

MIO-8288

MIO-8288 User Manual & Reference Guide

DAQ Remote Devices

Data Acquisition and Process Control

© Eagle Technology
24 Burg Street • Cape Town • South Africa
Phone +27 21 423 4943 • Fax +27 21 424 4637
Email eagle@eagle.co.za

Copyright

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or any means, electronic, mechanical, by photographing, recording, or otherwise without prior written permission.

Copyright © Eagle Technology, South Africa
April 2011
Revision 1.0

Information furnished in this manual is believed to be accurate and reliable; however no responsibility is assumed for its use, or any infringements of patents or other rights of third parties, which may result from its use.

Trademarks and Logos in this manual are the property of their respective owners.

Product Warranty

Eagle Technology, South Africa, warrants its products from defect in material and workmanship from confirmed date of purchase for a period of one year if the conditions listed below are met. The product warranty will call the Eagle Technology Data Acquisition Device short as **ETDAQD**.

- The warranty does not apply to an **ETDAQD** that has been previously repaired, altered, extended by any other company or individual outside the premises of Eagle Technology.
- That a qualified person configure and install the **ETDAQD**, and damages caused to a device during installation shall make the warranty void and null.
- The warranty will not apply to conditions where the **ETDAQD** has been operated in a manner exceeding its specifications.

Eagle Technology, South Africa, does not take responsibility or liability of consequential damages, project delays, damaging of equipment or capital loss as a result of its products.

Eagle Technology, South Africa, holds the option and final decision to repair or replace any **ETDAQD**. Proof of purchase must be supplied when requesting a repair.

TABLE OF CONTENTS

| | | |
|---|-----------------------------------|---|
| 1 | Introduction..... | 1 |
| | General Description: | 1 |
| | In the Box..... | 1 |
| | Device features:..... | 1 |
| | Connection Types..... | 1 |
| | Protocol Types..... | 1 |
| | Power Supply | 1 |
| | Data Inputs/Output Types | 2 |
| | Applications | 2 |
| | Software Support..... | 2 |
| | Contact Details..... | 2 |
| 2 | Installation..... | 2 |
| | Operating System Support | 2 |
| | Installation..... | 2 |
| | EDRE API..... | 2 |
| | USB Usart | 2 |
| 3 | Programing Guide..... | 6 |
| | EDR Enhanced API..... | 6 |
| | Digital Inputs..... | 6 |
| | Reading the digital inputs | 6 |
| | API-Call | 6 |
| | ACTIVEX Call | 6 |
| | Digital Outputs..... | 6 |
| | Writing the digital outputs | 6 |
| | API Call..... | 7 |
| | ACTIVX Call..... | 7 |
| | Analog Output..... | 7 |
| | Writing the Analog Outputs | 7 |
| | API Call..... | 7 |
| | ACTIVEX Call | 7 |
| | Analog Input | 8 |
| | Reading the Analog Inputs..... | 8 |
| | API Call..... | 8 |
| | ACTIVEX Call | 8 |
| | Single-Ended..... | 8 |
| | Differential-Ended..... | 9 |
| | Gain Settings..... | 9 |

| | |
|---|----|
| Range Settings..... | 9 |
| Query Function | 10 |
| API Call..... | 10 |
| Query Codes..... | 10 |
| Error Codes | 11 |
| 4 Interconnections | 12 |
| Communication Connection Types | 12 |
| Communication Connection Pin Assignments..... | 12 |
| RS-232 (DB9 Male) | 12 |
| RS-422/485 (5-Way Screw Terminal)..... | 12 |
| Ethernet (RJ-45)..... | 13 |
| Data Connection Types | 13 |
| Data Connection Pin Assignments | 13 |
| Analog Input | 13 |
| Analog Output..... | 13 |
| Digital Input | 14 |
| Digital Output | 14 |
| Power Connection Types..... | 14 |
| Power Connection Pin Assignments | 14 |
| 8 - 40 VDC | 14 |
| Pin Descriptions | 15 |
| AI1...8 | 15 |
| AO1...2..... | 15 |
| DI1A...8A..... | 15 |
| DI1B...8B | 15 |
| NO1...8..... | 15 |
| NC1...8 | 15 |
| COM1...8..... | 15 |
| AGND | 15 |
| GND or DGND | 15 |
| 5 Hardware Introduction | 16 |
| Physical | 16 |
| MIO-8288 Panel View | 16 |
| Dimensions..... | 16 |
| Operation..... | 16 |
| Load Default IP Settings | 16 |
| Hardware Reset | 17 |
| RUN LED | 17 |
| Firmware Upgrade | 17 |

| | |
|--|----|
| BatchISP..... | 17 |
| Activating the ISP | 17 |
| Driver Installation | 17 |
| Loading Firmware..... | 20 |
| A. Specifications | 2 |
| Digital Input/Output Characteristics | 2 |
| Analog Input/Output Characteristics | 3 |
| Serial Interface | 3 |
| LAN Interface | 3 |
| Power Requirements | 3 |
| Environmental / Physical | 3 |
| Isolation..... | 3 |
| Data Storage..... | 3 |
| B. MODBUS Codes and Address Registers | 4 |
| Function Codes | 4 |
| 0x01 Read Coils..... | 4 |
| 0x05 Write Single Coil..... | 4 |
| 0x0F Write Multiple Coils | 6 |
| 0x04 Read Input Registers | 7 |
| 0x10 Write Multiple Registers | 8 |
| 0x03 Read Holding Registers (Query function) | 9 |
| 0x08 Diagnostics..... | 10 |
| 0x0B Get Communication Event Count (Serial Line Only)..... | 11 |
| General Description | 12 |
| Protocol Description | 12 |
| Data Encoding..... | 12 |
| C. Troubleshooting & Maintenance..... | 13 |
| Troubleshooting | 13 |
| Device Unavailable on Serial Connection | 13 |
| Maintenance | 13 |
| Calibration..... | 13 |
| D. Ordering Information | 14 |

Table of Figures

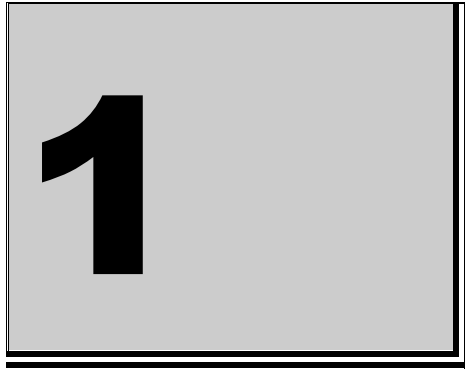
| | |
|--|----|
| FIGURE 2-1: FOUND NEW HARDWARE WIZARD..... | 4 |
| FIGURE 2-2: INSTALL FROM A LIST OR SPECIFIC LOCATION | 4 |
| FIGURE 2-3: BROWSE DIRECTORY..... | 5 |
| FIGURE 2-4: FINISH..... | 5 |
| FIGURE 2-5: USB USART PORT NUMBER IN DEVICE MANAGER..... | 6 |
| FIGURE 5-1: MIO-8288 PANEL VIEW..... | 19 |
| FIGURE 5-2: FOUND NEW HARDWARE WIZARD..... | 21 |
| FIGURE 5-3: INSTALL FROM A LIST OR SPECIFIC LOCATION. | 21 |
| FIGURE 5-4: BROWSE DIRECTORY..... | 22 |
| FIGURE 5-5: FINISH | 22 |
| FIGURE 5-6: TYPICAL BATCHISP COMMAND LINE..... | 23 |
| FIGURE B-1: GENERAL MODBUS FRAME..... | 34 |

Table of Tables

| | |
|---|----|
| TABLE 3-1: DAC SOFTWARE CHANNEL TO HARDWARE CONNECTIONS..... | 8 |
| TABLE 3-2: EDRE_DAWRITE FUNCTION. | 8 |
| TABLE 3-3: EDRE_ADCREAD FUNCTION..... | 9 |
| TABLE 3-4: SINGLE ENDED ASSIGNED SOFTWARE CHANNEL TO HARDWARE CONNECTIONS. | 10 |
| TABLE 3-5: DIFFERENTIAL-ENDED ASSIGNED SOFTWARE CHANNEL TO HARDWARE CONNECTIONS. | 10 |
| TABLE 3-6: SINGLE-ENDED GAIN SETTINGS. | 10 |
| TABLE 3-7: DIFFERENTIAL-ENDED GAIN SETTINGS. | 11 |
| TABLE 3-8: RANGE SETTINGS..... | 11 |
| TABLE 3-9: QUERY CODES | 13 |
| TABLE 3-10: ERROR CODES..... | 14 |
| TABLE 4-1: RS-232 | 15 |
| TABLE 4-2: RS-422/485 | 15 |
| TABLE 4-3: ETHERNET | 16 |
| TABLE 4-4: ANALOG INPUT | 16 |
| TABLE 4-5: ANALOG OUTPUT | 16 |
| TABLE 4-6: OPTICALLY ISOLATED INPUTS | 17 |
| TABLE 4-7: ELECTRO-MECHANICAL RELAY | 17 |
| TABLE 4-8: POWER SUPPLY CONTACT..... | 18 |
| TABLE 5-1: DEFAULT IP SETTINGS..... | 20 |
| TABLE B-1: COIL ADDRESS REGISTER FOR DIGITAL INPUTS | 27 |
| TABLE B-2: COIL ADDRESS REGISTER FOR RELAYS | 28 |
| TABLE B-3: INPUT REGISTER MAP..... | 29 |
| TABLE B-4: OUTPUT REGISTER MAP..... | 30 |

TABLE B-5: HOLDING REGISTER MAP31

TABLE B-6: DIAGNOSTIC SUB-FUNCTIONS32



1 Introduction

General Description:

The MIO-8288 is much more than just a Data Logger. It is the next generation in data acquisition hardware.

