drop5.asm

; PULSES SWITCH
RTS

; NOT
EOR $5F
AND $0610
JSR PUTFAA
JSR SWITCH3
RTS

; LOAD A PORT OF VIA AT A000 MAILBOX
ORA $5F
JSR PUTFAA
JSR SWITCH4 ; PULSES SWITCH
RTS

; NOT
EOR $5F
AND $0610
JSR PUTFAA
JSR SWITCH4
RTS

; SWITCH 1
LDA $5W1
JSR PUTFAA
JSR NEXT
LDA $5W1
JSR NEXT
JSR PUTFAA
RTS

; SWITCH 2
LDA $5W2
JSR PUTFAA
JSR NEXT
LDA $5W2
JSR NEXT
JSR PUTFAA
RTS

; SWITCH 3
LDA $5W3
JSR PUTFAA
JSR NEXT
LDA $5W3
JSR NEXT
JSR PUTFAA
RTS

; SWITCH 4
LDA $5W4
JSR PUTFAA
JSR NEXT
LDA $5W4
JSR NEXT
JSR PUTFAA
RTS

; PUTAA SUBROUTINE PLACES ACCUMULATOR IN A PORT
; OF A000 VIA
-------------------

; FOR THE RELAYS
STA $A001
STA $A009
RTS

; THIS SUBROUTINE DOES NOT DO XORING. IT'S USED
STA $A010
STA $A0E1
RTS

;***************************************
; BAR Code Reader
;***************************************

; JSR SOUTH
; WAIT FOR SOUTH TRACK TO BE CLEAR
LDA $5F
STA SOVRD
; KILL SOUTH RELAY TRACK
LDA $5D
STA $4000
; DISPLAY 'F0' FOR FORWARD SLOW
JSR GETF2
; WAIT FOR 1-2-RIGHT
LDA $5D
STA SOVRD
; REPOWER SOUTH RELAY TRACK
JSR SOUTH
; WAIT FOR SOUTH TRACK TO HAVE A TRAIN
LDA $5D
STA LCAR
; INITIALIZE LAST CAR MAILBOX TO 00
LDA $9F
EOR $FF
STA $8001
; STROBE CLEAR BIT ON SHIFT REGISTER
LDA $5F
STA $8001
; STOP STROBING CLEAR BIT
READ JSR SHIFT
; READ BITS IN SHIFT REGISTER AND DISPLAY ON
; COMPUTER TIL DISPLAYS
; LCAR IS UPDATED EACH TIME THIS SUBROUTINE IS CALLED
LDA $9F
BNE STRAIN
; IF TRAIN HASN'T PASSED SOUTH CLEAR, KEEP CHECKING
; FOR MORE CARS
RTS

;***************************************
; SHIFT: READ SHIFT REGISTER
;***************************************

SHIFT LDA $8001
; LOAD VIA INTO A VARIABLE
STA VAR3
; ROTATE ONE BIT TO THE RIGHT
ROR A
AND $8F
; KEEP ONLY TWO LEFTMOST BITS
STA $4000
; DISPLAY CAR NO.
STA LCAR
; STORE CAR NO. IN VARIABLE LCAR
RTS

END
ABSTRACT

This project automatically controls the actions of multiple trains simultaneously by sensing their location and sequencing and actuating their motion (via track kills and re-energizing) and direction (via switch throwing). This is implemented through a train station framework with switches and optical sensors. All components are controlled and coordinated through use of a single board computer (SBC). The project successfully addresses the important transportation issues faced by railway engineers regarding train organization and automatic collision avoidance.

Matt Morris
David Larson
May 23, 1997
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTRODUCTION</td>
<td>3</td>
</tr>
<tr>
<td>DESCRIPTION OF TRACK LAYOUT</td>
<td>4</td>
</tr>
<tr>
<td>SENSOR DESCRIPTION</td>
<td>5</td>
</tr>
<tr>
<td>ACTUATOR DESCRIPTION</td>
<td>7</td>
</tr>
<tr>
<td>TERMINAL INTERFACE</td>
<td>9</td>
</tr>
<tr>
<td>COMPUTER INTERFACE</td>
<td>10</td>
</tr>
<tr>
<td>PAL 16L8</td>
<td>12</td>
</tr>
<tr>
<td>PROGRAM EXPLANATION</td>
<td>13</td>
</tr>
<tr>
<td>Program Flow Charts</td>
<td>14</td>
</tr>
<tr>
<td>Program Listing</td>
<td>29</td>
</tr>
<tr>
<td>CIRCUIT DIAGRAMS</td>
<td>39</td>
</tr>
<tr>
<td>Track 0 &amp; 1 optical sensor and RS flip flop</td>
<td>40</td>
</tr>
<tr>
<td>Track 0 &amp; 1 Relay Kill</td>
<td>41</td>
</tr>
<tr>
<td>UART/T Daughter Board</td>
<td>42</td>
</tr>
<tr>
<td>Switch Actuator Circuit</td>
<td>43</td>
</tr>
<tr>
<td>North Optical Sensor</td>
<td>44</td>
</tr>
</tbody>
</table>

Very creative
UART/excellent
execution excel
Project is robust!

A

May 23, 1997
Introduction

This project uses a single board computer (SBC) with a 6502 microprocessor to control the operations of as many as 15 trains on a track given power by a Hornby master controller unit. The features that are unique to this project are four optical sensors, four track switches, and a UAR/T receiver transmitter chip which allows the computer to display information on a Perkin-Elmer 550 Terminal.

The premise behind Robust Junction is that trains proceed around the main track (north and south tracks) until they are marked as trains which should switch off of the north track onto the station track. Once a train is on the station track, there are three possible scenarios:
1) The station (tracks 1 and 0) is empty. In this case, the train will roll into track 0.
2) Either track 1 or track 0 is occupied. The train will simply move into the empty slot.
3) Both tracks are occupied. In this event, the train will stop just after activating a sensor before the station and will wait until the train that has been in the station longer leaves. However, the newly arriving train will not begin to move until the departed train is safely back on the north track, as indicated by another sensor on the north track. This prevents, for example, a fast incoming train from running into a slow outgoing one.

The Perkin-Elmer terminal displays which trains are slated for arrival at the track, the status of each of the station tracks (1 and 0), and the state of each of the four switches (straight or switched).
Description of Track Layout
Sensor Description

Inside the TIL139 the infrared light emitting diode is always on. When the reflective tape of an engine passes in front of the sensor, the infrared light is reflected back to the sensor and turns on the transistor. Now the amount of light reflected creates a potential. This potential is fed into a LM 339 comparator which compares it to a voltage obtained by a voltage divider. A Schmitt trigger was considered but not used because only the knowledge of when the train first passed in front of the sensor was necessary (only the upward transition important). The value of the comparing voltage in the voltage divider was determined by trial and error.

**Basic Optical Sensor Circuit Diagram**
Basic RS Flip Flop Circuit Diagram

The flip-flop takes the inputs from the optical sensor and the computer and produces an output. There are two states on each of the lines. From the optical sensor (O) a high means an engine is passing (or is stopped) in front of the optical sensor (It assumes a reflective piece of tape is placed in the proper location atop the bar code of the engine in accordance with the labeling requirements stated in Volume 1 of the MAE 412 readings). A low from the optical sensor means there are no engines in front of the sensor. A high from the computer (C) should force the track on immediately and will stay on as long as (C) is high. A low from the computer will allow the sensor to decide whether the track is on or off.

The flip-flop is initialized to be on (track on). Also, the computer is set to a low and it is assumed there are no engines on the track. Thus the optical sensor is a low. This is the quiescent state and the track remains on, until the RESET line is brought high which is the case when the optical sensor goes high due to an engine passing in front of the sensor. The flip-flop then immediately goes low, turning the track off and stopping the engine safely. As the engine slides by the sensor the optical sensor will again go back to a low, but because of the flip-flop the track will stay off.

The track will stay off until the computer goes high and sets the flip flop back on, turning on the track and allowing the engine to leave the station. Then the computer can go back to low, allowing the sensor to kill the track the next time a train passes in front.
Actuator Description

There are two types of actuation that are used in this project: track kills (and restarts) and track switching. Both involve relays. The diagram below shows the basic method of track kill.

