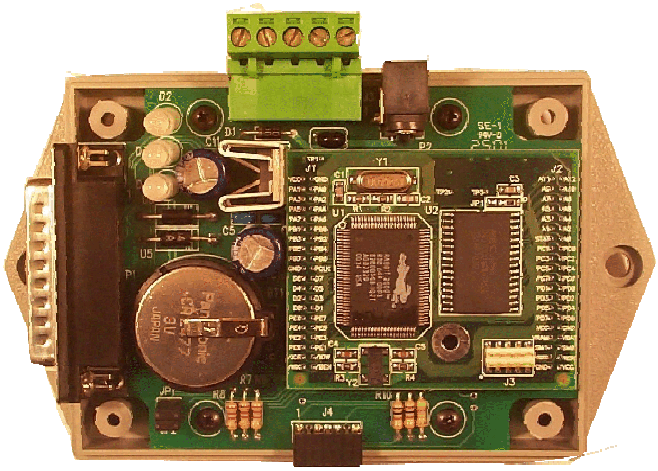
Model: 485-ADR

RS-485 to RS-232
Addressable Converter
with data buffers
V1.0 Firmware



Integrity Instruments

P.O. Box 451
Pine River, MN 56474 USA
Tech phone: (218)587-3120

Order phone: (800)450-2001
Web: http://www.integrityusa.com

Table of Contents

Introduction - Features

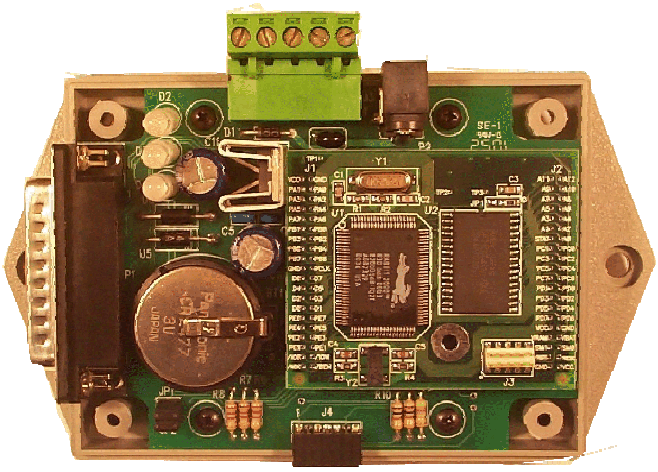


Figure 1

Features:

- Bi-Colored LED Status Indicators
- Independently Configurable RS-485 Baud Rate
- Independently Configurable RS-232 Baud Rate
- Support for most RS-232 byte frames
- Separate Status Monitor
- Built in connector for module setup.

The Integrity Instruments 485-ADR addressable RS-485 to RS-232 converter provides an easy method of utilizing existing point to point devices, such as card readers or CNC machining equipment, to build a point to multi-point communication systems. Each unit is capable of storing up to 2K of data received from the RS-232 port in memory until the host request that information.

Communication parameters for the RS-485 port and the RS-232 port are independently configurable, providing you, the system integrator with maximum flexibility in your designs. Communication to and from the unit from a host controller utilizes a packet protocol that is based on the popular ModBus® protocol. Each unit also comes equipped with a monitor/setup port. During normal operation, the unit transmits status messages out the monitor port, which may be used for on line troubleshooting. This port is also used for unit configuration (See note 1).

Each unit has three bi-colored LED's that are used to indicate the state of conditions within the unit. One LED indicates the operational status of the unit, normal or setup mode. The second LED indicates the status of the RS-232 buffer system. The third LED is used to indicate the status of the RS-485 buffer system. For more detailed explanation see page 5

The RS-232 port supports the most popular formats. Those formats include 7 or 8 bit byte frames, 1 or 2 stop bits, " Even, Odd, or No" parity, and RTS/CTS hardware flow control.

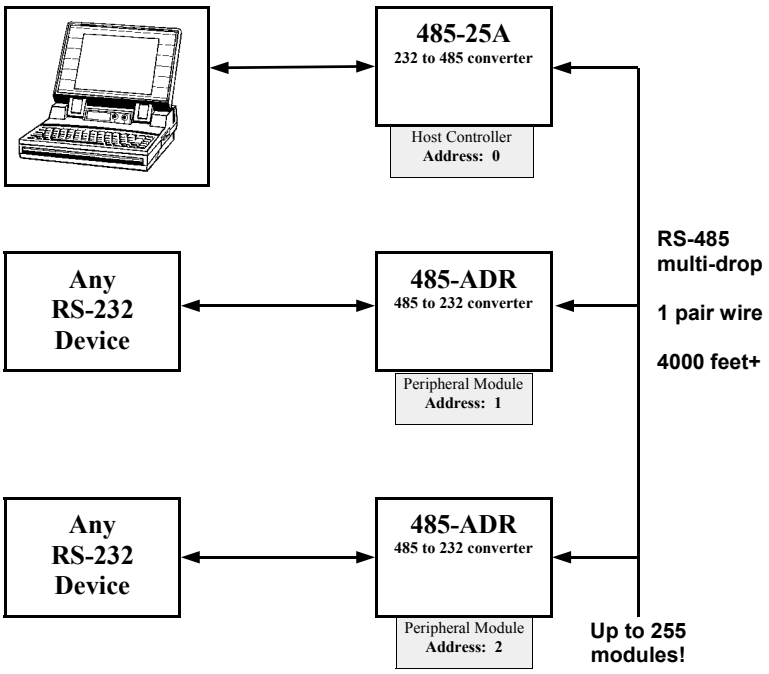
Both the RS-232 port and the RS-485 port support independently configurable baud rates from 110 to 115200 baud. The monitor/setup port is fixed at 9600 baud, 8 bit, 1 stop, no parity (See Note 2.).

