Signaling Made Easier Example Software and Program Documentation

By Bruce Chubb

This example uses the track plan shown in Fig. 1, which is the same as used in Part 3 of the **Signaling Made Easier** series in the March 2004 issue of **Model Railroader**. The major difference with the material presented herewith is that, due to page-space limitations, the MR article illustrated only portions of the program listing where here we are able to show the complete listing. Additionally, with increased space available, I have enhanced the example program documentation to help clarify many of its pertinent features.

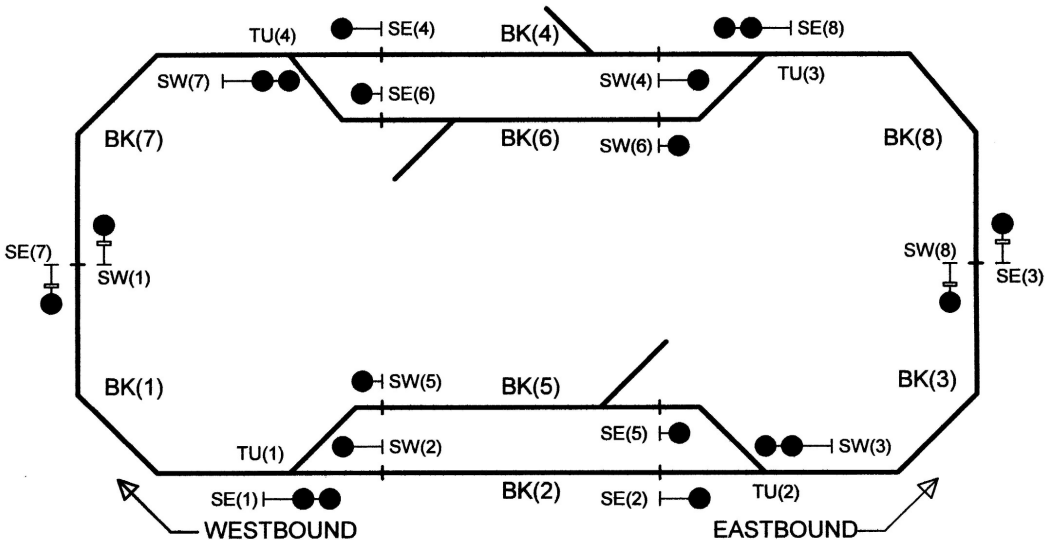


Fig. 1. Three color signaling with dual-head signals leading into sidings

This example employs dual-headed signals at the facing point end of each of the passing siding turnout. The upper head governs the main route and the lower-head governs the divergent route, i.e. entering the passing siding. Although this example assumes using 3-lead type searchlight signal LEDs yielding green, yellow and red capability, very little software changes are involved whether using 2-lead or 3-lead searchlight LEDs, or for that matter signals using a separate LED per color, PRR position light, B&O color position light or even semaphores.