**Basic Track Kill Circuit Diagram**

![Circuit Diagram]

The track kill relay is relatively simple. The signal from the RS flip flop circuit comes in and is inverted in a 74LS04 hex inverter. This signal is used in a transistor: when the inverted signal is high, the transistor gives no current to the relay, so the track is switched off; when the inverted signal is low, the relay is energized, throws the switch, and the track turns on. The diode across the relay coil is there to ensure that current flows only one way. This is necessary due to the high inductive load.

The switch circuit (shown in the circuit diagrams section) uses a 4N40 optical SCR, four transistors, and four relays. The computer sends four signals, S1-S4, to the board. These signals are then buffered. S4 goes to the 4N40 and determines whether power is sent to relay 1 to initiate switching. S1 goes to relay 1 and its state decides whether a main or station switch is to be altered. S2 goes to relay 2 and decides whether left or right switches are being switched. S3 goes to both relay 3 and relay 4 and decides which way the switch is to throw. The diode (1N4004) across the switch coil protects it from transient currents.
Terminal Interface

A unique aspect of the Robust Junction is its train board display. The project uses a standard Universal Asynchronous Receiver/Transmitter (UART) chip and RS-232 DB-25 connector to connect to a Perkin-Elmer 550 terminal. The UART requires a bit rate generator, crystal, and line driver, as well as a buffer, inverter, and power-up reset circuit.

The Robust Junction UART is only set up to transmit data, which is all that is required of a train board. This simplifies the circuit significantly. The UART is set to 8 data bits, even parity, and 2 stop bits. These values were chosen somewhat arbitrarily. They seem to work best with the Perkin-Elmer 550.

The bit rate generator uses the crystal to lay out a precise frequency clock signal. The highest BAUD rate the Perkin-Elmer 550 can handle is 9600, so the bit rate generator is set to 16 times this desired BAUD rate. The UART requires a clock signal at sixteen times the desired transmitting BAUD rate.

The line driver takes the serial TTL (+5V, ground) signal from the UART and converts it to an RS-232 signal of (-12V, +12V). Now the ±12V sources are taken from the Perkin-Elmer 550 terminal. This is a NON-STANDARD feature of this particular brand of terminal. In order for the terminal to put out this supply voltage its DIP mode switches number 6 and 7 must be in the ON position.

The reset circuit is identical to the one used on the single board computer, and merely resets the UART and bit rate generator whenever power is applied.

The Perkin Elmer 550 terminal must be Online, full duplex, printer off, 9600 BAUD and have the following mode switch settings:

<table>
<thead>
<tr>
<th>Switch #</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>line button depressed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setting</td>
<td>off</td>
<td>on</td>
<td>off</td>
<td>off</td>
<td>off</td>
<td>ON</td>
<td>ON</td>
<td>off</td>
<td>ON</td>
<td></td>
</tr>
</tbody>
</table>

The Perkin-Elmer takes a few seconds to warm up, and if the computer needs to be reset, first hold down CONTROL and press CLEAR on the terminal and then reset the computer. This clears the terminal screen of any extraneous characters.
## Computer Interface

This section shows all the connections from the computer to the other parts of the train board. All connections except for those to the UAR/T board are through a 44-pin edge connector. The UAR/T connections are through a ribbon that plugs directly into the SBC. Shown below are the edge pin and UAR/T ribbon connections to the VIAs. The respective circuit diagrams for each of the indicated boards give pin locations for each of pins 12 through 22. Those pins are the ones unique to this project. All output lines on the SBC go through a 74LS244 buffer and then to the 44 edge pin connector. The only tricky one is the force-the-north-track-clear line. This one goes to an open collector logic hex inverter, and the output of this inverter goes through the 44 edge pin connector to pin 7 on the DB-25 connector going to the test stand board.

The input lines unfortunately do not get buffered on their way into the VIAs, but on the daughter boards they are buffered on their way out, which accounts for this difference. The inputs jump directly from the 44 edge pin connector to the VIA.

<table>
<thead>
<tr>
<th>Pin#</th>
<th>Color</th>
<th>Description</th>
<th>VIA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>black</td>
<td>logic ground</td>
<td>GND</td>
</tr>
<tr>
<td>2</td>
<td>red</td>
<td>logic power</td>
<td>+5V</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>black</td>
<td>test stand</td>
<td>1,PA</td>
</tr>
<tr>
<td>5</td>
<td>white</td>
<td>test stand</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>yellow</td>
<td>test stand</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>blue</td>
<td>test stand</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>test stand</td>
<td>test stand</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>green</td>
<td>test stand</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>brown</td>
<td>test stand</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>red</td>
<td>test stand</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>grey</td>
<td>track 1 and 0 optical sensor circuit</td>
<td>1, PA0</td>
</tr>
<tr>
<td>13</td>
<td>purple</td>
<td>track 1 and 0 optical sensor circuit</td>
<td>1, PA1</td>
</tr>
<tr>
<td>14</td>
<td>blue</td>
<td>on-deck optical sensor circuit</td>
<td>1, PA2</td>
</tr>
<tr>
<td>15</td>
<td>orange</td>
<td>north track optical sensor circuit</td>
<td>2, PB1</td>
</tr>
<tr>
<td>16</td>
<td>blue</td>
<td>switch circuit</td>
<td>1, PA4</td>
</tr>
<tr>
<td>17</td>
<td>grey</td>
<td>switch circuit</td>
<td>1, PA5</td>
</tr>
<tr>
<td>18</td>
<td>purple</td>
<td>switch circuit</td>
<td>1, PA6</td>
</tr>
<tr>
<td>19</td>
<td>white</td>
<td>switch circuit</td>
<td>1, PA7</td>
</tr>
<tr>
<td>20</td>
<td>white</td>
<td>on-deck optical sensor circuit</td>
<td>2, PB5</td>
</tr>
<tr>
<td>21</td>
<td>brown</td>
<td>track 1 and 0 optical sensor circuit</td>
<td>2, PB4</td>
</tr>
<tr>
<td>22</td>
<td>orange</td>
<td>track 1 and 0 optical sensor circuit</td>
<td>2, PB3</td>
</tr>
<tr>
<td>A</td>
<td>green</td>
<td>track ground</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>yellow</td>
<td>track power</td>
<td></td>
</tr>
</tbody>
</table>

**UAR/T #**

- Description: VIA
- port

1. see UAR/T diagram
2. PB0
3. PA0
4. PB7
5. PA1
6. PA2
7. PA3
8. PA4
9. PA5
10. PA6
11. PA7

3:31 PM 10 May 23, 1997 373
The address decode logic for the SBC was implemented using a PAL 16L8 chip. Included here are a schematic of the PAL design, diagrams for all of the logic functions, and a diagram of the actual PAL chip.
logic diagram (positive logic)

Fuse number = First fuse number + Increment
Program Explanation

The best way to understand the program is to take a brief look at the flow charts following this basic text. The first describes the initializing and the main loop. Then there is another dealing with a different train on either the north (NEWNTRN) or south (NEWSTRN) track, as well as a different console command (NEWRDAT). These are the primary subroutines called by the main program.

There are a number of secondary subroutines that the primary subroutines use at various times. The most significant one of these is (STOPIT) which is called when a new train has entered the north track and it is scheduled to stop at the station. This subroutine will move that train to the right track, clear that track if there's a train on it, and call the appropriate routines to switch the switches and update the terminal display.

All of the procedures rely on some basic routines that switch the switches (SWTTT), update various aspects of the terminal (UPRB) - train board, DISPLST - train list, SWTSTAT - switch status) and the most important is the routine that takes an ASCII message in the EEPROM and transfers it to the terminal (DISPMSG). In addition, in order to clear the North track so Lecky will allow another train on after a train has been stopped at the station, a procedure was written and an open collector logic gate added (FCLR).

A few of the procedures are run only during initialization. These include one to set up all the nice looking blocks and labels on the terminal (BRDSET). Another one (CLRLST) clears the train list and sets them so they all stop at the station upon reset. And one sets all the switches to straight and updates the switch status board at the same time (INTSWT).

As one can see by the program listing, the Robust Junction program is long, and looks rather complicated. But it can be broken down into only a few basic sections. The majority of the running time is spent in the main loop checking whether or not a different train enters on the North or South track, and whether or not a different command is entered from the Hornby Console to computer #5. If one of these events happens then the program jumps to the appropriate subroutine. If not, the program stays in this loop until something changes.
Program Flow Charts

The flow charts follow the standard of a round rectangle for an operation, and a parallelogram for a decision. Frequently the word RUN is used. This refers to a jump subroutine command in assembly language. Upon completion of the subroutine the program will return to the line right after the RUN block.

