///AMASSER ECDIO

Embedded Control Digital I/O Module
User's Reference Manual

Updated: February 5, 2004
http://www.amassdata.com

Table of Contents

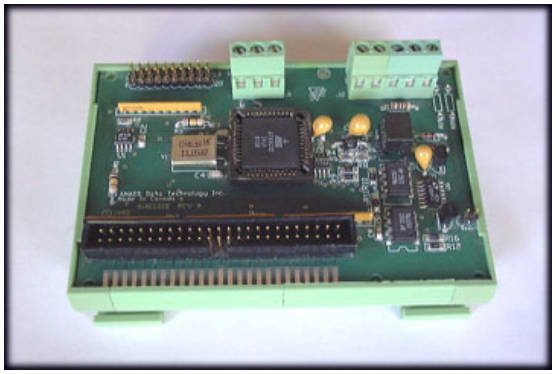
AMASS Data Embedded Control
Digital I/O Module
Pliant Technology Specialists

Pliant: readily yielding to influence

1.0 AMASSER ECDIO Digital I/O Module

The 2500 Vrms isolated ECDIO Digital I/O Module is a compact DIN Rail mountable, full function digital input/output (I/O) subsystem that is compatible with the AMASS RS485 Multi-drop Embedded Control Network (ECN). This module provides firmware based intelligent digital I/O functions for its 24 I/O lines.

1.1 AMASSER ECDIO Overview



The ///AMASSER ECDIO Digital I/O Module provides a standalone, full function digital input/output subsystem which is fully compatible with the ECN network. The unit provides three 8-bit parallel I/O ports. One port dedicated output, the second port is dedicated input and the third port is independently programmable as either an input or output port. Three ALS645 octal buffer devices are provided to buffer the I/O ports and provide +/- 24 ma. drive capacity. This is sufficient to directly interface to standard opto-module racks. Connection to the I/O points is via a 50-pin ribbon connector, which can be connected directly to industry standard digital I/O termination panels.

The communications port is a 2500Vrms optically isolated RS485 multi-drop interface and can be configured for half of full duplex operation. Network termination resistor sockets are provided on the module. An on board DC-DC converter provides the RS485 interface circuitry with 2500 Vrms isolated +5V dc power.

1.2 Firmware Support

The ECDIO module has been designed to provide the user with intelligent Digital I/O functions that are easily executed over a robust & reliable network from a COMM port on your industrial PC. The firmware provides input, output, network and specialized functions which are detailed in the *AMASSER ECDIO Command Set* section.

Input functions include Read Bits, Read Byte, Read Timer Registers & Read Periodic On/Off Registers. Output functions include Write Byte, Set Timer Register & Set Periodic On/Off Registers. The basic clock tick by which the firmware operates is 10 milliseconds. Output timers are 16 bit and provide 10 msec. to 655.35 second interval timing.

Also provided is firmware that provides a watchdog function for the communications port. It is user programmable and allows an orderly shutdown of I/O in case of communications failure. Also provided are two outputs that can be programmed to provide periodic maintenance until communications have been restored.

2.0 ECN Protocol

2.1 Physical

The Embedded Control Network (ECN) is configured as an RS-485 based multi-drop environment supporting up to 32 modules over a distance of 4000 feet on single or dual twisted pair cable. The network operates at 9.6 or 19.2 Kbaud rates with the ability to increase these rates up to 115.2 Kbaud. An additional pair is also specified for the routing of 24Volt or 5Volt DC power. An important feature of the AMASSER ECxxx line of products is that each module provides 2500 Vrms isolation between the network and the I/O functions being provided.

2.2 Protocol

The Embedded Control Network protocol is configured as a master slave environment supporting a simple & reliable multi-node ASCII based command - response message system. Message strings begin with an address byte followed by the message body, followed by a 8 bit hexadecimal checksum and terminated with a carriage return byte. The protocol allows broadcast messages, module identification, and individual module firmware reset ability.