The MIO-8288 comes standard with a selection of four different interface ports. These can be used to control the units 8 analog input channels, 2 analog output channels, 8 optically isolated digital inputs and 8 electro-mechanical relay outputs. All IO's are fitted with removable screw terminals blocks for easy and fast installation.

The MIO-8288 provides the user with the option of setting the analog input from +/-10 Volt input range to 4-20mAmp input range with the flick of a switch. No external components or accessories needed.

The MIO-8288 uses an SD Card for data storage, providing it with stand-alone data logging capability. Data logger setup is easily done in a web browser. The data is then logged on the SD Card in CSV (comma separated value) format that can be easily loaded into any spreadsheet. No driver or any software installation necessary.

The key to the MIO-8288 concept is that it is easy to use, flexible and cost effective. You do not need engineering experience in complicated data acquisition hardware. Anyone familiar with a personal computer and some programming skills can construct a data acquisition system. The devices can be interfaced to any personal computer with a USB or serial port or to a PLC.

In the Box

MIO-8288 package will contain the following:

- MIO-8288 Multi IO DAQ unit.
- Documentation and Software CD.
- Din Rail Mounting Kit.
- Quick Start Guide.

Device features:

Connection Types

- Serial RS-232
- Serial RS-422/RS-485
- Ethernet 10/100Mbps
- USB

Protocol Types

- EDRE
- MODBUS RTU (Remote Terminal unit)

Power Supply

- 8-40 VDC, 4.2W Max

Data Inputs/Output Types

- 8 Analog Inputs.
- 2 Analog Outputs.
- 8 Optically Isolated Inputs.
- 8 Electro-Mechanical Relay Outputs.

Applications

The MIO-8288 can be used in many different applications:

- Automation test equipment.
- Plant/Factory process control.
- Controlling and monitoring of equipment.
- Mobile computing.
- Laboratory applications.
- Interface with PLC.
- Remote Data Logging Applications

Software Support

The MIO-8288 have support for different communication protocols. The most well known is MODBUS. The MIO-8288 supports the industry standard MODBUS RTU (Remote Terminal Unit) protocol. Because the device supports a standard protocol, NO driver and NO software installation is needed.

The MIO-8288 also supports the EDRE software protocol. This makes the interface faster and easier.

Each device comes with example code. The example software will help you to get your hardware going very quickly. It also makes it easy to develop complicated control applications. All operating system are supported as no drivers are needed.

For further support information see the Contact Details section.

Contact Details

Eagle Technology

PO Box 4376
Cape Town
8000
South Africa

Eagle Technology

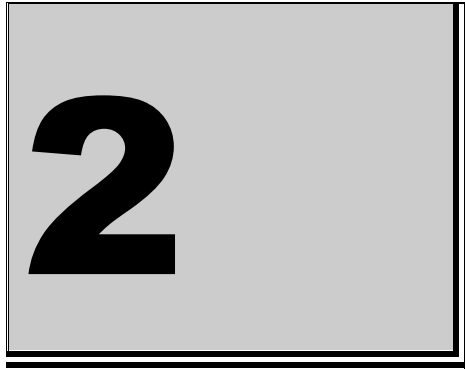
24 Burg Street
Cape Town
8001

Telephone +27 (021) 423 4943

Fax +27 (021) 424 4637

E-Mail eagle@eagle.co.za

Website <http://www.eagledaq.com>



2 Installation

Because this Data Logger is configured via an Internet Browser and DAQ control can be done using the MODBUS protocol, no drivers are necessary and no software needs to be installed. If you however want to use the EDRE protocol, you need to install the EDRE API.

Operating System Support

- Windows 2000
- Windows XP (32 bit)
- Windows Vista (32 bit)
- Windows Vista (64 bit)
- Windows 7 (32 bit)
- Windows 7 (64 bit)

Installation

There are two parts to the installation. Installing the EDRE API and installing the USB Usart driver.

EDRE API

Run the file EDREAPI.exe on the CD that was supplied.
Follow the installation instructions to complete the installation.

USB Usart

Ensure that the USB port is configured as a Usart. Connect the MIO-8288 device to the PC's USB port. Ensure that the MIO-8288 unit is power up. Doing this for the first time will open a new hardware installation window.

Choose not to connect to Windows Update for this installation and click “Next”.



Figure 2-1: Found New Hardware Wizard.

On the next screen, select “Install from a list or specific location (Advanced)” and click “Next”.

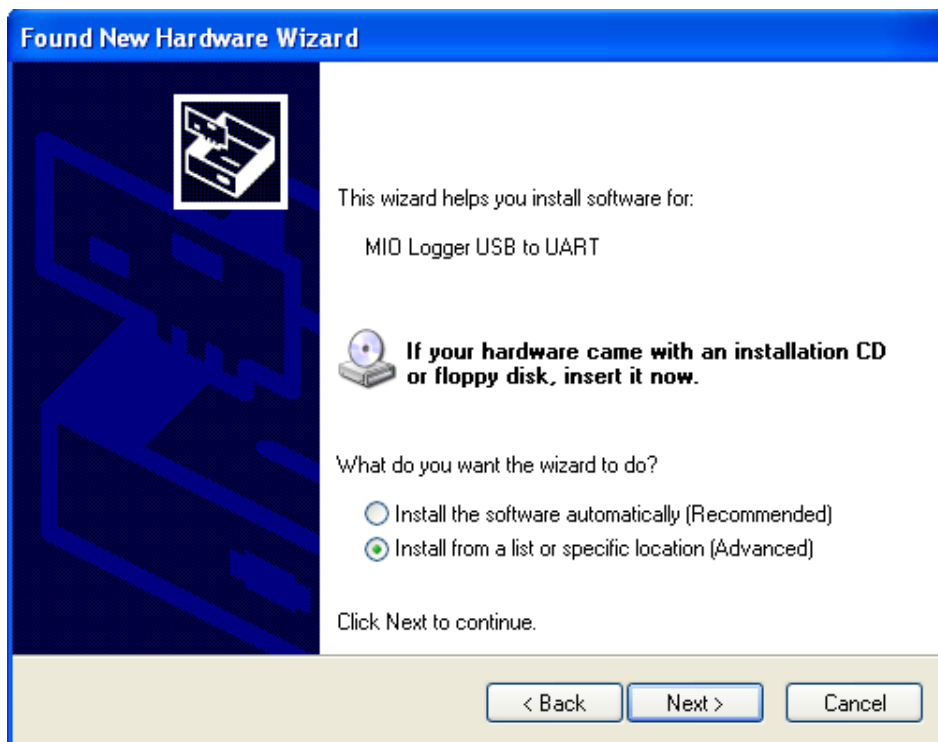


Figure 2-2: Install from a list or specific location

Then request to search in the “\Drivers\USB Usart\” on the CD that was include and click “Next”.

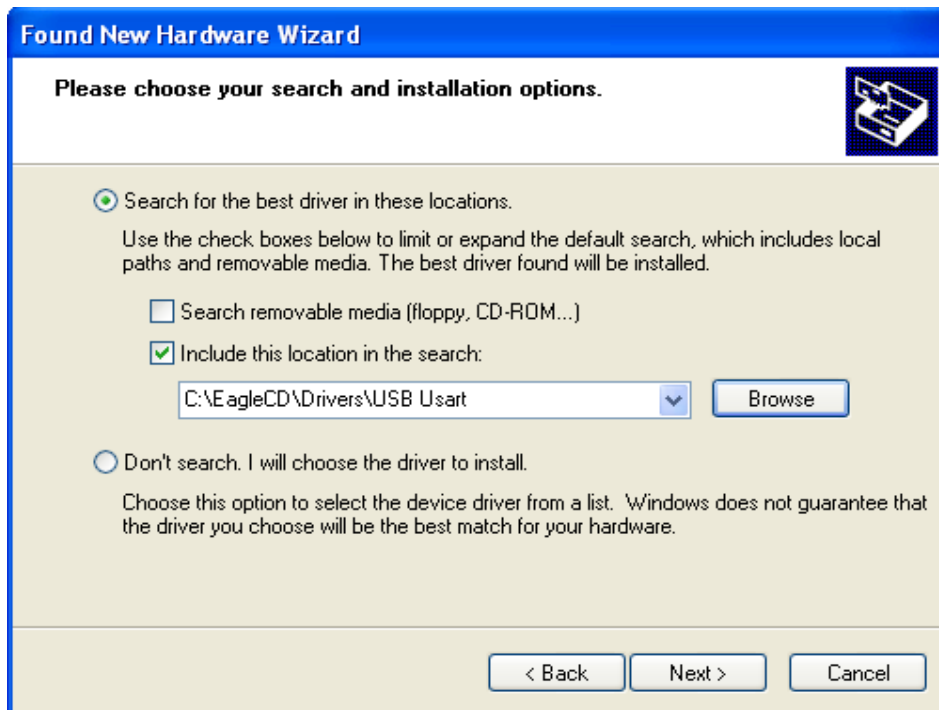


Figure 2-3: Browse Directory.

