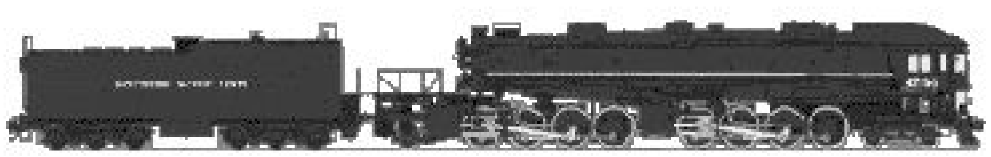


THE COMPUTER/ MODEL RAILROAD INTERFACE (C/MRI)

USER'S MANUAL
Version 3.1a

By Dr. Bruce A. Chubb, MMR

Chapter 4: Super Mini-Node Interface Card (SMINI)

www.jlcenterprises.net

Chapter 4

Super Mini-Node Interface Card (SMINI)

This chapter is devoted to showing how to set up your C/MRI as a serial interface, including step-by-step instruction for building the Super Mini-Node Interface Card (SMINI). Fig. 1-1, in Chapter 1, shows a photograph of the SMINI card. From the perspective of previous C/MRI components the SMINI combines enhanced features of the original design Universal Serial Interface Card, the USIC, with two output cards and an input card, placing everything on a single board not much larger than the original USIC. This combination provides a complete system node with 48 outputs and 24 inputs on a single card.

The SMINI provides everything required to set up a very capable node on a single card, at a very reasonable cost. For example, a single SMINI can drive 48 signal LEDs and read 24 detector and switch position inputs. Depending upon the size of your layout, this single card may be all you need to add signaling to your railroad.

If you desire more I/O than provided by a single SMINI, you can simply distribute additional SMINI cards around your layout to build up to whatever level of capability you desire. With an addressable capacity of 128 SMINI nodes, with 72 I/O lines per node, the maximum capacity of the C/MRI system using the SMINI is 9,216 I/O lines.

This distributed approach minimizes all local I/O wiring. Any device you wish to connect such as a wayside signal, occupancy detector, switch motor, switch contact, pushbutton, panel display LED, or almost anything else electrical, simply plugs into the nearest node.

At node locations requiring more than 72 I/O lines, you can group SMINI cards together. However, often a more attractive option is to use the new **Super USIC**. Using the **SUSIC** is the ideal approach for locations requiring large amounts of concentrated I/O such as at a lever-type CTC machine. The SUSIC option is covered in Chapter 10.

SERIAL INTERFACE STANDARDS

The SMINI and the SUSIC can make use of four interface standards: Universal Serial Bus (USB), RS232, RS422 and a “full duplex” implementation of RS485. All four standards can be used to connect your computer to an SMINI and to an SUSIC. Additionally, in every case, the RS422 or the RS485 full duplex implementation is used to make the “daisy-chain” connection between multiple nodes.

The only requirement for connecting the C/MRI is that your computer has an RS232 serial port and/or a Universal Serial Bus (USB). This is the case for nearly every personal computer, although the RS232 serial port has mostly been phased out of the latest PCs. It can, however, frequently be added as an expansion item. By contrast, some Macintosh models provide an RS422/RS485 serial port. Each can work just fine with the C/MRI. Alternatively, if your computer has USB ports only, then it’s possible to add a USB to RS232 or USB to RS422/485 Converter Cable to make the computer to C/MRI connection.

The most straightforward approach, and the one that typically provides the best overall system performance, is using a computer with a built-in RS232 Serial Port. In general, this is the preferred option and Fig. 4-1 shows the typical setups.

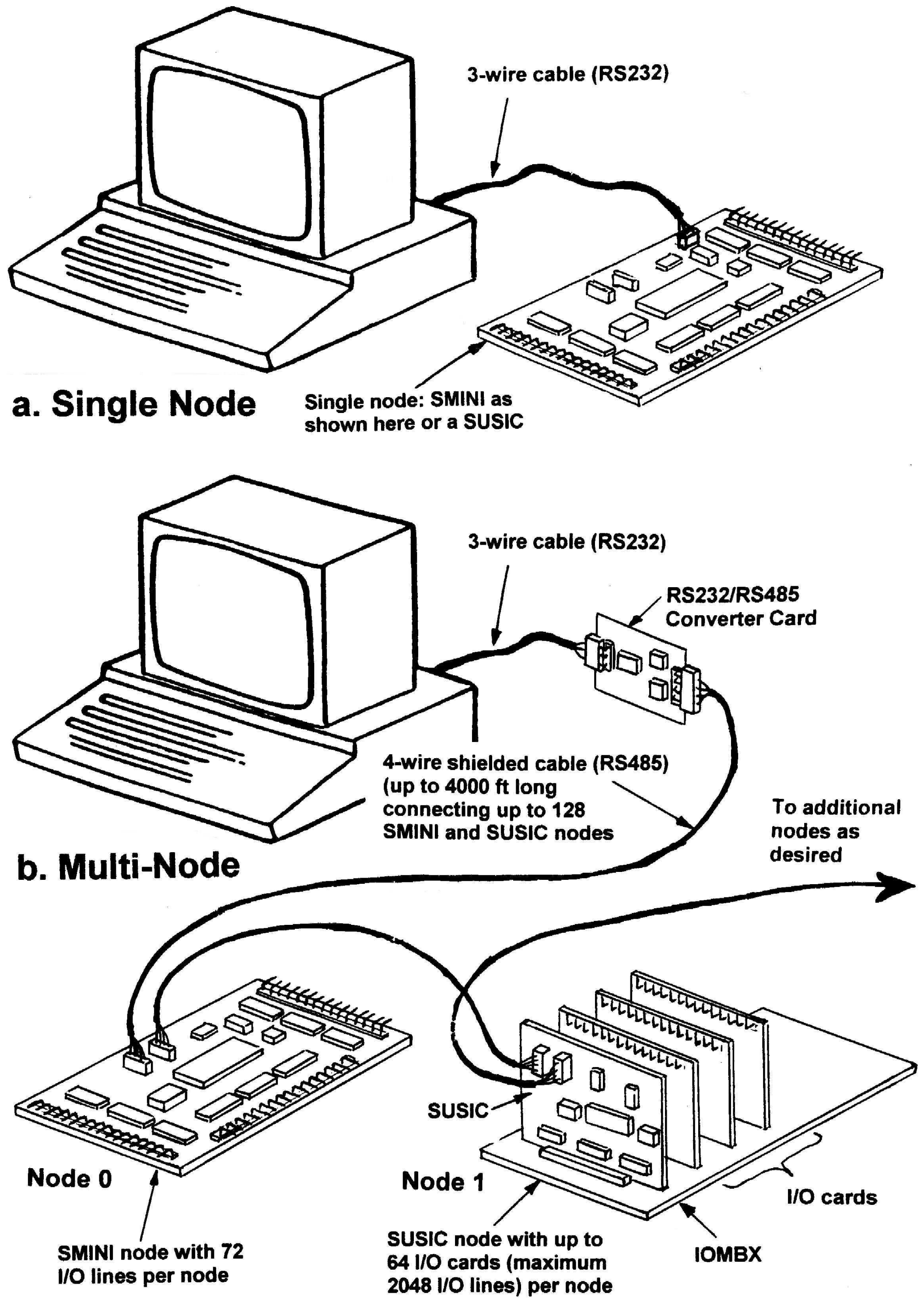

3-wire cable (RS232)
a. Single Node
Single node: SMINI as shown here or a SUSIC
3-wire cable (RS232)
RS232/RS485 Converter Card
4-wire shielded cable (RS485) (up to 4000 ft long connecting up to 128 SMINI and SUSIC nodes
To additional nodes as desired
b. Multi-Node
SUSIC
Node 0
Node 1
I/O cards
SMINI node with 72 I/O lines per node
SUSIC node with up to 64 I/O cards (maximum 2048 I/O lines) per node
IOMBX

Fig. 4-1. Serial interfacing example systems employing RS232 and RS422/485

Fig. 4-1a illustrates the simplest arrangement for systems comprising just a single node. It requires a single 3-wire RS232 cable to connect between the computer’s RS232 serial port and the RS232 header on the SMINI or the SUSIC. Alternatively, if your computer is limited to USB only, then you will need to incorporate a USB to RS232 Converter Cable to make the connection.

The typical multi-node setup is illustrated in Fig. 4-1b. Fundamentally, every multi-node application requires that all nodes, plus the connection leading toward the computer, need to be “daisy-chain” connected using RS422/485, a 4-wire cable. Each SMINI and SUSIC includes an adjacent pair of RS485 pins making the daisy-chaining easy to accomplish. An RS232 to RS485 Converter Card, such as the RS485 card supplied by JLC Enterprises and covered in detail near the end of this chapter, provides the required conversion to make the connection to the computer’s RS232 serial port. Alternatively, if your computer has USB only, then you will need to incorporate either a USB to RS232 Converter Cable between the computer and the RS485 card or a USB to RS422/485 Converter Cable connected between the computer and the RS485 header on the SMINI or SUSIC.

If you are starting out with a single node, with plans to expand at a later date to multiple nodes, then it’s a good idea to configure your first node for RS485. Then, you can use the Fig. 4-1 setup for connecting to your computer and have everything ready simply to connect additional nodes as your system expands.

Although many C/MRI users effectively make use of USB Converter Cables, their application can pose some challenges. These include adding another layer of complexity and added system overhead. The latter may not be a problem with small systems. However, it can substantially degrade system real-time performance with larger systems. Thus, whenever possible, I recommend implementing the simpler of the two choices, i.e. making use of a computer that has an available RS232 Serial Port. However, how to apply each of the different serial interfacing options, including detailed wiring information for each approach, is presented near the end of this chapter.

For readers interested in more information regarding serial interfacing, the following four subsections highlight each of the applicable serial interface standards. If you would rather skip the detail, please do not hesitate to jump ahead to the section, ***BAUD RATE SETTINGS.***

Universal Serial Bus (USB)

The Universal Serial Bus (USB) was developed to make it fundamentally easier to connect external devices to PCs by replacing the multiplicity of different connectors typically located at the back of PCs. In addition it addresses the usability issues of existing interfaces, simplifies software configuration of all devices connected to USB, and permits greater bandwidths for external devices. Its first release was in 1996 and it has been gaining popularity ever since. As a result, USB has become the “commercial standard” for connecting all types of computer peripherals such as the mouse, keyboards, printers, scanners, digital cameras, personal media players, joy sticks, card readers, flash drives and external hard drives. Similiarly, the USB can be used for connecting to the C/MRI.

The most common USB connector, referred to as Type A, has 4-conductors. This is the connector found on most modern peripheral devices and installed on most, if not all modern PCs. Two wires deliver 5Vdc power and ground. The remaining two wires provide ballanced-line, half-duplex communication. This means that in general, peripherals cannot communicate with the host, the PC, unless the host specifically requests communication. Data rates are up to 12Mbps for USB1.0 and even higher for USB2.0 and USB3.0. All of these rates are much higher than needed with our railroad applications and the C/MRI.

Although basic data rates are high, there is extra overhead and subsequent delays in the manner in which USB handles data to and from the computer. These delays, typically 1 millisecond or greater, are of little importance when driving a printer or scanning a document. However, with real-time applications such as the C/MRI the overall system performance can be degraded when using USB. This is especially true for large systems, such as the SVOS where we have 6 nodes with 140 I/O cards with the corresponding need to transfer 5480 bits during each pass through the real-time loop.

RS232 Standard

Prior to the domination now exhibited by USB, the RS232 standard was historically the most common serial interface. It uses a voltage of -12Vdc to define the Mark signal (Logic 1), and +12Vdc to define the Space (Logic 0).

Typically, RS232 ports use either a standard 9- or 25-pin, D-type male plug connector called a DB9P or DB25P respectively. Thus, to mate with them you need a female socket connector, a DB25S or DB9S. Pins 2 and 3 are the data wires, and are called the Transmit Data and Receive Data lines. Specific allocation of these two pins depends on whether the connected or peripheral device is configured as *Data Terminal Equipment* (DTE) or *Data Communication Equipment* (DCE). A DTE transmits on Pin 2 and receives on Pin 3, but DCE transmits on Pin 3 and receives on Pin 2. Some computers have DTE ports, while others have DCE, so the first step in setting up any RS232 interface is to determine the direction of data flow on Pins 2 and 3. The standard RS232 connections, used on most computer ports, are summarized in Table 4-1.

Table 4-1. Typical RS232 connections

	Connector Type and pin number	
Function:	**DB25P**	**DB9P**
Transmit Data	2	3
Receive Data	3	2
Signal Ground	7	5

Transmission and reception directions are defined from the perspective of the computer. These three wires (Pins 2, 3, and 7 or 5) suffice for bi-directional communication between two devices. The other pins, 22 in number for the DB25, are for handshaking between the computer and the peripheral device. For example, Pin 6, Data Set Ready on the DB25, is one that a printer might use to tell the computer it is ready to accept the next character.

There is less standardization in how computers and peripherals use the hand-shaking lines. Besides the interchange of Pin 2 and 3 functions, and the use in some terminals of a male DB25P connector while others have a female DB25S, there are good reasons why the RS232 is sometimes called "the most non-standard standard."

Length of cable also affects RS232 transmission capability and limits the maximum transmission rate. The specified maximum cable length with RS232 is 50 feet without an audio modem (modulator-demodulator) at each end. In practice, however, RS232 signals are routinely run for 100 to 200 feet and operated at high baud rates without problems. Such installations are not standard, but they do work. For more reliable, error-free transmission over long distances, you should use RS485.

The RS232 circuitry used with the original design USIC required both + and –12Vdc supply inputs. The updated circuitry used with the new cards generates its own + and – signal requirements (actually ±10Vdc) so that the latest design SMINI, SUSIC and RS485 cards only require +5Vdc power.

RS232 is limited to one receiver and one transmitter. That is suitable for many applications such as connecting a printer, modem or a single SMINI or SUSIC to your computer. However, to support a distributed system with multiple SMINIs (and/or SUSICs), RS232 will not do the job. You need to upgrade to RS485 or RS422, and later I will show you how to build an RS232 to RS485 converter card. Additionally, if your computer is limited to USB ports only, then you need to employ a USB to RS232 Converter Cable or as we will observe later in this chapter, a USB to RS422/485 Converter Cable.

RS422 Standard

With the exception of some versions of the Macintosh that provide a direct RS422/485 connection, few if any other personal computers directly support RS422 or RS485. However the advantages of RS422, and even more importantly RS485 make their application important to the C/MRI. I will cover RS422 first and then look at the differences when taking advantage of added capability provided by RS485.

The main advantages of RS422, when compared to RS232, arise from RS422 using twisted-pair, balanced transmission lines. The difference between the voltages on the two wires signals a Mark or a Space. If the difference is positive by more than 200mV, the receiver reads a Mark; if the difference is negative by more than 200mV, the receiver reads a Space. A secondary benefit arising from these levels is that the transmitters and receivers can easily be operated with the normal +5Vdc logic power supply.

RS422 is a full-duplex system, meaning that a device using RS422 can send and receive information simultaneously. Four wires are employed. One twisted pair is used for sending data and a second twisted pair for receiving data. Whereas RS232 can support communication between only two devices, an RS422 port can be connected with up to 16 devices, or nodes. Each additional node is simply **daisy-chained**, with the same 4-wire cable running from the computer to the first node, on to the second, to the third and so on up to a possible total of 16 nodes.

On one hand you can think of both RS232 and RS422 transmitting signals differentially, but RS232 suffers from using the voltage difference between the signal-wire and ground-wire. This is inferior because each end of the ground wire is locally grounded. Any difference in potential at the ends of the link causes current to flow through the ground wire, and the wire's resistance ensures a difference in ground potential at each end that can cause errors in the data.

RS422 grounding requirements are much less critical since local ground potential does not affect the Mark or Space signals. With the ground potential problem reduced, the RS422 can send its Mark and Space signals closer together to operate reliably at higher baud rates and over greater distances.

RS485 Standard

In many ways the RS485 standard is similar to RS422. The principal exception is that the RS485 specification is Half-duplex, meaning that it cannot send and receive data simultaneously. Although there is significant added complexity in controlling Half-duplex data flow with RS485, its advantage is that the 4-wire requirement with RS422 is reduced to 2-wires with RS485. Additionally, because RS485 was developed later than RS422, the RS485 transceiver ICs, a combination transmitter and receiver, exhibit higher input impedance and output drive capability. The result is an increase in maximum node handling capability from 16 to 128.

Although the C/MRI does not have a requirement for two-way simultaneous communication, there are definite advantages to its using a separate wire pair for sending and receiving data - namely reduced design complexity and easier system debug. For example, if there is activity on the one pair, we know data is flowing from the PC to the external hardware. Similarly, with activity on the other pair, we know that data is flowing from the external hardware to the PC. You avoid needing “data direction control circuitry. You also avoid the need of an oscilloscope to check for the direction of data flow and the possibility of device failure causing bus contention whereby data tries to flow in both directions simultaneously on the same pair of wires. Additionally, with separate send and receive wire pairs, it is easy to incorporate sending and receiving data LEDs as implemented on the JLC provided SMINI, SUSIC and RS485 cards.

The 4-wire design, as implemented by the C/MRI, takes advantage of both worlds, RS422 and RS485, by dedicating one set of the more advanced RS485 transceivers to sending data from the PC and a second set for handling data being received by the PC. And, because we are using the more advanced RS485 ICs, we expand our maximum node capability from 16 to 128. Because of this doubling up of the number of RS485 transceiver ICs and the corresponding use of 4-wires, this arrangement is typically referred to throughout the commercial and industrial world as a “Full Duplex” implementation of RS485. However, just because the 4-wire arrangement has the capacity for supporting simultaneous communication in both directions, it does not mean that the capability is always utilized, as is the situation with the C/MRI.

Because typical computer interfacing distances are very short compared to the 4000-foot limit of RS485, the type of cable used is not overly important. For some applications, regular 4-wire telephone cable provides acceptable performance. However, the model railroad environment can create a lot of electrical noise, especially when using high-frequency power-pulse systems like DCC. This electrical noise can easily couple into the RS485 cabling to create data transmission errors. Such cross-coupling effects become especially prevalent where RS485 cable runs are long and in close parallel proximity to DCC or Railcommand track wiring. To be on the conservative side, it is always best to use dual twisted pair shielded transmission cable for RS485.