2.3 Command Messages

In order to communicate with the AMASSER ECDIO Unit, the host controller sends a message to the module. The first byte of the message string is an "address" byte, which consists of an ASCII character which represents the AMASSER ECDIO's node address. In order to support up to 32 remote node addresses on a network, these addresses range from ASCII "0" (30H) to ASCII "O" (4FH). Following the Node address byte is a one or two byte command which allow various I/O operations to be initiated. The commands available to the host machine for the ECDIO module are listed below. Following the command byte is the body of the message which contains the parameters required for the execution of the command. Following the body of the message are two checksum bytes which are the ASCII hex value of the complement of the 8 bit checksum of all the bytes in the message up to and not including the two checksum bytes. The last character of the message is always "CR" (0DH). This "CR" character must never be used any where else in the message string. The host controller must turn off its transmitter and enable its receiver within 1 millisec after sending the "CR" in half duplex operation mode.

2.4 Response Messages

The AMASSER ECDIO responds to the host computer by sending a response message to the host computer. The first byte of the response message string is an "address" byte, which consists of an ASCII character which represents the AMASSER ECDIO's node address. The response address ranges from ASCII "0" (30H) to ASCII "O" (4FH) and is set by jumpers located on the ECDIO module. Following the address byte is the body of the response message which contains the parameters (if any) requested by the host computer. Following the body of the message are two checksum bytes which are the ASCII hex value of the complement of the 8 bit checksum of all the bytes in the message up to and not including the two checksum bytes. The last character of the response message is always "CR" (0DH). The AMASSER ECDIO turns off its transmitter within 1 millisec after sending the "CR" in half duplex operation mode.

2.5 Message Length

A string of bytes or characters will be at least four bytes in length and not more than 36 bytes in length.

2.6 Byte Frame Format

All transmission will be asynchronous having 1 start bit, 8 data bits, no parity and 1 stop bit.

2.7 Baud Rate

19,200 - installed or 9600 - not Jumper Block Position 1-2 (see section *Jumper Block Configuration*)

3.0 AMASSER ECDIO Command Set

3.1 Command Symbol Definitions:

a : address byte - Jumper Block Positions 3-4 to 11-12 (see section *Jumper Block Configuration*). No jumpers installed equals address 0 where 30H < a < 4FH for the host commands and for AMASSER ECDIO Responses.

xxx.xx : 16 bit decimal ASCII representation of time in seconds i.e. 256.45 or 0.01 seconds

b : Output Identifier i.e Output 2, b = 2

+ : Delimiter character - used to separate parameters i.e. +2+256.45+.01+

<xx>h : 2 ASCII Hex bytes representing 8 bit data where '0'< x > '9' or 'a' < x > 'f'

+**<b0><b1><b2>.....<b7> :** Binary Format Data i.e. +11100100B = e4H or 228

cksum <x><x> : the complement of the ASCII hex value of the 8 bit checksum of all the bytes in the message not including the two checksum bytes and the carriage return byte. Ex: Message '0!ae' where 'ae' is the hex value of the complement of the checksum of the ASCII characters '0' and '!'. ASCII '0' is 30H and ASCII '!' is 21H which when added and "complemented" results in 'ae' which is then appended to the command.

Command bytes : one or two byte ASCII character command. Ex.: R6

Identification String : "10AMASSDataECDIO xxx" where xxx = Revision Level

3.2 ECN Set Commands: Set ECDIO Parameters

3.2.1 Write Byte to Output Port S0

- command: aS0<xxh><cksum><cr>
- response: a<cksum><cr>

This command is used to set all output ports at once. Writing to the output ports using this command overrides all other settings by the user. For instance, all outputs can be turned on simultaneously with this single command.

Example:

The active host address is 0 and you wish to turn on outputs 7, 3 and 0. This corresponds to 10001001 binary which is 89 hex (see table below). Therefore the following command would be required:

Command: 0S089bd
Response: 0cf

output 7	output 6	output 5	output 4	output 3	output 2	output 1	output 0
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

3.2.2 Write Count Register S3

- command: aS3+b+<xxx.xx>+<cksum><cr>
- response: a<cksum><cr>

This command allows you to set the Count register of the current output bit. The count register is a timer which allows the user to program the on-time of any given output bit. This feature is to be used as a one-shot facility since the on-time is not to be repeated.