Windows will then process the installation of the driver. Once completed, click “Finish”.



Figure 2-4: Finish.

The USB Usart should now be listed as “MIO Logger USB to UART” under “Ports (COM & LPT)” in the “Device Manager”. It will also indicate the assigned COM number for the port.

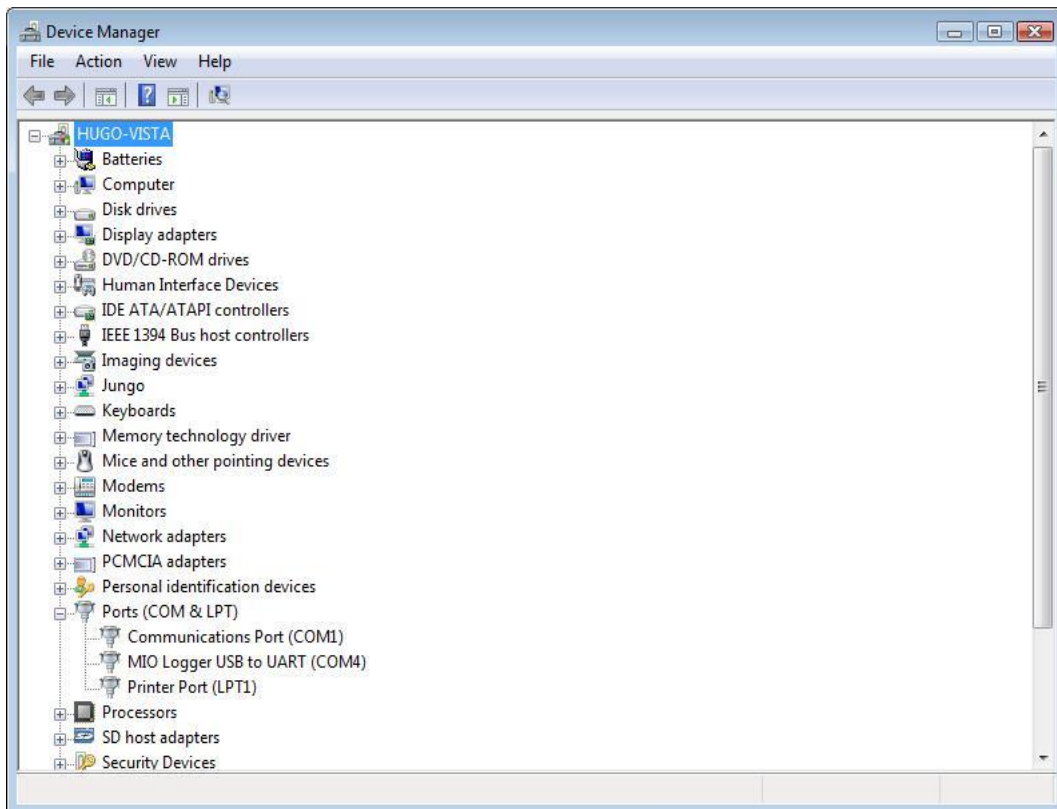
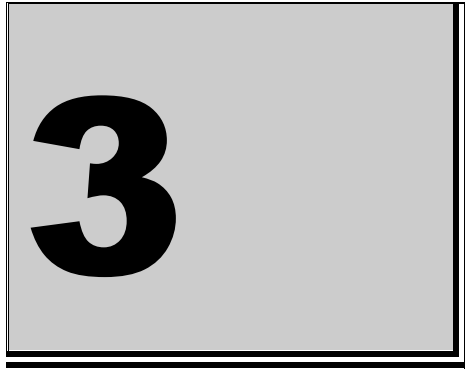


Figure 2-5: USB Usart Port Number in Device Manager.



3 Programing Guide

Even tho the MIO-8288 can be controlled and configured without installing any drivers or using any custom software, the EDRE SDK is still provided.

The EDRE comes with drivers and a common application program interface (API). The API also serves as a hardware abstraction layer (HAL) between the control application and the hardware. The EDRE API makes it possible to write an application that can be used on all hardware with common sub-systems.

EDR Enhanced API

The EDR Enhanced SDK comes with both ActiveX controls and a Windows DLL API. Examples are provided in many different languages and serves as tutorials. EDRE is also supplied with a software manual and user's guide.

The EDRE API hides the complexity of the hardware and makes it really easy to program the MIO-8288 device. It has got functions for each basic sub-system and is real easy to learn.

Digital Inputs

The MIO-8288 have 8 optically isolated input channels. The 8 inputs are situated on a 8-bit port.

Reading the digital inputs

A single call is necessary to read a digital Input. The **Port** parameter will always be equal to 0 because there is only one digital input port.

API-Call

long EDRE_DioRead(unsigned long SerialNumber, unsigned long Port, unsigned long *Value);

The serial number, port and pointer to variable that return the result must be passed by the calling function. A return code will indicate if any error occurred.

ACTIVEX Call

long EDREDioX.Read(long Port);

Only the port number needs to be passed and the returned value will either hold an error or the value read. If the value is negative an error occurred.

Digital Outputs

The MIO-8288 have 8 electro-mechanical relays. The 8 outputs are situated on a 8-bit port.

Writing the digital outputs

A single call is necessary to write a digital output. The **Port** parameter will always be equal to 0 because there is only one digital output port.

API Call

long EDRE_DioWrite(unsigned long SerialNumber, unsigned long Port, unsigned long Value);

The serial number, port and a value must be passed by the calling function. A return code will indicate if any error occurred.

ACTIVX Call

long EDREDioX.Write(long Port, long Value);

The port number and value to be written needs to be passed and the return value will indicate if an error occurred.

Analog Output

The MIO-8288 has 2 12-bit analog output channels with a range of ± 10 volt. These channels are very easy to program. A single command is used to write to them.

Writing the Analog Outputs

A single call is necessary to write voltage levels to a analog output channel. The table below shows the relation between the software channel and the channel on the connector.

| Assigned Software Channel | Assigned Connector Pin |
|---------------------------|------------------------|
| 0 | AO1 |
| 1 | AO2 |

Table 3-1: DAC Software Channel to Hardware Connections.

API Call

long EDRE_DAWrite(unsigned long SerialNumber, unsigned long Channel, long uVoltage);

The serial number, DAC channel and micro-voltage level is needed to set the DAC output. A return code will indicate if any error occurred.

| Parameter | Type | Description |
|---------------|---------------|---|
| Serial Number | unsigned long | Device Serial Number |
| Channel | unsigned long | The analog channel to write. |
| uVoltage | long | The voltage to write to the analog output channel in micro volts. |

Table 3-2: EDRE_DAWrite function.

ACTIVEX Call

long EDREDAX.Write(unsigned long Channel, long uVoltage);

The DAC channel and micro-voltage is needed to set a DAC channel's voltage. A return code will indicate if any errors occurred.

Analog Input

The MIO-8288 have 8 analog input channels.

The inputs can be used as 8 single-ended inputs, 4 differential-ended inputs or combination of the two.

Reading the Analog Inputs

A single call is necessary to configure a analog input channel and at the same time read the voltage level on the analog input.

API Call

long EDRE_ADCSingle(unsigned long SerialNumber, unsigned long Channel, unsigned long Gain, unsigned long Range, long *uVoltage);

The serial number, ADC channel, gain, range and a pointer to the uVoltage variable must be passed. A return code will indicate if an error occurred.

| Parameter | Type | Description | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---------------|-------------------|--|--------------|--------------------|--------------------|---|------|------|---|--|------|---|--|-----|---|--|-----|---|--|-------|---|--|-----|---|--|--------|---|--|-----|
| Serial Number | unsigned long | Device Serial Number. | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Channel | unsigned long | The channel to sample. | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Gain | unsigned long | Gain contains the gain setting of the channel that is being sampled. | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | <table border="1"> <thead> <tr> <th>Value</th> <th>Single-Ended</th> <th>Differential-Ended</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>±10V</td> <td>±20V</td> </tr> <tr> <td>1</td> <td></td> <td>±10V</td> </tr> <tr> <td>2</td> <td></td> <td>±5V</td> </tr> <tr> <td>3</td> <td></td> <td>±4V</td> </tr> <tr> <td>4</td> <td></td> <td>±2.5V</td> </tr> <tr> <td>5</td> <td></td> <td>±2V</td> </tr> <tr> <td>6</td> <td></td> <td>±1.25V</td> </tr> <tr> <td>7</td> <td></td> <td>±1V</td> </tr> </tbody> </table> | Value | Single-Ended | Differential-Ended | 0 | ±10V | ±20V | 1 | | ±10V | 2 | | ±5V | 3 | | ±4V | 4 | | ±2.5V | 5 | | ±2V | 6 | | ±1.25V | 7 | | ±1V |
| | | Value | Single-Ended | Differential-Ended | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 0 | ±10V | ±20V | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 1 | | ±10V | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 2 | | ±5V | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 3 | | ±4V | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 4 | | ±2.5V | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 5 | | ±2V | | | | | | | | | | | | | | | | | | | | | | | | | |
| 6 | | ±1.25V | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 7 | | ±1V | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Range | unsigned long | Range contains the range setting of the channel that is being sampled. | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| uVoltage | Pointer to a long | A pointer to the uVoltage variable that must return the result of the channel being sampled, in micro Voltage. | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table 3-3: EDRE_ADCRead function.

