```
        JSR SWITCH3     ;PULSES SWITCH
        RTS

SW3T    LDA #REL2
        EOR #$FF        ;"NOT"
        AND $0610       ;AND WITH A PORT OF VIA AT A000 MAILBOX
        JSR PUTAA2
        JSR SWITCH3
        RTS

SW4F    LDA $0610       ;LOAD A PORT OF VIA AT A000 MAILBOX
        ORA #REL2       ;OR WITH RELAY BITS
        JSR PUTAA2      ;PLACES ACCUMULATOR INTO VIA AND MAILBOX
        JSR SWITCH4     ;PULSES SWITCH
        RTS

SW4T    LDA #REL2
        EOR #$FF        ;"NOT"
        AND $0610       ;AND WITH A PORT OF VIA AT A000 MAILBOX
        JSR PUTAA2
        JSR SWITCH4
        RTS

SWITCH1 JSR FRAM1
        LDA #SW1        ;SWITCH 1
        JSR PUTAA       ;PLACES ACCUMULATOR INTO VIA AND MAILBOX
        JSR NEXT        ;WAITS 3 FRAMES
        JSR NEXT
        LDA #SW1        ;SWITCHES BACK OFF
        JSR PUTAA       ;
        RTS

SWITCH2 JSR FRAM1
        LDA #SW2        ;SWITCH 2
        JSR PUTAA       ;PLACES ACCUMULATOR INTO VIA AND MAILBOX
        JSR NEXT
        JSR NEXT
        LDA #SW2        ;SWITCHES BACK OFF
        JSR PUTAA       ;
        RTS

SWITCH3 JSR FRAM1
        LDA #SW3        ;SWITCH 3
        JSR PUTAA       ;PLACES ACCUMULATOR INTO VIA AND MAILBOX
        JSR NEXT
        JSR NEXT
        LDA #SW3        ;SWITCHES BACK OFF
        JSR PUTAA       ;
        RTS

SWITCH4 JSR FRAM1
        LDA #SW4        ;SWITCH 4
        JSR PUTAA       ;PLACES ACCUMULATOR INTO VIA AND MAILBOX
        JSR NEXT
        JSR NEXT
        LDA #SW4        ;SWITCHES BACK OFF
        JSR PUTAA       ;
        RTS

;*********************************************
;PUTAA SUBROUTINE PLACES ACCUMULATOR IN A PORT
;OF A000 VIA
;*********************************************
PUTAA   EOR $0610
        STA $A001
        STA $0610
        RTS

PUTAA2  STA $A001       ;THIS SUBROUTINE DOES NOT DO XORING.  ITS USED
        STA $0610       ;FOR THE RELAYS
        RTS


;************************
;BAR CODE READER
;************************
BARCODE JSR NORM
        JSR SOUTH0      ;WAIT FOR SOUTH TRACK TO BE CLEAR

        LDA #$FF
        STA SOVRD       ;KILL SOUTH RELAY TRACK

        LDA #$F0
        STA $4000       ;DISPLAY 'F0' FOR FORWARD SLOW

        JSR GETHF2      ;WAIT FOR 1-2-RIGHT

        LDA #$00
        STA SOVRD       ;REPOWER SOUTH RELAY TRACK

        JSR SOUTH1      ;WAIT FOR SOUTH TRACK TO HAVE A TRAIN

        LDA #$00
        STA LCAR        ;INITIALIZE LAST CAR MAILBOX TO 00

        LDA #CLEAR
        EOR #$FF
        STA $8001       ;STROBE CLEAR BIT ON SHIFT REGISTER

        LDA #$FF
        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 STRAIN
        BNE READ        ;IF TRAIN HASN'T PASSED SOUTH CLEAR, KEEP CHECKING
                        ;FOR MORE CARS
        RTS

;************************
;SHIFT: READ SHIFT REGISTER
;************************
SHIFT   LDA $8001
        STA VAR3        ;LOAD VIA INTO A VARIABLE

        ROR A           ;ROTATE ONE BIT TO THE RIGHT
        AND #BBITS      ;KEEP ONLY TWO LEFTMOST BITS
        STA $4000       ;DISPLAY CAR NO.
        STA LCAR        ;STORE CAR NO. IN VARIABLE LCAR

        RTS

        END
```

Robust Junction
MAE 412

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

INTRODUCTION 3

DESCRIPTION OF TRACK LAYOUT 4

SENSOR DESCRIPTION 5

ACTUATOR DESCRIPTION 7

TERMINAL INTERFACE 9

COMPUTER INTERFACE 10

PAL 16L8 12

PROGRAM EXPLANATION 13

Program Flow Charts 14

Program Listing 29

CIRCUIT DIAGRAMS 39

Track 0 & 1 optical sensor and RS flip flop 40
Track 0 & 1 Relay Kill 41
UAR/T Daughter Board 42
Switch Actuator Circuit 43
North Optical Sensor 44

. Very creative
. UART/excellent
. execution excel
. Project is robust

(A)

Introduction

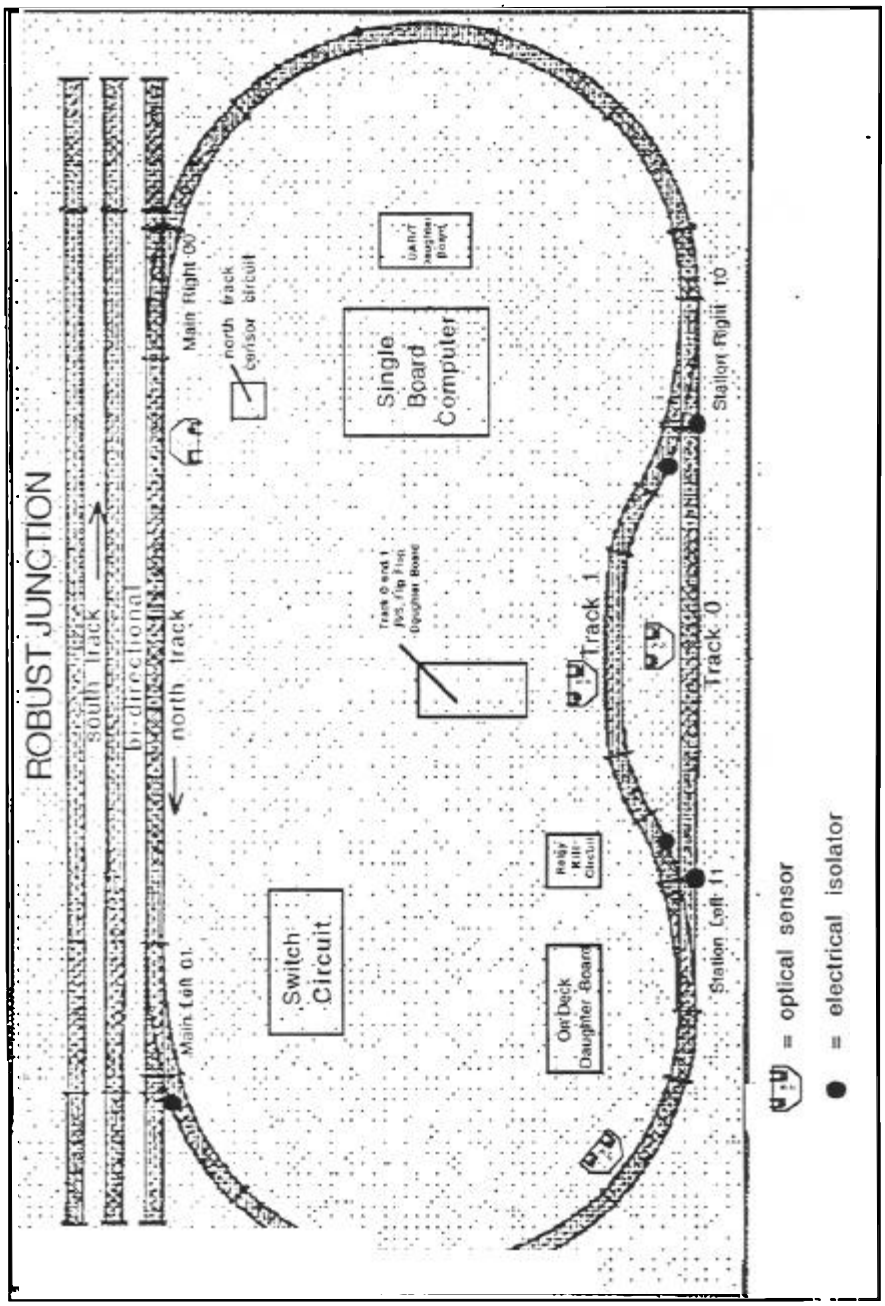
This project uses a single board computer (SBC) with a 6502 microprocessor to control the operations of as many as 15 trains on a track given power by a Hornby master controller unit. The features that are unique to this project are four optical sensors, four track switches, and a UAR/T receiver transmitter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

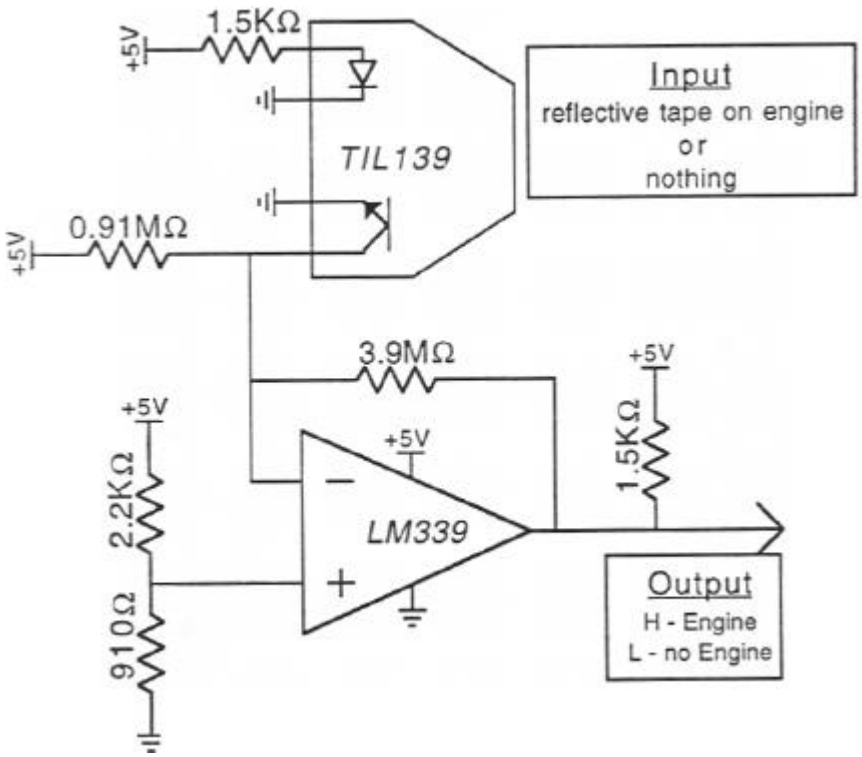
Description of Track Layout



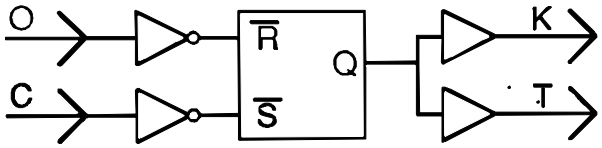
Sensor Description

Inside the TIL139 the infrared light emitting diode is always on. When the reflective tape of an engine passes infront 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 passsed 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 Ciruit Diagram



Basic RS Flip Flop Ciruit Diagram



Inputs

0 - from optical sensor
(H - engine present)
(L - no engine)
C - from computer
(H - track on)
(L - optical sensor decides)

Outputs

K - to track kill circuit
T - to computer
(H - track on)
(L - track off)

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 loaction atop the bar code of the engine in accordance with the labelling 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.

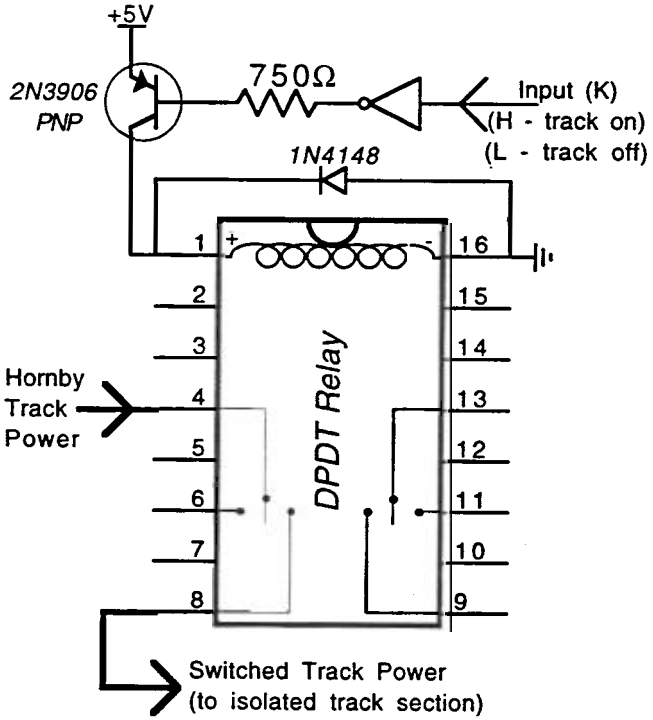
The flip-flop is initialized to be on (track on). Also, the computer is set to a low and it is assumed there are no engines on the track. Thus the optical sensor is a low. This is the quiescient state and the track remains on, until the RESET line is brought high which is the case when the optical sensor 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

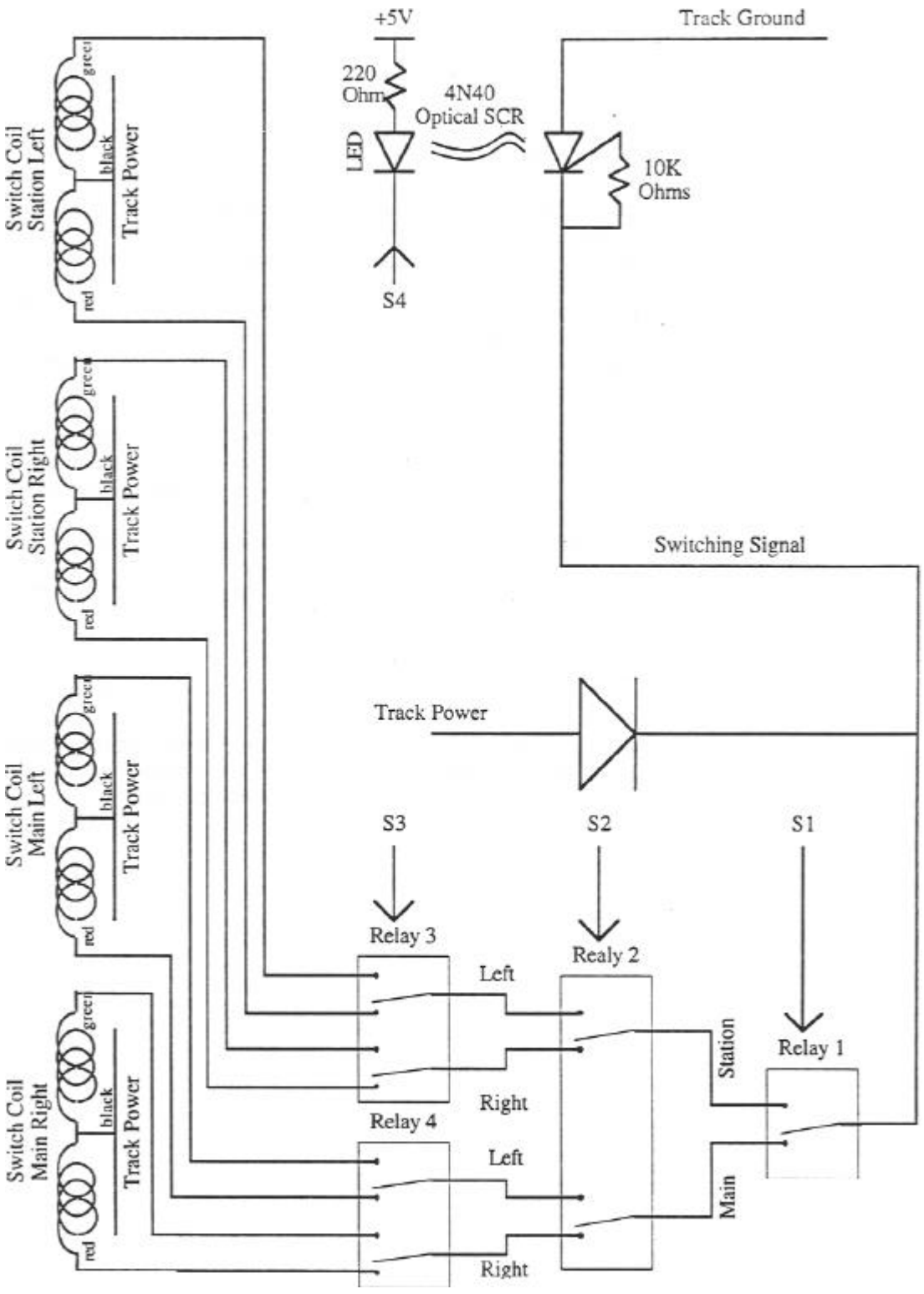
There are two types of actuation that are used in this project: track kills (and restarts) and track switching. Both involve relays. The diagram below shows the basic method of track kill.

Basic Track Kill Ciruit 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.


+5V
Track Ground
220 Ohm
4N40 Optical SCR
LED
10K Ohms
S4
Switch Coil Station Left
Switch Coil Station Right
Switch Coil Main Left
Switch Coil Main Right
green
black
red
Track Power
Switching Signal
Track Power
S3
S2
S1
Relay 3
Realy 2
Relay 1
Relay 4
Left
Right
Station
Main

Terminal Interface

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

The Robust Junction UAR/T is only set up to transmit data, which is all that is required of a train board. This simplifies the circuit signifigantly. The UAR/T 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 UAR/T requires a clock signal at sixteen times the desired transmitting BAUD rate.

The line driver takes the serial TTL (+5V, ground) signal from the UAR/T 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 UAR/T 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 :

Switch #	1	2	3	4	5	6	7	8	9	line button
Setting	off	on	off	off	off	ON	ON	off	ON	depressed

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.