Example:

The active host address is 0, the current output bit is 3 and you wish to turn it on for 33.5 seconds. The proper command would be:

Command: 0S3+3+33.5+cc
Response: 0cf

3.2.3 Write Periodic Registers S4

- command: aS4+b+<onx.xx>+<off.xx>+<cksum><cr>
- response: a<cksum><cr>

This command allows the user to write to the Periodic registers of the current output. The periodic registers enable one to set the on-time and off-time of output bits that are to be run cyclically. Their value are in fact reset values for the count register which in turn controls the output.

Example:

The active host address is 2, the current output bit is 4 and you wish to set an on-time of 3.45s and an off-time of 5.43s. The proper command would be:

Command: 2S4+4+3.45+5.43+d2
Response: 2cd

Output 4 would immediately begin the cycle: on for 3.45 s, off for 5.43s, on for 3.45s and so on. This would persist unless the Failsafe time elapses with no commands being sent from the host PC at which time all registers would be automatically reset to zero (see *Write FailSafe Register S5*).

3.2.4 Write FailSafe Register S5 (setting Failsafe Register=0 disables this facility)

- command: aS5+<xxx.xx>+<cksum><cr>
- response: a<cksum><cr>

This command allows the user to set the Failsafe register. The Failsafe register is in fact a communications Watchdog timer and as such is an automatic reset facility in the event of host inactivity. The user defines a time interval of "communications inactivity" after which all registers of all output bits are reinitialized to some value. This value is zero on standard units but can be configured to some other value if the customer so wishes. The selection of a suitable Failsafe time depends on the process that is being controlled, safety issues, etc… Due to this built-in intelligence the ECDIO will intervene in the event of loss of host management .

Example:

The active host address is 4 and you wish to have a Failsafe time of 45.0 seconds. The proper command would be:

Command: 4S5+45.00+f6

A successful transmission would be followed by :

Response: 4cb

Following this exchange therefore the host PC must contact the ECDIO at least every 45.0 seconds otherwise all registers are reset to zero (typically).

Note that this facility is disabled when the Failsafe register is set to zero. The longest period that it may be set at is 655.35 seconds.

3.2.5 Write AutoFeed Registers S6

→ Initiates upon ECN Comm Failsafe operation where 0=autofeed off

- command: aS6+<onxxx.xx>+<offxxx.xx>+<cksum><cr>
- response: a<cksum><cr>

This command is available when the ECDIO is configured for aluminum smelter applications. The Autofeed registers are in fact dedicated Periodic registers and are set in the same manner. These registers come into effect when all other registers have been reset following a period of host inactivity (see *Write FailSafe Register*).

Command: 2R645
Response: 210.00120.00bd

This indicates that the Autofeed registers are to be switched on for 10.00 s, then off for 120.00 s and so on.

3.3 ECN Read Commands: Read Bits & Bytes & Timer Registers

3.3.1 Read Hexadecimal Input Bytes R0

- command: aR0<cksum><cr>
- response: a<byte0><byte1><cksum><cr>

This command instructs the ECDIO to return the values of its input bytes at ports 2 (byte0) and 3 (byte1). Note that it reads port 3 regardless of whether it is configured as output or input.

Example:

The active host address is 2 and you wish to read the input bytes. The proper command would be:

Command: 2R04b
Response: 2ffHffHa5

3.3.2 Read Binary Input Bytes R1

- command: aR1<cksum><cr>
- response: a<+b0><+b1><+b2>.....<+b15><cksum><cr>

Just like the 'R0' command but presents the data in binary format.

Example:

The active host address is 2 and you wish to read the input bytes. The proper command would be:

Command: 2R14a
Response: 2+11111111B+11111111Be4

3.3.3 Get Shift Register Data R2

- command: aR2<cksum><cr>
- response: a<SRData><cksum><cr>

The shift register is the 8 bit binary representation of the jumper block configuration (see *Jumper Block Configuration*). Upon start-up the jumper block configuration of the first 16 pins is read and stored in the shift register. The 'R2' command allows the user access this data.

Example:

The active host address is 2 and jumpers 5-6, 7-8, 11-12 and 15-16 are installed. The exchange would be as follows:

Command: 2R249
Response: 2acHc1

The shift register is therefore 'ac' hexadecimal. To understand this consider the following chart:

jumper 20-19	jumper 18-17	jumper 16-15	jumper 14-13	jumper 12-11	jumper 10-9	jumper 8-7	jumper 6-5	jumper 4-3	jumper 2-1
none	none	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

Therefore the binary representation of our jumper block configuration is 10101100 which is 'ac' hexadecimal.

