines value
<i>lpnStatus</i>	returned status, 0 on success, < 0 failure, > 0 num bytes read

2.1.3.16 DioReadOutMemory()

```
MAGRESWCH_API void DioReadOutMemory (
    dio_handle_h nHandle,
    uint32_t * lnpdwVector,
    uint32_t dwStart,
    uint32_t dwSize,
    int16_t * lpnStatus)
```

Reads a block of output memory from the specified board.

Board must be in a HALT state before calling these functions.

Parameters

<i>nHandle</i>	master board handle
<i>lndwVector</i>	Array of double words up to 16K elements
<i>dwStart</i>	Starting address to read
<i>dwSize</i>	Number of steps to read
<i>lpnStatus</i>	returned status, 0 on success, < 0 failure, > 0 number of bytes read

2.1.3.17 DioReadProgramCounter()

```
MAGRESWCH_API void DioReadProgramCounter (
    dio_handle_h nHandle,
    int32_t * lpdwCounter,
    int16_t * lpnStatus)
```

Returns the specified master board program counter value.

Represents the address of the current step being executed.

Parameters

<i>nHandle</i>	master board handle
<i>lpdwCounter</i>	returned program counter value
<i>lpnStatus</i>	returned status, 0 on success, < 0 failure, > 0 num bytes read

2.1.3.18 DioReadSPI()

```
MAGRESWCH_API void DioReadSPI (
    dio_handle_h nMasterHandle,
    uint8_t addr,
    uint8_t * read_buff,
    size_t len,
    int16_t * lpnStatus)
```

Reads the SPI.

Accomplishes this by writing zeroes the SPI peripheral at the address, and fill `read_buff` with any incoming data

Address 0 is the FPGA, and it is recommended to use the DIO API to communicate with it.

Parameters

<i>nMasterHandle</i>	master board handle
<i>addr</i>	address of the SPI peripheral 0-7.
<i>write_buff</i>	pointer to byte-array of what to write
<i>read_buff</i>	pointer to byte-array of what to read
<i>len</i>	length of both the read and write buffer
<i>lpnStatus</i>	returned status, 0 on success < 0 on failure

2.1.3.19 DioReadStatusRegister()

```
MAGRESWCH_API void DioReadStatusRegister (
    dio_handle_h nHandle,
    int16_t * lpwData,
    int16_t * lpnStatus)
```

Reads the status register from the specified board.

bits	id	description
15-5	R	Reserved
4-2	STATE	board state↔ : 000 HALT 011 Pause 1xx Run
1-0	UNUSED	

Parameters

<i>nHandle</i>	master board handle
<i>lpwData</i>	returned status register value
<i>lpnStatus</i>	returned status, 0 on success, < 0 failure.

2.1.3.20 DioReset()

```
MAGRESWCH_API void DioReset (
    dio_handle_h nHandle,
    int16_t * lpnStatus)
```

Changes state of specified master board to HALT.

Parameters

<i>nHandle</i>	master board handle
<i>lpnStatus</i>	returned status, 0 on success, < 0 failure.

2.1.3.21 DioResetDDS()

```
MAGRESWCH_API void DioResetDDS (
    dio_handle_h nMasterHandle,
    int16_t * lpnStatus)
```

Resets and initializes the DDS. Can only be sent during HALT.

Parameters

<i>nMasterHandle</i>	master board handle
<i>lpnStatus</i>	returned status, 0 on success < 0 on failure

2.1.3.22 DioSendOLEDBuff()

```
MAGRESWCH_API void DioSendOLEDBuff (
    dio_handle_h nMasterHandle)
```

Sends the current local screen buffer to the OLED screen.

Parameters

<i>nMasterHandle</i>	master board handle
----------------------	---------------------

2.1.3.23 DioSetupOLED()

```
MAGRESWCH_API void DioSetupOLED (
    dio_handle_h nMasterHandle,
    int16_t * lpnStatus)
```

Sets up and turns on the OLED.

Must be called before any function manipulating the screen is called.

Parameters

<i>nMasterHandle</i>	master board handle
<i>lpnStatus</i>	returned status, 0 on success < 0 on failure

2.1.3.24 DioSetupTriggerMode()

```
MAGRESWCH_API void DioSetupTriggerMode (
    dio_handle_h nMasterHandle,
    uint8_t nPauseMode,
    uint8_t nRunMode,
    int16_t * lpnstatus)
```

Sets the external trigger mode during the PAUSE and RUN states.

The board can be triggered from an external line, but defaults to ignoring any changes on this external line. This function sets the action of the board in either the PAUSE or the RUN state. The different functionality is described in [dio_ext_trigger_modes_e](#)

The PAUSE state will transition to the RUN state depending on the value sent in *nPauseMode*. The RUN state will transition to the PAUSE state depending on the value sent in *nRunMode*.

Parameters

<i>nMasterHandle</i>	master board handle
<i>nPauseMode</i>	trigger mode during the pause state
<i>nRunMode</i>	trigger mode during the run state
<i>lpnStatus</i>	returned status, 0 on success < 0 on failure

2.1.3.25 DioStep()

```
MAGRESWCH_API void DioStep (
    dio_handle_h nMasterHandle,
    int32_t dwSteps,
    int16_t * lpnStatus)
```

Executes the specified master board program for a number of steps.

Board must be in a PAUSE state.

Each step is executed by changing the board state from PAUSE to RUN state and then to PAUSE state again.

Parameters

<i>nMasterHandle</i>	master board handle
<i>dwSteps</i>	Number of steps to execute
<i>lpnStatus</i>	returned status, 0 on success, < 0 failure.

2.1.3.26 DioStreamSPI()

```
MAGRESWCH_API void DioStreamSPI (
    dio_handle_h nMasterHandle,
    uint8_t addr,
    uint8_t * write_buff,
    uint8_t * read_buff,
    size_t len,
    int16_t * lpnStatus)
```

Streams the SPI.

Writes the contents of *write_buff* to the SPI peripheral at the given address, and read any incoming data that is coming in simultaneously to fill the *read_buff*.

Address 0 is the FPGA, and it is recommended to use the DIO API to communicate with it.

