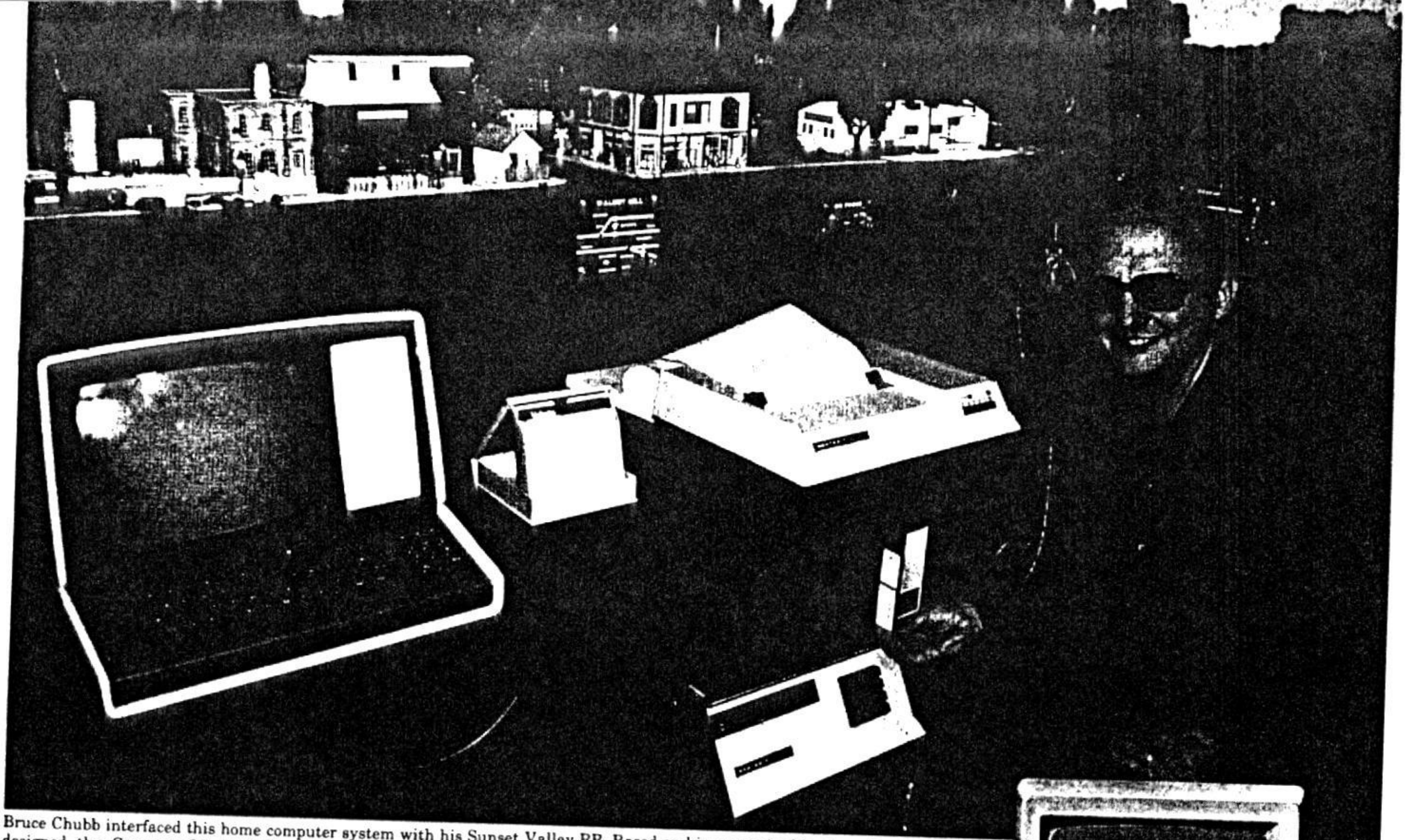

Bruce Chubb interfaced this home computer system with his Sunset Valley RR. Based on his customized hookup, Bruce designed the Computer/Model Railroad Interface (C/MRI) to be compatible with most popular home computers.

The C/MRI: A computer/model railroad interface

Part 1: How to connect a home computer to your layout

BY BRUCE CHUBB

HERE is your invitation to join in a major advancement in our hobby: the application of computers to model railroading. Obviously a computer can perform many useful paperwork and record-keeping functions for a model railroad. The benefits really expand, however, when the computer is electrically interfaced with the railroad for a wide variety of control and signal applications.

I'm using a Computer/Model Railroad Interface (C/MRI) to connect a home computer to my Sunset Valley line, and this is the first of a series of articles that will show you how to build your own C/MRI and use it with your railroad.

WHY A COMPUTER

I resisted "computerizing" the Sunset Valley for quite a few years. When I was writing my book, *How to Operate Your Model Railroad,* Linn Westcott often used to ask me when I was going to put a computer on my railroad. I would respond with something like, "I really don't plan to do it — it works just great the way it is," or "I like it without a computer," and sometimes more strongly, "I don't want a computer running my railroad!"

But things have changed. Now I have a computer hooked to the Sunset Valley through the C/MRI, and I love it. I still don't let the computer run the railroad, but my friends and I who do run it are extremely pleased with how much the computer has enhanced our enjoyment.

The main thing I wanted the computer to do for the SV was to operate the signal system prototypically, including accepting controls from and providing displays for the dispatcher's Centralized Traffic Control or CTC panel. [Bruce described his CTC installation in the January 1984 MODEL RAILROADER.] In addition, I wanted to be able to simplify the railroad wiring and use the computer to make a quick diagnosis of any problems.

Also, while fully automatic operation is not my goal, with the C/MRI the computer could quite easily do it. Eventually I think I will use it for automatic and/or semiautomatic operation, but just to run trains in a display mode so I can pay attention to my guests.

I use the same computer to automatically generate switchlists for operating sessions, for railroad-related word processing and printing, and even to let visitors who aren't model railroaders play video games. In this series, however, I'll be dealing specifically with building and using the interface between computer and model railroad: the C/MRI.

I'll even show you how to do something I don't need on the Sunset Valley. I'm using the CTC-16 command control system, but I'll show you how to use the C/MRI-equipped computer for automatic block power routing, a form of computerized cab control that takes the place of block toggles or rotary switches.

COMPUTER SELECTION

I could easily write a whole article on selecting the best computer to match a set of needs. My main advice is not to get locked into a small system without good expansion capabilities. Almost every system claims great expandability, great software, and great peripherals, but make them prove it by demonstration before you buy.

For the Sunset Valley I chose to build

a Heathkit H-8 computer kit. Ready-built machines are fine, but I like the kit approach because when you've built it you really know what you have. The H-8 has more room for internal expansion than the newer all-in-one designs like the Heath H-89 or Radio Shack TRS-80. The front panel display on the H-8 also lets me observe the condition of any internal system variable in real time, which is handy for debugging. Because they are being superseded by newer designs, used H-8s are often available at very low cost.

I started out with only 16K of memory and cassette I/O (*I*nput/*O*utput), but soon found that far too confining. Disk I/O for program and data storage is the only way to go. My present system consists of these components:

H-8 Computer with 56K memory
H-19 CRT display/keyboard
Three 5.25" disk drives
H-14 line printer
Bus extender card

The bus extender card is the basic element of the C/MRI, a home-built interface that lets the computer connect to the railroad as if it were a part of the computer's memory or I/O structure, and building it will be covered in the next part of this series.

For now I want you to know that you don't need this large a computer system to handle your railroad, even if it's a larger layout than the Sunset Valley. A computer with as little as 16K memory and one disk drive can do wonders for a model railroad of almost any size.

And, while I like my Heathkit, I've designed the C/MRI to let you use any of the most popular home computers. For that reason I call the bus extender a Universal Bus Extender Card (UBEC), and in part three I'll explain how to configure it for and connect it to these machines in addition to the H-8:

Apple II, II+, and IIe
Commodore 64
Commodore VIC-20
IBM PC
TRS-80, Models 1 and 3
TRS Color Computer

Even if your machine isn't on this list, I'll be showing you enough different ways to connect into a computer bus that, with the help of your computer's manuals, you should be able to figure out what you need to do.

THE COMPUTER'S ROLE

Figure 1 shows in a general way how the SV computer interfaces with the dispatcher's CTC panel and the railroad itself, and how CTC-16 fits in. Engineers use walkaround throttles to run trains with the CTC-16 system. They walk with their trains and obey the trackside signals, and follow switchlists and timetable schedules to operate prototypically. With command control there are no artificial electrical blocks restricting where a train may go, so the computer's contribution in controlling signals and turnouts is especially helpful.

As I said before, I use the computer to prepare paperwork prior to the operating session, but once the session starts the computer works directly coupled to

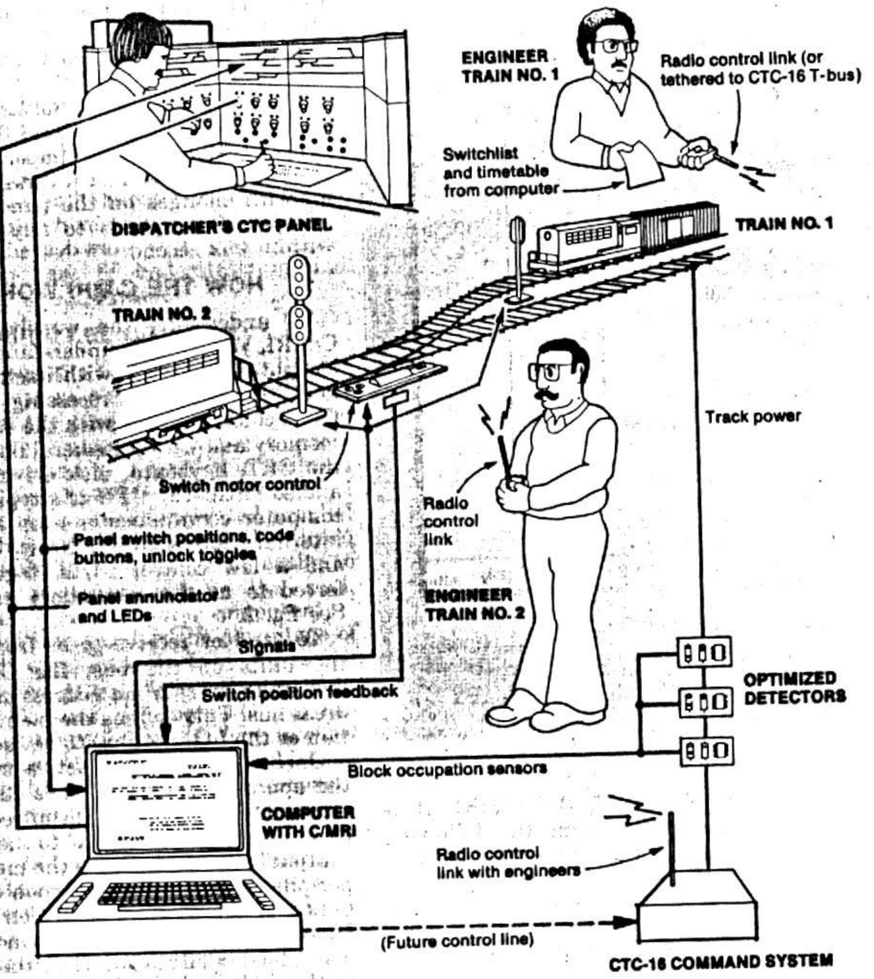


Fig. 1 THE COMPUTER'S ROLE ON THE SUNSET VALLEY

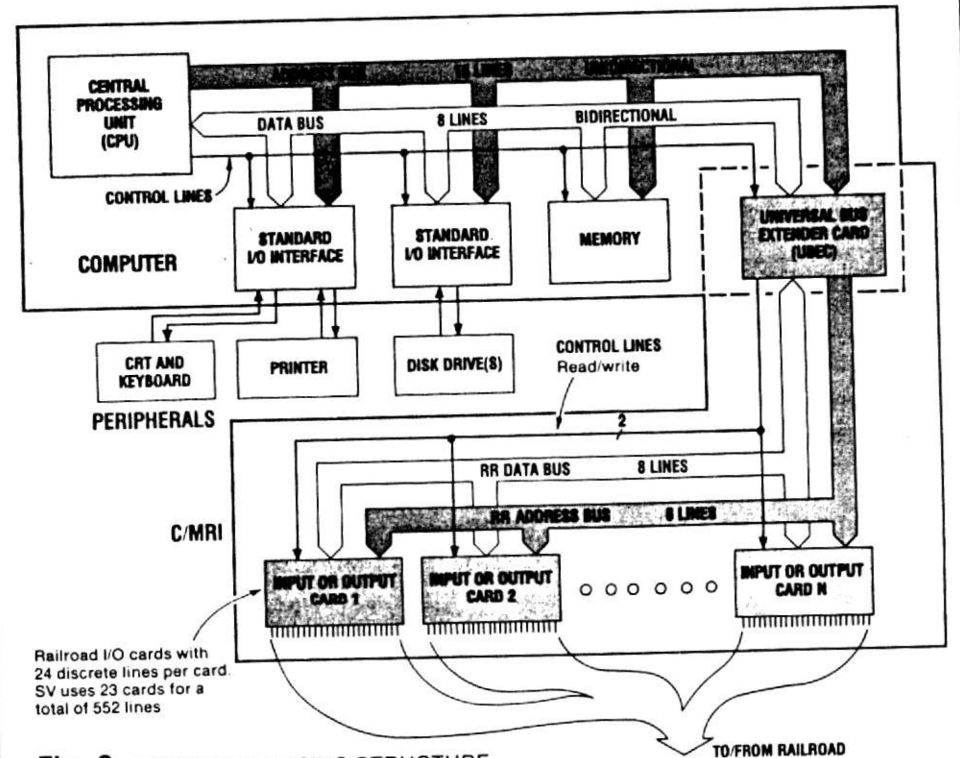


Fig. 2 COMPUTER BUSING STRUCTURE

FUNCTION PERFORMED	INPUT	OUTPUT
Turnout position	16	
Panel lever positions	35	
Code buttons	15	
Panel LEDs		62
Occupation detectors	61	
Turnout unlocks	10	
Signal LEDs		193
Switch motor control		20
Motor off-power	14	
Annunciator horn		1
Reverse block relay		1
Turnout enable		1
Grade crossing sensors	28	
Grade crossing control		5
Crossing manual override	5	
Cab signal control		3
CTC-16 throttle control		16
Spare	32	34
INPUT & OUTPUT TOTALS	216 (9)	336 (14)
TOTAL I/O DISCRETE LINES	552 (23)	

Note: Numbers in parentheses are numbers of I/O cards required at 24 lines per card

Fig. 3 I/O REQUIREMENTS FOR THE SUNSET VALLEY

the railroad through the C/MRI. It is programmed to perform the following functions in a continuous cyclic loop:

1. Read the CTC panel for the positions of switch and signal levers, switch lock and power toggles, and code start buttons.
2. Read the railroad for turnout positions and train locations.
3. Calculate, according to prototypical procedures, the appropriate setting of every trackside signal, switch motor, grade-crossing signal, reverse block relay, and panel indicator or annunciator.
4. Send electrical signals to set all railroad and panel devices as calculated in step 3.
5. Go back to step 1 and start the loop all over again.

The SV computer performs this loop about 16 times every second. The actual rate of calculation isn't important, since the computer is so fast it can easily keep up with changes on the panel and the railroad. It responds to any command within .062 second.

HOW THE C/MRI WORKS

To understand the workings of the C/MRI, you need to understand how the computer interfaces with itself. That is, how the Central Processing Unit, or CPU, communicates with the computer's memory and with peripheral devices like the CRT, keyboard, disk drive, printer, and so forth. The CPU of a typical home computer communicates over a 16-wire internal address bus, an 8-wire data bus, and a few control wires, together referred to as the computer's main bus. See fig. 2.

To send or receive to or from one of the units on the bus, the CPU first transmits or "sets" an address on the address bus. This defines the memory location or the I/O interface to be "accessed."

Once the address is set, a signal on the appropriate control lines causes the message (data) to be transmitted on the data bus, either as "input" to the CPU or "output" from the CPU to the memory or peripheral. Every transmission over the data bus goes to every device on the bus, but the only device that responds is the one whose address matches the setting of the address bus. Note that while the data bus is bidirectional the address bus is unidirectional.

A straightforward way to interface the computer with the railroad is to use a parallel I/O interface card. Such cards are available as add-ons for many popular computers. They plug into spare slots on the bus and provide eight separate input/output lines, each of which can be directly or indirectly connected to devices on your railroad. However, at eight lines each, the cards needed for even a simple model railroad would soon fill the spare slots in any home computer, as well as rapidly drain your pocketbook.

The C/MRI is a more efficient and economical way, and it's also illustrated in fig. 2. The UBEC or Universal Bus Extender Card extends the computer's internal bus to an external set of home-built I/O cards, and each one of these handles 24 separate lines to or from your railroad. On the Sunset Valley I use 23 I/O cards for a total of 552 discrete lines (including a few spares), as listed in fig. 3.

With large amounts of I/O it's a good idea, and usually necessary, to have an electrical buffer between the railroad I/O cards and the computer bus, and the UBEC performs this function too. This lets you use a separate power source for the railroad I/O cards and keeps the C/MRI from loading down the computer's internal bus.

The 16 address bus lines and the 8 data bus lines are essentially standard for the machines we'll be dealing with, but there are many differences in the control lines. The UBEC lets you get around these differences by making changes in its cable connections and in a few program jumpers on the card itself. That's what I'll be showing you in the third installment of this series.

For my H-8 I made a special bus extender card to fit right into one of the machine's spare card slots. Putting the bus extender card right inside the computer is a very neat way to go, so long as you have space for it. The H-8 has several spare slots, as do the Apple II family and the IBM PC. The approach of

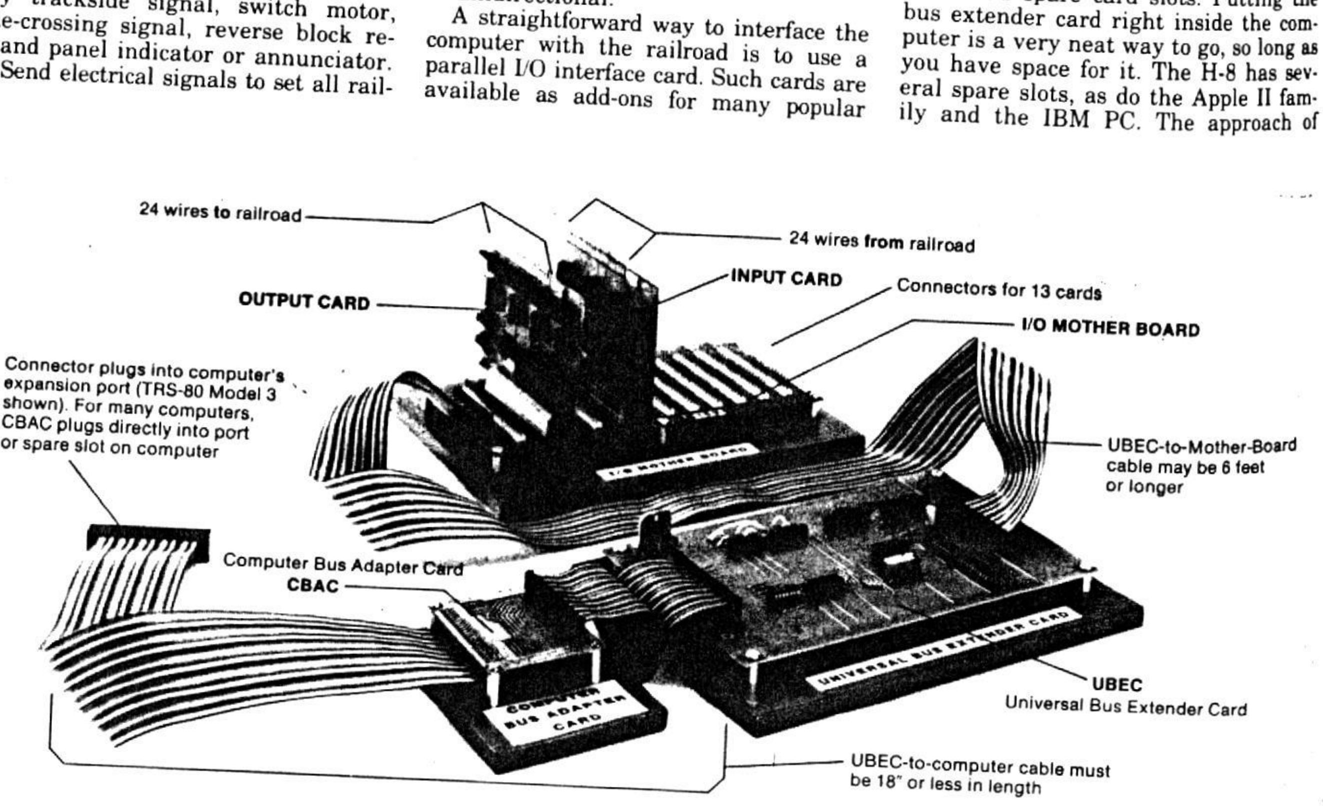

24 wires to railroad
24 wires from railroad
INPUT CARD
OUTPUT CARD
Connectors for 13 cards
I/O MOTHER BOARD
Connector plugs into computer's expansion port (TRS-80 Model 3 shown). For many computers, CBAC plugs directly into port or spare slot on computer
UBEC-to-Mother-Board cable may be 6 feet or longer
Computer Bus Adapter Card
CBAC
UBEC
Universal Bus Extender Card
UBEC-to-computer cable must be 18" or less in length
I/O MOTHER BOARD
UNIVERSAL BUS EXTENDER CARD
COMPUTER BUS ADAPTER CARD

Fig. 4. C/MRI SYSTEM. This is the hardware required to interface a home computer with a model railroad. The C/MRI series will explain how to build it, how to connect it to your computer and your layout, and how to write programs (software) so it can help you operate your model railroad.

this series, however, is not to build up a special card for each brand of computer. There are just too many different computers on the market and more coming out every year.

Also, many smaller computers don't have spare card slots anyway. To keep size and cost down they are designed with the bus coming out to an external connector, which goes by a variety of names, including expansion connector, cartridge connector, and game connector. (Note that a computer without either spare slots or an external connector is not a good candidate for interfacing with the C/MRI.)

The Universal Bus Extender Card, the UBEC, is designed to sit just outside your computer. It can plug into the expansion connector, if that's how your machine is equipped, or if your computer has spare slots, the UBEC cable can reach inside and plug into one of them.

THE C/MRI SYSTEM

Figure 4 illustrates what you'll need to build for your C/MRI. As I've said, the UBEC and its computer connection are the only parts of the system that have to be tailored to specific machines. On the other or railroad side of the UBEC, everything is the same no matter which computer you use.

Each of the input and output cards has 24 lines, three 8-bit ports each, which in turn can be connected to a varied assortment of electrical devices on your railroad, including signals, block occupancy detectors, panel switches and indicators, turnouts, and so forth. As a matter of fact, they can connect to any kind of electrical device, not just those that you'd use on a model railroad.

I'll show you how to connect various model railroad devices to the I/O cards and also how to prepare your own computer programs to operate them. This software will include signaling and automatic block power assignment, as well as test programs to make sure your C/MRI is working properly.

In designing the C/MRI I have selected components and laid out the circuit cards so that construction and testing will be a straightforward matter, and I have built and tested UBECs on most popular home computers. Also, I've written the software with primary emphasis on ease of understanding and simplicity of adaptation.

PARALLEL OR SERIAL?

As shown, the C/MRI is a parallel interface, with all the I/O cards on the extension of the computer's main bus. The alternative is a serial interface, a three-wire bus extending around your railroad to reach dispersed I/O cards. See fig. 5. I feel that the more concentrated parallel approach is best for most model railroad applications. It is lower in cost, easier to assemble and use, and as I've said, proven on several popular computers.

A serial interface has the advantage of using less wire, since the I/O cards can be located close to the devices they read or control. However, for most I/O you can use low-cost, small-diameter wire to offset the expense of running long cables. In fact, surplus multi-conductor telephone cable is ideal for most applications.

The parallel interface has a great advantage in speed, since it receives or transmits on all eight lines at once with the execution of a single computer instruction. A serial interface transmits each digital data bit individually. Not only is that slower, but since the computer itself processes data in a parallel format, the data must be converted into serial format before it can be sent. Then the receiving device must collect the data and reconstruct it into parallel format for use.

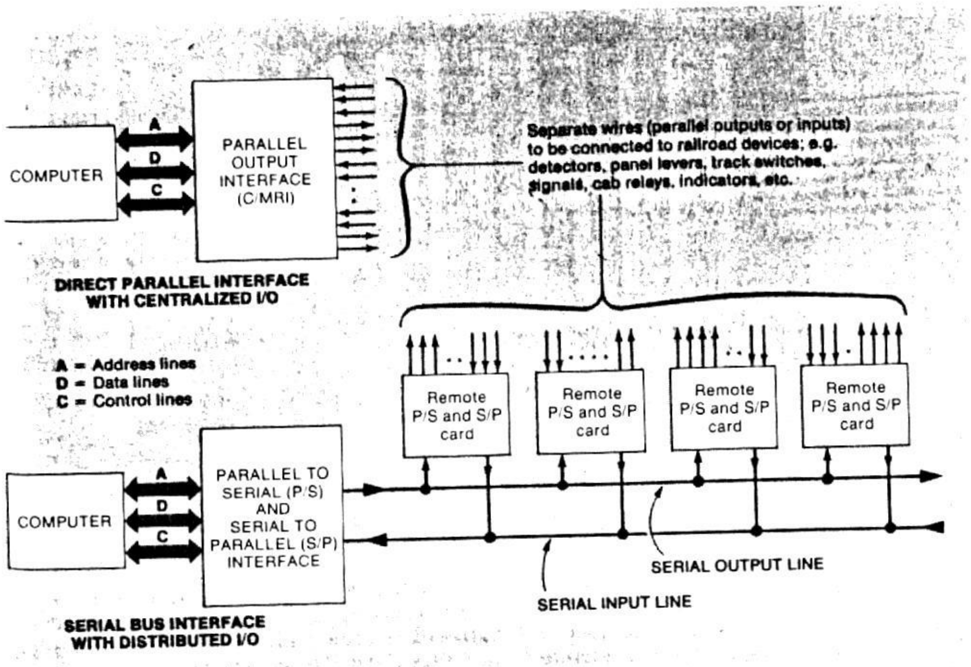

COMPUTER
A
D
C
PARALLEL OUTPUT INTERFACE (C/MRI)
Separate wires (parallel outputs or inputs) to be connected to railroad devices; e.g. detectors, panel levers, track switches, signals, cab relays, indicators, etc.
DIRECT PARALLEL INTERFACE WITH CENTRALIZED I/O
A = Address lines
D = Data lines
C = Control lines
Remote P/S and S/P card
Remote P/S and S/P card
Remote P/S and S/P card
Remote P/S and S/P card
COMPUTER
A
D
C
PARALLEL TO SERIAL (P/S) AND SERIAL TO PARALLEL (S/P) INTERFACE
SERIAL OUTPUT LINE
SERIAL INPUT LINE
SERIAL BUS INTERFACE WITH DISTRIBUTED I/O

Fig. 5 PARALLEL VS. SERIAL INTERFACING

BUNKHOUSE A

Our latest HO release, Bunkhouse A, Kit No. 230 was designed with the beginning modeler of craftsman type structure kits in mind.

All wood parts are cut to size for you, only the die-cut window openings need to be cut out with the use of a hobby knife. Bunkhouse A measures 1-1/2" x 3" and is the first of a series of beginner kits (more expierienced modelers will enjoy them too) other variations of the Bunkhouse will follow with different Door, Window and Porch arrangements. This Bunkhouse is not only fun and easy to construct it is astonishingly low priced at only $4.00.

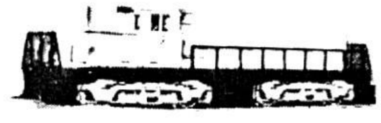
Now you have no reason not to give this one a try, or try several. The Bunkhouse and all CSM items are currently in stock (except Roadbed & Lichen) and available at your Local Hobby Shop or direct from us. Send SSAE for current price list.

Campbell Scale Models
P.O. BOX 121 • TUSTIN CA 92681

LEVENTON'S
HOBBY SUPPLY

American Flyer S-Gauge
(Formerly Dan Olson's Train House)

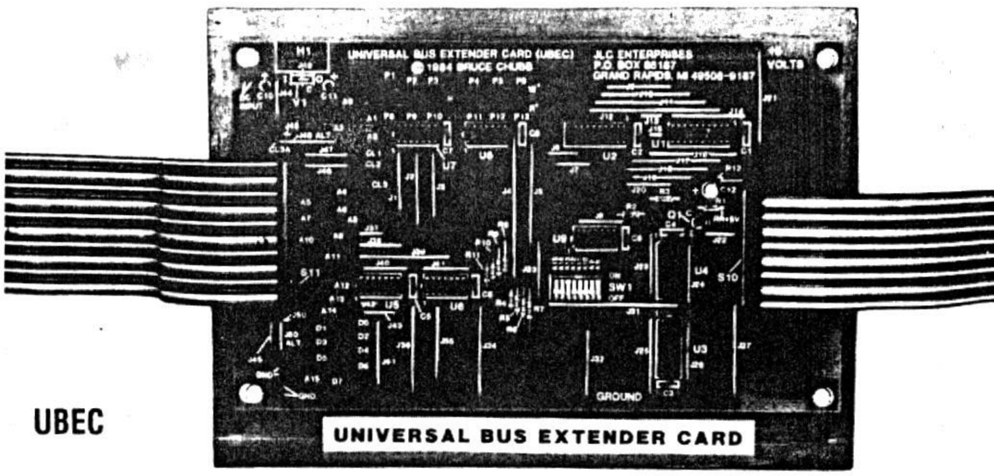
Offering:
Parts for cars, engines, accessories, sets
American Flyer - MPC (1979 to present)
Decals & Dry Transfers
Quickest Service
50 Page Sales Catalog
Repair Service
Rolling Stock (1946 - 1966)



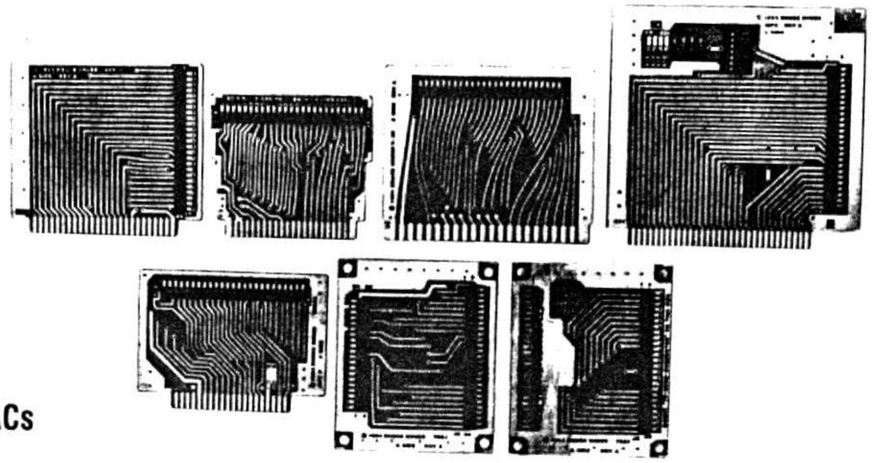
Shop hours: 10:00 a.m.-4:30 p.m.
Monday - Friday
1411 S.W. Johnson Ave.
Chehalis, WA 98532
(206) 748-3643

Coupon Special

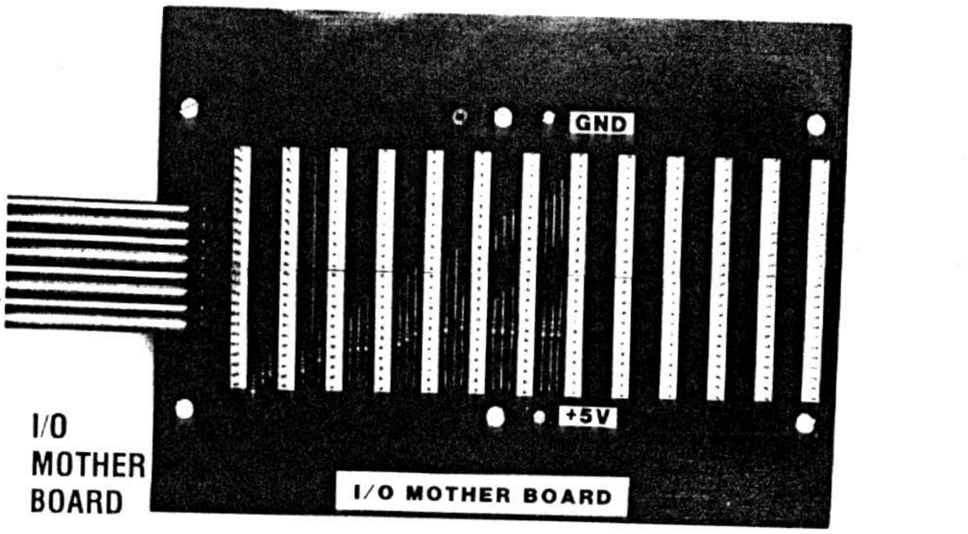
Mail in this coupon and get the 1984-85 Sales Catalog Free
Photo Copy Accepted
or Call for a Free Catalog


UNIVERSAL BUS EXTENDER CARD (UBEC)
© 1984 BRUCE CHUBB
JLC ENTERPRISES
P.O. BOX 88187
GRAND RAPIDS, MI 49508-9187
UNIVERSAL BUS EXTENDER CARD
GROUND

UBEC



CBACs



I/O MOTHER BOARD

Addresses also have to be transmitted over the serial bus to identify which device is being accessed, and these too must be converted from parallel to serial and back again at each end. A serial bus system needs a communications protocol, too, which often involves extra bits to separate addresses from data and to determine the direction of data flow.

It's not hard to see the time and cost penalty paid for using a serial bus. The interface devices have to be more complicated, too, whereas we'll see that constructing a parallel I/O port is a relatively simple task.

Toward the end of this series, space and interest permitting, I may be able to explain how you can cut down on cable wiring requirements with time-multiplexing over serial transmission cables, and remote decoding at field locations around your layout. That's likely to be useful and cost-effective only on very large model railroads, however, so we'll concentrate on the faster, simpler parallel-output approach of the basic C/MRI.

BUILDING YOUR OWN INTERFACE

Please don't worry if all this sounds too complicated now. I'm going to lead you through each step of the construction and installation, and in the course of building the hardware and programming the software, you'll find yourself learning what you need to know about computers. If you dig in and follow this series, you'll enjoy the new experiences and be proud of the end result.

For those of you who feel comfortable with electronics, I will be presenting complete circuit schematics, along with block diagrams to illustrate circuit functions. However, you won't need to pay attention to these unless you want to understand more about how the C/MRI works. The assembly instructions are written with the novice in mind, and don't even refer to the schematics or block diagrams.

If you have questions or comments as the C/MRI series proceeds, please send them to: MODEL RAILROADER, C/MRI, 1027 N. Seventh St., Milwaukee, WI 53233. MR will forward them to me for response, and I'll save those of general interest for a possible followup article after the series is complete.

If you want to learn more about computers and electronics than I can explain in the series, there are many books that can help you. I especially recommend:

- *Using TTL Integrated Circuits, Books I and II,* by Peter B. Rony, Howard W. Sans & Co. This two-volume set makes understanding and using TTL integrated circuits easy for anyone who has ever hooked up two wires to run a train set.
- *Logic and Memory Experiments using TTL Integrated Circuits, Books I and II,* by Peter B. Rony, Howard W. Sans & Co.
- *Digital Electronics, a Hands-on Learning Approach,* by George Young, Hayden Publishing Co.
- *TTL Cookbook,* by Don Lancaster, Howard W. Sans & Co.

The cost of your C/MRI will depend upon the number of interface lines you need and the functions they perform, and how much you bargain shop for parts using the numerous mail-order surplus electronic parts catalogs. I'll provide detailed parts lists and suggestions on sources.

The baseline C/MRI is very cost-effective. Assuming a basic interface with, say, 48 I/O lines, a ball-park cost is $300. (Obviously, this does not include the computer). Adding additional I/O capacity will cost in the neighborhood of $2 per line.

To build the C/MRI you will need to assemble six different kinds of circuit cards: a UBEC, a Computer Bus Adapter Card (CBAC), an I/O Mother Board, as many Input and Output cards as your railroad requires, and a Computer Output Test Card. These are shown in fig. 6. You will also need a 5VDC regulated power supply for the I/O cards; I'll cover that when we build the cards in part 4.

Fig. 6. C/MRI COMPONENTS

UBEC. The Universal Bus Extender Card is the heart of the C/MRI, and its construction will be covered in the next installment of this series. The UBEC is installed outside the computer's case, connected by a ribbon cable at most 18" long, and may be built into its own enclosure. The UBEC's job is to extend the main bus of a home computer — address, data, and control lines — to your model railroad. It lets the computer treat the railroad as part of the computer's memory or I/O structure. It's also an electrical buffer that isolates the computer's electronics from the railroad and amplifies transmissions to make weak, noisy signals strong and sharp. To let you use your computer for other purposes when you're not railroading, the UBEC automatically turns off the interface when the railroad is powered down.

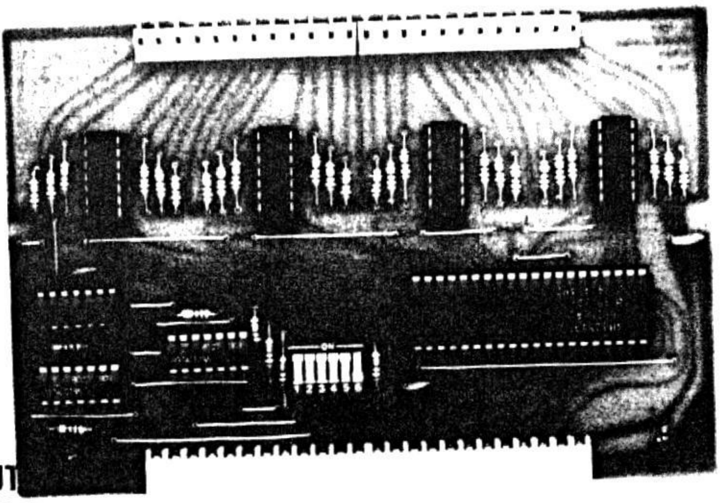
CBACs. Computer Bus Adapter Cards adapt the UBEC's ribbon cable connections to a variety of different computers. From the top left, CBACs for the Apple II (including the II+ and IIe), Commodore 64 and VIC-20, IBM PC, and TRS Color Computer plug right into the computer's spare card slots or expansion connectors. The TRS-80 Model 1 and 3 CBACs use a second short ribbon cable to connect to their machines. With some computers, like the Heathkit H-8 and the TRS-80s, you can eliminate a CBAC altogether by hard-wiring the ribbon cable to the appropriate holes in the UBEC. Part 3 of the C/MRI series will cover CBACs in detail as it explains how to connect the UBEC to different computers.

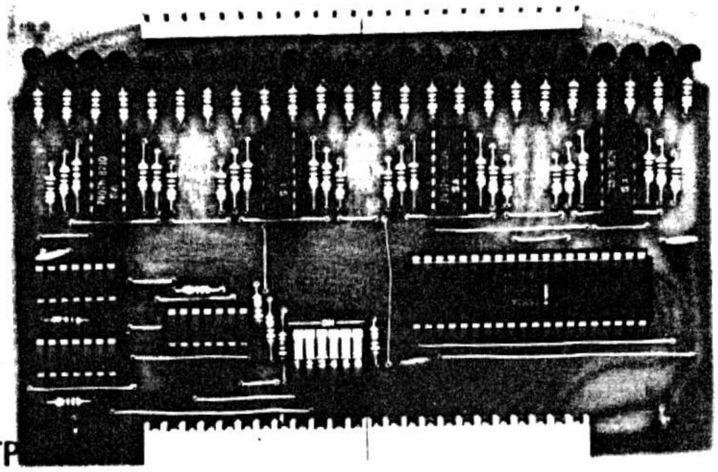
I/O MOTHER BOARD. The C/MRI's Mother Board provides 13 plug-in bus slots for input and output cards. If necessary additional Mother Boards can be ganged to the first. The Mother Board and its I/O cards can be installed at some distance from the UBEC: Bruce Chubb uses a 6-foot cable so he can roll his castered computer table away from the wall where his Mother Boards are mounted under the Sunset Valley. The Mother Board also supplies power for the I/O cards from a 5VDC power supply. Construction of the Mother Board, the input and output cards, and the power supply will be described in C/MRI part 4.

INPUT AND OUTPUT (I/O) CARDS. The C/MRI meets your railroad at the connector prongs on top of the general-purpose input and output cards, where wires from 24 separate devices can be attached to each card. The connector at the bottom of each I/O card plugs directly into the Mother Board. The cards are general purpose in that they can fit any slot in the Mother Board and are given unique addresses only when you program the six-segment Dual Inline Package or "DIP" switch.

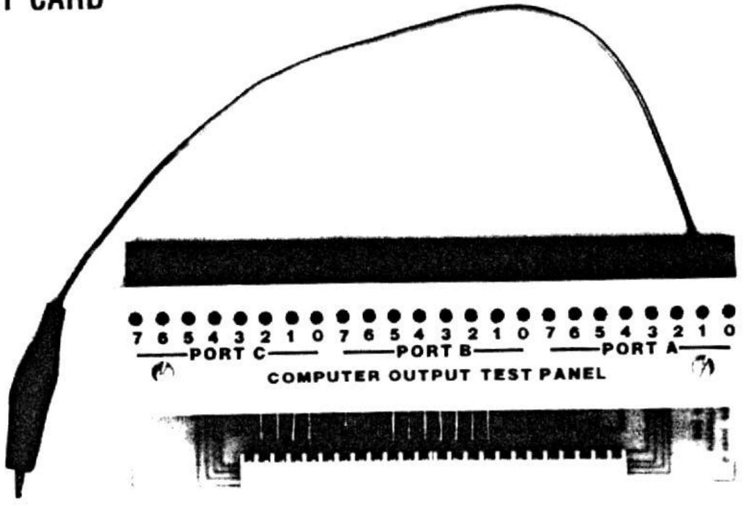
TEST CARD. You'll test your C/MRI with this simple circuit-test card that attaches to the I/O card connector prongs. The test card is described in part 5, along with two short BASIC programs that use your computer to check its own interface.

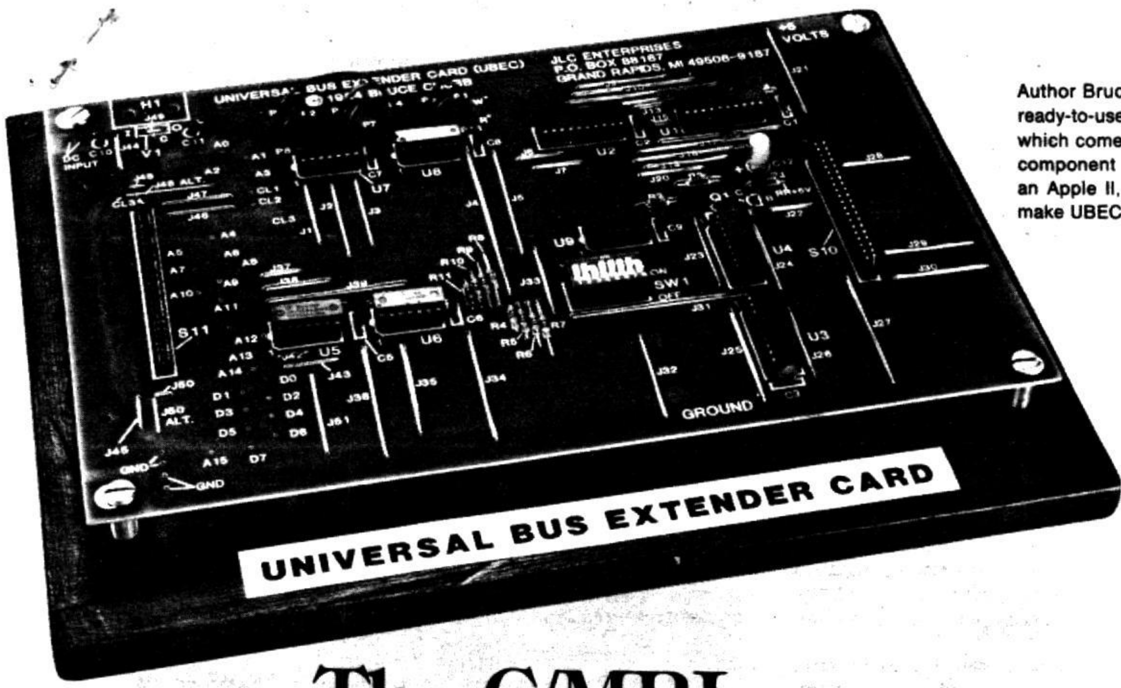
I hope you'll build a C/MRI for your railroad. The Sunset Valley's own computer interface has been on line since early 1980. It's been very dependable and has made the railroad more fun to operate. In the next installment [March MR] we'll get you started on your Computer/Model Railroad Interface by building the heart of the system, the Universal Bus Extender Card. See you then. ◊


INPUT CARD


OUTPUT CARD

TEST CARD





Author Bruce Chubb assembled this UBEC on a ready-to-use PC card made by JLC Enterprises, which comes with the parts layout printed on its component side. Bruce configured this one for an Apple II, II+, or IIe, but he explains how to make UBECs for seven other computers as well.

The C/MRI: A computer/model railroad interface

Part 2: Building the Universal Bus Extender Card.

BY BRUCE CHUBB

PHOTOS BY PAUL A. ERLER

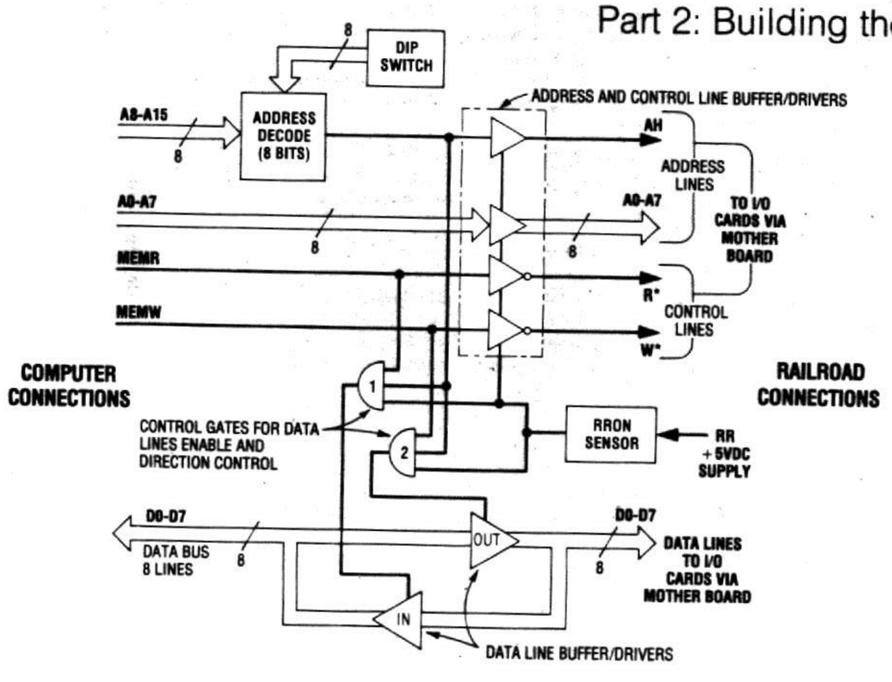
THIS MONTH we'll start building the C/MRI, the Computer/Model Railroad Interface, with step-by-step instructions for building the Universal Bus Extender Card, or UBEC. I'll also explain how to tailor its construction for a variety of computers, so we'll be ready to hook it up in C/MRI part 3 next month.

You don't need to understand electronic circuits to build the C/MRI. The circuit cards are readily available, and all you have to do is follow my instructions. But if you are interested in how it works, and to help in debugging in case you have any trouble, I'll explain what the UBEC does and give you a quick tour of its schematic. If you really just want to get started, skip ahead now to "UBEC PARTS" on page 95.

UBEC FUNCTIONS

Figure 1 is a functional look at what the UBEC does. The example uses "memory-mapped" I/O, with the computer providing separate memory-read and memory-write control lines. Next month I'll go into more detail about memory-mapped vs. port I/O — the C/MRI can use either, depending mainly on which computer you use. The UBEC's three functions are signal buffering, some address decoding, and line enabling.

In fig. 1 the arrows indicate the direction of signal flow, and double lines with a slash and number indicate that multiple parallel wires are involved — eight in our case. The low-order address lines,



CONTROL GATE LOGIC

IF:	THEN:
— Railroad is turned on (RRON high) and Railroad is being addressed (AH high) and Computer is executing a write command (MEMW high)	— Data bus is enabled for data flow from computer to railroad
— Railroad is turned on (RRON high) and Railroad is being addressed (AH high) and Computer is executing a read command (MEMR high)	— Data bus is enabled for data flow from railroad to computer
— Neither of the above sets of statements has all three conditions true	— Data bus is disconnected

Fig. 1 UBEC FUNCTIONS

A0 through A7, pass directly through the buffers from the computer side to the railroad side.

The buffers provide a degree of isolation between railroad and computer, and serve as line-driver amplifiers to make weak, noisy pulse inputs strong and sharp enough to drive several remotely located I/O cards. Without this buffering your railroad's electrical demands would so overload the computer's address bus that it couldn't work properly.

High-order address lines A8 through A15 are decoded by the UBEC. "Decoding" means that when input lines A8-A15 match the railroad's starting address code set on the DIP (*D*ual *I*nline *P*ackage) switch, decoding output line AH ("*A*ddress lines *H*igh-order bits") goes high.

I'll explain how to define and set the starting address next month; a "bit" is a single numeral in the binary code of digital logic, either a 0 or a 1. To say that any digital line or device "goes high" means that it is carrying or putting out +5VDC, the signal for the numeral 1 in digital logic. The other possible condition is "low": 0V, the signal for logic 0. When AH is high the computer is set to communicate with the railroad; when low the computer cannot communicate with the railroad.

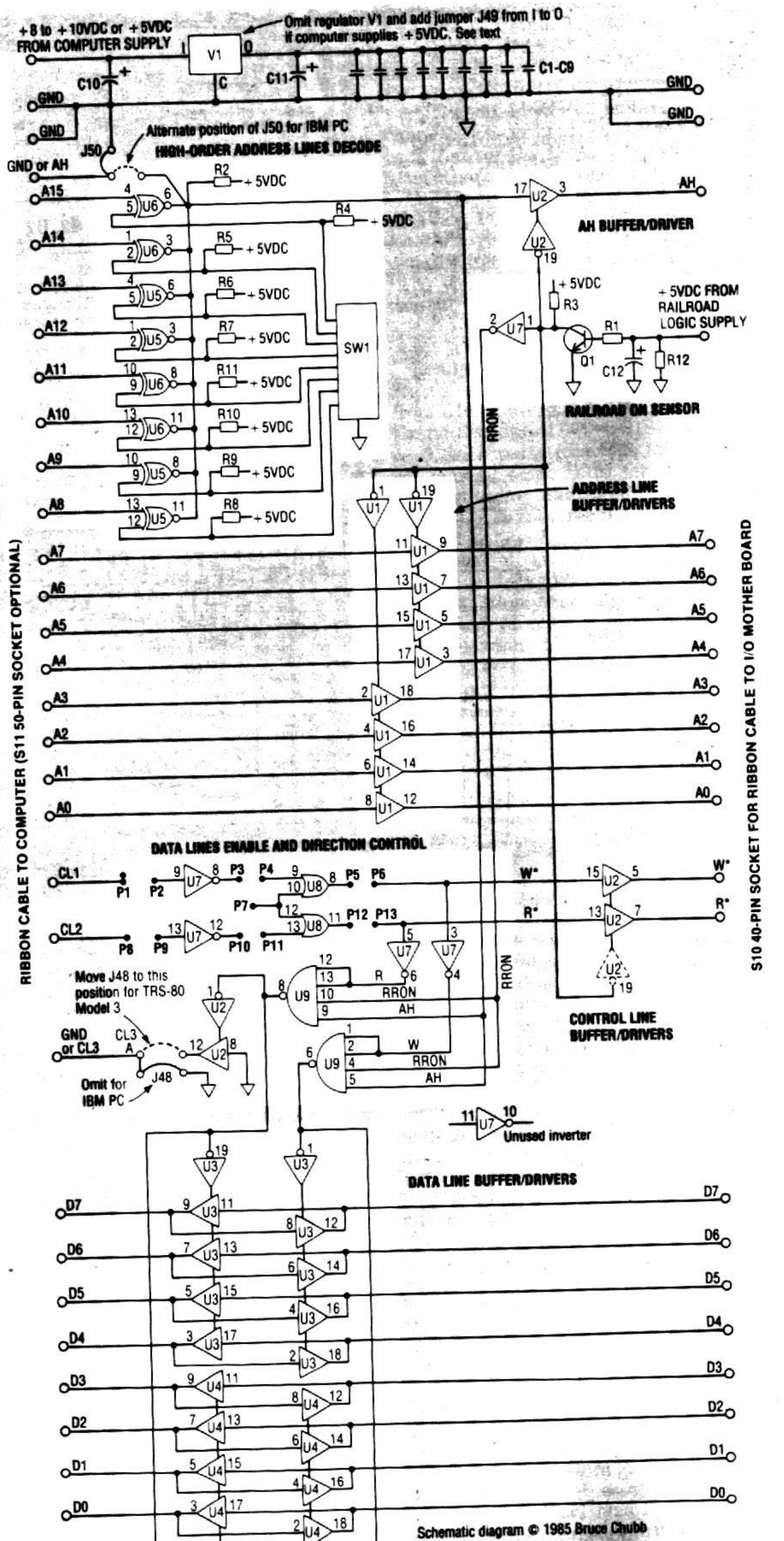
The memory-read (MEMR) and memory-write (MEMW) control lines also pass through the buffers from computer to railroad. With memory-mapped I/O we are treating the railroad as an expansion of the computer's memory, so MEMR and AH high simultaneously let the computer read data from the railroad. MEMW and AH high at the same time let the computer send data to the railroad.

The data is sent back and forth on lines D0 through D7, the data bus. Since data flow must be bidirectional, the data lines are double buffered. One group of eight buffers, labeled "IN" in fig. 1, handles data flow into the computer, and the group labeled "OUT" handles data out to the railroad.

Control Gates 1 and 2 logically determine which set of buffers is active by combining the AH address and MEMW and MEMR control signals with the railroad-on signal RRON (the UBEC's RRON sensor reads the railroad's +5VDC power supply). The control gate logic is summarized in the table included in fig. 1. Note that if the railroad is turned off, the address lines, the data bus, and the control lines cannot be enabled, so the railroad is effectively disconnected from the computer.

UBEC SCHEMATIC

The schematic in fig. 2 is simply an expansion of the function diagram in fig. 1. Digital circuits can appear complicated because there are a lot of parallel lines doing basically the same thing, but the same thing is true of conventional model railroad cab-control circuits. Have you ever tried to sit down and draw a single schematic for the block wiring on your railroad? Such a schematic would be very complex, but once you understand the basics of how one regular block and one reverse block were wired, the rest of the diagram would quickly become plain. The same is true for digital logic circuits.



+8 to +10VDC or +5VDC FROM COMPUTER SUPPLY
Omit regulator V1 and add jumper J49 from I to O if computer supplies +5VDC. See text
V1
C10
C11
C1-C9
GND
GND or AH
J50
Alternate position of J50 for IBM PC
HIGH-ORDER ADDRESS LINES DECODE
A15
A14
A13
A12
A11
A10
A9
A8
A7
A6
A5
A4
A3
A2
A1
A0
U5
U6
R2
R4
R5
R6
R7
R8
R9
R10
R11
+5VDC
SW1
AH
AH BUFFER/DRIVER
U2
U1
+5VDC FROM RAILROAD LOGIC SUPPLY
R1
R3
R12
C12
Q1
U7
RAILROAD ON SENSOR
RRON
ADDRESS LINE BUFFER/DRIVERS
RIBBON CABLE TO COMPUTER (S11 50-PIN SOCKET OPTIONAL)
S10 40-PIN SOCKET FOR RIBBON CABLE TO I/O MOTHER BOARD
DATA LINES ENABLE AND DIRECTION CONTROL
CL1
CL2
P1
P2
P3
P4
P5
P6
P7
P8
P9
P10
P11
P12
P13
U8
W*
R*
Move J48 to this position for TRS-80 Model 3
GND or CL3
CL3
A
J48
Omit for IBM PC
U9
R
W
CONTROL LINE BUFFER/DRIVERS
Unused inverter
DATA LINE BUFFER/DRIVERS
U3
U4
D7
D6
D5
D4
D3
D2
D1
D0
Schematic diagram © 1985 Bruce Chubb

TRUTH TABLE FOR RRON LOGIC

RAILROAD STATE	74LS04	
	PIN 2	PIN 1
On (+5VDC)	high (+5VDC)	low (0V)
Off (0V)	low (0V)	high (+5VDC)

Fig. 2
UNIVERSAL BUS EXTENDER CARD SCHEMATIC

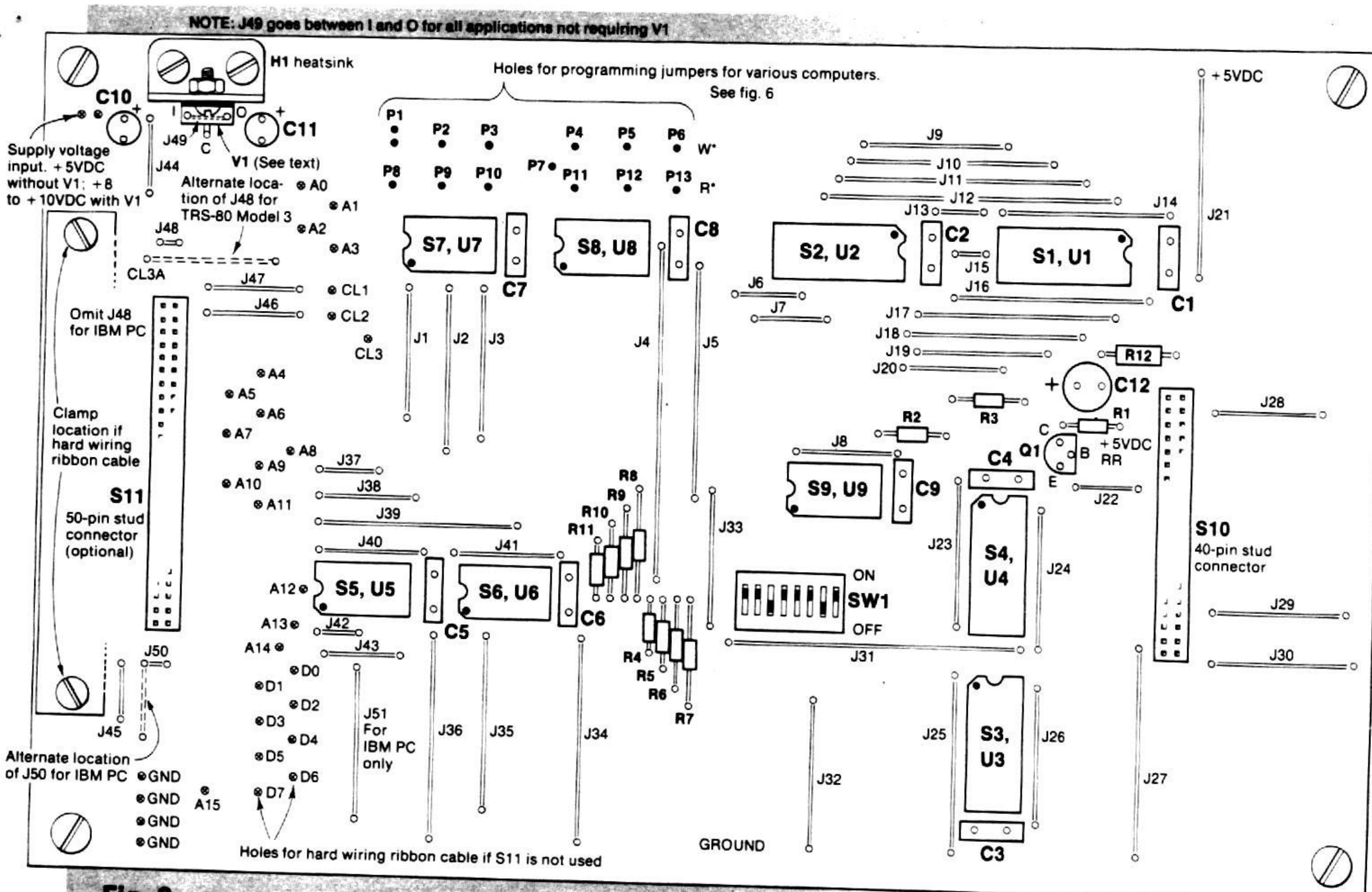


Fig. 3 UBEC PARTS LAYOUT

Integrated circuits, especially the TTL (*T*ransistor-*T*ransistor *L*ogic) types shown here, are easy to work with because you can think of them as building blocks. Each block performs a particular function, and each is designed to be "plug compatible" with the next. By connecting them in different patterns, you build up whatever overall function you wish — a crossing flasher, a signal decoder, or a computer bus extender card.

With that in mind, let's take a closer look at the UBEC schematic, starting with the address decoding function. The address bus's 16 lines, A0 through A15, allow the computer to use 16-bit address words to access 65,536 different locations. The majority of these addresses are taken up directly by the computer memory, but we need only a small subset of locations for our railroad I/O.

Address lines A0 and A1 feed directly through the UBEC to determine the internal port status on the railroad I/O cards. Lines A2 through A7 carry six-bit addresses for the individual I/O cards. Six bits can address 64 I/O cards, which is plenty — as I said last month I'm using only 23 on my Sunset Valley RR.

Lines A8 through A15 carry the high-order bits defining the starting address for the railroad I/O cards. Each of these lines is fed into the bus address decode circuit, ICs (*I*ntegrated *C*ircuits) U5 and U6. Pin 17 of the U2 IC goes high only if every input A8 through A15 matches the corresponding settings on the eight-segment DIP switch. This input is then further buffered by U2 and sent out on the railroad bus as AH.

The important point is that AH is high if the computer is addressing the railroad. With AH high, the specific railroad I/O card that is being addressed is defined by A2 through A7, and which specific I/O port by A0 and A1. More on this later in the C/MRI series when we talk about the specifics of the railroad I/O cards.

For now we'll look at the UBEC's second function: buffering. The four ICs U1, U2, U3, and U4 provide the desired buffering of the address, control, and the data bus lines. Each of these ICs is an octal buffer/line driver — "octal" means each IC can handle eight lines. These ICs provide "tri-state" logic: in addition to the usual two digital states, high or low, their output may also be a high impedance, effectively an open circuit.

The address bus and control lines are unidirectional, and so are their buffers. The data bus must be bidirectional, so two buffers are hooked up back-to-back for each of the eight data lines. If the right-most vertical column of data buffers in fig. 2 is enabled (pin 1s low), data can be transmitted from computer to railroad, while if pin 19s are low, data can be transmitted from railroad to computer. If pin 1s and pin 19s are high simultaneously, then all the buffers are open circuits, and there is no data connection between computer and railroad. Pretty straightforward, isn't it?

The next buffering substep in our circuit drives the enable inputs on the data line buffers so as to control the direction of data flow, either to and from the railroad. Remember, the CPU controls data flow on the computer's data bus with the write and read control lines MEMW and MEMR in fig. 1. These also control our data line buffers through control gates 1 and 2, and are extended to the railroad as control lines W* and R*.

The card's third purpose is to switch the whole bus extender to the open-circuit state when the railroad is turned off, in effect disconnecting the railroad while the computer is used for other tasks. This is accomplished by the railroad-on sensor. If the railroad is on transistor Q1 conducts, causing Q1's collector and pin 1 of U7 to be low. Since U7 is a 74LS04 inverter IC this causes the output on pin 2 to be high. If the railroad is off, the states are reversed. For ease in understanding, such logic is often put into a format called a "truth table," like the brief example included with the schematic in fig. 2.

This table illustrates an important concept for understanding digital circuits: regular or noninverted *vs.* inverted logic. The pin 2 output is regular logic in

UBEC PARTS LIST

J1-J51*	Jumpers, make from no. 24 uninsulated bus wire (Belden no. 8022)
R1, R2	1000Ω resistors [brown-black-red]
R3-R11*	2200Ω resistors [red-red-red]
R12	110Ω, ½W resistor [brown-brown-brown]
S1-S4	20-pin DIP sockets (Digi-Key C8920)
S5-S9*	14-pin DIP sockets (Digi-Key C8914)
S10	40-pin stud connector (Jameco 923865R)
S11**	50-pin stud connector (Jameco 923866R)
SW1*	8-segment DIP switch (Digi-Key SW1018-ND)
V1*	5V, 1.5A voltage regulator (Digi-Key LM7805CT)
H1*	Heatsink bracket for V1, make as shown in fig. 5
C1-C9*	.1μF, 50V ceramic disk capacitors (Jameco DC.1/50)
C10, C11	2.2μF, 35V tantalum capacitors (Jameco TM2.2/35)
C12	100μF, 10V radial-lead electrolytic capacitor (Digi-Key P6014 or equivalent with .098" lead spacing)
Q1	2N3904 NPN small signal transistor (Digi-Key)
P1-P13	Program jumpers, make from no. 24 insulated wire per fig. 6
U1-U4*	74LS244 octal buffer/driver, noninverting (Jameco)
U5, U6	74LS136 quad two-input exclusive-OR gates(Jameco)
U7	74LS04 hex inverter (Jameco)
U8*	74LS32 quad two-input OR gate (Jameco)
U9	74LS20 dual four-input NAND gate (Jameco)

Author's recommendations for suppliers given in parentheses above, with part numbers where applicable. Equivalent parts may be substituted. Resistors are ¼W, 5 percent except as specified, and color codes are given in brackets.
* Quantity and/or specifications vary for different computers. See table below.
** Not required if ribbon cable from computer is hard wired to UBEC

UBEC PART SUBSTITUTIONS AND/OR DELETIONS

PARTS	Heathkit H-8	TRS-80 Model 1	TRS-80 Model 3	TRS Color Computer	Apple II, II+, and IIe	VIC 20	Commodore 64	IBM PC
SW1, U5, U6, C5, C6, R4-R11, and J34-J43	YES	NO	NO	YES	YES	YES	YES	YES
V1 or J49	V1	J49	J49	J49	J49	J49	J49	J49
U8 and C8	NO	NO	YES	YES	YES	YES	YES	YES
U1, U3, and U4	240	244	244	244	244	244	244	244

Notes on jumpers: J44-J48 not required if computer cable is hard wired to UBEC, (i.e., S11 connector omitted).
Use alternate location of J48 for TRS-80 Model 3; omit J48 altogether for IBM PC.
Use alternate location of J50 for IBM PC and add jumper J51

that a high (+5V) output corresponds to a high input to the railroad-on sensor — the railroad's +5V. Pin 1 is the inverse of pin 2 and follows inverted logic: a low on pin 1 corresponds to the railroad being on. The small circles at the output of internal IC symbols in the schematic diagrams indicate that the output is inverted from the input.

Using the symbol RRON to represent the on-off state of the railroad (ON = logic 1 = +5V, and OFF = logic 0 = 0V), we would define the pin 2 output as RRON and pin 1 as RRON*. The asterisk following a symbol in digital logic circuitry means that the signal logic at that point is inverted (a bar or "macron" over a variable's symbol, i.e., $\overline{\text{RRON}}$, is another way of indicating the same thing). Note in fig. 2 that all of the UBEC outputs to the railroad I/O cards are noninverted, except for W* and R*. Thus, the railroad I/O cards work with noninverted signals for the most part.

For this example I'm assuming noninverted address and data signals from the computer, which is the case with most of them. The schematic thus shows U1, 3, and 4 as 74LS244s, since that IC doesn't have built-in logic inversion. However, my Heathkit H-8's address and data lines *are* inverted. All I had to do to use the same UBEC design was to substitute 74LS240s for these 74LS244s, providing the same functions but with inversion of the signals, and reverse the settings of DIP switch SW1. Typically, every IC function is available with either inverted or noninverted outputs.

The two 74LS20 NAND gates (U9) are used to drive the enable inputs of the bidirectional data buffers. If no data is set to be transferred, the data buffers are placed in open-circuit condition. If data is set up for transmission to the railroad or to the computer, the appropriate U9 gate enables the corresponding data buffers for flow in the required direction and places the buffers in the opposing direction in open circuit.

My H-8 uses individual voltage regulators on each board, as do some other computers. For these cases I have included circuitry for a +5VDC regulator on the UBEC. If your computer bus supplies the regulated +5VDC directly, then omit V1 as noted on the schematic and install jumper J49 in its place.

Because of the differences in control lines between machines, the UBEC has terminals CL1, CL2, and CL3, as well as 13 program jumper pads P1 through P13. By connecting CL1-3 to the appropriate computer control lines and installing appropriate jumpers, we can handle the peculiarities of various computers' control lines.

Still using our example of a computer with memory-mapped I/O and separate, noninverted memory-write and memory-read control lines, MEMW and MEMR, CL1 would connect to MEMW and CL2 to MEMR. Program jumpers would be installed from P1 to P2, P3 to P6, P8 to P9, and P10 to P13. CL3 is used only for the TRS-80 Model 3 and is otherwise connected to ground *via* jumper J48, except that for the IBM PC you'd omit J48 altogether.

On the other hand, if your computer uses port I/O, parts U5, U6, SW1, C5, C6, R4 through R11, and J34 through J43 usually aren't required. For some applications you won't need U8 (a 74LS32 quad 2-input OR gate) or C8 either.

UBEC PARTS

Ready-to-use printed circuit boards — "cards" — for the C/MRI are available from JLC Enterprises, P. O. Box 88187, Grand Rapids, MI 49508-9187. The UBEC card, with parts locations printed on its component side, sells for $42 plus a $2.50 shipping and handling charge per order. (Michigan residents must add 4 percent sales tax.)

Because of the number and size of the C/MRI cards, and because many are double-sided and thus difficult to etch on your own, circuit card artwork will not be included in the C/MRI series. However, you can obtain free, full-size art for the UBEC and other C/MRI cards by sending a business-size, stamped, self-addressed envelope to MODEL RAILROADER, C/MRI PC Layouts, 1027 N. Seventh St., Milwaukee, WI 53233.

The other parts for the UBEC are listed at the upper left, and I can recommend two mail-order parts suppliers:

- Digi-Key Corp., Box 677, Thief River Falls, MN 56701; (800) 344-4539.
- Jameco Electronics, 1355 Shoreway Rd., Belmont, CA 94002; (415) 592-8097.

Both have free catalogs for the asking.

Along with the parts list there's a table showing the substitutions and deletions to tailor the UBEC for different computers. Follow this table to assemble a UBEC to match the computer you want to interface. "YES" in the table means install the indicated parts, while "NO" means omit those parts. For U1, U3, and U4, the table gives the last three digits of the appropriate 74LS___ IC. Note that U2 is always a 74LS244.

The 50-pin stud connector S11 is for a ribbon-cable connection to the Computer Bus Adapter Card (CBAC), but it is optional. You may choose instead to hard-wire the CBAC cable directly to the UBEC, soldering the cable's conductors into holes provided in the card, as I'll explain in detail next month. If you'll be using the TRS-80 Model 1 or Model 3 computers, hard-wiring will let you save the expense of the CBAC and the extra ribbon cable; if you use the Heathkit H-8 you might as well hard-wire it as there is no CBAC for that machine.

It is easier and there will be less chance of error if you use S11 and let the CBAC make the connections to the computer for you. On the other hand, cable connectors are always less reliable than good hard-wiring. If you like, you can wait until you see what's involved next month and decide then whether or not to install S11.

Figure 3 shows the parts locations on

TEST METERS

TO BUILD AND TEST the C/MRI you will need a volt-ohm meter, or VOM. Two suitable meters available from Radio Shack are the no. 22-189 digital VOM and no. 22-204 analog VOM — either is fine for this project.

The digital meter is more expensive, but it is also easier to read and more accurate, because you have less chance of misinterpreting its measurements.

When using an analog meter you must pay careful attention to setting the scale so that you don't damage the meter or make an incorrect reading. If you don't get the prescribed voltage when testing with an analog meter, the first thing to do is make sure you're reading the correct scale.

We'll be using the VOM to perform two basic kinds of tests: continuity/resistance tests and voltage tests.

Continuity testing. First make sure that all power is turned off in the circuit being tested. Set the meter on the ohms scale (an ohm is a unit of electrical resistance), then connect one lead of the meter to one point of the circuit and the other lead to another point. The meter will measure the resistance of the circuit paths between the two points under test.

If you read 0 ohms, or very near 0, you have a solid conduction path, or a short, joining the two points. If the meter doesn't move, i.e., shows infinite resistance, you have an open circuit. Readings in between define the actual resistance of the circuit, components, and so forth under test. Get to know your meter by using it to measure the resistance of a few resistors. Compare your readings to the resistor values to make sure you are doing it correctly.

Voltage testing. We'll be making all voltage measurements with respect to a common point, called the "ground." To do this, first verify that you have selected the correct range, then connect one of the meter's probes to the ground and the other to the point where the voltage is to be measured.

Whenever possible use an alligator clip to firmly attach the ground probe to the circuit card. If you try to hold both test leads and the card at the same time while looking at the meter, you are asking for trouble, as it's just too easy to slip and short something out.

For all but a very few select tests that I'll call out in the instructions, the negative (-) probe connects to the ground, and the positive (+) probe touches the point to be tested. Make sure that your probe touches only the point to be tested and that it doesn't bridge between closely spaced leads such as IC pins. Don't let the probe slip and touch another component lead accidentally.

If you haven't made voltage measurements before, use your meter to measure your track voltage for different throttle settings to become familiar with its operation. When you measure a DC voltage with an analog meter and the needle tries to go the wrong way, reverse the two leads.

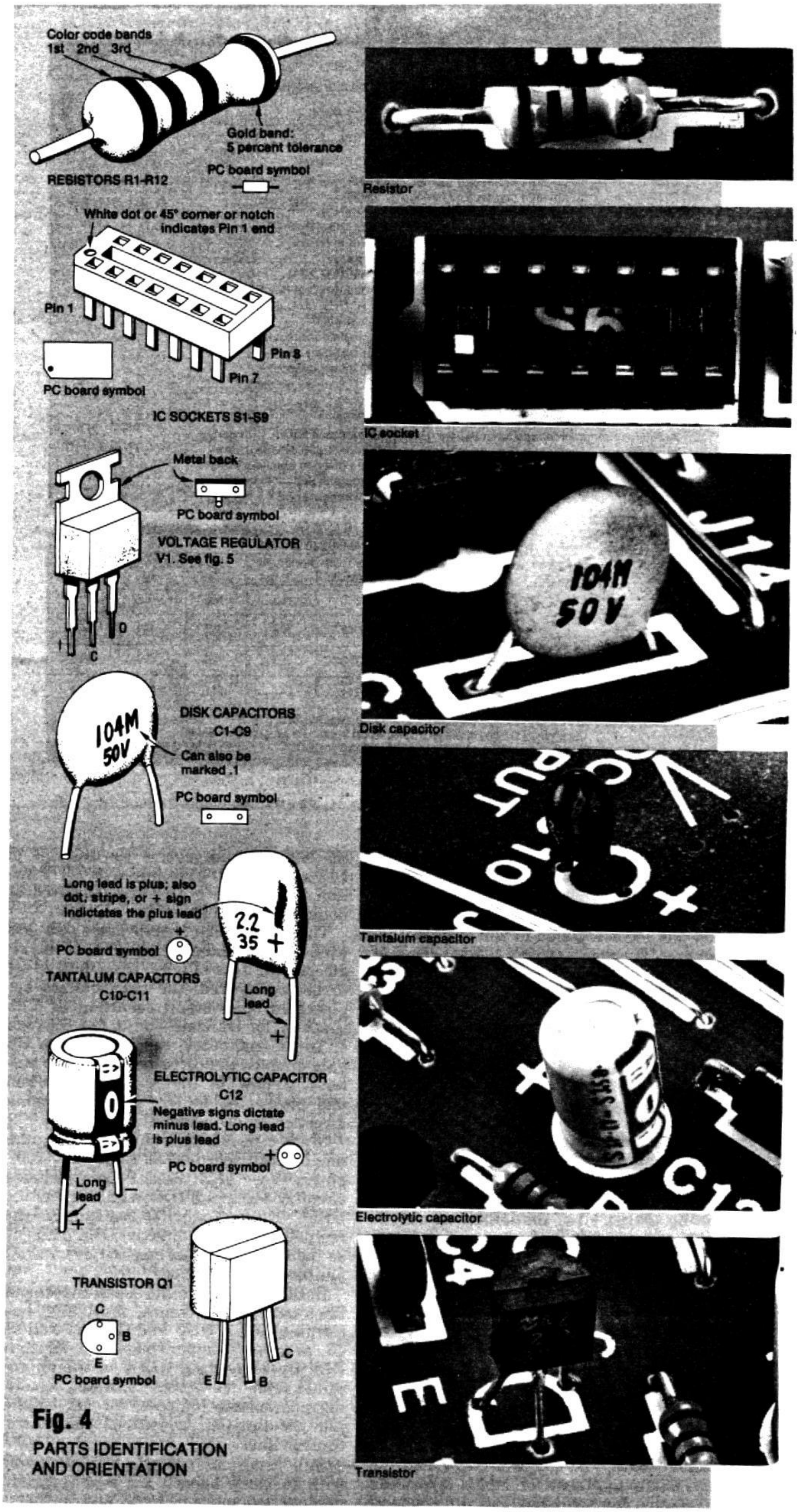


Fig. 4
PARTS IDENTIFICATION AND ORIENTATION

the top or component side of the UBEC, and you should refer to this diagram as you select and mount parts. Keep the card oriented the same way as the drawing while you work to help you place each part in its proper holes with correct orientation.

You should also take care to see that you are installing the right components. Sometimes their markings are so small you need a magnifying glass to read them, but extra effort at this stage is worthwhile. Check resistors against the color codes included in the parts list — in addition to those groups of three color bands, there should also be a gold band, indicating tolerance of ±5 percent.

For a few parts the correct orientation on the card is very important. These are capacitors C10, C11, and C12; transistor Q1; voltage regulator V1; DIP switch SW1; and both IC sockets S1-S9 and the ICs themselves, U1-U9. Figure 4 shows how to determine the correct orientation for each type of part. To remind you to check this as you build your UBEC, I've 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've soldered it to the card!

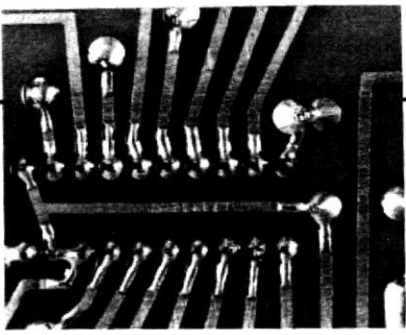
UBEC ASSEMBLY

The basic skill for building the C/MRI is PC-card soldering, and if that's a new one for you, the sidebar at the right will fill you in on the tools and techniques. Do all soldering on the back or bottom side of the card, the surface with the metal circuit traces.

The order of parts assembly on the UBEC is not critical, but for the sake of having a plan, follow the steps in order and check off the boxes as you complete each one. These instructions cover a "general case" UBEC with all components installed. I've used asterisks in the instruction steps, as in the parts list and its accompanying table, to identify parts that may be subject to substitution or deletion. Refer to the table every time you encounter an asterisk, and modify your UBEC to suit your computer.

☐ **Card test.** Use your VOM (*V*olt-*O*hm *M*eter: see "Test meters" sidebar on page 96 opposite) to test trace continuity on the circuit card. Make sure there are no open-circuit traces, such as a conductor with a hairline crack, and no shorts between adjacent traces or pads. You'll seldom find a problem in a bare card, but if you find one and correct it now you'll save hours in later debugging. If you find an open-circuit trace, solder a small piece of wire across the crack. If you find a short circuit, use a knife to scrape away the offending material.

☐ **J1 through J51*.** Form each jumper wire with sharp 90-degree bends at both ends, so that the end leads just fit in the holes. Each jumper should be straight and taut with no chance of touching an adjacent wire or component. After soldering each jumper in place, trim its leads flush with the tops of the solder tents on the back side of the card.


This closeup of the solder connections for IC socket S1 shows the kind of area where you have to be especially careful to avoid solder bridges. Study the pads before you start soldering to note which ones are joined — like the second and third from the left in the bottom row — and which separate, then make sure they're still that way when you're done.

PC CARD SOLDERING

GOOD SOLDERING is the most important requirement for building a successful C/MRI, and that starts with the right equipment. Soldering guns or large irons are absolute killers on circuit cards, but so is an iron with too little wattage. To do a good job and do it easily, what you need is a "just right" soldering iron.

I use a Weller W-TCP temperature-controlled soldering station. At around $70 it's an expensive tool, but the results it gives make it a wise investment.

You can assemble an adequate soldering station for about $20 using the following Radio Shack parts from the 1984 catalog:

- Cool-grip handle, no. 64-2080
- 33-watt heating element, no. 64-2082
- Light-duty tips (will require tinning), no. 64-2084
- Soldering iron stand and sponge holder, no. 64-2078

Do not use the Radio Shack 25-watt pencil iron; its tip temperature is too low for proper PC card soldering. And do not use a higher wattage iron as that can easily damage the PC card.

You'll also need good solder, a 60/40 rosin-core type — NEVER use acid-core solder for electronics projects. It's important, too, to use small-diameter solder in PC card work, .035" or less. Radio Shack no. 64-005 fits this description; however, so does Jameco no. SN60, and it's a better bargain on a 1-pound spool. You won't need any additional flux, as rosin-core solder has all you need for a good job.

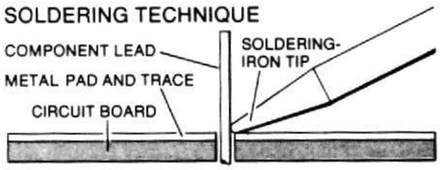
The drawings show the steps in making a good solder joint on a PC card. Both the soldering tip and the parts have to be clean for efficient heat transfer. Wipe the tip on a damp sponge after soldering every few joints, to keep it bright and shiny. If you find a component lead that has oxidized and is no longer shiny, scrape it with a knife to remove the oxidation before soldering.

For pads that are joined, heat each separately. Don't feed a blob of solder to one pad and hope it will get to the other before cooling. Heating each pad-and-lead combination separately also prevents heat damage to the components.

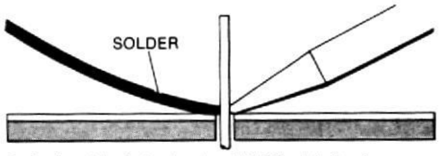
Learn to solder quickly and correctly, but don't rush either. When you complete a group of joints, look them over with a magnifying glass under a strong light and from different angles. The drawings show what to look for in good and bad joints. If you find any bad traits reheat the joint, pad and lead, and apply a bit more solder to get a good connection.

I've made soldering the C/MRI cards as easy as I could by designing good-sized cards with large pads and ample space between traces. Still, there is no substitute for careful soldering with the right equipment.

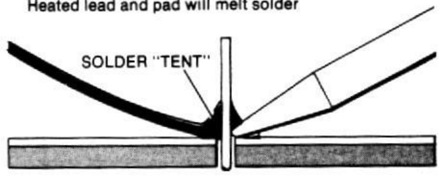
SOLDERING TECHNIQUE



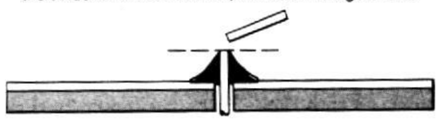
1. Press soldering-iron tip firmly against both lead and pad



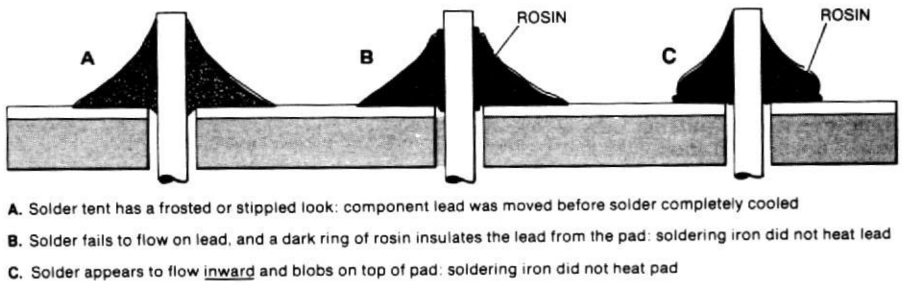
2. Apply solder to lead and pad, NOT soldering iron. Heated lead and pad will melt solder



3. Let solder melt until it flows completely around the lead and pad, forming a conical "tent." Then withdraw first the solder and then the iron, without moving the lead


4. Once the solder has cooled, clip off the excess lead flush with the top of the solder tent

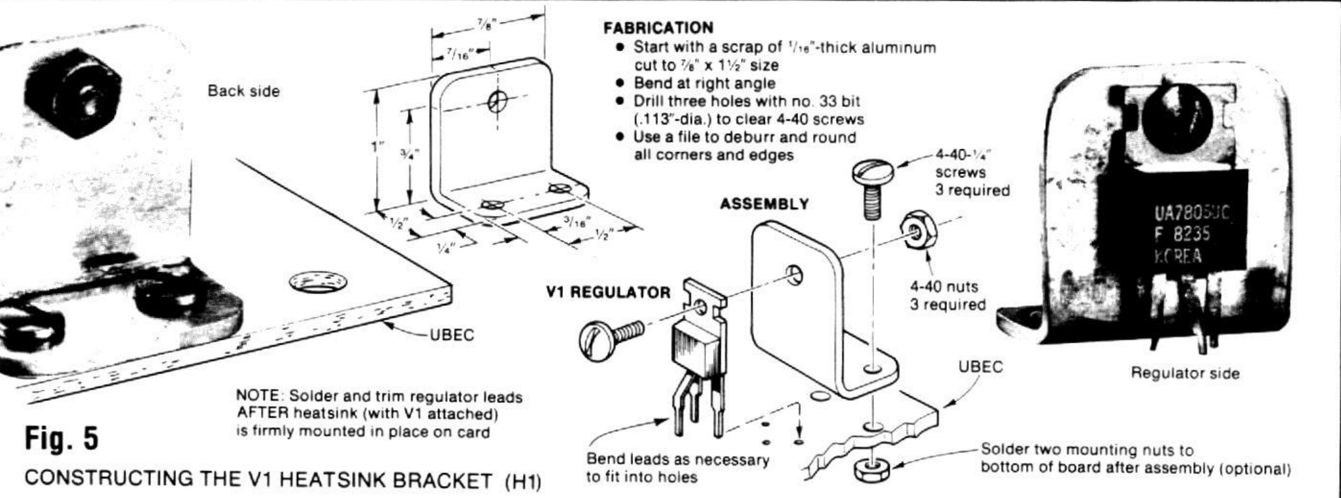
COMMON SOLDERING FAULTS



A. Solder tent has a frosted or stippled look: component lead was moved before solder completely cooled

B. Solder fails to flow on lead, and a dark ring of rosin insulates the lead from the pad: soldering iron did not heat lead

C. Solder appears to flow inward and blobs on top of pad: soldering iron did not heat pad



Back side
UBEC
7/8"
7/16"
1"
3/4"
1/2"
1/4"
3/16"
1/2"
FABRICATION
• Start with a scrap of 1/16"-thick aluminum cut to 7/8" x 1 1/2" size
• Bend at right angle
• Drill three holes with no. 33 bit (.113"-dia.) to clear 4-40 screws
• Use a file to deburr and round all corners and edges
ASSEMBLY
4-40-1/4" screws 3 required
4-40 nuts 3 required
V1 REGULATOR
UBEC
Bend leads as necessary to fit into holes
Solder two mounting nuts to bottom of board after assembly (optional)
NOTE: Solder and trim regulator leads AFTER heatsink (with V1 attached) is firmly mounted in place on card
Regulator side
UA7805UC F 8235 KOREA

Fig. 5 CONSTRUCTING THE V1 HEATSINK BRACKET (H1)

☐ **R1-R12*.** 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 the leads.

☐ **S1-S9*[+].** To avoid mixing them up, install 20-pin IC sockets S1-S4 first, then 14-pin sockets S5-S9, making certain that pin-1 orientation is correct for each. Push each socket tight against the card and hold it that way as you solder. DO NOT install the ICs at this time. We'll do some card and cable continuity checking in a later installment, and this is better accomplished without the ICs installed.

☐ **SW1*[+].** Mount the DIP switch using your VOM to be sure it is oriented so the contacts are closed when the switches are thrown toward U9. This direction will be "ON," and away from U9 will be "OFF." Hold the switch securely against the card as you solder it.

☐ **S10.** Install the 40-pin stud connector for the ribbon cable to the I/O Mother Board, holding it securely against the card as you solder.

☐ **S11.** If you decide to use this connector instead of hard-wiring, install it just as you did S10.

☐ **V1*[+].** If the +5V regulator is required for your C/MRI, install it as shown in fig. 5.

☐ **C1-C9*.** Insert with capacitor disk standing perpendicular to the card, solder, and trim.

☐ **C10-C12 [+].** Make sure that the + leads go into the + holes as shown in figs. 3 and 4. Incorrect polarity will damage these capacitors. Solder and trim.

☐ **Q1 [+].** Slightly spread the leads of this transistor to fit the three holes, making sure the center (base) lead goes into the hole closest to S10, and that the flat side of Q1 also faces this same connector. Push in only far enough to fit snugly without putting stress on the leads. Solder and trim.

☐ **P1-P13.** Install the appropriate program jumpers to suit your computer as shown in fig. 6.

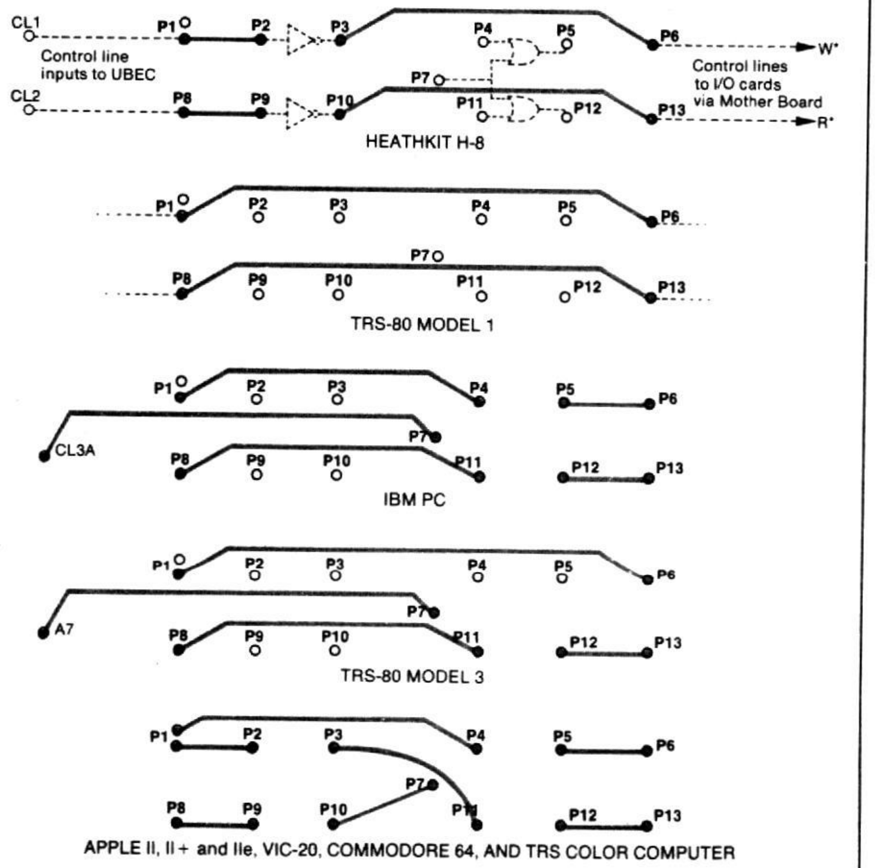
CLEANUP AND INSPECTION

For a professional-looking job and to help with inspection, spray the trace side of the card with rosin flux remover (Radio Shack 64-2324, for example) and lightly scrub the surface with a stiff-bristle brush. Then examine every solder joint closely, preferably with a magnifying glass, looking for faults like those shown in the sidebar on page 97.

Touch up any bad-looking joints by reheating them and applying additional solder as needed to form a uniform, shiny tent around each lead. Also look for and remove any solder bridges between adjacent pads and/or circuit traces.

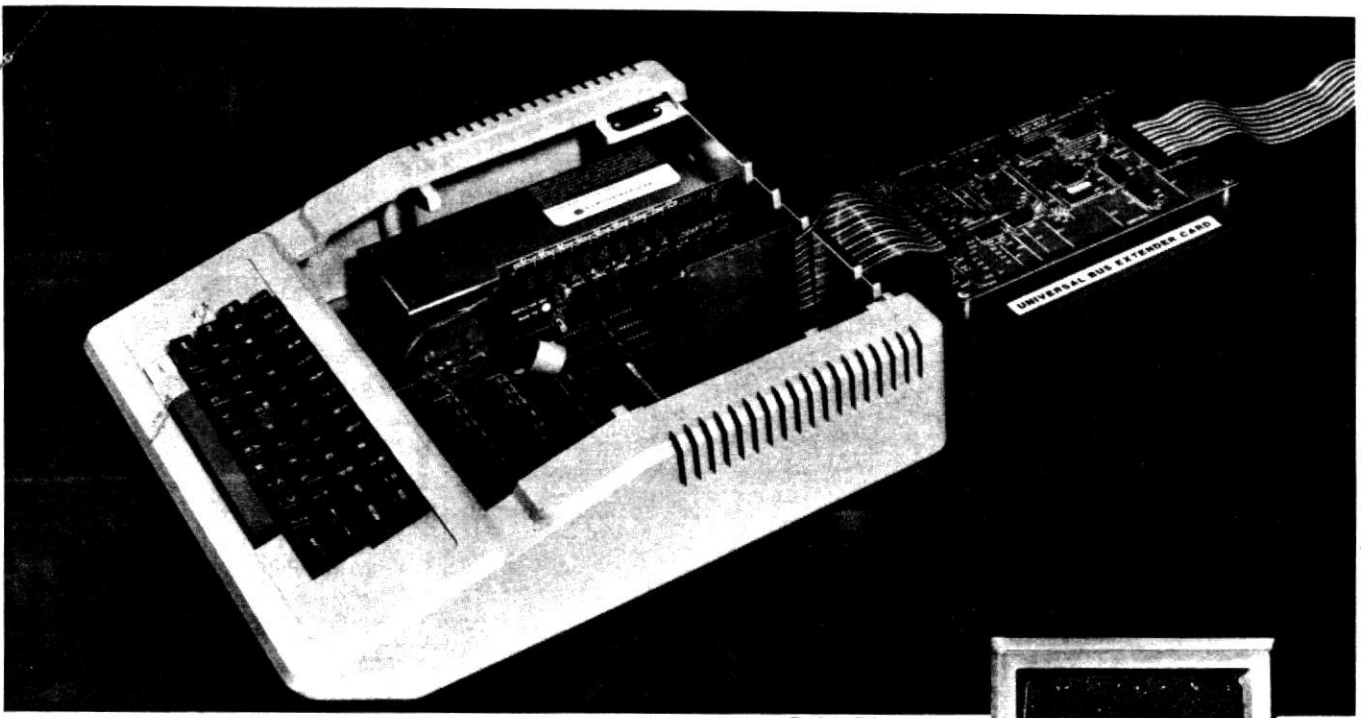
When everything looks good, spray the bottom of the card with a protective coat of clear acrylic — I like Krylon no. 1303. If you will be hard-wiring your computer cable, postpone this spray until you've completed all soldering on your UBEC.

With your Universal Bus Extender Card done, you've taken the first big step in building your own C/MRI. I'll see you next month, to show you how to connect the UBEC to your computer. ♦



CL1
CL2
Control line inputs to UBEC
P1
P2
P3
P4
P5
P6
P7
P8
P9
P10
P11
P12
P13
W*
R*
Control lines to I/O cards via Mother Board
HEATHKIT H-8
TRS-80 MODEL 1
CL3A
IBM PC
A7
TRS-80 MODEL 3
APPLE II, II+ and IIe, VIC-20, COMMODORE 64, AND TRS COLOR COMPUTER

Fig. 6 PROGRAM JUMPER WIRING Jumper wires (shown in color) required for various makes of computers. As seen from component side of UBEC


Paul A. Erler

Here's the computer end of the C/MRI, a Universal Bus Extender Card (UBEC) connected to an Apple II+ by a special Computer Bus Adapter Card (CBAC) plugged into bus slot 4 in the Apple's CPU. Author Bruce Chubb tells how to connect the UBEC to this and seven other home computers.

C/MRI-3

The C/MRI: A computer/model railroad interface

Part 3: Connecting the UBEC to your computer

BY BRUCE CHUBB

IN THIS installment we'll work on the computer end of the C/MRI, the Computer/Model Railroad Interface, and connect the Universal Bus Extender Card or UBEC to your home computer. I'll cover connections for eight popular computers, really more counting three kinds of Apples and the various "compatibles" of the IBM PC. We'll move on to the railroad end of the C/MRI next month.

As in the last installment, you don't need to understand exactly what goes on inside the computer or why the C/MRI is connected as it is. I'll give you detailed instructions, and with Computer Bus Adapter Cards and ready-to-use ribbon cables you'll find it's easy to plug the C/MRI right into most computers.

I will explain something of how and why the computers work and how the C/MRI works with them. If you run into any trouble the knowledge should be helpful. For just getting on with it, though, you can skip ahead to "RIBBON CABLE AND HARD WIRING" on page 92. When you get to the instructions for specific computers you can ignore everything except your own machine, then jump on to "UBEC ENCLOSURES" at the end of the article.

MEMORY-MAPPED OR PORT I/O

The main buses of the computers I'll cover are standardized with 16 address lines and 8 data lines, except that the IBM PC has 20 address lines. As fig. 1 shows, however, there are many differences in the control lines. For example, my Heathkit H8, the TRS-80 Model 1, and the IBM PC each have four control lines of interest: MEMORY-WRITE, MEMORY-READ, I/O WRITE, and I/O READ. Each has two possible states, high or low, the basic conditions of digital logic. "High" means +5VDC and is also known as "logic 1" or "on"; "low" means 0V and is also known as "logic 0" or "off."

The CPU sets the appropriate control line high (or low) for a fraction of a second at the instant data is to be transferred. The two memory control lines determine whether the transfer is to be to or from memory. The input/output control lines control whether the transfer is to be to or from one of the peripheral I/O devices.

On the other hand, the Apple IIs, the VIC-20, the Commodore 64, and the Color Computer don't have separate MEMORY-READ and -WRITE control lines. They use a single line, READ/WRITE*, which is set low when the CPU sends data to memory and is otherwise high. A separate line, called the clock signal, is set high at the instant the data is to be transferred.

When a computer uses any of the memory control lines to control peripheral

CONTROL LINES/SIGNALS	HEATHKIT H8	TRS-80 MODEL 1	TRS-80 MODEL 3	TRS COLOR COMPUTER	APPLE II, II+ and IIe	VIC 20	COMMODORE 64	IBM PC
Memory write	**MEMW (23)**	WR* (13)	—	—	—	—	—	**MEMW* (B11)**
Memory read	**MEMR (28)**	RD* (15)	—	—	—	—	—	**MEMR* (B12)**
I/O write	I/OW (21)	**OUT* (12)**	**XOUT* (35)**	—	—	—	—	I/OW* (B13)
I/O read	I/OR (26)	**IN* (19)**	**XIN* (33)**	—	—	—	—	I/OR* (B14)
Combined memory Read/Write*	—	—	—	**R/W* (18)**	**R/W* (18)**	**CR/W* (18)**	**R/W* (5)**	—
Clock signal	—	—	—	**E (6)**	**ϕ0 (40)**	**Sϕ2 (V)**	**ϕ2 (E)**	—

Fig. 1
CONTROL LINE VARIATIONS

Color indicates signal used with C/MRI
Asterisks indicate signals that are active low (inverted logic)
Dashes indicate signals not provided by the computer
Computer connector pinouts for signals are given in parentheses

ıput and output devices, the peripherals re treated as extensions of the memory ddress space. This is "memory-mapped O." That's as opposed to "port I/O," the ind of operation in which the I/O WRITE nd I/O READ lines control the data ransfer through addresses designated as xternal connections, or "ports."

To summarize, in the most general case ve have a choice between two types of /O: memory-mapped I/O and port I/O. If he control-line terminals on the UBEC onnect to the computer's memory control ines, we are using memory-mapped I/O, ınd if connected to the input-output conrol lines, we are using port I/O. Some computers limit you to one or the other, out other machines give you the choice.

Either will work. In BASIC programming, you use the IN (or INP) and OUT instructions with port I/O, but replace those with PEEK and POKE for memory-mapped I/O. I'll give you examples of both methods later on in the C/MRI series.

We adapt the UBEC to different control lines with the arrangement of its program jumper wires and with component selection, as covered last month in C/MRI Part 2. There will also be variations in the cable connections from each computer to the UBEC, as you'll see later in this installment.

BINARY NUMBER SYSTEM

When we talk to a computer we use the decimal number system for the most part, the digits 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9. The computer, however, uses a "binary" number system with only two digits, 0 and 1, to do everything we do with 0-9.

Computers use binary because it is very easy to handle this system electronically. Inside a computer are millions of electronic switches — transistors — that store and control the flow of electrical signals. You can think of these as if they were switches like those in fig. 2.

Each switch is separately controlled and can be either on or off, representing or sending binary digit 1 when on and 0 when off. A single switch can send only two codes, but a pair can send four, equivalent to decimal numbers 0 through 3. More switches can send more codes, twice as many with each additional switch. Three switches can send eight binary codes, and four switches (as used in CTC-16 command control) can send 16.

The short term for a single binary digit is "bit." A bit can have only two states, but they may be referred to as on or off, true or false, and +5V or 0V, as well as 1 or 0. Most home computers are eight-bit machines, which means that their electronic switches are arranged in groups of eight. A group of eight contiguous bits is called a byte, and as fig. 2 shows, it takes eight binary switches to send or store one byte of information.

All eight switches on indicate binary 11111111, equivalent to decimal 255. The decimal equivalent is the sum of the decimal equivalent values of each switch position that is turned on, and from right to left each switch doubles in decimal equivalent value. All switches off equals zero, so one byte can represent all decimal numbers between 0 and 255, a total of 256 binary codes.

Binary is good for computers but cumbersome for us, so we won't deal in binary any more than absolutely necessary. One reason binary is cumbersome is that it takes eight 1s or 0s to represent a byte. A decimal number from 0 to 255 can also define the contents of a byte, but it's difficult to see the correspondence between the decimal number and the binary bit pattern. This is where the hexadecimal number system comes to the rescue.

HEXADECIMAL NUMBER SYSTEM

In hexadecimal we have 16 "digits" to work with: 0 through 9 plus A through F. The letters are symbols used as digits, with A = 10, B = 11, C = 12, D = 13, E = 14, and F = 15. These 16 digits, 0 through F, can represent all possible bit combinations. Figure 3 shows corresponding hexadecimal, decimal, and eight-bit binary-coded numbers. A complete table would take 256 rows, 0 through 255, so I've eliminated many higher-order rows.

That we can eliminate these, yet still have all the information we need, is one reason hexadecimal is so handy. It takes only two hexadecimal digits to represent any eight-bit binary code. That's convenient, but the main advantage of hexadecimal is the ease of converting binary to hex and hex to binary — much easier than with decimal numbers.

The colored part of fig. 3 is the only part we need to understand, as everything else can be derived from it. For example, a hexadecimal 11 is binary 00010001, a hex BB is 10111011, and a hex 1B is 00011011. The left hex character defines the left four binary bits, and the right hex character defines the right four bits. This will come in very handy when you write software to use with your C/MRI.

When I use hexadecimal numbers I'll precede them with a "$," so hex number D9 will appear as $D9. This follows common programming conventions and is a good shorthand notation.

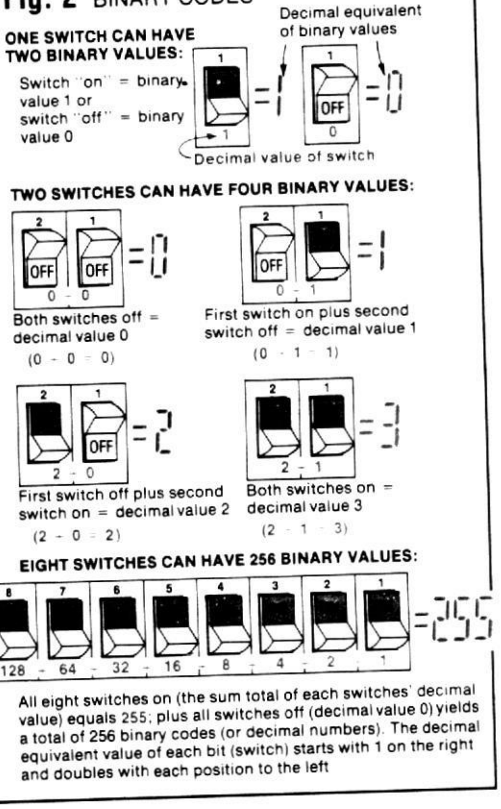
DETERMINING THE RAILROAD'S STARTING ADDRESS

To interface with a computer you must find a range of unused addresses — specific locations that can store or send one byte each, also called cells — in its memory or I/O system. Such blocks of unused addresses are called "holes," and the first address in the hole you use for your C/MRI is your railroad's starting address.

The minimum acceptable size of the hole is 4 times the number of I/O Cards you plan to use, to a maximum of 256 bytes. This is the maximum because only address lines A0 through A7 (8 bits) are decoded to select the particular I/O Card being addressed. With 8 bits you are limited to 256 bytes, providing a maximum of 256 ÷ 4 or 64 I/O Cards. Since each card controls 24 I/O lines, you could handle 1536 railroad I/O connections with a 64-card system.