3.3.4 Read Count Register R3

- command: aR3+b+<cksum><cr>
- response: a+<xxx.xx>+<cksum><cr>

This command allows you to read the Count register of the current output bit. The count register is a timer which allows the user to program the on-time of any given output bit. This feature is to be used as a one-shot facility since the on-time is not periodic.

Example:

The active host address is 2, the current output is 0 and you wish to read its Count register.

Command: 2R3+0+c2
Response: 20.000f

Which indicates that output 0 is currently off. To turn on an output for a specified amount of time you must set the Count register using the 'S3' command.

3.3.5 Read Periodic Registers R4

- command: aR4+b+<cksum><cr>
- response: a+<onx.xx>+<off.xx>+<cksum><cr>

This command allows the user to read the Periodic registers of the current output. The periodic registers enable one to set the on-time and off-time of output bits that are to be run cyclically. Their value are in fact reset values for the count register.

Example:

The active host address is 2, the current output is 0 and you wish to read its Periodic registers.

Command: 2R4+0+c1
Response: 20.00+0.0026

Which indicates that output 0 is not currently operating cyclically. To set the Periodic registers use command 'S4'.

3.3.6 Read FailSafe Register R5

- command: aR5<cksum><cr> 0=timer off
- response: a+<xxx.xx>+<cksum><cr>

This command allows the user to read the Failsafe register. The Failsafe register is in fact a communications Watchdog timer and as such is an automatic reset facility in the event of host inactivity. The user defines a time interval of "communications inactivity" after which all registers of all output bits are reinitialized to some value. This value is zero on standard units but can be configured to some other value if the customer so wishes. The selection of a suitable Failsafe time depends on the process that is being controlled, safety issues, etc… Due to this built-in intelligence the ECDIO will intervene in the event of loss of host management over the prescribed time period.

Example:

The active host address is 2 and you wish to read the value of the Failsafe register.

Command: 2R546
Response: 230.00dc

Which indicates that the Failsafe time is set at 30.00 seconds. Therefore, if a period of 30.00 s elapses where no commands have been sent all registers would automatically be reset to zero.

To set the Failsafe register use the 'S5' command.

Note that a Failsafe time of 0.00 disables this automatic reset facility.

3.3.7 Read AutoFeed Registers R6

- command: aR6<cksum><cr>
- response: a+<onxxx.xx>+<offxxx.xx>+<cksum><cr>

This command is available when the ECDIO is configured for aluminum smelter applications. The Autofeed registers are in fact dedicated Periodic registers and are set in a likewise manner. The 'R6' command allows one to set their on-time and off-time. These registers come into effect when all other registers have been reset following a period of host inactivity (see *Read FailSafe Register*).

Command: 2R645
Response: 210.00120.00bd

This indicates that the Autofeed registers are to be switched on for 10.00 s, then off for 120.00 s and so on.

3.4 ECN Network Commands:

3.4.1 Change Node Address: "A" Command

- command: aAc<cksum><cr> where c = new node address
- response: c<cksum><cr>

This command is used to change the node address of the ECDIO. Successful communication between the host and the ECDIO can only take place if the host and ECDIO node addresses are the same. Therefore the host address must be updated following the use of this command (see *ECDIO Operation*).

Upon start-up the default node address is set by the jumper block configuration (see *Jumper Block Configuration*) so that the user must adjust the host address accordingly.

Example:

The active host address is 2 and you wish to change it to B.

Command: 2AB4a
Response: Bbd

The host address must now be changed to B in order to communicate with the ECDIO.

3.4.2 Send ECN Identification String I

- command: aI<cksum><cr>
- response: a<identification string><cksum><cr>

This command is used to obtain identification of the ECDIO.

3.4.3 Send acknowledgment !

- command: a!<cksum><cr>
- response: a<cksum><cr>

This command allows the user to confirm the communications link with the ECDIO without having to send a Read or Set command.

Example:

The active host address is 2 and you wish to confirm your link with the ECDIO.
Command: 2!ac
Response: 2cd

3.4.4 Reset The Module

- command: a#<cksum><cr>
- response: none Wait 10 milliseconds, then send Acknowledgment

This command allows remote resetting of the ECDIO.