List of Flow Charts

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAIN LOOP</td>
<td>15</td>
</tr>
<tr>
<td>NEWSTRN</td>
<td>16</td>
</tr>
<tr>
<td>NEWNTRN</td>
<td>17</td>
</tr>
<tr>
<td>STOPIT</td>
<td>18</td>
</tr>
<tr>
<td>BRDSET</td>
<td>20</td>
</tr>
<tr>
<td>DISPMSG</td>
<td>21</td>
</tr>
<tr>
<td>CLRRLST</td>
<td>22</td>
</tr>
<tr>
<td>FCLR</td>
<td>22</td>
</tr>
<tr>
<td>DISPLST</td>
<td>23</td>
</tr>
<tr>
<td>NEWRDAT</td>
<td>24</td>
</tr>
<tr>
<td>SWTSTAT</td>
<td>25</td>
</tr>
<tr>
<td>SWITI</td>
<td>26</td>
</tr>
<tr>
<td>INITSWT</td>
<td>27</td>
</tr>
<tr>
<td>UPRBJ</td>
<td>28</td>
</tr>
</tbody>
</table>
MAIN Loop

Initialization

set BLKID to computer #5

Run Lecky's INIT

Initialize variables

Set up VIA's

RUN BRDSET sets up terminal

RUN CLRLST clears train list

RUN INITSWT sets all switches to straight

RUN UPRBJ sets up the track status board

Main Program

Main Loop (LOOPIT)

Check for different Train on North Track

same

different

Run NEWNTRN

Check for different Train on South Track

same

different

Run NEWSTRN

Check for different Hornby Command

same

different

Run NEWRDAT
NEWSTRN

Check if south track clear

not clear
Set up ASCII message saying "Train #_" positioned bottom right of terminal
RUN DISPMSG puts this message on the terminal
Return from subroutine

clear
Set up ASCII message saying "Clear" positioned bottom right of terminal
RUN DISPMSG puts this message on the terminal
NEWNTRN

Check if north track clear

- clear
  - Set up ASCII message saying "Clear" positioned bottom left of terminal
  - RUN DISPMSG puts this message on the terminal

- not clear
  - Set up ASCII message saying "Train #_" positioned bottom left of terminal
  - RUN DISPMSG puts this message on the terminal

Check if this train should stop at the station

- don't stop
  - RUN STOPIT stops train at station

- stop
  - Return from subroutine
STOPIT

Set SWTDAT to code equivalent to main left turnout

RUN SWTIT (switches SWTDAT code)

Check which track to put train on

Track 1
Set SWTDAT to code equal to station left turn
RUN SWTIT
Set SWTDAT to code equal to station right turn
RUN SWTIT
Change Board to show train on Track 1 is departing (UPRBJ)

Track 0
Set SWTDAT to code equal to station left straight
RUN SWTIT
Set SWTDAT to code equal to station right straight
RUN SWTIT
Change Board to show train on Track 0 is departing (UPRBJ)

Check whether Train has reached on-deck sensor

no train
Set SWTDAT to code equivalent to main left switch straight
RUN SWTIT

train

Check NXTRK variable to see which track train is moving onto

Track 0
Track 1
Check TOTRN variable to see if there is a train on Track 0

- Yes
  - Set SWTDAT to main right turn
  - RUN SWTIT
  - Turn on Track 0 section

- No
  - Check north sensor
    - Yes
      - Set SWTDAT to main right straigh
      - RUN SWTIT
      - Update Board to show train as stopped
      - Run FCLR (forces N clear)
      - Turn the on deck track on
      - Check Track 0 sensor
    - No
      - Update Board to show this train arriving on 0
      - Set the next track to be filled as track 1 (NXTRK)

- No train
  - Update Board to show arrived
  - Set the next track to be filled as track 1 (NXTRK)

Return from this subroutine

Check TOTRN variable to see if there is a train on Track 1

- Yes
  - Set SWTDAT to main right turn
  - RUN SWTIT
  - Turn on ondeck track section

- No
  - Check Track 1 sensor
    - Yes
      - Set SWTDAT to main right straigh
      - RUN SWTIT
      - Update Board to show train as stopped
      - Run FCLR (forces N clear)
      - Turn the on deck track on
      - Check Track 1 sensor
    - No
      - Update Board to show this train arriving on 1
      - Set the next track to be filled as track 0 (NXTRK)

- No train
  - Update Board to show arrived
  - Set the next track to be filled as track 0 (NXTRK)
BRDSET

Put the High Byte of the address of North/South block ASCII message into zero page location MESGH

Put the low byte of the address of North/South block ASCII message into the zero page location MESGL

RUN DISPMMSG

Put the High Byte of the address of Train list block ASCII message into zero page location MESGH

Put the low byte of the address of Train list block ASCII message into the zero page location MESGL

RUN DISPMMSG

Put the High Byte of the address of switch status block ASCII message into zero page location MESGH

Put the low byte of the address of switch status ASCII message into the zero page location MESGL

RUN DISPMMSG

Put the High Byte of the address of Train board block ASCII message into zero page location MESGH

Put the low byte of the address of Train board block ASCII message into the zero page location MESGL

RUN DISPMMSG

Return from BRDSET Subroutine
DISPMSG

Load in the next character from message

Is this character a <NUL> (00)?

- NO
  - Send this 8 bit character out on VIA to UART
    - Wait for the End of Character (EOC) line to go low
    - Pulse the Data Strobe (DS) line low then return it to high
    - Wait for the End of Character (EOC) line to go high

- YES
  - Return from DISPMSG subroutine
CLRLST

CLRLST

Put binary number 1000000 into all 16 train status locations $0000 - $05FF

Return from BRDSET Subroutine

FCLR

FCLR

Bring VIA port connected to open collector logic inverter and pin 7 on DB-25 to a high (forcing north track clear)

Wait 256 cycles

Reset the VIA to its former states, bringing force clear port back to a low

Return from Subroutine
DISPLST

Put 4 x 4 grid of engine numbers and cursor addressing characters in ram ($0600)

Check to see if this engine should stop at station (uses train list $0500-$050F)

No stop

Put an ASCII space over its engine number in RAM ($0600)

Stop

Have all the Engines been checked?

Move on to the next one

Store the address of the message ($0600) to the MESGH, MESGL bytes

RUN DISPMSG

Return from Subroutine
NEWRDAT

Check if it's a <- 9 command

Yes
Clear the train from track 1
Return from Subroutine

NO

Check if it's a <- 8 command

Yes
Clear the train from track 0
Return from Subroutine

NO

Check if left arrow was pressed

yes
Add ten to number pressed
Invert the most significant bit of the train list byte corresponding to that engine number

no
Don't add ten to number pressed

RUN DISPLST

Return from Subroutine
SWTSTAT

Check bit 7 of SWTBUF

High
Show Left Main as Turnout

Low
Show Left Main as Straight

Check bit 6 of SWTBUF

High
Show Right Main as Turnout

Low
Show Right Main as Straight

Check bit 5 of SWTBUF

High
Show Left Station as Turnout

Low
Show Left Station as Straight

Check bit 5 of SWTBUF

High
Show Right Station as Turnout

Low
Show Right Station as Straight

Return from Subroutine
SWTIT

Set the S1-S4 bits on the VIA to bits 7-4 on the SWTDAT code

Output these to the VIA
(set the relays so switching signal goes to proper switch and direction)

Wait for Frame 1 to come across on Hornby signal
(Lecky mailbox FRANUM == 1)

Bring the S4 line on the VIA to a low
(energizing the switch coil)

Wait for Frame 2 to come across on the Hornby signal
(Lecky mailbox FRANUM == 2)

Bring the S4 line on the VIA back to a high

Update the SWTBUF variable to reflect the proper switch changes

RUN SWTSTAT
(to update switch status terminal block)

Return from Subroutine
IN T WT

- **INITSWT**
  - Load in code into SWTDAT to switch main right switch straight
    - **RUN SWTIT**
      - Load in code into SWTDAT to switch main left switch straight
        - **RUN SWTIT**
          - Load in code into SWTDAT to switch station right switch straight
            - **RUN SWTIT**
              - Load in code into SWTDAT to switch station left switch straight
                - **RUN SWTIT**
                  - Return from Subroutine
UPRBJ