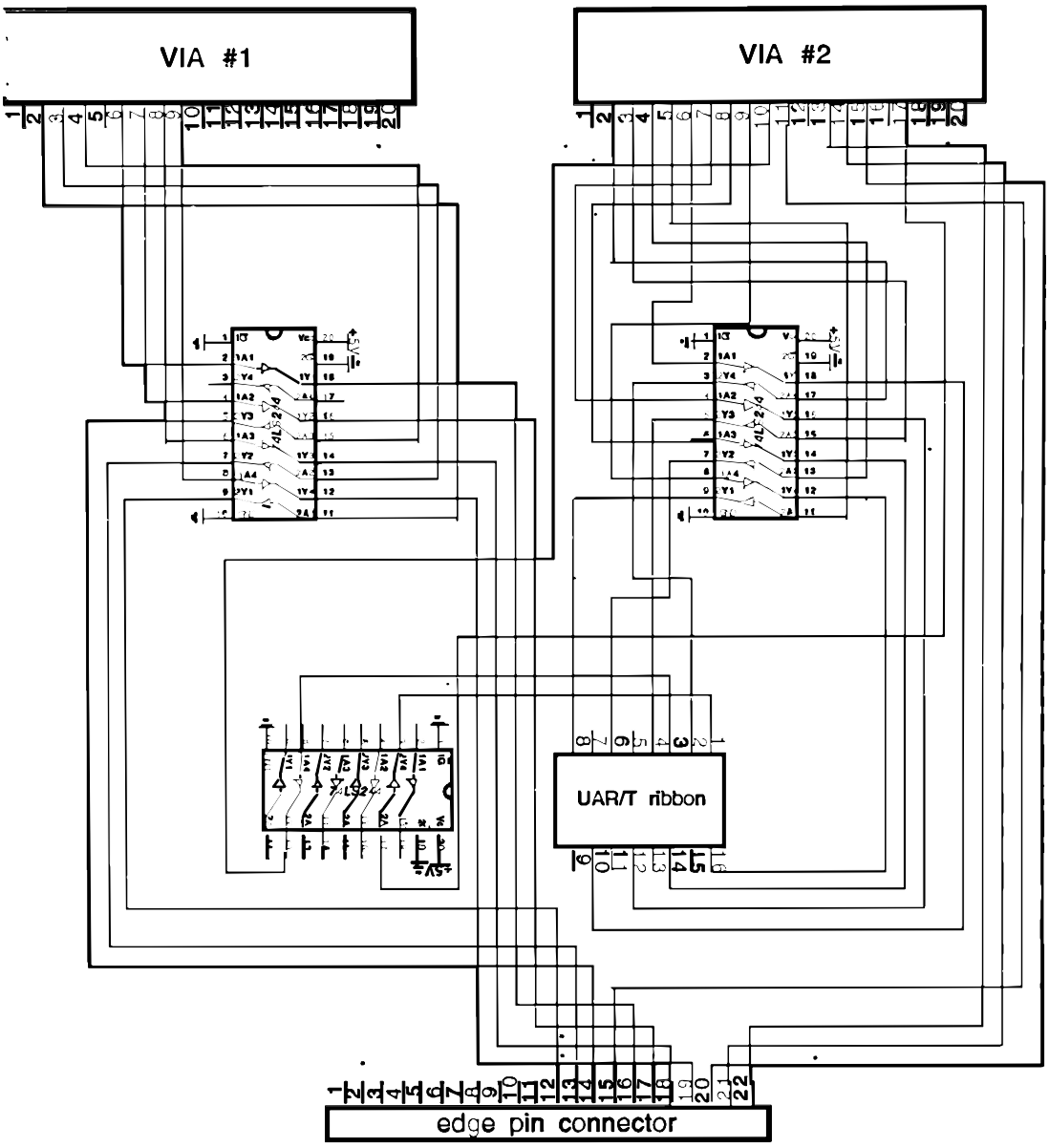
Computer Interface

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.

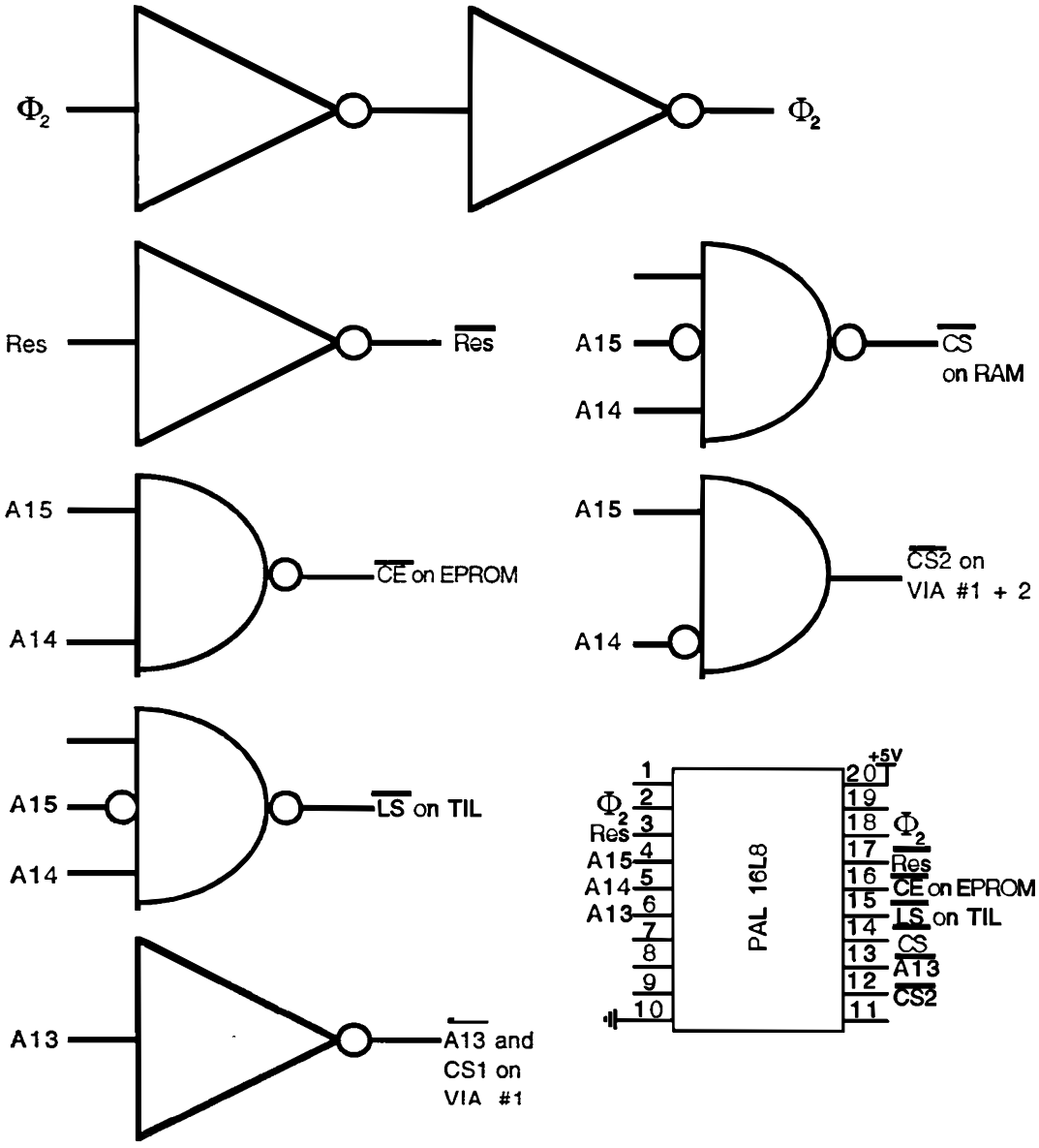
Pin#	Color	Description	VIA port
1	black	logic ground	GND
2	red	logic power	+5V
3			
4	black	test stand	1,PA
5	white	test stand	
6	yellow	test stand	
7	blue	test stand	
8	test stand	test stand	
9	green	test stand	
10	brown	test stand	
11	red	test stand	
12	grey	track 1 and 0 optical sensor circuit	1,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

UAR /T #	Description	VIA port
1	see UAR/T diagram	2, PB0
2		2, PA0
3		2, PB7
4		2, PA1
6		2, PA2
8		2, PA3
10		2, PA4
12		2, PA5
14		2, PA6
16		2, PA7

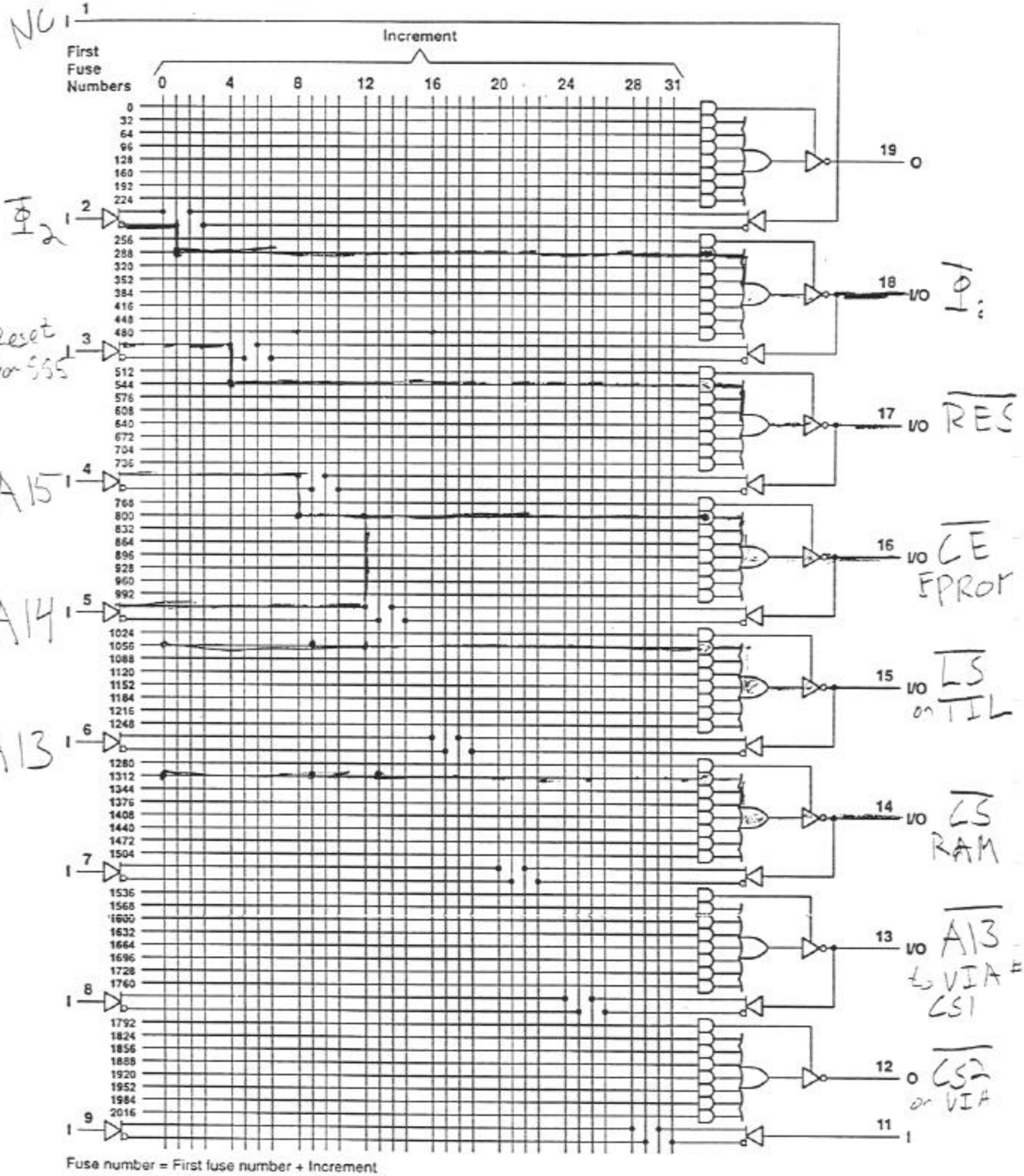

VIA #1
VIA #2
UART ribbon
edge pin connector

PAL 16L8

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 (SWTIT), update various aspects of the terminal (UPRBJ - 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 (INITSWT).

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

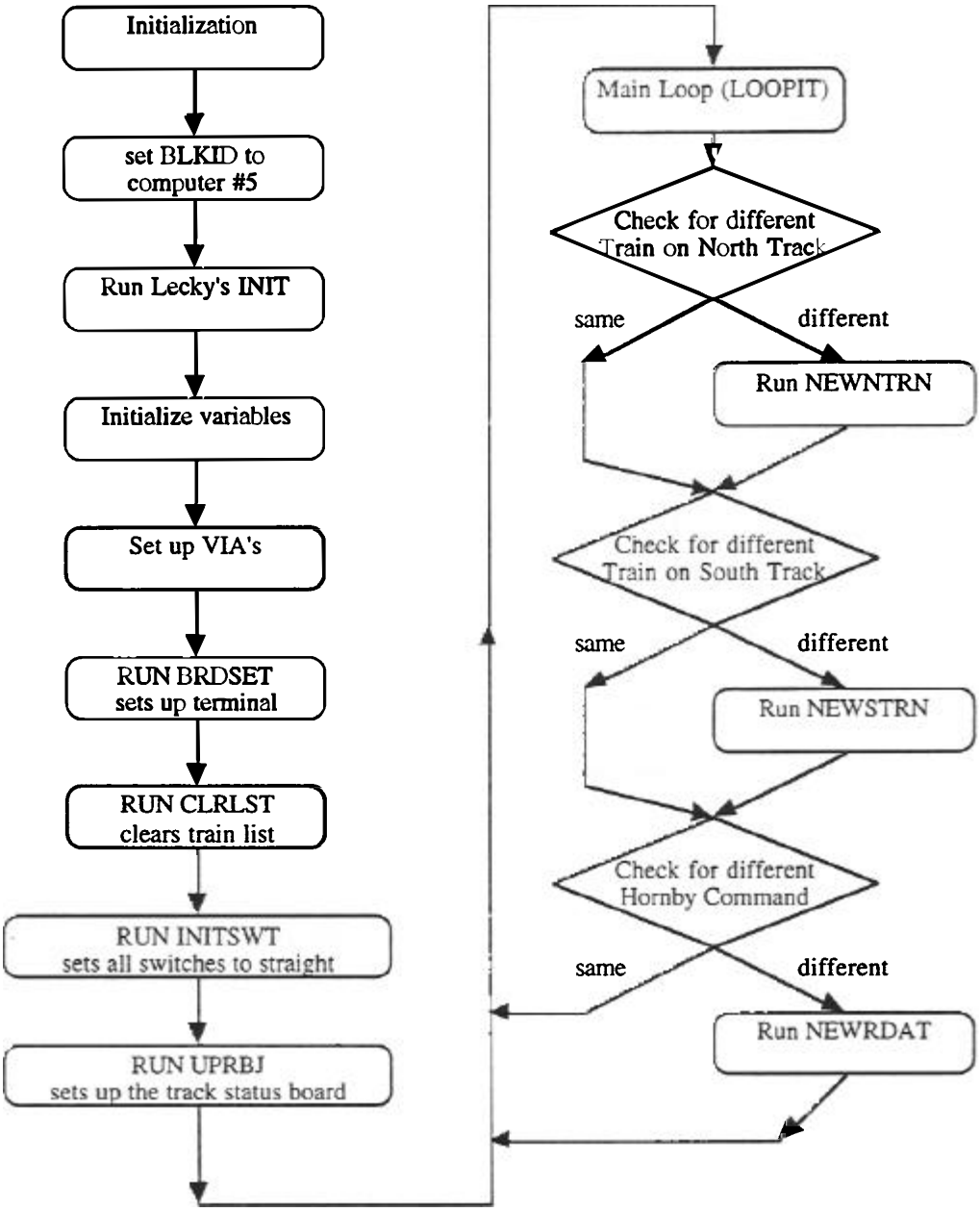
The flow charts follow the standard of a round rectangle for an operation, and a parallelogram for a decision. Frequently the word RUN is used. This refers to a jump subroutine command in assembly langauge. Upon completion of the subroutine the program will return to the line right after the RUN block.

