Emerald-MM-8EL Driver Software User Manual

Emerald-MM-8EL 4- or 8-Port Serial Module

High Performance PCI/104-Express 4- or 8-Port RS-232/422/485 Serial I/O Module with Opto-isolation

For Universal Driver Version 7.0.0 and later

Revision A.0 May 2015

Revision	Date	Comment
A.0	5/13/2015	Initial release

FOR TECHNICAL SUPPORT PLEASE CONTACT:

support@diamondsystems.com

Diamond Systems Corporation
555 Ellis Street
Mountain View, CA 94043 USA
Tel 1-650-810-2500
Fax 1-650-810-2525
www.diamondsystems.com

CONTENTS

1. INTRODUCTION

This user manual contains all essential information about the Emerald-MM-8EL Demo applications, programming guidelines and usage instructions. This manual also includes the Emerald-MM-8EL driver API descriptions with usage examples.

2. HARDWARE OVERVIEW

2.1 Description

The Emerald-MM-8EL-XT is a family of high performance PCIe/104 "OneBank" serial I/O modules offering 4 or 8 multiprotocol serial ports with software-controlled configuration and optional opto-isolation.

The serial ports are based on a high speed PCIe octal UART with 256-byte TX/RX FIFOs and auto RS-485 transmit control. Each serial port can be independently configured for RS-232, RS-422, or RS-485 protocols, along with programmable 120-ohm line termination. Each port is independently isolated with an isolated power + signal chip, plus additional isolators for control signals. The board features intelligent power management that limits inrush current on power-up and also enables power-down of unused serial ports for power savings.

Opto-isolated models feature independent 2500VRMS isolation circuits for enhanced reliability in vehicle or long cable applications. All ports also feature +/-15KV ESD protection. Each serial port is available on an independent latching connector for increased isolation and ruggedness. With its wide operating temperature range and high resistance to shock and vibration, the EMM-8EL-XT fits a wide variety of rugged and on-vehicle embedded serial I/O application needs.

EMM-8EL-XT also offers 8 digital/analog I/O lines which are programmable from the on-board microcontroller. Each I/O line can be configured for digital input or output. Seven of the I/O lines can be configured for 12-bit A/D input with selectable 0-2.048V or 0-3.3V input ranges.

EMM-8EL-XT contains no configuration jumpers; all configuration and control is done with an onboard microcontroller. All configuration settings are stored in the microcontroller's flash memory and are automatically loaded on power-up.

2.2 Specifications

- 8 or 4 RS-232/422/485 Serial ports with programmable protocol
- Data rates: RS-232 mode up to 1Mbps
RS-422/485 modes up to 10Mbps
- XR17V358/XR17V354 PCIe interface UART with 256-byte FIFOs.
- Models available with or without opto-isolation.
- SP336 multi-protocol transceivers with high speed, ESD, and open-circuit protection.
- RS-422/485 termination programmable.
- +/-15KV ESD protection on all serial ports.
- Independent 2500VRMS isolation port by port with individual I/O connectors for each port.
- Staggered turn-on of isolated power devices for reduced power-on inrush current.
- 8 digital I/O or analog input lines.
- PCIex1 host interface using PCIe/104 “OneBank™” connector.
- Onboard microcontroller to manage all configuration options with flash memory for configuration storage and automatic recall.
- Latching connectors for increase ruggedness.
- Comprehensive software suite enables easy configuration and control.

3. GENERAL PROGRAMMING GUIDELINES

3.1 Initialization and exit function calls

The functions should be called prior to the calling of any other EMM-8EL-XT board specific functions.

- EMM8EL_INIT (), this function initializes the EMM-8EL-XT.

At the termination of the demo applications, the user should call EMM8EL_Free () function to close the file handles.

These calls are important in initializing and freeing resources used by the driver. Here is an example of the framework for an application using the driver:

```
#ifdef WIN32
#include <windows.h>
#include <winioctl.h>
#endif
#include "emm8elxt.h"
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <time.h>
ERRPARAMS errorParams;

int main()
{
      if (EMM8EL_INIT() != 0)
      {
            DSCGetLastError (&errorParams);
            Printf ("EMM8EL_INIT error: %s %s\n", dscGetErrorString
            (errorParams.ErrCode),
            errorParams.errstring);
            return 0;
      }
      EMM8EL_Free();

      return 0;
}
```

3.2 Error handling

All the Emerald-MM-8EL driver functions provide a basic error handling mechanism that stores the last reported error in the driver. If your application is not behaving properly, you can check for the last error by calling the function DSCGetLastError (). This function takes an ERRPARAMS structure pointer as its argument.