Additionally, connections between nodes should be contiguous from the computer to the first node, then on to the next and then the next. A “star” configuration, where there are branches going off to different nodes from a central point, should be avoided. Also, it is generally recommended that cable be terminated with a resistor/capacitor combination equivalent to the characteristic impedance of the cable. Proper cable termination, a topic covered near the end of this chapter becomes increasingly important as cable length is increased.

BAUD RATE SETTINGS

It is important to understand that all four serial interface standards allow different baud or data-transmission rates. This is significant because a serial interface transmits data sequentially, one bit after another, as opposed to a parallel interface, which for example might transmit one byte (eight bits) at a time.

In a serial interface the receiving and transmitting devices ***must*** operate at the same baud rate. The SMINI and the SUSIC are designed to operate with standard baud rates of 9600, 19200, 28800, 57600 and 115200 bits per second or bps. This range is considerably faster than the standard rates of 150, 300, 600, 1200, 2400, 4800, 9600 and 19200 provided with the original Classic USIC. Thus, combining nodes using the Classic USIC with the newer SMINI and SUSIC nodes requires operating at either the 9600 or 19200 baud rates – the rates where the two ranges overlap. At this point, let’s just focus in on the SMINI.

SUPER MINI-NODE INTERFACE CARD (SMINI)

Fig. 4-2 shows the parts layout for the SMINI card. It is a complete self contained node, being sort of a gee-wiz do-all type of card. I find it easiest to think of the SMINI card as being four cards combined into one, an SUSIC combined with two 24-bit output cards and one 24-bit input card.

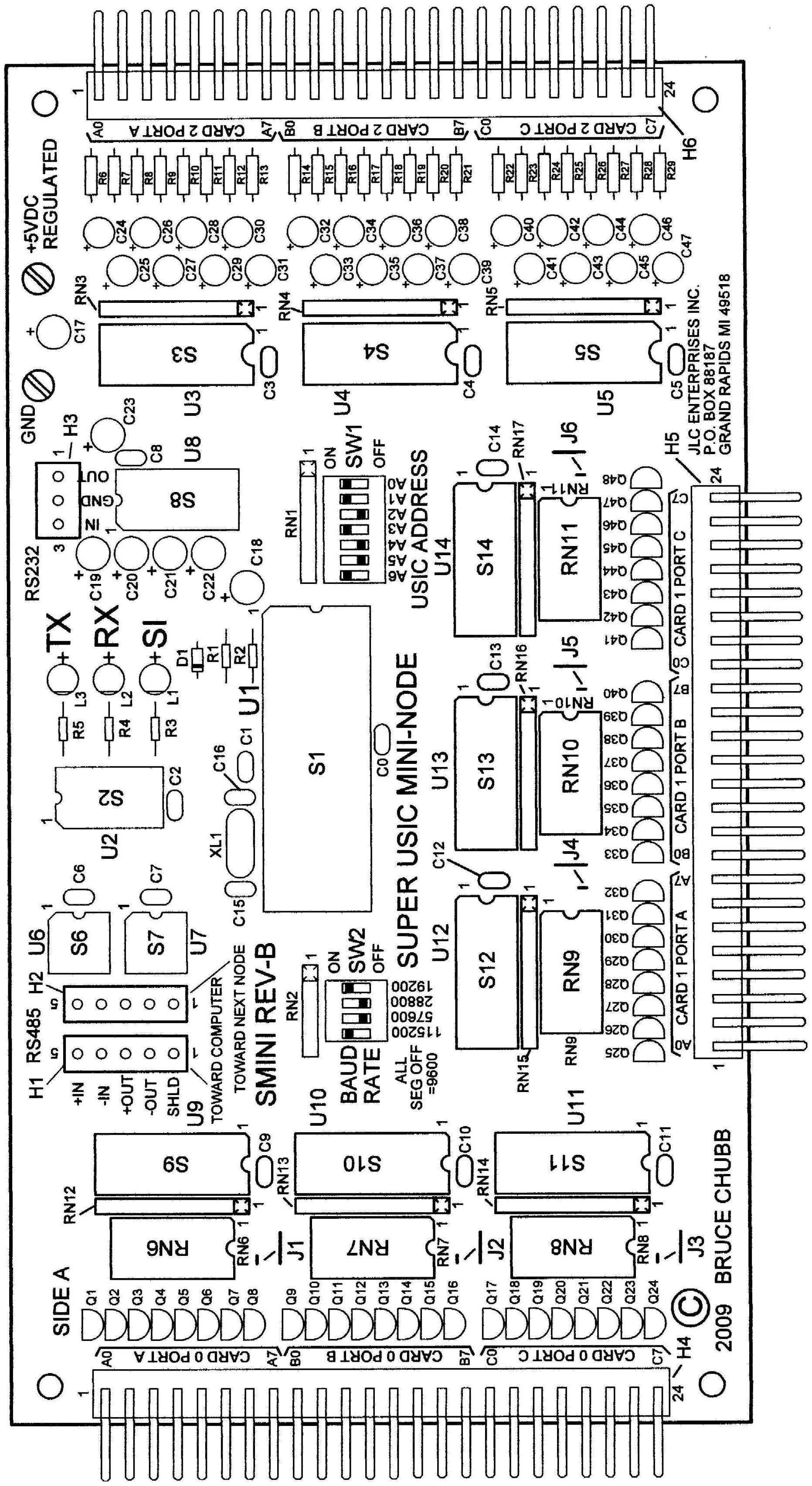


Fig. 4-2. SMINI parts layout

I will refer to these “cards within a card” as “Cards 0 and 1” for providing the 48-outputs and as “Card 2” for providing the 24-inputs. You will find these nomenclatures used along the left, bottom and right edges of the SMINI card.**[1]**

When using RS232, the serial connections between the computer and SMINI are made using the 3-pin header along the top right edge of the card. Two 5-pin headers are provided for use with RS485. The left header connects with the cable running back toward the computer and the right header connects with the cable running to the next node. This setup facilitates easy daisy-chaining from node to node. The fifth, or bottom pin in each RS485 header, is provided to maintain continuity for the shield normally used with RS485 cabling.

Because there can be up to 128 SMINI cards connected on the same 4-wire cable, the system must be able to discern between which SMINI is involved with communications at any point in time. This is achieved by giving each SMINI a unique address, 0 through 127, set by using the 7-segment address DIP switch SW1. These address settings follow the powers-of-two binary number scheme we established in Chapter 2 and the resulting DIP switch settings for typical card numbers are illustrated in Fig. 2-16.

The 4-segment DIP switch, SW2, sets the baud rate with all segments off corresponding to 9600 bps. To achieve each of the other 4 baud rates, the corresponding segment is set to the on, or up, position. If more than one segment happens to be turned on, the software within the SMINI defaults to using the baud rate corresponding to the highest switch segment with an “on” setting.

Connections between the SMINI and the railroad are made using Molex-style right-angle header connectors. These connectors, used on all previous C/MRI board designs, have proven to be extremely rugged, dependable and cost effective. The SMINI includes circuitry for two 24-pin output cards and one 24-pin input card.

Although all the I/O lines are physically on the same SMINI card it is convenient to talk about them as being separate (I/O) cards within the same (SMINI) card. For example, the two sets of 24-pin outputs are noted as Cards 0 and 1 located along the left and bottom edge of the card. The 24-pin inputs are noted as Card 2 along the right edge. Each 24-pin grouping is subdivided into three ports labeled A, B and C. The 8-pins within the different ports are defined as A0 through A7, B0 through B7 and C0 through C7.

Each railroad device being interfaced simply connects directly to one or more of these pins. In “computer-ese” we refer to each pin, or line, as a bit and 8 bits become a byte. Thus each port handles 8 bits, or 1 byte of data, either input or output. The use of the terms bit and byte will become second nature as you begin to put the interface to use.

Power connections, +5Vdc and ground, use the two screw terminals at the top right corner of the card. The supply must be regulated with a tolerance range of +4.75Vdc to +5.25Vdc. Most surplus computer power supplies easily meet this requirement and are typically available for zero or near zero cost.

The heart of the SMINI is part U1, a Microchip Technology PIC16F877-20/P Microcontroller with built-in FLASH memory, making it a **smart interface**. PIC processors are the most popular family of microcontrollers and the 877 is ideally suited for our interfacing needs. This very capable Microcontroller IC is a single 40-pin

[1] Note: The first SMINI production run cards labeled these “cards within a card” as 1, 2 and 3 versus the 0, 1 and 2 description provided in this manual. If you happen to have one of these early SMINI cards simply ignore the 1, 2 and 3 printing on the card and treat them as if they read 0, 1 and 2. See section ***Counting Cards/Nodes Using Number Zero*** in Chapter 2 for change in approach.

DIP containing a Micro-Processor Unit (MPU), 8K 14-bit words of FLASH Program Memory, 368 bytes of RAM Data Memory, 256 bytes of EEPROM data memory, a Serial Communications Interface (SCI), built-in baud generator, three programmable timers, and an abundance of I/O pins. It is a very rugged static protected design with built-in TTL compatibility including high sink/source current capability of 25mA per I/O line. These superior capabilities give the SMINI lots of power in a very rugged package.

The main advantage of a smart serial interface with built-in MPU and memory is great flexibility. The same design works whether we are using it plugged into the SMINI or the SUSIC and whether we are using RS232, RS422 or RS485 I/O. For RS232 the interface uses only the Pin-2 and Pin-3 data lines and the Pin-5 or Pin-7 ground lines, so it is independent of the sometimes unpredictable RS232 handshaking lines.

The PIC16F877 performs all the bookkeeping as to which particular bit the computer is sending and whether it is part of an address or data. When the computer is to read data from the SMINI input ports, it simply tells the PIC16F877, which gathers the data from the 3 input ports, sets it up in the prescribed format, and transmits it bit-by-bit back to the computer.

The three LEDs, L1-L3, indicate the operational status of the SMINI. L1 (green) blinks to show that U1’s internal program is operating correctly and also provides error codes when not operating correctly. The error codes have been significantly expanded over those provided with the original Classic USIC. LED L2 (amber) blinks when the SMINI is receiving data from the PC and L3 (red) blinks when the SMINI is sending data back to the PC.

Each of the 24 SMINI input lines feature optional input line filtering for maximizing immunity to electrical noise frequently encountered with pulse-type command control systems such as DCC and Railcommand. Each of the SMINI output ports is configurable for standard current-sinking or alternate current-sourcing. The majority of C/MRI applications use current-sinking whereby railroad devices are activated, i.e. turned on, by the SMINI completing the ground connection. Alternatively, with current-sourcing the SMINI provides the actual voltage that activates the railroad device. Current-sourcing is mainly used when driving 3-lead searchlight signal LEDs or for other special applications where railroad devices share a common ground connection. I will make use of both arrangements in later chapters.

Programming the PIC16F877 FLASH memory requires a special PIC Micro Programmer package. As a bare minimum the package needs to include a programming editor and an assembler that turns the user written assembly language code into 1s and 0s for loading into the PIC16F877 Microcontroller. A big advantage of the 877’s FLASH memory is that the ultraviolet erase cycle is not required as it was for the MC68701, used with the previous USIC. With the PIC16F877 everything is handled electronically.

Do not worry about having to do any PIC programming. It is provided pre-programmed and tested in a working SUSIC from JLC Enterprises. The cost is $34. Besides being more than twenty times faster and much more powerful than the MC68701 used in the original design USIC, the programmed PIC16F877 is about half the cost.

You do not need to understand electronics to build the SMINI. Circuit cards and programmed PIC16F877 chips are readily available from JLC Enterprises; all you have to do is follow my instructions. If you prefer, all the circuit boards, including the SMINI, can be purchased fully assembled and tested or as complete kits from SLIQ Electronics. If you are interested in how the SMINI works, and in debugging it in case of trouble, I will explain what the SMINI does and give you a tour of its schematic. If you really just want to get started, skip ahead to ***SMINI Parts***. If you have purchased your SMINI already assembled and tested, you may skip to ***Computer Connections***.

SMINI FUNCTIONS

Fig. 4-3 is a functional schematic illustrating what the SMINI does.

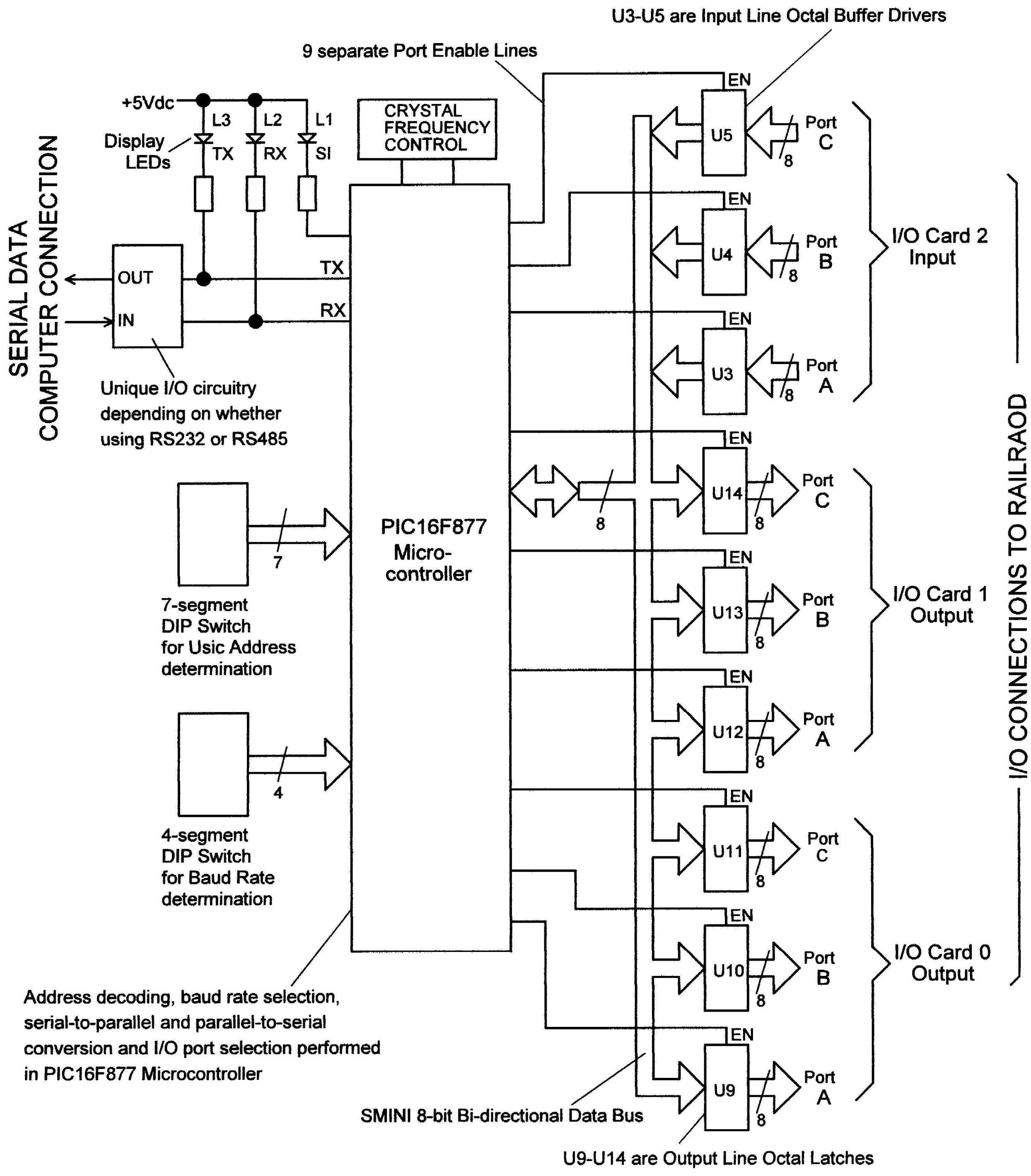

U3-U5 are Input Line Octal Buffer Drivers
9 separate Port Enable Lines
+5Vdc
L3
L2
L1
Display LEDs
TX
RX
SI
CRYSTAL FREQUENCY CONTROL
SERIAL DATA COMPUTER CONNECTION
OUT
IN
TX
RX
Unique I/O circuitry depending on whether using RS232 or RS485
EN
U5
U4
U3
U14
U13
U12
U11
U10
U9
Port C
Port B
Port A
8
I/O Card 2 Input
I/O Card 1 Output
I/O Card 0 Output
I/O CONNECTIONS TO RAILRAOD
PIC16F877 Micro-controller
7
7-segment DIP Switch for Usic Address determination
4
4-segment DIP Switch for Baud Rate determination
Address decoding, baud rate selection, serial-to-parallel and parallel-to-serial conversion and I/O port selection performed in PIC16F877 Microcontroller
SMINI 8-bit Bi-directional Data Bus
U9-U14 are Output Line Octal Latches

Fig. 4-3. SMINI functional diagram

Its main hardware functions are address decoding, baud rate generation, parallel-to-serial and serial-to-parallel conversion, input/output port selection, input port buffering and output port latching.

Most functions are handled directly by the PIC16F877 including all the parallel-to-serial and serial-to-parallel conversions and determining the operational timings for both the serial and parallel lines. It also keeps track of which information is data and which is address and determines when to activate each of the port enable lines for communication with the SMINI's 9 different I/O ports – 3 for inputs and 6 for outputs. The arrows in Fig. 4-3 indicate the direction of signal flow, and double lines with a slash and number indicate multiple parallel wires – four, seven or eight in our case.

The SMINI has six 8-bit output ports labeled as Ports A, B and C for "Cards 0 and 1." Similarly, the three 8-bit input ports are labeled as Port A, B and C for "Card 2." Each I/O port includes an important buffer between the railroad's connection to the SMINI card and the PIC16F877. The functional blocks, noted as U9 through U13, perform as buffer/line-drivers for the 6 output ports. Also, these blocks provide a latching function to keep the output data constant during the period between output data updates. Blocks U3 through U5 perform as the input data buffers.