List of Flow Charts

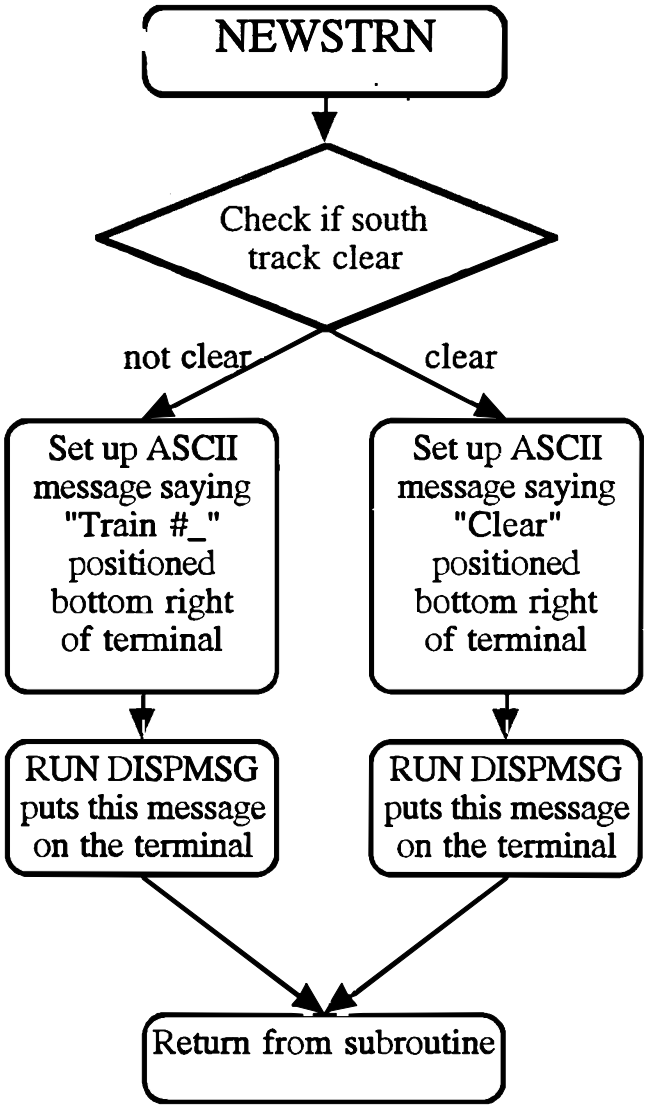
MAIN LOOP	15
NEWSTRN	16
NEWNTRN	17
STOPIT	18
BRDSET	20
DISPMSG	21
CLRLST	22
FCLR	22
DISPLST	23
NEWRDAT	24
SWTSTAT	25
SWTIT	26
INITSWT	27
UPRBJ	28

MAIN Loop

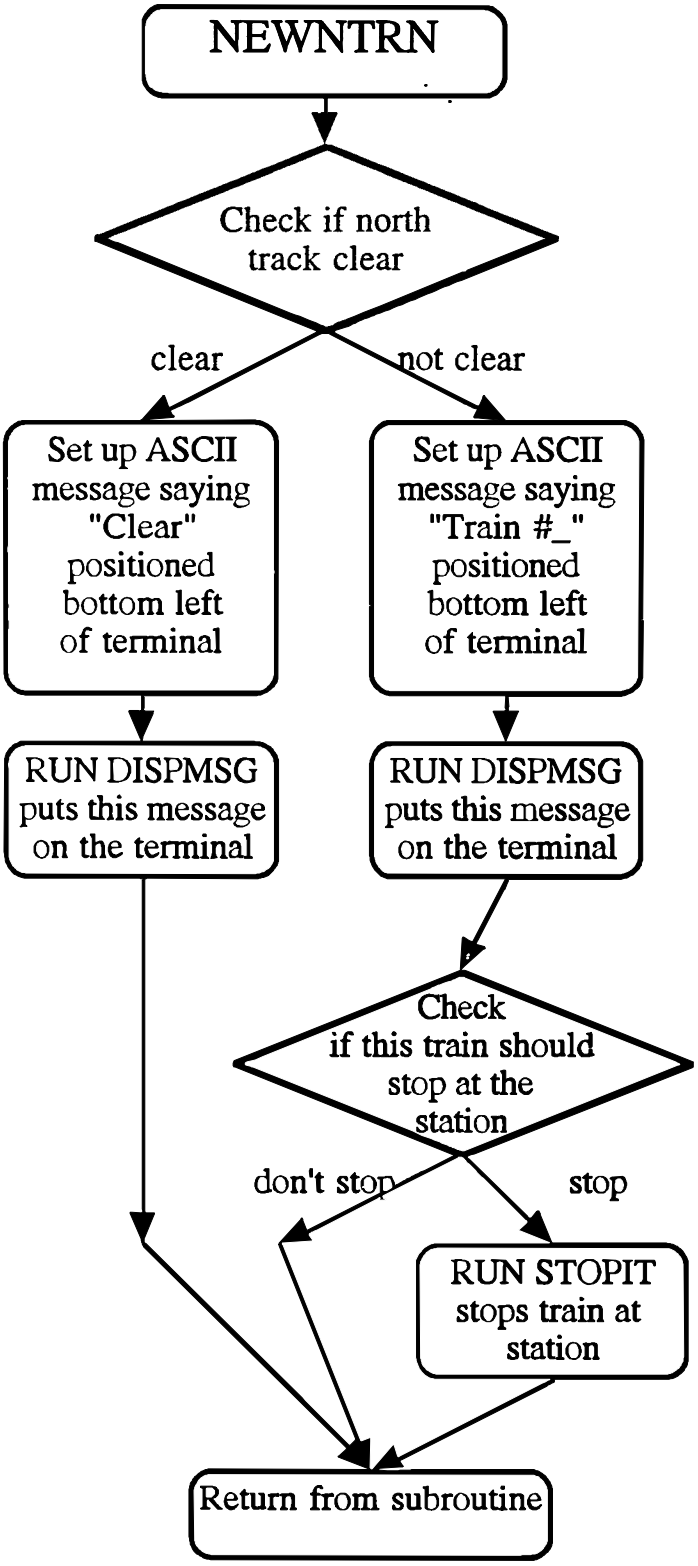
Main Program



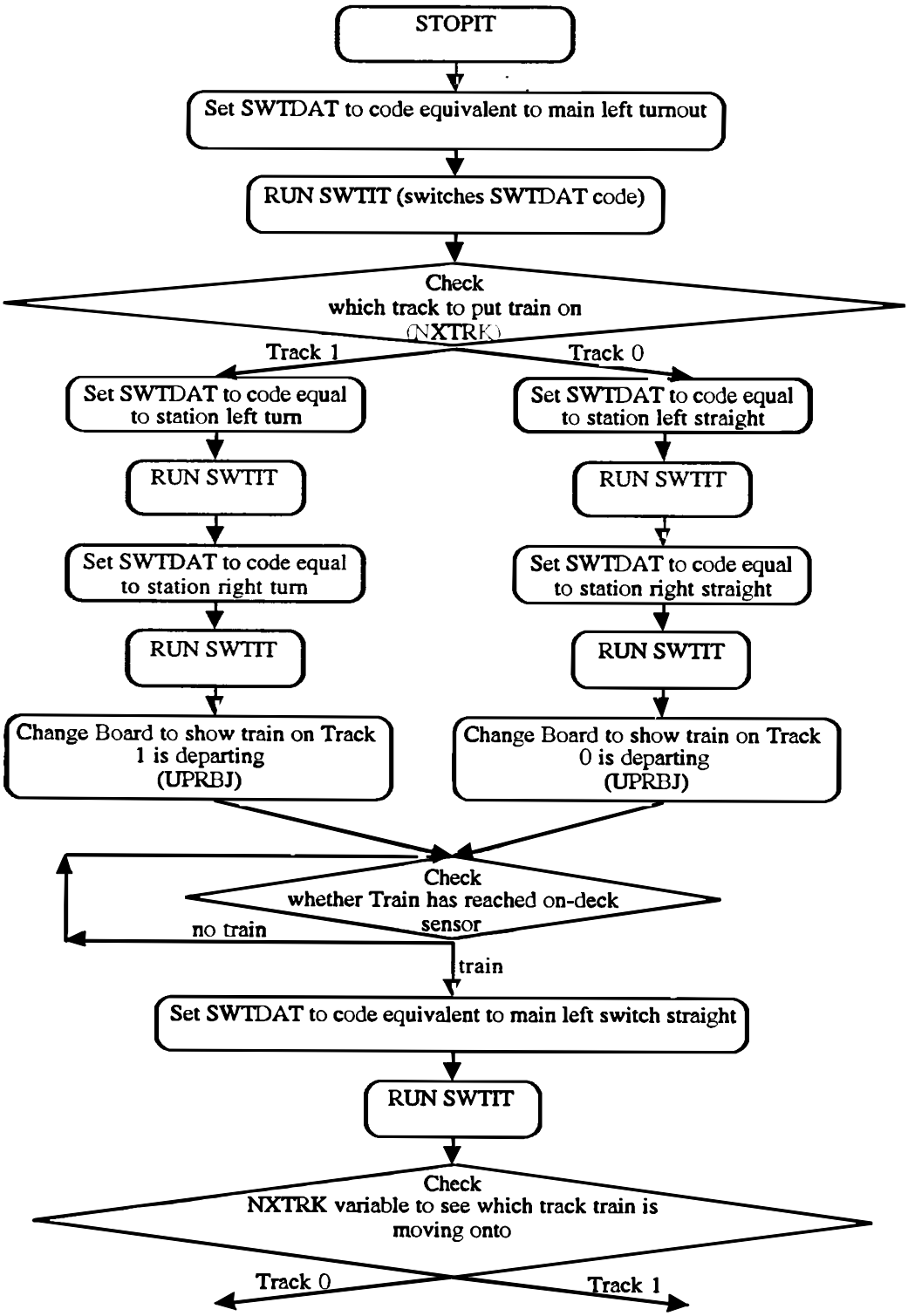
NEWSTRN

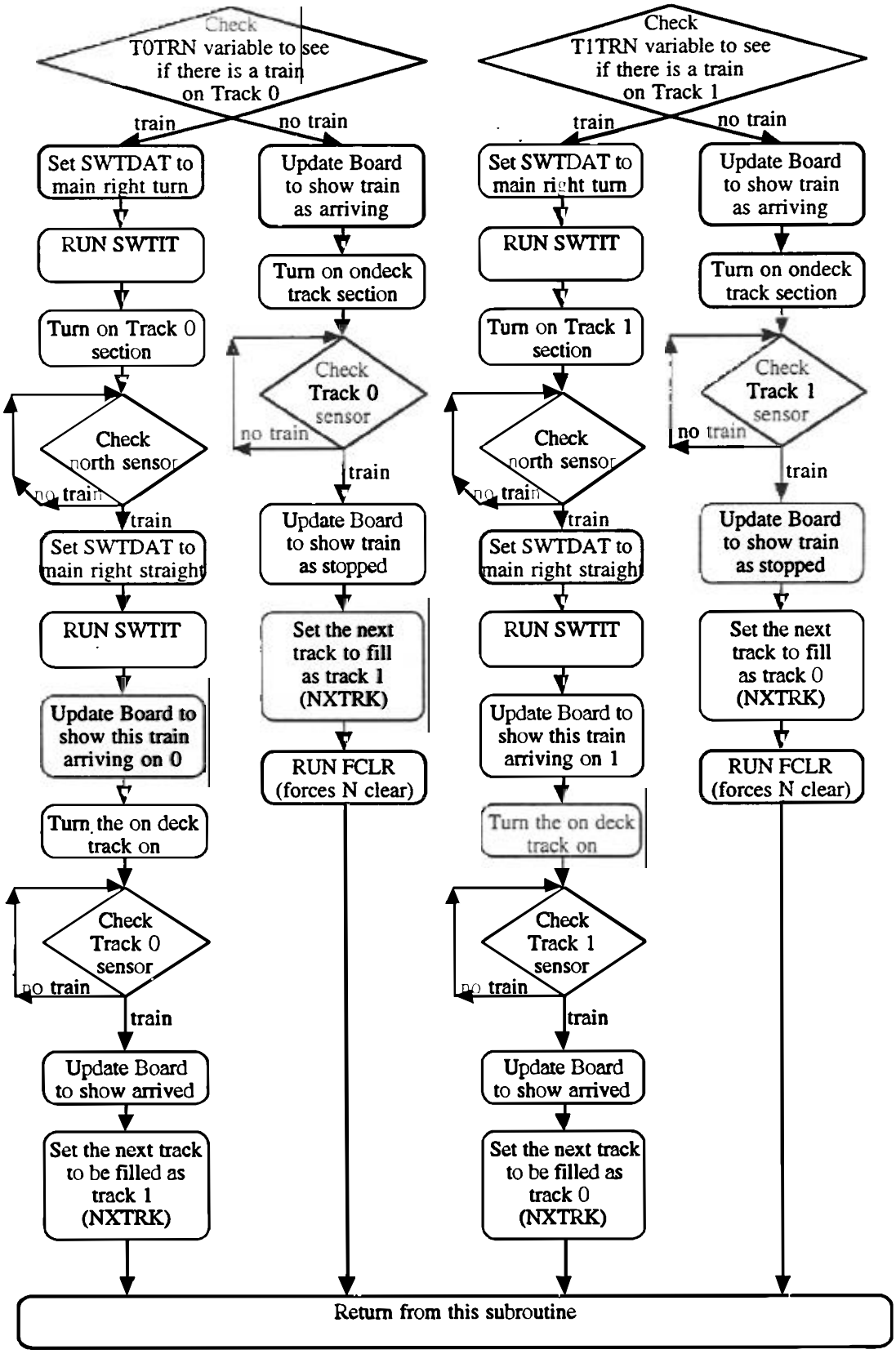

NEWSTRN
Check if south track clear
not clear
clear
Set up ASCII message saying "Train #_" positioned bottom right of terminal
Set up ASCII message saying "Clear" positioned bottom right of terminal
RUN DISPMSG puts this message on the terminal
RUN DISPMSG puts this message on the terminal
Return from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

NEWNTRN

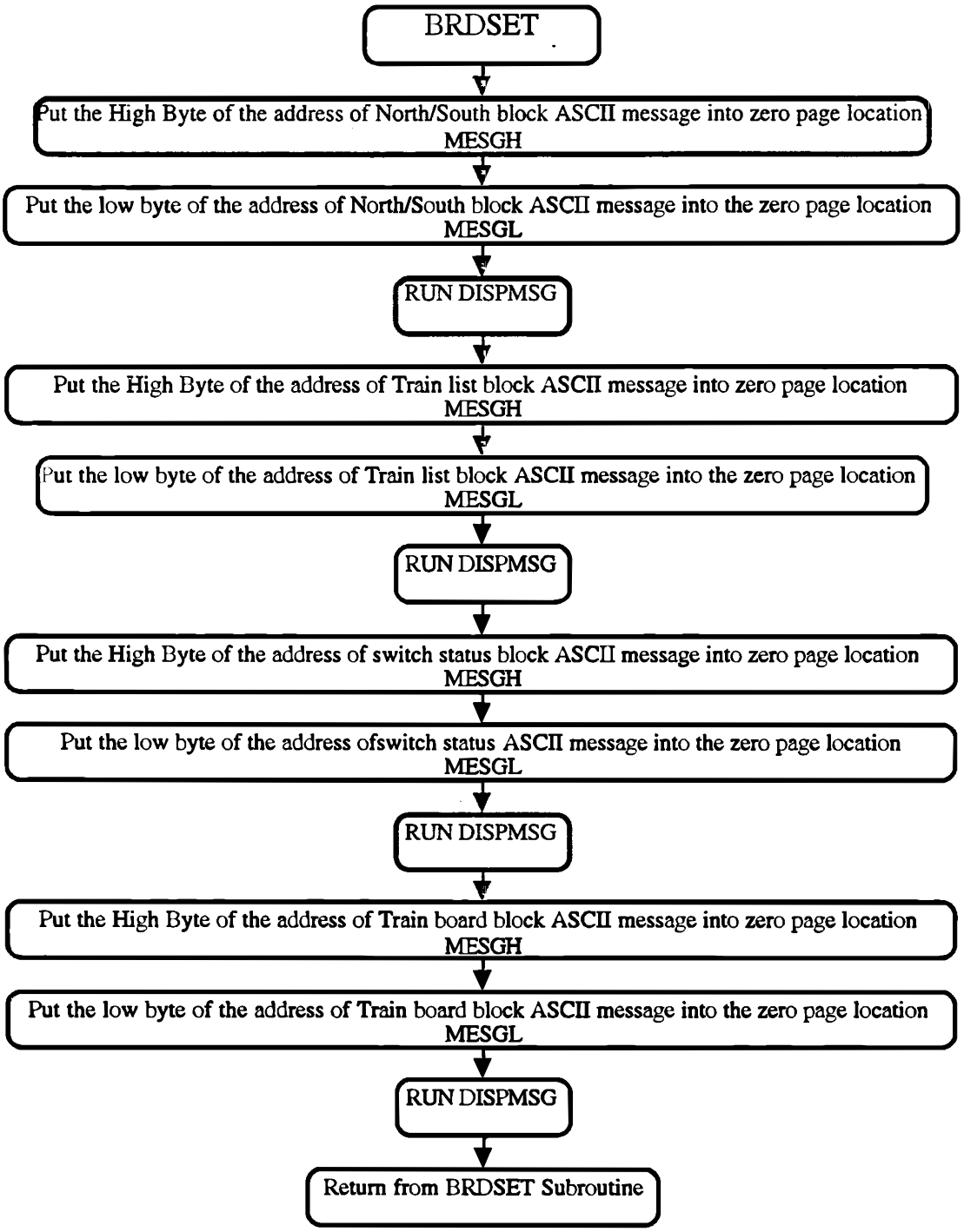

NEWNTRN
Check if north track clear
clear
not clear
Set up ASCII message saying "Clear" positioned bottom left of terminal
Set up ASCII message saying "Train #_" positioned bottom left of terminal
RUN DISPMSG puts this message on the terminal
RUN DISPMSG puts this message on the terminal
Check if this train should stop at the station
don't stop
stop
RUN STOPIT stops train at station
Return from subroutine

STOPIT

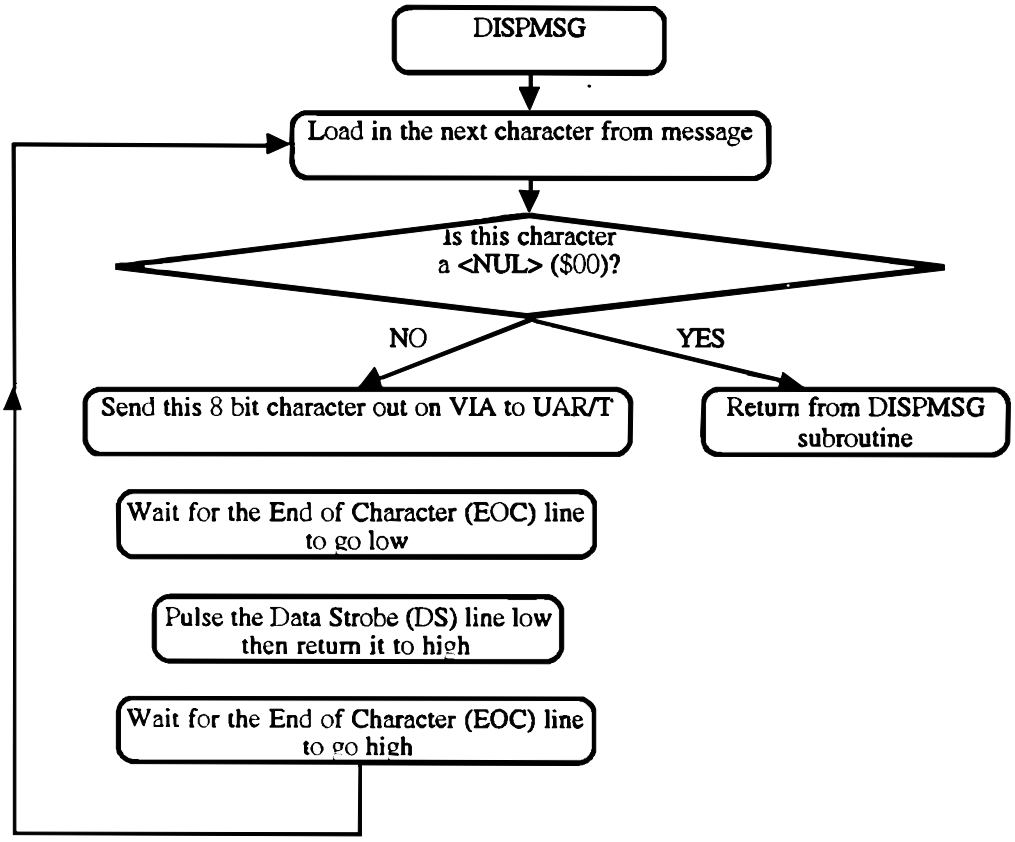

STOPIT
Set SWTDAT to code equivalent to main left turnout
RUN SWTIT (switches SWTDAT code)
Check
which track to put train on
(NXTRK)
Track 1
Track 0
Set SWTDAT to code equal to station left turn
RUN SWTIT
Set SWTDAT to code equal to station right turn
RUN SWTIT
Change Board to show train on Track 1 is departing (UPRBJ)
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
train
Set SWTDAT to code equivalent to main left switch straight
RUN SWTIT
Check
NXTRK variable to see which track train is moving onto
Track 0
Track 1