4.0 Host Software Operation

The ECDIO user-interface is menu-driven host software that enables the user to send and receive the proper commands that are described above. To initiate the program type "ECDIO". Possible command line options include: "ECDIO -*s com#*" where # represents the port number which is being used. When the default port is being used the latter is unnecessary. Another command line option is to use "ECDIO -a #" to initiate the program with host address # (host addresses are actually numbered from 0 to 9 and A to V).

The main menu allows the user to establish contact with the ECDIO unit and as such contains the ECN network commands. Once the link is established one can begin to access and write to the ECDIO using the Read and Set commands in the sub-menus.

The main menu is as follows:

[Addr=0] [Output=0] Command? (Command line prompt)

+ - Increment host address (Main menu)
- - Decrement host address
n - Change node address
a - Request acknowledgment
i - Read sensor identification
r - READ current value of ECDIO parameters
s - SET ECDIO parameters
m - Send manual command
c - Toggle continuous mode
q - quit

Response: 0 (Infoline message)

Sending: '0!ae' (Communications section)
Received: '0bf'

The *Change node address* item sets the ECDIO node address to something other than the default address set by the jumper block (see *Jumper Block Configuration)* . To reestablish communication after such a change the host address (the current value of which is always shown in the command line at the top of the screen. Ex: 0) must be set to the new ECDIO node address by using '+' and '-' . A successful link occurs when the Infoline message displays the host address (Ex: 0), otherwise it displays "No response from sensor". One can always confirm that the current host address is active by using the *acknowledge* or *identify* items.

Item 'm' allows the user to type any command which appears in the *AMASSER ECDIO Command Set* section. This is done at the *command line* at the following prompt: "Manual cmd:". When entering manual commands however, the host address and checksum need not be entered since these are added by the software before being sent to the PSE-SDI. For instance, the command aR4+b+<checksum> should be entered as R4+b+.

The sub-menus allow the user to send the Read and Set commands described above. The output which is the target of these commands is displayed in the command line and controlled from the sub-menus as shown below for the Read commands. To return to the main menu press the *ESC* key.

[Addr=0] [Output=0] Command? (Command line prompt)

+ - Increment output target (READ sub-menu)
- - Decrement output target
0 - Read hexadecimal input bytes
1 - Read binary input bytes
2 - Read shift register data
3 - Read count register
4 - Read periodic register
5 - Read failsafe register
6 - Read autofeed register
c - Toggle continuous mode

Response: +1.00+2.00 (Infoline message)

Sending: '0R4+0+c3' (Communications section)
Received: '0+1.00+2.00+cf'

This is a typical response when selecting item '4' to read the periodic register of the current output, i.e. 0. In this instance the on-time is 1.00 second and the off-time is 2.00 seconds as set in the *Set* sub-menu since default values are always zero.

Press 'c' to enter the *continuous mode*. In this mode any subsequent readings will be continuously updated at the Infoline. For instance, if one views the count register in the example above as it cycles through the preset values of the periodic registers one would see the following : 1.00 to 0, 2.00 to 0, 1.00 to 0, and so on. Any *Read* menu item may be selected while in continuous mode. Manual commands may also be viewed in continuous mode, but from the main menu. To do this press 'm' so that the last manual command that was entered is continuously sent. Note that this is usually done for "read" commands only since one would not normally "write" continuously to a sensor such as the PSE-SDI. When in the *continuous mode* the command line reads: [Addr=0] Continuous command ?.

The SET sub-menu is as follows:

Enter hexadecimal byte value (ex:ff) : (Command line prompt)

+ - Increment output target (SET sub-menu)
- - Decrement output target
0 - Write byte to output port
3 - Write count register
4 - Write periodic register
5 - Write failsafe register
6 - Write autofeed register

This is the prompt that appears when item '0' is selected. The command line editor uses a buffered input which allows the user to edit what is typed at the prompt before it is entered. It also allows you to recall previous entries by using the up and down arrow keys.