With memory-mapped I/O you'll ordinarily have no problem locating a hole of 256 contiguous bytes. With port I/O most machines use only the lower eight bits for I/O addresses, and it's usually impossible to use a full 256 bytes for your C/MRI since room must be left for other peri-

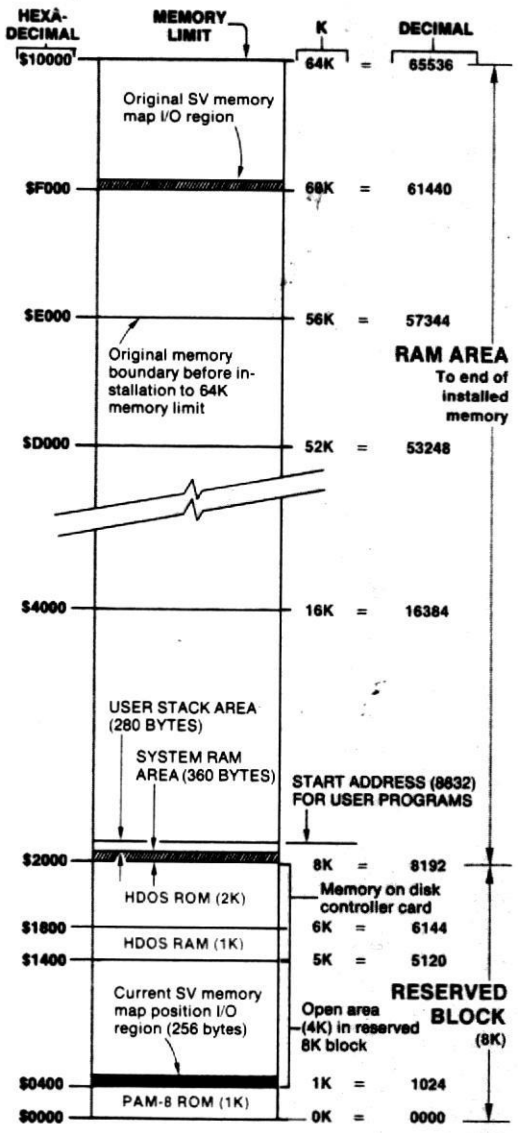
Fig. 2 BINARY CODES


Decimal equivalent of binary values
ONE SWITCH CAN HAVE TWO BINARY VALUES:
Switch "on" = binary value 1 or switch "off" = binary value 0
1
=1
OFF
=0
1
0
Decimal value of switch
TWO SWITCHES CAN HAVE FOUR BINARY VALUES:
2 1
=0
Both switches off = decimal value 0 (0 + 0 = 0)
=1
First switch on plus second switch off = decimal value 1 (0 + 1 = 1)
=2
First switch off plus second switch on = decimal value 2 (2 + 0 = 2)
=3
Both switches on = decimal value 3 (2 + 1 = 3)
EIGHT SWITCHES CAN HAVE 256 BINARY VALUES:
8 7 6 5 4 3 2 1
=255
128 + 64 + 32 + 16 + 8 + 4 + 2 + 1

All eight switches on (the sum total of each switches' decimal value) equals 255; plus all switches off (decimal value 0) yields a total of 256 binary codes (or decimal numbers). The decimal equivalent value of each bit (switch) starts with 1 on the right and doubles with each position to the left

Fig. 3 HEXADECIMAL, DECIMAL AND BINARY NUMBERS

HEXA-DECIMAL	DECIMAL	BINARY
0[illegible]	[illegible]	0000[illegible]
0[illegible]	[illegible]	0000[illegible]
0[illegible]	[illegible]	0000[illegible]
0[illegible]	[illegible]	0000[illegible]
0[illegible]	[illegible]	0000[illegible]
0[illegible]	[illegible]	0000[illegible]
0[illegible]	[illegible]	0000[illegible]
0[illegible]	[illegible]	0000[illegible]
0[illegible]	[illegible]	0000[illegible]
0[illegible]	[illegible]	0000[illegible]
0[illegible]	[illegible]	0000[illegible]
0[illegible]	[illegible]	0000[illegible]
0[illegible]	[illegible]	0000[illegible]
0[illegible]	[illegible]	0000[illegible]
0[illegible]	[illegible]	0000[illegible]
0[illegible]	[illegible]	0000[illegible]
10	16	00010000
11	17	00010001
12	18	00010010
13	19	00010011
14	20	00010100
15	21	00010101
16	22	00010110
17	23	00010111
18	24	00011000
19	25	00011001
1A	26	00011010
1B	27	00011011
1C	28	00011100
1D	29	00011101
1E	30	00011110
1F	31	00011111
	: :	
2A	42	00101010
	: :	
6B	107	01101011
	: :	
8B	139	10001011
	: :	
BB	187	10111011
BC	188	10111100
	: :	
D5	213	11010101
	: :	
FD	253	11111101
FE	254	11111110
FF	255	11111111



HEXA-DECIMAL
MEMORY LIMIT
K
DECIMAL
$10000
64K = 65536
Original SV memory map I/O region
$F000
60K = 61440
$E000
56K = 57344
Original memory boundary before installation to 64K memory limit
RAM AREA
To end of installed memory
$D000
52K = 53248
$4000
16K = 16384
USER STACK AREA (280 BYTES)
SYSTEM RAM AREA (360 BYTES)
START ADDRESS (8832) FOR USER PROGRAMS
$2000
8K = 8192
HDOS ROM (2K)
Memory on disk controller card
$1800
6K = 6144
HDOS RAM (1K)
$1400
5K = 5120
Current SV memory map position I/O region (256 bytes)
RESERVED BLOCK (8K)
Open area (4K) in reserved 8K block
$0400
1K = 1024
PAM-8 ROM (1K)
$0000
0K = 0000

Fig. 4

SAMPLE MEMORY MAP (HEATHKIT H8)

pherals on the bus. Few model railroads will need 64 I/O Cards — my Sunset Valley has only 23 — so you should be able to find enough of a hole in most machines.

If you use memory-mapped I/O for your C/MRI, and you don't have the full memory capacity of your computer installed, locating the hole is very simple. For example, if your computer has 48K of installed memory, the starting address for the block of memory-mapped I/O can start at address 48K or 49152 (48 x 1024, as 1K or "kilobyte" = 2^{10} or 1024 bytes). In hex that's $C000.

You might think that it should start at 48K + 1. However, with 48K of memory installed, the locations used are 0 through 48K-1, and the memory-mapped railroad I/O can start at 48K.

If your machine has its full memory capacity installed, i.e. 64K in a machine that can address only 64K, then you must look for a block of 256 bytes of unused memory in the "reserved" area. Typically a few kilobytes of memory are reserved by the manufacturer for various functions. There's often space for future expansion in this area, and that's where we'll find our 256 bytes.

To find the available space you need a map of your computer's memory layout. Often that's included in your computer's user manual, but if not, check with your dealer or service center.

For example, fig. 4 is the memory map of my Heathkit H8. The bottom 8K is defined as "reserved" by the manufacturer and contains the ROM (*R*ead-*O*nly *M*emory) chips for the panel monitor (PAM-8) and the Heath disk operating system (HDOS).

When I first connected the C/MRI, I had only 48K of RAM (*R*andom *A*ccess *M*emory) installed, so the lower address of unused memory 56K (8K reserved + 48K installed). For simplicity I set the railroad starting address for memory-mapped I/O at 60K, midway in the unused space. I later added the full memory and moved railroad I/O into the blank reserved area at address 1K, also shown in fig. 4.

Figure 5 shows how to determine both the binary code for address 60K and the corresponding DIP switch settings for the UBEC. Try calculating one yourself. Say we want the railroad starting address to be 1K (1024), as it is now on the SV. The resulting DIP switch positions would be A10 = 1 and all others 0.

RIBBON CABLE AND HARD WIRING

You'll need at least one and perhaps two multiconductor ribbon cables to connect the UBEC to your computer. I'll specify these as part of the instructions for connecting specific computers. Ribbon cables are easy to use, and with preassembled connectors your chance of a wiring error will be slight.

Whenever possible I've specified cables and arranged terminals to provide multiple ground connections. With my H8, for example, a 40-wire cable allows tripling each of the four ground connections. The cable conductors are 28-gauge wire, but 12 in parallel give a ground equivalent to just larger than 18-gauge wire.

If you use a Heathkit H8 like mine, you'll most likely want to hard wire it to your UBEC. With the TRS-80 Models 1 and 3, you have the choice of hard wiring or using ribbon cables with standard connectors and a Computer Bus Adapter Card (CBAC). For the other five machines I'll cover you must use a CBAC that plugs into a female edge-connector socket, so there's not much point in hard wiring them to the UBEC.

The first step in hard wiring is to make and install the clamp shown in fig. 6. Clamp the cable in place so that its last 6"-8" extend onto the UBEC. Separate the insulated conductors all the way to the cable clamp.

Next, take one conductor at a time and trim it to length to reach into the appropriate hole. Then remove about 1/4" of insulation, insert the wire end into the hole, and solder. The UBEC holes involved are A0-A15, D0-D7, CL1-CL3, ground (GND), and + supply (DC INPUT). See fig. 3 of C/MRI part 2, in last month's MODEL RAILROADER.

Which wire goes where depends on your computer, and later on I'll list the computer-pinout/UBEC connections for each machine. The best way to make sure that each conductor is correctly connected is to make end-to-end continuity checks with your VOM.

CBACS

The Computer Bus Adapter Card is designed to make the correct computer/UBEC connections for you, so it's easier than hard wiring and offers less chance of error. While cable sockets are always less reliable than good hard wiring, I'd expect that to be a smaller problem for most people than the possibility of crossed connections. In general, the best way to make the computer connections is to use a CBAC and a cable or cables with preassembled connectors, and to install the 50-pin stud connector S11 on the UBEC.

Ready-to-use CBAC PC cards are available from JLC Enterprises, P. O. Box 88187, Grand Rapids, MI 49508-9187. To order use the CBAC designations listed below, and add a $2.50 shipping and handling charge per order (Michigan residents must add 4 percent sales tax.)

Computer	CBAC	Price
TRS-80 Model 1	TRS1	$18
TRS-80 Model 3	TRS3	$18
TRS Color Computer	TRSCC	$24
Apple II, II+, or IIe	APP-II	$24
Commodore 64	C-64	$24
Commodore VIC-20	V-20	$24
IBM PC	IBPC	$32

As with all C/MRI cards, you can obtain free, full-size art for the CBACs by sending a business-size, stamped, self-addressed envelope to MODEL RAILROADER, C/MRI PC Layouts, 1027 N. Seventh St., Milwaukee, WI 53233. However, the CBACs are not easy projects for home etching. They are double-sided and need plated-through holes, and most also need edge-connector tabs.

COMPUTER CONNECTIONS

Now I'll detail the connections for each computer, but first some general advice. **Warning: Always be sure that your computer and any other related power supplies are turned off when making any kind of connections.** Then double check to make sure that your plug connections are correctly oriented BEFORE turning the power on. If you follow the instructions carefully, it's unlikely that anything in the C/MRI could damage your computer, but the worst cases would be making incorrect connections or accidental misconnections under power.

I'll specify the connector pinouts for each computer and their UBEC connections, but don't take my listings blindly. Manufacturers always reserve the right to change these and may also slip in some new capability which changes the memory mapping. In most cases I've tested the hardware and software, but your machine might be different. Check your computer's reference manual to verify every cable connection and the available port addresses or memory holes before connecting the UBEC to your machine.

I've tried to include as much information about the interface arrangements as I thought might be helpful, but the real basics of each case are shown in figs. 7-15. Double-check those pinouts, assemble the proper parts the right way, set the UBEC DIP switch as shown, and you'll very likely be home free.

HEATHKIT H8

The H8 uses an 8080A microprocessor which supports both port I/O and memory-mapped I/O. With port I/O, however, the H8 limits us to 64 "user-available," contiguous port addresses, enough for only 64 ÷ 4 or 16 I/O Cards. It's possible to sneak in more, but there's no point as the H8 is a natural for memory-mapped I/O.

The H8 has a 50-wire bus and 50-pin in-line stud connectors on its mother board with standard .100" pin spacing. Figure 7 shows the interfacing setup and lists the computer/UBEC pinouts. As I explained above, the 40-wire ribbon cable lets you use three conductors for each of the four ground connections.

TRS-80 MODEL 1

The TRS-80 Model 1 uses the Z80 8-bit microprocessor, an enhanced 8080A, which supports both port and memory-mapped I/O. The Model 1 is designed to use memory-mapped I/O for most peripheral devices, but it works out best for us to use port I/O with the C/MRI.

Port I/O works best because there's typically only one I/O port assigned by the system, at address 255 ($FF). That leaves port addresses 0 through 254 free for the C/MRI, enough for 63 I/O Cards. The Model 1's bus lines are brought out to a 40-pin male card-edge connector. The 16 address and 8 data lines are non-inverted, +5V is available, and there are separate port I/O read and write control lines, IN* and OUT* respectively.

The hookup in fig. 8 is straightforward. It uses a standard 40-wire ribbon cable with a 40-pin female card-edge connector preassembled on one end and a 40-pin female stud socket connector on the other end to connect to the CBAC. A second ribbon cable, this one with 50 conductors and female stud socket connectors on both ends, connects the CBAC to the UBEC.

You can eliminate this second ribbon cable, the CBAC, and S11 on the UBEC by hard wiring as explained above. Get a 40-conductor ribbon cable with the female connector on one end and the other end open (no connector). This alternate assembly is also shown in fig. 8, and the computer/UBEC pinouts are listed there too.

TRS-80 MODEL 3

The TRS-80 Model 3 uses the same Z80 microprocessor as the Model 1, but incorporates peripheral features such as disk drives and CRT inside the same console as the CPU. With all this capability built-in, the designers apparently chose not to bring the high-order address bus signals (A8-A15) or the memory-read and memory-write (RD* and WR*) control lines out to an external connector. Thus we must use port I/O with the Model 3.

The bus connection on the Model 3 is a 50-pin male edge connector. All lines on one side, even-numbered pins 2 through 50, are signal ground. As shown in fig. 9, a 50-wire ribbon cable with a female card-edge connector on one end and a female stud-connector socket on the other connects the computer to the CBAC. A second cable, also 50-conductor but with female stud sockets on both ends, connects the CBAC to the UBEC.

As with the Model 1 you can hard wire the UBEC connections and eliminate this second ribbon cable, the CBAC, and S11 on the UBEC. This alternate assembly is also shown in fig. 9, along with the computer/UBEC pinouts.

Note that pin 43 connects to CL3 on the UBEC. The Model 3's data bus is fully buffered, with the direction of data flow controlled by the external hardware such as our UBEC. The signal for this is called EXTIOSEL*. It must be pulled low by the user for an INP command and left high for an OUT command. The UBEC does

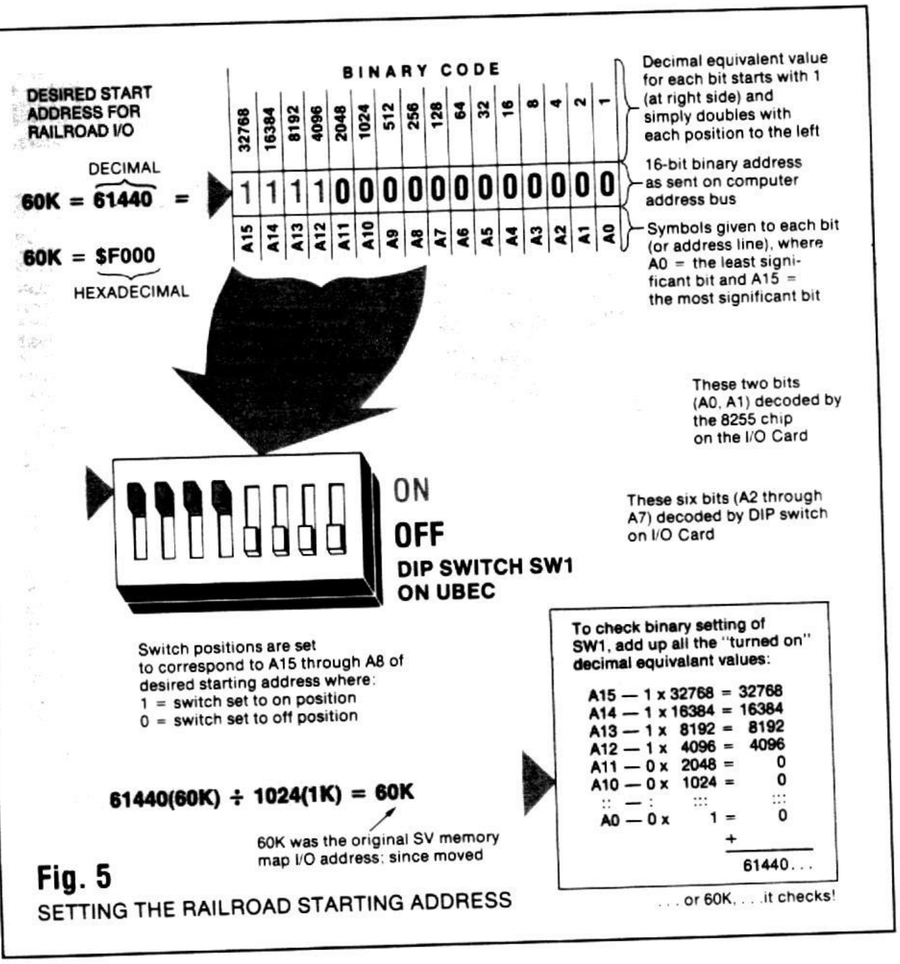
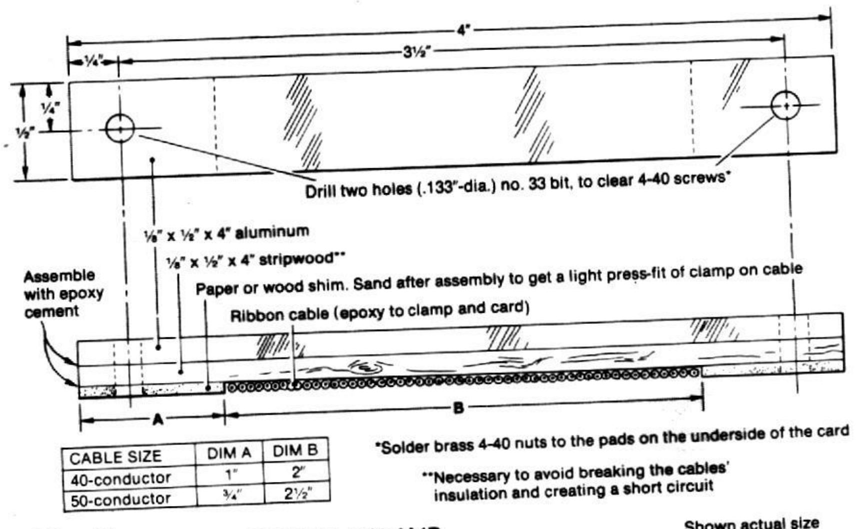
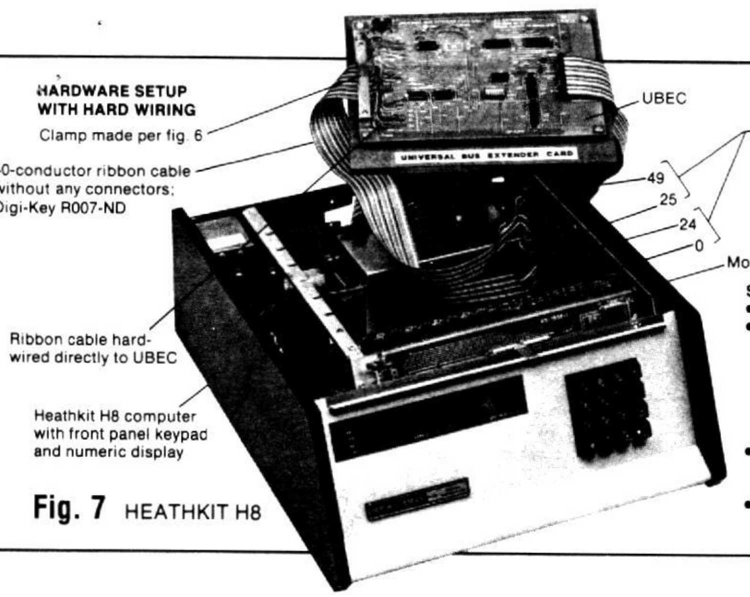

DESIRED START ADDRESS FOR RAILROAD I/O
DECIMAL
60K = 61440 =
60K = $F000
HEXADECIMAL
BINARY CODE
32768 16384 8192 4096 2048 1024 512 256 128 64 32 16 8 4 2 1
1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0
A15 A14 A13 A12 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0
Decimal equivalent value for each bit starts with 1 (at right side) and simply doubles with each position to the left
16-bit binary address as sent on computer address bus
Symbols given to each bit (or address line), where A0 = the least significant bit and A15 = the most significant bit
These two bits (A0, A1) decoded by the 8255 chip on the I/O Card
These six bits (A2 through A7) decoded by DIP switch on I/O Card
ON
OFF
DIP SWITCH SW1 ON UBEC
Switch positions are set to correspond to A15 through A8 of desired starting address where: 1 = switch set to on position 0 = switch set to off position
61440(60K) ÷ 1024(1K) = 60K
60K was the original SV memory map I/O address; since moved
To check binary setting of SW1, add up all the "turned on" decimal equivalant values:
A15 — 1 x 32768 = 32768
A14 — 1 x 16384 = 16384
A13 — 1 x 8192 = 8192
A12 — 1 x 4096 = 4096
A11 — 0 x 2048 = 0
A10 — 0 x 1024 = 0
:: — : ::: :::
A0 — 0 x 1 = 0
+
61440...
... or 60K, ... it checks!

Fig. 5 SETTING THE RAILROAD STARTING ADDRESS

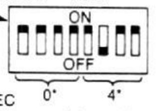

4"
3½"
¼"
¼"
½"
Drill two holes (.133"-dia.) no. 33 bit, to clear 4-40 screws*
⅛" x ½" x 4" aluminum
⅛" x ½" x 4" stripwood**
Assemble with epoxy cement
Paper or wood shim. Sand after assembly to get a light press-fit of clamp on cable
Ribbon cable (epoxy to clamp and card)
A
B
*Solder brass 4-40 nuts to the pads on the underside of the card
**Necessary to avoid breaking the cables' insulation and creating a short circuit
Shown actual size

CABLE SIZE	DIM A	DIM B
40-conductor	1"	2"
50-conductor	¾"	2½"

Fig. 6 UBEC RIBBON-CABLE CLAMP


HARDWARE SETUP WITH HARD WIRING
Clamp made per fig. 6
40-conductor ribbon cable without any connectors; Digi-Key R007-ND
UBEC
49
25
24
0
Mother board
HEATH CONNECTORS
2 each Part no. 432-948 25-hole connector shells
32 each Part no. 432-866 Spring connectors
Ribbon cable hard-wired directly to UBEC
Heathkit H8 computer with front panel keypad and numeric display

SOFTWARE SETUP
- Use memory-mapped I/O
- For railroad starting address $0400 = 1024 set UBEC DIP switch to:
  Note that DIP switch for H8 is set inverted (0 = on and 1 = off) since the high-order address bits are active low going to UBEC



- DIP switches on I/O cards will be set with positive logic since UBEC inverts the address bits going to the I/O cards
- Use PEEK and POKE instructions

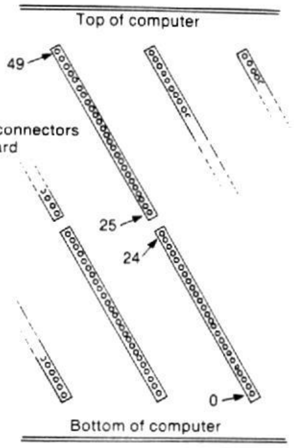


Fig. 7 HEATHKIT H8

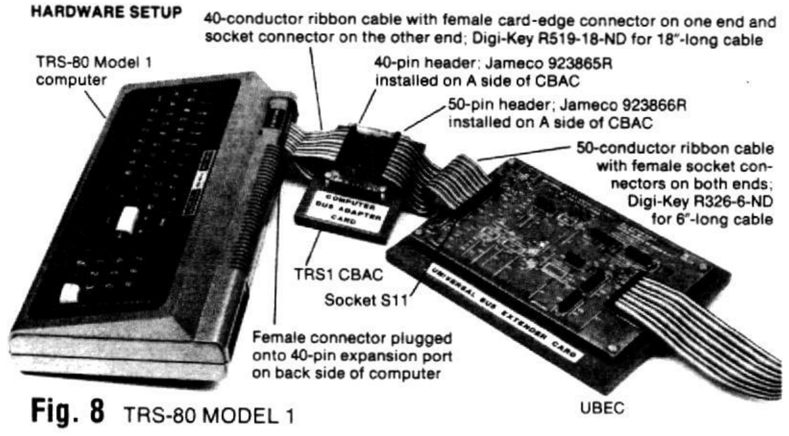
this for the Model 3 as an output via CL3.

Another consideration is that the Model 3's expansion bus can be disabled under software control. In fact, when the Model 3 is powered up, its expansion port comes up "dead," in tri-state open-circuit condition. It will not respond to any INP or OUT instruction until bit 4 at output port 236 is made a logic 1, easily accomplished with the BASIC instruction OUT 236,16.

Unfortunately, it seems that BASIC occasionally writes into this port unexpectedly, again disabling the port. For example, executing the CLS command may disable external I/O. If you are running a program on your Model 3 and your interface suddenly stops functioning, try inserting extra OUT 236,16 commands in your program loop. This may very well keep your I/O operating.

BASIC operation in the immediate mode also disables the I/O port after executing each instruction. This makes immediate-mode I/O through the C/MRI impossible, though this isn't too great a handicap because you can easily make all your C/MRI applications operate in the program mode.

Finally, the designers of the Model 3 didn't bring a +5VDC power supply out


HARDWARE SETUP
40-conductor ribbon cable with female card-edge connector on one end and socket connector on the other end; Digi-Key R519-18-ND for 18"-long cable
TRS-80 Model 1 computer
40-pin header; Jameco 923865R installed on A side of CBAC
50-pin header; Jameco 923866R installed on A side of CBAC
50-conductor ribbon cable with female socket connectors on both ends; Digi-Key R326-6-ND for 6"-long cable
COMPUTER BUS ADAPTER CARD
TRS1 CBAC
Socket S11
UNIVERSAL BUS EXTENDER CARD
Female connector plugged onto 40-pin expansion port on back side of computer
UBEC

SOFTWARE SETUP
- Use port I/O; UBEC DIP switch SW1 not required
- Use railroad starting address $0000 = 0
- Use INP and OUT instructions

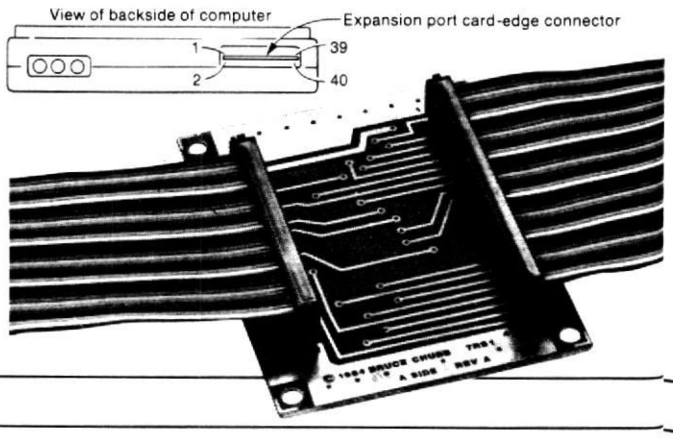
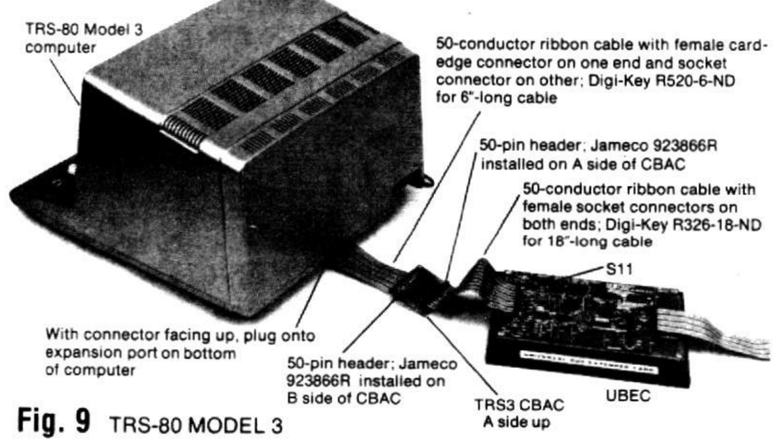
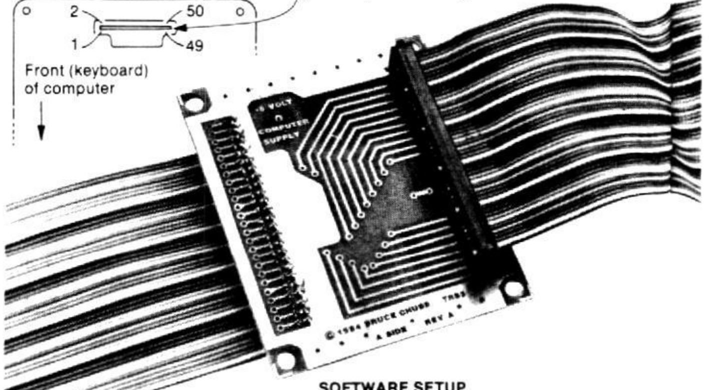

View of backside of computer
Expansion port card-edge connector
1
2
39
40
© 1984 BRUCE CHUBB TRS1 A SIDE REV A

Fig. 8 TRS-80 MODEL 1


HARDWARE SETUP
TRS-80 Model 3 computer
50-conductor ribbon cable with female card-edge connector on one end and socket connector on other; Digi-Key R520-6-ND for 6"-long cable
50-pin header; Jameco 923866R installed on A side of CBAC
50-conductor ribbon cable with female socket connectors on both ends; Digi-Key R326-18-ND for 18"-long cable
S11
With connector facing up, plug onto expansion port on bottom of computer
50-pin header; Jameco 923866R installed on B side of CBAC
TRS3 CBAC A side up
UBEC


View of underside of computer
Expansion port card-edge connector
2
50
1
49
Front (keyboard) of computer
5 VOLT TO COMPUTER SUPPLY
© 1984 BRUCE CHUBB TRS3 A SIDE REV A

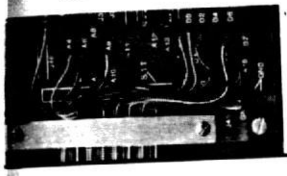
SOFTWARE SETUP
- Use port I/O; UBEC DIP switch SW1 not required
- Use railroad starting address $0000 = 0
- Use INP and OUT instructions

Fig. 9 TRS-80 MODEL 3

PINOUTS (MEMORY-MAPPED I/O)

PIN NO.	SIGNAL	PIN NO.	SIGNAL	PIN NO.	SIGNAL
0	GND	23	MEMW	38	A8*
1	GND	24	GND	39	A9*
10	D0*	28	MEMR	40	A10*
11	D1*	30	A0*	41	A11*
12	D2*	31	A1*	42	A12*
13	D3*	32	A2*	43	A13*
14	D4*	33	A3*	44	A14*
15	D5*	34	A4*	45	A15*
16	D6*	35	A5*	48	+8VDC
17	D7*	36	A6*	49	+8VDC
18	GND	37	A7*		

NOTES: MEMW connects to CL1 on UBEC
MEMR connects to CL2 on UBEC


HARD WIRING
Ribbon cable hardwired directly to UBEC

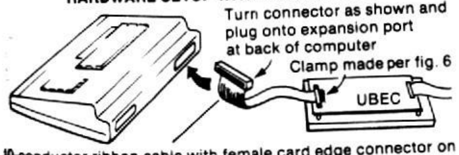
to the expansion connector either. There are three ways to take care of this. The neatest is to disassemble your Model 3 and connect an internal jumper on the main system board to bring the +5VDC out to spare pin 45 on the connector. The Model 3 CBAC is designed to receive the +5VDC input from this pin. If you don't care to disassemble your Model 3 that far, you can run a separate wire from

PINOUTS (PORT I/O)

PIN NO.	SIGNAL	PIN NO.	SIGNAL	PIN NO.	SIGNAL
8	GND	26	D3	35	A5
12	OUT*	27	A1	36	A7
18	D4	28	D5	37	GND
19	IN*	29	GND	38	A6
20	D7	30	D0	39	+5VDC
22	D1	31	A4	40	A2
24	D6	32	D2		
25	A0	34	A3		

NOTES: OUT* connects to CL1 on UBEC
IN* connects to CL2 on UBEC

HARDWARE SETUP WITH HARD WIRING

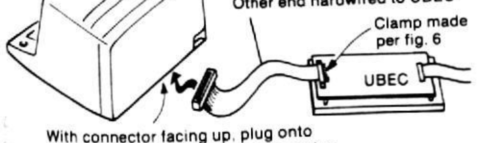

Turn connector as shown and plug onto expansion port at back of computer
Clamp made per fig. 6
UBEC
40-conductor ribbon cable with female card edge connector on one end; Digi-Key R509-36-ND. Other end hardwired to UBEC

PINOUTS (PORT I/O)

PIN NO.	SIGNAL	PIN NO.	SIGNAL	PIN NO.	SIGNAL
1	D0	15	D7	29	A6
3	D1	17	A0	31	A7
5	D2	19	A1	33	IN*
7	D3	21	A2	35	OUT*
9	D4	23	A3	43	EXTIOSEL*
11	D5	25	A4	2-50	GND
13	D6	27	A5		(all even pins)

NOTES: OUT* connects to CL1 on UBEC
IN* connects to CL2 on UBEC
EXTIOSEL* connects to CL3 on UBEC

HARDWARE SETUP WITH HARD WIRING


50-conductor ribbon cable with female card-edge connector on one end; Digi-Key R510-36-ND. Other end hardwired to UBEC
Clamp made per fig. 6
UBEC
With connector facing up, plug onto expansion port on bottom of computer

the internal +5VDC power supply, and bring it out alongside the ribbon cable to the pad provided on the CBAC.

Radio Shack reserves the right to void warranties or refuse service if any modifications are made to the computer. To get around this you can purchase a 120VAC-to-9VDC adapter. The UBEC needs about 320ma of current, and Radio Shack adapter no. 273-1651, rated at 9VDC and 500ma, works nicely.

Cut off the adapter's output plug, separate the two leads, and strip about ¼" of insulation from each. Plug the adapter into an active outlet, and use your VOM to find which lead is the +9VDC. Then unplug the adapter, and solder that lead in one of the UBEC's DC INPUT holes and the other in a GND hole.

You *must* install voltage regulator V1 on your UBEC if you use adapter power, to convert the +9VDC to the regulated +5VDC required by the ICs. Also make sure that you have the adapter plugged into an outlet that goes on and off with your computer.

The Model 3 system uses quite a few I/O addresses in the 128 through 255 range, so I've arranged the UBEC program-jumper wiring to use the lower range of 0 through 127. This is enough for 32 I/O cards and so should not be much of a limitation on your C/MRI.

To keep the C/MRI from halting CPU operation by pulling EXTIOSEL* low during reading of internal ports in the 128-256 range, I've arranged address line A7 to be jumped on the UBEC to be gated with XIN*.

TRS COLOR COMPUTER

Radio Shack's Color Computer uses the 6809E 8-bit microprocessor, with some 16-bit capability. This chip doesn't support separate I/O instructions, so we have to use memory-mapped I/O. There is a 40-pin female edge connector on the Color Computer's PC board that's handy for attaching our C/MRI to the bus. Referred to as the "ROM cartridge port" or "cartridge connector," this connector is meant for use with cartridge programs such as games, but it provides all the bus signals we need for our UBEC.

The memory map in my reference manual shows the address range $C000 through $FEFF set aside for a plug-in ROM cartridge, a region of 16,128 locations for specialized I/O. With the C/MRI plugging into the cartridge connector, this memory range is an excellent spot for our block of I/O addresses. Address $C000 corresponds to 49152 decimal or 48K and is a good starting address for the railroad.

Figure 10 shows the C/MRI setup for the Color Computer. The 50-wire ribbon cable lets you run a considerable number of parallel ground lines. The figure also lists the pinouts for the Color Computer's UBEC connections.

APPLE II, II+, AND IIe

The Apples use the 6502 8-bit microprocessor which doesn't support separate instructions for port I/O, so they handle all I/O by memory mapping. Figure 11 shows the interfacing setup for the Apples and also lists the computer/UBEC pinouts. The 50-wire ribbon cable has a standard socket connector on each end.

The Apple II, II+, and IIe are nice machines to interface with, as they include up to eight 50-pin female card-edge connectors in their cases expressly for system expansion. These "slots" are numbered 0-7, and the C/MRI can use any slot except 0, which is reserved by the manufacturer.

Although slot assignments aren't critical, a typical setup might be:

Slot	Assignment
0	RAM expansion to 64K
1	Printer interface
2	Open
3	Modem and/or 80 column display card
4	Open
5	Extra disk controller for more than two drives or hard disk
6	Disk controller for two drives
7	Open

You can see that there should be plenty of room in the Apple II for our CBAC. Either slot 4 or 5 would be a good choice, as they don't seem to be used for other applications as often as some others. Slot 7 is not as handy, as its opening in the back of the case is shorter than for the other slots, making it more difficult to run the ribbon cable.

The Apple designers made interfacing easy and attractive by providing a separate address decoding line for each of slots 1 through 7 (pin 1 in each of the 50-pin connectors) called I/O SELECT*. The CPU sets this signal low for a given slot (remember that the * means active when low) to communicate with the address range of that slot.

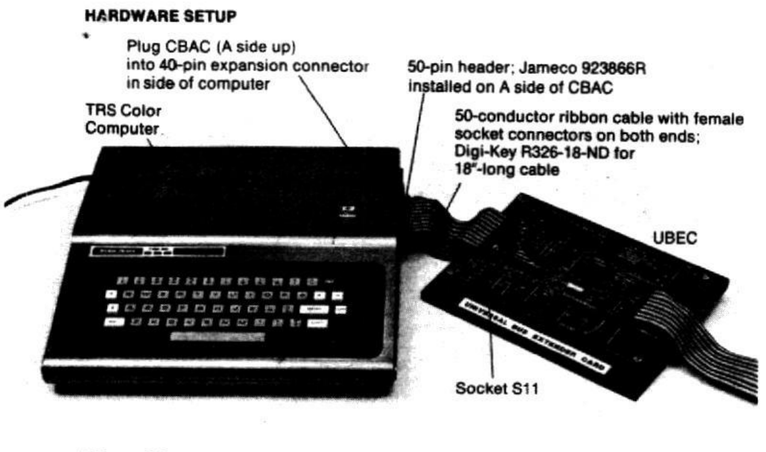
The table in fig. 11 defines the I/O SELECT* address allocations, and you can see that there's a block of 256 bytes for each slot. That corresponds exactly to the maximum 64 I/O Cards we can handle with our C/MRI.

Since this built-in decoding performs the same function as the U5-U6 circuits on the UBEC, it's tempting to use the Apple's I/O SELECT* line to drive the UBEC's AH signal line directly. This has three disadvantages. First, I/O SELECT* is inverted logic and would have to be inverted again to give positive logic for AH.

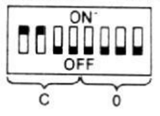
More importantly, using I/O SELECT* makes our C/MRI slot-dependent, and there are potential timing problems in that I/O SELECT* is already pregated with the phase zero clock (more about this signal later). For these reasons I recommend that you use the address decoding features of the UBEC rather than the built-in Apple slot decoding features.

Apple combines the read and write control features into a single line labeled R/W* (pin 18). R/W* is brought high to set up a read operation and low to set up a write. The phase zero clock signal (pin 40) must be gated with the R/W* line to mark the precise times when data on the data bus is valid.

Since the hookup shown is not slot dependent, you are free to use any of the 256-byte address blocks not used by


HARDWARE SETUP
Plug CBAC (A side up) into 40-pin expansion connector in side of computer
TRS Color Computer
50-pin header; Jameco 923866R installed on A side of CBAC
50-conductor ribbon cable with female socket connectors on both ends; Digi-Key R326-18-ND for 18"-long cable
UBEC
Socket S11

SOFTWARE SETUP
- Use memory-mapped I/O
- Use railroad starting address \$C000 = 49152
- Set UBEC DIP switch to:
- Use PEEK and POKE instructions



PINOUTS (MEMORY-MAPPED I/O)

PIN NO.	SIGNAL	PIN NO.	SIGNAL	PIN NO.	SIGNAL
6	E(clock)	18	R/W*	28	A9
9	+5VDC	19	A0	29	A10
10	D0	20	A1	30	A11
11	D1	21	A2	31	A12
12	D2	22	A3	33	GND
13	D3	23	A4	34	GND
14	D4	24	A5	37	A13
15	D5	25	A6	38	A14
16	D6	26	A7	39	A15
17	D7	27	A8		

NOTES: R/W* connects to CL1 on UBEC
E (clock signal) connects to CL2 on UBEC

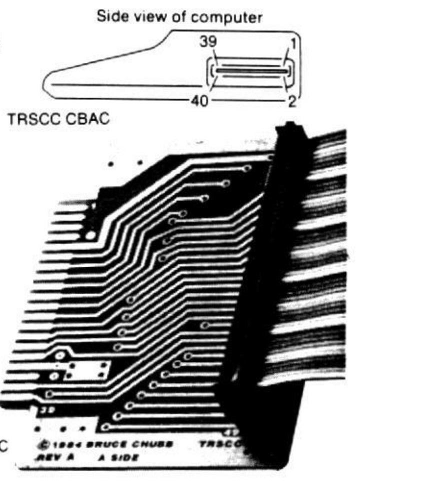
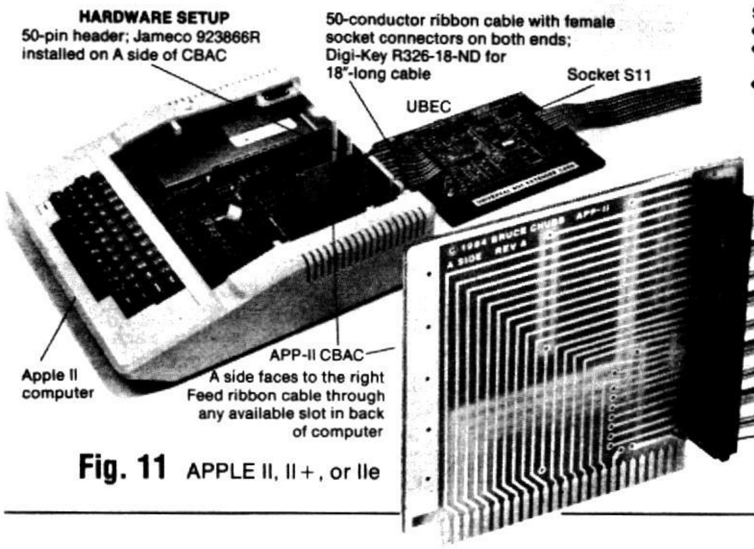
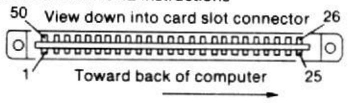


Fig. 10 TRS-COLOR COMPUTER


HARDWARE SETUP
50-pin header; Jameco 923866R installed on A side of CBAC
50-conductor ribbon cable with female socket connectors on both ends; Digi-Key R326-18-ND for 18"-long cable
Socket S11
UBEC
APP-II CBAC
A side faces to the right
Feed ribbon cable through any available slot in back of computer
Apple II computer
©1984 BRUCE CHUBB APP-II
A SIDE REV A

SOFTWARE SETUP
- Use memory-mapped I/O
- For railroad starting address (depending on slot used for CBAC) set UBEC DIP switch as shown:
- Use PEEK and POKE instructions



PINOUTS (MEMORY-MAPPED I/O)

PIN NO.	SIGNAL	PIN NO.	SIGNAL	PIN NO.	SIGNAL
2	A0	12	A10	42	D7
3	A1	13	A11	43	D6
4	A2	14	A12	44	D5
5	A3	15	A13	45	D4
6	A4	16	A14	46	D3
7	A5	17	A15	47	D2
8	A6	18	R/W*	48	D1
9	A7	25	+5VDC	49	D0
10	A8	26	GND		
11	A9	40	Φo(clock)		

NOTES: R/W* connects to CL1 on UBEC
Φo (clock signal) connects to CL2 on UBEC

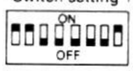
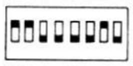
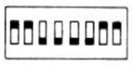
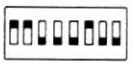
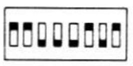
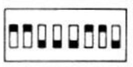
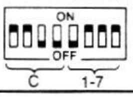
I/O select interface slot:	Address range available	UBEC DIP Switch setting
1	49408-49663 \$C100-\$C1FF	
2	49664-49919 \$C200-\$C2FF	
3	49920-50175 \$C300-\$C3FF	
4	50176-50431 \$C400-\$C4FF	
5	50432-50687 \$C500-\$C5FF	
6	50688-50943 \$C600-\$C6FF	
7	50944-51199 \$C700-\$C7FF	

Fig. 11 APPLE II, II+, or IIe

other I/O devices. Corresponding UBEC DIP-switch settings are defined in the fig. 11 address table.

VIC-20

The Commodore VIC-20 uses the same 6502 microprocessor chip as the Apple IIs and so uses memory-mapped I/O. The 6502 can address 64K, but to keep initial cost low the VIC-20 usually comes with only 29K of its memory space utilized. The remaining capacity is available through the memory-expansion connector on the rear of the case for expansion of ROM or RAM, or for specialized I/O like our C/MRI.

All of the bus lines needed for the C/MRI come out to the expansion connector, but they are a bit different from the ones I've explained so far. There are eight data lines, but they're labeled CD0 through CD7 instead of D0 through D7.

However, only the 14 low-order address lines, CA0 through CA13, are brought out to the connector. The VIC-20 manages its memory in 8 blocks of 8K each. Only 13 address bits are needed to address 8K, so 14 is 1 more than required. Each memory block is activated by one of the eight block-select signal lines: BLK1*, BLK2*, BLK3*, BLK5*, RAM1*, RAM2*, RAM3*, I/O1*, and I/O2*.

Block 5 (BLK5*) is often used for ROM programs — games — plugged into the memory expansion port, so it's a prime candidate for our C/MRI. It has a total space of 8K bytes from address 40960 through 49151. As part of the procedure for addressing this space, the VIC-20 sets block-select line BLK5* low. With BLK5* (pin 13) connected by the CBAC to the A14 address bit on the UBEC, and corresponding segment A14 of DIP switch SW1 set to "off," the UBEC can detect the computer addressing Block 5 and your railroad.

The A15 input is not used with the VIC-20 and is jumped to ground on the CBAC. Segment A15 of the UBEC DIP switch must also be "off" to achieve a comparison so that AH can go high. The rest of the high-order address bits, A8-A13, operate normally to identify the starting address for our railroad I/O.

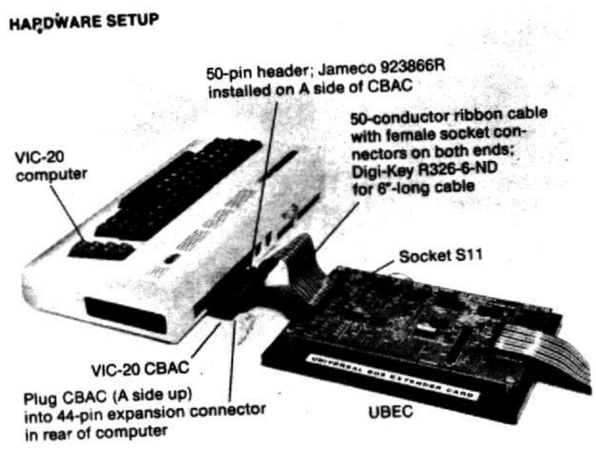
The VIC-20's memory expansion connector is a 44-pin female card-edge connector with 22 contacts on each side and .156" contact spacing. (All other computers in this series use .100" contact spacing.) Figure 12 shows the interface setup and the computer pinouts.

COMMODORE 64

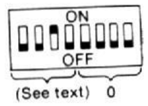
The Commodore 64 uses the 6510 microprocessor. This chip has the same basic architecture as the 6502, so like the Apple, the Commodore 64 handles all I/O by memory-mapping. The basic system's memory has 64K of RAM and 20K of ROM (including built-in BASIC software interpreter, operating system software, and standard display character set), plus 4K for I/O. That adds up to a memory capacity of 88K!

How does a computer with a 16-bit address bus address a range of 88K? (Remember that 16 bits can address only 2^{16} or 65,536 locations = 64K). It does it by overlaying different memory maps, that is, a given range of addresses may at one time be represented as RAM and at another time as ROM or I/O. Which condition is in effect at any time depends on the condition of special internal (or external) signal control lines.

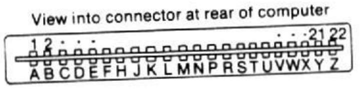
There are two I/O slots that very handily provide memory address holes for the C/MRI. These are listed in the Commodore 64 reference manual as "reserved for future expansion" and go by the names "open I/O slots nos. 1 and 2." Figure 13 defines these two slot address ranges and their corresponding UBEC SW1 DIP-switch settings. Each range includes 256 bytes, which is the ideal size for our C/MRI.


HARDWARE SETUP
50-pin header; Jameco 923866R installed on A side of CBAC
50-conductor ribbon cable with female socket connectors on both ends; Digi-Key R326-6-ND for 6"-long cable
VIC-20 computer
Socket S11
VIC-20 CBAC
Plug CBAC (A side up) into 44-pin expansion connector in rear of computer
UBEC

SOFTWARE SETUP
- Use memory-mapped I/O
- Use railroad starting address $A000 = 40960; set UBEC DIP switch to:
- Use PEEK and POKE instructions



View into connector at rear of computer



PINOUTS (MEMORY-MAPPED I/O)

PIN NO.	SIGNAL	PIN NO.	SIGNAL	PIN NO.	SIGNAL
1	GND	18	CR/W*	J	CA6
2	CD0	21	+5VDC	K	CA7
3	CD1	22	GND	L	CA8
4	CD2	A	GND	M	CA9
5	CD3	B	CA0	N	CA10
6	CD4	C	CA1	P	CA11
7	CD5	D	CA2	R	CA12
8	CD6	E	CA3	S	CA13
9	CD7	F	CA4	V	Sϕ2(clock)
13	BLK5*	H	CA5	Z	GND

NOTES: CR/W* connects to CL1 on UBEC
Sϕ2 (clock signal) connects to CL2 on UBEC
BLK5* connects to A14 on UBEC

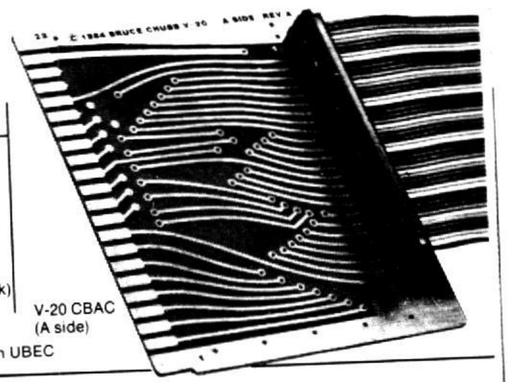
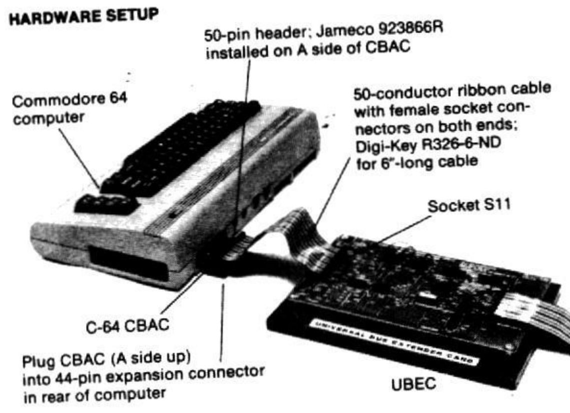
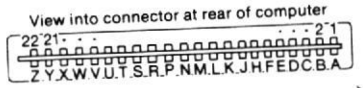


Fig. 12 VIC-20


HARDWARE SETUP
50-pin header; Jameco 923866R installed on A side of CBAC
Commodore 64 computer
50-conductor ribbon cable with female socket connectors on both ends; Digi-Key R326-6-ND for 6"-long cable
Socket S11
C-64 CBAC
Plug CBAC (A side up) into 44-pin expansion connector in rear of computer
UBEC

SOFTWARE SETUP
- Use memory-mapped I/O
- Set UBEC DIP switch for desired railroad address depending on I/O slot used; see address table:
- Use PEEK and POKE instructions

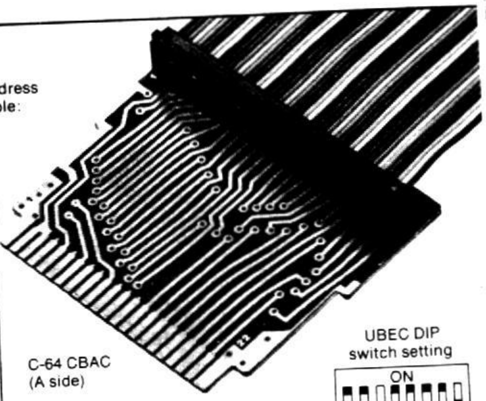
View into connector at rear of computer



PINOUTS (MEMORY-MAPPED I/O)

PIN NO.	SIGNAL	PIN NO.	SIGNAL	PIN NO.	SIGNAL
1	GND	21	D0	P	A8
2	+5VDC	22	GND	R	A7
3	+5VDC	A	GND	S	A6
5	R/W*	E	ϕ2(clock)	T	A5
14	D7	F	A15	U	A4
15	D6	H	A14	V	A3
16	D5	J	A13	W	A2
17	D4	K	A12	X	A1
18	D3	L	A11	Y	A0
19	D2	M	A10	Z	GND
20	D1	N	A9		

NOTES: R/W* connects to CL1 on UBEC
ϕ2 (clock signal) connects to CL2 on UBEC



AVAILABLE ADDRESS HOLES

Open I/O slot no.	Address range available	
	Decimal	Hexadecimal
1	56832-57087	$DE00-$DEFF
2	57088-57343	$DF00-$DFFF

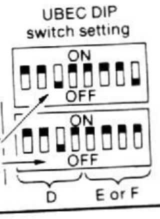


Fig. 13 COMMODORE 64

The bus lines come out at the expansion port connector at the rear of the Commodore 64. This connector, also known as the game connector, is a 44-pin female card-edge connector with 22 contacts on each side and .100" contact spacing. It provides all the bus signals we need for our UBEC.

Figure 13 shows the setup for the Commodore 64 and lists the computer pinouts. The 50-wire ribbon cable lets you run a considerable number of parallel ground lines.

IBM PC AND COMPATIBLES

The Cadillac of the computers covered in this series is the IBM PC, and it's becoming a pseudo-standard for other manufacturers to emulate. It uses the Intel 8088 which, thanks to IBM, has become the most popular "16-bit" microprocessor, in contrast to all the other machines we've covered, which use 8-bit structures.

The 8088's data bus is still 8 bits, however, whereas a true 16-bit processor would have a 16-bit bus. This is an advantage for us since our UBEC is set up for an 8-bit bus. The address bus of the IBM PC has 20 bits instead of 16, allowing the PC to directly address up to 1,048,576 bytes of memory, 16 times that provided by typical 8-bit home computers! The 8088 supports both memory-mapped and port I/O, with up to 65,536 I/O ports available.

The standard PC system board holds up to 262,144 bytes of RAM (65,536 if built before May 1983) and 40,960 bytes of ROM containing the BASIC interpreter, I/O drivers, and so on. There are five expansion slots on the standard machine and eight on the XT version. We can plug our railroad interface into any of these except slot 8 of the XT, which is wired differently.

There's usually a disk controller card in one slot, a video card takes up another, and, in larger systems, a multi-function I/O-memory-expansion-clock/calendar card occupies a third. That still leaves at least two slots, or five in the XT, and our C/MRI needs just one.

The question with the PC is whether to use memory-mapped or port I/O. This machine can address 2^{10} or 1024 I/O ports, four times the port I/O of any other computer we've considered, and four times what the C/MRI can use. However, in spite of this seeming abundance, the PC's port I/O isn't as attractive (for the C/MRI) as you might think. Depending on the individual configuration of your machine, it can be hard to find a contiguous block of 256 unused addresses.

The PC's memory capacity makes memory-mapped I/O the best choice, but the UBEC is not set up to decode the PC's four highest address bits, A16-A19. I've gotten around that with a CBAC that decodes these bits. Figure 14 includes the schematic, parts layout, and parts list for this special CBAC. The decoding circuitry is just like the UBEC's, but only four bits wide instead of eight. To assemble this CBAC, insert components on the A side of the card and solder them on the B side.

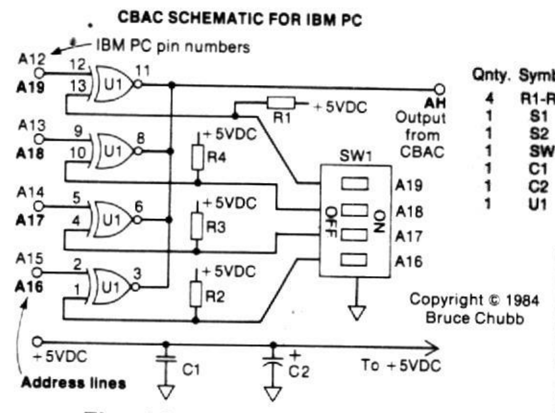
☐ **R1-R4.** Bend and insert the leads of each resistor, solder, and trim.

☐ **S1 [+].** Be sure to install this DIP socket with pin-1 orientation as shown on the parts layout and the CBAC itself. Hold the socket tightly against the card as you solder it in place. DO NOT insert IC U1 at this time, as we'll be doing some testing later on that is best accomplished without the IC installed.

☐ **S2.** Insert the 50-pin stud connector and hold it tightly against the card as you solder it in place.

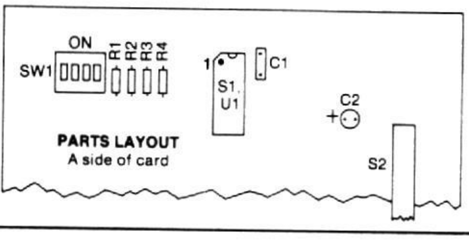
☐ **SW1 [+].** Install the DIP switch, using your VOM to make sure it is oriented so that the contacts are closed when set toward the "ON" label on the CBAC. Hold the switch tightly against the card as you solder it.

☐ **C1.** Insert with the capacitor disk perpendicular to the card, solder, and trim.



PARTS LIST FOR IBMPC CBAC

Qnty.	Symbol	Description
4	R1-R4	2200Ω, ¼W, 5 percent resistors [red-red-red]
1	S1	14-pin DIP socket (Digi-Key C8914)
1	S2	50-pin stud connector header (Jameco 923866R)
1	SW1	4-segment DIP switch (Digi-Key SW1014-ND)
1	C1	.1µF, 50V ceramic disk capacitor (Jameco DC.1/50)
1	C2	2.2µF, 35V tantalum capacitor (Jameco TM2.2/35)
1	U1	74LS136 quad two-input exclusive-OR gate (Jameco)



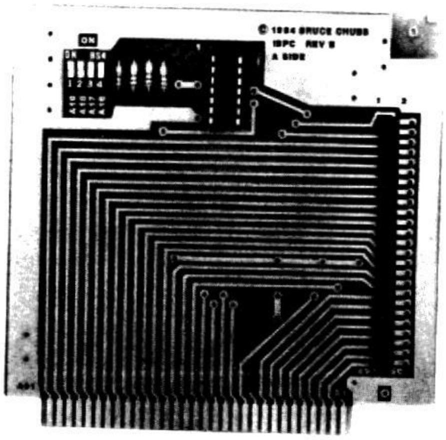


Fig. 14 SPECIAL CBAC FOR THE IBM PC

☐ **C2 [+].** Insert with positive lead in the hole nearest S1, as shown on the parts layout. Incorrect polarity will damage this capacitor. Solder and trim.

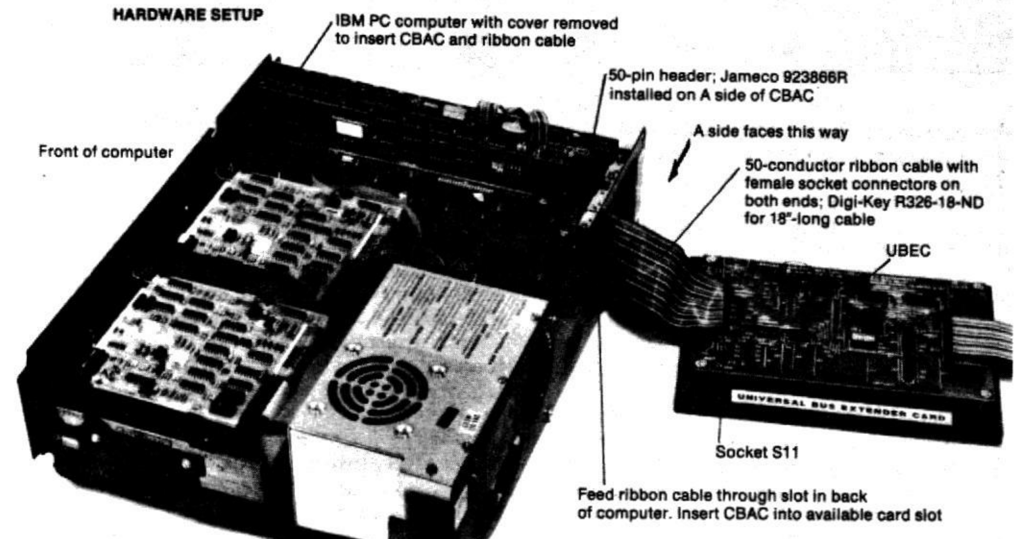
Figure 15 shows the setup for the IBM PC, along with the listing of computer/UBEC pinouts. The 50-wire ribbon cable lets you run a considerable number of parallel ground lines. Note that to set the railroad's starting address you must set both SW1s, the CBAC's as well as the UBEC's.

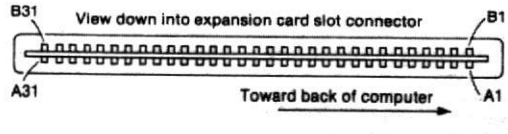
Note that pin A11 connects to program jumper pad P7 on the UBEC *via* pad CL3A. With the PC, devices other than the CPU can control the address and control lines. When the CPU is in control, the PC pulls its AEN (*A*ddress *EN*able) line low. Pin A11 is that AEN line, and, to keep other devices from inadvertently writing to the C/MRI, it is jumped on the UBEC to be gated with MEMW* and MEMR*.

OTHER COMPUTERS

If you have a computer that I haven't covered, please don't give up on the C/MRI. Study the reference manual for your computer and compare its bus description to examples I've covered. You should find enough similarities to be able to tailor the UBEC for your computer, and connect your machine to the C/MRI.

The main thing to consider in any adaptation is that the computer's data bus, address bus, and control lines must pass through your UBEC to come out as the logic signals shown at the right of fig. 2 in C/MRI Part 2, the UBEC schematic. Whatever the logic state of these signals coming out of the computer, they must emerge from the UBEC with data lines D0-D7, and address lines A0-A7, plus AH, noninverted. The RD* and WR* lines, on the other hand, must be inverted. This is essential so that your UBEC properly interfaces with the I/O Cards through the Mother Board.

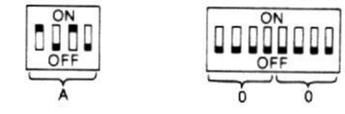

HARDWARE SETUP
IBM PC computer with cover removed to insert CBAC and ribbon cable
50-pin header; Jameco 923866R installed on A side of CBAC
A side faces this way
Front of computer
50-conductor ribbon cable with female socket connectors on both ends; Digi-Key R326-18-ND for 18"-long cable
UBEC
UNIVERSAL BUS EXTENDER CARD
Socket S11
Feed ribbon cable through slot in back of computer. Insert CBAC into available card slot



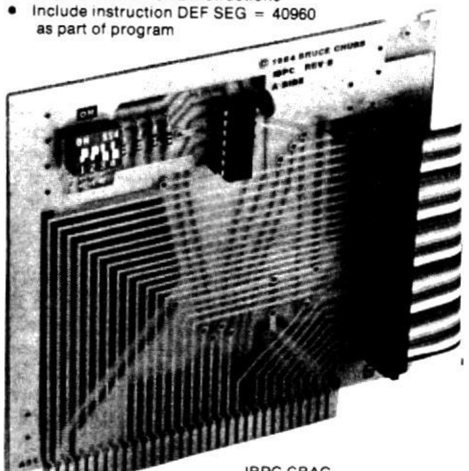
PINOUTS (MEMORY-MAPPED I/O)

PIN NO.	SIGNAL	PIN NO.	SIGNAL	PIN NO.	SIGNAL
A2	D7	A15	A16	A27	A4
A3	D6	A16	A15	A28	A3
A4	D5	A17	A14	A29	A2
A5	D4	A18	A13	A30	A1
A6	D3	A19	A12	A31	A0
A7	D2	A20	A11	B1	GND
A8	D1	A21	A10	B3	+5VDC
A9	D0	A22	A9	B10	GND
A11	AEN	A23	A8	B11	-MEMW
A12	A19	A24	A7	B12	-MEMR
A13	A18	A25	A6	B29	+5VDC
A14	A17	A26	A5	B31	GND

SOFTWARE SETUP
- Use memory-mapped I/O
- For railroad starting address $A0000 = 655360 set CBAC DIP switch to: Set UBEC DIP switch to:



- Use PEEK and POKE instructions
- Include instruction DEF SEG = 40960 as part of program



IBPC CBAC
(A side)

PINOUT NOTES:
IBM uses minus sign to designate active low. Thus -MEMW = MEMW*
MEMW* connects to CL1 on UBEC
MEMR* connects to CL2 on UBEC
AEN connects to P7 via CL3A on UBEC
AH output from CBAC connects to AH input on UBEC via jumpers J50 in alternate position and J51

Fig. 15 IBM PC

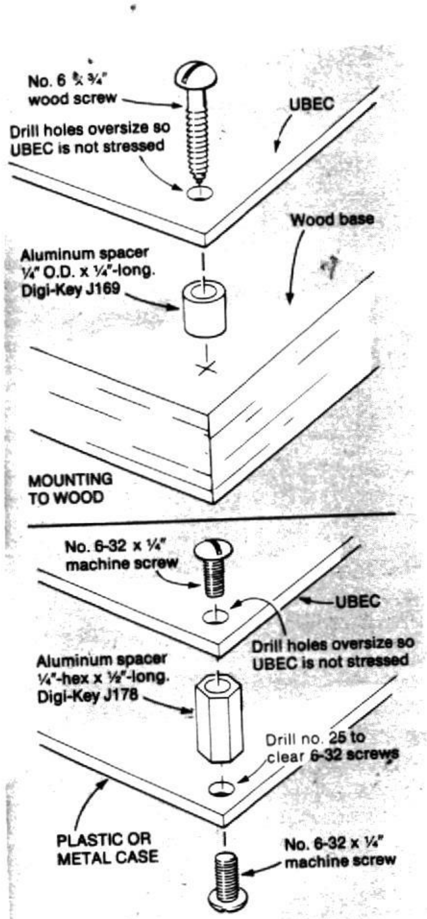

No. 6 x ¾" wood screw
UBEC
Drill holes oversize so UBEC is not stressed
Wood base
Aluminum spacer ¼" O.D. x ¼"-long. Digi-Key J169
MOUNTING TO WOOD
No. 6-32 x ¼" machine screw
UBEC
Drill holes oversize so UBEC is not stressed
Aluminum spacer ¼"-hex x ½"-long. Digi-Key J178
Drill no. 25 to clear 6-32 screws
PLASTIC OR METAL CASE
No. 6-32 x ¼" machine screw

Fig. 16 MOUNTING THE UBEC

UBEC ENCLOSURES

I suggest that you simply mount your UBEC on a small piece of plywood until you've completed testing and debugging. Having it secured on a block of wood helps avoid accidental shorts against the underside of the card. Drill out the four corner holes with a no. 25 bit or larger to clear 6-32 screws, and mount the card with standoffs as shown in fig. 16.

Once you get everything working, it's a good idea to mount your UBEC in some type of enclosure or box. If you choose a metal enclosure, make sure that none of the circuit traces or components touch the enclosure. Again, using standoffs as in fig. 16 will protect the underside of the card.

The UBEC should be close enough to your computer to be joined by no more than 18" of ribbon cable, so the enclosure helps keep things neat around your machine. On the other hand, the Mother Board with its I/O Cards can be mounted further away under your layout, so you won't need enclosures for them. On the Sunset Valley I have everything except the UBEC mounted on a sheet of plywood secured to the basement wall under the layout. A 6-foot cable runs from the UBEC to the Mother Board, so I can move my computer on a castered table.

Next time we'll move on to the railroad side of the C/MRI. I'll show you how to build the I/O Mother Board, a husky 5VDC power supply, and the General Purpose I/O Cards. See you then. ♁

HOMA-BED® **Roadbed from HOMASOTE**

TWO PIECES

HO Roadbed Dimensions: ¼" T. x 2" W. x 32" L.

- For All Type Track—Curve To 15" Radius—NMRA certified
- 48' HO $17.50 ($15.00 + $2.50 UPS) MONEY BACK
- 72' HO $25.00 ($22.00 + $3.00 UPS) GUARANTEE
- 32" UPS sample/details • HO $2.50 • S and O $3.00
- DEALERS WANTED! 8' pack in hobby shops

BO Mfg., RD3, Box 357B-M, Kingston, NY 12401

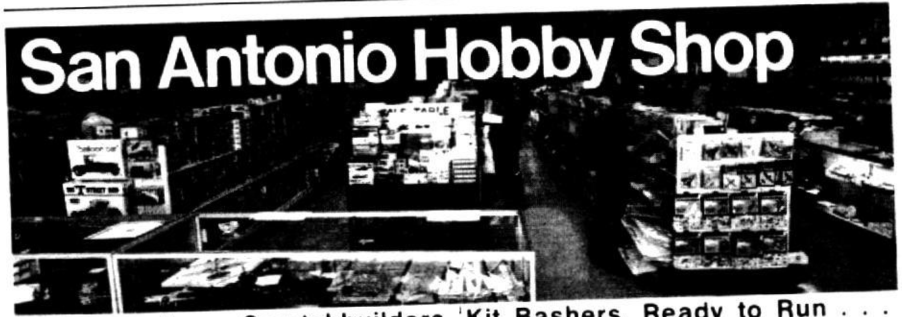
N, Z SCALERS

Great prices, fast service on the complete line of KADEE rolling stock, trucks, and coupler conversions. Send a long SSAE for our latest list.

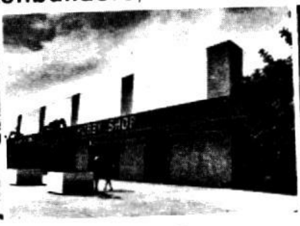
TIMBERLINE ASSOCIATES

P O BOX 423 Lombard, Ill 60148

San Antonio Hobby Shop



For Brass Fans, Scratchbuilders, Kit Bashers, Ready to Run . . .



All the great names at San Antonio Hobby Shop
Alco • NWSL • Hallmark • PFM • Lambert • Key • Precision Scale • Kemtron • Cary • Cal-Scale • Athearn • Tyco • Atlas • Roundhouse • Campbell • Chooch • Grandt Line • Magnuson • E&B Valley • Kadee • Detail Assoc. • Details West

ORDER BY PHONE: use your BankAmericard or Master Charge. **(415) 941-1278** for same day service. Or visit us in the San Antonio Shopping Center, next to Mervyns. Just 30 minutes south of San Francisco, off U.S. 101, on San Antonio Road.

Books, Magazines, Decals, Paints, Tools, Air Brushes, Sound Systems

San Antonio Hobby Shop
2550 West El Camino
Mountain View, CA 94040

Vollmer promotes higher education.

There's the first bell! Grab your lunchbox and an apple for the teacher!

This city schoolhouse was converted from the Vollmer Post Office kit by adding figures and scenic details.

Whether you're learning the basics or have graduated to other modeling materials, you'll enjoy building Vollmer kits.

Simple instructions, plus styrene parts that fit perfectly, make construction easy and fun! All parts are molded in color so no painting is needed.

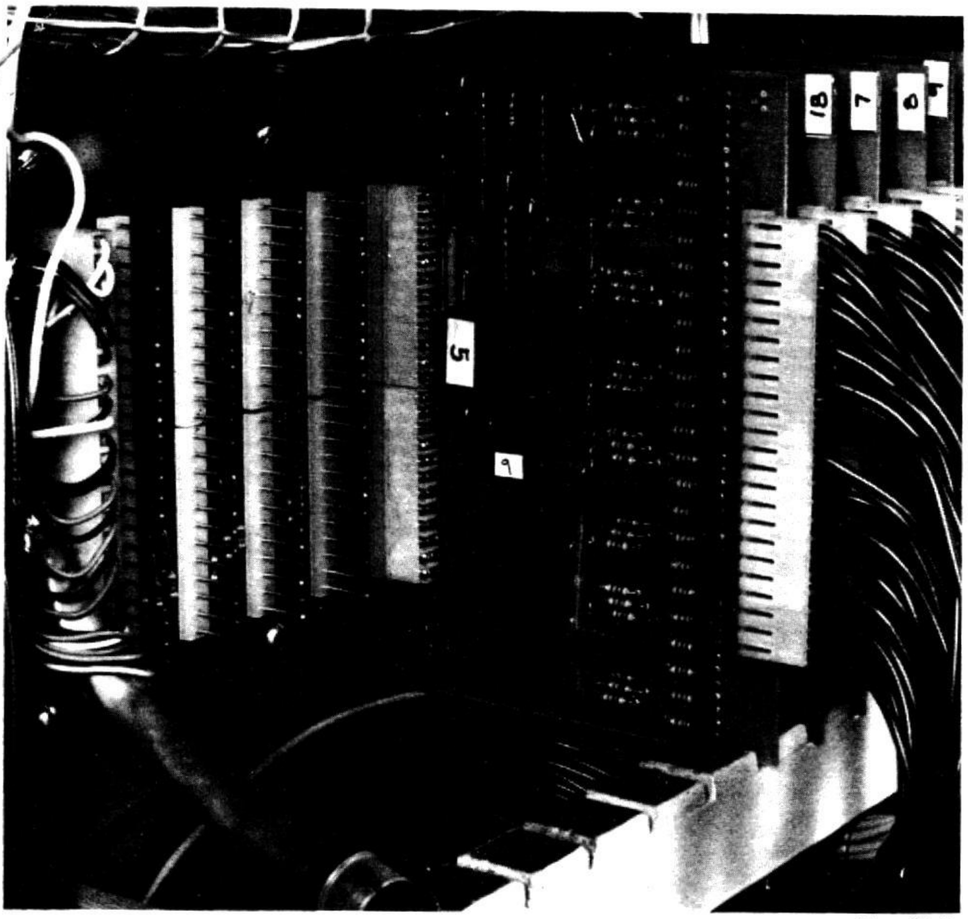
You can learn more about Vollmer structures in **The World of HO Scale -1985**
913-626 $11.95.

Your Dealer can get it from Walthers.

770-3765 Post Office $43.25
American flag not included.

© 1985 Wm. K. Walthers, Inc.

VOLLMER
Imported from West Germany and marketed by Walthers.
5601 W. Florist Ave.
Milwaukee, WI 53218



Bruce Chubb

This is the railroad end of the C/MRI, underneath author Bruce Chubb's Sunset Valley RR. At left you see the I/O Mother Board mounted against the wall, with I/O Cards plugged into the Mother Board at right. The large cable at the bottom is the Mother Board's connection to the UBEC and computer — in his own original installation Bruce did not use the ribbon cables later adopted for the C/MRI series. At right are 24-wire connectors bringing wires from the railroad to the I/O Cards. The slotted wooden rail keeps the I/O Cards lined up with their Mother Board connectors. The nearest Output Card is an early version using a hard-wired DIP header to set the card address instead of the six-segment DIP switch shown in this article.

C/MRI-4

The C/MRI: A computer/model railroad interface

Part 4: Building the I/O circuits

BY BRUCE CHUBB

THIS TIME we'll get to the railroad end of the C/MRI. I'll cover the I/O Mother Board and the general-purpose I/O Cards, and tell you how to build a 5VDC power supply for the I/O Cards.

Full-size circuit art for the Mother Board and I/O Cards, for etching your own circuit cards, is available free by sending a stamped, self-addressed, business-size envelope to MODEL RAILROADER, C/MRI PC Layouts, 1027 N. Seventh St., Milwaukee, WI 53233. Ready-to-use cards are available from JLC Enterprises, P. O. Box 88187, Grand Rapids, MI 49508-9187. Mother Board Card IOMB is $38, Input Card CIN24 is $22, Output Card COUT24 is $24, and there is a shipping and handling charge of $2.50 per order (Michigan residents must add 4 percent sales tax).

Let's start by building the Mother Board.

I/O MOTHER BOARD

The I/O Mother Board is simply a set of 13 "slots" or connectors to let you plug up to 13 I/O Cards into the extended computer bus. The photo at the top of this page shows a Mother Board with I/O Cards plugged into its multiple connectors. Any railroad device that connects with the computer is connected to one or more of the pins on the I/O Cards.

Figure 1 includes the parts list and shows the parts layout for the Mother Board. To fill the card to capacity you need 26 of the Molex connectors, 2 for each card slot. These can hold up to 13 I/O Cards, for a total of 312 discrete wires to the railroad — 13 cards x 24 wires per card. This should be ample for most layouts.

If you do need more you can connect two or more Mother Boards together — I use two on my Sunset Valley RR. You simply jump each of the 24 bus traces from one board to the next. Only the first board in the series needs the 2.2K resistors. These perform as line-terminator pull-up resistors and are good design practice if your UBEC-Mother Board cable is more than a couple feet long.

Here are the assembly steps for the Mother Board:

☐ **Card test.** Use your VOM to check that there are no open-circuit traces and no shorts between adjacent traces or pads.

☐ **J1, J2.** Form straight, taut jumpers from no. 24 solid, bare bus wire.

☐ **R1-R19.** Install as shown in fig. 1. For R13-R19, the resistor leads themselves are too short to reach bus lines 3 through 9. Extend them with bus wire as shown, but slip spaghetti insulation or heat-shrink tubing over the joints to keep adjacent splices from touching. For two or more Mother Boards, install R1-R19 only on the first.

☐ **S1.** Install the 40-post stud connector for the UBEC cable.

☐ **S2-S27.** Install two Molex 12-post stud connectors for each I/O Card slot, pushing them tight against the card as you solder.

☐ **Power terminal screws.** Insert 4-40 screws in the +5VDC and GND connection holes from the top or component side of the card. Add 4-40 brass nuts on the trace side, tighten firmly, and solder the nuts to the circuit traces.

☐ **Labels.** Cut out the "+5VDC" and "GND" labels from fig. 1, or from a photocopy, and glue them next to the power terminals as shown.

☐ **Adjacent trace test.** Set your VOM on scale R x 100 and check the resistance between each adjacent pair of pins on any one set of the card slot connectors. Between 1 and 2 you should read 0Ω (zero ohms) since these are both ground. Between all the other adjacent pairs — 2 and 3, 3 and 4, through 23 and 24 — you should read either infinite resistance for an open circuit, close to 4.4KΩ for two 2.2K resistors in series, or 2.2KΩ between lines 23 and 24. A reading close to 0Ω for any of these indicates a solder bridge between adjacent pads somewhere along the two bus lines under test. Locate it by inspection, and reheat the pads to draw the solder away from the gap. Then retest to make sure you have solved the problem.

☐ **Ground test.** Keep your VOM on the R x 100 scale, but this time connect one lead to the ground screw (GND). Touch the other lead to each pin in one pair of card slot connectors one at a time. Pins 1, 2, and 19 are grounds and should read 0Ω, but all the rest should be open circuits. If you get different readings, again look for the offending solder bridge and remove it. The most likely spot for a solder bridge is among the pads for S1.

☐ **Cleanup.** Clean and seal the trace side of the card, making sure not to get sealer spray onto any connector contact surfaces.

This completes the assembly and standalone testing of the I/O Mother Board. Set it aside for now. Before getting into the construction of the I/O Cards, I'll explain in more detail how the computer addresses a particular railroad input or output through the C/MRI. While an understanding of this could be helpful in later debugging, you may skip ahead to "OUTPUT CARD CONSTRUCTION" on page 95 if you prefer.

ADDRESS DECODING

Figure 2 shows how the 16-bit railroad address is decoded in a three-level process. Address bits A8 through A15 are decoded by the UBEC as we saw last month in C/MRI Part 3. Remember, all eight of these must match the setting of the UBEC's DIP switch — the railroad's starting address — for signal AH, Address *H*igh-order-bits, to go to logic 1 or high. This is the first level of address decoding.

Address bits A2 through A7 pass directly through the UBEC to be decoded by the I/O Cards, each of which includes a six-segment DIP switch to be set for the individual card's address as I'll explain later. If lines A2-A7 match the setting of the DIP switch, signal AM, Address *M*iddle-order-bits, goes high. This is the second level of address decoding.

Signals AH and AM are combined in a NAND gate. The output of such a gate is logic 0 *only* if both inputs are logic 1. The output of the NAND gate is the CS* or Chip Select (active low) input on the 8255 I/O chip. Thus the chip on the addressed I/O card is turned on (CS* = 0) only when address lines A2-A15 correspond to the DIP switch settings of both the UBEC *and* the I/O Card. This enables the third level of address decoding, which is performed by the 8255.

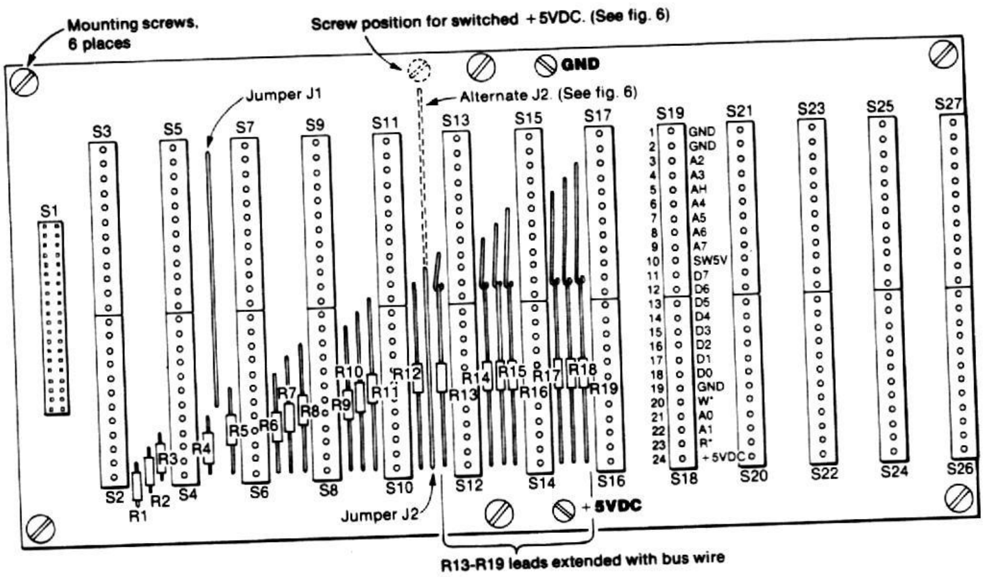
THE I/O CHIP

The heart of each I/O Card is its 8255 IC, a large, 40-pin DIP that goes by the complex-sounding name of "Programmable Peripheral Interface." This chip is a general-purpose, programmable I/O device designed as an external interface for microprocessors. It gives the lowest cost per bit of any parallel-I/O circuit on the market, and it helps to make our C/MRI affordable and easily expandable.

The 8255 can operate in many different modes, but its most basic, mode 0, meets the needs of the C/MRI. Figure 3 summarizes our use of the 8255. Imagine that it's a high-speed, 8-pole, 4-position switch, switching the 8 data lines from the UBEC to Port A, B, or C, or to an open-circuit state with no port connected.

The four control-line inputs, RD*, WR*, RESET, and CS*, and the two address lines A0 and A1, determine which port is selected, if any, and the direction of data flow. The port switch is open-circuited any time that RESET is high or CS* is high. Which port is connected at a given instant depends on the address lines A0 and A1, and at the same time the status of RD* and WR* determine if the data transfer is a read or write operation.

In this most basic mode we can control the 8255 with just the two control bytes shown in fig. 3. You'll understand their



I/O MOTHER BOARD PARTS LIST (In order of assembly)

Qnty.	Symbol	Description
2	J1, J2	Jumpers, make from no. 24 uninsulated bus wire (Belden no. 8022)
19	R1-R19	2200Ω resistors [red-red-red]
1	S1	40-pin stud connector (Jameco 923865R)
26	S2-S27	12-pin connector (Molex 09-64-1121)
2	—	4-40 brass nuts
2	—	4-40 pan-head machine screws

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

LABELS

GND GND

+5VDC +5VD

Fig. 1 INPUT/OUTPUT MOTHER BOARD PARTS LAYOUT

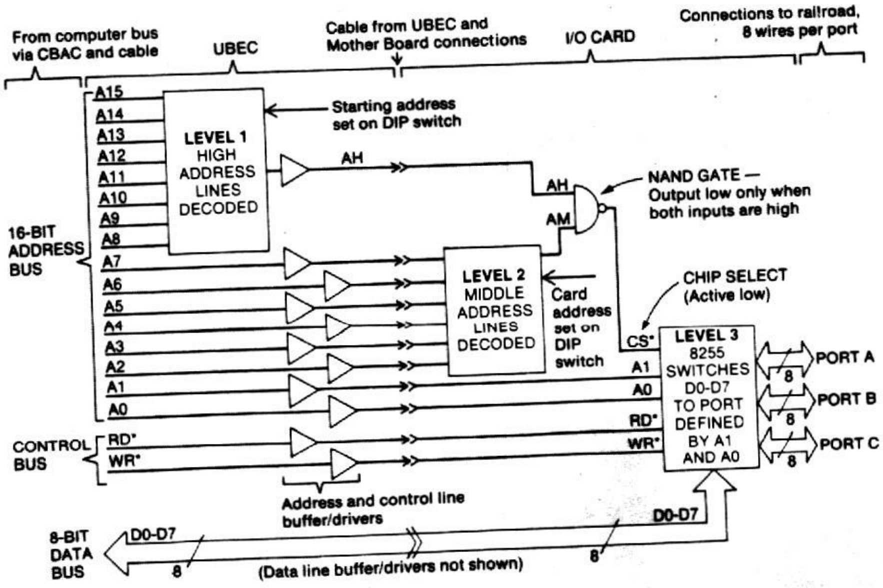
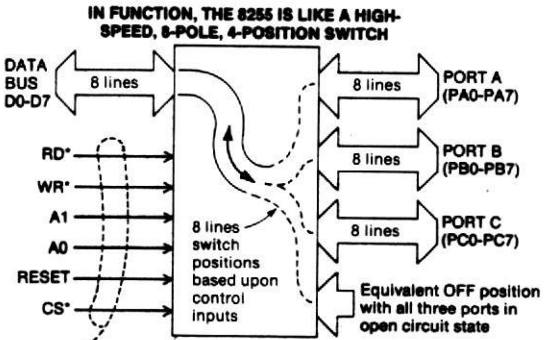

From computer bus via CBAC and cable
UBEC
Cable from UBEC and Mother Board connections
I/O CARD
Connections to railroad, 8 wires per port
16-BIT ADDRESS BUS
A15
A14
A13
A12
A11
A10
A9
A8
A7
A6
A5
A4
A3
A2
A1
A0
LEVEL 1 HIGH ADDRESS LINES DECODED
Starting address set on DIP switch
AH
AM
NAND GATE — Output low only when both inputs are high
LEVEL 2 MIDDLE ADDRESS LINES DECODED
Card address set on DIP switch
CHIP SELECT (Active low)
CS*
RD*
WR*
LEVEL 3 8255 SWITCHES D0-D7 TO PORT DEFINED BY A1 AND A0
PORT A
PORT B
PORT C
8
CONTROL BUS
Address and control line buffer/drivers
8-BIT DATA BUS
D0-D7
(Data line buffer/drivers not shown)

Fig. 2 ADDRESS DECODING; A 3-LEVEL PROCESS

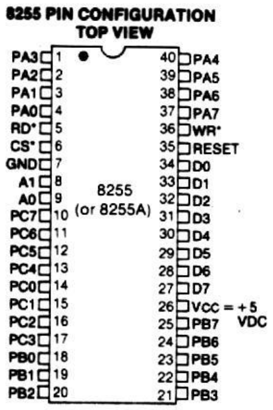

IN FUNCTION, THE 8255 IS LIKE A HIGH-SPEED, 8-POLE, 4-POSITION SWITCH
DATA BUS D0-D7
8 lines
PORT A (PA0-PA7)
PORT B (PB0-PB7)
PORT C (PC0-PC7)
RD*
WR*
A1
A0
RESET
CS*
8 lines switch positions based upon control inputs
Equivalent OFF position with all three ports in open circuit state

These control inputs determine which port is selected, if any, and the direction of data flow

Switch is in OFF (i. e., all 3 ports = open circuit state) if RESET = 1 or chip select (CS*) = 1. Data transfer can take place any time that RESET = 0 and CS* = 0, according to the table below:

					DATA TRANSFERRED	
	A1	A0	RD*	WR*	FROM	TO
INPUT	0	0	0	1	Port A	Data bus
	0	1	0	1	Port B	Data bus
	1	0	0	1	Port C	Data bus
OUTPUT	**0**	**0**	1	0	**Data bus**	**Port A**
	0	**1**	1	0	**Data bus**	**Port B**
	1	**0**	1	0	**Data bus**	**Port C**
	1	**1**	1	0	**Data bus**	**Control byte**

Control bytes	Binary	Hex	Decimal
All ports output	10000000	$80	128
All ports input	10011011	$9B	155



PIN NAMES

D0-D7	Data bus (bidirectional)
RESET	Reset input
CS*	Chip select (active low)
RD*	Read input (active low)
WR*	Write input (active low)
A0-A1	Port address
PA0-PA7	Port A (bit)
PB0-PB7	Port B (bit)
PC0-PC7	Port C (bit)
Vcc	+5VDC
GND	0V

Fig. 3 BASIC FEATURES OF THE 8255 PROVIDE FOR EASY I/O

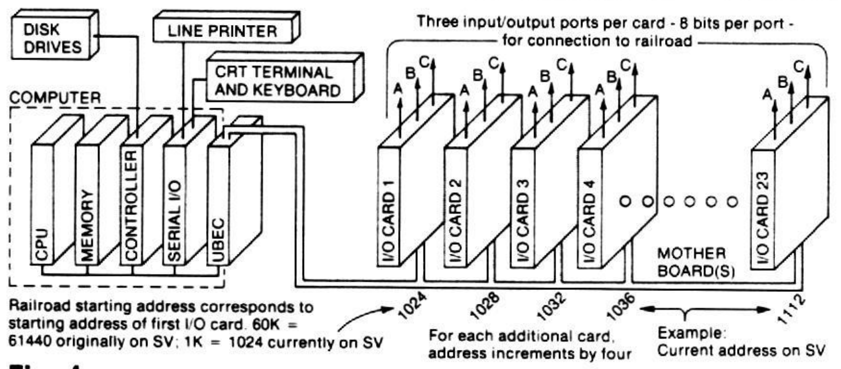

DISK DRIVES
LINE PRINTER
CRT TERMINAL AND KEYBOARD
COMPUTER
CPU
MEMORY
CONTROLLER
SERIAL I/O
UBEC
Three input/output ports per card - 8 bits per port - for connection to railroad
A B C
I/O CARD 1
I/O CARD 2
I/O CARD 3
I/O CARD 4
I/O CARD 23
MOTHER BOARD(S)
Railroad starting address corresponds to starting address of first I/O card. 60K = 61440 originally on SV; 1K = 1024 currently on SV
1024
1028
1032
1036
1112
For each additional card, address increments by four
Example: Current address on SV

Fig. 4 I/O CARD ADDRESS INCREMENTS

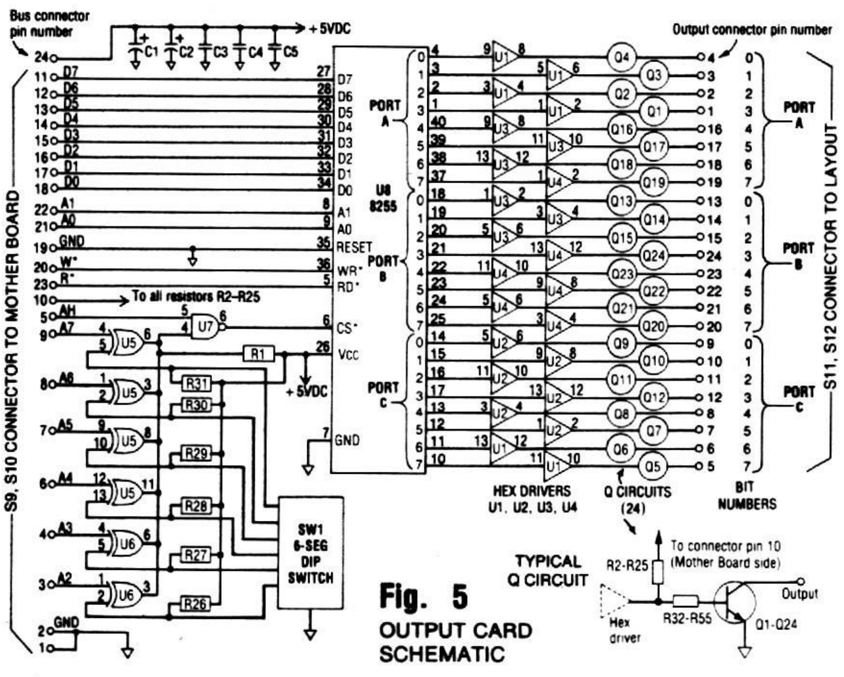


Fig. 5 OUTPUT CARD SCHEMATIC

meanings as we get into the C/MRI software. For now all you need is this general idea of what the 8255 does for us.

MESSAGE EXAMPLE

Let's look at how the computer sends a message to, say, address 1K (1024), Port A of the first I/O Card, and on to the railroad. Once you understand this you'll have a working knowledge of how the whole C/MRI operates.

To transmit a message to the railroad via Port A, the computer places 8 bits of data, the message, on the 8 data lines. It simultaneously places the desired address, 1K, on the 16 address bus lines. Every device connected to the address bus tries to decode the 1K to see if it is the location addressed, but only our I/O Card at 1K gets the proper decoding to make it receptive to the data lines.

In less than a microsecond (one millionth of a second) the lines will have stabilized and the address will have been decoded, and then the computer triggers the WR* (*WR*ite, active low) line for another fraction of a microsecond. This causes the 8255 to read the data lines and switch them into the 8 lines of Port A, so the data is available to turn devices on the railroad on or off as desired.

Our first I/O Card responds any time the computer addresses 1024, 1025, 1026, or 1027. This is because for these four addresses (or any similar sequence), lines A2 through A15 are identical and only A0 and A1 become binary 00, 01, 10, and 11 respectively. A0 and A1 pass directly to the 8255, where they are decoded as shown in the colored portion of the data transfer table in fig. 3.

Thus we'd program the computer to transmit address 1024 when we want Card 1 Port A, 1025 for Card 1 Port B, 1026 for Card 1 Port C, and 1027 when we want to transmit an 8255 control byte to Card 1. (We'll need a programmed control byte transmission to "initialize" the 8255, set it for startup, whenever we power up the railroad and computer.)

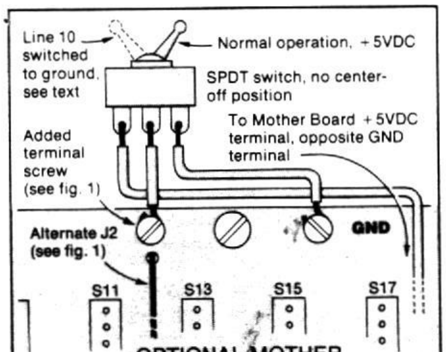
To address the next card we simply add four to the previous address, and for the next add four again, as shown in fig. 4.

OUTPUT CARD SCHEMATIC

Figure 5 is the schematic for the general-purpose Output Card. The Input Card, which I'll cover later, is nearly identical. Every Output Card is interchangeable with every other Output Card, and the same is true for the Input Cards. DIP switch SW-1 is used to set each card's "slot address" as described above. As we've seen, the 8255 chip does most of the I/O work for us, but a few other parts are needed too.

From the Mother Board, data lines D0-D7, address lines A0-A1, the grounded RESET line, and the two UBEC control lines W* and R* are connected directly to the 8255. Address lines A2 through A7 are decoded by two 74LS136 ICs, U5 and U6, just as the higher address lines were decoded on the UBEC. However, for the I/O Cards only 6 bits need to be decoded, so a 6-segment DIP switch is all we need to set the card address.

The address decode circuit — U5, U6, U7, R1, R26-R31, and SW1 — is de-


Line 10 switched to ground, see text
Normal operation, +5VDC
SPDT switch, no center-off position
To Mother Board +5VDC terminal, opposite GND terminal
Added terminal screw (see fig. 1)
Alternate J2 (see fig. 1)
GND
S11
S13
S15
S17
Fig. 6 OPTIONAL MOTHER BOARD GROUND SWITCH

signed so that pin 6 of U7 is high except when the CPU is addressing this specific card, that is, when AH is high and lines A2-A7 correspond to SW-1. When this is the case, pin 6 of U7 goes low, sending a low signal (logic 0) to the CS* input of the 8255, pin 6 of U8, and enabling data transfer across the I/O chip.

For a write operation, the addressed port — either A, B, or C — is latched. This means that the signals on data lines D0-D7 when CS* is low are transferred to the appropriate port and remain fixed even after CS* goes high. The output remains latched until either CS* goes low again or a new control byte is transmitted to the 8255. This latching is important, since you don't want the outputs — signal indications, say, or turnout settings — to disappear while the computer is doing calculations or communicating with other devices. The outputs stay constant until the CPU sends new data.

Each port output line from the 8255 is connected through a 7407 hex noninverting buffer (U1, U2, U3, or U4) to a 2N3904 NPN output transistor. The open-collector transistor provides drive capability for most model railroad loads, including LEDs, GOW lamps, and relays, as well as other logic circuits.

The 7407 is an open-collector device itself, and the top ends of the 1K pull-up resistors (R2-R25) are all connected to pin 10 of the bus connector to the Mother Board. Bus line 10 on the Mother Board is usually wired *via* jumper J2 so that it is held at +5VDC.

As an option, J2 can be relocated to the alternate position connecting line 10 to the extra screw-pad terminal, and the optional ground switch installed. See fig. 6. That way line 10 can be switched between +5VDC, for normal operation, and ground, to force all Output Card outputs to the open-circuit state.

This is handy to keep the 8255 from delivering random outputs when you have the railroad turned on but have not activated the computer program. Otherwise, this circumstance can cause unwanted conditions on the railroad. On the Sunset Valley, for example, the annunciator horn on the dispatcher's CTC panel might sound continuously.

BUILDING THE OUTPUT CARD

Figure 7 is the parts layout and list for the Output Card. All parts mount on the top side of the board, the plain surface, and are soldered on the bottom or circuit-trace side. I've indicated steps involving polarity- and orientation-sensitive parts with a plus sign in brackets: [+]. Refer to fig. 4, C/MRI Part 2, March MODEL RAILROADER, for help with orienting parts.

Note that U8, the 8255, is static-sensitive. You won't be installing this part until C/MRI Part 5, next month, but you ought to know that 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 which could damage the chip.

You'll need one Output Card for every 24 lines of output to your railroad. Here's how to assemble one:

☐ **Card test.** Use your VOM to check that there are no open-circuit traces and no shorts between adjacent traces or pads.

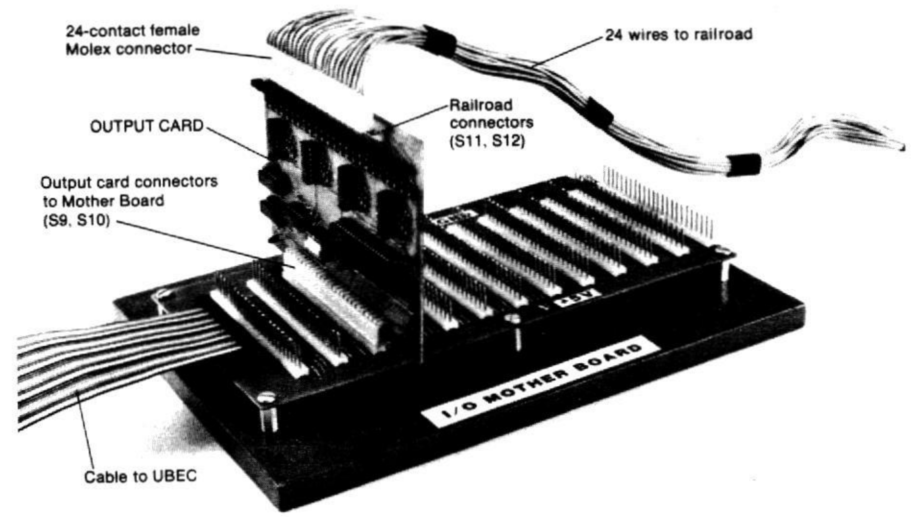
☐ **J1-J23.** Form straight, taut jumpers from no. 24 solid, bare bus wire.

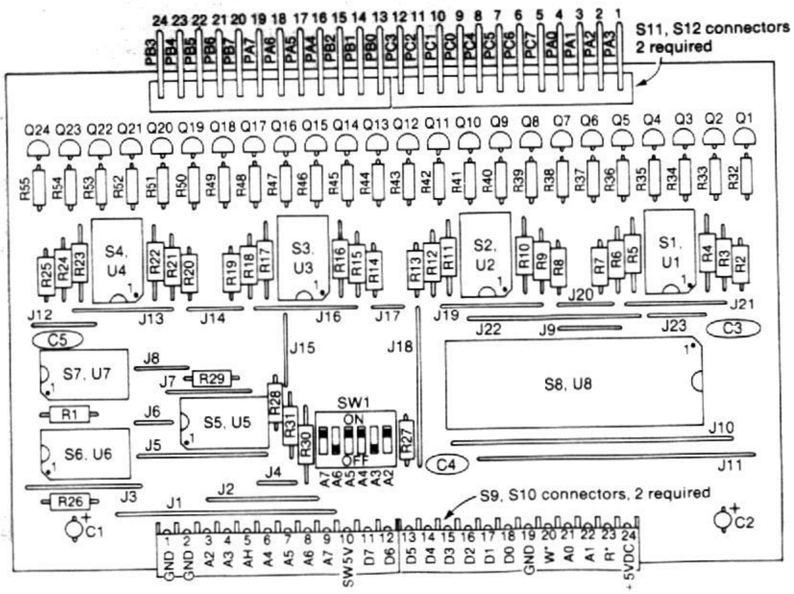
☐ **R1-R55.** Install as shown in fig. 7. Use the color codes in the parts list to select the correct resistor for each location.

☐ **S1-S8 [+].** Install the IC sockets with pin-1 orientation as shown in fig. 7. DO NOT install ICs U1-U8 in the sockets at this time.

☐ **SW1 [+].** Use your VOM to make sure you install this DIP switch so its contacts are closed when thrown toward S3. This direction will be "ON," and away from S3 will be "OFF."

☐ **S9, S10.** Install the Molex 12-contact


24-contact female Molex connector
24 wires to railroad
OUTPUT CARD
Railroad connectors (S11, S12)
Output card connectors to Mother Board (S9, S10)
I/O MOTHER BOARD
Cable to UBEC


S11, S12 connectors 2 required
PB3 PB4 PB5 PB6 PB7 PA7 PA6 PA5 PA4 PB2 PB1 PB0 PC3 PC2 PC1 PC0 PC4 PC5 PC6 PC7 PA0 PA1 PA2 PA3
Q24 Q23 Q22 Q21 Q20 Q19 Q18 Q17 Q16 Q15 Q14 Q13 Q12 Q11 Q10 Q9 Q8 Q7 Q6 Q5 Q4 Q3 Q2 Q1
S4, U4
S3, U3
S2, U2
S1, U1
S7, U7
S5, U5
S6, U6
S8, U8
SW1
ON
OFF
S9, S10 connectors, 2 required
GND GND A2 A3 AH A4 A5 A6 A7 SW5V D7 D6 D5 D4 D3 D2 D1 D0 GND W* A0 A1 R* +5VDC

OUTPUT CARD PARTS LIST (In order of assembly)

Qnty.	Symbol	Description
23	J1-J23	Jumpers, make from no. 24 uninsulated bus wire (Belden no. 8022)
25	R1-R25	1000Ω resistors [brown-black-red]
24	R26-R31	2200Ω resistors [red-red-red]
24	R32-R55	120Ω resistors [brown-red-brown]
7	S1-S7	14-pin DIP socket (Digi-Key C8914)
1	S8	40-pin DIP socket (Digi-Key C8940)
1	SW1	6-segment DIP switch (Digi-Key SW1016-ND)
2	S9, S10	12-pin connectors (Molex 09-52-3121)
2	S11, S12	12-pin connectors (Molex 09-66-1121)
24	Q1-Q24	2N3904 NPN small signal transistors (Digi-Key)
2	C1, C2	2.2μF, 35V tantalum capacitors (Jameco TM2.2/35)
3	C3-C5	.1μF, 50V ceramic disk capacitors (Jameco DC.1/50)
4	U1-U4	7407 hex buffer/drivers (Jameco)
2	U5, U6	74LS136 quad two-input exclusive-OR gate (Jameco)
1	U7	74LS00 quad two-input NAND gate (Jameco)
1	U8	8255 or 8255A programmable peripheral interface (Jameco)
	Mating connector for railroad cable	
1	—	24-pin connector shell (Molex 09-50-3241)
24	—	Model 247 crimp terminals (Molex 08-50-0106)

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

Fig. 7 OUTPUT CARD PARTS LAYOUT

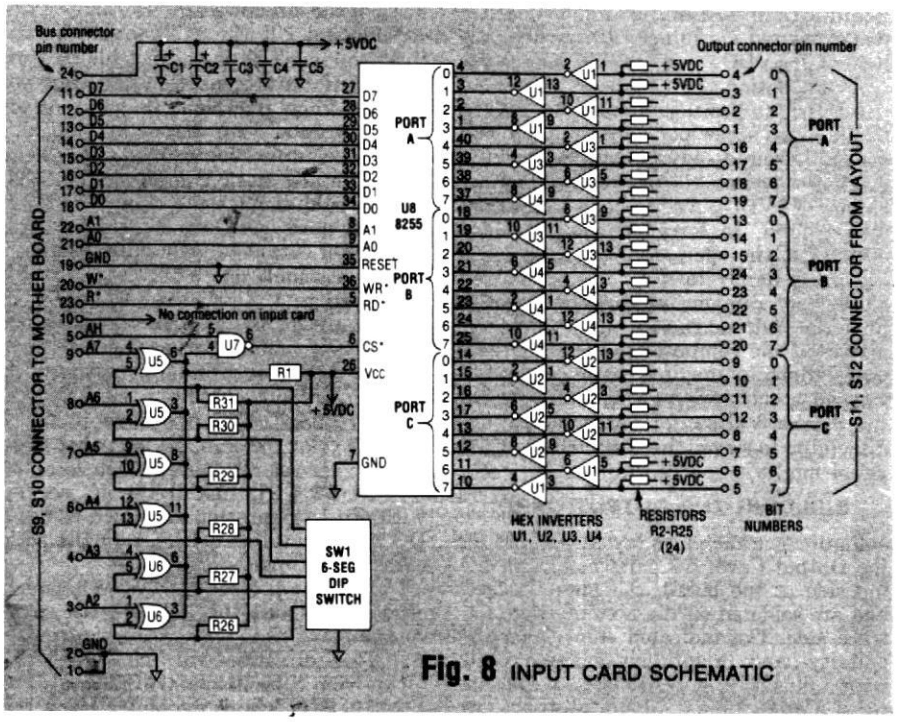
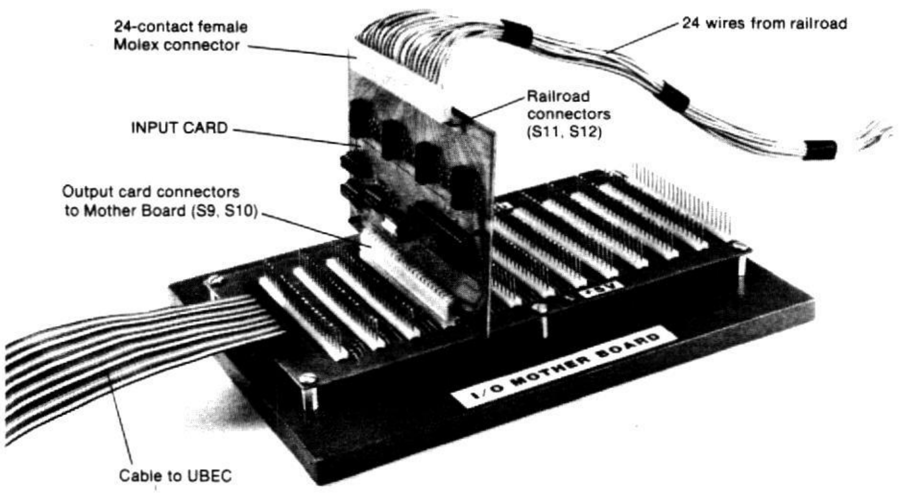

Bus connector pin number
+5VDC
Output connector pin number
PORT A
PORT B
PORT C
U8 8255
RESET
No connection on input card
SW1 6-SEG DIP SWITCH
HEX INVERTERS U1, U2, U3, U4
RESISTORS R2-R25 (24)
BIT NUMBERS
S9, S10 CONNECTOR TO MOTHER BOARD
S11, S12 CONNECTOR FROM LAYOUT

Fig. 8 INPUT CARD SCHEMATIC


24-contact female Molex connector
24 wires from railroad
Railroad connectors (S11, S12)
INPUT CARD
Output card connectors to Mother Board (S9, S10)
I/O MOTHER BOARD
Cable to UBEC

connectors by first hooking their nylon retaining fingers over the card edge, then feeding the metal contact pins through the card holes. Make sure all 12 pins of each connector pass through the holes. Hold the connector shell tightly against the card as you solder.

☐ **S11, S12.** Install the Molex 12-prong right-angle connectors, and hold them tightly against the card as you solder.

☐ **Q1-Q24 [+].** Slightly bend the leads of each transistor to fit its holes, and orient each with its flat side facing the R32-R55 resistors, with its center (base) lead in the hole nearest the resistors. You may have covered some of these holes with solder when installing the resistors. Either use desoldering braid (Radio Shack 64-2090) to remove the excess solder, or redrill the hole from the component side with a no. 72 bit (.025"). For a neat installation with Q1-Q24 all lined up in a row, I put a length of ⅛"-square stripwood under the curved back sides of the transistors, then press each one down until it just rests on the wood. Once all 24 transistors are soldered in place, I remove the wood and trim the leads.

☐ **C1, C2 [+].** Install these capacitors with their plus (+) leads in the plus holes as indicated in fig. 7. Reversed polarity will damage the capacitors.

☐ **C3-C5.** Insert, solder, and trim.