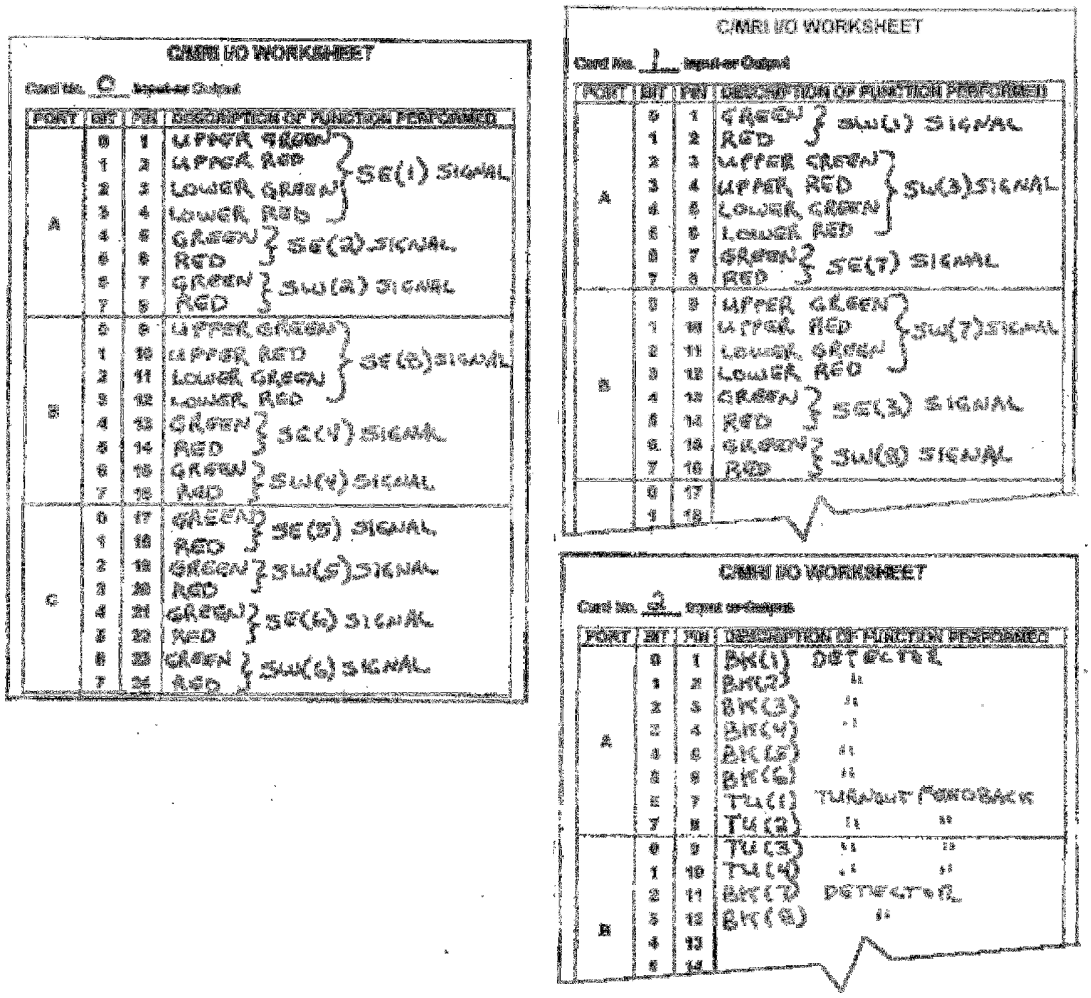
Additionally, the example assumes using a single SMINI node. However very little change is required if a SUSIC-based Maxi-node was desired. Lastly, the example assumes that Command Control, such as DCC, is used for train control. Making this assumption lets us focus our full attention on signaling.

The corresponding I/O worksheets, which are equal to those illustrated in the March 2004 issue of MR, are repeated here for completeness as Table 1.

As published in MR, this example does not include separate detectors per powered turnout (frequently called OS sections). Their inclusion is essential for true CTC-type operation and to protect against a powered track switch from being thrown when the turnout is occupied. However

leaving them out for this example saves the cost of four DCCOD occupancy detectors, about $45 for Do-It Yourselfers.

Table 1. I/O worksheets for searchlight signal example using SMINI



C/MRI I/O WORKSHEET

Card No. 0 Input or Output

PORT	BIT	PIN	DESCRIPTION OF FUNCTION PERFORMED
A	0	1	UPPER GREEN } SE(1) SIGNAL
	1	2	UPPER RED
	2	3	LOWER GREEN
	3	4	LOWER RED
	4	5	GREEN } SE(2) SIGNAL
	5	6	RED
	6	7	GREEN } SW(A) SIGNAL
	7	8	RED
B	0	9	UPPER GREEN } SE(B) SIGNAL
	1	10	UPPER RED
	2	11	LOWER GREEN
	3	12	LOWER RED
	4	13	GREEN } SE(4) SIGNAL
	5	14	RED
	6	15	GREEN } SW(4) SIGNAL
	7	16	RED
C	0	17	GREEN } SE(5) SIGNAL
	1	18	RED
	2	19	GREEN } SW(5) SIGNAL
	3	20	RED
	4	21	GREEN } SE(6) SIGNAL
	5	22	RED
	6	23	GREEN } SW(6) SIGNAL
	7	24	RED

C/MRI I/O WORKSHEET

Card No. 1 Input or Output

PORT	BIT	PIN	DESCRIPTION OF FUNCTION PERFORMED
A	0	1	GREEN } SW(1) SIGNAL
	1	2	RED
	2	3	UPPER GREEN } SW(3) SIGNAL
	3	4	UPPER RED
	4	5	LOWER GREEN
	5	6	LOWER RED
	6	7	GREEN } SE(7) SIGNAL
	7	8	RED
B	0	9	UPPER GREEN } SW(7) SIGNAL
	1	10	UPPER RED
	2	11	LOWER GREEN
	3	12	LOWER RED
	4	13	GREEN } SE(3) SIGNAL
	5	14	RED
	6	15	GREEN } SW(8) SIGNAL
	7	16	RED
	0	17	
	1	18	