4.1 Host Software Protocol Drivers

The host is a C program that provides the interface described above and operates the ECDIO according to ECN protocol (see *ECN Protocol*). Note that the commands that appear in the *AMASSER ECDIO Command Set* section are the final form that are sent to and from the ECDIO. When the function doSDIMsg() (see below) is called the command message is typically something like "R3+b+". The host address is then inserted at the beginning of the string. The doSDIMsg() function call is:

doSDIMsg(sdiAddr, Message, sdiRsp);

where int sdiAddr; /*A global variable that is equal to the current host address. Its value is changed with the upaddressCmd() and the downaddressCmd() functions. */

char *Message; /* A pointer to a string such as: "R3+b+".

unsigned char sdiRsp[256];

The complement of the checksum is then calculated and appended in sdiSend() so that the final form is typically:

aR3+b+<cksum><cr> as it appears in the command set.

```
char asciiconv [36] = { "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ" };
doSDIMsg( a, s, d )
int a;
char *s, *d;
{
        char buf[ 128 ];
        char tmp[ 128 ];

        if( a == 32 ) {
                sprintf( tmp, "*%s", s );
        } else {
                sprintf( tmp, "%c%s", asciiconv [a], s );  /* Precede command message with host address*/
        }
        sdiSend(tmp);
        sprintf( buf, "Sending: '%s'", tmp );
        osScrollUp( MsgLine );      /*Prints command message to the "Communications section" of */
        osPutLine( 24, buf );       /*          the screen (see Host  Software Operation section). */
        if( sdiRecv( d ) ) {
                noMagics( tmp, d );
                sprintf( buf, "Received: '%s'", tmp );
                osScrollUp( MsgLine );                  /*Prints response message to the screen.*/
                osPutLine( 24, buf );
                return( 1 );
        } else {
                osScrollUp( MsgLine );
                osPutLine( 24, "Receiver timeout" );
                return( 0 );
        }
}
```

```
sdiPutChar( c )     /* This function is responsible for outgoing messages. Processes one character at a time.
*/
int c;
{
        while( ( inp( addr+5 ) & 0x40 ) == 0 ) ;
        outp( addr, c );
}

char hexconv [16] = { "0123456789abcdef" };

void
sdiSend( s )
char *s;
{
        unsigned char t;
        unsigned char checksum[2];

        while( sdiCharWaiting() ) sdiGetChar();
        for( t=0; *s; t+=*s++ ) {
                sdiPutChar( *s );
        }
        t = 0xff - t;               /*Determine the complement of the checksum*/
        sdiPutChar (hexconv [t >> 4]);                  /*Append the complement of the checksum  */
        sdiPutChar (hexconv [t & 0x0f]);                /* and the carriage return character to the      */
        sdiPutChar (13);                                /*         outgoing command message.           */
        *(s) = hexconv[t >> 4];       /*Append the complement of the checksum to the command  */
        *(s + 1) = hexconv[t & 0x0f];           /* message to be displayed on the screen. */
        *(s + 2) = '\0';
}
```

```
int
sdiRecv( buf )
char *buf;
{
        int c, l;
        unsigned char t, new;
        int kludge;

        for( t=l=0; l < 32; ++l ) {
                c = sdiGetChar ();
                if (c == -1) return 0;
                if (c == 13) {
                        if (l != 0) {
                                *buf = 0;
                                t -= buf [-1];
                                t -= buf [-2];
                                sscanf (&buf [-2], "%x", &kludge);
                                new = kludge;
                                t = 0xff - t;
                                return (t == new);
                        } else {
                                return 0;
                        }
                }
                t += (*buf++ = c);
        }
        return 0;
}
```

5.0 ECDIO HARDWARE Configuration:

The AMASSER ECDIO module has three 8 bit ports, a total of 24 digital I/O. Port 1 (numbered 0 to 7) is a dedicated output port, Port 2 (#8 to #15)is a dedicated input port and port 3 (#16 to #23) is user-selectable via jumper - (jumper installed = INPUT). Port 3 if selected as an output port does not have the periodic mode of operation but does maintain the one-shot facilities (see sections 3.2.2 and 3.2.3).

When the ECDIO is configured for aluminum smelter applications output 2 of port 1 is dedicated as the autofeed register (see section 3.2.5) and output 7 as the 1 second clock.