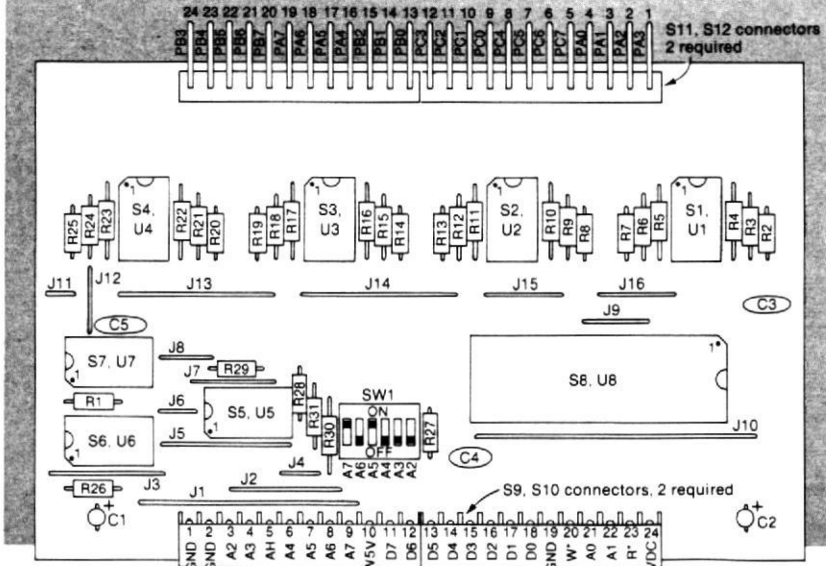
☐ **Inspection.** Look for and correct any soldering faults, then clean and seal the trace side of the board. Be careful not to get any sealer spray on the connector contact surfaces.

That completes the Output Card. I'll discuss the Input Card's schematic next, but if you wish you may skip ahead to "BUILDING THE INPUT CARD."

INPUT CARD SCHEMATIC

Figure 8 is the Input Card schematic. It's like the Output Card except for the area at the right, between the 8255's A, B, and C ports and the railroad connections. The input lines could be wired from the railroad socket directly into the 8255, but I've found it best to include a buffer in each input line with U1-U4.

I use these four 74LS04 hex inverters


S11, S12 connectors 2 required
S4, U4
S3, U3
S2, U2
S1, U1
S7, U7
S5, U5
S6, U6
S8, U8
SW1
ON
OFF
S9, S10 connectors, 2 required

INPUT CARD PARTS LIST (In order of assembly)

Qnty.	Symbol	Description
16	J1-J16	Jumpers, make from no. 24 uninsulated bus wire (Belden no. 8022)
1	R1	1000Ω resistor [brown-black-red]
30	R2-R31	2200Ω resistors [red-red-red]
7	S1-S7	14-pin DIP socket (Digi-Key C8914)
1	S8	40-pin DIP socket (Digi-Key C8940)
1	SW1	6-segment DIP switch (Digi-Key SW1016-ND)
2	S9, S10	12-pin connectors (Molex 09-52-3121)
2	S11, S12	12-pin connectors (Molex 09-66-1121)
2	C1, C2	2.2μF, 35V tantalum capacitors (Jameco TM2.2/35)
3	C3-C5	.1μF, 50V ceramic disk capacitors (Jameco DC.1/50)
4	U1-U4	74LS04 hex inverters (Jameco)
2	U5, U6	74LS136 quad two-input exclusive-OR gate (Jameco)
1	U7	74LS00 quad two-input NAND gate (Jameco)
1	U8	8255 or 8255A programmable peripheral interface (Jameco)
	Mating connector for railroad cable	
1	—	24-pin connector shell (Molex 09-50-3241)
24	—	Model 247 crimp terminals (Molex 08-50-0106)

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

Fig. 9 INPUT CARD PARTS LAYOUT

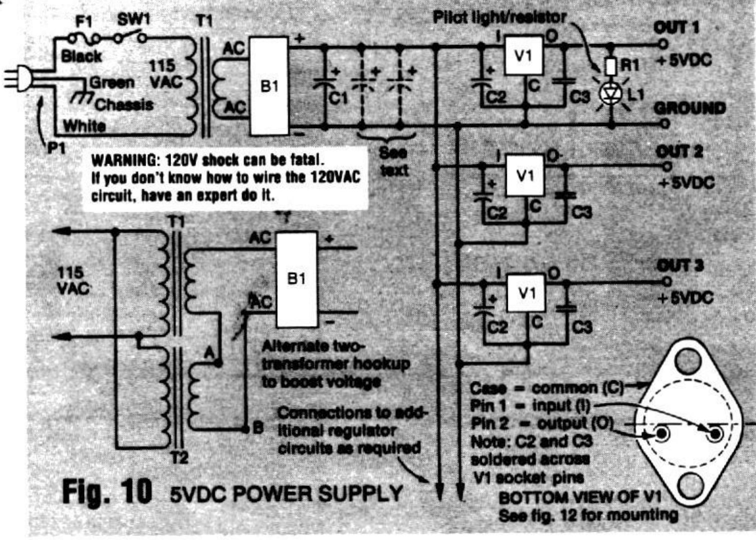

F1
SW1
T1
Black
115 VAC
Green
Chassis
White
P1
AC
B1
AC
+
−
C1
See text
Pilot light/resistor
I
O
V1
C
C2
C3
R1
L1
OUT 1
+5VDC
GROUND
OUT 2
+5VDC
OUT 3
+5VDC
WARNING: 120V shock can be fatal.
If you don't know how to wire the 120VAC circuit, have an expert do it.
115 VAC
T1
T2
A
B
Alternate two-transformer hookup to boost voltage
Connections to additional regulator circuits as required
Case = common (C)
Pin 1 = input (I)
Pin 2 = output (O)
Note: C2 and C3 soldered across V1 socket pins
BOTTOM VIEW OF V1
See fig. 12 for mounting

Fig. 10 5VDC POWER SUPPLY

5-VOLT POWER SUPPLY PARTS LIST (In order of assembly)

Qnty.	Symbol	Description
1	P1	Three-conductor power cord w/plug (Jameco 17236)
1	—	Strain relief (Radio Shack 278-1636)
1	F1	Panel-mount fuse holder (Radio Shack 270-365)
1	—	2A fuse (Radio Shack 270-1275)
1	SW1	Toggle switch (Radio Shack 275-602)
1	T1	Power transformer (See text and sidebar)
1	B1	Bridge rectifier (Varco 30A, 100-PIV VK148), or (Radio Shack 25A, 50-PIV 276-1185), or (Digi-Key 25A, 50PIV BR2505-ND)
as reqd.	C1	Filter capacitor(s) (see text)
as reqd.	V1	5V fixed-voltage regulator (Fairchild 5A μA78H05 or μA78P05 10A capacity) (see text)
as reqd.	—	TO-3 socket (Radio Shack 276-029)
as reqd.	—	TO-3 mounting hardware (Radio Shack 276-1371)
as reqd.	C2	1μF, 35V tantalum capacitor (Jameco TM1/35)
as reqd.	C3	.1μF, 50V ceramic disk capacitor (Jameco DC.1/50)
as reqd.	H1	Fin-type heatsinks for B1 and all V1s with TO-3 case (Jameco DUDE-4 or equivalent)
1	R1	160Ω resistor [brown-blue-brown]
1	L1	Diffused LED (Jameco XC556R)
1		Terminal strip (Radio Shack 274-688)
as reqd.		Grommets (Radio Shack 64-3025)

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

not so much for logic inversion (either kind of input logic can be handled with software) as to protect the 8255. If you couple the 8255 directly to the railroad and accidentally connect an input line to, say, track or switch machine power, you blow the 8255 and about $5. It's better to blow a 35-cent 74LS04. Since installing my C/MRI in January 1981, this precaution has saved me four 8255s.

No latching is required when the 8255 is used as an input device. Once the computer has performed the READ operation, the data read is stored in memory.

BUILDING THE INPUT CARD

You'll need one Input Card for every 24 lines of input from your railroad. Figure 9 gives its parts layout and parts list, and here's how to build one:

☐ **Card test.** Use your VOM to check that there are no open-circuit traces and no shorts between adjacent traces or pads.

☐ **J1-J16.** Form straight, taut jumpers from no. 24 solid, bare bus wire.

☐ **R1-R31.** Install as shown in fig. 9. Use the color codes in the parts list to select the correct resistor for each location.

☐ **S1-S8 [+].** Install the IC sockets with pin-1 orientation as shown in fig. 9. DO NOT install ICs U1-U8 in the sockets at this time.

☐ **SW1 [+].** Use your VOM to make sure you install this DIP switch so its contacts are closed when thrown toward S3. This direction will be "ON," and away from S3 will be "OFF."

☐ **S9, S10.** Install the Molex 12-contact connectors, making sure all 12 pins of each connector pass through the holes.

☐ **S11, S12.** Install the Molex 12-prong right-angle connectors.

☐ **C1, C2 [+].** Install these capacitors with their plus (+) leads in the plus holes as indicated in fig. 9. Reversed polarity will damage the capacitors.

☐ **C3-C5.** Insert, solder, and trim.

☐ **Inspection.** Look for and correct any soldering faults, then clean and seal the trace side of the board. Be careful not to get any sealer spray on the connector contact surfaces.

This completes the assembly of the general-purpose Input Card.

BUILDING A 5VDC POWER SUPPLY

You can purchase 5VDC supplies from electronics suppliers, but you can build your own a lot less expensively. MR has published power supply articles, but for the C/MRI you will need current capacity greater than the typical 1A.

Power requirements vary from railroad to railroad, so it's a good idea to rough out what yours will be. Simply total the items you expect to drive with the 5VDC supply. Each I/O card takes about 250 milliamps, each LED turned on draws about 20mA, and add any other loads you desire. Here's a breakdown of the 5VDC power requirements on my Sunset Valley:

Load	Current
1500 LEDs @ 20mA (assume no more than ⅓ turned on at a time)	= 10A
24 I/O Cards @ 250mA	= 6A
Dispatcher's CTC panel	= 2A
Walnut Hill tower panel	= 2A
All other misc. circuitry	= 4A
TOTAL	24A

Even if you need a lot less, you will still need a hefty supply. Figure 10 shows a schematic that can be tailored to meet your C/MRI requirements. Parts T1, the power transformer, and C1, a large filter capacitor, can be costly, but you may use low-priced surplus parts for these. See the sidebar, "Transformers and capacitors," on page 98.

The transformer output should be at least 6.3V, and 7.5V to 8.5V would be better. Some transformers have taps on the primary for power line variations, i.e., 107V to 125V. If so, wiring your line-cord input to the lowest-voltage primary winding will help raise the output voltage.

Another voltage booster is shown as an option in fig. 10. Connect the secondary of T2, a second transformer with a 1.5V to 2V output, in series with T1's. The current capacities of both T1 and T2 should be greater than you need from your power supply. If the combined voltage decreases instead of increases when you connect the secondaries, you have them out of phase. Simply reverse the secondary leads (marked A and B in fig. 10) from one of the transformers to correct the phasing.

A rule of thumb for the size of filter capacitors like C1 is to allow 5000 to 10,000μF per amp of output. You needn't try to find all the capacitance you need in a single capacitor, however. In fig. 10, the dotted additional capacitors in parallel with C1 show how to add as many of these parts as it takes to get a total value of around 7500μF per amp. Two 52,000μF capacitors, for example, would be good for about 14 amps.

C1's voltage rating isn't important as long as it's 15VDC or greater. When you order capacitors, make sure you also get mounting clamps for easy installation.

Figure 10 shows multiple V1 circuits, too. Each delivers up to 5 amps using a Fairchild μA78H05 regulator as V1, or up to 10 amps with the μA78P05 regulator. On the SV I have six 5A regulators and one 10A regulator; a 25A, 6.6V transformer in series with a 30A, 2V transformer (Fair Radio 5950-829-8618); a Varco 30A bridge rectifier; and four 48,000 μF capacitors.

Each SV regulator powers a separate function on the railroad, so currents stay well below maximum ratings, and a short in one function doesn't affect others. My 10A regulator supplies power to all of the C/MRI I/O Cards (about 6 amps), and my 5A regulators supply trackside signals, the CTC dispatcher's panel, the Walnut Hill tower panel, the Fillmore yard panel, and the overhead track diagram panels.

Figure 11 shows a 5VDC, 4A power supply that's adequate for small layouts. The basic structure is a 7″ x 13″ x 2″ aluminum chassis (Bud no. AC-409) with 1/16″ x 7″ x 13″ aluminum sheet used as the component-mounting surface on top. Power supplies usually end up under the layout, so a complete enclosure isn't necessary, but for safety you should enclose the high-voltage connections to prevent accidental shock.

How you lay out your power supply isn't critical, but use the checklist on the next page so you won't forget to include a necessary item. All parts on the checklist should be mounted before you begin wiring, except V1 and its heatsink.

☐ Fuse holder F1.
☐ Toggle switch SW1.
☐ Transformer T1, and T2 if required.
☐ Bridge rectifier B1, with heatsink if required.
☐ Filter capacitor C1, or two or more C1s in parallel.
☐ Voltage regulator V1, or two or more V1s in parallel, with a separate heatsink for each.
☐ Pilot light L1.
☐ Output screw terminals, including at least one ground terminal and one +5VDC terminal for each V1 regulator.

V1 must be mounted on a heatsink, and for currents of more than 5A, B1 should have a heatsink too. B1's case is insulated and V1's is the common or ground connection, so the heatsinks don't need to be insulated from each other.

HIGH-VOLTAGE WIRING

Install the high-voltage wiring as follows:

Warning: Readers unfamiliar with electrical safety precautions should not make the 115VAC connections. These must be made properly to avoid the possibility of a dangerous, potentially fatal electrical shock.

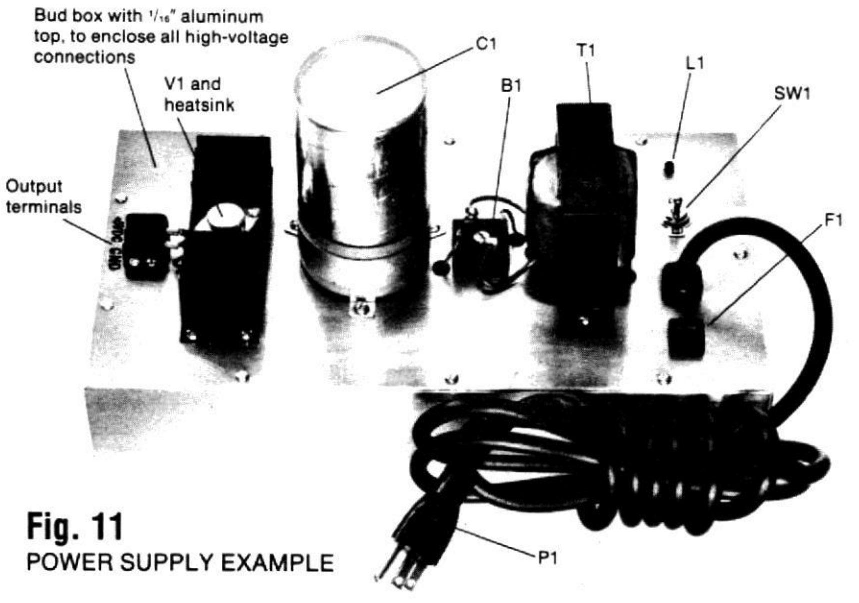
☐ **Power line cord.** Trim about 6" of the outer insulation off the end of the cord. Separate its three wires, then trim about ¾" of insulation from each.

☐ **Green wire (ground).** Solder to a lug mounted under any one of the transformer mounting screws. This grounds both the metal frame of the transformer and the metal chassis, if your house wiring is properly grounded.

☐ **White wire/T1.** Solder the wire to one of the transformer primary lugs. If exposed, insulate this connection with heat-shrink tubing or electrical tape.

☐ **Black wire/F1.** Solder the wire to the end tab of the fuse holder.

☐ **F1/SW1.** Run a piece of no. 20 insulated hookup wire from the side tab of the fuse holder to one tab of the toggle and solder both connections.



Bud box with 1/16" aluminum top, to enclose all high-voltage connections
V1 and heatsink
C1
T1
B1
L1
SW1
Output terminals
F1
P1

Fig. 11
POWER SUPPLY EXAMPLE

☐ **SW1/T1.** Run a second piece of no. 20 hookup wire from the other tab of the toggle to the other transformer primary lug. If the latter connection is exposed, insulate it with heat-shrink tubing or electrical tape.

☐ **Inspection.** Examine all connections to make sure that the high-voltage wiring is as shown in fig. 10 and that any exposed connections are insulated.

☐ **Voltage test.** Set your VOM to measure a voltage of around 10VAC. Place a 2A fuse in the fuse holder, plug in the line cord, turn the toggle on, and measure the voltage at the transformer secondary. It should be just a bit higher than the rated value for your transformer — i.e., 7.6V for a 6.3V transformer without a load. Make a note of your reading: ________VAC. If it isn't between 6.3 and 10.3VAC you either have the transformer(s) wired incorrectly or an unsuitable transformer. Unplug the line cord and correct this before going on.

LOW-VOLTAGE WIRING

Make sure the power line cord is unplugged before proceeding with the low-voltage connections.

☐ **T1/B1 (1 of 2).** Run a length of no. 18 insulated hookup wire from either of the transformer secondary lugs to either of the B1's AC input lugs and solder.

☐ **T1/B1 (2 of 2).** Run a length of no. 18 insulated hookup wire from the other transformer secondary lug to the other B1 AC input lug, and solder.

☐ **B1/C1 (positive).** Run a no. 18 insulated hookup wire from the positive (+) lug of B1 to the positive terminal of C1, or to each C1's positive terminal if you use more than one filter capacitor.

☐ **B1/C1 (negative).** Run a no. 18 insulated hookup wire from the negative (-) lug of B1 to the negative terminal of C1, or to each C1's negative terminal if you

TRANSFORMERS AND CAPACITORS

A good source for surplus power transformers is Fair Radio Sales, P. O. Box 1105, 1016 E. Eureka St., Lima, OH 45802. Here's a list of some suitable transformers in their 1984 catalog:

Part no.	Output: Volts	Amps	Weight (approx.)	Price
4094*	6.3	15	11 lbs.	$20.00
	6.3	15		
FT2550*	6.5	11	16 lbs.	$ 8.95
	6.5	6		
5950-647-6219*	6.5	24	22 lbs.	$14.95
	6.5	6		
5950-829-8618	2.0	30	4 lbs.	$ 2.95

*Dual windings, connect in parallel to increase current capacity (i.e., 4094 with windings in parallel supplies 30 amps).

The last transformer is good for use as T2, in series with T1 to raise output voltage as in fig. 10.

If you don't mind spending a bit more, you can buy name-brand transformers from electronics suppliers. The examples in the next column are made by Stancor:

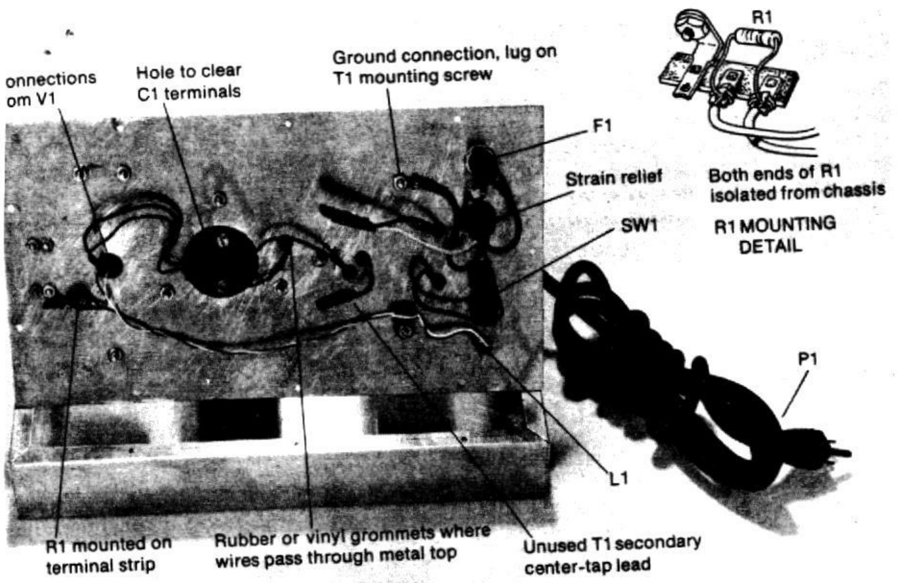
Part no.	Output: Volts	Amps	Weight (approx.)	Price
P-4089	6.3	6	3.5 lbs.	$21.31
P-6464	6.3	10	3.5 lbs.	$26.18
P-8189	6.3	20	6.9 lbs.	$34.29
P-6138	7.5	8	4.7 lbs.	$28.47
P-6457	7.5	21	8.0 lbs.	$47.71

If you're in doubt as to current capacity, choose the larger value. Weight shows how much iron is in the core laminations. For a given current capacity, more iron gives better regulation, for a more constant output voltage under changing loads.

Fair Radio is also a good source for surplus computer-grade capacitors, such as:

40000μF	20VDC	$2.75
52000μF	20VDC	$2.85
76000μF	40VDC	$4.75

New-component prices are $15 to $30 for the same items, so you can see the advantage of shopping surplus for your power transformers and filter capacitors.



ENCLOSED CONNECTIONS

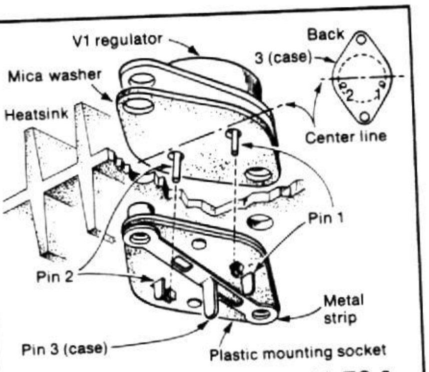


Fig. 12 ISOLATING V1, with TO-3 socket and hardware

are using more than one filter capacitor.

☐ **Voltage test.** Plug in the line cord and set your VOM to measure about 25VDC. Touch the meter's plus (+) and minus (-) leads to the plus and minus terminals of C1, and make a note of your measurement: ________VDC. If it isn't between 7.5 and 13.1VDC, you have made a wiring error, a measurement error, or have a faulty B1 or C1(s).

Warning: Until your power supply is finished and connected to a load, voltage will be stored in the capacitor even after you turn off the SW1 power switch and unplug the AC line cord. To avoid shocks, you should discharge the capacitor by placing the leads of a 100Ω resistor across the capacitor terminals for about 20 seconds. You may also want to connect your VOM across the capacitor and watch the voltage drop as the resistor discharges it.

☐ **V1 [+].** Attach the regulator to its heatsink. If you mount the heatsink on the same metal chassis you've grounded through the green wire of the power line cord, you must electrically isolate V1 from its heatsink. Figure 12 shows how to isolate V1 using a TO-3 transistor socket and a TO-3 mounting hardware kit (Radio Shack 276-029 and 276-1371), with heatsink grease (Radio Shack 276-1372) for heat transfer.

Apply a film of heatsink grease to the bottom of V1 and fit the nonconductive mica washer — the grease will hold it in place. Make sure the holes line up, then apply grease to the bottom of the washer.

Fit the socket to the heatsink from the bottom, making sure the socket's plastic bushings project into the mounting holes and that the socket connections for the two regulator pins line up with the corresponding clearance holes in the heatsink. If things don't line up or fit correctly, you have the socket backwards or the wrong heatsink.

Hold the socket in place and plug in V1, making sure the washer's screw holes stay in line with the regulator's holes. Then insert the two mounting screws and tighten until the socket is tight against the bottom of the heatsink, with the plastic bushings in their holes.

If your assembly doesn't require isolation, you can eliminate the mica washer, or even eliminate the socket by bolting V1 directly to the heatsink and soldering the connecting wires directly to the regulator pins and a solder lug attached to one of the mounting screws.

☐ **Isolation test.** With your VOM set on scale R x 1, connect one VOM lead to the heatsink. Touch the second lead to the heatsink too and adjust your meter to read 0 ohms. Then touch the second lead to all three lugs of the socket. The two side or pin lugs should read very high resistance, an open circuit. For an isolated assembly using the mica washer, the center pin should also read a very high resistance; with a nonisolated assembly it should read 0. If you get 0 on a lug where you should find a high resistance, that lug is touching the heatsink and is not isolated. Rework your assembly to get the necessary isolation.

☐ **V1 connections.** Solder a different-color no. 18 insulated wire to each regulator socket pin, and note your colors below:

Color	Pin	Function
________	1 (B)	Input
________	2 (E)	Output
________	3 (case)	Common (ground)

If you use the same Radio Shack socket that I used, the letters in parentheses match the markings on the bottom of the socket. Pin 3 is the tab in the middle of the metal strip connecting to the two mounting screws.

☐ **C2 [+].** Wrap the plus (+) lead around V1 socket pin 1, and the minus (-) lead around V1 socket pin 3, keeping both exposed leads as short as possible so they won't touch anything else. Solder the connection at pin 1, being careful not to unsolder the wire previously attached.

☐ **C3.** Wrap one lead around V1 socket pin 2 and the other around V1 socket pin 3, keeping both exposed leads as short as possible so they won't touch anything else. Solder both connections, being careful not to unsolder the previous connections on these pins.

☐ **Heatsink installation.** Attach the heatsink to your power-supply chassis, making sure that you maintain V1's isolation by not letting bare wires or pins touch the chassis or heatsink.

☐ **V1/C1 (negative).** Connect the wire from V1 pin 3 to the minus (-) terminal of C1.

☐ **V1/C1 (positive).** Connect the wire from V1 pin 1 to the plus (+) terminal of C1.

☐ **Labels.** Cut out the labels from fig. 1 (or a photocopy) and glue them beside the power supply output terminals to mark one screw terminal as the +5VDC output and the other as ground.

☐ **Output connection.** Connect the wire from V1 pin 2 to the +5VDC output terminal.

☐ **Ground connection.** Run a no. 18 wire from the ground output terminal to the minus (-) terminal of C1.

☐ **L1 (pilot light).** This LED lights up to indicate when the power supply is turned on. Use small, solid wire to connect it, along with a 160Ω resistor, as shown in fig. 10.

☐ **Final test.** Plug in the line cord and turn on the toggle; the pilot light should come on. Set your VOM to the 10VDC range, and attach the plus (+) meter lead to the +5VDC supply terminal and the negative (-) meter lead to the ground terminal. Make a note of your measurement: ________. If everything is working correctly your output reading will be between 4.8 and 5.2VDC, the tolerance limits of the regulator. This completes the 5VDC power supply.

You can also use these instructions for building supplies with greater current capacity. Simply use a larger chassis box, use a higher-current transformer, mount the bridge rectifier on a heatsink, mount larger (and/or more, parallel) filter capacitors, and allow space for more V1 regulator/heatsinks, using the same circuit shown in fig. 10.

I've now covered the construction of all the basic C/MRI hardware. In the next installment I'll explain how to test the operation of the complete interface, including two brief BASIC programs you will use to test the I/O Cards and their connections to your computer through the UBEC. See you next month. ♦

In this fifth installment, Bruce Chubb explains how to test the C/MRI hardware, shown here with the Computer Output Test Panel and the other test equipment that you will need. The computer is included because the article presents programs that turn it into a C/MRI tester.

Paul A. Erler

C/MRI-5

The C/MRI: A computer/model railroad interface

Part 5: Testing the system with software

BY BRUCE CHUBB

IF YOU have been building along with me, you've now built all the hardware for your C/MRI. You can be proud of that, and soon you'll be even prouder when you see it at work with your model railroad. This month we'll test the system to see that everything is assembled and connected correctly.

We'll start out with an initial VOM check of your C/MRI system, less the I/O Cards. Then I'll show how to build the Test Panel that will let you use software to verify the operation of your entire C/MRI, including the I/O Cards.

INITIAL SYSTEM TESTING

The following tests check your intracard and most of your intercard connections. Systematic checking takes time, but it also goes a long way toward ensuring that your C/MRI will operate correctly when you first turn it on.

The most likely sources of trouble will be poor soldering and incorrect parts insertion. The next thing to look for is cable trouble, like CBAC connections that are not correct for your particular computer, or an open or poor contact at one of the ribbon-cable connector pins.

If you haven't already done it, get out your computer reference manual and make sure that all the connections are correct for *your* computer. If the computer pinouts aren't listed in the manuals you have, then at least double check against the tables for your machine in C/MRI part 3, figs. 7-15. Check each box only after you get the correct results.

☐ **Initial connections.** Connect the cable(s), CBAC, and UBEC as in C/MRI part 3, but don't connect anything to your computer. No UBEC (or IBM PC CBAC) ICs should be installed, and the Mother Board should not be connected to the UBEC.

☐ **Input line tests.** Set your VOM scale to R x 100 and clip its minus (–) lead to ground on the UBEC, either to the wide circuit trace where J32 connects to the edge of the card or to J32 itself. To reach into IC sockets with the plus (+) meter lead, clamp a short piece of no. 24 bus wire in the meter lead's clip, or in a clip lead on the meter probe. Insert the wire "probe" into the pin-7 hole of S9, which is grounded. A reading of zero ohms (0Ω) shows that both your VOM and test connections are okay.

Check each point listed in fig. 1 to see that no UBEC or CBAC input lines are accidentally grounded by soldering. You should read an open circuit with infinite resistance for each test — the needle of an analog meter shouldn't move — except where otherwise indicated in the figure. As each line passes, check off the boxes at the left of fig. 1.

A reading of 0Ω indicates either a solder bridge grounding the input line, or a case where the line should be grounded, as noted at the bottom of fig. 1. If the line should be grounded, check the test as passed. If it shouldn't be, follow the circuit path away from the IC pin under test until you find the bridge to ground, then remove it. Retest the line and be sure it passes before checking it off.

☐ **Output line tests.** Repeat the procedure of the previous test, using the test points in fig. 2, to see that none of the UBEC output lines are grounded. Every point should give an open-circuit reading.

☐ **Computer-UBEC connection tests.** Keep your VOM on R x 100 and follow fig. 3 to check the continuity of your computer-UBEC connections. Place one meter lead on the indicated IC pin or test point, and the other on the corresponding connector pin on the computer side of the CBAC (or in the female ribbon-cable connector). Skip tests that don't apply to your computer. If your computer isn't covered in this series, list the pinouts for your machine in the rightmost column of fig. 3. When you read continuity (0Ω), check off the box for that line and go on.

Wherever you don't read continuity you may have trouble. First check that you've counted connector pins correctly, as it's easy to start from the wrong side or end of a connector. If you're sure there's a fault, work your meter leads along the circuit path to track it down and correct it. If it's at a cable connection, unplug and replug that connector: breaking down oxidation on the studs may correct the problem.

☐ **UBEC/Mother Board connection tests.** Use a 40-conductor ribbon cable to connect the I/O Mother Board to the UBEC, but don't connect its power supply or plug in any I/O Cards. Jameco offers custom ribbon cables of any length for the cost of the cable and connectors plus $5. Order no. S40-XX-S40 Custom, substituting the desired cable length in feet for the Xs.

Keep your VOM on R x 100 and follow fig. 4 to check the continuity of each line: attach one meter lead to each pin of any one I/O Card slot in succession, and touch the other lead to the matching UBEC IC pin or test point.

Again, if you read continuity check off that line and go on, and if you don't, find and correct the fault.

☐ **Computer power test.** Set your VOM to measure +5VDC, then clip the – lead to ground on the UBEC and the positive + lead to the +5V line: the wide trace where J21 connects at the edge of the card or to J21 itself.

Warning: Always make sure your computer is turned off whenever you connect or disconnect the C/MRI, and whenever you install or remove ICs on the UBEC with the C/MRI connected.

With the computer off, connect the UBEC as shown in C/MRI part 3. Now turn on the computer. You should read +5VDC on your meter if the computer is supplying power to the UBEC.

If you read 0V, recheck your connections. If correct, use the VOM to check for +5V at the appropriate computer connector pins, on the CBAC, and at each cable connector. Find where you lose voltage, then establish a path for it to the UBEC.

If you have a TRS-80 Model 1 and don't get +5VDC on your UBEC, it may be that this power is not brought out to pin 39 of the expansion connector, as is the case on some later versions of the Model 1. If this is the problem, cut the CBAC circuit trace going to pin 39 and follow the procedure explained in C/MRI part 3 for the TRS-80 Model 3, adding V1 to the UBEC and powering the card with a separate 9VAC adapter.

☐ **Computer operation test.** Try running some games or other programs with the C/MRI connected. A malfunction at this stage indicates a problem with the CBAC or UBEC, most likely a solder bridge touching one of the address lines, data lines, or control lines.

With the computer off, examine the stud connectors, IC socket pins, and other closely spaced joints for solder bridges. To help isolate the problem, repeat the operation test with first the Mother Board disconnected, then the UBEC disconnected, and finally the CBAC.

If you've isolated a faulty card but can't see the problem, use your VOM to recheck each address, data, and control input to that card to find the offending line. Most likely a solder bridge will connect one of the input lines to another input, to ground, or to +5VDC. Trace until you find the solder bridge and remove it. Repeat until the computer operates properly with the CBAC, UBEC, and Mother Board all connected.

☐ **UBEC DIP switch test.** With the computer turned on and the C/MRI connected, connect your – meter lead to ground and move the + lead to the IC socket pinouts listed in fig. 5. At each position turn the switch segment on and off to see if the corresponding IC socket point switches between 0V and +5VDC. The off or open position should give +5V at the IC pin, and the on or closed position should read 0V.

If all readings are reversed, you most likely have the DIP switch installed backwards. Heat each pin and use desoldering braid to wick up the solder until you can remove SW1 and reinstall it correctly. If only some segments are incorrect, check around U5, U6, SW1, and R4-R11 for solder bridges or open circuits. Repeat until all eight switch segments work as they should.

☐ **IC installation.** Turn off your computer, then install ICs U1-U9 on the UBEC, and also U1 on the CBAC for the IBM PC. Be sure you have the ICs specified for your machine in C/MRI part 2, and that you put them in the right sockets with correct pin-1 orientation. Figure 6 shows how to insert and extract ICs.

Also, now is the time to set SW1 to your railroad's starting address. Either use the setting for your computer shown in C/MRI part 3, or calculate the starting address for another machine or application following the guidelines in that installment. If you have an IBM PC, set SW1 on the CBAC as well.

☐ **Computer operation test.** Check your computer's operation by again running some programs. If it doesn't work properly, recheck the placement and orientation of all ICs in the UBEC. If necessary remove ICs from their sockets one by one — turning the computer off and on again each time — to see if your problem is in the part or in the circuit itself. If you do trace a problem to an IC, replace it with a new part. It's handy to keep at least one spare of each IC type in your C/MRI for debugging and replacement.

☐ **UBEC IC power test.** Set your VOM to measure +5VDC. With the computer on, follow fig. 7 to check that each UBEC IC is receiving +5VDC. If you find an IC not getting power, work along the circuit path until you locate the open circuit and correct it.

☐ **I/O Mother Board power test.** Turn off the computer and connect your power supply to the Mother Board's +5VDC and GND terminals. With the power supply off, turn on the computer and check its operation, then turn on your power supply and make sure the computer still operates correctly. Turning power on and off while a program is running should have little or no effect on the computer's operation.

If this stops the program, it may be causing a line transient in your house wiring. Try plugging the computer and power supply into outlets on different circuits; this may correct the problem.

☐ **RRON test.** Connect your VOM's – lead to UBEC ground and its + lead to pin 4 of U9, then turn the +5VDC supply on and off. The meter should read 0V when the supply is off, and +5V when it's on. If not, check along the circuit path for a bad solder joint, solder bridge, missing jumper, or a bad or incorrectly inserted U7 and/or Q1.

☐ **Output Card power test.** Set the DIP switch on your first Output Card to your railroad's starting address, as shown in C/MRI parts 3 and 4. Turn off the +5VDC supply and plug the Output Card into any Mother Board slot. Turn the power back on and check that your computer operates properly, and that your power supply delivers between +4.8 and +5.2VDC.

If not, your Output Card probably has a short circuit. Look for and correct reversed ICs and/or solder bridges. If necessary, remove ICs one at a time (with power off) to help isolate the problem.

☐ **Output Card IC power test.** Follow fig. 8 to see that power reaches each IC on the Output Card. If not, look for a missing jumper or bad solder joint. Keep checking until your C/MRI passes all tests.

TEST PANEL

Our next tests will use software, the instructions that make the hardware work. Our first set of instructions, or program, will be a short one to show that a message can be sent from the computer to the Output Card. We'll test the computer's ability to write to each of 24 output lines; to do that efficiently we'll need both the test program and a Computer Output Test Panel to mount on the Output Card. See fig. 9.

The Test Panel's LEDs display the status of each of the 24 output lines at any time. This makes the initial checkout a snap, and can also come in handy any time you need to debug the C/MRI or confirm that it's operating properly.

A circuit card for the Test Panel can be purchased from JLC Enterprises, P. O. Box 88187, Grand Rapids, MI 49508-9187. Order card OUTEST-24, $16 each, and include $2.50 per order for shipping and handling (Michigan residents must add 4 percent sales tax). Free artwork for etching your own card is available by sending a stamped, self-addressed, business-size envelope to C/MRI PC Layouts, MODEL RAILROADER Magazine, 1027 N. Seventh St., Milwaukee, WI 53233.

Figure 9 includes the schematic, parts layout, and parts list for the Test Panel. I'll just name those assembly steps like those you've already performed on other cards, and give details on anything new.

☐ **Card inspection.**
☐ **J1-J12.**
☐ **R1-R24.**
☐ **S1, S2.**

INPUTS TO UBEC			
	UBEC		
✓	IC SOCKET	PIN	LINE SYMBOL
	S4	2	D0
	S4	4	D1
	S4	6	D2
	S4	8	D3
	S3	2	D4
	S3	4	D5
	S3	6	D6
	S3	8	D7
	S1	8	A0
	S1	6	A1
	S1	4	A2
	S1	2	A3
	S1	17	A4
	S1	15	A5
	S1	13	A6
	S1	11	A7
	S5	13	A8
	S5	10	A9
	S6	13	A10
	S6	10	A11
	S5	1	A12
	S5	4	A13
	S6	1	A14
*	S6	4	A15
	V1 PAD I		+5V
	PAD P1		CL1
	PAD P2		CL2
**	PAD CL3A		CL3
***	S5	3	AH

* Should read ground for VIC-20
** Should read ground for all but TRS-80 Model 3 and IBM PC
*** Used for IBM PC only

Fig. 1 UBEC INPUT TESTS

OUTPUTS FROM UBEC			
	UBEC		
✓	IC SOCKET	PIN	LINE SYMBOL
	S4	18	D0
	S4	16	D1
	S4	14	D2
	S4	12	D3
	S3	18	D4
	S3	16	D5
	S3	14	D6
	S3	12	D7
	S1	12	A0
	S1	14	A1
	S1	16	A2
	S1	18	A3
	S1	3	A4
	S1	5	A5
	S1	7	A6
	S1	9	A7
	S2	3	AH
	S2	5	W*
	S2	7	R*

Fig. 2
UBEC OUTPUT TESTS

FIGURE 1 NOTES:
- CL3 line connected to ground for all applications except TRS-80 Model 3 and IBM PC
- A8-A15 not used for TRS-80 Model 3, or for TRS-80 Model 1 with port I/O
- For VIC-20 the A-15 line is connected to ground on CBAC
- AH line used only for IBM PC
- +5VDC on Heathkit H8 is actually +8VDC

	UBEC			COMPUTER CONNECTOR PINOUTS							
✓	IC SOCKET	PIN	LINE SYMBOL	HEATHKIT H8	TRS-80 Model 1	TRS-80 Model 3	TRS Color Computer	Apple II, II+ and IIe	VIC-20	Commodore 64	IBM PC
	S1	8	A0	30	25	17	19	2	B	Y	A31
	S1	6	A1	31	27	19	20	3	C	X	A30
	S1	4	A2	32	40	21	21	4	D	W	A29
	S1	2	A3	33	34	23	22	5	E	V	A28
	S1	17	A4	34	31	25	23	6	F	U	A27
	S1	15	A5	35	35	27	24	7	H	T	A26
	S1	13	A6	36	38	29	25	8	J	S	A25
	S1	11	A7	37	36	31	26	9	K	R	A24
	S5	13	A8	38			27	10	L	P	A23
	S5	10	A9	39			28	11	M	N	A22
	S6	13	A10	40			29	12	N	M	A21
	S6	10	A11	41			30	13	P	L	A20
	S5	1	A12	42			31	14	R	K	A19
	S5	4	A13	43			37	15	S	J	A18
	S6	1	A14	44			38	16	13	H	A17
	S6	4	A15	45			39	17	***	F	A16
	S4	2	D0	10	30	1	10	49	2	21	A9
	S4	4	D1	11	22	3	11	48	3	20	A8
	S4	6	D2	12	32	5	12	47	4	19	A7
	S4	8	D3	13	26	7	13	46	5	18	A6
	S3	2	D4	14	18	9	14	45	6	17	A5
	S3	4	D5	15	28	11	15	44	7	16	A4
	S3	6	D6	16	24	13	16	43	8	15	A3
	S3	8	D7	17	20	15	17	42	9	14	A2
	S1	10	GND	0	8	2	33	26	1	1	B31
	V1 PAD I		+5V	48*	39	**	9	25	21	2	B29
	PAD P1		CL1	23	12	35	18	18	18	5	B11
	PAD P8		CL2	28	19	33	6	40	V	E	B12
	S2	12	CL3			43					
	S5	3	AH								****
	PAD P7		AEN								A11

* For Heathkit H8 +5VDC line is actually +8VDC
** For TRS-80 Model 3 +5VDC must be supplied externally or via pin 45
*** For VIC-20 A15 line on UBEC is connected to ground on CBAC
**** For IBM PC AH continuity should be to S1 pin 3 on CBAC

Fig. 3 COMPUTER-UBEC CONTINUITY TESTS

☐ **Panel Plate.** Make the LED display panel as shown in fig. 10. You can lay the printed panel face over the aluminum and prick-punch through the paper to mark the metal for drilling.

☐ **L1-L24, positioning [+].** Install the LEDs with the + leads in the holes next to the resistors and the – leads in the holes closest to the connectors — fig. 10 shows how to identify LED leads. DO NOT solder the LEDs at this time.

☐ **Panel assembly.** Mount the panel on the card as in fig. 10, carefully working each LED into its panel hole.

☐ **L1-L24, soldering.** One at a time, push the lip of each LED firmly against the panel, then solder and trim the leads. Solder the two 4-40 brass nuts to their circuit board pads.

☐ **Clip lead.** Cut the clip lead in two leaving one end about 12" long. Strip off ⅛" of insulation from the cut end and install as shown in fig. 10.

☐ **Cleanup and inspection.**

☐ **Panel face.** Cut the panel face out of fig. 10 or a photocopy. Dismount the panel and use spray adhesive to cement the face to it, with the dots centered over the holes. Use an X-acto knife with a no. 11 blade to trim excess paper off the edges, then work from the face side to trim out the 26 holes. Remount the panel.

☐ **Card test.** Attach the card's clip lead to the +5VDC terminal of your power supply. Attach another clip lead to the ground terminal and, with the supply turned on, touch its other end to each of the 24 connector inputs. Only one LED should light for each input, according to the test sequence in fig. 10. If more than one lights you have a solder bridge. If one doesn't light you have a poor solder joint, a bad LED, a reversed LED, or a faulty resistor. Debug until all 24 LEDs work correctly.

This completes the assembly and test of the Test Panel. Next we'll use some software to drive those LEDs!

INTRODUCTION TO SOFTWARE

"Software," as I noted above, means the instructions placed in a computer's memory to tell the computer what action to take. These instructions can be mathematical or logical. A group of instructions is referred to as a program, and a program loaded into a computer defines, step-by-step, what kind of a machine the computer will be.

When we load a check-balancing program, the computer helps us balance the family checkbook. Load a space-war program and the same computer is a space-war game. We'll eventually need railroad programs, but our first program simply drives the Test Panel LEDs to make the computer a C/MRI tester.

Programs can be written in many different programming languages, but the most common one for home computers is BASIC. It's easy to learn and comes with almost every home computer.

On the other hand, BASIC has limitations for "real-time" operation with a model railroad, interacting with the railroad as it runs. Because it is an interpreted language, not separately compiled into digital machine code, BASIC is very slow in executing instructions. When we get to applications like signaling and automatic block-power assignment, I'll discuss using other languages for faster response, but BASIC is more than adequate for getting started.

	MOTHER	UBEC		
✓	BOARD PIN NO.	IC SOCKET	PIN	LINE SYMBOL
	1	S9	7	GND
	2	S2	10	GND
	3	S1	16	A2
	4	S1	18	A3
	5	S3	3	AH
	6	S1	3	A4
	7	S1	5	A5
	8	S1	7	A6
	9	S1	9	A7
	10	NO TEST		SW5V
	11	S3	12	D7
	12	S3	14	D6
	13	S3	16	D5
	14	S3	18	D4
	15	S4	12	D3
	16	S4	14	D2
	17	S4	16	D1
	18	S4	18	D0
	19	S7	7	GND
	20	S2	5	W*
	21	S1	12	A0
	22	S1	14	A1
	23	S2	7	R*
	24	End of R1 next to S10		5V

Fig. 4 MOTHER BOARD CONTINUITY TESTS

	UBEC		
✓	IC SOCKET	PIN	DIP SWITCH SEGMENT
	S5	12	A8
	S5	9	A9
	S6	12	A10
	S6	9	A11
	S5	2	A12
	S5	5	A13
	S6	2	A14
	S6	5	A15

With switch segment set to "ON" IC socket pin should read 0V. With switch segment set to "OFF" IC socket pin should read +5VDC

Fig. 5 UBEC DIP SWITCH TESTS

✓	IC	+METER LEAD ON PIN NO.	-METER LEAD ON PIN NO.
	U1	20	10
	U2	20	10
	U3	20	10
	U4	20	10
	U5	14	7
	U6	14	7
	U7	14	7
	U8	14	7
	U9	14	7

Some applications don't use all ICs. You should read +5VDC on all that are installed

Fig. 7 UBEC POWER TESTS

✓	IC	+METER LEAD ON PIN NO.	-METER LEAD ON PIN NO.
	U1	14	7
	U2	14	7
	U3	14	7
	U4	14	7
	U5	14	7
	U6	14	7
	U7	14	7
	U8	26	7

Each line should read +5VDC

Fig. 8 I/O CARD IC POWER TESTS

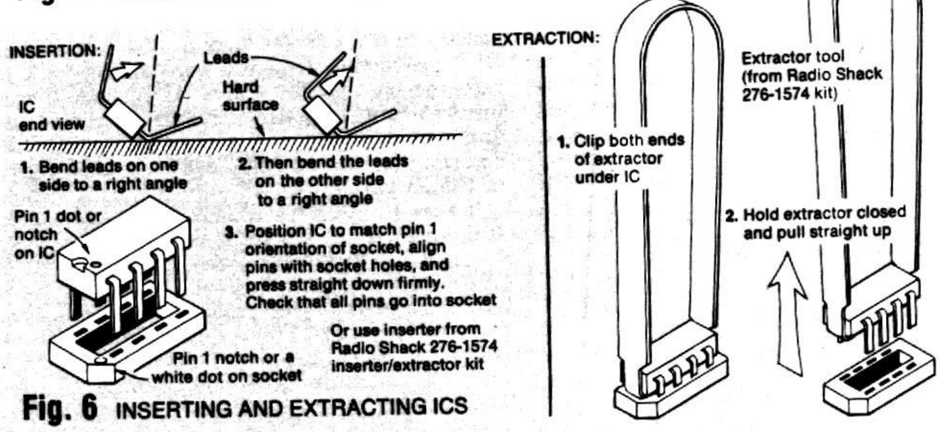

INSERTION:
Leads
Hard surface
IC end view
1. Bend leads on one side to a right angle
2. Then bend the leads on the other side to a right angle
Pin 1 dot or notch on IC
3. Position IC to match pin 1 orientation of socket, align pins with socket holes, and press straight down firmly. Check that all pins go into socket
Or use inserter from Radio Shack 276-1574 inserter/extractor kit
Pin 1 notch or a white dot on socket
EXTRACTION:
1. Clip both ends of extractor under IC
Extractor tool (from Radio Shack 276-1574 kit)
2. Hold extractor closed and pull straight up

Fig. 6 INSERTING AND EXTRACTING ICS

BASIC instructions vary a bit from one manufacturer to another. The BASIC I'll use in this series is MicroSoft Interpreted BASIC from Heath/Zenith. Dialects vary from machine to machine, but MicroSoft BASIC is the most widely used version of BASIC high-level language, and it's found on such popular computers as the TRS-80, IBM PC, and many others. MicroSoft BASIC programs are easy to convert into other BASICs. If you understand what each instruction means, often all you need to do is substitute the appropriate command words or symbols for your version.

I can't teach you BASIC here, but most of the BASIC instructions we'll need are easy to understand — they're very close to plain English. Almost every home computer comes with a BASIC programming manual, and computer stores have a wide assortment of books on BASIC. One title I've always liked is *BASIC BASIC* — you can hardly get more basic than that!

INITIALIZING CARD OUTPUTS

With your C/MRI connected and BASIC loaded into the computer, plug the Test Panel onto an Output Card installed on your Mother Board.

Warning: Turn off the +5VDC power supply whenever you add or remove an I/O Card on the Mother Board. Turn off both the computer and the +5VDC supply when plugging in or unplugging the UBEC and/or CBAC.

You need not turn off your computer or your +5VDC supply when making connections to or disconnections from the I/O Cards, such as the Test Panel.

With the Test Panel plugged in, connect the card's clip lead to the +5VDC terminal on the Mother Board. The Test Panel's LEDs should light in a random pattern, some on and some off. If none light, make sure you have the +5VDC supply turned on and connected. If you installed the optional switch for controlling Mother Board bus line 10, set it for +5VDC. Now you can use computer instructions to first initialize all the LEDs to off, then control them as you desire.

Before you can write data to the LEDs, you must set up the Output Card's 8255 as an output device by sending it the appropriate output control byte — the 128 shown in fig. 3 of C/MRI part 4. If you're using memory-mapped I/O, this is accomplished using BASIC's POKE instruction. For port I/O the OUT instruction gives the same result. I'll cover the memory-mapped case first.

POKE lets you transmit any data value from 0 to 255 into any defined address location from 0 to 65535. For example POKE 36192,73 transmits a value of 73 into address location 36192.

To intialize the 8255, enter the following statement, replacing the Xs with the decimal number you set as the port A address for the Output Card under test: POKE XXXXX+3,128.

For example, if your Output Card was at address 1028, you'd enter POKE 1028 +3,128, or POKE 1031,128. The 3 must be added to the 1028, as address 1028 would be Port A, 1029 Port B, and 1030 Port C; 1031 would be the address for the control byte.

This statement sets all three 8255 ports as outputs, and initializes and latches all 24 output lines to logic zero, 0V. That causes all the output transistors to turn off, or not conduct, which turns off all the LEDs on the Test Panel.

If you're using a railroad starting address above 65535 with an IBM PC, you must precede any use of PEEK and POKE instructions with a DEF SEG instruction. I'll explain this in more detail with the first output test program.

Once you've initialized the 8255 for output, you can use further POKE instructions to display any combination of outputs. For example POKE 1028+0,1 turns on bit 0 of port A, while POKE 1028+0,0 turns it off. POKE 1028+1,7 turns on bits 2, 1, and 0 of port B. Figure out some instructions of your own, and try to visualize the displays before you enter them.

For port I/O the 8255 is initialized for all outputs using the instruction OUT XXX+3,128, with the Xs replaced by the decimal address set as the port A address for the Output Card under test. For example if your Output Card was at port address 0 (a typical value for the first I/O card slot with the TRS-80 Model 1 or 3), you would enter: OUT 0+3,128, or OUT 3,128.

As in the memory-mapped example, this instruction should turn off all 24 LEDs on the Computer Output Test Panel. Then you can use subsequent OUT instructions to set up output displays, just as with the POKE instructions I described above.

If you're using the TRS-80 Model 3 you'll need to remember a couple of other points. As I explained in C/MRI part 3, with the Model 3 you must precede any use of external port I/O with the instruction OUT 236,16. Also, BASIC operation in the immediate mode causes the Model 3's user I/O port to be disabled after each instruction is executed. Use two program instructions followed by a RUN command to initialize your 8255, such as:

```
10 OUT 236,16
20 OUT 0+3,128
RUN
```

IN CASE OF DIFFICULTY

If these instructions, tailored for your own computer and card address, don't control the LEDs on your Test Panel, your software isn't communicating with your Output Card. You have what's known as an "interface communication problem." The question is, "how do I fix it?" The solution will probably be very simple once you find it, but first you have to find it!

The best way to start is to patiently and carefully recheck your entire C/MRI per installments 2, 3, and 4. Do you have the CBAC and UBEC set up correctly? Are all the ribbon-cable connectors plugged in correctly? Are the program jumpers on the UBEC correct for your computer? Have you used the right ICs in the UBEC for your computer, are they all installed with correct pin-1 orientation, and are all the pins properly pushed into their corresponding socket holes? And so on, checking everything methodically.

If you have a TRS-80 Model 3, is J48 on the UBEC in the alternate position? If you have an IBM PC, have you left J48 out altogether, installed J50 in the alternate position, and added J51?

Also, reread the sections of your computer's BASIC manual that cover the POKE or OUT instructions, to be doubly sure your BASIC commands are correct.

If you've built two Output Cards you can try again with the second one, but set its DIP switch to the same address so you're making the same test.

Make sure you have the correct railroad starting address set on the UBEC DIP switch, and the correct card-slot setting on the Output Card DIP switch. Are all the switches installed with correct "on/off" orientation? If all switch settings aren't correct for your C/MRI configuration, and if they don't correspond exactly to the address in your POKE or OUT instruction, then you won't get any response on the Test Panel. If you are unsure about the settings, review binary and hexadecimal numbers, as well as determining the starting address, back in C/MRI part 3.

If everything above is correct and you still don't get any response, then go back to page 80 and repeat the "INITIAL SYSTEM TESTING" sequence, making sure that your C/MRI passes each test. Re-examine each card carefully to see that all parts are inserted correctly, that all solder joints are good, and that there are no solder bridges between adjacent pins or traces.

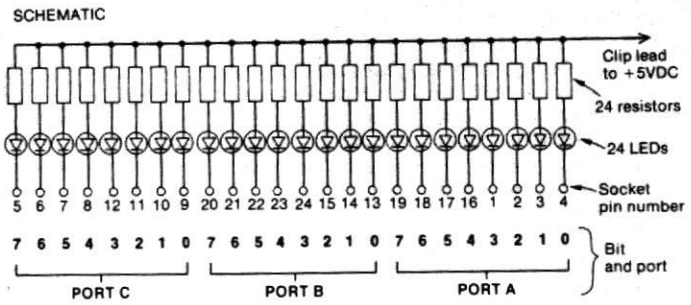
Remember that most problems can be traced to faulty soldering, parts insertion, or cabling. Keep searching and retesting until you find the error, correct it, and achieve successful operation.

If you still can't find the trouble, get a friend, preferably one with some electronics background, to go through all the checks with you. Often someone else will find something wrong where the person who made the error passes right over it. As you perform the checks, have your friend challenge you with questions like, "How do you know that is the correct part?" and, "Are all the wires in that connector making contact?" As you try to explain what you're doing at each step you will very likely find your problem. In industry this is referred to as a "design walk-through."

If you tailored your own CBAC and/or UBEC to adapt the C/MRI to a computer not covered in this series, you may have a line or lines hooked up wrong. See fig. 2 in C/MRI part 2, and check for improper logic inversions between the computer bus and the I/O Cards.

If you can't find any errors, and plugging in new ICs doesn't help, try to get help from a computer expert in your area. In many computer clubs you'll find hobbyists just looking for tough problems to solve. The technician at your computer service center may be able to help you on the side.

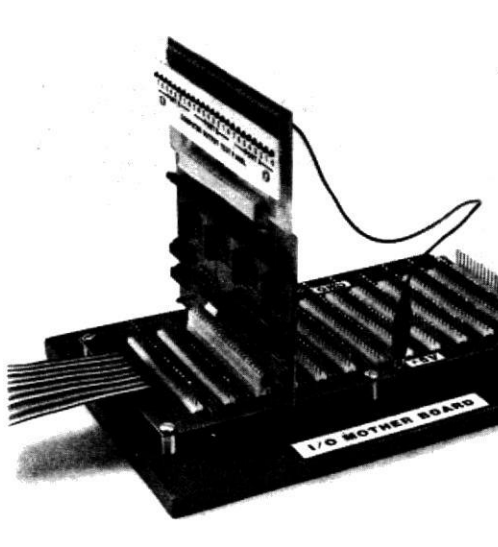
If you live near a college or school teaching electronics, check out the digital electronics classes. A student may be able to help you and earn course credit for doing it, plus a payment (from you) for helping out. An experienced person with the aid of an oscilloscope and/or a



TEST PANEL PARTS LIST

Qnty.	Symbol	Description
12	J1-J12	Jumpers, make from uninsulated bus wire (Belden no. 8022 or equivalent)
24	R1-R24	150Ω resistors [brown-green-brown]
2	S1, S2	12-pin connectors (Molex 09-52-3121)
24	L1-L24	Red diffused LED (Jameco XC209R)
2	—	¼"-dia. x ¼"-long standoffs (Digi-Key J167)
2	—	4-40 x ½"-long pan-head machine screws
1	—	Clip lead cut in half (Radio Shack 278-001)
1	—	1/16" x 1¼" x 6¼" aluminum LED panel

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



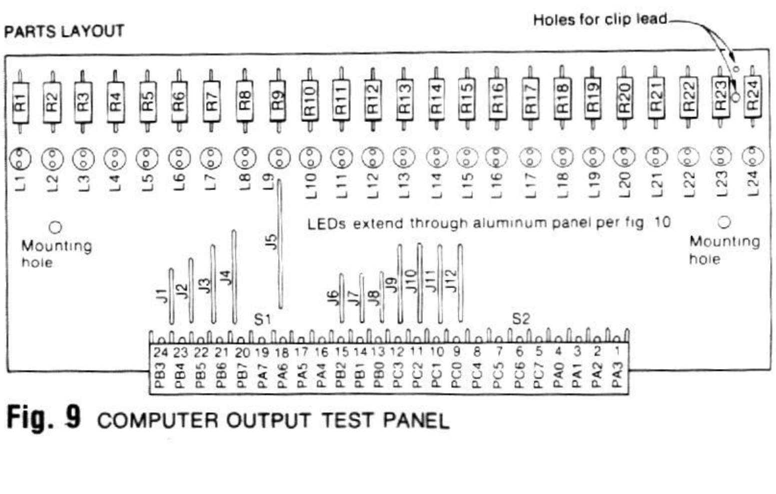
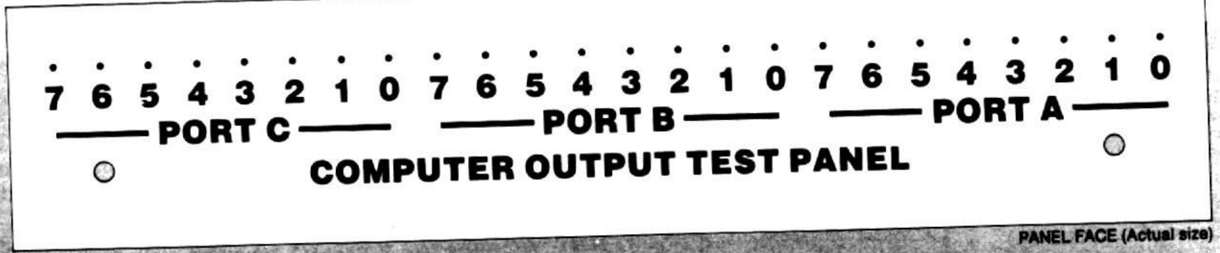

PARTS LAYOUT
Holes for clip lead
R1 R2 R3 R4 R5 R6 R7 R8 R9 R10 R11 R12 R13 R14 R15 R16 R17 R18 R19 R20 R21 R22 R23 R24
L1 L2 L3 L4 L5 L6 L7 L8 L9 L10 L11 L12 L13 L14 L15 L16 L17 L18 L19 L20 L21 L22 L23 L24
LEDs extend through aluminum panel per fig 10
Mounting hole
Mounting hole
J1 J2 J3 J4 J5 J6 J7 J8 J9 J10 J11 J12
S1
S2
24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
PB3 PB4 PB5 PB6 PB7 PA7 PA6 PA5 PA4 PB2 PB1 PB0 PC3 PC2 PC1 PC0 PC4 PC5 PC6 PC7 PA0 PA1 PA2 PA3

Fig. 9 COMPUTER OUTPUT TEST PANEL



PANEL FACE (Actual size)

CONNECTOR-PIN/LED CONTINUITY TESTS

✓	SOCKET PIN NO.	LED TURNED ON	
		PORT	BIT
	1	A	3
	2	A	2
	3	A	1
	4	A	0
	5	C	7
	6	C	6
	7	C	5
	8	C	4
	9	C	0
	10	C	1
	11	C	2
	12	C	3
	13	B	0
	14	B	1
	15	B	2
	16	A	4
	17	A	5
	18	A	6
	19	A	7
	20	B	7
	21	B	6
	22	B	5
	23	B	4
	24	B	3

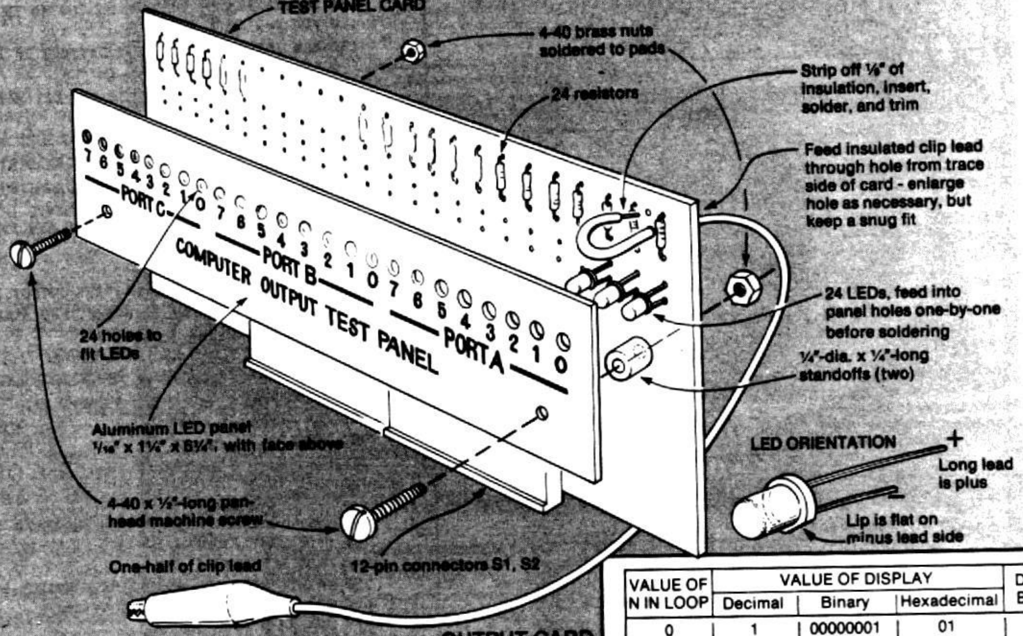

TEST PANEL CARD
4-40 brass nuts soldered to pads
24 resistors
Strip off ⅛" of insulation, insert, solder, and trim
Feed insulated clip lead through hole from trace side of card - enlarge hole as necessary, but keep a snug fit
24 LEDs, feed into panel holes one-by-one before soldering
¼"-dia. x ¼"-long standoffs (two)
LED ORIENTATION
Long lead is plus
Lip is flat on minus lead side
24 holes to fit LEDs
Aluminum LED panel 1/16" x 1¼" x 8¾", with face above
4-40 x ½"-long pan-head machine screw
One-half of clip lead
12-pin connectors S1, S2
COMPUTER OUTPUT TEST PANEL
PORT C
PORT B
PORT A

Fig. 10 TEST PANEL ASSEMBLY

Fig. 11 OUTPUT CARD LED-DRIVER TEST PROGRAM

VALUE OF N IN LOOP	VALUE OF DISPLAY			DISPLAY BIT "ON"
	Decimal	Binary	Hexadecimal	
0	1	00000001	01	0
1	2	00000010	02	1
2	4	00000100	04	2
3	8	00001000	08	3
4	16	00010000	10	4
5	32	00100000	20	5
6	64	01000000	40	6
7	128	10000000	80	7

BIT PATTERN FOR RELATIONSHIP 2↑N

```
1Ø DIM PN$(2)
2Ø PN$(Ø)="A":  PN$(1)="B":  PN$(2)="C"
4Ø REM DEFINE 8255 CONTROL BYTE
5Ø CO=128
6Ø REM DEFINE PORT A ADDRESS FOR CARD UNDER TEST
7Ø REM**VALUE OF MM IS APPLICATION DEPENDENT**
8Ø MM=1Ø28
9Ø PRINT "OUTPUT CARD LED DRIVER TEST PROGRAM"
1ØØ REM INCREMENT PORT TO BE TESTED IN A LOOP
11Ø FOR P=Ø TO 2
12Ø REM OUTPUT 8255 CONTROL BYTE
13Ø POKE MM+3,CO
14Ø REM INCREMENT DISPLAYED BIT NUMBER IN A LOOP
15Ø FOR N=Ø TO 7
16Ø REM OUTPUT TEST STATUS TO CRT
17Ø PRINT "PORT IS",PN$(P),"BIT NUMBER IS",N
18Ø REM OUTPUT LED DISPLAY DATA TO CARD
19Ø D=2↑N
2ØØ POKE MM+P,D
21Ø REM WAIT FOR CARRIAGE RETURN TO INCREMENT
22Ø INPUT "ENTER RETURN TO CONTINUE TEST",CR
23Ø REM COMPLETE LOOPS
24Ø NEXT N
25Ø NEXT P
26Ø REM REPEAT TEST
27Ø GOTO 9Ø
```

Changes if exponent not available

```
15 DIM D(7)
25 D(Ø)=1:  D(1)=2:  D(2)=4:  D(3)=8:  D(4)=16
26 D(5)=32:  D(6)=64:  D(7)=128
19Ø **DELETE THE LINE**
2ØØ POKE MM+P,D(N)
```

Changes for PORT I/O

```
13Ø OUT MM+3,CO
2ØØ OUT MM+P,D
```

Changes for TRS-80 Model 3

```
85 REM ENABLE USER PORT I/O
86 OUT 236,16
```

Changes for IBM PC

```
85 REM DEFINE USER REGISTER SEGMENT
86 REM **VALUE IN DEF SEG IS APPLICATION DEPENDENT**
87 DEF SEG 4Ø96Ø
```

logic analyzer should be able to find most any interface problem with minimum effort.

The most important requirement for problem-solving is persistence. The reward will be worth it.

OUTPUT CARD TEST PROGRAM

The previous testing established a basic level of communication between the software and the Test Panel. Now I'll take you one step further with a BASIC program, listed in fig. 11, that will demonstrate the complete output capabilities of your interface.

The program is for memory-mapped I/O; for port I/O substitute OUT instructions for the two POKE instructions as noted in fig. 11. The program uses seven variables defined as follows:

PN$( ): a three-position array used to print out the port being tested; PN$(0) = "A", PN$(1) = "B" and PN$(2) = "C".

CO: 8255 output control byte = 128.

MM: memory map address of port A for the Output Card under test = decimal address for your first Output Card.

P: numeric variable identifying the port to be tested; 0 = port A, 1 = port B, and 2 = port C.

N: bit number (LED) to be lit, 0-7.

D: data output byte to be sent to the LEDs on the test circuit.

CR: dummy variable for inputting a carriage return from the keyboard.

Since this is the first program in this series, I'll briefly explain each line of code. Line 10 sets aside three locations in memory and labels them PN$(0), PN$(1), and PN$(2). Line 20 stores the symbol "A" in PN$(0), "B" in PN$(1), and "C" in PN$(2).

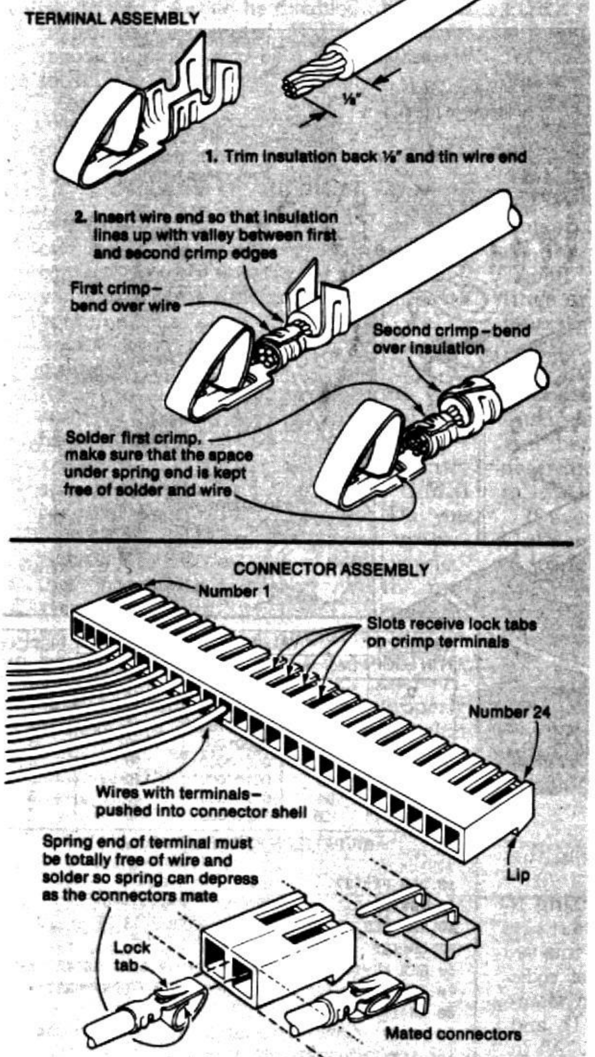
BASIC lines beginning with REM, for *REM*ark, are not instructions to be carried out. Their function is simply to carry comments that help readers understand the program. Thus line 40 explains what line 50 does: it gives control-byte variable CO the numeric value 128.

The starting address for the Sunset Valley's memory-mapped I/O is 1024, but my first card is an Input Card, and my first Output Card, the card this program tests, is at the second card address: 1024 + 4 or 1028. Thus my line 80 sets MM to 1028, but this is an "application-dependent" variable that you must set to reflect your own C/MRI's address for the card you are testing.

Line 90 causes the computer to display the program title on the CRT. Line 110 begins a test loop by incrementing variable P from 0 to 1 to 2, to test all three ports. Line 130 sends the output control byte to the 8255 to initialize the ports for output. Line 150 sets up an inner loop whereby the bit number (LED) to be lit (N) is incremented from 0 to 7. To keep track of what's being tested, line

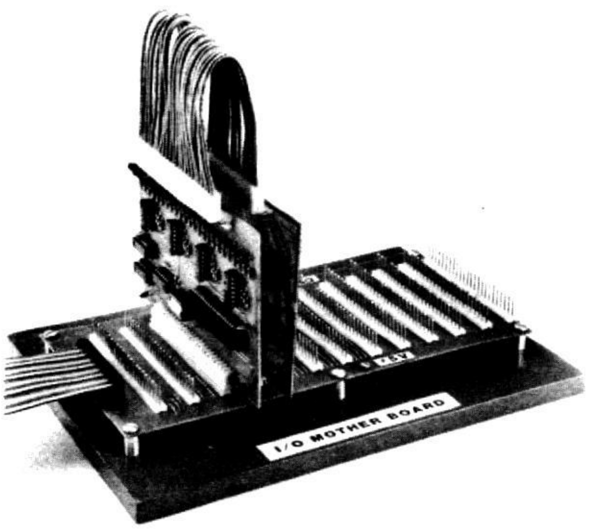
WRAPAROUND CABLE PARTS LIST

2	Molex no. 09-50-3241 connector shell
48	Molex no. 08-50-0106 crimp terminals
24	15"-lengths 20-AWG stranded insulated wire (Belden no. 8502 or equivalent)



TERMINAL ASSEMBLY
1/8"
1. Trim insulation back 1/8" and tin wire end
2. Insert wire end so that insulation lines up with valley between first and second crimp edges
First crimp–bend over wire
Second crimp–bend over insulation
Solder first crimp, make sure that the space under spring end is kept free of solder and wire
CONNECTOR ASSEMBLY
Number 1
Slots receive lock tabs on crimp terminals
Number 24
Wires with terminals–pushed into connector shell
Spring end of terminal must be totally free of wire and solder so spring can depress as the connectors mate
Lip
Lock tab
Mated connectors

Fig. 12 WRAPAROUND TEST CABLE



```
10 DIM PN$(2)
20 PN$(0)="A": PN$(1)="B": PN$(2)="C"
40 REM DEFINE 8255 CONTROL BYTE
50 CO=128
55 CI=155
60 REM DEFINE PORT A ADDRESS FOR CARDS UNDER TEST
70 REM **FOLLOWING CARD ADDRESSES ARE APPLICATION DEPENDENT**
80 MO=1028
82 MI=1024
90 PRINT "WRAPAROUND TEST PROGRAM FOR I/O CARDS"
100 REM INCREMENT PORT TO BE TESTED IN A LOOP
110 FOR P=0 TO 2
120 REM INCREMENT TEST BIT PATTERN IN A LOOP
130 FOR D= 0 TO 255
140 REM OUTPUT CONTROL BYTE AND BIT PATTERN TO OUTPUT CARD
150 POKE MO+3,CO
160 POKE MO+P,D
170 REM OUTPUT CONTROL BYTE AND READ BIT PATTERN FROM INPUT CARD
180 POKE MI+3,CI
190 C=PEEK(MI+P)
200 REM PRINT TEST STATUS OUTPUT AND INPUT ON CRT
210 HO$=HEX$(D)
220 HI$=HEX$(C)
230 PRINT "PORT IS",PN$(P),"HEX OUTPUT IS",HO$,"HEX INPUT IS",HI$
240 REM COMPARE INPUT TO OUTPUT AND BRANCH IF NO ERROR FOUND
250 IF C=D THEN 300
260 REM ERROR FOUND SO PRINT TO CRT
270 PRINT "ERROR FOUND IN ABOVE I/O, ENTER RETURN TO CONTINUE TEST"
280 INPUT CR
290 REM COMPLETE LOOPS
300 NEXT D
310 NEXT P
320 GOTO 40
```

Changes if hexadecimal conversions not available

```
210 **DELETE LINE**
220 **DELETE LINE**
230 PRINT "PORT IS",PN$(P),"DEC OUTPUT IS",D,"DEC INPUT IS",C
```

Changes to simplify manual hex conversions

```
130 FOR N= 0 TO 7
155 D=2↑N
300 NEXT N
```

Changes for PORT I/O

```
150 OUT MO+3,CO
160 OUT MO+P,D
180 OUT MI+3,CI
190 C=INP(MI+P)
```

Changes for TRS-80 Model 3

```
85 REM ENABLE USER PORT I/O
86 OUT 236,16
```

Changes for IBM PC

```
85 REM DEFINE USER REGISTER SEGMENT
86 REM **VALUE IN DEF SEG IS APPLICATION DEPENDENT**
87 DEF SEG 40960
```

Fig. 13 AUTOMATIC WRAPAROUND TEST PROGRAM

SAMPLE CRT OUTPUT OF AUTOMATIC WRAPAROUND TEST PROGRAM

NO ERROR

```
PORT  IS       A              HEX OUTPUT IS  0       HEX INPUT IS  0
PORT  IS       A              HEX OUTPUT IS  1       HEX INPUT IS  1
PORT  IS       A              HEX OUTPUT IS  2       HEX INPUT IS  2
PORT  IS       A              HEX OUTPUT IS  3       HEX INPUT IS  3
PORT  IS       A              HEX OUTPUT IS  4       HEX INPUT IS  4
PORT  IS       A              HEX OUTPUT IS  5       HEX INPUT IS  5
PORT  IS       A              HEX OUTPUT IS  6       HEX INPUT IS  6
PORT  IS       A              HEX OUTPUT IS  7       HEX INPUT IS  7
PORT  IS       A              HEX OUTPUT IS  8       HEX INPUT IS  8
PORT  IS       A              HEX OUTPUT IS  9       HEX INPUT IS  9
PORT  IS       A              HEX OUTPUT IS  A       HEX INPUT IS  A
PORT  IS       A              HEX OUTPUT IS  B       HEX INPUT IS  B
PORT  IS       A              HEX OUTPUT IS  C       HEX INPUT IS  C
PORT  IS       A              HEX OUTPUT IS  D       HEX INPUT IS  D
PORT  IS       A              HEX OUTPUT IS  E       HEX INPUT IS  E
PORT  IS       A              HEX OUTPUT IS  F       HEX INPUT IS  F
PORT  IS       A              HEX OUTPUT IS  10      HEX INPUT IS  10
PORT  IS       A              HEX OUTPUT IS  11      HEX INPUT IS  11
  :            :                     :                     :
```

ERROR INDUCED

```
PORT  IS       A              HEX OUTPUT IS  0       HEX INPUT IS  1
ERROR FOUND IN ABOVE I/O, ENTER RETURN TO CONTINUE TEST
?
PORT  IS       A              HEX OUTPUT IS  1       HEX INPUT IS  1
PORT  IS       A              HEX OUTPUT IS  2       HEX INPUT IS  3
ERROR FOUND IN ABOVE I/O, ENTER RETURN TO CONTINUE TEST
?
PORT  IS       A              HEX OUTPUT IS  3       HEX INPUT IS  3
PORT  IS       A              HEX OUTPUT IS  4       HEX INPUT IS  5
  :            :                     :                     :
```

170 gives a CRT display of the port under test and the bit number to be lit.

The next step is to calculate and send the bit patterns to the port under test, so we can see if they get through correctly to light the proper LED on the Test Panel. Line 190 does the calculation by setting D equal to 2 raised to the power N. This arithmetical statement sets the bit pattern relationships shown in fig. 11 for each value of N from 0 through 7. Line 200 transmits D to the Output Card to light bit number N of port P.

Line 220 is a CRT prompt message asking for input from the operator if the test is to continue. Lines 240 and 250 loop back to lines 150 and 110 to increment N and P. Once the program has sent an output to all eight bits in all three ports, it reaches line 270, which repeats the test by branching back to line 90.

As I said earlier, there can be subtle differences between one dialect of BASIC and another. For example, your particular BASIC may use a symbol other than the colon (:) I used for including more than one statment on a line, as in line 20.

Figure 11 includes some special-case instructions, too. If your BASIC doesn't have the exponent function used in line 190, you can get the same result by deleting that line and adding/changing the other four lines as indicated. For port I/O, use OUT instructions instead of POKEs in lines 130 and 200, and for the TRS-80 Model 3, add lines 85 and 86 to enable the user I/O port.

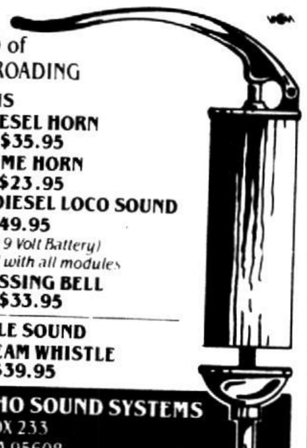
If you have an IBM PC and set your railroad starting address above 65535, the maximum address BASIC can handle in PEEK and POKE instructions, you'll need to use a DEF SEG (*DEF*ine *SEG*ment) instruction. For example, to use the suggested PC starting address of $A0000 = 640K = 655360 from C/MRI part 3, include a DEF SEG = 40960 instruction before any PEEK or POKE instructions. The value of DEF SEG is the starting address divided by 16, 655360 ÷ 16 = 40960 in the example above.

Once you have defined the current segment of storage, a subsequent PEEK or POKE definition identifies the address of any railroad I/O Card as an offset into this segment. For example, a POKE 3,128 with the segment defined as 40960 places a 128 into address location $40960 \times 16 + 3 = 655360 + 3$, or 655363. That would set the 8255 in the first I/O Card slot as an all-output device.

To perform the test enter the program with adjustments for your application, SAVE it to disk so you'll have it to use again, then RUN the program. Compare the port and bit combination on the CRT with the LED that lights up on the Test Panel. If they agree, press RETURN to increment the display. Every time you do this the program will loop, and the next LED to the left should come on.

If the LEDs don't agree with the CRT display, you have some debugging to do. First make sure your program is correct by LISTing it and checking it against fig. 11. Once one port works correctly, any remaining problem is almost certainly on the Output Card itself. Check the card for bad solder joints, solder bridges, reversed ICs, and so on. By following the schematic

The **SOUNDS** of MODEL RAILROADING



DIESEL HORNS
- **3 CHIME DIESEL HORN** (less speaker) **$35.95**
- **SINGLE CHIME HORN** (less speaker) **$23.95**
- **"O" SCALE DIESEL LOCO SOUND MODULE $49.95** (less speaker & 9 Volt Battery) 16V.A.C. needed with all modules
- **GRADE CROSSING BELL** (with speaker) **$33.95**

STEAM WHISTLE SOUND
- **5 CHIME STEAM WHISTLE** (less speaker) **$39.95**

WESTWARD HO SOUND SYSTEMS
POST OFFICE BOX 233
CARMICHAEL, CA 95608

Whether you're building 1980's diesels, or 1880's structures...

Start with a new catalog.

For 1985 we've added 22 new manufacturers including Roco, Alloy Forms, Pike Stuff, Pacer/Robart, MZZ, Quality Products and Maxon Electronics, just to name a few. Other manufacturers have thousands of their newest products in **The World of HO Scale — 1985** which have never been shown before!

If you're looking for new ways of doing things don't miss 'The Magic of Model Railroading' section. You'll find 33 pages of full color model, layout and diorama photos, loaded with new ideas and inspiration for your modeling!

Order your copy of **The World of HO Scale — 1985** and get going on your special projects!

WALTHERS
5601 W. Florist Ave.
Milwaukee, WI 53218
Enclosed is my check for $11.95
Please send me **The World of HO Scale — 1985** *(913-626)* and a list of participating Walthers Dealers in my area.
Please print M

Name ____________________

Address ____________________

City, State, Zip ____________________

© 1985 Wm. K. Walthers, Inc.

with your VOM set for +10VDC, you can look directly at the port-bit outputs of the 8255 and the 7407 buffers to help isolate any particular line failure.

Success with the test program proves not only that the Output Card works correctly, but also that the CBAC, UBEC, cabling, address decoding, and software are okay. It's an effective visual test of C/MRI's output. Once you get one Output Card working on all three ports, the hardest part of the C/MRI project is behind you. Total interface success is just a small step ahead — testing the Input Card. Since you know everything else checks out, this will be much easier.

AUTOMATED TEST PROGRAM

We could test the Input Card by connecting a DIP switch to the card inputs and using a test program to read the switch settings and display the results on the CRT. However, it's much easier to use the computer to perform the whole test automatically. That's one of the advantages of having a computer connected to your railroad.

Since you've already tested your first Output Card, you can use it to test all the Input Cards you need. In the computer industry this kind of I/O check is called a "wraparound test," since the computer's outputs are simply wrapped around and connected to its inputs.

For that you need the wraparound cable shown in fig. 12, to connect the Output Card to the Input Card you want to test. Each pin of one connector is simply wired straight to the corresponding pin of the other connector. Figure 12 shows how to crimp the terminals to the wires and insert them into the connector shells. Later you'll do the same thing to connect railroad devices to your I/O cards.

The automated-wraparound-test program in fig. 13 writes a bit pattern to the Output Card and then reads the Input Card. It compares what it reads with what it wrote, and if they don't agree it has found an I/O error and displays an error message on the CRT. Since you've already proven the Output Card, the fault will most likely be in the Input Card.

If the program reads exactly what it wrote, it changes the bit pattern and repeats the write/read cycle. Once all bit patterns pass the test, both cards in the test loop are proven good.

Since the wraparound test program closely parallels the LED-driver test program we just used to test the Output Card, I'll explain only the significant differences. Line 55 defines the 8255 control byte for input and line 82 the Port A address for the Input Card under test. Lines 170 through 190 set up and read the Input Card. Lines 200 through 220 convert output-bit pattern D and input-bit pattern C to hexadecimal displays.

If your BASIC doesn't have a decimal-to-hexadecimal conversion function, the first change at the bottom of the listing in fig. 13 will print out D and C directly. That's not as handy because you'll have to do conversions by hand to locate bad bit(s), but it still gets the job done. The next change simplifies that process, so if you need to make the first, you'll want to make the second.

The other changes at the bottom of fig. 13 cover port instead of memory-mapped I/O, and special instructions for the TRS-80 Model 3 and IBM PC.

The important comparison test is the IF, THEN statement in line 250. If C equals D there is no error, and the program branches to line 300 to increment to the next bit pattern D and repeat the cycle. If C doesn't equal D there's an error, and a message is displayed on the CRT. To continue the test you simply press RETURN. The program runs continuously, looping through all combinations of bit patterns for each port.

CONDUCTING THE TEST

To conduct the automated wraparound test follow these four steps: first connect the wraparound test cable between the Output and Input Cards you want in the test loop; then enter the program from fig. 13, changing lines 80 and 82 to set your port A addresses for the cards under test; SAVE the program so you'll have it to use again; and then RUN the program. Depending on your version of BASIC, you may have to alter the format of some instructions to get the program to RUN correctly.

Figure 13 includes a few lines of sample program output as seen on the CRT. I forced that error message by grounding one of the input connector pins on the Input Card with a clip lead. If you do get errors, first check to see that you have the correct addresses set in the DIP switches on each card, and that these settings correspond to the addresses in the program. If, for example, you accidentally interchanged the addresses, you would be trying to write to the Input Card and read from the Output Card, and nothing would work.

Seeing which bits fail can help locate where on the Input (or Output) card you have bad soldering, a part incorrectly inserted, or a faulty part. To find which bit or bits are faulty, expand the hex representations of C and D into their binary equivalents using the table back in fig. 3 in C/MRI part 3. Whichever bits don't compare define the area of trouble. Probe those parts of the card with your VOM, looking for 0V or +5VDC on the appropriate lines, to pinpoint the problem.

Once you have a proven Input Card, you can use it to automatically test the rest of your Output Cards, just as your first proven Output Card can be used to test Input Cards. Keep your test programs on a diskette for use if you ever suspect faulty operation in your C/MRI.

At this point you can feel very proud. Your C/MRI is fully operational, and you've written and executed two interface programs.

In part 6 of the C/MRI series I'll show how to connect a variety of model railroad devices to your interface, and in part 7 I'll tell you how to build and install an optimized track-occupancy detector. Then we'll move on to programming for real C/MRI applications, computer-controlled signaling and computer cab control. Once you have a couple of these applications under your belt and start experimenting with software, the excitement can easily get out of control. ⚪

Bruce Chubb

On Bruce Chubb's Sunset Valley RR, C/MRI output lines control trackside signals, panel lights, and switch machines, while input lines carry information on block occupancy, turnout position, and control switches back to the computer. This installment of Bruce's C/MRI series tells how to make output and input connections from your layout to the General Purpose I/O Cards; block-occupancy detection is coming up next month.

C/MRI-6

The C/MRI: A computer/ railroad inte

Part 6: Connecting your railro

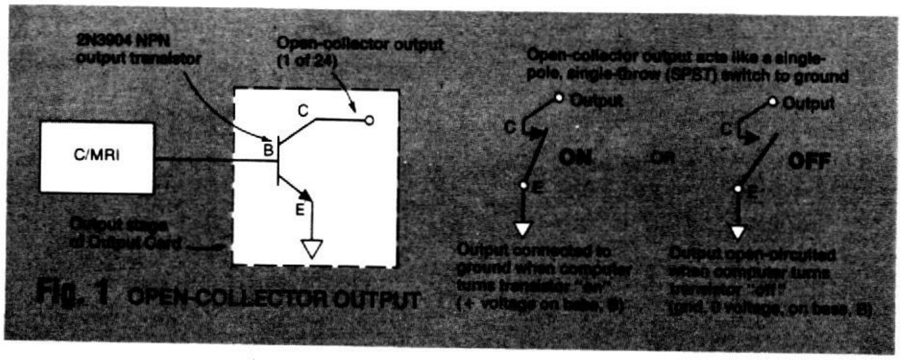
BY BRUCE CHUBB

NOW it's time to start connecting your tested and operational C/MRI to your railroad. This month I'll show you how lamps, LEDs, relays, logic gates, opto-isolators, panel switches, and switch machines connect to the C/MRI I/O Cards. Let's start by looking at how easily the General Purpose Output Card can drive a variety of model railroad devices.

OUTPUT TO THE RAILROAD

Remember that each Output Card can transmit 24 discrete signals to the railroad using open-collector transistors as the output devices. The easiest way to understand the open-collector transistor circuit is to think of it as an electrical switch with one end connected to ground, as in fig. 1. The computer, under software control as I'll explain in later installments, simply turns the transistor on and off.

When the transistor is on, its collector


2N3904 NPN output transistor
Open-collector output (1 of 24)
C/MRI
B
C
E
Output stage of Output Card
Open-collector output acts like a single-pole, single-throw (SPST) switch to ground
Output
C
E
ON
OR
OFF
Output connected to ground when computer turns transistor "on" (+ voltage on base, B)
Output open-circuited when computer turns transistor "off" (gnd, 0 voltage, on base, B)
Fig. 1 OPEN-COLLECTOR OUTPUT

or C terminal is, in effect, connected to ground, so the switch is closed. When the transistor is off, the collector is in effect an open circuit, and the switch is open.

Thinking of each C/MRI output line as a simple switch to ground, it's easy to see how to connect typical model railroad devices. Figure 2 shows circuits for several common items. GOW lamp operation, fig. 2a, is straightforward: switch closed (to ground), lamp on; switch open, lamp off. The LED in fig. 2b is the same except that you need a current-limiting resistor in series with the diode.

With a relay, fig. 2c, the power supply voltage must be compatible with the relay and a diode must be added across the coil. The diode protects the output transistor by suppressing inductively induced voltage spikes when the relay is turned off. Diode polarity is important, and in this case the banded end of the diode should be toward the + power supply terminal.

Figure 2d needs a bit more explanation. With the output transistor turned off (switch open), you have +5V into the TTL IC input, and with the transistor on (switch closed), the collector is grounded and you get 0V into the TTL input. The 2.2KΩ resistor pulls the TTL input up to +5VDC when the transistor is in the off (open circuit) state. That's why this is often called a "pull-up" resistor.

(In fact the collector voltage levels are not exactly 0V and 5V. On the high end — collector open — there is a small current draw from the TTL input, which causes a small voltage drop through the 2.2KΩ pull-up resistor. Nor is the low zero, because of "diode losses" in the transistor junctions. The TTL gate input swing will actually be from a low of +.25V to a high of about 4.75V, but that's okay as TTL circuits are designed to analyze any input less than .8V as low and above 2.4V as high.)

The connections in circuit 2e are identical to 2d's, except that CMOS (*C*omplementary *M*etal *O*xide *S*emiconductor) circuits typically use a power supply in the range of 9 to 12VDC.

If you have excessively noisy conditions (electrically) and/or some special output that must be electrically isolated from the C/MRI, you can use an opto-isolator as in fig. 2f. This is an LED and a photo diode in a single package, and light is the only coupling between the two devices.

When the output transistor is on it turns on the LED, and that in turn puts the photo diode into its low-resistance (on) state. When the transistor is off, the LED is off too and the photo diode approaches an open circuit (off) state.

In all cases except TTL or CMOS ICs, supply voltages and loads can be varied as necessary, such as a 24V lamp with a 24V supply. The only limitations are that the load musn't draw more than .3A and the supply must be less than 40VDC — the ratings of the output transistor.

Supply voltage can be different for different output lines, but it's best to stay with 5VDC for as many applications as possible. Then it's easy to combine loads on a single computer output line. For example, fig. 3 shows one output line controlling four LEDs, a relay, a lamp, and two TTL logic gates.

DRIVING HEAVY LOADS

For loads exceeding the output transistor ratings — 40VDC and .3A — you need a booster circuit between the Output Card and the load. Figure 4 shows the one I use to drive the Sunset Valley's switch machines: surplus rotary relays powered by a 32VDC, 35A supply.

These relay switch machines have a holding coil and a throwing coil. A built-in SPST contact cuts out the throwing coil once it throws, and the machine is then held in reverse position by the holding coil. When power goes off, a return spring restores the machine to normal (unenergized) position. The throw current is around 4A, far exceeding the capacity of the Output Card's transistor.

The circuit in fig. 4 handles a steady 10A and surges of 15A, and supply voltages up to 60V, all from your computer! Q2 is controlled by PNP transistor Q1. When the input terminal is brought low, by the Output Card transistor turning on, Q1 turns on, and it turns on power Darlington Q2. With Q2 on, the output switches from open-circuit to ground, energizing the switch machine.

Again, any time you drive an inductive load — one with a coil such as a relay or switch machine — add a diode across the coil with its banded end toward the + power supply terminal.

SWITCH MACHINES

You could use fig. 4 booster circuits for twin-solenoid switch machines, one for each coil, but it wouldn't be a good idea. If an error in your program allowed a switch machine output line to be low for more than the .25 second or so needed to throw the machine, or set both lines to a given machine low at the same

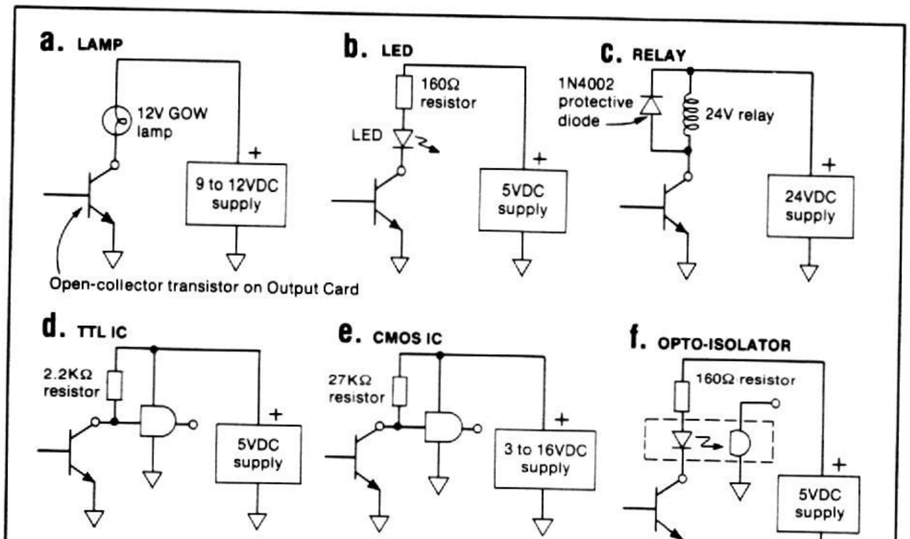
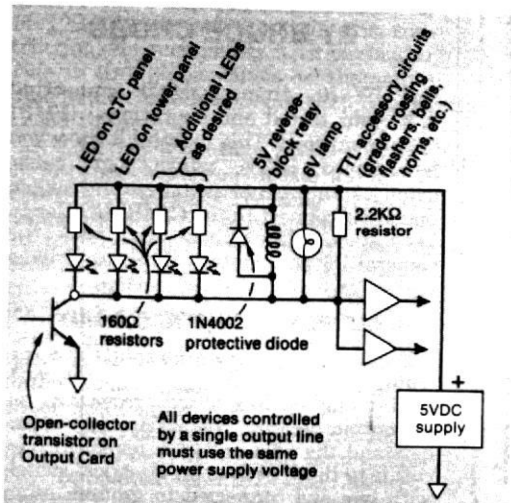
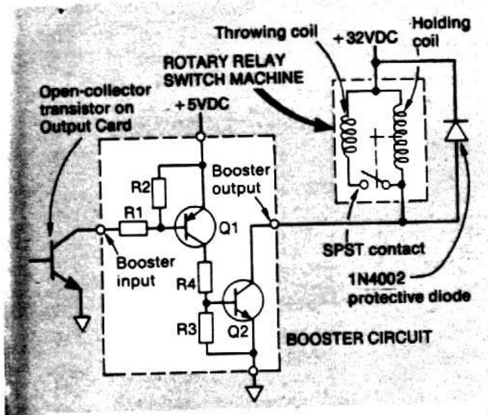


Fig. 2 CONNECTIONS FOR C/MRI OUTPUT


LED on CTC panel
LED on tower panel
Additional LEDs as desired
5V reverse-block relay
6V lamp
TTL accessory circuits (grade crossing flashers, bells, horns, etc.)
2.2KΩ resistor
160Ω resistors
1N4002 protective diode
+
5VDC supply
Open-collector transistor on Output Card
All devices controlled by a single output line must use the same power supply voltage
Fig. 3 MULTIPLE LOADS ON A SINGLE C/MRI OUTPUT LINE
(Typical Sunset Valley RR practice)

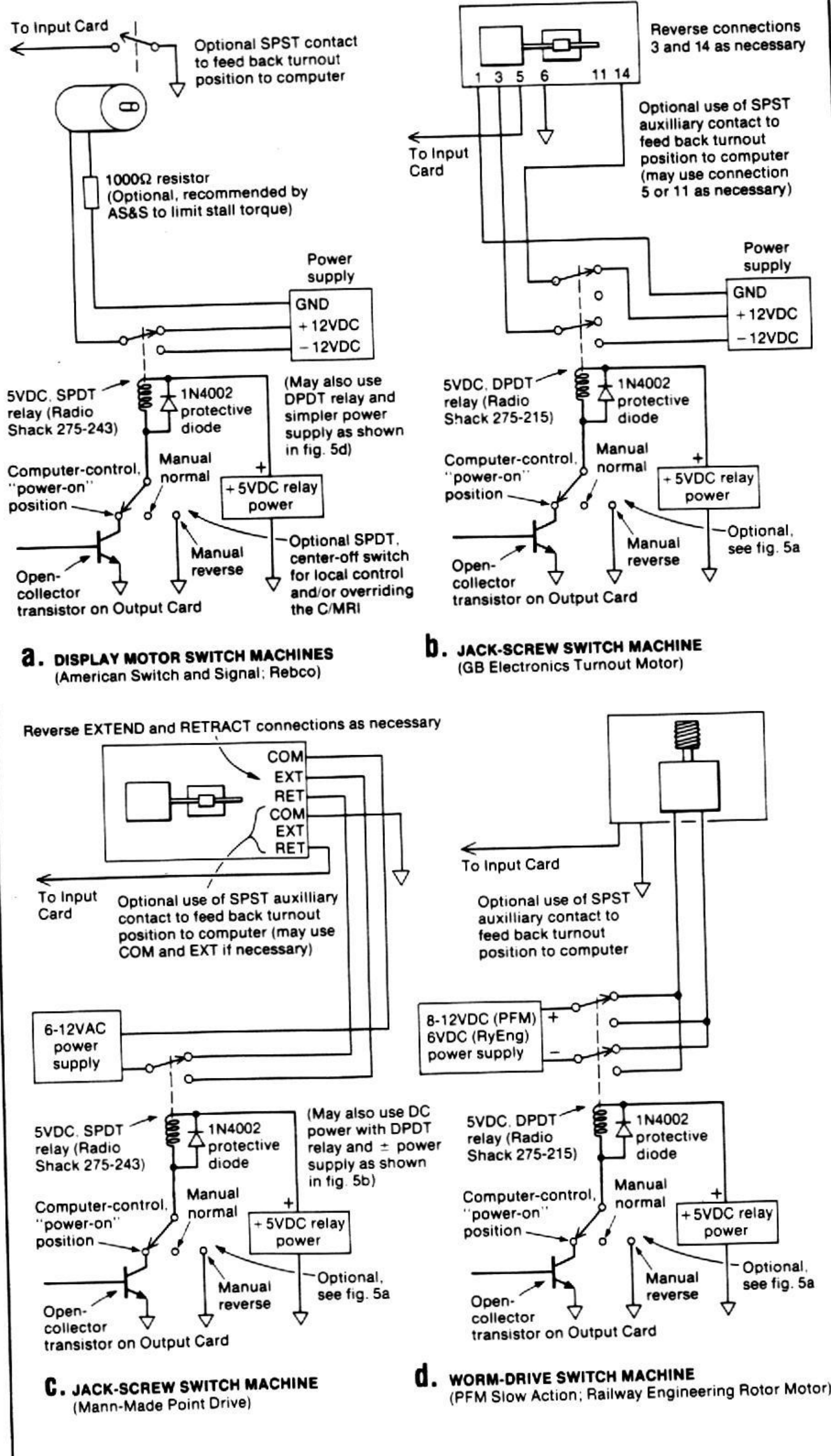


BOOSTER CIRCUIT PARTS LIST

R1	470Ω resistor [yellow-violet-brown]
R2, R3	1000Ω resistor [brown-black-red]
R4	47Ω, ½W resistor [yellow-violet-black]
Q1	2N2907 PNP small signal transistor (Jameco PN2907)
Q2	2N6387 NPN power Darlington (Mouser 597-2N6387)

Author's recommendations for suppliers given in parentheses above, with part numbers where applicable. Resistors are ¼W, 5 percent except as specified, and color codes are given in brackets. Mouser Electronics' address is 11433 Woodside Ave., Santee, CA 92071, (619)449-2222. Equivalent parts may be substituted

Fig. 4 BOOSTER CIRCUIT FOR HEAVY OUTPUT LOADS



To Input Card
Optional SPST contact to feed back turnout position to computer
1000Ω resistor (Optional, recommended by AS&S to limit stall torque)
Power supply
GND
+12VDC
-12VDC
5VDC, SPDT relay (Radio Shack 275-243)
1N4002 protective diode
(May also use DPDT relay and simpler power supply as shown in fig. 5d)
Computer-control, "power-on" position
Manual normal
+5VDC relay power
Open-collector transistor on Output Card
Manual reverse
Optional SPDT, center-off switch for local control and/or overriding the C/MRI
a. DISPLAY MOTOR SWITCH MACHINES (American Switch and Signal; Rebco)
Reverse connections 3 and 14 as necessary
1 3 5 6 11 14
Optional use of SPST auxilliary contact to feed back turnout position to computer (may use connection 5 or 11 as necessary)
To Input Card
Power supply
GND
+12VDC
-12VDC
5VDC, DPDT relay (Radio Shack 275-215)
1N4002 protective diode
Computer-control, "power-on" position
Manual normal
+5VDC relay power
Manual reverse
Optional, see fig. 5a
Open-collector transistor on Output Card
b. JACK-SCREW SWITCH MACHINE (GB Electronics Turnout Motor)
Reverse EXTEND and RETRACT connections as necessary
COM
EXT
RET
COM
EXT
RET
To Input Card
Optional use of SPST auxilliary contact to feed back turnout position to computer (may use COM and EXT if necessary)
6-12VAC power supply
5VDC, SPDT relay (Radio Shack 275-243)
1N4002 protective diode
(May also use DC power with DPDT relay and ± power supply as shown in fig. 5b)
Computer-control, "power-on" position
Manual normal
+5VDC relay power
Manual reverse
Optional, see fig. 5a
Open-collector transistor on Output Card
c. JACK-SCREW SWITCH MACHINE (Mann-Made Point Drive)
To Input Card
Optional use of SPST auxilliary contact to feed back turnout position to computer
8-12VDC (PFM) 6VDC (RyEng) power supply
5VDC, DPDT relay (Radio Shack 275-215)
1N4002 protective diode
Computer-control, "power-on" position
Manual normal
+5VDC relay power
Manual reverse
Optional, see fig. 5a
Open-collector transistor on Output Card
d. WORM-DRIVE SWITCH MACHINE (PFM Slow Action; Railway Engineering Rotor Motor)

Fig. 5 OUTPUT FOR MOTOR-DRIVEN SWITCH MACHINES

time, you might easily smoke a switch machine. Even with perfect software, when you initialize the system the 8255 can come up with random outputs.

If you installed the optional Mother Board ground switch shown in C/MRI part 4, setting that to ground would keep things safe during initialization. You could still have trouble, though, if it were left in the +5V position at the wrong time.

If you are building a new layout, or if you want to change switch machines for some other reason, it would be easiest not to use dual-coil machines at all with your C/MRI. The many kinds of motor-driven switch machines are simpler to control with a computer, and their slow movement is prototypical and easier on turnouts. Figure 5 shows how to use C/MRI-controlled relays, like we saw in fig. 2c, to drive several types of motor switch machines.

If you already have dual-coil machines installed and don't want to change them, you *can* safely drive them from the C/MRI. I'll show you two methods.

The first, circuit SM1 in fig. 6, is easy and direct. Two C/MRI output lines drive the coils *via* two booster circuits and an individual capacitor-discharge (CD) power supply. The boosters are like the one in fig. 4, but with resistor values so that everything, including the CD supply, can use the same 16-24VDC power. Each SM1 circuit can be mounted close to the machine it controls, with all SM1s fed by a central 16-24VDC filtered supply.

Using a separate CD supply for each machine protects the coils, because the capacitor stores just enough energy to throw the machine once, and the supply circuit won't let it recharge until the C/MRI signals return to high state. That makes an accidental burnout impossible, and also makes the circuit very immune to electrical noise.

Here are the steps for building the SM1 on the JLC circuit card:

☐ **Card inspection.**

☐ **Terminal screws.** Insert seven 4-40 screws from top side, tighten nuts on bottom, and solder nuts to traces.

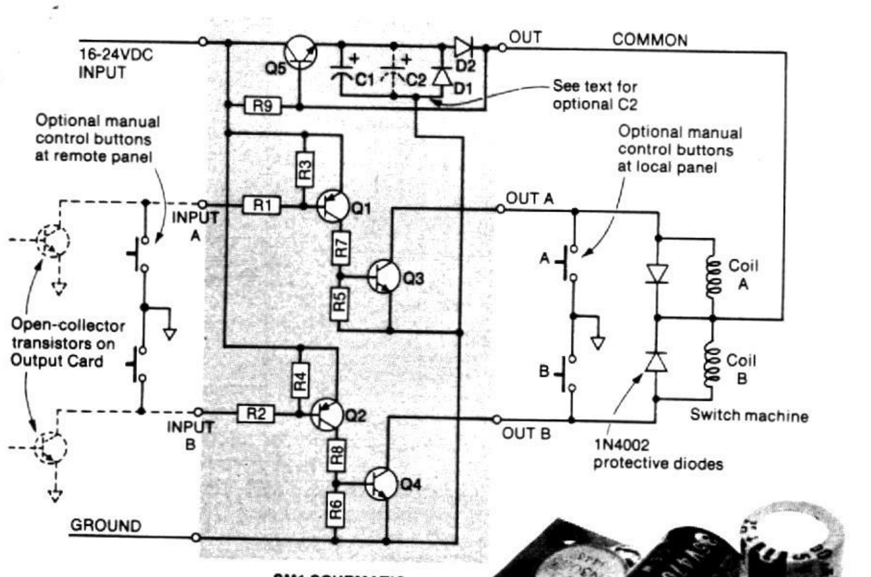
☐ **R1-R9.**

☐ **D1, D2 [+].** Orient with banded ends as shown in fig. 6.

☐ **C1, C2 [+].** C2 is optional, see parts list.

☐ **Q1, Q2 [+].**

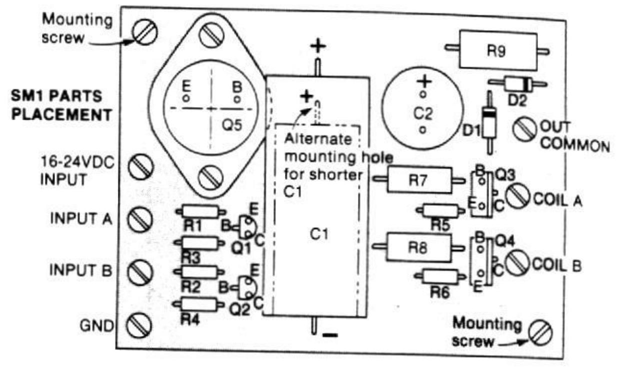
☐ **Q3, Q4 [+].** Bend the center (collector) leads *back*, and orient as shown in fig. 6, with the metal tabs toward the terminal screws. Push in until the wide parts of the leads touch the card, solder, and trim.