C/MRI I/O WORKSHEET

Card No. 2 Input or Output

PORT	BIT	PIN	DESCRIPTION OF FUNCTION PERFORMED
A	0	1	BK(1) DETECTOR
	1	2	BK(2) "
	2	3	BK(3) "
	3	4	BK(4) "
	4	5	BK(5) "
	5	6	BK(6) "
	6	7	TU(1) TURNOUT THROW/BACK
	7	8	TU(2) " "
B	0	9	TU(3) " "
	1	10	TU(4) " "
	2	11	BK(7) DETECTOR
	3	12	BK(8) "
	4	13	
	5	14	

The wiring for all three of the I/O cards built into the SMINI is very straightforward. For example, the output of each of the eight occupancy detectors simply connects to their corresponding input pin on the SMINI. Four additional inputs are derived directly from each of the passing siding turnouts. This can be something as simple as using a contact on the hand-throw turnout or in this example by using one of the DPDT contact sets built into a Tortoise switch motor. Simply connect the center pole to logic ground and the wire the corresponding contact that is closed when the turnout is aligned for the diverging route, i.e. the passing siding, to the designated input card pin.

Because this example assumes that the passing siding turnouts are controlled by a local tower or station operator, each of the control toggles can be wired directly to their corresponding switchmotor, i.e. not pass through the computer. Alternatively, we could assume that the railroad was being operated using “poor mans” CTC, where the train crew is required to align the turnout. In either case, it is appropriate to wire the toggles directly to the switch motor. Alternatively, the

turnouts could be hand-throw with a built-in contact used to connect the corresponding SMINI input pin to ground when the turnout is aligned for the divergent route, i.e. the siding.

However, if for some reason you desired to have the switch motor control pass through the computer software, e.g. to prevent changing turnout alignment when occupied, then it is a straightforward process to connect the output of each toggle to an SMINI input line and then use SMINI output lines to drive the switch motor.

The 3-lead bi-color signal LEDs are connected directly to SMINI outputs as shown in Fig. 2

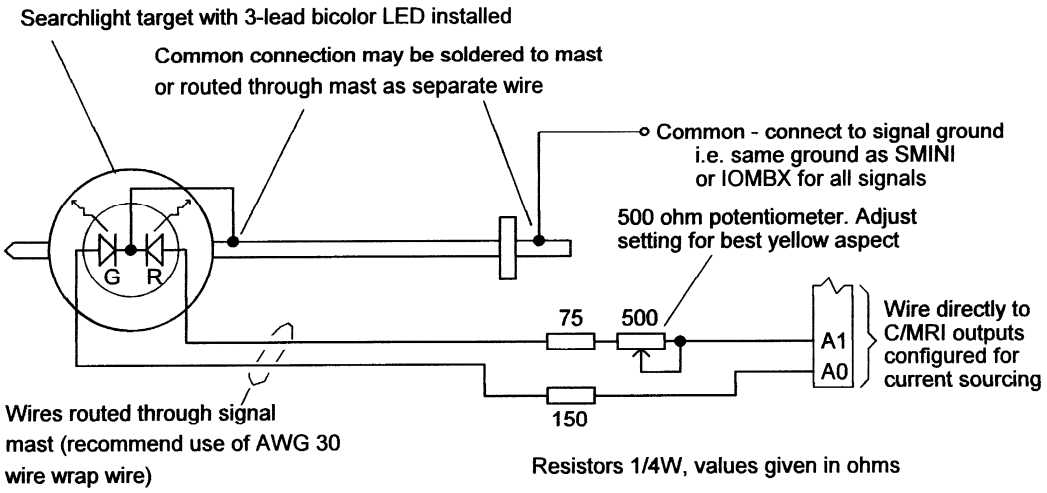


Fig. 2. Connecting 3-lead Searchlight Signal LEDs to C/MRI Outputs

The center lead is connected to signal ground, either feeding a wire through the mast or using the mast itself as the conductor. The two outside leads from the LED are fed through the mast for connecting to SMINI outputs via resistor/potentiometer board. Each bi-color LED requires 2 adjacent C/MRI output lines configured for alternate current sourcing. The software simple needs to activate one output to achieve a green aspect the other a red aspect and activating both outputs generates a yellow. Resistors are required to limit the LED current. Adjusting the potentiometer varies the brightness of the red which in turn controls the quality of the yellow aspect.