Check T0TRN variable to see if there is a train on Track 0
train
no train
Set SWTDAT to main right turn
RUN SWTIT
Turn on Track 0 section
Check north sensor
no train
train
Set SWTDAT to main right straight
RUN SWTIT
Update Board to show this train arriving on 0
Turn the on deck track on
Check Track 0 sensor
no train
train
Update Board to show arrived
Set the next track to be filled as track 1 (NXTRK)
Update Board to show train as arriving
Turn on ondeck track section
Check Track 0 sensor
no train
train
Update Board to show train as stopped
Set the next track to fill as track 1 (NXTRK)
RUN FCLR (forces N clear)
Check T1TRN variable to see if there is a train on Track 1
train
no train
Set SWTDAT to main right turn
RUN SWTIT
Turn on Track 1 section
Check north sensor
no train
train
Set SWTDAT to main right straight
RUN SWTIT
Update Board to show this train arriving on 1
Turn the on deck track on
Check Track 1 sensor
no train
train
Update Board to show arrived
Set the next track to be filled as track 0 (NXTRK)
Update Board to show train as arriving
Turn on ondeck track section
Check Track 1 sensor
no train
train
Update Board to show train as stopped
Set the next track to fill as track 0 (NXTRK)
RUN FCLR (forces N clear)
Return from this subroutine

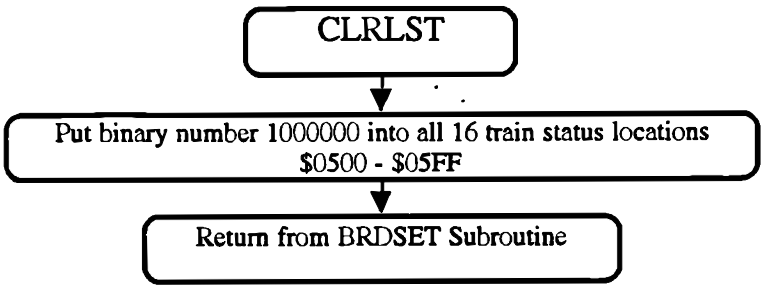
BRDSET


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 DISPMSG
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 DISPMSG
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 ofswitch status ASCII message into the zero page location MESGL
RUN DISPMSG
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 DISPMSG
Return from BRDSET Subroutine

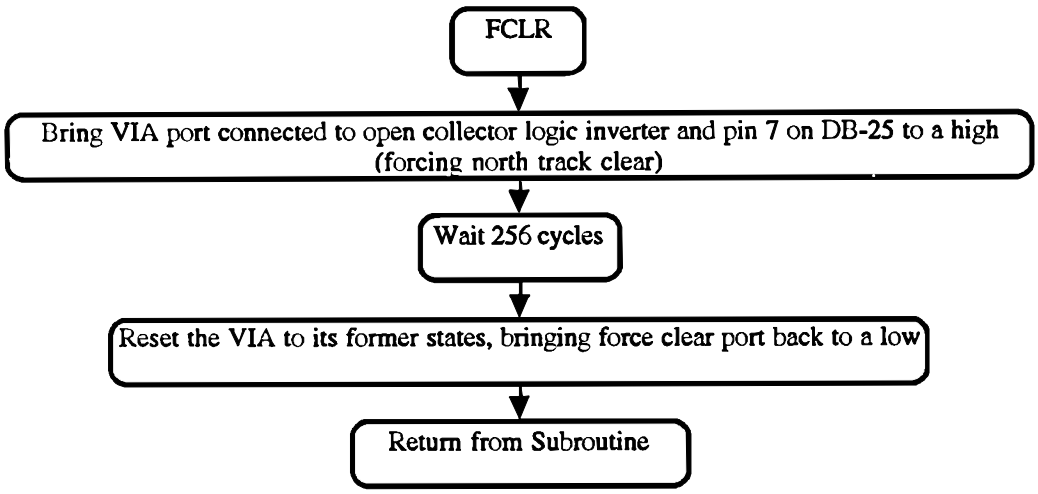
DISPMSG


DISPMSG
Load in the next character from message
Is this character a <NUL> ($00)?
NO
YES
Send this 8 bit character out on VIA to UAR/T
Return from DISPMSG subroutine
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

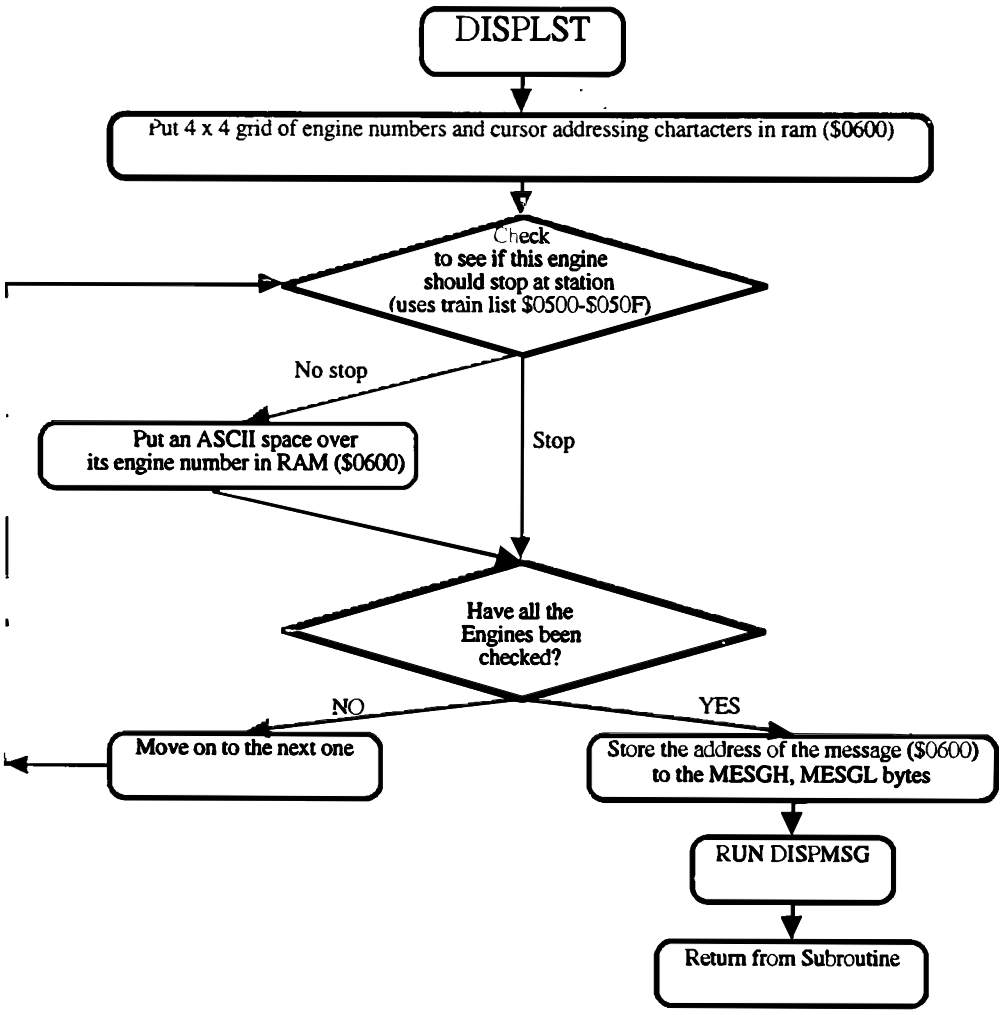
CLRLST


CLRLST
Put binary number 1000000 into all 16 train status locations $0500 - $05FF
Return from BRDSET Subroutine

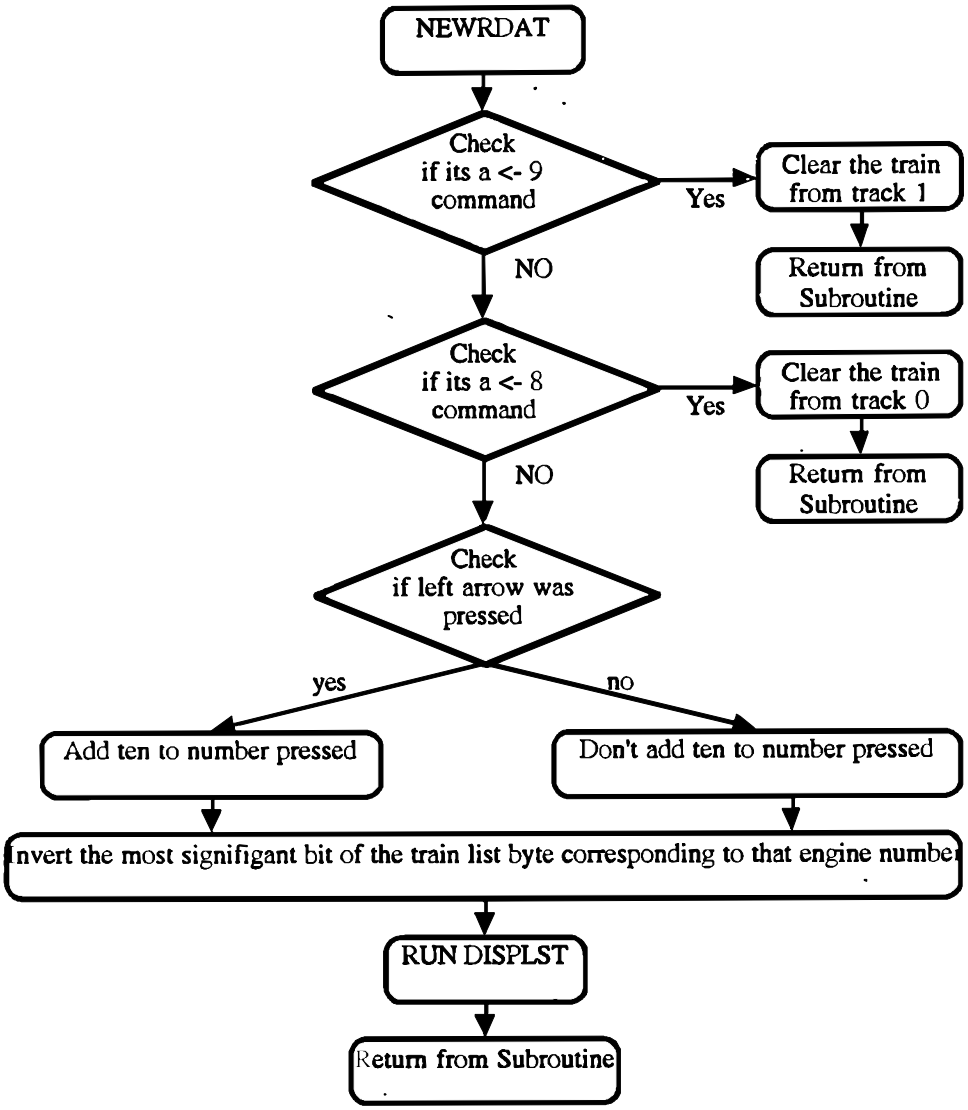
FCLR


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

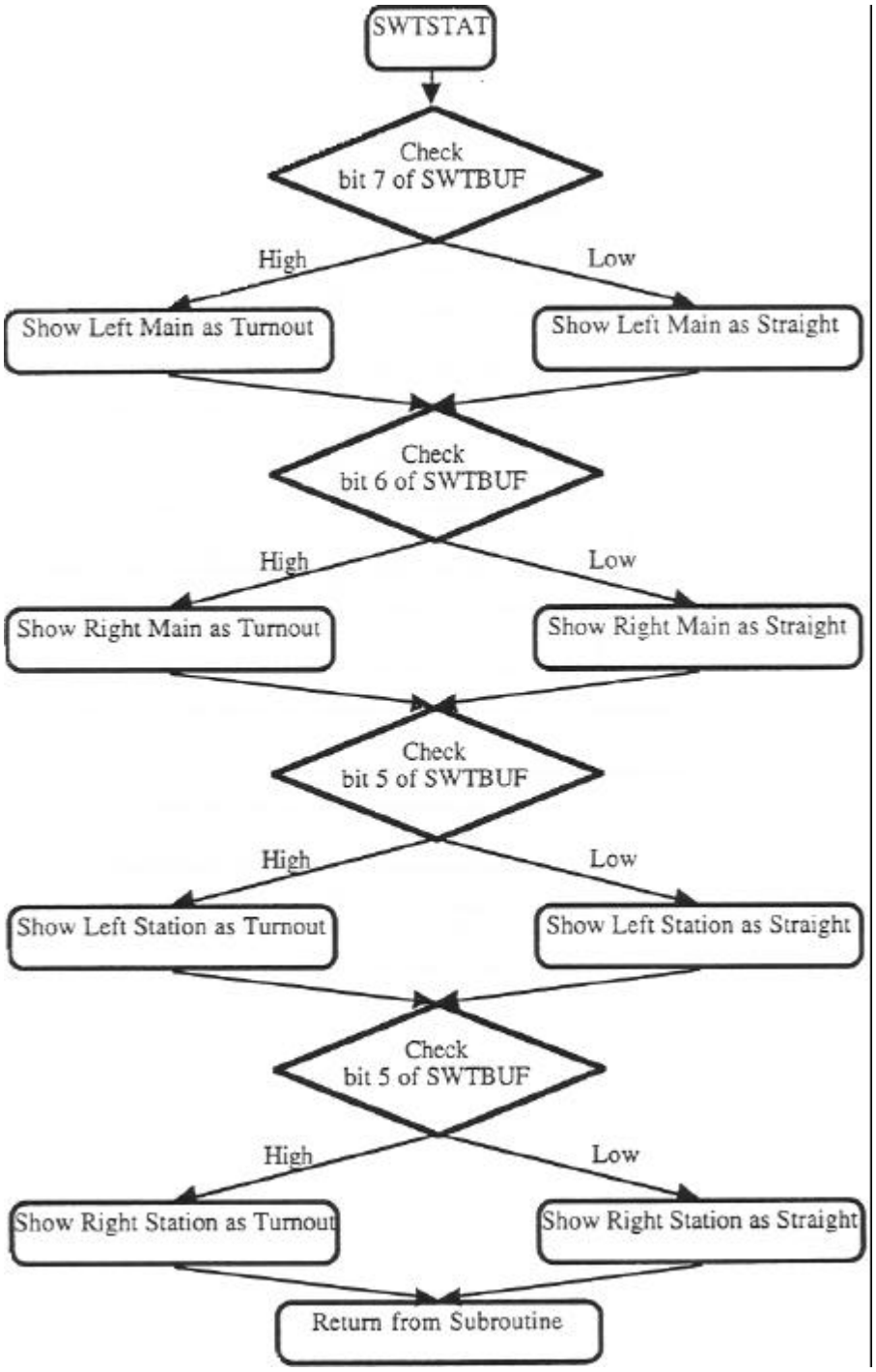
DISPLST


DISPLST
Put 4 x 4 grid of engine numbers and cursor addressing chartacters in ram ($0600)
Check
to see if this engine
should stop at station
(uses train list $0500-$050F)
No stop
Stop
Put an ASCII space over
its engine number in RAM ($0600)
Have all the
Engines been
checked?
NO
YES
Move on to the next one
Store the address of the message ($0600)
to the MESGH, MESGL bytes
RUN DISPMSG
Return from Subroutine

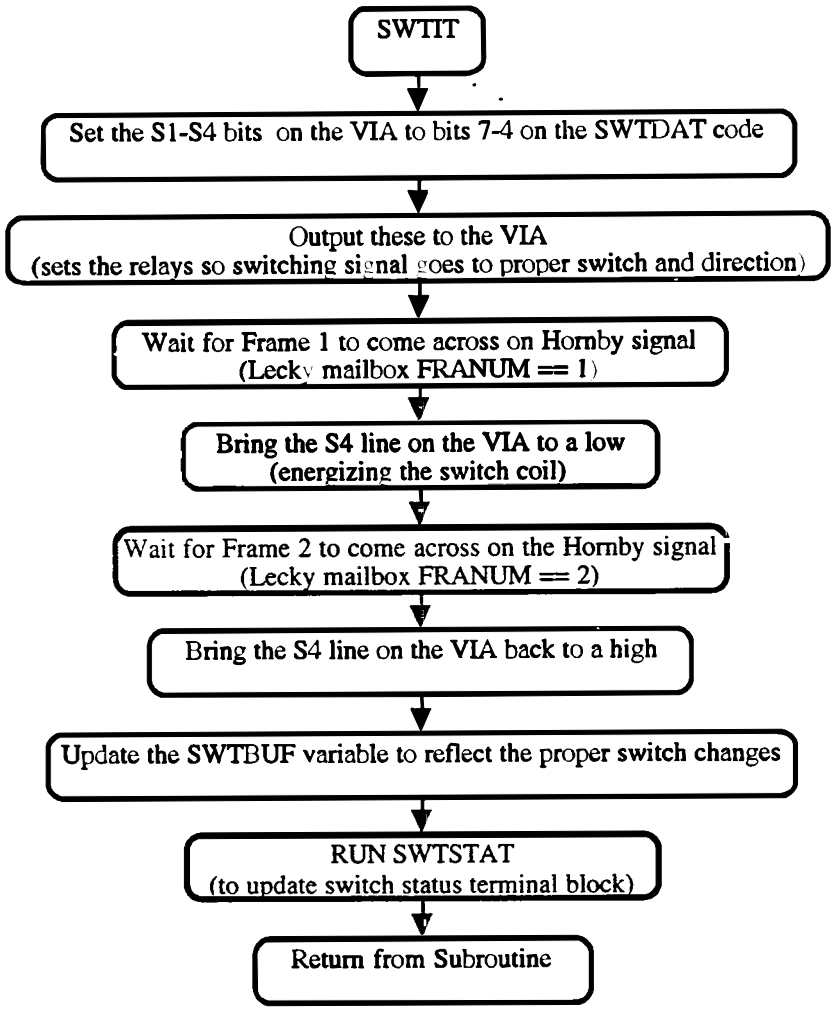
NEWRDAT


NEWRDAT
Check if its a <- 9 command
Yes
Clear the train from track 1
Return from Subroutine
NO
Check if its a <- 8 command
Yes
Clear the train from track 0
Return from Subroutine
NO
Check if left arrow was pressed
yes
no
Add ten to number pressed
Don't add ten to number pressed
Invert the most signifigant bit of the train list byte corresponding to that engine numbe
RUN DISPLST
Return from Subroutine