A single 8-wire I/O data bus connects the PIC16F877 to each of the 9 I/O buffering functions. There are also 9 separate I/O control wires – typically called port select or enable lines – joining the 877 to each of the 9 I/O buffering functions. One of these port select lines connects directly to the enable (EN) inputs for each of the port I/O buffer functions. By selectively activating only one of these 9 lines at any one time, the 16F877 sets up direct communication to and from the desired port via the single function 8-wire bi-directional I/O data bus.

For example, if the 877 wants to read data from Input Port A on Card 2 it activates the port enable line connected to the enable input on Input Buffer U3. Correspondingly, U3 transfers the input railroad data, via U3's 8 input pins to the 8-wire data bus to be read into the 877.

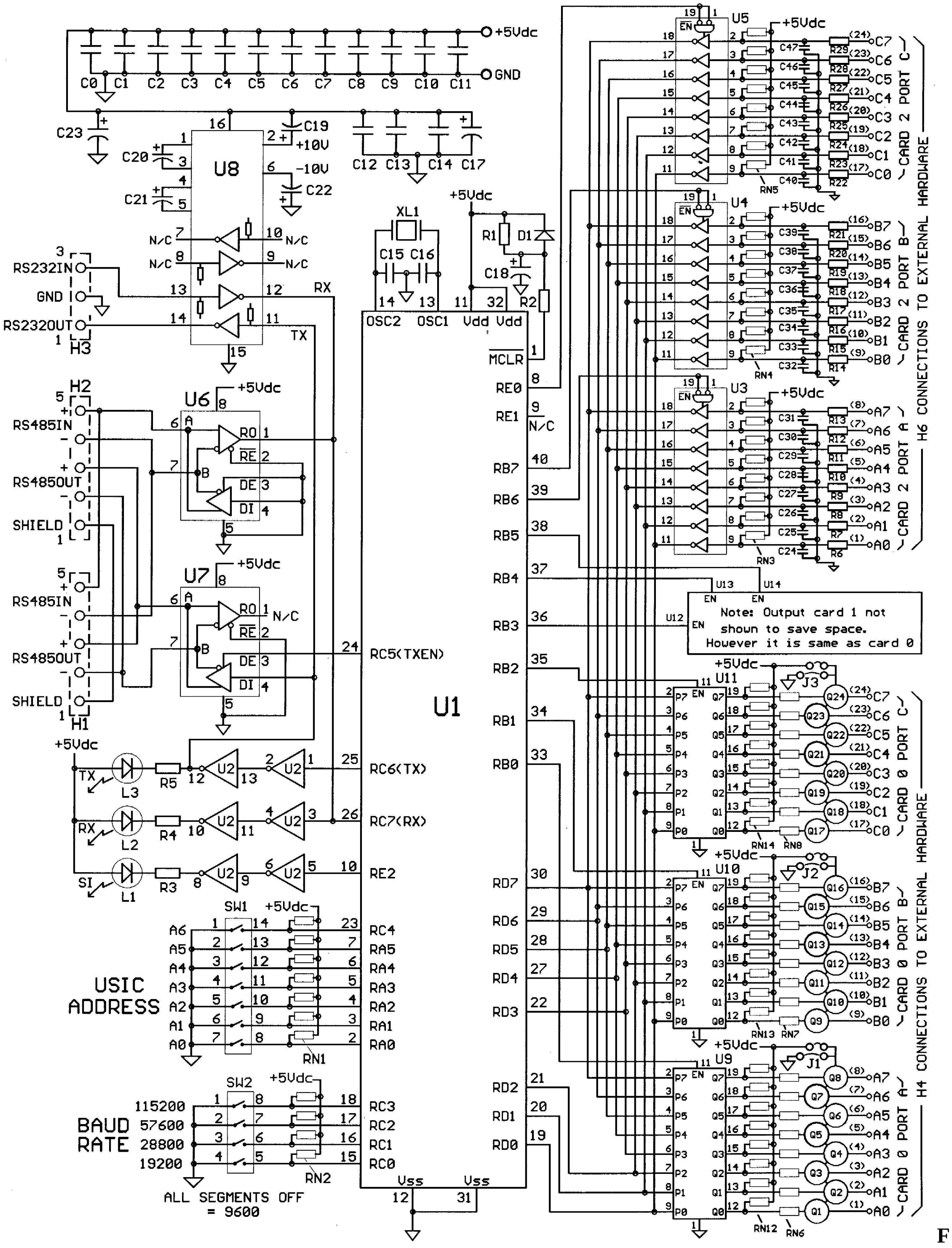
For SMINI outputs, say Output Port C on Card 0, the PIC16F877 places the desired output data on the 8-wire I/O data bus and then activates the port enable line connected to the enable input on Output Buffer U11. Correspondingly, U11 takes the data found on the I/O data bus and transfers (and latches) it to the correct 8 output pins connected to the railroad.

On the serial side of the SMINI, one of two special I/O circuits couples the PIC16F877 to the serial I/O lines coming to SMINI from the main computer, the PC. Which circuit is used depends on whether you are using RS232, or RS485 (RS422) I/O. The basic function is the same, however, to buffer and convert the serial signal levels to be compatible with the +5Vdc and 0V logic levels used by the PIC16F877's Transmit (TX) and Receive (RX) lines.

To place multiple SMINI cards on the same RS485 4-wire cable, we must be able to set a unique SMINI Address, UA, for each card. This is handled by the 7-segment DIP switch feeding directly into the PIC16F877. The 7-segment switch can set 128 unique addresses, allowing up to 128 SMINI cards in an RS485 system. The 4-segment DIP switch feeding directly into the PIC16F877 sets the baud rate generator built into the 877 to the desired serial transmission rate.

SMINI SCHEMATIC

The schematic in Fig. 4-4 is simply an expansion of the functional diagram in Fig. 4-3. It looks complicated but breaking it down into its various components and visualizing that many wires run in parallel to identical devices performing the same function, it becomes quite easy to understand.



F
ig. 4-4. SMINI schematic

Everything focuses on U1, the PIC16F877, so I will cover it first. Of its 40 pins, 33 are I/O lines, configured in three 8-bit ports, one 6-bit port and one 3-bit port. Software within the 877 can configure each line of each port as either input or output.

Table 4-2 summarizes the utilization of the 877's I/O port lines for the SMINI application. Seven input lines, RA0-RA5 and RC4, are connected to DIP Switch SW1 for reading the SMINI's node address, UA. These input lines have pull-up resistors, built into resistor network RN1, so each line is held high, +5Vdc, unless the corresponding segment of SW1 is closed, switching the line to ground.

Table 4-2. PIC16F877 port utilization for SMINI application

Port symbol	Number of bits	Bit/line identifier(s) and function within SMINI
RA	6	**RA0 – RA5** used as inputs to read in the SMINI Address DIP switch segments A0 – A5.
RB	8	**RB0 – RB5** used as outputs to drive the 6 latch enable lines for separately activating each of the 6 SMINI output ports. **RB6 – RB7** used as outputs to drive 2 of the 3 input buffer enable lines for separately activating 2 of the 3 SMINI input ports.
RC	8	**RC0 – RC3** used as inputs to read in the SMINI baud rate DIP switch segments. **RC4** used as input to read in the SMINI address DIP switch segment A6. **RC5** used as output to drive the transmitter enable line for enabling the RS485 transmissions from SMINI back to the PC. **RC6** used as output for sending serial data back to the PC. **RC7** used as input for reading in serial data coming from the PC.
RD	8	**RD0-RD7** used bi-directionally as inputs to read the 8-bit parallel data coming in from the 3 SMINI input ports and as outputs to send the 8-bit parallel data going to the 6 SMINI output ports.
RE	3	**RE0** used as output to drive the 3rd SMINI input port enable. **RE1** spare – not used for SMINI application. **RE2** used as output to drive the green SMINI status indicator LED.

Four input lines, RC0-RC3, connected to DIP Switch SW2, set the SMINI's baud rate. Pull-up resistor network RN2 keeps each line held high, +5Vdc, unless the corresponding segment of SW2 is closed, switching the line to ground.

U6 and U7 are identical RS485 transceiver ICs capable of transmitting and receiving over the same 2-wire transmission line to support half-duplex operation – two wires transmitting bi-directionally. The C/MRI is set up for full-duplex operation – two wires for transmitting from the PC and two wires for reception by the PC. Thus, two ICs are used where U6 is set up to receive data from the PC and U7 is set up to transmit data to the PC. This provides easier system operation and debugging capability.

U8, a dual RS232 transmitter and receiver IC, handles communication in both directions when using RS232. Three wires are used with RS232. These are signals from the PC, signals to the PC and ground.

Line RC7, Pin 26, is U1's serial data input Receive (RX) line, while RC6, Pin 25, is the serial data output Transmit (TX) line. These connect U1 to the SMINI's interface-unique circuitry for either RS232, or RS485 I/O. If you are using RS232 you simply install U8, or for RS485 you install U6 and U7. You should **NEVER** have U6 and U7 installed if you have U8 installed and vice versa. If you do, then U6 and U8 both fight to control the RX line resulting in faulty operation and possible IC damage.

Looking at RC5, Pin 24, the Transmit Enable (TXEN) line, this is a little more complicated. To support a distributed serial system, multiple SMINI receiver and transmitter ICs, U6 and U7, are connected in parallel on the same two sets of two wires leading back to the PC. Fig. 4-5 illustrates the situation.

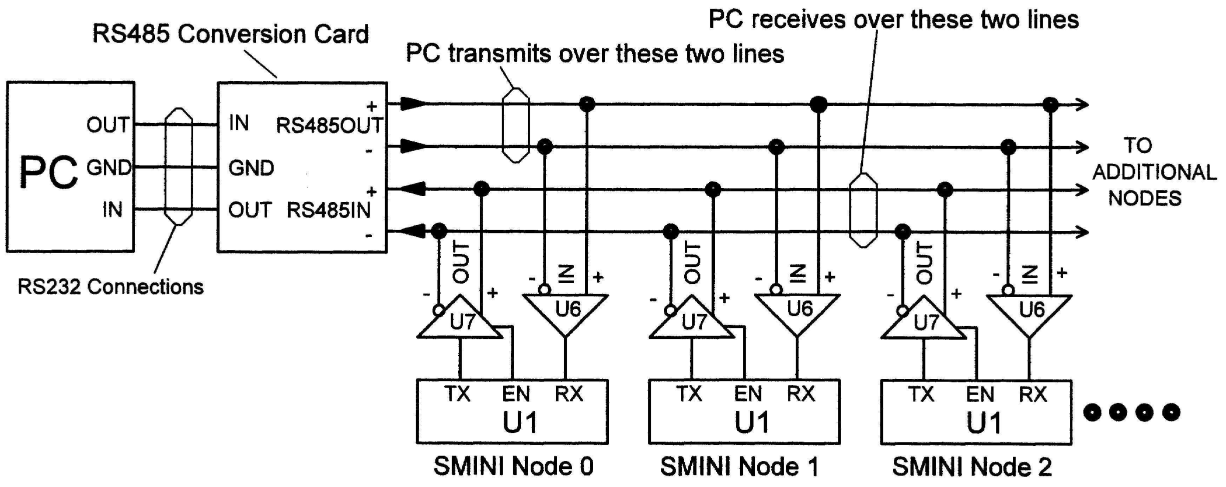


Fig. 4-5. Multiple RS485 transmitter and receiver ICs connected in parallel

The arrows indicate the direction of data transmission for each two wire set. Two wires are used to send data from the PC to the SMINI cards and two wires are used to send data from the SMINI cards to the PC.

When the PC transmits data, every SMINI card listens for its address. Having multiple SMINI cards listen is not a problem because:

- Only one device is ever talking, which is the PC via the RS485 card. This way only one device, the driver IC on the RS485 card, is in command of the two transmit lines.

- The receiving device on each SMINI, the U6 receiver IC, has a high input impedance corresponding to a ¼ unit load. The driver IC on the RS485 card is capable of driving 32 unit loads or 128 of the ¼ load devices. That is from where the limit of 128 nodes maximum per distributed system arises.

When it comes to the PC receiving data from the SMINI cards, the situation is more complex. Only one SMINI, as determined by the PC, may be permitted to communicate back to the PC at any point in time. When an SMINI is talking to the PC, the SMINI card's U7 transmit IC is in effect operating in a low impedance state driving the two bus lines with '1s' and '0s'. For this to happen without interference, the outputs from the U7 transmit ICs on all the other SMINI cards must be held in a high impedance state, effectively an open circuit to prevent loading down the bus.

This means that only the U7 transmitter on one particular SMINI can be permitted to be active at a time, otherwise there would be more than one U7 trying to pull the transmission line high or low at the same time resulting in a confused state. Such undesired action – called bus contention – must be avoided. For the system to work, only the transmitter IC from one SMINI can be allowed to seize, i.e. transmit over, the bus at any one time and all other SMINI cards must have their U7 ICs held in their high impedance (open circuited) states. This is accomplished by using RC5, Pin 24, as a transmit enable line connected to U7's driver enable (DE) input.

When the C/MRI application software, resident in the PC, requests that a given SMINI node send the railroad data to be received as inputs on its 3 input ports, the U1 gathers the data, converts it to serial format, then activates U7's DE enable line, which then seizes the serial bus for sending the input data to the PC. Because none of the other SMINI cards were addressed, their corresponding U7's are held in their high impedance tri-state, effectively open circuited, to prevent conflict with the operation of the addressed SMINI.

The 8-lines of U1’s Port RD (Pins 19-22 and 27-30) are the bi-directional I/O data bus connecting U1 to the SMINI’s 3 input ports and 6 output ports. When U1 needs to read data from an input port, U1 software switches the eight Port RD lines to be inputs. When U1 needs to send data to an output port the U1 software switches the eight Port RD lines to be outputs.

Lines RB0-RB7 and RE0 perform as port enable lines to define which I/O port IC (U3-U5 for input or U9-U14 for output) is activated for each input and output operation over the I/O data bus connecting all the ICs to U1’s Port RD. For example, if U1 needs to write to I/O Card 0 Port A, it first sets Port RD to be output then it places the data for output on the Port RD lines. Then it pulls line RB0, Pin 33, high to activate, or enable, U9. With U9 enabled it transfers the data from the Port RD lines, the I/O bus, to the 8-output pins associated with Card 0 Port A that are connected to the railroad. As soon as U1 pulls the U9 enable line low, U9 latches the output data so that it stays constant until the next output data update.

Likewise, when U1 needs to read from I/O Card 2 Port C, it first sets Port RD to be input then it pulls line RE0, Pin 8, low to activate, or enable U5 and then it reads the input data that U5 places on the Port RD lines.

Each of the port latch outputs from U9 through U14 pass through a transistor, Q1 through Q48, before reaching the SMINI card’s 6 output ports. These transistors, typically 2N4401 NPN transistors, operate in an open collector configuration to provide a sinking capability of .25A for loads up to 40Vdc. Railroad loads such as lamps, LEDs, relays, and switch motors are turned on and off by your PC software turning each output transistor on (transistor conducting with collector connected to ground) or off (transistor open circuited). By substituting a 2N4403 PNP transistor and changing card jumper locations it is also possible to obtain a current-sourcing output configuration which I will cover in the next section.

The R1, R2, D1 and C18 circuit ensures that the U1 Master Clear (active low) line is held low during the power-up sequence. This insures correct reset operation when power is cycled to the SMINI. Including the diode provides an ultra fast reset to minimize the wait time required when cycling power.

The three LEDs, L1-L3, indicate the operational status of the SMINI. L1 (green) blinks to show that U1's internal program is operating correctly and also provides error codes when the SMINI is not operating correctly. These error codes are significantly expanded over those supplied with the original USIC. L2 (yellow) blinks when the SMINI is receiving data and L3 (red) blinks when the SMINI is sending data.

OPTIONAL OUTPUT CONFIGURATIONS – SMINI CARD

The vast majority of C/MRI outputs make use of standard current-sinking, where the transistor switch, built within every C/MRI output line, completes a connection to ground to activate the connected railroad device. However for special cases that may arise, it is easy to vary the output configuration to handle the alternative current-sinking case. That is where the C/MRI’s transistor switch sources the voltage to activate the load by completing a connection to +5Vdc. Each SMINI output port is separately configurable by selecting the appropriate output transistor and installing the appropriate jumper.

Typically the alternative current-sourcing option is used only when driving:

- 3-lead searchlight signal LEDs
- Color light signals that are prewired with a common ground connection

If you do not have either one of these situations you can simply skip ahead to ***Input Line Filtering***.

Fig. 4-6 shows a sub-schematic detailing the latch and output transistor portion of the SMINI card, configured for both the standard current-sinking and optional current-sourcing. Only one output line is illustrated because all are identical. For illustration I selected the line associated with transistor Q1.

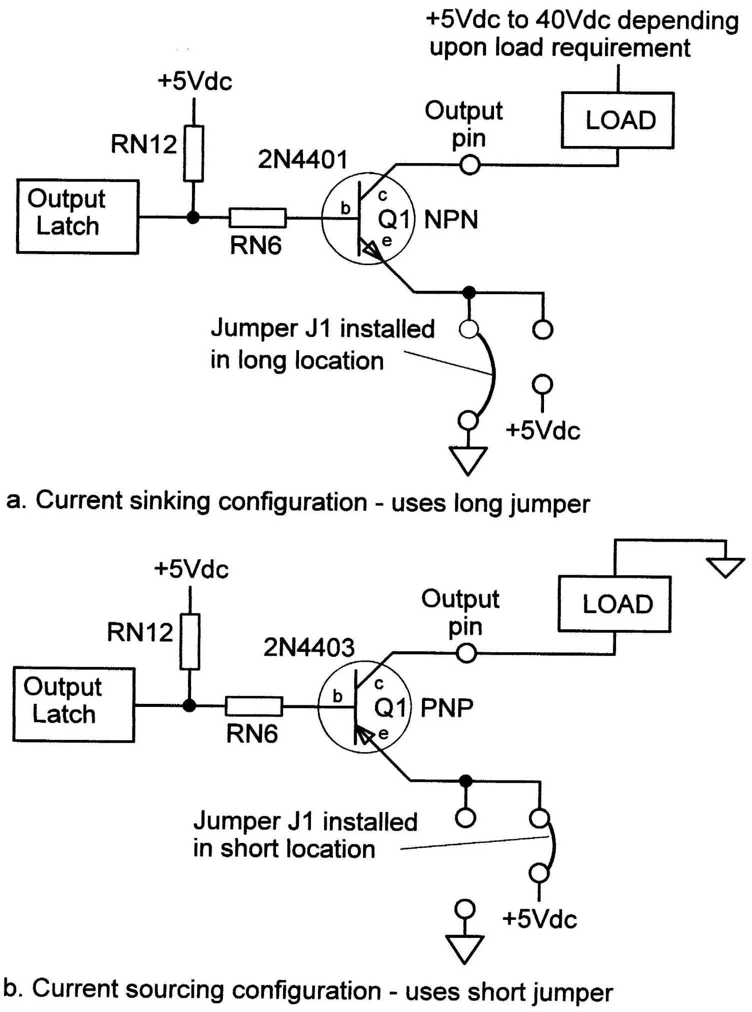

+5Vdc to 40Vdc depending upon load requirement
+5Vdc
RN12
Output pin
LOAD
2N4401
Output Latch
b
c
Q1
NPN
RN6
e
Jumper J1 installed in long location
+5Vdc
a. Current sinking configuration - uses long jumper
+5Vdc
RN12
Output pin
LOAD
2N4403
Output Latch
b
c
Q1
PNP
RN6
e
Jumper J1 installed in short location
+5Vdc
b. Current sourcing configuration - uses short jumper

Fig. 4-6. Current-sinking and current-sourcing configurations – SMINI

I will cover the standard current-sinking case first, which uses the 2N4401 NPN output transistor and the port configuration jumper installed in the long position, as shown in Fig. 4-6a. When the output of the latch goes high, the pull-up resistor RN12 pulls the output of the latch to +5Vdc. That causes Q1 to conduct and activate the load. When the latch output is low, Q1 is turned off and the load is de-activated. Software is used to define the latch output, and therefore we have total control of the load's operation.