The resistors and potentiometer can be mounted on a small piece of perforated printed circuit board such as that available from Radio Shack. Alternatively, the parts can be mounted on a JLC provided RSST circuit board as covered elsewhere on this Website. Alternatively, you can make use of a special JLC provided circuit board, the Remote Searchlight Signal Driver (RSSD), which includes the necessary resistors and the potentiometer and transistors enabling the use of standard current sinking outputs to drive the 3-lead bi-color LEDs.

It should be noted that the I/O tables do not include any inputs for the industrial spur turnouts, i.e. to indicate when a spur turnout is either unlocked or not aligned for the main track. Using such inputs, the software can set the signals to red leading into the blocks where spurs are either unlocked or when a spur turnout is left in the aligned-for-spur condition. Instead, this example assumes that the spurs are protected by incorporating a shunting resistor which results in the block containing a spur to show up as occupied any time that the spur turnout is aligned for the spur which in turn causes the signals leading into the block to display a stop indication.

The example program listing, written in Microsoft QuickBASIC Version 4.5, is shown as follows, i.e. in Fig. 3.

```
REM**Print out title information
    PRINT "THREE COLOR SIGNALING FOR BASIC LOOP EXAMPLE"
    PRINT "PROGRAM ASSUMES COMBINATION OF ABS AND LOCAL INTERLOCKING"
    PRINT "PROGRAM ASSUMES BASIC TUMBLEDOWN FEATURE OF APB SIGNALING"
    PRINT "PROGRAM ASSUMES 3-LEAD SEARCHLIGHT LEDs"

REM**Define variable types and array sizes
    DEFINT A-Z              'Define all variables as integer
    DIM SHARED OB(60), IB(60), CT(15), TB(60)
    DIM BK(8), SE(8), SW(8), TU(4), DOT(8)

REM**Define constants for packing/unpacking I/O bytes
    B0 = 1: B1 = 2: B2 = 4: B3 = 8: B4 = 16: B5 = 32: B6 = 64: B7 = 128
    W1 = 1: W2 = 3: W3 = 7: W4 = 15: W5 = 31: W6 = 63: W7 = 127

REM**Define general constants
       CLR = 0      'Clear
       OCC = 1      'Occupied
       NDT = 0      'No direction-of-traffic
       WBD = 1      'Westbound
       EBD = 2      'Eastbound (east is even)
       TUN = 0      'Turnout normal alignment
       TUR = 1      'Turnout reverse alignment

 REM**Define signal aspect constants
     DRK = 0      'Dark      00
     GRN = 1      'Green     01
     RED = 2      'Red       10
     YEL = 3      'Yellow    11
     GRNRED = 9   'Green-over-red   1001
     REDRED = 10  'Red-over-red     1010
     YELRED = 11  'Yellow-over-red  1011
     REDYEL = 14  'Red-over-yellow  1110

 REM**Define interface constants**
     UA = 0                'USIC node address
     COMPORT = 1           'PC communications port = 1, 2, 3 or 4
     BAUD100 = 96          'Baud rate of 9600 divided by 100
     DL = 0                'USIC transmission delay
     NDP$ = "M"            'Node definition parameter
     NS = 0                'Number of 2-lead searchlight LEDs
     NI = 3                'Number of input ports
     NO = 6                'Number of output ports
     MAXTRIES = 10000      'Maximum read tries before abort inputs

 REM**Initialize SMINI
     GOSUB INIT            'Invoke initialization subroutine

 REM**BEGIN REAL-TIME LOOP
BRTL:

 REM**Read and unpack input bytes to separate out device inputs
     GOSUB INPUTS
     BK(1) = IB(1) AND W1         'SMINI CARD 2 PORT A
     BK(2) = IB(1) \ B1 AND W1
     BK(3) = IB(1) \ B2 AND W1
     BK(4) = IB(1) \ B3 AND W1
```

```
     BK(5) = IB(1) \ B4 AND W1
     BK(6) = IB(1) \ B5 AND W1
     TU(1) = IB(1) \ B6 AND W1
     TU(2) = IB(1) \ B7 AND W1

     TU(3) = IB(2) AND W1         'SMINI CARD 2 PORT B
     TU(4) = IB(2) \ B1 AND W1
     BK(7) = IB(2) \ B2 AND W1
     BK(8) = IB(2) \ B3 AND W1

 REM**Calculate aspects for signals entering sidings & adjacent main**
     SE(1) = REDRED           'Signal SE(1)
     IF TU(1) = TUN THEN
        IF BK(2) = OCC THEN GOTO SE8
        IF SE(2) <> RED THEN SE(1) = GRNRED ELSE SE(1) = YELRED
     ELSE
        IF BK(5) = CLR THEN SE(1) = REDYEL
     END IF

SE8: SE(8) = REDRED           'Signal SE(8)
     IF TU(3) = TUN THEN
        IF BK(4) = OCC THEN GOTO SW3