ACTIVE X Call

long EDREADX.SingleRead(long Channel);

Make sure to set the gain and Range properties of the ActiveX control. This will in turn set range and gain when reading the ADC channel. A return code will indicate if any errors occurred.

Single-Ended

| Value | AI1 | AI2 | AI3 | AI4 | AI5 | AI6 | AI7 | AI8 | GND |
|-------|-----|-----|-----|-----|-----|-----|-----|-----|----------|
| 0 | + | | | | | | | | - (AGND) |
| 1 | | + | | | | | | | - (AGND) |
| 2 | | | + | | | | | | - (AGND) |

| Value | AI1 | AI2 | AI3 | AI4 | AI5 | AI6 | AI7 | AI8 | GND |
|-------|-----|-----|-----|-----|-----|-----|-----|-----|----------|
| 3 | | | | + | | | | | - (AGND) |
| 4 | | | | | + | | | | - (AGND) |
| 5 | | | | | | + | | | - (AGND) |
| 6 | | | | | | | + | | - (AGND) |
| 7 | | | | | | | | + | - (AGND) |

Table 3-4: Single Ended Assigned Software Channel to Hardware Connections.

Differential-Ended

| Value | AI1 | AI2 | AI3 | AI4 | AI5 | AI6 | AI7 | AI8 |
|-------|-----|-----|-----|-----|-----|-----|-----|-----|
| 0 | + | - | | | | | | |
| 1 | | | + | - | | | | |
| 2 | | | | | + | - | | |
| 3 | | | | | | | + | - |
| 4 | - | + | | | | | | |
| 5 | | | - | + | | | | |
| 6 | | | | | - | + | | |
| 7 | | | | | | | - | + |

Table 3-5: Differential-Ended Assigned Software Channel to Hardware Connections.

Gain Settings

| Value | Voltage Range |
|-------|---------------|
| 0 | ±10 |

Table 3-6: Single-Ended Gain Settings.

| Value | Voltage Range | Gain x |
|-------|---------------|--------|
| 0 | ±20 | 1 |
| 1 | ±10 | 2 |
| 2 | ±5 | 4 |
| 3 | ±4 | 5 |
| 4 | ±2.5 | 8 |
| 5 | ±2 | 10 |
| 6 | ±1.25 | 16 |
| 7 | ±1 | 20 |

Table 3-7: Differential-Ended Gain Settings.

Range Settings

| Value | Range |
|-------|--------------------|
| 0 | Single-Ended |
| 1 | Differential-Ended |

Table 3-8: Range Settings.

Query Function

The Query function can be used to get or set more information about the MIO-8288 device.

API Call

long EDRE_Query(unsigned long SerialNumber, unsigned long QueryCode, unsigned long Param);

A serial number, query code and parameter must be specified when doing a query. A return value will indicate the result. The return code will be negative if an error occurred.

Query Codes

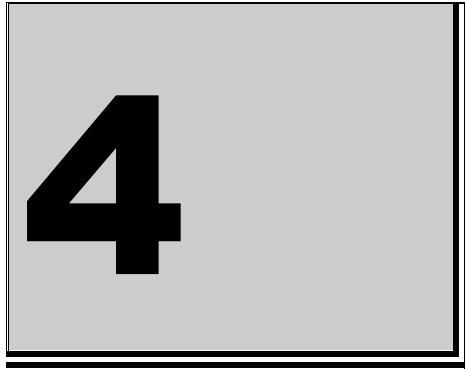
| Name | Value | Description |
|----------------------|-------|--|
| APIMAJOR | 1 | Query EDRE API major version number. |
| APIMINOR | 2 | Query EDRE API minor version number. |
| APIBUILD | 3 | Query EDRE API build version number. |
| APIOS | 4 | Query API OS type. |
| APINUMDEV | 5 | Query number of devices installed. |
| BRDTYPE | 10 | Query a board's type. |
| BRDREV | 11 | Query a board's revision. |
| BRDYEAR | 12 | Query a board's manufactured year. |
| BRDMONTH | 13 | Query a board's manufactured month. |
| BRDDAY | 14 | Query a board's manufactured day. |
| BRDSERIALNO | 15 | Query a board's serial number. |
| DRVMAJOR | 20 | Query driver's major version number. |
| DVRMINOR | 21 | Query driver's minor version number. |
| DRVBUILD | 22 | Query driver's build version number. |
| FRMMAJOR | 23 | Query the firmware major version number. |
| FRMMINOR | 24 | Query the firmware minor version number. |
| FRMBUILD | 25 | Query the firmware build version number. |
| ADNUMCHAN | 100 | Query number of ADC channels. |
| DANUMCHAN | 200 | Query number of DAC channels. |
| DIONUMPORT | 400 | Query number of digital I/O ports. |
| NETSETIPADDRESS | 604 | Set board's IP Address. |
| NETSETMASK | 605 | Set board's Mask Address. |
| NETSETGATEWAY | 606 | Set board's Gateway. |
| NETGETIPADDRESS | 608 | Get board's IP Address. |
| NETGETMASK | 609 | Get board's Mask Address. |
| NETGETGATEWAY | 610 | Get board's Gateway. |
| NETGETMACCOMPANYID | 613 | Get board's company MAC ID. |
| NETGETMACEXTENSIONID | 614 | Get board's extension MAC ID. |
| SRLGETBAUD | 700 | Query the device Baud Rate. |
| SRLSETBAUD | 701 | Set the device Baud Rate. |
| RTCSETTIME | 800 | Set device time. |
| RTCGETTIME | 801 | Get device time. |

Table 3-9: Query Codes

Error Codes

| Name | Value | Description |
|----------------------|-------|---|
| EDRE_OK | 0 | Function successful. |
| EDRE_FAIL | -1 | Function call failed. |
| EDRE_BAD_FN | -2 | Invalid function call. |
| EDRE_BAD_SN | -3 | Invalid serial number. |
| EDRE_BAD_DEVICE | -4 | Invalid device. |
| EDRE_BAD_OS | -5 | Function not supported by operating system. |
| EDRE_EVENT_FAILED | -6 | Wait on event failed. |
| EDRE_EVENT TIMEOUT | -7 | Event timed out. |
| EDRE_INT_SET | -8 | Interrupt in use |
| EDRE_DA_BAD_RANGE | -9 | DAC value out of range. |
| EDRE_AD_BAD_CHANLIST | -10 | Channel list size out of range. |
| EDRE_BAD_FREQUENCY | -11 | Frequency out of range. |
| EDRE_BAD_BUFFER_SIZE | -12 | Data passed by buffer incorrectly sized. |
| EDRE_BAD_PORT | -13 | Port value out of range. |
| EDRE_BAD_PARAMETER | -14 | Invalid parameter value specified. |
| EDRE_BUSY | -15 | System busy. |
| EDRE_IO_FAIL | -16 | IO call failed. |
| EDRE_BAD_ADGAIN | -17 | ADC-gain out of range. |
| EDRE_BAD_QUERY | -18 | Query value not supported. |
| EDRE_BAD_CHAN | -19 | Channel number out of range. |
| EDRE_BAD_VALUE | -20 | Configuration value specified out of range. |
| EDRE_BAD_CT | -21 | Counter-timer channel out of range. |
| EDRE_BAD_CHANLIST | -22 | Channel list invalid. |
| EDRE_BAD_CONFIG | -23 | Configuration invalid. |
| EDRE_BAD_MODE | -24 | Mode not valid. |
| EDRE_HW_ERROR | -25 | Hardware error occurred. |
| EDRE_HW_BUSY | -26 | Hardware busy. |
| EDRE_BAD_BUFFER | -27 | Buffer invalid. |
| EDRE_REG_ERROR | -28 | Registry error occurred. |
| EDRE_OUT_RES | -29 | Out of resources. |
| EDRE_IO_PENDING | -30 | Waiting on I/O completion. |