Circuit designers will be quick to observe that RN12 is not really required because the high output of the latch would turn on Q1 independent of having RN12. However, I have included RN12 for two reasons: it is required for the current-sourcing option and keeping it in place for the current-sinking option assists debugging by allowing easy determination of whether an output line failure is within the latch or the output transistor.

The current-sinking output format should work for most applications. It performs as if the computer interface were a large collection of toggle switches, where one side of each switch is connected to ground and the other side is available for connecting to external hardware. Each switch is separately controllable by software giving you total freedom to turn each circuit on and off as desired. Current-sinking is the preferred method of using digital electronics to drive external hardware.

However, you may find situations where you need to connect the interface to external hardware that is prewired with a common ground and what you need to do is switch the hot, or power side, of the loads. These situations can be handled by replacing the 2N4401 NPN transistor with a 2N4403 PNP, and installing the port configuration jumper in the short position as illustrated in Fig. 4-6b.

In this current-sourcing case, Q1's emitter is connected to +5Vdc and the logic between the software and the load is reversed. When the latch output goes high, RN12 insures that the base of the transistor is pulled up to +5Vdc, turning the transistor off to deactivate the load. When the output of the latch goes low the base of the 2N4403 transistor falls below the emitter, causing the transistor to conduct to activate the load. Including the pull-up resistor is mandatory for the current-sourcing case to insure that the base voltage of the transistor is pulled sufficiently high when the latch is high so that the transistor is turned fully off.

Table 4-3 summarizes the logical relationship between latch output and the connected railroad device, both when it has been turned on (activated) and turned off (deactivated). The situation is exactly reversed when using alternative current-sourcing rather than standard current-sinking.

Table 4-3. Operational logic for current-sinking versus current-sourcing

Mode of Operation	State of Latch Output	Transistor Status	Railroad Device Status (load)
Current-sinking	High	On	Activated
Current-sinking	Low	Off	Deactivated
Current-sourcing	High	Off	Deactivated
Current-sourcing	Low	On	Activated

With current-sinking, having the latch high turns on the load while with current-sourcing having the latch high turns off the load. Because your application software is controlling the latch output, it is easy simply to reverse the software outputs when driving current-sourcing outputs. Using this approach, all the other logic in the application program remains unchanged. I will follow this procedure in all of our application program examples.

The 2N4403 PNP transistor provides a drive capability of .3A. In the current-sourcing case the sum of all the load currents pass through the smaller +5Vdc card traces, whereas the larger ground traces are used in the current sinking case. It can therefore become necessary to limit the size and the number of loads that are turned on simultaneously when using current-sourcing. However, for railroading applications, current-sourcing is primarily used for driving 3-lead searchlight signal bicolor LEDs. Their maximum current draw, corresponding to displaying a yellow aspect with red and green both turned on, is only about .03A per signal. Current loading in this case is not much of an issue. The current-sinking design is more tolerant of driving heavier loads since the ground traces are wider and there are three ground pins per I/O card. Also, current-sourcing with the SMINI is limited to driving 5Vdc loads while current-sinking can drive loads up to 40Vdc.

INPUT LINE FILTERING – SMINI CARD

The SMINI card is designed so that each input line is wired from the external hardware directly into the appropriate pin on input buffers U3-U5. This works well for most applications, and is how the original input cards (the CIN24 cards using the 8255) function. However, the interface input lines connected from many hardware applications contain extensive electrical noise. Adding a noise filter to each input line can reduce the effects of this noise.

It is my personal experience that few applications require input line filtering. However, by contrast, there are many users that think input line filtering is one of the C/MRI's greater assets. If you are unsure of the level of noise filtering you need, or for that matter if you need it at all, then a good approach is to assemble the SMINI cards without installing R6-R29 and C24-C47. Also, this reduces parts cost and reduces assembly time. Then later, if you find it desirable to add filtering, it is a simple task to cut the trace on the bottom side of the board and insert the desired parts. Thus, if you are not currently interested in implementing input line filtering, simply skip ahead to ***SMINI Parts***.

Adding a low-pass RC (*R*esistor-*C*apacitor) filter to each input line is easy with SMINI. Simply cut the narrower circuit trace located on the bottom side of the card under each of the resistor locations, and install the corresponding R6-R29 resistors and C24-C47 capacitors. The R and C combination is called a low-pass filter because it lets DC and low-frequency signals pass through with practically zero attenuation while attenuating high-frequency signals.

The product of the resistor and capacitor values (i.e. R x C), is called the *filter time constant*. The larger is this constant the greater is the filtering. The filter's cut-off frequency (in units of cycles/second, or Hz), is calculated as 1/(6.28 x R x C), where R is expressed in ohms and C is expressed in farads. AC signals or noise above the cut-off frequency are severely attenuated, while signals below this frequency pass through with little attenuation.

The big question is, "How do we go about selecting the best values for R and C?" Well, first off the resistor should never be greater than about 100Ω. Larger values cause the input voltage into U3-U5 representing a logic low to rise to a point where it is too close to the .8Vdc switching threshold of the ICs, thereby actually increasing noise susceptibility.

Using less resistance is not good, because lower values do not adequately limit current surges resulting from the capacitor being discharged when the input line is switched to ground by the external hardware. Conclusion, fix the resistor at 100Ω and then adjust C to achieve your desired filtering level.

The maximum capacitor that fits on the card is 470μF, 10Vdc (Digi-Key P6217) which when combined with the 100Ω resistor produces a filter time constant of .047 seconds (calculated as .000470 x 100) or 47ms, and a cut-off frequency of 3.4Hz. This means that the noise would have to hold the input in the incorrect state for approximately 47ms before the software would receive an incorrect input. With the 3.4Hz cutoff, the input line would be very sluggish in responding to valid changes in input. This is extremely heavy filtering and probably more than needed for even the most severe situations.

Balancing the amount of filtering versus how fast you desire to track valid input changes is important. For example, using QuickBASIC with my Pentium in data acquisition, I read input bytes 200,000 times per second. If you are interested in such fast tracking of inputs, and you installed the filter time constant of 47ms, it would take about 9,400 samples before you detect an input change, a totally unacceptable situation when seeking high speed data acquisition. Thus, the upper bound on your filter time constant should be about half the time period you expect between reads. In this case the desired maximum capacitor value would be about .02μF, which is quite a difference.

Large capacitors also have a disadvantage by creating longer-lasting surge-currents when card inputs are switched to ground by the external hardware. Using the 100Ω resistor, this current is 5V (the normal charge on the capacitor) divided by 100Ω or 50mA. Therefore, you would not want to drive this level of filtering if you are using TTL circuits in your external hardware to switch input lines to ground. Larger capacitors make these high-level currents present for longer time-periods exacerbating the drive requirement.

Table 4-4 provides examples for a wide selection of filter capacitor values, along with the resulting filter time constant and cut-off frequency. For example, if your requirement does not demand a fast response time for reading inputs, and you desire heavy filtering, I would suggest using the 100Ω resistor with the 10 or 22µF capacitor. You can adjust these values up and down per Table 4-4 as needed, based upon the speed of your program and the filtering level desired.

A frequently used technique is to start with the 100Ω resistor and the smaller size capacitors, and then gradually increase the capacitor size until any noise problem is solved. Select one size larger, for a design margin, and install the capacitors. Remember to always keep your filter time constant under half the time it takes your software to execute one loop, so that you read data in a timely manner. Also, for applications where you desire to filter out high frequency noise, it is better to use tantalum, disk ceramic or monolithic capacitors in place of the frequently lower-priced electrolytic capacitors, which are less effective at the higher frequencies.

Table 4-4. Input filter time constant and capacitor selection

Capacitor Value (µF)	Filter Time Constant	Filter Break Frequency
.0047	.47 µsec	340 kHz
.01	1 µsec	160 kHz
.047	4.7 µsec	34 kHz
.1	10 µsec	16 kHz
.47	47 µsec	3.4 kHz
1	100 µsec	1.6 kHz
4.7	470 µsec	340 Hz
10	1 msec	159 Hz
22	2.2 msec	72 Hz
47	4.7 msec	34 Hz
100	10 msec	16 Hz
470	47 msec	3.4 Hz

Note: Table assumes an R value of 100Ω

In place of using hardware filtering, you can perform software filtering in your application program. The software simply checks to see that the input has remained stable for two or more samplings before it accepts any change in input. A disadvantage of software filtering is that it does require extra processing time, therefore slowing down your real-time loop. Its advantage is that extra hardware is not required and it is easy to vary the effective filter time constant. However, for most C/MRI users, myself included, I do not recommend employing software filtering. Doing so is contrary to my software motto which is: Keep it simple, straightforward and easy to understand!

Any time I find I need to add input line filtering, I would much rather provide it by adding the simple RC hardware components rather than to complicate the software. I just cannot overemphasize the importance of keeping your software **simple, straightforward and easy to understand!**

SMINI PARTS

Because it is possible that the SMINI might be your first C/MRI circuit card assembly, I will describe it in more detail than I will provide for the rest of the C/MRI cards. Ready-to-assemble SMINI circuit boards are available from JLC Enterprises and fully assembled-and-tested boards and complete kits are available from SLIQ Electronics.

For easy reference and assembly, a component overlay is printed on the PC card. However if you have not yet purchased your card, you can refer to Fig. 4-2 for a copy of the parts layout on the top side of the SMINI card. Table 4-5 is the parts list.

Table 4-5. SMINI parts list (in order of recommended assembly)

Qnty.	**Symbol**	**Description**
2	-	4-40 x 1/4" long pan head machine screws (Digi-Key H142)
2	-	4-40 hex nuts (Digi-Key H216)
1	R1	10KΩ resistor [brown-black-orange]
4	R2-R5	330Ω resistors [orange-orange-brown]
1	D1	1A, 100V 1N4002 diode (Jameco 76961 or Mouser 625-1N4002-E3/73)
1	S1	40-pin DIP socket (Jameco 112311)
1	S2	14-pin DIP socket (Jameco 112214)
3	S3-S5	20-pin DIP sockets (Jameco 112248)
6	S9-S14	20-pin DIP sockets (Jameco 112248)
1	RN1	2.2KΩ 7-element SIP resistor network (Mouser 652-4608X-1LF-2.2K)
1	RN2	2.2KΩ 5-element SIP resistor network (Mouser 652-4606X-1LF-2.2K))
3	RN3-RN5	2.2KΩ 9-element SIP resistor networks (Jameco 97893)
6	RN6-RN11	470Ω 8-elemenet DIP resistor networks (Jameco 108581)
6	RN12-RN17	4.7KΩ 9-element SIP resistor networks (Jameco 24660)
1	SW1	7-segment DIP switch (Digi-Key CT2067)
1	SW2	4-segment DIP switch (Digi-Key CT2064)
3	H4-H6	24-pin right angle headers (Mouser 538-26-48-1242)
15	C0-C14	.1μF, 50V monolithic capacitors (Jameco 332672)
2	C15, C16	18pF, 100V monolithic ceramic disk capacitors (Digi-Key 490-9076-3)
2	C17, C18	10μF, 16V tantalum capacitors (Jameco 94060)
48	Q1-Q48	2N4401 NPN small signal transistors (Jameco 38421) for standard current-sinking or 2N4403 PNP small signal transistors (Jameco 38447) when using current-sourcing
6	J1-J6	Jumpers (make from excess resistor leads and install either long jumper for current-sinking or short jumper when using current-sourcing – see text for details)
1	L1	Diffused green T1¾ size LED (Jameco 334086)
1	L2	Diffused yellow T1¾ size LED (Jameco 334108)
1	L3	Diffused red T1¾ size LED (Jameco 333973)
1	XL1	18.432mHz crystal (Digi-Key X179)
1	U1	Microcontroller PIC16F877-20/P with USIC programmed FLASH Memory (JLC U1B)
1	U2	74LS04 hex inverter (Jameco 46316)
3	U3-U5	74LS540 octal buffer/line drivers – inverting (Jameco 47861 or Jameco 250915)
6	U9-U14	74HCT573 octal D-type latched flip-flops (Jameco 45090)
Used only for RS-485 and RS422		
2	S6, S7	8-pin DIP sockets (Jameco 112206)
2	H1, H2	5-pin straight headers (cut from 24-pin straight header - Mouser 538-26-48-1241)
2	U6, U7	RS485 transceivers MAX487CPA (Mouser 700-MAX487CPA)
Used only for RS-232		
1	S8	16-pin DIP socket (Jameco 112222)
5	C19, C23	1.0μF, 35V tantalum capacitors (Jameco 33662)
1	H3	3-pin straight header (cut from 24-pin straight header – Mouser 538-26-48-1241)
1	U8	Dual RS232 transmitter & receiver (Jameco 24811)
Mating connectors for cable to computer, to next node and to external hardware		
1 or 2	—	One 3-pin terminal housing for RS232 or two 5-pin terminal housings for RS485/RS422 (cut from 12-pin terminal housing Mouser 538-09-50-3121)
6	—	12-pin terminal housings (Mouser 538-09-50-3121)
82	—	Crimp terminals (Mouser 538-08-50-0106 for wire sizes 18-20 or 538-08-50-0108 for wire sizes 22-26)

Table 4-5. SMINI Parts List – Continuation

Fixed resistor and most popular alternate capacitors for different levels of filtering (see text)

24	R6-R29	100Ω resistors [brown-black-brown]
24	C24-C47	.1µF, 50V monolithic capacitors (Jameco 332672)
24	C24-C47	1.0µF, 35V tantalum capacitors (Jameco 33662)
24	C24-C47	10µF, 16V tantalum capacitors (Jameco 94060)
24	C24-C47	22µF, 16V radial lead electrolytic capacitors (Digi-Key P6224)
24	C24-C47	47µF, 16V radial lead electrolytic capacitors (Digi-Key P6226)

Note: Lead spacing for capacitors should be between .079”(2mm) and .1”(2.5mm)

Author’s recommendations for suppliers given in parentheses above with part numbers where applicable. Equivalent parts may be substituted. Resistors are ¼W, 5 percent and color codes are given in brackets.

Refer to the parts layout as you select and mount parts. Keep the card oriented the same way as the drawing while you work, to help place each part in its proper holes with correct orientation. Ensure that you are installing the right components. Sometimes the markings are so small you may need a magnifying glass to read them, but extra effort at this stage is well worthwhile. Check resistors against the color codes included in the parts list and if still in doubt use the ohms range on your VOM to verify the required resistance.

Insert all components from the component or A side of the board, the side with the printed component overlay. Do all soldering on the back or B side. For many parts the correct orientation on the card is **extremely** important. These are diode D1, resistor networks RN1 through RN17, DIP switches SW1 and SW2, capacitors C17 through C23, the three LEDs L1-L3, the transistors Q1-Q48, all IC sockets and the ICs themselves. To remind you to check this as you build your SMINI, I have marked the orientation-critical parts with a plus sign in brackets [+] in the instructions below.

Double-check your selection and location of each component **before** soldering it in place. It is a lot easier to remove an incorrect component before you have soldered it to the card! This is especially true with plated-through-holes since as you solder the part in place, the solder flows all the way through the hole.

The basic skill for building the SMINI is PC card soldering. If that is new to you, make doubly sure that you have thoroughly digested the section on ***PC Card Soldering*** in Chapter 1, to acquaint you fully with the tools and techniques of good soldering. I have updated the descriptive material on good soldering practice to include the newly recommended use of special electronic low-flux content solders and improved commercial flux removers. I find using these new products makes life much easier, so if you have not done so already, please reread this important section covering ***PC Card Soldering!***

Note that U1, the PIC16F877, is static-sensitive. Before handling it you should ground your hands by touching them to a large metal object. This helps discharge any static charge on your body that could damage the chip.

SMINI ASSEMBLY

The order of assembly is not critical, but for the sake of having a plan, follow the steps in order and check mark the boxes as you complete each one. I list the common steps first, then the unique steps to configure your SMINI for one of the two interface standards.

Card inspection. Each SMINI card is factory tested for correct continuity between pads and to ensure that no shorts exist between traces that should be isolated. It is still a good idea to look over both sides of your card to make sure that no scratches have occurred during shipment and handling that may result in a short or open circuit. If there is a questionable area, use your VOM (Volt-Ohm Meter: review the ***Test Meters*** section of Chapter 1). You will seldom find a problem in a card that has been factory tested, but if you do find one and correct it first you will save time in later debugging. If you find an open-circuit trace, scrape away the solder mask coating and then solder a small piece of wire across the damaged section. If you find a short circuit, use a knife to scrape away the offending material. Each new card is guaranteed against manufacturing defects. Therefore, if you suspect your new card has a defect, you may send it back to JLC Enterprises for replacement.