(updates train board on terminal)

Copy cursor addressing characters and track 1 and 0 message to RAM
(the train numbers and status will be filled in soon)

Load in train number on track 0
(T0TRN)

Convert to ASCII character (1 - F)

Store this character in proper message location in RAM

Read in status of train on track 0
(T0STAT)

Copy message corresponding to status code into proper message location in RAM

Load in train number on track 1
(T1TRN)

Convert to ASCII character (1 - F)

Store this character in proper message location in RAM

Read in status of train on track 1
(T1STAT)

Copy message corresponding to status code into proper message location in RAM

RUN DISPMSG
(nuts message from RAM onto terminal)

Return from Subroutine
Program Listing

.include LECHY3.LASM

; Set up and data locations $V1 for VIA $5A000, V2 for VIA $5B000
; SET for setup (0-input 1 output) DAT for data

; V2ASET EQU $E003
; V2BSET EQU $E002
; V2ADAT EQU $E001
; V2ADAT EQU $E000

V2ASET EQU $8003
V2BSET EQU $8002
V2ADAT EQU $8001
V2ADAT EQU $8000

TIL EQU $4000

; START OF PROGRAM
ORG $2000

HYINIT SEI
CLD
cld
LEX $FF
TIX

; set up as computer $05
LDA $505
STA BLKD

JSR INI

; run Lecky initialization

V2ADAT LDA $FF :set up UART/T outputs (0 data bits)
STA V2ASET

; use VIA $8000 port A

STA V2ASET

; Outputs for:
; switches track kills, force clear
; Switch4 (1 - no switch 0 - switch it)
; Switch3 (1 - Straight 0 - Turn)
; Switch2 (1 - Left 0 - Right)
; Switch1 (1 - Station 0 - Main)
; Switch2 (1 - XTrack H - Force clear)

; deck (H - on L - sensor decides)
; PA1 - Track 1 (H - on L - sensor decides)
; PA2 - Track 2 (H - on L - sensor decides)
; uses VIA $5A000 port A

STA V2ADAT

; initialize outputs

LDA $FF :set up buffer for this port
STA V2ADAT

; and don't force track clear

; STK V2ADAT

; Main program loop

:loop

; display north/south trains on TIL

LDA XTRAIN

; check if a new/no train on north track

CMP OTRN

BEQ SAME

STA OTRN

JSR MEDITR

; If it is the same continue

; in main loop

SANE

; check if a change in

; handle it

LDA STRAIN

CMP OTRN

BEQ SAME

STA OTRN

JSR MEDITR

; If it is a change in

; handle it

SANE

; check to see if a new

; console command

LDA EDTA

; output to TIL for user to see

CMP ORDATA

BEQ SAME

STA EDTA

JSR MEDITA

; If it is a new one handle it

SANE

; Procedure: MEDITA

; a new train has entered south track
or a train has left the south track
display as status message to terminal
Uses a few bytes of ram on page $0A00 to assemble a message

; display a few bytes of ram on page $0A00 to assemble a message
STXT  EQU $0A00
STX1  EQU $0A01

; if south track is now clear, deal with it
LDA NUXTXT, X ; load in ASCII character equivalent to train number
STA STXT
LDA $010
STA STXT1 ; puts an end of message at end

; display last half of message
LDA #HIGH NMTSGA
STA MESGN
LDA #LOW NMTSGA
STA MESGL
JSR DISPMSG

; display train number
LDA #HIGH NMTSGA
STA MESGN
LDA #LOW NMTSGA
STA MESGL
JSR DISPMSG

; display 2nd half of message
LDA #HIGH NMTSGB
STA MESGN
LDA #LOW NMTSGB
STA MESGL
JSR DISPMSG

; now check if this train is to stop at station
LDA LSTLOC, X ; stops at station or not
BPL DONTSTOP ; if not then let it pass
JSR STOPT ; if it is on the list then stop it at station

; deal with $00 if the north track is clear
NOSTRN LDA #HIGH NMTSGA
STA MESGN
LDA #LOW NMTSGA
STA MESGL
JSR DISPMSG

; deal with $00 if the north track is clear
NOSTRN LDA #HIGH NMTSGA
STA MESGN
LDA #LOW NMTSGA
STA MESGL
JSR DISPMSG

; deal with $00 if the north track is clear
NOSTRN LDA #HIGH NMTSGA
STA MESGN
LDA #LOW NMTSGA
STA MESGL
JSR DISPMSG

; display train number
LDA #HIGH NMTSGA
STA MESGN
LDA #LOW NMTSGA
STA MESGL
JSR DISPMSG

; display 2nd half of message
LDA #HIGH NMTSGB
STA MESGN
LDA #LOW NMTSGB
STA MESGL
JSR DISPMSG

; display message in NMTSGA
LDA #HIGH NMTSGA
STA MESGN
LDA #LOW NMTSGA
STA MESGL
JSR DISPMSG

; display message in NMTSGA
LDA #HIGH NMTSGA
STA MESGN
LDA #LOW NMTSGA
STA MESGL
JSR DISPMSG

; display message in NMTSGA
LDA #HIGH NMTSGA
STA MESGN
LDA #LOW NMTSGA
STA MESGL
JSR DISPMSG

; display message in NMTSGA
LDA #HIGH NMTSGA
STA MESGN
LDA #LOW NMTSGA
STA MESGL
JSR DISPMSG

; display message in NMTSGA
LDA #HIGH NMTSGA
STA MESGN
LDA #LOW NMTSGA
STA MESGL
JSR DISPMSG

; display message in NMTSGA
LDA #HIGH NMTSGA
STA MESGN
LDA #LOW NMTSGA
STA MESGL
JSR DISPMSG

; display message in NMTSGA
LDA #HIGH NMTSGA
STA MESGN
LDA #LOW NMTSGA
STA MESGL
JSR DISPMSG

; display message in NMTSGA
LDA #HIGH NMTSGA
STA MESGN
LDA #LOW NMTSGA
STA MESGL
JSR DISPMSG

; display message in NMTSGA
LDA #HIGH NMTSGA
STA MESGN
LDA #LOW NMTSGA
STA MESGL
JSR DISPMSG

; display message in NMTSGA
LDA #HIGH NMTSGA
STA MESGN
LDA #LOW NMTSGA
STA MESGL
JSR DISPMSG

; display message in NMTSGA
LDA #HIGH NMTSGA
STA MESGN
LDA #LOW NMTSGA
STA MESGL
JSR DISPMSG

; display message in NMTSGA
LDA #HIGH NMTSGA
STA MESGN
LDA #LOW NMTSGA
STA MESGL
JSR DISPMSG

; display message in NMTSGA
LDA #HIGH NMTSGA
STA MESGN
LDA #LOW NMTSGA
STA MESGL
JSR DISPMSG

; display message in NMTSGA
LDA #HIGH NMTSGA
STA MESGN
LDA #LOW NMTSGA
STA MESGL
JSR DISPMSG

; display message in NMTSGA
LDA #HIGH NMTSGA
STA MESGN
LDA #LOW NMTSGA
STA MESGL
JSR DISPMSG

; display message in NMTSGA
LDA #HIGH NMTSGA
STA MESGN
LDA #LOW NMTSGA
STA MESGL
JSR DISPMSG

; display message in NMTSGA
LDA #HIGH NMTSGA
STA MESGN
LDA #LOW NMTSGA
STA MESGL
JSR DISPMSG

; display message in NMTSGA
LDA #HIGH NMTSGA
STA MESGN
LDA #LOW NMTSGA
STA MESGL
JSR DISPMSG

; display message in NMTSGA
LDA #HIGH NMTSGA
STA MESGN
LDA #LOW NMTSGA
STA MESGL
JSR DISPMSG

; display message in NMTSGA
LDA #HIGH NMTSGA
STA MESGN
LDA #LOW NMTSGA
STA MESGL
JSR DISPMSG

; display message in NMTSGA
LDA #HIGH NMTSGA
STA MESGN
LDA #LOW NMTSGA
STA MESGL
JSR DISPMSG

; display message in NMTSGA
LDA #HIGH NMTSGA
STA MESGN
LDA #LOW NMTSGA
STA MESGL
JSR DISPMSG