Notes:

1. The Setup/Monitor port is an asynchronous TTL serial signal.
2. The Setup/Monitor port requires the 485-ADRPM unit in order to communicate with a PC.

485-ADR Specifications:

MPU:	Rabbit® 2000
RAM:	128 KB
FLASH:	256 KB
EEPROM:	Microchip 25C040
MPU Clock:	18.432 Mhz
RTC Clock:	32.768 KHz
Interface:	RS-485 (multidrop up to 254 nodes) to RS-232 (single ended)
Monitor Port:	Asynchronous TTL, 9600 Baud (Requires 232-TTL-ADR)
Baud Rates:	110 to 115200 (User Programmable)
Status LED:	Bicolor diagnostic LED
RS-485 Buffer LED:	Bicolor diagnostic LED
RS-232 Buffer LED:	Bicolor diagnostic LED
Buffers:	RS-232 Receive: 2KB RS-232 Transmit: 2KB RS-485 Receive: 2KB RS-485 Transmit: 2KB
Watchdog:	MPU has built-in watchdog timer
POR:	MPU contains timed Power On Reset circuitry
Brownout:	MPU brownout detection ciruictry built-in
OperatingTemperature:	0° to 70°C
PCB:	FR4
Power:	7.5Vdc - 24.0Vdc, approx. 200 ma.
Transient:	ZNR surge supression on +V power input TransZorb protection on RS-485 data lines
Turn around response	After data is received, unit responds 5 milliseconds minimum.



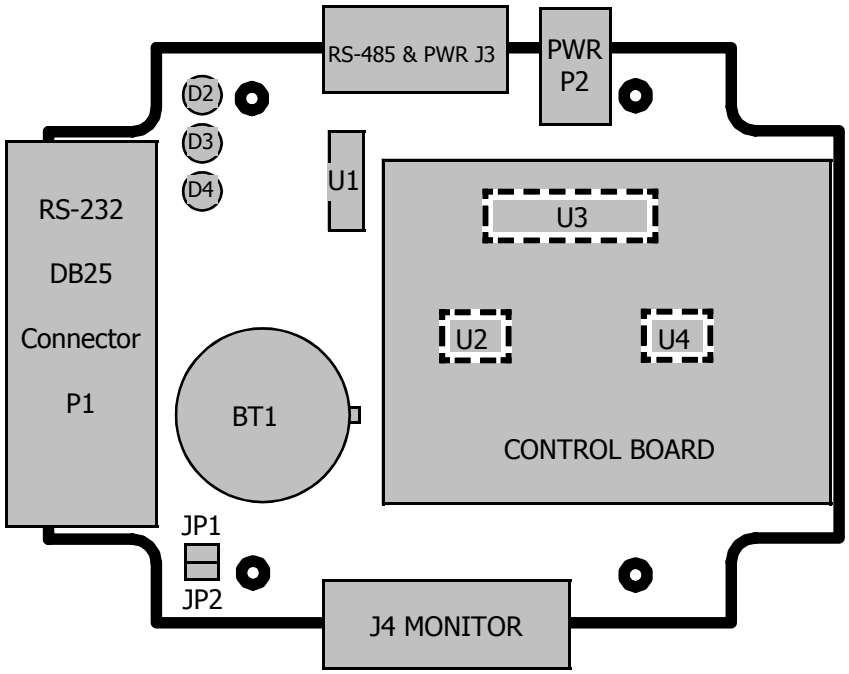


Figure 2

IC Description

U1	7805 Voltage Regulator [TO-220]
U2*	LTC1487 RS-485 driver [8 pin DIP]
U3*	MAX202 RS-232 driver [16 pin DIP]
U4	25C040 EEPROM [8 pin DIP]
Control Board	RCM2020

Connectors

J3	RS-485 and Power
J4	Monitor and parameter port
P1	DB25 RS-232 port
P2	2.5 mm Power jack

Note

Power can be supplied to J3 or P2. Power can be daisy chained via J3 to other devices to the limit of the power capability of the supplying unit.

LED Operation

D2	Operational Status:	Normal Operation	Flashing green, unit active.
		Setup Mode	Flashing green and red
D3	RS-232 Buffer	Data in receive buffer	Green
		Data in transmit buffer	Red
D4	RS-485 Buffer	Data in receive buffer	Green
		Data in transmit buffer	Red