Power terminals. Insert 4-40 screws in the +5Vdc and GND connection holes from the top or component side of the card. Add 4-40 hex nuts on the trace side, tighten firmly, and solder the nuts to the circuit pads. Use spade lugs when you attach your power supply wires to these screws for the best heavy duty connection.

R1-R5. Match the color code of each resistor to the parts list. Make 90-degree bends in the leads of each resistor so it is centered between its two holes and the leads just fit. Insert and solder while holding the part flat against the card, then trim its leads flush with the tops of the solder tents on the back side of the card. Additionally, if you are unsure of the resistor values or have difficulty in reading the color coding bands, as might be the case if substituting 1 percent resistors, which have extra bands or because color recognition may not be clear, it is a good idea to use a VOM set to its resistance range to check the resistor values before insertion.

D1[+]. Make 90-degree bends in the leads of the diode so it is centered between its two holes and the leads just fit. Making sure that the banded end of the diode is oriented as shown in the parts layout. Insert and solder while holding the part flat against the card. Then trim its leads flush with the tops of the solder tents on the back side of the board.

S1-S14[+]. To avoid mixing them up, install 40-pin IC socket S1 first, then 14-pin socket S2 followed by 20-pin sockets S3-S5 and S9-S14. To make certain that Pin-1 orientation is correct for each socket, the notch on the end of the socket should correspond to the notch printed on the parts legend. The Pin 1 hole also has the square pad for easy identification. As for any multi-pin part, solder only a couple pins first, i.e. on opposite corners of each socket. Reheat as necessary to make certain that the socket is firmly against the board. Then solder the remaining pins.

RN1-RN5[+] and RN12-RN17[+]. Install these SIP (Single In-line Package) resistor networks, making sure you have the lead common to each resistor, typically marked with a dot or small vertical line, in the hole with the square pad and marked with the numeral ‘1’. You can also use your VOM to locate the common lead. Solder the two ends first. Once you have checked that the part is firmly against the board, solder the remaining pins.

RN6-RN11. Install these DIP (Dual In-line Package) resistor networks. Orientation is not important but I still like to keep the end-notch on the part lined up with the notch printed on the parts legend. Solder two corner pins first. Once you have checked that the part is firmly against the board, then solder the remaining pins.

Note: The following assembly step can be a frequent source of error. The ON-OFF labeling printed on DIP switches from different manufacturers varies. Some label some of their switches ON when the switch circuit is closed and others are labeled ON when the switch circuit is open. Some switches have their segments numbered from right to left while others are numbered left to right. It is easy for us to totally circumvent all this potential confusion if you simply ignore all the labeling printed on the switches and use your VOM to insure correct switch orientation as described in the following step.

SW1, SW2[+]. Mount the two DIP switches using your VOM to be sure they are oriented so their contacts are closed when the switches are thrown toward the ON label as shown on the parts layout and printed on the SMINI card. Solder two corners first. Check that the part is firmly against the board, then solder the remaining pins.

H4-H6. Install the 24-pin right angle headers and hold them tightly against the card as you solder. It is best to first solder only one pin located near each header end. Then place the board and header over a screwdriver clamped in a vice with the blade protruding up. Lay the board bottom side up with the plastic part of the header resting on the screwdriver blade next to a soldered pin. Reheat the pin while pressing firmly down on the board and the header will tend to "snap" in place. Repeat for the second pin. Repeat the process for a couple more pins if required to get the complete bottom surface of the connector firmly seated. Once verified that the header's bottom surface is pressed flat and tight against the board, then solder the remaining pins.

C0-C14. Insert with capacitor disk standing perpendicular to the card, solder, and trim. These capacitors typically have a number 104 marking otherwise they can appear very similar to C15 and C16. To avoid possible error, ensure that you have the C15 and C16 parts separated out for the next step and that each of the C0-C14 parts have the correct markings before installation.

C15, C16. Insert with capacitor standing perpendicular to the card, solder, and trim. These parts typically have a unique bend in their leads so they stand up a short distance from the card when installed.

C17, C18[+]. Make sure that the + leads (typically the longer of the two leads and marked with a small + sign) go into the + holes as shown in the parts layout and printed on the circuit board. The 'plus' hole also has the square pad for easy identification. Incorrect polarity will damage these capacitors. Solder and trim. These capacitors are different in value from C19-C23 so make certain that you have the correct values installed before you solder and trim.

L1-L3[+]. Install the LEDs with the + leads, typically the longer of the two leads, in the + holes, i.e. those closest to U13 and with the square pad. Check that you have the correct colored LED in the correct position before you solder and trim.

XL1. Hold case firmly against card while soldering leads and trim.

Note: Before installing jumpers J1-J6 and transistors Q1-Q48 you must decide if you want each output port to be set up for standard current-sinking or the alternative current-sourcing. Almost all applications will use standard current-sinking with the 2N4401 transistors. However, you may have a few special applications requiring current-sinking where 2N4403 transistors must be fitted. If you are not sure at this point simply postpone the following two steps until we develop a better understanding of how railroad devices are connected to the SMINI.

J1-J6. Make these jumpers from excess resistor leads and install only one jumper, long or short, at each location. Install only the long jumper for each port being setup for standard current-sinking i.e. using the 2N4401 transistors. Install only the short jumper for each port being setup for the current-sourcing alternative , i.e. using the 2N4403 transistors.

Warning: Each location J1-J6 should have only 1 long or 1 short jumper installed. Installing both long and short jumpers at any given jumper location results in a direct short between the +5Vdc supply and ground.

Q1-Q48[+]. These transistors are in groups of eight per output port. For each port that is configured for standard current-sinking (long jumper installed), install 8 of the 2N4401 transistors. For each port that is configured for alternative current-sourcing (short jumper installed), install 8 of the 2N4403 transistors.

In each case slightly bend the leads of each transistor to fit its holes, and orient each with its flat side facing the direction shown in the parts layout drawing and as printed on the card. The transistor's center, or base, lead needs to be inserted in the hole farthest from the flat side. Push each transistor into its corresponding holes until there is about 3/16 inch of space between the board and the bottom of the transistor. Once all the transistors are lined up evenly, I solder only one lead of each transistor first and then recheck their alignment. I then solder the next lead of every transistor. After again rechecking the alignment, I then trim the leads that have been soldered. Finally I solder and trim the remaining leads.

U2-U5, U9-U14[+]. See Fig. 1-7 for IC insertion and extraction procedures. Be sure you have the ICs specified, that you put them in the right sockets with correct Pin-1 orientation and that all pins go into the socket. The notch in the end of each IC should line up with the notch in the end of the IC socket and the notch printed on the parts legend.

That completes the general SMINI assembly. Next you need only install the parts for the specific interface standard you have chosen, i.e. either RS232 or RS485. Alternatively, you may elect to install the passive components for both interface standards. This makes it easy to switch your interface back and forth between RS232 and RS485 if the need should arise.

Note: If you do install the passive components for both standards, make sure you always leave the ICs out of their respective sockets for the standard not being used.

RS485:

S6, S7[+]. Install these 8-pin IC sockets making certain that the end notches line up for correct Pin-1 orientation. Push each socket tightly against the card and hold it that way as you solder.

H1, H2, Cut off two sets of 5-pins from a 24-pin straight header. Insert and solder following the procedure used for H4-H6 noted above, except that only solder the center pin first.

U6, U7[+]. Be sure you have the ICs specified, that you install them with the notches lined up for correct Pin-1 orientation and that all pins go into the socket. Do not install these ICs if you intend using RS232.

Skip down to ***Input Line Filtering*** unless you are also installing the RS232 components.

RS232:

S8[+]. Install the 16-pin IC socket making certain that the end notches line up for correct Pin-1 orientation. Push the socket tightly against the card and hold it that way as you solder.

C19-C23[+]. Make sure that the + leads go into the + holes as shown in the parts layout drawing and printed on the PC board. Incorrect polarity will damage these capacitors. Solder and trim.

H3. Cut off 3-pins from a 24-pin straight header. Insert and solder following the procedure used for H4-H6 noted above, except that only solder the center pin first.

U8[+]. Be sure you have the IC specified, that you install it with the notches lined up for correct Pin-1 orientation and that all pins go into the socket. Do not install this IC if you intend using RS485.

Input Line Filtering:

Note: I find that many C/MRI applications do not require the use of input line filtering. Some C/MRI Users however swear by it and think its incorporation is an important C/MRI feature. To skip this feature, simply proceed directly to U1 installation. As presented in the text on input line filtering, unless you are sure you want it, I recommend that you first build up the cards without filtering. Then if you decide that filtering is desired you can incorporate it by implementing the next four steps.

To implement input line filtering, cut the narrowed-down traces on the bottom side of the board in the area under the center of each resistor in the locations for R6-R29, choosing only those traces for the lines you require to filter. Use an X-Acto™ knife or a cut-off wheel in a Dremel™-type hand power tool. The cut should be at least 1/32-in wide and need not be any deeper than the trace itself.

Use your VOM to make sure that you have an open circuit across each of the cuts.

R6-R29. Match the color code of each resistor to the parts list and install them as you did for R1-R5.

C24-C47[+]. Install with the value of your choice per Table 4-4 and corresponding text discussion. For those that are polarity sensitive (tantalum and electrolytic) make sure their plus (+) leads go into the plus holes.

U1[+]. The PIC16F877 is a static-sensitive chip that must be handled with care. Avoid unnecessary handling, and keep it protected in its conductive wrapper until ready to insert it. Before touching it, ground your hands by touching a large metal object, to help discharge any static charge on your body that could damage the chip. Make sure you have the correct Pin-1 orientation and that all pins go into the socket.

Cleanup and Inspection. For a professional-looking job and to help ensure your card functions properly, follow the specific steps covered in Chapter 1 regarding cleanup and inspection. This is a vital step so do not shorten it!

That is all there is to assembling an SMINI card.

MAKING COMPUTER CONNECTIONS

Computers vary in their serial I/O capabilities. For example, earlier PCs tended to feature one or two RS232 Serial I/O ports, frequently providing one 9-pin and one 25-pin connector. Later designs, and the majority of the computers sold today, feature multiple USB ports. Generally, computers sold in the transition period retained a single RS232 port, almost exclusively the 9-pin connector, and multiple USB ports. These transitional models tend to be good choices for C/MRI applications. Additionally, connection requirements can vary as a function of whether you are implementing a single node or a multiple node system. With all these options in mind, I'll cover the most common connections for the single node first and then we'll look at the connection options for handling multiple nodes.

Single Node Applications

When you have a single node and your computer has an RS232 Serial Port, a "straight through" RS232 to RS232 cable is the simplest approach. As indicated in Fig. 4-7a, one end needs a standard 9-pin or 25-pin RS232 connector, to mate with the connecter on your computer. The opposite end requires a 3-pin Molex Shell connector to mate with the 3-pin header on the SMINI and SUSIC. Any 3-wire stranded cable will work. However, I typically use 2-pair AWG 22 stranded communication cable leaving one spare wire.

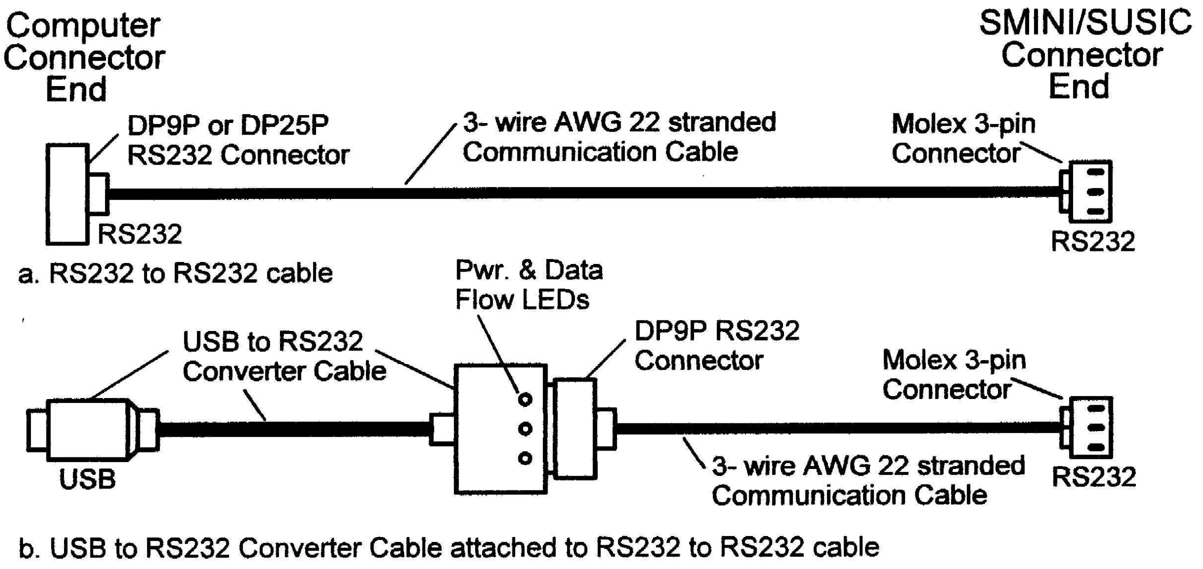

Computer Connector End
SMINI/SUSIC Connector End
DP9P or DP25P RS232 Connector
3- wire AWG 22 stranded Communication Cable
Molex 3-pin Connector
RS232
RS232
a. RS232 to RS232 cable
Pwr. & Data Flow LEDs
USB to RS232 Converter Cable
DP9P RS232 Connector
Molex 3-pin Connector
USB
3- wire AWG 22 stranded Communication Cable
RS232
b. USB to RS232 Converter Cable attached to RS232 to RS232 cable

Fig. 4-7. Connecting computer to SMINI or to SUSIC – single node applications

If your computer provides USB only, then you need to use a USB to RS232 Converter Cable to plug into the computer's USB and use the aforementioned RS232 cable to connect between the converter cable and the 3-pin RS232 header on the SMINI or SUSIC as shown in Fig. 4-7b.

There are a large number of USB to RS232 Converter Cables on the market from which to choose and typical prices range between $15 and $50. Each of these adapters, or converters, contains electronics, and the success rate depends on the capabilities of the electronics and the device driver software that is supplied with the converter to communicate with the electronics over the USB bus. The real challenge is in selecting which converter is best suited to a given application. Sometimes, the adapter that works well for one user's application does not function as well or it functions even better for another user's application. Consequently, which converter is best is a frequent discussion topic on the C/MRI User's Group. The end result, however, is an ever increasing number of C/MRI applications successfully using USB converter cables.

Just as a personal note, based upon the limited testing of three different USB to RS232 Converter Cables, the one we have found that works best for the SVOS application is the Keyspan Model USA-19HS (www.Keyspan.com). However, as we continue to explore other selections we may discover that there is another converter that we like even better.

In contrast to the USB arrangement, the connection in Fig. 4-7a is just about as easy and straightforward as one can achieve. Once determined that you have made use of the correct RS232 pinouts, as defined in Table 4-1, you have a functional connection. Additionally, the setup has minimum software and hardware overhead resulting in maximized real-time system performance.

Multiple Node Applications

Connecting your computer for multi-node applications requires using the SMINI and SUSIC RS485 5-pin headers. Different connection possibilities are summarized in Fig. 4-8.

As illustrated in Fig. 4-8a, the simplest setup for computers having RS232 is to use the JLC provided RS232 to RS485 Converter Card to be covered in detail shortly. Simply use the aforementioned RS232 cable for connecting between your computer's RS232 serial port and the RS485 card. Then use a 4-wire communications cable to connect between the RS485 card and the SMINI or SUSIC.

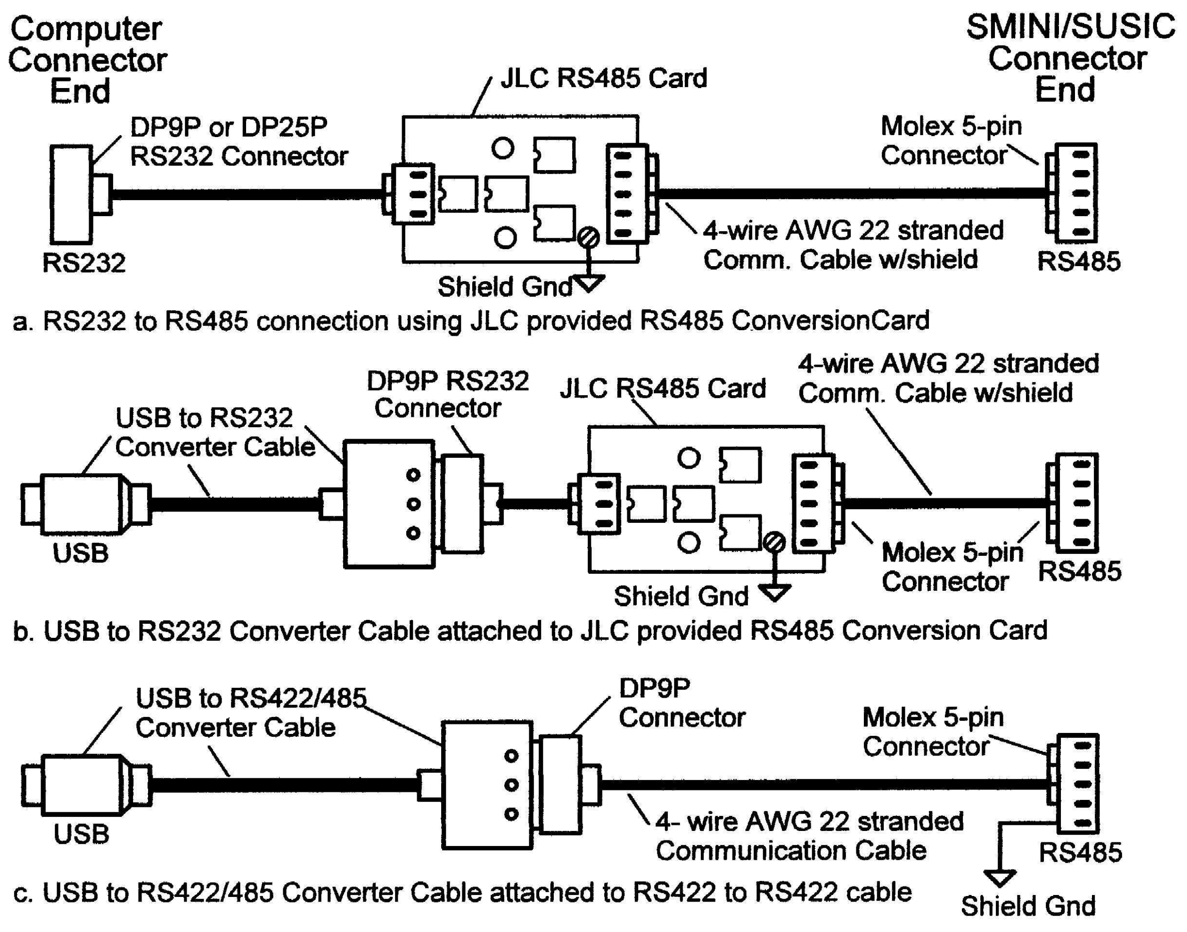

Computer Connector End
SMINI/SUSIC Connector End
JLC RS485 Card
DP9P or DP25P RS232 Connector
Molex 5-pin Connector
4-wire AWG 22 stranded Comm. Cable w/shield
RS232
RS485
Shield Gnd
a. RS232 to RS485 connection using JLC provided RS485 ConversionCard
DP9P RS232 Connector
JLC RS485 Card
4-wire AWG 22 stranded Comm. Cable w/shield
USB to RS232 Converter Cable
USB
Molex 5-pin Connector
RS485
Shield Gnd
b. USB to RS232 Converter Cable attached to JLC provided RS485 Conversion Card
USB to RS422/485 Converter Cable
DP9P Connector
Molex 5-pin Connector
USB
4- wire AWG 22 stranded Communication Cable
RS485
c. USB to RS422/485 Converter Cable attached to RS422 to RS422 cable
Shield Gnd

Fig. 4-8. Connecting computer to SMINI or to SUSIC – multiple node applications

Alternatively, for computers having only USB, you can use the same USB to RS232 Converter Cable as used in the previous subsection followed by the JLC provided RS232 to RS485 Converter Card as illustrated in Fig. 4-8b. However, for applications where you have not acquired the USB to RS232 Converter Cable or the RS232 to RS485 Conversion Card, one option is to purchase a USB to RS422/485 Converter Cable as indicated in Fig. 4-8c.