; display message in NMTSGA
LDA #HIGH NMTSGA
STA MESGN
LDA #LOW NMTSGA
STA MESGL
JSR DISPMSG

; display message in NMTSGA
LDA #HIGH NMTSGA
STA MESGN
LDA #LOW NMTSGA
STA MESGL
JSR DISPMSG

; display message in NMTSGA
LDA #HIGH NMTSGA
STA MESGN
LDA #LOW NMTSGA
STA MESGL
JSR DISPMSG

; display message in NMTSGA
LDA #HIGH NMTSGA
STA MESGN
LDA #LOW NMTSGA
STA MESGL
JSR DISPMSG

; display message in NMTSGA
LDA #HIGH NMTSGA
STA MESGN
LDA #LOW NMTSGA
STA MESGL
JSR DISPMSG

; display message in NMTSGA
LDA #HIGH NMTSGA
STA MESGN
LDA #LOW NMTSGA
STA MESGL
JSR DISPMSG

; display message in NMTSGA
LDA #HIGH NMTSGA
STA MESGN
LDA #LOW NMTSGA
STA MESGL
JSR DISPMSG

; display message in NMTSGA
LDA #HIGH NMTSGA
STA MESGN
LDA #LOW NMTSGA
STA MESGL
JSR DISPMSG

; display message in NMTSGA
LDA #HIGH NMTSGA
STA MESGN
LDA #LOW NMTSGA
STA MESGL
JSR DISPMSG

; display message in NMTSGA
LDA #HIGH NMTSGA
STA MESGN
LDA #LOW NMTSGA
STA MESGL
JSR DISPMSG

; display message in NMTSGA
LDA #HIGH NMTSGA
STA MESGN
LDA #LOW NMTSGA
STA MESGL
JSR DISPMSG

; display message in NMTSGA
LDA #HIGH NMTSGA
STA MESGN
LDA #LOW NMTSGA
STA MESGL
JSR DISPMSG

; display message in NMTSGA
LDA #HIGH NMTSGA
STA MESGN
LDA #LOW NMTSGA
STA MESGL
JSR DISPMSG

; display message in NMTSGA
LDA #HIGH NMTSGA
STA MESGN
LDA #LOW NMTSGA
STA MESGL
JSR DISPMSG

; display message in NMTSGA
LDA #HIGH NMTSGA
STA MESGN
LDA #LOW NMTSGA
STA MESGL
JSR DISPMSG

; display message in NMTSGA
LDA #HIGH NMTSGA
STA MESGN
LDA #LOW NMTSGA
STA MESGL
JSR DISPMSG

; display message in NMTSGA
LDA #HIGH NMTSGA
STA MESGN
LDA #LOW NMTSGA
STA MESGL
JSR DISPMSG

; display message in NMTSGA
LDA #HIGH NMTSGA
STA MESGN
LDA #LOW NMTSGA
STA MESGL
JSR DISPMSG
STA SWDC
JMP STP1

JMP STP1 :go on with station stop

JMP STP1 :show departing on train board for track 0

: set the switches to track 0
LDA #11110000 : switch the station left to straight
STA SWDC
JSR SWDC

LDA #11110000 : switch the station right to straight
STA SWDC
JSR SWDC

STP1: JSR UPRRJ : update the train board

LDA VDBD : now wait for train to get to ondeck location
AND #00010000 : makes all other bits zero except for ondeck sensor
BNE WATTON : if train not there yet keep waiting

LDA #11110000 : switch the main left switch to straight
STA SWDC
JSR SWDC

LDA DTTN : figure out which track is the next one to open up
BEQ NTTK1 : if track 0 then go to track 1 as next track

JMP NTTK1 : if not then go to track 1 as next track

; track 0 is the destination for train currently at the on deck track
; must check and see if there is a train currently at the station on track 0

NEXTK1 LDA TOTP : if no train skip the part about moving a train off
BEQ NOTOTP : the train on track 0 has to be moved off and back onto north track

; to continue on its merry way;
LDA #11000000 : switch the main right switch to turnout
STA SWDC
JSR SWDC

; energize the track 0 section
LDA VIAJ : energize the on deck section of track
AND #11111111 : bring PA0 to a low
EDR #00000001 : bring PA0 to a high
STA VIAJ : output this to VIA

; now track 0 section are energized, and train begins to move away
LDA VIAJ : restore the track 0 to be killed when next train crosses in front of sensor
STA VIAJ

; wait until the leaving train clears the north track optical sensor
WATTTR LDA VIAJ : check PA2 for a train passing the north track optical
AND #00000010 : makes other bits 0
BEQ WAITTR : if bit 2 is low keep waiting

; now train has moved off track 0 and the on deck train can be moved on
LDA #11100000 : switch main right switch to straight
STA SWDC
JSR SWDC

; change the train number on the north track to this one
LDA TOTP : leaving station
STA NTERRY : this could be rule violation

; update the train board for the new train number to be moved on
LDA ONETR : set the new train number on train board
STA TOTP

LDA #01 : status code that refers to arriving
STA TOTP
JSR UPRRJ : update the train board

; so energize the on deck track section
LDA VIAJ : load in current output
AND #11111111 : bring the 2 bit to a low
EDR #00000001 : bring bit 2 to a high
STA VIAJ : turn on on deck track

; now wait for train to make it to track zero
WATTTR LDA VDBD : check the track 0 sensor and #00010000 : force all bits except for bit 4 to a zero
BNE WAITTR : if bit 4 is high then no train yet keep waiting
LDA VIAJ : restore the ondeck track
STA VIAJ : section to kill when sensor detects a train

; update the train board to show that new train is stopped at the station
LDA #03 : status code equivalent to stopped at station
STA TOTP
JSR UPRRJ : update train board

; change the NTTK variable to point to track board with the next train
LDA #02 : everything done and ready for next train
STA TOTP

; situation: moving a train onto track 0 and no train currently on track 0

NOTOTP LDA ONETR : update train board to show this train number as arriving
STA TOTP

LDA #01 : status code that refers to arriving
STA TOTP
JSR UPRRJ : update train board

; energize the on deck section of track
LDA VIAJ : energize the on deck section of track
AND #11111111 : bring PA0 to a low
EDR #00000001 : bring PA0 to a high
STA VIAJ : output this to VIA

; now track 0 section are energized, and train begins to move away
LDA VIAJ : restore the track 0 to be killed when next train crosses in front of sensor
STA VIAJ

; wait until the leaving train clears the north track optical sensor
WATTTR LDA VIAJ : check PA2 for a train passing the north track optical
AND #00000010 : makes other bits 0
BEQ WAITTR : if bit 2 is low keep waiting

; now train has moved off track 0 and the on deck train can be moved on
LDA #11100000 : switch main right switch to straight
STA SWDC
JSR SWDC

; force a clear on the north track
JSR FCLA

RTS : all done and ready for next train

; do the same except on track 1 instead of track 0
; must check and see if there is a train currently at the station on track 1

NEXTK1 LDA TOTP : if no train skip the part about moving a train off
BEQ NOTOTP
the train on track 1 has to be moved off and back onto north track

\[\text{LDA } \#10000000 \text{; switch the main right switch to turnout} \]
\[\text{STA TWIN} \]
\[\text{JSR ENMT} \]

; energize the track 1 section
\[\text{LDA VIARU} \]
\[\text{AND } \#11111101 \text{; bring PA1 to a low} \]
\[\text{EOR } \#00000001 \text{; bring PA1 to a high} \]
\[\text{AND } \#00000000 \text{; makes other bit 0} \]
\[\text{lea leaves other bits unchanged} \]
\[\text{STA VIADAT} \text{; output this to the VTA} \]

; now track 1 section energizes, and train begins to move away

; wait until the leaving train clears the north track optical sensor
\[\text{WAITINS LDA V2BDA} \]
\[\text{; check PB2 for a train passing the north track} \]
\[\text{optical} \]
\[\text{AND } \#00000000 \text{; makes other bit 0} \]
\[\text{lea leaves bit 2 unchanged} \]
\[\text{BRQ WAITINS} \text{; if bit 2 is low keep waiting} \]

; now train has moved off track 1 and the on deck train can be moved on
\[\text{LDA } \#11000000 \text{; switch main right switch to straight} \]
\[\text{STA SNMT} \]