Jumpers

JP1-JP2	Installed	RS-485 active termination at unit

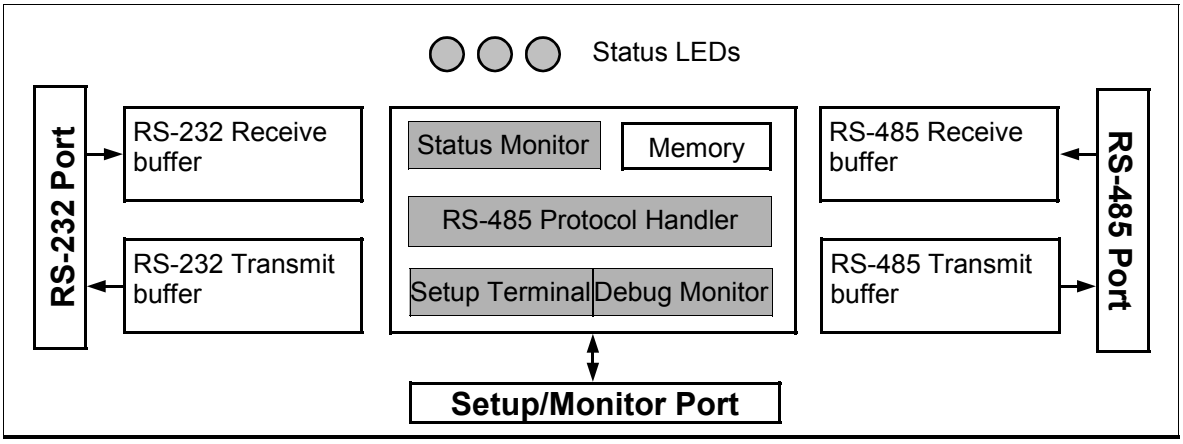


Figure 3

The Integrity Instruments models **485-ADR** converts and buffers data from the RS-485 bus and re-transmits that information out the RS-232 port. RS-485 data is carried in via a bus protocol that includes a CRC style checksum. Any information transmitted from an RS-232 device is buffered until the host requests that the information be sent over the RS-485 bus. The RS-485 and RS-232 baud rates are individually configurable allowing you greater flexibility in your system design.

RS-232 Interface:

- RS-232 operates Full Duplex (unless otherwise noted)
- Data Rates: 110 to 115200 BPS
- Data bits: 7 or 8
- Stop bits: 1 or 2 (Full Duplex not supported with 2 stop bits)
- Parity: None, Even, or Odd
- Hardware Handshaking: None or RTS/CTS

RS-485 Interface:

- RS-485 operates Half Duplex
- Data rates: 110 to 115200 BPS
- Each module (node) on the bus has a unique Address 1 to 254 (0x01-0xFE hex)
- We use the latest Linear Technologies RS-485 bus drivers (LTC1487) allowing up to **255 nodes** on the RS-485 multi-drop bus
- Address 0 (0x00 hex) is reserved for the Host controller
- Address 255 (0xFF hex) is reserved for Broadcast messages - **accepted by all modules on the RS-485 bus.**

RS-485 Packet Field Description

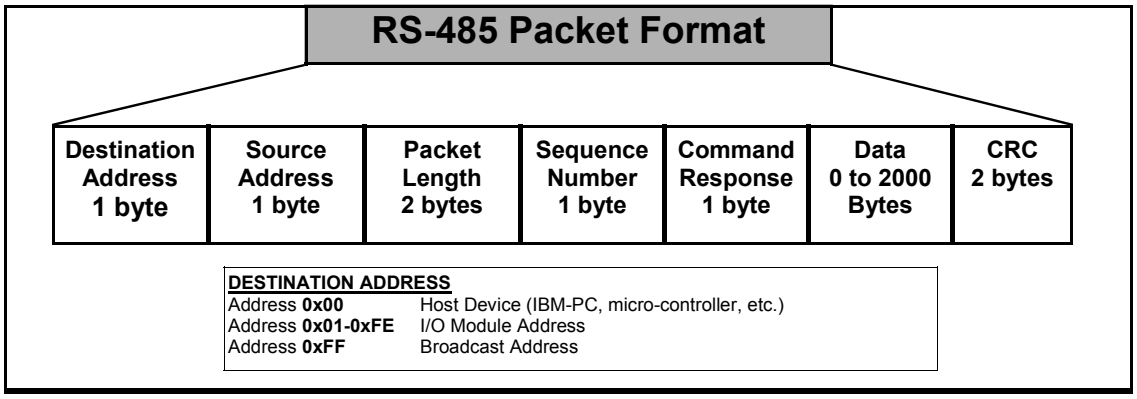


Figure 4

Field	Description
Destination Address	This field may be in the range of 0 to 255. This is the number used as an address for a packet of information being sent to a unit.
Source Address	This field may be in the range of 0 to 255. This is the number used as the destination address in a response packet.
Packet Length	This is a 16 bit value indicating the number of bytes following this field, starting with and including the sequence number, Command/ Response, Data and the CRC (See note 1).
Sequence Number	This 8 bit value is used to uniquely identify each packet. The host will generate a new sequence number for each new packet that it generates. This number will also be included in the response packet from a unit. A target unit may choose to ignore an incoming packet if this sequence number is equal to the previous sequence number.
Command Response	This field is used to access the different functions of an end point. See table on the next page.
Data	This is the only variable length field. It is used to information to/or from an endpoint. It's meaning is dependant on the command/ response byte.
CRC	This 16 bit value is used to validate any data packet. The CRC computation starts with and includes the Destination address, and ends with and includes the last byte of data not including this CRC. For more information please refer to the CRC Generation section of this manual (See note 1).

Notes:

1. As is the case in ModBus®, all multi-byte fields are set big-endian. In other words the most significant byte is transmitted first, and the least significant byte is sent last.

Firmware v1.0 Command/Response

Command and Response

The table below illustrates the Integrity Instruments ADR module commands and responses.