Nearly all the available functions in the Emerald-MM-8EL driver API return a BYTE value upon completion. This value represents an error code that will inform the user whether the function call was successful or not. The User should always check if the result returns a DE_NONE value (signifying that no errors were reported), as the code below illustrates:

```
BYTE result;
ERRPARAMS errparams;
if (EMM8EL_INIT() != DE_NONE)
{
      DSCGetLastError (&errorParams);
      Printf ("EMM8EL_INIT error: %s %s\n", dscGetErrorString
      (errorParams.ErrCode),
      errorParams.errstring);
      return 0;
}
```

In this code snippet, the BYTE result of executing a particular driver function (EMM8EL_INIT () in this case) is stored and checked against the expected return value (DE_NONE). At any point of time, if a function does not complete successfully, an error code other than DE_NONE will be generated, and the current API function will be terminated. The function dscGetErrorString () provides a description of the error occurred.

4. EMERALD-MM-8EL DRIVER API DESCRIPTION

4.1 EMM8EL_GetConfig

Function Definition
BYTE EMM8EL_GetConfig (int RegType, int* Config);

Function description
This function returns the board's complete configuration data from the controller's flash or RAM.

Function Parameters

Name	Description
RegType	0 = RAM, 1 = Flash, 2 = RAM and Flash
Config	Pointer to board's complete configuration data

Return Value
0 - Success or Error code

Usage Example
To get boards configuration from RAM,

```
RegType = 0;
EMM8EL_GetConfig (RegType, &Config);
```

4.2 EMM8EL_SetConfig

Function Definition
BYTE EMM8EL_SetConfig (int RegType, int* Config);

Function description
This function stores a complete configuration dataset into the on-board controller's flash and/or RAM.

Function Parameters

Name	Description
RegType	0 = RAM, 1 = Flash, 2 = RAM and Flash
Config	Pointer to board's complete configuration data.

Return Value
0 - Success or Error code

Usage Example
To set all the digital I/O port direction to input mode,

```
int *config;
int RegType;
RegType = 0;
config = (int *)malloc(sizeof(int));
config [0] = 0x1d;
EMM8EL_SetConfig (RegType, &config);
```

4.3 EMM8EL_SerialPortConfig

Function Definition
BYTE EMM8EL_SerialPortConfig (int RegType, int port, int prot_type, int line_term);

Function description
This function configures a serial port for protocol and termination.

Function Parameters

Name	Description
RegType	0 = RAM, 1 = Flash, 2 = RAM and Flash
Port	Serial port number
Prot_type	Protocol type (0 – RS-232, 1 – RS-422, 2 – RS-485)
Line_term	Line termination values (0 – disabled, 1 – enabled).

Return Value
0 - Success or Error code

Usage Example
To configure serial port 1 for RS-232 protocol and disabled termination,

```
RegType=2;
Port =1;
Prot_type =0;
Line_term=0;
EMM8EL_SerialPortConfig (RegType, Port, prot_type, Line_term);
```

4.4 EMM8EL_SerialPortEnable

Function Definition
BYTE EMM8EL_SerialPortEnable (int RegType, int port, int power_en);

Function description
This function configures a serial port for power up or power down.

Function Parameters

Name	Description
RegType	0 = RAM, 1 = Flash, 2 = RAM and Flash
Port	Serial port number
Power_en	1 – Enable, 0 – disable.

Return Value
0 - Success or Error code

Usage Example
To configures a serial port for power up,

```
RegType=2;
port =1;
power_en=1;
EMM8EL_SerialPortEnable (RegType, port, power_en);
```

4.5 EMM8EL_SerialPowerConfig

Function Definition
BYTE EMM8EL_SerialPowerConfig (int RegType, int init_delay, int inter_delay);

Function description
This function configures the power-on timing of the serial port isolators.