; change the train number on the north track to
\[\text{LDA TITRN} \]
\[\text{; train number of train leaving station} \]
\[\text{STA NTRAIN} \text{; (new Leacky mailbox for this could be rule violation)} \]
\[\text{LDA VIARU} \text{; restore the track 1 to be} \]
\[\text{killed when next train} \]
\[\text{; crosses in front of sensor} \]

; now train has moved off track 1, and the on deck
\[\text{train can be moved on} \]
\[\text{LDA } \#01 \text{; set the new train number on train board} \]
\[\text{STA TITRN} \]
\[\text{LDA } \#01 \text{; status code that refers} \]
\[\text{STA TISSTAT} \]
\[\text{JSR UPRRJ} \text{; update train board} \]

; iso energize the on deck track section
\[\text{LDA VIARU} \text{; load in current output} \]
\[\text{values} \]
\[\text{AND } \#11111011 \text{; bring the 2 bit to a low} \]
\[\text{lea leaves other bits unchanged} \]
\[\text{EOR } \#00000000 \text{; bring bit 2 to a high} \]
\[\text{ent turns on ondeck track} \]
\[\text{STA VIADAT} \text{; turn on on deck track} \]

; now wait for train to make it to track 1
\[\text{WAITIT LDA V2BDA} \text{; check the track 1 sensor} \]
\[\text{AND } \#00000000 \text{; force all bits except for} \]
\[\text{bit 3 to a zero} \]
\[\text{BRQ WAITIT} \text{; if bit 3 is high then no} \]
\[\text{train yet keep waiting} \]
\[\text{LDA VIARU} \text{; restore the ondeck track} \]
\[\text{section to kill when} \]
\[\text{sensor detects a train} \]

; update the train board to show that this new
\[\text{LDA } \#03 \text{; status code equivalent to} \]
\[\text{STA TISSTAT} \]
\[\text{JSR UPRRJ} \text{; update train board} \]

; change the NXTK variable to point to track 0
\[\text{LDA } \#00 \text{; with the next train} \]
\[\text{STA NTRAIN} \]
\[\text{RTC} \text{; everything done and ready for the} \]
\[\text{next train} \]

; situation: moving a train onto track 1 and no
\[\text{train currently on track 1} \]

; update the train board to show this train number
\[\text{as arriving} \]
Programming

Robust Junction

Message: NSBKLK
; used to set borders around the north and south track blocks

NSBKLK DB $1B, 'X0', $0D

DB '---------', $1B, 'Y', '---------'

DB ', $0D, $0A

DB '| NORTH TRACK |', $1B, 'Y', ' | SOUTH TRACK |', $1B, 'Y', ' |

DB '----------', $0D, $0A

DB ' |', $1B, 'Y', ' |

DB ' |', $0D, $0A

DB ' |', $1B, 'Y', ' |

DB ' |', $0D, $0A

DB '----------', $1B, 'Y', '----------'

DB $0D

; Message: TPLBLK
; used to set borders around block that displays which trains will stop at the station

TPLBLK DB $1B, 'Y', '----------'

DB ', $0A

DB ' |', $1B, 'Y', ' |', $0A

DB ' |', $1B, 'Y', ' |', $0A

DB ' |', $1B, 'Y', ' |', $0A

DB ' |', $1B, 'Y', ' |', $0A

DB ' |', $0A

DB '----------'

TPLBLK DB $1B, 'Y', '----------'

DB ', $0A

DB ' |', $1B, 'Y', ' |', $0A

DB ' |', $1B, 'Y', ' |', $0A

DB ' |', $1B, 'Y', ' |', $0A

DB ' |', $1B, 'Y', ' |', $0A

DB ' |', $0A

DB '----------'

DB $0D

; Message: SNWBLK
; sets up the block on the terminal to display the status of the various switches

SNWBLK DB $1B, 'X0', '----------'

DB ', $0A

DB ' |', $1B, 'Y', ' |', $0A

DB ' | SWITCH STATUS |', $0A

DB ' | Main | Station |', $0A

DB ' | Left | Right | Left | Right |', $0A

DB ' |', $0A

DB ' |', $1B, 'Y', ' |', $0A

DB ' |', $0A

DB '----------'

DB $0D

; Message: RAJBLK
; sets up block to display which train on which station track.

; and its status (in future maybe add time of arrival/departure)

RAJBLK DB $1B, 'X', $1B, 'Y', '----------'

DB ' |', $0A, $0D

DB ' |', $0A, $0D

DB ' |', $0A, $0D

DB ' |', $0A, $0D

DB ' |', $0A, $0D

DB ' |', $0A, $0D

DB ' |', $0A, $0D

DB $0D

; Procedure: DISPLST
; Displays in the trains to stop at station block.

; which trains will visit the station.

; depends on the most significant bit of table value

; Procedure: CLRLST
; Sets up the train list in memory on page $2500-$250F
; one byte for each train.

; the Most Significant Bit signifies whether

; stop that train at the station or not

; 111111 - stop this train, 000000 - let this train pass

; Reserves $0500 - $055F for its use

; Uses

; ListLoc EQU $0500

CLRLST LDA $80

; default is all trains

; stop at the station

; Procedure: DISPLST
; Displays in the trains to stop at station block.

; which trains will visit the station.

; depends on the most significant bit of table value

; Procedure: DISPLST
; Displays in the trains to stop at station block.

; which trains will visit the station.

; depends on the most significant bit of table value

; Procedure: DISPLST
; Displays in the trains to stop at station block.

; which trains will visit the station.

; depends on the most significant bit of table value

; Procedure: DISPLST
; Displays in the trains to stop at station block.

; which trains will visit the station.

; depends on the most significant bit of table value
MAE 412

Programming

Robust Junction

: Assembles its message to send to the
  terminal on the $6600 page
MGSASM EQU $6600

DISPLAY LDX $500

: Copy the 4 x 4 grid of
  characters
NXTDS LDA BLANKL, X

: STG ASSEMBLY PAGE
REC DOUNES
INX
JMP NXTDS

DOUNES NOP

STARTS LDA LSTLOC, X

: If flag set go on to next
NEXTTR one

: If no flag set then
LDA 0

: Move space in
STG MGSASM, Y

: Message location over the
  Loco number to not stop

NEXTTR INX

CPS $110
BNE STARTS

LDA HIGH MGSASM

: Finally output
STG ASSEMBLY message to terminal
LDA #LOW MGSASM
STG MGSASM
JSR DISPMSG

RTS

: Table: LSTTAB

: The address refers to locomotive number.
  and the value
  refers to number of bytes from $6600 that
  that locomotive
  number is in the assembled message
LSTTAB

: for loco # 1 2 3
DB $04, $09, $0C, $0F

: for loco # 4 5 6 7
DB $14, $19, $1C, $1F

: for loco # 8 9 A B
DB $24, $29, $2C, $2F

: I think you can
DB $34, $39, $3C, $3F

: see the pattern
DB $00

: Message: BLANKL

: clears the trains to stop at station block
BLANKL

: DB $1B, 'X'
DB $1B, 'Y', ', ', 1, 2, 3', $0D, $0A, $0A
DB $1B, 'Y', ',', 4, 5, 6, 7', $0D, $0A, $0A
DB $1B, 'Y', ',', '8', '9', 'A', 'B', $0D, $0A, $0A
DB $1B, 'Y', ',', 'C', 'D', 'E', '
DB $00

: Procedure: NEWDAT

: a new console command has been issued so
  must be handled
  commands "> then numbers 1 9 correspond to
  taking train # 1 9
  on and off the train list, while
  commands "< then numbers 0 correspond to
  taking train # A F
  on and off the train list
  other commands to be assigned later
NEWDAT LDA RDATA