5.1 Jumper Block Configuration:

	Uninstalled	Installed
1-2 Baud Rate	9600	19200
3-4 Bit 0 Node Address	0	1
5-6 Bit 1 Node Address	0	1
7-8 Bit 2 Node Address	0	1
9-10 Bit 3 Node Address	0	1
11-12 Bit 4 Node Address	0	1
13-14 Not Currently Used		
15-16 Port 3 I/O Configuration	Output	Input
17-18 TD Watchdog Timer	1.2sec	150 ms.
19-20 Hardware Reset	Install & Remove	

Example:
You are using a baud rate of 9600, a node address of 5, TD Watchdog Timer is 1.2 sec and port 3 is configured for input. The following jumpers would have to be installed: 3-4, 7-8 and 15-16. Jumper 15-16 is installed because of the configuration of port 3 whereas jumpers 3-4 and 7-8 are installed to represent node address 5 in binary. This is because the binary representation of 5 is 00101, where jumpers 3-4 and 11-12 are the least and most significant bits respectively. As mentioned in the *ECN Protocol* section the ECDIO supports 32 node addresses.

Note however that the jumper configuration for node address is simply the default value when the ECDIO is reset. The address can then be changed by executing the *Change Address Command* by typing ‘n’ from the main menu (see *Host Software Operation*).

6.0 Installation

All AMASSER ECN products, including the ECDIO, are DIN rail mountable. The mounting rails may be NS 15, NS 35/7,5, NS 35/15 or NS 32.

The ECDIO is powered with 5 V. If only 24 V is available you may order the AMASSER DC2405/15W 5V Power Supply which is installable on the same mount.

To make the proper connections to the ECDIO refer to the following section and to the *Jumper Block Configuration* section.

6.1 Connectors

1 50 Pin Ribbon Connector
1 5 screw contact terminal (RS485 Connections to host PC)
1 3 screw contact terminal (Power Supply Connections)

50 Pin Ribbon Connector

I/O #23 ---------- 1	2 ---------- GND
I/O #22 ---------- 3	4 ---------- GND
I/O #21 ---------- 5	6 ---------- GND
I/O #20 ---------- 7	8 ---------- GND
I/O #19 ---------- 9	10 ---------- GND
I/O #18 ---------- 11	12 ---------- GND
I/O #17 ---------- 13	14 ---------- GND
I/O #16 ---------- 15	16 ---------- GND
Input #15 ---------- 17	18 ---------- GND
Input #14 ---------- 19	20 ---------- GND
Input #13 ---------- 21	22 ---------- GND
Input #12 ---------- 23	24 ---------- GND
Input #11---------- 25	26 ---------- GND
Input #10 ---------- 27	28 ---------- GND
Input #9 ---------- 29	30 ---------- GND
Input #8 ---------- 31	32 ---------- GND
Output #7 ---------- 33	34 ---------- GND
Output #6 ---------- 35	36 ---------- GND
Output #5 ---------- 37	38 ---------- GND
Output #4 ---------- 39	40 ---------- GND
Output #3 ---------- 41	42 ---------- GND
Output #2 ---------- 43	44 ---------- GND
Output #1 ---------- 45	46 ---------- GND
Output #0 ---------- 47	48 ---------- GND
Vcc (5 V) --------- 49	50 ---------- GND

5 screw contact terminal
(RS485 Connections)

RGND ---------- 1
RXD ---------- 2
RXD* ---------- 3
TXD ---------- 4
TXD* ---------- 5

If operating in half duplex mode RXD* and TXD* are interconnected as are RXD and TXD.

3 screw contact terminal
(Power Supply Connections)

Vcc (5 V) --------- 1
GND ---------- 2
EARTH GND ---- 3

7.0 Specifications

Processor - Atmel 89C52 @ 11.0592MHz.
Word Size - 8 bit data - 8 bit instruction
Memory - 89C52, 8 Kbytes FLASH
256 bytes RAM

Connectors
ECNet - 10 pin ribbon header type
Compatible connectors:
Flat cable - 3M 3473-6010,TB
Ansley 609-100M or equivalent
DIN connector - 96/64 PC mount
Compatible connectors:
Flat cable - GW Elco 00-8259-096-124 or equivalent
Discrete Wire - GW Elco 60-8257-3017 or equal
DIGITAL I/O - 50 pin pcb mount: Molex 15-25-5501
SBX Connector - As per Intel iSBX Expansion Bus Specification.
Digital Inputs
Ports 0,3: present 1 LSTTL load
Ports 1,2,4,5: VIH = 2.0 VDC min
VIL = 0.8 VDC max.
(no pullups)
Leakage (IIL) = 1uA max.
Digital Outputs
Ports 0,3: installed 74LS225 buffer drives 60 LSTTL loads
VOH = 2.0 VDC min
VOL = 0.8 VDC max
IOH = -15mA max
IOL = 24mA min.