Parameters

<i>nMasterHandle</i>	master board handle
<i>addr</i>	address of the SPI peripheral 0-7.
<i>write_buff</i>	pointer to byte-array of what to write
<i>read_buff</i>	pointer to byte-array of what to read
<i>len</i>	length of both the read and write buffer
<i>lpnStatus</i>	returned status, 0 on success < 0 on failure

2.1.3.27 DioTrig()

```
MAGRESWCH_API void DioTrig (
    dio_handle_h nMasterHandle,
    int16_t * lpnStatus)
```

Takes the specified master board state from PAUSE to RUN.

Parameters

<i>nMasterHandle</i>	master board handle
<i>lpnStatus</i>	returned status, 0 on success, < 0 failure.

2.1.3.28 DioWriteCtrlMemory()

```
MAGRESWCH_API void DioWriteCtrlMemory (
    dio_handle_h nHandle,
    uint32_t * lpdwMemory,
    uint32_t dwStart,
    uint32_t dwSize,
    int16_t * lpnStatus)
```

Writes a block of data to the control memory of the specified board.

Board must be in a HALT state before calling these functions.

Parameters

<i>nMasterHandle</i>	master board handle
<i>lpdwMemory</i>	array of double words
<i>dwStart</i>	starting address to write
<i>dwSize</i>	number of steps to write.
<i>lpnStatus</i>	returned status, 0 on success, < 0 failure.

2.1.3.29 DioWriteCurrentIO()

```
MAGRESWCH_API void DioWriteCurrentIO (
    dio_handle_h nHandle,
    int32_t dwCounter,
    int16_t * lpnStatus)
```

Writes the specified master board current output value.

The board must be in pause.

Parameters

<i>nHandle</i>	master board handle
<i>dwCounter</i>	new program counter value
<i>lpnStatus</i>	returned status, 0 on success, < 0 failure, > 0 num bytes read

2.1.3.30 DioWriteOLED2DUnpackedArrImm()

```
MAGRESWCH_API void DioWriteOLED2DUnpackedArrImm (
    dio_handle_h nMasterHandle,
    uint8_t ** arr)
```

Writes 128x64 bytes to the screen at once, given a single dimensional array of that size.

The array MUST be that size, otherwise program will segfault.

Parameters

<i>nMasterHandle</i>	master board handle
<i>arr</i>	array to write

2.1.3.31 DioWriteOLEDAreaArrBuff()

```
MAGRESWCH_API void DioWriteOLEDAreaArrBuff (
    dio_handle_h nMasterHandle,
    dio_oled_write_mode_e mode,
    uint8_t x,
    uint8_t y,
    uint8_t col_width,
    uint8_t height,
    uint8_t * arr)
```

Writes an array of length `col_width*height` to the local screen buffer.

The array is one dimensional, but written to the screen two-dimensionally at a y offset and single pixel offset in the x direction. The array is split into strips of `col_width`, and is written in `height` rows. The array should be formatted such that the first byte is the top-left of the image, and the last byte is the bottom right.

Undefined behavior if coordinates are outside bounds of display

Parameters

<i>nMasterHandle</i>	master board handle
<i>mode</i>	buffer overwrite mode
<i>x</i>	x offset
<i>y</i>	y offset
<i>col_width</i>	width of image to write
<i>height</i>	height of image to write
<i>arr</i>	pointer to the array

2.1.3.32 DioWriteOLEDAreaArrImm()

```
MAGRESWCH_API void DioWriteOLEDAreaArrImm (
    dio_handle_h nMasterHandle,
    uint8_t x,
    uint8_t y,
    uint8_t col_width,
    uint8_t height,
    uint8_t * arr)
```

Writes an array of length `col_width*height` to the screen immediately.

The array is one dimensional, but written to the screen two-dimensionally at a y offset and single pixel offset in the x direction. The array is split into strips of `col_width`, and is written in `height` rows. The array should be formatted such that the first byte is the top-left of the image, and the last byte is the bottom right.

Undefined behavior if coordinates are outside bounds of display

Parameters

<i>nMasterHandle</i>	master board handle
<i>mode</i>	buffer overwrite mode
<i>x</i>	x offset
<i>y</i>	y offset
<i>col_width</i>	width of image to write
<i>height</i>	height of image to write
<i>arr</i>	pointer to the array

2.1.3.33 DioWriteOLEDAreaBMPBuff()

```
MAGRESWCH_API void DioWriteOLEDAreaBMPBuff (
    dio_handle_h nMasterHandle,
    dio_oled_write_mode_e mode,
    uint8_t x,
    uint8_t y,
    dio_bmp_h bmp_handle)
```

Writes a BMP to the local screen buffer at an x and y offset.

The origin of the BMP is the top left corner, and the offset is the distance between this corner and the BMP origin.

Parameters

<i>nMasterHandle</i>	master board handle
<i>mode</i>	buffer overwrite mode
<i>x</i>	x offset
<i>y</i>	y offset
<i>bmp_handle</i>	handle of the bmp file

2.1.3.34 DioWriteOLEDAreaBMPImm()

```
MAGRESWCH_API void DioWriteOLEDAreaBMPImm (
    dio_handle_h nMasterHandle,
    uint8_t x,
    uint8_t y,
    dio_bmp_h bmp_handle)
```

Writes a BMP to the screen immediately at an x and y offset.

The origin of the BMP is the top left corner, and the offset is the distance between this corner and the BMP origin.

Parameters

<i>nMasterHandle</i>	master board handle
<i>x</i>	x offset
<i>y</i>	y offset
<i>bmp_handle</i>	handle of the bmp file

2.1.3.35 DioWriteOLEDArrBuff()

```
MAGRESWCH_API void DioWriteOLEDArrBuff (
    dio_handle_h nMasterHandle,
    dio_oled_write_mode_e mode,
    uint8_t x,
    uint8_t y,
    uint8_t * arr,
    size_t len)
```

Writes an array of length len to the local screen buffer.

The array is written to the screen at a y offset and single pixel offset in the x direction, nibble calculations to move the array across by one pixel are done internally.

Undefined behavior if coordinates are outside bounds of display

Parameters

<i>nMasterHandle</i>	master board handle
<i>mode</i>	buffer overwrite mode
<i>x</i>	x offset
<i>y</i>	y offset
<i>arr</i>	pointer to the array
<i>len</i>	length of the array

2.1.3.36 DioWriteOLEDArrImm()

```
MAGRESWCH_API void DioWriteOLEDArrImm (
    dio_handle_h nMasterHandle,
    uint8_t x,
    uint8_t y,
    uint8_t * arr,
    size_t len)
```

Writes an array of length len to screen immediately.