Master Command	Slave Response	Master Data Bytes	Slave Data Bytes	Description
0x01	0x81 or 0x82	N Data Bytes	0	Send packet payload data to the RS-232 transmit buffer. This command checks the sequence number. If the slave returns a 0x82, then the data in the RS-232 buffer has been corrupted due to a buffer overrun. If the slave returns a 0x81 the packet payload was received O.K.
0x03	0x83	0	2	RS-232 transmit buffer character count
0x04	0x84	2	N Data Bytes	Retrieve up to the requested number of characters from the RS-232 receive buffer.
0x05	0x85	0	2	Get the current character count in the RS-232 receive buffer
0x06	0x86	0	2	Get the current 485-ADR Version number. Data: Major-Minor version number.
0x07	0x87	0	0	Reset buffers

Note:

In all cases of commands and responses: If no response is received, your software program should re-issue the command. The reason for this is if the data is corrupted, the CRC checksum process within the ADR unit will disregard any packet with corrupted data. As there will be several devices on the RS-485 link, and the data from the host is received by all devices, there is no way that all devices can report back with a bad data response.

Even though the RS-232 buffers are 2K bytes, it is suggested the RS-232 data be limited to 1.5K bytes to stay within reasonable error detection boundaries of the 16 bit CRC.

Typical Software Flow

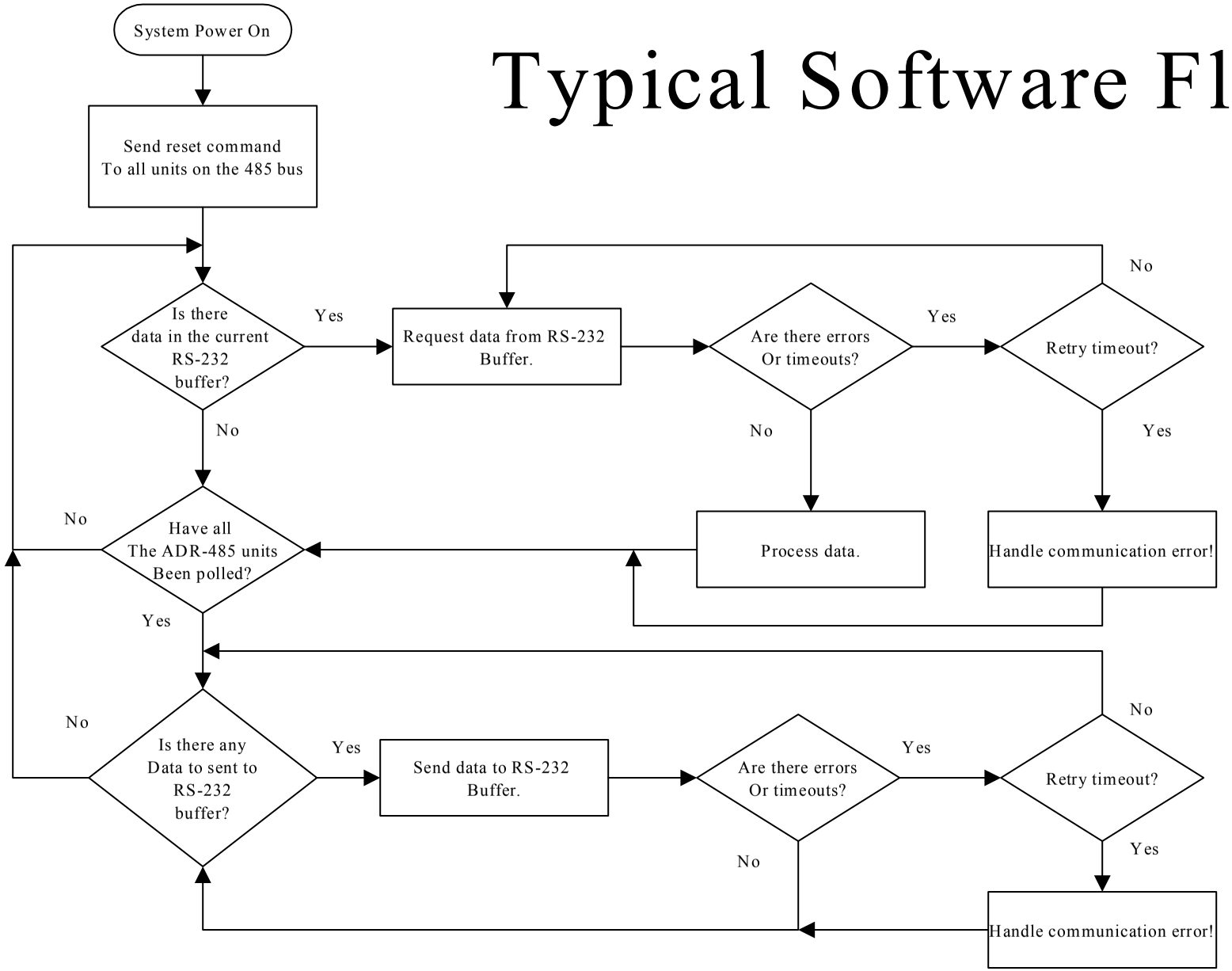


Figure 5

CRC Generation

The Cyclical Redundancy Check (CRC) field is two bytes, containing a 16-bit binary value. The CRC value is calculated by the transmitting device, which appends the CRC to the message. The receiving device recalculates a CRC during receipt of the message, and compares the calculated value to the actual value it received in the CRC field. If the two values are not equal, an error results.