Function Parameters

Name	Description
RegType	RegType: 0 = RAM, 1 = Flash, 2 = RAM and Flash
Init_delay	ADUM5402 Initial power-on delay in **2ms** increments
Inter_delay	ADUM5402 inter-device power-on interval in **1ms** increments

Return Value
0 - Success or Error code

Usage Example
To configures the power-on timing of init_delay=2ms and inter_delay =1ms of register type RAM,

```
RegType =0;
init_delay = 2;
inter_delay = 1;
EMM8EL_SerialPowerConfig (RegType, init_delay, inter_delay);
```

4.6 EMM8EL_IOConfig

Function Definition
BYTE EMM8EL_IOConfig (int io_config, int io_dir);

Function description
This function configures the I/O for A/D or DIO; for DIO the direction is also defined.

Function Parameters

Name	Description
Io_config	I/O configuration value (0 – DIO, 1 - Analog)
Io_dir	0 – input, 1 – output.

Return Value
0 - Success or Error code

Usage Example
To configure the I/O for DIO with input direction,

```
io_config = 0;
io_dir=0;
EMM8EL_IOConfig (io_config, io_dir);
```

4.7 EMM8EL_ADConfig

BYTE EMM8EL_ADConfig (int ref_volt, int channel);

Function description
This function configures the A/D reference voltage and differential channel.

Function Parameters

Name	Description
Ref_volt	A/D Reference voltage 0 = +3.3V, 1 = +2.048V
Channel	Differential mode channel 1,2,3,4,6

Return Value
0 - Success or Error code

Usage Example
To configure Reference voltage of +3.3V and channel 2,

```
Ref_volt=0;
Channel=2;
EMM8EL_ADConfig (ref_volt, Channel);
```

4.8 EMM8EL_ADConvert

Function Definition
BYTE EMM8EL_ADConvert (int channel, signed int ADSample);

Function description
This function performs an A/D conversion.

Function Parameters

Name	Description
Channel	Desired A/D channel
ADSample	Output the AD sample value (-4096 to +4095)

Return Value
0 - Success or Error code

Usage Example
To perform A/D conversion of sample value 4000 on 1st A/D channel,

```
Channel =1;
ADSample=4000;
EMM8EL_ADConvert (channel, ADSample);
```

4.9 EMM8EL_DIOInput

Function Definition
BYTE EMM8EL_DIOInput (byte* data);

Function description
This function reads all DIO bits configured for input arranged in a byte and returns it in the location specified by the pointer to data.

Function Parameters

Name	Description
Data	pointer to receive the data read from the port

Return Value
0 - Success or Error code

Usage Example
To read the Digital I/O data value,

EMM8EL_DIOInput (&data);

4.10 EMM8EL_DIOInputBit

Function Definition
BYTE EMM8EL_DIOInputBit (int bit, byte* data);

Function description
This function reads one DIO bit configured for input and returns it in the location specified by the pointer to data.

Function Parameters

Name	Description
bit	(0-7) bits
Data	pointer to receive the data read from the port

Return Value
Error code or 0.

Usage Example
To read 2nd bit from the Digital I/O data register,

```
bit =2;
EMM8EL_DIOInputBit (bit, &data);
```

4.11 EMM8EL_DIOOutput

Function Definition
BYTE EMM8EL_DIOOutput (byte data);

Function description
This function sets all DIO bits configured for output.

Function Parameters

Name	Description
data	output value

Return Value
0 - Success or Error code

Usage Example
To output the 0x3F data to DIO pins.

```
Data = 0x3F;
EMM8EL_DIOOutput (Data);
```

4.12 EMM8EL_DIOOutputBit

Function Definition
BYTE EMM8EL_DIOOutputBit (int bit, byte data);

Function description
This function writes the bit to the specified digital I/O pin.

Function Parameters

Name	Description
Bit	bit position of the DIO
Data	output value

Return Value
0 - Success or Error code

Usage Example
To set the 2nd pin as output,

bit =2;
`data` =1;
`EMM8EL_DIOOutputBit` (bit, data);

4.13 EMM8EL_LED

Function Definition
BYTE EMM8EL_ LED (int state);

Function description
This function initializes the board.

Function Parameters

Name	Description
state	1 – ON, 0 – OFF

Return Value
0 - Success or Error code

Usage Example
To turn on the LED,

state =1;
`EMM8EL_` LED (`state`);

4.14 EMM8EL_INIT

Function Definition
BYTE EMM8EL_INIT ();

Function description
This function identifies the first COM port number of EXAR device and gets the handle to communicate with EXAR driver.