The array is written to the screen at a y offset and single pixel offset in the x direction, nibble calculations to move the array across by one pixel are done internally.

Undefined behavior if coordinates are outside bounds of display

Parameters

<i>nMasterHandle</i>	master board handle
<i>x</i>	x offset
<i>y</i>	y offset
<i>arr</i>	pointer to the array
<i>len</i>	length of the array

2.1.3.37 DioWriteOLEDCharBuff()

```
MAGRESWCH_API void DioWriteOLEDCharBuff (
    dio_handle_h nMasterHandle,
    dio_oled_write_mode_e mode,
    uint8_t x,
    uint8_t y,
    char letter)
```

Writes a single character to the local screen buffer.

Parameters

<i>nMasterHandle</i>	master board handle
<i>x</i>	x offset
<i>y</i>	y offset
<i>letter</i>	letter to write

2.1.3.38 DioWriteOLEDCharImm()

```
MAGRESWCH_API void DioWriteOLEDCharImm (
    dio_handle_h nMasterHandle,
    uint8_t x,
    uint8_t y,
    char letter)
```

Writes a single character immediately to the screen.

Parameters

<i>nMasterHandle</i>	master board handle
<i>x</i>	x offset
<i>y</i>	y offset
<i>letter</i>	letter to write

2.1.3.39 DioWriteOLEDLineBuff()

```
MAGRESWCH_API void DioWriteOLEDLineBuff (
    dio_handle_h nMasterHandle,
    dio_oled_write_mode_e mode,
    uint8_t x1,
    uint8_t y1,
    uint8_t x2,
    uint8_t y2,
    uint8_t val1,
    uint8_t val2)
```

Writes a line to the local screen buffer that begins at (x1, y1), with value val1, to (x2, y2) with value val2.

val1 and val2 are linearly interpolated.

Undefined behavior if coordinates are outside bounds of display

Parameters

<i>nMasterHandle</i>	master board handle
<i>mode</i>	buffer overwrite mode
<i>x1</i>	starting x value
<i>y1</i>	starting y value
<i>x2</i>	ending x value
<i>y2</i>	ending y value
<i>val1</i>	starting brightness value
<i>val2</i>	ending brightness value

2.1.3.40 DioWriteOLEDDLineImm()

```
MAGRESWCH_API void DioWriteOLEDDLineImm (
    dio_handle_h nMasterHandle,
    uint8_t x1,
    uint8_t y1,
    uint8_t x2,
    uint8_t y2,
    uint8_t val1,
    uint8_t val2)
```

Immediately writes a line to the screen that begins at (x1, y1), with value val1, to (x2, y2) with value val2.

val1 and val2 are linearly interpolated.

Undefined behavior if coordinates are outside bounds of display

Parameters

<i>nMasterHandle</i>	master board handle
<i>x1</i>	starting x value
<i>y1</i>	starting y value
<i>x2</i>	ending x value
<i>y2</i>	ending y value
<i>val1</i>	starting brightness value
<i>val2</i>	ending brightness value

2.1.3.41 DioWriteOLEDPixelBuff()

```
MAGRESWCH_API void DioWriteOLEDPixelBuff (
    dio_handle_h nMasterHandle,
    dio_oled_write_mode_e mode,
    uint8_t x,
    uint8_t y,
    uint8_t val)
```

Writes a pixel to the screen at the designated coordinate in the local screen buffer.

Out of bounds values wrap around.

Parameters

<i>nMasterHandle</i>	master board handle
<i>mode</i>	buffer overwrite mode
<i>x</i>	pixel x value, 0-256
<i>y</i>	pixel row
<i>val</i>	pixel pair value, the upper nibble being the leftmost

2.1.3.42 DioWriteOLEDPixelImm()

```
MAGRESWCH_API void DioWriteOLEDPixelImm (  
    dio_handle_h nMasterHandle,  
    uint8_t row,  
    uint8_t column,  
    uint8_t val)
```

Writes a pixel to the screen at the designated coordinate immediately.

Out of bounds values wrap around.

Parameters

<i>nMasterHandle</i>	master board handle
<i>x</i>	pixel x value, 0-256
<i>y</i>	pixel row
<i>val</i>	pixel pair value, the upper nibble being the leftmost

2.1.3.43 DioWriteOLEDPixelPairBuff()

```
MAGRESWCH_API void DioWriteOLEDPixelPairBuff (  
    dio_handle_h nMasterHandle,  
    dio_oled_write_mode_e mode,  
    uint8_t column,  
    uint8_t y,  
    uint8_t val)
```

Writes a value to a pair of pixels in the local screen buffer.

This is not written to the screen until the buffer is sent to the device. The x position of the pixel pair is a column, and goes from 0-127, since every column is two pixels wide. Pixel rows are indexed as follows: zero being the top of the screen, 63 being the lowest part of the screen.

Parameters

<i>nMasterHandle</i>	master board handle
<i>mode</i>	buffer overwrite mode
<i>column</i>	pixel column
<i>y</i>	pixel row
<i>val</i>	pixel pair value, the upper nibble being the leftmost pixel

2.1.3.44 DioWriteOLEDPixelPairImm()

```
MAGRESWCH_API void DioWriteOLEDPixelPairImm (
    dio_handle_h nMasterHandle,
    uint8_t column,
    uint8_t y,
    uint8_t val)
```

Writes a value to a pair of pixels immediately.

The x position of the pixel pair is a column, and goes from 0-127, since every column is two pixels wide. Pixel rows are indexed as follows: zero being the top of the screen, 63 being the lowest part of the screen

Parameters

<i>nMasterHandle</i>	master board handle
<i>column</i>	pixel column
<i>y</i>	pixel row
<i>val</i>	pixel pair value, the upper nibble being the leftmost pixel

2.1.3.45 DioWriteOLEDRowArrBuff()

```
MAGRESWCH_API void DioWriteOLEDRowArrBuff (
    dio_handle_h nMasterHandle,
    dio_oled_write_mode_e mode,
    uint8_t y,
    uint8_t * arr)
```

Writes an array of length 128 to a row in the local screen buffer.