        IF SE(4) <> RED THEN SE(8) = GRNRED ELSE SE(8) = YELRED
     ELSE
        IF BK(6) = CLR THEN SE(8) = REDYEL
     END IF

SW3: SW(3) = REDRED           'Signal SW(3)
     IF TU(2) = TUN THEN
        IF BK(2) = OCC THEN GOTO SW7
        IF SW(2) <> RED THEN SW(3) = GRNRED ELSE SW(3) = YELRED
     ELSE
        IF BK(5) = CLR THEN SW(3) = REDYEL
     END IF

SW7: SW(7) = REDRED           'Signal SW(7)
     IF TU(1) = TUN THEN
        IF BK(2) = OCC THEN GOTO SW2
        IF SW(4) <> RED THEN SW(7) = GRNRED ELSE SW(7) = YELRED
     ELSE
        IF BK(5) = CLR THEN SE(1) = REDYEL
     END IF

 REM**Calculate aspects for signals leaving siding (and adjacent main)
SW2: SW(2) = RED: SW(5) = RED          'Signals SW(2) and SW(5)
     IF BK(1) = OCC THEN GOTO SW4
     IF TU(1) = TUN THEN
        IF SW(1) <> RED THEN SW(2) = GRN ELSE SW(2) = YEL
     ELSE
        IF SW(1) <> RED THEN SW(5) = GRN ELSE SW(5) = YEL
     END IF

SW4: SW(4) = RED: SW(6) = RED          'Signals SW(4) and SW(6)
     IF BK(8) = OCC THEN GOTO SE2
     IF TU(3) = TUN THEN
        IF SW(8) <> RED THEN SW(4) = GRN ELSE SW(4) = YEL
     ELSE
        IF SW(8) <> RED THEN SW(6) = GRN ELSE SW(6) = YEL
     END IF

SE2: SE(2) = RED: SE(5) = RED          'Signals SE(2) and SE(5)
     IF BK(3) = OCC THEN GOTO SE4
     IF TU(2) = TUN THEN
```

```
        IF SE(3) <> RED THEN SE(2) = GRN ELSE SE(2) = YEL
     ELSE
        IF SE(3) <> RED THEN SE(5) = GRN ELSE EW(5) = YEL
     END IF

SE4: SE(4) = RED: SE(6) = RED          'Signals SE(4) and SE(6)
     IF BK(7) = OCC THEN GOTO SE3
     IF TU(4) = TUN THEN
        IF SE(7) <> RED THEN SE(4) = GRN ELSE SE(4) = YEL
     ELSE
        IF SE(7) <> RED THEN SE(6) = GRN ELSE SE(6) = YEL
     END IF

 REM**Calculate aspect for intermediate block signals
SE3: SE(3) = RED                       'Signal SE(3)
     IF BK(8) = OCC THEN GOTO SE7
     IF SE(8) <> REDRED THEN SE(3) = GRN ELSE SE(3) = YEL

SE7: SE(7) = RED                       'Signal SE(7)
     IF BK(1) = OCC THEN GOTO SW1
     IF SE(1) <> REDRED THEN SE(7) = GRN ELSE SE(7) = YEL

SW1: SW(1) = RED                       'Signal SW(1)
     IF BK(7) = OCC THEN GOTO SW8
     IF SW(7) <> REDRED THEN SW(1) = GRN ELSE SW(1) = YEL

SW8: SW(8) = RED                       'Signal SW(8)
     IF BK(3) = OCC THEN GOTO DOT
     IF SW(3) <> REDRED THEN SW(8) = GRN ELSE SW(8) = YEL

 REM**When occupied, set direction-of-traffic
DOT: IF BK(1) = OCC AND DOT(7) <> EBD THEN DOT(1) = WBD
     IF BK(3) = OCC AND DOT(8) <> WBD THEN DOT(3) = EBD
     IF BK(7) = OCC AND DOT(1) <> WBD THEN DOT(7) = EBD
     IF BK(8) = OCC AND DOT(3) <> EBD THEN DOT(8) = WBD

 REM**Clear direction-of-traffic when single track becomes clear
     IF BK(1) = CLR AND BK(7) = CLR THEN DOT(1) = NDT: DOT(7) = NDT
     IF BK(3) = CLR AND BK(8) = CLR THEN DOT(3) = NDT: DOT(8) = NDT

 REM**Set head block signals at red if train approaching on single track
     IF DOT(1) = WBD THEN SE(4) = RED: SE(6) = RED
     IF DOT(3) = EBD THEN SW(4) = RED: SW(6) = RED
     IF DOT(7) = EBD THEN SW(2) = RED: SW(5) = RED
     IF DOT(8) = WBD THEN SE(2) = RED: SE(5) = RED

 REM**Implement approach lighting (delete code if feature not desired)
     FOR I = 1 TO 8
         IF BK(I) = CLR THEN SE(I) = DRK: SW(I) = DRK
     NEXT I

 REM**Pack output bytes and send them to railroad
     OB(1) = SE(1)              'Card 0 port A
     OB(1) = SE(2) * B4 OR OB
     OB(1) = SW(2) * B6 OR OB
     OB(1) = OB(1) XOR 255

     OB(2) = SE(8)              'Card 0 port B
     OB(2) = SE(4) * B4 OR OB
     OB(2) = SW(4) * B6 OR OB
     OB(2) = OB(2) XOR 255

     OB(3) = SE(5)              'Card 0 port C
```

```
    OB(3) = SW(5) * B2 OR OB
    OB(3) = SE(6) * B4 OR OB
    OB(3) = SW(6) * B6 OR OB
    OB(3) = OB(3) XOR 255
    OB(4) = SW(1)              'Card 2 port A
    OB(4) = SW(3) * B2 OR OB
    OB(4) = SE(7) * B6 OR OB
    OB(4) = OB(4) XOR 255

    OB(5) = SW(7)              'Card 2 port B
    OB(5) = SE(3) * B4 OR OB
    OB(5) = SW(8) * B6 OR OB
    OB(5) = OB(5) XOR 255

    OB(6) = 0                  'Card 2 port C is spare
    GOSUB OUTPUTS

 REM**Return to beginning of real-time loop**
    GOTO BRTL

 [Insert a copy of the GOSUB version of the Standard Serial Protocol
Subroutines at this location]
```