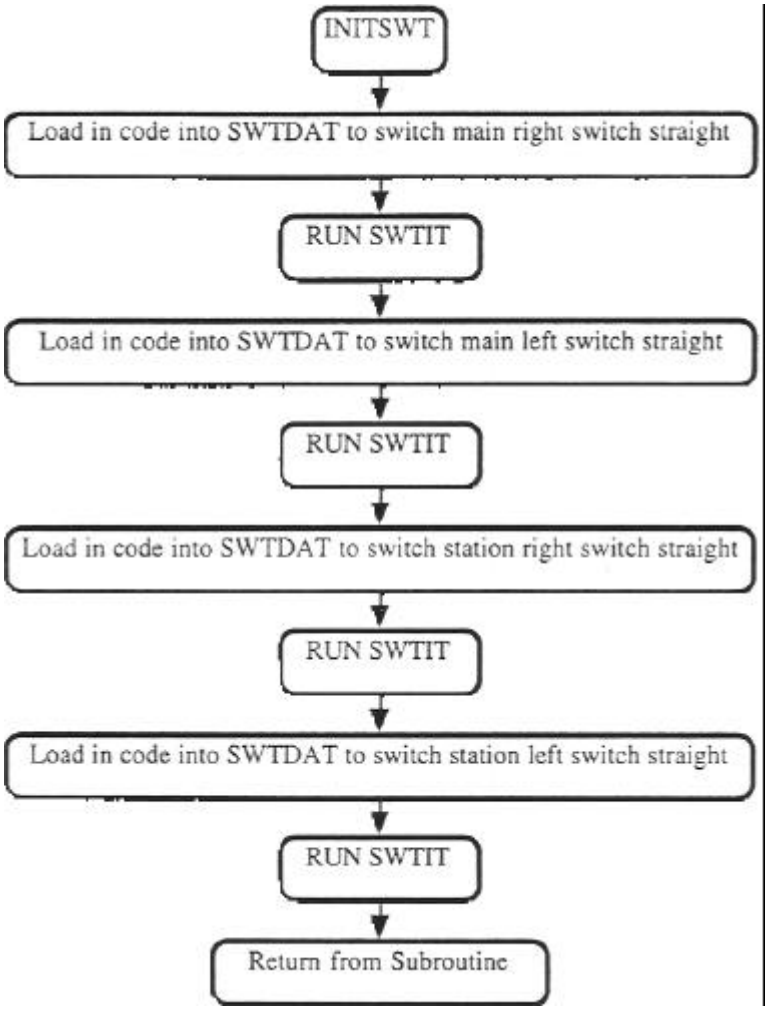
SWTSTAT


SWTSTAT
Check bit 7 of SWTBUF
High
Low
Show Left Main as Turnout
Show Left Main as Straight
Check bit 6 of SWTBUF
High
Low
Show Right Main as Turnout
Show Right Main as Straight
Check bit 5 of SWTBUF
High
Low
Show Left Station as Turnout
Show Left Station as Straight
Check bit 5 of SWTBUF
High
Low
Show Right Station as Turnout
Show Right Station as Straight
Return from Subroutine

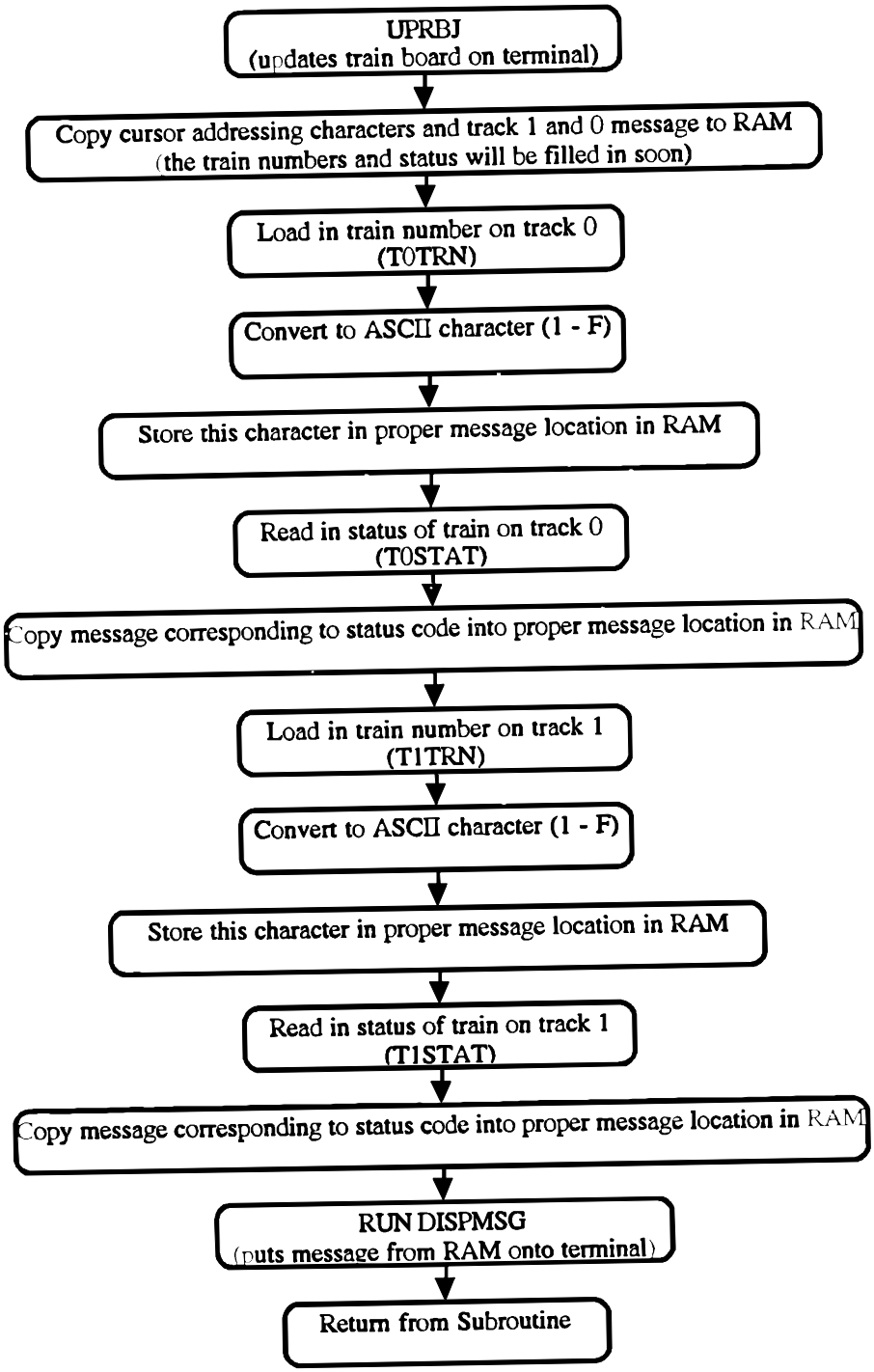
SWTIT


SWTIT
Set the S1-S4 bits on the VIA to bits 7-4 on the SWTDAT code
Output these to the VIA
(sets 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


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
(puts message from RAM onto terminal)
Return from Subroutine