The array is written to the screen at a y offset.

Parameters

<i>nMasterHandle</i>	master board handle
<i>y</i>	y offset
<i>arr</i>	pointer to the array

2.1.3.46 DioWriteOLEDRowArrImm()

```
MAGRESWCH_API void DioWriteOLEDRowArrImm (
    dio_handle_h nMasterHandle,
    uint8_t y,
    uint8_t * arr)
```

Writes an array of length 128 to a row on screen immediately.

The array is written to the screen at a y offset.

Parameters

<i>nMasterHandle</i>	master board handle
<i>y</i>	y offset
<i>arr</i>	pointer to the array

2.1.3.47 DioWriteOLEDRowValImm()

```
MAGRESWCH_API void DioWriteOLEDRowValImm (  
    dio_handle_h nMasterHandle,  
    uint8_t y,  
    uint8_t val)
```

Writes singular value to a row on screen immediately.

The value is written to the screen at a y offset.

Parameters

<i>nMasterHandle</i>	master board handle
<i>y</i>	y offset
<i>val</i>	value to write

2.1.3.48 DioWriteOLEDScreenArrBuff()

```
MAGRESWCH_API void DioWriteOLEDScreenArrBuff (  
    dio_handle_h nMasterHandle,  
    dio_oled_write_mode_e mode,  
    uint8_t * arr)
```

Writes 128x64 bytes to the local screen buffer, given a single dimensional array of that size.

Each byte represents two pixels, the upper nibble representing the leftmost pixel of the byte. The array MUST be that size, otherwise program will segfault.

Parameters

<i>nMasterHandle</i>	master board handle
<i>mode</i>	buffer overwrite mode
<i>arr</i>	array to write

2.1.3.49 DioWriteOLEDScreenArrImm()

```
MAGRESWCH_API void DioWriteOLEDScreenArrImm (  
    dio_handle_h nMasterHandle,  
    uint8_t * arr)
```

Writes 128x64 bytes to the screen at once, given a single dimensional array of that size.

Each byte represents two pixels, the upper nibble representing the leftmost pixel of the byte. The array MUST be that size, otherwise program will segfault.

Parameters

<i>nMasterHandle</i>	master board handle
<i>arr</i>	array to write

2.1.3.50 DioWriteOLEDDStrBuff()

```
MAGRESWCH_API void DioWriteOLEDDStrBuff (
    dio_handle_h nMasterHandle,
    dio_oled_write_mode_e mode,
    uint8_t x,
    uint8_t y,
    char * str)
```

Writes a string to the local screen buffer.

Parameters

<i>nMasterHandle</i>	master board handle
<i>x</i>	x offset
<i>y</i>	y offset
<i>str</i>	string to write

2.1.3.51 DioWriteOLEDDStrImm()

```
MAGRESWCH_API void DioWriteOLEDDStrImm (
    dio_handle_h nMasterHandle,
    uint8_t x,
    uint8_t y,
    char * str)
```

Writes a string immediately to the screen.

Parameters

<i>nMasterHandle</i>	master board handle
<i>x</i>	x offset
<i>y</i>	y offset
<i>str</i>	string to write

2.1.3.52 DioWriteOutMemory()

```
MAGRESWCH_API void DioWriteOutMemory (
    dio_handle_h nHandle,
    uint32_t * lpdwMemory,
    uint32_t dwStart,
    uint32_t dwSize,
    int16_t * lpnStatus)
```

Writes a block of data to the output memory of the specified board.

Board must be in a HALT state before calling these functions.

Parameters

<i>nMasterHandle</i>	master board handle
<i>lpdwMemory</i>	array of double words
<i>dwStart</i>	starting address to write
<i>dwSize</i>	number of steps to write.
<i>lpnStatus</i>	returned status, 0 on success, < 0 failure.

2.1.3.53 DioWriteProgramCounter()

```
MAGRESWCH_API void DioWriteProgramCounter (  
    dio_handle_h nHandle,  
    int32_t dwCounter,  
    int16_t * lpnStatus)
```

Writes the specified master board program counter value.

The board must be in pause state for this function to be successful.

Parameters

<i>nHandle</i>	master board handle
<i>dwCounter</i>	new program counter value
<i>lpnStatus</i>	returned status, 0 on success, < 0 failure, > 0 num bytes read

2.1.3.54 DioWriteSPI()

```
MAGRESWCH_API void DioWriteSPI (  
    dio_handle_h nMasterHandle,  
    uint8_t addr,  
    uint8_t * write_buff,  
    size_t len,  
    int16_t * lpnStatus)
```

Writes to the SPI.

Writes the `write_buff` to the SPI peripheral at the address, and ignore any any incoming data

Address 0 is the FPGA, and it is recommended to use the DIO API to communicate with it.

Parameters

<i>nMasterHandle</i>	master board handle
<i>addr</i>	address of the SPI peripheral 0-7.
<i>write_buff</i>	pointer to byte-array of what to write
<i>len</i>	length of both the read and write buffer
<i>lpnStatus</i>	returned status, 0 on success < 0 on failure