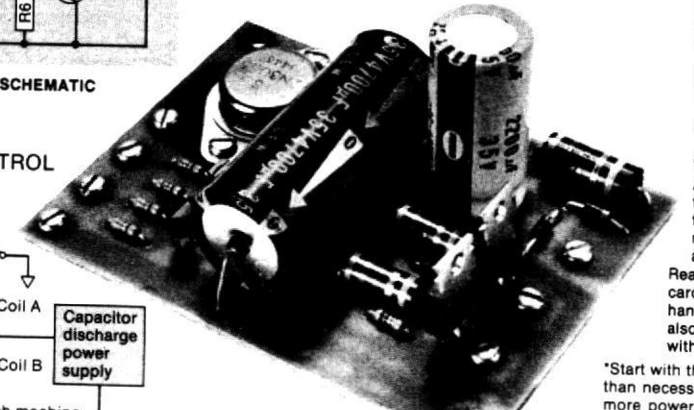


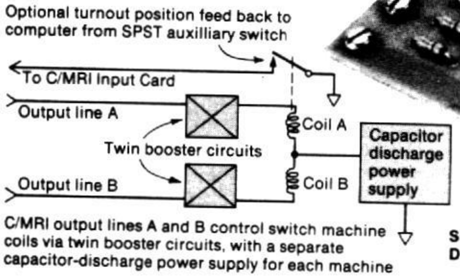
SM1 SCHEMATIC

Fig. 6

SM1, DUAL-OUTPUT SWITCH MACHINE CONTROL






Optional turnout position feed back to computer from SPST auxilliary switch
To C/MRI Input Card
Output line A
Output line B
Twin booster circuits
Coil A
Coil B
Capacitor discharge power supply

C/MRI output lines A and B control switch machine coils via twin booster circuits, with a separate capacitor-discharge power supply for each machine

SM1 BLOCK DIAGRAM

SM1 PARTS LIST

Symbol	Description
R1-R4	4700Ω resistors [yellow-violet-red]
R5-R6	1000Ω resistors [brown-black-red]
R7, R8	470Ω, 1W resistors [yellow-violet-brown]
R9	470Ω, 2W resistor [yellow-violet-brown]
D1, D2	1N4002 diodes
C1	4700μf, 35V electrolytic capacitor (Digi-Key P5068)*
C2	Optional*
Q1, Q2	2N2907 PNP small signal transistors (Jameco PN2907)
Q3, Q4	2N6387 NPN power Darlingtons (Mouser 597-2N6387)
Q5	2N3055 NPN power transistor (Jameco)

Author's recommendations for suppliers given in parentheses above, with part numbers where applicable. See fig. 4 for Mouser Electronics address. Equivalent parts may be substituted. Resistors are ¼W, 5 percent except as specified, and color codes are given in brackets. Ready-to-use circuit cards available from JLC Enterprises; card SM1, $6.00 each plus $2.50 per order shipping and handling. Michigan residents add 4 percent sales tax. Card also requires 9 4-40 x ¼"-long panhead machine screws with brass nuts for mounting Q5 and for use as terminals

*Start with the 4700μf C1. If that throws your switch machine harder than necessary, substitute a smaller capacitor as listed below. For more power, add C2. Try the 1000μf value first, and the 2200μf if that's still not enough power

C1 2200μf, 35VDC Digi-Key P5066 C2 1000μf, 35VDC Digi-Key P6058
C1 3300μf, 35VDC Digi-Key P5067 C2 2200μf, 35VDC Digi-Key P6059

☐ **Q5 [+].** Orient as shown in fig. 6, and secure with 4-40 screws and brass nuts. Solder and trim the two leads, then solder the nuts to the traces.

☐ **Cleanup and inspection.**

Circuit SM2, fig. 7, uses just one output line to control the two coils, taking advantage of the fact that a switch machine (like the turnout it operates) is a binary device: its two states are its normal and reverse positions. I've labled the single input to the logic driver X* (active low), so we can use X* = 0 for normal position and X* = 1 for reverse.

Outputs from pins 3 and 8 of the 74LS00 NAND gates drive the coils *via* booster circuits. The 74LS00's output is low only if both inputs are high. One of the inputs for each gate comes from the 555 timer IC, which when triggered puts out a high pulse for about .25 second. The other two gate inputs are the X* input and its inversion, or X. Thus a low input energizes one coil, a high input the other, and in either case only during the .25 second of the 555's pulse output.

The 74LS86 exclusive-OR IC, along with R1 and C1, triggers the 555, which requires a negative-going, narrow-pulse input. The 74LS86 output goes low only when its two inputs are identical (00 or 11). Since X* and X can't be identical, by definition, it might seem that the 74LS86 would always be a logic high.

However, R1 and C1 cause a small delay in the X path every time X* changes state, so the X signal arrives at the 74LS86 a bit later than the X*. That gives a 1 microsecond logic 0 trigger-pulse input to the 555, which in turn energizes the desired coil for .25 second. To vary the length of the coil pulse, change the value of R2: decreasing R2 shortens the pulse proportionately.

The SM2 isn't as noise-immune as the SM1, but R2 and C7 do help. Usually a noise pulse just causes the machine to try to move the way it's already thrown.

Here are the steps for building the SM2, again on the JLC circuit card:

☐ **Card inspection.**

☐ **Terminal screws.** Five 4-40 screws and nuts.

☐ **J1, J2.**

☐ **R1-R12.**

☐ **C1, C2, C5, and C6.**

☐ **C3, C4, and C7 [+].**

☐ **S1-S3 [+].**

☐ **Q1, Q2 [+].**

☐ **Q3, Q4 [+].**

☐ **U1-U3 [+].**

☐ **Cleanup and inspection.**

However you control it, you can use an auxilliary SPST contact on the switch machine to feed back the turnout position, as shown in figs. 5, 6, and 7. The feedback would reach the computer as an input through a C/MRI Input Card, as I'll explain shortly.

If you use SM1s or SM2s and don't feed back turnout position to the computer, your software must include a power-up initialization procedure that throws each turnout to a known position. Otherwise the turnout position can get out of step with the computer's memory.

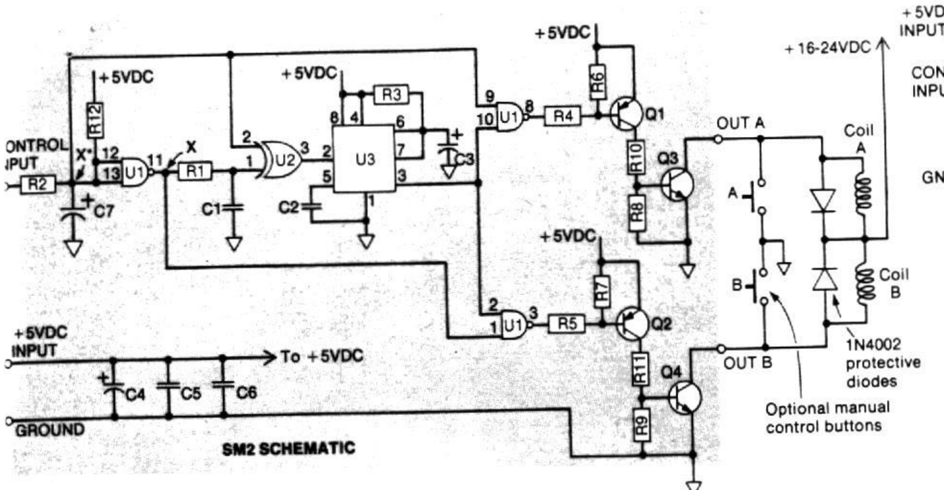
Local, "manual" control of switch machines is easy to add with a switch (fig. 5) or push buttons (figs. 6 and 7) to ground. With the relay controls in fig. 5, make the turnout's normal position the open-circuit, switch-off state with the relay not energized, so the center off position of the toggle can be the normal setting under manual control. Note that with the SM1, fig. 6, you can use both local and remote buttons. Turnout-position feedback is necessary if you use any manual controls.

INPUT FROM THE RAILROAD

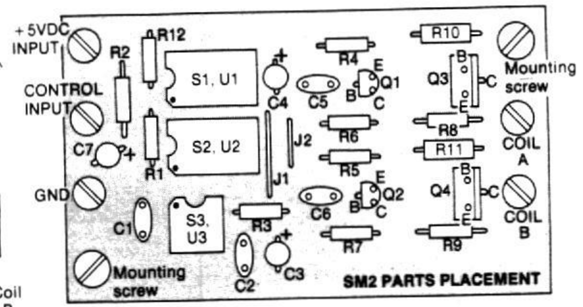
Now we'll look at connections to the Input Cards, to tell your computer which blocks are occupied, which turnouts are reversed, and which way a cab direction switch is set. When the Input Card's 24 input lines are open-circuited, each is held high, at +5V or logic 1, by a built-in 2.2K pull-up resistor. Simply grounding an input line changes it to logic 0.

Figure 8 shows C/MRI input-line connections for various devices. The circuit in fig. 8a lets the computer read the position of an SPST switch. If the switch is open the computer reads +5V or logic 1 through the pull-up resistor. Closing the switch forces the input line to ground with the 5V dropped across the resistor, and the computer reads 0V or logic zero.

Probably the most common application of this is to read the position of a turnout, using an SPST contact, or one side of an SPDT contact, as we just saw. With the turnout in one position the computer will read logic 1, and in the other logic 0.



SM2 SCHEMATIC

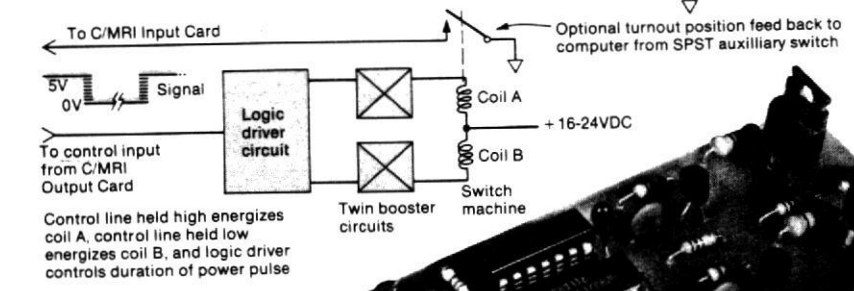


SM2 PARTS PLACEMENT

SM2 PARTS LIST

Symbol	Description
J1-J2	Jumpers, make from no. 24 uninsulated bus wire (Belden no. 8022 or equivalent)
R1, R2	47Ω resistors [yellow-violet-black]
R3	200KΩ resistor [red-black-yellow]
R4, R5	470Ω resistors [yellow-violet-brown]
R6-R9	1000Ω resistors [brown-black-red]
R10, R11	47Ω,½W resistors [yellow-violet-black]
R12	2200Ω resistor [red-red-red]
C1	.022µf, 50V ceramic disk capacitor (Jameco DC.022/50)
C2	.01µf, 50V ceramic disk capacitor (Jameco DC.01/50)
C3, C4	2.2µf, 35V tantalum capacitors (Jameco TM2.2/35)
C5, C6	.1µf, 50V ceramic disk capacitors (Jameco DC.1/50)
C7	1µf, 35V tantalum capacitor (Jameco TM1/35)
S1, S2	14-pin DIP sockets (Digi-Key C9814)
S3	8-pin DIP socket (Digi-Key C9808)
Q1, Q2	2N2907 PNP small signal transistors (Jameco PN2907)
Q3, Q4	2N6387 NPN power Darlingtons (Mouser 597-2N6387)
U1	74LS00 quad two-input NAND gate (Jameco)
U2	74LS86 quad exclusive-OR gate (Jameco)
U3	555 timer (Jameco NE555V)

Author's recommendations for suppliers given in parentheses above, with part numbers where applicable. See fig. 4 for Mouser Electronics address. Equivalent parts may be substituted. Resistors are ¼W, 5 percent except as specified, and color codes are given in brackets. Ready-to-use circuit cards available from JLC Enterprises; card SM2, $6.00 each plus $2.50 per order shipping and handling. Michigan residents add 4 percent sales tax. JLC card also requires 5 4-40 x ¼"-long panhead machine screws with brass nuts for use as terminals


To C/MRI Input Card
Optional turnout position feed back to computer from SPST auxilliary switch
5V
0V
Signal
To control input from C/MRI Output Card
Logic driver circuit
Coil A
+16-24VDC
Coil B
Twin booster circuits
Switch machine
Control line held high energizes coil A, control line held low energizes coil B, and logic driver controls duration of power pulse
SM2 BLOCK DIAGRAM

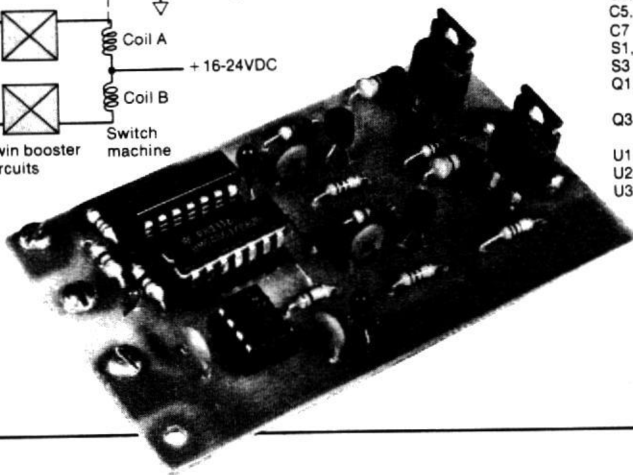


Fig. 7
SM2, SINGLE-OUTPUT SWITCH MACHINE CONTROL

To read a three-position switch, use circuit 8b. If line 1 is logic 0 the switch is in position 1, if line 2 is logic 0 the switch is in position 2, and if neither line is logic 0 the switch is in position 3.

Input lines can also be driven by an open-collector transistor switch as in fig. 8c. If the transistor is off, not conducting, the collector (C) is in effect an open circuit and the computer reads logic 1. When the transistor is on, the collector is tied to ground and the computer reads logic 0.

Figure 8d shows the direct hookup of a standard TTL circuit. The input card already has a built-in pull-up resistor on each line, so either regular TTL chips or those with open-collector outputs can be coupled directly to the Input Card.

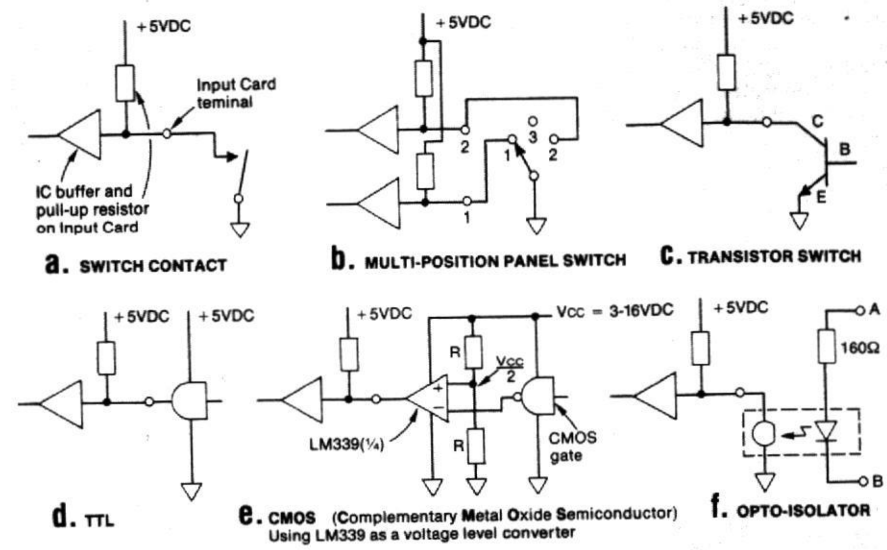
Figure 8e shows how to drive C/MRI input lines with logic voltage levels other than +5VDC and 0. For example, a CMOS gate can operate with anywhere from 3 to 16VDC, with 10V being a frequently-used value. To make the conversion, this circuit uses part of an LM339 quad voltage comparator, a very handy IC to have in your repertoire.

Each of the four comparators in the LM339 has two inputs and an open-collector output. If the voltage applied at the + input is greater than that at the – input, the output is an open circuit. If the voltage at the + input is less than at the –, the output is pulled to ground.

The two resistors (R) are of equal value — 10KΩ works well — and hold the + input voltage constant at the midpoint of the logic switching levels out of the CMOS gate. As an example using a Vcc = 10VDC supply, the + input would be held at 5VDC, which then is the switching level for the voltage applied to the – input terminal.

Figure 8f shows how to connect an opto-isolator. Current through inputs A and B lights the LED, which is "seen" by the photodiode, changing the signal to the Input Card from logic 1 to logic 0.

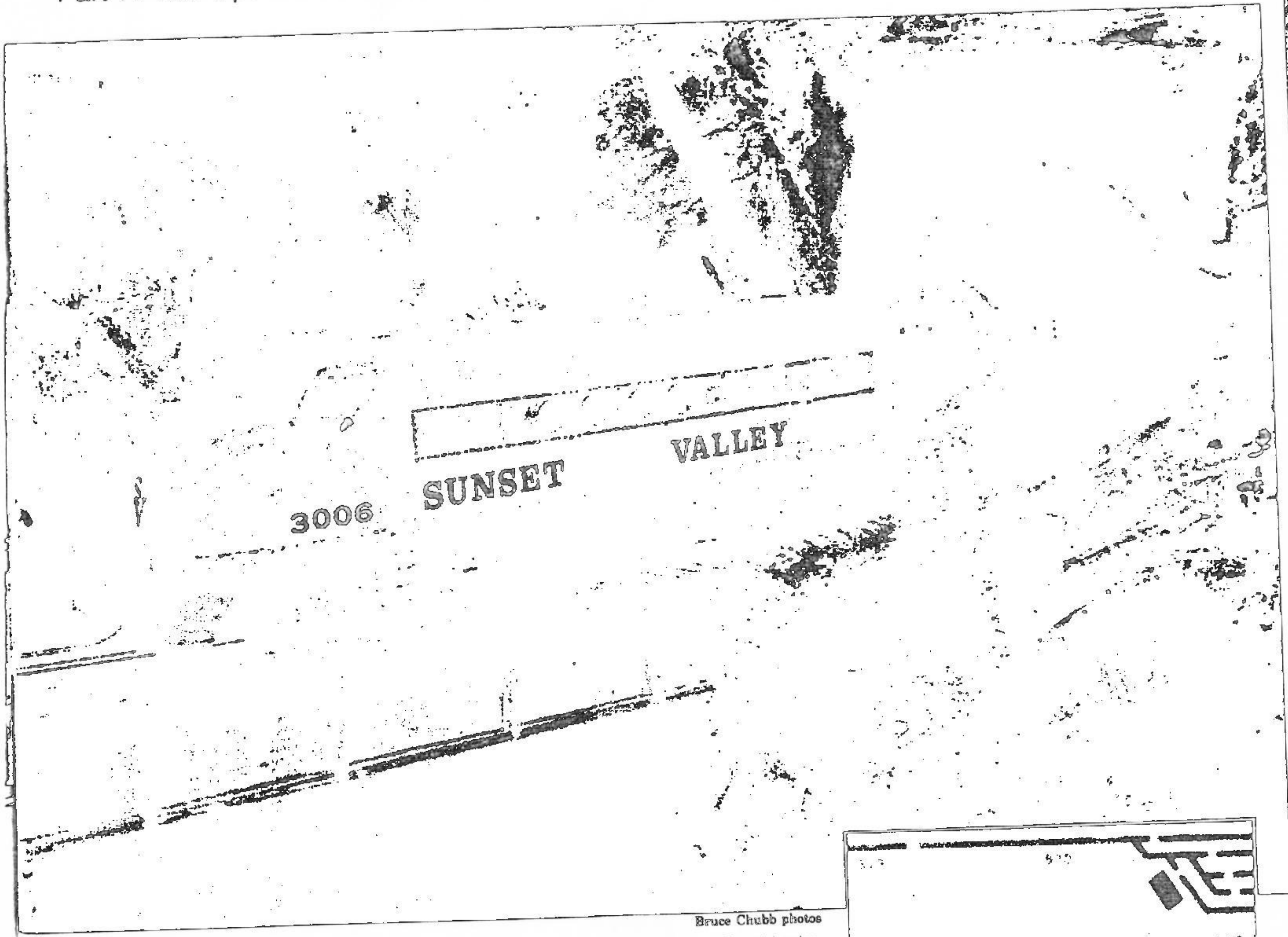
The C/MRI input you'll need most is block-occupancy detection, to let your computer know the whereabouts of every train on your layout. This is the primary input for signaling, automatic assignment of block power, and semi- or completely automatic train control. Next month I'll tell you about the Optimized Detector, which I think is the best model railroad occupancy detector in the world. ♦


+5VDC
Input Card terminal
IC buffer and pull-up resistor on Input Card
a. SWITCH CONTACT
+5VDC
2
1
3
2
1
b. MULTI-POSITION PANEL SWITCH
+5VDC
C
B
E
C. TRANSISTOR SWITCH
+5VDC
+5VDC
d. TTL
+5VDC
Vcc = 3-16VDC
R
Vcc/2
R
LM339(¼)
CMOS gate
e. CMOS (Complementary Metal Oxide Semiconductor) Using LM339 as a voltage level converter
+5VDC
A
160Ω
B
f. OPTO-ISOLATOR
Fig. 8 CONNECTIONS FOR C/MRI INPUT

C/MRI-7

The C/MRI: A computer/model railroad interface

Part 7: The Optimized Detector — it's useful even if you aren't computerizing



Bruce Chubb photos

The panel light shows the dispatcher that the freight led by Sunset Valley no. 3006 is at Mount Louise. In this month's C/MRI part 7, Bruce Chubb explains the Optimized Detector that makes this possible.

BY BRUCE CHUBB

BLOCK-OCCUPANCY input through the C/MRI tells your computer where the trains are on your layout, for computer cab control, signaling, and semi- or completely automatic train control. Many block-occupancy detectors will work with the C/MRI, but I recommend the Optimized Detector shown in fig. 1.

As I said last month, I think this is the best little model railroad block-occupancy detector in the world. My friend Paul Zank, who is both an N scale model railroader and an electronics engineer, helped in its design. It's easy to build and gives super performance. Even if you aren't building the C/MRI, you still might want to use the Optimized Detector on your railroad. Here are a few of its important advantages:

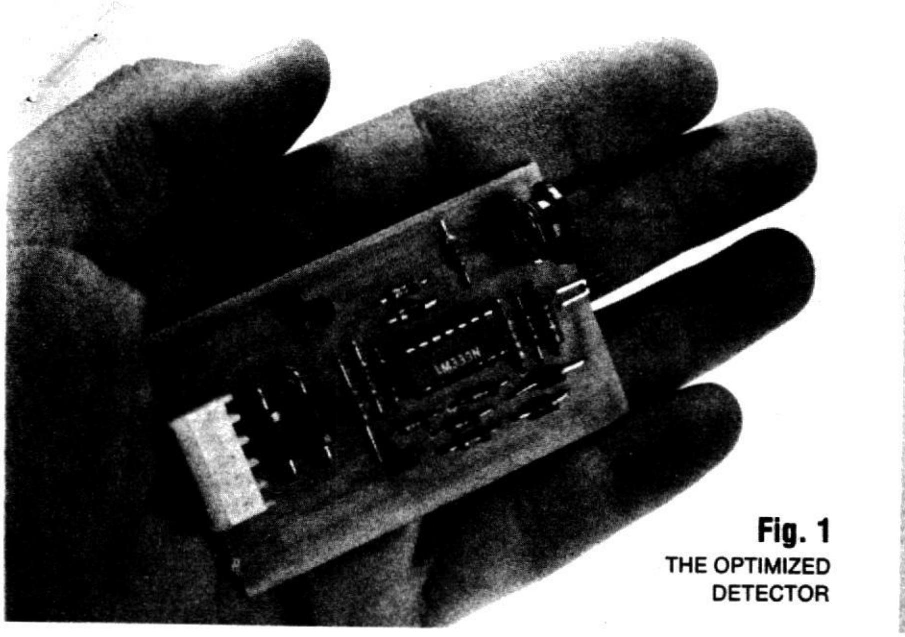


Fig. 1
THE OPTIMIZED DETECTOR

OPTIMIZED DETECTOR PARTS LIST (In order of assembly)

Qnty.	Symbol	Description
1	R1	10Ω resistor [brown-black-black]
3	R3-R5	10KΩ resistors [brown-black-orange]
1	R6	3600Ω, ½W resistor [orange-blue-red]
2	R7, R8	2200Ω resistors [red-red-red]
1	R9	2200Ω, ½W resistor [red-red-red]
1	R10	10KΩ resistor [brown-black-orange]
1	R11	2.2MΩ resistor [red-red-green]
1	R12	220KΩ resistor [red-red-yellow]
1	R13	330KΩ resistor [orange-orange-yellow]
1	R2	10KΩ trim potentiometer (Digi-Key Q0A14)
2	D1, D2	3A, 50-PIV diodes (Digi-Key 1N5400)
3	D3-D5	1A, 50-PIV diodes (Digi-Key 1N4001)
1	S1	14-pin DIP socket (Digi-Key C8914)
1	U1	LM339N quad voltage comparator (Jameco)
1	S2	5-pin connector (Molex P/N 09-52-3051) (Mating connector for Mother Board)
1	—	5-pin connector (Molex P/N 09-64-1051)
1	Q1	2N3904 NPN small signal transistor (Digi-Key)
1	C1	.1µF, 50V ceramic disk capacitor (Jameco DC.1/50)
1	C2	1.5µF, 35V tantalum capacitor (Jameco TM1.5/35)
1	L1	Red, diffused LED (Jameco XC209R)

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

- Its sensitivity is easy to adjust with a trim potentiometer and LED monitor.
- Its built-in turn-on delay of 0.75 second and turn-off delay of 3.5 seconds eliminates problems from dirty track and other causes of intermittent contact.
- Its monitor LED comes on before the time delays, giving an instant occupancy indication to help in setting sensitivity.
- It has only two active components, one IC and one transistor, so it's easy to debug and maintain.
- Its open-collector transistor output allows easy connection to LEDs, TTL logic circuits, relays, and C/MRI Input Cards.
- It works with conventional DC, AC, pulse power, all forms of command control, and any sound system.
- It handles track currents up to 3A and would handle even more if you substituted higher-current diodes.
- It's a small, modular unit (one per block), so it's ideal for plug-in circuit-card construction. That eases system debugging and maintenance, but alternate connection methods are also provided.
- Its price is very reasonable.

If you don't already have occupancy detection on your railroad, I'm sure you'd be happy with this circuit. If you have detection already, you can most likely adapt it for C/MRI input. The output of a Twin-T with booster, for example, can be fed directly into the Input Card. Still, the Optimized Detector beats the Twin-T by 10 to 1 in my book. I've used 54 of them on the Sunset Valley since February 1980 (I described the circuit in MR's March 1982 Symposium on Electronics and updated it in the March 1983 Symposium). Their performance has been exceptional, and I have not had a single failure.

As shown in the schematic and parts list in fig. 2, the Optimized Detector handles up to 3A of steady-state track current and surges up to 50A. If you need more, replace D1 and D2 with higher-current diodes. For example, Digi-Key no. 6A05s give steady-state capacity of 6A and surge capacity up to 400A.

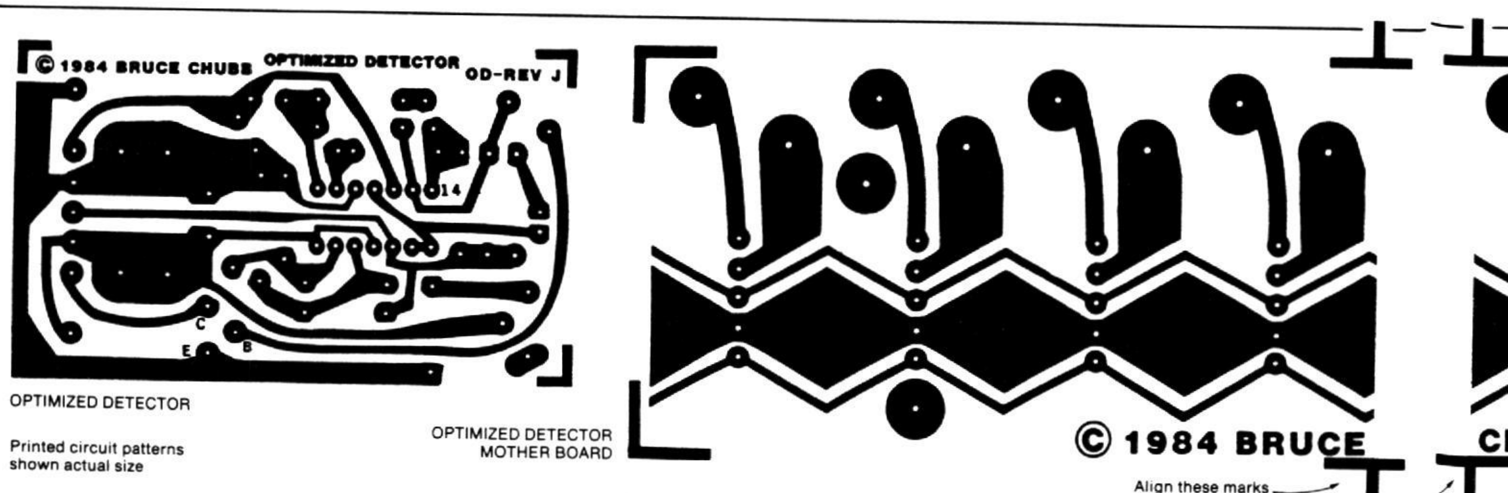
The product of R11 and C2 determines the turn-off delay, and the product of R13 and C2 determines the turn-on delay as long as R13 is considerably smaller than R11. Thus you can easily change the delay times if you wish. I enjoy the rather long 3.5 second turn-off delay, which not only completely solves the problem of intermittent contact, but also simulates the massive, slow-moving relays in prototype detection circuits.

The value of R6 can also be varied if you like. I selected 3.6K for reasonably high drive capability from the output transistor to handle loads as high as .2A and still maintain a good logic low for TTL connections. For example, on the SV I use a single detector to drive parallel loads of 10 LEDs and four TTL logic gates, a total load of about .2A, but still with a logic low under .8V.

Reducing R6 to a lower value, such as 1K, would take more current from the power supply, but would allow driving output loads up to .3A at 40V — the ratings of the 2N3904. For values of R6 at the 3.6K level or lower, you would need to use a ½W resistor.

BUILDING THE OPTIMIZED DETECTOR

If you've already sent for C/MRI PC layouts from MODEL RAILROADER, you'll find circuit art for the Optimized Detector



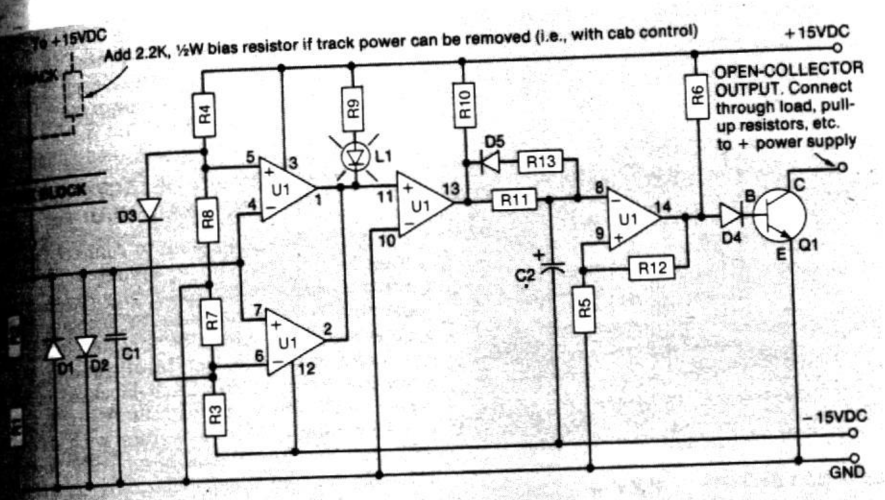
OPTIMIZED DETECTOR

Printed circuit patterns shown actual size

OPTIMIZED DETECTOR MOTHER BOARD

These printed circuit board patterns are copyrighted 1984 by Bruce A. Chubb. They may be locally reproduced as an aid to your own personal construction, but distributing photocopies or any circuit cards made from these diagrams is prohibited

Fig. 3 CIRCUIT-CARD ART



+15VDC
Add 2.2K, ½W bias resistor if track power can be removed (i.e., with cab control)
+15VDC
OPEN-COLLECTOR OUTPUT. Connect through load, pull-up resistors, etc. to + power supply
R4
R9
R10
R6
L1
D5
R13
R11
R12
U1
D3
D4
Q1
B
C
E
C2
R8
R7
R5
R3
D1
D2
C1
-15VDC
GND

Fig. 2 OPTIMIZED DETECTOR SCHEMATIC © 1985 Bruce A. Chubb

and its Mother Board (see below) in that package. For those of you who just want to use Optimized Detectors, without the C/MRI, the full-size circuit art is included here as fig. 3.

Ready-to-use cards are available from JLC Enterprises, P. O. Box 88187, Grand Rapids, MI 49508-9187. The cost of Optimized Detector card OD is $5, Mother Board card ODMB is $18, and there is a shipping and handling charge of $2.50 per order (Michigan residents must add 4 percent sales tax).

Figure 4 gives the parts layout for the Optimized Detector. You need one for each railroad block you want your computer to read. Here's how to assemble it:

☐ **Card test.**

☐ **R1, R3-R13.** Note that R6 and R9 are larger, ½W resistors.

☐ **R2.** Install this trim potentiometer as in fig. 4, pushing the three prongs all the way into the holes as you solder.

☐ **D1, D2 [+].** Use needle-nose pliers to bend the heavy leads of these power diodes at right angles so they drop into the holes. The banded ends must face in opposite directions as in fig. 4. Slip a 1/16" stripwood spacer between the card and the diodes as you solder, then remove the wood. The space helps ventilate the diodes and protects the card.

☐ **D3-D5 [+].**

☐ **S1 [+].**

☐ **U1 [+].**

☐ **S2.** (Optional, for use with Detector Mother Board. Alternate holes, with wider spacing, for hard wiring, are provided for track and common connections.)

☐ **Q1 [+].**

☐ **C1.**

☐ **C2 [+].**

☐ **L1 [+].** Note orientation of flat side and + hole (longer lead) in fig. 4. With a needle-nose pliers, hold the leads securely next to the housing and bend at right angles as in the fig. 4 detail. The LED sticks out over the edge of the card so you can see it when the Detectors are plugged into their Mother Board.

☐ **Cleanup and inspection.**

DETECTOR MOTHER BOARD AND POWER SUPPLY

If you build your Detectors with the S2 connectors, you will also need to build up one or more of the Detector Mother Boards in fig. 5. Each Mother Board accommodates 12 Detectors; if you need more you can parallel each of the 3 bus traces from one board to the next — the SV RR uses 5.

Figure 5 includes the parts list and shows the parts layout for the Mother Board. Here are the assembly steps:

☐ **Card test.**

☐ **Terminal screws.** Install 27 terminal screws with nuts soldered to the circuit traces: +15VDC, COMMON, −15VDC, 12 track connections, and 12 detector outputs.

☐ **S1-S12.**

☐ **Adjacent trace test.** Set your VOM to R x 100 and check the resistance between the COMMON terminal and the +15V and −15V terminals. It should be infinite, an open circuit. Then clip one lead to the +15V terminal and touch the other to each of the track terminal screws for each card slot: each should also read open circuit. Then for each card slot touch one lead to the track terminal and the other to the output terminal. Again, you should read open circuit.

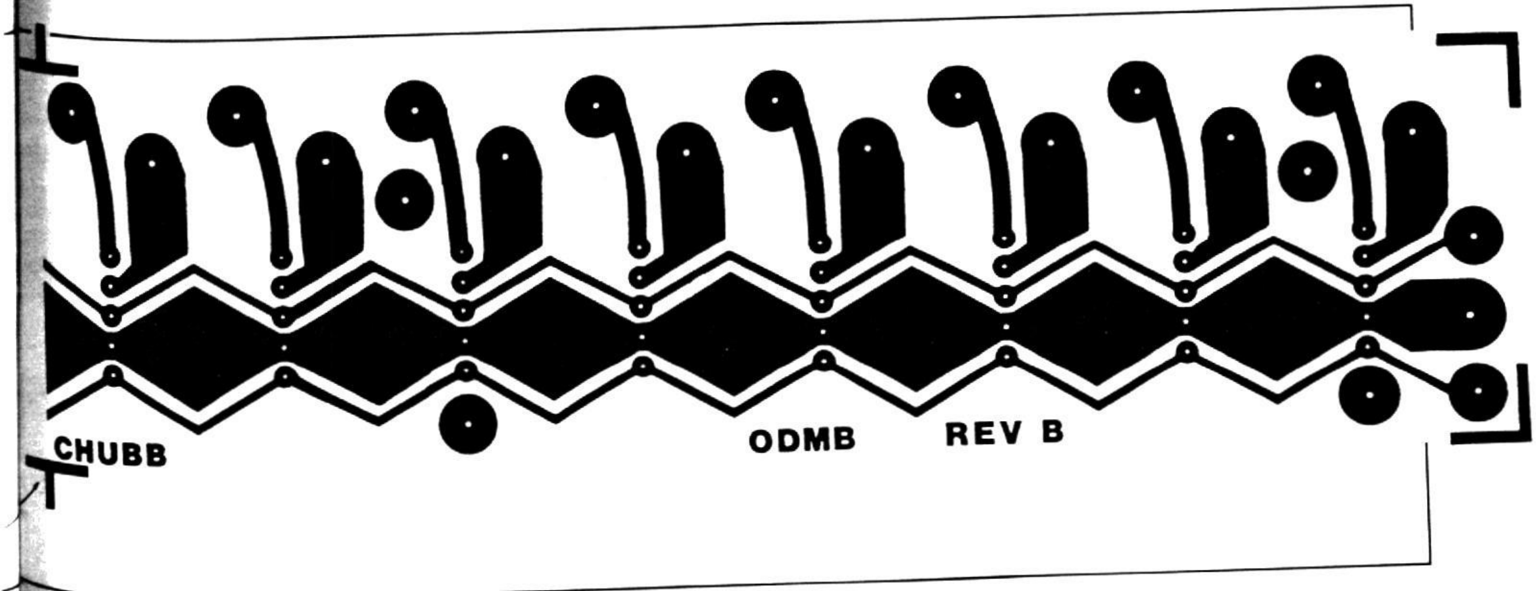
A reading close to 0 for any of these indicates a solder bridge between adjacent pads somewhere along the two bus lines under test. Locate it, remove it, and retest to be sure.

☐ **Cleanup and inspection.**

For mounting, use six ¼"-long standoffs as for the UBEC in part 2. The installation photo in fig. 6 shows how I put my Mother Boards on a wall under my layout and added a wood frame with notches to hold the Detector cards in line. I painted the edge of the notched rail white and used black transfers to apply the railroad block number for each slot.

The Optimized Detectors need a polarized (plus-and-minus, ±) 15VDC power supply. Any voltage between 5 and 15 would work, but the nearer the high end the better, and the + and − values must be the same. Each Detector draws about 10mA (not counting external current on the open-collector output connection), so a 1.5A supply handles up to 150.

Figure 7 shows the schematic for the Sunset Valley's supply, along with the



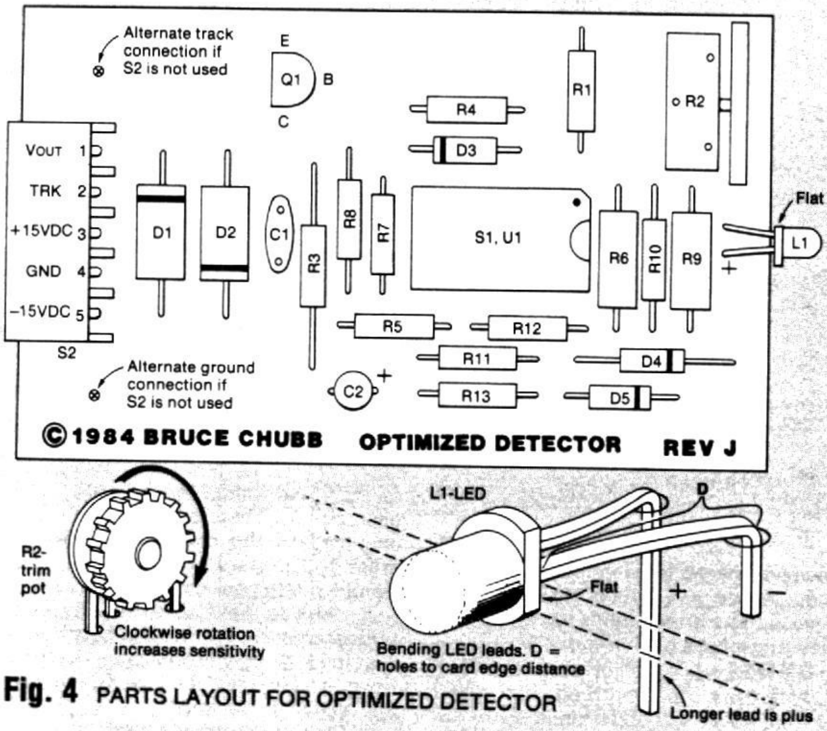
parts list. It's built just like the +5VDC supply, except that regulators V1 and V2 must be insulated from each other. Since the currents are lower the heat sinks could be smaller, but I like the reliabilility of an overdesigned supply.

CONNECTING DETECTORS TO YOUR RAILROAD

Figure 8a shows how to connect Optimized Detectors on a layout with cab control. Each block has a bias resistor to provide a current path to activate the detector if the block is occupied but the selector switch is set to off, and/or the assigned cab is off. Each block so occupied will draw about 10mA from the +15V supply; for a system with many blocks, you could use a separate power supply for the bias resistors.

If you use command control, like the CTC-16 on the SV, track power is always present, so you can eliminate the bias resistors (and, obviously, the selector switches). You still need a DPDT reverser for each reversing block, but with your C/MRI you can use a DPDT relay and have the computer switch the reversing block automatically.

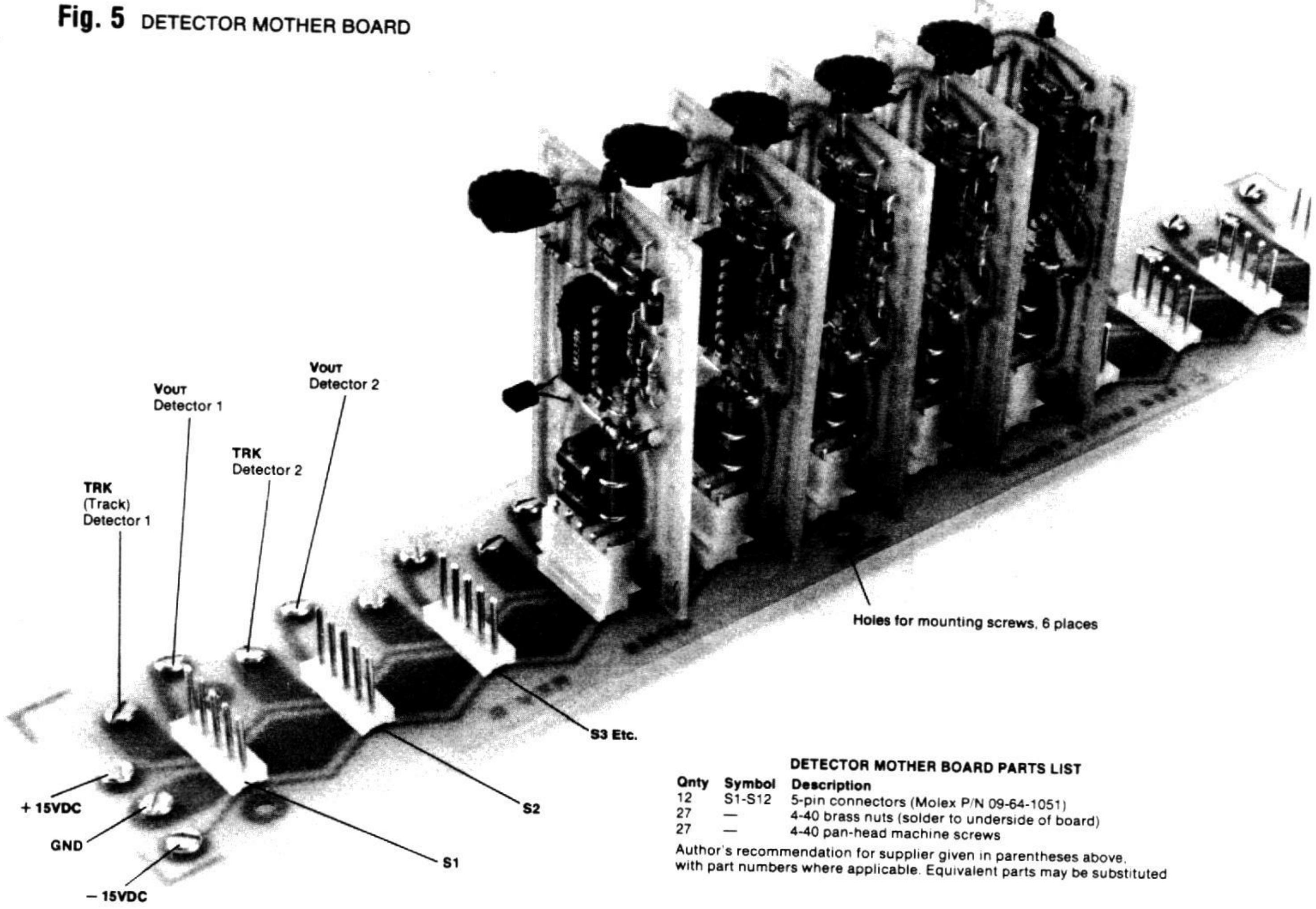
Connecting devices to the Detector output is just like connecting them to a C/MRI Output Card. Figure 8b shows connections for the most common items.



Alternate track connection if S2 is not used
E
Q1
B
C
R4
D3
R1
R2
Vout 1
TRK 2
+15VDC 3
GND 4
−15VDC 5
S2
D1
D2
C1
R3
R8
R7
S1, U1
R6
R10
R9
Flat
L1
+
R5
R12
R11
D4
C2
R13
D5
Alternate ground connection if S2 is not used
©1984 BRUCE CHUBB OPTIMIZED DETECTOR REV J
R2-trim pot
Clockwise rotation increases sensitivity
L1-LED
D
Flat
+
−
Bending LED leads. D = holes to card edge distance
Longer lead is plus

Fig. 4 PARTS LAYOUT FOR OPTIMIZED DETECTOR

Fig. 5 DETECTOR MOTHER BOARD



DETECTOR MOTHER BOARD PARTS LIST

Qnty	Symbol	Description
12	S1-S12	5-pin connectors (Molex P/N 09-64-1051)
27	—	4-40 brass nuts (solder to underside of board)
27	—	4-40 pan-head machine screws

Author's recommendation for supplier given in parentheses above, with part numbers where applicable. Equivalent parts may be substituted

GOOD GROUND WIRING

Any model railroad should have good ground wiring, but when you introduce electronics to a layout — whether command control, solid-state signaling logic, sound, or the C/MRI — good ground wiring is particularly important.

With TTL or CMOS circuits, a voltage shift of only a volt or 2 can change a logic state. Typical IC response times are in the range of a few nanoseconds (1 billionth of a second), while model railroads are full of large current surges. When a train derails or shorts across a gap, or a switch machine throws, it's not unusual to get large-step current changes of 5 to 10 amps or more.

A 10A current in a 50-foot ground line of no. 18 copper wire (.64Ω per 100 feet) will cause an end-to-end voltage differential of 3.2V — enough to give any TTL circuit fits. Throw in a few resistive solder joints and some dirty relay or switch contacts, and the loss can be big even for short lengths of wire. Smaller wire makes the problem even worse.

You don't have to understand the theory of voltage loss, or worry about it either, as long as you follow three basic practices for good ground wiring.

1. Use heavy wire for ground lines. You may get by with no. 18 wire on a very small layout, but no. 16 is better. For larger systems the investment in no. 14 or 12 would be worthwhile, with no. 10 best for the main feeder grounds on very large systems. (Note that in American wire gauge or AWG sizes, lower numbers mean bigger wire.)

2. Use separate ground wires for high-current circuits (like track feeders and switch machines) and for logic and computer circuits. In electronics these two types of grounds are often called "power ground" and "signal ground."

3. Tie all of your various ground lines together at only one central point, as close as possible to your power supplies.

These practices should eliminate significant voltage shifts in signal grounds and also reduce the noise level at IC inputs. Figure 9 shows an example of good ground wiring for a model railroad.

Good ground wiring practices go hand-in-hand with the good signal line practices we've been following in the C/MRI series, always using either an open-collector transistor output (as in the Output Card and Optimized Detector) or a special driver IC (like the 74LS244 in the UBEC) when driving signals long distances.

By the way, if your I/O Mother Board is more than 12 feet from your UBEC, and/or if you use more than 40 I/O Cards, it would be best to use 74S244s for U1-U4 instead of 74LS244s. The 74S244s have nearly three times the drive capability, although with them your UBEC will draw about .6A. With some computers that may require a separate power supply for the UBEC, as described in Part 3 for the TRS-80 Model 3.

Large (48,000μF) electrolytic capacitors liberally distributed throughout your wiring system will also enhance its immunity to electrical noise, and these are shown in fig. 9 too. However, while a large electrolytic capacitor is an excellent ripple filter for a power supply, it can't provide much high-frequency bypassing. In fact, at high frequencies an electrolytic capacitor usually acts more like a noise source than a noise filter.

The large electrolytics need 1.5μF tantalum and .1μF ceramic-disk capacitors parallel with them for good noise suppression. The big capacitors can be almost anywhere, but the small ones should be as close as possible to potential sources of high-frequency noise, such as the IC chips themselves and motor commutators.

TTL and CMOS circuits make a lot of noise on their own. When their outputs change state at nanosecond speeds, ICs draw heavy current spikes from their supply lines. These must be kept from propagating through the supply system, "ringing" the lines with oscillating transients and interferring with other circuits.

Local "despiking" capacitors momentarily provide energy to stabilize supply lines during output state transitions. Use small ceramic-disk capacitors in the .01 to .1μF range, with leads cut as short as possible. They should be distributed around any TTL or CMOS ICs, with at least one for every four 14- to 16-pin DIP packages, and a separate capacitor for each larger MSI (*M*edium-*S*cale *I*ntegration) package like the 40-pin 8255. It's also good practice to include a 2 to 10μF tantalum capacitor on each circuit card where the +5VDC supply line enters the card.

These smaller capacitors are already included in our C/MRI circuits. For example, the UBEC has a .1μF despiking capacitor for each IC! For reliability, use

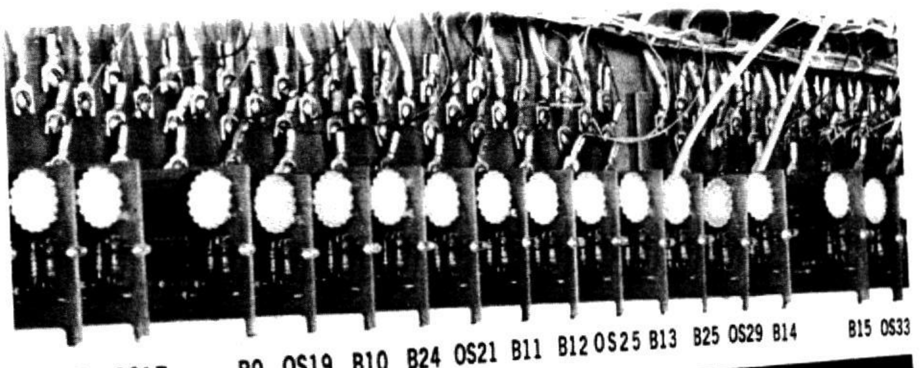
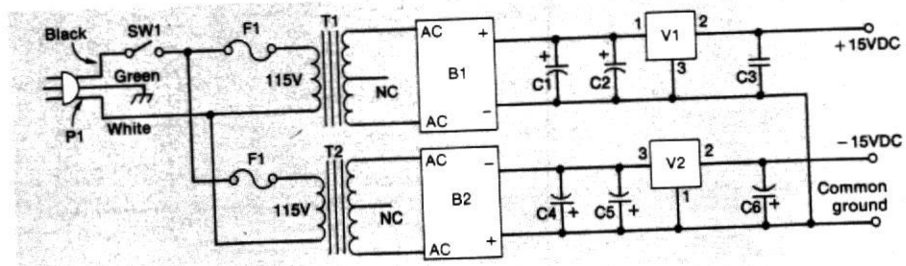
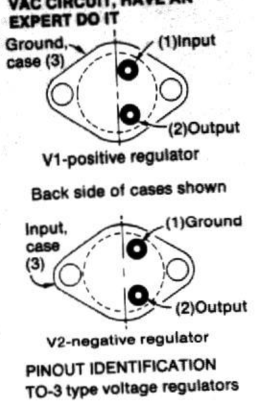


Fig. 6 OPTIMIZED DETECTORS UNDER THE SUNSET VALLEY



WARNING: 120VAC SHOCK CAN BE FATAL. IF YOU DON'T KNOW HOW TO WIRE THE 120 VAC CIRCUIT, HAVE AN EXPERT DO IT



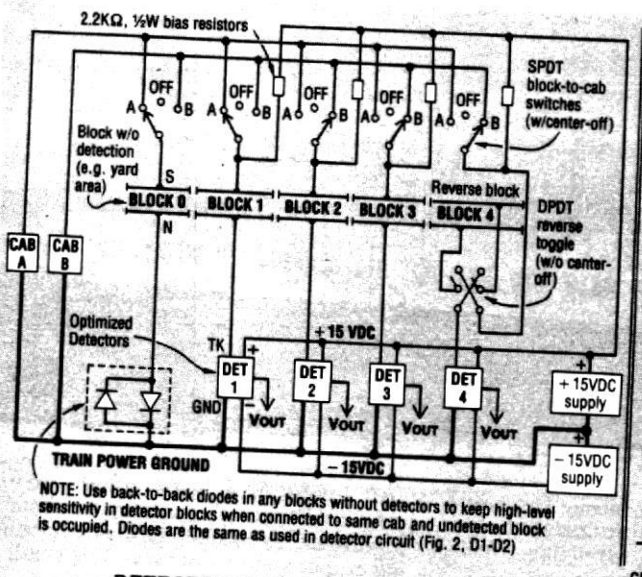
PINOUT IDENTIFICATION
TO-3 type voltage regulators

Fig. 7
±15VDC DETECTOR POWER SUPPLY

±15VDC POWER SUPPLY PARTS LIST (In order of assembly)

Qnty	Symbol	Description
1	P1	Three-conductor power cord w/plug (Jameco 17236)
1	—	Strain relief (Radio Shack 278-1636)
2	F1	Panel-mount fuse holder (Radio Shack 270-365)
2	—	2A fuse (Radio Shack 270-1275)
1	SW1	Toggle switch (Radio Shack 275-602)
2	T1, T2	18V, 2A power transformer (Radio-Shack 273-1515)
2	B1, B2	6A, 50-PIV bridge rectifier (Radio Shack 276-1180)
2	C1, C4	10,000μF, 25V electrolytic capacitors (Digi-Key P6509)
1	C2	.22μF, 35V tantalum capacitor (Jameco TM.22/35)
1	C3	.1μF, 50V ceramic disk capacitor (Jameco DC.1/50)
1	C5	2.2μF, 35V tantalum capacitor (Jameco TM2.2/35)
1	C6	1μF, 35V tantalum capacitor (Jameco TM1/35)
1	V1	Positive 15V, 1.5A fixed voltage regulator (Fairchild μA7815KC or National LM7815KC or Unitrode UC7815KC)
1	V2	Negative 15V, 1.5A fixed voltage regulator (Fairchild μA7915KC or National LM7915KC or Unitrode UC7915KC)
2	—	TO-3 socket (Radio Shack 276-029)
2	—	TO-3 mounting hardware (Radio Shack 276-1371)
2	H1, H2	Fin-type heatsinks for V1 and V2 with TO-3 case (Jameco DUDE-4 or equivalent)

Author's recommendations for suppliers given in parentheses above, with part numbers where applicable. Equivalent parts may be substituted



2.2KΩ, ½W bias resistors
OFF
A
B
SPDT block-to-cab switches (w/center-off)
Block w/o detection (e.g. yard area)
S
N
BLOCK 0
BLOCK 1
BLOCK 2
BLOCK 3
BLOCK 4
Reverse block
DPDT reverse toggle (w/o center-off)
CAB A
CAB B
Optimized Detectors
TK
GND
+15 VDC
DET 1
DET 2
DET 3
DET 4
VOUT
+15VDC supply
−15VDC supply
−15VDC
TRAIN POWER GROUND
NOTE: Use back-to-back diodes in any blocks without detectors to keep high-level sensitivity in detector blocks when connected to same cab and undetected block is occupied. Diodes are the same as used in detector circuit (Fig. 2, D1-D2)

Fig. 8a DETECTOR WIRING WITH CONVENTIONAL CAB CONTROL

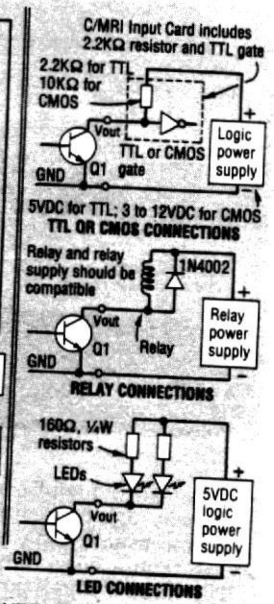


Fig. 8b DETECTOR OUTPUT CONNECTIONS

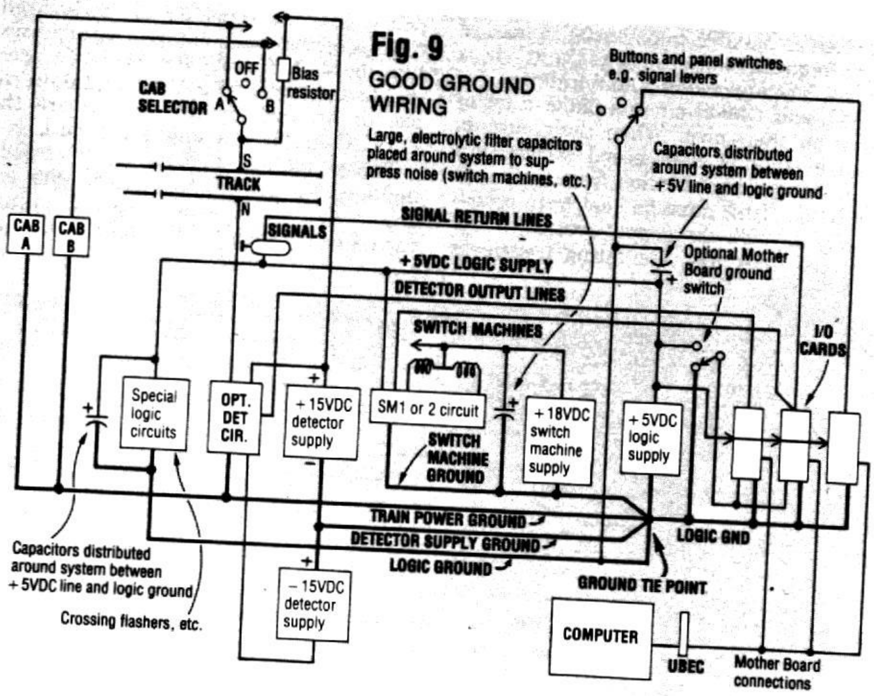
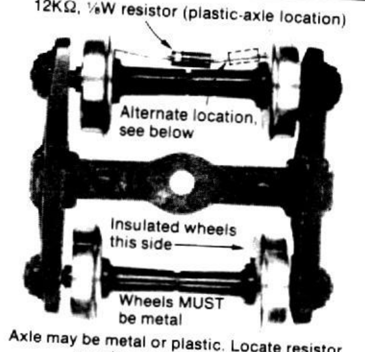


Fig. 9 GOOD GROUND WIRING



12KΩ, ⅛W resistor (plastic-axle location)
Alternate location, see below
Insulated wheels this side
Wheels MUST be metal
Axle may be metal or plastic. Locate resistor near insulated wheel on metal axles to avoid short from lead to axle

Fig. 10 DETECTING UNLIGHTED CARS

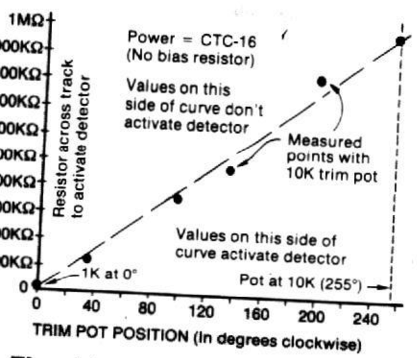


Fig. 11 LINEAR CONTROL OF OPTIMIZED DETECTOR SENSITIVITY

these same solid design practices in any circuits you add to your C/MRI.

CAR DETECTION

When you get your Optimized Detectors installed, turn sensitivity-adjustment pot R2 on each one up as high (clockwise) as it will go. If the LED lights to show the block occupied when it is not, back off the pot until the LED goes out.

Powered locomotives or lighted cars entering a block should activate the Detector and turn on the LED. To protect unlighted cars, you need a resistive path across at least one wheelset on each car. There are two ways (at least) to do this: silver-based resistance paint, or a small resistor across a pair of metal wheels.

Peter Thorne describes the resistance- and conductive-paint method in his Kalmbach book, *Practical Electronic Projects for Model Railroaders.* It can be used with plastic wheels and/or axles, as well as metal ones. Peter recommends a resistance of about 5KΩ. If too low, the resistance paint can get hot enough to damage plastic at full track voltage. He also points out that conductive paint eventually wears off plastic wheels.

I prefer to use a 12KΩ, ⅛W resistor on one wheelset of each unlighted car, as in fig. 10 — the wheels must be metal, of course. The resistor has a definite value, and wear can't cause any loss of detection. At 12KΩ per car and about 350 such cars on the SV, the total current drain on my whole system is still less than ½A. You may use less resistance (Thorne's 5KΩ) with less-sensitive detectors.

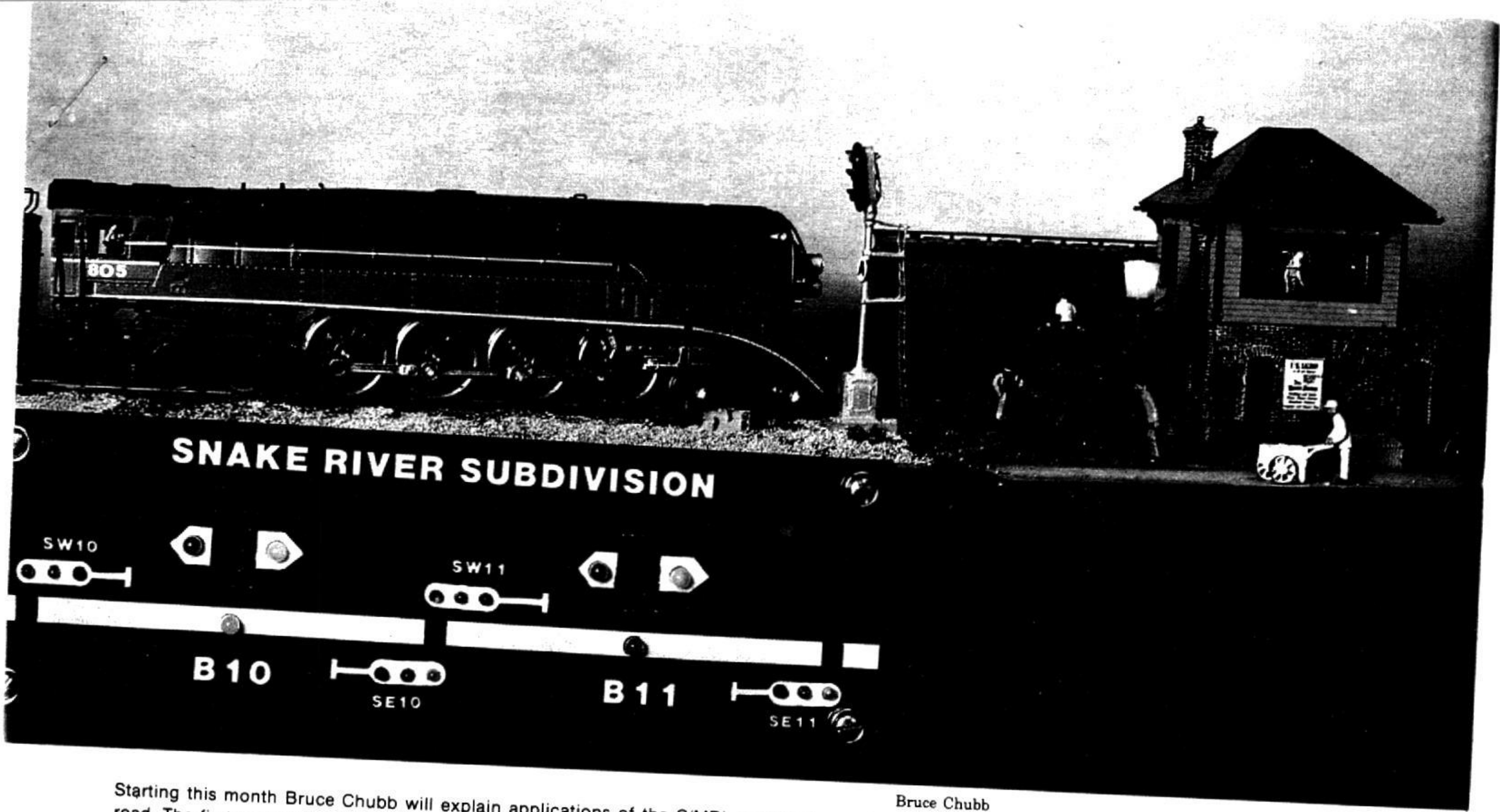
I drill a no. 78 (.016") hole in each wheel, poke the lead wires through the holes, cut them off, and then bend the ends over against the wheel faces to secure the resistor. The leads fit tightly in the holes and make good contact.

I adjust the Optimized Detectors for single-car protection in each block by clipping a 120KΩ resistor across the track and setting the sensitivity pot so that the LED just lights. That way the 12KΩ car resistor activates the detector with a 10-to-1 margin of safety!

Humidity sometimes caused false occupancy indications with my old Twin-Ts, but if that happens with the Optimized Detector you can simply reset the sensitivity control to a lower level. Once a visitor spilled Coke on the track during an SV operating session. The occupancy light came on, but I just turned down the pot until the monitor LED went out, removing the false indication. After cleaning up the spill, we ran a train into the block to see if it would be detected, and it was!

Figure 11 demonstrates the sensitivity range of the optimized detector as a linear function of the potentiometer position. With the sensitivity pot set to maximum (fully clockwise) the detector triggers with 1MΩ or less across the track. Set to minimum sensitivity you need 1KΩ across the track before the Detector activates, so the 10KΩ pot gives you a 1000-to-1 linear range in sensitivity adjustment.

We'll skip next month, but in October I'll be back to show you how to put the Detectors and C/MRI to work in a computer cab control system, with computerized signaling to follow.



Bruce Chubb

Starting this month Bruce Chubb will explain applications of the C/MRI, or how to use it on your railroad. The first application is Computer Cab Control (CCC) — the computer switches block power automatically and operates trackside signals, so you can have more fun running the trains. The panel in the photo displays the cab assignment, cab direction, and track occupancy of two computer-controlled blocks.

C/MRI-8

The C/MRI: A computer/model railroad interface

Part 8: Computer Cab Control — hardware

BY BRUCE CHUBB

WE ALL try to achieve realism on our model railroads, and for many of us that includes operating the trains as parts of a prototypical railroad system. It's just as important, then, to follow prototype practice in the way we run our models as in the way we make them look. The C/MRI can make running our trains more realistic, and easier too.

Cab control in its typical manual forms — the conventional way of running two or more trains simultaneously — imposes unrealistic requirements on our operations. Switching cab power from block to block doesn't correspond to any procedure on a real railroad.

Obviously, real engineers don't need block switches to control their locomotives. As a model railroad engineer, however, you can spend as much time throwing toggles as actually running your train, especially on a small layout. Many of us just get used to this, but only at some cost in realism and fun.

Command control is one alternative to manual cab control. As most of you probably know, with command control a receiver in each locomotive picks up coded signals sent through the rails, so each train can run independently on a railroad wired as if it were one big electrical block. It's very prototypical in that you never have to worry about running into a block controlled by another cab and losing control of your train.

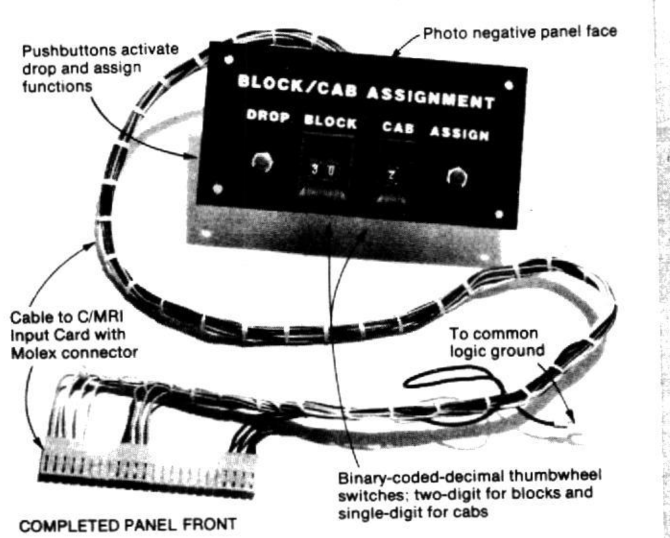
I use command control myself ("Command control comes to the Sunset Valley," January 1983 MODEL RAILROADER), but I realize that this isn't the best choice for everyone, for any number of reasons. Fortunately, command control isn't the only alternative to manual cab control. The C/MRI makes automated computer cab control practical.

COMPUTER CAB CONTROL

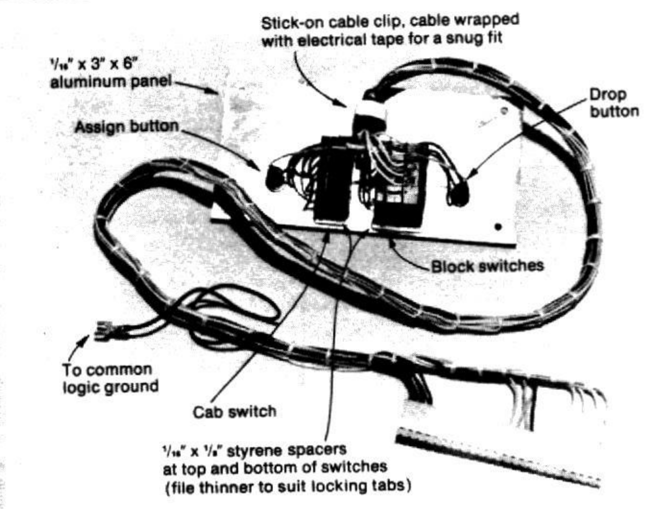
Automatic cab control systems have been around at least since the 1950s, but you haven't seen them on a lot of model railroads. Until recently they have been awkward and inflexible in implementation, and limited in their capabilities. Now that we can use the C/MRI to apply the flexibility and logic capacity of a home computer to this task, Computer Cab Control (CCC) will be easier to use than any earlier automated system.

How does Computer Cab Control work? Simply and almost entirely automatically. You'll do a bit of manual setup at the start of a run to tell the computer to start your cab from the block where your train is waiting. Then CCC automatically connects (assigns) blocks to your cab one step ahead of your train and disconnects blocks behind as your last car clears. Clear blocks can be reassigned to other cabs automatically.

CCC-driven trackside signals will show when your train can enter the next block. When you stop in a siding for a meet and set your cab direction switch in its neutral/off position, only the block(s) your train occupies stays assigned to your cab. If you reverse direction, the next block in the other direction will be assigned to your cab, that is, if it isn't occupied or already assigned to another cab. CCC follows turnout settings to assign blocks on branching routes. At the end of the run a little more manual input tells the CCC computer to disconnect your cab from the block where your train tied up, so you can use the same cab to run another train.


Pushbuttons activate drop and assign functions
Photo negative panel face
BLOCK/CAB ASSIGNMENT
DROP BLOCK CAB ASSIGN
Cable to C/MRI Input Card with Molex connector
To common logic ground
Binary-coded-decimal thumbwheel switches; two-digit for blocks and single-digit for cabs

COMPLETED PANEL FRONT

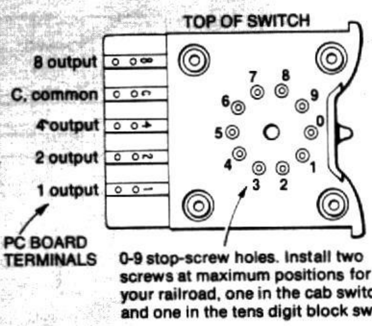

Stick-on cable clip, cable wrapped with electrical tape for a snug fit
1/16" x 3" x 6" aluminum panel
Assign button
Drop button
Block switches
To common logic ground
Cab switch
1/16" x 1/8" styrene spacers at top and bottom of switches (file thinner to suit locking tabs)

COMPLETED PANEL REAR

BLOCK/CAB ASSIGNMENT PANEL PARTS LIST

Qnty.	Description
3	Unimax 10-position BCD thumbwheel switch (Jameco SF-21)
2	Unimax thumbwheel switch end plates-pair (Jameco SF-EP)
2	Stop-screws for thumbwheel switch (0-80 x 1/4" flathead screws)
1	Stick-on cable clip (Radio-Shack 278-1640)
2	Normally-open SPST pushbutton (Jameco MS102)

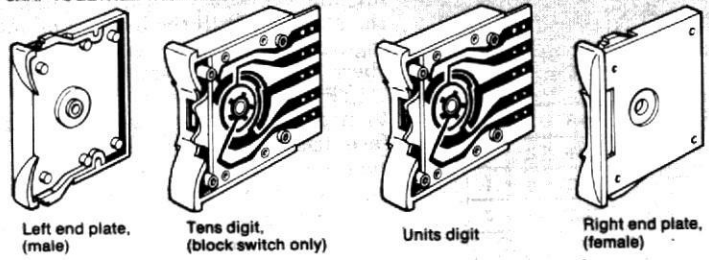
Author's recommendations for suppliers given in parentheses above, with part numbers where applicable. Equivalent parts may be substituted

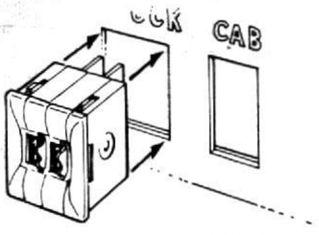

TOP OF SWITCH
8 output
C, common
4 output
2 output
1 output
PC BOARD TERMINALS
7 8 9 0 1 2 3 4 5 6
0-9 stop-screw holes. Install two screws at maximum positions for your railroad, one in the cab switch and one in the tens digit block switch

LOGIC TABLE
TEN-POSITION BCD SWITCH

Dial no. ↓	Common, C connection to terminal 8	4	2	1
0				
1				●
2			●	
3			●	●
4		●		
5		●		●
6		●	●	
7		●	●	●
8	●			
9	●			●

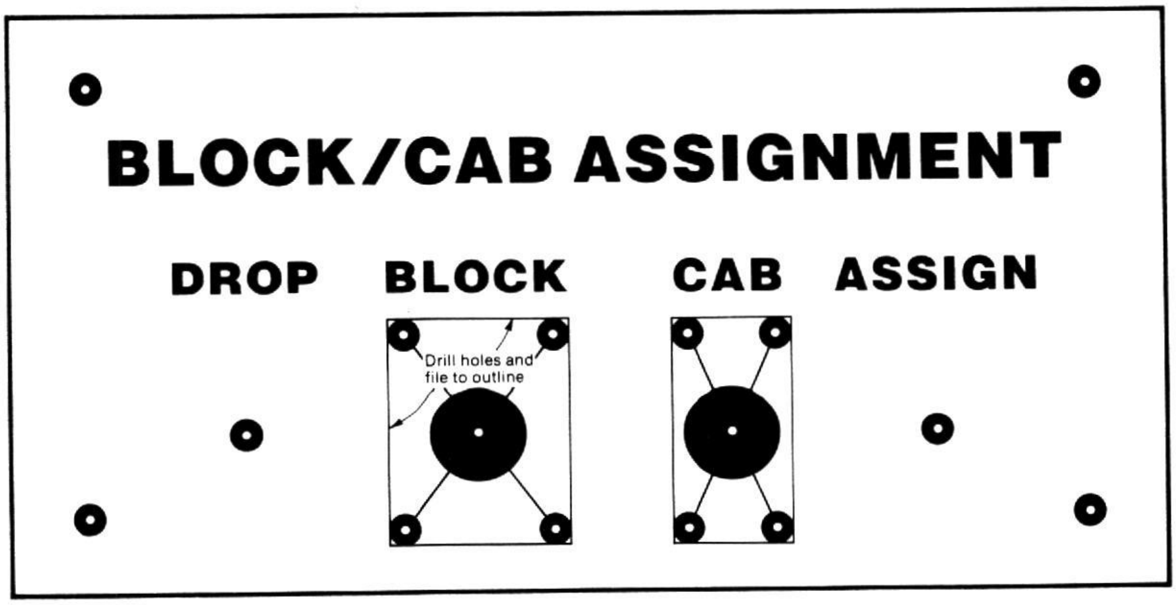
SNAP TOGETHER THUMBWHEEL SWITCHES



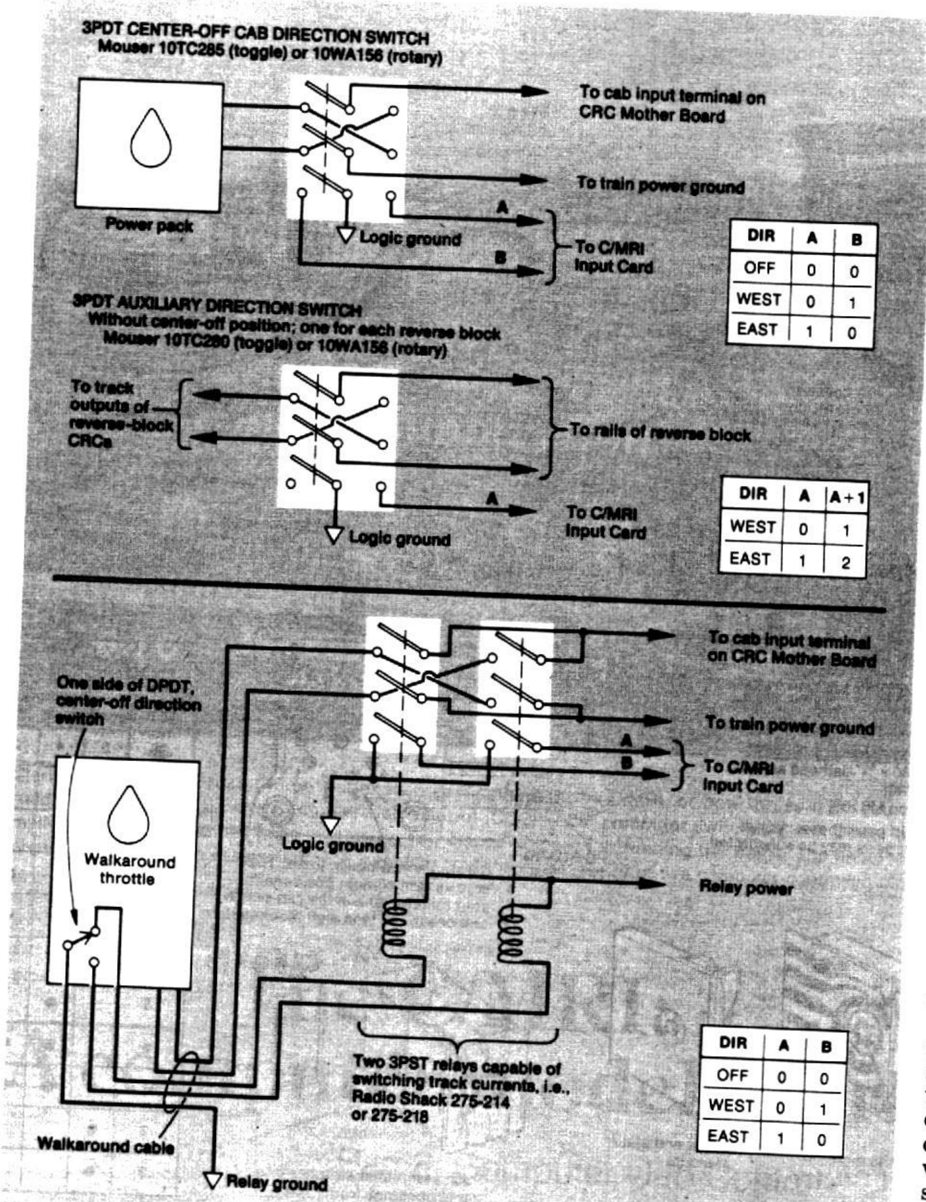


Push assembled switches into panel openings from front

Fig. 1 BLOCK/CAB ASSIGNMENT PANEL



Actual size panel face


3PDT CENTER-OFF CAB DIRECTION SWITCH
Mouser 10TC285 (toggle) or 10WA156 (rotary)
To cab input terminal on CRC Mother Board
To train power ground
Power pack
Logic ground
A
B
To C/MRI Input Card
3PDT AUXILIARY DIRECTION SWITCH
Without center-off position; one for each reverse block
Mouser 10TC280 (toggle) or 10WA156 (rotary)
To track outputs of reverse-block CRCs
To rails of reverse block
To C/MRI Input Card
One side of DPDT, center-off direction switch
To cab input terminal on CRC Mother Board
To train power ground
To C/MRI Input Card
Walkaround throttle
Logic ground
Relay power
Two 3PST relays capable of switching track currents, i.e., Radio Shack 275-214 or 275-218
Walkaround cable
Relay ground

DIR	A	B
OFF	0	0
WEST	0	1
EAST	1	0

DIR	A	A+1
WEST	0	1
EAST	1	2

DIR	A	B
OFF	0	0
WEST	0	1
EAST	1	0

Fig. 2 DIRECTION SWITCH INPUT FOR CCC

While CCC handles cab-control functions automatically, YOU run the train. Free to concentrate on operating realistically, or just to watch the train roll through the scenery, you'll have more fun being the engineer with the computer doing the donkey work.

In addition to the C/MRI itself you'll need some special hardware for CCC. Most basically, you'll need the block switching circuitry, to assign one of several cabs to each block by computer command. You'll want a display of the status of each block, to show whether or not it is occupied and which cab, if any, is assigned. I'll show you how to do this with a separate LED cab number display digit for each block on a track-diagram panel, and/or using the computer's CRT.

You'll also need one or more control stations for initial cab assignments and reassignments. Let's build this Block/Cab Assignment Panel (B/CAP) first, since it's used to initialize the CCC system.

BLOCK/CAB ASSIGNMENT PANEL

I've already described the B/CAP in use, assigning a block to a cab at the start of a run and canceling the assignment at the end. You could do this from the computer keyboard, but mixing keyboard input into a tight-loop, real-time program like we'll need for CCC slows down the processing. It can also require "interrupt" processing, complicating the program. Besides, engineers can be anywhere around your railroad when they need to make or change assignments, and the keyboard may not be handy.

Figure 1 shows the B/CAP. It uses three binary-coded decimal (BCD) thumbwheel switches: two to identify the block and one to identify the cab. Two normally-open pushbuttons activate the assignment and drop functions.

To assign a block to a cab, you dial the block and cab numbers you want, then press the ASSIGN button. To cancel cab assignment to a block, dial the block number and press the DROP button. The cab setting doesn't matter in the drop case: any cab assigned to the block you dial will be dropped.

The output of BCD switches can go directly into the C/MRI, and thumbwheel types let you enter more block numbers than you could with readily available rotary switches. Two thumbwheel switches can select up to 100 blocks (0 through 99), and three up to 1000. The single switch for cab input handles to 9 cabs, with 0 used for no cab, or off. If you need more, a hexadecimal switch can handle up to 15 cabs, designated 1 through 9 and A through F.

You may use as many B/CAPs as you need so they are convenient wherever trains start and end their runs on your railroad. Assemble each one as follows:

☐ **Thumbwheel switches.** Place stop screws at the maximum block- and cab-number positions for your railroad. Snap together the two block number switches, with one pair of end plates, and the cab number switch, also with a pair of end plates, as shown in fig. 1.

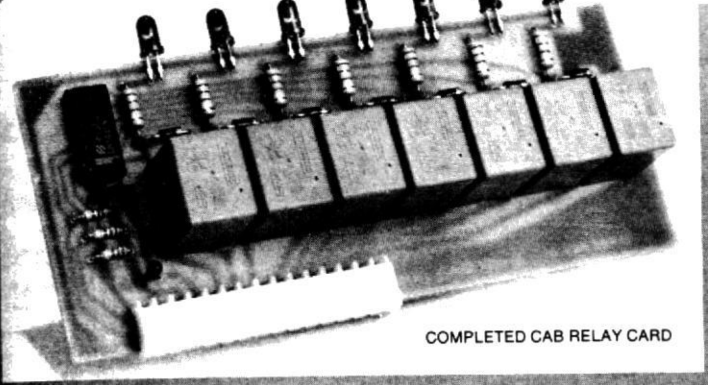
☐ **Panel plate.** Lay the printed panel face from fig. 1 over the aluminum and prick-punch through the paper to mark the metal for drilling. Drill the eight corner holes no. 7, being careful to keep the drill inside the outline of the switch openings. Drill a ½" hole for file clearance in the center of each switch opening, then file the openings out to their outlines, filing the drilled-out corners square to fit the switches. Drill the button and mounting-screw holes and sand smooth, then check the fit of all parts and screws.

☐ **Panel face.** Have a one-to-one photo negative made of the printed panel face, using Kodak graphic art reproduction reversal film, specifying that the emulsion be on the back side. A commercial printer who does photolithographic work can do this for you. Spray the back side of the negative any color(s) you desire and allow several days to dry. Then spray the painted side of the negative with spray adhesive — I use 3M's. Attach the negative to the panel, being careful to align it with the holes. Then use a pointed knife from the front side to cut out the film at each hole, and to trim away excess film around the panel edge.

Of course you could simply use the paper panel face from fig. 1, or paint the aluminum and apply your own lettering. I use film negatives for all panels on the Sunset Valley because they look great and the lettering and colors can't be scratched or worn off. I demonstrate this method in the Kalmbach videotape, *The Basics of Model Railroad Wiring.*

☐ **Assembly.** To install the switches, epoxy the spacer strips to the back of the panel as shown in fig. 1. When those have set, push the switches into position from the front side of the panel so the locking tabs snap into place. Then mount the two pushbuttons.

☐ **Wiring.** Connect a common logic ground lead to the C position of each switch and to one side of each button. Then connect 13 logic line wires, 4 to each block-number switch, 3 to the cab-number switch, and 1 each to the but-


COMPLETED CAB RELAY CARD

CAB RELAY CARD PARTS LIST (In order of assembly)

Qnty.	Symbol	Description
6	J1-J6	Jumpers, make from no. 20 (to carry track current) uninsulated bus wire (Belden no. 8020)
•	R1-R7	510Ω, ½W resistor [green-brown-brown]
3	R8-R10	2200Ω, ¼W resistor [red-red-red]
•	D1-D7	1A, 100-PIV diode (Digi-Key 1N4002)
1	S1	16-pin DIP socket (Digi-Key C8916)
1	S2	15-pin connector (Molex 09-52-3121)
1	U1	7445 BCD-to-1-of-10 decoder/driver (Jameco SN7445N)
•	RY1-RY7	SPDT relay, 12V coil with contact rated currents as: 3A = (Digi-Key Z411-ND); 7A = (Digi-Key Z416-ND); 12A = (Digi-Key Z421-ND); 15A = (Digi-Key Z424-ND)
1	C1	.1µF, 50V ceramic disk capacitor (Jameco DC.1/50)
1	C2	2.2µF, 35V tantalum capacitor (Jameco TM2.2/35)
•	L1-L7	Red diffused LED (Digi-Key P300)

Author's recommendations for suppliers given in parentheses above, with part numbers where applicable. Equivalent parts may be substituted. All resistors 5 percent, color codes are given in brackets

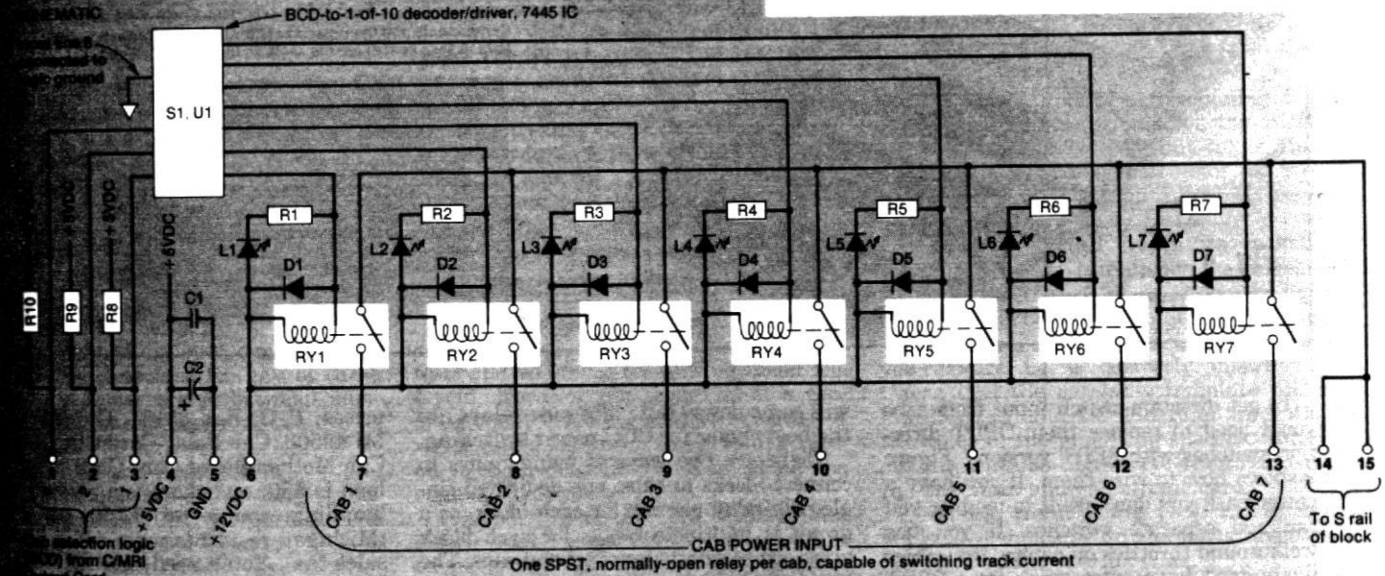
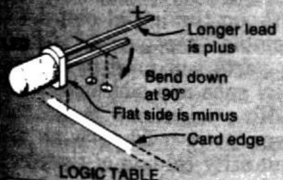
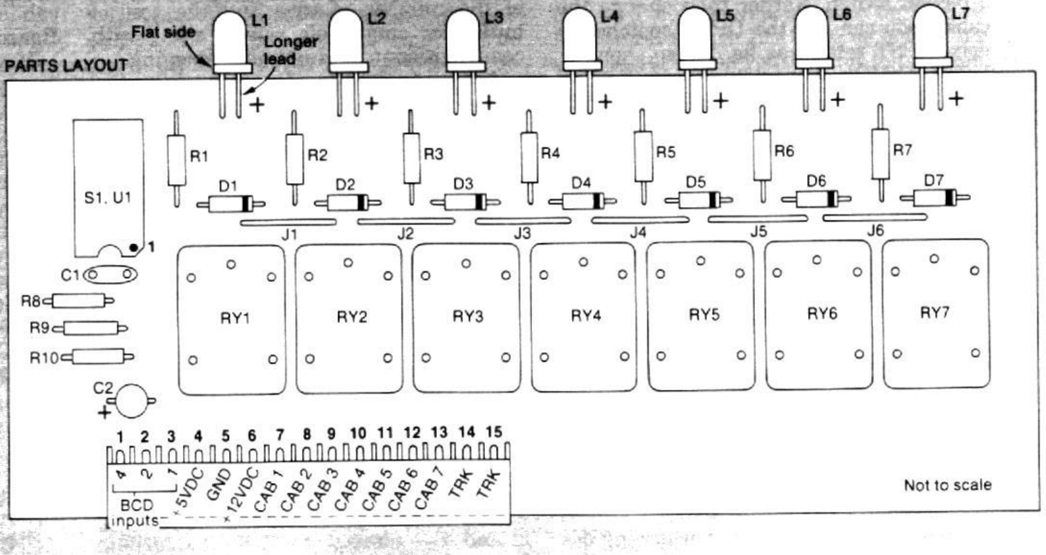


Fig. 3 CAB RELAY CARD



LOGIC TABLE

INPUTS 4	INPUTS 2	INPUTS 1	ACTIVATED RELAY	✓
0	0	0	None	
0	0	1	1	
0	1	0	2	
0	1	1	3	
1	0	0	4	
1	0	1	5	
1	1	0	6	
1	1	1	7	



tons. Lace all the wires together to form a cable, and secure that to the panel plate with a cable clamp as shown.

We won't test the B/CAP until it's connected to the C/MRI, when we can test it easily with software.

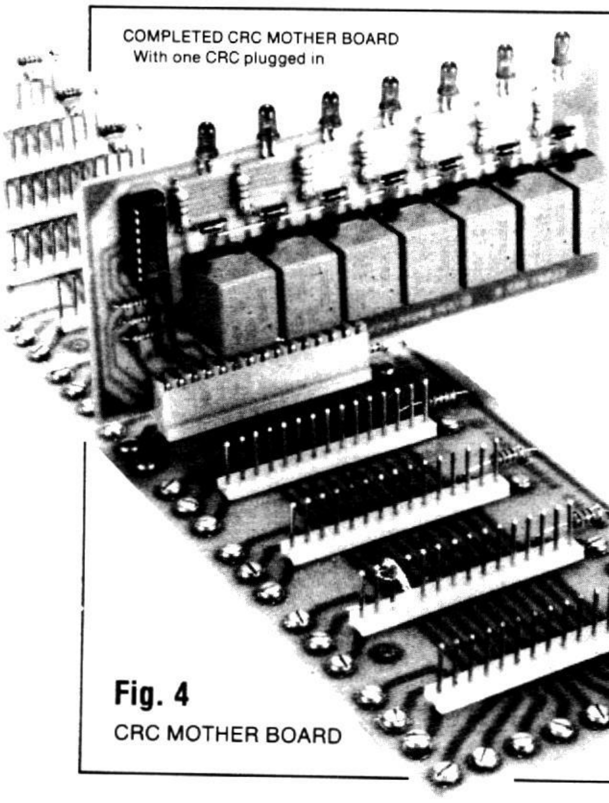
DIRECTIONAL INPUT

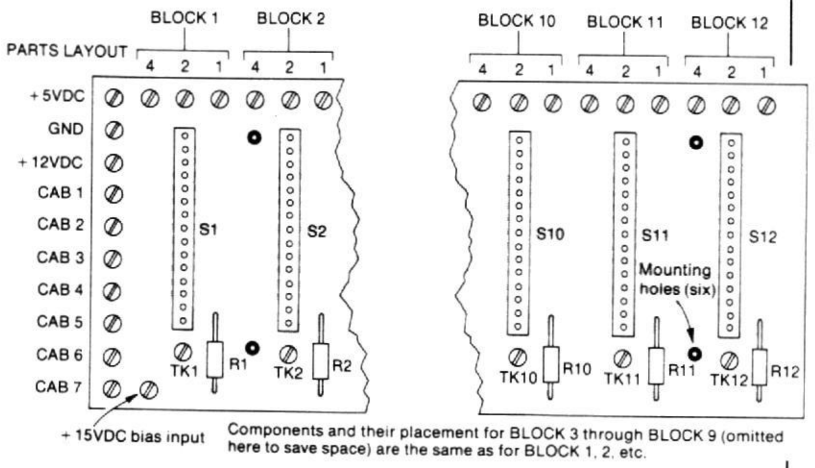
In addition to B/CAP input, the CCC computer needs input for the direction of travel set on each cab. This could come from direction-sensitive occupancy detectors like the Twin-T, or by direct feedback from each cab's direction switch.

For DC trains, the output of a direction-sensitive detector is reliable as long as sufficient voltage is applied to the occupied block. However, when the throttle is off, or nearly so, a direction-sensitive detector can't do the job.

Also, directional input from detectors takes two lines (wires) per detector to the computer, compared to just one for occupancy. You will have many more blocks than cabs, so monitoring direction from direction switch input will take less wiring and require fewer C/MRI Input Cards.

Finally, in order to control return loops and other turning tracks automatically, CCC needs direction input from reverse blocks *before* those blocks are occupied. For that the computer must read the auxiliary direction switch (or its equivalent) that controls a reverse block and not a detector.





Components and their placement for BLOCK 3 through BLOCK 9 (omitted here to save space) are the same as for BLOCK 1, 2, etc.

CRC MOTHER BOARD PARTS LIST (In order of assembly)

Qnty.	Symbol	Description
59	—	4-40 x ¼"-long pan-head machine screw
59	—	4-40 brass nuts
12	S1-S12	15-pin connector (Molex 09-64-1151)
12	R1-R12	2200Ω, ½W resistor [red-red-red]

Author's recommendations for suppliers given in parentheses above, with part number where applicable. Equivalent parts may be substituted. Resistors are 5 percent, color code given in brackets

Fig. 4
CRC MOTHER BOARD

To get direction-switch input from cabs you'll need to replace their DPDT direction switches with 3PDT versions. Figure 2 shows how to wire them. If you have a power pack you don't want to modify, you can add separate switches. If you use walkaround throttles on cables, the cables likely don't have extra wires for a 3PDT direction switch. Rather than change the cables you can use the DPDT switches to drive 3PST relays, as fig. 2 also shows.

The C/MRI connections are identical from either a switch or a relay, with two lines to an Input Card for each cab. One line low indicates that the cab is set in one direction, and the other low indicates the other direction. Neither line low indicates that the cab is in the center-off position, not set for either direction. A center-off position is a necessity for cab direction switches with CCC.

BLOCK POWER SWITCHING

The biggest challenge in CCC is the block power switching itself. Current levels can be high, considering short circuits from derailments or wheels across turnout gaps, long passenger trains with lighted cars, multiple powered units, and especially the larger scales.

There's a variety of power-switching devices to choose from, including stepping switches, power opto-isolators, solid-state photovoltaic relays (PVRs), gate-turnoff devices (GTOs), and triacs. Still, it's hard to beat the old standard mechanical relay.

Relays are easy to understand, to wire, and to debug. They can provide years of dependable, trouble-free operation. Relays are also a lot cheaper than the newest solid-state devices, the PVRs and GTOs, even though the high-current relays we need are fairly expensive even as surplus. New solid-state devices will be developed, and prices of existing ones will come down, but right now relays are the best choice for CCC power switching.

There are two ways of using relays to connect blocks to cabs: one dedicated single-pole relay per cab for each block, or a group of multipole relays for each block arranged to select one of several cabs. The trade-offs between the two are a function of the number of cabs and the cost of multi- vs. single-pole relays, along with complexities in wiring and debugging.

A single-pole relay for each cab is simpler, costs the least per relay, and lets you start with two cabs and add relays for more as needed. There is no theoretical limit to the number of cabs you can add.

On the other hand, the multipole approach requires fewer relays for a given number of cabs. Up to seven cabs would take three: one SPDT, one DPDT and one 4PDT. This could give lower overall cost, but takes more complicated wiring and limits the total number of CCC cabs. For simplicity, flexibility, and expandability, I've designed this CCC system with one single-pole relay per cab per block.

Figure 3 shows a Cab Relay Card (CRC) that connects one of up to seven cabs with one block using single-pole relays. Three BCD logic lines — labeled 1, 2, and 4 — come to the CRC as outputs from a C/MRI Output Card to determine which cab is to be selected. The three lines are decoded as in the fig. 3 logic table, by the 7445 BCD-to-1-of-10 decoder/driver, to select which of the seven cab relays is activated (closed). The LEDs light to indicate which relay is closed, and there's a protective diode across each of the relay coils.

BUILDING THE CAB RELAY CARD

Figure 3 includes the parts layout and parts list for the CRC. Ready-to-use circuit cards are available from JLC Enterprises, P. O. Box 88187, Grand Rapids, MI 49508. Cab Relay Card CRCD7 is $9, Cab Mother Board Card CMB12 (see below) is $30, and there is a shipping and handling charge of $2.50 per order (Michigan residents must add 4 percent sales tax). You'll need one CRC for each block in which the computer is to control cab assignments, and one Cab Mother Board for every 12 CRCs.

If you don't have seven cabs or just want to keep the initial cost down, you can assemble "depopulated" CRCs with just the parts you need to handle your number of cabs and add more cabs later by installing additional parts. Parts in this category are marked by an asterisk in the quantity column of the parts list. You need one each of these parts for each cab. Install them first in the cab 1 position and work up.

Refer to fig. 3 and follow these steps:

☐ **Card test.**

☐ **J1-J6.** Use no. 20 wire, rather than the lighter wire used on previous cards in this series, to better handle track current. All six jumpers are required even if you are not installing all seven relays.

☐ **R1-R10.**

☐ **D1-D7 [+].**

☐ **S1[+].**

☐ **S2.**

☐ **U1[+].**

☐ **RY1-RY7.** Insert each relay, making sure all five pins pass through the holes and that the relay case is held firmly against the card as you solder. For some relays you may need to elongate the holes to get a proper fit.

☐ **C1.**

☐ **C2[+].**

☐ **L1-L7[+].** Note orientation of flat side and + hole (longer lead) in fig. 3. Hold leads securely next to the housing with needle-nose pliers, and bend at right angles so LEDs stick out over the edge of the card (so you can see them when the CRCs are plugged into their Mother Board). Then insert, solder, and trim.

☐ **Cleanup and inspection.**

We'll hold off testing the card until we have the Mother Board assembled.

BUILDING THE CRC MOTHER BOARD

Each CRC Mother Board accommodates 12 CRCs as shown in fig. 4. If you need more you can connect as many Mother Boards as you need with inputs in parallel. Resistors R1-R12 are the bias resistors described in C/MRI part 7; they provide a current path to let the detectors read an occupied block even when the track power is turned off. Here are the Mother Board assembly steps:

☐ **Card test.**

☐ **Terminal screws.** Install the 59 terminal screws and nuts, and solder the nuts to the circuit traces.

☐ **S1-S12.**

☐ **R1-R12.** Install with the leads toward S1-S12 a bit longer than the others, as shown in fig. 4, to allow clearance for the CRC connectors.

☐ **Adjacent trace test.** Set your VOM to R x 100 and check the resistance between each of the adjacent traces. It should be infinite, an open circuit.

A reading close to 0 for any of these indicates a solder bridge between adjacent pads or traces. Locate it, remove it, and retest to be sure.

☐ **Cleanup and inspection.**

For mounting, use six ¼"-long standoffs as for the UBEC in part 2. The boards can be mounted in an open frame as shown for the IOMB in part 4.

RELAY POWER SUPPLY

The relays need a 12VDC power supply. Combined relay and LED current for each occupied block is about 45mA, and assuming that only about a third of your blocks are ever assigned at any one time, a 1.5A supply as shown in fig. 5 would handle up to 100 blocks. Build it like the 5VDC and ±15VDC supplies except, of course, for the 12VDC regulator.

Run a heavy (no. 16 or larger) ground line from your 12VDC supply's ground terminal to your common-ground single-tie-point as I described under "Good Ground Wiring" in part 6. To connect your supplies to the Cab Mother Board, run no. 16 or larger wire from the ground-tie point to Mother Board terminal screw no. 2, and no. 18 or larger wires from +5VDC to screw 1 and from +12VDC to screw 3.

TESTING THE CAB RELAY CARDS

The following tests check that your CRCs function correctly. I'll assume that your CRCs have the full complement of 7 relays each. If they have fewer, ignore the tests for those you haven't installed.

☐ **Plug in.** Place the CRC to be tested in the first slot (block 1) on the Cab Mother Board, taking care that all socket pins go into the proper holes, then turn on the 5VDC and 12VDC supplies. The cab 7 LED should light. If not, check that power from both supplies is reaching the CRC, and that U1 is inserted correctly.

☐ **Zero (no) cabs assigned.** Using clip leads, ground the 1, 2, and 4 terminals for block 1. All the LEDs should be off. Set your VOM on R x 100 and connect one lead to the block 1 track terminal, TK1, then touch cab input screws 4 through 10 (cabs 1-7) with the other. All should read an open circuit, showing no cabs connected to the block. If you find a cab connected, you most likely have a solder bridge; find and correct it before going on.

☐ **Cabs 1 through 7.** Rearrange your three clip leads to ground block 1 logic terminals 1, 2, and 4 according to the logic table in fig. 4: attach a lead where the table shows a 0 (logic 0 = ground), and remove any lead where the table shows a 1. For each case the indicated relay should close and the LED light. Use your VOM to check continuity between the TK1 terminal and the terminal for the activated cab, and to see that there is no continuity to the terminal of any unactivated cab. Check the table in fig. 4 as you pass each test. If you have problems, look for a faulty solder joint, a part left out, or a solder bridge. Correct and continue until you pass all eight conditions, and repeat for each CRC.

CAB-ASSIGNMENT PANEL DISPLAY

The CRCs' LEDs indicate which cab is assigned to a block. They're handy for tests and when you're working under the layout. However, you'll also want a cab-assignment display where you can see it when you're assigning and dropping cabs, on a track diagram panel or along the edge of the layout.

Figure 6 shows the Cab Display Card (CDC) used for the display panel shown on page 66. The LED digit (L4) displays the cab assignment for the block, and the LED below that (L1) indicates block occupancy. The LEDs left and right of the digit in the photo (L2, L3) indicate the direction set on the assigned cab.

Ready-to-use CDC circuit cards are available from JLC Enterprises. Card CDC costs $5 and there is a shipping and handling charge of $2.50 per order (Michigan residents must add 4 percent sales tax). You will need one CDC for each CCC block.

Figure 6 includes the parts layout, parts list, and a mounting template for the CDC. Here's how to assemble it:

☐ **Card test.**

☐ **R1-R13.**

☐ **S1, S2[+].**

☐ **U1[+].**

☐ **Spacers.** Install as shown and secure with 4-40 nuts.

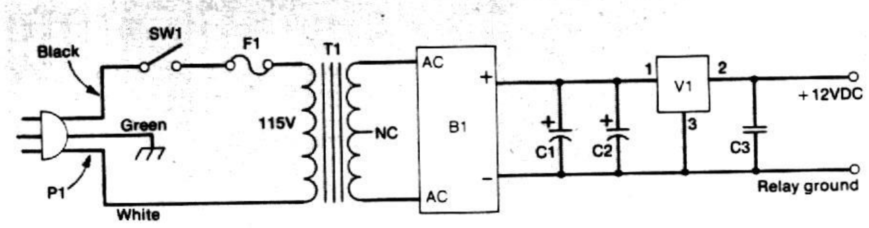
☐ **Panel mounting template.** The template also serves as an LED assembly jig. Lay the printed pattern over the aluminum and prick punch through the paper to mark the drill centers. Drill the mounting holes and check the fit to the CDC by screwing to the spacers. Once this fits, drill and file the opening for the LED digit as you did for the BCD switches on the B/CAP, but make the center hole ⅜". Then drill holes for the occupancy and direction LEDs (I used no. 10 for the Digi-Key LEDs).

☐ **L4 [+].** Install L4 in its socket, making sure its orientation is correct with the decimal point as shown in fig. 6. (We won't be using the decimal point, so paint over it with black paint once you're sure L4 is correctly installed.) Check that the assembly template both fits down around L4 and lines up with the holes in the mounting spacers, filing out the holes as necessary. Then remove the two nuts and the circuit card.

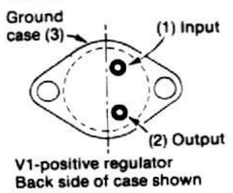
☐ **L1-L3, positioning [+].** Insert + leads in + holes as on the parts layout. DO NOT solder at this time.

☐ **Template assembly.** Mount the circuit card on the template as in fig. 6, carefully working each LED into its hole in the template as far as you'd like it to protrude.

☐ **L1-L3, soldering.** One at a time, hold each LED in position and solder and trim its leads.



WARNING: 120VAC SHOCK CAN BE FATAL. IF YOU DON'T KNOW HOW TO WIRE THE 120VAC CIRCUIT, HAVE AN EXPERT DO IT.