Return Value
0 - Success or Error code

Usage Example
To identify the first COM port number of EXAR device and gets the handle to communicate with EXAR driver,

```
EMM8EL_INIT ();
```

4.15 EMM8EL_Write

Function Definition
BYTE EMM8EL_Write (int RegType, int address, byte data);

Function description
This function writes a byte to the on-board controller's flash or RAM.

Function Parameters

Name	Description
RegType	0 – RAM, 1 – FLASH, 2 – RAM and FLASH
Address	register address
Data	input data

Return Value
0 - Success or Error code

Usage Example
To write data=0x80 in the address= 0x13,

```
RegType = 0;
address = 0x13;
data = 0x80;
EMM8EL_Write (RegType, address, data);
```

4.16 EMM8EL_Read

Function Definition
BYTE EMM8EL_ Read (int RegType, int address, byte* data);

Function description
This function reads a byte from the on-board controller's flash or RAM

Function Parameters

Name	Description
RegType	0 – RAM, 1 – FLASH, 2 – RAM and FLASH
address	register address
Data	pointer to the output data

Return Value
0 - Success or Error code

Usage Example
To read the byte from the address 0x11,

```
RegType =0;
address =0x11;
EMM8EL_ Read (RegType, address, & data);
```

4.17 EMM8EL_Reset

Function Definition
BYTE EMM8EL_Reset ();

Function description
This function resets the on-board controller and causes a reload of all configuration settings.

Return Value
0 - Success or Error code

Usage Example
To reset the on-board controller,

```
EMM8EL_Reset ();
```

4.18 EMM8EL_Status

Function Definition
BYTE EMM8EL_Status (byte *present, byte* cfg, byte* revision, byte* LEDstatus);

Function description
This function detects the status of the controller / verifies that it is listening, and returns the model Configuration (CFG pins) and software revision number

Function Parameters

Name	Description
Present	1 – board is present, 0 –not present
Cfg	Pointer to receive the configuration register value.
Revision	Pointer to receive the software revision code.
LEDstatus	Pointer to receive the LED status.

Return Value
Error code or 0.

Usage Example
To detect the status of the controller / verifies that it is listening

```
EMM8EL_Status (&present, &cfg, & revision, &LEDstatus);
```

5. EMERALD-MM-8EL DRIVER DEMO APPLICATION DESCRIPTION

The Emerald-MM-8EL driver supports the following application for EMM-8EL-XT board:

- EMM8ELXT

5.1 EMM8ELXT

This application provides various options i.e. Reset, Board Presence detection, ADUM5402 enable configuration, ADUM5402 turn-on delay configuration, LED control, ProtocolSelect_LineTermination, GPIO Configuration, GPIO Input, GPIO Input Bit, GPIO Output, GPIO Output Bit, Read operation, Write operation, A/D Configuration, A/D Sample, Configuration, load Configuration from file, store Configuration to Flash, store Configuration to RAM, read RAM and Flash configuration from board, store RAM configuration to Flash, store RAM configuration to file, store Flash configuration to File

6. EMERALD-MM-8EL DRIVER DEMO APPLICATION USAGE INSTRUCTIONS

6.1 EMM8ELXT

E.g.1: To Write 0x12 in address 0x3A of register type RAM

Enter an option from below
1. Reset
2. Board Presence detection
3. ADUM5402 enable configuration
4. ADUM5402 turn-on delay configuration
5. LED control
6. ProtocolSelect_LineTermination
7. GPIO Configuration
8. GPIO Input
9. GPIO Input Bit
10. GPIO Output
11. GPIO Output Bit
12. Read operation
13. Write operation
14. A/D Configuration
15. A/D Sample
16. Configuration
q. Quit the program
13
Enter Register type <0 = RAM, 1 = RAM and Flash; default: 0 >0
Enter address in Hex <0x00 – 0x3F >:0x3A
Enter Data in Hex <0x00 – 0xFF >: 0x12