2.2 MAGResWCH.h

[Go to the documentation of this file.](#)

```

00001
00005
00006 #pragma once
00007
00008 #ifndef MAGRESWCH_EXPORTS
00009 #define MAGRESWCH_API __declspec(dllexport)
00010 #else
00011 #define MAGRESWCH_API __declspec(dllimport)
00012 #endif
00013
00014
00015 #include <stdint.h>
00016
00017 #ifdef __cplusplus
00018 extern "C" {
00019 #endif
00020
00022 typedef struct dio_handle_s *dio_handle_h;
00023
00025 typedef struct dio_bmp_s *dio_bmp_h;
00026
00028 typedef enum {
00030     ERR_CANNOT_ALLOCATE_LPMASTERHANDLE = -1,
00032     ERR_INVALID_CH367_HANDLE = -2,
00034     ERR_CANNOT_GET_IO_REG_ADDR = -3,
00036     ERR_STEP_SIZE_LESS_EQUAL_ZERO = -4,
00038     ERR_WRITE_LONGER_THAN_AVAIL_MEM = -5,
00040     ERR_WRITE_ZERO_BYTES = -6,
00042     ERR_READ_ZERO_BYTES = -7,
00044     ERR_SPI_READ_ERR = -8,
00046     ERR_SPI_STREAM_ERR = -9,
00048     ERR_SPI_WRITE_ERR = -10,
00050     ERR_SPI_ADDR_OUT_OF_BOUNDS = -11,
00052     ERR_BMP_INVALID_FILE = -12
00053 } dio_errors_e;
00054
00056 typedef enum {
00058     DIO_EXT_TRIG_IGNORE = 0,
00060     DIO_EXT_TRIG_RISING_EDGE,
00062     DIO_EXT_TRIG_FALLING_EDGE,
00064     DIO_EXT_TRIG_HIGH_HOLD,
00066     DIO_EXT_TRIG_LOW_HOLD
00067 } dio_ext_trigger_modes_e;
00068
00069
00070 typedef enum {
00071     DIO_ADRSR_PIN_LOW = 0,
00072     DIO_ADRSR_PIN_HIGH
00073 } dio_adrsr_pin_state_e;
00074
00076 typedef enum {
00078     DIO_OLED_MODE_OVERWRITE = 0,
00080     DIO_OLED_MODE_OR,
00082     DIO_OLED_MODE_GREATER_THAN,
00084     DIO_OLED_MODE_XOR,
00086     DIO_OLED_MODE_NONZERO
00087 } dio_oled_write_mode_e;
00088
00089 /*
00090 typedef enum {
00091     DIO_OLED_BUFF_SEND_MODE_FULL = 0, //sends the full buffer
00092     DIO_OLED_BUFF_SEND_MODE_DIRTY, //sends the 'dirty area' of the buffer
00093     DIO_OLED_BUFF_SEND_MODE_QUAD_1 //divides into quads and sends dirty quad
00094 } dio_oled_buff_send_mode_e;
00095 */
00096
00110 MAGRESWCH_API void DioInitialize(short nMasterNum, dio_handle_h *lpnMasterHandle, int16_t* lpnStatus);
00111
00120 MAGRESWCH_API void DioArm(dio_handle_h nMasterHandle, int16_t* lpnStatus);
00121
00131 MAGRESWCH_API void DioHalt(dio_handle_h nMasterHandle, int16_t* lpnStatus);
00132
00142 MAGRESWCH_API void DioPause (dio_handle_h nMasterHandle, int16_t* lpnStatus);
00143
00144
00151 MAGRESWCH_API void DioTrig(dio_handle_h nMasterHandle,
00152     int16_t* lpnStatus);
00153
00168 MAGRESWCH_API void DioReadCtrlMemory (dio_handle_h nHandle,
00169     uint32_t* lnpdwVector,
00170     uint32_t dwStart,
00171     uint32_t dwSize,

```



```

00172     int16_t* lpnStatus);
00173
00174
00189 MAGRESWCH_API void DioReadOutMemory (dio_handle_h nHandle,
00190     uint32_t* lnpdwVector,
00191     uint32_t dwStart,
00192     uint32_t dwSize,
00193     int16_t* lpnStatus);
00194
00195
00208 MAGRESWCH_API void DioWriteCtrlMemory(dio_handle_h nHandle,
00209     uint32_t* lpdwMemory,
00210     uint32_t dwStart,
00211     uint32_t dwSize,
00212     int16_t* lpnStatus);
00213
00226 MAGRESWCH_API void DioWriteOutMemory(dio_handle_h nHandle,
00227     uint32_t* lpdwMemory,
00228     uint32_t dwStart,
00229     uint32_t dwSize,
00230     int16_t* lpnStatus);
00231
00242 MAGRESWCH_API void DioReadProgramCounter(dio_handle_h nHandle,
00243     int32_t* lpdwCounter,
00244     int16_t* lpnStatus);
00245
00256 MAGRESWCH_API void DioWriteProgramCounter(dio_handle_h nHandle,
00257     int32_t dwCounter,
00258     int16_t* lpnStatus);
00259
00268 MAGRESWCH_API void DioReadCurrentIO(dio_handle_h nHandle,
00269     int32_t* lpdwCounter,
00270     int16_t* lpnStatus);
00271
00282 MAGRESWCH_API void DioWriteCurrentIO(dio_handle_h nHandle,
00283     int32_t dwCounter,
00284     int16_t* lpnStatus);
00285
00286
00287
00301 MAGRESWCH_API void DioReadStatusRegister(dio_handle_h nHandle,
00302     int16_t* lpwData,
00303     int16_t* lpnStatus);
00304
00305
00306 /*
00307  * @brief Reads/writes x register from/to the specified board
00308  *
00309  * called to simulate external event lines value.
00310  *
00311  * @param nHandle      master board handle
00312  * @param lpwData      returned register value
00313  * @param wData        value to be written
00314  * @param lpnStatus    returned status, 0 on success, < 0 failure.
00315  */
00316 /*
00317 MAGRESWCH_API void DioReadXRegister(dio_handle_h nHandle,
00318     int32_t* lpwData,
00319     int16_t* lpnStatus);
00320
00321 MAGRESWCH_API void DioWriteXRegister(dio_handle_h nHandle,
00322     int32_t* wData,
00323     int16_t* lpnStatus);
00324 */
00325
00326
00333 MAGRESWCH_API void DioReset(dio_handle_h nHandle,
00334     int16_t* lpnStatus);
00335
00336 /*
00337  * @brief Sets up the specified master board external event lines trigger mode
00338  *
00339  * DioTrig function or the external trigger control line can trigger the board
00340  * regardless of the trigger mode set by this function. Board must be in a PAUSE
00341  * state to enable triggering.
00342  *
00343  * @param nHandle      master board handle
00344  * @param nMode        specifies one of several trigger modes
00345  * @param lpnStatus    returned status, 0 on success, < 0 failure.
00346  */
00347 /*
00348 MAGRESWCH_API void DioSetupTriggerMode(dio_handle_h nHandle,
00349     int16_t nMode,
00350     int16_t* lpnStatus);
00351 */
00352
00353