Program Listing

```
%include  LECKY3_1.ASM


;VIA setup and data locations   (V1 for VIA @$A000,
                                V2 for VIA @$8000)
;SET for setup (0-input 1 output)   DAT for data

V1ASET   EQU $A003
V1BSET   EQU $A002

V1ADAT   EQU $A001
V1BDAT   EQU $A000

V1ABUF   EQU $70      ;zero page location that holds
                      the
                              ;current  values
                      for the VIA outputs
                              ;(Allows  program
                      to just change one
                              ; output, and not
                      the others from
                              ; anywhere in the
                      program)

V2ASET   EQU $8003
V2BSET   EQU $8002

V2ADAT   EQU $8001
V2BDAT   EQU $8000

TIL      EQU $4000


;START of MY PROGRAM
         ORG $E000

MYINIT   SEI
         CLD
         CLC
         LDX #$FF
         TXS
         LDA #$05         ;set up as computer #05
         STA BLKID
         JSR INIT         ;run Lecky initialization

VIAINIT  LDA #$FF         ;set up UAR/T outputs (8
                      data bits)
         STA V2ASET   ;use VIA @$8000 port A

         STA V1ASET   ;Outputs for :
                                 ;switches,
                      track kills, force clear
                                 ;PA7 -
                      Switch4 (1 - no switch 0 -
                      switch it)
                                 ;PA6 -
                      Switch3 (1 - Straight  0 -
                      Turn)
                                 ;PA5 -
                      Switch2 (1 - Left      0 -
                      Right)
                                 ;PA4 -
                      Switch1 (1 - Station   0 -
                      Main)
                                 ;PA3 - Force
                      North track clear (H - force
                      clear)
                                 ;PA2 - On
                      Deck (H - on  L - sensor
                      decides)
                                 ;PA1 - Track
                      1 (H - on  L - sensor decides)
                                 ;PA0 - Track
                      0 (H - on  L - sensor decides)
                                 ;uses VIA
                      @$A000 port A

         STA V1ADAT   ;initialize outputs
         LDA #$F0         ;let sensors decide track
                      kills
         STA V1ADAT   ;and don't force track clear
         STA V1ABUF   ;set up buffer for this port

         LDA #%01000011  ;set up UAR/T control lines
                                 ;PB7 End Of
                      Char from UAR/T (input)
                                 ;PB0 data
                      strobe line (output)

                                 ;Optical/kill
                      Sensors (0 - train, track off)
                                 ;PB5 ondeck
                      sensor (input)
                                 ;PB4 track
                      zero sensor (input)
                                 ;PB3 track
                      one sensor (input)

                                 ;optical
                                                ;(1 - train
                      passing by,  0 - no train)
                                                ;PB2 NORTH
                      TRACK sensor (input)
         STA V2BSET   ;Use VIA @$8000 port B


; uses local variables :
ONTRN    EQU $10          ;train # on north track
                      last loop
OSTRN    EQU $11          ;train # on south track
                      last loop
ORDATA   EQU $12          ;command from console last
                      loop

;initialize some variables
         LDA #$FF         ;initialize variables to
                      $FF
         STA ONTRN        ;this way first value from
                      Lecky
         STA OSTRN        ;will always be handled and
                      terminal
         STA ORDATA   ;instantly updated


;Set up the terminal, train list, and clock
         JSR BRDSET   ;setups up the blocks on the
                      terminal
         JSR CLRLST   ;sets up the train list in
                      memory and clears it
         JSR INITCLO  ;initialize the real time
                      clock, and show time

;set up the switches to default positions
         LDA #$00         ;initialize variables to
                      $00
         STA SWTBUF   ;start off with all the
                      switches switched straight
         JSR INITSWT  ;switch all switches to
                      straight

;set up the track status board
         LDA #$00         ;initialize some variables
                      to $00
         STA NXTRK        ;first train to station
                      will stop on track 0
         STA T0TRN
         STA T0STAT
         STA T1TRN
         STA T1STAT

         JSR UPRBJ        ;update the track status
                      board on terminal
;
;************************
;* MAIN PROGRAM LOOP     *
;*                       *
;************************
;
LOOPIT   NOP

          ;JSR DISPTRN      ;display north/south
                      trains on TIL

         LDA NTRAIN       ;check if a new/no train
                      on north track
         CMP ONTRN
         BEQ SAMEN        ;if its the same continue
                      in main loop
         STA ONTRN
         JSR NEWNTRN      ;if it is a change in
                      train handle it
SAMEN    NOP

         LDA STRAIN       ;check if a new/no train
                      on south track
         CMP OSTRN
         BEQ SAMES        ;if its the same continue
                      main loop
         STA OSTRN
         JSR NEWSTRN      ;if it is a change in
                      train handle it
SAMES    NOP

         LDA RDATA        ;check to see if a new
                      console command
         STA TIL          ;output to TIL for user to
                      see
         CMP ORDATA
         BEQ SAMER        ;if its the same command
                      continue
         JSR NEWRDAT      ;if its a new one handle
                      it
SAMER    NOP

         JMP LOOPIT

;
; Procedure : NEWSTRN
;         a new train has entered south track
```

```
;         or a train has left the south track
                    leaving it empty
;         display as status message to terminal
;
;Uses a few bytes of ram on page $0A00 to assembly
                    a message

STXT      EQU $0A00
STXT1     EQU $0A01

NEWSTRN   TAX
          BEQ NOSTRN        ;if south track is now
                    clear deal with it
          LDA NUM2TXT, X    ;Load in ASCII character
                    equivalent to train number
          STA STXT
          LDA #$00
          STA STXT1         ;puts an end of message at
                    end

;display  1st half of message
          LDA #HIGH STMSGA
          STA MESGH
          LDA #LOW STMSGA
          STA MESGL
          JSR DISPMSG

;display  train number
          LDA #HIGH STXT
          STA MESGH
          LDA #LOW STXT
          STA MESGL
          JSR DISPMSG

;display  2nd half of message
          LDA #HIGH STMSGB
          STA MESGH
          LDA #LOW STMSGB
          STA MESGL
          JSR DISPMSG
          RTS

;Deal with $00 if the south track is clear
NOSTRN    LDA #HIGH NSTMSG
          STA MESGH
          LDA #LOW NSTMSG
          STA MESGL
          JSR DISPMSG
          RTS


; MESSAGE  : NSTMSG
;          message shown when south tracks are clear
;          (no train on the south tracks)
;
NSTMSG    DB $1B, 'X4'
          DB $1B, 'Yc'
          DB '  CLEAR  '
          DB $00


; Message  : STMSGB
;          message outputted after the train number,
                    when
;          the train on south track changes
STMSGB    DB ' '
          DB $00    ;end of message


; Message  : STMSGA
;          message outputted before train number when
                    a train is on
;          the south track
STMSGA    DB $1B, 'X4'
          DB $1B, 'Yc'
          DB 'Train #'
          DB $00    ;end of message


; Procedure : NEWNTRN
;          a new train has entered north track
;          or a train has left the north track
                    leaving it empty
;          display as status message to terminal
;
;Uses a few bytes of ram on page $0A00 to assembly
                    a message

NTXT      EQU $0A00
NTXT1     EQU $0A01

NEWNTRN   TAX
          BEQ NONTRN        ;if north track is now
                    clear deal with it
          LDA NUM2TXT, X    ;Load in ASCII character
                    equivalent to train number
          STA NTXT
          LDA #$00
          STA NTXT1         ;puts an end of message at
                    end

;display  1st half of message
          LDA #HIGH NTMSGA
          STA MESGH
          LDA #LOW NTMSGA
          STA MESGL
          JSR DISPMSG

;display  train number
          LDA #HIGH NTXT
          STA MESGH
          LDA #LOW NTXT
          STA MESGL
          JSR DISPMSG

;display  2nd half of message
          LDA #HIGH NTMSGB
          STA MESGH
          LDA #LOW NTMSGB
          STA MESGL
          JSR DISPMSG

;now check if this train is to stop at station
          LDX ONTRN         ;checking the train list
                    to see if this train
          LDA LSTLOC, X     ;stops at station or not
          BPL DONENT        ;if not then let it pass
                    through

          JSR STOPIT        ;if it is on the list then
                    stop it at station

DONENT    RTS

;Deal with $00 if the north track is clear
NONTRN    LDA #HIGH NNTMSG
          STA MESGH
          LDA #LOW NNTMSG
          STA MESGL
          JSR DISPMSG
          RTS


; Message  : NTMSGB
;          message outputted after the train number,
                    when
;          the train on north track changes
NTMSGB    DB ' '
          DB $0A    ;Line Feed
          DB $0D    ;carriage return
          DB $00    ;end of message

;
; Message  : NTMSGA
;          message outputted before train number when
                    a train is on
;          the north track
NTMSGA    DB $1B, 'X4'
          DB $1B, 'Y8'
          DB 'Train #'
          DB $00    ;end of message


;
; Table  : NUM2TXT
;          lookup table that goes from train number
                    to an ASCII character
;
NUM2TXT   DB '0'
          DB '1'
          DB '2'
          DB '3'
          DB '4'
          DB '5'
          DB '6'
          DB '7'
          DB '8'
          DB '9'
          DB 'A'
          DB 'B'
          DB 'C'
          DB 'D'
          DB 'E'
          DB 'F'
          DB 'G'


;
;   Procedure:   STOPIT
;
;          Stop this train on the north track at the
                    station
;
NXTRK     EQU $E0 ;value for which station track to
                    be used next
                  ;$00 for track 0,  $FF for track 1

;Train will be stopping at station
STOPIT    LDA #%10100000    ;switch the main left
                    switch to turnout
          STA SWTDAT
          JSR SWTIT

          LDX #$02          ;status code equivalent to
                    departing
          LDA NXTRK         ;figure out which track
                    this train is going to
          BEQ NT0           ;if its $00 then its track
                    0

;set switches to track 1
          STX T1STAT        ;show that the train on
                    track 1 will be departing

          LDA #%10110000    ;switch the station left
                    to turnout
```

```
        STA SWTDAT
        JSR SWTIT

         LDA #%10010000  ;switch the station right
                         to turnout
        STA SWTDAT
        JSR SWTIT

         JMP STP1        ;go on with station stop

NT0     STX T0STAT       ;show departing on train
                         board for track 0

;set the switches to track 0
         LDA #%11110000  ;switch the station left
                         to straight
        STA SWTDAT
        JSR SWTIT

         LDA #%11010000  ;switch the station right
                         to straight
        STA SWTDAT
        JSR SWTIT

STP1    JSR UPRBJ        ;update the train board

WAITON  LDA V2BDAT       ;now wait for train to get
                         to ondeck location
         AND #%00100000  ;makes all other bits zero
                         except for ondeck sensor
         BNE WAITON      ;if train not there yet
                         keep waiting

         LDA #%11100000  ;switch the main left
                         switch to straight
        STA SWTDAT
        JSR SWTIT

         LDA NXTRK       ;figure out which track is
                         the next one to open up
         BEQ NXTRK0      ;if its $00 then go to
                         track 0 as next track
         JMP NXTRK1      ;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
NXTRK0  LDA T0TRN
         BEQ NOT0TRN     ;if no train skip the part
                         about moving a train off

;the train on track 0 has to be moved off and back
                         onto north track
;to continue on its merry way

          LDA #%10000000  ;switch the main right
                         switch to turnout
        STA SWTDAT
        JSR SWTIT

;energize the track 0 section
        LDA V1ABUF
         AND #%11111110  ;bring PA0 to a low
         EOR #%00000001  ;bring PA0 to a high
                         leaving other bits unchanged
         STA V1ADAT      ;output this to the VIA

;now track 0 section energizes, and train begins
                         to move away

         LDA V1ABUF      ;restore the track 0 to be
                         killed when next train
         STA V1ADAT      ;crosses in front of
                         sensor

;Wait until the leaving train clears the north
                         track optical sensor
WAITNOS LDA V2BDAT       ;check PB2 for a train
                         passing the north track
                         optical
         AND #%00000100  ;makes other bits 0
                         leaves bit 2 unchanged
         BEQ WAITNOS     ;if bit 2 is low keep
                         waiting

;now train has moved off track 0 and the on deck
                         train can be moved on

          LDA #%11000000  ;switch main right switch
                         to straight
        STA SWTDAT
        JSR SWTIT

;change the train number on the north track to
                         this one
         LDA T0TRN       ;train number of train
                         leaving station
         STA NTRAIN      ;(use Lecky mailbox for
                         this (could be rule violation)

;update the train board for the new train number
                         to be moved on
;and show arriving as its status
         LDA ONTRN       ;set the new train number
                         on train board
        STA T0TRN

         LDA #$01        ;status code that refers
                         to arriving
        STA T0STAT

         JSR UPRBJ       ;update the train board

;so energize the on deck track section
         LDA V1ABUF      ;load in current output
                         values
         AND #%11111011  ;bring the 2 bit to a low
                          (leaves others unchanged)
         EOR #%00000100  ;bring bit 2 to a high
                          (turns on ondeck track)
         STA V1ADAT      ;turn on on deck track

;now wait for train to make it to track zero
WAITT0A LDA V2BDAT       ;check the track 0 sensor
         AND #%00010000  ;force all bits except for
                         bit 4 to a zero
         BNE WAITT0A     ;if bit 4 is high then no
                         train yet keep waiting

         LDA V1ABUF      ;restore the ondeck track
                         section to kill when
         STA V1ADAT      ;sensor detects a train

;update the train board to show that this new
                         train is stopped at the
                         station
         LDA #$03        ;status code equivalent to
                         stopped at station
        STA T0STAT

         JSR UPRBJ       ;update train board

;change the NXTRK variable to point to track
                         with the next train
        LDA #$FF
        STA NXTRK

         RTS      ;everything done and ready f
                         next train


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

;update the train board to show this train number
                         as arriving
NOT0TRN LDA ONTRN
        STA T0TRN

         LDA #$01        ;status code that refers
                         to arriving
        STA T0STAT

         JSR UPRBJ       ;update train board

;energize the on deck section of track
         LDA V1ABUF      ;load in current output
                         values
         AND #%11111011  ;bring the 2 bit to a low
                          (leaves others unchanged)
         EOR #%00000100  ;bring bit 2 to a high
                          (turns on ondeck track)
         STA V1ADAT      ;turn on on deck track

;now wait for train to make it to track zero
WAITT0B LDA V2BDAT       ;check the track 0 sensor
         AND #%00010000  ;force all bits except for
                         bit 4 to a zero
         BNE WAITT0B     ;if bit 4 is high then no
                         train yet keep waiting

         LDA V1ABUF      ;restore the ondeck track
                         section to kill when
         STA V1ADAT      ;sensor detects a train

;update the train board to show that this new
                         train is stopped at the
                         station
         LDA #$03        ;status code equivalent to
                         stopped at station
         STA T0STAT

         JSR UPRBJ       ;update train board

;set the NXTRK variable to point to track one with
                         the next train
        LDA #$FF
        STA NXTRK

;force a clear on the north track
        JSR FCLR

         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
NXTRK1  LDA T1TRN
         BEQ NOT1TRN     ;if no train skip the part
                         about moving a train off
```

```
;the train on track 1 has to be moved off and back
                    onto north track
;to continue on its merry way

          LDA #%10000000  ;switch the main right
                    switch to turnout
          STA SWTDAT
          JSR SWTIT

;energize the track 1 section
          LDA V1ABUF
          AND #%11111101  ;bring PA1 to a low
          EOR #%00000010  ;bring PA1 to a high
                    leaving other bits unchanged
          STA V1ADAT      ;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
WAITNS    LDA V2BDAT      ;check PB2 for a train
                    passing the north track
                    optical
          AND #%00000100  ;makes other bits 0
                    leaves bit 2 unchanged
          BEQ WAITNS      ;if bit 2 is low keep
                    waiting

;now train has moved off track 1 and the on deck
                    train can be moved on

          LDA #%11000000  ;switch main right switch
                    to straight
          STA SWTDAT
          JSR SWTIT

;change the train number on the north track to
                    this one
          LDA T1TRN       ;train number of train
                    leaving station
          STA NTRAIN      ;(use Lecky mailbox for
                    this (could be rule violation)

          LDA V1ABUF      ;restore the track 1 to be
                    killed when next train
          STA V1ADAT      ;crosses in front of
                    sensor

;now train has moved off track 1 and the on deck
                    train can be moved on

;update the train board for the new train number
                    to be moved on
;and show arriving as its status
          LDA ONTRN       ;set the new train number
                    on train board
          STA T1TRN

          LDA #$01        ;status code that refers
                    to arriving
          STA T1STAT

          JSR UPRBJ       ;update the train board

;so energize the on deck track section
          LDA V1ABUF      ;load in current output
                    values
          AND #%11111011  ;bring the 2 bit to a low
                    (leaves others unchanged)
          EOR #%00000100  ;bring bit 2 to a high
                    (turns on ondeck track)
          STA V1ADAT      ;turn on on deck track

;now wait for train to make it to track 1
WAITT1A   LDA V2BDAT      ;check the track 1 sensor
          AND #%00001000  ;force all bits except for
                    bit 3 to a zero
          BNE WAITT1A     ;if bit 3 is high then no
                    train yet keep waiting

          LDA V1ABUF      ;restore the ondeck track
                    section to kill when
          STA V1ADAT      ;sensor detects a train

;update the train board to show that this new
                    train is stopped at the
                    station
          LDA #$03        ;status code equivalent to
                    stopped at station
          STA T1STAT

          JSR UPRBJ       ;update train board

;change the NXTRK variable to point to track 0
                    with the next train
          LDA #$00
          STA NXTRK

          RTS       ;everything done and ready for the
                    next train

;situation:  moving a train onto track 1 and no
                    train currently on track 1

;update the train board to show this train number
                    as arriving

NOT1TRN   LDA ONTRN
          STA T1TRN

          LDA #$01        ;status code that refers
                    to arriving
          STA T1STAT

          JSR UPRBJ       ;update train board

;energize the on deck section of track
          LDA V1ABUF      ;load in current output
                    values
          AND #%11111011  ;bring the 2 bit to a low
                    (leaves others unchanged)
          EOR #%00000100  ;bring bit 2 to a high
                    (turns on ondeck track)
          STA V1ADAT      ;turn on deck track

;now wait for train to make it to track 1
WAITT1B   LDA V2BDAT      ;check the track 1 sensor
          AND #%00001000  ;force all bits except for
                    bit 3 to a zero
          BNE WAITT1B     ;if bit 3 is high then no
                    train yet keep waiting

          LDA V1ABUF      ;restore the ondeck track
                    section to kill when
          STA V1ADAT      ;sensor detects a train

;update the train board to show that this new
                    train is stopped at the
                    station
          LDA #$03        ;status code equivalent to
                    stopped at station
          STA T1STAT

          JSR UPRBJ       ;update train board

;set the NXTRK variable to point to track 0 with
                    the next train
          LDA #$00
          STA NXTRK

;force a clear on the north track
          JSR FCLR

          RTS             ;all done and ready for
                    next train

;
; MESSAGE : NNTMSG
;         message shown when north tracks
;         (no train on the north tracks)
;
NNTMSG    DB $1B, 'X4'
          DB $1B, 'Y#'
          DB '  CLEAR  '
          DB $00

;
; Procedure : DISPTRN
;
;         takes the north and south train numbers
                    from Lecky
;         and displays them on the TIL
;         North on TIL high nibble,  South TIL on
                    the low nibble
;
;         uses BTRNS  local variable  located @
                    $00F0

BTRNS     EQU $F0

DISPTRN   LDA NTRAIN
          ASL A
          ASL A
          ASL A
          ASL A
          STA BTRNS
          LDA STRAIN
          ADC BTRNS
          STA TIL
          RTS

; Procedure : BRDSET
;
; Sets up the blocks on the train table on
                    terminal
;
; Uses :
BRDSET    LDA #HIGH NSBLKS
          STA MESGH
          LDA #LOW NSBLKS
          STA MESGL
          JSR DISPMSG

          LDA #HIGH TRLBLK1
          STA MESGH
          LDA #LOW TRLBLK1
          STA MESGL
          JSR DISPMSG

          LDA #LOW TRLBLK2
          STA MESGL
          LDA #HIGH TRLBLK2
          STA MESGH
          JSR DISPMSG
```

```
        LDA #LOW SWTBLK
        STA MESGL
        LDA #HIGH SWTBLK
        STA MESGH
        JSR DISPMSG

        LDA #LOW RBJBLK1
        STA MESGL
        LDA #HIGH RBJBLK1
        STA MESGH
        JSR DISPMSG

        LDA #LOW RBJBLK2
        STA MESGL
        LDA #HIGH RBJBLK2
        STA MESGH
        JSR DISPMSG

        RTS

; Message : NSBLKS
;
; used to set borders around the north and south
                    train blocks
;
NSBLKS  DB $1B, 'X0', $0D
          DB '/-------------\', $1B, 'Y', '/-------
                    ------\', $0D, $0A
          DB '| NORTH TRACK |', $1B, 'Y', '| SOUTH
                    TRACK |', $0D, $0A
          DB '|-------------|', $1B, 'Y', '|-------
                    ------|', $0D, $0A
          DB '|             |', $1B, 'Y', '|
                    |', $0D, $0A
          DB '|             |', $1B, 'Y', '|
                    |', $0D, $0A
          DB '|             |', $1B, 'Y', '|
                    |', $0D, $0A
          DB '\-------------/', $1B, 'Y', '\-------
                    ------/'
          DB $00

; Message : TRLBLK
;
; used to set the borders around block that
                    displays which trains
; will stop at the station
;
TRLBLK1 DB $1B, 'H'       ;homes the cursor
          DB $1B, 'Y', '/--------------\', $0A
          DB $1B, 'Y', '|Trains to STOP|', $0A
          DB $1B, 'Y', '|at the STATION|', $0A
          DB $1B, 'Y', '|--------------|', $0A
          DB $1B, 'Y', '|              |', $0A
          DB $1B, 'Y', '|              |', $0A
          DB $1B, 'Y', '|              |', $0A
          DB $00  ;end of message

TRLBLK2 DB $1B, 'Y', '|              |', $0A
          DB $1B, 'Y', '|              |', $0A
          DB $1B, 'Y', '|              |', $0A
          DB $1B, 'Y', '|              |', $0A
          DB $1B, 'Y', '|              |', $0A
          DB $1B, 'Y', '|              |', $0A
          DB $1B, 'Y', '\--------------/', $0A
          DB $00  ;end of message

; Message : SWTBLK
;
;         sets up the block on the terminal to
                    display the status
;         of the various switches
;
SWTBLK  DB $1B, 'X0'      ;move to line 17
          DB $1B, 'Y;', '/-----------------------\',
                    $0A   ;col. 28, linefeed
          DB $1B, 'Y;', '|      SWITCH STATUS      |',
                    $0A
          DB $1B, 'Y;', '|    Main     |  Station  |',
                    $0A
           DB $1B, 'Y;', '|Left |Right|Left |Right|',
                    $0A
           DB $1B, 'Y;', '|-----------|-----------|',
                    $0A
          DB $1B, 'Y;', '|     |     |     |     |',
                    $0A
           DB $1B, 'Y;', '\-----------------------/'
          DB $00                          ;end of
                    message

; Message : RBJBLK
;
;         Sets up block to display which train on
                    which station track,
;         and its status
;                   (in future maybe add time of
                    arrival/ departure)
;
RBJBLK1 DB $1B, 'X ', $1B, 'Y '           ;move to
                    line 1, column 1
          DB '/---------------------------------\',
                    $0A, $0D      ;LF, CR
          DB '|                                 |',
                    $0A, $0D
          DB '|         Robust Junction         |',
                    $0A, $0D
          DB '|                                 |',
                    $0A, $0D
          DB '|           TRAIN #----STATUS---|',
                    $0A, $0D
          DB '|             |           |       |',
                    $0A, $0D
          DB '| TRACK 0     |           |       |',
                    $0A, $0D
          DB $00  ;end of message

RBJBLK2 DB '|             |           |       |',
                    $0A, $0D
          DB '|-------------|-----------|-------|',
                    $0A, $0D
          DB '|             |           |       |',
                    $0A, $0D
          DB '| TRACK 1     |           |       |',
                    $0A, $0D
          DB '|             |           |       |',
                    $0A, $0D
          DB '\---------------------------------/',
                    $0A, $0D
          DB $00  ;end of message

; Procedure : DISPMSG
;
;Displays the string of characters located at
; ADL at $50 on zero page