Fig. 3. Program example for 3-color signaling using SMINI

Removing the page space limitation that exists with magazine articles, it is possible here to provide a more in-depth description of the program. For example, statement lines beginning with REM, standing for Remarks, are inserted for the benefit of the reader and have no bearing on the execution of the program. Statements where a branch-to is required, or were we need a specific reference point for program documentation, I have included a meaningful alphabetic label such as BRTL: for beginning real-time loop, and SE8: for the statement location where the Signal SE(8) aspect calculations begin and DOT: where direction-of-traffic calculations commence. The colon after each label simply tells the QuickBASIC software that the preceding characters are a label.

Initializing. As with every C/MRI application, the program starts with an initialization sequence. The B0 through B7 and W1 through W7 and their use in the unpacking and packing operations are standard for every interface application program, as are the general constants that follow. The signal aspect constants shown reflect standard single- and dual-head 3-lead searchlight signal LED. These stay identical when using 2-lead bicolor LEDs but vary somewhat when using color light signals.

Initializing the SMINI is identical to most other applications by defining the node address via UA = 0, the PC COM port and baud rate. The NDP$ = “M” statement defines the node as an SMINI and with an SMINI the NS variable is set equal to the number of 2-lead searchlight signals which in this case is zero, and NI is set equal to the number of input ports, 3 for an SMINI, and NO the number of output ports, which is 6 for the SMINI. Once all the SMINI initialization constants are defined, a simple GOSUB INIT is all that is required to complete the node initialization.

Unpacking Inputs. The first operation within the real-time loop is to read the railroad inputs via the GOSUB INPUTS statement followed by unpacking the various railroad devices from their respective input bytes. An entire chapter is the V3.0 User’s Manual is devoted to the explaination of unpacking and packing. In summary however, unpacking inputs takes one statement per railroad device, and the format is always identical: device = IB(x)\By AND Wz where you substitute the byte number for x, the starting bit location within the byte for y, and how many

input lines wide the device is for z. For example, the Table 1 I/O worksheet shows that the occupancy detector input for block 1, namely BK(1), is located within IB(1) at bit position 0 and is 1 line wide. The corresponding unpacking statement is BK(1) = IB(1) \ B0 AND W1. However, because B0 has the value 1, and dividing any number by 1 equates to the number itself, we can shorthand the statement to simply read as BK(1) = IB(1) AND W1.