There are many different USB to RS422/485 Converter Cables on the market. Prices range somewhat higher than their RS232 counterparts but are still typically less than $80. However, a little extra care is required when purchasing USB to RS485 Converter Cables. This is because some suppliers fail to include in their advertising that their USB to RS485 Converter Cable is capable of handling the “Full Duplex” implementation of RS485, i.e. the 4-wire option where separate RS485 transceivers are used for output and input. This 4-wire adaptation is a requirement for use with the C/MRI. Standard RS485 Half Duplex will not work with the C/MRI.

It is important to point out that the multiple node connections shown in Fig. 4-8 are the more general in terms of capability. That is, all the multiple node examples can be used to connect to a single node system. However, the converse is not true; you cannot use either of the single node examples, as illustrated in Fig. 4-7, to connect to a multi-node application. Therefore, if you are starting out using a single node system but have plans to expand later on to multiple nodes, then it is prudent to start by using one of the connection schemes illustrated in Fig. 4-8.

For testing purposes, I selected two different USB to RS485 Converter Cables. Both are from *www.usconverters.com* with model numbers UT890 and U485G. The U485G, being designed for industrial applications, is optoisolated with an approximately 1/3 higher cost. Both units include the desirable data flow display LEDs.

Initial testing proved that both converters worked perfectly when interfacing to a single SUSIC node that I use for testing I/O cards. However, to date, both have failed when connecting to the 7-node, 3800 I/O line C/MRI system used on the SVOS. That is, they initialize and communicate properly with some nodes but not others. I should point out however, that the SVOS with its extensive amount of I/O, represents a severe test for any distributed system. Thus the problems I detect with it probably will never be seen by other users. Conversely, if something works on the SVOS it will most likely work everywhere.

The source of our problem might just be limited drive capability resulting from both of the USB to RS422/485 Converter Cables under test receiving their 5Vdc power from the PCs USB port which is typically limited to 100ma. If this is the case, then utilizing a converter that has its own built-in power supply or is connectable to an external supply may well solve our problem. Bob Jacobsen, the founding creator of JMRI, helped confirm our rationale by informing me that his JMRI users have certainly seen RS-485 adapters that don’t have enough drive for handling large C/MRI systems

Encouraging Trend Using USB Converter Cables

Tom Gordon, one of the C/MRI Users exploring the application of VB.net is very successfully using the same USConverters’ Model UT890 for connecting to three SMINIs. His application program uses VB.net running on an Intel i7 processor operating at 2.67GHz with Windows 7/64. The drivers for his USB-based virtual RS485 serial port came as part of a Windows 7 update.

While the performance numbers for Tom’s application, which we will look at shortly, are very encouraging, it’s important to emphasize that Tom’s application is programmed in VB.net which is significantly different than programming in VB6. For example, the VB version of the Serial Protocol Subroutine Package (SPSVBM) introduced with the V3.0 User’s Manual will not function with VB.net.
For this reason, Tom converted them to operate with VB.net and his results can be found in the “files” section of the C/MRI User’s Group.

The overriding point to make about Tom’s setup is that, using a baud rate of 28800, his real-time loop iteration rate ranges between .030 and .031 seconds, where the slower time occurs when he is using the same computer to surf the net, play music and read/write email in addition to simultaneously interfacing with the three SMINI cards on his railroad.

The .030s cycle time that Tom is achieving using the USConverters' Model UT890 USB to RS485 Converter Cable is very good. Why do I say that? Well, first off, it has no problems keeping up with pushbutton presses. However, it is really more than that. Communicating with 3 SMINIs requires that 810 bits (calculated as 270 data bits + 540 overhead bits) be transmitted to and from the railroad every pass through the real-time loop. At a transmission rate of 28800 bits/sec, the theoretical serial I/O time is calculated as 810/28800 or .028 seconds. This means that the railroad oriented processing time within Tom's PC plus any USB overhead time is .030 - .028 or .002 seconds, or 2 milliseconds. Certainly, not much time is being wasted in Tom's USB to RS485 Converter.

However, even at this point, Tom's results appear to confirm a desirable trend that with faster computers and with the even faster USB3.0 becoming available on an increasing number of such computers, we can expect continuing improvement in making use of USB for real-time applications such as the C/MRI.

RS485 CONVERSION CARD

Assuming your selected connection method makes use of the JLC provided RS485 card, then this section is of importance. However, if you do not have need for such a card, then feel free to skip ahead to ***RS232 and RS485 Wiring Connections.***

Fundamentally, if you want to use RS485 I/O with a computer not equipped for it, then assembling an RS232 to RS485 Conversion Card, as shown in Fig. 4-9, can be a wise investment.

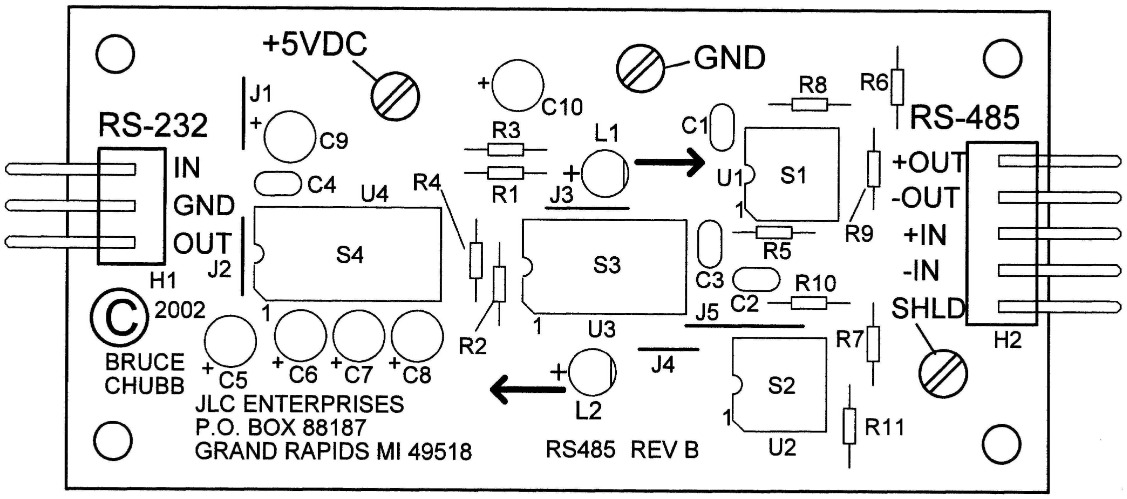


Fig. 4-9. RS485 conversion card parts layout

Ready-to-assemble conversion printed circuit cards are available from JLC Enterprises and fully assembled-and-tested cards and complete kits are available from SLIQ Electronics.

The RS485 card contains many of the same parts as used with the SMINI and the SUSIC. The 3-pin right angle header connector on the left provides the RS232 cable connections to your computer. The 5-pin right angle header connector on the right provides the RS485 cable connection to your SMINI, or SUSIC. The fifth pin, used for connecting to the cable's shield is also connected to a screw terminal enabling easy shield grounding. Also, two terminal screws are provided for connecting the required +5Vdc and ground for powering the card.

Two LEDs are employed to monitor the card's operation. The L1 yellow LED blinks when data is being converted from RS232 to RS485, that is going from the PC to the SMINI, or SUSIC. The red L2 LED blinks when data is being converted from RS485 to RS232, that is going from the SMINI, or SUSIC, to the PC. These come in handy for checking out system operation.

The schematic for the RS485 conversion card is shown as Fig. 4-10.

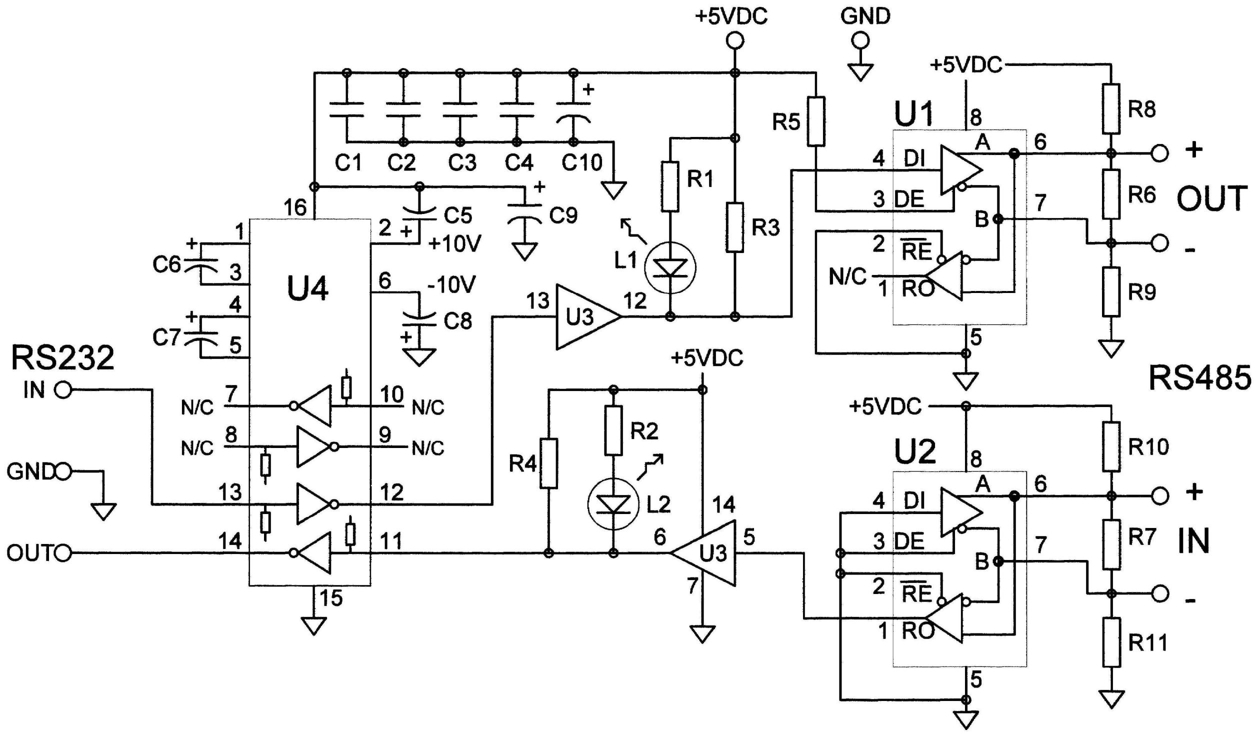


Fig. 4-10. RS485 conversion card schematic

Resistors R6 and R7 are line termination resistors for the RS485 cable. These match the typical impedance of the transmission cable to minimize reflections, to help reduce serial transmission errors at the higher baud rates. Resistors R8-R11 act as pull-up and pull-down resistors to help maintain the differential voltages on the transmission line in their idle state (an absolute value less than 200mV) when no data is being handled. Table 4-7 provides the parts list.

The RS485 assembly steps are much like those for the SMINI, and SUSIC, so I will only comment on the highlights as follows:

Card inspection.

J1-J5. Make these jumpers from excess resistor leads and install, solder and trim

R1-R11. Check the color codes for each resistor against those listed in the parts lists before installing

Terminals. Insert 4-40 screws into each of the 3 terminal holes from the top or component side of the board. Add 4-40 hex nuts on the bottom side, tighten firmly, and solder the nuts to the circuit pads.

S1-S4[+]. Make certain that the notch at the end of each socket lines up with the notch printed on the parts legend.

Table 4-7. RS485 conversion card parts list (in recommended order of assembly)

Qnty.	Symbol	Description
5	J1-J5	Jumpers - make from spare resistor leads
2	R1, R2	330Ω resistors [orange-orange-brown]
3	R3-R5	2.2KΩ resistors [red-red-red]
2	R6, R7	120Ω resistors [brown-red-brown]
4	R8-R11	1.0KΩ resistors [brown-black-red]
3	-	4-40 x ¼” pan-head machine screws (Digi-Key H142)
3	-	4-40 hex nuts (Digi-Key H216)
2	S1, S2	8-pin DIP sockets (Jameco 112206)
1	S3	14-pin DIP socket (Jameco 112214)
1	S4	16-pin DIP socket (Jameco 112222)
4	C1-C4	.1µF, 50V monolithic capacitors (Jameco 332672)
5	C5-C9	1µF, 35V tantalum capacitors (Jameco 33662)
1	C10	2.2µF, 16V tantalum capacitor (Jameco 94001)
1	H1	3-pin Waldom right angle header (make from 24-pin right angle header Mouser 538-26-48-1242 – see text)
1	H2	5-pin Waldom right angle header (make from 24-pin right angle header Mouser 538-26-48-1242 – see text)
1	L1	Diffused yellow T1¾ size LED (Jameco 334108)
1	L2	Diffused red T1¾ size LED (Jameco 333973)
2	U1, U2	Dual RS485/RS422 transceivers (Mouser 700-MAX487CPA)
1	U3	7407 hex buffer/driver open collector, non-inverting (Jameco 49120)
1	U4	Dual RS232 transmitter & receiver (Jameco 24811)

Author’s recommendations for suppliers given in parentheses above with part numbers where applicable. Equivalent parts may be substituted. Resistors are ¼W, 5 percent and color codes are given in brackets.

C1-C4. Polarity is not important. Insert, solder and trim.

C5-C10[+]. These capacitors are polarity sensitive so be sure to insert the + lead into the + hole which has the square pad. Note that C10 is a different value so you might want to identify it and install it first before installing C5-C9.

H1-H2. Cut off the required number of pins from a 24-pin right angle header. Install and hold them tightly against the card as you solder. It is best to first solder only one pin near the center. Then place the board and header over a screwdriver clamped in a vice with the blade protruding up. Lay the board bottom side up with the plastic part of the header resting on the screwdriver blade next to a soldered pin. Reheat the pin while pressing firmly down on the board and the header will tend to “snap” in place. Having verified that the header’s bottom surface is pressed flat and tight against the board, solder the remaining pins.

L1, L2 [+]. Make sure you have the correct color LED and that the + lead, typically the longer one, is inserted into the + hole which has the square pad.

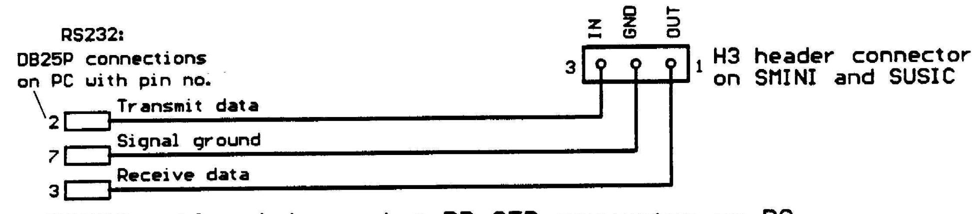
U1-U4[+]. Make sure that you have the correct part number for each location and that the ICs are inserted with their end notch lined up with the notch at the end of the socket and printed on the legend.

That is all there is to assembling the RS485 conversion card.

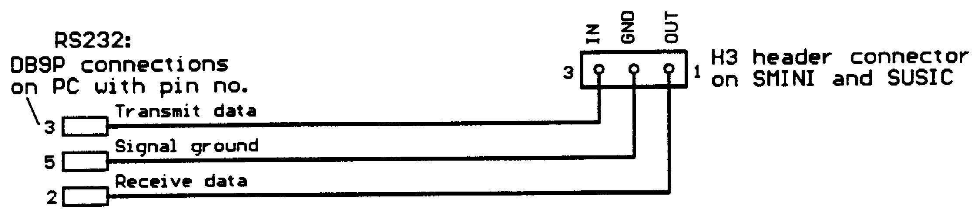
RS232 AND RS485 WIRING CONNECTIONS

Fig. 4-11 summarizes the wiring connections between your computer and the SMINI or SUSIC. It also shows connections to and from the RS485 Conversion Card for supporting multi-node applications.