; ADH at $51 on zero page

MESGH   EQU $51           ;High byte of location of
                    text message
MESGL   EQU $50           ;Low byte of location of
                    text message

;    uses NUL ($00) as end of message

DISPMSG LDY #$FF          ;reset character counter

NEXTCHR INY               ;step up counter to next
                    character in memory
        LDA (MESGL), Y    ;read in that character
        BEQ DONEMSG       ;if its a <NUL> ($00) then
                    stop (done with message)

        STA V2ADAT        ;put this characters 8
                    bits to the VIA #2 port A
                          ;also connected to the
                    transmit data lines on UAR/T
WAITH   LDA V2BDAT        ;Check the EOC line from
                    the UAR/T  (PB7 on VIA#2)
        BPL WAITH         ;if its high, keep
                    checking until it goes low
                          ;(this waits for the last
                    character to be sent to
                    terminal)

        LDA #$00          ;Pulse the Data Strobe
                    line of the UAR/T low
        STA V2BDAT        ;Tells UAR/T next
                    character's data is stable
        LDA #$01          ;and ready to be read into
                    the UAR/T buffer
        STA V2BDAT        ;complete the pulse by
                    putting the DS back to High

WAITL   LDA V2BDAT
        BMI WAITL

        JMP NEXTCHR

DONEMSG RTS

; Procedure : CLRLST
;
;         Sets up the trail list in memory on page
                    $0500-$050F
;         one byte for each train
;          the Most Significant Bit signifies whether
                    to
;         stop that train at the station or not
;         (1#######  - stop this train,  0####### -
                    let this train pass)
;
;         Reserves $0500 - $050F for its use
;        Uses :
LSTLOC  EQU $0500

CLRLST  LDA #$80          ;default is all trains
                    stop at the station
        LDX #$0F
NXTLOC  STA LSTLOC, X
        DEX
        BPL NXTLOC

        RTS

; 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
;
```

```
;         Assembles its message to send to the
                    terminal on the $0600 page
MSGASM    EQU $0600

DISPLST LDX #$00          ;copy the 4 x 4 grid of
                    loco numbers
NXTDS     LDA BLANKL, X    ;and text location
                    characters
          STA MSGASM, X    ;to message assembly page
          BEQ DONEDS
          INX
          JMP NXTDS

DONEDS    NOP

          LDX #$00
STARTR    LDA LSTLOC, X
          BMI NEXTTR       ;if flag set go on to next
                    one
          LDY LSTTAB, X    ;if no flag set then
          LDA #' '         ;store a space in the
                    assembled
          STA MSGASM, Y    ;message location over the
                    loco number to not stop


NEXTTR    INX
          CPX #$10
          BNE STARTR

          LDA #HIGH MSGASM  ;Finally output
                    assembled message to terminal
          STA MESGH
          LDA #LOW MSGASM
          STA MESGL
          JSR DISPMSG

          RTS

;  Table  : LSTTAB
;
;         The address refers to locomotive number,
                    and the value
;         refers to number of bytes from $0600 that
                    that locomotive
;         number is in the assembled message
;
LSTTAB    DB $06, $09, $0C, $0F    ;for loco #'s  0,
                    1, 2, 3
          DB $16, $19, $1C, $1F    ;for loco's 4, 5,
                    6, 7
          DB $26, $29, $2C, $2F    ;loco's 8, 9, A, B
          DB $36, $39, $3C, $3F    ;I think you can
                    see the pattern

;  Message  : BLANKL
;
;         clears the trains to stop at station block
;
BLANKL    DB $1B, 'X%'
          DB $1B, 'Yc', '   1  2  3', $0D, $0A, $0A
          DB $1B, 'Yc', '4  5  6  7', $0D, $0A, $0A
          DB $1B, 'Yc', '8  9  A  B', $0D, $0A, $0A
          DB $1B, 'Yc', 'C  D  E  F'
          DB $00


;  Procedure  : NEWRDAT
;
;         a new console command has been issued so
                    must be handled
;         commands -> then numbers 1 - 9 correspond
                    to taking train #'s 1-9
;         on and off the train list, while
;         commands <- then numbers 0-5 correspond to
                    taking train #'s A-F
;         on and off the train list
;
;         other commands to be assigned later
;
NEWRDAT   LDA  RDATA
          STA ORDATA       ;store this new value in
                    variable

          CMP #$89         ;if its a <- 9 command
                    then
          BEQ LEFT9        ;deal with the a left nine

          CMP #$88         ;if its a <- 8 command
                    deal with that
          BNE CHL7
          JMP LEFT8

          CMP #$87         ;if its a <- 7 command
                    deal with that
          BNE CHL6
          JMP LEFT7

CHL6      CMP #$86         ;if its a <- 6 command
                    deal with that
          BNE CONTOR
          JMP LEFT6

                    ;if its another command
                    then corresponds
                    ;to a train number to
                    either add
                    ;or take away from trains
                    to stop at
                    ;station list
CONTOR    LDA ORDATA       ;reload which command was
                    pressed to reset N flag
          BPL LOWTRN       ;if left arrow not pressed
                    go to lowtrns
          AND #$0F         ;turn off the high bit
          CLC
          ADC #$0A         ;add ten to get the high
                    train numbers
LOWTRN    TAX
          LDA LSTLOC, X    ;load in train list byte
          EOR #$80         ;if it was on list take
                    off, and vice versa
          STA LSTLOC, X

          JSR DISPLST      ;update the train list on
                    terminal

          RTS

LEFT9     NOP              ;clear the train off of
                    track 1

;update the train board to show track 1 train is
                    departing
          LDA #$02         ;status code equivalent to
                    departing
          STA T1STAT
          JSR UPRBJ

WAITNC    LDA NTRAIN       ;wait until the train on
                    north track clears
          BNE WAITNC

          LDA #$FF
          STA NOVRD        ;override the north track
                    to occupied (I think)

          LDA #%10010000   ;switch the station right
                    to turnout
          STA SWTDAT
          JSR SWTIT

          LDA #%10000000   ;switch the main right to
                    turnout
          STA SWTDAT
          JSR SWTIT

;energize the track 1 section
          LDA V1ABUF
          AND #%11111101   ;bring PA1 to a low
          EOR #%00000010   ;bring PA1 to a high
                    leaving other bits unchanged
          STA V1ADAT       ;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
WAITNSC LDA V2EDAT         ;check PB2 for a train
                    passing the north track
                    optical
          AND #%00000100   ;makes other bits 0
                    leaves bit 2 unchanged
          BEQ WAITNSC      ;if bit 2 is low keep
                    waiting

;now train has moved off track 1

          LDA #%11000000   ;switch main right switch
                    to straight
          STA SWTDAT
          JSR SWTIT

;update the train board to reflect current
                    conditions
          LDA #$00         ;status code referring to
                    empty

          ---- -----

          JSR UPRBJ

;wait for train to clear the train board
          LDA #$08
          STA $0F          ;use a zero page location
                    for another counter register

          LDX #$00
          LDY #$00
DELAY1A   DEX
          BNE DELAY1A
          DEY
          BNE DELAY1A
          DEC $0F
          BNE DELAY1A

;turn the north track override off
          LDA #$00
          STA NOVRD

;all done
          RTS
```

```
LEFT8   NOP             ;clear the train off of
                        track 0

;update the train board to show track 0 train is
                        departing
        LDA #$02        ;status code equivalent to
                        departing
        STA T0STAT
        JSR UPRBJ

WAITND  LDA NTRAIN      ;wait until the train on
                        north track clears
        BNE WAITND

        LDA #$FF
        STA NOVRD       ;override the north track
                        to occupied (I think)

        LDA #%11010000  ;switch the station right
                        to straight
        STA SWTDAT
        JSR SWTIT

        LDA #%10000000  ;switch the main right to
                        turnout
        STA SWTDAT
        JSR SWTIT

;energize the track 0 section
        LDA V1ABUF
        AND #%11111110  ;bring PA0 to a low
        EOR #%00000001  ;bring PA0 to a high
                        leaving other bits unchanged
        STA V1ADAT      ;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
WAITNSE LDA V2BDAT      ;check PB2 for a train
                        passing the north track
                        optical
        AND #%00000100  ;makes other bits 0
                        leaves bit 2 unchanged
        BEQ WAITNSE     ;if bit 2 is low keep
                        waiting

;now train has moved off track 0

        LDA #%11000000  ;switch main right switch
                        to straight
        STA SWTDAT
        JSR SWTIT

;update the train board to reflect current
                        conditions
        LDA #$00        ;status code referring to
                        empty
        STA T0STAT
        STA T0TRN

        JSR UPRBJ


;wait for train to clear the train board
        LDA #$08
        STA $0F         ;use a zero page location
                        for another counter register

        LDX #$00
        LDY #$00
DELAY1B DEX
        BNE DELAY1B
        DEY
        BNE DELAY1B
        DEC $0F
        BNE DELAY1B

;turn the north track override off
        LDA #$00
        STA NOVRD

;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 :
SWTBUF  EQU     $A0     ;zero page location of
                        switch status byte
                        ; high is turnout,  low is
                        straight
                        ;bit 7 is for Left Main
                        switch
                        ;bit 6 is for Right Main
                        switch
                        ;bit 5 is for LEFT STATION
                        switch
                        ;bit 4 is for RIGHT
                        STATION switch
                        ;remaining bits will be
                        kept as zeros
SWTSTAT NOP
        LDA SWTBUF
        AND #%10000000          ;bring all bits
                        except bit 7 to a low
        BNE LMT                 ;if bit 7 is high,
                        then left main turnout

LMS     LDA #HIGH LMSMSG        ;if its low then
                        left main straight
        STA MESGH               ;display that
                        message
        LDA #LOW LMSMSG
        STA MESGL
        JSR DISPMSG
        JMP RMDEC               ;done with that
                        switch go to next

LMT     LDA #HIGH LMTMSG        ;if its high then
                        left main switch is turnout
        STA MESGH
        LDA #LOW LMTMSG
        STA MESGL
        JSR DISPMSG

RMDEC   LDA SWTBUF              ;now make decision
                        about right main switch
        AND #%01000000          ;bring all other
                        bits to zeros
        BNE RMT                 ;if its high then
                        turnout branch to that

RMS     LDA #HIGH RMSMSG        ;if its low then
                        straight write that message
        STA MESGH
        LDA #LOW RMSMSG
        STA MESGL
        JSR DISPMSG
        JMP LSDEC               ;done with that go
                        to decision for left station

RMT     LDA #HIGH RMTMSG        ;if its a high,
                        then switch is turnout
        STA MESGH               ;Show the TURN
                        message in proper location
        LDA #LOW RMTMSG
        STA MESGL
        JSR DISPMSG

LSDEC   LDA SWTBUF              ;Decide about the
                        left station switch
        AND #%00100000          ;bring all bits
                        except bit 5 to a low
        BNE LST                 ;if bit 5 is high
                        then left station is turnout

LSS     LDA #HIGH LSSMSG        ;if its low then
                        switch is straight
        STA MESGH               ;display that
                        message
        LDA #LOW LSSMSG
        STA MESGL
        JSR DISPMSG
        JMP RSDEC               ;done here go to
                        decision for right station

LST     LDA #HIGH LSTMSG        ;if its high show
                        TURN message in right spot
        STA MESGH
        LDA #LOW LSTMSG
        STA MESGL
        JSR DISPMSG

RSDEC   LDA SWTBUF              ;Make decision
                        about the right station switch
        AND #%00010000          ;bring all bits
                        except for bit 4 to a low
        BNE RST                 ;if bit 4 is high
                        then switch is turnout

RSS     LDA #HIGH RSSMSG        ;if not then its
                        straight
        STA MESGH               ;display the STR8
                        message
        LDA #LOW RSSMSG
        STA MESGL
        JSR DISPMSG
        RTS                     ;return from
                        subroutine,  done displaying

RST     LDA #HIGH RSTMSG        ;switch is turnout
                        show that message
        STA MESGH
        LDA #LOW RSTMSG
        STA MESGL
        JSR DISPMSG
        RTS                     ;done return from
                        subroutine

   MESAGES   RM, LM, RS, LS,  -  S or T  - MSG
```

```
          first two letters correspond to which
                    switch   (L left, R Right
          M - main line, S - station line)  then the
                    next letter corresponds
          to the switches state (S - straight,  T
                    for turnout)
;
RMSMSG    DB $1B, 'X5', $1B, 'YB' ;line 22, column
                    35
          DB 'STR8'
          DB $00

RMTMSG    DB $1B, 'X5', $1B, 'YB' ;line 22, column
                    35
          DB 'TURN'
          DB $00

LMSMSG    DB $1B, 'X5', $1B, 'Y<' ;line 22, column
                    29
          DB 'STR8'
          DB $00

LMTMSG    DB $1B, 'X5', $1B, 'Y<' ;line 22, column
                    29
          DB 'TURN'
          DB $00

RSSMSG    DB $1B, 'X5', $1B, 'YN' ;line 22, column
                    47
          DB 'STR8'
          DB $00

RSTMSG    DB $1B, 'X5', $1B, 'YN' ;line 22, column
                    47
          DB 'TURN'
          DB $00

LSSMSG    DB $1B, 'X5', $1B, 'YH' ;line 22, column
                    41
          DB 'STR8'
          DB $00

LSTMSG    DB $1B, 'X5', $1B, 'YH' ;line 22, column
                    41
          DB 'TURN'
          DB $00

; Procedure : SWTIT
;
;         Uses the information passed to it in
                    SWTDAT, to switch the
;         desired switch the desired direction
;         also updates the SWTBUF, and calls SWTSTAT
                    to update display on terminal
;
; USES :
SWTDAT  EQU $A1   ;zero page location for data
                    indicating which switch
                  ;and which direction to switch
                  ;bit 7 should always be high
                  ;bits 0-3 should always be low

                  ;bits 6, 5, 4 correspond to which
                    switch and which direction
                  ;according to following table
                  ;000     Right Main Turnout
                  ;001     Right Station Turnout
                  ;010     Left Main Turnout
                  ;011     Left Station Turnout
                  ;100     Right Main Straight
                  ;101     Right Station Straight
                  ;110     Left Main Straight
                  ;111     Left Station Straight
ANDVAL  EQU $A2           ;two temporary variables
                    used in updating terminal
INVVAL  EQU $A3

SWTIT   NOP

SWTNEXT LDA V1ABUF        ;load in current outputs
                    on VIA
          AND #%00001111  ;erase the S1-S4 bits to
                    zeros
          EOR SWTDAT      ;now set the S1-S4 bits to
                    proper settings
          STA V1ABUF      ;store these in buffer
          STA V1ADAT      ;output them to VIA  (note
                    S4 is high so no switching
                    yet)

;set up Y register as number of frame 1's to
                    switch through
          LDY #$02        ;activate the switch coil
                    for 2 consecutive frame 1's
NUMF1S    DEY

WAITF1    LDA FRANUM      ;wait for frame number 1
                    to come across on HORNBY
          CMP #$01
          BNE WAITF1

          LDA V1ABUF
          EOR #%10000000  ;bring the S4 line to a
                    low (start switching switch)
          STA V1ADAT      ;output to the VIA (switch
                    coil is energized)

WAITF2    LDA FRANUM      ;wait for frame 2 to come
                    across
          CMP #$02
          BNE WAITF2

          LDA V1ABUF      ;Stop energizing the
                    switch coil
          STA V1ADAT
          .
          .
          TYA             ;test to see if done
                    switching
          BNE NUMF1S      ;if Y register is not yet
                    zero switch another frame

;Now time to update switch status board on
                    terminal
          LDA SWTDAT      ;transfer the switch data
                    to X
          LSR A           ;move the S1-S4 bits to
                    the least significant bits
          LSR A
          LSR A
          LSR A
          TAX
          LDA SWTTAB, X   ;from the table, figure
                    out which bit changes
                          ;of the SWTBUF to update
                    display
          STA ANDVAL      ;put this in a temporary
                    variable
          EOR #$FF        ;invert this and store in
                    a temporary variable
          STA INVVAL

          LDA SWTBUF      ;load in state of switches
                    before this switch
          AND ANDVAL      ;bring the bit
                    corresponding to switch
                    switched to low
          STA SWTBUF      ;store back in switch
                    buffer

          BIT SWTDAT      ;loads in S3 (switched
                    direction to V status
                    register)
          BVS DOIT        ;if overflow is set
                    (switch is straight update
                    terminal)

          LDA SWTBUF      ;if overflow was clear
                    (switch is turnout)
          EOR INVVAL      ;change the bit on the
                    SWTBUF variable
          STA SWTBUF      ;update it

DOIT      JSR SWTSTAT     ;update the terminal's
                    switch status display

          RTS

; Table : SWTTAB
;
;         The address corresponds to the 4 most
                    significant bits of the SWTDAT
;         value, and the data corresponds to a 0 in
                    the desired bit location
;         and ones everywhere else
SWTTAB    DB %11111111    ;0000    ;addresses with
                    bit 3 low not used ($FF)
          DB %11111111    ;0001
          DB %11111111    ;0010
          DB %11111111    ;0011
          DB %11111111    ;0100
          DB %11111111    ;0101
          DB %11111111    ;0110
          DB %11111111    ;0111
          DB %10111111    ;1000    Right main turnout
          DB %11101111    ;1001    Right Station
                    Turnout
          DB %01111111    ;1010    Left Main Turnout
          DB %11011111    ;1011    Left Station
                    Turnout
          DB %10111111    ;1100    Right Main
                    Straight
          DB %11101111    ;1101    Right Station
                    Straight
          DB %01111111    ;1110    Left Main Straight
          DB %11011111    ;1111    Left Station
                    Straight

;   Procedure : INITSWT
;
;         Sets all switches to straight (no
                    turnouts)
;         and switches them
INITSWT LDA #%11000000  ;switch main right switch
                    straight
          STA SWTDAT
          JSR SWTIT

          LDA #%11010000  ;switch station right
                    switch straight
          STA SWTDAT
          JSR SWTIT

          LDA #%11100000  ;switch main left switch
                    straight
```

```
        STA SWTDAT
        JSR SWTIT

         LDA #%11110000  ;switch station left
                         switch straight
        STA SWTDAT
        JSR SWTIT

        RTS              ;all switches straight

;   Procedure : FCLR
;
;       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
                         oncoming traffic
;
;force a north track clear
FCLR    LDA V1ABUF       ;load in current VIA
                         states
        AND #$F7         ;brings bit 3 to 0, (the
                         force track clear bit)
        STA V1ABUF
        EOR #$08         ;brings bit 3 to 1,
                         leaving other bits the same
        STA V1ADAT       ;output these values to
                         the VIA (forces track clear)

        LDX #$00         ;send this force clear as
                         a pulse
PAUSE   DEX              ;256 cycles long
        BNE PAUSE

        LDA V1ABUF       ;load back in same bits,
                         now with force clear low
        STA V1ADAT       ;output these normal
                         values to VIA

        RTS

    Procedure : UPRBJ

        Reads in 4 bytes of the zero page @ $B0 -
                         $B3
;       updates the track board on the terminal
                         according to their values
;  USES :
T0TRN   EQU $B0          ;train number on track 0
                         ($00 for no train)
T0STAT  EQU $B1          ;status of train on track
                         zero
T1TRN   EQU $B2          ;train number on track 1
                         ($00 for no train)
T1STAT  EQU $B3          ;status of train on track
                         one
                         ;STATUS BYTES
                         ;$00    -       EMPTY
                         (track is clear)
                         ;$01    -       ARRIVING
                         (train is arriving)
                         ;$02    -       DEPARTING
                         (train's leaving track)
                         ;$03    -       STOPPED
                         (train is safely stopped)

RTXT    EQU $0B00        ;temporary locations to
                         hold
RTXTNUM EQU $0B06        ;the train number message
                         including character addressing
RTXTST  EQU $0B0A        ;status string starting
                         location

UPRBJ   LDX #$00         ;reset counter and copy
                         train number message to RTXT
NEXTT0  LDA T0MSG,X
        STA RTXT,X
        BEQ DONET0       ;if reach end of message
                         then stop copying
        INX
        JMP NEXTT0

DONET0  LDX T0TRN        ;load in the train number
                         of train on track 0
         LDA NUM2TXT, X  ;get the ASCII character
                         corresponding to that train
         STA RTXTNUM     ;store that character in
                         proper location

         LDA #HIGH RTXT  ;display the train number
                         on terminal
         STA MESGH       ;and position for the
                         status string
        LDA #LOW RTXT
        STA MESGL
        JSR DISPMSG

         LDA T0STAT      ;load in the status of the
                         train number
         CMP #$00        ;check if its  EMPTY
         BNE NOT1        ;if not then try another
                         one

         LDA #HIGH MTMSG ;output to terminal the
                         EMPTY string message
        STA MESGH
        LDA #LOW MTMSG
        STA MESGL
        JSR DISPMSG

        JMP DT0

NOT1    CMP #$01         ;check if its ARRIVING
        BNE NOT2         ;if not try another one

        LDA #HIGH ARRMSG         ;output to
                         terminal the ARRIVING message
        STA MESGH
        LDA #LOW ARRMSG
        STA MESGL
        JSR DISPMSG

        JMP DT0

NOT2    CMP #$02         ;check if its DEPARTING
        BNE NOT3         ;if not try another one

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

        JMP DT0

NOT3    LDA #HIGH STPMSG          ;Must be STOPPED
        STA MESGH
        LDA #LOW STPMSG
        STA MESGL
        JSR DISPMSG

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

DTC     LDX #$00         ;reset counter and copy
                         train number message to RTXT
NEXTT1  LDA T1MSG,X
        STA RTXT,X
        BEQ DONET1       ;if reach end of message
                         then stop copying
        INX
        JMP NEXTT1

DONET1  LDX T1TRN        ;load in the train number
                         of train on track 1
         LDA NUM2TXT, X  ;get the ASCII character
                         corresponding to that train
         STA RTXTNUM     ;store that character in
                         proper location

         LDA #HIGH RTXT  ;display the train number
                         on terminal
         STA MESGH       ;and position for the
                         status string
        LDA #LOW RTXT
        STA MESGL
        JSR DISPMSG

         LDA T1STAT      ;load in the status of the
                         train on track 1
         CMP #$00        ;check if its  EMPTY
         BNE NT1         ;if not then try another
                         one

         LDA #HIGH MTMSG ;output to terminal the
                         EMPTY string message
        STA MESGH
        LDA #LOW MTMSG
        STA MESGL
        JSR DISPMSG

        JMP DT1

NT1     CMP #$01         ;check if its ARRIVING
        BNE NT2          ;if not try another one

        LDA #HIGH ARRMSG         ;output to
                         terminal the ARRIVING message
        STA MESGH
        LDA #LOW ARRMSG
        STA MESGL
        JSR DISPMSG

        JMP DT1

NT2     CMP #$02         ;check if its DEPARTING
        BNE NT3          ;if not try another one

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

        JMP DT1

NT3     LDA #HIGH STPMSG          ;Must be STOPPED
        STA MESGH
        LDA #LOW STPMSG
        STA MESGL
        JSR DISPMSG

DT1     RTS              ;but for now we'll just
                         end it here
```

```
      age:   T0MSG

         used to position cursor in the right spot
                    for the train number,
         and then the status (whichever it may be)

T0MSG    DB $1B, 'X&', $1B, 'Y/'          ;position
                    at line 7, col. 16
         DB 'A'                           ;train
                    number goes here
         DB $1B, 'Y6'                     ;position
                    at col. 23 now
         DB $00                           ;status
                    string goes here

;   Message:   T1MSG
;
;         used to position cursor in the right spot
                    for the train number,
;         and then the status (whichever it may be)
;
T1MSG    DB $1B, 'X*', $1B, 'Y/'          ;position
                    at line 11, col. 16
         DB 'A'                           ;train
                    number goes here
         DB $1B, 'Y6'                     ;position
                    at col. 23 now
         DB $00                           ;status
                    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
                    00:00:00
;         zero hours, zero minutes, zero seconds
                    past the origin
INITCLO NOP       ;will be added if there is time
         RTS

;   Procedure : UPCLOC
;
;         Checks the T1 counter on VIA #2, and
                    updates the clock
;         accordingly.   Should be called every $FFFF
                    microseconds
;
UPCLOC   NOP       ;will be added in if there is time
         RTS


          END
```