It is vitally important when writing unpacking statements to always use the backslash (\) which signifies an integer divide, in comparison to a forward slash (/) signifying a conventional divide. If you need some practice unpacking, look at the I/O worksheet and try writing the unpacking statements for a few of the other devices. When you have finished, compare them to those in the Fig. 3 program. For further insight, consult Chapter 8 of the V3.0 C/MRI User's Manual.

Calculating Basic Signal Aspects. Within the real-time loop, there is a close similarity between all of the signal logic statements. The main difference is where the signal logic is expanded to cover double-head signals. Single-head signals are initialized to red and double-headed signals to red-over-red. The program then checks input conditions such as block occupancy and turnout alignment and where it is safe to do so, calculates a less restrictive aspect.

To illustrate how this can be accomplished, let's take a look at the Fig. 1 track diagram and examine the logic statements used to calculate the appropriate aspect for double-headed signal SE(1). The first step is to initialize the signal to REDRED, its most restrictive aspect. Then, because there are two possible routes, i.e. remaining on the main-track route or diverging into the siding, the program needs to branch based upon the alignment of Turnout TU(1) to check out the appropriate route. For example if TU(1) is aligned for the main track, i.e. TU(1) = TUN, the program checks the occupancy status of BK(2). If occupied it retains SE(1) at REDRED by jumping directly to calculate the aspect for the next signal of interest, namely SE(8). For the condition when BK(2) is clear, i.e. not occupied, the logic looks at the signal in advance of SE(1), in this case SE(2). If SE(2) is not equal to RED (where the < > notation in BASIC reads as Not Equal), then the logic sets SE(1) to GRNRED, i.e. proceed (on main track) the next two blocks are clear while for the condition where SE(2) equals RED the logic sets SE(1) = YELRED, i.e. proceed on the main track approaching next signal prepared to stop.

Although this example does not do so, we could follow exactly the same logical steps to check out the siding route for the case when TU(1) is aligned for the siding,. That is, if BK(5) = OCC then retain the REDRED aspect by jumping directly to calculate the aspect for the next signal of interest, namely SE(8). For the condition when BK(5) is clear, i.e. not occupied, have the logic look at the signal in advance of SE(1), in this case SE(5). If SE(5) is not RED, then the logic would set SE(1) to REDGRN while for the condition where SE(5) equals RED the logic would set SE(1) = REDYEL, i.e. proceed on the divergent route at the prescribed speed and approach the next signal, at the end of the siding, prepared to stop.

Most prototype situations, and model as well, where trains are entering a siding, they are doing so in preparation for a meet with another train. Under these conditions, the majority of trains entering a siding need to be prepared to stop at the signal at the end of the siding. This is typically true even for "running meets" because the train when entering the siding never really knows for sure if the opposing train will be totally in the clear before reaching the end of the siding. Consequently, many prototype railroads find it desirable to limit divergent route aspects by requiring all trains entering the siding to approach the next signal, i.e. the signal at the end of the siding, prepared to stop. The primary driving force behind this simplification is reduced signaling complexity and lower cost. This example assumes this simplified approach and as a result it takes only a simple single one line statement to check out the divergent route, i.e. IF BK(5) = CLR

THEN SE(1) = REDYEL. Consequently, if BK(5) is occupied, not clear, then the coding after the THEN is ignored and SE(1) is retained at REDRED.

I find that most modelers, and I believe most prototypes as well, are happy with this simplified approach. However, if you are modeling a prototype that uses more involved divergent route aspects, then it does not take much added effort to refine the C/MRI code accordingly.

Calculating the aspect of signals leaving the siding and adjacent main track is even easier to grasp. For example, let's examine the statements used for calculating SW(2) and SW(5). First both signals are initialized to RED. Then if BK(5) is occupied, the program simply retains both signals at red by jumping to calculate the aspect of the next signal of interest, namely SW4. For the condition that BK(5) is clear, i.e. not occupied, the program simply checks the alignment of Turnout TU(1) and if normal then checks the signal in advance of SW(2), namely SW(1), and if not red sets SW(2) equal to green else SW(1) is red and the program logic sets SW(2) equal to yellow, i.e. approach prepared to stop at next signal. Essentially identical logic is used to calculate the correct aspect for Signal SW(5) for the condition where TU(1) is aligned for the divergent route.

Calculating the correct aspect for the intermediate signals, namely SE(3), SE(7), SW(1) and SW(8) in our example, is extremely straightforward because no turnout alignment checking is involved. Basically, once you have one of each type of signal situation understood, all the others of the same type fall into place, and the statements are repeated, with the exception of changing the numbers. This makes "copy and paste" an important tool set used in generating most large C/MRI system programs.