Table 3-10: Error Codes



4 Interconnections

Communication Connection Types

- RS-232 (DB9 Male)
- RS-422/485 (5-Way Screw Terminal)
- USB as Usart (Type B)
- Ethernet (RJ-45)

Communication Connection Pin Assignments

RS-232 (DB9 Male)

| Pin Number | RS-232 Signal |
|------------|---------------|
| 2 | RXD |
| 3 | TXD |
| 5 | GND |
| 7 | RTS |
| 8 | CTS |

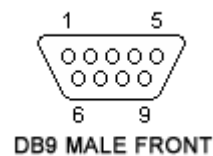


Table 4-1: RS-232

RS-422/485 (5-Way Screw Terminal)

| Pin Number | RS-485 Signal |
|------------|---------------|
| 1 | TXD+ |
| 2 | TXD- |
| 3 | RXD+ |
| 4 | RXD- |
| 5 | GND |

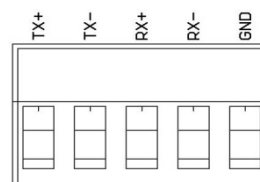
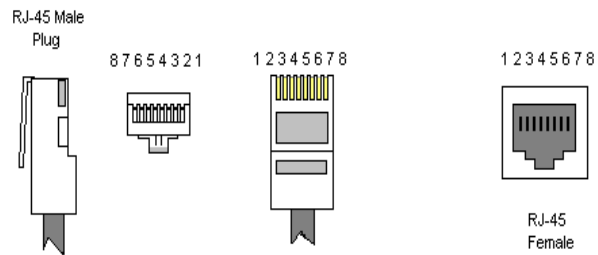


Table 4-2: RS-422/485

Ethernet (RJ-45)

| Pin Number | Ethernet Signal |
|------------|-----------------|
| 1 | TD+ |
| 2 | TD- |
| 3 | RD+ |
| 4 | TDCT |
| 5 | RDCT |
| 6 | RD- |
| 8 | GND |

**Table 4-3: Ethernet****Data Connection Types**

- Analog Input (2x 5-Way Screw Terminal)
- Analog Output (3-Way Screw Terminal)
- Digital Input (2x 8-Way Screw Terminal)
- Digital Output (2x 12-Way Screw Terminal)

Data Connection Pin Assignments**Analog Input**

| Pin Number | Signal |
|------------|--------|
| 1 | AI1 |
| 2 | AI2 |
| 3 | AGND |
| 4 | AI3 |
| 5 | AI4 |
| 6 | AI5 |
| 7 | AI6 |
| 8 | AGND |
| 9 | AI7 |
| 10 | AI8 |

Table 4-4: Analog Input**Analog Output**

| Pin Number | Signal |
|------------|--------|
| 1 | AO1 |
| 2 | AGND |
| 3 | AO2 |

Table 4-5: Analog Output

Digital Input

| Pin Number | Signal |
|------------|--------|
| 1 | DI1A |
| 2 | DI1B |
| 3 | DI2A |
| 4 | DI2B |
| 5 | DI3A |
| 6 | DI3B |
| 7 | DI4A |
| 8 | DI4B |
| 9 | DI5A |
| 10 | DI5B |
| 11 | DI6A |
| 12 | DI6B |
| 13 | DI7A |
| 14 | DI7B |
| 15 | DI8A |
| 16 | DI8B |

Table 4-6: Optically Isolated Inputs**Digital Output**

| Pin Number | Signal |
|------------|--------|
| 1 | NO1 |
| 2 | COM1 |
| 3 | NC1 |
| 4 | NO2 |
| 5 | COM2 |
| 6 | NC2 |
| 7 | NO3 |
| 8 | COM3 |
| 9 | NC3 |
| 10 | NO4 |
| 11 | COM4 |
| 12 | NC4 |
| 13 | NO5 |
| 14 | COM5 |
| 15 | NC5 |
| 16 | NO6 |
| 17 | COM6 |
| 18 | NC6 |
| 19 | NO7 |
| 20 | COM7 |
| 21 | NC7 |
| 22 | NO8 |
| 23 | COM8 |
| 24 | NC8 |

Table 4-7: Electro-Mechanical Relay**Power Connection Types**

- 8 – 40 VDC (3-Way Screw Terminal)

Power Connection Pin Assignments**8 - 40 VDC**

| Pin Number | Signal |
|------------|----------------|
| 1 | + Power Supply |
| 2 | - Power Supply |
| 3 | EARTH |

Table 4-8: Power Supply Contact

Pin Descriptions

AI1...8

Analog Input channels 1 to 8.

AO1...2

Analog Output channels 1 and 2.

DI1A...8A

Optically Isolated Input channels input pin A. This is the Positive input terminal for the optically isolated input channel

DI1B...8B

Optically Isolated Input channels input pin B. This is the Negative or reference input terminal for the optically isolated input channel.

NO1...8

Electro-Mechanical Relay Normally Open Contact Pin. When the relay is in the ON position, this pin is connected to its relevant COM pair and not connected or open when the relay is in the OFF position.

NC1...8

Electro-Mechanical Relay Normally Close Contact Pin. When the relay is in the OFF position, this pin is connected to its relevant COM pair and not connected when the relay is in the ON position.

COM1...8

Electro-Mechanical Relay Common Contact Pin. This Pin is either connected to its relevant NO or NC contact pin depending on the state of the relay.

AGND

Analog Ground used in conjunction with analog inputs and analog outputs.

GND or DGND

Digital Ground. This is the same as the “-” or Negative input on the Power Supply (8–40VDC) connector.

5

5 Hardware Introduction

Physical

MIO-8288 Panel View

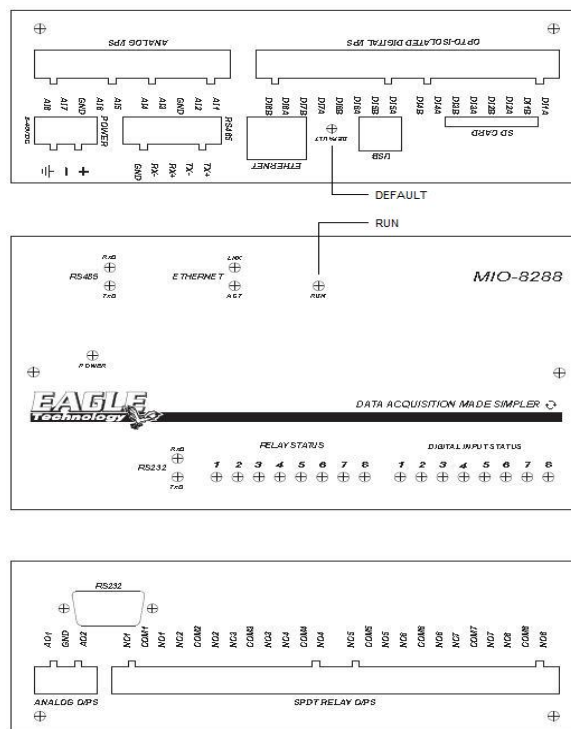


Figure 5-1: MIO-8288 Panel View

Dimensions

| | |
|---------------|------------------------------------|
| (W x D x H) | 163 x 108 x 60 mm (With Feet) |
| | 6.42 x 4.25 x 2.36 in(With Feet) |
| | 163 x 108 x 70 mm (DIN Rail) |
| | 6.42 x 4.25 x 2.75 in(DIN Rail) |

Operation

Load Default IP Settings

DEFAULT Button – Press the DEFAULT button continuously for 10 seconds to load the factory default IP settings: Use a pointed object, such as a straightened paper clip or toothpick, to press the DEFAULT button. This will cause the RUN LED to stop blinking. The

LED will change state after 10 seconds and the default IP settings will be loaded. At this state you should release the DEFAULT button. To finish loading the default settings, a **Hardware Reset** should be done.

| Description | Default Settings |
|-------------|------------------|
| IP Address | 192.168.0.2 |
| GATEWAY | 192.168.0.1 |
| NET MASK | 255.255.255.0 |

Table 5-1: Default IP Settings

Hardware Reset

To do a hardware reset the power must be disconnected for ± 5 Seconds and then be reconnected. The device will now reboot and reload all settings. This might take a couple of seconds.

RUN LED

The RUN LED will indicate when the device is used as a Data Logger and busy logging data to the SD/MMC storage device.

Firmware Upgrade

Firmware is the software the runs on a micro controller. It is like the O/S and application program all in one running on you PC.

When the firmware on the device needs to be upgrade, two components are needed. The program that is used to load the firmware and the new firmware.

BatchISP

The application that is used to load the firmware is called "BatchISP".

BatchISP is a command line tool that allows to program parts containing an embedded Atmel ISP. It comes with FLIP 3. To install BatchISP, first install FLIP 3 using its installer. You can find the application on the CD. "Flip_Installer_3.4.2.exe".

Activating the ISP

To load new firmware, the ISP must be activated.

- This is done by disconnecting the power to the MIO-8288 device.
- Connect the MIO-8288 device to the PC via a USB cable.
- Push and hold the **DEFAULT** button while the MIO-8288 device is power up.
- This will activate the SPI.