The CRC is started by first preloading a 16-bit register to all 1's. Then a process begins of applying successive eight-bit bytes of the message to the current contents of the register. Only the eight bits of data in each character are used for generating the CRC. Start and stop bits, and the parity bit, do not apply to the CRC.

During generation of the CRC, each eight-bit character is exclusive ORed with the register contents. The result is shifted in the direction of the least significant bit (LSB), with a zero filled into the most significant bit (MSB) position. The LSB is extracted and examined. If the LSB was a 1, the register is then exclusive ORed with a preset, fixed value. If the LSB was a 0, no exclusive OR takes place.

This process is repeated until eight shifts have been performed. After the last (eighth) shift, the next eight-bit character is exclusive ORed with the register's current value, and the process repeats for eight more shifts as described above. The final contents of the register, after all the characters of the message have been applied, is the CRC value.

Generating a CRC
Step 1 Load a 16-bit register with FFFF hex (all 1's). Call this the CRC register.

Step 2 Exclusive OR the first eight-bit byte of the message with the low order byte of the 16-bit CRC register, putting the result in the CRC register.

Step 3 Shift the CRC register one bit to the right (toward the LSB), zerofilling the MSB. Extract and examine the LSB.

Step 4 If the LSB is 0, repeat Step 3 (another shift). If the LSB is 1, Exclusive OR the CRC register with the polynomial value A001 hex (1010 0000 0000 0001).

Step 5 Repeat Steps 3 and 4 until eight shifts have been performed. When this is done, a complete eight-bit byte will have been processed.

Step 6 Repeat Steps 2 ... 5 for the next eight-bit byte of the message. Continue doing this until all bytes have been processed.

Result The final contents of the CRC register is the CRC value.

Step 7 When the CRC is placed into the message, its upper and lower bytes must be swapped as described below.

Example
An example of a C language function performing CRC generation is shown on the following pages. All of the possible CRC values are preloaded into two arrays, which are simply indexed as the function increments through the message buffer. One array contains all of the 256 possible CRC values for the high byte of the 16-bit CRC field, and the other array contains all of the values for the low byte.
Indexing the CRC in this way provides faster execution than would be achieved by calculating a new CRC value with each new character from the message buffer.

Note: This function performs the swapping of the high/low CRC bytes internally. The bytes are already swapped in the CRC value that is returned from the function. Therefore the CRC value returned from the function can be directly placed into the message for transmission.

The function takes two arguments:
unsigned char *puchMsg ; /* A pointer to the message buffer containing binary data to be used for generating the CRC */

unsigned short usDataLen ; /* The quantity of bytes in the message buffer. */

The function returns the CRC as a type unsigned short.

CRC Generation Function

```
unsigned short CRC16(puchMsg, usDataLen)
unsigned char *puchMsg ;                /* message to calculate CRC upon      */
unsigned short usDataLen ;              /* quantity of bytes in message       */
{
        unsigned char uchCRCHi = 0xFF ; /* high CRC byte initialized          */
        unsigned char uchCRCLo = 0xFF ; /* low CRC byte initialized           */
        unsigned uIndex ;               /* will index into CRC lookup         */

               /* table */
        while (usDataLen--)             /* pass through message buffer        */
        {
          uIndex = uchCRCHi ^ *puchMsg++ ;        /* calculate the CRC  */
          uchCRCHi = uchCRCLo ^ auchCRCHi[uIndex] ;
          uchCRCLo = auchCRCLo[uIndex] ;
        }
        return (uchCRCHi << 8 | uchCRCLo) ;
}
```

High Order Byte Table

```
/* Table of CRC values for high-order byte */
static unsigned char auchCRCHi[] = {
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40
} ;
```

Low Order Byte Table

```
/* Table of CRC values for low-order byte */
static char auchCRCLo[] = {
0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06,
0x07, 0xC7, 0x05, 0xC5, 0xC4, 0x04, 0xCC, 0x0C, 0x0D, 0xCD,
0x0F, 0xCF, 0xCE, 0x0E, 0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09,
0x08, 0xC8, 0xD8, 0x18, 0x19, 0xD9, 0x1B, 0xDB, 0xDA, 0x1A,
0x1E, 0xDE, 0xDF, 0x1F, 0xDD, 0x1D, 0x1C, 0xDC, 0x14, 0xD4,
0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3,
0x11, 0xD1, 0xD0, 0x10, 0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3,
0xF2, 0x32, 0x36, 0xF6, 0xF7, 0x37, 0xF5, 0x35, 0x34, 0xF4,
0x3C, 0xFC, 0xFD, 0x3D, 0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A,
0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38, 0x28, 0xE8, 0xE9, 0x29,
0xEB, 0x2B, 0x2A, 0xEA, 0xEE, 0x2E, 0x2F, 0xEF, 0x2D, 0xED,
0xEC, 0x2C, 0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26,
0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0, 0xA0, 0x60,
0x61, 0xA1, 0x63, 0xA3, 0xA2, 0x62, 0x66, 0xA6, 0xA7, 0x67,
0xA5, 0x65, 0x64, 0xA4, 0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F,
0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB, 0x69, 0xA9, 0xA8, 0x68,
0x78, 0xB8, 0xB9, 0x79, 0xBB, 0x7B, 0x7A, 0xBA, 0xBE, 0x7E,
0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C, 0xB4, 0x74, 0x75, 0xB5,
0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71,
0x70, 0xB0, 0x50, 0x90, 0x91, 0x51, 0x93, 0x53, 0x52, 0x92,
0x96, 0x56, 0x57, 0x97, 0x55, 0x95, 0x94, 0x54, 0x9C, 0x5C,
0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E, 0x5A, 0x9A, 0x9B, 0x5B,
0x99, 0x59, 0x58, 0x98, 0x88, 0x48, 0x49, 0x89, 0x4B, 0x8B,
0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C,
0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42,
0x43, 0x83, 0x41, 0x81, 0x80, 0x40
} ;
```