```

```

00354  /*
00355  * @brief Sets up the specified master board external event lines source.
00356  *
00357  *
00358  * @param nMasterHandle master board handle
00359  * @param nSource      0 source is external event lines/1 X register
00360  * @param lpnStatus    returned status, 0 on success, < 0 failure.
00361  */
00362  /*
00363  MAGRESWCH_API void DioSetupTriggerXEventSource(dio_handle_h nMasterHandle,
00364      int16_t nSource,
00365      int16_t* lpnStatus);
00366  */
00367
00380  MAGRESWCH_API void DioStep(dio_handle_h nMasterHandle,
00381      int32_t dwSteps,
00382      int16_t* lpnStatus);
00383
00392  MAGRESWCH_API void DioReadMaxDataLines(dio_handle_h nHandle,
00393      int32_t* lpdwCounter,
00394      int16_t* lpnStatus);
00395
00396
00403  MAGRESWCH_API void DioClearCtrlMemory(dio_handle_h nMasterHandle, int16_t* lpnStatus);
00404
00411  MAGRESWCH_API void DioClearOutMemory(dio_handle_h nMasterHandle, int16_t* lpnStatus);
00412
00413
00431  MAGRESWCH_API void DioStreamSPI(dio_handle_h nMasterHandle,
00432      uint8_t addr,
00433      uint8_t *write_buff,
00434      uint8_t *read_buff,
00435      size_t len,
00436      int16_t* lpnStatus);
00437
00453  MAGRESWCH_API void DioWriteSPI(dio_handle_h nMasterHandle,
00454      uint8_t addr,
00455      uint8_t *write_buff,
00456      size_t len,
00457      int16_t* lpnStatus);
00458
00475  MAGRESWCH_API void DioReadSPI(dio_handle_h nMasterHandle,
00476      uint8_t addr,
00477      uint8_t *read_buff,
00478      size_t len,
00479      int16_t* lpnStatus);
00480
00487  MAGRESWCH_API void DioResetDDS(dio_handle_h nMasterHandle, int16_t* lpnStatus);
00488
00506  MAGRESWCH_API void DioSetupTriggerMode(dio_handle_h nMasterHandle,
00507      uint8_t nPauseMode,
00508      uint8_t nRunMode,
00509      int16_t* lpnStatus);
00510
00520  MAGRESWCH_API void DioClose(dio_handle_h nMasterHandle, int16_t* lpnStatus);
00521
00522  MAGRESWCH_API void DioSetADRSR_Low(dio_handle_h nMasterHandle, uint8_t io_byte);
00523  MAGRESWCH_API void DioSetADRSR_High(dio_handle_h nMasterHandle, uint8_t io_byte);
00524
00533  MAGRESWCH_API void DioSetupOLED(dio_handle_h nMasterHandle, int16_t* lpnStatus);
00534
00535
00549  MAGRESWCH_API void DioWriteOLEDPixelPairImm(dio_handle_h nMasterHandle, uint8_t column, uint8_t y,
00550      uint8_t val);
00550
00566  MAGRESWCH_API void DioWriteOLEDPixelPairBuff(dio_handle_h nMasterHandle, dio_oled_write_mode_e mode,
00567      uint8_t column, uint8_t y, uint8_t val);
00568
00579  MAGRESWCH_API void DioWriteOLEDPixelImm(dio_handle_h nMasterHandle, uint8_t row, uint8_t column,
00580      uint8_t val);
00580
00593  MAGRESWCH_API void DioWriteOLEDPixelBuff(dio_handle_h nMasterHandle, dio_oled_write_mode_e mode,
00594      uint8_t x, uint8_t y, uint8_t val);
00595
00612  MAGRESWCH_API void DioWriteOLEDLineImm(dio_handle_h nMasterHandle,
00613      uint8_t x1,
00614      uint8_t y1,
00615      uint8_t x2,
00616      uint8_t y2,
00617      uint8_t val1,
00618      uint8_t val2);
00619
00620
00638  MAGRESWCH_API void DioWriteOLEDLineBuff(dio_handle_h nMasterHandle,
00639      dio_oled_write_mode_e mode,
00640      uint8_t x1,
00641      uint8_t y1,

```

```

00642     uint8_t x2,
00643     uint8_t y2,
00644     uint8_t val1,
00645     uint8_t val2);
00646
00662 MAGRESWCH_API void DioWriteOLEDArrImm(dio_handle_h nMasterHandle, uint8_t x, uint8_t y, uint8_t *arr,
size_t len);
00663
00664
00681 MAGRESWCH_API void DioWriteOLEDArrBuff(dio_handle_h nMasterHandle, dio_oled_write_mode_e mode,
00682     uint8_t x, uint8_t y, uint8_t *arr, size_t len);
00683
00695 MAGRESWCH_API void DioWriteOLEDScreenArrImm(dio_handle_h nMasterHandle, uint8_t *arr);
00696
00709 MAGRESWCH_API void DioWriteOLEDScreenArrBuff(dio_handle_h nMasterHandle, dio_oled_write_mode_e
00710     mode, uint8_t *arr);
00711
00712
00722 MAGRESWCH_API void DioWriteOLEDRowArrImm(dio_handle_h nMasterHandle, uint8_t y, uint8_t *arr);
00723
00724
00734 MAGRESWCH_API void DioWriteOLEDRowArrBuff(dio_handle_h nMasterHandle, dio_oled_write_mode_e mode,
00735     uint8_t y, uint8_t *arr);
00736
00746 MAGRESWCH_API void DioWriteOLEDRowValImm(dio_handle_h nMasterHandle, uint8_t y, uint8_t val);
00747
00767 MAGRESWCH_API void DioWriteOLEDAreaArrImm(dio_handle_h nMasterHandle, uint8_t x, uint8_t y,
00768     uint8_t col_width, uint8_t height, uint8_t *arr);
00769
00789 MAGRESWCH_API void DioWriteOLEDAreaArrBuff(dio_handle_h nMasterHandle, dio_oled_write_mode_e
00790     mode, uint8_t x, uint8_t y,
00791     uint8_t col_width, uint8_t height,
00792     uint8_t *arr);
00793
00803 MAGRESWCH_API void DioWriteOLED2DUnpackedArrImm(dio_handle_h nMasterHandle, uint8_t **arr);
00804
00810 MAGRESWCH_API void DioSendOLEDBuff(dio_handle_h nMasterHandle);
00811
00817 MAGRESWCH_API void DioClearOLEDBuff(dio_handle_h nMasterHandle);
00818
00824 MAGRESWCH_API void DIOClearOLEDImm(dio_handle_h nMasterHandle);
00825
00834 MAGRESWCH_API dio_bmp_h DioOpenBMP(dio_handle_h nMasterHandle, char *path, int16_t *lpnStatus);
00835
00836
00843 MAGRESWCH_API void DioCloseBMP(dio_handle_h nMasterHandle,
00844     dio_bmp_h *bmp_handle);
00845
00846
00858 MAGRESWCH_API void DioWriteOLEDAreaBMPImm(dio_handle_h nMasterHandle, uint8_t x, uint8_t y,
00859     dio_bmp_h bmp_handle);
00860
00873 MAGRESWCH_API void DioWriteOLEDAreaBMPBuff(dio_handle_h nMasterHandle,
00874     dio_oled_write_mode_e mode, uint8_t x, uint8_t y, dio_bmp_h bmp_handle);
00875
00884 MAGRESWCH_API void DioWriteOLEDCharImm(dio_handle_h nMasterHandle, uint8_t x, uint8_t y, char letter);
00885
00886
00895 MAGRESWCH_API void DioWriteOLEDCharBuff(dio_handle_h nMasterHandle,
00896     dio_oled_write_mode_e mode,
00897     uint8_t x, uint8_t y,
00898     char letter);
00899
00900
00909 MAGRESWCH_API void DioWriteOLEDStrImm(dio_handle_h nMasterHandle,
00910     uint8_t x, uint8_t y,
00911     char* str);
00912
00921 MAGRESWCH_API void DioWriteOLEDStrBuff(dio_handle_h nMasterHandle,
00922     dio_oled_write_mode_e mode,
00923     uint8_t x, uint8_t y,
00924     char* str);
00925
00933 MAGRESWCH_API const char* DioGetErrorStr(dio_handle_h nMasterHandle, int16_t lpnStatus);
00934
00935 #ifdef __cplusplus
00936 }
00937 #endif