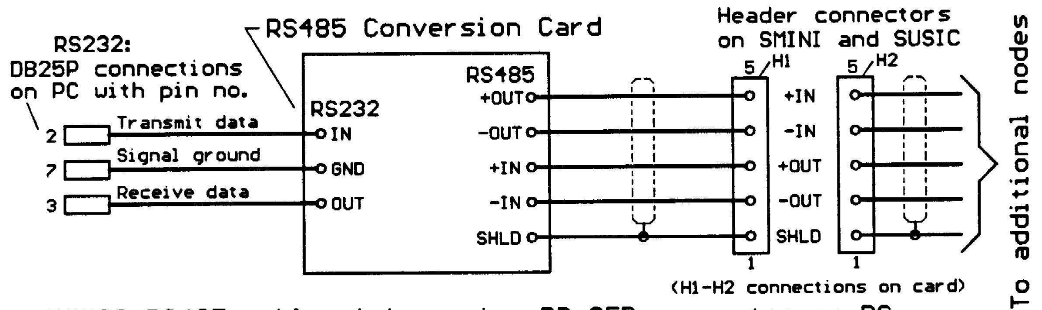
It is important to note when connecting RS232, that the transmit data out of the PC connects to the RS232 IN terminal on the SMINI and SUSIC or on the RS485 Conversion Card. Similarly, the receive data into the PC connects to the RS232 OUT terminal on the SMINI and SUSIC or on the RS485 Converter Card. Regarding the RS485 connections, the +OUT connects to the +IN and the -OUT connects to the -IN. Quite rationally, the output of one device becomes the input to the next.



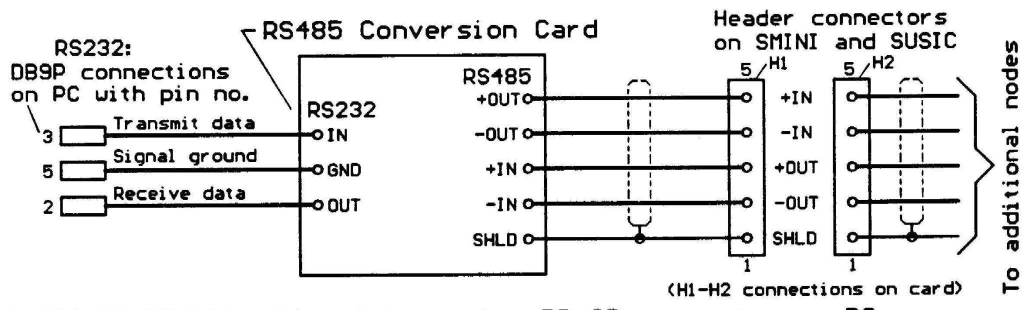
a. RS232 cable wiring using DB-25P connector on PC



b. RS232 cable wiring using DB-9P connector on PC



c. RS232/RS485 cable wiring using DB-25P connector on PC



d. RS232/RS485 cable wiring using DB-9P connector on PC

Fig. 4-11. Computer connections using RS232 and RS232/RS485

RS485 LAST NODE TERMINATION RESISTORS

Serial transmission cables or lines operating at high speed data rates can generate signal reflections, whereby the pulses being sent over the line, on reaching the opposite end, can bounce or reflect back toward the transmitter. The reflection problem is particularly troublesome for data rates above 1Mbps and frequently, commercial systems run up to 600Mbps or higher. Fortunately, at the relatively low bit rates used with the C/MRI, which are up to a maximum of 115200, or .1152Mbps, reflections are not a serious issue.

A good design rule of thumb is that as long as the bit width is 40 or more times greater than the transmission delay through the cable, any reflections will have settled by the time the receiver reads the bits. Using the relationship that an electrical signal travels through copper wire at approximately 1.5ηs (1.5 x 10^{-9} seconds) per foot, one can calculate the maximum cable length as a function of baud rate before line termination resistance is important. This relationship is summarized in Table 4-8. This shows that at the lower baud rates, say 28800bps and below, we need not be concerned about adding line termination resistors, that is, unless your RS485 cable becomes greater than 600 feet.

Table 4-8. Maximum cable length before require line termination resistors

Baud rate (bps)	9600	19200	28800	57600	115200
Bit width (μsec)	104	52	34.7	17.3	8.6
Max cable length w/o line termination (feet)	1736	867	595	289	144

However, at 57600bps and above, and with long cables, adding termination resistors at the far end of the RS485 cable (i.e. at the last node) can become important. Should you desire to add such termination resistors, you can easily do so by assembling a termination resistor connector as shown in Fig. 4-12. Once assembled, simply plug the connector onto an open set of RS485 header connector pins on your very last node, i.e. the furthest from the computer. This node can be either an SMINI or SUSIC.

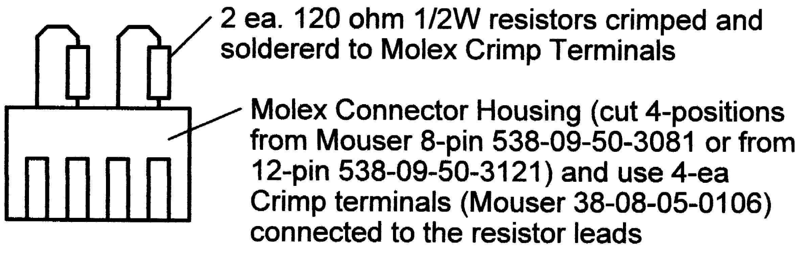

2 ea. 120 ohm 1/2W resistors crimped and soldererd to Molex Crimp Terminals
Molex Connector Housing (cut 4-positions from Mouser 8-pin 538-09-50-3081 or from 12-pin 538-09-50-3121) and use 4-ea Crimp terminals (Mouser 38-08-05-0106) connected to the resistor leads

Fig. 4-12. RS485 termination resistor connector

Make sure the connector is on the 4-signal lines and not the shield pin. Normally however, you will not need this connector. For example, for testing purposes, I have run the SV Oregon System's 7-node system for a very long time both with and without the plug installed, with no noticeable performance difference.

The higher baud rates provided by an all new SMINI and/or SUSIC distributed node systems can provide a great speed advantage. The higher baud rates can be especially important in keeping serial I/O times in line with response time requirements when creating distributed systems with a large number of nodes.

Because most serial interfaces do not begin to approach the 128 node capacity of the C/MRI and because of the high-speed capabilities of computers and the relatively low speed requirements of our railroads, most C/MRI users need not be concerned about serial I/O time. However for those that do, I cover ***Serial I/O processing time*** **and** ***writing large C/MRI programs*** in the **Railroader's C/MRI Applications Handbook**.

Additionally, if you are not interested in pursuing USB applications, I recommend that you jump ahead to the next chapter. However, for those interested, I'll add some closing sections covering my personal experience, and what I have observed from others, applied to USB C/MRI applications.

ADDED RESEARCH REQUIRED TO BEST APPLY USB TO C/MRI APPLICATIONS

There are a large number of USB to RS232 and USB to RS422/485 Converter Cables on the market and a significant number of C/MRI users are successfully using many different brands. However, based upon my discussion with many users and postings to the C/MRI User's Group, it appears that some brands work fine for some users while the identical model does not work as well, or it might even work better, for other users. The problems encountered appear often to be software related in terms of:

- Which computer operating system is being employed with which cable, and
- The capability and universality provided within the "Device Driver" software supplied with the particular converter cable and the corresponding level of provided customer support.

Personally, I avoid the very low priced converters in anticipation of lower level customer support. In general, there appears to be two different brand of processor ICs used within the converters, "Prolific" and "FTDI". Although I am not sure of how universal is the thinking, C/MRI user Jean-Francois posted in Message 13368 to the C/MRI Group that the "FTDI" based devices are a lot simpler to install. Although far from conclusive, my observations seem to support his input.

Because user experience varies between different brands of converter cables and even when using the identical model number device it is difficult to make specific recommendations. However, for the benefit of alerting users to the potential challenges that they might face in trying to incorporate USB converter cables, I will discuss my latest findings. Remember however, that the experience of others may be entirely opposite. With more experimentation and research efforts being performed by other C/MRI users, what is found as unworkable today may very well be workable tomorrow. Additionally, a simple wiring error or an incorrect parameter set in the driver software may lead to reporting that a particular converter doesn't do the job. Then, later on, when the error is corrected the converter may provide good performance.

To support my own experimentation, I purchased five different USB Converter Cables but found none that really performed to my high-level expectations. Most functioned successfully when connected to a single node used for testing I/O cards, but failed or were unreliable when attached to the large multi-node SVOS application involving 7 nodes with 134 I/O cards supporting nearly 3800 I/O lines.

- The USB to RS232 cables that I purchased include *www.cablestogo.com* Part Number 26886, *www.usconverters.com* Model UT880 and *www.Keyspan.com* Model USA-19HS.
- The USB to RS422/485 cables include the USConverters Models UT890 and U485G.

One design feature I like with the UT880, UT890 and U485G is their having three display LEDs. One illuminates when power is applied, the second when data is flowing from the PC and the third when data is flowing to the PC. I find these display LEDs very useful when checking and debugging system operation. By contrast, the Keyspan USA-19HS has a single LED that blinks when data is being transmitted or received and the Cables-to-GO has no display LED.

I did find when first attempting to use the Cables-to-Go converter, with my Laptop operating with VISTA, that I received an "Error 820" message when trying to read C/MRI data to the PC. A quick email to *techsupport@cablestogo.com* instructed me to download a new driver for Vista which after doing so solved the "Error 820" problem but I still could not read input data.

Taking advantage of Message 13358, as posted by Jean-Francois to the C/MRI User's Group, we downloaded the two Microsoft updates which he found required when using USB with Vista and VB6. However, our experience so far is that even with these updates installed, we still were unable to read input data from the large SVOS system using the Cables-to-Go converter cable.

C/MRI User Chris Elliott posted with his Message 13479 to the C/MRI User's Group denoting a "little gem" he found on the JMRI list that said: "Some of the newer adapters require the following pins connected together DSR-DTR (pins 4 and 6 on a DB9) and RTS-CTS (pins 7 and 8 on a DB9)". However, with just so many other avenues to investigate, and publication deadlines looming, we have not yet taken the opportunity to continue with such testing to see if making these added connections or following some other avenue-of-attack solves our problem when using the Cables-to-Go converter.

By contrast, once we installed the driver software that came with the USConverters Model UT880, we found that it worked right-out-of-the-box in communicating in both directions with the SVOS. However, the resulting real-time loop response was very sluggish. Typically, and as covered in the Railroader's Application Handbook in detail, the iteration time of the real-time loop on the SVOS using VB6 and operating at 28800 baud is .26 seconds. Out of the total .26s, the actual serial I/O time (the physical time it takes to transmit and receive data over the RS232 lines, including overhead) is .19s and the internal processing time of the VB6 application is .07s.

With everything else held identical except for substituting the UT880 in place of the direct RS232 connection, the loop response time increased from the .26s up to 1.68s. This is unacceptable. For example, you need to hold a "pushbutton pressed" for close to 2s to be assured that the button press was recognized by the application software.

An email to the support team at *mail@usconverters.com* came back with the following response: *"Thank you for your detailed description (of your problem). I understand what you mean. There will always be latencies with using a USB to serial adapter because of the drivers and data conversion. Unfortunately, there is nothing we can do to improve the adapter itself, we would have to find you another high-speed adapter that fits your exact application."* Then they went on to suggest other converter cables that I should purchase for testing to see if they did a better job. Well, one could go broke pretty fast following that approach.

While waiting for the response, we sought support from the C/MRI User's Group. Bob Jacobsen, one of the responders and the founding creator of JMRI, provided significant input with his explanation in Message 13552 that:

> *"Serial-to-USB convertors have to decide when to forward the input data to the computer. Ideally, this would be done on each character, but some adapters default to only forwarding when a control character is received, and some default to only forwarding on CR or LF. Since C/MRI messages don't end in a CR or LF, that last type can delay reception until a timeout expires, typically 250 msec. It sounds like that's what's happening here.*
>
> *If bad defaults for the adapter are the cause of this, the program needs to set some options. Setting the "framing character" to 0x03 really helps, because then the USB adapter knows to forward the contents immediately when the end of message is received. Setting the receive timeout to something small (JMRI uses 10 milliseconds) can also help, at some cost in increased processor load.*
>
> *It would be straight-forward to test for the control leads causing the delay. Take the DTR signal on pin 4 (as an active level; or use a power supply) and jumper it to pin 1, 6 and 8. (CD, DSR,*

CTS) Only CTS (clear to send; permission for the computer to output characters) might make a difference. If you find that it does, the port is probably trying to do "hardware flow control"; the program can explicitly turn that off.

I have no idea how to do this in VB, but here are the corresponding Java statements:

```
activeSerialPort.enableReceiveFraming(0x03);
activeSerialPort.enableReceiveTimeout(10);
activeSerialPort.setFlowControlMode(SerialPort.FLOWCONTROL_NONE);"
```

Based upon Bob's input, we delved into implementing his recommendations, first via the driver software supplied with the UT880. Low and behold, and certainly not told to us by the USConverters' support team, the UT880 driver software provided an option to set the Latency Time. Adjusting the Latency Time from its "default" setting of 16ms to the lowest accepted setting of 1ms, resulted in a loop response time change from 1.68s to 1.2s. A significant improvement, but still far short of that required for acceptable real-time response.

At that point in our test/experimentation process, we received the Keyspan USA-19HS USB to RS232 adapter that is advertised as "Low Latency". Using its driver's "default setting" the measured real-time response was .9s. We then changed the driver's "compatible interrupt" setting to "high performance bulk" and the timing improved to .84s. Next, changing the "Transmit Completion Timing Advance (TCTA)" from the default setting of "standard" to "faster" the timing improved to .5s. Lastly, changing the "receive First-In-First-Out (FIFO) buffer size" from its default value of "16" to "none" improved the loop time to .46 seconds.

When operating at the .46s rate, we could very acceptably move levers and press code start buttons on both the US&S and the GRS CTC Machines, and operate the three N-X Interlocking Machines and achieve acceptable system performance. That made us pretty happy.

To further test the Keyspan converter, SVOS Operator Don Jones moved it to our Laptop computer operating with Vista. With the identical VB6 application software operating the railroad, Don cycled through all combinations of the "driver options". The best loop response achieved was .46s, which is identical to that achieved with the XP machine. Additionally, the options that provided this response were, "high performance bulk", "TCTA = faster", and "FIFO" set to 16 although a setting of 56 provided the same results. However, reducing "FIFO" to "none" increased the loop time from .46s to .58s.

As we will see in a following section, I am very confident that if the Keyspan Driver software would permit changing from "high performance bulk" to "isochronous transfer", that we could achieve significantly faster real-time loop response. However, to date, I have not yet located a USB Converter Cable that permits Isochronous Transfers. C/MRI serial system specialist Alan Anderson is supporting this research with a special test setup that he is generating.

Besides its improved operation, the Keyspan USA-19HS provides a downloadable 34-page Users Manual. Additionally, a "Keyspan Serial Assistant" software package is provided that can capture and display the bytes going and coming on the port. It includes timestamps and has good "display" and "save to file" features. The "Open Data Monitor Window" sets up the traffic capture feature.

SUBSTANTIAL USER TESTING USB APPLICATION

Another point of reference is the extensive testing performed by Ken Cameron, who among his many activities supports a C/MRI system on Jim Heidt's freelanced Ogdensburg and Norwood Railway which was a real railroad name that Jim's family owned for years that ran between Ogdensburg, NY along the St. Lawrence River to Norwood, NY where it interchanged with Penn Central. Its 5-node C/MRI system comprises 3 SMINI cards, an SUSIC with 5 DOUT32 and 1 DIN32 and a second SUSIC with 1 DOUT32 and 1 DIN32. The system's function is to monitor 85 DCCODs and 50 turnout sensors and control all wayside signaling. Also, part of the job of the one SUSIC node is running 27 stepper motors that drive the room lighting dimmers and another 6 for clock speed control. Everything feeds into and out of JMRI via the C/MRI.

Two different computer setups are employed with the data for each summarized as:

1. 800MHz Pentium running Windows ME, JMRI 2.9.6, on Java 16.0_17, and a real RS232 serial port operating at 19200 baud and using the default settings within JMRI for the serial port control. The real-time loop cycle time for this set up using RS232 Serial Port is .2695s seconds.

2. Dual Core 2.6GHz computer with USB2.0 ports running Windows XP, JMRI 2.9.6, on Java 16.0_20 and incorporating the Keyspan USA-19HS USB to RS232 Converter Cable operating at 19200 baud. The real-time loop cycle time for this setup using the USB Converter Cable is .1564 seconds.

The first setup is used to support normal railroad operations and the second, making use of a laptop, is used for application program development and system debugging.

What is encouraging regarding Ken's data is that he is achieving faster real-time loop response when using the USB port than when using the RS232 port, albeit on a much faster computer. Based upon Ken's findings, it might be safe to assume that as computer speeds increase, the applicability of USB to adequately handle real-time requirements will become increasingly attractive.

To obtain a better handle on equivalent timings using the same computer, Ken set up additional testing using a computer having both an RS232 Serial Port and a USB connection. For the record, it was an IBM Thinkpad A21, 200MHz P3 Pentium with 384Meg RAM operating with Windows XP. The real-time loop cycle time when using the RS232 Serial Port setup was .13 seconds and when using the Keyspan USA-19HS Converter Cable was .16 seconds. Table 4-9 summarizes the data that Ken collected.

Table 4-9. Loop timing test results for different computers and port connections**[1]**

Computer (see text for details)	**Operating System**	**Computer's Port Connection**	**Loop Timing (seconds)[2]**	**Programs Internal Processing Time[3] (seconds)**
Pentium 800MHz	Windows ME	RS232	.27	.19
Dual Core 2.6GHz	Windows XP	USB2.0 using Keyspan USA-19HS	.16	.08 [4]
Pentium 200MHz	Windows XP	RS232	.13	.05
Pentium 200MHz	Windows XP	USB1.0 using Keyspan USA-19HS	.16	.08 [4]

[1] All test data supplied by joint C/MRI and JMRI user Ken Cameron with all testing conducted with a C/MRI System consisting of 3 SMINIs and 2 SUSICs with 8 I/O cards (32-bit) using a baud rate of 19,200
[2] Loop timings are rounded to the nearest .01 seconds.
[3] Assumes theoretical serial I/O time, as calculated below, of .08 seconds (rounded to nearest .01 seconds)
[4] Includes possible added delay resulting from USB Converter Cable

Numerous conclusions can be reached by studying the results presented in Table 4-9:

- All four loop timings listed in Table 4-9 yield acceptable performance,

- However, when the same computer is used for both tests as indicated with the last two rows, the operation with the USB cable is slower by 23%.