E.g.2: To Store RAM configuration to file

Enter an option from below
1. Reset
2. Board Presence detection
3. ADUM5402 enable configuration
4. ADUM5402 turn-on delay configuration
5. LED control
6. ProtocolSelect_LineTermination
7. GPIO Configuration
8. GPIO Input
9. GPIO Input Bit
10. GPIO Output
11. GPIO Output Bit
12. Read operation
13. Write operation
14. A/D Configuration
15. A/D Sample
16. Configuration
q. Quit the program
16
Enter an option for configuration operation
1. Load Configuration from file
2. Store Configuration to Flash
3. Store Configuration to RAM
4. Read RAM and Flash configuration from board
5. Store RAM configuration to Flash
6. Store RAM configuration to file
7. Store Flash configuration to File

8. Main menu
q. Quit
6
Enter filename to store RAM configuration to file: ram.ini

7. COMMON TASK REFERENCE

7.1 Data Acquisition Feature Overview

EMM-8EL-XT offers 8 programmable digital I/O or analog input lines. All 8 I/O lines can be configured for digital I/O and seven can be configured for 12-bit analog input.

Digital I/O signals
EMM-8EL-XT offers 8 digital/analog I/O lines which are programmable from the on-board microcontroller. Each I/O line can be configured for digital input or output.

A/D
Seven of the I/O lines can be configured for 12-bit A/D input with selectable 0-2.048V or 0-3.3V input ranges.

7.2 Data Acquisition Software Task Reference

This section describes the various data acquisition tasks that may be performed with the EMM-8EL-XT board and gives step by step instructions on how to achieve them using the Emerald-MM-8EL Driver functions. Tasks include:

- Program entry / exit sequence
- Digital I/O
- A/D

Program Entry/Exit sequence

1. All driver usage begins with the function EMM8EL_INIT (). This function must be called prior to any other function involving the EMM-8ELXT.
2. At the termination of the program the programmer may use EMM8EL_Free (), but this is not required. This function is normally used in a development environment where the program is being repeatedly modified and rerun.

Digital I/O operations
Configure the port in output or input mode using the following function,

BYTE EMM8EL_IOConfig (io_config,io_direction);

Byte read/write enables 8 bits of digital I/O to be updated at once. Bit operation enables a single bit to be updated.

```
BYTE EMM8EL_DIOInput (byte* data);
BYTE EMM8EL_DIOInputBit (int bit, byte* data);
BYTE EMM8EL_DIOOutput (byte data);
BYTE EMM8EL_DIOOutputBit (int bit, byte data);
```

LED control
The EMM-8EL-XT contains a LED that is user-programmable. This can be used as a visual indication that the board is responding to commands. Turn the LED ON or OFF by using the EMM8EL_LED () function.

7.3 Performing Digital IO Operations

Description

The driver supports four types of direct digital I/O operations: input bit, input byte, output bit, and output byte. Digital I/O is fairly straightforward - to perform digital input, you provide a pointer to the storage variable and indicate the port number and bit number if relevant. To perform digital output, you provide the output value and the output port and bit number, if relevant.

The five Emerald-MM-8EL Driver functions described here are EMM8EL_IOConfig(), EMM8EL_DIOInput() , EMM8EL_DIOInputBit(), EMM8EL_DIOOutput (), EMM8EL_DIOOutputBit().

Step-By-Step Instructions

Call EMM8EL_IOConfig function to configure the port in input/output mode to read/write the port. Call the selected digital I/O function. Pass it an int port value, and either a pointer to or a constant byte digital value. If you are performing bit operations, then you will also need to pass in an int value telling the driver which particular bit (0-7) of the DIO port you wish to operate on.