Ports 1,2,4,5: VOH = 3.0 VDC min
VOL = 0.4 VDC max
IOH = -2.5 mA max
IOL = 2.5 mA min
leakage (IOL) = 10 uA max

Power Requirements
+ 5V +/- 5% @ 250mA max.

Physical Characteristics
Height - 13.77 mm. (0.542in.)
Width - 100.0 mm (3.93 in.)
Depth - 200.00 mm (8.65 in.)
Weight - 0.175 Kg (0.5 lb.)
Mounting - Standard single height, double depth Eurocard, with IEC 603-2-IEC-C064-M connector.

Environmental Characteristics
Temperature - At 61.5 Linear meters/min air velocity: Operating -40 to +85 C
Storage - -55 to +105 C
Humidity: <= 90% non-condensing

The above information is believed to be true at the time of printing. AMASS Data reserves the right to modify specifications without notice.

AMASS Data Technologies Inc., 34 Chemin Helene, Val des Monts, Quebec, CANADA J8N 2L7
TEL: 1-819-457-4926 (in USA 1-315-393-3793) FAX: 1-819-457-9802

APPENDIX A : ECDIO Command Set: Quick Reference

ECN Set Commands: Set ECDIO Parameters

1. Write Byte to Output Port S0
   - command: aS0<xxh><cksum><cr>
   - response: a<cksum><cr>
2. Write Count Register S3
   - command: aS3+b+<xxx.xx>+<cksum><cr>
   - response: a<cksum><cr>
3. Write Periodic Registers S4
   - command: aS4+b+<onx.xx>+<off.xx>+<cksum><cr>
   - response: a<cksum><cr>
4. Write FailSafe Register S5 (setting Failsafe Register=0 disables autofeed)
   - command: aS5+<xxx.xx>+<cksum><cr>
   - response: a<cksum><cr>
5. Write AutoFeed Registers S6 - Initiates upon ECN Comm Failsafe operation where 0=autofeed off
   - command: aS6+<xxx.xx>+<xxx.xx>+<cksum><cr>
   - response: a<cksum><cr>

ECN Read Commands: Read Bits & Bytes & Timer Registers

1. Read Hexadecimal Input Bytes R0
   - command: aR0<cksum><cr>
   - response: a<byte0><byte1><cksum><cr>
2. Read Binary Input Bytes R1
   - command: aR1<cksum><cr>
   - response: a<+b0><+b1><+b2>.....<+b15><cksum><cr>
3. Get Shift Register Data R2
   - command: aR2<cksum><cr>
   - response: a<SRData><cksum><cr>
4. Read Count Register R3
   - command: aR3+b+<cksum><cr>
   - response: a+<xxx.xx>+<cksum><cr>
5. Read Periodic Registers R4
   - command: aR4+b+<cksum><cr>
   - response: a+<onx.xx>+<off.xx>+<cksum><cr>
6. Read FailSafe Register R5
   - command: aR5<cksum><cr> 0=timer off
   - response: a+<xxx.xx>+<cksum><cr>
7. Read AutoFeed Registers R6
   - command: aR6<cksum><cr>
   - response: a+<xxx.xx>+<xxx.xx>+<cksum><cr>

ECN Network Commands:

1. Change Node Address Command A
   - command: aAc<cksum><cr> where c = new node address
   - response: c<cksum><cr>
2. Send ECN Identification String I
   - command: aI<cksum><cr>
   - response: a<identification string><cksum><cr>
3. Send acknowledgment !
   - command: a!<cksum><cr>
   - response: a<cksum><cr>
4. Reset The Module #
   - command: a#<cksum><cr>
   - response: none Wait 10 milliseconds, then send Acknowledgment