Driver Installation

Activating the ISP for the first time will open a new hardware installation window. Choose not to connect to Windows Update for this installation and click "Next".



Figure 5-2: Found New Hardware Wizard.

On the next screen, select “Install from a list or specific location (Advanced)” and click “Next”.



Figure 5-3: Install from a list or specific location.

Then request to search in the “USB” folder of the FLIP installation directory as shown below and click “Next”.

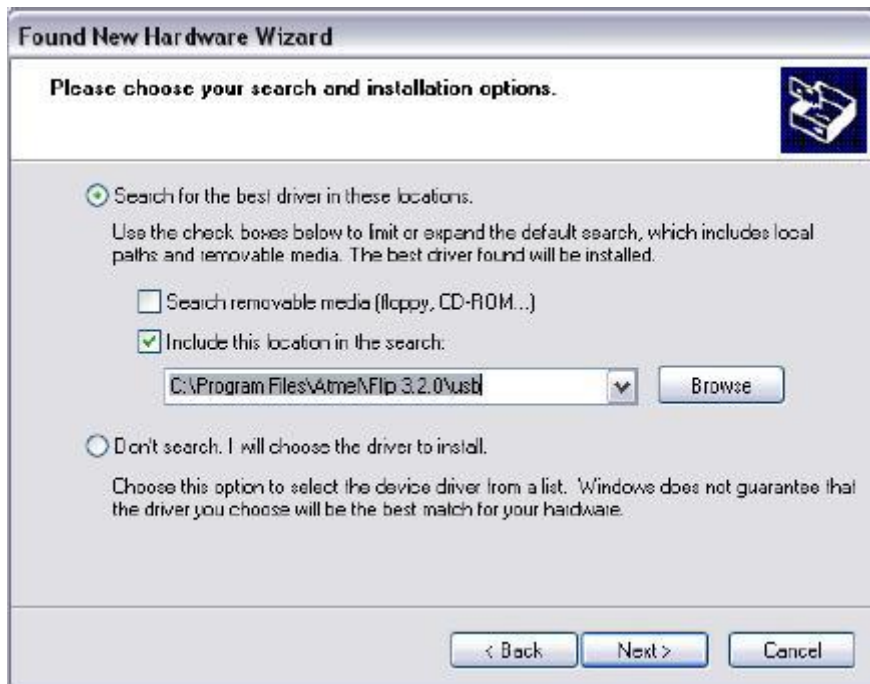


Figure 5-4: Browse Directory.

Windows will then process the installation of the driver corresponding to the ISP of the connected part. Once completed, click “Finish”.



Figure 5-5: Finish

Loading Firmware

First see Activating the ISP.

To launch BatchISP, open a command prompt. Windows or Cygwin command prompt can be used provided that the `bin` folder of the FLIP installation directory is in the `PATH` (Windows' or Cygwin's) environment variable.

When running BatchISP on AT32UC3A0512, the target part has to be specified with `-device at32uc3a0512` and the communication port with `-hardware usb`. Commands can then be placed after `-operation`. These commands are executed in order. BatchISP options can be placed in a text file invoked using `-cmdfile` rather than on the command line.

BatchISP works with an internal ISP buffer per target memory. These ISP buffers can be filled from several sources. All target operations (program, verify, read) are performed using these buffers.

A typical BatchISP command line programming an application will look like this:

```
batchisp -device at32uc3a0512 -hardware usb -operation erase f memory flash
blankcheck loadbuffer MIO-8288.elf program verify start reset 0
```

A Hardware Reset must now be done.

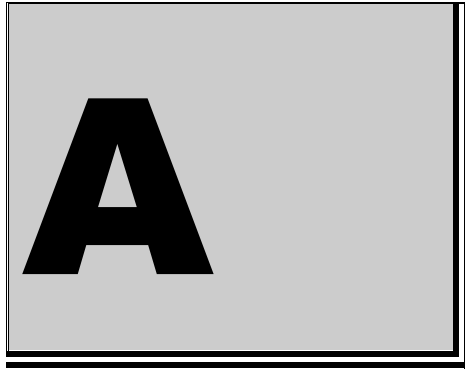
```
C:\WINDOWS\System32\cmd.exe
C:\Program Files\Atmel\Flip 3.4.2\bin>batchisp -device at32uc3a0512 -hardware us
b -operation erase f memory flash blankcheck loadbuffer MIO-8288.elf program ver
ify start reset 0
Running batchisp 1.2.4 on Wed Apr 06 11:49:21 2011

AT32UC3A0512 - USB - USB/DFU

Device selection..... PASS
Hardware selection..... PASS
Opening port..... PASS
Reading Bootloader version..... PASS      1.0.3
Erasing..... PASS
Selecting FLASH..... PASS
Blank checking..... PASS      0x00000 0x7ffff
Parsing ELF file..... PASS      MIO-8288.elf
WARNING: The user program and the bootloader overlap!
Programming memory..... PASS      0x00000 0x4745f
Verifying memory..... PASS      0x00000 0x4745f
Starting Application..... PASS      RESET 0

Summary: Total 11 Passed 11 Failed 0
C:\Program Files\Atmel\Flip 3.4.2\bin>
```

Figure 5-6: Typical BatchISP Command Line



A. Specifications

Digital Input/Output Characteristics

Electro Mechanical Relay Characteristics

| | |
|----------------------------|---------------------------------|
| DC Rating | |
| Rated Current / Voltage DC | 6 Amp / 28 VDC |
| Max. Switching Power | 336W |
| Max. Switching Voltage | 110VDC |
| AC Rating | |
| Rated Current / Voltage AC | 5 Amp / 250 VAC |
| Max. Switching Power | 1250VA |
| Max. Switching Voltage | 380VAC |
| Contact Resistance | < 100m Ω |
| Operate Time | < 10mS |
| Release Time | < 5mS |
| Vibration resistance | 10~55Hz double amplitude 1.5mm |
| Shock resistance | 100m/s ² 11mS |
| Terminals Strength | 5N |
| Contact Configuration | Single Pole Double Throw (SPDT) |
| Ambient temperature range | -55 ~ 85°C |
| Relative Humidity | 93% (@ 40°C) |

Optically Isolated Characteristics

| | |
|---------------------------|--|
| Frequency Response | Up to 10 kHz |
| Logic Levels | High: 3V min, 24V max Low: 0.0V min, 1.2V max |
| Isolated Voltage | 2500V |
| Input Current | 30mA Continuous 1A Peak (100 μ s Pulse) |
| Max forward current [LED] | 50mA |

Analog Input/Output Characteristics

Analog Input Characteristics

| | |
|---------------------------------|---|
| Number of Channels | 8 single-ended, 4 differential-ended |
| Input Coupling | DC |
| Maximum Working Voltage | ±20V relative to module ground |
| Resolution | 14 Bit |
| Relative Accuracy | ±2 LSB |
| Input range, single-ended | ±10V |
| Input range, differential-ended | ±20, ±10, ±5, ±4, ±2.5, ±2, ±1.25, ±1 (V) |
| Input impedance | 2 MΩ |

Analog Output Characteristics

| | |
|--------------------|---------|
| Number of Channels | 2 |
| Maximum Output | +10V |
| Minimum Output | -10V |
| Output Current | 5mA |
| Resolution | 12 Bits |
| Zero offset error | 5mV |
| Full scale error | 30mV |

Serial Interface

| | |
|------------------------|--|
| Connection Type | USB (USB as USART) RS-232 RS-485/RS-422 |
| Communication Protocol | MODBUS RTU (Remote Terminal Unit) protocol EDRE |
| Baud Rate | 1200 - 230400 Baud |
| Word length | 8 |
| Parity | NONE |
| Stop Bits | 1 |

LAN Interface

| | |
|------------------------|--|
| Connection Type | RJ-45 |
| Communication Protocol | MODBUS RTU (Remote Terminal Unit) protocol EDRE |
| Port | 7070 |

Power Requirements

| | |
|-------|-----------------------|
| Power | 8 to 40 VDC, 4.2W Max |
|-------|-----------------------|

Environmental / Physical

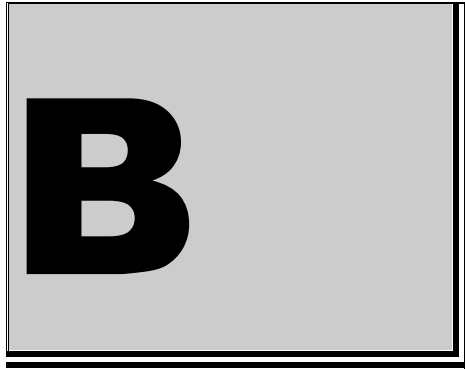
| | |
|-------------------------|--|
| Relative Humidity | 5% to 90% (non-condensing) |
| Operating Temperature | 0°C to 55°C (32 to 131°F) |
| Housing | Powder coated Aluminium |
| Dimension (W x D x H) | 163 x 108 x 60 mm (With Feet) 6.42 x 4.25 x 2.36 in(With Feet) 163 x 108 x 70 mm (DIN Rail) 6.42 x 4.25 x 2.75 in(DIN Rail) |

Isolation

| | |
|---------------------|--------------------------|
| Magnetic Isolation | 1.5 KV for Ethernet |
| Optically Isolation | 2.5 KV for Digital Input |

Data Storage

| | |
|----------------|---------------------|
| SD Card or MMC | 2 Giga Byte (Max) |
|----------------|---------------------|



B. MODBUS Codes and Address Registers

Function Codes

0x01 Read Coils

This function code is used to read from 1 to 8 contiguous status of coils in a remote device. The Request PDU specifies the starting address, i.e. the address of the first coil specified, and the number of coils. In the PDU Coils are addressed starting at zero. Therefore coils numbered 1-8 are addressed as 0-7.