However, at this point we need to be a little careful because up to this point we have calculated what one might refer to as being the "basic ABS-type signal aspects." These aspects provide protection for following movement only which is the role of ABS signals. To protect for opposing movements on stretches of single track between passing sidings, we need to borrow and an important feature of APB signaling.

Establishing Direction-of-Traffic. The three blocks of code after the direction-of-traffic label, "DOT:" are important features of APB signaling. It is important to include its "direction-of-traffic" feature here because most modelers are going to want it included in any signal system they care to use. Fundamentally, when a train enters single track such as BK(1) in Fig. 1, headed westbound, signals SE(4) and SE(6) both turn red. This is essential to prevent an opposing train from entering the same stretch of single track.

To code this action for this example, a direction-of-traffic variable is established for each signal block between passing sidings. The first block of code establishes these DOT variables. For example, in the first line, if BK(1) becomes occupied and DOT(7) is not already set for eastbound, then DOT(1) is set to westbound. This may seem a bit complicated at first but a bit of study should help clarify the situation. Saying the same thing in different words, if Block 1 becomes occupied and the direction-of-traffic through Blocks 1 and 7 has not already been pre-set as eastbound, then the only safe course is to assume that Block 1 became occupied by a westbound train. Therefore set DOT(1) = WBD. The same logic works for setting DOT(3), DOT(7) and DOT(8). Using this logic, following movements through each stretch of single track are permitted but not opposing movements.

The next block of code resets the direction-of-traffic to "NDT" (no direction-of-traffic), when both single-track blocks become clear. When both blocks are clear, a train can enter the single

track from either direction. Once it does, it will set the DOT variable to prevent opposing movements. The function of the third block of code sets opposing signals to stop.

Approach Lighting. The FOR-NEXT loop implements approach lighting. If this feature is not desired, then simply leave out the code. If you include it, the code will set each signal to dark if the block *in- approach-to* the signal is clear. Alternatively, if you desire that only the intermediate signals be approach lighted – which is the procedure followed by most prototype using approach lighting – then simply replace the FOR-NEXT loop statements with the following four statements:

```
IF BK(1) = CLR THEN SW(1) = DRK
IF BK(3) = CLR THEN SE(3) = DRK
IF BK(7) = CLR THEN SE(7) = DRK
IF BK(8) = CLR THEN SW(8) = DRK
```

Signal Calculation Procedure. The recommended procedure for calculating signal aspects is to repetitively set each signal to a more refined value as the program works its way through the real-time loop. For instance, in this example the signals are first initialized to red or red over red. Then ABS procedures are employed, and wherever it was safe to do so, the signals are set to less restrictive aspects. Then, calculations considering direction-of-traffic are used to set opposing signals to red. Lastly and when desired, approach lighting considerations set all signals dark unless the block in approach to the signal is occupied. All the above are simply intermediary calculations used to arrive at the finalized prototypically correct setting for each signal which is the only setting that is transmitted out to the railroad.

Packing Outputs. The last operation within the real-time loop is to pack the outputs into their corresponding bytes and write them out to the railroad via the GOSUB OUTPUT statement. Like unpacking, packing takes one statement per output device and follows a similar procedure as explained in detail in Chapter 8 of the V3.0 C/MRI User's Manual. You start by setting each output byte equal to the first device within its port followed by repeated use of the statement OB(x) = device * By OR OB(x) where you substitute the byte number for x and the starting bit location within the byte for y. For example, on the I/O worksheet we see that the first output byte is to contain SE(1), SE(2) starting at bit position 2 and SW(2) starting at bit position 6. Thus the corresponding packing statements are written as:

```
OB(1) = SE(1)
OB(1) = SE(2) * B4 OR OB(1)
OB(1) = SW(2) * B6 OR OB(1)
OB(1) = OB(1) XOR 255
```

Because this example, connecting 3-lead searchlight LEDs directly to the SMINI, requires that output ports to be configured in their alternate current-sourcing the XOR operation is included to compensate for the inverted hardware logic built into the SMNI, and the DOUT32 cards, when output ports are configured for current-sourcing. With packing of the output bytes complete, they are then sent to the railroad using the standard statement GOSUB OUTPUTS. When that is complete, the GOTO BRTL statement branches the program back to the beginning of the real-time loop to begin the process all over again. This read the inputs, calculate the outputs e.g. our signal aspects, and transmit the outputs to the railroad. This cyclic-loop is repeated multiple times a second; thereby always keeping each signal at the correct prototypical aspect.