: store this new value in
  variable
STA ORDATA

: if its a <= 9 command
CHP #59

: deal with the a left nine
BEQ LEFT79

: if its a <= 8 command
CHP #18

: deal with that
BNE CHL6
JMP LEFT76

: if its a <= 7 command
CHP #27

: deal with that
BNE CHL6
JMP LEFT77

: if its a <= 6 command
CHP #86

: deal with that
BNE CONTOR
JMP LEFT76

: if its another command
CONTOR LDA ORDATA

: load which command was
  pressed to reset X flag
BPL LOWNTR

: if left arrow not pressed
AND #50P

: turn off the high bit
CLC

: add ten to get the high
  train numbers
ADC #0A

: LOWNTR
LDA LSTLOC, X

: load in train list byte
EOR $50

: if it was on list take
  off, and vice versa
STA LSTLOC, X

: update the train list on
JSR DISPLAY
  terminal

RTS

LEFTS NOP

: clear the train off of
  track 1

: update the train board to show track 1 is
  departing
LDA #$02

: status code equivalent to
BNE $10013000

: STA TSTTAT

: JSR UPPEJ

WAITNC LDA NTPAIN

: wait until the train on
  north track clears
JSR DISPMG

LDA #$5F

: STA NOVRD

: override the north track
  to occupied (I think)
STA #$10010000

: STA SWTDAT

: JSR SWTIT

: LDA #$10000000

: STA SWTDAT

: JSR SWTIT

: energize the track 1 section
LDA VIADAT

: bring PA to a low
EOR #$00000000

: STA VIADAT

: output this to the VTA

: now track 1 section energizes, and train begins
  to move away

: while the leaving train clears the north
  track optical sensor
WAITNSC LDA VZDAT

: check PS2 for a train passing the north track

: AND #$00000000

: makes other bits 0

: leaves bit 2 unchanged

: BNE WARTSC

: if bit 2 is low keep
  waiting

: now train has moved off track 1
LDA #$11000000

: LDA SWTIT

: switch main right switch

: to straight

: JSR SWTIT

: update the train board to reflect current
  conditions
LDA #$50

: status code referring to
  empty

: JSR UPPEJ

: wait for train to clear the train board
LDA #$58

: use a zero page location
  for another counter register
STA $0F

: LDX $50

: LDX $50

: DELAYL DEX

: BNE DELAYL

: DEC 0F

: BNE DELAYL

: turn the north track override off
LDA #$00

: STA NOVRD

: all done
RTS
MAE 412  Programming  Robust Junction

LEPT6  NOP  ;clear the train off of track 0
LEPT6  LDA #500  ;update the train board to show track 0 train is departing
LEPT6  STA TOSSTAT  ;LDA 0#502 code equivalent to departing
LEPT6  JSR UPRDJ

WAITND  LDA MTRAIN  ;wait until the train on north track clears
WAITND  STA NOVND  ;override the north track to occupied (I think)
WAITND  LDA #10000000  ;switch the station right to straight
WAITND  STA SWTIT  ;LDA #10000000 ;switch the main right to turnout
WAITND  JSR SWTIT

;energize the track 0 section
WAITND  LDA #11111110  ;bring PA0 to a low
WAITND  STA VIADAT  ;output this to the VIA
;now track 1 section energizes, and train begins to move away
;wait until the leaving train clears the north track optical sensor
WAITND  LDA #00000010  ;check PB2 for a train passing the north track optical
WAITND  STA #00000010  ;makes other bits 0 leaves bit 2 unchanged
WAITND  STA WAITND  ;if bit 2 is low keep waiting

;now train has moved off track 0
LDA #11000000  ;switch main right switch to straight
STA SWTIT  JSR SWTIT

;update the train board to reflect current conditions
LDA #500  ;status code referring to empty
LDA TOSSTAT  STA TOSFN  JSR UPRDJ

;wait for train to clear the train board
LDA #508  ;use a zero page location for another counter register
LDX #500  LDY #500

DELAY18  DEX  STA DELAY18  DEX  STA DELAY18  DEX  STA DELAY18

;turn the north track override off
LDA #500  STA NOVND

;all done
RTS

LEFT7  NOP  ;for use in future expansion
RTS

LEFT6  NOP  ;for use in future expansion
RTS

; Procedure : SWTSTAT
; Reads in from a zero page variable the direction of each switch
; and displays it on the terminal in the switch block
; USES : SWTSTP EQU #A0 ;zero page location of switch status byte
;       ;high is turnout, low is straight
; bit 7 is for Left Main

SHTINF  LDA #01000000  ;bring all bits except bit 7 to a low
SHTINF  LDA #01000000  ;if bit 7 is high, then left main turnout

LNS  LDA #HIGH LNMNG  ;if its low then display that
LNS  STA MESSG  STA MESOL  JSR DISPMG  JMP RDREC  ;switch go to next

LMT  LDA #HIGH LMTNSG  ;if its high then left main switch is turnout
LMT  STA MESSG  STA MESOL  JSR DISPMG

RMDEC  LDA SMTRIP  ;now make decision about right main switch
RMDEC  STA #01000000  ;bring all bits to zeros
RMDEC  STA #01000000  ;if its high then turnout branch to that
RMDEC  STA #01000000  ;if its high then
RMDEC  STA #01000000  ;straight write that message
RMDEC  STA MESSG  LDA #LOW RMSNG  STA MESOL  JSR DISPMG  JMP LDSD  ;done with that go
to decision for left station
RMDEC  LDA #HIGH RMTNG  ;if its high then
RMDEC  STA MESSG  LDA #LOW RMTNG  STA MESOL  JSR DISPMG

LSDEC  LDA SMTRIP  ;left station switch
LSDEC  LDA #00000000  ;bring all bits except bit 5 to a low
LSDEC  LDA #00000000  ;if bit 5 is high then left station is turnout

LSS  LDA #HIGH LSSNG  ;if its low then switch is straight
LSS  STA MESSG  STA MESOL  JSR DISPMG  JMP RDDEC  ;done here go to
decision for right station
LSS  LDA #HIGH LSSNG  ;if its low then
LSS  STA MESSG  LDA #LOW LSSNG  STA MESOL  JSR DISPMG

RSDEC  LDA SMTRIP  ;Make decision about the right station switch
RSDEC  LDA #00000000  ;bring all bits except for bit 4 to a low
RSDEC  STA MESSG  LDA #LOW RSMSG  STA MESOL  JSR DISPMG  JMP RDST  ;return from subroutine, done displaying

RSDEC  LDA #HIGH RMTNG  ;switch is turnout
RSDEC  STA MESSG  LDA #LOW RMTNG  STA MESOL  JSR DISPMG  RTS  ;done return from

MESSAGES RM, LM, RS, LS. - 5 or T - MSG

3:32 PM  35  May 23, 1997  399
Programming

Robust Junction

first two letters correspond to which switch (L left, R right
K - main line, S - station line) then the next letter corresponds
to the switches state (S - straight, T for turnout)

; RENMSG DB $1B, 'X5:', $1B, 'YB': line 22, column 35
DB 'STR8': DB $0C
; RMNMSG DB $1B, 'X5:', $1B, 'YB': line 22, column 35
DB 'STR8': DB $0C
; LSMSG DB $1B, 'X5:', $1B, 'YB': line 22, column 29
DB 'STR8': DB $0C
; LMNMSG DB $1B, 'X5:', $1B, 'YB': line 22, column 29
DB 'STR8': DB $0C
; Procedure : SWIT
; Uses the information passed to it in
SWDAT, to switch the
desired switch and sends a message to
the display on terminal

; USES:
SWDAT EQU $A1 :zero page location for data
indicating which switch
and which direction to switch
; bits 5-0 should always be high
; bits 6, 7 should always be low
; bits 6, 7, 8 correspond to which switch
; and its direction; (5-0) corresponding to following table
; RST: Right Main Turnout
; LST: Left Main Turnout
; RSP: Right Station Turnout
; LSP: Left Station Turnout
; RSM: Right Main Straight
; LSM: Left Main Straight
; RMS: Right Station Straight
; LMS: Left Station Straight

; Table : SMTAB
; The address corresponds to the 4 most
; significant bits of the SWDAT
; value, and the data corresponds to a 0 in
; the desired bit location
; and ones everywhere else

SMTAB DB $11111111 :0000 :addresses with
bit 3 low not used (SFM)
DB $11111111 :0001
DB $11111111 :0010
DB $11111111 :0011
DB $11111111 :0100
DB $11111111 :0101
DB $11111111 :0110
DB $11111111 :0111
DB $11111111 :1000
DB $11111111 :1001
DB $11111111 :1010
DB $11111111 :1011
DB $11111111 :1100
DB $11111111 :1101
DB $11111111 :1110
DB $11111111 :1111

ANDVAL EQU $A2 :two temporary variables

INVVAL EQU $A3 :used in updating terminal

SWIT NP

SMTWXY LDA VIABUF on VIA
AND #40000111 :erase the SI-S4 bits to
zeros
EOR SWDAT :now set the SI-S4 bits to
proper settings
STA VIABUF STA VIAJAT :output these in buffer
; S4 is high so no switching
yet!