The coils in the response message are packed as one coil per bit of the data field. Status is indicated as 1= ON and 0= OFF. The LSB of the first data byte contains the output addressed in the query. The other coils follow toward the high order end of this byte, and from low order to high order in subsequent bytes.

If the returned output quantity is not a multiple of eight, the remaining bits in the final data byte will be padded with zeros (toward the high order end of the byte). The Byte Count field specifies the quantity of complete bytes of data.

Request

| | | |
|--------------------------|---------|------------------|
| Function Code | 1 Byte | 0x01 |
| Starting Address | 2 Bytes | 0x0000 to 0x0007 |
| Quantity of Coils | 2 Bytes | 0x0001 to 0x0008 |

Response

| | | |
|----------------------|---------|--------------|
| Function Code | 1 Byte | 0x01 |
| Byte Count | 1 Bytes | N* |
| Coil Status | n Bytes | n=N or N + 1 |

*N = Quantity of Outputs / 8, if the remainder is different of 0 → N = N + 1

Error

| | | |
|-----------------------|---------|----------------------|
| Function Code | 1 Byte | 0x81 |
| Exception Code | 1 Bytes | 01 or 02 or 03 or 04 |

| Address Register | Description |
|------------------|-------------|
| 0x00 | OPTO 1 |
| 0x01 | OPTO 2 |
| 0x02 | OPTO 3 |
| 0x03 | OPTO 4 |
| 0x04 | OPTO 5 |
| 0x05 | OPTO 6 |
| 0x06 | OPTO 7 |
| 0x07 | OPTO 8 |

Table B-1: Coil Address Register for Digital Inputs

0x05 Write Single Coil

This function code is used to write a single output to either ON or OFF in a remote device. The requested ON/OFF state is specified by a constant in the request data field. A value of FF

00 hex requests the output to be ON. A value of 00 00 requests it to be OFF. All other values are illegal and will not affect the output.

The Request PDU specifies the address of the coil to be forced. Coils are addressed starting at zero. Therefore coil numbered 1 is addressed as 0. The requested ON/OFF state is specified by a constant in the Coil Value field. A value of 0xFF00 requests the coil to be ON. A value of 0x0000 requests the coil to be off. All other values are illegal and will not affect the coil.

The normal response is an echo of the request, returned after the coil state has been written.

Request

| | | |
|-----------------------|---------|------------------|
| Function Code | 1 Byte | 0x05 |
| Output Address | 2 Bytes | 0x0000 to 0x0007 |
| Output Value | 2 Bytes | 0x0000 or 0xFF00 |

Response

| | | |
|----------------------|---------|------------------|
| Function Code | 1 Byte | 0x05 |
| Byte Count | 2 Bytes | 0x0000 to 0x0007 |
| Coil Status | 2 Bytes | 0x0000 or 0xFF00 |

Error

| | | |
|-----------------------|---------|----------------------|
| Function Code | 1 Byte | 0x85 |
| Exception Code | 1 Bytes | 01 or 02 or 03 or 04 |

0x0F Write Multiple Coils

This function code is used to force each coil in a sequence of coils to either ON or OFF in a remote device. The Request PDU specifies the coil references to be forced. Coils are addressed starting at zero. Therefore coil numbered 1 is addressed as 0.

The requested ON/OFF states are specified by contents of the request data field. A logical '1' in a bit position of the field requests the corresponding output to be ON. A logical '0' requests it to be OFF.

The normal response returns the function code, starting address, and quantity of coils forced.

Request

| | | |
|----------------------------|-------------|------------------|
| Function Code | 1 Byte | 0x0F |
| Starting Address | 2 Bytes | 0x0000 to 0x0007 |
| Quantity of Outputs | 2 Bytes | 0x0001 to 0x0008 |
| Byte Count | 1 Byte | N* |
| Outputs Value | N* x 1 Byte | |

*N = Quantity of Outputs / 8, if the remainder is different of 0 → N = N + 1

Response

| | | |
|----------------------------|---------|------------------|
| Function Code | 1 Byte | 0x0F |
| Starting Address | 2 Bytes | 0x0000 to 0x0007 |
| Quantity of Outputs | 2 Bytes | 0x0001 or 0x0008 |

Error

| | | |
|-----------------------|---------|----------------------|
| Function Code | 1 Byte | 0x8F |
| Exception Code | 1 Bytes | 01 or 02 or 03 or 04 |

| Address Register | Description |
|------------------|-------------|
| 0x00 | Relay 1 |
| 0x01 | Relay 2 |
| 0x02 | Relay 3 |
| 0x03 | Relay 4 |
| 0x04 | Relay 5 |
| 0x05 | Relay 6 |
| 0x06 | Relay 7 |
| 0x07 | Relay 8 |

Table B-2: Coil Address Register for Relays

0x04 Read Input Registers

This function code is used to read from 1 to approx. 125 contiguous input registers in a remote device. The Request PDU specifies the starting register address and the number of registers. In the PDU Registers are addressed starting at zero. Therefore input registers numbered 1-16 are addressed as 0-15.

The register data in the response message are packed as two bytes per register, with the binary contents right justified within each byte. For each register, the first byte contains the high order bits and the second contains the lower order bits.

Request

| | | |
|------------------------------------|---------|------------------|
| Function Code | 1 Byte | 0x04 |
| Starting Address | 2 Bytes | 0x0000 to 0x000F |
| Quantity of Input Registers | 2 Bytes | 0x0001 to 0x0010 |

Response

| | | |
|------------------------|-------------|-------|
| Function Code | 1 Byte | 0x04 |
| Byte Count | 1 Bytes | 2 x N |
| Input Registers | N x 2 Bytes | |

N = Quantity of Input Registers.

Error

| | | |
|-----------------------|---------|----------------------|
| Function Code | 1 Byte | 0x84 |
| Exception Code | 1 Bytes | 01 or 02 or 03 or 04 |

| Address Register | Description |
|------------------|-------------|
| 0x00 | ADC 1 MSB |
| 0x01 | ADC 1 LSB |
| 0x02 | ADC 2 MSB |
| 0x03 | ADC 2 LSB |
| 0x04 | ADC 3 MSB |
| 0x05 | ADC 3 LSB |
| 0x06 | ADC 4 MSB |
| 0x07 | ADC 4 LSB |
| 0x08 | ADC 5 MSB |
| 0x09 | ADC 5 LSB |
| 0x0A | ADC 6 MSB |
| 0x0B | ADC 6 LSB |
| 0x0C | ADC 7 MSB |
| 0x0D | ADC 7 LSB |
| 0x0E | ADC 8 MSB |
| 0x0F | ADC 8 LSB |

Table B-3: Input Register Map

0x10 Write Multiple Registers

This function is used to write a block of contiguous registers, from 1 to approx. 120 registers, in a remote device.

The requested written values are specified in request data field. Data is packed as two bytes per register.

The normal response returns the function code, starting address, and quantity of registers written.

Request

| | | |
|------------------------------|-------------|------------------|
| Function Code | 1 Byte | 0x10 |
| Starting Address | 2 Bytes | 0x0000 to 0x005F |
| Quantity of Registers | 2 Bytes | 0x0001 to 0x0008 |
| Byte Count | 1 Byte | 2 x N |
| Registers Value | N x 2 Bytes | Value |

N = Quantity of Registers.

Response

| | | |
|------------------------------|---------|------------------|
| Function Code | 1 Byte | 0x10 |
| Starting Address | 2 Bytes | 0x0000 to 0x005F |
| Quantity of Registers | 2 Bytes | 0x0001 to 0x0008 |

Error

| | | |
|-----------------------|---------|----------------------|
| Function Code | 1 Byte | 0x90 |
| Exception Code | 1 Bytes | 01 or 02 or 03 or 04 |

| Address Register | Description |
|------------------|-------------|
| 0x0000 | DAC 1 MSB |
| 0x0001 | DAC 1 LSB |
| 0x0002 | DAC 2 MSB |
| 0x0003 | DAC 2 LSB |
| ... | ... |
| ... | ... |
| 0x0050 | ADC Gain 1 |
| 0x0051 | ADC Gain 2 |
| 0x0052 | ADC Gain 3 |
| 0x0053 | ADC Gain 4 |
| 0x0054 | ADC Gain 5 |
| 0x0055 | ADC Gain 6 |
| 0x0056 | ADC Gain 7 |
| 0x0057 | ADC Gain 8 |
| 0x0058 | ADC Range 1 |
| 0x0059 | ADC Range 2 |
| 0x005A | ADC Range 3 |
| 0x005B | ADC Range 4 |
| 0x005C | ADC Range 5 |
| 0x005D | ADC Range 6 |
| 0x005E | ADC Range 7 |
| 0x005F | ADC Range 8 |

Table B-4: Output Register Map

0x03 Read Holding Registers (Query function)

This function is used to query specific holding registers.