V1-positive regulator
Back side of case shown

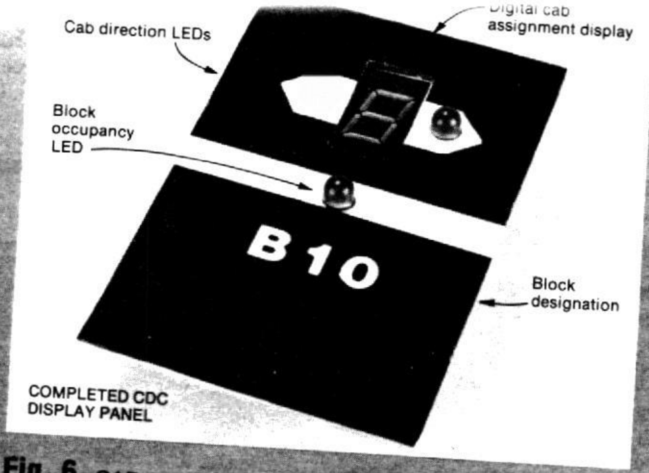
PINOUT IDENTIFICATION
TO-3 type voltage regulator

Fig. 5 +12VDC POWER SUPPLY

+12VDC RELAY POWER SUPPLY PARTS LIST (In order of assembly)

Qnty.	Symbol	Description
1	P1	Three-conductor power cord w/plug (Jameco 17236)
1	—	Strain relief (Radio Shack 278-1636)
1	—	Panel-mount fuse holder (Radio Shack 270-365)
1	—	2A fuse (Radio Shack 270-1275)
1	SW1	Toggle switch (Radio Shack 275-602)
1	T1	18V, 2A power transformer (Radio Shack 273-1515)
1	B1	6A, 50-PIV bridge rectifier (Radio Shack 276-1180)
1	C1	10,000μF, 25V electrolytic capacitor (Digi-Key P6509)
1	C2	.22μF, 35V tantalum capacitor (Jameco TM.22/35)
1	C3	.1μF, 50V ceramic disk capacitor (Jameco DC.1/50)
1	V1	7812K positive 12V, 1A voltage regulator (Jameco)
1	—	Fin-type heatsink for V1 with TO-3 case (Jameco DUDE-4 or equivalent)
1	—	TO-3 socket (Radio Shack 276-029)
1	—	TO-3 mounting hardware (Radio Shack 276-1371)

Author's recommendations for suppliers given in parentheses above, with part numbers where applicable. Equivalent parts may be substituted



COMPLETED CDC DISPLAY PANEL

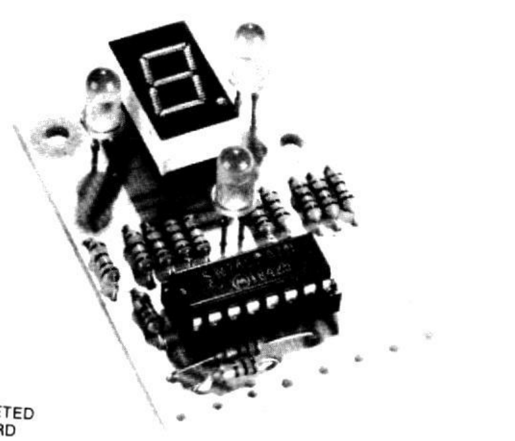
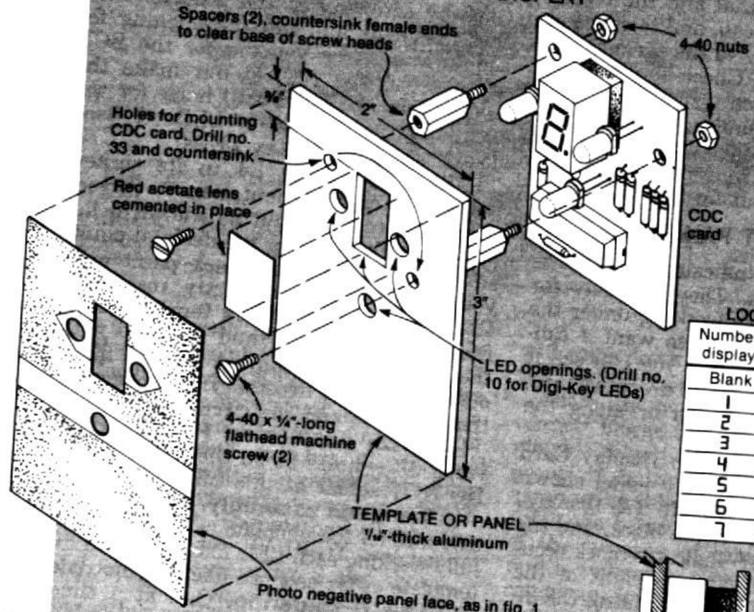
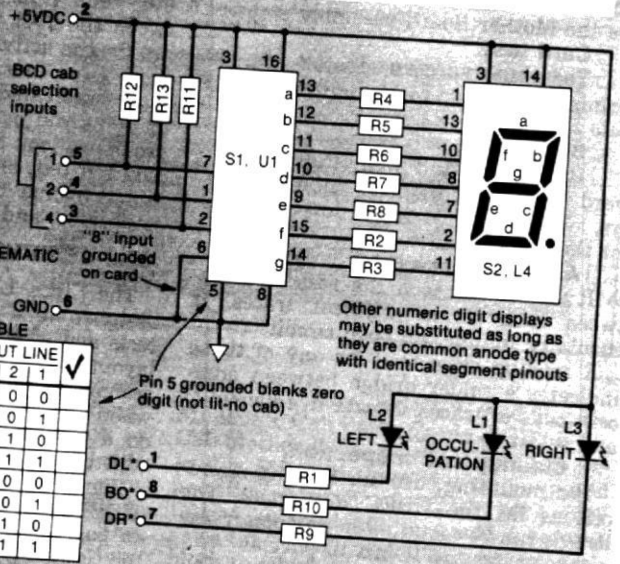

COMPLETED CDC CARD

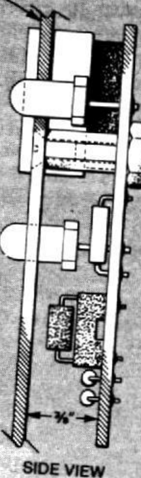
Fig. 6 CAB-ASSIGNMENT PANEL DISPLAY


Spacers (2), countersink female ends to clear base of screw heads
4-40 nuts
3/8"
2"
Holes for mounting CDC card. Drill no. 33 and countersink
Red acetate lens cemented in place
CDC card
3"
LED openings. (Drill no. 10 for Digi-Key LEDs)
4-40 x 1/4"-long flathead machine screw (2)
TEMPLATE OR PANEL 1/16"-thick aluminum
Photo negative panel face, as in fig. 1

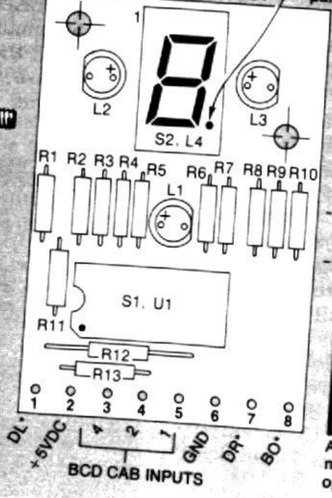

+5VDC 2
BCD cab selection inputs
R12
R13
R11
1 5
2 4
4 3
"8" input grounded on card
SCHEMATIC
GND 6
S1, U1
R4
R5
R6
R7
R8
R2
R3
S2, L4
Other numeric digit displays may be substituted as long as they are common anode type with identical segment pinouts
Pin 5 grounded blanks zero digit (not lit-no cab)
L2 LEFT
L1 OCCU-PATION
L3 RIGHT
DL* 1
BO* 8
DR* 7
R1
R10
R9

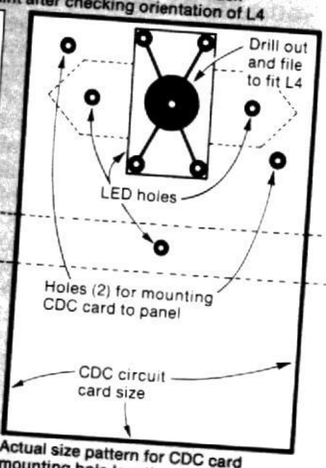
LOGIC TABLE

Number display	INPUT LINE 4	2	1	✓
Blank	0	0	0	
1	0	0	1	
2	0	1	0	
3	0	1	1	
4	1	0	0	
5	1	0	1	
6	1	1	0	
7	1	1	1	



SIDE VIEW


CDC PARTS LAYOUT
Paint out decimal point with black paint after checking orientation of L4
L2
S2, L4
L3
R1 R2 R3 R4 R5 R6 R7 R8 R9 R10
L1
S1, U1
R11
R12
R13
1 2 3 4 5 6 7 8
DL*
+5VDC
4 2 1
GND
DR*
BO*
BCD CAB INPUTS


Drill out and file to fit L4
LED holes
Holes (2) for mounting CDC card to panel
CDC circuit card size
Actual size pattern for CDC card mounting hole locations. Use to locate openings on aluminum template

CAB ASSIGNMENT DISPLAY PARTS LIST (In order of assembly)

Qnty.	Symbol	Description
10	R1-R10	270Ω resistor [red-violet-brown]
3	R11-R13	2200Ω resistor [red-red-red]
1	S1	16-pin DIP socket (Digi-Key C8916)
1	S2	14-pin DIP socket (Digi-Key C8914)
1	U1	74LS47 BCD-to-seven segment decoder/driver (Jameco)
1	L1	Amber diffused LED (Digi-Key P306)
2	L2, L3	Green diffused LED (Digi-Key P303 or may prefer smaller LEDs such as Jameco XC209)
1	L4	1-digit numeric display LED (Digi-Key P326)
2	—	Hex-threaded .375"-long male/female spacer (Digi-Key J211)
2	—	4-40 x 1/4"-long flat head machine screws
2	—	4-40 nuts

Author's recommendations for suppliers given in parentheses above, with part numbers where applicable. Equivalent parts may be substituted. Resistors are 1/4W, 5 percent and color codes are given in brackets

☐ **Cleanup and inspection.**

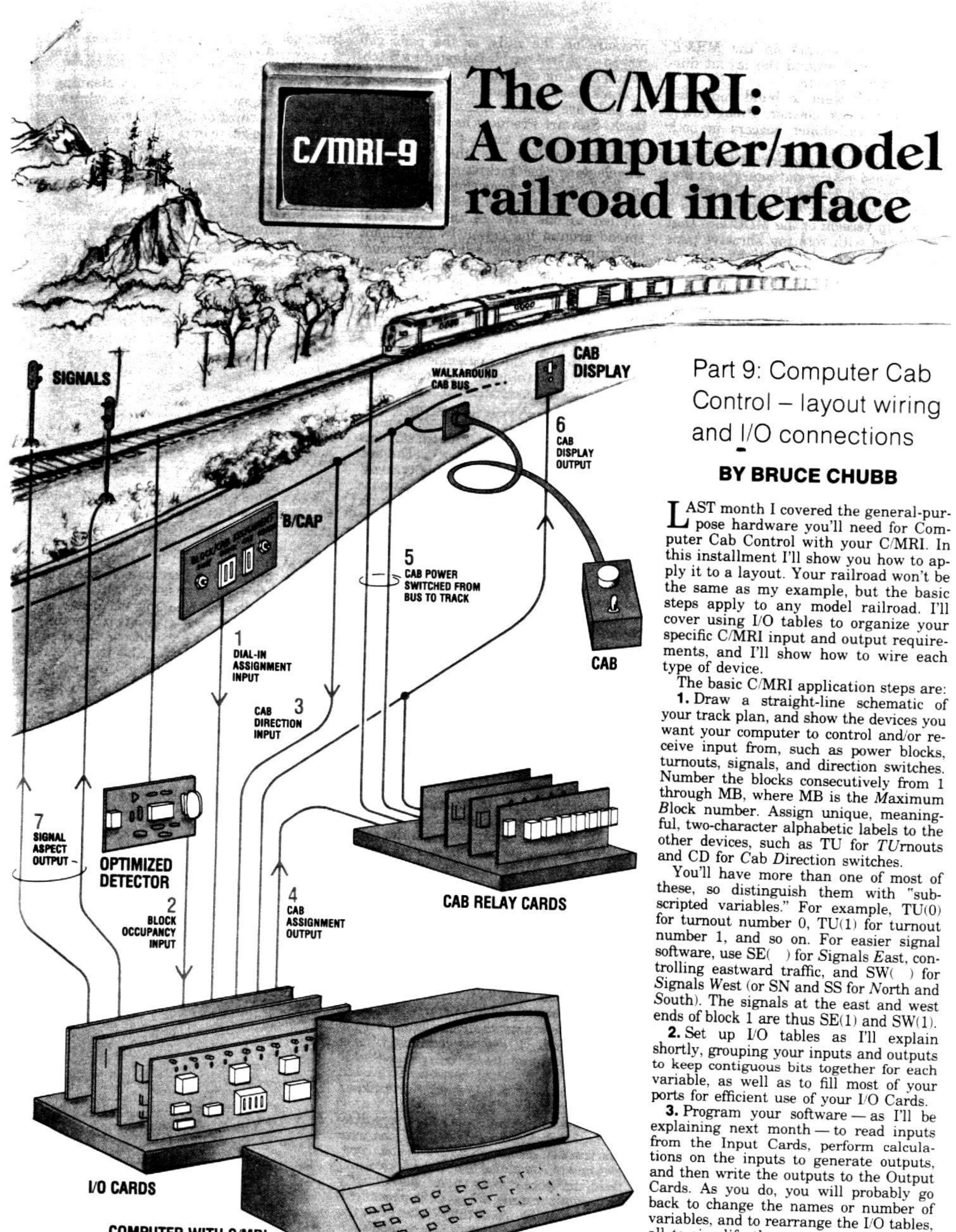
☐ **Card test.** Temporarily solder a ground test lead to card input hole 6 and a +5VDC test lead to hole 2. Connect these to the 5VDC supply and turn on the power. The LED digit should light up with a numeral 7. If not check for improper insertion of L4 or U1, or a solder bridge. Now check the digits from 7 down to 0, following the logic test diagram in fig. 6 to temporarily ground the input lines 1, 2, and 4 so as to check the digits from 7 down to 0. Once these pass, ground input hole 1 to check the left direction LED, hole 7 to check the right LED, and hole 8 to test the occupation LED. If any of these fail, check for a solder bridge or a reversed or faulty LED.

Keep the template handy to drill your panels for mounting CDCs. If you mount the CDCs directly to a panel as shown in fig. 6, secure the screws and spacers to the panel with ACC so they won't come loose when you take the nuts off to remove the card.

Cut a lens of transparent red acetate to just fit over the face of L4. This makes the unlit segments darker so the displayed digit is easier to read. You can get red acetate at office supply stores that stock materials for overhead projector transparencies.

That's all the specialized hardware you need for CCC. Next month I'll show you how to connect it to your railroad and to your C/MRI, and in the installment after that I'll explain how to do the programming to make your Computer Cab Control system work. ◘

The C/MRI: A computer/model railroad interface



Part 9: Computer Cab Control – layout wiring and I/O connections

BY BRUCE CHUBB

LAST month I covered the general-purpose hardware you'll need for Computer Cab Control with your C/MRI. In this installment I'll show you how to apply it to a layout. Your railroad won't be the same as my example, but the basic steps apply to any model railroad. I'll cover using I/O tables to organize your specific C/MRI input and output requirements, and I'll show how to wire each type of device.

The basic C/MRI application steps are:

1. Draw a straight-line schematic of your track plan, and show the devices you want your computer to control and/or receive input from, such as power blocks, turnouts, signals, and direction switches. Number the blocks consecutively from 1 through MB, where MB is the *M*aximum *B*lock number. Assign unique, meaningful, two-character alphabetic labels to the other devices, such as TU for *TU*rnouts and CD for *C*ab *D*irection switches.

You'll have more than one of most of these, so distinguish them with "subscripted variables." For example, TU(0) for turnout number 0, TU(1) for turnout number 1, and so on. For easier signal software, use SE( ) for *S*ignals *E*ast, controlling eastward traffic, and SW( ) for *S*ignals *W*est (or SN and SS for *N*orth and *S*outh). The signals at the east and west ends of block 1 are thus SE(1) and SW(1).

2. Set up I/O tables as I'll explain shortly, grouping your inputs and outputs to keep contiguous bits together for each variable, as well as to fill most of your ports for efficient use of your I/O Cards.

3. Program your software — as I'll be explaining next month — to read inputs from the Input Cards, perform calculations on the inputs to generate outputs, and then write the outputs to the Output Cards. As you do, you will probably go back to change the names or number of variables, and to rearrange the I/O tables, all to simplify the programming. This is normal procedure in engineering design.

4. Connect devices to your I/O cards as defined by your I/O tables. As you make

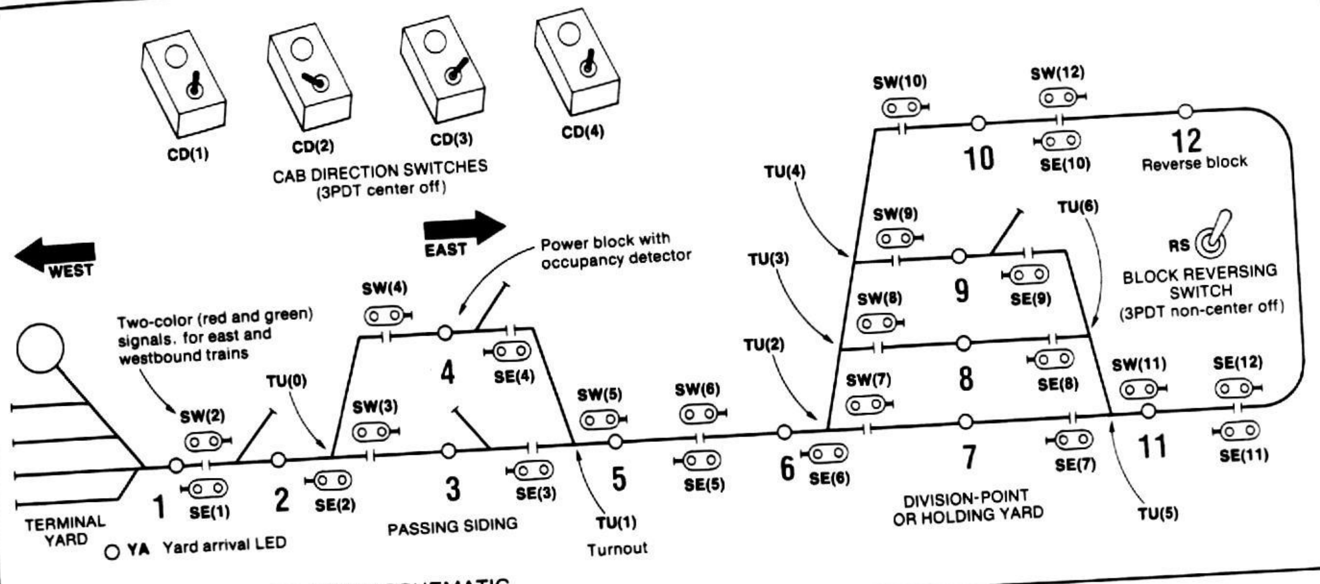


Fig. 1 STRAIGHT-LINE LAYOUT SCHEMATIC

connections to Output Cards you can write short BASIC test programs to turn the devices on and off, to see that everything works as you go. Test your inputs by writing a BASIC program to operate in a loop to read an input port for, say, turnout positions, separate the bits that define individual turnouts — TU(0), TU(1), and so forth — and print them on the CRT. Throw turnouts and your display will be updated, making it easy to check your connections.

5. Test and debug the complete system. You tested the C/MRI in part 5, and if you do a good job of testing I/O connections as you go, most of your problems will be in the software. To help, I'll show you a short but effective simulation that uses keyboard input to let you test your program independently of actual C/MRI inputs.

EXAMPLE LAYOUT

In the rest of this installment I'll go into details on carrying out application steps 1 and 2 above for CCC with the basic trackside signaling that tells engineers not to run into a block controlled by another cab. I'll cover signals in more detail and as an independent system in part 11, but as you'll see next month, signal-control output is almost a byproduct of the CCC software.

Figure 1 is the result of step 1. The layout in this example has 12 controlled blocks, numbered 1 through 12. It also has an assortment of representative features: a terminal yard, a passing siding, a reversing loop, a division-point or staging yard, and spurs off the signaled route.

Seven turnouts, TU(0) through TU(6), control train routing through the 12 powered blocks. For simplicity in this example, I'm assuming the turnouts are not controlled through the C/MRI; they could be either manually or remotely controlled. In general each block has two signals — one at the east end and one at the west. The exception is block 1, a stub terminal yard with no west-end signal.

For the sake of example I'll assume the layout has four road cabs, though CCC could handle up to seven just as easily. The cab direction switches are labeled CD(1) through CD(4). For block 12, the reverse-loop block, the auxilliary *R*eversing *S*witch is labeled RS. For more than one, use RS with subscript notation: RS(1), RS(2), and so forth.

I/O TABLES

We are now ready for step 2, setting up our I/O tables. Figure 2 shows tables for the fig. 1 layout made out on the forms I like to use. The three left-hand columns list ports, bits, and card pinouts (line/wire numbers) for the S11 and S12 I/O Card connectors. In the blanks on the right I write in the function of each I/O line, including the variable name to be used in the software. For this example cards 1 and 2 are Input Cards, and 3 through 6 are Output Cards.

The line locations are mostly arbitrary, but it's handy to group related or similar lines. When it takes more than one bit to represent a variable, the bits should be contiguous on the same port. An example of this is the Cab-number thumbwheel *I*nput, IC, bits 2-4 on card 1, port A.

I likewise grouped the two block-number thumbwheels, four bits each, to fill card 1, port B. Grouping them is not only logical but expedient in programming, since the program can skip past the complete port-and-bit decoding if its decoding of port A shows that neither the assign nor drop buttons were pushed. (To use more than one B/CAP, give each its own pair of ports on separate Input Cards.)

Similarly, I grouped the turnout position inputs to fill a single port, even though that "wastes" one line. The advantage is that a simple comparison of the complete input byte with that of the previous iteration will show if any turnout positions have changed. If none have, subsequent operations involving turnout position input can be skipped.

The center-off cab direction switches need two bits each. The lines for each switch go together, and the group of four cabs neatly fills one port. Direction switch RS for block 12 is not a center-off type. One line can sense two positions, so I tucked it in back on card 1, port A.

We have only four basic types of C/MRI output for the example layout:

- *C*ab *N*umber, CN( ), to drive the Cab Relay Cards and the digital readouts of the Cab Display Cards.
- *D*irection of *T*ravel, DT( ), to drive the direction LEDs on the Cab Display Card panels.
- Signals East and West, SE( ) and SW( ) as explained earlier.
- A single line to drive a *Y*ard-*A*rrival indicator, YA, an LED to tell the yardmaster that a mainline train is approaching his yard.

Each cab-number output needs three contiguous bits to handle our four-cab example, and in fact the same three bits are enough for up to seven cabs. Grouping two cab numbers per port leaves two bits left over in each port for signal lines. Following this pattern, cards 3 and 4 take care of all the cab assignment numbers for the 12 blocks. Card 5 fills out the rest of the signal lines and the yard-arrival LED. I've used card 6 to handle the direction-of-travel LED lines, two each per block.

LAYOUT WIRING

Figure 3 illustrates the general wiring of the layout, with emphasis on the hookups to the C/MRI Input/Output Cards. Since the wiring is repetitive, I've only shown examples of each kind of device.

As described back in part 8, good ground wiring is essential, so make sure you keep your logic grounds separate from your power grounds, that all the ground lines tie together at only one central point close to your power supplies and C/MRI, and that you use heavy wire for ground lines — never less than no. 18 even for the smallest layout, and no. 16 up through no. 10 for larger layouts.

C/MRI I/O WORKSHEET

CARD No. 1 INPUT ~~OR OUTPUT~~

PORT	BIT	PIN	DESCRIPTION OF FUNCTION PERFORMED
A	0	4	ASSIGN PUSHBUTTON – AS
	1	3	DROP PUSHBUTTON – DP
	2	2	1 } CAB NUMBER THUMBWHEEL INPUT – IC
	3	1	2
	4	16	4
	5	17	BLOCK REVERSE SWITCH – RS
	6	18	} SPARE
	7	19	
B	0	13	1 } BLOCK NUMBER LOW ORDER THUMBWHEEL DIGIT – BL
	1	14	2
	2	15	4
	3	24	8
	4	23	1 } BLOCK NUMBER HIGH ORDER THUMBWHEEL DIGIT – BH
	5	22	2
	6	21	4
	7	20	8
C	0	9	TURNOUT 0 POSITION – TU(0)
	1	10	TURNOUT 1 POSITION – TU(1)
	2	11	TURNOUT 2 POSITION – TU(2)
	3	12	TURNOUT 3 POSITION – TU(3)
	4	8	TURNOUT 4 POSITION – TU(4)
	5	7	TURNOUT 5 POSITION – TU(5)
	6	6	TURNOUT 6 POSITION – TU(6)
	7	5	SPARE TURNOUT

C/MRI I/O WORKSHEET

CARD No. 2 INPUT ~~OR OUTPUT~~

PORT	BIT	PIN	DESCRIPTION OF FUNCTION PERFORMED
A	0	4	BLOCK 1 OCCUPATION – BO(1)
	1	3	BLOCK 2 OCCUPATION – BO(2)
	2	2	BLOCK 3 OCCUPATION – BO(3)
	3	1	BLOCK 4 OCCUPATION – BO(4)
	4	16	BLOCK 5 OCCUPATION – BO(5)
	5	17	BLOCK 6 OCCUPATION – BO(6)
	6	18	BLOCK 7 OCCUPATION – BO(7)
	7	19	BLOCK 8 OCCUPATION – BO(8)
B	0	13	BLOCK 9 OCCUPATION – BO(9)
	1	14	BLOCK 10 OCCUPATION – BO(10)
	2	15	BLOCK 11 OCCUPATION – BO(11)
	3	24	BLOCK 12 OCCUPATION – BO(12)
	4	23	SPURS IN BLOCK 2 – SP(2)
	5	22	SPURS IN BLOCK 3 – SP(3)
	6	21	SPURS IN BLOCK 4 – SP(4)
	7	20	SPURS IN BLOCK 9 – SP(9)
C	0	9	} CAB 1 DIRECTION SWITCH CD(1)
	1	10	
	2	11	} CAB 2 DIRECTION SWITCH CD(2)
	3	12	
	4	8	} CAB 3 DIRECTION SWITCH CD(3)
	5	7	
	6	6	} CAB 4 DIRECTION SWITCH CD(4)
	7	5	

C/MRI I/O WORKSHEET

CARD No. 3 ~~INPUT OR~~ OUTPUT

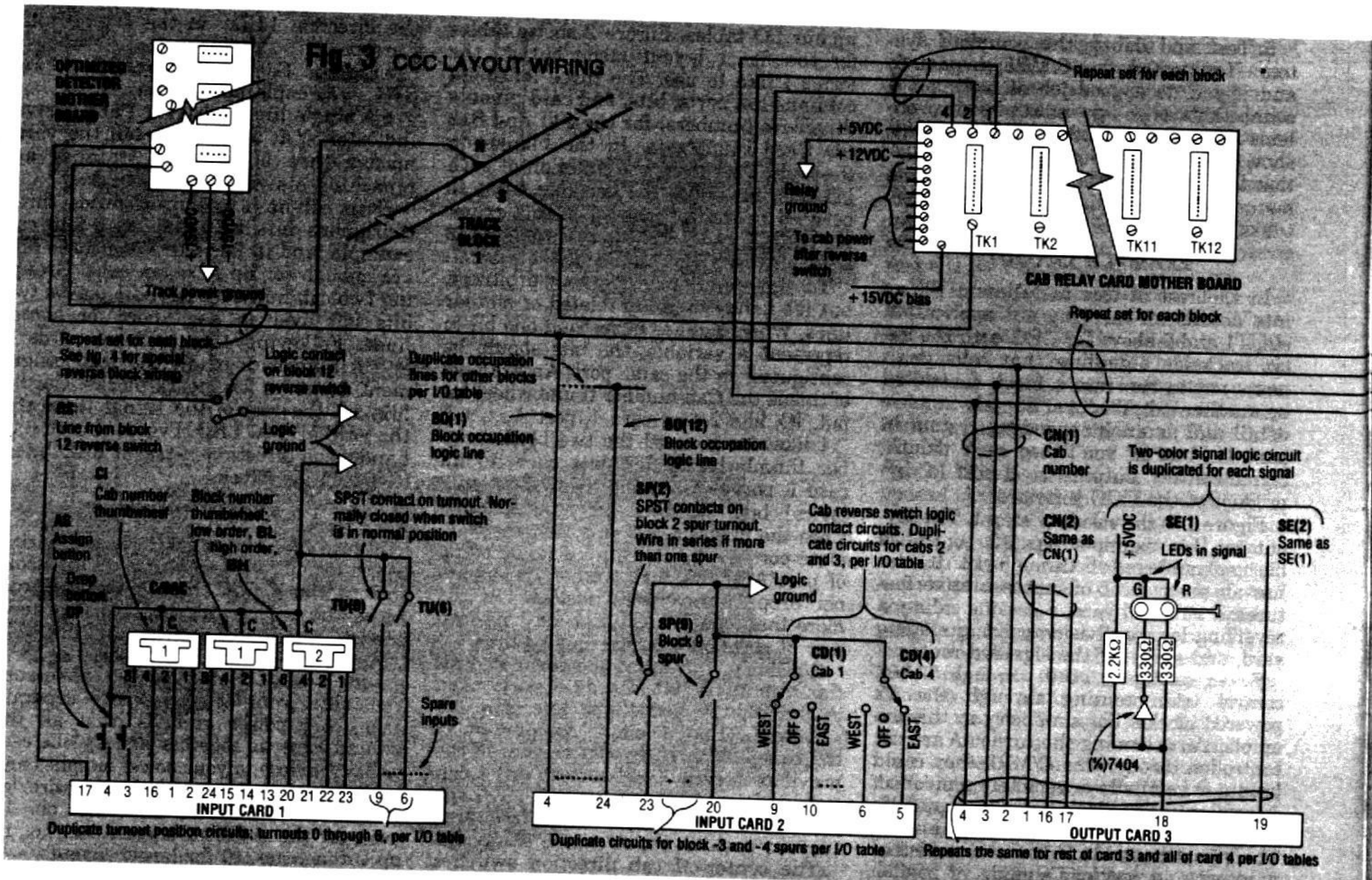
PORT	BIT	PIN	DESCRIPTION OF FUNCTION PERFORMED
A	0	4	} CAB NUMBER BLOCK 1 – CN(1)
	1	3	
	2	2	
	3	1	} CAB NUMBER BLOCK 2 – CN(2)
	4	16	
	5	17	
	6	18	SIGNAL EAST BLOCK 1 – SE(1)
	7	19	SIGNAL EAST BLOCK 2 – SE(2)
B	0	13	} CAB NUMBER BLOCK 3 – CN(3)
	1	14	
	2	15	
	3	24	} CAB NUMBER BLOCK 4 – CN(4)
	4	23	
	5	22	
	6	21	SIGNAL EAST BLOCK 3 – SE(3)
	7	20	SIGNAL EAST BLOCK 4 – SE(4)
C	0	9	} CAB NUMBER BLOCK 5 – CN(5)
	1	10	
	2	11	
	3	12	} CAB NUMBER BLOCK 6 – CN(6)
	4	8	
	5	7	
	6	6	SIGNAL EAST BLOCK 5 – SE(5)
	7	5	SIGNAL EAST BLOCK 6 – SE(6)

CARD No

PORT	B
A	
B	
C	

2 C/MRI I/O WORKSHEET EXAMPLES

Fig. 3 CCC LAYOUT WIRING



OPTIMIZED DETECTOR MOTHER BOARD
Track power ground
N
S
TRACK BLOCK 1
Repeat set for each block. See fig. 4 for special reverse block wiring
Logic contact on block 12 reverse switch
RS Line from block 12 reverse switch
Logic ground
Duplicate occupation lines for other blocks per I/O table
BO(1) Block occupation logic line
BO(12) Block occupation logic line
CI Cab number thumbwheel
Block number thumbwheel: low order, BL high order, BH
AS Assign button
Drop button DP
CABS
C
1
2
8 4 2 1
SPST contact on turnout. Normally closed when switch is in normal position
TU(0)
TU(6)
Spare inputs
17 4 3 16 1 2 24 15 14 13 20 21 22 23 9 6
INPUT CARD 1
Duplicate turnout position circuits: turnouts 0 through 6, per I/O table
SP(2) SPST contacts on block 2 spur turnout. Wire in series if more than one spur
SP(9) Block 9 spur
Logic ground
Cab reverse switch logic contact circuits. Duplicate circuits for cabs 2 and 3, per I/O table
CD(1) Cab 1
CD(4) Cab 4
WEST
OFF
EAST
4 24 23 20 9 10 6 5
INPUT CARD 2
Duplicate circuits for block -3 and -4 spurs per I/O table
Repeat set for each block
+5VDC
+12VDC
Relay ground
To cab power after reverse switch
+15VDC bias
4 2 1
TK1 TK2 TK11 TK12
CAB RELAY CARD MOTHER BOARD
Repeat set for each block
CN(1) Cab number
CN(2) Same as CN(1)
Two-color signal logic circuit is duplicated for each signal
+5VDC
SE(1)
LEDs in signal
G
R
2.2KΩ
330Ω
330Ω
(¼)7404
SE(2) Same as SE(1)
4 3 2 1 16 17 18 19
OUTPUT CARD 3
Repeats the same for rest of card 3 and all of card 4 per I/O tables

ALTERN
The alte
isolates
transisto
grouping

2N39
trans

C/MRI I/O WORKSHEET

ARD No. 4 ~~INPUT OR~~ OUTPUT

ORT	BIT	PIN	DESCRIPTION OF FUNCTION PERFORMED
A	0	4	
	1	3	CAB NUMBER BLOCK 7 – CN(7)
	2	2	
	3	1	
	4	16	CAB NUMBER BLOCK 8 – CN(8)
	5	17	
	6	18	SIGNAL EAST BLOCK 7 – SE(7)
	7	19	SIGNAL EAST BLOCK 8 – SE(8)
B	0	13	
	1	14	CAB NUMBER BLOCK 9 – CN(9)
	2	15	
	3	24	
	4	23	CAB NUMBER BLOCK 10 – CN(10)
	5	22	
	6	21	SIGNAL EAST BLOCK 9 – SE(9)
	7	20	SIGNAL EAST BLOCK 10 – SE(10)
C	0	9	
	1	10	CAB NUMBER BLOCK 11 – CN(11)
	2	11	
	3	12	
	4	8	CAB NUMBER BLOCK 12 – CN(12)
	5	7	
	6	6	SIGNAL EAST BLOCK 11 – SE(11)
	7	5	SIGNAL EAST BLOCK 12 – SE(12)

C/MRI I/O WORKSHEET

CARD No. 5 ~~INPUT OR~~ OUTPUT

PORT	BIT	PIN	DESCRIPTION OF FUNCTION PERFORMED
A	0	4	SIGNAL WEST BLOCK 2 – SW(2)
	1	3	SIGNAL WEST BLOCK 3 – SW(3)
	2	2	SIGNAL WEST BLOCK 4 – SW(4)
	3	1	SIGNAL WEST BLOCK 5 – SW(5)
	4	16	SIGNAL WEST BLOCK 6 – SW(6)
	5	17	SIGNAL WEST BLOCK 7 – SW(7)
	6	18	SIGNAL WEST BLOCK 8 – SW(8)
	7	19	SIGNAL WEST BLOCK 9 – SW(9)
B	0	13	SIGNAL WEST BLOCK 10 – SW(10)
	1	14	SIGNAL WEST BLOCK 11 – SW(11)
	2	15	SIGNAL WEST BLOCK 12 – SW(12)
	3	24	YARD ARRIVAL LED – YA
	4	23	LOOP TIMER LED – XØ
	5	22	
	6	21	
	7	20	
C	0	9	
	1	10	SPARE
	2	11	
	3	12	
	4	8	
	5	7	
	6	6	
	7	5	

C/MRI I/O WORKSHEET

CARD No. 6 ~~INPUT OR~~ OUTPUT

PORT	BIT	PIN	DESCRIPTION OF FUNCTION PERFORMED
A	0	4	DIRECTION TRAVEL BLOCK 1 – DT(1)
	1	3	
	2	2	DIRECTION TRAVEL BLOCK 2 – DT(2)
	3	1	
	4	16	DIRECTION TRAVEL BLOCK 3 – DT(3)
	5	17	
	6	18	DIRECTION TRAVEL BLOCK 4 – DT(4)
	7	19	
B	0	13	DIRECTION TRAVEL BLOCK 5 – DT(5)
	1	14	
	2	15	DIRECTION TRAVEL BLOCK 6 – DT(6)
	3	24	
	4	23	DIRECTION TRAVEL BLOCK 7 – DT(7)
	5	22	
	6	21	DIRECTION TRAVEL BLOCK 8 – DT(8)
	7	20	
C	0	9	DIRECTION TRAVEL BLOCK 9 – DT(9)
	1	10	
	2	11	DIRECTION TRAVEL BLOCK 10 – DT(10)
	3	12	
	4	8	DIRECTION TRAVEL BLOCK 11 – DT(11)
	5	7	
	6	6	DIRECTION TRAVEL BLOCK 12 – DT(12)
	7	5	

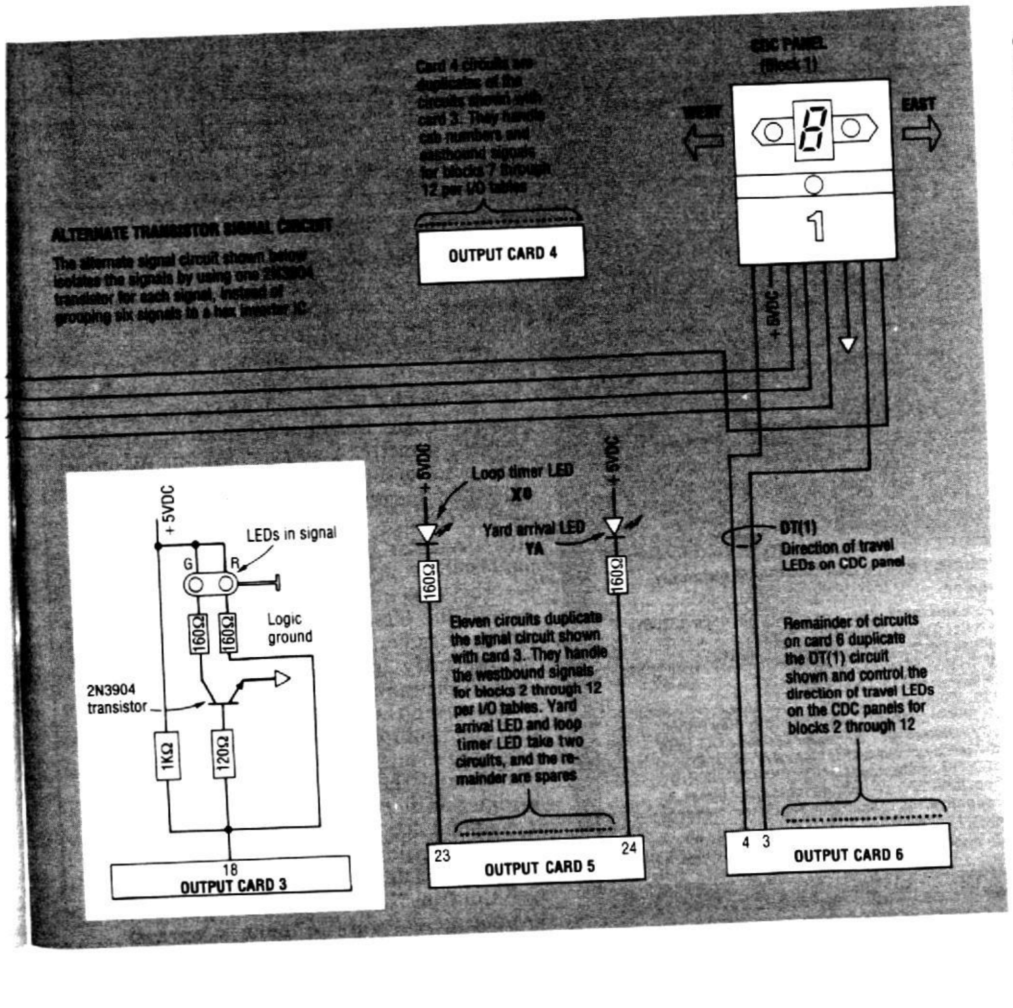

CDC PANEL
(Block 1)
WEST
EAST
1
Card 4 circuits are duplicates of the circuits shown with card 3. They handle cab numbers and eastbound signals for blocks 7 through 12 per I/O tables
OUTPUT CARD 4
ALTERNATE TRANSISTOR SIGNAL CIRCUIT
The alternate signal circuit shown below isolates the signals by using one 2N3904 transistor for each signal, instead of grouping six signals to a hex inverter IC
+5VDC
LEDs in signal
G
R
160Ω
160Ω
Logic ground
2N3904 transistor
1KΩ
120Ω
18
OUTPUT CARD 3
+5VDC
Loop timer LED
XØ
Yard arrival LED
YA
+5VDC
160Ω
160Ω
Eleven circuits duplicate the signal circuit shown with card 3. They handle the westbound signals for blocks 2 through 12 per I/O tables. Yard arrival LED and loop timer LED take two circuits, and the remainder are spares
23
24
OUTPUT CARD 5
DT(1)
Direction of travel LEDs on CDC panel
Remainder of circuits on card 6 duplicate the DT(1) circuit shown and control the direction of travel LEDs on the CDC panels for blocks 2 through 12
4 3
OUTPUT CARD 6

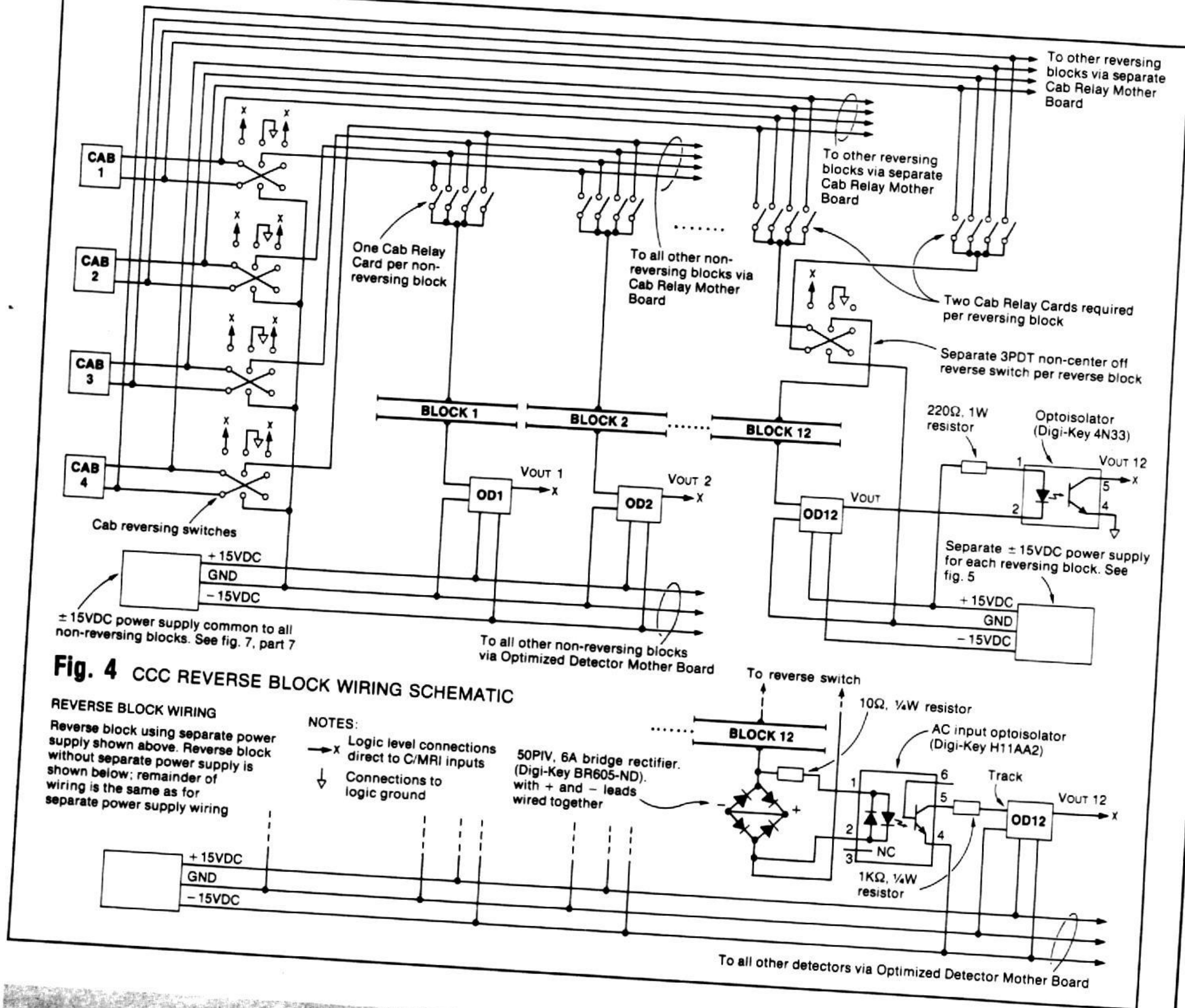
Figure 3 also shows two circuits to decode line 18, card 3, for signal SE(1) — all others are identical. The 7404 is a hex inverter, so one IC can handle six signals, or you can use an individual transistor for each signal. A low output turns on the red LED, and high turns on the green.

The only exception to the repetitiveness of the layout wiring is block 12, the reverse block. It needs special attention as shown in fig. 4, a schematic of CCC wiring for a reverse block (12) with regular blocks (1 and 2) included for comparison.

Notice that for each reverse block you need two Cab Relay Cards, instead of one, to switch the two power lines from each cab ahead of the cabs' reversing switches. The two CRCs for each reversing block have their 1, 2, and 4 logic lines wired in parallel, so they'll switch simultaneously to the same cab-assignment state on command from the C/MRI.

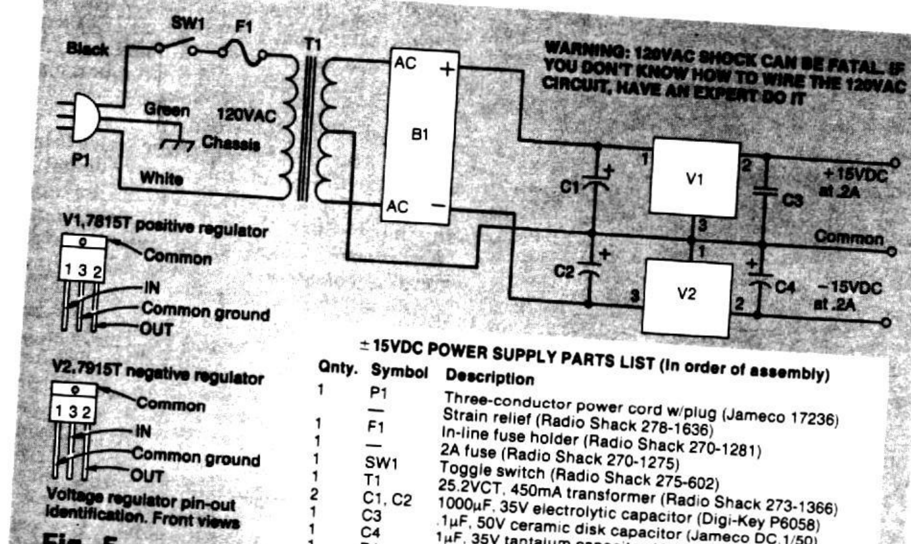
For a layout that has three or more reversing blocks, it would be handiest to use separate CRC Mother Boards (or cut-off portions of the CMB cards) for each set of cab power leads. If you have only one or two reverse blocks, you might simply hard wire the reverse-block CRCs instead of plugging them into CRC Mother Boards.

There are two ways to connect an Optimized Detector to a reverse block. The fig. 4 main schematic shows the Cadillac approach, which retains the Detector's full 1MΩ sensitivity but requires a separate ±15VDC power supply for each reverse block and an optoisolator at the Detector



To other reversing blocks via separate Cab Relay Mother Board
CAB 1
CAB 2
CAB 3
CAB 4
To other reversing blocks via separate Cab Relay Mother Board
One Cab Relay Card per non-reversing block
To all other non-reversing blocks via Cab Relay Mother Board
Two Cab Relay Cards required per reversing block
Separate 3PDT non-center off reverse switch per reverse block
BLOCK 1
BLOCK 2
BLOCK 12
220Ω, 1W resistor
Optoisolator (Digi-Key 4N33)
VOUT 1
VOUT 2
VOUT
VOUT 12
OD1
OD2
OD12
Cab reversing switches
+15VDC
GND
−15VDC
Separate ±15VDC power supply for each reversing block. See fig. 5
±15VDC power supply common to all non-reversing blocks. See fig. 7, part 7
To all other non-reversing blocks via Optimized Detector Mother Board
To reverse switch
10Ω, ¼W resistor
AC input optoisolator (Digi-Key H11AA2)
Track
50PIV, 6A bridge rectifier. (Digi-Key BR605-ND). with + and − leads wired together
NC
1KΩ, ¼W resistor
To all other detectors via Optimized Detector Mother Board
REVERSE BLOCK WIRING
Reverse block using separate power supply shown above. Reverse block without separate power supply is shown below; remainder of wiring is the same as for separate power supply wiring
NOTES:
Logic level connections direct to C/MRI inputs
Connections to logic ground

Fig. 4 CCC REVERSE BLOCK WIRING SCHEMATIC



SW1
F1
T1
Black
Green
120VAC
Chassis
White
P1
AC +
B1
AC −
WARNING: 120VAC SHOCK CAN BE FATAL. IF YOU DON'T KNOW HOW TO WIRE THE 120VAC CIRCUIT, HAVE AN EXPERT DO IT
C1
C2
C3
C4
V1
V2
+15VDC at .2A
Common
−15VDC at .2A
V1,7815T positive regulator
Common
IN
Common ground
OUT
V2,7915T negative regulator
Common
IN
Common ground
OUT
Voltage regulator pin-out identification. Front views

Fig. 5
±15VDC POWER SUPPLY FOR REVERSE BLOCK

±15VDC POWER SUPPLY PARTS LIST (In order of assembly)

Qnty.	Symbol	Description
1	P1	Three-conductor power cord w/plug (Jameco 17236)
	—	Strain relief (Radio Shack 278-1636)
1	F1	In-line fuse holder (Radio Shack 270-1281)
1	—	2A fuse (Radio Shack 270-1275)
1	SW1	Toggle switch (Radio Shack 275-602)
1	T1	25.2VCT, 450mA transformer (Radio Shack 273-1366)
2	C1, C2	1000µF, 35V electrolytic capacitor (Digi-Key P6058)
1	C3	.1µF, 50V ceramic disk capacitor (Jameco DC.1/50)
1	C4	1µF, 35V tantalum capacitor (Jameco TM 1/35)
1	B1	1.5A, 100-PIV bridge rectifier (Digi-Key RB152-ND)
1	V1	7815T positive 15V, 1A voltage regulator (Jameco)
1	V2	7915T negative 15V, 1A voltage regulator (Jameco)

Author's recommendations for suppliers given in parentheses above, with part numbers where applicable. Equivalent parts may be substituted

output. The separate supply powers just the one Detector and its optoisolator, so fig. 5 shows an adequate supply with a current capacity of less than .2A.

The bottom of fig. 4 shows a way to connect the Detector without a separate power supply, by moving the optoisolator ahead of the Detector to read the track current directly. You must use an AC-input optoisolator because the polarity of the track current will change, and wire it with a bridge rectifier in parallel and a 10Ω resistor in series to protect the optoisolator against high track currents — otherwise any track short circuit would exceed the 60mA rating of this part.

The disadvantages of this method are significantly lower detection sensitivity and slight jumps in train speed entering and leaving the reverse block. The front-end optoisolator reduces the Optimized Detector's sensitivity from 1MΩ to about 30KΩ, and the added diode drop in the bridge rectifier will make the track voltage about .8V less in the reverse block than in regular blocks.

That's the layout wiring for Computer Cab Control. Next month I'll explain how to program software to make it work. ♁

The C/MRI: A computer/model railroad interface

Part 10: Computer Cab Control software

BY BRUCE CHUBB

IN the last two C/MRI installments I've covered the hardware for Computer Cab Control and how to wire it into a model railroad. This month I'll explain the CCC program, the software to make the hardware and wiring work.

CCC is an on-line application, with railroad and computer connected through the C/MRI to interact as you run the trains. Conversely, using a computer to generate switch lists is an off-line application. Figure 1 outlines the CCC program: once initialized, the computer loops through its set of instructions over and over, for as long as you want to operate the railroad. Such an operation is called a "tight-infinite" or "real-time" loop.

In its initialization step the program defines variables and constants, initializes the C/MRI by sending command bytes to the 8255s, and so on. Once into the real-time loop, the program first reads the Input Cards for the status of the railroad, checking Block/Cab Assignment Panels, turnout positions, block occupancy, and cab direction switch settings.

From these inputs the program determines whether any blocks need to be assigned or dropped, and calculates the settings for the signals and cab displays. Once the outputs are calculated, the program updates the CRT display (if used) and writes the outputs to the railroad. Then it branches back to reread the inputs and begin the loop over again.

The loop repeats in a couple of seconds or less. As panel inputs change and trains move from block to block, new cab assignments are made, block assignments no longer needed are dropped, and the signals change automatically.

INITIALIZATION

The main program for CCC with two-color signaling on our example layout (fig. 1 in C/MRI Part 9) is listed in fig. 2. As in Part 5, I've used Heath/Zenith MicroSoft BASIC. I've avoided exotic codes, so most of the statements should work on any computer with few changes. Later I'll explain some special coding for Apples. Since this is our first operational C/MRI program, I'll walk you through it.

For easy understanding, the program is grouped in blocks, with a REMark line at the beginning of each to describe the code lines that follow. Anything that follows an apostrophe (') in a code line is also explanatory and will not be executed. Figure 3 defines the important variables for your reference. Don't try to understand all of them now; they'll be clear once you understand the program.

Line 10 prints the program title on the CRT. Line 30 is a MicroSoft BASIC statement for my H8 defining program variables starting with letters A through Z as integers, meaning they can have only whole-number values — 0, 1, 2, 3, and so forth. C/MRI programs don't need real numbers (with fractions), while integer variables use less memory and run faster on some machines. If it doesn't apply to your computer, line 30 can be omitted.

Lines 40 through 70 DIMension, or define the size of, each array variable. For example, variable BO is used to store the *B*lock *O*ccupancy status for each block on the railroad. There are 12 blocks, so BO is dimensioned to BO(12). In use, BO(7) = 0 will mean that block 7 is clear and BO(7) = 1 that it's occupied. The string variables, those followed by a $, store letters, "alpha characters," for the optional CRT display subroutines.

Lines 90 and 100 set up special constants to unpack and pack the input and output bytes received and transmitted over the C/MRI. This lets us make the most effective use of our I/O hardware by packing lots of information into each byte. Each I/O bit, each separate line, may be connected to a separate device on the railroad, or groups of bits may go to single devices. We talk to the railroad a byte at a time using PEEK and POKE or INP and OUT instructions. For input the software must unpack or separate the bits to assign their values to appropriate variables. For output the software packs or combines the values of different variables into bytes for transmission to the railroad.

Lines 120 through 240 attach two-letter symbols to frequently-used constants. As

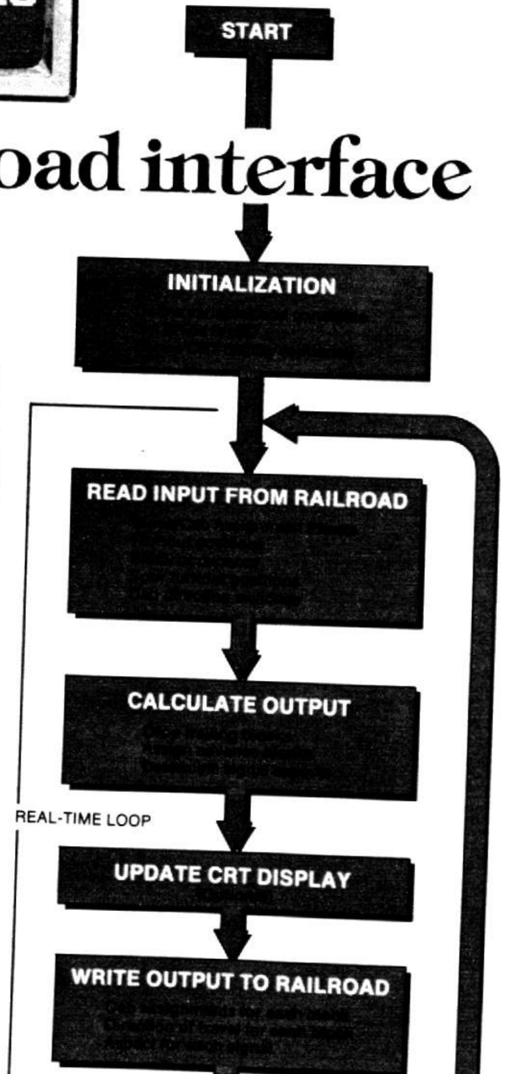


Fig. 1 CCC PROGRAM STRUCTURE

an example, for the program 1 means westbound and 2 eastbound, but lines 190 and 200 let us use WB and EB so we don't have to remember the numbers. Also, line 240 makes the program more general-purpose by defining variable MB for the *M*aximum *B*lock number. If your layout has 40 blocks, simply set MB = 40 and all other relationships which depend on MB are automatically set for 40 blocks.

Lines 260 and 270 are layout-dependent and define the constant next-east or -west blocks, those that don't change with turnout position. Look at the track schematic and they should be obvious. The *N*ext block *E*ast of 1 is 2, thus NE(1) = 2, and likewise NE(5) = 6.

The next blocks east of 2, 3, and 4

```
10
20
30
40
50
60
70
80
90
100
110
120
130
140
150
160
170
180
190
200
210
220
230
240
250
260
270
280
290  C
300  C
310  S
320  R
330  G
340  R
350  P
360  P
370  P
380  P
390  P
400  P
410  R
420  S
430  R
440  X
450  R
460  G
470  I
480  A
490  D
500  I
510  R
520  R
530  T
540  I
550  B
560  B
570  B
580  B
590  B
600  B
610  B
620  B
630  I
640  B
650  B
660  B
670  B
680  S
690  S
700  S
710  S
720  I
730  C
740  C
750  C
760  C
770  R
780  I
790  I
800  B
810  B
820  B
830  C
```

F

```
1Ø   'PRINT "COMPUTER CAB CONTROL-EXAMPLE PROGRAM"
2Ø   REM**Define variable types and array sizes**
3Ø   DEFINT A-Z
4Ø   DIM BO(12),CN(12),DT(12),BS(12),CS(12),DS(12)
5Ø   DIM NE(12),NW(12),SP(12),SE(12),SW(12),ES(12)
6Ø   DIM WS(12),BE(12),BW(12),SS(12),TU(6),TS(6),CD(4)
7Ø   DIM CC(4),DD$(2),DO$(1),DC$(4),DR$(1)
8Ø   REM**Define constants for packing/unpacking bits**
9Ø   BØ=1: B1=2: B2=4: B3=8: B4=16: B5=32: B6=64: B7=128
1ØØ  W1=1: W2=3: W3=7: W4=15: W5=31: W6=63: W7=127
11Ø  REM**Define general constants**
12Ø  CL=Ø        'CLear
13Ø  OC=1        'OCcupied
14Ø  RD=1        'ReD
15Ø  GN=Ø        'GreeN
16Ø  NB=99       'No next Block
17Ø  ND=Ø        'No Direction
18Ø  NC=Ø        'No Cab assigned
19Ø  WB=1        'WestBound
2ØØ  EB=2        'EastBound
21Ø  TN=1        'Turnout Normal position
22Ø  TR=Ø        'Turnout Reverse position
23Ø  PP=1        'Pushbutton Pressed
24Ø  MB=12       'Maximum Block number
25Ø  REM**Define next east/west constant blocks**
26Ø  NE(1)=2: NE(5)=6: NE(1Ø)=12: NE(11)=12: NE(12)=11
27Ø  NW(1)=NB: NW(2)=1: NW(6)=5: NW(12)=1Ø
28Ø  REM**Define I/O interface constants**
29Ø  CI=155          '8255 control byte for input
3ØØ  CO=128          '8255 control byte for output
31Ø  SA=1Ø24         'Starting Address of the I/O
32Ø  REM**Initialize CRT status display-optional**
33Ø  GOTO 3Ø2Ø
34Ø  REM**Transmit control bytes to set up I/O ports**
35Ø  POKE SA+3,CI   'card 1 input
36Ø  POKE SA+7,CI   'card 2 input
37Ø  POKE SA+11,CO  'card 3 output
38Ø  POKE SA+15,CO  'card 4 output
39Ø  POKE SA+19,CO  'card 5 output
4ØØ  POKE SA+23,CO  'card 6 output
41Ø  REM**Force initial iteration thru turnout logic**
42Ø  S1=99 'Set turnout save byte to off-beat value
43Ø  REM**Initialize count variable for loop timing**
44Ø  X=Ø
45Ø  REM**Read inputs and separate out bit areas**
46Ø  GOTO 1Ø1Ø '&&&& LINE FOR TEST TO BYPASS C/MRI INPUT
47Ø  IB=PEEK(SA)       'Card 1 port A
48Ø  AS=IB/BØ AND W1
49Ø  DR=IB/B1 AND W1
5ØØ  IC=IB/B2 AND W3
51Ø  RS=IB/B5 AND W1
52Ø  REM **read port B later only if ASign/DRop pressed**
53Ø  T1=PEEK(SA+2)     'Card 1 port C
54Ø  IB=PEEK(SA+4)     'Card 2 port A
55Ø  BO(1)=IB/BØ AND W1
56Ø  BO(2)=IB/B1 AND W1
57Ø  BO(3)=IB/B2 AND W1
58Ø  BO(4)=IB/B3 AND W1
59Ø  BO(5)=IB/B4 AND W1
6ØØ  BO(6)=IB/B5 AND W1
61Ø  BO(7)=IB/B6 AND W1
62Ø  BO(8)=IB/B7 AND W1
63Ø  IB=PEEK(SA+5)     'Card 2 port B
64Ø  BO(9)=IB/BØ AND W1
65Ø  BO(1Ø)=IB/B1 AND W1
66Ø  BO(11)=IB/B2 AND W1
67Ø  BO(12)=IB/B3 AND W1
68Ø  SP(2)=IB/B4 AND W1
69Ø  SP(3)=IB/B5 AND W1
7ØØ  SP(4)=IB/B6 AND W1
71Ø  SP(9)=IB/B7 AND W1
72Ø  IB=PEEK(SA+6)     'Card 2 port C
73Ø  CD(1)=IB/BØ AND W2
74Ø  CD(2)=IB/B2 AND W2
75Ø  CD(3)=IB/B4 AND W2
76Ø  CD(4)=IB/B6 AND W2
77Ø  REM **If assign button pressed perform assign**
78Ø  IF AS<>PP GOTO 85Ø
79Ø  IB=PEEK(SA+1)     'Card 1 port B
8ØØ  BL=IB/BØ AND W4
81Ø  BH=IB/B4 AND W4
82Ø  BI=BL+(1Ø*BH)
83Ø  CN(BI)=IC
84Ø  REM **If drop button pressed perform drop**
85Ø  IF DR<>PP GOTO 92Ø
86Ø  IB=PEEK(SA+1)     'Card 1 port B
87Ø  BL=IB/BØ AND W4
88Ø  BH=IB/B4 AND W4
89Ø  BI=BL+(1Ø*BH)
9ØØ  CN(BI)=NC
91Ø  REM**If turnout changed decode/update next blocks**
92Ø  IF T1=S1 THEN GOTO 115Ø
93Ø  S1=T1
94Ø  TU(Ø)=T1/BØ AND W1
95Ø  TU(1)=T1/B1 AND W1
96Ø  TU(2)=T1/B2 AND W1
97Ø  TU(3)=T1/B3 AND W1
98Ø  TU(4)=T1/B4 AND W1
99Ø  TU(5)=T1/B5 AND W1
1ØØØ TU(6)=T1/B6 AND W1
1Ø1Ø IF TU(Ø)=TN THEN NE(2)=3: NW(3)=2: NW(4)=NB: GOTO 1Ø3Ø
1Ø2Ø NE(2)=4: NW(3)=NB: NW(4)=2
1Ø3Ø IF TU(1)=TN THEN NW(5)=3: NE(3)=5: NE(4)=NB: GOTO 1Ø5Ø
1Ø4Ø NW(5)=4: NE(3)=NB: NE(4)=5
1Ø5Ø NW(7)=NB: NW(8)=NB: NW(9)=NB: NW(1Ø)=NB
1Ø6Ø NE(7)=NB: NE(8)=NB: NE(9)=NB
1Ø7Ø IF TU(2)=TN THEN NE(6)=7: NW(7)=6: GOTO 111Ø
1Ø8Ø IF TU(3)=TR THEN NE(6)=8: NW(8)=6: GOTO 111Ø
1Ø9Ø IF TU(4)=TR THEN NE(6)=9: NW(9)=6: GOTO 111Ø
11ØØ NE(6)=1Ø: NW(1Ø)=6
111Ø IF TU(5)=TN THEN NW(11)=7: NE(7)=11: GOTO 115Ø
112Ø IF TU(6)=TR THEN NW(11)=8: NE(8)=11: GOTO 115Ø
113Ø NW(11)=9: NE(9)=11
114Ø REM **Set direction of travel for assigned blocks**
115Ø FOR N=1 TO MB
116Ø DT(N)=ND
117Ø IF CN(N)>NC THEN DT(N)=CD(CN(N))
118Ø NEXT N
119Ø DT(12)=RS+1
12ØØ REM**Drop block if not valid advance block**
12Ø5 FOR N=1 TO MB
121Ø IF BO(N)=OC THEN GOTO 126Ø
1215 IF CN(N)=NC THEN GOTO 126Ø
122Ø IF DT(N)=ND THEN GOTO 1255
1225 IF DT(N)=WB THEN AB=NE(N): GOTO 1235
123Ø AB=NW(N)
1235 IF AB=NB THEN GOTO 1255
124Ø IF BO(AB)=CL THEN GOTO 1255
1245 IF CN(AB)<>CN(N) THEN GOTO 1255
125Ø GOTO 126Ø
1255 CN(N)=NC
126Ø NEXT N
128Ø REM**Assign available advance block to desired cab**
1285 FOR N=1 TO MB
129Ø IF BO(N)=CL THEN GOTO 137Ø
1295 IF CN(N)=NC THEN GOTO 137Ø
13ØØ IF DT(N)=ND THEN GOTO 137Ø
131Ø IF DT(N)=WB THEN AB=NW(N): GOTO 133Ø
132Ø AB=NE(N)
133Ø IF AB=NB THEN GOTO 137Ø
134Ø IF BO(AB)=OC THEN GOTO 137Ø
135Ø IF CN(AB)>NC THEN GOTO 137Ø
136Ø CN(AB)=CN(N)
137Ø NEXT N
138Ø REM**compute signal indications**
139Ø FOR N=1 TO MB
14ØØ SE(N)=RD: SW(N)=RD: IE=NE(N): IW=NW(N)
141Ø IF IE=NB THEN GOTO 149Ø
142Ø IF BO(IE)=OC THEN GOTO 149Ø
143Ø IF SP(IE)=TR THEN GOTO 149Ø
144Ø IF BO(N)=OC THEN GOTO 147Ø
145Ø IF CN(IE)>NC THEN GOTO 149Ø
146Ø GO TO 148Ø
147Ø IF CN(IE)<>CN(N) THEN GOTO 149Ø
148Ø SE(N)=GN
149Ø IF IW=NB THEN GOTO 157Ø
15ØØ IF BO(IW)=OC THEN GOTO 157Ø
151Ø IF SP(IW)=TR THEN GOTO 157Ø
152Ø IF BO(N)=OC THEN GOTO 155Ø
153Ø IF CN(IW)>NC THEN GOTO 157Ø
154Ø GOTO 156Ø
155Ø IF CN(IW)<>CN(N) THEN GOTO 157Ø
156Ø SW(N)=GN
157Ø NEXT N
158Ø REM**Set reverse block signals red if polarity wrong
159Ø IF DT(1Ø)<>DT(12) THEN SE(1Ø)=RD: SW(12)=RD
16ØØ IF DT(11)=ND THEN SE(11)=RD: SE(12)=RD: GOTO 163Ø
161Ø IF DT(11)=DT(12) THEN SE(11)=RD: SE(12)=RD
162Ø REM**Set yard approach LED **
163Ø YA=Ø: IF DT(2)=WB THEN YA=1
164Ø REM**update CRT (optional)**
165Ø GOTO 351Ø
166Ø REM**Set XOut bit for timing**
167Ø XO=X AND W1
168Ø REM**Write outputs to railroad**
169Ø OB=Ø                  'Card 3 port A
17ØØ OB=(7-CN(1))*BØ OR OB
171Ø OB=(7-CN(2))*B3 OR OB
172Ø OB=SE(1)*B6 OR OB
173Ø OB=SE(2)*B7 OR OB
174Ø POKE SA+8,OB
175Ø OB=Ø                  'Card 3 port B
176Ø OB=(7-CN(3))*BØ OR OB
177Ø OB=(7-CN(4))*B3 OR OB
178Ø OB=SE(3)*B6 OR OB
179Ø OB=SE(4)*B7 OR OB
18ØØ POKE SA+9,OB
181Ø OB=Ø                  'Card 3 port C
182Ø OB=(7-CN(5))*BØ OR OB
183Ø OB=(7-CN(6))*B3 OR OB
184Ø OB=SE(5)*B6 OR OB
185Ø OB=SE(6)*B7 OR OB
186Ø POKE SA+1Ø,OB
187Ø OB=Ø                  'Card 4 port A
188Ø OB=(7-CN(7))*BØ OR OB
189Ø OB=(7-CN(8))*B3 OR OB
19ØØ OB=SE(7)*B6 OR OB
191Ø OB=SE(8)*B7 OR OB
192Ø POKE SA+12,OB
193Ø OB=Ø                  'Card 4 port B
194Ø OB=(7-CN(9))*BØ OR OB
195Ø OB=(7-CN(1Ø))*B3 OR OB
196Ø OB=SE(9)*B6 OR OB
197Ø OB=SE(1Ø)*B7 OR OB
198Ø POKE SA+13,OB
199Ø OB=Ø                  'Card 4 port C
2ØØØ OB=(7-CN(11))*BØ OR OB
2Ø1Ø OB=(7-CN(12))*B3 OR OB
2Ø2Ø OB=SE(11)*B6 OR OB
2Ø3Ø OB=SE(12)*B7 OR OB
2Ø4Ø POKE SA+14,OB
2Ø5Ø OB=Ø                  'Card 5 port A
2Ø6Ø OB=SW(2)*BØ OR OB
2Ø7Ø OB=SW(3)*B1 OR OB
2Ø8Ø OB=SW(4)*B2 OR OB
2Ø9Ø OB=SW(5)*B3 OR OB
21ØØ OB=SW(6)*B4 OR OB
211Ø OB=SW(7)*B5 OR OB
212Ø OB=SW(8)*B6 OR OB
213Ø OB=SW(9)*B7 OR OB
214Ø POKE SA+16,OB
215Ø OB=Ø                  'Card 5 port B
216Ø OB=SW(1Ø)*BØ OR OB
217Ø OB=SW(11)*B1 OR OB
218Ø OB=SW(12)*B2 OR OB
219Ø OB=YA*B3 OR OB
22ØØ OB=XO*B4 OR OB
221Ø POKE SA+17,OB
222Ø OB=Ø                  'Card 6 port A
223Ø OB=DT(1)*BØ OR OB
224Ø OB=DT(2)*B2 OR OB
225Ø OB=DT(3)*B4 OR OB
226Ø OB=DT(4)*B6 OR OB
227Ø POKE SA+2Ø,OB
228Ø OB=Ø                  'Card 6 port B
229Ø OB=DT(5)*BØ OR OB
23ØØ OB=DT(6)*B2 OR OB
231Ø OB=DT(7)*B4 OR OB
232Ø OB=DT(8)*B6 OR OB
233Ø POKE SA+21,OB
234Ø OB=Ø                  'Card 6 port C
235Ø OB=DT(9)*BØ OR OB
236Ø OB=DT(1Ø)*B2 OR OB
237Ø OB=DT(11)*B4 OR OB
238Ø OB=DT(12)*B6 OR OB
239Ø POKE SA+22,OB
24ØØ REM Increment loop counter
241Ø X=X+1: IF X>31ØØØ THEN X=Ø
242Ø GOSUB 4Ø2Ø    FOR TEST ONLY
243Ø REM**Return to beginning**
244Ø GOTO 45Ø
```

Fig. 2 CCC MAIN PROGRAM

aren't constant and so aren't defined here. For example NE(2) is 3 if *TU*rnout 0, TU(0), is normal, and 4 if TU(0) is reversed. The program decides this after reading turnout positions in the real-time loop. Also note that NW(1) is NB for *N*o *B*lock, since the whole terminal yard is block 1 and there is no block farther west. When you write a CCC program for your own layout, change these statements to fit your track plan.

Lines 290 through 310 define the input and output control bytes for the I/O Cards, and the railroad's *S*tarting *A*ddress. SA = 1024 is the Sunset Valley's starting address; this must be changed to the specific starting address for your computer as in C/MRI Part 3. The DEF SEG statement for an IBM PC can be inserted as line 315, as can the OUT 236,16 statement for a TRS-80 Model 3.

In line 330 the GOTO statement branches to a subroutine to initialize a status display on the CRT. I'll come back to that later, but you don't really need the CRT display for CCC. To omit it, omit lines 320 and 330.

Lines 350 through 400 set the 8255s on the first two cards for input and on cards 3 through 6 for output; change these lines to match the number and arrangement of your own I/O cards. The code given is for memory-mapped I/O; for port I/O substitute OUT for POKE instructions.

Line 420 sets S1, the saved turnout-position byte, to an unlikely value. The program won't have to unpack the turnout-position byte on every iteration, but this forces it to do so on the first one.

Line 440 sets variable X to 0 (zero), to be incremented by +1 each time through the real-time loop. This can be printed on the CRT to show that the program is running and to let you time the loops.

INPUT

The real-time loop starts at line 450 with the reading of inputs and separation of values for assignment to variables. Line 460 is a temporary instruction to skip reading the railroad so you can test the program with simulated input from the keyboard. For normal operation using railroad input, omit 460.

Line 470 reads the *I*nput *B*yte from port A of card 1 at address SA and stores it at location IB. Again, the example is for memory-mapped I/O; for port I/O substitute IN (or INP) instructions for PEEK. After each PEEK we unpack the bits once for each variable read from that port.

As in the I/O tables in Part 9, card 1 port A includes four variables: *AS*sign button AS, *DR*op button DR, *I*nput from *C*ab thumbwheel setting IC, and block 12 auxilliary *R*everse *S*witch RS. All take one bit each except IC, which takes three.

The pattern for each unpacking statement is identical. The variable to be defined is calculated by first dividing the Input Byte by a bit-position constant, B0 through B7, depending on the position of the variable's Least Significant Bit or LSB. This effectively shifts the byte to the right by the required number of bits so that the LSB is in the B0 position.

That result is ANDed with a second constant, W1 through W5, depending on the variable's *W*idth in bits. The AND operation zeroes out or masks all but the bits defining the variable in question.

Fig. 3 TABLE OF VARIABLES

BO(N):	Array dimensioned to the maximum block number used to define the current occupation status of the Nth block; e.g. BO(7)=1 for block 7 occupied and BO(7)=Ø for block 7 clear.
BS(N):	Array dimensioned to the maximum block number used to Save the last iteration of the block occupation array BO(N)
N:	Array index (pointer) used for subscripted variable notation.
CN(N):	Array dimensioned to the maximum block number used to define the current cab assignment(if any) of the Nth block; e.g. CN(5)=Ø for no cab assigned to block 5 and CN(5)=3 for cab number 3 assigned.
CS(N):	Array dimensioned to the maximum block number used to Save the last iteration of the CN(N) array.
DT(N):	Array dimensioned to the maximum block number used to define when a cab is in the Nth block the direction of travel(if any); e.g. DT(7)=Ø if the cab assigned to block 7 is set for no direction or if there is no cab assigned to block 7, DT(7)=1 if the cab assigned to block 7 is set for Westbound, DT(7)=2 if the cab assigned to block 7 is set for Eastbound.
DS(N):	Array dimensioned to the maximum block number used to Save the last iteration of the DT(N) array.
NE(N):	Array dimensioned to the maximum block number used to define the current Next Eastward block number connected to the Nth block; e.g. NE(1)=2 defines that the next eastward block for block 1 is block 2. Block number is set to NB (where NB=No Block=99) if there is no next block connected.
BE(N):	Array dimensioned to the maximum block number used to save the last iteration of the NE(N) array.
NW(N):	Array dimensioned to the maximum block number used to define the current Next Westward block number connected to the Nth block; e.g. NW(6)=5 defines that the next westward block for block 6 is block 5. Block number is set to NB (where NB=No Block=99) if there is no next block connected.
BW(N):	Array dimensioned to the maximum block number used to save the last iteration of the NW(N) array.
SP(N):	Array dimensioned to the maximum block number used to define the current status of any spurs left open in the Nth block; e.g. SP(4)=Ø indicates all spurs(if any) are set for normal position in block 4 and SP(4)=1 indicates that one or more spurs in block 4 are in the reverse(unsafe position).
SS(N):	Array dimensioned to the maximum block number used to Save the last iteration of the SP(N) array.
SE(N):	Array dimensioned to the maximum block number used to define the current state of the Signal at the East end of the Nth block; e.g. SE(12)=Ø defines a red signal and SE(12)=1 defines a green signal.
ES(N):	Array dimensioned to the maximum block number used to save the last iteration of the SE(N) array.
SW(N):	Array dimensioned to the maximum block number used to define the current state of the Signal at the West end of the Nth block; e.g. SW(12)=Ø defines a red signal and SW(12)=1 defines a green signal.
EW(N):	Array dimensioned to the maximum block number used to save the last iteration of the SW(N) array.
TU(N):	Array dimensioned to the maximum turnout number used to define the current position status of route selection turnouts; e.g. TU(Ø)=1 defines that turnout Ø is set for the turnout normal route and TU(Ø)=Ø defines that turnout Ø is set for the turnout reversed route.
TS(N):	Array dimensioned to the maximum turnout number used to Save the last iteration of the TU(N) array.
CD(N):	Array dimensioned to the maximum cab number defining Cab Direction i.e. position of the Nth cabs reverse switch; e.g.: CD(3)=Ø if cab 3 reverse switch is set at center off(no direction), CD(3)=1 if cab 3 reverse switch is set for westbound, CD(3)=2 if cab 3 reverse switch is set for eastbound.
CC(N):	Array dimensioned to the maximum block number used to Save the last iteration of the CD(N) array.
DD$(2):	Array used to define the Displayed Direction used on the CRT for direction of travel; i.e.: DD$(Ø)=" " (a blank) for no direction, DD$(1)="<" for westbound, DD$(Ø)=">" for eastbound.
DO$(1):	Array used to define the Displayed Occupation used on the CRT: DO$(Ø)=" " (a blank) for clear, DO$(1)="1" for occupied.
DC$(4):	Array used to define the Displayed Cab number used on the CRT: DC$(Ø)=" " (a blank) for no cab, DO$(1)="1", DO$(2)="2", DO$(3)="3" and DO$(4)="4".
DR$(1):	Array used to define the Displayed Red signal used on the CRT: DR$(Ø)=" " (a blank) for green, DR$(1)="R" for red.
BØ, B1, B2, B3, B4, B5, B6, and B7:	Constants used to define the low order Bit position for each variable to be unpacked or packed in an input or output byte.

It's easier to do than to explain. For example, the I/O table shows you that AS is 1 bit wide at position 0, so you write line 480 as AS = IB/B0 AND W1. Variable IC is 3 bits wide with its LSB at position 2, so line 500 is IC = IB/B2 AND W3. Once all the variables from one port are defined, you move on to the next with another PEEK to read the next input byte.

That's all there is to reading railroad inputs. Lines 530 through 1000 follow the same process: a PEEK to bring in the port data followed by repeated executions of the same basic unpacking instruction format to separate the input bits defining the respective variables.

You can group your input lines to cut the cycle time of your real-time loop. For example input and decoding of card 1, port B is completely skipped unless the reading of port A shows that either the assign or drop button is pressed.

Likewise I purposely grouped all the turnout position inputs into a single port. Line 530 reads the port and stores the input byte as variable T1. In line 920 the present T1 is compared with the last iteration's T1 value, stored as S1.

If no turnouts have been thrown, T1 = S1 and the program branches to line 1150, bypassing the unpacking of the individual bits and the calculation of the next blocks which depend on turnout position. If you have more than eight CCC routing turnouts you can skip each group of eight by using variables T2 and S2 for the second group, followed by T3 and S3 for the third group, and so on.

We need to go back and see how the thumbwheel switches are handled. Line 780 invokes a branch to skip to line 850 if AS is not equal to PP (AS<>PP), the variable defined in line 230 for *P*ush button *P*ressed. AS = PP if the button is pressed, and line 790 reads card 1 port B. Line 800 sets BL to the *B*lock *L*ow-order thumbwheel switch value (units digit), and 810 sets BH to the *B*lock *H*igh-order value (tens digit). The total *B*lock number *I*nput is then calculated as BI = BL+(10*BH) in line 820.

Since the assign button is pressed, line 830 sets the cab number in block BI to the number read from thumbwheel switch

W1, W2, W3, W4, W5, W6, W7:	Constants used to define the number of bits Wide taken up by a variable in the input or output byte.
YA:	Yard Arrival indication used to inform the yardmaster that a train has entered single track heading for the yard: YA=1 indicates train arriving, YA=Ø indicates train not arriving.
X$:	String variable used to temporarily store the Next East or Next West block numbers used on the CRT.
FNDC$(X,Y):	Direct Cursor function used to direct the cursor to the point on the CRT screen defined by the argument input coordinates where X=line number and Y=column number.
CHR$(I):	Intrinsic function used to return a string whose element has the ASCII code corresponding to the input argument I.
LEN(X$):	Intrinsic function used to return the length of the string X$ with non-printing characters and blanks counted.
CI:	Variable set to the Control Input byte for the 8255; CI=155.
CO:	Variable set to the Control Output byte for the 8255; CO=128.
SA:	User defined variable to be set to the C/MRI's Starting Address e.g. SA=1Ø24 for the Sunset Valley.
IB:	Variable used for temporary storage of the Input Byte read in from an input port.
OB:	Variable used for temporary storage of the Output Byte being formulated to send to an output port.
AS:	Variable used to define the status of the ASsign pushbutton on the cab/block initialization panel; AS=1 if pushbutton pressed and AS=Ø if pushbutton not pressed.
DR:	Variable used to define the status of the DRop pushbutton on the cab/block initialization panel; DR=1 if pushbutton pressed and DR=Ø if pushbutton not pressed.
IC:	Variable used to define the Input Cab thumbwheel switch position: IC=Ø defines no cab number selected, IC=1 through 7 defines the cab number input selected.
BH:	Defines the input Block High order thumbwheel switch setting(Ø thru 9).
BL:	Defines the input Block Low order thumbwheel switch setting(Ø thru 9).
BI:	Defines the Block Input selected on the cab/block initialization panel as calculated by BI=BL+(1Ø*BH).
RS:	Defines the position the the block 12 reverse switch (w/o center off): RS=Ø for eastbound and RS=1 for westbound.
T1:	Defines the 1st(and only for this example) Turnout input byte status.
S1:	Used to Save the 1st(and only for this example) Turnout input byte for comparison to see if a turnout has changed status.
CB:	Working variable used to temporarily define a specific CaB number entry in CN( ).
AB:	Working variable used to temporarily define the Advance Block and Approach Block numbers.
IE:	Working variable used to temporarily define the next East block number.
IW:	Working variable used to temporarily define the next West block number.
X:	Loop counter used for program timing.
XO:	Lowest order bit of X used for C/MRI display Output for timing program iteration rate.
CL:	Variable set to constant=Ø for CLear.
OC:	Variable set to constant=1 for OCcupied.
RD:	Variable set to constant=1 for ReD.
GN:	Variable set to constant=Ø for GreeN.
NB:	Variable set to constant=99 for No next Block.
ND:	Variable set to constant=Ø No Direction.
NC:	Variable set to constant=Ø for No Cab assigned.
WB:	Variable set to constant=1 for WestBound.
EB:	Variable set to constant=2 for EastBound (East is even).
TN:	Variable set to constant=1 for Turnout Normal.
TR:	Variable set to constant=Ø for Turnout Reversed.
PP:	Variable set to constant=1 for Pushbutton Pressed.
MB:	Variable set to maximum block number (MB=12 for this example).
VA$:	String variable used to enter the two-alpha-character code when using simulated input.
NI:	Variable used to enter the index to the array position to be altered when using simulated input.
VA:	Variable used to enter the numeric value for parameter to be altered when using simulated input.

IC. Lines 850 through 900 handle the drop case in the same manner except that CN(BI) is set to 0, NC for *No Cab*.

Lines 1010 through 1130 are layout dependent and define those next blocks east and west that vary with turnouts. In line 1010, for example, if turnout 0 is set to *Turnout Normal*, TN, then NE(2) = 3, NW(3) = 2, and NW(4) = NB, and the program branches around line 1020. If turnout 0 is reversed, not normal, line 1020 is executed to set NE(2) = 4, NW(3) = NB and NW(4) = 2. This all follows from the track plan, and 1030 and 1040 do the same thing for turnout 1.

The three-track yard — blocks 7, 8, and 9 — and branch into block 10 require a bit more logic. Since at most times NE and NW for blocks 7, 8, and 9, as well as NW for 10 will be no block, these are set to NB in lines 1050 and 1060. Then 1070 through 1130 reset whichever next blocks become valid when turnouts are thrown.

Lines 1150 through 1180 set the *Di*rection of *T*ravel, DT, for each block. First 1160 sets DT(N) for *No Direction*, ND. Then 1170 resets DT for each block that has a cab assigned (CN(N) greater than 0), equal to the direction set on the assigned cab. Line 1190 takes care of reverse block 12, where DT(12) is determined by the block 12 auxilliary reversing switch, not the cab direction switch.

BLOCK/CAB ASSIGNMENT

It's best to first have the program automatically drop any assigned blocks no longer needed — such as trailing blocks that have become unoccupied — then search for and assign clear blocks ahead of moving trains. That way any blocks that were dropped are immediately available for reassignment. For 12 blocks or 120, the 12 code statements in lines 1205 through 1260 will do the job.

To help you understand this code, fig. 4 shows the logic flow for dropping cab assignments. Rectangular boxes in this diagram represent calculations performed by the computer. Diamonds represent decision points such as IF statements. The statement numbers let you compare the program listing to the chart.

To find cab assignments that can be dropped the program loops through each block from 1 to MB. If line 1210 finds the Nth block occupied or 1215 that it has no cab assigned, no action is taken and the program branches to line 1260 to increment N for testing the next block. If at 1220 it finds that there is no direction of travel setting, the cab assignment is dropped *via* a branch to 1255.

For each clear block with a cab assigned and a direction of travel identified, program lines 1225 and 1230 look for an *A*pproach *B*lock, AB, in the direction from which the train is approaching — opposite the direction of travel. If there is no approach block, or if it is also clear, or if it is assigned to another cab, the program again branches to program line 1255 to drop the assignment for the Nth block. If it gets past all of the IF statements for a given block, the program again reaches line 1260 and the assignment is retained.

Lines 1285 through 1370 automatically assign cabs to the blocks ahead of each moving train. Again, these lines stay the same whatever the track plan or size of your layout! The program takes another look at each block. For each one it finds occupied, and with both a cab assignment and a direction of travel, the program looks for an *A*dvance *B*lock, also called AB, the next one in the direction of travel. If there is an AB, and if it is both clear and unassigned, the program makes the assignment. See fig. 5.

Any time an IF condition is true the program makes no assignment, but instead branches to line 1370 to increment N for testing the next block. If it gets past all the IF condiditons for a given block, the program reaches line 1360 and sets the cab number for the advance block equal to the cab number for the Nth block.

SIGNALING

Once all the block occupancy, turnout position, and cab assignment data is in the computer, it takes only a few added steps to operate trackside signals on a CCC layout. I'll cover signals in more detail in the February MR. For now I'll just explain software to drive a simple two-color signal system to make CCC operation easier for your railroad's engineers.

Lines 1390 through 1570 perform all the general logic for the signals, and again these 19 lines are the same for any layout. The software loops through each block and initially sets the signals at the east and west ends to red. Also, to save execution time by minimizing use of subscribed array variables, working variables IE and IW are temporarily set equal to the next-east and -west block numbers.

Lines 1410 through 1480 then determine if it is safe to change the east end signal from red to green. Figure 6 shows the logic flow. If the next block east is no block, if it is occupied, or if it has a spur turnout reversed, the signal guarding the block is kept red by having the program branch to line 1490.

If these tests all fall through, the program reaches line 1440 to check occupancy of the Nth block. If occupied, 1440 branches to line 1470, an important check of whether the cab assigned to the Nth block is the same one running the approaching train. If the cab numbers

VIRNEX DECALS and STEWART PRODUCTS

VIRNEX	STEWART
DeLuxe Railroad Emblem Decals	Track Cleaning Adapter
Railroad Roman Alphabet	Traction Tire Tool N & HO
Also VIRNEX "No Need To Trim"	Super Traction Tires N & HO
Pilot Stripe Decals	I Beam Crane Hoist HO
Bold Gothic Alphabet & Numbers	Diesel Service Facilities
Narrow Stripes	Yard Service Facility Work Pit
Sill Stripes For Athearn SW 1500	Universal Cleaning Solvent
Logo Shapes	

ASK FOR THESE ITEMS AT YOUR FAVORITE HOBBY SHOP

VIRNEX Industries, Inc.

Rte. 1 Box 154-B Herwig Rd. Reedsburg, WI 53959

Save by mail-ordering from Germany.

You can save "real" money by mail ordering from us, here in Germany. We carry the full range of European makers of model trains and accessories.

Our service is fast and efficient. Send $2 and SAE for our NEW list.

Modellbahn Drechsler
Detmolder Str. 184
48 Bielefeld, F.R.G.

Free
AIR BRUSH
Catalog

"America's most comprehensive mail order airbrush catalog."
40 pages filled with the top names in airbrush and graphic arts: Efbe, Iwata, Paasche, T & C, Badger, DeVilbiss, Binks, AirWaves, Deka, Liquitex, Magic Color, Tombow, Frisk, Designaire, and more. Complete supplies, parts and books for the professional, hobbyist and student. Save on special sets.
Write or call now for free catalog!

ARTISTIC AIRBRUSH

TOLL-FREE 800-547-9750 ext. 387
(Oregon, Alaska, Hawaii & Canada 503/249-6298)
Dept. 387 P.O. Box 3318 Portland, OR 97208

are not equal, a branch to 1490 keeps the signal red. If the assigned cabs are identical, 1480 changes the signal at the east end of the block to green.

If, back in 1440, the Nth block is unoccupied, line 1450 is executed: if the next block east has a cab assigned, a branch to line 1490 keeps the signal red. If there is no cab assigned to the next block, line 1480 is reached and the signal at the east end of the block changes to green. Lines 1490 through 1560 are just like 1420-1480, except that they govern the signal at the west end of the block.

That's the standard signal logic, but reverse-loop block 12 is a special case. How nice it would be to have the computer interlock the block 12 auxilliary reversing switch with the appropriate cab's direction switch, and then set the signals so that a train wasn't permitted to enter or leave block 12 unless the track polarities were correct!

With the C/MRI, all sorts of nice things are possible just by adding a little software. Lines 1590, 1600, and 1610 operate according to the reverse-loop-polarity logic shown in fig. 7 to control the signals both in and out of block 12. Obviously, these lines would have to be changed to reflect whichever block or blocks are reversing blocks on your railroad.

Line 1630 sets the *Y*ard *A*pproach LED variable YA, so that the LED lights to tell the yardmaster a train is approaching. This line initializes YA to off and then, if the Direction of Travel in block 2 is WestBound, resets YA to on. Again, change the determining block number to suit your railroad.

Line 1650 is a GOTO instruction for use of a CRT status display as part of your real-time loop. I'll cover that subroutine later, but if you don't want this feature simply leave out the line.

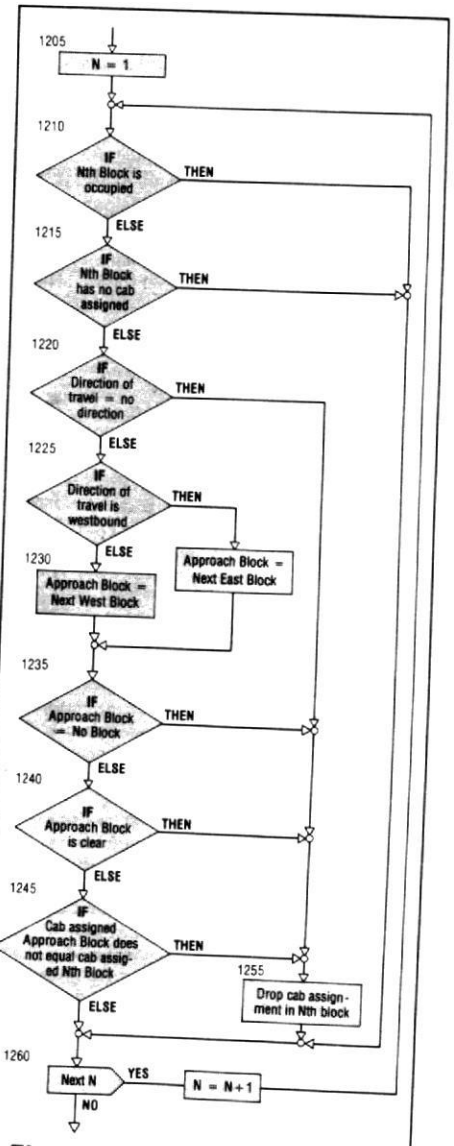
Line 1670 gives you a hardware indication for timing your software. X increments by 1 each time through the real-time loop, so that the LSB of X alternates between 0 and 1. By setting X0 = X AND W1, X0 takes on the value of the LSB of X. X0 can be sent as output to an LED (see figs. 2 and 3 in Part 9), and the diode will change state, flash on or off, each time the program loops. How long the LED stays on or off shows you the iteration time of your real-time loop.

OUTPUT

The last part of our main program sends the calculated variables to the railroad as output. As for inputs, we'll use repetitions of a standard procedure to pack variables into bytes and send output to the railroad a byte at a time.

Taking card 3, port A as an example, the first step is to initialize *O*utput *B*yte OB to zero. The bit-packing instruction format is OB = the name of the variable to be packed multiplied by the position constant of the variable's LSB, then ORed with the previous value of OB. This instruction format is repeated once for each variable to go into the byte.

At first glance line 1700 appears more complicated than you might expect. It packs variable CN(1), the Cab Number assigned to block 1. The input lines on CRC and CDC cards are active high,



1205
N = 1
1210
IF Nth Block is occupied
THEN
ELSE
1215
IF Nth Block has no cab assigned
THEN
ELSE
1220
IF Direction of travel = no direction
THEN
ELSE
1225
IF Direction of travel is westbound
THEN
ELSE
1230
Approach Block = Next West Block
Approach Block = Next East Block
1235
IF Approach Block = No Block
THEN
ELSE
1240
IF Approach Block is clear
THEN
ELSE
1245
IF Cab assigned Approach Block does not equal cab assigned Nth Block
THEN
ELSE
1255
Drop cab assignment in Nth block
1260
Next N
YES
NO
N = N+1

Fig. 4
LOGIC FLOW FOR DROPPING CAB ASSIGNMENTS

positive logic, while Output Card output transistors are active low — when on they in effect connect their output lines to ground, 0V. This negative logic is best for most C/MRI loads, but it has to be reversed for the CN( ) outputs. The correct values are obtained by outputting 7 − CN(1), instead of CN(1) itself.

Following that format, line 1700 packs the cab number for block 1 starting at bit position 0, and line 1710 the cab number for block 2 starting at bit position 3 — cab numbers 0-4 or 0-7 take 3 bits each. Lines 1720 and 1730 pack the settings for the east-end signals for blocks 1 and 2 into bit positions 6 and 7.

Once all the bits are packed, a POKE (or OUT) instruction sends the Output Byte to the appropriate port of an Output Card. The remainder of the output lines follow the same procedure. Only the variable to be output is changed along with the bit position and card/port address, all following the I/O tables.

To close off the program, line 2410 in-

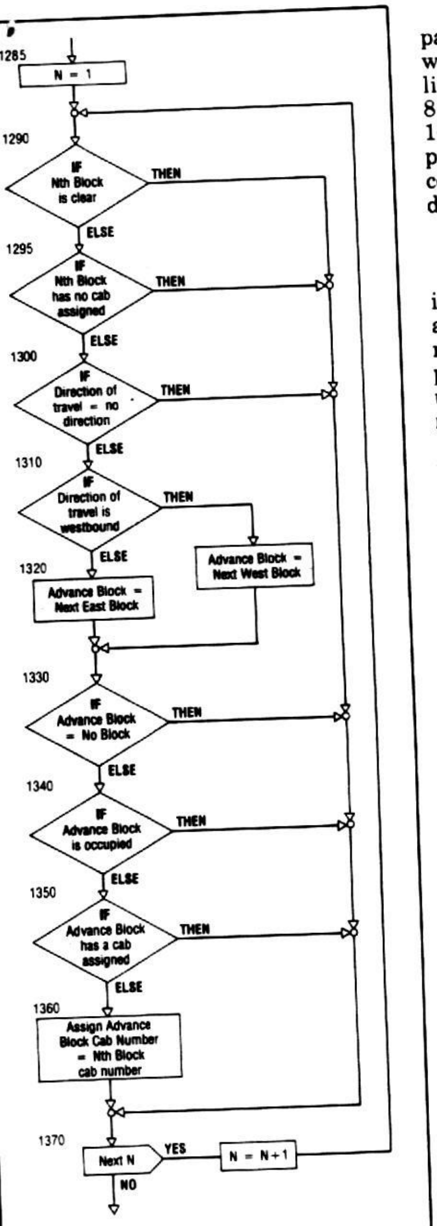

1285
N = 1
1290
IF Nth Block is clear
THEN
ELSE
1295
IF Nth Block has no cab assigned
THEN
ELSE
1300
IF Direction of travel = no direction
THEN
ELSE
1310
IF Direction of travel is westbound
THEN
ELSE
Advance Block = Next West Block
1320
Advance Block = Next East Block
1330
IF Advance Block = No Block
THEN
ELSE
1340
IF Advance Block is occupied
THEN
ELSE
1350
IF Advance Block has a cab assigned
THEN
ELSE
1360
Assign Advance Block Cab Number = Nth Block cab number
1370
Next N
YES
N = N+1
NO

Fig. 5 LOGIC FLOW FOR ASSIGNING CABS TO BLOCKS

crements the loop counter, and resets it to 0 when it exceeds 31000. Line 2420 is only for using the simulated-input test subroutine I'll cover later. The last main program statement is line 2440: it completes the real-time loop with a GOTO branch back to line 450, to repeat the loop.

APPLE I/O

I've used AND and OR logic in packing and unpacking I/O bytes for simplicity and speed. I've tested this software on every computer covered in C/MRI Part 3, and it works for all except the Apples. Applesoft BASIC and Apple's Integer BASIC both use simplified functions rather than true Boolean AND and OR logic.

The simplified logic gives the wrong answers from AND and OR operations with the Apple BASICs. You can test your machine by entering two BASIC commands:

```
A = 5 AND 2
PRINT A
```

If you get A = 0 the packing and unpacking operations explained above will work. If you get A = 1 or TRUE, replace lines 450 through 1000 as shown in fig. 8, and change every OR to + in lines 1680 through 2390. This version of the packing logic takes a few more lines of code and more execution time, but it does let you use the Apple BASICs.

CRT STATUS DISPLAY

Once you have the main CCC program, it's easy to display block occupancy, cab assignments and directions, and more right on your computer's CRT. The display can be simple or complex, up to full use of color graphics with a track schematic like a modern CTC dispatcher's.

However, graphics capabilities vary from computer to computer, and graphics commands are anything but standard. Also, the display would have to be tailored for each track plan. Full use of graphics is therefore beyond the scope of the C/MRI series, but I can show you a subroutine for a general-purpose, tabular status display. That way you can modify and upgrade your CRT display without changing the main program.

Figure 9 shows typical screen output in a tabular CRT display for our example layout. This is a monochrome display, but for sake of explanation, parts that don't change are on a colored background, and those that are updated on white.

The top row includes the program title and "ITERATION =," followed by the current value of X, the loop counter variable. The next row is the display title, and the next lists Block Numbers 1-12.

The fixed parts of succeeding rows contain the symbols for block occupation, cab number assiged, direction of travel, signal east, signal west, next-east block, next-west block, and spur position. This is followed by fixed title lines for turnouts 0-6 and cab direction switches 1-4.

The last line isn't part of the display subroutine, but is the data input request line for the simulation test subroutine I'll cover later. I've included it in fig. 9 so you can see how it will appear in conjunction with the status display.

The status printouts, on the white background, are updated as required each time the program completes an iteration of the real-time loop. The block occupancy indicators ("1s" in fig. 9) can be any graphic symbol in your computer's repertoire: a solid rectangle, for example, would avoid confusion with cab 1.

The cab-number line shows assignments by number, with a blank for no assignment. Greater-than and less-than symbols serve as arrows to show direction of travel, with blanks for no direction. Each signal is displayed as a blank if green and as an "R" if red.

The next blocks east and west are indicated by number, with 99 used for no block. This next-block display is handy for debugging, but once your main program's next-block logic is working this display isn't much use and can be deleted.

For the spur and route turnout status I chose a blank if normal and a 1 if reversed. The cab direction display is like the direction of travel for each block.

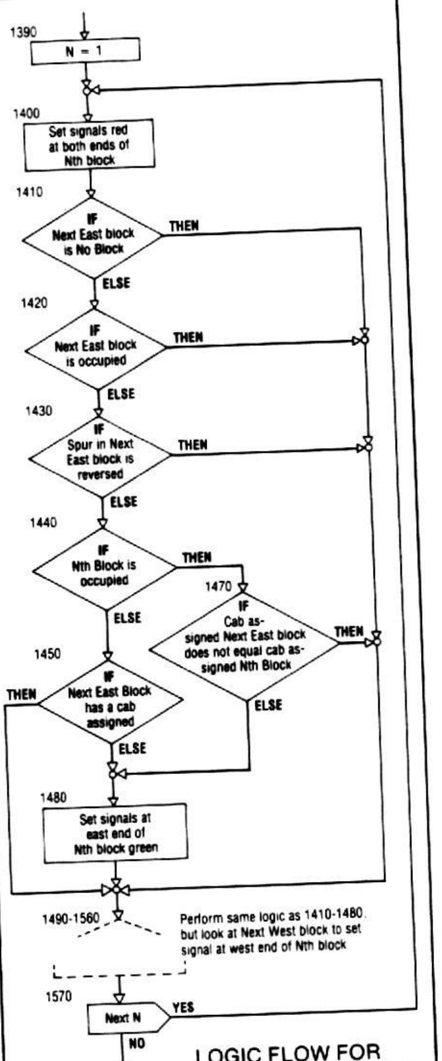

1390
N = 1
1400
Set signals red at both ends of Nth block
1410
IF Next East block is No Block
THEN
ELSE
1420
IF Next East block is occupied
THEN
ELSE
1430
IF Spur in Next East block is reversed
THEN
ELSE
1440
IF Nth Block is occupied
THEN
ELSE
1470
IF Cab assigned Next East block does not equal cab assigned Nth Block
THEN
ELSE
1450
IF Next East Block has a cab assigned
THEN
ELSE
1480
Set signals at east end of Nth block green
1490-1560
Perform same logic as 1410-1480 but look at Next West block to set signal at west end of Nth block
1570
Next N
YES
NO

Fig. 6 LOGIC FLOW FOR TRACKSIDE SIGNALS

In fig. 9, we see that cab 3 is running an eastbound train in blocks 2 and 3. The computer has already assigned block 5, which is clear, to cab 3, and signal SE(3) is green to let cab 3's train enter block 5. Cab 3's train has cleared the trailing block, so block 1's cab assignment has automatically been dropped. That leaves the block available for reassignment.

Cab 2 has a train switching in block 4, the passing siding, with the spur off the siding in reverse position. Note that the signals at both ends of the passing track are red, restricting cab 2's train to the siding as it works the local industries.

Cab 1 has a train in block 10, moving toward reverse-loop block 12. The track polarities are correct, as shown by the direction arrows as well as by signal SE(10) being green for entry into block 12.

Once cab 1's train enters block 12, the computer will automatically assign cab 1 to block 11. When his train is entirely in block 12, the cab-1 engineer will reverse the cab direction switch to westbound, allowing the CCC program to clear signal SE(12). That shows the cab-1 engineer that his train can enter block 11. Turnout 5 is reversed, routing cab 1's train into yard block 9.

RS	DT(12) = RS+1	Polarity set for movement
0	1	Block 11 to block 10
1	2	Block 10 to block 11

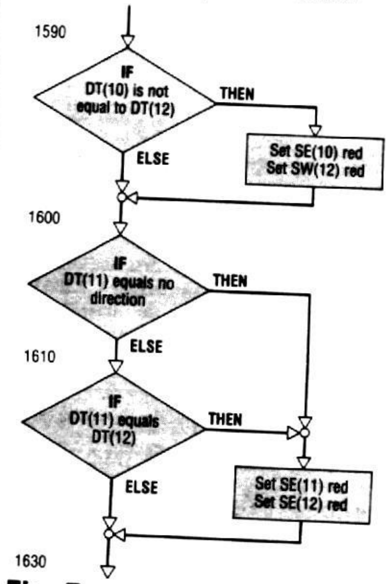


Fig. 7

LOGIC FLOW FOR REVERSING BLOCK SIGNALS

CRT DISPLAY SUBROUTINES

You can add as many extra features as you like to your CCC program, but it's important to keep the execution time of the real-time loop as short as possible, because that is the index of how fast the computer reacts to changes on the railroad. You want the CRT display program to update only things that have changed since the last iteration, and let anything that hasn't changed stay the same rather than being written over.

The title information doesn't change, so I've handled that with a separate display initialization subroutine invoked just once, before the beginning of the real-time loop. A second subroutine is invoked each time through the loop to provide updates as necessary.

Figure 10 lists both subroutines for the CRT display. As is typical of display software, some lines are necessarily peculiar to the MicroSoft BASIC dialect of the Heath/Zenith H-8 and H-89 machines. Even for different computers, this should give you a good start, and much of the code will be the same.

In the display initialization subroutine, line 3020 defines a user function for my Heathkit terminal to move the cursor to a position denoted by coordinates X (column number) and Y (row number) — 1,1 is the upper left corner of the screen. Lines 3040, 3050, and 3060 define the special display string characters. Line 3080 is needed on my computer to avoid automatic insertion of a carriage return and line feed any time output exceeds 80 characters.

Line 3090 is used to clear the screen on my H8; substitute the statement that does the same thing on your machine. Lines 3100 through 3160 are PRINT statements for the fixed titles. Line 3170 completes the subroutine with a GOTO back to the next line after the GOTO 3020 statement that called the subroutine from the main program (GOSUB/RETURN is slower).

In the update subroutine, all PRINTs are made in the direct cursor control mode. The semicolons in the print lines suppress the carriage-returns/line-feeds at the end of each print sequence.

For example, line 3510 drives the cursor to row 1, column 48 at which point it prints out the loop counter variable X. The subroutine then works its way across the screen looking at first for all variables associated with block 1, then block 2, up to MB. If, and only if, it finds a value changed from the last iteration, it jumps the cursor to the required position for the printout. Each printout is followed by an update of the saved last-iteration variable, giving it the value just printed.

Since the next-east and -west block numbers can be either one or two digits, lines 3650 and 3660 are needed to maintain the column spacing. Line 3650 sets the alignment using the intrinsic LEN function to determine whether a given block number has one or two digits, then 3660 adds leading blanks to X$ so the printout of line 3670 is always three characters long. Lines 3690 and 3700 do the same for the next-west block number.

Once all the block updates are printed, the same procedures are used to print updates for the turnouts and cab direction switches, beginning with line 3750.

SIMULATING RAILROAD INPUT

A handy way to check the execution of your real-time CCC program is to simulate the railroad input with the computer's keyboard. You can enter variable changes that would occur in normal operation and see the program's respond on the CRT. Figure 11 lists the subroutine I used to debug the CCC main program.

Line 4020 directs the cursor to row 16, column 1 and prints the prompt message "ENTER VA$,NI,VA." Again note the semicolon used after each PRINT command to suppress the automatic carriage-return/line-feed. Following the INPUT command in line 4030 the program waits for you to enter variable changes.

Simply key in any of the two-character alpha codes in lines 4050 through 4100 (such as BO for block occupancy), followed by a comma and the array index number (such as 1 for block 1), followed by another comma and the new value you want to give the variable (0, clear, or 1, occupied), followed by a carriage return. The entry format is fixed, but you can loop through as many variable changes as you wish.