- Also of special interest is that the performance with the 200MHz Pentium, both with RS232 and with USB, is significantly faster than the performance provided by the 800MHz Pentium using RS232. This result may be attributed to non-optimality of the Windows ME operating system applied to real-time applications.

- Equally interesting is that there is no measurable improvement in loop timing when using the 2.6GHz Dual Core processor with USB2.0 as compared with the 200MHz Pentium using USB1.0. It would appear that both performances are being limited by the performance of the Keyspan USA-19HS.

To expand a little on the last conclusion, it is important to understand that in every application, there is a portion of the loop timing that is essentially independent of computer speed. This is the actual transition time that it takes for the bits to be transmitted to and from the railroad. This time is dependent upon the baud rate, which is the number of bits that are transmitted per second, and not computer speed.

Table 4-10 shows a “work table” that can be used for calculating the theoretical Serial I/O time for any C/MRI system as a function of its number of nodes, the overall compliment of cards making up the nodes, and the baud rate being employed. I have filled out the table, using the numbers noted in italics, for the C/MRI system used for collecting the test data shown in Table 4-9. For those interested, the derivation of Table 4-10 is presented in Chapter 15 of the Railroader’s Application Handbook.

Table 4-10. Serial I/O time calculation for Ogdensburg and Norwood Railway (Employing 3 SMINIs, 2 SUSICS and 8 I/O cards (32 bit)

Arithmetic Function to be performed:	**No. of Bits and Serial I/O time**
Enter total number of 24-bit I/O cards here __*0*__ then multiply by 30	*0*
Enter total number of 32-bit I/O cards here __*8*__ then multiplied by 40	*320*
Enter total number of SMINI cards here __*3*__ then multiplied by 90	*270*
Enter total number of serial nodes here ___*5*_ (including all USIC, SUSIC and SMINI nodes) then multiply by 180 to determine total number of “overhead bits” per pass through real-time loop	*900*
Perform addition of the above 4 entries in the rightmost column to arrive at the total number of bits that need to be transmitted each pass through the real-time loop	*1490*
Enter the baud rate here ___*19200*___ (with valid entries being 9600, 19200, 28800, 57600 or 115200) and use it to divide into the above total number of system bits to arrive at the serial I/O time in seconds	*.0776s*

Table 4-10 shows us that communicating with 3 SMINIs and 2 SUSICs with a combined total of 8 I/O cards of 32 bits each, requires 1490 bits (calculated as 590 data bits + 900 overhead bits) being transmitted to and from the railroad every pass through the real time loop. At a transmission rate of 19200 bits/sec, the theoretical serial I/O time is calculated as 1490/19200 or .0776 seconds. This is the actual time it takes for RS232 to transmit all the required I/O bits, including overhead, between the PC and the railroad each pass through the real-time loop.

From this, we can determine that the internal JMRI processing time, including any extra delay caused by the USB conversion as simply the measured Loop Timing, the second column from the right in Table 4-9, minus the .08 seconds. This subtraction is the time reported in the rightmost column in Table 4-9.

USB TRANSPARENCY AND DATA TRANSFER METHODS

A key point in using a USB to Serial Port converter cable is that the resulting conversion is transparent. You simply compose your program just as if you were using a "standard" RS232 serial port. The only difference you are likely to notice is a change in the COMPORT number. Dependent upon which USB port you plug into, the Operating Systems' Device Manager assigns an equivalent COMPORT number, usually between 4 and 9. To determine the specific assignment in your situation, Click on "Control Panel" and then click on "Device Manager".

If the assignment is greater than 4, and you are using the Serial Protocol Subroutines supplied with the V3.0 User's Manual, you will encounter the "COMPORT must be 1, 2, 3 or 4" error message. To correct the problem, replace the SPSVBM.bas module in your application program with the one contained on the CD associated with this manual. Assuming your computer has an available expansion slot, then an alternative to purchasing a USB to RS232 Converter Cable is to invest in a plug-in serial interface card that supports RS232 and sometimes RS422/RS485 combined with RS232. In that case direct connections can be made between the computer and the SMINI and SUSIC.

****Key Points Using USB with C/MRI****

1. Once the USB to RS232 Converter Driver is recognized by the Operating System, the fact that you are using a USB Port in place of having an RS232 Port is transparent to the user.
2. Simply set up your I/O identically to how you would when using RS232 which is exactly how all of the presented C/MRI examples are created.

The main reasoning behind using a "real" RS232 port, contrasted to a "virtual" RS232 port as set up when using USB, is that physical RS232 ports are integrated more closely with the computer's hardware and application software. Typically with the real RS232 port, the application software has direct or almost direct accessibility to the port's performance. Settings like baud rate, number of data bits and stop bits, and use of hardware-software flow control can frequently be controlled from within the application program.

By contrast the USB interface is much more complex with much added overhead. Additionally, it does not provide a close tie to the application software. This can result in the user having less effective control of how the I/O is handled. To try and provide "transparent" RS232 performance under USB, a second device driver is necessary with its purpose being to emulate an RS232 UART, which is the IC that handles the RS232 data transfers, but still communicates through the USB.

The challenge with real-time applications, such as the C/MRI, is that specific timing constraints apply to I/O handling. With RS232 ports closely integrated into the computer's core architecture, data transfer can be very closely controlled when running QuickBASIC applications under DOS. Even when operating with Visual Basic under Windows, which can interfere with I/O efficiency, there is still reasonably tight control. For the most part, the application software communicates directly, or via a small and very efficient device driver, to the RS232's UART. Thus, everything happens within a quite closely defined time frame.

The setup with the USB bus is much more complex and much more divorced from the application software. Additionally, having the double device driver layer, with an RS232 driver functioning on top of the complex USB driver, is prone to adding extra communication overhead, resulting in delays. In our quest to maximize system responsiveness when using USB it is important to be aware of the four different types of transfers employed with USB. These are:

Control Transfers – are typically used for command and status operations and they are the only type that is specified by the USB specification. They enable the host, in our case the PC, to read information about a device, set a device's address and select its configuration. They may also be used to send requests that result in sending and receiving data for any other purpose. This "send request" capability is essential because USB bus operation is a "polled system" whereby USB devices on the bus can send information to the host only when requested to do so by the host.

Interrupt Transfers – are typically non-periodic, small device "initiated" communications whereby the device must receive the PCs attention or the PC must obtain the device's attention with a bounded latency. For example, mouse movement and key press data make use of interrupt transfers.

Isochronous Transfers – can occur continuously and periodically. They typically contain time sensitive information and enjoy a guaranteed delivery time. Typical applications include transferring audio and video files to be played in real-time. If there were a delay or retry of data in a video stream, then you would expect some erratic video containing glitches or with an audio file the "beat" may no longer be in synchronism. On the contrary, if a frame or packet of data was dropped every now and then, the loss would be less likely noticed. Thus, there is CRC checking of isochronous transferred data but there is no automated retry if data is found to contain errors. Consequently, if using isochronous transfers, occasional errors must be tolerated.

Bulk Transfers – are intended to support moving large "bursty" data under situations where the timing to initiate the transfer is not critical. Example applications are sending a file to a printer, or receiving data from a scanner. In these applications, data transfer can typically wait for a reasonable period of time. Fundamentally, bulk transfers use spare un-allocated bandwidth on the bus after all other transactions have been allocated. For example, if a bus is busy with isochronous and/or interrupt transfers, then bulk transfers may just slowly trickle over the bus. As a result, bulk transfers should not be used for time sensitive communication as there is no guarantee of latency. On the other hand, if the bus is idle, bulk transfers can be very fast.

Almost everything I read and study regarding USB points to the fact that real-time applications, such as transferring audio and video files to be played in real-time, robotics, machine tool control and the C/MRI, are best served by having the USB port operate in its Isochronous Transfer mode. However, up to this point in time, I haven't discovered a USB Converter Cable that permits operating in this mode. Possibly, searching for adapter cables designed to support the audio and video editing market, would provide fruitful results.

Although the management of all transfers is under the direction of the PC, it is the responsibility of the end-device, e.g. the printer, scanner, mouse, and so forth, to inform the PC as to its data transfer needs. Once these needs are understood, the PC operates under a strict "time slicing protocol" to manage all transfers. Dependent upon such factors as the particular version of USB being used, the type of device with which it is communicating and the transfer method being employed, three different operational speeds are involved. These are denoted as "low speed", "full speed" and "high speed". Traffic is managed by dividing time into slices called "frames", or in the case of high speed transfers "microframes". Frames, used for low and full speed transfers, are 1ms duration. For high speed data transfers, the PC divides each frame into eight 125µs Microframes. Low speed devices, such as mouse and keyboards, must use either Control or Interrupt transfers, while Isochronous and Bulk transfers must use full or high speed. However, USB1.0 only supports low and full speed while USB2.0 automatically uses high speed whenever possible, switching to low and full speeds when necessary.

For general applications, where you plug a USB compatible device, such as a printer, mouse or scanner directly into a USB socket on your computer, the PC automatically interrogates the attached device to determine its capabilities and needs and then adjusts the operation of the USB port, such as setting the transfer mode, operational speed, receive buffer size, etc., to “best service” the device. To provide this “plug and play” capability, every manufactured USB device must receive a Vendor ID and a Product ID that are embedded in the device to identify it to the PC’s operating system. Additionally, “device specific” parameters are transferred as part of the interrogation that in turn enable automatic optimization of the USB port to “best match” the needs of the device.

FUNDAMENTAL CHALLENGE USING USB CONVERTER CABLES

However, once a USB to Serial Port converter cable is employed to attach a non-USB equipped device, such as the C/MRI, the merits provided by “plug and play” loose their luster. The host no longer knows if the end-point device is a printer, scanner, digital camera, robot, piece of laboratory test equipment or production machinery, or a C/MRI. It simply knows that a particular brand and model number adapter cable is attached. The knowledge that the PC needs to “optimize” its USB bus performance is missing.

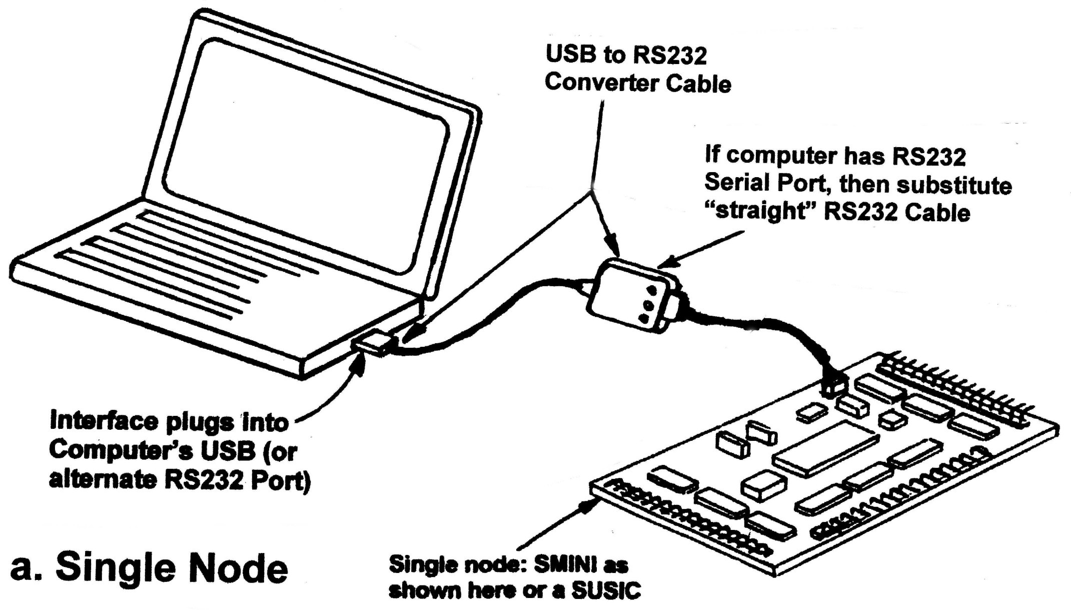
To help overcome this limitation, USB to Serial Port Converter Cable providers include options in the “Driver Software” supplied with their cable. Then, working within the constraints of the driver software the cable’s end user has latitude in tailoring the operation of the USB port to best fit the intended application.

For the majority of applications where USB to Serial Port converter cables are employed, such as attaching a non-USB equipped printer or a reasonably sized C/MRI application, simply applying the “default options” is adequate. Other general applications that have somewhat more time-critical transfer requirements, including medium sized C/MRI application, may require some level of “driver parameter” tweaking.

Fundamentally, being able to “best set” driver parameters becomes increasingly important as the demands of the real-time system expand. With the C/MRI, this “expanse of the system” is measured in terms of the number of bits that need to be transmitted between the PC and the railroad during each pass through the real-time loop. As indicated within Table 4-10 this number is directly related to the total number of Nodes, SMINIs, SUSICs and 24-bit and 32-bit I/O cards.

Experience has proven for a growing number of USB C/MRI users that obtaining a suitable USB adapter cable and then, if and when required, making appropriate driver parameter adjustments, can and will continue to provide successful results. This is especially true for the smaller, less-demanding, real-time applications. The challenge comes, I believe, when working to apply USB adapter cables to the larger real-time systems.

While this “research level” activity is continuing, my recommended course of action, and especially so for the large system application users, is to continue where you have a choice to make use of a computer that still employs a RS232 connector. However, as the technology and our application of it improves, I fully expect that the utilization of USB for C/MRI applications will continue to expand. Furthermore, as they expand to make increasing use of USB ports as integrated into all later model computers, the typical C/MRI setups will be reflective of those illustrated in Fig. 4-13.


USB to RS232 Converter Cable
If computer has RS232 Serial Port, then substitute "straight" RS232 Cable
Interface plugs into Computer's USB (or alternate RS232 Port)
a. Single Node
Single node: SMINI as shown here or a SUSIC

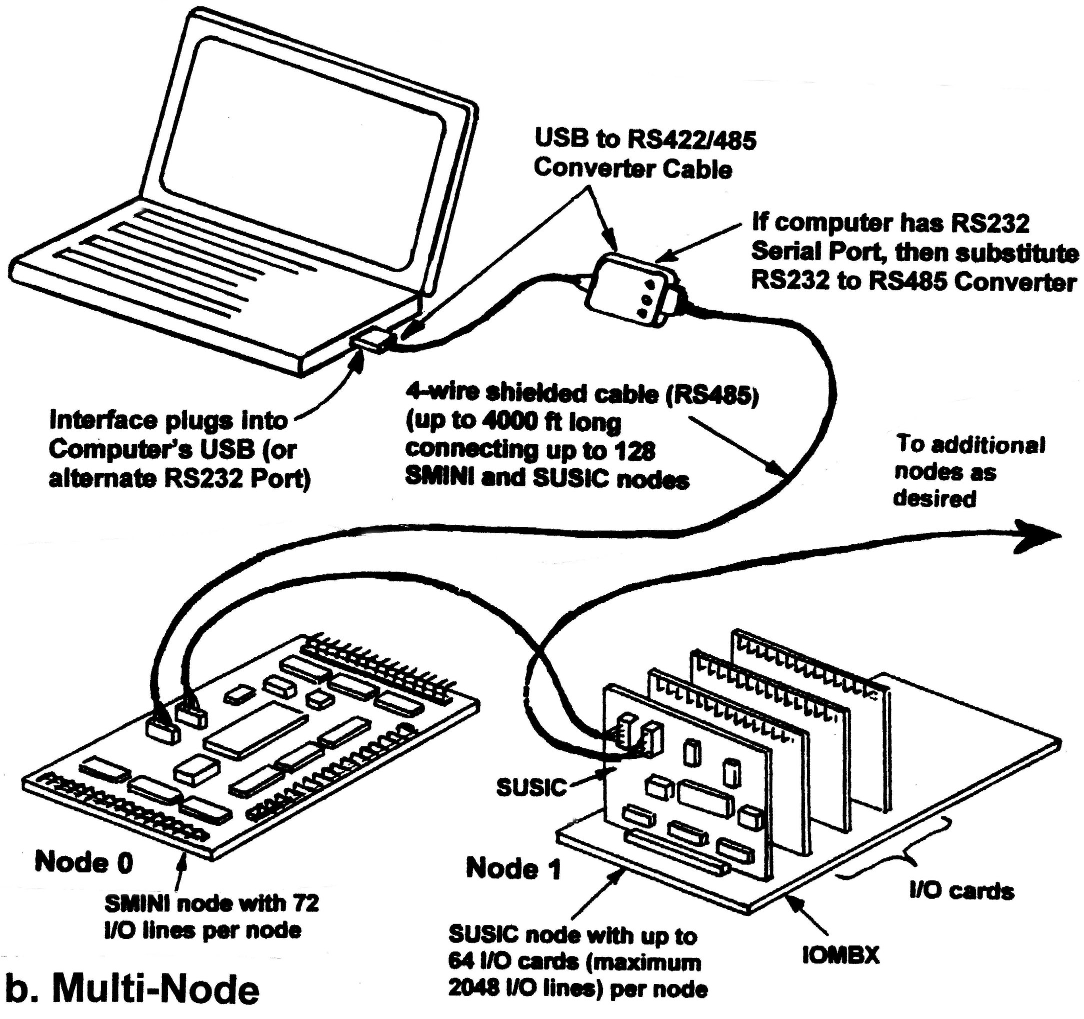

USB to RS422/485 Converter Cable
If computer has RS232 Serial Port, then substitute RS232 to RS485 Converter
Interface plugs into Computer's USB (or alternate RS232 Port)
4-wire shielded cable (RS485) (up to 4000 ft long connecting up to 128 SMINI and SUSIC nodes
To additional nodes as desired
SUSIC
Node 0
Node 1
SMINI node with 72 I/O lines per node
SUSIC node with up to 64 I/O cards (maximum 2048 I/O lines) per node
I/O cards
IOMBX
b. Multi-Node

Fig. 4-13. Serial interfacing example systems employing Universal Serial Bus (USB)