```


Index

dio_errors_e
MAGResWCH.h, [7](#)

DIO_EXT_TRIG_FALLING_EDGE
MAGResWCH.h, [7](#)

DIO_EXT_TRIG_HIGH_HOLD
MAGResWCH.h, [7](#)

DIO_EXT_TRIG_IGNORE
MAGResWCH.h, [7](#)

DIO_EXT_TRIG_LOW_HOLD
MAGResWCH.h, [7](#)

DIO_EXT_TRIG_RISING_EDGE
MAGResWCH.h, [7](#)

dio_ext_trigger_modes_e
MAGResWCH.h, [7](#)

DIO_OLED_MODE_GREATER_THAN
MAGResWCH.h, [8](#)

DIO_OLED_MODE_NONZERO
MAGResWCH.h, [8](#)

DIO_OLED_MODE_OR
MAGResWCH.h, [8](#)

DIO_OLED_MODE_OVERWRITE
MAGResWCH.h, [8](#)

DIO_OLED_MODE_XOR
MAGResWCH.h, [8](#)

dio_oled_write_mode_e
MAGResWCH.h, [7](#)

DioArm
MAGResWCH.h, [8](#)

DioClearCtrlMemory
MAGResWCH.h, [8](#)

DioClearOLEDBuff
MAGResWCH.h, [8](#)

DIOClearOLEDImm
MAGResWCH.h, [9](#)

DioClearOutMemory
MAGResWCH.h, [9](#)

DioClose
MAGResWCH.h, [9](#)

DioCloseBMP
MAGResWCH.h, [9](#)

DioGetErrorStr
MAGResWCH.h, [10](#)

DioHalt
MAGResWCH.h, [10](#)

DioInitialize
MAGResWCH.h, [10](#)

DioOpenBMP
MAGResWCH.h, [11](#)

DioPause
MAGResWCH.h, [11](#)

DioReadCtrlMemory
MAGResWCH.h, [11](#)

DioReadCurrentIO
MAGResWCH.h, [12](#)

DioReadMaxDataLines
MAGResWCH.h, [12](#)

DioReadOutMemory
MAGResWCH.h, [12](#)

DioReadProgramCounter
MAGResWCH.h, [13](#)

DioReadSPI
MAGResWCH.h, [13](#)

DioReadStatusRegister
MAGResWCH.h, [13](#)

DioReset
MAGResWCH.h, [14](#)

DioResetDDS
MAGResWCH.h, [14](#)

DioSendOLEDBuff
MAGResWCH.h, [14](#)

DioSetupOLED
MAGResWCH.h, [15](#)

DioSetupTriggerMode
MAGResWCH.h, [15](#)

DioStep
MAGResWCH.h, [15](#)

DioStreamSPI
MAGResWCH.h, [16](#)

DioTrig
MAGResWCH.h, [16](#)

DioWriteCtrlMemory
MAGResWCH.h, [17](#)

DioWriteCurrentIO
MAGResWCH.h, [17](#)

DioWriteOLED2DUnpackedArrImm
MAGResWCH.h, [17](#)

DioWriteOLEDAreaArrBuff
MAGResWCH.h, [18](#)

DioWriteOLEDAreaArrImm
MAGResWCH.h, [18](#)

DioWriteOLEDAreaBMPBuff
MAGResWCH.h, [19](#)

DioWriteOLEDAreaBMPImm
MAGResWCH.h, [19](#)

DioWriteOLEDArrBuff
MAGResWCH.h, [19](#)

DioWriteOLEDArrImm
MAGResWCH.h, [20](#)