When you are done changing variables

```
450  REM **Read inputs and separate
            out bit areas**
460  REM GOTO 1Ø1Ø '&&&& LINE FOR TEST
            TO BYPASS C/MRI INPUT
47Ø  IB=PEEK(SA)      'Card 1 port A
471  DR=Ø:  IC=Ø:  RS=Ø
48Ø  IF IB>W5 THEN RS=1:  IB=IB-B5
49Ø  IF IB>W4 THEN IC=B2:  IB=IB-B4
491  IF IB>W3 THEN IC=IC+B1:  IB=IB-B3
492  IF IB>W2 THEN IC=IC+BØ:  IB=IB-B2
5ØØ  IF IB>W1 THEN DR=1:  IB=IB-B1
51Ø  AS=IB
52Ø  REM **Read port B later only if
            ASign/DRop pressed**
53Ø  T1=PEEK(SA+2)     'Card 1 port C
54Ø  IB=PEEK(SA+4)     'Card 2 port A
541  FOR N=1 TO MB: BO(N)=CL: NEXT N
55Ø  IF IB>W7 THEN BO(8)=OC: IB=IB-B7
56Ø  IF IB>W6 THEN BO(7)=OC: IB=IB-B6
57Ø  IF IB>W5 THEN BO(6)=OC: IB=IB-B5
58Ø  IF IB>W4 THEN BO(5)=OC: IB=IB-B4
59Ø  IF IB>W3 THEN BO(4)=OC: IB=IB-B3
6ØØ  IF IB>W2 THEN BO(3)=OC: IB=IB-B2
61Ø  IF IB>W1 THEN BO(2)=OC: IB=IB-B1
62Ø  BO(1)=IB
63Ø  IB=PEEK(SA+5)     'Card 2 port B
631  FOR N=1 TO MB: SP(N)=TN: NEXT N
64Ø  IF IB>W7 THEN SP(9)=TR: IB=IB-B7
65Ø  IF IB>W6 THEN SP(4)=TR: IB=IB-B6
66Ø  IF IB>W5 THEN SP(3)=TR: IB=IB-B5
67Ø  IF IB>W4 THEN SP(2)=TR: IB=IB-B4
68Ø  IF IB>W3 THEN BO(12)=OC: IB=IB-B3
69Ø  IF IB>W2 THEN BO(11)=OC: IB=IB-B2
7ØØ  IF IB>W1 THEN BO(1Ø)=OC: IB=IB-B1
71Ø  BO(9)=IB
72Ø  IB=PEEK(SA+6)     'Card 2 port C
721  FOR N=1 TO 4:  CD(N)=ND: NEXT N
73Ø  IF IB>W7 THEN CD(4)=EB: IB=IB-B7
731  IF IB>W6 THEN CD(4)=WB: IB=IB-B6
74Ø  IF IB>W5 THEN CD(3)=EB: IB=IB-B5
741  IF IB>W4 THEN CD(3)=WB: IB=IB-B4
75Ø  IF IB>W3 THEN CD(2)=EB: IB=IB-B3
751  IF IB>W2 THEN CD(2)=WB: IB=IB-B2
76Ø  IF IB>W1 THEN CD(1)=EB: IB=IB-B1
761  IF IB=W1 THEN CD(1)=WB
77Ø  REM **If assign button pressed
            perform assign**
78Ø  IF AS<>PP GOTO 85Ø
79Ø  IB=PEEK(SA+1)     'Card 1 port B
791  BL=Ø:    BH=Ø
8ØØ  IF IB>W7 THEN BH=B3:  IB=IB-B7
8Ø1  IF IB>W6 THEN BH=BH+B2:  IB=IB-B6
8Ø2  IF IB>W5 THEN BH=BH+B1:  IB=IB-B5
8Ø3  IF IB>W4 THEN BH=BH+BØ:  IB=IB-B4
81Ø  IF IB>W3 THEN BL=B3:  IB=IB-B3
811  IF IB>W2 THEN BL=BL+B2:  IB=IB-B2
812  IF IB>W1 THEN BL=BL+B1:  IB=IB-B1
813  IF IB=W1 THEN BL=BL+BØ
82Ø  BI=BL+(1Ø*BH)
83Ø  CN(BI)=IC
84Ø  REM ** If drop button pressed
            perform drop**
85Ø  IF DR<>PP GOTO 92Ø
86Ø  IB=PEEK(SA+1)     'Card 1 port B
861  BL=Ø:  BH=Ø
87Ø  IF IB>W7 THEN BH=B3:  IB=IB-B7
871  IF IB>W6 THEN BH=BH+B2:  IB=IB-B6
872  IF IB>W5 THEN BH=BH+B1:  IB=IB-B5
873  IF IB>W4 THEN BH=BH+BØ:  IB=IB-B4
88Ø  IF IB>W3 THEN BL=B3:  IB=IB-B3
881  IF IB>W2 THEN BL=BL+B2:  IB=IB-B2
882  IF IB>W1 THEN BL=BL+B1:  IB=IB-B1
883  IF IB=W1 THEN BL=BL+BØ
89Ø  BI=BL+(1Ø*BH)
9ØØ  CN(BI)=NC
91Ø  REM**If turnout changed decode
            update next blocks**
92Ø  IF TI=S1 THEN GOTO 115Ø
93Ø  S1=T1
931  FOR N=1 TO 6:  TU(N)=TR: NEXT N
94Ø  IF T1>W6 THEN TU(6)=TN: T1=T1-B6
95Ø  IF T1>W5 THEN TU(5)=TN: T1=T1-B5
96Ø  IF T1>W4 THEN TU(4)=TN: T1=T1-B4
97Ø  IF T1>W3 THEN TU(3)=TN: T1=T1-B3
98Ø  IF T1>W2 THEN TU(2)=TN: T1=T1-B2
99Ø  IF T1>W1 THEN TU(1)=TN: T1=T1-B1
1ØØØ TU(Ø)=T1
```

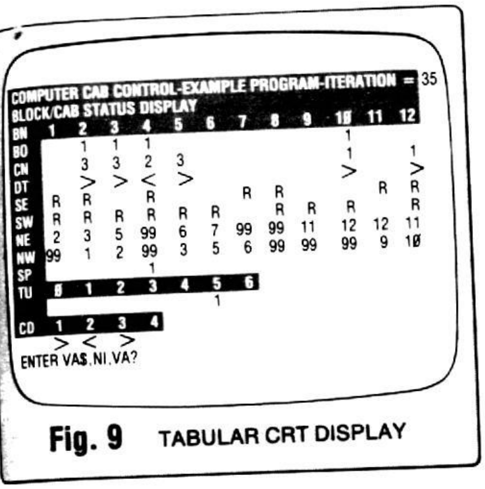
Fig. 8

UNPACKING CODE FOR APPLE BASICS

Replaces lines 450-1000 in fig. 2

Also change each OR to a + (plus sign) in lines 1680-2390 in fig. 2

```
3ØØØ
3Ø1Ø
3Ø2Ø
3Ø3Ø
3Ø4Ø
3Ø5Ø
3Ø6Ø
3Ø7Ø
3Ø8Ø
3Ø9Ø
31ØØ
311Ø
312Ø
313Ø
314Ø
315Ø
316Ø
317Ø
```



```
COMPUTER CAB CONTROL-EXAMPLE PROGRAM-ITERATION = 35
BLOCK/CAB STATUS DISPLAY
BN  1   2   3   4   5   6   7   8   9  1Ø  11  12
BO      1   1   1                       1
CN      3   3   2   3                   1       1
DT      >   >   <   >                   >       >
SE  R   R       R           R   R           R   R
SW  R   R   R   R   R   R       R   R   R       R
NE  2   3   5  99   6   7  99  99  11  12  12  11
NW 99   1   2  99   3   5   6  99  99  99   9  1Ø
SP              1
TU  Ø   1   2   3   4   5   6
                        1
CD  1   2   3   4
    >   <   >
ENTER VA$,NI,VA?
```

Fig. 9 TABULAR CRT DISPLAY

for a given iteration, enter EN for *EN*d followed by any two "dummy" numbers, and line 4040 branches to line 4120. That turns control back to the main program at the line just after the GOSUB 4020 line which invoked the test subroutine.

You can use keyboard input to "move" imaginary trains around your railroad, assign or drop cabs, set turnouts, and change direction switches, then watch as the CRT displays the results of your actions in automatic block/cab assignments, changing signals, and so on. Remember, however, that to use this test subroutine you MUST include the GOTO statement at line 460 in the main program. That bypasses the normal C/MRI input so it won't get mixed up with keyboard input.

PROGRAM TIMING

BASIC programming is easy to use, and with it you can get a lot of on-line software running in short order. It's disadvantage is that it is slow in execution. If you expand the example program for a fairly large railroad and use the CRT display option, you may find that the loop runs too slowly for good CCC performance. Even so, an all-BASIC program is a good way to start, and once that's debugged there are ways to speed it up.

The speed of BASIC processing can vary widely with different versions of the langauge and different computers. My H8 has an Intel 8080A processor and some static RAM, and is quite slow by today's standards. Running on the H8, the program in fig. 2 looped in about 3.2 seconds. That's a tolerable response time, acceptable in the worst case.

For CCC that means that when a train enters a new block, the signal would drop to red and the cab would be assigned to the next advance block some time between 0 and 3.2 seconds later. The exact response time would depend on where the program was in its loop at the instant the train entered the new block, and on average it would probably be close to 2.2 seconds. When I added the CRT-display-update subroutine in fig. 10, the iteration time increased to about 4.7 seconds.

My friend Bruce Wahl's newer Heath H89 has a Z80 processor, and the same program's times on it were 2.6 seconds without the display subroutine and 3.8 seconds with. When he compiled the program with Microsoft's BASCOM compiler software, that cut the H89 iteration times to 1.3 seconds without display and 1.9 seconds with, which is very respectable.

If your loop takes longer, you don't have to go out and buy a faster computer. Even staying entirely with BASIC there are simple measures that can increase processing speed. These include renumbering to lower line numbers, deleting all REM lines, removing division and multiplication by B0 (since it equals 1), and making sure that your most frequently used constants are defined first. Also make sure that your program is branching around unecessary blocks of calculations, as I've already explained.

For the fastest possible operation, you can recode your BASIC program into the assembly language of your computer. As an example, I have the Sunset Valley's CTC program coded in 95 pages worth of 8080A assembly language. It runs in 15K bytes of memory, handles 552 discrete I/O lines, calculates a large number of prototypical signal functions, and drives well over 1000 signal and panel LEDs. Still, an iteration of its real-time loop takes only .045 seconds. It's more than 200 times faster than an equivalent BASIC program, and means that all the I/O and all 55 signal settings are being recalculated 22 times each second!

That's faster than necessary, and an update rate of once each second or two would be fine for most model railroads. If you can't get into that range by streamlining your BASIC, converting small but frequently-repeated parts of a BASIC program to assembly language should give you enough speed. The railroad I/O and the bit unpacking/packing operations are excellent places to start. These are repetitive and slow in BASIC, and they make ideal assembly language subprograms. For most railroads, their conversion alone should provide enough speed.

Unfortunately I can't cover assembly language here, since it varies from computer to computer. Typically you'll only need to learn a couple dozen instruction types in assembly language to have all you'll need to speed up your program.

Another way would be to use a high-order, truly-compiled language like FORTRAN, PASCAL, C, or FORTH. If a high-order language for your machine permits operations at the bit and byte level it will really speed things up. However, compilers can be expensive and they take time to learn. Unless you really want to try one, I think your best bet is to stick with BASIC and supplement it with a bit of assembly language as I've suggested.

That's all you need to use Computer Cab Control on your railroad. We'll skip next month, and I'll be back in February with another C/MRI application, a more detailed, general-purpose signal system. ♁

```
3ØØØ  REM**SUBROUTINE TO INITIALIZE CRT DISPLAY (OPTIONAL)
3Ø1Ø  REM**Define Direct Cursor to X,Y position function**
3Ø2Ø  DEF FNDC$(X,Y)=CHR$(27)+"Y"+CHR$(31+X)+CHR$(31+Y)
3Ø3Ø  REM**Define CRT Display symbols**
3Ø4Ø  DD$(Ø)=" ":  DD$(1)="<":  DD$(2)=">":  DO$(Ø)="1"
3Ø5Ø  DO$(1)=" ":  DC$(Ø)=" ":  DC$(1)="1":  DC$(2)="2"
3Ø6Ø  DC$(3)="3":  DC$(4)="4":  DR$(Ø)=" ":  DR$(1)="R"
3Ø7Ø  REM**Initialize CRT status display**
3Ø8Ø  WIDTH 255
3Ø9Ø  PRINT CHR$(27)+"E";      'clear screen
31ØØ  PRINT "COMPUTER CAB CONTROL-EXAMPLE PROGRAM-ITERATION="
311Ø  PRINT "BLOCK/CAB STATUS DISPLAY"
312Ø  PRINT "BN  1  2  3  4  5  6  7  8  9 1Ø 11 12"
313Ø  PRINT "BO":  PRINT "CN":  PRINT "DT":  PRINT "SE"
314Ø  PRINT "SW":  PRINT "NE":  PRINT "NW":  PRINT "SP"
315Ø  PRINT "TU  Ø  1  2  3  4  5  6":  PRINT
316Ø  PRINT "CD  1  2  3  4"
317Ø  GOTO 35Ø
```

```
35ØØ  REM **SUBROUTINE TO UPDATE CRT DISPLAY (OPTIONAL)**
351Ø  PRINT FNDC$(1,48);X; 'Print loop counter for timing
352Ø  REM **Update status for each block CRT display**
353Ø  FOR N=1 TO NB
354Ø  IF BO(N)=BS(N) THEN GOTO 356Ø
355Ø  PRINT FNDC$(4,2+(3*N));DO$(BO(N));: BS(N)=BO(N)
356Ø  IF CN(N)=CS(N) THEN GOTO 358Ø
357Ø  PRINT FNDC$(5,2+(3*N));DC$(CN(N));: CS(N)=CN(N)
358Ø  IF DT(N)=DS(N) THEN GOTO 36ØØ
359Ø  PRINT FNDC$(6,2+(3*N));DD$(DT(N));: DS(N)=DT(N)
36ØØ  IF SE(N)=ES(N) THEN GOTO 362Ø
361Ø  PRINT FNDC$(7,2+(3*N));DR$(SE(N));: ES(N)=SE(N)
362Ø  IF SW(N)=WS(N) THEN GOTO 364Ø
363Ø  PRINT FNDC$(8,2+(3*N));DR$(SW(N));: WS(N)=SW(N)
364Ø  IF NE(N)=BE(N) THEN GOTO 368Ø
365Ø  X$=STR$(NE(N))
366Ø  IF LEN(X$)<3 THEN X$=" "+X$: GOTO 366Ø
367Ø  PRINT FNDC$(9,(3*N));X$;: BE(N)=NE(N)
368Ø  IF NW(N)=BW(N) THEN GOTO 372Ø
369Ø  X$=STR$(NW(N))
37ØØ  IF LEN(X$)<3 THEN X$=" "+X$: GOTO 37ØØ
371Ø  PRINT FNDC$(1Ø,(3*N));X$;: BW(N)=NW(N)
372Ø  IF SP(N)=SS(N) THEN GOTO 374Ø
373Ø  PRINT FNDC$(11,2+(3*N));DO$(SP(N));: SS(N)=SP(N)
374Ø  NEXT N
375Ø  FOR N=Ø TO 6
376Ø  IF TU(N)=TS(N) THEN GOTO 378Ø
377Ø  PRINT FNDC$(13,5+(3*N));DO$(TU(N));: TS(N)=TU(N)
378Ø  NEXT N
379Ø  FOR N=1 TO 4
38ØØ  IF CD(N)=CC(N) THEN GOTO 382Ø
381Ø  PRINT FNDC$(15,2+(3*N));DD$(CD(N));: CC(N)=CD(N)
382Ø  NEXT N
383Ø  GOTO 167Ø
```

Fig. 10 OPTIONAL CRT DISPLAY SUBROUTINES

```
4ØØØ  REM**&&&&SUBROUTINE USED
           FOR STANDALONE TESTING**
4Ø1Ø  REM**Provides simulated C/MRI
           inputs via keyboard**
4Ø2Ø  PRINT FNDC$(16,1);: PRINT
           "ENTER VA$,NI,VA";
4Ø3Ø  INPUT VA$,NI,VA
4Ø4Ø  IF VA$="EN" THEN GOTO 412Ø
4Ø5Ø  IF VA$="BO" THEN BO(NI)=VA
4Ø6Ø  IF VA$="CD" THEN CD(NI)=VA
4Ø7Ø  IF VA$="TU" THEN TU(NI)=VA
4Ø8Ø  IF VA$="CN" THEN CN(NI)=VA
4Ø9Ø  IF VA$="SP" THEN SP(NI)=VA
41ØØ  IF VA$="RS" THEN RS=VA
411Ø  GOTO 4Ø2Ø
412Ø  RETURN
```

Fig. 11
OPTIONAL
KEYBOARD-INPUT TEST

C/MRI-11

The C/MRI: A computer/model railroad interface

Part 11: Computerized signals — prototype and model systems, and I/O wiring

BY BRUCE CHUBB

A FUNCTIONAL trackside signal system is a natural application for the C/MRI. Signals can do a lot to increase the realism of your model railroad, to smooth out the rough edges of its operation, and to add color and animation.

Remember the photos on the first page of C/MRI Part 6, in the July 1985 MODEL RAILROADER? Consider this scenario:

A high-tonnage freight pulled by a set of RS units patiently waits on the siding, held by the red dwarf signal until an opposing train can clear. The dispatcher, miles away in our imagination, has set up the meet on his Centralized Traffic Control or CTC panel and is watching the colored lights on his console which track the oncoming train's progress.

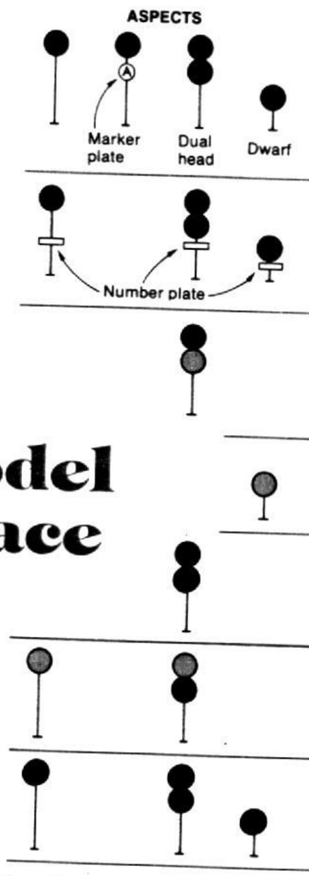
Once the opposing train is clear, the dispatcher throws the turnout and signal levers on the panel and presses the code-start button below them. The switch points swing over, the dwarf clears to green, the engineer revs his Alcos, and the waiting freight pulls out onto the main track. The meet is carried out smoothly, realistically, and without a word being spoken.

Prototypical signals, a realistic setting, reliable equipment, and a crew that strives for perfection: we put these all together at every Sunset Valley operating session to create railroading with a capital R. You can do it on your railroad too, be it large or small.

You can have realistic signaling without a computer, but the computer makes it easier, and easier to enjoy. To appreciate that takes some knowledge of the prototype.

UNDERSTANDING SIGNALS

Railroads use signals to help trains move safely, efficiently, and quickly.



ASPECTS	NAME	INDICATION
	Stop	Stop and stay, "absolute stop"
	Stop and proceed	Stop, then proceed at restricted speed to the next signal — "permissive stop"
	Medium approach or Diverging approach	Proceed at prescribed speed on diverging route
	Restricting	Proceed at restricted speed prepared to stop short of any obstruction
	Medium clear or Diverging clear	Proceed on diverging route at medium speed
	Approach	Proceed prepared to stop at next signal. Reduce to medium speed
	Clear	Proceed

Fig. 1
SIGNAL ASPECTS AND INDICATIONS

These or equivalent aspects may be displayed on any type of signal. Marker and number plates apply only to red and red-over-red aspects; their presence or absence has no effect on less restrictive aspects

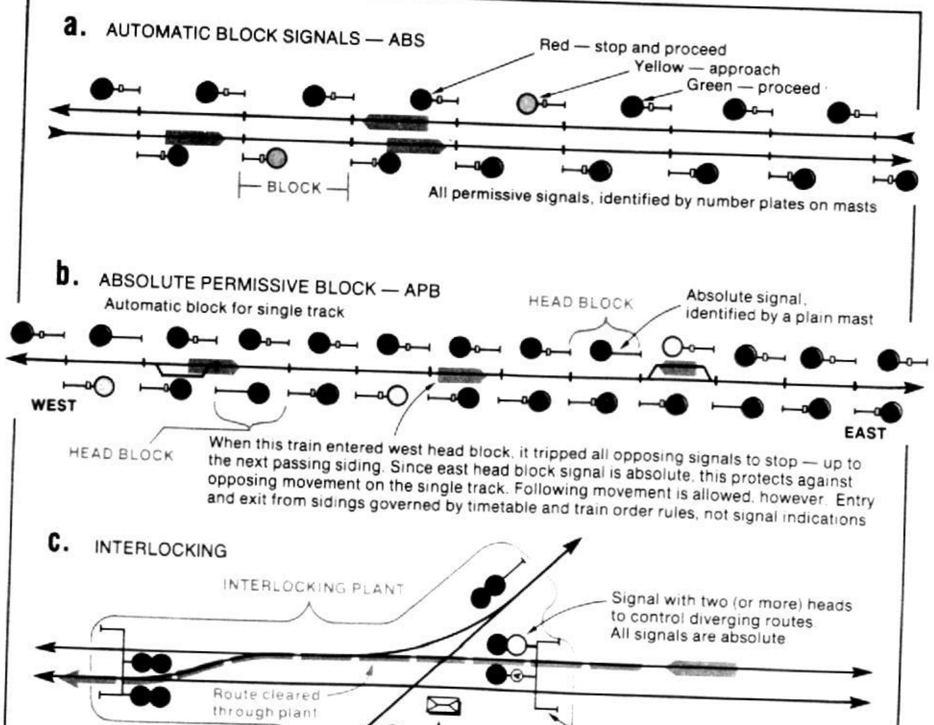

a. AUTOMATIC BLOCK SIGNALS — ABS
Red — stop and proceed
Yellow — approach
Green — proceed
BLOCK
All permissive signals, identified by number plates on masts
b. ABSOLUTE PERMISSIVE BLOCK — APB
Automatic block for single track
HEAD BLOCK
Absolute signal, identified by a plain mast
WEST
EAST
HEAD BLOCK
When this train entered west head block, it tripped all opposing signals to stop — up to the next passing siding. Since east head block signal is absolute, this protects against opposing movement on the single track. Following movement is allowed, however. Entry and exit from sidings governed by timetable and train order rules, not signal indications
c. INTERLOCKING
INTERLOCKING PLANT
Signal with two (or more) heads to control diverging routes. All signals are absolute
Route cleared through plant
Signal bridge
Signals (and turnouts) in plant controlled by operator in local tower, or remotely controlled from another location. Simpler plants may be automated

Fig. 2
BLOCK AND INTER-LOCKING SYSTEMS

Signals allow a higher traffic density, more trains, on a given line than would be safely practical without them. They protect trains following each other, passing through crossings and junctions, and running in opposite directions on a single track. Except that lives and property aren't at stake, they do just as much good on a model railroad.

Prototype signaling and how to model it are bigger subjects than I can explain in detail here. I'll cover just the basics, so you'll see how computerized signals can make your railroad easier and more realistic to operate. For more information on the real thing, see chapters 9 and 10 of my book, *How to Operate Your Model Railroad*, and John Armstrong's *All About Signals*, both published by Kalmbach (MR's publisher).

For immediate reference fig. 1 is a table of some commonly used signal aspects, names, and indications. These can vary from railroad to railroad, or even on different parts of the same railroad. To follow a specific prototype you'll need to learn its practices, but fig. 1 is still a useful starting point.

Broadly speaking, prototype signals fall into two categories:

Block signals maintain braking distance between trains running in one direction on the same track, to prevent rear-end collisions. See fig. 2a. They are also used to protect trains moving in opposite directions on single track, as in fig. 2b. Block signals can be of any type, but in most applications they have only a single head capable of displaying red, yellow, and green, or equivalent aspects.

The red aspect of a block signal generally indicates a "permissive stop," meaning that a train may proceed at restricted speed after stopping, so that a circuit failure won't hold a train at some remote location for hours. Where trains must stop and stay, as at the "headblock" entrances to a stretch of single track, railroads use an "absolute stop" indication. Permissive signals are identified by number plates on their masts; some roads put a marker plate with a letter "A" on the mast of an absolute signal, others define any signal without a number plate as absolute.

The normal aspect of a block signal is green, indicating at least two clear or unoccupied blocks ahead. To save bulbs and batteries, some railroads use "approach lighting," which keeps the signal dark until a train enters the block that it faces.

Interlocking signals control movement through junctions and crossings, through danger points such as drawbridges, and sometimes through passing sidings on single track and crossovers on multiple tracks. See fig. 2c. The signals and turnouts within the "plant" are interlocked, connected to each other mechanically and/or electrically so that simultaneous train movements on conflicting routes are not allowed. The normal aspect of interlocking signals is red, and they must be deliberately cleared to allow a train into the plant.

Interlocking signals may be of any type, but typically have two or more heads in vertical alignment on a mast. The topmost head governs movement on the main route, and the lower head(s) the diverging route(s). Dwarf or "pot" signals near rail level may also be used.

Interlocking signals are always absolute, but usually an all-red aspect on two or more heads is defined as an absolute-stop indication and no other marking is necessary.

A CTC system like the one I described at the start of this installment combines interlocking and block signals. The dispatcher-controlled turnouts and their adjacent signals form a series of remote-control interlocking plants, while long stretches of track between the control points or "OS sections" are protected by automatic block signals. See fig. 3.

On CTC railroad districts the dispatcher-controlled signals take precedence over timetables and train orders, and trains run or wait according to the signal indications. CTC is a communication as well as a safety system and eliminates a lot of the telephone conversation that would otherwise be needed to dispatch trains efficiently even with automatic block signals.

SIGNALING EXAMPLES

Because CTC combines features of both block and interlocking signals, I'll show you how to operate a CTC signal system with the C/MRI. Once you understand that, you'll have the basic knowledge you'll need to duplicate other kinds of railroad signaling as well.

As you might expect, railroads use computers to operate today's CTC systems, and dispatchers work at keyboards watching graphic "track model" displays on CRTs. It's possible to model such systems with the C/MRI, but because graphics commands aren't standardized, that's beyond the scope of this series. I'll explain how to model CTC with the earlier kind of panel console or "machine" that I use on the Sunset Valley. I've described the SV CTC machine in the January 1984 MR and covered dispatching by CTC

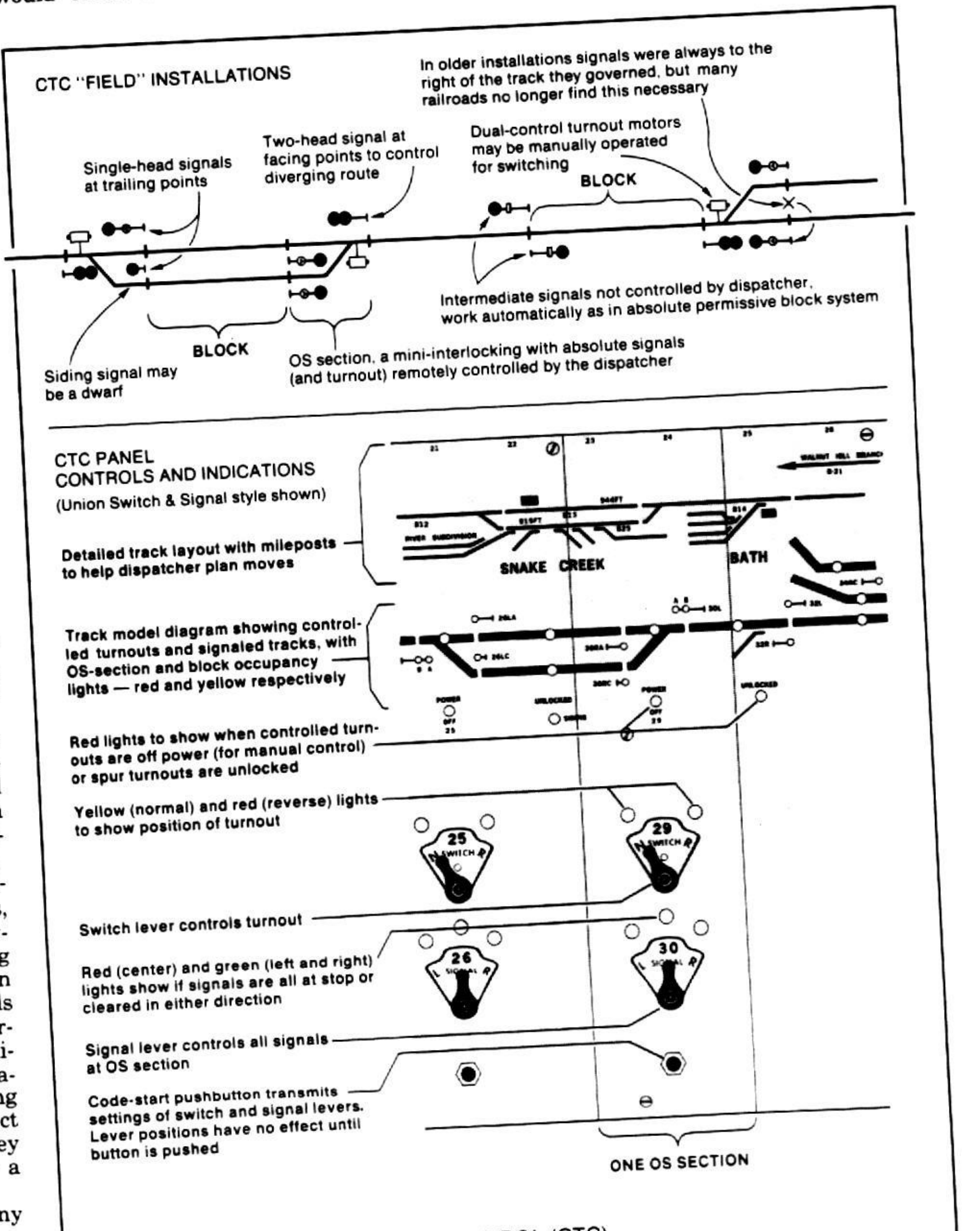

CTC "FIELD" INSTALLATIONS
In older installations signals were always to the right of the track they governed, but many railroads no longer find this necessary
Single-head signals at trailing points
Two-head signal at facing points to control diverging route
Dual-control turnout motors may be manually operated for switching
BLOCK
Intermediate signals not controlled by dispatcher, work automatically as in absolute permissive block system
Siding signal may be a dwarf
BLOCK
OS section, a mini-interlocking with absolute signals (and turnout) remotely controlled by the dispatcher
CTC PANEL CONTROLS AND INDICATIONS (Union Switch & Signal style shown)
Detailed track layout with mileposts to help dispatcher plan moves
SNAKE CREEK
BATH
Track model diagram showing controlled turnouts and signaled tracks, with OS-section and block occupancy lights — red and yellow respectively
Red lights to show when controlled turnouts are off power (for manual control) or spur turnouts are unlocked
Yellow (normal) and red (reverse) lights to show position of turnout
25
29
Switch lever controls turnout
Red (center) and green (left and right) lights show if signals are all at stop or cleared in either direction
26
30
Signal lever controls all signals at OS section
Code-start pushbutton transmits settings of switch and signal levers. Lever positions have no effect until button is pushed
ONE OS SECTION

Fig. 3 CENTRALIZED TRAFFIC CONTROL (CTC)

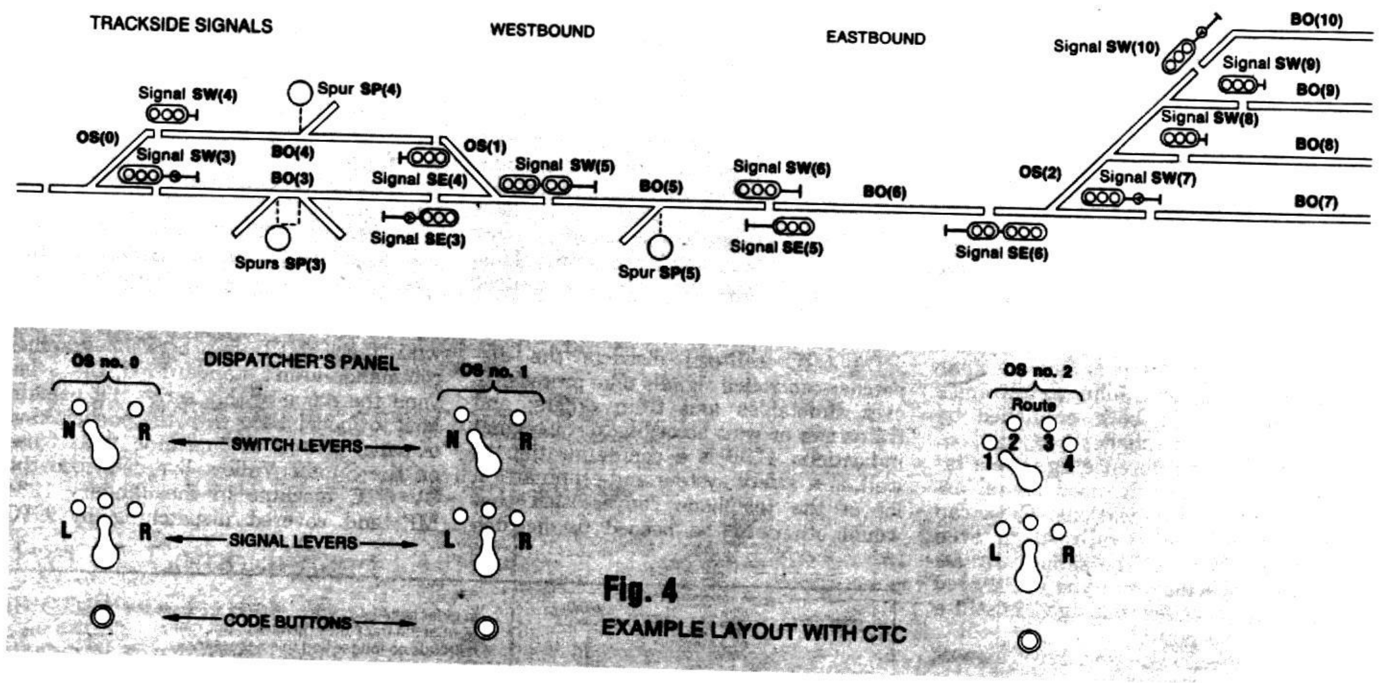


Fig. 4
EXAMPLE LAYOUT WITH CTC

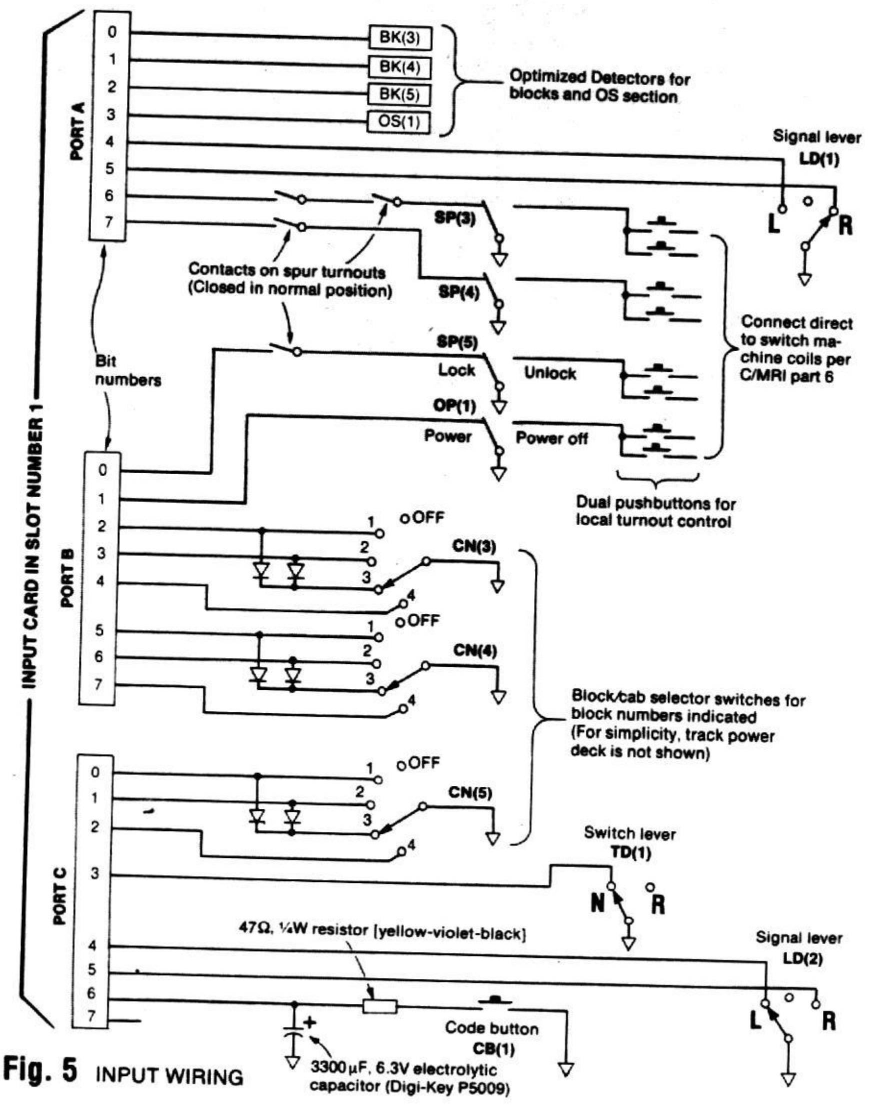

INPUT CARD IN SLOT NUMBER 1
PORT A
PORT B
PORT C
BK(3)
BK(4)
BK(5)
OS(1)
Optimized Detectors for blocks and OS section
Signal lever LD(1)
L
R
SP(3)
SP(4)
SP(5)
Contacts on spur turnouts (Closed in normal position)
Bit numbers
Lock
Unlock
OP(1)
Power
Power off
Connect direct to switch machine coils per C/MRI part 6
Dual pushbuttons for local turnout control
OFF
CN(3)
CN(4)
CN(5)
Block/cab selector switches for block numbers indicated (For simplicity, track power deck is not shown)
Switch lever TD(1)
N
R
47Ω, ¼W resistor [yellow-violet-black]
Signal lever LD(2)
Code button CB(1)
3300µF, 6.3V electrolytic capacitor (Digi-Key P5009)

Fig. 5 INPUT WIRING

indications in chapter 10 of *How to Operate Your Model Railroad.*

Because the SV is set in the early 1950s, I chose not to have the dispatcher work from the CRT. The SV computer plays an invisible role in the background. It does what precomputer prototypes accomplished with thousands of relays, without any relays at all! Software logic does the job.

For this example I'll use part of the same 12-block layout as in the Computer Cab Control installments. Instead of doing the whole thing I'll cover just the triad of interlocking signals at a typical OS section and one block signal. These are the building blocks that make up almost any signal system you might want.

Figure 4 shows the relationship of the trackside signals, turnouts, spur unlocks, and switch motors to the dispatcher panel control levers, LEDs, and code buttons. It covers blocks 3 through 10 of our example layout.

Compare fig. 4 to fig. 1 in C/MRI Part 9 to see the features added to the signal system. These include three-color multihead signals, separate OS sections at controlled turnouts, simulated dual-control switch motors with an off-power position to allow "manual" control by train crews for switching, electrically locked spurs, approach lighting, and a prototypical CTC machine. Although all this could be included with CCC, I'll show wiring for manual cab control so you can see how that's affected by signaling.

EXAMPLE I/O — INPUT

You'd want to plan out all of your I/O requirements on worksheets as shown in C/MRI Part 9, but for this installment I'll cover just what's needed for the examples. Figure 5 shows typical Input Card wiring for the example part of our 12-block railroad, with card 1 handling the inputs. (Card 2, an Output Card, will be covered later.)

I've shown a manual cab control setup for four cabs using five-position rotary switches to make the cab assignments to each block. The computer must read these switches to ensure cab continuity across block gaps before signals are cleared.

If you use your C/MRI for CCC as well as signaling, then in most cases the three lines per block that indicate cab assignments would move from Input Cards to Output Cards as in Part 9. However, you still may want some of these switches hooked up as inputs for terminal yard departure tracks and staging tracks, so operators can simply dial up whatever cab they want the computer to put in control of a given train. In these cases the three lines per switch would drive the CRCs and CDCs directly in parallel with the C/MRI inputs.

CAB SELECTOR INPUTS

Cab-selector switches are the trickiest inputs to connect, so I've gone into more detail showing various types of switches in fig. 6. For the common two-cab system, simply use a DPDT toggle with a center-off position (fig. 6a). One pole is wired in the conventional manner for track power, and the other has the center terminal wired to logic ground and each of the two end terminals wired to separate but adjacent input lines on a C/MRI Input Card.

Line A is grounded when cab 1 is selected, line B is grounded when cab 2 is selected, and neither line is grounded when neither cab is selected. On the Input Card each input line goes through an inverter, so the software sees logic 1 when the input is grounded and logic 0 when it's open. That way the binary codes of the two lines correspond to the cab numbers: 00 = no cab, 01 = cab 1, and 10 = cab 2.

In fact, two lines can cover three cabs plus the off position as shown in fig. 6b. Use a 2-pole, 4-position rotary switch, plus two diodes. The off, cab 1, and cab 2 positions work as in fig. 6a. In the cab 3 position, the two diodes pull down both lines A and B to achieve the code 11, the binary equivalent for cab 3.

For as many as seven cabs and an off position, three input lines are enough. Figure 6c shows a circuit using diodes. An alternate approach is to use a four-deck switch wired as shown in fig. 6d, or a single digit of a BCD thumbwheel switch as for the BCAP in C/MRI Part 8. The BCD switch is the most straightforward and easiest.

Of course, if you use command control you can drop these cab-number input lines all together, as well as the program lines in the next installment that deal with the CN( ) variables.

OTHER INPUTS

Returning to fig. 5, an SPDT non-center-off toggle is used to lock the spurs in each block. They can be on the CTC panel as on the SV or at the "field" locations as on many prototypes. In unlocked position train crews have direct control of the spur turnouts using the local pushbuttons; in locked position the local control is disabled.

The lock input line to the C/MRI

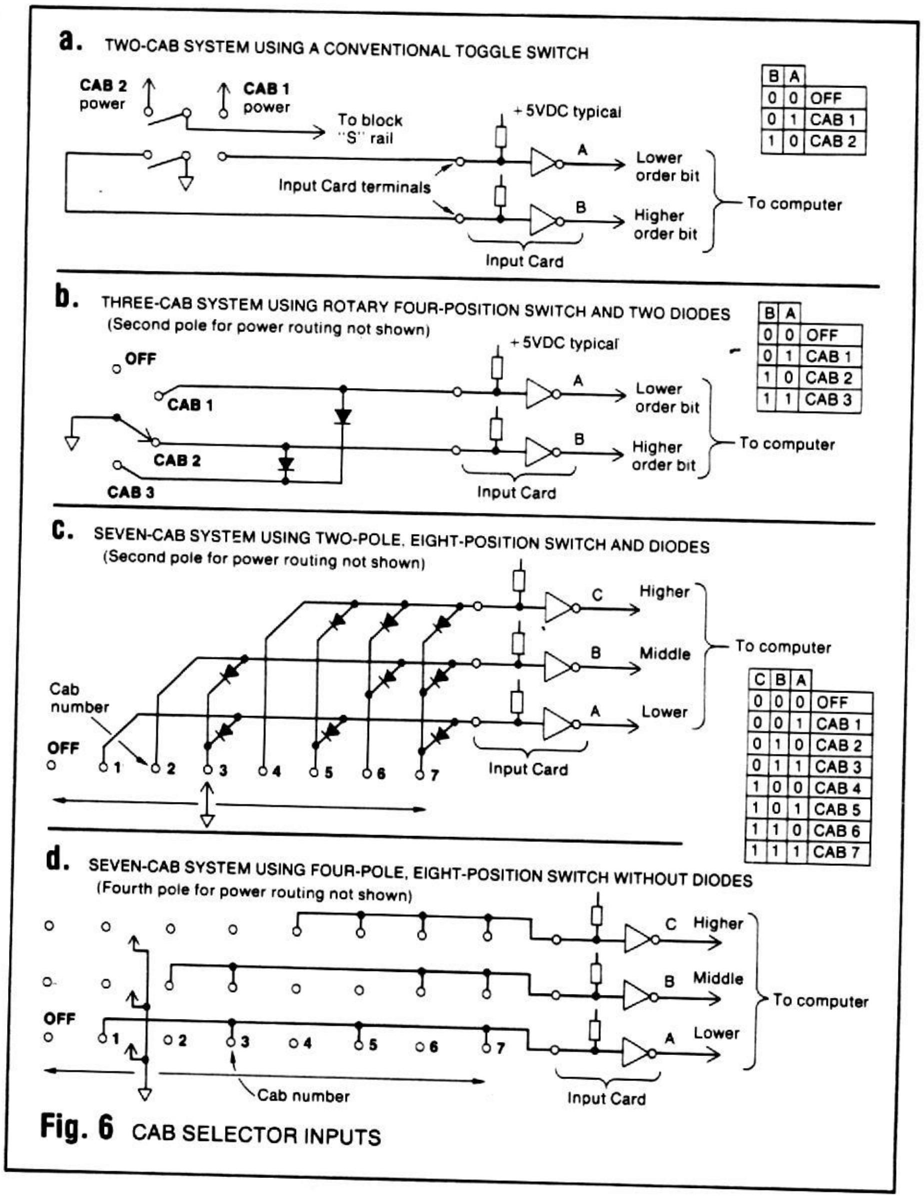


Fig. 6 CAB SELECTOR INPUTS

passes through a switch machine contact for each spur in the signaled block, wired in series. The contacts should be closed when the spurs are aligned for the normal route. This ensures that the spur-input-locked circuit can be a logic low only when the toggle is in locked position AND each spur turnout is in its normal position.

The off-power toggle for each dispatcher-controlled turnout is wired the same way as the spurs, except that it does not use a contact on the switch machine. When the toggle is set for off-power, train crews may operate the dual-control switch motor with local pushbuttons, in simulated manual mode.

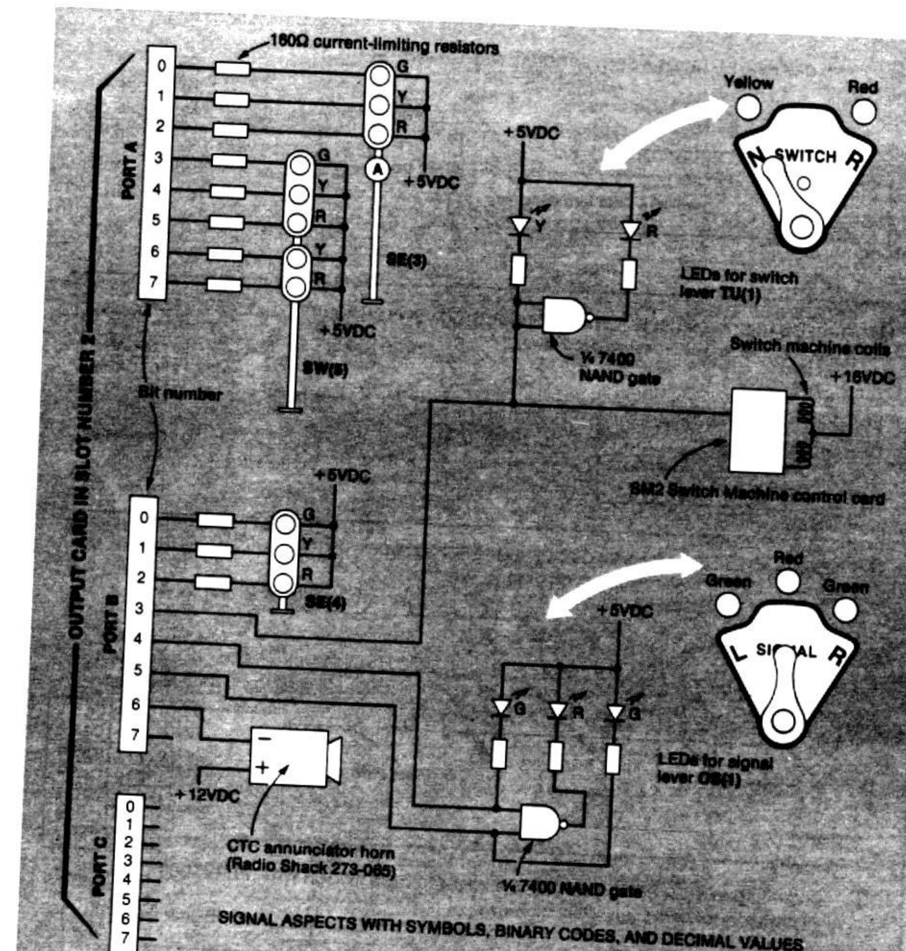
The CTC signal levers have three possible positions — stop (binary 00), clear left (01) and clear right (10) — so they need two input lines each. The CTC switch levers have just two positions — normal (1) and reverse (0) — so they need only one input line. The normally-open code-start button is wired to the Input Card to ground its input line when pressed. The detectors are wired following the guidelines in C/MRI Parts 7 and 9.

EXAMPLE I/O — OUTPUT

Figure 7 shows the connections from the Output Card in slot 2 in our example. Most lines will be for signals, so we'll start with them. The lines to any one signal should be arranged sequentially by bit number, and all on the same port. Start with green toward the LSB end as shown in the fig. 7 tables, so the decimal values for each aspect will match the software I'll explain next month.

I've shown LED signal lights as these are superior to grain-of-wheat (GOW) or other lamps, although the C/MRI can handle lamps as well. LEDs have deep, rich color and near-infinite life. They give off very little heat, and anyway there is no dye to burn off as happens with GOWs.

The LEDs must be wired to the C/MRI outputs through current-limiting resistors as shown. The 160Ω value is typical, but you may want to experiment. Staying with 160Ω for green LEDs and using a slightly higher value for yellow, say 270Ω, and then 330Ω for red, tends to make the different-colored LEDs appear equally bright.



180Ω current-limiting resistors
PORT A
0
1
2
3
4
5
6
7
G
Y
R
A
+5VDC
SE(3)
SW(5)
Bit number
OUTPUT CARD IN SLOT NUMBER 2
Yellow
Red
SWITCH
N
R
LEDs for switch lever TU(1)
¼ 7400 NAND gate
Switch machine coils
+18VDC
SM2 Switch Machine control card
PORT B
SE(4)
Green
Red
Green
SIGNAL
L
R
LEDs for signal lever OS(1)
+12VDC
CTC annunciator horn (Radio Shack 273-065)
PORT C
¼ 7400 NAND gate

SIGNAL ASPECTS WITH SYMBOLS, BINARY CODES, AND DECIMAL VALUES

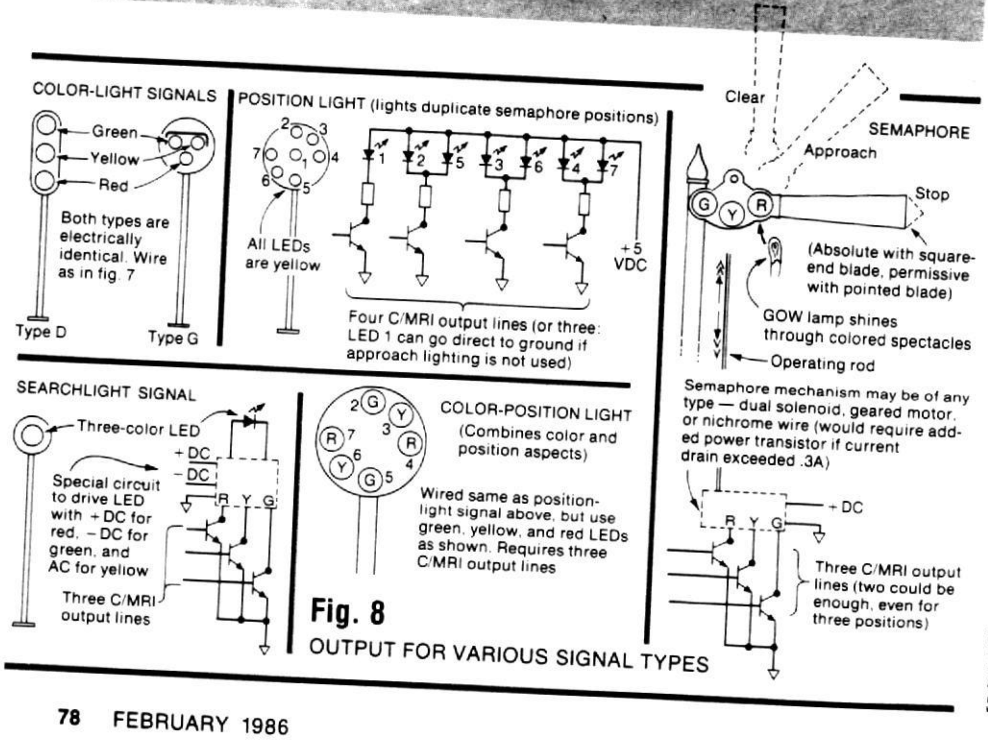
R Y G

ASPECT	SYMBOL	BINARY			DECIMAL
DarK	DK	0	0	0	0
GreeN	GN	0	0	1	1
YelloW	YW	0	1	0	2
ReD	RD	1	0	0	4
Decimal value of bit		4	2	1	
				LSB	

R Y   R Y G

ASPECT	SYMBOL	BINARY					DECIMAL
DarK	DK	0	0	0	0	0	0
Green-over-Red	GR	1	0	0	0	1	17
Yellow-over-Red	YR	1	0	0	1	0	18
Red-over-Red	RR	1	0	1	0	0	20
Red-over-Yellow	RY	0	1	1	0	0	12
Decimal value of bit		16	8	4	2	1	
						LSB	

Fig. 7 OUTPUT WIRING



COLOR-LIGHT SIGNALS
Green
Yellow
Red
Both types are electrically identical. Wire as in fig. 7
Type D
Type G
POSITION LIGHT (lights duplicate semaphore positions)
All LEDs are yellow
+5 VDC
Four C/MRI output lines (or three: LED 1 can go direct to ground if approach lighting is not used)
Clear
SEMAPHORE
Approach
Stop
G
Y
R
(Absolute with square-end blade, permissive with pointed blade)
GOW lamp shines through colored spectacles
Operating rod
Semaphore mechanism may be of any type — dual solenoid, geared motor, or nichrome wire (would require added power transistor if current drain exceeded .3A)
+DC
Three C/MRI output lines (two could be enough, even for three positions)
SEARCHLIGHT SIGNAL
Three-color LED
+DC
−DC
Special circuit to drive LED with +DC for red, −DC for green, and AC for yellow
Three C/MRI output lines
COLOR-POSITION LIGHT
(Combines color and position aspects)
Wired same as position-light signal above, but use green, yellow, and red LEDs as shown. Requires three C/MRI output lines

Fig. 8
OUTPUT FOR VARIOUS SIGNAL TYPES

For the example I've assumed two-coil switch machines controlled by SM-2 circuits from C/MRI Part 6, so only a single output line is required for each controlled turnout. The two panel LEDs for the switch lever can be driven off the output line using a 7400 NAND gate as shown, or a standard invertor/transistor as for the two-color signals in Part 9. If you prefer positive feedback of the turnout's actual position, you can use an SPDT contact on the switch machine.

The three panel LEDs for the signal lever are controlled by two output lines and the 7400 NAND gate. In a later installment I'll show you how, under certain conditions, you can use a separate decoding circuit to control the three signals protecting the OS section from these same two lines. That reduces the wiring requirements, but for overall flexibility and adaptability you can't beat the approach shown here with a separate output line to each signal LED.

With separate lines each LED is under software control, so you can set up approach lighting, flashing aspects, and any other options simply by changing your program. The extra wiring effort also pays off in improved fault isolation capability, as I'll also cover later in an overview of how to use your C/MRI for computer-automated diagnostics.

The CTC panel annunciator horn, such as Radio Shack 273-065, can be connected directly between an output line and a +12VDC supply.

OTHER SIGNAL TYPES

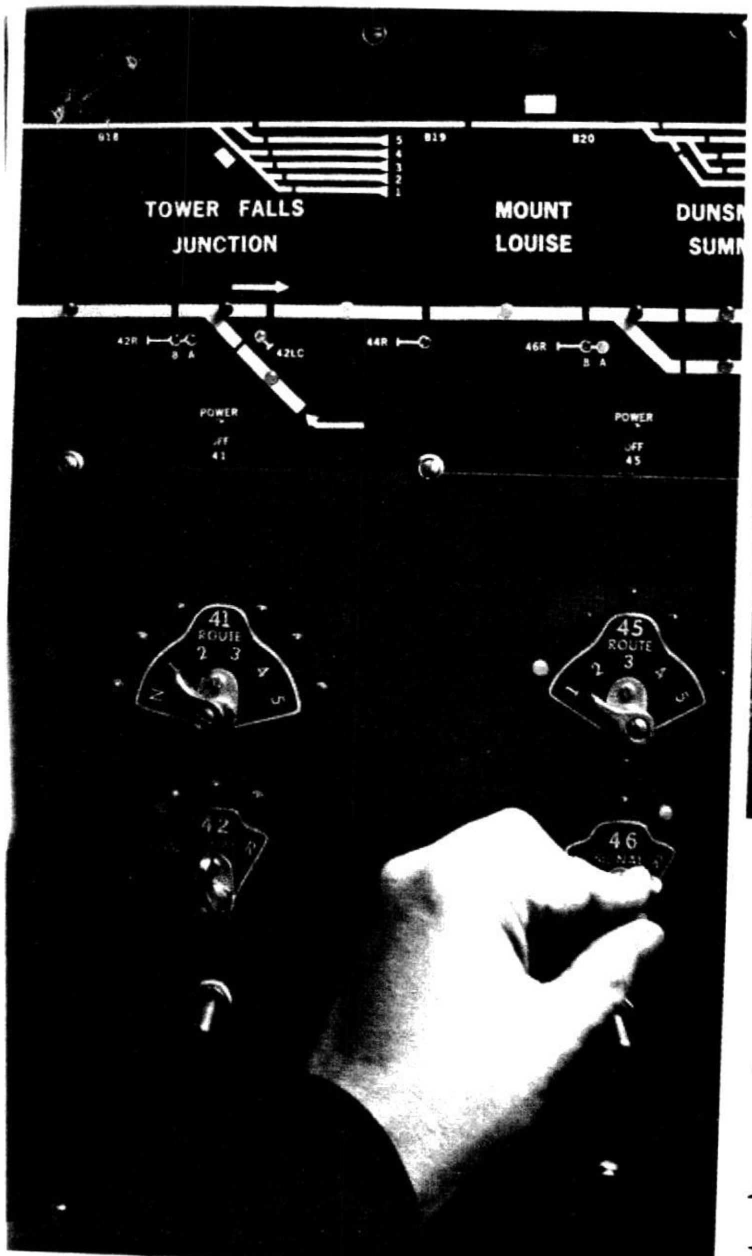
There are many types of signals, but they are all easily adapted to C/MRI output. Just remember that a connection to an output line is like a connection to one terminal of an SPST toggle switch with its other terminal connected to ground. The Output Card's maximum effective "switch" ratings are 40VDC and .3A, and the software simply turns the switch on or off.

Figure 8 shows circuit suggestions for several types of signals. The type D color-light signals (as used on the SV and for the example layout above), are the most straightforward, and type G color-lights are electrically identical.

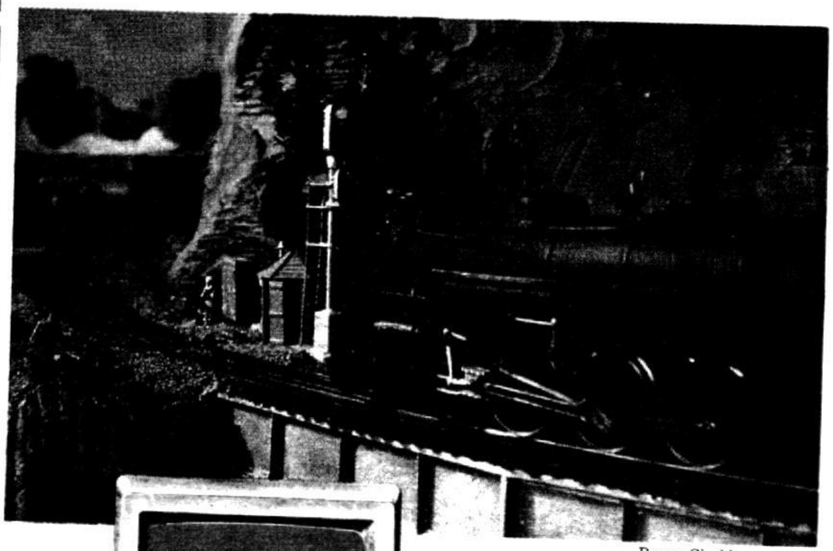
The semaphore drive mechanism can be dual solenoids, a geared motor with limit switches, or even a heated nichrome wire. The exact connections will depend upon the mechanism you use and whether you have two- or three-position semaphores. A lamp is better than an LED here as you need a clear bulb that will shine brightly through the semaphore's colored spectacles.

To make the software programming as straightforward as possible, always arrange the signal connections on the Output Cards so as not to split a given signal between ports, and to keep the connections consistent as in fig. 7. That way you can use the same symbols and decimal values to define the different aspects for any signal.

Next month I'll explain the programming to make the signals work and explain how you can add or delete features as you wish by changing the software. See you then. ♁



This month Bruce Chubb tells you how to write a computer program for realistic signaling action like this: the dispatcher lines up a route on his CTC panel, left, giving a clear signal to the train, below, heading into the Sunset Valley's Dunsmuir Summit yard.



Bruce Chubb

The C/MRI: A computer/model railroad interface

Part 12: Computerized signals — software

BY BRUCE CHUBB

WE'LL complete our look at computerized signaling this month with software to drive the signals and other hardware I showed you last time. One way to prepare your signal software is the general-purpose, "table-driven" approach, in which the program loops through the same block of statements once for each group of two or three signals. The only things that change are the subscripts defining the particular "table parameters."

That works well for the most basic situations, so that's what I described for the two-color block signals in Computer Cab Control. When you have only a few special situations, such as reversing block 12 in the CCC example layout, they can be handled with unique coding.

Another way is to simply program the required code for each signal. This takes more code and uses more computer memory, but execution of the real-time loop will be faster. Direct code is especially useful for signaling complex trackwork, as at multi-track junctions and large interlocking plants. I'll describe direct coding in this installment so you'll have examples of both approaches.

I'll show you how to set up a detailed, prototypical Centralized Traffic Control signal system for the example layout shown last month in C/MRI Part 11, fig. 4. As I said then, I'll cover just the building blocks for any kind of system you might want: the signals at OS(1) — SE(3), SE(4), and SW(5) — and block signal SW(6). The signal protecting the facing points of a dispatcher-controlled turnout in a signaled route is the most complex, so let's program SW(5) first.

FACING-POINT SIGNALS

SW(5) is a dual-head signal protecting the facing-point end of turnout TU(1). The upper three-color signal head governs movement past the turnout on the main track. The lower head governs movement on the diverging route, the passing track. In CTC territory the dispatcher clears this signal with a signal control lever on the CTC panel.

Basically, to clear a westbound train through OS(1) the dispatcher would throw signal lever LD(1) left and press code button CB(1). Throwing LD(1) right would clear a train to the east, while putting it in the center position sets all OS(1) signals to stop.

The signal aspect displayed at any instant is controlled by more than just the signal lever. It depends too on the position of the turnout, occupancy of the OS section, and occupancy of the next block. As the track plan gets more complicated the logic that must be tested increases significantly. Put in several signaled routes, a double crossover, and numerous spurs to check, and it can get darn complicated!

This is where the computer really ıys off, as all the logic can be handled ith a little bit of software. Once you ət the hang of it, you'll enjoy both preıaring and using your signal program, nd find that it's easier to make changes nd add improvements in the software ıan on the layout.

Figure 1 is the logic flow diagram for ignal SW(5). As we saw in C/MRI Part 10, such diagrams are pictures of what the software must do to compute the desired outputs, in this case the aspect of our facing-point signal. I've included the program line numbers in the flow chart for correlation to the program listing in ig. 6, which I'll cover later.

It's best to define your major program variables, flow chart your program logic, and set up your I/O tables before you start writing the program code. This results in a better-structured program with code that is easier to program and understand, and with many fewer bugs.

As for CCC, program variable names are assigned to signals, block and OS-section detectors, locks for spurs, switch motors, and so forth, and used in the flow chart too. With BASIC you are restricted to two-character variables, so use of subscripted array variables such as SE(1), SE(2), and SE(3) is a must.

Array variables are also essential with table-driven software as for CCC. However, with direct code there is little to be gained by using subscripted variables, and they typically slow down loop processing significantly.

In fact, if you use a language other than BASIC, one that doesn't limit variables to two characters, it's best to drop subscripts altogether and use longer but unique names. For example, I call the three signals controlled by signal lever 30 on the Sunset Valley CTC panel S30LAB, S30RA, and S30RC. These prototypical notations are easy to understand once you know that S30LAB stands for "signal 30 left, A and B heads."

Since the example program here will still be in BASIC, I'll use most of the same variables used in Part 10 for CCC. We will need a few new ones as listed in fig. 2. They're mainly for the OS sections, multi-head signal aspects, and the levers and buttons on the dispatcher's panel. If you look up a variable in fig. 2 and don't find it, its use is identical to that defined in Part 10, fig. 3.

Figure 7 in Part 11 showed how the signal aspects are defined with a separate bit for each LED. The decimal equivalent for each aspect is derived by adding up the decimal value for each bit turned on.

The ones and zeros in each position correspond to which LEDs are on, 1, and off, 0. With these codes established, as I've done for you, you can forget the numerics and simply use the two-letter alpha designators. For example, if you want to set signal SW(5) red-over-yellow you simply write SW(5) = RY. Things just couldn't be much easier than that!

FACING-POINT LOGIC

Getting back to fig. 1, the first step in the calculations is to set signal variable SW(5) = RR — red-over-red, the stop indication. The trackside signal DOES NOT instantly change to red-over-red; that wouldn't happen until the value of SW(5) was written to the output port. The program simply tells the computer's memory to hold that value for the variable until the program finds that some "better," less-restrictive indication would be safe.

This initial setting to the most restrictive aspect is a fail-safe approach. The program next performs a series of tests to see whether or not it can allow a better aspect. If it fails any test, variable SW(5) remains RR as preset, and the signal will be set red-over-red when all the outputs are written to the railroad.

The first test is whether the dispatcher's signal lever is "not left," either centered or to the right. LV(1) is the "effective-position" variable for this — see "CODE-BUTTON LOGIC" below. If LV(1) is not left, no action is taken and the branch to line 2090 keeps the signal at stop.

If the lever is left the program checks OS-section occupancy. If OS(1) is occupied the signal is kept at stop. If OS(1) is clear, the next test is whether remote, computer power is applied to the switch motor. If "off power," with a train crew in local control of the dual-control turnout and able to throw it at any instant, all the signals protecting the turnout must be kept at stop *via* the branch to 2090. These three tests could have been performed in any order, but it tends to speed up execution if you arrange your IF/THEN statements with the more probable conditions first.

If the program drops through each of these tests, it reaches 1960 and checks the position of TU(1). If the turnout is normal, the program looks to see if block 3 is occupied, if the cab assigned to block 5 is different from that assigned to block 3, and if the spurs in block 3 are unlocked. If

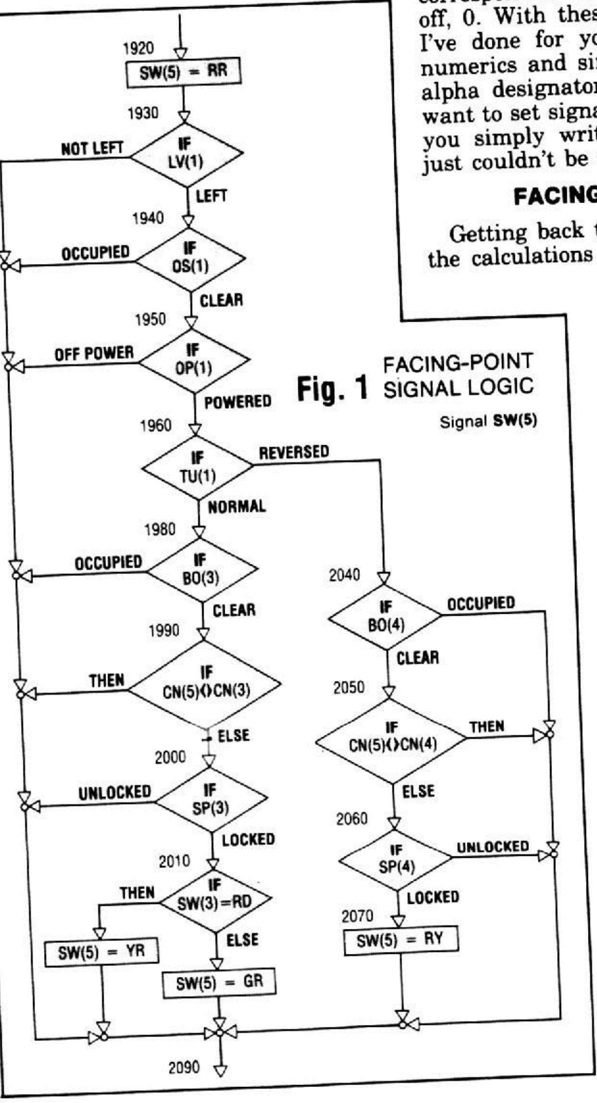


Fig. 1 FACING-POINT SIGNAL LOGIC
Signal **SW(5)**

TD(N):	Array dimensioned to the maximum OS number used to define the desired position of the switch lever; e.g. TD(3)=Ø for lever 3 at reverse, =1 for normal.
LD(N):	Array dimensioned to the maximum OS number used to define the desired position of the signal lever; e.g. LD(1)=Ø for lever 1 at stop, =1 for left and =2 for right.
LV(N):	Array dimensioned to the maximum OS number used to define the effective position of the signal lever; i.e. after the code button is pressed; e.g. LV(1)=Ø for lever 1 at stop, =1 for left and =2 for right.
OS(N):	Array dimensioned to the maximum OS number used to define the current occupation status of the OS secion; e.g. OS(2)=1 for OS 2 occupied and =Ø for clear.
OP(N):	Array dimensioned to the maximum OS number used to define the off powered status of the dual control switch motor; e.g. OP(2)=1 for turnout 2 powered and =Ø for off power.
HN:	Variable defining status of annunciator HorN (1=on, Ø=off).
MS:	Variable set to Maximum OS number.
DK:	Variable set to constant=Ø for signals DarK.
GN:	Variable set to constant=1 for GreeN.
YW:	Variable set to constant=2 for YelloW.
RD:	Variable set to constant=4 for ReD.
GR:	Variable set to constant=17 for Green-over-Red.
YR:	Variable set to constant=18 for Yellow-over-Red.
RR:	Variable set to constant=2Ø for Red-over-Red.
RY:	Variable set to constant=12 for Red-over-Yellow.
ST:	Variable set to constant=Ø for STop.
LE:	Variable set to constant=1 for LEft.
RI:	Variable set to constant=2 for RIght.
OP:	Variable set to constant=Ø for Off-Power.
UL:	Variable set to constant=Ø for UnLocked.

Fig. 2 TABLE OF VARIABLES

Variables not defined here are the same as Part 10, fig. 3

one of these is true the program branches to 2090, keeping SW(5) at stop.

If the program drops through all three tests for the normal route, it checks the next signal, SW(3). IF SW(3) is red, THEN everything is clear and set up only one block ahead, so SW(5) is set to YR (yellow-over-red, approach), ELSE everything is clear and set up at least two blocks ahead so SW(5) is set to GR (green-over-red, clear).

Back in line 1960, if TU(1) had been reversed, aligned for the passing siding, the program would have branched to 2040 to test the diverging instead of the main route. If siding block BO(4) were occupied, if there were no continuity of cab assignment, or siding spur SP(4) were unlocked, then SW(5) would be kept at stop.

If the program gets past all the IF statements, SW(5) is set to RY (red-over-yellow, proceed at prescribed speed on diverging route). In this case I didn't include a check of the next signal, so the example follows the practice of the SV and many prototypes in allowing no better aspect than red-over-yellow for entering the passing siding. That completes all the checks needed to calculate the setting for facing-point signal SW(5).

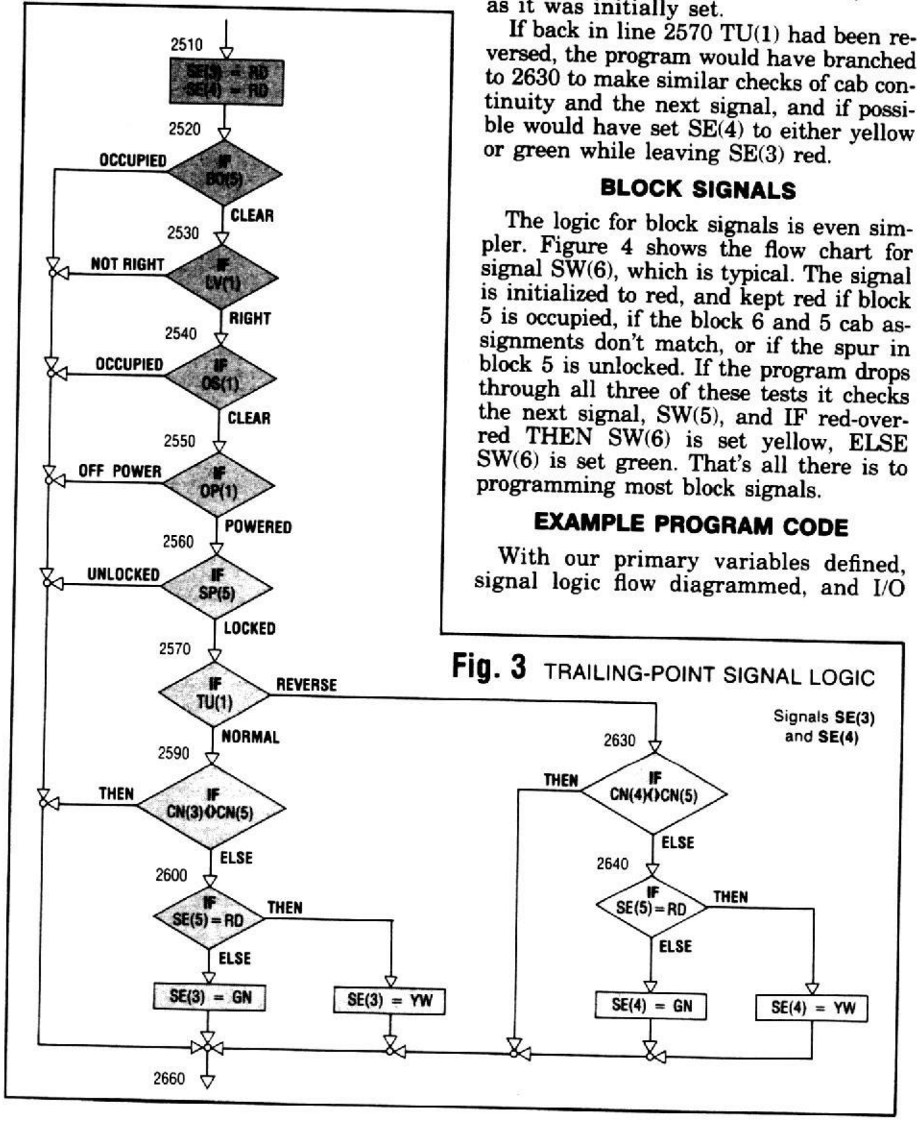


Fig. 3 TRAILING-POINT SIGNAL LOGIC

TRAILING-POINT SIGNALS

Let's now take a look at the logic flow chart for trailing-point signals SE(3) and SE(4), in fig. 3. Once you understand both facing- and trailing-point signal logic you'll be able to handle any set of route-interlocking signals.

Starting at the top, SE(3) and SE(4) are both initially set to RD (red, stop — both are absolute signals). Look at Part 11, fig. 4, and you'll understand that there's a group of common tests that must be passed for either of these signals to be cleared, whatever the turnout position. It's best to perform these five tests first: checking to see if BO(5) is occupied, LV(1) is not right, OS(1) is occupied, turnout 1 is off power, or SP(5) is unlocked. If any of these conditions is true, a branch to line 2660 keeps both signals red.

If the program drops through all five common IF statements, it reaches line 2570 and checks the position of turnout TU(1). If normal, line 2590 checks the cab assignments and if they aren't identical it keeps both signals red. If the cab assignments are identical the program checks the next signal, SE(5), and IF red THEN SE(3) is set to YW (yellow, approach), ELSE SE(3) is set to GN (green, clear), and in either case SE(4) stays red as it was initially set.

If back in line 2570 TU(1) had been reversed, the program would have branched to 2630 to make similar checks of cab continuity and the next signal, and if possible would have set SE(4) to either yellow or green while leaving SE(3) red.

BLOCK SIGNALS

The logic for block signals is even simpler. Figure 4 shows the flow chart for signal SW(6), which is typical. The signal is initialized to red, and kept red if block 5 is occupied, if the block 6 and 5 cab assignments don't match, or if the spur in block 5 is unlocked. If the program drops through all three of these tests it checks the next signal, SW(5), and IF red-over-red THEN SW(6) is set yellow, ELSE SW(6) is set green. That's all there is to programming most block signals.

EXAMPLE PROGRAM CODE

With our primary variables defined, signal logic flow diagrammed, and I/O tables established, we're ready to begin the program coding. Figure 5 shows the overall program structure, which is similar to the CCC program in Part 10. The initialization section performs the same basic functions, as does the real-time loop. The loop's internal details, however, are quite different.

Figure 6 gives the program listing, with colored dots to show where the code would be expanded to cover the other signals, I/O Cards, and so on. For CCC you'd insert the appropriate code lines from Part 10, fig. 2, to determine the next-east/next-west block designations, and to automatically drop and assign cabs. In this case you would need to set up two different arrays for block occupancy. One, called BX( ), would include the effects of the OS sections and the other, BO( ), would not.

The BX( ) array is calculated from the input read of BO( ) and OS( ) with this block of statements:

```
FOR N=1 TO MB
BX(N)=BO(N)
NEXT N
IF OS(1)=OC THEN BX(5)=OC
IF OS(2)=OC THEN BX(6)=OC
.
.
.
```

In the cab assignment code you would then use BX( ) in place of BO( ), and use BO( ) directly in the signal code for this installment.

Program lines 10 through 660 comprise the initialization section and follow the format and procedures used in Part 10. The real-time loop starts at line 810, and 810 through 980 read the Input Card and unpack the Input Bytes, IB. If you are using one of the Apple BASICS you will have to alter the unpacking lines as in Part 10, fig. 8.

CODE BUTTON LOGIC

The first calculation starts at line 1220, and calculates the effective positions of the levers on the CTC panel. Remember that the levers can be moved at will but their positions can only become effective when the dispatcher presses the code button. This condition is handled by defining two arrays each for the switch levers and signal levers:

TD( ): *T*urnout lever *D*esired position.
LD( ): signal *L*ever *D*esired position.
TU( ): effective *TU*rnout lever position.
LV( ): effective signal *LeV*er position.

TD( ) and LD( ) are read on each loop. When any code button is pressed during a given loop, and if the desired settings for its levers are found to be safe, then the effective lever positions are set equal to those read in. Figure 7 shows the corresponding logic flow for code button CB(1).

The capacitor and resistor connected to the code button input line, Part 11, fig. 5, hold the line low from the time the button is pushed until the read is executed. That means that the dispatcher doesn't have to hold the button for the duration of a program loop, which could be annoying with slower-running programs. Use capacitors of higher or lower value if your program needs more or less "hold time."

If the code button is not pressed no

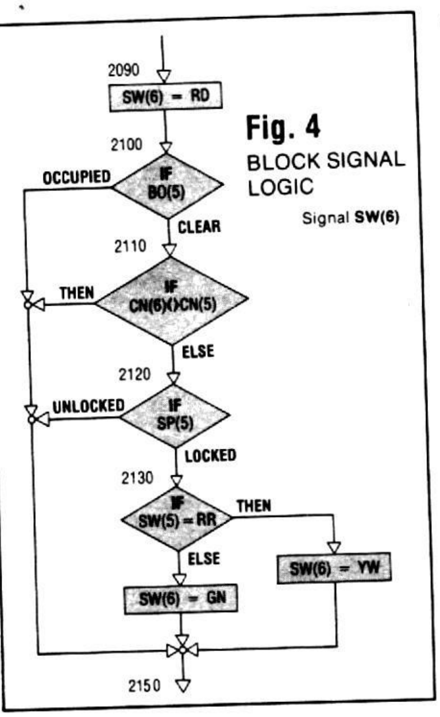


Fig. 4 BLOCK SIGNAL LOGIC

Signal **SW(6)**

action is taken and the program branches to line 1340 to test the next code button. The same is true if the button is pressed but the OS section is occupied or the powered turnout is off power. These latter two IF statements give you software protection against the turnout being thrown under a train — pretty neat isn't it? If the program drops through all three IF statements, then the effective turnout position variable TU(1) is set equal to the read-in lever position TD(1).

The program then sets the effective signal lever positions, beginning at line 1260. If LD(1) is at stop then LV(1) is set to stop, but if the dispatcher is trying to clear traffic through the OS section to the right (east) or left (west), the conditions of the adjacent signal levers must also be checked.

This is software interlocking, and an essential protection against the dispatcher's inadvertently letting two opposing trains enter single track. For example, suppose LV(2) is left and the dispatcher sets LD(1) right, then presses code button CB(1). There would be a potential conflict if the computer set LV(1) = right.

The program takes care of this in line 1270, making a branch to line 1310 which sets the effective lever position to stop and the annunciator horn variable, HN, to on (logic 1). Upon output this sounds the horn shown in Part 11, fig. 7. The alarm tells the dispatcher that he or she has tried to clear conflicting routes, which the "CTC machine," the computer, has ignored while keeping all OS(1) signals red. If LV(2) were not left — at stop or right — the program would drop through 1310 to set effective lever position LV(1) = LD(1), the dispatcher's desired command.

If the dispatcher is trying to clear a movement to the left at OS(1), the checks are more complex. If TU(0) is set opposite of TU(1), one for the main line and the other for the passing siding, then the LV(0) position need not be checked and 1280 branches to 1320 which sets LV(1) = LD(1). However, if TU(1) .= TU(0), both set either for the main or the siding, then 1290 checks LV(0). Only if LV(0) is not right is LV(1) set equal to LD(1). If LV(0) is right, LV(1) is set to stop and the horn sounds.

That takes care of code button CB(1). The logic for others could be slightly different to handle different track arrangements, but the basic checks required would be the same.

AUTOMATIC FOLLOWING

The next block of code, lines 1600 through 1640, is optional. Without it you get a feature used in some CTC installations called "automatic following." With this feature, once the dispatcher clears a train through an OS section in one direction, following trains automatically obtain favorable signals at that OS section as the block beyond clears. This can be handy for handling heavy traffic.

Lines 1600-1640 prevent automatic following by setting the effective signal lever to stop each time its OS section is occupied. The dispatcher must push the code button again to clear traffic through an OS section that has been occupied.

SIGNAL PROGRAMMING

The calculation of signal aspects begins at line 1650, with the first signal for which I show example code being SW(5), lines 1910 through 2070. This code simply implements the flow chart shown in fig. 1. If your particular BASIC won't accept the IF/THEN/ELSE statement as in line 2010, for example, you can achieve the same result by using two lines with the statement after the word ELSE moved up as a separate line above the IF.

I've placed the code for SW(6), following the logic shown in fig. 4, right after SW(5). It's best with three-color signals to first calculate all the signals governing travel in one direction, say westbound, and then all those for the opposite direction.

Arrange the code to calculate the farthest signals in each direction first — SW(3) and SW(4) before SW(5), SW(5) before SW(6), and in the reverse direction SE(6) before SE(5) and so on. This reduces the amount of logic needed to select green or yellow, because the program can simply check the next signal in advance, the next one facing a train in that direction, which has already been calculated.

After line 2130 you'd fit in the rest of the westbound signals, beginning with SW(7), followed by the eastbound signals. As an example, lines 2500 through 2640 show the coding required for signals SE(3) and SE(4) following the logic of fig. 3.

APPROACH LIGHTING

The code in lines 2800 through 2850 is another optional feature. It provides approach-lighted signals that light up only when the block which a signal faces is occupied. The code simply loops through each block 1 to MB and if occupied leaves the signals at each end of the block as calculated. If the block is clear line 2840 is executed to set both signals dark, DK or decimal 0, with no LEDs lit.

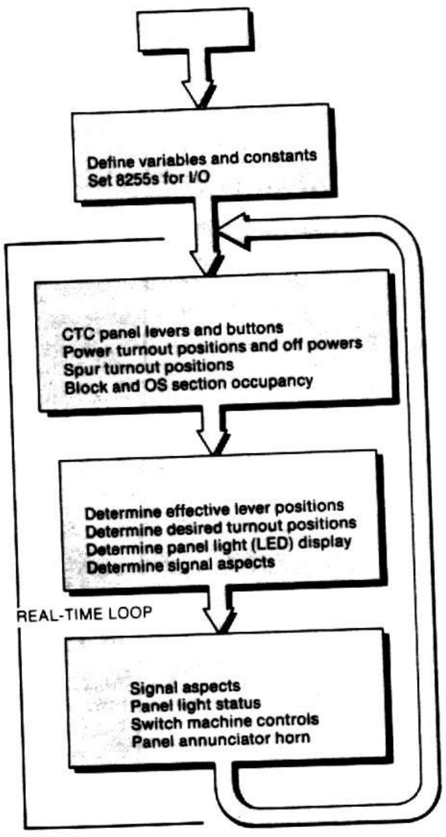

Define variables and constants
Set 8255s for I/O
CTC panel levers and buttons
Power turnout positions and off powers
Spur turnout positions
Block and OS section occupancy
Determine effective lever positions
Determine desired turnout positions
Determine panel light (LED) display
Determine signal aspects
REAL-TIME LOOP
Signal aspects
Panel light status
Switch machine controls
Panel annunciator horn

Fig. 5 CTC SIGNALING PROGRAM STRUCTURE

OUTPUT

With all calculations complete, the last remaining step is to write the outputs to the railroad. Lines 2910 through 3000 do this for the single Output Card in our example, following the same practices used in Part 10. Again, with the Apple BASICS you'll need to change the OR to a plus sign (+). Line 4010 resets the horn variable HN to off, logic 0, and 4030 branches back to the beginning of the real-time loop at line 810.

CHANGING CAPABILITIES

For the sake of example I've covered all of the major prototype tests for unlocks, signal levers, switch motors off-power, and so on, as well as the most common model railroad requirement for cab-power continuity. You can drop any features you don't want or need by not including those particular code statements in your program.

For example, if you use command control you'd leave out the checks for cab continuity. If you don't have a dispatcher's panel, leave out the signal and turnout lever position checks. And, if you aren't simulating dual-control switch motors, leave out the checks for these being off power.

Keep the program as simple as you can while including the features you want. It will be easy to add or change features, and at no cost when all you need to do is change the software. For example, you can rewrite your program for Automatic Permissive Block (APB) signaling instead of CTC.

```
10   PRINT "BASIC EXAMPLE SIGNAL PROGRAM USING CTC RULES"
20   REM••Define variable types and array sizes••
30   DEFINT A-Z
40   DIM BO(12),CN(12),SE(12),SW(12),OP(4),LD(4)
50   DIM LV(4),TD(4),TU(4),OS(4),SP(12)
60   REM••Define constants for packing/unpacking bits••
70   B0=1: B1=2: B2=4: B3=8: B4=16: B5=32: B6=64: B7=128
80   W1=1: W2=3: W3=7: W4=15: W5=31: W6=63: W7=127
200  REM••Define general constants••
210  CL=0        'CLear
220  OC=1        'OCcupied
230  DK=0        'DarK              00000
240  GN=1        'GreeN               001
250  YW=2        'YelloW              010
260  RD=4        'ReD                 100
270  GR=17       'Green-over-Red    10001
280  YR=18       'Yellow-over-Red   10010
290  RR=20       'Red-over-Red      10100
300  RY=12       'Red-over-Yellow   01100
310  ST=0        'STop
320  LE=1        'LEft
330  RI=2        'RIght
340  OP=0        'Off-Power
350  UL=0        'UnLocked
360  TN=1        'Turnout Normal
370  TR=0        'Turnout Reversed
380  PP=1        'Pushbutton Pressed
390  MB=12       'Mamimum Block number
400  MS=4        'Maximum OS number
600  REM••Define I/O interface constants••
610  CI=155         '8255 Control byte for Input
620  CO=128         '8255 Control byte for Output
630  SA=1024        'Starting Address of the I/O
640  REM••Transmit control bytes to set up I/O ports••
```

Fig. 6 CTC SIGNALING EXAMPLE PROGRAM

```
650  POKE SA+3,CI     'card 1 input
660  POKE SA+7,CO     'card 2 output
:
800  REM••Read inputs and separate out
          bit areas••
810  IB=PEEK(SA)         'Card 1 port A
820  BO(3)=IB/B0 AND W1
830  BO(4)=IB/B1 AND W1
840  BO(5)=IB/B2 AND W1
850  OS(1)=IB/B3 AND W1
860  LD(1)=IB/B4 AND W2
870  SP(3)=IB/B6 AND W1
880  SP(4)=IB/B7 AND W1
890  IB=PEEK(SA+1)       'Card 1 port B
900  SP(5)=IB/B0 AND W1
910  OP(1)=IB/B1 AND W1
920  CN(3)=IB/B2 AND W3
930  CN(4)=IB/B5 AND W3
940  IB=PEEK(SA+2)       'Card 1 port C
950  CN(5)=IB/B0 AND W3
960  TD(1)=IB/B3 AND W1
970  LD(2)=IB/B4 AND W2
980  CB(1)=IB/B6 AND W1
:
1200 REM••Calculate lever positions••
1210 REM••Code button 1
1220 IF CB(1)<>PP THEN GOTO 1340
1230 IF OS(1)=OC THEN GOTO 1340
1240 IF OP(1)=OP THEN GOTO 1340
1250 TU(1)=TD(1)
1260 IF LD(1)=ST THEN LV(1)=ST: GOTO 1340
1270 IF LD(1)=RI THEN GOTO 1310
1280 IF TU(1)<>TU(0) THEN GOTO 1320
1290 IF LV(0)<>RI THEN GOTO 1320
1300 LV(1)=ST: HN=1: GOTO 1340
1310 IF LV(2)=LE THEN LV(1)=ST: HN=1:
        GOTO 1340
1320 LV(1)=LD(1)
1330 REM••Code button 2
1340 IF CB(2)<>PP THEN GOTO XXXX
:
1600 REM••Set signal levers off if
        OS occupied••
1610 REM••Remove code if want automatic
        following
1620 FOR N=1 TO MS
1630 IF OS(N)=OC THEN LV(N)=ST
1640 NEXT N
1650 REM••Calculate signal aspects••
1660 REM••SW(2)••
:
1910 REM••SW(5)••
1920 SW(5)=RR
1930 IF LV(1)<>LE THEN GOTO 2090
1940 IF OS(1)=OC THEN GOTO 2090
1950 IF OP(1)=OP THEN GOTO 2090
1960 IF TU(1)=TR THEN GOTO 2040
1970 REM••normal route••
1980 IF BO(3)=OC THEN GOTO 2090
1990 IF CN(5)<>CN(3) THEN GOTO 2090
2000 IF SP(3)=UL THEN GOTO 2090
2010 IF SW(3)=RD THEN SW(5)=YR ELSE SW(5)=GR
2020 GOTO 2090
2030 REM••reverse route••
2040 IF BO(4)=OC THEN GOTO 2090
2050 IF CN(5)<>CN(4) THEN GOTO 2090
2060 IF SP(4)=UL THEN GOTO 2090
2070 SW(5)=RY
2080 REM••SW(6)
2090 SW(6)=RD
2100 IF BO(5)=OC THEN GOTO 2150
2110 IF CN(6)<>CN(5) THEN GOTO 2150
2120 IF SP(5)=UL THEN GOTO 2150
2130 IF SW(5)=RR THEN SW(6)=YW ELSE SW(6)=GN
2140 REM••SW(7)
2150 SW(7)=RD
:
2500 REM••SE(3) & SE(4)
2510 SE(3)=RD: SE(4)=RD
2520 IF BO(5)=OC THEN GOTO 2660
2530 IF LV(1)<>RI THEN GOTO 2660
2540 IF OS(1)=OC THEN GOTO 2660
2550 IF OP(1)=OP THEN GOTO 2660
2560 IF SP(5)=UL THEN GOTO 2660
2570 IF TU(1)=TR THEN GOTO 2630
2580 REM••normal route
2590 IF CN(3)<>CN(5) THEN GOTO 2660
2600 IF SE(5)=RD THEN SE(3)=YW ELSE SE(3)=GN
2610 GOTO 2660
2620 REM••reverse route••
2630 IF CN(4)<>CN(5) THEN GOTO 2660
2640 IF SE(5)=RD THEN SE(4)=YW ELSE SE(4)=GN
2650 REM••SE(2)
2660 SE(2)=RR
:
2800 REM••Turn off signals unless approach
        occupied••
2810 REM••Remove if don't want approach
        lighting••
2820 FOR N=1 TO MB
2830 IF BO(N)=OC THEN GOTO 2850
2840 SE(N)=DK: SW(N)=DK
2850 NEXT N
:
2900 REM••Write outputs to railroad••
2910 OB=0                  'Card 2 port A
2920 OB=SE(3)*B0 OR OB
2930 OB=SW(5)*B3 OR OB
2940 POKE SA+4,OB
2950 OB=0                  'Card 2 port B
2960 OB=SE(4)*B0 OR OB
2970 OB=TU(1)*B3 OR OB
2980 OB=LV(1)*B4 OR OB
2990 OB=HN*B6 OR OB
3000 POKE SA+5,OB
:
4000 REM••Reset horn to off••
4010 HN=0
4020 REM••Return to beginning of loop••
4030 GOTO 810
```

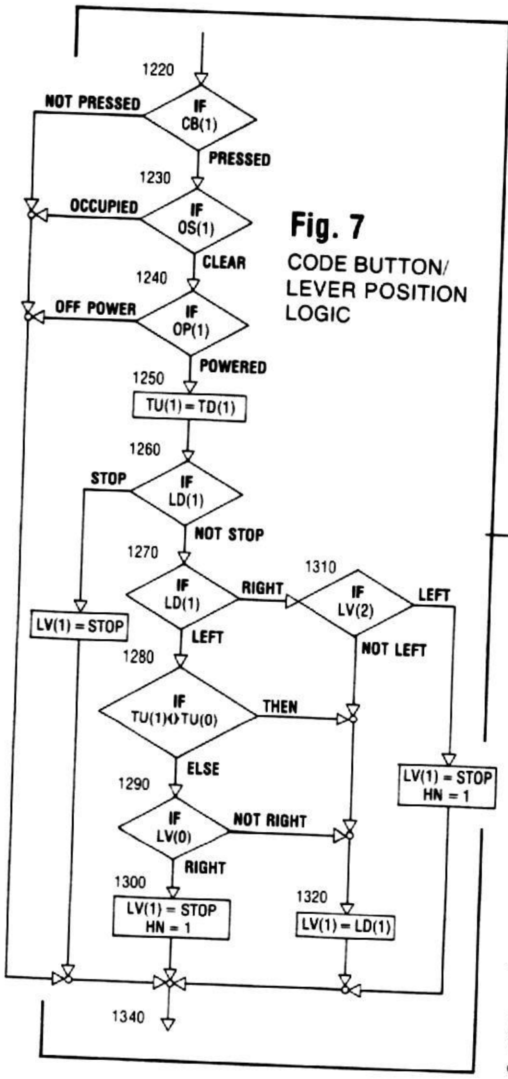


Fig. 7 CODE BUTTON/ LEVER POSITION LOGIC

Even if you have a CTC panel, you can operate informally without a dispatcher by having the software bypass the code buttons and signal levers. The logic would simply check that once a train in one direction entered single track, an opposing train was not allowed into the same section of single track until the first train had cleared.

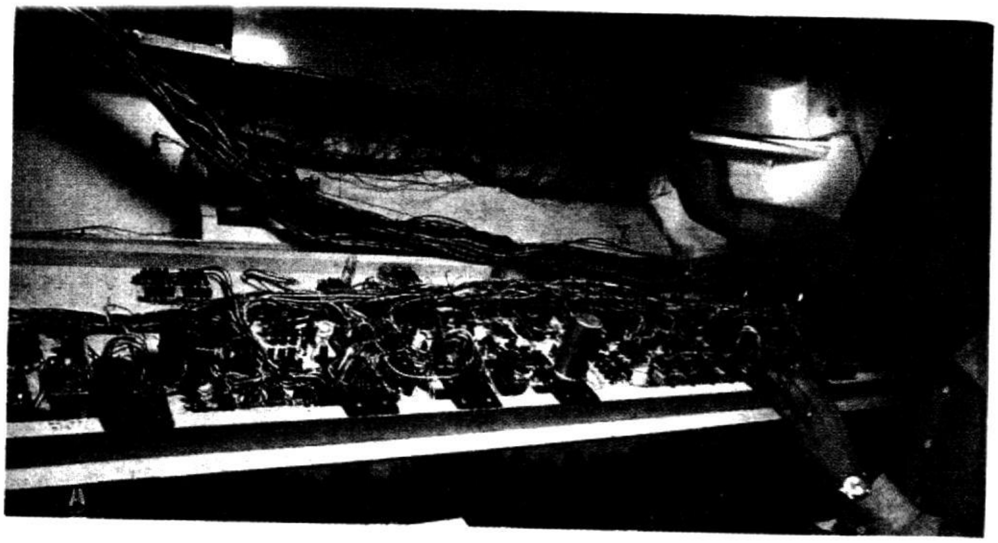
One feature I didn't include in the example, but that you may want to add once your baseline system is operating, is a feature used by some railroads called "preconditioning." It lets the dispatcher set levers at a given OS section for the next movement and hit the code button when the section is occupied.

The requested movement is not acted upon until the OS section is clear, and then is carried out automatically. Preconditioning is handy because it lets the dispatcher do his work for a meet as the trains arrive at the passing siding, then turn his attention somewhere else while the meet is completed automatically.

Calculations can also be added to automatically control the polarity in each reverse block and, as in Part 10, to set red signals leading in and out of a reverse block if the polarity isn't correct. Software can also operate cab signals, for a large club layout with elevated cabs or in any case where the engineers can't see the trackside signals.

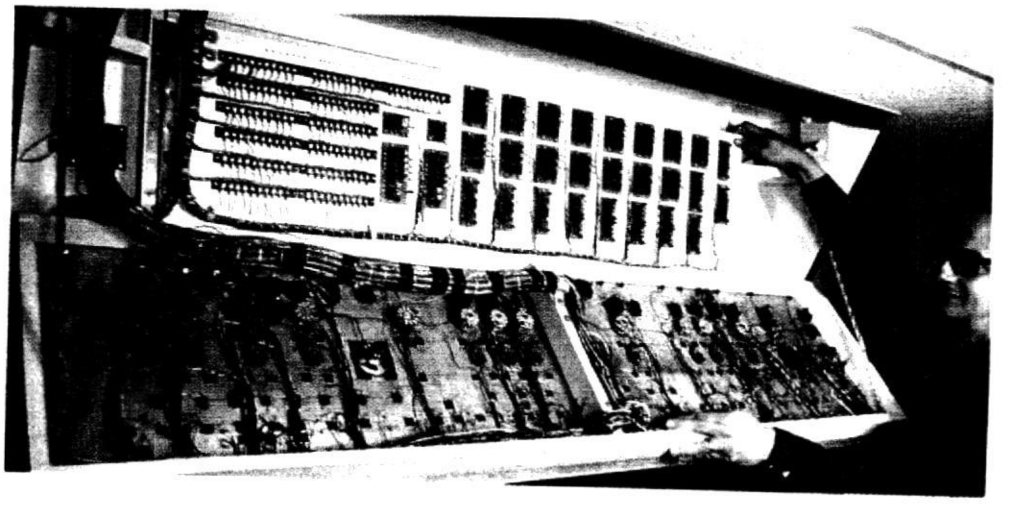
I've presented these examples of Computer Cab Control and computerized signaling to show you how the C/MRI can put more fun in your model railroading. Next time I'll explain how a C/MRI-equipped computer can simplify the wiring on your railroad and perform diagnostic tests automatically. ◘

C/MRI-13



The C/MRI: A computer/model railroad interface

Part 13: Simplified wiring and automated diagnostics



BY BRUCE CHUBB

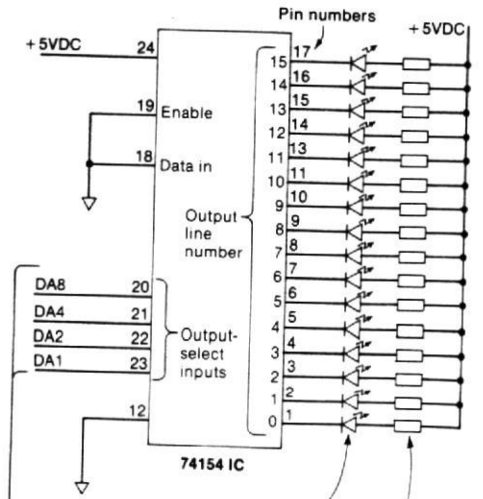
THE C/MRI can perform control or signal functions with many fewer wires than an equivalent noncomputer system, and its modular, plug-in circuit cards help keep your wiring neat and well organized. This month I'll show you how to make it simpler and easier still, first with decoding and encoding circuits that control or monitor groups of devices over single C/MRI I/O lines. That can lower the cost of the C/MRI — fewer I/O Cards — and reduce the wiring for a large railroad.

Then I'll explain how the computer can help you find and fix almost any troubles which develop. Since solid-state systems are very reliable to begin with, you'll have a much easier time maintaining the wiring that's left.

SIMPLIFIED WIRING

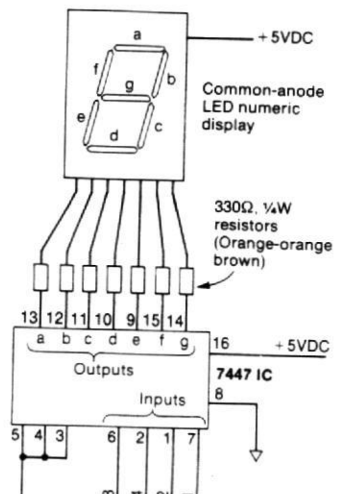
Yes, the C/MRI really can simplify your wiring. Take, for example, the Sunset Valley signal system published in the October 1970, April 1972, and May 1972 issues of MODEL RAILROADER. It worked well for years, but its complex, precomputer wiring involved thousands of wires and relay contacts, numerous multideck rotary switches, and multiple contacts on each switch machine — I had close to 24 wires running to every mainline powered turnout.

For all that, it only approximated the behavior of a real three-color signal system, and if a problem developed, like a dirty contact, it could take me a couple of hours to find the fault and fix it. The system was just too complicated, with wires running all over the place.



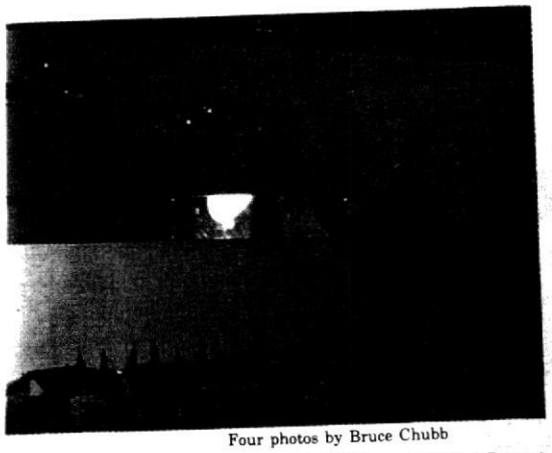
Four C/MRI output lines and a 74154 IC drive up to 16 display LEDs

Fig. 1 DECODING OUTPUTS



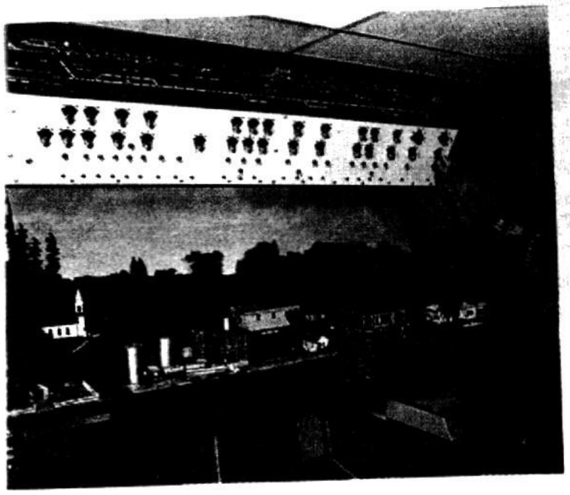
Four C/MRI output lines and a 7447 IC drive a seven-segment decimal display

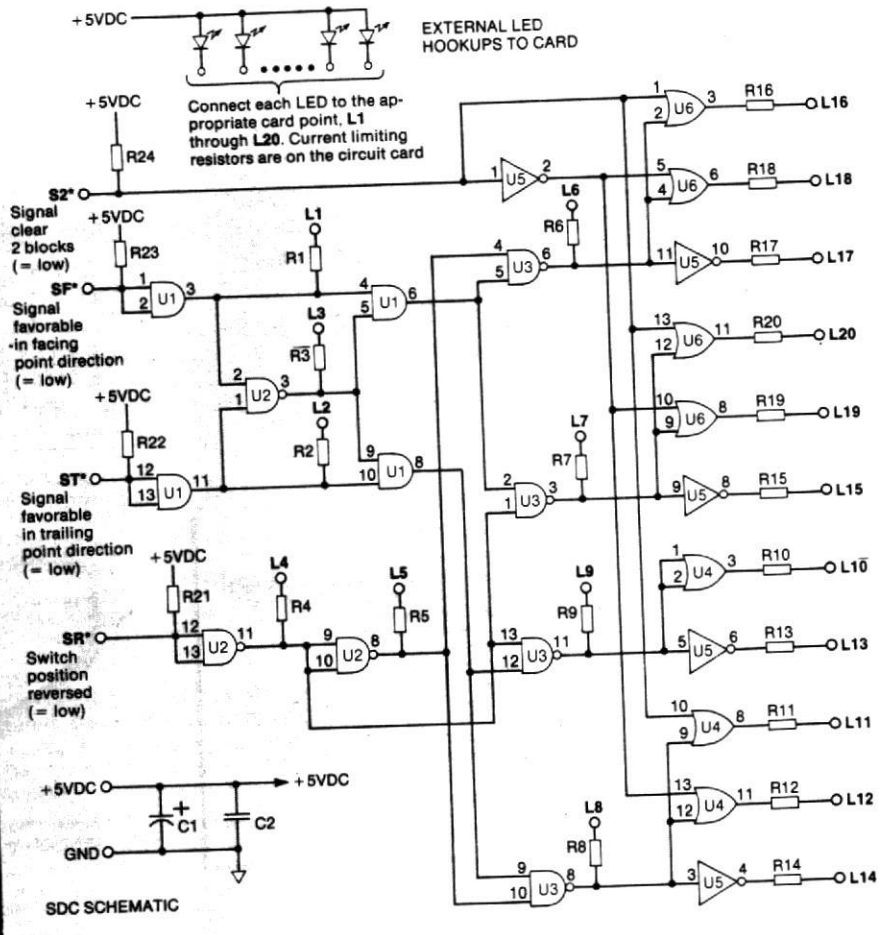
Fig. 2 DECODING FOR NUMERIC DISPLAY



Four photos by Bruce Chubb

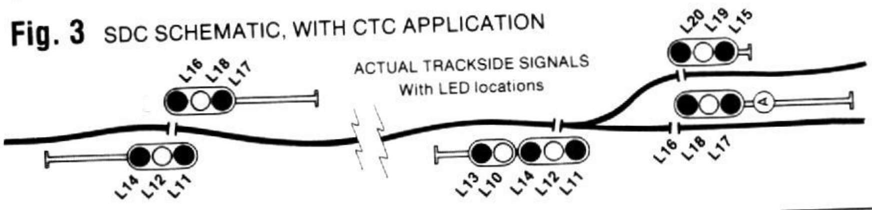
:e and Janet Chubb show how the C/MRI has simplified Sunset
ɜy wiring. The old Walnut Hill panel, above, was complex and
using inside, and controlled only about a quarter of the rail-
1. Below is the new panel, an auxiliary CTC board which can
:rol the whole line, and its wiring is neat and easy to follow.