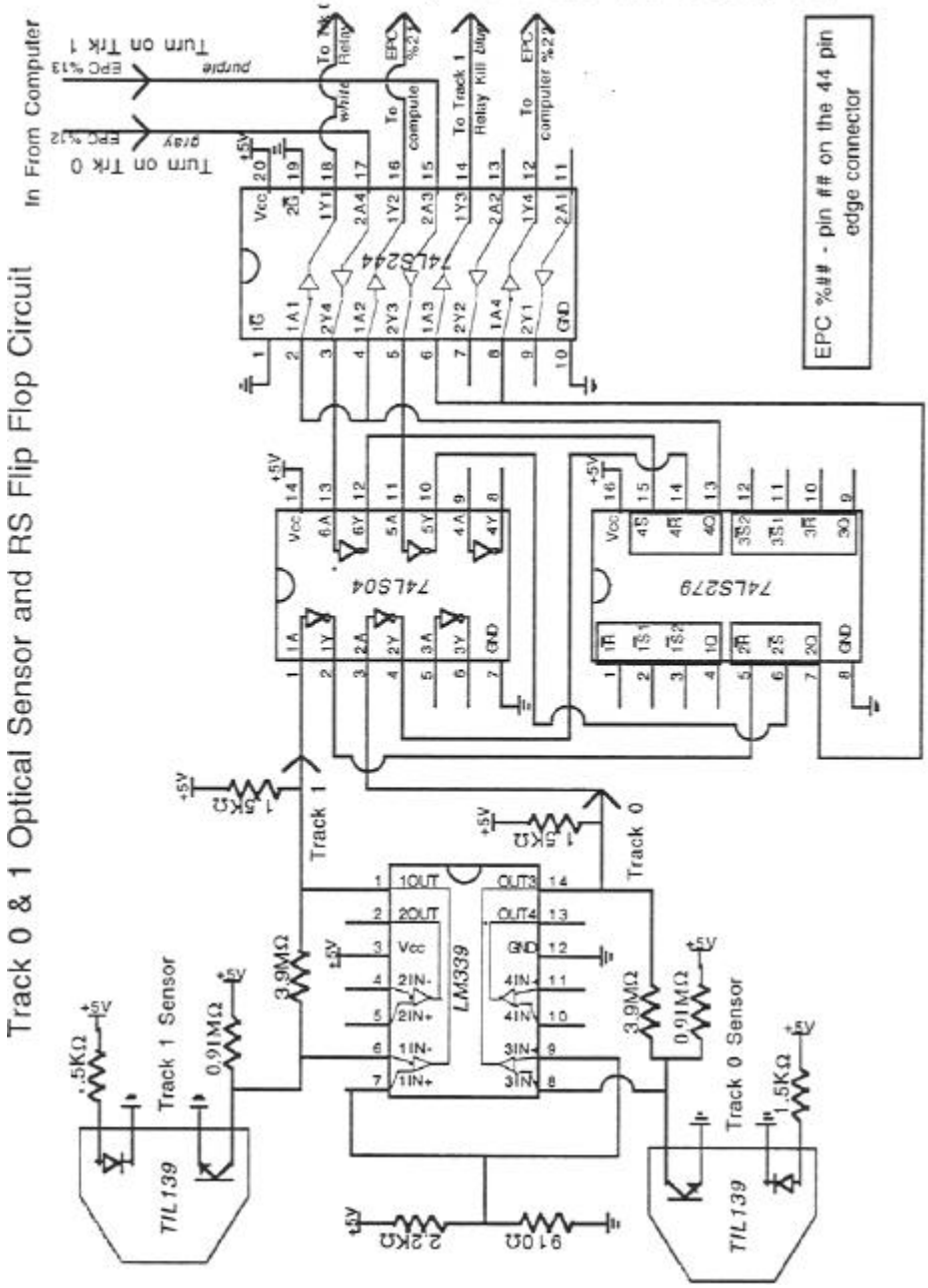
Circuit Diagrams

The following ciruit 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, thats what was used.

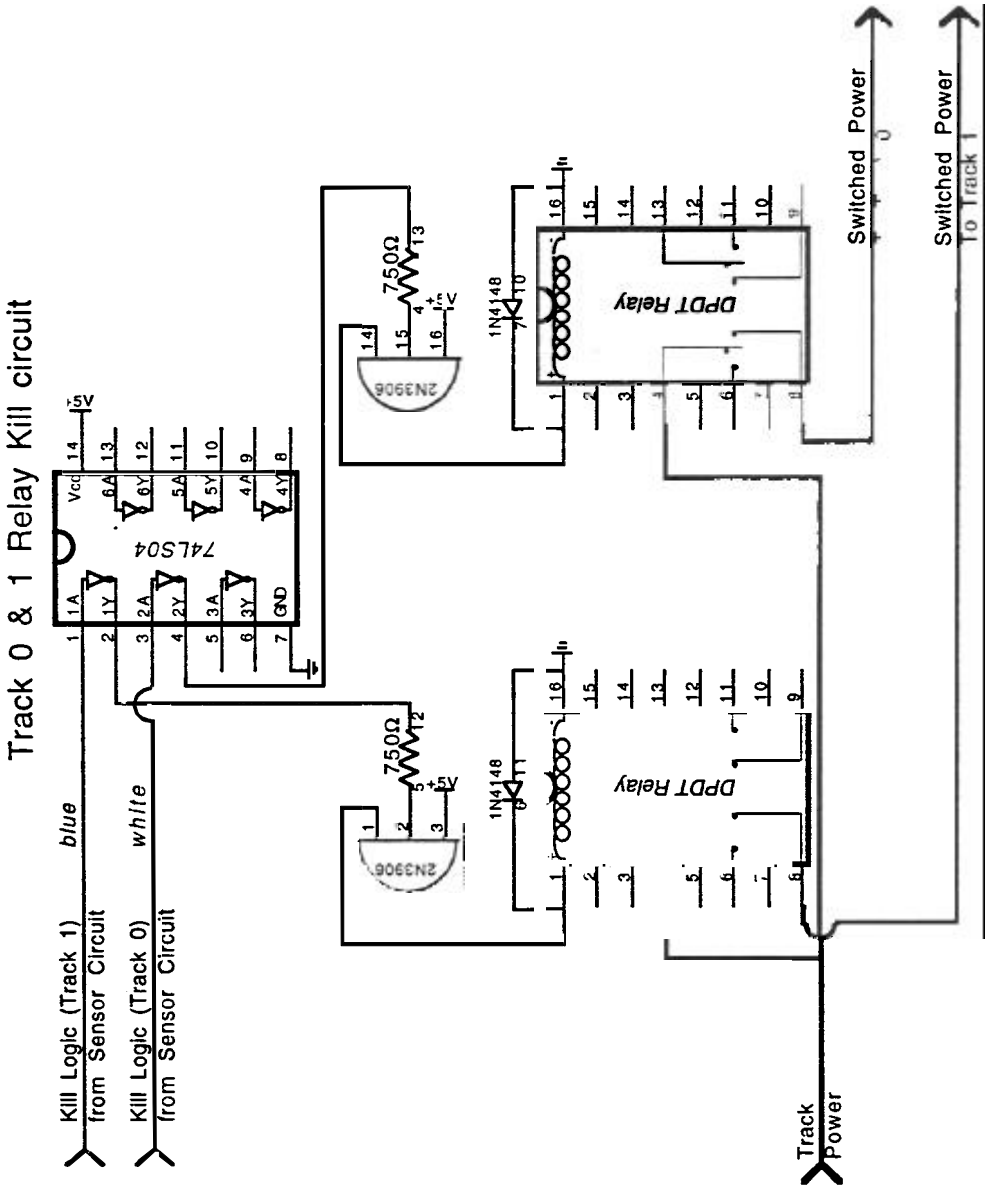
List of Circuits :

TRACK 0 & 1 OPTICAL SENSOR AND RS FLIP FLOP 40
TRACK 0 & 1 RELAY KILL 41
UART DAUGHTER BOARD 42
SWITCH ACTUATOR CIRCUIT 43
NORTH OPTICAL SENSOR 44

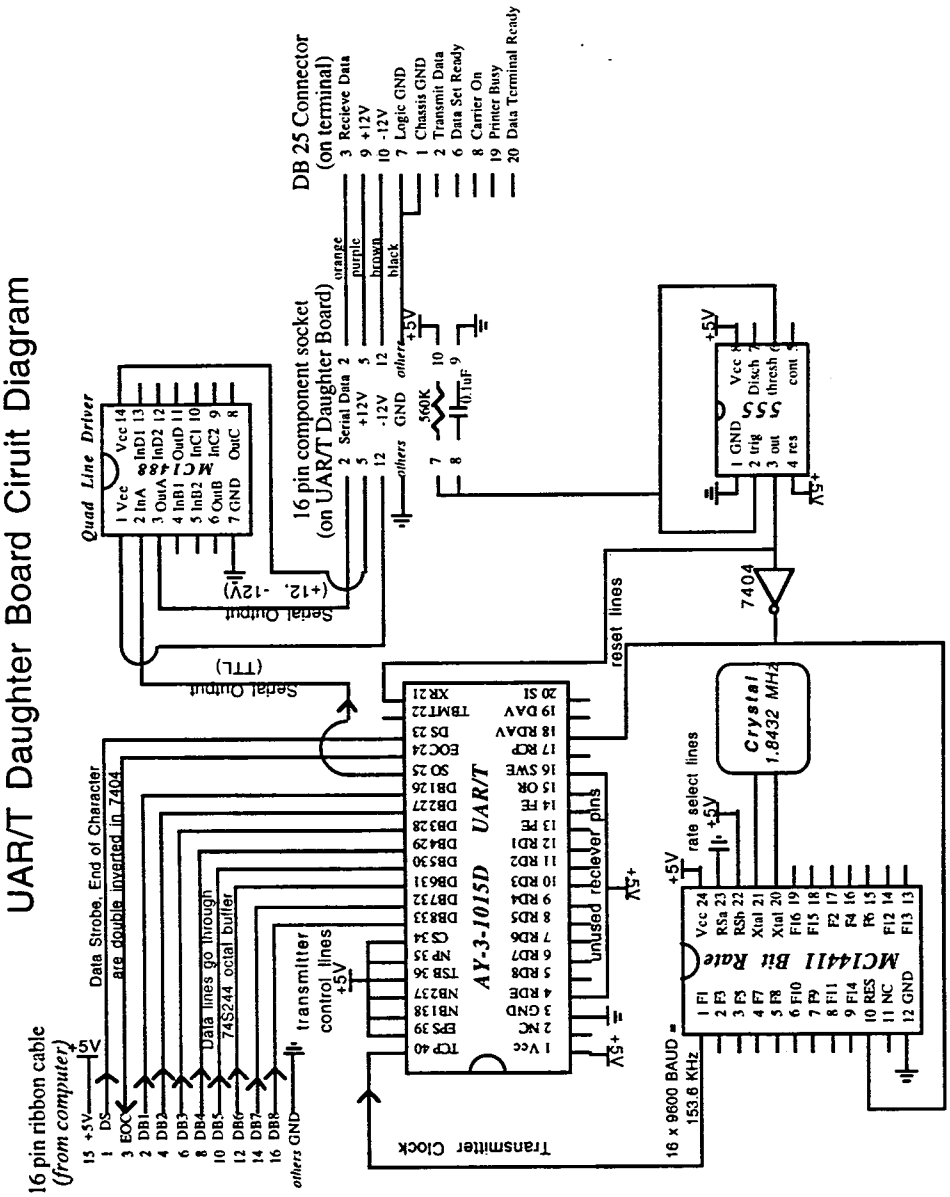
Track 0 & 1 optical sensor and RS flip flop

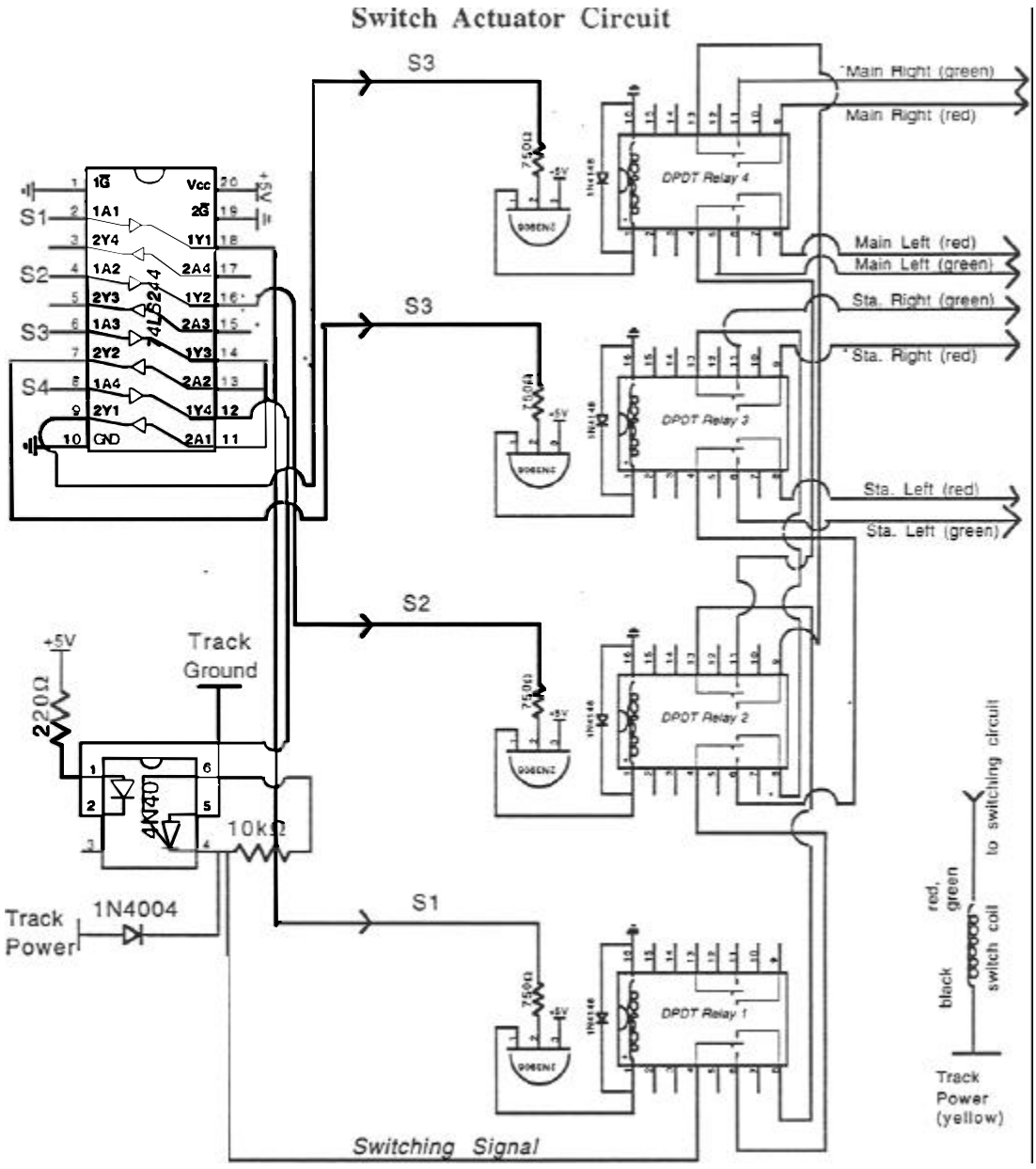


Track 0 & 1 Relay Kill

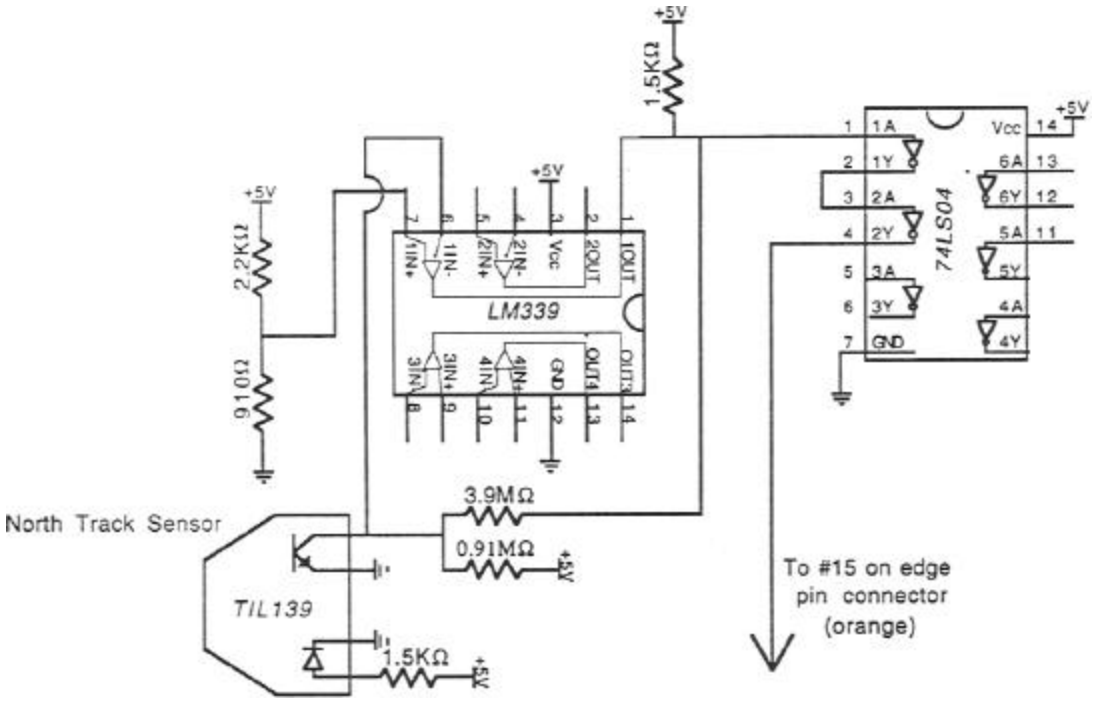


UAR/T Daughter Board




Switch Actuator Circuit
S3
S3
S2
S1
S1
S2
S3
S4
74LS244
+5V
Track Ground
220Ω
4N40
10kΩ
Track Power
1N4004
Switching Signal
DPDT Relay 4
DPDT Relay 3
DPDT Relay 2
DPDT Relay 1
Main Right (green)
Main Right (red)
Main Left (red)
Main Left (green)
Sta. Right (green)
Sta. Right (red)
Sta. Left (red)
Sta. Left (green)
to switching circuit
red, green
black
switch coil
Track Power (yellow)

North Optical Sensor



'Cops and Robbers'

Bradley Mendelson and Lina Jin
MAE 412
Professor M. Littman
May, 1996

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

Bradley Mendelson
Lina Jin

409