; set up Y register as number of frame 1's to
; switch through
LDY $502
; activate the switch coil
; for 2 consecutive frame 1's
NUNFIS DEY

WAIT1 LDA FRANUM :wait for frame number 1
to come across on NUNFIS
CMP $401
BNE WAIT1
LDA VIABUF :bring the S4 line to a
; low (start switching switch)
STA VIAJAT :output to the VIA (switch
cell is energized)

WAIT2 LDA FRANUM :wait for frame 2 to come
; across
CMP #502
BNE WAIT2
LDA VIABUF :Stop energizing the
; switch coil
STA VIAJAT

; test to see if done
BNE NUNFIS

; Now time to update
; switch status board on
; terminal
; transfer the switch data
to X
; move the SI-S4 bits to
; the least significant bit
; push this in a temporary
; variable
; invert this and store in
; a temporary variable
; load in state of switches
; before this switch
; bring the bit corresponding to
; switch to low
; store back in switch
; buffer
; loads in S3 (switched
to V status register)
; if overflow is set
; (switch is straight update
; terminal)
; if overflow was clear
; (switch is turnout)
; change the bit on the
; SNFBUF variable
; update it

DOIT JSR SWITAT :update the terminal's
; switch status display

RTS

; Procedure : INITWT
; sets all switches to straight (no
; turnouts) and switches them

; Procedure : INITWT
; sets all switches to straight (no
; turnouts) and switches them

; Procedure : INITWT
; sets all switches to straight (no
; turnouts) and switches them

3:32 PM 36 May 23, 1997 400
STA SOTDAT
JSR SWIT
LDA #11110000 ;switch station left
STA SOTDAT
JSR SWIT
RTS ;:all switches straight

; Procedure : FCLA
; Forces the North Track to a clear state
; will be used when train captured and
; safely stopped at station
; then the north track will be reopened for
; upcoming traffic
;
; FCLA
LDA VIA100
AND #$7F
STA VIA100

EOR #$08
STA VIA100

;:load in current VIA
;:states
;:brings bit 3 to 0, (the
;:force track clear bit)
;:brings bit 3 to 1,
;:leaving other bits the same
;:output these values to
;:the VIA (forces track clear)
;:send this force clear as a
;:pulse
;:256 cycles long

PAUSE DEX
BNE PAUSE

LDA VIA100
;:load back in same bits,
;
;:now with force clear low
;
;:output these normal
;
;:values to VIA

RTS

; Procedure : UPREJ
;
; Reads in 4 bytes of the zero page # $80 -
; $83
; updates the track board on the terminal
; according to their values
;
; USES
; TOTN EQU $80
; TOSTAT EQU $81
; TITN EQU $82
; TISTAT EQU $83
;
; RTXT EQU $8000
; RTXTNM EQU $8006
; RTXTST EQU $800A
;
; UPREJ
LDA #000
STA TOTN

LDA #000
STA TOSTAT

LDA #11110000
STA TITN

LDA #11110000
STA TISTAT

RTS

;:check if its ARRIVING
BNE NOT1
;if not try another one

LDA #HIGH ARRMGS
STA MESGH
LDA LOW ARRMGS
STA MESGL
JSR DISPMSG

JSR DTP

NOT1


;:check if its DEPARTING
BNE NOT3
;if not try another one

LDA #HIGH DEMPSG
STA MESGH
LDA LOW DEMPSG
STA MESGL
JSR DISPMSG

JSR DTP

NOT3

;:Must be STOPPED
LDA #HIGH STPMSG
LDA LOW STPMSG
STA MESGL
JSR DISPMSG


;:now do the same stuff but for the track no 1 line

LDA #000
;:reset counter and copy
;
;:train number message to RTXT
;
;:then stop copying
;
;NEXTT1
LDA TIMS2.X
BEQ DONET1
;:if reach end of message

LDA #000
;:then stop copying

INDEX
JMP NEXTT1

DONET1

LDA TITN
;:load in the train number of
;
;:train on track 1
;
;:get the ASCII character
;
;:corresponding to that train
;
;:in proper location
;
;:store that character in
;
;:a temporary location
;
;:and position for the
;
;:train number
;
;:display the train number on
;
;:terminal
;
;:position for the
;
;:status string
;
;:load in the status of the
;
;:train on track 1
;
;:CHECK if its EMPTY
;
;:if not then try another
;
;:one

LDA #HIGH MNMSG
STA LOW MNMSG
STA MESGL
JSR DISPMSG

JSR DTP

NT1

CME #101
;:check if its ARRIVING
BNE NT2
;if not try another one

LDA #HIGH ARRMGS
STA MESGH
LDA LOW ARRMGS
STA MESGL
JSR DISPMSG

JSR DTP

NT2

CME #102
;:check if its DEPARTING
BNE NT3
;if not try another one

LDA #HIGH DEMPSG
STA MESGH
LDA LOW DEMPSG
STA MESGL
JSR DISPMSG

JSR DTP

NT3

;:Must be STOPPED
LDA #HIGH STPMSG
LDA LOW STPMSG
STA MESGL
JSR DISPMSG

DSAT

;:but for now we'll just
;
;:end it here

3:32 PM 37 May 23, 1997 401
age: TOMSG
used to position cursor in the right spot
for the train number.
and then the status (whichever it may be)
TOMSG DB $1B, 'X4', $1B, 'Y'/
    at line 7, col. 16 ;position
    ;train
DB 'A'
DB $1B, 'Y6'
number goes here ;position
at col. 23 now ;status
DB $00
    string goes here

; Message: TINSG
; used to position cursor in the right spot
; for the train number.
; and then the status (whichever it may be)
TINSG DB $1B, 'X*', $1B, 'Y'/
    at line 11, col. 16 ;position
    ;train
DB 'A'
DB $1B, 'Y6'
number goes here ;position
at col. 23 now ;status
DB $00
    string goes here

; Message: MTMSG
; status string corresponding to track empty
MTMSG DB 'EMPTY'
DB $00

; Message: ARRMSG
; status string corresponding to train
; arriving
ARRMSG DB 'ARRIVING'
DB $00

; Message: DEPMSG
; status string corresponding to train
; departing
DEPMSG DB 'DEPARTING'
DB $00

; Message: STPMSG
; status string corresponding to train
; stopped at track
STPMSG DB 'STOPPED'
DB $00

; Procedure: INITCLO
; gets the T1 timer on VIA #2 going counting
down from $FFFF
; sets up the clock display on the terminal
; sets up certain variables in memory
; corresponding to the time
; and initializes current station time to
; zero hours, zero minutes, zero seconds
; past the origin
INITCLO NOP  ;will be added if there is time
RTS

; Procedure: UPLCLO
; checks the T1 counter on VIA #2, and
; updates the clock accordingly. Should be called every $FFFF
; microseconds
UPLCLO NOP
RTS

END
Circuit Diagrams

The following circuit diagrams were done in Aldus SuperPaint for the Macintosh. No disrespect was intended for GEDIT, but the author's learning curve in GEDIT was not progressing as fastly as he would have liked, so since the opportunity cost of using SuperPaint was much lower, that's what was used.

List of Circuits:

TRACK 0 & 1 OPTICAL SENSOR AND RS FLIP FLOP
TRACK 0 & 1 RELAY KILL
UART DAUGHTER BOARD
SWITCH ACTUATOR CIRCUIT
NORTH OPTICAL SENSOR
Track 0 & 1 Optical Sensor and RS Flip Flop

- Turn on TK1
- To 5V

TL139
0.024K

Flip Flop

- To 5V
- To 5V

Track 0 & 1 Optical Sensor and RS Flip Flop Circuit

EPC % # - pin # on the 44 pin edge connector
Track 0 & 1 Relay Kill

Track 0 & 1 Relay Kill circuit

Kill Logic (Track 1)
from Sensor Circuit

Kill Logic (Track 0)
from Sensor Circuit

Switched Power
to Track 1

Switched Power
North Optical Sensor
Bradley Mendelson and Lina Jin
MAE 412
Professor M. Littman
May, 1996

'This paper was written in accordance with University Regulations.'

Bradley Mendelson