Example of Usage for Digital I/O Operations

```
…..
int port;
BYTE input_byte;  // the value ranges from 0 to 255
BYTE output_byte;
int digital_value;

/* 1. Configure Port 0 in output mode */
if ( EMM8EL_IOConfig(io_config,io_direction) != 0)
{
      DSCGetLastError (&errorParams);
      printf ("EMM8EL_IOConfig error: %s %s\n",
      dscGetErrorString(errorParams.ErrCode),
      errorParams.errstring);
      return 0;
}
/* 2. input bit - read bit 6 (port  is in input mode) */
bit = 6; //DIO line 0-7 are port 0
if ( EMM8EL_DIOInputBit(bit,&data) != 0 )
{
      DSCGetLastError (&errorParams);
      printf ("EMM8EL_DIOInputBit error: %s %s\n",
      dscGetErrorString(errorParams.ErrCode),
      errorParams.errstring);
      return 0;
}
/* 3. input byte - read all 8 bits of port (port  is in input mode)  */
BYTE value;
if ( EMM8EL_DIOInput(&value) != 0)
{
      DSCGetLastError (&errorParams);
      printf ("EMM8EL_DIOInput error: %s %s\n",
      dscGetErrorString(errorParams.ErrCode),
      errorParams.errstring);
      return 0;
}
/* 4. output bit – set the bit 6 of port (assumes port  is in output mode) */
bit = 6;
digital_val = 1;
if ( (EMMDIO4MDIOOutputBit (bi,port,bit,digital_val) != DE_NONE) )
```

```
{
      dscGetLastError (&errorParams);
      printf ("EMMDIO4MDIOOutputBit error: %s %s\n",
      dscGetErrorString(errorParams.ErrCode), errorParams.errstring );
      return 0;
}
/* 5. Output byte (assumes port is in output mode) */
BYTE value = 0xFF;
if ( EMM8EL_DIOOutput(value) != 0)
{
      DSCGetLastError (&errorParams);
      printf ("EMM8EL_DIOOutput error: %s %s\n",
      dscGetErrorString(errorParams.ErrCode),
      errorParams.errstring);
      return 0;
}
```

7.4 Performing A/D Sample

Description

When you perform an A/D conversion, you are getting a digital reading of an analog voltage signal applied to one of the A/D board's analog input channels. The following Emerald-MM-8EL driver function is used for A/D conversion: EMM8EL_ADConvert ().

Step-By-Step Instructions

Create EMM8EL_ADConfig() configures the Reference voltage and input mode(Single ended or Differential ended), then call EMM8EL_ADConvert () to configure the A/D settings as requested by the user, and It then takes a single A/D sample value of a single channel.

Example of Usage for A/D Sample

```
if ( EMM8EL_ADConfig(ADRef,Diff_channel) != 0)
{
      DSCGetLastError (&errorParams);
      printf ("EMM8EL_ADConfig error: %s %s\n",
      dscGetErrorString(errorParams.ErrCode), errorParams.errstring );
      return 0;
}
if ( EMM8EL_ADConvert(channel,&ADSamples) != 0)
{
      DSCGetLastError (&errorParams);
      Printf ("EMM8EL_ADConvert error: %s %s\n", dscGetErrorString
      (errorParams.ErrCode), errorParams.errstring);
      return 0;
}
```

8. INTERFACE CONNECTOR DETAILS

This section describes the connectors on the Emerald-MM-8EL-XT module

8.1 Serial Ports 1-8 (Port 1-8)

The 4 or 8 serial ports are provided on miniature 10-pin headers with 1 port per header. Ports 1-4 are located on the right side of the board, and ports 5-8, if installed, are located on the left. The pin definition depends on the serial protocol selected. All ports have identical pinouts as shown below.

RS-232 Configuration:

NC	**1**	**2**	NC
TXD	**3**	**4**	RTS
RXD	**5**	**6**	CTS
NC	**7**	**8**	Iso Gnd

RS-422 Configuration:

NC	**1**	**2**	NC
TX-	**3**	**4**	TX+
RX+	**5**	**6**	RX-
NC	**7**	**8**	Iso Gnd’

RS-485 Configuration:

NC	**1**	**2**	NC
Data-	**3**	**4**	Data+
NC	**5**	**6**	NC
NC	**7**	**8**	Iso Gnd

Description: 2x4 2mm pitch through hole right angle locking pin header, standard

Part Number: JST S08B-PUDSS-1

8.2 Data Acquisition (DAQ)

Eight GPIO signals from the microcontroller are made available on a 10 pin header for auxiliary GPIO / A/D functions. All the A/D capable pins are grouped together at the top of the connector near the analog ground pin

AGnd	**1**	**2**	GPIO / A/D0
GPIO1 / A/D1	**3**	**4**	GPIO / A/D2
GPIO3 / A/D3	**5**	**6**	GPIO / A/D4
GPIO5 / A/D5	**7**	**8**	GPIO6 / A/D6
GPIO7	**9**	**10**	D Gnd

Description: 2x5 2mm pitch through-hole connector with 4mm high posts and gold flash plating