The register data in the response message are packed as four bytes per register, with the binary contents right justified within each byte. For each register, the first byte contains the high order bits and the fourth contains the low order bits.

Request

| | | |
|------------------------------------|---------|------------------|
| Function Code | 1 Byte | 0x03 |
| Starting Address | 2 Bytes | 0x0000 to 0xFFFF |
| Quantity of Input Registers | 2 Bytes | 0x0001 to 0x000f |

Response

| | | |
|------------------------|-------------|-------|
| Function Code | 1 Byte | 0x03 |
| Byte Count | 1 Bytes | 4 x N |
| Input Registers | N x 4 Bytes | |

N = Quantity of Input Registers.

Error

| | | |
|-----------------------|---------|----------------------|
| Function Code | 1 Byte | 0x83 |
| Exception Code | 1 Bytes | 01 or 02 or 03 or 04 |

| Address Register | Description |
|------------------|-------------------------|
| 0x000A | Device Type |
| 0x000B | Device Revision |
| 0x000C | Manufactured Year |
| 0x000D | Manufactured Month |
| 0x000E | Manufactured Day |
| 0x000F | Serial Number |
| 0x0017 | Firmware Major Revision |
| 0x0018 | Firmware Minor Revision |
| 0x0019 | Firmware Build Revision |
| 0x0260 | Get Network IP Address |
| 0x0261 | Get Network Mask |
| 0x0262 | Get Network Gate Way |
| 0x0265 | MAC Company ID |
| 0x0266 | MAC Extension ID |

Table B-5: Holding Register Map

0x08 Diagnostics

MODBUS function code 08 provides a series of tests for checking the communication system between a client (Master) device and a server (Slave), or for checking various internal error conditions within a server.

The function uses a two-byte sub-function code field in the query to define the type of test to be performed. The server echoes both the function code and sub-function code in a normal response. Some of the diagnostics cause data to be returned from the remote device in the data field of a normal response.

In general, issuing a diagnostic function to a remote device does not affect the running of the user program in the remote device. User logic, like discrete and registers, is not accessed by the diagnostics. Certain functions can optionally reset error counters in the remote device. A server device can, however, be forced into 'Listen Only Mode' in which it will monitor the messages on the communications system but not respond to them. This can affect the outcome of your application program if it depends upon any further exchange of data with the remote device. Generally, the mode is forced to remove a malfunctioning remote device from the communications system.

The following diagnostic functions are dedicated to serial line devices.

Request

| | | |
|----------------------|-------------|------|
| Function Code | 1 Byte | 0x08 |
| Sub-function | 2 Bytes | |
| Data | N X 2 Bytes | |

Response

| | | |
|----------------------|-------------|------|
| Function Code | 1 Byte | 0x08 |
| Sub-function | 2 Bytes | |
| Data | N x 2 Bytes | |

Error

| | | |
|-----------------------|---------|----------------------|
| Function Code | 1 Byte | 0x88 |
| Exception Code | 1 Bytes | 01 or 02 or 03 or 04 |

Here the list of **sub-function** codes supported by the serial line devices.

| Code | Description |
|--------|--|
| 0x0000 | Echo Query Data |
| 0x0001 | Reset Communications Option |
| 0x0004 | Force Listen Only Mode |
| 0x000A | Clear Counters and Diagnostic Register |
| 0x000C | Return Bus Communication Error Count |
| 0x000D | Return Bus Exception Count |
| 0x000E | Return Slave Message Count |

Table B-6: Diagnostic Sub-functions

0x0B Get Communication Event Count (Serial Line Only)

This function code is used to get a status word and an event count from the remote device's communication event counter.

By fetching the current count before and after a series of messages, a client can determine whether the messages were handled normally by the remote device.

The device's event counter is incremented once for each successful message completion. It is not incremented for exception responses, poll commands, or fetch event counter commands.

The event counter can be reset by means of the Diagnostics function (code 08), with a sub-function of Restart Communications Option (code 00 01) or Clear Counters and Diagnostic Register (code 00 0A).

The normal response contains a two-byte status word, and a two-byte event count. The status word will be all ones (FF FF hex) if a previously-issued program command is still being processed by the remote device (a busy condition exists). Otherwise, the status word will be all zeros.

Request

| | | |
|----------------------|--------|------|
| Function Code | 1 Byte | 0x0B |
|----------------------|--------|------|

Response

| | | |
|----------------------|---------|------------------|
| Function Code | 1 Byte | 0x0B |
| Status | 2 Bytes | 0x0000 to 0xFFFF |
| Event Count | 2 Bytes | 0x0000 to 0xFFFF |

Error

| | | |
|-----------------------|---------|----------------------|
| Function Code | 1 Byte | 0x8B |
| Exception Code | 1 Bytes | 01 or 02 or 03 or 04 |

General Description

Protocol Description

The MODBUS protocol defines a simple protocol data unit (**PDU**) independent of the underlying communication layers. The mapping of MODBUS protocol on specific buses or network can introduce some additional fields on the application data unit (**ADU**).

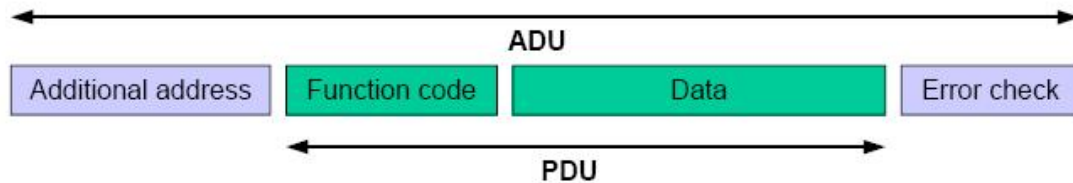


Figure B-1: General MODBUS frame

The MODBUS application data unit is built by the client that initiates a MODBUS transaction. The function indicates to the server what kind of action to perform. The MODBUS application protocol establishes the format of a request initiated by a client.

The size of the MODBUS PDU is limited by the size constraint inherited from the first MODBUS implementation on Serial Line network (max. RS485 ADU = 256 bytes).

Therefore:

MODBUS PDU for serial line communication = 256 - Server address (1 byte) - CRC (2 bytes) = 253 bytes.

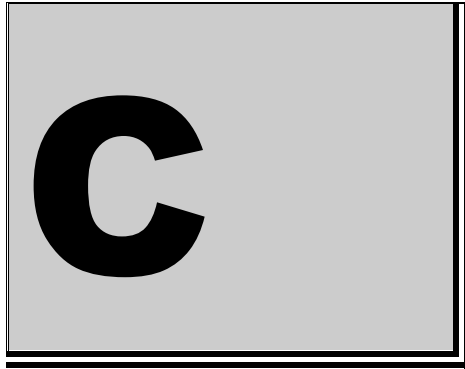
Consequently:

RS232 / RS485 ADU = 253 bytes + Server address (1 byte) + CRC (2 bytes) = 256 bytes.

TCP MODBUS ADU = 253 bytes + MBAP (7 bytes) = 260 bytes.

Data Encoding

MODBUS uses a 'big-Endian' representation for addresses and data items. This means that when a numerical quantity larger than a single byte is transmitted, the most significant byte is sent first.



C. Troubleshooting & Maintenance

Troubleshooting

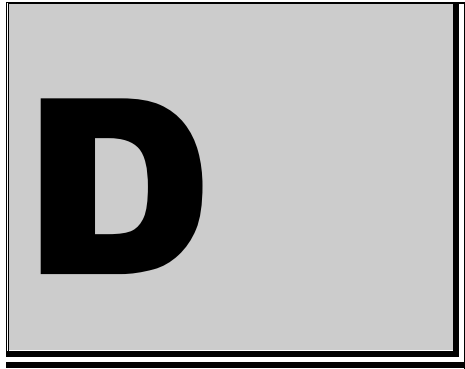
Device Unavailable on Serial Connection

- Make sure that the Slave address is correct, as selected.
- Check that correct serial port number is correct and correctly configured.
- Check that your Baud Rate is correct, as selected.
- Reset device after any changes was made to the device configuration settings.
- Reset device by disconnecting the power.

Maintenance

Calibration

The device should be returned every two years for recalibration. A service fee will be charged for all work done as well as for any parts that needs replacement. Courier and / or delivery cost will also be for the clients cost.



D. Ordering Information

For ordering information please contact Eagle Technology directly or visit our website.

Telephone +27 (021) 423 4943

Fax +27 (021) 424 4637

E-Mail eagle@eagle.co.za

Website <http://www.eagledaq.com>