- DioWriteOLEDCharBuff
 - MAGResWCH.h, [20](#)
- DioWriteOLEDCharImm
 - MAGResWCH.h, [21](#)
- DioWriteOLEDLineBuff
 - MAGResWCH.h, [21](#)
- DioWriteOLEDLineImm
 - MAGResWCH.h, [22](#)
- DioWriteOLEDPixelBuff
 - MAGResWCH.h, [22](#)
- DioWriteOLEDPixelImm
 - MAGResWCH.h, [23](#)
- DioWriteOLEDPixelPairBuff
 - MAGResWCH.h, [23](#)
- DioWriteOLEDPixelPairImm
 - MAGResWCH.h, [23](#)
- DioWriteOLEDRowArrBuff
 - MAGResWCH.h, [24](#)
- DioWriteOLEDRowArrImm
 - MAGResWCH.h, [24](#)
- DioWriteOLEDRowValImm
 - MAGResWCH.h, [25](#)
- DioWriteOLEDScreenArrBuff
 - MAGResWCH.h, [25](#)
- DioWriteOLEDScreenArrImm
 - MAGResWCH.h, [25](#)
- DioWriteOLEDStrBuff
 - MAGResWCH.h, [26](#)
- DioWriteOLEDStrImm
 - MAGResWCH.h, [26](#)
- DioWriteOutMemory
 - MAGResWCH.h, [26](#)
- DioWriteProgramCounter
 - MAGResWCH.h, [27](#)
- DioWriteSPI
 - MAGResWCH.h, [27](#)
- ERR_BMP_INVALID_FILE
 - MAGResWCH.h, [7](#)
- ERR_CANNOT_ALLOCATE_LPNMASTERHANDLE
 - MAGResWCH.h, [7](#)
- ERR_CANNOT_GET_IO_REG_ADDR
 - MAGResWCH.h, [7](#)
- ERR_INVALID_CH367_HANDLE
 - MAGResWCH.h, [7](#)
- ERR_READ_ZERO_BYTES
 - MAGResWCH.h, [7](#)
- ERR_SPI_ADDR_OUT_OF_BOUNDS
 - MAGResWCH.h, [7](#)
- ERR_SPI_READ_ERR
 - MAGResWCH.h, [7](#)
- ERR_SPI_STREAM_ERR
 - MAGResWCH.h, [7](#)
- ERR_SPI_WRITE_ERR
 - MAGResWCH.h, [7](#)
- ERR_STEP_SIZE_LESS_EQUAL_ZERO
 - MAGResWCH.h, [7](#)
- ERR_WRITE_LONGER_THAN_AVAIL_MEM
 - MAGResWCH.h, [7](#)
- ERR_WRITE_ZERO_BYTES
 - MAGResWCH.h, [7](#)
- MAGResWCH.h, [3](#)
 - dio_errors_e, [7](#)
 - DIO_EXT_TRIG_FALLING_EDGE, [7](#)
 - DIO_EXT_TRIG_HIGH_HOLD, [7](#)
 - DIO_EXT_TRIG_IGNORE, [7](#)
 - DIO_EXT_TRIG_LOW_HOLD, [7](#)
 - DIO_EXT_TRIG_RISING_EDGE, [7](#)
 - dio_ext_trigger_modes_e, [7](#)
 - DIO_OLED_MODE_GREATER_THAN, [8](#)
 - DIO_OLED_MODE_NONZERO, [8](#)
 - DIO_OLED_MODE_OR, [8](#)
 - DIO_OLED_MODE_OVERWRITE, [8](#)
 - DIO_OLED_MODE_XOR, [8](#)
 - dio_oled_write_mode_e, [7](#)
 - DioArm, [8](#)
 - DioClearCtrlMemory, [8](#)
 - DioClearOLEDBuff, [8](#)
 - DIOClearOLEDImm, [9](#)
 - DioClearOutMemory, [9](#)
 - DioClose, [9](#)
 - DioCloseBMP, [9](#)
 - DioGetErrorStr, [10](#)
 - DioHalt, [10](#)
 - DioInitialize, [10](#)
 - DioOpenBMP, [11](#)
 - DioPause, [11](#)
 - DioReadCtrlMemory, [11](#)
 - DioReadCurrentIO, [12](#)
 - DioReadMaxDataLines, [12](#)
 - DioReadOutMemory, [12](#)
 - DioReadProgramCounter, [13](#)
 - DioReadSPI, [13](#)
 - DioReadStatusRegister, [13](#)
 - DioReset, [14](#)
 - DioResetDDS, [14](#)
 - DioSendOLEDBuff, [14](#)
 - DioSetupOLED, [15](#)
 - DioSetupTriggerMode, [15](#)
 - DioStep, [15](#)
 - DioStreamSPI, [16](#)
 - DioTrig, [16](#)
 - DioWriteCtrlMemory, [17](#)
 - DioWriteCurrentIO, [17](#)
 - DioWriteOLED2DUnpackedArrImm, [17](#)
 - DioWriteOLEDAreaArrBuff, [18](#)
 - DioWriteOLEDAreaArrImm, [18](#)
 - DioWriteOLEDAreaBMPBuff, [19](#)
 - DioWriteOLEDAreaBMPIImm, [19](#)
 - DioWriteOLEDArrBuff, [19](#)
 - DioWriteOLEDArrImm, [20](#)
 - DioWriteOLEDCharBuff, [20](#)
 - DioWriteOLEDCharImm, [21](#)
 - DioWriteOLEDLineBuff, [21](#)
 - DioWriteOLEDLineImm, [22](#)
 - DioWriteOLEDPixelBuff, [22](#)
 - DioWriteOLEDPixelImm, [23](#)

DioWriteOLEDPixelPairBuff, [23](#)
DioWriteOLEDPixelPairImm, [23](#)
DioWriteOLEDRowArrBuff, [24](#)
DioWriteOLEDRowArrImm, [24](#)
DioWriteOLEDRowValImm, [25](#)
DioWriteOLEDScreenArrBuff, [25](#)
DioWriteOLEDScreenArrImm, [25](#)
DioWriteOLEDStrBuff, [26](#)
DioWriteOLEDStrImm, [26](#)
DioWriteOutMemory, [26](#)
DioWriteProgramCounter, [27](#)
DioWriteSPI, [27](#)
ERR_BMP_INVALID_FILE, [7](#)
ERR_CANNOT_ALLOCATE_LPNMASTERHANDLE,
[7](#)
ERR_CANNOT_GET_IO_REG_ADDR, [7](#)
ERR_INVALID_CH367_HANDLE, [7](#)
ERR_READ_ZERO_BYTES, [7](#)
ERR_SPI_ADDR_OUT_OF_BOUNDS, [7](#)
ERR_SPI_READ_ERR, [7](#)
ERR_SPI_STREAM_ERR, [7](#)
ERR_SPI_WRITE_ERR, [7](#)
ERR_STEP_SIZE_LESS_EQUAL_ZERO, [7](#)
ERR_WRITE_LONGER_THAN_AVAIL_MEM, [7](#)
ERR_WRITE_ZERO_BYTES, [7](#)