Unit Setup

The 485-ADR unit has a built in setup/monitor port that you use to configure the unit operation. You will need a 485-ADRPM unit before you can change the factory configuration data. There are two ways to configure the 485-ADR unit.

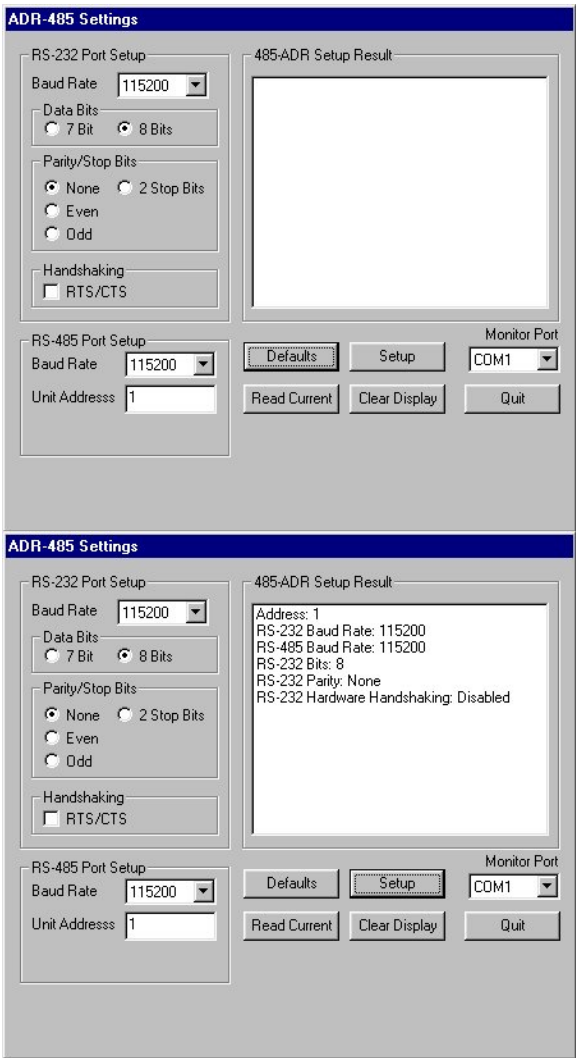
1. By using the convenient setup program available on our website.
2. By using any terminal emulation program, like Hyper Terminal

485-ADRPM PROGRAM MODULE MODE AND SWITCH SETTINGS

SWITCH	SETUP MODE	MONITOR MODE	FLASH MODE (factory use)
SW 1	ON	OFF	OFF
SW 2	OFF	OFF	ON
SW 3	OFF	OFF	ON
SW 4	OFF	OFF	ON

Using the Setup Program

Connect the 485-ADRPM DB-9 connection to a com port on your computer.
Set the dip switch for setup mode: 1-On, 2-Off, 3-Off, 4-Off. (status light will blink red and green)
Open the setup window by clicking on the product test icon. The following window should appear.

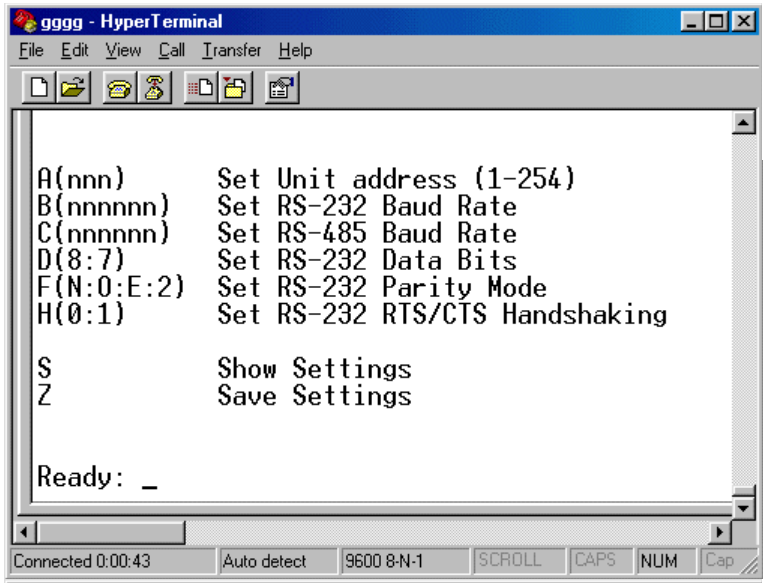


1. Select the com port that you are connected to on your PC.
2. Set the RS-232 port for the baud rate, data, parity, and stop bits, and if you want to use handshaking.
3. Set the RS-485 port for the baud rate you will be using, and set the address for the port. Valid numbers are from 1 to 255. **NOTE!** You can set the RS-485 for any valid number but the baud rate **must be the same** for all modules and devices on the RS-485 link.
4. Actuate the Setup button. The data will be sent to the unit, and will appear in the window.
5. To verify what you sent actuate the Clear Display button, then the Read Current button. The 485-ADR unit will send back its' current settings.
6. You are now ready to program another unit.