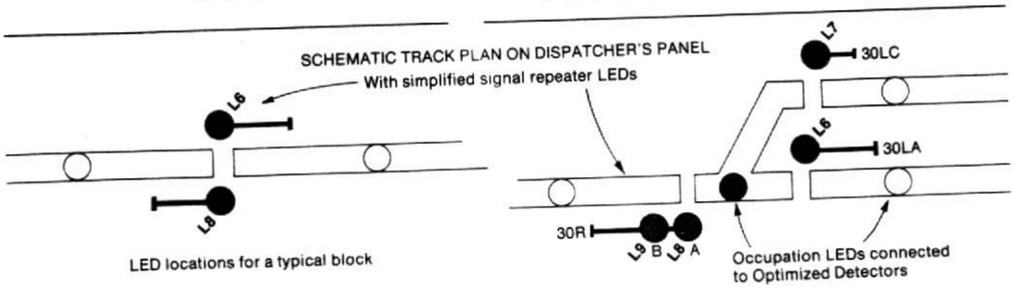




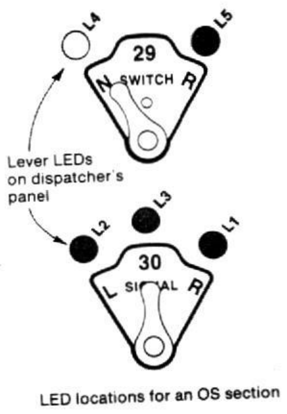
+5VDC
EXTERNAL LED HOOKUPS TO CARD
Connect each LED to the appropriate card point, L1 through L20. Current limiting resistors are on the circuit card
S2* Signal clear 2 blocks (= low)
SF* Signal favorable in facing point direction (= low)
ST* Signal favorable in trailing point direction (= low)
SR* Switch position reversed (= low)
GND
C1
C2
SDC SCHEMATIC
L16
L18
L17
L20
L19
L15
L10
L13
L11
L12
L14

Fig. 3 SDC SCHEMATIC, WITH CTC APPLICATION





SCHEMATIC TRACK PLAN ON DISPATCHER'S PANEL
With simplified signal repeater LEDs
L6
L8
LED locations for a typical block
L7
30LC
L6
30LA
30R
L9 B L8 A
Occupation LEDs connected to Optimized Detectors



L4
L5
29
N SWITCH R
Lever LEDs on dispatcher's panel
L2
L3
L1
30
L SIGNAL R
LED locations for an OS section

LED LOGIC TABLE

✓	INPUTS				LED OUTPUTS																			
	SF*	ST*	SR*	S2*	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	L11	L12	L13	L14	L15	L16	L17	L18	L19	L20
	I	I	I	I			●	●									●	●	●		●			
	I	I	I	O			●	●									●	●	●		●			
	I	I	O	I			●		●								●	●	●		●			
	I	I	O	O			●		●								●	●	●		●			
	O	I	I	I	●			●				●				●	●		●		●			
	O	I	I	O	●			●				●			●		●		●		●			
	O	I	O	I	●				●				●	●				●	●		●			
	O	I	O	O	●				●				●	●				●	●		●			
	I	O	I	I		●		●		●							●	●	●			●		
	I	O	I	O		●		●		●							●	●	●	●				
	I	O	O	I		●			●		●						●	●			●		●	
	I	O	O	O		●			●		●						●	●			●			●

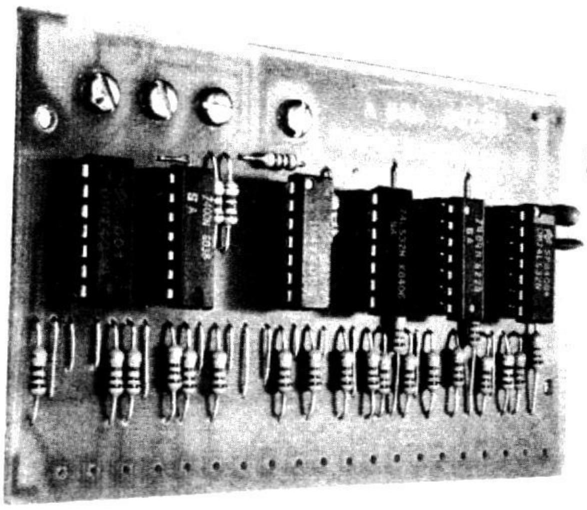
The C/MRI simplified everything, and the hundreds of relays are now for sale. Each of those mainline turnouts now has only a single wire to signal its position, close to a 24:1 reduction. Software signal logic instead of relays has reduced mechanical contacts by nearly 1000:1.

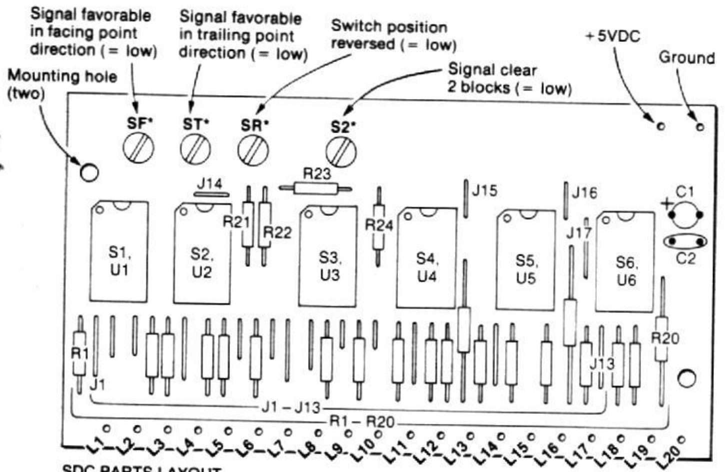
The photos above demonstrate the contrast. The old Walnut Hill tower controlled only a quarter of the railroad, but the new Walnut Hill tower is an auxiliary CTC panel which can control the whole railroad. Even so, it has less wire, simpler switches, and only one relay. Not only is the wiring simpler, but the computerized signaling is much more realistic than the old relay system.

REDUCING LAYOUT WIRING

So far, I've pretty much stuck with one C/MRI I/O line (wire) for each device controlled or monitored in CCC and computerized signaling. This fully parallel wiring is simple and easy to debug, but a large layout will require a large number of C/MRI I/O cards and will need long cables with many wires.

For example, the SV has 1400 feet of track, a fully signaled main line with 54 blocks, 1500 LEDs, overhead track diagram panels, a CTC dispatcher's panel,




Signal favorable in facing point direction (= low)
Signal favorable in trailing point direction (= low)
Switch position reversed (= low)
+5VDC
Ground
Signal clear 2 blocks (= low)
Mounting hole (two)
SF*
ST*
SR*
S2*
J14
R23
J15
J16
C1
R21
R22
R24
J17
C2
S1, U1
S2, U2
S3, U3
S4, U4
S5, U5
S6, U6
R1
R20
J1
J13
J1 – J13
R1 – R20
L1 L2 L3 L4 L5 L6 L7 L8 L9 L10 L11 L12 L13 L14 L15 L16 L17 L18 L19 L20

SDC PARTS LAYOUT

SDC PARTS LIST (In order of assembly)

Qnty.	Symbol	Description
4	—	4-40 x ¼" pan head machine screws with brass nuts
17	J1-J17	Jumpers, make from no. 24 uninsulated bus wire (Belden no. 8022 or equivalent)
20	R1-R20	160Ω resistors [brown-blue-brown]
4	R21-R24	2200Ω resistors [red-red-red]
6	S1-S6	14-pin DIP sockets (Digi-Key C8914)
1	C1	2.2µF, 35V tantalum capacitor (Jameco TM2.2/35)
1	C2	.1µF, 50V ceramic disk capacitor (Jameco DC.1/50)
1	U1	7408 quad two-input AND gate (Jameco SN7408N)
2	U2, U3	7400 quad two-input NAND gate (Jameco SN7400N)
2	U4, U6	7432 quad two-input OR gate (Jameco SN7432N)
1	U5	7404 hex inverter (Jameco SN7404N)

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

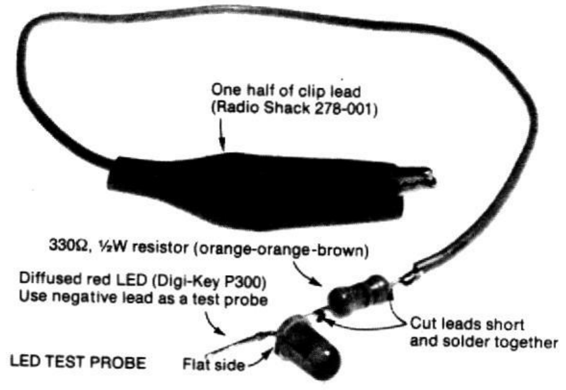

One half of clip lead (Radio Shack 278-001)
330Ω, ½W resistor (orange-orange-brown)
Diffused red LED (Digi-Key P300) Use negative lead as a test probe
Cut leads short and solder together
LED TEST PROBE
Flat side

Fig. 4 BUILDING THE SDC

and other features. I use 23 I/O Cards, and wires from the C/MRI to devices "in the field" are up to 100 feet long.

True, most lines carry low currents, such as LED outputs and panel switch inputs, and surplus multiconductor telephone cable is adequate. That helps keep wiring neat, and the prebundled, color-coded cables are easy to use. Talk to people renovating an old office or factory building, and you can often have what you need for the asking. Don't use telephone cable for train or switch machine power, however; it's too small for that.

Still, the smart way is to run no more wire than you need for a given job. With the C/MRI there are many things you can do to reduce layout wiring. For larger railroads, the most effective is to use a serial interface driving multiple I/O Mother Boards (IOMBs) distributed around your layout. I'll explain serial interfacing in the next two C/MRI installments, but simple I/O decoding and encoding circuits can also make significant wiring reductions. Let's look at decoding circuits first.

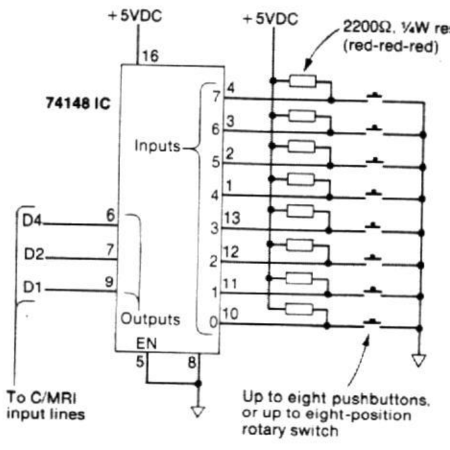
DECODING OUTPUT

Assume you have a yard with 16 tracks and you want your computer to light an LED on your panel to show which track is aligned. Using the methods we've explained, you'd run 16 wires and tie up 16 lines of one Output Card. However, you can do the job with only 4 wires as shown in fig. 1.

Since you need only 4 lines to define up to 16 binary codes, you can use a 74154 IC, a 1-of-16 data distributor, to get a 1-low-out-of-16 output whose address is selected by the binary code on the 4 input address lines. Send out a decimal 4 (binary 0100) on the input lines and the LED for track 4 will light. Send out a 9 (binary 1001) and track 9's LED would light.

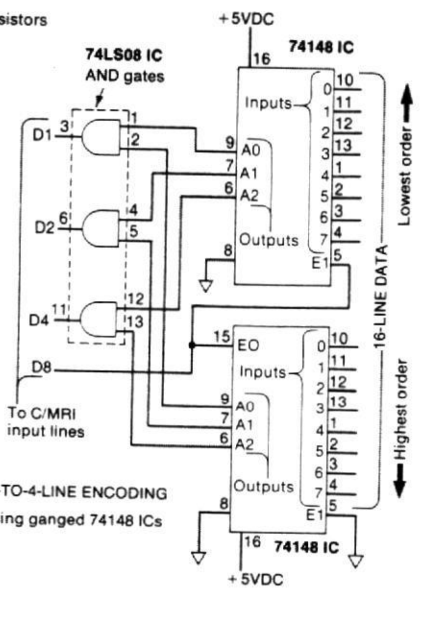
The same circuit will work, of course, for fewer tracks, and if necessary you could cascade 74154s to control up to 32 devices with 5 wires and up to 64 devices with only 6. For 10 or fewer tracks you can substitute a 7442 IC for the 74154.

Rather than have a separate LED for each track, you could use a single seven-


+5VDC
+5VDC
2200Ω, ¼W resistors (red-red-red)
74148 IC
Inputs
Outputs
EN
D4
D2
D1
To C/MRI input lines
Up to eight pushbuttons, or up to eight-position rotary switch

8-TO-3-LINE ENCODING

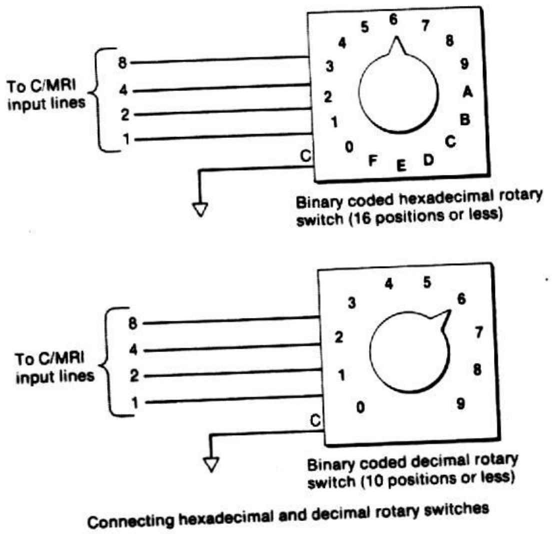
Encoding up to eight pushbutton (or rotary switch position) inputs to three C/MRI input lines



16-TO-4-LINE ENCODING

Using ganged 74148 ICs

Fig. 5 ENCODING INPUT



To C/MRI input lines
8
4
2
1
C
Binary coded hexadecimal rotary switch (16 positions or less)
To C/MRI input lines
Binary coded decimal rotary switch (10 positions or less)
Connecting hexadecimal and decimal rotary switches

Fig. 6 BINARY CODED SWITCH INPUT

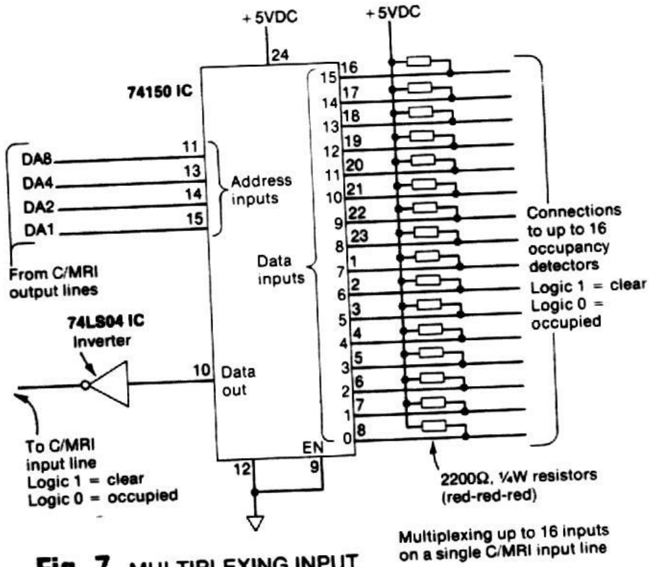


Fig. 7 MULTIPLEXING INPUT

segment numeric display as in fig. 2. It uses a 7447 IC, a BCD-to-seven-segment decoder/driver, to decode the four input lines and light LED segments to display the number sent on the four input lines.

The same decoding-at-destination approach can control other devices, too. You can use the 7442 or the 74154, for example, to provide control logic for automatic turntable alignment with one of up to 10 or 16 tracks, respectively. Or they could control a diode matrix for automatic alignment of a yard throat with up to 10 or 16 tracks.

Other general or specialized decoder circuits can handle other cases, but you should evaluate each one on its own merit. Parallel wiring with multiconductor cable is often easier to connect and debug, it requires no special circuitry outside of your C/MRI, and it can often be the more cost-effective solution.

On the SV I use some direct wiring — one line/one device — and some decoding at destinations. An example of the latter is that I use only two C/MRI output lines to control the group of nine LEDs at each OS section on the CTC dispatcher's panel (compare Part 11, fig. 7). A balanced approach with some direct control and some decoding generally gives the optimum utilization of your C/MRI.

SIGNAL DECODER CARD

Where you do want to use decoding, my Signal Decoder Circuit (SDC) can come in handy. It's an outgrowth of the circuit I use for those OS-section panel LEDs. With three extra ICs and the option for a third input line from your C/MRI, it's a universal circuit that can control up to 20 LEDs using only 3 C/MRI output lines. It can handle trackside and panel signals, either two- or three-color, for an OS section or any pair of adjacent block signals.

Figure 3 shows the SDC schematic and the LED connections for application to a CTC OS section or block signals. For the OS section, lines 1-3 are for the signal lever, 4 and 5 for the switch lever, 6-9 for simplified signal repeaters on the CTC panel track model, and 10-20 for the signals. LEDs 6-9 light green anytime the computer sets a favorable indication (yellow or green) for the more complete trackside signals. As you can see in fig. 3, the block signal case is simply a subset of the OS section.

The SDC has four inputs, all of which are active low. The three coming from the C/MRI are *S*ignal clear in *T*railing point direction (ST*), *S*ignal clear in *F*acing point direction (SF*), and *S*ignal clear *2* blocks ahead (S2*). The fourth input comes from an SPST contact on the switch machine showing *S*witch *R*eversed (SR*).

The truth table in fig. 3 defines the 20 different LED output states as functions of the 4 inputs. To use two-color signals, simply connect the S2* input to logic ground so that you never get an intermediate yellow aspect.

The benefits of the SDC are easy to see. Say you have 10 OS sections and want to use three-color signals as in Part 12, fig. 1. That's 11 trackside LEDs per OS section, or 110 total. With SDC cards you need only 30 C/MRI output lines to control these LEDs, *vs.* 110 lines with direct wiring. Even with the cost of the SDCs, that still gives almost a 3:1 cost reduction and nearly a 4:1 reduction in the amount of wiring.

Figure 4 shows the parts layout, parts list, and a photo of the completed SDC. Ready-to-use SDC cards are available from JLC Enterprises, P. O. Box 88187, Grand Rapids, MI 49518. Card SDC is $8, and there is a shipping and handling charge of $2.50 per order (Michigan residents must add 4 percent sales tax).

You can leave off the resistors for any LED outputs you don't want to use, and if you are using the circuit only for the LEDs on the CTC panel, leave off parts S4-S6, U4-U6, R10-R20, and R24. Here's how to assemble a "full-up" SDC.

☐ **Card inspection.**

☐ **Terminal screws.** Insert four 4-40 screws from the top side, tighten four nuts on the bottom, and solder nuts to traces.

☐ **J1-J17.**

☐ **R1-R24.**

☐ **S1-S6[+].**

☐ **C1[+].**

☐ **C2.**

☐ **U1-U6[+].**

☐ **Cleanup and inspection.**

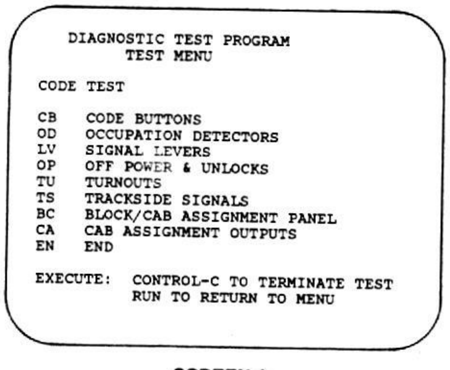
☐ **Card test.** Temporarily solder a test lead to the GND hole and another to the +5VDC hole. Connect these to the +5V supply and turn on the power. Make an LED test probe as shown in fig. 4, attach the clip to the +5VDC output from your supply, and sequentially touch the LED end to each of the 20 LED output pads. The test LED should light only for positions L3, L4, L13, L14, L15, and L17 as indicated by the top row in the logic table.

Arrange clip leads to ground the input screws according to each of the remaining rows in the logic table, attaching a lead where the table has a 0 (logic 0 = ground), and removing any lead where the table has a 1. For each case check all 20 pads, or all with parts installed, and make sure only those with a dot cause the test LED to light. Check the table in fig. 3 as you pass each test. If you have problems, look for a faulty solder joint, a part left out, or a solder bridge. Correct and continue until you pass all tests.

ENCODING INPUT

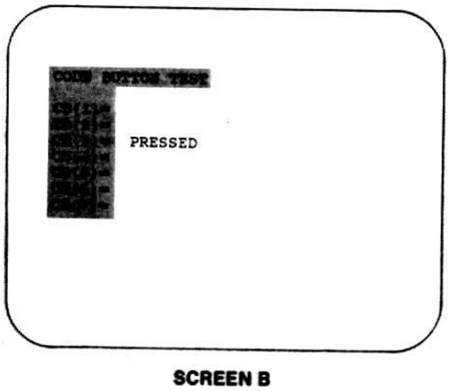
Encoding data at the source can also make significant wiring reductions. Say you have eight pushbuttons you want the computer to read, turnout-control buttons on a yard panel or code-start buttons on a dispatcher's CTC panel. Again, rather than running eight wires and tieing up eight Input Card lines, you can use just three wires as shown in fig. 5.

The 74148 IC, an eight-bit priority encoder, puts out a three-line binary equivalent of the most significant (largest) input. All inputs and outputs are active low, and two chips can be ganged together to provide 16-line-to-4-line encoding, as also shown in fig. 5. You could sense up to 16 positions of a rotary switch with only 4



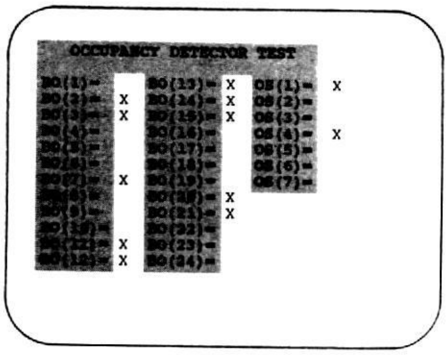
```
     DIAGNOSTIC TEST PROGRAM
            TEST MENU

CODE  TEST

CB    CODE BUTTONS
OD    OCCUPATION DETECTORS
LV    SIGNAL LEVERS
OP    OFF POWER & UNLOCKS
TU    TURNOUTS
TS    TRACKSIDE SIGNALS
BC    BLOCK/CAB ASSIGNMENT PANEL
CA    CAB ASSIGNMENT OUTPUTS
EN    END

EXECUTE:   CONTROL-C TO TERMINATE TEST
           RUN TO RETURN TO MENU
```

SCREEN A



```
CODE BUTTON TEST
CB(1)=
[illegible]
                 PRESSED
```

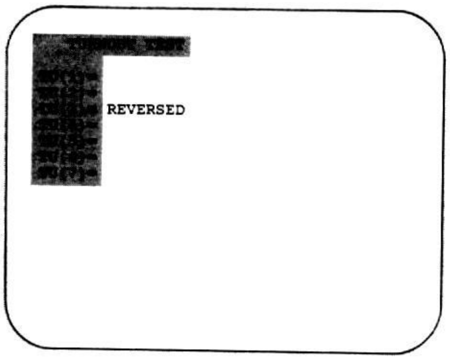
SCREEN B



```
    OCCUPANCY DETECTOR TEST
BO(1)=       BO(13)=  X   OS(1)=    X
BO(2)=   X   BO(14)=  X   OS(2)=
BO(3)=   X   BO(15)=  X   OS(3)=
BO(4)=       BO(16)=      OS(4)=    X
BO(5)=       BO(17)=      OS(5)=
BO(6)=       BO(18)=      OS(6)=
BO(7)=   X   BO(19)=      OS(7)=
BO(8)=       BO(20)=  X
BO(9)=       BO(21)=  X
BO(10)=      BO(22)=
BO(11)=  X   BO(23)=
BO(12)=  X   BO(24)=
```

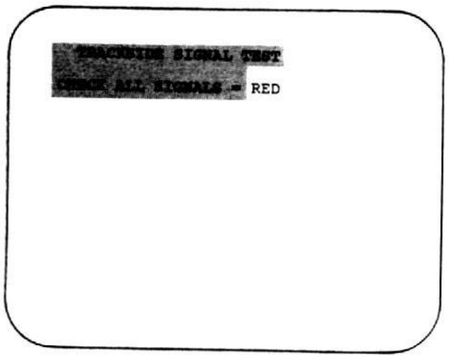
SCREEN C

Fig. 8 DIAGNOSTIC DISPLAYS



```
TURNOUT TEST
[illegible]
              REVERSED
```

SCREEN F



```
TRACKSIDE SIGNAL TEST
CHECK ALL SIGNALS = RED
```

SCREEN G

input lines to your C/MRI. If you have 9 or 10 lines to encode you could use one 74147 IC, a 10-bit-to-4-line encoder, in place of the two 74148s.

Rather than build your own encoding circuits to connect to panel switches, you could use digital-type switches that have the encoding built in. For example, the cab selector switch in the CTC-16e walk-around T-bus throttle is a 16-position hexadecimal rotary switch with positions 0-9 and A-F, but only 4 outputs to carry the 16 binary codes for its 16 positions.

Such a binary-coded hexadecimal or decimal switch can be directly connected to your C/MRI input card as shown in fig. 6, providing up to 16 or 10 inputs with only 4 wires. At the software end you simply read the decimal number of the switch position. You can't get things easier than that!

You can also employ a few output lines to make significant reductions in the number of input lines. The output lines are used as address lines to select at the source what particular input is to be read. The TTL chips that perform this selection for us are appropriately called data-selectors/multiplexers.

These ICs can be thought of as one-way logic versions of a toggle or rotary switch. The 74150 single 16-position selector, for example, works like an automatic 16-position rotary switch. The software sends a binary code on 4 address lines, representing a decimal number from 0 through 15, to set the "switch" to position for reading the desired 74150 input line. In addition to the 74150, you can get four 2-position selectors per package (74157), two 4-position selectors (74153), or one 8-position selector (74152).

As an example of how the data selector/multiplexers can cut down the number of I/O lines to your C/MRI, suppose you have up to 16 remote staging tracks, each with an occupancy detector, and you want your computer to periodically sample the detectors to determine which tracks are clear. Instead of running all 16 detector output lines directly to 16 inputs on an Input Card, you could use a 74150 IC to multiplex the 16 detector output lines onto a single C/MRI input line. See fig. 7.

In operation, the software would send binary code N on the four address lines to read the state of track N from the single input line. For example, to sample track 5 it would send 0101 (binary 5) on the address lines and read the input line.

Since with the C/MRI we always read a port at a time, and a port is eight bits, you could group eight 74150s together, driving them all from the same four address lines, but connecting each of the selector outputs to a different line on the same C/MRI input port. This way your software could increment through sending 16 different addresses (0-15) and sequentially read back the equivalent of 16 different input ports, but all on a single port. That's the same as 128 (16 x 8) discrete lines. You can use the other data-selector/multiplexer ICs I listed above to set up multiplexing schemes of your own.

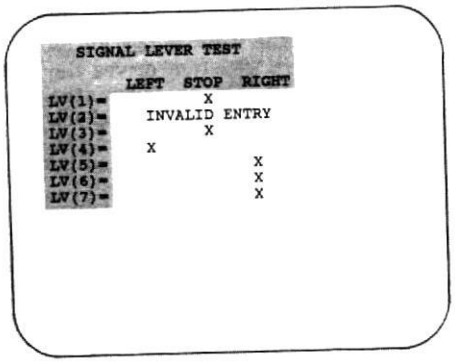
There are similar one-way "switches" you could use to distribute C/MRI output. Called data-distributors, these ICs include such chips as the 74154, which takes one input line and routes it to one of 16 output lines. However, the output lines are not latched, so data isn't retained when you switch addresses. Without adding even more circuitry, these devices won't help much to expand the output capability of the C/MRI.

Any serial/multiplexing scheme reduces the number of individual wires compared with an all-parallel approach, but you add complexity and lose speed both in the hardware and software. These are trade-offs you have to make on your own, but when in doubt, I suggest that you lean toward using one wire per device. It's simpler to use and easier to understand.

AUTOMATED DIAGNOSTICS

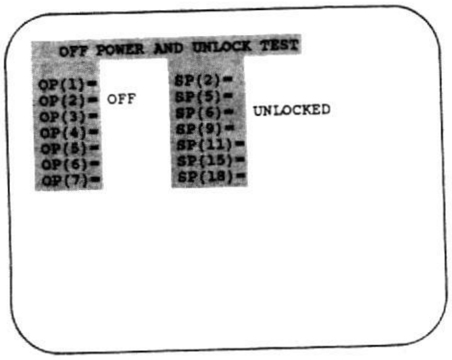
Through the C/MRI your computer has the power to check out your railroad's electrical system. I've used such an automated diagnostic system on the SV since I installed my computer in 1980, and it has saved me hundreds of hours of testing and debugging. Without the computer I could easily spend an hour or two running a train while throwing all combinations of panel switches to make sure that everything worked. Even so, there would be many situations left unchecked. If I did find a problem, say a lamp that wouldn't light, it would still take some time to track down the exact fault.

With the C/MRI you can make a very complete check of an entire layout's electrical system in less than 10 minutes. And by complete I mean checking every panel switch, pushbutton, LED, trackside signal, occupancy detector, and switch machine — both for control and position



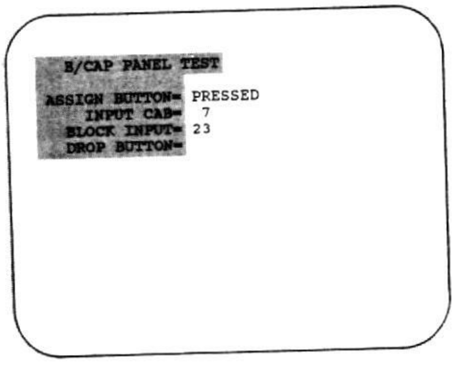
```
   SIGNAL LEVER TEST

           LEFT  STOP  RIGHT
LV(1)=             X
LV(2)=     INVALID ENTRY
LV(3)=             X
LV(4)=     X
LV(5)=                   X
LV(6)=                   X
LV(7)=                   X
```

SCREEN D



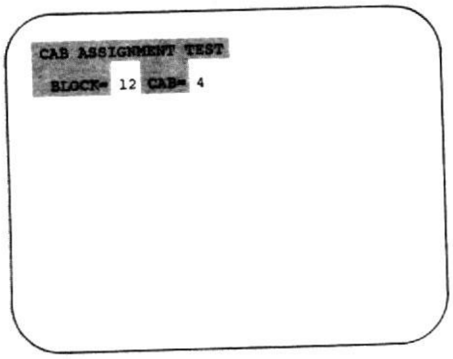
```
 OFF POWER AND UNLOCK TEST

OP(1)=            SP(2)=
OP(2)= OFF        SP(5)=
OP(3)=            SP(6)=  UNLOCKED
OP(4)=            SP(9)=
OP(5)=            SP(11)=
OP(6)=            SP(15)=
OP(7)=            SP(18)=
```

SCREEN E



```
  B/CAP PANEL TEST

ASSIGN BUTTON= PRESSED
    INPUT CAB=  7
  BLOCK INPUT= 23
  DROP BUTTON=
```

SCREEN H



```
CAB ASSIGNMENT TEST

   BLOCK= 12  CAB= 4
```

SCREEN I

feedback. For CCC you can check all CBAC panel inputs, Cab Display Cards (CDCs), and Cab Relay Cards (CRCs).

The tests include the wiring from each device to the I/O Cards, the I/O Cards themselves, the IOMB, the UBEC, and the CBAC, and all the DIP switches used to set card addresses. That's a thorough test and all in 10 minutes — indeed a lot of power!

Digital simplicity, documentation, and plug-in card replacement make C/MRI diagnosis an easy way to isolate and repair faults. When I find a problem with the diagnostic program, I usually have the defect located and repaired in a tenth of the time it took before the computer. And, since the C/MRI and its field devices are an almost all-solid-state system, there are many fewer failures in the first place. It's a much better system that gives me more time to enjoy railroading.

EXAMPLE DIAGNOSTIC DISPLAYS

Figure 8 shows example CRT displays from the SV diagnostic program, with some added to include testing for CCC. I won't detail the code, because once you've programmed CCC or signaling, the tests are simple. Screen A is the test menu that appears when you activate the program, listing all the tests you can perform. You simply key in any of the two-character alpha codes followed by a RETURN and the program branches to that test.

The programming for this branching follows the procedure used for simulating railroad input in Part 10. Entering a CONTROL-C command at any point ends the test in process, and then RUN returns you to the test menu. From the menu, entering EN for ENd terminates the diagnostic program and returns you to the operating system so that you are in position to invoke any other program, such as the railroad CCC or signal programs.

In screens B through I, the constant or standing part of each display is shown against a colored background, and the updated part on white. Screen B is the display for code button testing. The word PRESSED appears next to the symbol for any button the computer detects as being pressed, in this case CB(3), but otherwise this part of the display is blank.

By pushing each button in sequence and watching the CRT you see that each button is working properly, including all wiring back to the computer. If your panel is remote from the CRT you can have your diagnostic program blink a panel LED or beep the annunciator horn as you press each button, in addition to updating the CRT. This makes testing even faster, as you won't have to look at the CRT unless you detect a problem.

Screen C is the display for the occupancy detector test: the Xs show which blocks or sections are read as occupied. By running a train, or (faster but not as much fun) by touching a resistor across each block and OS section, you verify the operation of each detector and its wiring. With Optimized Detectors, use a resistor of the specified value for your maximum sensitivity adjustment, 30KΩ for example, to check the potentiometer setting of each Detector. Again it's also handy to have the horn beep for each new detection.

FULL-LINE HO CATALOG

structures • accessories • rolling-stock

20 pages $1.00 postpaid

ALEXANDER SCALE MODELS

PO Box 7121, Grand Rapids, Mich. 49510

Real RR Antiques & Artifacts

Dress up your train room with choice display pieces from America's leading specialist in diverse non-paper RR collector's items. Wide selection, museum quality (Smithsonian a repeat customer since 1972), satisfaction guaranteed. Send $1 for current issues of RR DISPATCH

Scott Arden—20457-M Hwy 126
Noti, OR 97461 (503/935-1619)

märklin **Maryland's largest antique train and toy store.**

Buy—Sell—Trade
Authorized Marklin dealer.
Visa or Mastercard accepted.
No inquiries without SSAE.

Antique Train & Toy World
3626 Falls Rd., Baltimore, MD 21211
Phone (301) 889-0040

1/2" SCALE PARTS
SHORTLINE FOUNDRY, INC.

Make your R/R special with our cast Zinc/Aluminum details. Sprung ARCH-BAR trucks, 2 types of grabirons w/mounting holes (BRASS), working knuckle couplers, whistle, bell, stake pockets, etc.

Send S.S.A.E. for information to:
412 Oak Glen/Plano, Tx 75074 214-578-9850

N, Z SCALERS

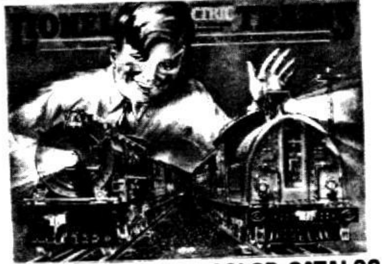
Great prices, fast service on the complete line of KADEE rolling stock, trucks, and coupler conversions. Send a long SSAE for our latest list.

TIMBERLINE ASSOCIATES
P.O. BOX 423 Lombard, Ill. 60148

TAKE COMMAND
DYNATROL®
Independent Train Control

Send stamped self-addressed envelope for info

POWER SYSTEMS INC
56 BELLIS CIRCLE
CAMBRIDGE, MA
02140
(617) 661 0660



LIONEL 1929 FULL COLOR CATALOG

DELUXE REPRODUCTION—46PAGES—CENTER FOLDOUT
Send $16.00 plus $3.00 handling (U.S. & Can.) N.J. add 6%
Foreign (air) $22.00 total. Checks — U.S. Funds/Banks only
Large list of catalogs & paper items $1.00 (Foreign $2.00)
Fine catalogs since 1969. Satisfaction guaranteed

KNOECKLEIN CATALOGS P.O. BOX 662A CHATHAM, N. J. 07928

Screen D is the test display for CTC signal levers. The Xs mark the read-in position of each lever, and you can move each lever to left, stop, and right to test all its connections. The SV program drives the signal-lever panel LEDs to follow the switch position as read, so you can check out the lever and all its wiring into the software and back out to the LEDs just by flipping the switch!

The program prints INVALID ENTRY when both lines for any lever are read as pulled to ground, as for LV(2) in this example. This appears to show that the switch is left *and* right simultaneously, and indicates either a faulty switch or, more likely, a line shorted to ground.

Screen E is the display I use to check the off-power and unlock circuits: flipping each toggle on and off in sequence checks out all actions back through the C/MRI.

Screen F is for the turnout test, which you can tailor to fit your application. If your turnouts aren't computer controlled, just display the state of each turnout-position feedback line. For CTC controlled turnouts, you can also display the lever position, throw the turnout, and set the panel LEDs. Another useful test is to have the program progressively throw each turnout back and forth automatically, as you watch for proper switch point operation and spring tension.

The trackside signal test is on screen G. The CRT display is simple but the test is powerful. The software first initializes all signal LEDs to red, and when you enter a RETURN all reds go out and all yellows go on, then another RETURN turns on all greens. For the SV's approach-lighted signals, another RETURN sets all signals dark.

For a quick overall "lamp test," you could have the program turn on all signal LEDs simultaneously, so you could check all signal LEDs with one walk around the layout. Or you could have the program automatically, without waiting for carriage return input, cycle through all dark, all red, all yellow, and all green. Then you could watch for correct, independent operation of each signal line without manual intervention.

Screen H is an example of a B/CAP test display for a CCC railroad. The two buttons are handled as before, and the numbers read from the thumbwheel switches are displayed. If your B/CAPs aren't near your CRT, have a friend help with this test or build in a parallel form of audio/visual display.

Screen I is a test display for CCC cab assignment. The program would initialize all blocks to no cab assigned, cab number 0. Then for each block 1 through MB, the cab assignment would be switched from 0 to 1 through 7 and back to 0. You would see a "chase light" pattern on the row of CRCs, indicating that each cab relay is being activated.

Well, that's it for this month. I'll be back in the June MR with a serial version of the C/MRI, using a Universal Serial Interface Card (USIC) to control several IOMBs. On large layouts this can save enormous amounts of wire over parallel connections. The USIC also allows interfacing with computers that use RS-232 serial channel I/O. See you then.

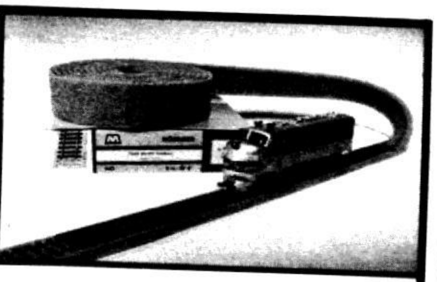
FOAM ROADBED
by Mössmer West Germany

One of the world's finest roadbeds. True to exact scale. Easy to use. No ballasting. Gray color. Sound-absorbent. Premolded.

ORDER NOW

ROADBED—16' 6" Rolls: Also **Märklin & LGB.**
HO Atlas $6.25 (also fits Model Power, Bach., Tyco)
N Atlas $4.75 (also fits Peco, Joeuf)
Z Märklin $4.75
SWITCH UNDERLAYS – in pairs (1 R/H & 1 L/H)
HO Std Atlas #4, $2.50, Mark II #4, $2.50, #6 s, $2.75
N Current #4-2580, $2.50; #6-2558, $2.75
Z 8561 or 8567, $2.50, 8560 or 8559 ea., $1.75

U.S. orders, add $2.50 for postage & handling; for orders over $25.00, add $3.00. Canadian orders, add $4.00; for orders over $25.00, add $4.75. Visa or MasterCard only, add 5% for handling.
DISTRIBUTOR and DEALER TERMS.



Exclusive Importers-USA-Canada
GAMCO PRODUCTS, INC.
2027 Wooddale Dr. NE
Huntsville, AL 35801
Call (205) 534-7836

- 24-Hour Service Year 'Round
- Send SSAE for complete price list
- Specify Scale

N SCALE TRAIN SALE SPECIAL

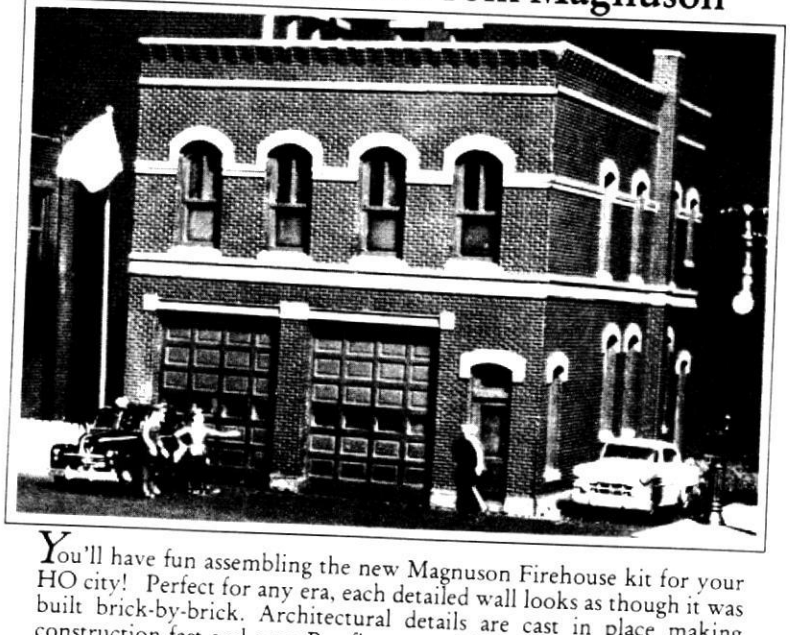
One year subscription (12 issues) of the all N scale Newsletter and an Atlas Davenport 0-6-0 diesel engine valued at $16.00 and $9.00 worth of discount coupons...

ALL FOR
$11.00

DAVENPORT 0-6-0
Rod-Connected Industrial Switcher
$16.00

CAMA TRAIN CO.
P.O. Box 09063, Chicago, IL 60609

A New Firehouse From Magnuson



You'll have fun assembling the new Magnuson Firehouse kit for your HO city! Perfect for any era, each detailed wall looks as though it was built brick-by-brick. Architectural details are cast in place making construction fast and easy. Roofing material, window "glass," modern doors and cast metal chimneys are included. Just paint and weather to give the Firehouse your personal touch! Best of all, the new Firehouse is affordably priced at just $26.95.

You'll find the complete Magnuson line illustrated in *The World of HO Scale—1986* (913-634, $12.95).

Your Dealer can get it from Walthers.

Shown here are the new Victoria Falls Firehouse (6-3/4 × 4-3/8") 439-547 $26.95 and assorted Magnuson vehicles (available separately).

Marketed by Walthers

Magnuson Models Inc.

5601 W. Florist Ave. Milwaukee, WI 53218

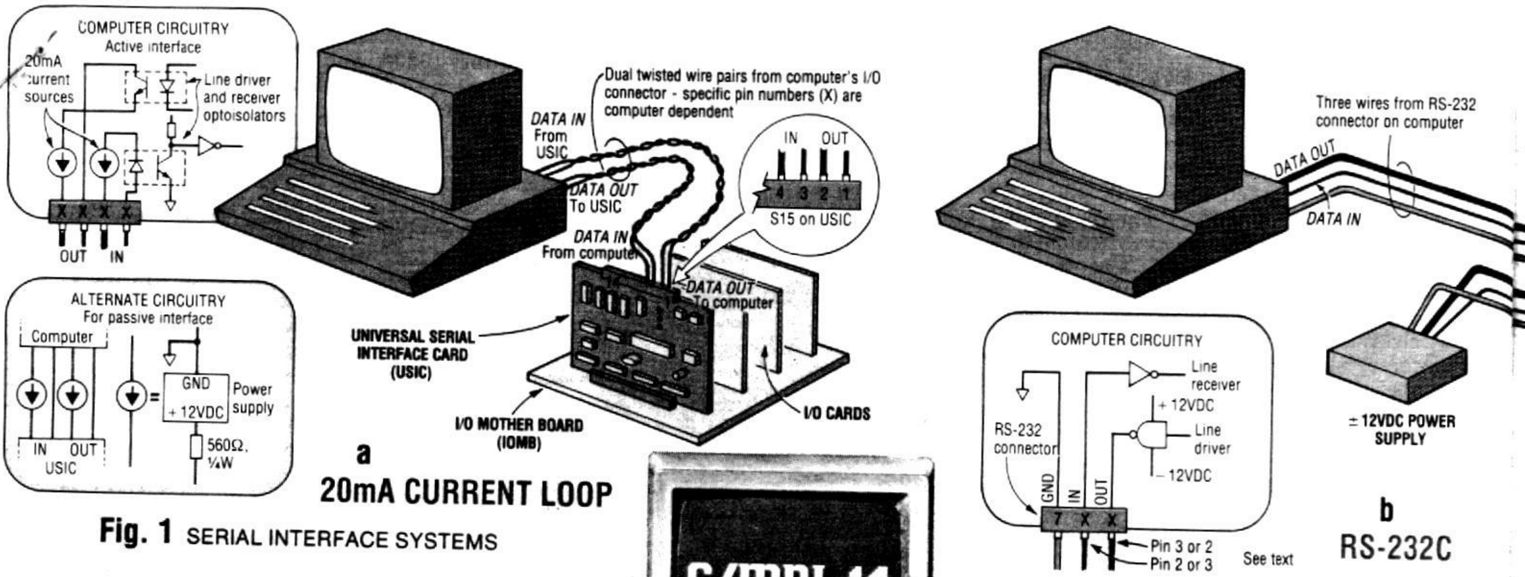


Fig. 1 SERIAL INTERFACE SYSTEMS

C/MRI-14

The C/MRI: A computer/model railroad interface

Part 14: Serial interface hardware

BY BRUCE CHUBB

BACK in C/MRI Part 1, I explained how the computer interface could be parallel or serial, but from then on I've shown only the parallel case. In this installment and the next I'll show you a serial interface, beginning with step-by-step instructions for building a *U*niversal *S*erial *I*nterface *C*ard, or USIC. For a serial C/MRI the USIC replaces the UBEC and CBAC, but the rest of the hardware stays the same.

As I explained in Part 1, the main disadvantage of a serial interface is slower speed, but there are at least three reasons why you might want a serial interface:

Your computer allows only serial I/O. Many computers don't provide internal expansion slots or bring out an expansion connector giving access to the address, data, and control lines needed for the UBEC and its parallel approach. The Apple IIc, Macintosh, and TI/99 are three popular computers in this category. You could bring out the required buses by soldering leads to appropriate traces on the internal circuit cards of these machines, but a serial interface would be easier.

Your computer is remote from your railroad. The UBEC must be close to the computer for direct connection to its high-speed internal bus, and the I/O Mother Board (IOMB) should be within 12 feet of the computer. With a serial interface you can make long-distance connections. For example, with an RS-422A four-wire transmission line, the computer can be as far as 4000 feet from your railroad.

You want minimum layout wiring. A serial interface can let you distribute I/O Cards near the railroad devices they read and control, which takes less wire. This may not matter on average-size layouts, but on large ones it can be a big advantage.

SERIAL INTERFACE STANDARDS

The USIC can use any one of three interface standards: 20mA current loop, RS-232C, or RS-422A. See fig. 1, and note that "data out" from the computer is "data in" to the USIC, and *vice versa*.

• **20mA current loop.** This "de facto standard," fig. 1a, is one of the oldest forms of computer interface and is based on teletype equipment. Many low-cost computers use it for simplicity. Current switched on and off transmits binary 1s and 0s, and coding follows teletype practice where Mark (logic 1) equals a 20mA current, and Space (logic 0) is no current.

Current-loop interfaces use low-impedance circuits and are very immune to electrical noise, making them well suited to our railroad environment. Distance between computer and USIC may be 1000 feet or more; data transmission speeds, called BAUD rates, are typically low.

• **RS-232C.** This most common serial interface, fig. 1b, uses voltage, not current, to define the signal. A Mark (logic 1) is −12V, and a Space (logic 0) is +12V. RS-232C systems use a standard 25-pin, D-type connector called a DB25 or RS-232.

Pins 2 and 3 are the data wires, called Transmitted and Received Data. Which is which depends on whether a device is configured as *D*ata *T*erminal *E*quipment (DTE) or *D*ata *C*ommunication *E*quipment (DCE). DTE transmits on pin 2 and receives on pin 3, but DCE transmits on 3 and receives on 2. Some computers have DTE ports while others have DCE, so the first step in setting up any RS-232C interface is determining the direction of data flow on pins 2 and 3.

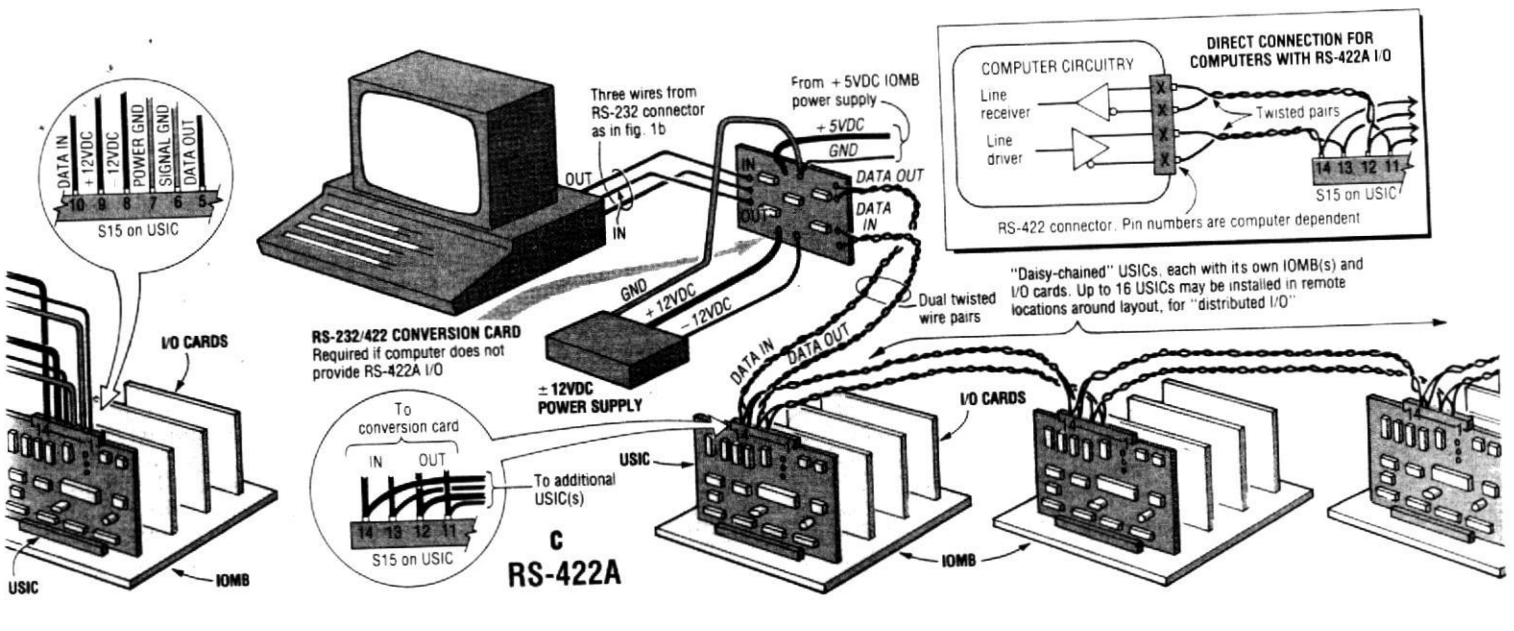
Pin 7 is the most standard of the pins and is the signal ground, the current-return path between the two ends of the transmission. These three wires (pins 2, 3, and 7) suffice for bidirectional communication between two devices. The other 22 pins are control lines for "handshaking" between the computer and the peripheral device, whether it's a printer, a modem, or a railroad. For example, pin 6, Data Set Ready, is one a printer might use to tell the computer it's ready to accept the next character.

The trouble is that there is little standardization in how computers and peripherals use the handshaking lines. Along with 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 RS-232C is sometimes called "the most nonstandard standard."

On the positive side, most computers provide RS-232C ports either as standard equipment or as an option, perhaps from an aftermarket supplier. Even so, a manufacturer's claim that a product is RS-232C-compatible usually only means that it generates and accepts a small fraction of RS-232C signals, and doesn't violate any other parts of the standard.

As fig. 1b shows, the RS-232C is limited to one receiver and one transmitter. That's fine for many applications, but if you want distributed I/O with multiple USICs around your railroad, the RS-232C won't do the job. You'll need to go to the RS-422A, and later I'll show you how to build an RS-232/422 conversion card.

Length of cable also affects RS-232C transmission capability and limits maxi-


DATA IN
+12VDC
−12VDC
POWER GND
SIGNAL GND
DATA OUT
10 9 8 7 6 5
S15 on USIC
I/O CARDS
USIC
IOMB
Three wires from RS-232 connector as in fig. 1b
OUT
IN
From +5VDC IOMB power supply
+5VDC
GND
DATA OUT
DATA IN
GND
+12VDC
−12VDC
RS-232/422 CONVERSION CARD
Required if computer does not provide RS-422A I/O
±12VDC POWER SUPPLY
To conversion card
IN OUT
To additional USIC(s)
14 13 12 11
S15 on USIC
USIC
DATA IN
DATA OUT
Dual twisted wire pairs
I/O CARDS
IOMB
COMPUTER CIRCUITRY
DIRECT CONNECTION FOR COMPUTERS WITH RS-422A I/O
Line receiver
Line driver
Twisted pairs
14 13 12 11
S15 on USIC
RS-422 connector. Pin numbers are computer dependent
"Daisy-chained" USICs, each with its own IOMB(s) and I/O cards. Up to 16 USICs may be installed in remote locations around layout, for "distributed I/O"

C
RS-422A

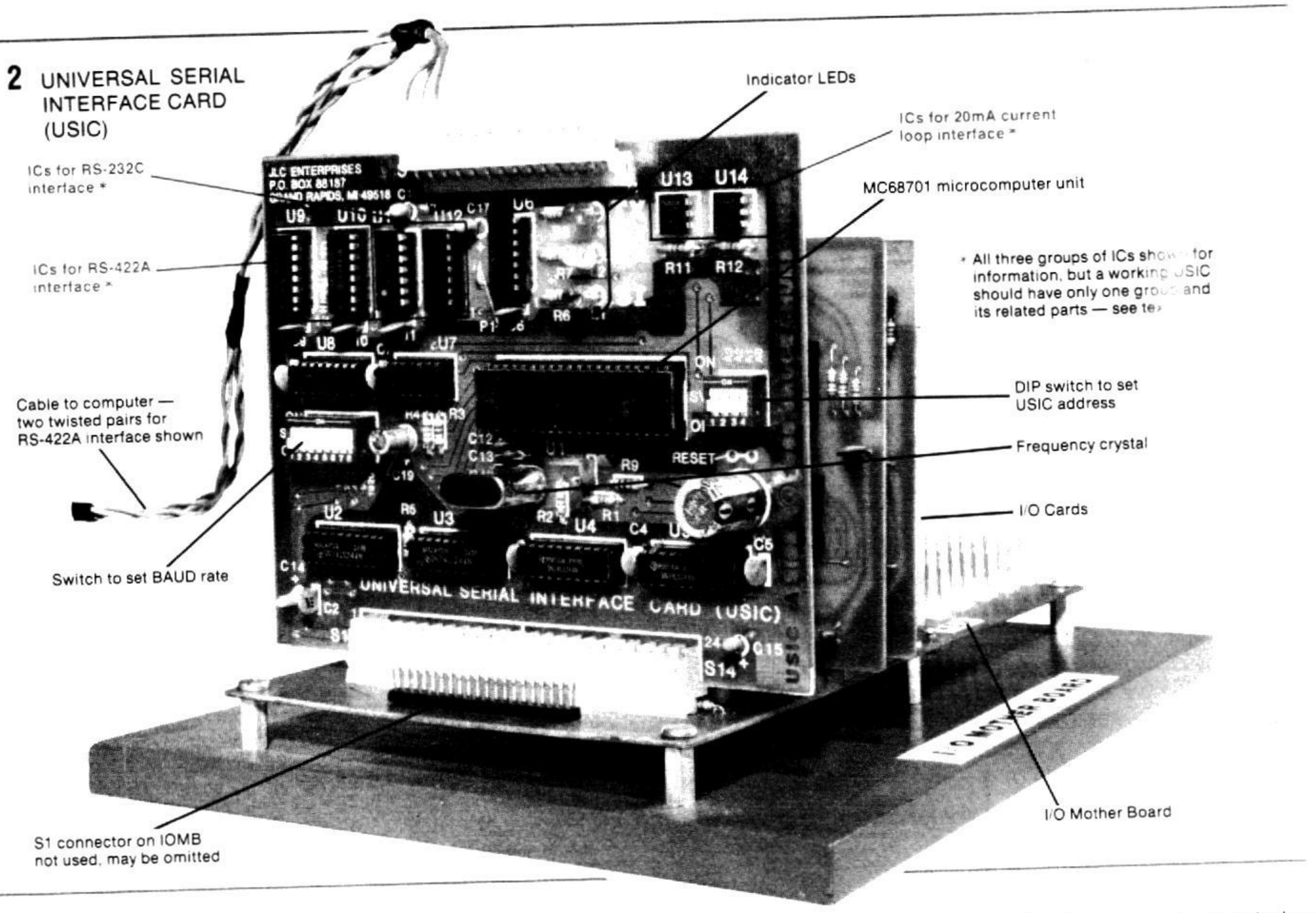

Indicator LEDs
ICs for 20mA current loop interface *
ICs for RS-232C interface *
MC68701 microcomputer unit
ICs for RS-422A interface *
* All three groups of ICs shown for information, but a working USIC should have only one group and its related parts — see text
Cable to computer — two twisted pairs for RS-422A interface shown
DIP switch to set USIC address
Frequency crystal
I/O Cards
Switch to set BAUD rate
I/O Mother Board
S1 connector on IOMB not used, may be omitted

Fig. 2 UNIVERSAL SERIAL INTERFACE CARD (USIC)

mum BAUD rate. The specified maximum cable length with RS-232C is 50 feet without an audio modem (modulator/demodulator) at each end. In practice, however, RS-232C signals are routinely run for 100 to 200 feet and operated at high BAUD rates without problems. Such installations aren't standard, but they do work. For more reliable, error-free transmission over long distances, use RS-422A.

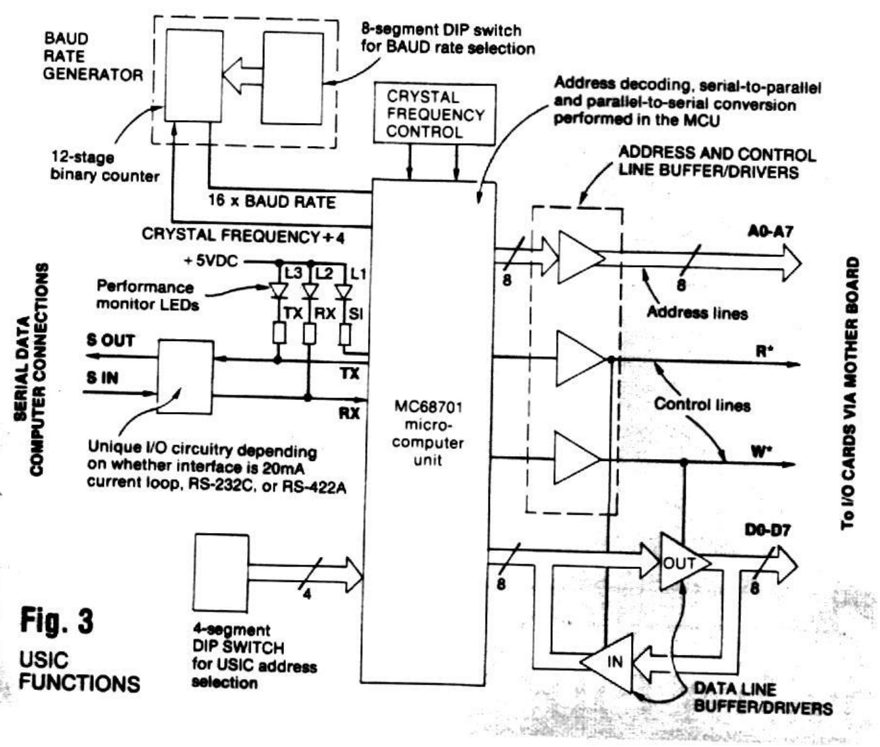
• **RS-422A.** Few personal computers besides the Macintosh provide this interface directly, but its advantages make it worth consideration. Compared to RS-232C it improves noise immunity, lengthens range, and drives multiple receivers from a single transmitter. These features arise from the RS-422's use of twisted-pair, balanced transmission lines, fig. 1c. The difference between the voltages on the two wires signals a Mark or a Space.

If the difference is positive more than 200mV, the receiver reads a Mark; if negative more than 200mV, the receiver reads a Space. A secondary advantage arising from these levels is that the transmitters and receivers can operate with the normal +5VDC logic power supply.

Figure 1c shows an RS-422A system with multiple USICs and two twisted-pair transmission lines, one pair for signals in each direction. The USICs are "daisy-chained," with the same transmission lines running from the computer to the first USIC, on to the second, and so on.

Both RS-232C and RS-422A transmit signals differentially, but RS-232C suffers from using the difference between the signal-wire and ground-wire voltages. This is inferior because each end of the ground wire is grounded. Any difference in potential at the ends of the link makes current flow through the ground wire, and the wire's inevitable resistance ensures a difference in potential that can cause errors in the data.

RS-422A grounding requirements are much less critical since local ground



BAUD RATE GENERATOR
8-segment DIP switch for BAUD rate selection
12-stage binary counter
CRYSTAL FREQUENCY CONTROL
Address decoding, serial-to-parallel and parallel-to-serial conversion performed in the MCU
ADDRESS AND CONTROL LINE BUFFER/DRIVERS
16 x BAUD RATE
CRYSTAL FREQUENCY ÷ 4
+5VDC
Performance monitor LEDs
L3 L2 L1
TX RX SI
SERIAL DATA COMPUTER CONNECTIONS
S OUT
S IN
TX
RX
Unique I/O circuitry depending on whether interface is 20mA current loop, RS-232C, or RS-422A
MC68701 micro-computer unit
A0-A7
8
Address lines
R*
Control lines
W*
D0-D7
OUT
IN
4
4-segment DIP SWITCH for USIC address selection
DATA LINE BUFFER/DRIVERS
To I/O CARDS VIA MOTHER BOARD

Fig. 3
USIC FUNCTIONS

potential doesn't affect the Mark or Space signals. With the ground-potential problem reduced, the RS-422A can send its Mark and Space signals closer together to operate reliably at higher BAUD rates.

It's important to understand that all three 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 our parallel interface which transmits one byte (eight bits) at a time.

In a serial interface the receiving and transmitting devices *must* operate at the same BAUD rate. The USIC is designed to operate with standard BAUD rates of 75, 110, 150, 300, 600, 1200, 2400, 4800, 9600, and 19,200 bits per second. Also, remember that the faster the transmission, the greater the susceptibility to errors from electronic noise.

THE USIC

Figure 2 shows a Universal Serial Interface Card (USIC) plugged into an IOMB. The Mother Board is just as described in C/MRI part 4, but note that with the USIC 40-pin stud connector S1 isn't required. The general purpose I/O Cards plug into the IOMB as before.

The serial connections between computer and USIC are made with the 14-pin Molex connector at the top of the USIC. Which pins you use depend on whether you're using 20mA current-loop, RS-232C, or RS-422A. With RS-232C you must also supply + and −12VDC power through this same connector.

With RS-422A a single computer I/O port can drive up to 16 USICs distributed around your layout, each with its own IOMB(s). Each USIC can drive up to 64 I/O cards at 24 lines each. That's capacity for 24,576 I/O lines! — 16 x 64 x 24.

The heart of the USIC is part U1, an MC68701 *M*icro*C*omputer *U*nit (MCU), making it a "smart interface." The very-capable MC68701 is a single 40-pin DIP containing a clock oscillator, a *Micro-Processor Unit* (MPU), 2048 bytes of ultraviolet-*E*rasable *P*rogrammable *R*ead-*O*nly *M*emory (EPROM), 128 bytes of *R*andom *A*ccess *M*emory or RAM, a *S*erial *C*ommunications *I*nterface (SCI), a programmable timer, and an abundance of I/O pins. It's a self-contained computer system on one chip, giving the USIC lots of power.

The main advantage of a smart serial interface with built-in MPU and memory is great flexibility. The same design works for 20mA current loop, RS-232C, and RS-422A I/O. For RS-232C the interface uses only the pin-2 and -3 data lines, along with the pin-7 ground, so it's independent of the unpredictable RS-232C handshaking lines.

The MC68701 performs all the bookkeeping as to which particular bit the computer is sending and whether it's part of address or data. When the computer is to read data from Input Cards on the IOMB, it simply tells the MC68701, which gathers the data from the cards, sets it up in the prescribed format, and transmits it bit-by-bit back to the computer.

Programming the MC68701 EPROM requires a special piece of hardware, an "EPROM programmer" or "software development system." The MC68701 is programmed by first applying high-intensity ultraviolet light to a window on top of the chip, erasing it to all "0s" (most other EPROMs erase to the opposite state, logic 1). Then the programmer is used to enter "1s" in the desired bit locations. To avoid inadvertent erasure of a programmed chip, the window should be covered with an opaque material to protect against accidental exposure to ultraviolet light.

Don't worry about having to do any EPROM programming yourself. I've arranged to have MC68701 chips programmed for the USIC and tested in a working card by The Chesapeake Group Inc., 7789 C Street, Chesapeake Beach, MD 20732. The cost is $59 each, and there is a $2.50 shipping and handling charge per order. (Maryland residents must add 5 percent sales tax.)

You don't need to understand electronics to build the USIC. Circuit cards and programmed MC68701 chips are readily available, and all you have to do is follow my instructions. But if you are interested in how it works, and in debugging in case of trouble, I'll explain what the USIC does. If you really just want to get started, skip ahead now to "USIC Assembly" on page 104.

USIC FUNCTIONS

Figure 3 is a functional look at what the USIC does. Its main hardware functions are unique I/O line selection, address decoding, BAUD rate generation, parallel-to-serial and serial-to-parallel conversion, and signal buffering. The MC68701 also performs software protocols for handling address, data, and control signals, which I'll cover next month.

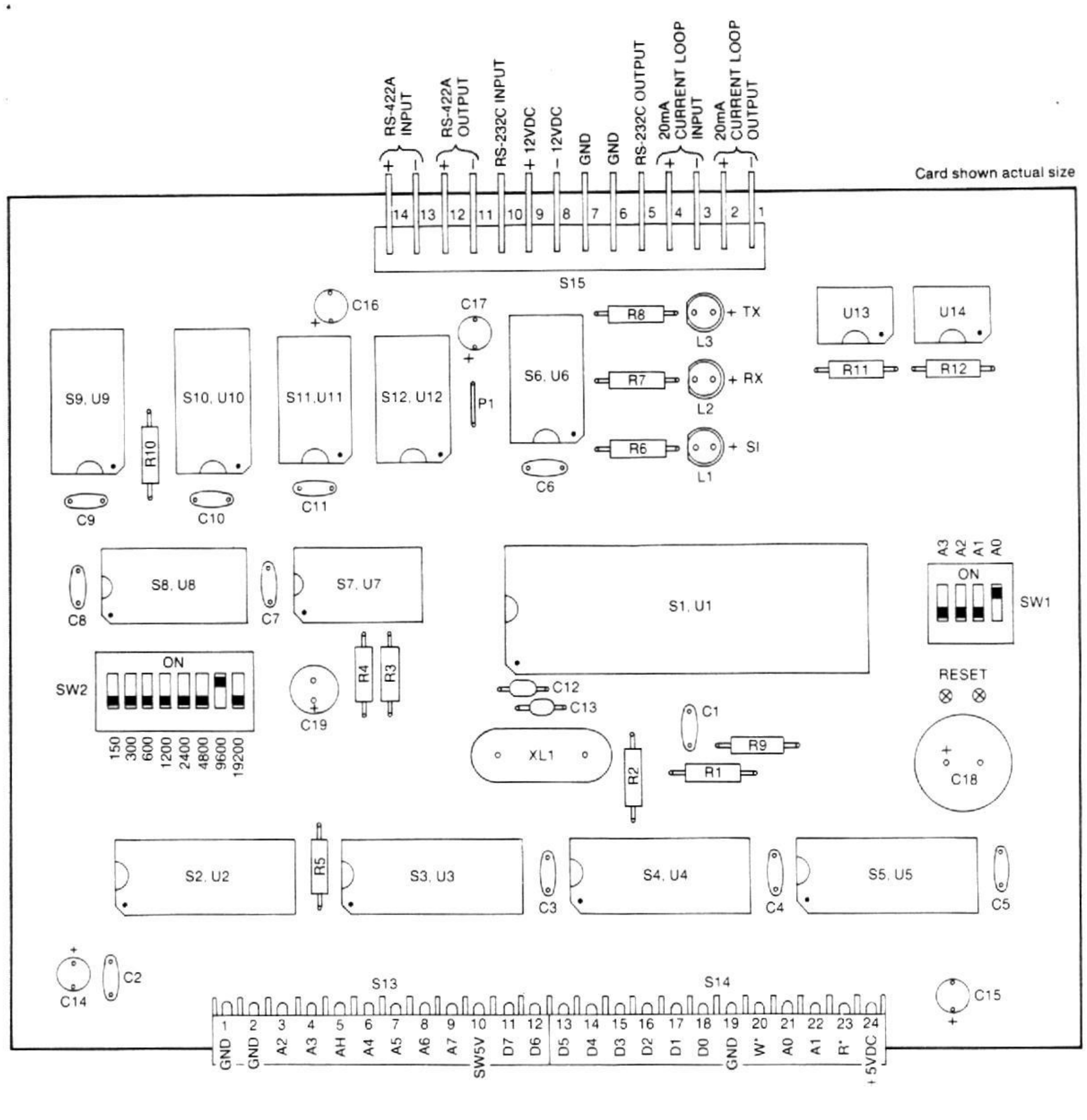
In fig. 3 the arrows indicate the direction of signal flow, and double lines with a slash and number indicate multiple parallel wires — four or eight in our case. The railroad connections are just like the UBEC's, so the USIC plugs right onto the IOMB. Since the USIC is powered from the railroad supply, the RRON sensing line isn't needed, so it's open-circuited or *N*o *C*onnection, NC. The Address High-order bits line AH is also hard-wired on the USIC, to +5VDC.

The I/O address lines, A0 through A7, the two Read and Write control lines, R* and W*, and the eight data lines, D0 through D7, all pass through buffers before going to the IOMB. Since data must flow both ways, the data lines are double buffered. One group of eight buffers, labeled "IN" in fig. 3, handles data flow into the MC68701, and the group labeled "OUT" handles data going to the railroad. The R* and W* lines control which set of buffers is active.

The buffers provide a degree of isolation between the MC68701 and the railroad, as well as the necessary drive capability to handle up to 64 I/O Cards. Without this buffering the railroad's electrical demands would so overload the MC68701 that it couldn't work properly.

One of three special I/O circuits couples the MC68701 to the serial I/O lines coming to USIC from the main computer. Which circuit is used depends on whether you are using 20mA current loop, RS-232C, or RS-422A 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 MC68701's *T*ransmit, TX, and *R*eceive, RX, lines.

To place multiple USICs on the same RS-422A transmission lines, we must be able to set a unique *U*SIC *A*ddress UA for each card. This is handled by a four-segment DIP switch feeding directly into the MC68701. The four-segment switch



USIC PARTS LIST (In order of assembly)

Qnty.	Symbol	Description
2	R1, R2	4700Ω resistors [yellow-violet-red]
1	R3	470Ω resistor [yellow-violet-brown]
1	R4	2200Ω resistor [red-red-red]
1	R5	1000Ω resistor [brown-black-red]
3	R6-R8	330Ω resistors [orange-orange-brown]
1	R9	51Ω resistor [green-brown-black]
1	S1	40-pin DIP socket (Digi-Key C8940)
4	S2-S5	20-pin DIP sockets (Digi-Key C8920)
2	S6, S7	14-pin DIP sockets (Digi-Key C8914)
1	S8	16-pin DIP socket (Digi-Key C8916)
2	S13, S14	12-pin connectors (Molex 09-52-3121)
1	S15	14-pin connector (Molex 09-66-1141)
1	SW1	4-segment DIP switch (Digi-Key CT2064)
1	SW2	8-segment DIP switch (Digi-Key CT2068)
8	C1-C8	.1μF, 50V ceramic disk capacitors (Jameco DC.1/50)
2	C12, C13	22μF, 50V ceramic disk capacitors (Jameco DC22/50)
2	C14, C15	2.2pF, 35V tantalum capacitors (Jameco TM2.2/35)
1	C18	1000μF, 10V radial lead electrolytic capacitor (Digi-Key P6108 or equivalent with .197" lead spacing)
1	C19	100μF, 10V radial lead electrolytic capacitor (Digi-Key P6014 or equivalent with .098" lead spacing)
1	XL1	2.4576mHz crystal (Digi-Key X047)
1	L1	Diffused green LED (Digi-Key P303)
1	L2	Diffused amber LED (Digi-Key P306)
1	L3	Diffused red LED (Digi-Key P300)
1	U1	MC68701 microcomputer unit with C/MRI programmed EPROM (The Chesapeake Group Incorporated 7789 C Street, Chesapeake Beach, MD 20732)
4	U2-U5	74LS244 octal buffer/drivers, noninverting (Jameco)
2	U6, U7	7404 hex inverters (Jameco SN7404N)
1	U8	CD4040 12-state binary/ripple counter (Jameco)

Used only for RS-232C

Qnty.	Symbol	Description
2	S11, S12	14-pin DIP sockets (Digi-Key C8914)
1	C11	.1μF, 50V ceramic disk capacitor (Jameco DC.1/50)
2	C16, C17	2.2μF, 35V tantalum capacitors (Jameco TM2.2/35)
1	U11	LM1489N quad line receiver, RS-232 (Jameco)
1	U12	LM1488N quad line driver, RS-232 (Jameco)

Used only for RS-422A

Qnty.	Symbol	Description
1	R10	1000Ω resistor [brown-black-red]
2	S9, S10	16-pin DIP sockets (Digi-Key C8916)
2	C9, C10	.1μF, 50V ceramic disk capacitors (Jameco DC.1/50)
1	U9	MC3486 quad differential line receiver, RS-422 (Jameco) May substitute LM3486, 26LS32 or SN75175
1	U10	MC3487 quad differential line driver, RS-422 (Jameco) May substitute LM3487, 26LS31 or SN75174

Used only for 20mA current loop

Qnty.	Symbol	Description
1	P1	Jumper, make from no. 24 uninsulated bus wire (Belden no. 8022 or equivalent)
2	R11, R12	330Ω resistors [orange-orange-brown]
2	U13, U14	4N33 optoisolators (Digi-Key)

Mating connector for USIC cable to computer

Qnty.	Symbol	Description
1	—	14-pin connector shell (Molex 09-50-3141)
14	—	Model 247 crimp terminals (Molex 08-50-0106)

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.

Fig. 4 USIC PARTS LAYOUT

✓	IC	+ METER LEAD ON PIN NO.	– METER LEAD ON PIN NO.	VOLTAGE READING
	U1	6	1	+5VDC*
	U1	7	1	+5VDC
	U1	8	1	+5VDC*
	U1	21	1	+5VDC
	U2	20	10	+5VDC
	U3	20	10	+5VDC
	U4	20	10	+5VDC
	U5	20	10	+5VDC
	U6	14	7	+5VDC
	U7	14	7	+5VDC
	U8	16	8	+5VDC
	U9	16	8	+5VDC
	U10	16	8	+5VDC
	U11	14	7	+5VDC
	U12	14	7	+12VDC
	U12	7	1	+12VDC

No application uses all ICs.
You should read indicated voltage on all that are installed.

*Reading may be slightly less than other +5VDC readings

Fig. 5 USIC POWER TESTS

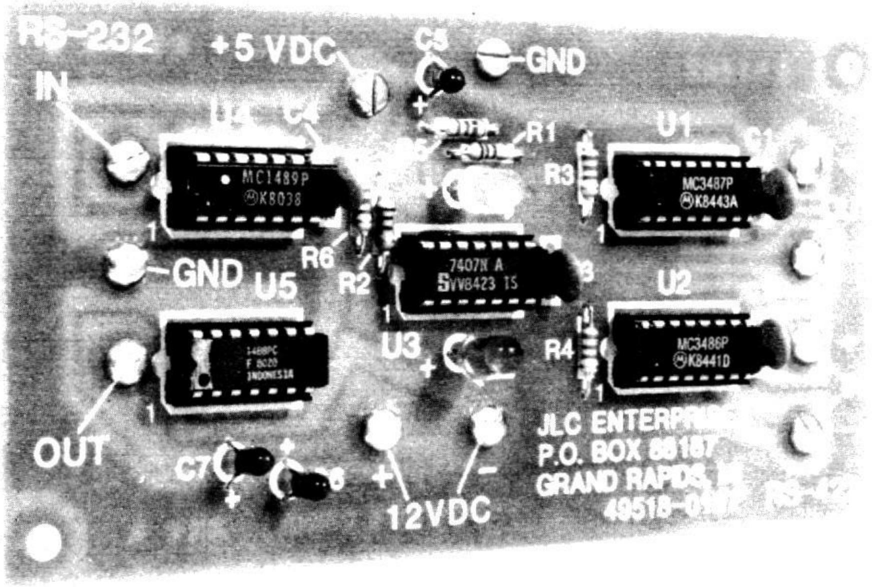


Fig. 6 RS-232/422 CONVERSION CARD

can set 16 unique addresses, allowing up to 16 USICs in an RS-422A system.

An eight-segment DIP switch sets the BAUD rate generator to the desired serial transmission frequency, which in turn is fed into the MC68701. The MC68701 provides all the parallel-to-serial and serial-to-parallel conversions, and determines 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 bring the R* and W* lines low for communication with the railroad. I'll explain the software for all this next month in Part 15.

To save space we are not including the USIC schematic drawing or my explanation of it here. You may obtain copies of these, along with full-size circuit-card art for the USIC and the RS232/422 Conversion Card, by sending a business-size, stamped, self-addressed envelope to MODEL RAILROADER, USIC, 1027 N. Seventh St., Milwaukee, WI 53233.

USIC ASSEMBLY

If this will be your first circuit card assembly, go back and study C/MRI Part 2, especially the sidebar, "PC Card Soldering." If you have trouble identifying any parts, referring to previous C/MRI installments should help.

Figure 4 includes the parts layout and parts list. You may make your own PC cards using the art available from MR, but ready-to-use USIC PC cards are available from JLC Enterprises, P. O. Box 88187, Grand Rapids, MI 49518-0187. Card USIC, with parts locations printed on its component side, sells for $32 plus a $2.50 shipping and handling charge per order. (Michigan residents must add 4 percent sales tax.)

The USIC is a double-sided PC card with circuit traces on each side interlaced *via* plated-through holes. Insert all components from the component, or "A," side, and do *all* soldering on the back, or "B," side. The order of assembly is not critical, but for the sake of having a plan, follow the steps in order and check off the boxes as you complete each one. Parts that must be inserted with a specific orientation or polarity are identified by a "[+]" in the instructions. If in doubt as to the correct orientation, see Part 2, fig. 4.

I'll list the common steps first, then the unique steps to configure your USIC for one of the three interface standards.

☐ **Card inspection.**
☐ **R1-R9.**
☐ **S1-S8[+].**
☐ **S13-S15.**
☐ **SW1, SW2[+].** Use your VOM to make sure these DIP switches are oriented so that the contacts are closed when set toward the "ON" label on the USIC.
☐ **C1-C8.**
☐ **C12, C13.**
☐ **C14, C15[+].**
☐ **C18, C19[+].**
☐ **XL1.** Hold case firmly against card while soldering leads.
☐ **L1-L3[+].**
☐ **U2-U8[+].** See Part 5, fig. 6 for IC insertion and extraction procedures.

That completes the general USIC assembly. Next install *only* the parts for the one interface standard you've chosen — 20mA current loop, RS-232C, or RS-422A.

20mA current loop:
☐ **P1.** Program jumper. Make from no. 24 uninsulated bus wire (Belden no. 8022).
☐ **R11, R12.**
☐ **U13, U14[+].** Insert these six-pin ICs directly into the PC board. Double check for correct pin-1 orientation, then hold each one firmly against the card while soldering each lead.

Skip down to "U1."

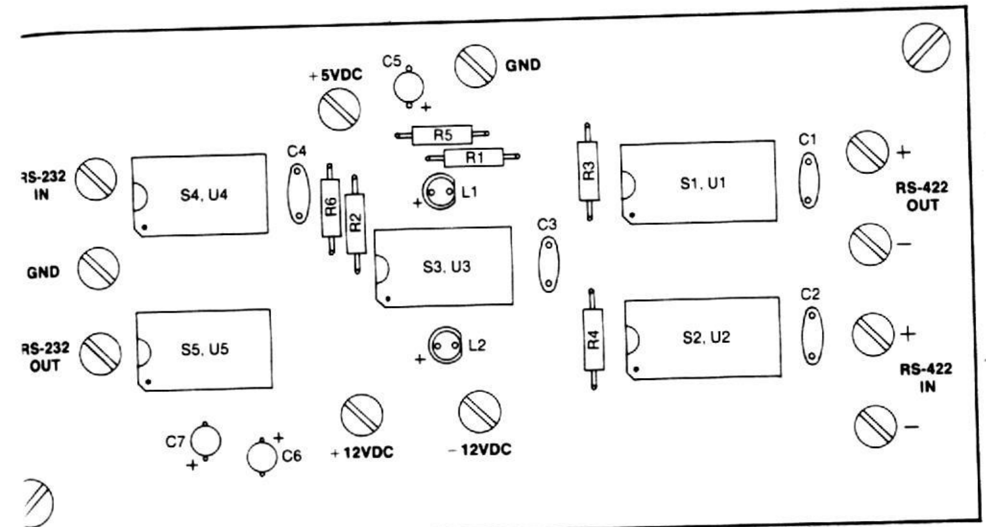
RS-232C:
☐ **S11, S12[+].**
☐ **C11.**
☐ **C16, C17[+].**
☐ **U11[+].**
☐ **U12[+].**
Skip down to "U1."

RS-422A:
☐ **R10.**
☐ **S9, S10[+].**
☐ **C9, C10.**
☐ **U9, U10[+].**

☐ **U1[+].** The MC68701 is a static-sensitive chip which 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 which could damage the chip.
☐ **Cleanup and inspection.**

USIC TESTING

The following steps check that the correct power is reaching each IC, and that U1 is correctly initialized when powered up and its internal software is working.
☐ **Set SW1.** All four segments OFF.
☐ **Set SW2.** Any one segment ON and all others OFF. (Which BAUD setting you use doesn't matter when the USIC isn't connected to the computer.)
☐ **Power-up test.** Turn off the IOMB's



Card shown actual size

RS-232 TO RS-422 CONVERSION CARD PARTS LAYOUT

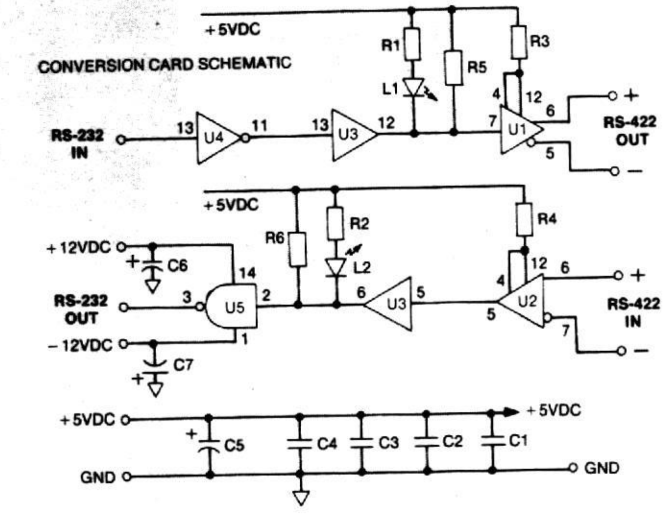
✓	IC	+ METER LEAD ON PIN NO.	– METER LEAD ON PIN NO.	VOLTAGE READING
	U1	16	8	+5VDC
	U2	16	8	+5VDC
	U3	14	7	+5VDC
	U4	14	7	+5VDC
	U5	14	7	+12VDC
	U5	7	1	+12VDC

RS-232/422 CONVERSION CARD PARTS LIST (In order of assembly)

Qnty.	Symbol	Description
11	—	4-40 pan-head machine screws with brass nuts
2	R1, R2	160Ω resistors [brown-blue-brown]
4	R3-R6	2200Ω resistors [red-red-red]
2	S1, S2	16-pin DIP sockets (Digi-Key C8916)
3	S3-S5	14-pin DIP sockets (Digi-Key C8914)
4	C1-C4	.1µF, 50V ceramic disk capacitors (Jameco DC.1/50)
3	C5-C7	2.2µF, 35V tantalum capacitors (Jameco TM2.2/35)
1	L1	Diffused amber LED (Digi-Key P306)
1	L2	Diffused red LED (Digi-Key P300)
1	U1	MC3487 quad differential line driver, RS-422 (Jameco) May substitute LM3847, 26LS31 or SN75174
1	U2	MC3486 quad differential line receiver, RS-422 (Jameco) May substitute LM3486, 26LS32 or SN75175
1	U3	7407 hex buffer/driver (Jameco SN7407N)
1	U4	LM1489N quad line receiver, RS-232 (Jameco)
1	U5	LM1488N quad line driver, RS-232 (Jameco)

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

CONVERSION CARD SCHEMATIC



+5VDC supply and remove any I/O Cards; then plug the USIC into the IOMB's first slot. Now turn on the 5V power; the green LED should start blinking about once every two seconds. It's controlled by a real-time-loop counter in U1's internal software and shows that the chip is working properly. The red and amber LEDs should be off unless you're using 20mA current loop I/O, in which case red should be off and amber on. Even if the LED indications aren't correct, go on to the next test.

☐ **IC power tests.** Set your VOM to measure DC voltage, and make the tests listed in fig. 5 to see that each IC installed is receiving the proper voltage. If you find one not getting power, work back along the circuit path to locate and correct the open circuit. A low voltage, less than 4.6V, most likely means a short circuit on your card caused by a solder bridge or a reversed part. Turn off the power and reexamine the card to find and correct the short. If necessary, remove ICs one at a time to isolate the problem.

For RS-232C you'll need a ±12VDC supply connected to S15 to check the U12 voltage. Follow Part 9, fig. 5, but substitute 7812 positive and 7912 negative 12V regulators for V1 and V2.

☐ **Recheck operation.** If IC power is correct in all tests, the green LED should blink continuously when +5VDC is applied to the USIC. If the green LED still doesn't blink, look for a bad solder joint, a solder bridge, or an incorrectly inserted or faulty part. Be persistent in searching for such problems. If necessary to isolate the fault you can remove U2-U6 and U9-U13, since they should have no effect on the green LED's indication. The cost of U1 includes program verification and full testing in a functional USIC, so it should be the very last thing to suspect in debugging your card.

That completes the USIC testing for this month. If you're using the 20mA Current Loop or RS-232C I/O you can skip the rest of this installment.

RS232/422 CONVERSION CARD

If you want to use RS-422A I/O with a computer not equipped for it, assemble an RS-232/422 Conversion Card as shown in fig. 6. You may make your own card using the art available from MR, but ready-to-use RS232/422 cards are available from JLC Enterprises, P. O. Box 88187, Grand Rapids, MI 49518-0187, for $18 each plus a $2.50 shipping and handling charge per order. (Michigan residents must add 4 percent sales tax.)

The assembly steps are as follows:

☐ **Card inspection.**

☐ **Terminals.** Solder wires directly to terminal pads, or enlarge holes with no. 33 drill for 4-40 terminal screws with nuts.

☐ **R1-R6.**

☐ **S1-S5[+].**

☐ **C1-C4.**

☐ **C5-C7[+].**

☐ **L1,L2[+].**

☐ **U1-U5[+].**

☐ **Cleanup and inspection.**

☐ **IC power test.** Connect +5VDC and ±12VDC power, using the same ground screw for both. Use your VOM to make the power tests listed in fig. 6; if you don't find correct voltage follow the circuit traces back to find the fault.

☐ **LED test.** Attach one end of a clip lead to the ground screw, and clip a bit of bare, solid wire in the other end as a test probe. Carefully touch the probe to pin 10 of U3: L1 should light. Touch pin 6 of U3: L2 should light. If not, check for poor soldering, or reversed or faulty LEDs.

That completes all the hardware for a serial interface. Next month we'll cover the system operation and software. ♦

C/MRI-15

The C/MRI: A computer/model railroad interface

Part 15: Serial interface software

BY BRUCE CHUBB

LAST month we built the hardware for a serial interface — the Universal Serial Interface Card or USIC. This time I'll show you the special software for the USIC and how to use it.

The USIC's main software requirement is a protocol to allow sending card addresses and data in serial over the same wire. The protocol software will automatically form packets of information, messages, to be transmitted. It will also insert special control characters to mark the beginning and end of transmissions, and define which bytes are addresses and which are pure data. This may sound complicated, but software engineer Bruce Wahl and I have done the hard work for you by preparing standardized subroutines for the USIC protocol.

The protocol uses four types of message: initialization, transmit data, poll request, and receive data. I'll describe the general message format first; then we'll look at the details of each type.

GENERAL MESSAGE FORMAT

Each message consists of framing and data. In serial I/O we use "data" as a general term for both the I/O Card addresses and the information being communicated to or from their ports. In serial I/O, addresses and information are all just signals sent over the wires. Figure 1 depicts a typical serial I/O message, bit by binary bit: a Mark in a given bit period represents a logic 1, while a Space represents a logic 0; in each byte the LSB (*L*east *S*ignificant *B*it) is sent first.

Framing separates one message from another and defines the type of data in each byte. Three standard control characters are used in framing the packets: STX for *S*tart-of-*T*e*X*t, ETX for *E*nd-of-*T*e*X*t, and DLE for *D*ata-*L*ink-*E*scape.

To understand the differences in how the serial channel handles control characters vs. data, it's important to know how alpha characters (letters) are stored in a computer. In computer hardware, the memory for example, numbers, letters, and special characters are all stored as binary codes made up of ones and zeroes.

Figure 2 is the American Standard Code for Information Interchange, ASCII. It defines standard relationships between bytes of eight-bit binary code and their multiple representations in decimal, hexadecimal, and "symbolic character" notation. Symbolic character notation includes function codes or control characters, and also other character types, including upper and lower case alpha characters, numeric characters (numerals) 0 through 9, and special characters such as punctuation.

ASCII defines function and character codes only for decimal 0 through 127, leaving 128 through 255 undefined except for the mathematical relationship between binary, decimal, and hexadecimal codes. Figure 2 is an important reference to keep handy as you do more and more C/MRI application programming. You may not have realized it, but we've already been using ASCII codes.

For example, in the Part 5 fig. 11 Output Card Test Program, when the computer's BASIC interpreter read the instruction PN$(0) = "A," it accepted the letter A as an ASCII symbol and stored the binary code 01000001 in the memory location for variable PN$(0). With an arithmetic statement such as G = 65, the interpreter would place the same 01000001 code in the memory location for variable G.

Note that the same binary code can represent decimal integers (numbers) and either alpha or control characters. It's up to the program to keep track of the correct interpretation for any given application. You'll see how this works as you learn more about message formats and protocol subroutines.

Returning to fig. 1, each message format begins with two bytes of all ones to synchronize the hardware. Next comes an STX to show that the text of the message begins with the next byte. The text is a series of bytes with each bit of each byte sent in serial, one bit after another, and each new byte following right behind its predecessor.

The first byte after the STX is the "offset" *U*SIC *A*ddress UA of the USIC being commanded. It is offset by adding decimal 65 to the USIC's DIP switch setting, written as UA + 65, to keep UA from conflicting with any control characters. That makes the legal values of UA + 65 for the 16 possible USICs correspond to alpha characters A through P.

Following the address byte is an alpha character defining the message type as either an "I" for *I*nitialize, "T" for *T*ransmit

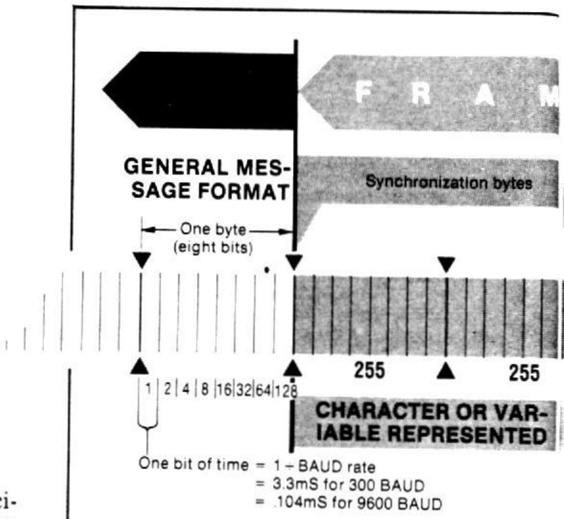
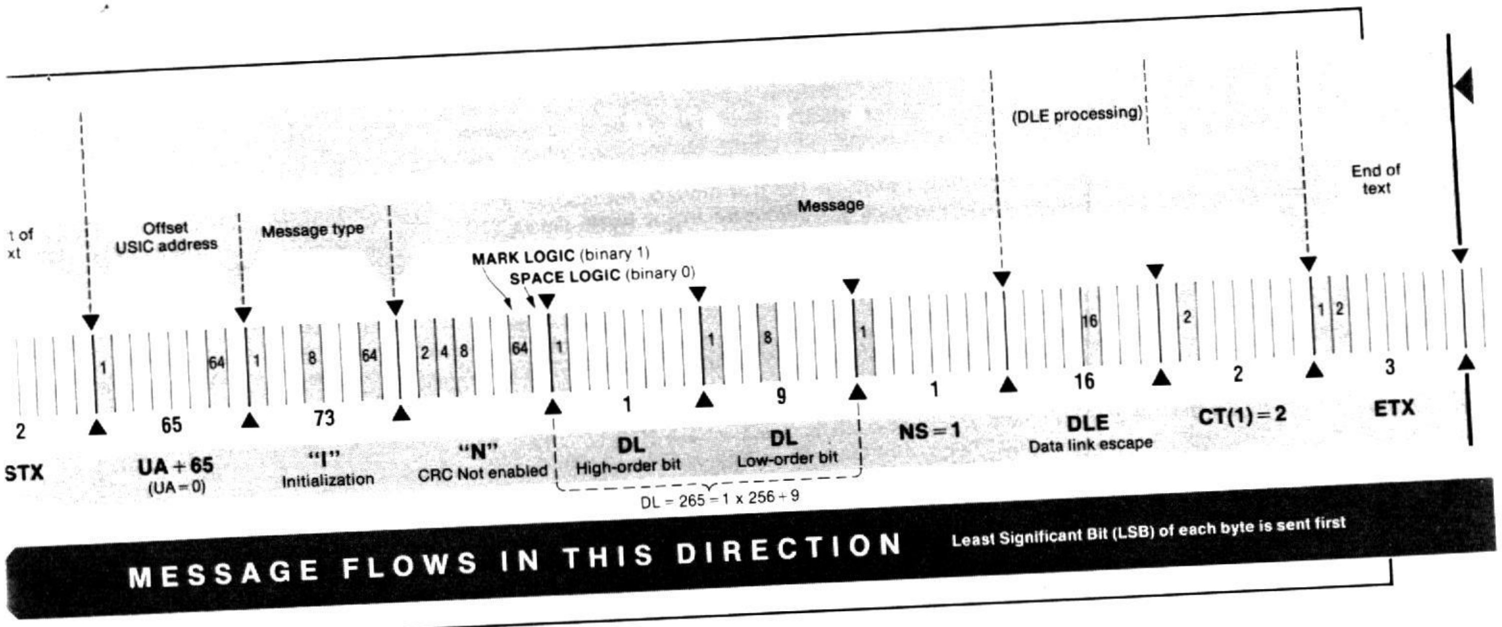


Fig. 1 SERIAL I/O MESSAGE PACKET

BINARY CODE	DECIMAL CODE	HEXADECIMAL CODE	FUNCTION CODE
00000000	0	00	NUL
00000001	1	01	SOH
00000010	2	02	STX
00000011	3	03	ETX
00000100	4	04	EOT
00000101	5	05	ENQ
00000110	6	06	ACK
00000111	7	07	BEL
00001000	8	08	BS
00001001	9	09	HT
00001010	10	0A	LF
00001011	11	0B	VT
00001100	12	0C	FF
00001101	13	0D	CR
00001110	14	0E	SO
00001111	15	0F	SI
00010000	16	10	DLE
00010001	17	11	DC1
00010010	18	12	DC2
00010011	19	13	DC3
00010100	20	14	DC4
00010101	21	15	NAK
00010110	22	16	SYN
00010111	23	17	ETB
00011000	24	18	CAN
00011001	25	19	EM
00011010	26	1A	SUB
00011011	27	1B	ESC
00011100	28	1C	FS
00011101	29	1D	GS
00011110	30	1E	RS
00011111	31	1F	US
00100000	32	20	SP



MEANING	BINARY CODE	DECIMAL CODE	HEXADECIMAL CODE	CHARACTER	BINARY CODE	DECIMAL CODE	HEXADECIMAL CODE	CHARACTER	BINARY CODE	DECIMAL CODE	HEXADECIMAL CODE	CHARACTER
All zero character	00100001	33	21	!	01000010	66	42	B	01100011	99	63	c
Start of heading	00100010	34	22	"	01000011	67	43	C	01100100	100	64	d
Start of text	00100011	35	23	#	01000100	68	44	D	01100101	101	65	e
End of text	00100100	36	24	$	01000101	69	45	E	01100110	102	66	f
End of transmission	00100101	37	25	%	01000110	70	46	F	01100111	103	67	g
Enquiry	00100110	38	26	&	01000111	71	47	G	01101000	104	68	h
Acknowledge	00100111	39	27	'	01001000	72	48	H	01101001	105	69	i
Bell	00101000	40	28	(	01001001	73	49	I	01101010	106	6A	j
Backspace	00101001	41	29	)	01001010	74	4A	J	01101011	107	6B	k
Horizontal tabulation	00101010	42	2A	*	01001011	75	4B	K	01101100	108	6C	l
Linefeed	00101011	43	2B	+	01001100	76	4C	L	01101101	109	6D	m
Vertical tabulation	00101100	44	2C	,	01001101	77	4D	M	01101110	110	6E	n
Form feed	00101101	45	2D	-	01001110	78	4E	N	01101111	111	6F	o
Carriage return	00101110	46	2E	.	01001111	79	4F	O	01110000	112	70	p
Shift out	00101111	47	2F	/	01010000	80	50	P	01110001	113	71	q
Shift in	00110000	48	30	0	01010001	81	51	Q	01110010	114	72	r
Data link escape	00110001	49	31	1	01010010	82	52	R	01110011	115	73	s
Device control 1	00110010	50	32	2	01010011	83	53	S	01110100	116	74	t
Device control 2	00110011	51	33	3	01010100	84	54	T	01110101	117	75	u
Device control 3	00110100	52	34	4	01010101	85	55	U	01110110	118	76	v
Device control 4	00110101	53	35	5	01010110	86	56	V	01110111	119	77	w
Negative acknowledge	00110110	54	36	6	01010111	87	57	W	01111000	120	78	x
Synchronous idle	00110111	55	37	7	01011000	88	58	X	01111001	121	79	y
End of transmission block	00111000	56	38	8	01011001	89	59	Y	01111010	122	7A	z
Cancel	00111001	57	39	9	01011010	90	5A	Z	01111011	123	7B	{
End of medium	00111010	58	3A	:	01011011	91	5B	[	01111100	124	7C	|
Substitute	00111011	59	3B	;	01011100	92	5C	\	01111101	125	7D	}
Escape	00111100	60	3C	<	01011101	93	5D	]	01111110	126	7E	~
File separator	00111101	61	3D	=	01011110	94	5E	^	01111111	127	7F	DEL
Group separator	00111110	62	3E	>	01011111	95	5F	_	.	.	.	.
Record separator	00111111	63	3F	?	01100000	96	60	`	.	.	.	.
Unit separator	01000000	64	40	@	01100001	97	61	a	11111110	254	FE	
Space	01000001	65	41	A	01000010	98	62	b	11111111	255	FF	

Fig. 2 AMERICAN STANDARD CODE FOR INFORMATION INTERCHANGE (ASCII)

data, "P" for *P*oll-request, or "R" for *R*eceive data. The details of the remaining message bytes depend on the message type, and I'll describe them separately.

The DLE or data-link-escape character is used when it's necessary to include binary codes in the data for decimal numbers 2, 3, and 16, which are the same codes as for control characters STX, ETX, and DLE. In the fig. 1 example, a decimal 2 is sent by first sending a DLE byte, decimal 16, and following that with a byte of 2. An ETX character marks the end of each message.

Now refer to fig. 3 as we take a closer look at each message format.

Initialization message. This message tells the USIC how many Input and Output Cards are connected to its IOMB, and which types are at which address. Once the USIC's U1, the MC68701, receives this information, it automatically initializes the 8255s on the I/O cards.

The format of the initialization message is given in fig. 3a. "N" signifying CRC *N*ot Enabled, USIC *D*e*L*ay DL, and variable NS giving the *N*umber of Card *S*ets of four will be explained with the initialization protocol subroutine on pages 89 and 90.

Card types are defined by a CT( ) byte for each group of four cards. A byte allows two bits to define each card, and we'll use 0 = no card, 1 = Input Card, 2 = Output Card, and 3 = reserved for expansion. Figure 4 shows how to calculate CT( ) bytes.

Transmit-data message. For this message, fig. 3b, the byte immediately following the address is a "T." The data section includes three binary bytes per Output Card on the IOMB. The first byte corresponds to port A of the first card, followed by ports B and C, followed by the same data format for the second Output Card, the third, and so forth. The DLE processing described above is used in the data transmission to allow transmission of binary code equivalents of the control characters.

Poll-request and receive-data messages. Two messages are needed for the computer to receive data. First the computer sends a *P*oll-request or "P" message, fig. 3c, to tell the USIC to send back the state of the Input Card input lines connected to its IOMB. When the USIC receives the "P" message, it reads its Input Cards and forms the *R*eceive-data or "R" message, fig. 3d, which carries the information back to the computer. The data section of the "R" message contains three bytes per input card on the IOMB, and again it uses DLE processing.

PROTOCOL SUBROUTINES

Figure 5 is a listing of the protocol subroutines used for USIC communications. Lines 200 through 1970 contain the source code and are a standard block you put at the top of every BASIC application program for the USIC serial interface.

The protocol code is ready-to-use and essentially application-independent, so I'll just cover the high points to give you a general understanding of how it works. As before, I've used Heath/Zenith Micro-Soft BASIC and avoided exotic codes, so most of it should work on any computer with little change. I'll point out the few lines which are computer-dependent and need to be tailored to your own machine.

There are three user-invoked subroutines for *T*ransmit data, T, *R*eceive data, R, and *I*nitialize, I, but first we need to cover a couple of utility subroutines which are called by T, R, and I. Including the two utilities, the five protocol subroutines are:

- **Receive one input byte.** Lines 310 through 350 are a utility subroutine that receives a single *I*nput *B*yte IB over the C/MRI *via* Port Address PA. I've made it a separate subroutine because this function is used over and over again. However, once you have your protocol package fully debugged you may want to embed duplicates of these lines at each location where they're called, to increase operating speed.

Lines 320 and 330 are user-dependent and check that the computer I/O port connected to the C/MRI is operating correctly to receive input, and that a byte has been properly accumulated and is ready to be placed on the computer's data bus.

My Heathkit serial card uses an 8250 Asynchronous Communication Element (ACE) chip which is quite popular in other makes of computer. If your computer uses the same device to control its RS-232C port, the two statements will be similar if not identical. For other machines you will have to study your reference/operations manuals to derive similar tests and alter the code accordingly to provide the same functions.

With the 8250, program line 320 examines port PA's "Line Status Register," equivalent to a special memory byte at Port Address PA + 5. This register lets the program determine the status of the port's I/O lines at a given moment.

For example bit 1, "Overrun Error," is set to one to signify that data is arriving faster than it can be processed. To check for this condition, 320 ANDs the contents of PA + 5 with decimal 2, a binary 00000010 which strips out all bits except bit 1, and checks for it being a one. (For Apple BASICs replace ANDs with equivalent arithmetic operations to strip out and test appropriate bits.)

If the result is a one, line 320 prints "OVERRUN" on the CRT as an alert that this is happening. If you get such an error message, you can increase USIC DeLay variable DL to purposely slow down the USIC and give your computer more time to process input.

Line 330 performs a similar test of the line status register to check when port PA has received a complete byte of input data and has converted it to parallel format so it can be placed on the computer's data bus. This is accomplished by examining bit 0, the "Data Ready Bit," of port PA's Line Status Register.

If bit 0 is a zero, data isn't available, so the program loops back through line 320 until bit 0 becomes a one, showing that data is available. Once data is available, the program drops through the IF statement to execute line 340 to input the byte and return control to the calling program.

- **Form packet and send.** Lines 400-680 are a utility subroutine to form a message packet and send it over the C/MRI. It's a separate subroutine since it is used by the Transmit-data, Receive-data, and Initialization subroutines. Once more, for increased speed you can embed copies of this code "in line" once your program is operational.

a. INITIALIZATION, "I"
255, 255, STX, UA + 65, "I", "N", DL high-order bit, DL low-order bit, NS, CT(1),... CT(NS), ETX

b. TRANSMIT DATA, "T"
255, 255, STX, UA + 65, "T", OB(1),.... OB(NO), ETX. (OB is output data array.)

c. POLL REQUEST, "P"
255, 255, STX, UA + 65, "P", ETX

d. RECEIVE DATA, "R"
255, 255, STX, UA + 65, "R", IB(1),... IB(NI), ETX. (IB is input data array.)

Fig. 3 MESSAGE FORMATS

The array variable TB( ), the *T*ransmit *B*uffer, contains the data bytes in sequence for the ports of each Output Card, to be transmitted to the USIC for the transmit-data message. TB( ) also holds data for the initialization message.

As we've seen, the first and second bytes are set to all 1s, decimal 255, by lines 410 and 420. Line 430 sets the third byte to STX (decimal 2), 440 sets the fourth byte to UA + 65, and 450 sets the fifth byte to the message type. These five bytes use five positions in the TB array, so line 460 sets *T*ransmit *P*ointer TP to 6, the next available buffer position.

Line 470 then checks to see if the message type is a "P" or poll request. If it is, then all necessary data is in the buffer except for the ETX (decimal 3), which is then added by the branch to 550. If the message is not a poll request, if it's either a "T" or an "I," the program falls through the test at 470 to execute line 480.

The loop in lines 480-540 sets up the data part of the message packet by transferring the desired Output Byte data array, OB( ), to the Transmit Buffer. The next available TB variable, beginning with TB(6), is set equal to the next OB variable beginning with OB(1), except for the cases when a particular OB( ) equals a 2, 3, or 16.

In these cases the data to be sent is in the same code as a control character, and per our protocol an extra TB( ) value of 16, a DLE character, is inserted ahead of the 2, 3, or 16 byte. TP( ) is incremented by one and the 2, 3, or 16 data byte is placed after the DLE. Once all OB( ) values are in TB( ), line 550 defines the end of the data by setting TB(TP) equal to decimal 3 for ETX.

With the formulated message packet in the Transmit Buffer, the loop in lines 640-670 transmits the message over the serial channel a byte at a time. Line 650 is user-dependent and checks that the computer I/O port connected to the C/MRI is free to transmit a byte. In my H8's serial card with the 8250 ACE, this is done by looking at port PA's line status register bit 5, the "Transmitter Holding Register Empty" bit.

If this bit is a one, the port is ready to accept a parallel data byte for transmission over the serial port. If the bit is a

SLOT LOCATION IN SET OF 4	CARD		MULTIPLICATION CONSTANT	PRODUCT OF CARD TYPE TIMES CONSTANT
	TYPE	TYPE NO.		
1			1	
2			4	
3			16	
4			64	
			CARD SET CODE	

No card = 0
Input card = 1
Output card = 2

Sum of above numbers goes here

Steps to calculate each CT( ) card set code

1. Enter card type and type numbers for each set of 4 cards (slot 1 location always closest to USIC)
2. Multiply each card type number by the card position constant 1, 4, 16, or 64
3. Sum the products to get the card set code
4. Repeat process for each card set of 4

Example:

1st card set of 4	I	1×1=1
	O	2×4=8
	O	2×16=32
	I	1×64=64
		CT(1)=105
2nd card set of 4	I	1×1=1
	I	1×4=4
	O	2×16=32
	none	0×64=0
		CT(2)=37

Fig. 4 CALCULATING CARD TYPE DEFINITION ARRAY VARIABLES

zero the port isn't ready, and the program continues to branch back on line 650 until the bit becomes a one, showing that the port has become available. Then line 660 is executed to output the Ith byte TB(I) to the C/MRI. Once all bytes are transmitted, line 680 returns control to the calling program.

● **Transmit data.** Lines 800-830 are the subroutine used to transmit data to a set of Output Cards connected to a USIC. Line 810 sets *M*essage *T*ype variable MT = decimal 84, ASCII "T." With this definition complete and the Output Byte array OB( ) established by the application program, all that's needed is to invoke the Form-Packet-and-Send subroutine with the GOSUB 410 in line 820.

● **Receive data.** Lines 1000-1230 are the subroutine used to receive data from a set of Input Cards connected to a USIC. As I've explained, the program first sends a "P" message to the USIC with lines 1010 and 1020. It then loops on reading input bytes, *via* the GOSUB 320 in line 1070 and the test in line 1080, until it receives a byte of decimal 2, STX, from the USIC.