Using a Terminal Emulation Program

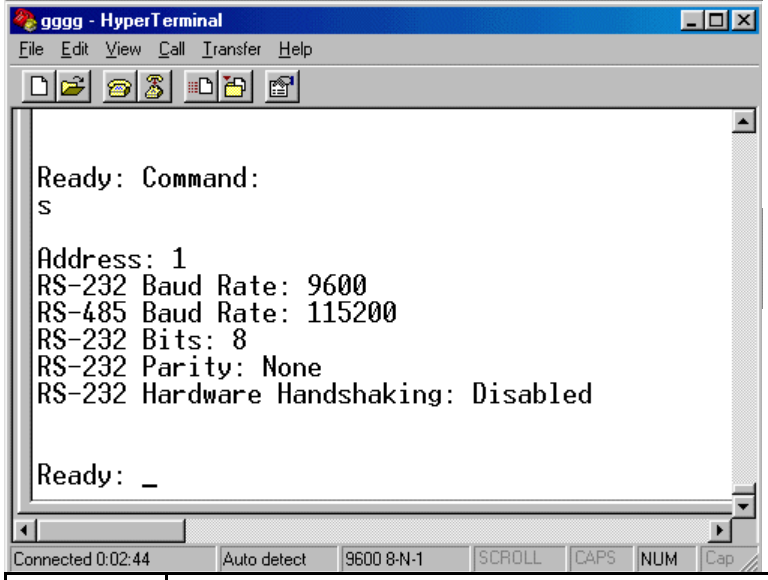
Connect the 485-ADRPM DB-9 connection to a com port on your computer.
Set the dip switch for setup mode: 1-On, 2-Off, 3-Off, 4-Off (status light will blink red and green).

Start your favorite terminal emulation program and configure it for the com port that you connect the 485-ADRPM unit to. Set the port to: 9600 baud, 8 data bits, 1 stop bit, no parity, and no hand-shaking.
Press the enter key.



You should see a screen that looks like this

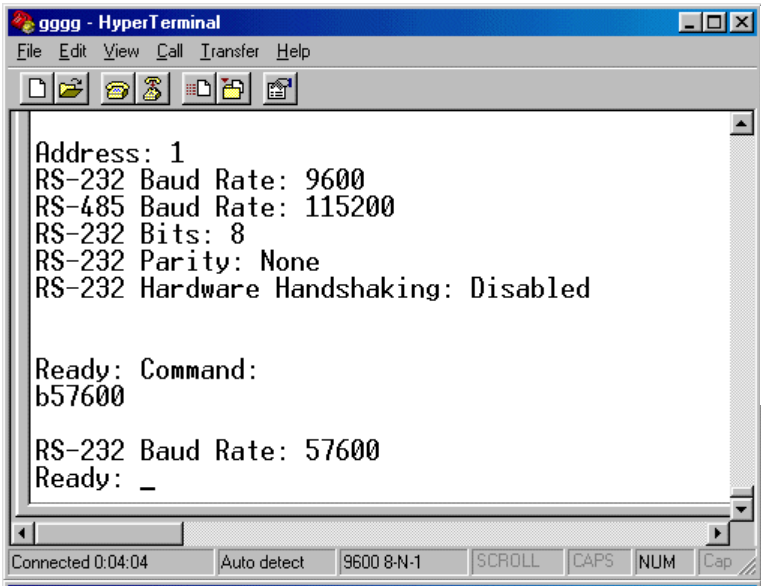
This is the setup menu terminal. From this menu you can configure all of the operational parameters, as well as view the current or proposed settings.



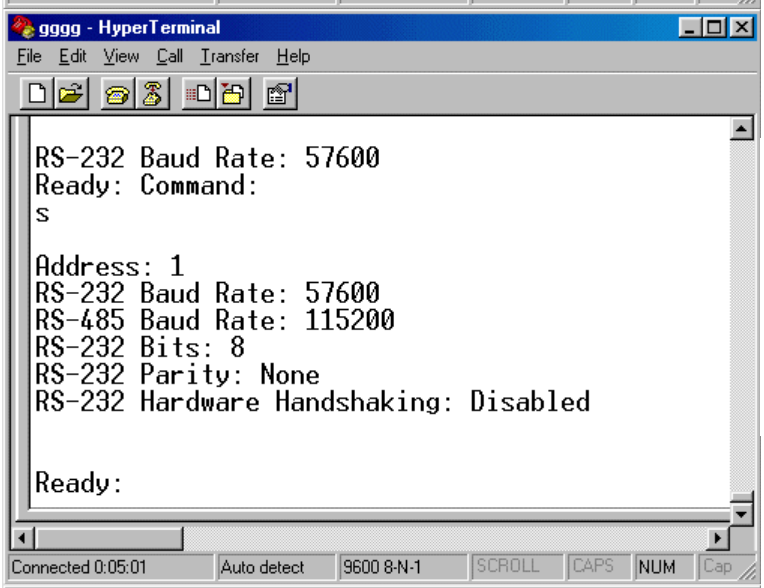
To view the current settings press “S” followed by the enter key.
You should see a screen that looks like this:

To change any of the parameters press the alpha character associated with the parameter that you want to change followed immediately by the desired parameter then press enter

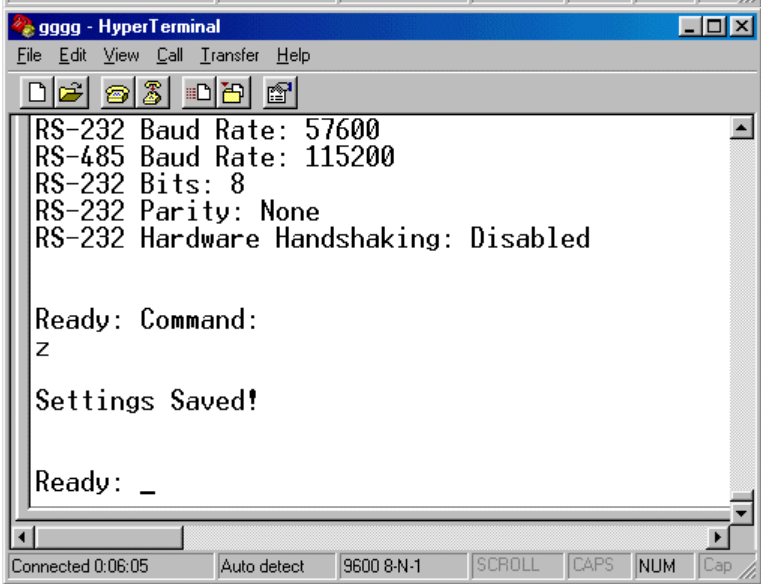
COMMAND	PARAMETER SET	VALID VALUES
A	UNIT ADDRESS ON THE RS-485 LINK	1 TO 254
B	RS-232 BAUD RATE FOR THE RS-232 DEVICE	110 THRU 115,200
C	RS-485 BAUD RATE ON THE RS-485 LINK	110 THRU 115,200
D	NUMBER OF RS-232 DATA BITS	7 OR 8
F	RS-232 PARITY MODE	NONE ODD EVEN 2
H	RS-232 RTS/CTS HANDSHAKING	0 DISABLED 1 ENABLED



To change any of the parameters press the alpha character associated with the parameter that you want to change followed immediately by the desired parameter then press enter. For example to change the RS-232 baud rate to 57600 type B57600 <enter>. You should see this.



The current settings may be viewed at any time during the setup process. To view the current settings press “S” followed by the enter key.
You should see a screen that looks like this

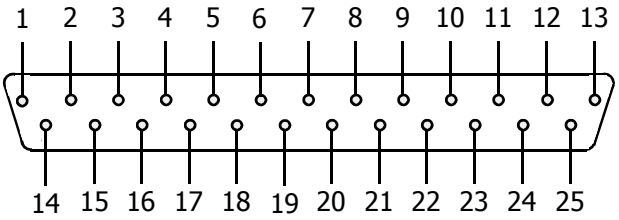


When you have finished configuring your unit, the new settings must be saved before they can be used. This is accomplished by entering Z <enter>.

This will write all of the current settings into non-volatile memory within the unit. As soon as the unit is returned to normal operating mode, these settings will take effect.

Return to normal operating mode by setting the switches on the 485-ADRPM to off.

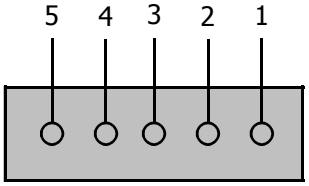
Connector Pinouts



Front View

DB25 RS-232 PINOUTS

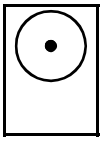
DB25 Pin	Description
1	N/C
2	Transmit Data
3	Receive Data
4	RTS
5	CTS
6	N/C
7	GND
8	N/C
9	N/C
10	N/C
11	N/C
12	N/C
13	N/C
14	N/C
15	N/C
16	N/C
17	N/C
18	N/C
19	N/C
20	N/C
21	N/C
22	N/C
23	N/C
24	N/C
25	N/C



Front View

Power/RS-485 Pinouts

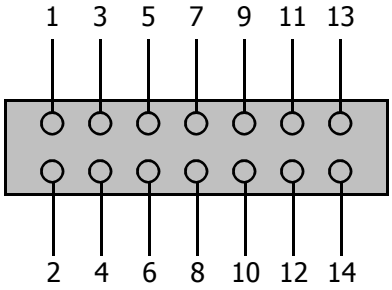
Header Pin	Description
1	9-25 Volts
2	Power Ground
3	Signal Ground
4	B
5	A



Power Input

Power Pin	Description
Center	9-25 Volts
Edge	Power Ground

Connector Pinouts



Front View

Monitor/Setup Port

Header Pin	Description
1	Rx Data (TTL)
2	GND
3	CLKA
4	Vcc
5	Reset
6	Tx Data (TTL)
7	N/C
8	Status
9	SMode0
10	SMode1
11	PE7
12	PE6
13	N/C
14	N/C

WARRANTY
Integrity Instruments warranties **all** products against defective workmanship and components for the life of the unit. Integrity Instruments agrees to repair or replace, at it's sole discretion, a defective product if returned to Integrity Instruments with proof of purchase. Products that have been mis-used, improperly applied, or subject to adverse operating conditions fall beyond the realm of defective workmanship and are not convered by this warranty.