Once an STX is received, line 1090's GOSUB 320 pulls in the next input byte, the offset USIC address UA + 65. Line 1100 removes the offset and line 1110 checks to make sure the address received is correct. If not, the transmission is assumed to be garbled. An error message is printed on the CRT, and the program branches to line 1010 and tries again.

If the addresses do match, line 1120 is executed to bring in the next input byte. This should be the message type sent by the MC68701, a decimal 82 for "R." If the byte is not an 82 the transmission is again assumed garbled. Another error message is printed, and the program branches back to 1010 and tries again.

If both these inputs are received correctly, the program proceeds to pull in the data from the Input Cards. Lines 1140 through 1200 loop through each input port 1 to NI. In line 1150 GOSUB 320 is invoked to read in the Input Byte IB. If line 1160 finds that it's a 2, STX, the transmission must be garbled or out of synchronization, and the program prints another error message and branches back to 1010 to try once more. If line 1170 finds that IB is a 3, ETX, it again must be garbled, resulting in another error message and another retry.

If IB is a 16, DLE, the program skips that byte and invokes GOSUB 320 to read the next IB, the actual data byte following the data-link-escape character. The value read in as IB is then placed in its proper array position IB(I), and the program increments loop variable I to process the next input port.

Once the program has received the input bytes for all the NI input ports, it reads the next IB with line 1210 and checks in 1220 to make sure it is ETX, decimal 3. If not a 3, the program prints an error message on the CRT and RETURNS control back to the user program in line 1230.

● **Initialization.** Lines 1800-1970 are the subroutine that initializes the USIC and its I/O Cards. It starts by setting the BAUD rate for the host computer — in my H8 serial interface the 8250 ACE puts the BAUD rate and basic serial format under software control. Many other computers also use the 8250, making my code applicable to them, while for others you may need to add a jumper wire or set a DIP switch in your computer's hardware to set the BAUD rate.

In the case of the 8250, line 1810 sets a "software-controlled switch" so that when the computer writes out to the port it actually writes to the 8250's internal BAUD rate generator to "latch in" a divisor to be used for generating the BAUD rate from the computer's basic clock frequency — very much as the CD4040 counter chip and DIP switch do on the USIC.

Once the latch is made accessible by line 1810, lines 1820 and 1830 set the BAUD rate. The constants I've shown set it for 9600; changing to OUT PA,128 and OUT PA + 1,1 would set the BAUD rate at 300.

Different constants, as defined in the H8 serial operations manual for the H8 serial card, can be used to set any other standard BAUD rate. Study your computer's reference/operations manuals to find the procedures for setting the BAUD rate with your machine. Similarly, line 1840 formats the 8250 to handle eight data bits, no parity, and one stop bit, which are also requirements for proper USIC operation. Line 1850 defines the MT variable as "I," decimal 73.

I've chosen not to use the Cyclic Redundancy Check or CRC feature provided by the Chesapeake Group Inc's. MC68701 program. CRC helps ensure to a high probability that data bits aren't being lost or garbled, but it takes processing time. It also requires additional lines in the protocol subroutine to handle CRC codes and limits the MC68701 to processing no more than 16 Output Cards at a time unless you add other additional protocol lines. All that takes still more processing time, but in BASIC real-time application programs for serial I/O there is *no* spare time for CRC error checking.

Besides, I've operated the USIC in RS-232C, RS-422A, and 20mA current loop modes for hours on end, with a long transmission cable interwoven with other railroad wiring, all without a noted error. Also, as you've seen, I've included enough error messages in the prototcol subroutines so that if you have problems with electrical noise, or with something like a BAUD-rate generator set incorrectly, you will soon see error messages on your CRT.

The MC68701's program expects to receive CRC codes, and will generate and transmit them if the CRC function is enabled. Line 1860 disables it by setting OB(1) = 78, an ASCII "N" for No CRC checking.

Lines 1870 and 1880 set the USIC DeLay variable DL to be sent to the USIC. The value of this variable causes the USIC software to build in a delay of about 10 microseconds per unit of DL. A value of DL = 0 yields no added delay, and DL = 6000 adds a delay of 60mS per transmitted byte.

This delay keeps the USIC from sending data faster than your computer can receive and process it. Since DL values can easily go over 255, the maximum decimal value we can store in one byte, it's necessary to use two bytes, high- and low-order, set by lines 1870 and 1880.

In line 1890 variable OB(4) is equated to the Number-of-card-Sets-of-four variable NS. A loop then fills following OB array positions with the Card Type or CT variables for each set of four cards. With all card types defined, the line 1950 GOSUB 410 is invoked to form the complete initialization packet and transmit it to the USIC. The resulting transmission initializes the USIC's MC68701, and it in turn automatically sets the 8255s on each of the attached I/O Cards.

With its initialization work complete,

```
200   REM***STANDARD USIC PROTOCOL SUBROUTINES***
210   REM**REVISION 16-MAR-86
310   REM**RECEIVE ONE INPUT BYTE**
320*  IF (INP(PA+5) AND 2) <> Ø THEN PRINT "OVERRUN"
330*  IF (INP(PA+5) AND 1) = Ø THEN GOTO 32Ø
340   IB=INP(PA) 'received byte from port PA
350   RETURN
400   REM**FORM PACKET AND SEND
410   TB(1)=255 'set 1st start byte to all 1's
420   TB(2)=255 'set 2nd start byte to all 1's
430   TB(3)=2 'define start-of-text
440   TB(4)=UA+65 'add 65 offset to USIC Address
450   TB(5)=MT 'define message type
460   TP=6 'next position for transmit pointer
470   IF MT=80 THEN GOTO 55Ø 'poll request;end message
480   FOR I=1 TO LM 'loop to setup output data
490   IF OB(I)=2  THEN TB(TP)=16: TP=TP+1: GOTO 52Ø
500   IF OB(I)=3  THEN TB(TP)=16: TP=TP+1: GOTO 52Ø
510   IF OB(I)=16 THEN TB(TP)=16: TP=TP+1
520   TB(TP)=OB(I) 'move byte to transmit buffer
530   TP=TP+1 'increment to next byte position
540   NEXT I
550   TB(TP)=3 'set end-of-text
630   REM**send packet
640   FOR I=1 TO TP 'loop through transmit buffer
650*  IF (INP(PA+5) AND 32)=Ø THEN GOTO 65Ø
660   OUT PA,TB(I) 'byte transmitted
670   NEXT I
680   RETURN
800   REM**TRANSMIT DATA**
810   MT=84 'message type ="T"
820   GOSUB 41Ø 'form packet and send
830   RETURN
1000  REM**RECEIVE DATA**
1010  MT=8Ø 'send out poll request "P" for input data
1020  GOSUB 41Ø 'form packet and send
1060  REM**receive data back from USIC
1070  GOSUB 32Ø 'receive input byte
1080  IF IB<>2 THEN GOTO 1Ø7Ø 'loop till get start-of-text
1090  GOSUB 32Ø 'receive input byte as USIC address
1100  IB=IB-65 'subtract offset for address check
1110  IF IB<>UA THEN PRINT"ERROR RECEIVED BAD UA":GOTO 1Ø1Ø
1120  GOSUB 32Ø 'receive input byte as "R"
1130  IF IB<>82 THEN PRINT"ERROR:RECEIVE NOT=R":GOTO 1Ø1Ø
1140  FOR I=1 TO NI 'loop through number of input ports
1150  GOSUB 32Ø 'receive input byte
1160  IF IB=2 THEN PRINT"ERROR:NO DLE AHEAD OF 2":GOTO 1Ø1Ø
1170  IF IB=3 THEN PRINT"ERROR:NO DLE AHEAD OF 3":GOTO 1Ø1Ø
1180  IF IB=16 THEN GOSUB 32Ø 'DLE so next byte read
1190  IB(I)=IB 'store as valid data byte
1200  NEXT I
1210  GOSUB 32Ø 'receive input byte as end-of-text
1220  IF IB<>3 THEN PRINT"ERROR ETX NOT PROPERLY RECEIVED"
1230  RETURN
1800  REM**INITIALIZATION**
1810* OUT PA+3,128 'turn CR bit 7 on to access BAUD rate
1820* OUT PA,12  'set LS latch for 9600 BAUD rate
1830* OUT PA+1,Ø 'set MS latch for 9600 BAUD rate
1840* OUT PA+3,3 'set up for 8 data bits and 1 stop bit
1850  MT=73 'message type="I"
1860  OB(1)=78 'CRC setup not used="N"
1870  OB(2)=INT(DL/256) 'set USIC delay high order byte
1880  OB(3)=DL-(OB(2)*256) 'set USIC delay low order byte
1890  OB(4)=NS 'number card sets of 4
1900  LM=4 'establish length of message
1910  FOR I=1 TO NS 'loop through number of card sets
1920  LM=LM+1 'accumulate message length
1930  OB(LM)=CT(I) 'define card type array
1940  NEXT I
1950  GOSUB 41Ø  'form packet and send
1960  LM=NO 'initialize length message for "T" type
1970  RETURN
```

Fig. 5 USIC PROTOCOL SUBROUTINES

the subroutine executes line 1960 to set *M*essage *L*ength variable LM equal to NO, the *N*umber of *O*utput Card ports. LM will be used by the Form-Packet-and-Send subroutine during repetitive calls from the Transmit-Data subroutine. With this initialization complete, control RETURNs back to the user program in line 1970. That's all of the protocol subroutines, so now let's see how to put them to use in application programs.

INITIALIZATION OF VARIABLES

The protocol subroutines do all the I/O work for us, but for them to work properly certain variables must be "handed-off" between your main program and the subroutines. For example, before invoking the Transmit-Data subroutine you must define the output byte data array, OB( ), to be transmitted to the Output Cards. Before invoking the Initialization subroutine you must define USIC Address UA, and also the CT( ) arrays to specify what I/O Cards are plugged into the USIC's IOMB.

In short, all variables except the IB( ) array must be defined before invoking the protocol subroutines, and each of the variable symbols CT( ), IB( ), NI, NO, NS, OB( ), PA, and UA must be written with exactly these alphabetical notations. Also, certain variables used internally in and/or between the protocol subroutines should be treated as "reserved variables" not to be used in an application program. These include ML, IB, MT, TB( ), and TP.

The loop index variable I is used internally in the protocol subroutines, but "I" can be used in your application programs. Just remember that its value will most likely be changed any time you invoke and return from a protocol subroutine.

The numeric values, of course, have to be adjusted to fit your own C/MRI system and whatever application programs you write. For example, you must allow one element in the CT( ) array for each group of four I/O Card address slots (and for any tail-end group of less than four) used in a given application, and set NS equal to the largest subscript needed to define your CT( ) array.

When you invoke the Receive-Data subroutine, it returns with the Input Byte data array IB( ) as read from the Input Cards. The main program must be ready to unpack data in this form.

APPLICATION PROGRAMS

To put everything in perspective I'll describe a hypothetical application program and see how it interfaces with the protocol subroutines. To make it more general I'll use an application with two USICs, each with a different assortment of I/O cards as shown in fig. 6.

The first step is to calculate the CT( ) array variables to let the USICs "know" what I/O Cards are in which address slots. The five CT( ) values shown in fig. 6 were calculated following the procedure in fig. 4. Try calculating them based on the card complements in this example to make sure you understand the method and that you get the same values.

The general outline for an application program using the hardware of fig. 6 is shown in fig. 7. It's basically straightforward, but note that in line 20 the GOTO 3000 branches around the block of protocol subroutines, lines 200-1970. Your application program, whether it's a test/diagnostic program, a signaling program, CCC, some combination of these, or something else altogether, always starts at line 3000.

The initialization section first defines the general protocol variables, those independent of the particular USIC. That's followed by the specific initialization for the first USIC, followed by GOSUB 1810 and a repeat of the initialization procedure for the second USIC.

The real-time loop is set up as our previous examples for CCC and signaling; namely, receive C/MRI inputs and unpack, calculate output variables from input variables, pack outputs, and transmit to C/MRI. The only difference with more than one USIC is that the receive and unpack operations are grouped together

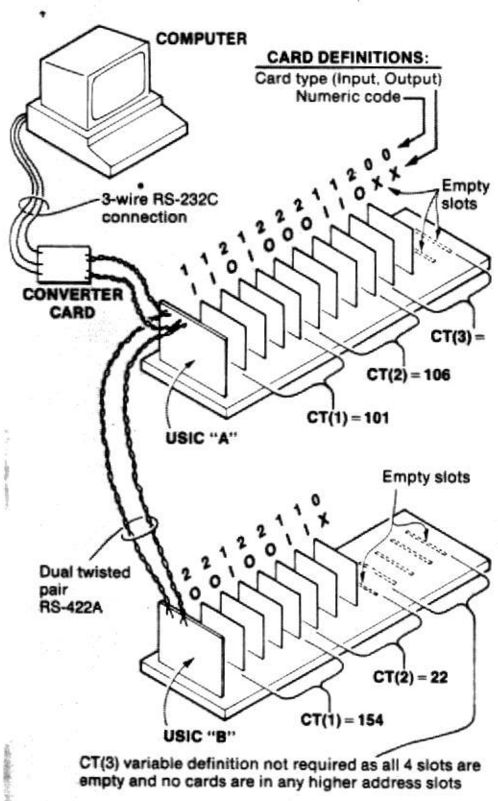
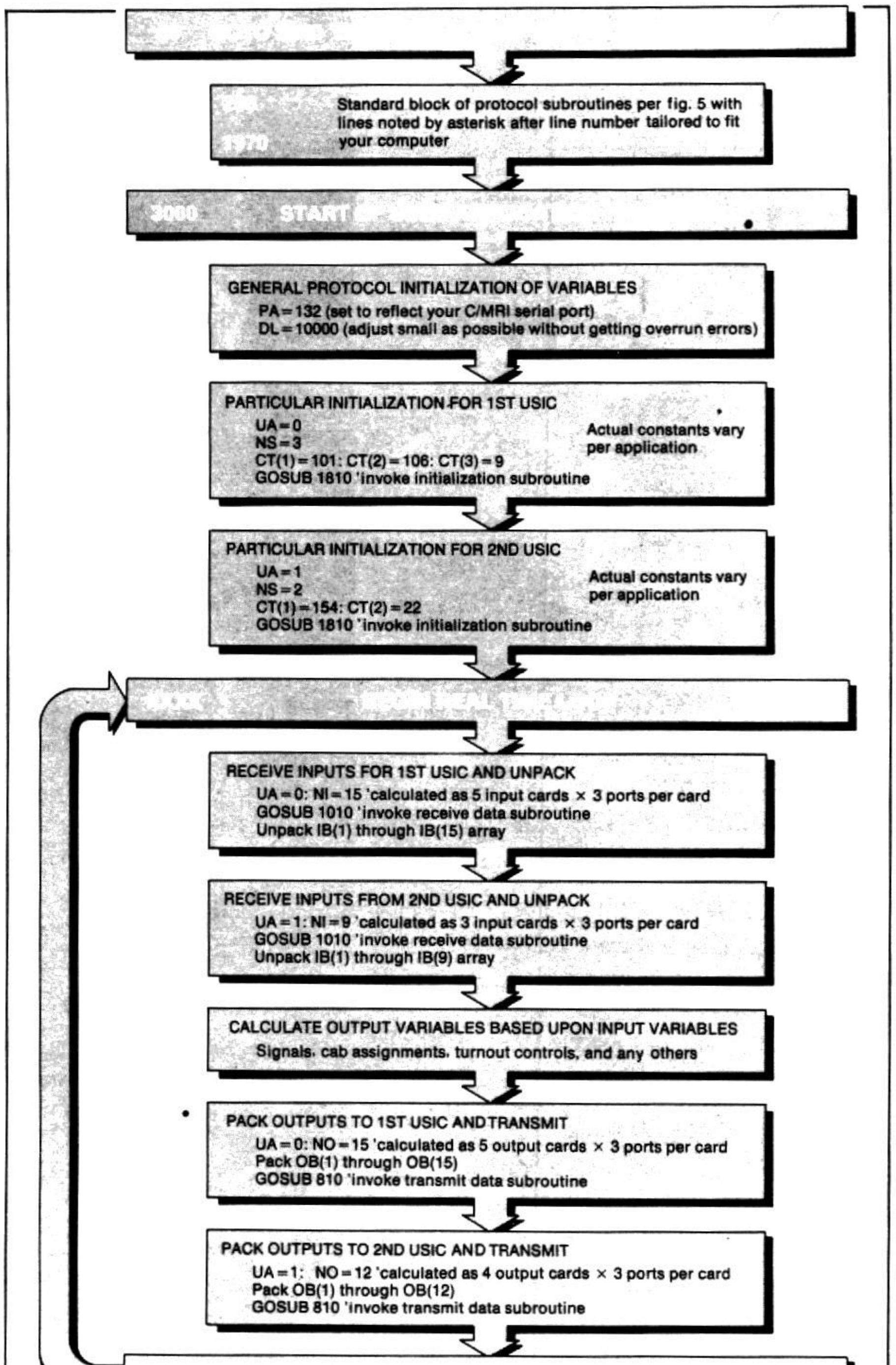


Fig. 6 EXAMPLE SERIAL C/MRI WITH TWO USICS



1970
Standard block of protocol subroutines per fig. 5 with lines noted by asterisk after line number tailored to fit your computer
3000 START
GENERAL PROTOCOL INITIALIZATION OF VARIABLES
PA = 132 (set to reflect your C/MRI serial port)
DL = 10000 (adjust small as possible without getting overrun errors)
PARTICULAR INITIALIZATION FOR 1ST USIC
UA = 0
NS = 3
CT(1) = 101: CT(2) = 106: CT(3) = 9
GOSUB 1810 'invoke initialization subroutine
Actual constants vary per application
PARTICULAR INITIALIZATION FOR 2ND USIC
UA = 1
NS = 2
CT(1) = 154: CT(2) = 22
GOSUB 1810 'invoke initialization subroutine
Actual constants vary per application
RECEIVE INPUTS FOR 1ST USIC AND UNPACK
UA = 0: NI = 15 'calculated as 5 input cards × 3 ports per card
GOSUB 1010 'invoke receive data subroutine
Unpack IB(1) through IB(15) array
RECEIVE INPUTS FROM 2ND USIC AND UNPACK
UA = 1: NI = 9 'calculated as 3 input cards × 3 ports per card
GOSUB 1010 'invoke receive data subroutine
Unpack IB(1) through IB(9) array
CALCULATE OUTPUT VARIABLES BASED UPON INPUT VARIABLES
Signals, cab assignments, turnout controls, and any others
PACK OUTPUTS TO 1ST USIC AND TRANSMIT
UA = 0: NO = 15 'calculated as 5 output cards × 3 ports per card
Pack OB(1) through OB(15)
GOSUB 810 'invoke transmit data subroutine
PACK OUTPUTS TO 2ND USIC AND TRANSMIT
UA = 1: NO = 12 'calculated as 4 output cards × 3 ports per card
Pack OB(1) through OB(12)
GOSUB 810 'invoke transmit data subroutine

Fig. 7 APPLICATION PROGRAM FORMAT

for USIC A, then repeated for USIC B, and the same is true for the pack and transmit operations.

I've prepared USIC versions of the Output Card Test Program and Automated Wraparound Test Program from Part 5, and of the Computer Cab Control Program from Part 10. They'd take up too much space to include here, but you can get them free, along with my explanations and USIC-C/MRI test procedures, by sending a stamped, self-addressed, business-size envelope to MODEL RAILROADER, PROGRAMS, 1027 N. Seventh St., Milwaukee, WI 53233.

PROGRAM TIMING

One point to keep in mind with the serial interface is that operational speed will be slower. For example, at 300 BAUD we are sending 300 bits per second. Our 12-block CCC railroad example from Parts 8, 9, and 10 used six I/O Cards at three ports each. That's 18 bytes, 144 bits, to be transmitted on each real-time loop. Thus it would take 480 milliseconds (mS), almost half a second, just to transmit the data. Going to 9600 BAUD would speed that up to 15mS.

On the other hand, an I/O setup the size of the Sunset Valley's — 23 I/O Cards, 69 bytes, 552 bits — would have transmission times of 1.8 sec at 300 BAUD and 57mS at 9600 BAUD. For medium to large layouts you definitely need the higher BAUD rates. Compared to typical processing speeds for a BASIC real-time loop of a second or two, these hardware speeds at the higher BAUD rate still aren't bad, although they are much slower than in parallel I/O.

The major slowness of the serial interface comes from the machine cycle overhead of the protocol subroutines. It's hard to give meaningful times for this, as it varies greatly from BASIC to BASIC and from machine to machine. Even though using noncompiled BASIC may be too slow for your "final" operational program, it's still a good way to get started developing your system.

The easiest way to make a significant speedup is to use a compiled BASIC, which has a double benefit as it speeds up your whole program and not just the protocol subroutines. Much faster operation can also be achieved by converting the protocol subroutines to your computer's assembly language or AL. Using AL your serial I/O speeds will approach the times of BASIC parallel I/O, except for the small overhead factor associated with formating messages and the address bytes sent along with the data.

Well, that's it for the serial interface. Next month I'll conclude the C/MRI series by showing how to add both digital-to-analog and analog-to-digital convertors to your C/MRI, and another form of block control using one computerized throttle for each block. See you then. ✪

Why computer speed control?

Solo operation: computer runs through trains while you run a yard switcher or a local freight.

The C/MRI: A computer/model railroad interface

Part 16 (conclusion): Digital/analog and analog/digital converters, and computerized throttles

BY BRUCE CHUBB

UP TO NOW we've been working with digital devices using on-off binary codes: either +5VDC or 0V, but nothing in between. In this final installment I'll explain how the C/MRI can monitor and control analog devices, those with a continuous range of input or output voltage.

A DC locomotive motor is an analog device; its speed range is continuous from stop to full. A power pack controls it with an analog signal, a continuously variable voltage. With the *D*igital-to-*A*nalog-*C*onverter Output Card, or DAC Card, C/MRI software can generate analog voltages for throttle control, light dimming, or even simulating locomotive gauges with meters.

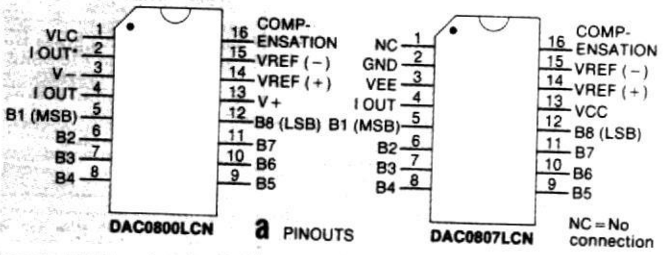
I'll also cover *A*nalog-to-*D*igital *C*onverter Input Cards, ADC Cards, to let the C/MRI receive analog voltage inputs. That way your computer can read track voltage or potentiometer settings. Finally, we'll see a *C*omputer-*C*ontrolled *T*hrottle or CCT that's useful for a different approach to controlling trains with blocks, Computer Block Control. First we'll look at controlling train speed with the C/MRI.

COMPUTER SPEED CONTROL

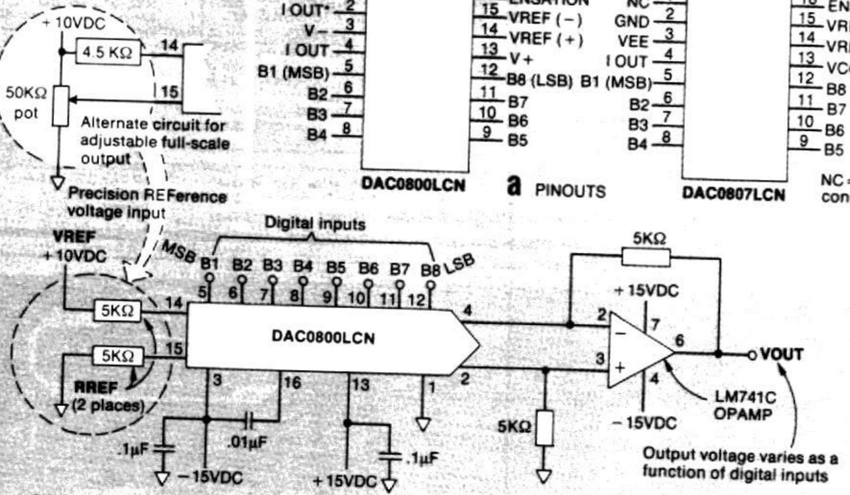
Most of us like to play the role of locomotive engineer and run our own trains. Why would we want the computer to run the train? Here are three good reasons.

Solo operation. The computer could run some trains while you ran one yourself, to add realism and fun when you operate alone. You might run a way freight, for example, or play the role of yardmaster, while the computer ran through trains automatically.

Display operation. Wouldn't it be great for the computer to run the railroad while you entertained guests? You could chat and answer questions while the computer orchestrated meets between trains, had passenger trains make station stops, and even used helpers to push tonnage freights up your big hill.

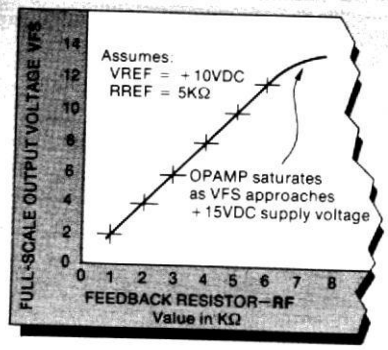


a PINOUTS


+10VDC
4.5 KΩ
50KΩ pot
Alternate circuit for adjustable full-scale output
Precision REFerence voltage input
VREF +10VDC
Digital inputs
MSB B1 B2 B3 B4 B5 B6 B7 B8 LSB
5KΩ
DAC0800LCN
RREF (2 places)
.1μF
.01μF
-15VDC
+15VDC
5KΩ
+15VDC
-15VDC
VOUT
LM741C OPAMP
Output voltage varies as a function of digital inputs

b CIRCUIT FOR SYMMETRICAL ±9.92 VDC OUTPUT ABOUT GROUND

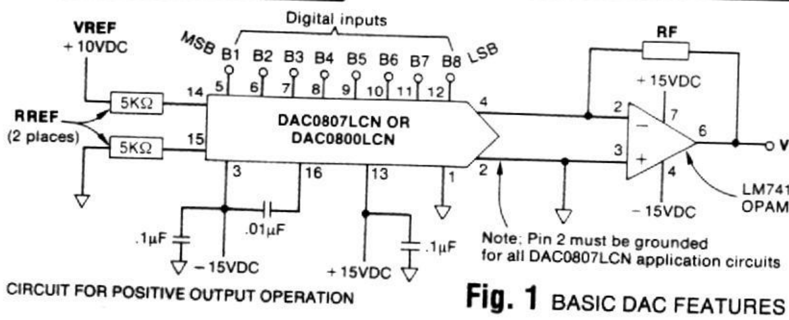
	B1	B2	B3	B4	B5	B6	B7	B8	EO
Pos. Full-scale	1	1	1	1	1	1	1	1	+9.920
Pos. Full-scale-LSB	1	1	1	1	1	1	1	0	+9.840
(+) Zero scale	1	0	0	0	0	0	0	0	+0.040
(−) Zero scale	0	1	1	1	1	1	1	1	−0.040
Neg. Full-scale+LSB	0	0	0	0	0	0	0	1	−9.840
Neg. Full-scale	0	0	0	0	0	0	0	0	−9.920



EQUATION FOR OUTPUT VOLTAGE – **VOUT**

$$\textbf{VOUT} = \text{VFS}\left[\frac{B1}{2}+\frac{B2}{4}+\frac{B3}{8}+\frac{B4}{16}+\frac{B5}{32}+\frac{B6}{64}+\frac{B7}{128}+\frac{B8}{256}\right]$$

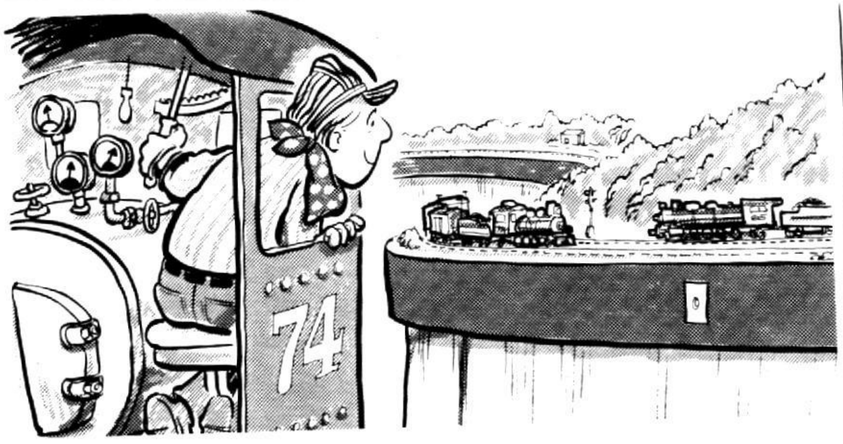
$$\text{Full-scale output voltage} = \frac{255}{256}\times\frac{\text{VREF}}{\text{RREF}}\times \text{RF} = .0019921\ \text{RF}$$



c CIRCUIT FOR POSITIVE OUTPUT OPERATION

Fig. 1 BASIC DAC FEATURES

Display operation: computer automatically runs several trains on your railroad, making meets and station stops, while you entertain your guests.



Cab realism: computer "in the loop" between you and your train simulates the effects of prototype locomotive controls and the readings on cab gauges.

Cab realism. With the computer as an intermediary between you and the locomotive, you can simulate prototype cab controls. Transistor throttles do this to some extent by mimicking momentum effects and brake functions, but with a computer "in the loop" the sky is the limit. You could even build a full-fledged locomotive/train simulator.

DAC CHIPS

A DAC converts digital signals into analog voltage (or current). This output can go directly into most transistor throttles as a replacement for the speed control potentiometer, enabling the DAC to control train speed. That's one way C/MRI software can run trains on your railroad.

The output of the DAC is a "stairstep" approximation of continuous voltage. An eight-bit DAC gives steps of $^1/_{255}$ of the full analog output. For example, for an eight-bit DAC set for a range of 0 to +12VDC, the smallest voltage step, called the "resolution," would be 12V ÷ 255, about .047V.

A digital 0 command to the DAC would give 0V output, a digital 1 would give +.047VDC, a 2 would give +.94VDC, and so on up to 255 for the full-scale +12VDC. This would give you smooth speed control, but note that a DAC's resolution is a function of the number of bits that carry the digital information. There are 10-, 12-, and 16-bit converters, but I'll stick to 8-bit (1 byte) converters which have ample resolution for C/MRI uses.

DAC ICs are flexible devices that, with different external circuit components, can give a variety of outputs. To keep costs down I'll cover only two inexpensive chips: the DAC0807LCN, the lowest-priced DAC IC in the Jameco catalog; and the DAC0800LCN, a slightly higher-priced but more flexible chip. Both come in 16-pin DIPs, and I've designed the C/MRI DAC Output Card to work with either chip.

Figure 1 summarizes the primary features of these ICs including pinouts and typical applications. In each application circuit I've attached an off-loading *OP*erational *AMP*lifier or OPAMP to the DAC outputs. The OPAMP buffers and amplifies the weak signal current coming from the DAC, to help maintain conversion accuracy and to drive heavier loads than could the DAC alone.

Looking at the pinouts in fig. 1a, the 8-bit digital input to both DACs is *via* pins 5-12. The MSB (*M*ost *S*ignificant *B*it) is B1, pin 5, and the LSB (*L*east *S*ignificant *B*it) is B8, pin 12. Both chips have multiplying (MDAC) capability, but for our use we'll keep the differential analog reference input on pins 14 and 15 constant, so the analog output is simply proportional to the digital input.

The 0800's output is a differential current between pins 4 and 2, while the 0807's output current is single-ended on pin 4 with pin 2 as ground. Pin 1 of the 0800 sets the logic switching levels of the digital input. For the C/MRI, pin 1 is grounded for logic levels compatible with standard TTL logic (+5VDC/0V).

Figure 1b shows a typical application circuit for the more "elegant" of the two ICs. Varying the decimal value of the digital input from 0 to 255 (by either a POKE A,N or an OUT A,N instruction in the software, where A is the DAC's 8255 port address and N is the desired value from 0 to 255) would vary the analog output VOUT between −9.92 and +9.92VDC as in the fig. 1b table. The full-scale value, here 9.92V, can be made adjustable by inserting the optional trim potentiometer in the input reference circuit.

Figure 1c is a simplification of 1b, for positive analog output only. Either chip can be used but the 0807 is usually least expensive. The equation for the output voltage is given to show how the stackup of the input bits is done in binary powers-of-two fashion to arrive at the analog output. Each B value is a 1 if the software has that bit position "on" and 0 if "off."

The full-scale voltage, VFS, is a function of the VREF and RREF values and the OPAMP feedback resistor RF. By using standard values of VREF = +10VDC and RREF = 5KΩ, then selecting RF from a range of 1K to 6KΩ, you can achieve full-scale outputs between 2V and 12V. The relationship of VOUT's full-scale value to RF is shown by the graph in fig. 1c; it is linear up to the point where the OPAMP saturates as VFS approaches the +15VDC supply voltage.

DAC OUTPUT CARD

Figure 2 shows a completed DAC Output Card. It's designed to plug onto the IOMB just like the general-purpose Output Card. The DAC Card uses three DAC ICs for three independent analog outputs. Your layout connects to the 10-pin Molex connector at the top; its lines include +

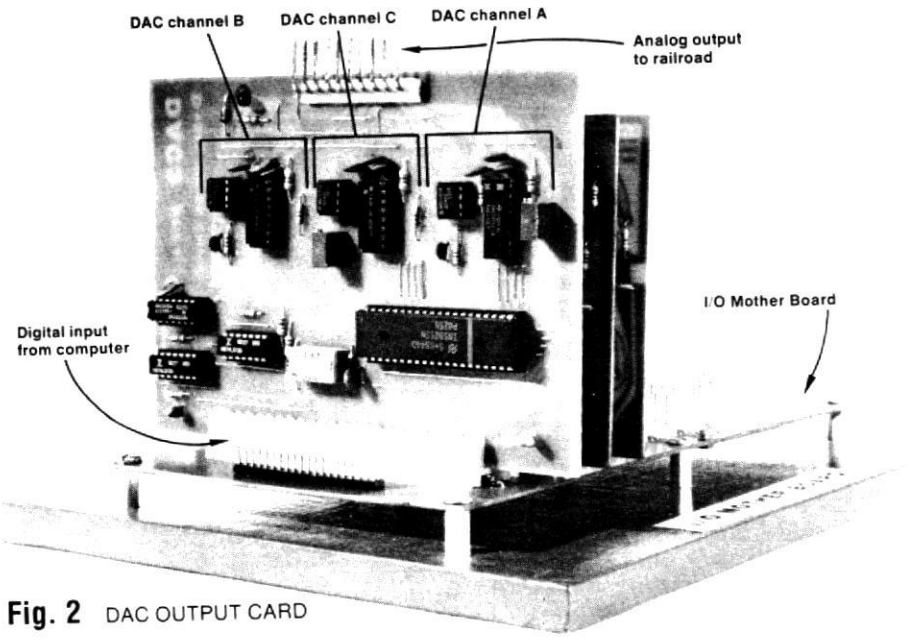


Fig. 2 DAC OUTPUT CARD

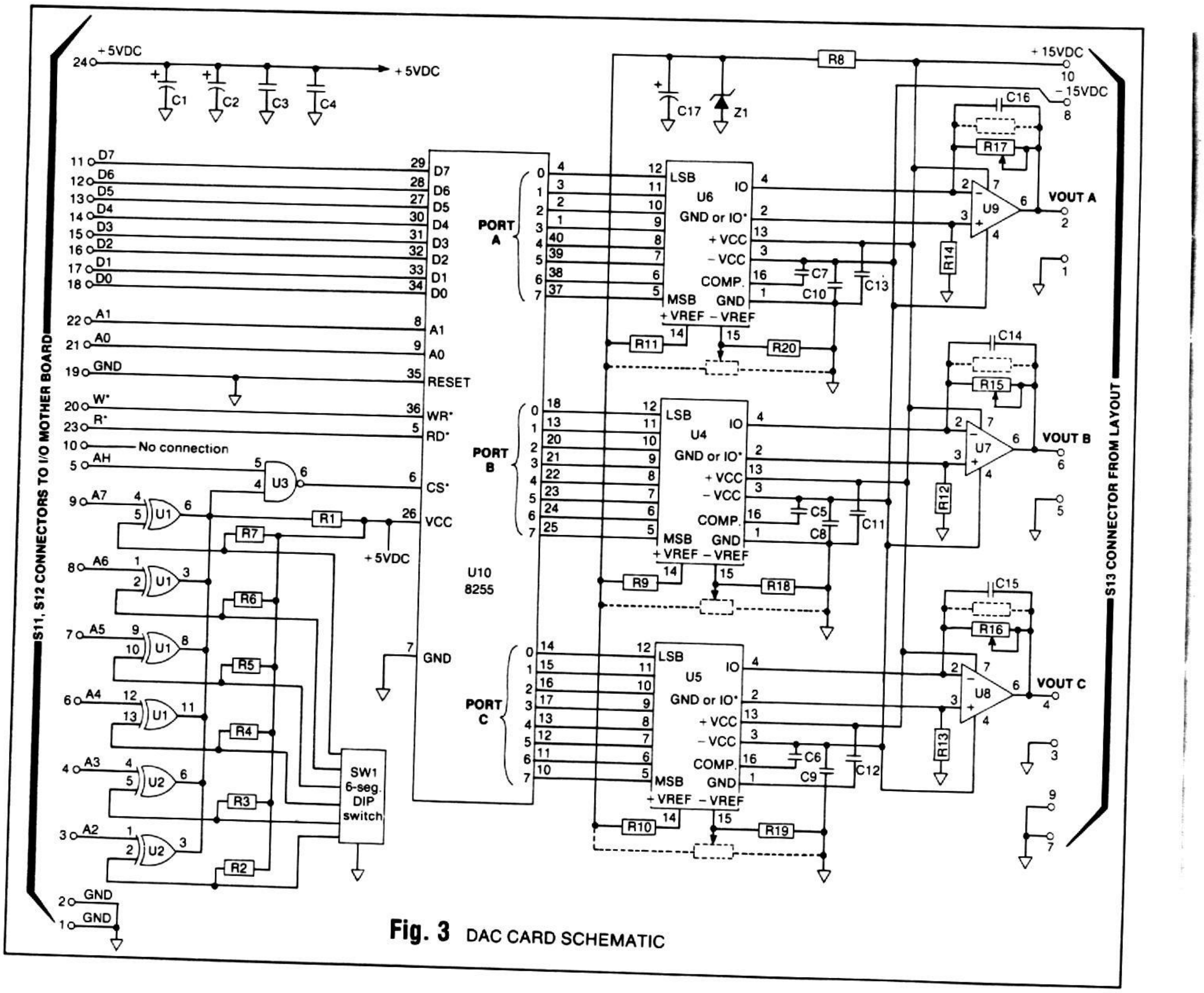


Fig. 3 DAC CARD SCHEMATIC

and −15VDC power for the DACs and their OPAMPS, three analog outputs, and five ground connections. The DAC Card uses the same DIP-switch circuit for address decoding as the Output Card, and the same 8255 chip for switching data bytes to the appropriate DAC IC.

The three independent DAC circuits let you install different parts and jumpers in each, for symmetrical operation about ground with the 0800, or positive-only output voltages with either the 0800 or the 0807. Also, you can install fixed resistors for fixed full-scale settings or trim pots for adjustable full-scale values.

The example illustrates the DAC Card's flexibility. I've configured the analog channel A circuit, corresponding to Port A of the 8255, to use the 0800 IC for symmetrical output about ground and an adjustable full-scale value. The channel B circuit is set up to use the 0807 IC for positive-only output with a fixed full-scale value of +12VDC. Channel C is the same as B except that the full-scale value is adjustable.

Figure 3 is the DAC Card schematic. Its left side, including the 8255, is identical to that of the Output Card. The 8255's eight-bit ports A, B, and C connect directly to the digital inputs of the three DACs. One +10VDC zener diode, Z1, is the +VREF source for all three DACs.

The four resistors around each DAC/OPAMP combination vary in different configurations. For example, you'd use 5KΩ resistors as R12-R14 for symmetrical output with the 0800, but replace them with jumpers (0Ω resistance) for positive-only output. Note that R12-R14 must ALWAYS be jumpers when using the 0807. The OPAMP feedback resistors R15-R17 can be fixed resistors selected for one full-scale value, or may be replaced by a trim pot for adjusting output. Likewise, reference input resistors R18-R20 can be fixed resistors or a trim pot.

Feedback capacitors C14-C16 are in parallel with the OPAMP feedback resistors to reduce the amplifier's high frequency gain, which improves circuit stability and reduces output noise.

BUILDING THE DAC CARD

Figure 4 includes the parts layout and parts list for the DAC Card. Ready-to-use circuit cards are available from JLC Enterprises, P. O. Box 88187, Grand Rapids, MI 49518. The DAC3 is $24 and there is a shipping and handling charge of $2.50 per order (Michigan residents must add 4 percent sales tax).

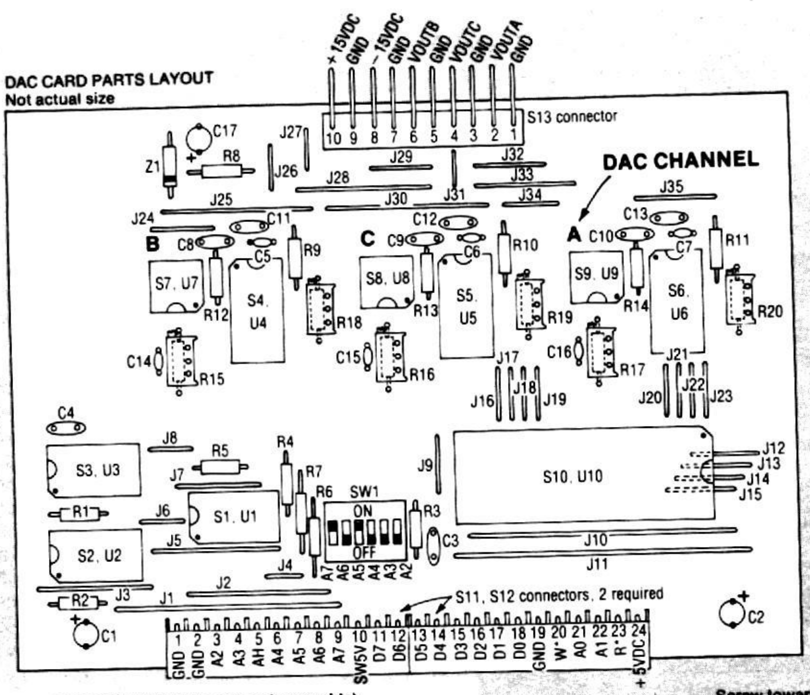
In the parts list, the parts which vary depending on how you want to configure each DAC circuit are marked with asterisks; select those for your applications as described above. Trim pots are more expensive than fixed resistors, but they can be adjusted without being unsoldered.

For R9-R20 I've called out the nearest standard ¼W, 5-percent resistor values, for example 5100Ω in place of 5000Ω for RREF as in fig. 1b. That's fine for most applications, but for greater accuracy and less shifting of output voltage with temperature variation (i.e., warmup), substitute metal film resistors with 1-percent tolerance. These are listed in the Digi-Key catalog, and their price is about double that of 5-percent resistors. They do come in finer increments, so you can get, for example, a 4.99KΩ resistor for RREF.

To assemble the DAC Card refer to fig. 4 and follow these steps:

☐ **Card test.**

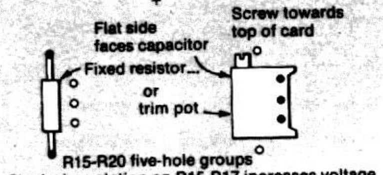
DAC CARD PARTS LAYOUT
Not actual size



DAC PARTS LIST (In order of assembly)

Qnty.	Symbol	Description
35	J1-J35	Jumpers, make from no. 24 uninsulated bus wire (Belden no. 8022 or equivalent)
1	R1	1000Ω resistor [brown-black-red]
6	R2-R7	2200Ω resistors [red-red-red]
1	R8	150Ω, ½W resistor [brown-green-brown]
3*	R9-R11	5100Ω resistors [green-brown-red] or substitute 4700Ω resistors [yellow-violet-red] if using 50KΩ trim pot for R18-R20
3*	R12-R14	Use jumpers (R = 0Ω) of no. 24 uninsulated bus wire (Belden no. 8022) for use with DAC0807LCN and also with DAC0800LCN for single-ended output as in fig. 1c. Use 5100Ω resistors [green-brown-red] with DAC-0800LCN for +/− symmetrical output as in fig. 1b
3*	R15-R17	Use 5100Ω resistors [green-brown-red] with DAC0800LCN if +/− symmetrical output is desired as in fig. 1b, otherwise select fixed resistor values per graph or equation in fig. 1c for desired full-scale output or install 10KΩ trim potentiometers (Digi-Key CFG14) for adjustable full-scale output voltage
3*	R18-R20	Use 5100Ω resistors [green-brown-red] or with DAC0800LCN install 50KΩ potentiometer (Digi-Key CFG54) for adjustable full-scale output
1	Z1	10V, 1W zener diode (Digi-Key 1N4740A)
3	S1-S3	14-pin DIP sockets (Digi-Key C8914)
3	S4-S6	16-pin DIP sockets (Digi-Key C8916)
3	S7-S9	8-pin DIP sockets (Digi-Key C8908)
1	S10	40-pin DIP socket (Digi-Key C8940)
2	S11, S12	12-pin connectors (Molex 09-52-3121)
1	S13	10-pin connector (Molex 09-66-1101)
1	SW1	6-segment DIP switch (Digi-Key CT2066)
2	C1, C2	2.2μF, 35V tantalum capacitors (Jameco TM2.2/35)
2	C3, C4	.1μF, 50V ceramic disk capacitors (Jameco DC.1/50)
3	C5-C7	.01μF, 50V ceramic disk capacitors (Jameco DC.01/50)
6	C8-C13	.1μF, 50V ceramic disk capacitors (Jameco DC.1/50)
3	C14-C16	.001μF, 50V ceramic disk caps. (Jameco DC.001/50)
1	C17	10μF, 25V tantalum capacitor (Jameco TM10/25)
2	U1,U2	74LS136 quad two-input exclusive-OR gate (Jameco)
1	U3	74LS00 quad two-input NAND gate (Jameco)
3	U4-U6	8-bit D/A converters (Jameco DAC0807LCN) or for special high-accuracy applications (see text) use Jameco DAC0800LCN
3	U7-U9	Operational amplifiers (Jameco LM741CN)
1	U10	8255 or 8255A programmable peripheral interface (Jameco)

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



R15-R20 five-hole groups
Clockwise rotation on R15-R17 increases voltage
Clockwise rotation on R18-R20 decreases voltage

IC POWER TEST

✓	IC	+ METER LEAD ON PIN NO.	− METER LEAD ON PIN NO.	VOLTAGE READING
	U1	14	7	+5VDC
	U2	14	7	+5VDC
	U3	14	7	+5VDC
	U4	13	1	+15VDC
	U4	1	3	+15VDC
	U5	13	1	+15VDC
	U5	1	3	+15VDC
	U6	13	1	+15VDC
	U6	1	3	+15VDC
	U7	7	*	+15VDC
	U8	7	*	+15VDC
	U9	7	*	+15VDC
	U7	*	4	+15VDC
	U8	*	4	+15VDC
	U9	*	4	+15VDC
	U10	26	7	+5VDC

* = U4 pin number 1

Fig. 4
DAC CARD CONSTRUCTION

Note: Program listing shown assumes UBEC with H-8 memory-mapped I/O. For other machines, change CA to your card address and change/add other statements per Part 6, fig. 11 and 12. For USIC, revise per Part 15 to include protocol subroutines

Fig. 5 VOUT TEST PROGRAM

```
10    PRINT "VOUT TEST PROGRAM FOR DAC3"
20    CA = 1024   'set for your own card address
30    PN = 0  'for Chan. A, =1  Chan. B, =2  Chan. C
40    POKE CA-3,128  'initialize 8255 for output
50    FOR N= 0 TO 255
60    PRINT N
70    POKE CA-PN,N  'output to DAC card
80    NEXT N
90    GOTO 10    'repeat test
```

☐ **J1-J35.** Jumpers J12-J15 will lie partially underneath S10.

☐ **R1-R20.** Where parts are marked by asterisks be sure to install the right component for the type of DAC output you want, either fixed resistors, jumpers (R12-R14 positions), or trim pots (R15-R20 positions). There are five holes at each set of R15-R20 locations; use them as shown in the fig. 4 inset drawing.

☐ **Z1[+].**

☐ **S1-S10[+].** Be sure S10 is tight against the card, not held away by J12-J15.

☐ **S11-S13.**

☐ **SW1[+].** Install so DIP switch contacts are "ON" when thrown toward S13.

☐ **C1,C2[+].**

☐ **C3-C16.**

☐ **C17[+].**

☐ **U1-U10[+].**

☐ **Cleanup and inspection.**

☐ **IC power test.** Set SW1 to the desired card slot address. Plug the DAC Card into your IOMB, and connect pins 7-10 of S13 to a ±15VDC supply — use the same supply as for your Optimized Detectors, Part 7 fig. 7, or a separate one as in Part 9 fig. 5. Turn on both the ±15VDC and your +5VDC IOMB supplies, then use your VOM and the fig. 4 table to see that power reaches each IC. If not, look for a missing or shorted jumper, an IC reversed, or bad soldering. Keep checking until you pass all power tests.

☐ **Reference voltage test.** Touch your VOM's + lead to the banded end of Z1 and the − lead to U4 pin 1 — you should read between +9.5 and +10.5VDC. If not, then Z1 is backwards or defective, your soldering is faulty, or one or more of ICs U4-U6 is defective. Locate and correct the problem before proceeding.

☐ **Channel A output voltage test.** Attach the − lead of your VOM to Channel A S13 pin 1 (ground), and the + lead to pin 2 (VOUTA). Enter and RUN the VOUT Test Program in fig. 5.

The VOM should be initialized to 0VDC, then each program loop should increment the output voltage by one step, reaching full scale when N = 255. The voltage of each step should equal N x full-scale value ÷ 255. If your card has trim pots, you can stop the program at N = 255 and turn the pot adjustment screw to set the full-scale value. The 25-turn pots I've specified allow precise settings.

If you see no voltage when the program runs, you either have an error in the software, an address error in your UBEC/USIC or DAC Card DIP switches, or a fault in the DAC Card itself. You can check the software and addresses by replacing the DAC Card with a good Output Card and the Output Test Panel connected as in Part 5 fig. 9.

Set the Output Card's DIP switch to the same setting as your DAC Card's, then RUN the VOUT Test Program. The LEDs for port A should initialize to all off, then with each loop they should display the binary pattern corresponding to each value of N. If this doesn't work, unplug all other cards on the IOMB to make sure you don't have an address conflict problem, two cards with the same DIP switch settings. Also recheck your address calculations.

Once the program works with an Out-

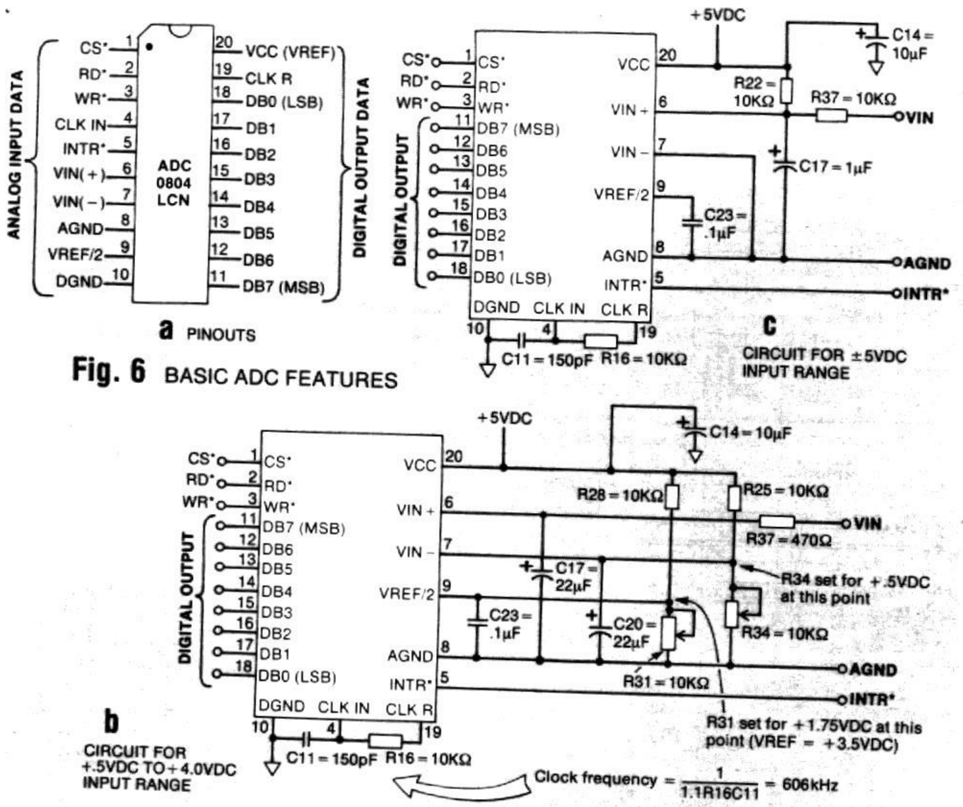


Fig. 6 BASIC ADC FEATURES

put Card, replace the DAC Card and VOM leads, and RUN the program again. If it works you're all set; if not, debug the card itself, especially for faulty soldering.

☐ **Channel B and C tests.** Use the same procedure to test the other two DAC channels, moving the VOM leads to appropriate S13 pins, and changing program line 40 to PO = 1 for channel B and PO = 2 for C. This completes the DAC Card.

ADC CHIPS

An ADC converts analog voltage into digital data, like a DAC in reverse. Again the precision of conversion is a function of the number of data bits. ADCs are necessarily more expensive than DACs, and for C/MRI use I'll stick to one eight-bit chip, the ADC0804LCN. It comes in a 20-pin DIP, and is very flexible even though it's the lowest priced ADC in Jameco's catalog.

Figure 6 summarizes the features of the 0804. Let's look at the pinouts first in fig. 6a. Pins 6 and 7 can receive differential analog input, or, if the pin 7 VIN(−) input is grounded, pin 6 can receive single-ended input. Differential input reduces common-mode noise, as in RS-422A I/O. Pin 7 can also be used to automatically subtract a fixed voltage value from the input reading. For maximum accuracy and noise immunity the *A*nalog *GrouND* or AGND on pin 8 is separate from the *D*igital *GrouND* or DGND on pin 10.

The 0804 ADC generates an eight-bit digital output on pins 11-18 in proportion to the analog input voltage measured across pins 6 and 7. The MSB is DB7, pin 11, and the LSB is DB0, pin 18. The eight output lines are all logic zero when the differential voltage across pins 6 and 7 is zero (off), and all ones (full scale) when the voltge across pins 6 and 7 equals reference voltage VREF.

Unless the 0804 receives an input on pin 9, VREF equals supply voltage VCC from pin 20. With +5VDC on pin 20, the digital output on pins 11-18 would be all ones when the differential analog input across pins 6 and 7 reached +5VDC.

For greater flexibility, pin 9's VREF/2 input can adjust the span or range of the analog input voltage. For example, suppose you want to handle a dynamic input range of +.5 to +4VDC, a span of 3.5V. Apply +.5VDC to the VIN(−) pin to absorb the +.5VDC zero offset, and make the VREF/2 input equal to 3.5 ÷ 2, or +1.75VDC. Now the ADC encodes the input on pin 6 so that +.5VDC gives a digital output byte of all zeros, and +4VDC gives a full-scale output byte of all ones. That way you get full eight-bit resolution even with a reduced input range.

Figure 6b shows an application circuit for the example above. Resistor R25 and pot R34 are selected and set so that VIN(−), pin 7, is held at the zero-offset value of +.5VDC. Likewise R28 and R31 hold VREF/2, pin 9, at +1.75VDC. Capacitors C17, C20, and C23 suppress electrical noise into the 0804 which could reduce the conversion's accuracy, and 10µF tantalum capacitor C14 minimizes transients on the supply/reference line.

The conversion process is triggered by simultaneous lows on the Chip Select (CS*) and WRite (WR*) control lines—the 0804 is designed to hang right on a computer's bus and has its own CS*, RD*, and WR* inputs just like the 8255. The ADC makes its conversion in a sequential manner using "successive approximation logic," a test to see which bits must be turned on to best match the input voltage.

The 0804 tests the highest-order bit first, and after eight comparisons (64 clock cycles) the eight-bit code is set on the output pins. C11 and R16 together determine the chip's internal clock frequency, in this case 606kHz. Total conversion time, from request until the digital byte can be read, is under 100ms.

When the output byte is ready, the 0804 sets pin 5 low, an *INTeR*upt or INTR* signal to tell the "outside world" that the ADC data can be read. To read the data, lines RD* (ReaD) and CS* are both brought low, enabling the chip's tri-state output buffer.

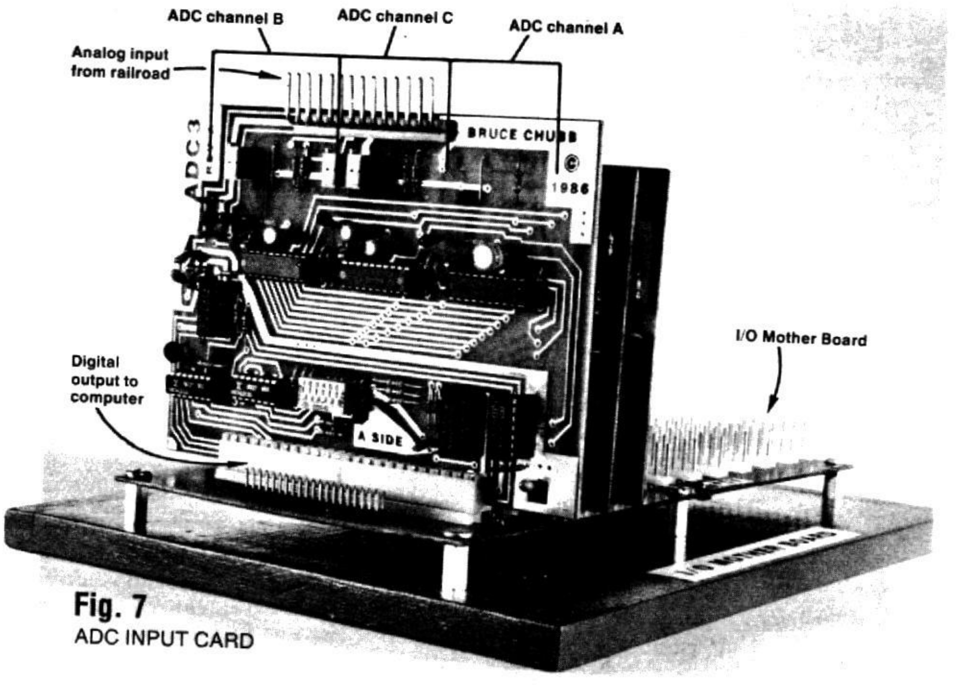


Fig. 7 ADC INPUT CARD

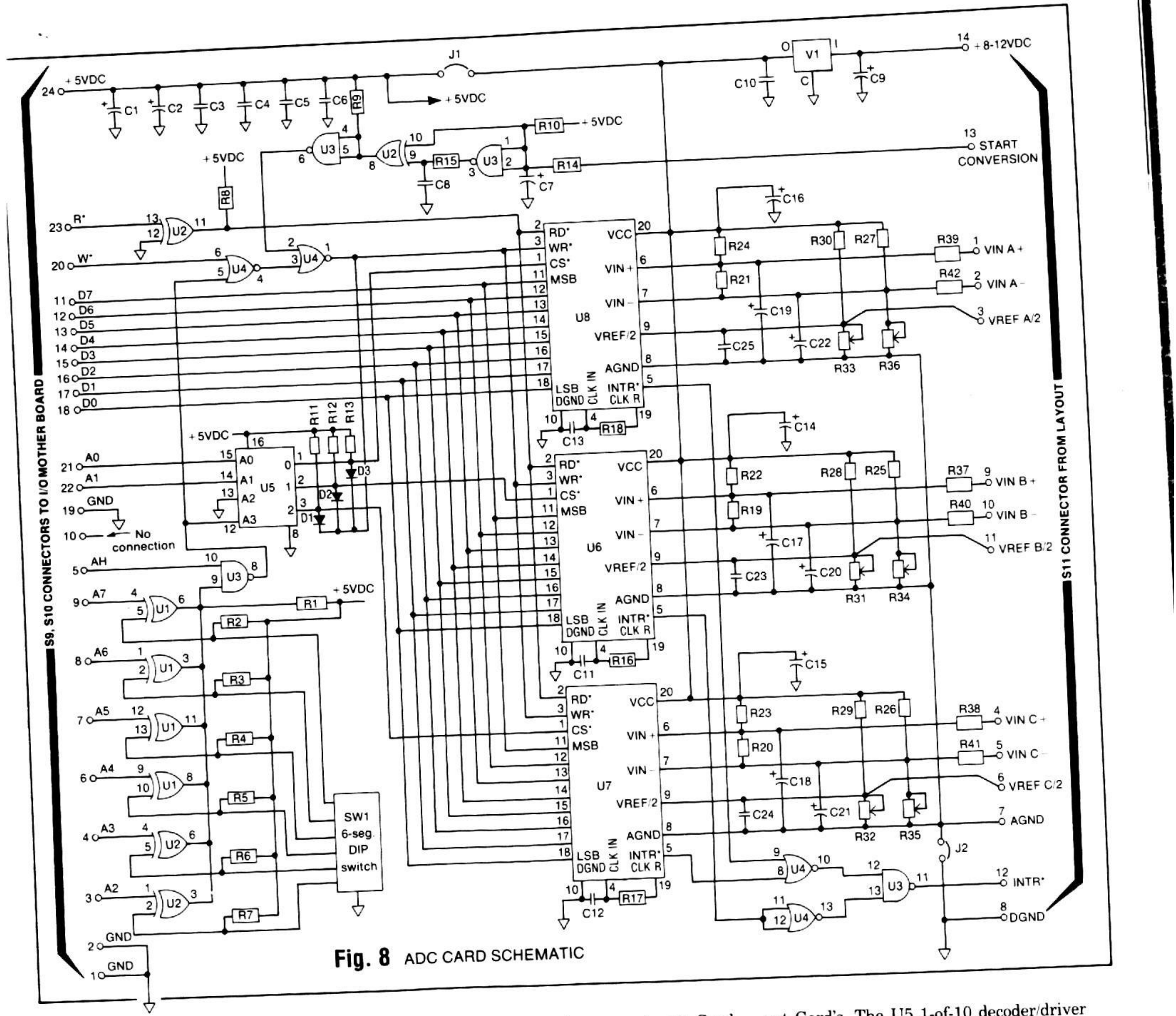


Fig. 8 ADC CARD SCHEMATIC

Figure 6c shows the configuration to read an analog input range of ±5VDC. Its operation is easy to understand if you start with the end points of the range. When VIN is −5VDC, and with R22 equal to R37, pin 6 is midway between +5 and −5VDC, 0V, giving a digital output of all zeroes. At the other end of the range, with VIN equal to +5VDC, pin 6 is +5VDC, the full-scale VREF value, so the digital output is all ones. Any input between the end points yields a proportional in-between digital output.

Change R22 to 5KΩ, and add an R19 of 10KΩ between pin 6 and AGND pin 8, and the effective input range changes to ±10VDC. Keeping R19 at 10KΩ, changing R37 to 14KΩ, and deleting R22 shifts the input range to 0-12VDC.

ADC INPUT CARD

Figure 7 shows a completed ADC Input Card, which plugs into the IOMB just like a general-purpose Input Card. The ADC Card uses three 0804 ICs for three independent analog inputs. Your layout connects to S11, the 14-pin Molex connector at the top, and the address decoding/DIP switch circuit is like the Input Card's.

Each of the three ADC circuits is designed so you can use different parts and jumpers to tailor it for reading inputs with different zero offsets and dynamic ranges. You can install fixed resistors for preset ranges or trim pots for adjustment.

The example illustrates the ADC Card's flexibility. I've configured the Analog A circuit with fixed resistors to handle an input range of 0 to +5VDC. The Analog B circuit is set up for a range of 0 to some positive full-scale voltage set by trim pot adjustment. The Analog C circuit is adapted for a range of +0.5 to +4VDC.

Figure 8 is the ADC Card schematic. The bottom left side is identical to the Input Card's. The U5 1-of-10 decoder/driver decodes the A0 and A1 inputs, along with the card-selected signal from U3's pin 8, to pull the proper CS* (Chip Select) line low to activate the ADC being addressed.

The U5 outputs have open collectors, so when W* goes low along with pin 8 of U3, U4's pin 1 output goes low. Through diodes D1-D3, that pulls all ADC CS* inputs low to simultaneously trigger data conversions. When the R* line goes low to read the digital data, only the addressed CS* line goes low, enabling only one ADC output buffer at a time.

In a serial interface the ADC Card is defined as an input card in the CT( ) array. You can't write to an input card through a USIC, so I've included an alternate conversion-start circuit on the ADC Card, using leftover gates from U2 and U3.

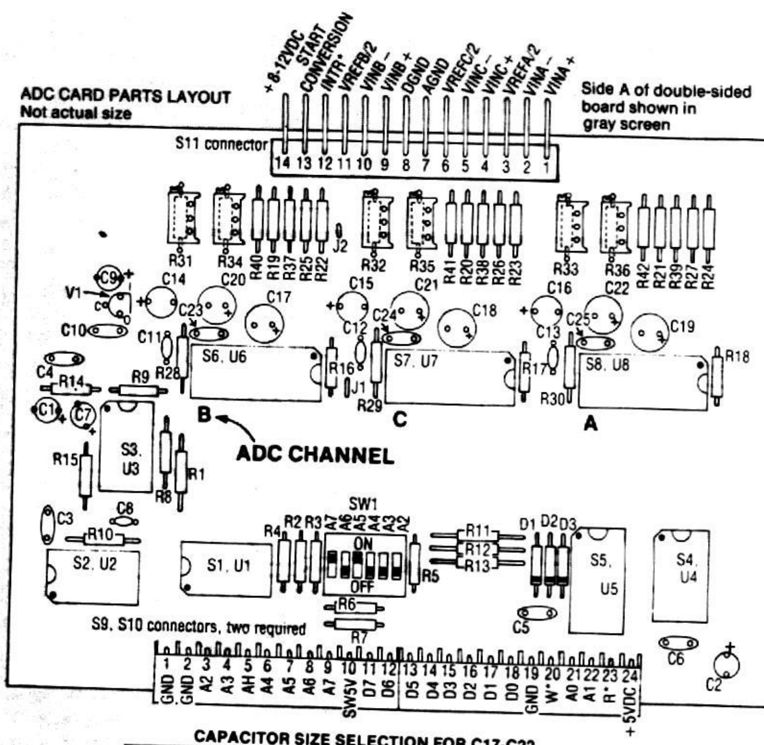
The conversion-start circuit operates just like the trigger circuit for the 555 timer IC in the SM2 single-output switch

ADC PARTS LIST (In order of assembly)

Qnty.	Symbol	Description
2	J1,J2	Jumpers, make from no. 24 uninsulated bus wire. (Belden no. 8022). Omit J1 if using alternate power supply with V1 as noted below. Omit J2 to separate analog and digital grounds
1	R1	1000Ω resistor [brown-black-red]
12	R2-R13	2200Ω resistors [red-red-red]
2	R14, R15	47Ω resistors [yellow-violet-black]
3	R16-R18	10KΩ resistors [brown-black-orange]
3*	R19-R21	Open circuit for VIN+ full scale less than +5VDC or install 10K resistors [brown-black-orange] if VIN+ full scale to be greater than +5VDC
3*	R22-R24	Open circuit for VIN+ only positive values or select as fixed per fig. 6c and text to handle negative VIN+ voltage values
3*	R25-R27	Open circuit for single-ended inputs with VIN− grounded or install 10KΩ resistors [brown-black-orange] to set zero-offset via R34-R36
3*	R28-R30	Open circuit for VREF = VCC or install 10KΩ resistors [brown-black-orange] to set VREF/2 via R31-R33
3*	R31-R33	Open circuit for VREF = VCC or install fixed resistor or 10KΩ potentiometer (Digi-Key CFG14) to set VREF/2
3*	R34-R36	Jumper for single-ended inputs with VIN− grounded or select as either fixed resistor or 10KΩ potentiometer (Digi-Key CFG14) for setting non-zero zero-offset (along with R25-R27) per fig. 6b
3*	R37-R39	470Ω resistors [yellow-violet-brown] for VIN+ full scale less than +5VDC or install greater fixed resistor to attenuate (along with R19-R21) VIN+ values greater than +5VDC
3*	R40-R42	Open circuit for single-ended inputs with VIN− grounded, otherwise install 470Ω resistor [yellow-violet-brown]
3	D1-D3	1N4148 switching diodes (Jameco)
4	S1-S4	14-pin dip sockets (Digi-Key C8914)
1	S5	16-pin DIP socket (Digi-Key C8916)
3	S6-S8	20-pin DIP sockets (Digi-Key C8920)
2	S9, S10	12-pin connectors (Molex 09-52-3121)
1	S11	14-pin connector (Molex 09-66-1141)
1	SW1	6-segment DIP switch (Digi-Key CT2066)
2	C1, C2	2.2μF, 35V tantalum capacitors (Jameco TM2.2/35)
4	C3-C6	.1μF, 50V ceramic disk capacitors (Jameco DC.1/50)
1	C7	1μF, 35V tantalum capacitor (Jameco TM1/35)
1	C8	.022μF, 50V ceramic disk capacitor (Jameco DC.022/50)
3	C11-C13	150pF, 500V ceramic disk capacitors (Digi-Key P4102)
3	C14-C16	10μF, 25V tantalum capacitors (Jameco TM10/25)
3*	C17-C19	Select using equation in text and pick nearest standard part in table between 1μF and 220μF
3*	C20-C22	Open circuit when R34-R36 is jumper or install 1μF, 35V tantalum capacitor (Jameco TM1/35) with non-zero zero-offset or for differential input of VIN− select using equation in text and pick nearest standard part in table between 1μF and 220μF
3	C23-C25	.1μF, 50V ceramic disk capacitors (Jameco DC.1/50)
2	U1, U2	74LS136 quad two-input exclusive-OR gate (Jameco)
1	U3	74LS00 quad two-input NAND gate (Jameco)
1	U4	74LS02 quad two-input NOR gate (Jameco)
1	U5	74LS145 1-of-10 decoder/driver with open collector outputs (Jameco)
3	U6-U8	8-bit analog-to-digital (ADC) converters (Jameco ADC0804LCN)
Optional parts used for separate ADC power supply		
1	V1	100mA, 5V positive regulator (Jameco 78L05AC)
1	C9	.33μF, 35V tantalum capacitor (Jameco TM.33/35)
1	C10	.1μF, 50V ceramic disk capacitor (Jameco DC.1/50)

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

ADC CARD PARTS LAYOUT
Not actual size



+8-12VDC
START CONVERSION
INTR
VREFB/2
VINB−
VINB+
DGND
AGND
VREFC/2
VINC−
VINC+
VREFA/2
VINA−
VINA+
Side A of double-sided board shown in gray screen
S11 connector
14 13 12 11 10 9 8 7 6 5 4 3 2 1
R31 R34 R19 R40 R37 R25 R22 J2 R32 R35 R41 R20 R38 R26 R23 R33 R36 R42 R21 R39 R27 R24
V1 C9 C14 C20 C17 C15 C21 C18 C16 C22 C19
C10 C11 C23 C12 C24 C13 C25
S6, U6 S7, U7 S8, U8
R16 R28 J1 R29 R17 R30 R18
C4 R14 R9
C1 C7
B C A
ADC CHANNEL
R15 S3, U3 R8 R1
C3 C8 R10
SW1 R4 R2 R3 A7 A6 A5 A4 A3 A2 ON OFF
R11 R12 R13 D1 D2 D3
S2, U2 S1, U1 R5 S5, U5 S4, U4
R6 R7 C5 C6 C2
S9, S10 connectors, two required
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
GND GND A2 A3 AH A4 A5 A6 A7 SW5V D7 D6 D5 D4 D3 D2 D1 D0 GND W* A0 A1 R* +5VDC

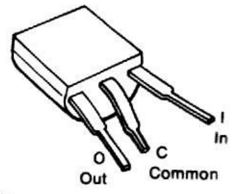
CAPACITOR SIZE SELECTION FOR C17-C22

Capacitor	Source	Lead spacing
1μF, 35V tantalum	Jameco TM1/35	Variable
2.2μF, 35V tantalum	Jameco TM2.2/35	Variable
4.7μF, 100V electrolytic	Digi-Key P5092	.098"
10μF, 63V electrolytic	Digi-Key P6079	.098"
22μF, 35V electrolytic	Digi-Key P6051	.098"
47μF, 16V electrolytic	Digi-Key P6026	.098"
100μF, 10V electrolytic	Digi-Key P6014	.098"
220μF, 10V electrolytic	Digi-Key P6015	.138"

IC POWER TEST

✓	IC	+METER LEAD ON PIN NO.	−METER LEAD ON PIN NO.	VOLTAGE READING
	U1	14	7	+5VDC
	U2	14	7	+5VDC
	U3	14	7	+5VDC
	U4	14	7	+5VDC
	U5	16	8	+5VDC
	U6	20	10	+5VDC
	U7	20	10	+5VDC
	U8	20	10	+5VDC

Clockwise rotation on all potentiometers increases voltage



Pinouts for regulator V1 with TO-92 case. Bend leads as shown to fit holes in card

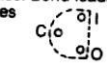


Fig. 9 ADC CARD CONSTRUCTION

machine control from Part 6 fig. 7, except that available NAND gates of U3 are used for the inversion function. Each transition of the conversion-start line (high-to-low or low-to-high) triggers the ADCs to start their conversions. You can drive any number of ADC Cards with a single line from an Output Card which changes state each time through the real-time loop, like the loop-timer LED output from pin 23 of card 5 in Part 9 fig. 3.

That way, each time your program writes to the Output Cards it toggles the conversion-start line and triggers the ADC conversions. Since the 100μs the ADCs take to make conversions is quick compared to real-time loop processing, by the time your program reads back the Input Cards, the ADCs are updated to reflect last-iteration data.

For the greatest accuracy use a separately regulated supply for ADC reference voltage, and separate grounds AGND and DGND. Connect a +8 to 12VDC supply through pin 14 of S11 to 5V regulator V1. This serves as a stable local reference-voltage supply for the ADCs.

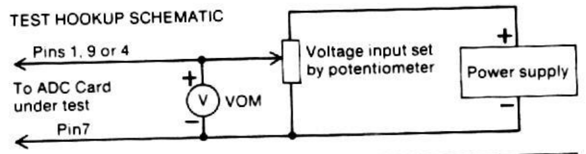
If accuracy isn't that important in your application, V1, C9, and C10 can be omitted, and jumper J1 used to connect the ADC reference line to the +5VDC IOMB power. If you use V1, *do not* install J1. Noise-filter capacitors C17 through C25 are all tied to AGND. For applications where separate grounds aren't necessary, use jumper J2 to tie AGND into the common card digital ground on the IOMB.

If you want VREF = VCC = +5VDC, do not install resistors R28-R33. If you want lower VREF values for reduced input range, use 10KΩ fixed resistors for R28-R30, and select R31-R32 for the desired pin 9 VREF/2 voltage. If you want a zero-offset voltage, install R25-R27 (10KΩ fixed resistors) and select R34-R36 for the desired pin 7 offset voltage.

When you want neither differential input or an offset zero, replace resistors R34-R36 with jumpers for 0Ω resistance. Also, delete C20-C22, R25-R27, and R40-R42 if jumpers are substituted for R34-R36. If the range of VIN is to include negative values, install R22-R24. Similarly, if the range of VIN is to go above +5VDC, install R19-R21.

Resistors R37-R42 should never be less than 470Ω, to help protect the ADC if you accidentally connect the VIN pins of S11 to high voltages. If you want the effective input range to include voltages above +5VDC, use 10KΩ resistors for R19-R21, and select R37-R39 to provide a level of attenuation such that full-scale input corresponds to +5VDC at pin 6.

To minimize the chance of a noise spike input causing an output error, it's good practice to make capacitors C17-C19, as well as C20-C22 when used, as large as the board area and update rate permit. Larger values give more filtering, but values can be too large, causing "sampling errors" in which the ADC inputs lag behind input at S11 by a period



```
10     PRINT "VIN TEST PROGRAM FOR ADC3"
20     CA = 1024   'set for your own card address
30     PN = 0  'for Chan. A, =1  Chan. B, =2  Chan. C
40     POKE CA-3,128  'write to start conversion
50     N = PEEK (CA-PN)  'read input from ADC card
60     PRINT N    'print value on CRT
90     GOTO 30   'loop till stop program
```

Note: Program listing shown assumes UBEC with H-8 memory-mapped I/O. For other machines, change CA to your card address and change/add other statements per Part 6, fig. 11 and 12. For USIC, revise per Part 15 to include protocol subroutines, delete line 40, and add a start-conversion signal to change state each time through loop for sending out to ADC Card

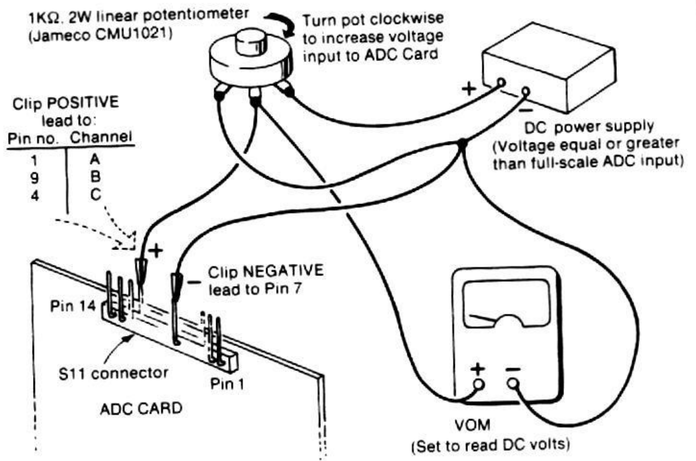

1KΩ, 2W linear potentiometer (Jameco CMU1021)
Turn pot clockwise to increase voltage input to ADC Card
DC power supply (Voltage equal or greater than full-scale ADC input)
Clip POSITIVE lead to:
Pin no. Channel
1 A
9 B
4 C
Clip NEGATIVE lead to Pin 7
Pin 14
S11 connector
Pin 1
ADC CARD
VOM (Set to read DC volts)

Fig. 10 VIN TEST PROGRAM

longer than the time between conversion updates. For maximum accuracy, keep the capacitors' values on the high side but not greater than the time between conversions (in seconds) divided by the product of five times the value (in ohms) of R37-R42.

For example, in fig. 6b I've assumed a real-time-loop update period of .06 seconds, as on the Sunset Valley, so the largest value recommended for C17 with R37 at 470Ω is .06 ÷ (5 x 470), or 26μF — I picked the nearest standard value of 22μF. In fig. 6c R37 is 10KΩ, so for the same update period C17 is decreased to 1μF. The table in fig. 9 lists recommended standard capacitors to fit the card.

BUILDING THE ADC CARD

Figure 9 includes the parts layout and parts list for the ADC Card. Ready-to-use circuit cards are available from JLC Enterprises, P. O. Box 88187, Grand Rapids, MI 49518. The ADC3 is double sided with plated-through holes and sells for $30, and there is a shipping and handling charge of $2.50 per order (Michigan residents add 4 percent sales tax).

As with the DAC, asterisks in the parts list mark parts to be substituted for the particular ADC circuit configurations you want. For R16-R42 you may want to substitute 1 percent precision resistors.

To assemble the ADC Card refer to fig. 9 and follow these steps:

☐ **Card test.**

☐ **J1, J2.** Omit J1 to use V1, and omit J2 to separate AGND from DGND.

☐ **R1-R42.** For parts marked by asterisks, be sure to install components for the configuration(s) you want. Options are no part (open circuit), fixed resistor, trim pot, or jumper (0Ω). Where you use a trim pot, its adjustment screw should face S11.

☐ **D1-D3[+].**

☐ **S1-S8[+].**

☐ **S9-S11.**

☐ **SW1[+].** Install so DIP switch contacts are "ON" when thrown toward S11.

☐ **C1,C2[+].**

☐ **C3-C6.**

☐ **C7[+].**

☐ **C8, C11-C13.**

☐ **C14-C16[+].**

☐ **C17-C19[+].** Check fig. 9 for values.

☐ **C20-C22[+].** Omit if you are substituting jumpers for resistors R34-R36.

☐ **C23-C25.**

☐ **U1-U8[+].**

Skip to "Cleanup and inspection" unless you're using alternate supply V1.

☐ **V1[+].** Install this regulator like the transistors on the Output Card, noting the pinout diagram inset in fig. 9.

☐ **C9[+].**

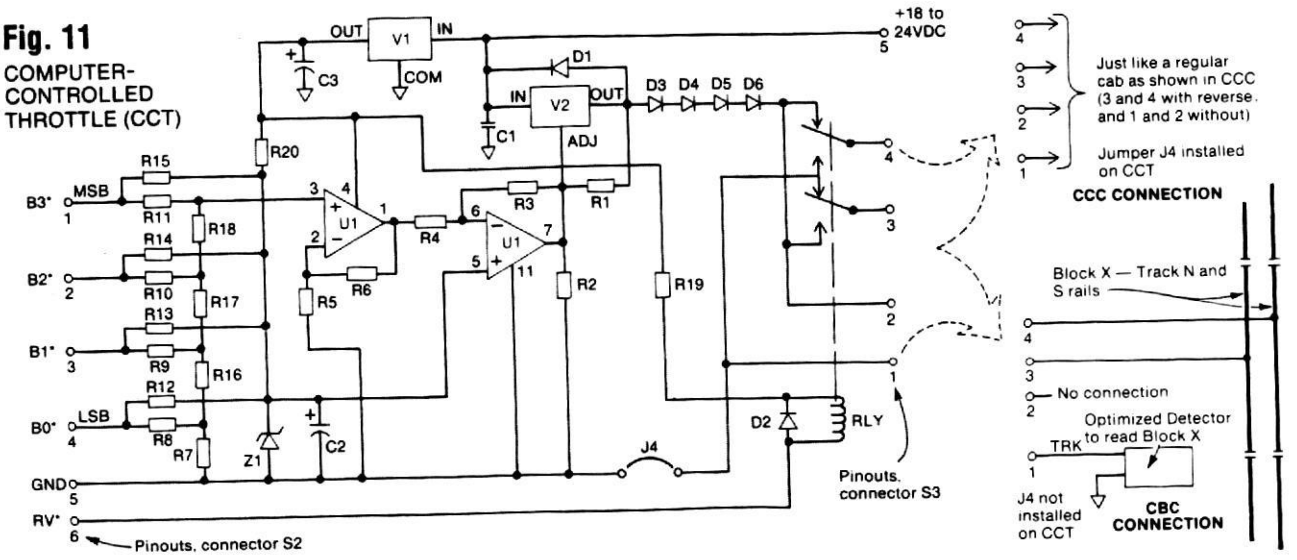
☐ **C10.**

☐ **Cleanup and inspection.**

☐ **IC power test.** Set SW1 to the desired card slot address, and plug the ADC Card into your IOMB. If using V1, connect a separate +8 to 12VDC input to pin 14 of S11, and the corresponding ground to pin 8. Turn on the IOMB supply, and the V1 supply if used, and use your VOM and the fig. 9 table to see that power reaches each IC. Correct any problems before proceeding.

☐ **Input voltage test.** Connect an external potentiometer as in fig. 10 to provide a variable input voltage to pin 1 of S11, the channel A ADC input. Enter and RUN the VIN Test Program in fig. 10 to

Fig. 11 COMPUTER-CONTROLLED THROTTLE (CCT)



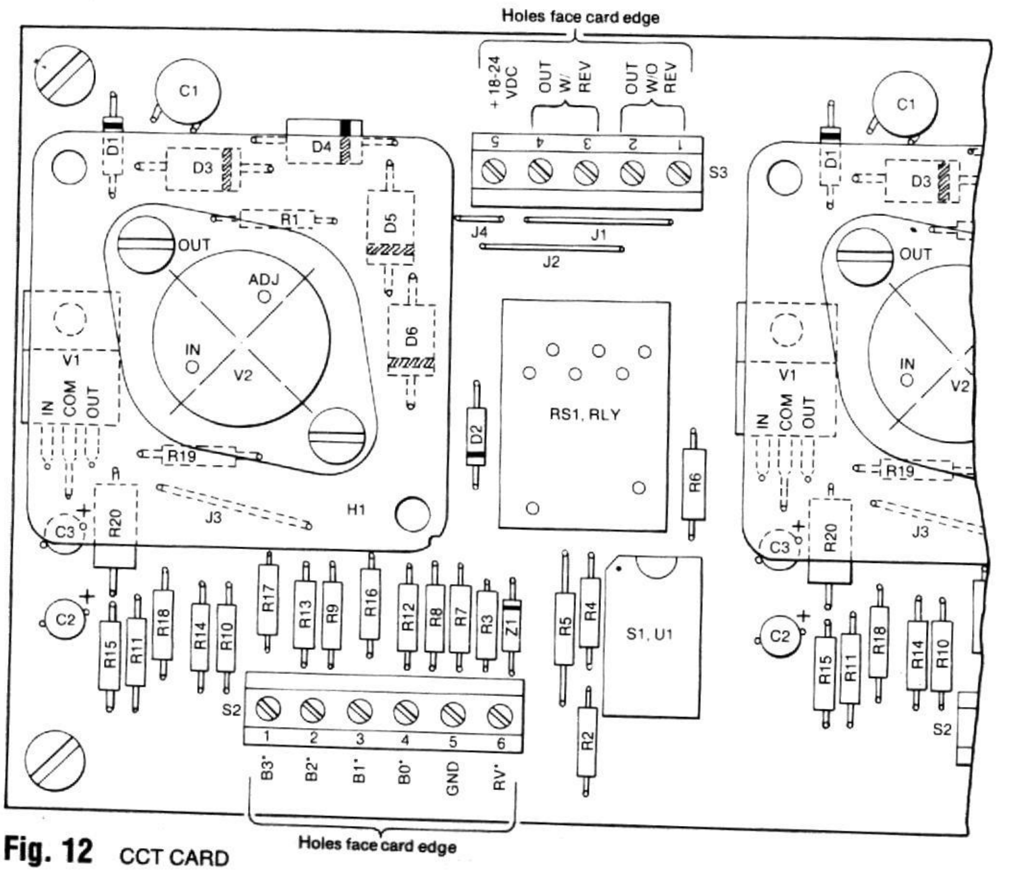
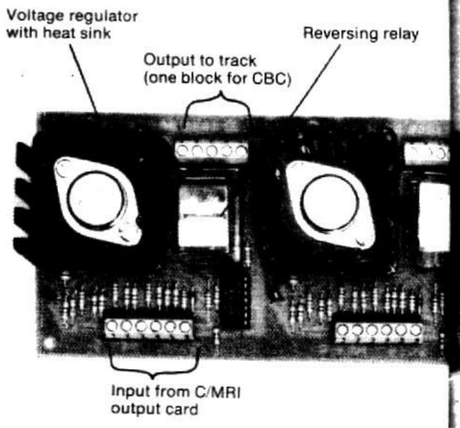
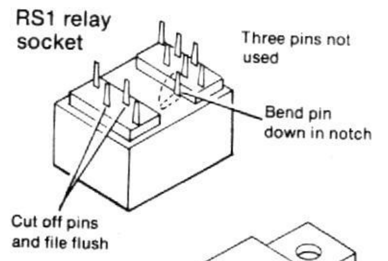
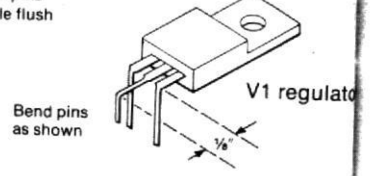


Fig. 12 CCT CARD


Voltage regulator with heat sink
Output to track (one block for CBC)
Reversing relay
Input from C/MRI output card





COMPUTER CONTROLLED THROTTLE (CCT) PARTS LIST (in order of assembly)

Qnty.	Symbol	Description
4	J1-J4	Jumpers, make from no. 20 uninsulated bus wire (Belden no. 8020, to carry track current). Omit J4 if using CCT as separate throttle per block for Computer Block Control (CBC) with Optimized Detector
1	R1	240Ω resistor [red-yellow-brown] except for V2 equal LM338K use 120Ω resistor [brown-red-brown]
1	R2	470Ω resistor [yellow-violet-brown]
3	R3-R5	10KΩ resistors [brown-black-orange]
1	R6	12KΩ resistor [brown-red-orange]
5	R7-R11	36KΩ resistors [orange-blue-orange]
4	R12-R15	3000Ω resistors [orange-black-red]
3	R16-R18	18KΩ resistors [brown-grey-orange]
1	R19	47Ω resistor [yellow-violet-black]
1	R20	160Ω, ½W resistor [brown-blue-brown]
2	D1,D2	1A, 100-PIV diodes (Digi-Key 1N4002)
4	D3-D6	3A, 100-PIV diodes (Digi-Key 1N5401) May replace 1 or 2 diodes with jumper(s) made from no. 20 uninsulated bus wire (Belden no. 8020, to carry track current), if such a low throttle-off output voltage isn't needed. For V2 equal LM338K substitute 6A diodes (Digi-Key 6A1)
1	Z1	6.8V, 1W zener diode (Jameco 1N4736)
1	C1	.1μF, 50V ceramic disk capacitor (Jameco DC 1/50)
2	C2,C3	1μF, 35V tantalum capacitors (Jameco TM1/35)
1	S1	14-pin DIP socket (Digi-Key C8914)
1	S2	6-position terminal block (Mouser ME153-2006)
1	S3	5-position terminal block (Mouser ME153-2005)
1	V1	1A, 15V positive regulator (Jameco 7815T)
1	V2	1.5A positive adjustable regulator (Jameco LM317K) or 3A positive adjustable regulator (Jameco LM350K) or 5A positive adjustable regulator (Jameco LM338K)
1	H1	25W heatsink forTO-3 case (Digi-Key HS103-1.25). For 5A regulator substitute larger off-card heatsink such as Digi-Key HS118
2	—	Hex 6-32 threaded ½"-long spacer (Digi-Key J178)
2	—	6-32 x ½"-long pan head machine screw (Digi-Key H160)
2	—	6-32 x ¼"-long pan head machine screw (Digi-Key H154)
1	RS1	Relay socket (Potter & Brumfield 27E128 or Mouser ME151-VP2135201)
1	RLY	2PDT, 12VDC relay (Potter & Brumfield R10E1Y2V185 or Mouser ME433-VP25ACAB12)
1	U1	LM324N quad low power operational amplifier (Jameco)
		****Optional item mounted external to card in track circuit****
1	CB	2A circuit breaker, (Radio Shack 270-1310)

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

Z1 options, see text

Zener diode		Approx. max. track voltage*
Part no.	Reference voltage	
1N4736	6.8V	12
1N4735	6.2V	10.8
1N4734	5.6V	9.6
1N4733	5.1V	8.6
1N4732	4.7V	7.8

*Based upon 3 diode drops D3-D6

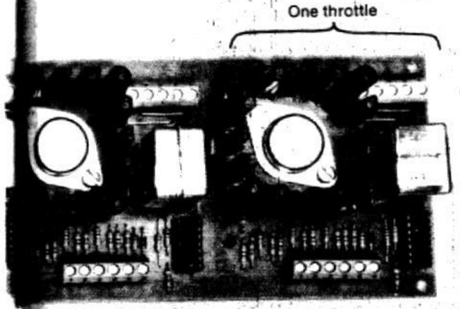
Fig. 13 CCT CARD CONTRUCTION

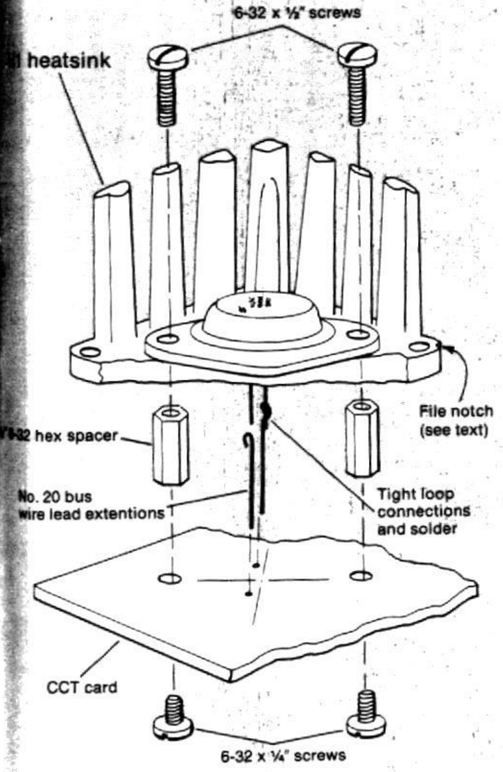
Voltage test table

INPUTS B0*-B3* SET AT	MINIMUM OUTPUT VOLTAGE TESTS		MAXIMUM OUTPUT VOLTAGE TESTS	
	OPEN CIRCUITED		GROUNDED	
+ meter lead on *	✓	Voltage reading	✓	Voltage reading
U1 pin 1		13.9		.71
U1 pin 2		6.3		.34
U1 pin 3		6.3		0.00
U1 pin 4		15.3		15.3
U1 pin 5		6.9		6.9
U1 pin 6		7.4		6.8
U1 pin 7		.94		12.9
Case V2		2.3		14.2
D3 cathode●		1.6		13.4
D4 cathode●		1.1		12.7
D5 cathode●		.60		12.0
D6 cathode●		.06		11.3

* – meter lead on S2 terminal 5
● banded end

Above readings taken with load of 50Ω, 10W resistor (Radio Shack 271-133) connected between S3 term 2 and S2 term 5





Decimal number in software

N	B3*	B2*	B1*	B0*	VOUT
0					0
1				●	.8
2			●		1.6
3			●	●	2.4
4		●			3.2
5		●		●	4.0
6		●	●		4.8
7		●	●	●	5.6
8	●				6.4
9	●			●	7.2
10	●		●		8.0
11	●		●	●	8.8
12	●	●			9.6
13	●	●		●	10.4
14	●	●	●		11.2
15	●	●	●	●	12.0

● Indicated line pulled low by software (or manually with clip lead to ground)

Input-output table

display the decimal equivalent, between 0 and 255, of the input voltage — the CRT readout should vary as you turn the pot. Once channel A passes, repeat the test for the other channels. That completes the ADC Card.

COMPUTER BLOCK CONTROL

You can use the DAC Card to control almost any throttle. Computer-controlled throttles can connect to your layout like any other throttle used for manual cab control, and they can be used for Computer Cab Control (CCC) by feeding the throttles' output through CRC Cards.

However, as alternative to CCC, you can use one computer-controlled throttle for each block on your layout: *C*omputer *B*lock *C*ontrol or CBC.

CBC's separate throttles for each block eliminate block power switching and give great control flexibility. You could run a train with a cab consisting simply of a pot and a switch to generate speed and direction input to be read by an ADC Card and a general-purpose Input Card, respectively. Or use a hexadecimal rotary switch (Part 13, fig. 6) as a speed control: its output can be read directly by an Input Card.

The software would keep track of which blocks are occupied by the train you are controlling, and simultaneously do the same for all other trains and cabs. With that data the software would calculate and output settings for each of the separate CBC block throttles. Wherever you ran a train over your layout, the train would respond to your cab through the CBC software. The CRC Cards limited CCC to seven cabs, but with CBC you can have up to N – 1 trains in simultaneous independent operation on a railroad with N blocks.

CBC may sound costly, but if the block throttles are simple and inexpensive, the overall system can be very cost effective. Figure 11 shows such a Computer-Controlled Throttle or CCT. It's a digital-input modification of a voltage-regulator throttle introduced by Joe Giannovario and improved by Don Hansen in MR's January 1981 Symposium on Electronics. In fact, the CCT can be used in any computerized-throttle system, not just CBC.

The digital modification is the resistor-ladder network in front of OPAMP U1, which serves as a four-bit DAC. A second stage of U1 is used to make the logic inputs active low, so the throttle is off if the inputs are not connected. The C/MRI can drive the CCT directly with five lines from an Output Card. Lines B0*-B3* control train speed, and RV* controls direction by activating reversing relay RLY.

The 4 speed-control lines give 16 discrete levels of motor voltage. Regulator V1 provides a constant 15VDC for the OPAMP, and zener diode Z1 and R20 set the 6.8V reference for the ladder network.

Regulator V2 controls the track voltage. Its output can only be controlled down to about 2V, but the drop across series diodes D3-D6 cancels the effect of the minimum VOUT, and drops minimum track voltage near to zero.

Depending upon how much voltage it takes to make your engines creep, you may get by with only one, two, or three of the diodes, replacing those omitted with jumpers. Each diode gives about a .7V drop in track voltage. If you use Optimized Detectors, they already give you a diode drop, which is one diode you can leave off the CCT.

The full-scale track voltage can be adjusted with software, by selecting the voltage rating of zener diode Z1, or a combination of both. The fig. 13 table gives several choices for Z1 ratings; the change in the full-scale voltage will be twice the change in the rated zener voltage.

Resistor R19 gives the correct voltage across the relay coil when RV* is pulled low by the C/MRI. Diodes D1 and D2 protect V1 and the Output Card, respectively.

Figure 12 shows a CCT Card which includes four identical throttles. Power is 18-24VDC applied to terminal 5 of S3, with ground on terminal 5 of S2. For CBC you need just one set of power connections per card, and multiple cards may be powered in parallel. You may cut the CCT Card along the dashed lines on the trace side to make four individual CCTs to install close to the blocks they control. If you use Optimized Detectors in a CBC system, omit jumper J4 on each throttle, and connect the Detector as in fig. 11.

Figure 11 also shows how the CCT may be used as a conventional (non-CBC) throttle, using the four output terminals on each S3. Terminals 1 and 2 are before the reversing relay while 3 and 4 are after it, for CCC use as shown in Part 9, fig. 4. Jumper J4 is installed, and you use a separate 18-24VDC supply for each throttle. To isolate these, cut away the two outside circuit traces between each throttle at the dashed arrows on the card.

The throttle power supply must be filtered but need not be regulated. It can be built following Part 8, fig. 5, with V1, C2, and C3 omitted and T1 selected to have a 14-18V secondary. The transformer and bridge rectifier B1, must have current capacity greater than the expected maximum simultaneous current requirement from each group of throttles. C1's value should be increased or other capacitors added in parallel so the total capacitance equals about 7500μF per amp of supply capacity. The bridge and the filter capacitors should be rated for at least 100V.

BUILDING THE CCT CARD

Figure 12 includes the parts layout for the CCT Card; the parts list is in fig. 13. Ready-to-use circuit cards are available from JLC Enterprises, P. O. Box 88187, Grand Rapids, MI 49518. The CCT4 is $24 and there is a shipping and handling charge of $2.50 per order (Michigan residents must add 4 percent sales tax).

You can build up to four CCTs on each card, but as all are identical I've repeated the parts nomenclature from throttle to throttle. The parts list quantities are for a single throttle.

I've listed three V2 regulators for different maximum currents. This is a function of the maximum "steady-state" current you expect to need for your heaviest train. The higher the regulator's current rating and the higher the input voltage to the card, the more power needs to be dissipated by the regulator when you have a short circuit in a block. Thus it's best to use the lowest-current regulator that you can get by with. The regulators will handle surge currents greater than their continuous ratings, and if overloaded they automatically go into thermal shutdown without damage. When the short is removed, V2 will quickly cool and automatically resume operation.

If you need the 3A regulator, you may want to add metal to the heat sink, and it's best to mount the 5A regulator off the card on a large heatsink like Digi-Key HS118. Or you can use a circuit breaker (Radio Shack 270-1310) in one of the track feeders, so the breaker will open in case of a prolonged short.

To assemble a CCT Card refer to figs. 12 and 13, and follow these steps:

☐ **Card test.**

☐ **J1-J4.** Omit J4 if CCT is for CBC with Optimized Detectors.

☐ **R1-R20.**

☐ **D1, D2[+].**

☐ **D3-D6[+].** Install 1/16" away from the card, like Optimized Detector diodes D1-D2, Part 7 page 89. If you omit any diodes, use no. 20 or larger jumpers in their place.

☐ **Z1[+].**

☐ **C1.** Once soldered, bend flat against the card as in the parts layout.

☐ **C2, C3[+].**

☐ **S1-S3[+].**

☐ **V1[+].** Bend leads as shown in the fig. 13 inset; install flat against the card.

☐ **V2[+].** Mount V2 to the heatsink as in fig. 13. Then make and solder tight hook joints between two 2" lengths of no. 20 jumper wire and the V2 leads. V2's leads are only slightly off center: to avoid blowing the regulator by installing it backwards, file a notch in the heatsink as shown in fig. 13.

Feed the jumpers through the IN and ADJ holes in the card, making sure the notch filed in the heat sink is near D2 and the leads are in the correct holes, and secure the assembly with 6-32 x 1/4" screws. After soldering, check that the jumpers are tight and not touching, and use your VOM to make sure the leads are not shorted to the case of V2.

☐ **RS1, RLY[+].** Modify the pins of relay socket RS1 as shown in the fig. 13 inset drawing, and insert relay RLY into the socket. If you use Mouser parts, snap the hold-down clip into the socket and position it over the top of the relay. For the Potter & Brumfield socket the hold-down is available separately as part no. 20C49. Hold RS1 (with RLY in place) firmly against the card as you solder, so the pins extend as far as possible on the trace side.

☐ **U1[+].**

☐ **Power-up test.** Leaving the track terminals open, connect +18 to 24VDC power to the card; V1 and V2 should remain cool. Touch the − lead of your VOM to U1 pin 11; you should read about +15V with the + lead on U1 pin 4, and +6.8V with the + lead on U1 pin 5.

☐ **Relay test.** Use a clip lead to temporarily ground the RV* terminal of S2. The relay should throw, and the voltage drop across the relay coil should be close to 12VDC. If off more than a few volts, vary R19 until you get the correct value.

☐ **Operational voltage test.** Use clip leads to attach a 50Ω, 10W resistor between S2 terminal 5 and S3 terminal 2. Attach the − lead of your VOM to S2 terminal 5, and touch the + lead to the test points in the fig. 13 test table. Check the boxes next in the "open circuited" column as you pass each test. A reading off by more than 1V indicates a soldering problem, an incorrect resistor, or a faulty U1. Use clip leads to ground the four B0*-B3* inputs and recheck the previous test points, comparing voltages read to the table's "grounded" column. Again, a reading off by more than 1V indicates a problem.

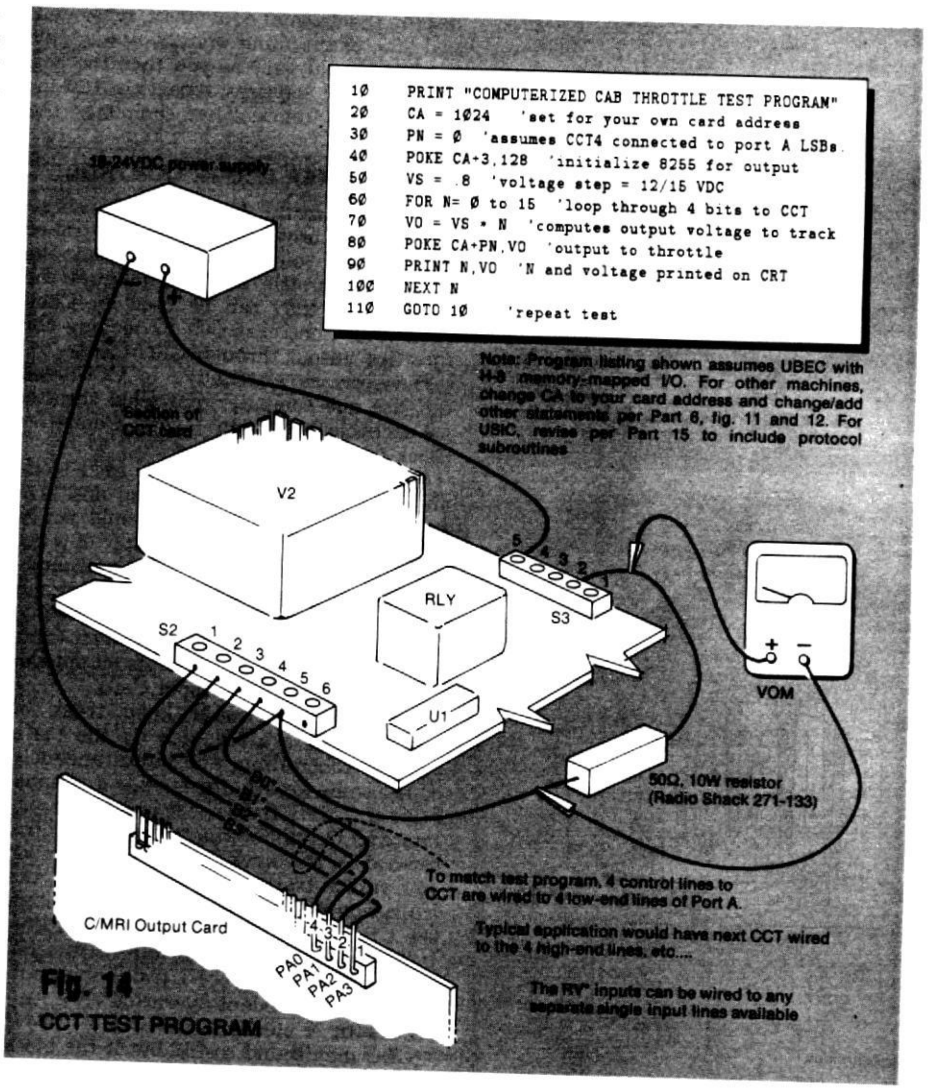
☐ **Variable voltage test.** Use clip leads to temporarily ground various combinations of the B0* to B3* terminals of S2. Grounding all four terminals should give a maximum output near 12V. Each other grounding combination should give voltage proportional to the decimal equivalent of the binary code set on B0*-B3*, according to the fig. 13 input-output table.

☐ **Train control test.** Connect S3 terminals 3 and 4 to an isolated section of track; if J4 is not installed, temporarily connect S2 terminal 5 to S3 terminal 1. Put a locomotive on the track and repeat the previous clip-lead tests to check the locomotive's operation. If it runs sluggishly, measure the voltage across the power input terminal screws to make sure you are not loading down your power source. For proper operation of the CCT, voltage into the CCT Card must be at least 18V. Grounding terminal RV* should throw the reverse relay and reverse the locomotive.

☐ **Software test.** Wire the S2 terminal to a C/MRI Output Card as in fig. 14, and enter and RUN the test program. The track voltage should vary as in the fig. 13 input-output table each time through the real-time loop. You've already tested the CCT Card, so any problems here are in your software, card addresses, or Output Card to CCT Card wiring.

That completes the CCT Card, and you now have the capability to put almost everything on your railroad under software control. I hope you've learned from the C/MRI series and that you have found building your own interface both enjoyable and rewarding.

If you've followed along but haven't yet jumped in to build your first card, now's the time. Working one step at a time, the steps are meaningful and easy to understand, and before you realize it you'll have a working system. The Sunset Valley's computer has made its operation easier and more fun, and I'm sure that a C/MRI can do the same for your layout. Happy railroading! ♁



```
10    PRINT "COMPUTERIZED CAB THROTTLE TEST PROGRAM"
20    CA = 1Ø24    'set for your own card address
3Ø    PN = Ø  'assumes CCT4 connected to port A LSBs.
4Ø    POKE CA+3,128  'initialize 8255 for output
5Ø    VS = .8  'voltage step = 12/15 VDC
6Ø    FOR N= Ø to 15  'loop through 4 bits to CCT
7Ø    VO = VS * N  'computes output voltage to track
8Ø    POKE CA+PN,VO  'output to throttle
9Ø    PRINT N,VO  'N and voltage printed on CRT
1ØØ   NEXT N
11Ø   GOTO 1Ø     'repeat test
```

Note: Program listing shown assumes UBEC with H-8 memory-mapped I/O. For other machines, change CA to your card address and change/add other statements per Part 6, fig. 11 and 12. For USIC, revise per Part 15 to include protocol subroutines

To match test program, 4 control lines to CCT are wired to 4 low-end lines of Port A.

Typical application would have next CCT wired to the 4 high-end lines, etc....

The RV* inputs can be wired to any separate single input lines available

Fig. 14
CCT TEST PROGRAM

Corrections. Part 14 fig. 4 parts list: C12-C13 should be 22pf, C14-C15 should be 2.2µf, and C18 part number should be P6018. Part 15 fig. 1: the decimal value of "N," for CRC Not enabled, is 78. Part 15 fig. 6: USIC "A's" CT(3) value is 9.