A

excellent report
(complete)

very solid project

1st working comput
in class

1st workins project

410

Table of Contents

1. Introduction

The objective of this project was to make use of an N-gauge model train layout to learn about microcomputer control. The requirements were that the system of components coupled with the software had to sense, actuate, and sequence. A number of physical restraints were placed upon the design. All projects had to conform with the dimensions and traffic flow of the test stand. This consisted of a 2' x 4' board with three sections of railroad track (Figure 1). The South track was that closest to the edge of the board while the North track was the furthest inside. Traffic flowed to the left on the South track and to the right on the North. A bi-directional track was positioned in between the other two to remain consistent with the possible intents of other projects which could be sequenced with ours in the future. This maintained the "standard board" philosophy which allows for all projects to be linked while without modification. The entire project interfaced with the Hornby Zero-One^{TM} control unit which provided power and data to the tracks and trains. A microcomputer using the 6502 microprocessor was constructed to allow communication with the Hornby and to interpret and react to any prescribed events (Figure 2). The computer ran the Lecky2_0 collision avoidance system and interacted with the test stand through the signal sensor board.

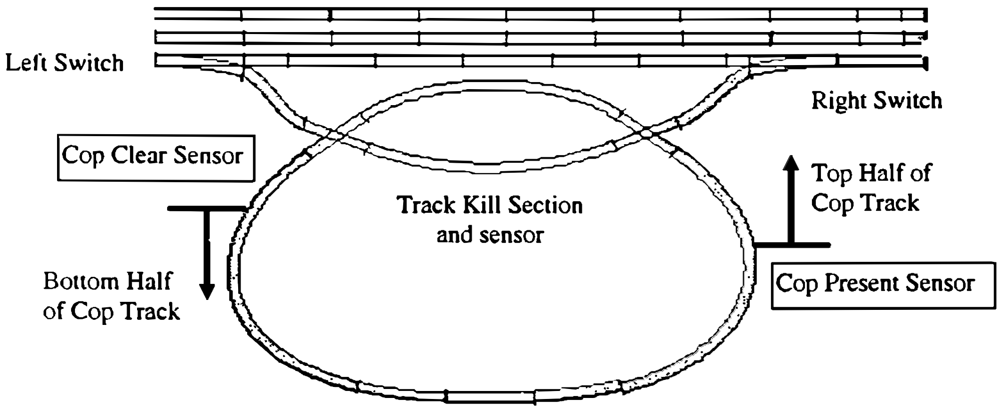
We chose to design our project so that it was capable of running almost indefinitely without input or instructions from a human controller. The problem which we tackled was essentially one of collision avoidance. Two separate flows of traffic were utilized and the task was to ensure that they never ran into one another. The primary vehicle was a standard engine ("robber"). It ran on the main lines continuously until events occurred which triggered alternate behavior. The secondary train was a small, DC powered unit ("cop") which ran on an independently powered, nearly circular track on the inside of the board.

All action contained in our project hinged upon the position of this cop car. Upon approaching our board, the microcomputer made a decision as to where to direct the robber train depending upon the location of the cop. Three different scenarios were possible. The first and most commonly observed one was that the robber train would pass through the standard board along the North track without any action taking place. This corresponded to the cop car being positioned on the front half of its circular track at the moment when the robber train passed the North track bar code sensor. If, on the other hand, the cop was instead on the back half of its track at this moment, a second event took place. These circumstances triggered two switches on the North track to their 'turn' positions so as to redirect the robber train toward the inside of the board where it crosses over the cop track. After this had occurred, two possibilities remained. Either the robber train passed unhindered through and back onto the main track ("robbing the bank") or it stopped due to a track kill and waited for power on its tracks to return ("caught at the bank"). The latter of these events only occurred if the cop car entered the front half of its tracks after the robber train had already followed the switch and entered the inside part of its route. In this case the robber waited until the cop had cleared to the back half of its track before again receiving power and heading back to the main line. Upon passing the

North track clear sensor, all switches and sensors were reset to their original values and the sequence was allowed to repeat. Below the three chains of events are summarized as according to what may happen when the robber passes the North bar code reader. The are listed in order of frequency of occurrence.

1. Cop car present on top half of track, Robber passes through without any action
2. Cop car not present, Robber switched to inside track, Cop still not present, Robber continues on and leaves ("the bank was robbed").
3. Cop car not present, Robber switched to inside track, Cop now present, Robber waits for cop to clear and then leaves ("the robber was caught robbing the bank").

The project accomplished the three goals of sensing, sequencing, and actuating. It performed its operations without any user intervention and performed reliably with a minimum of hardware components and software code.



All track edges 2.5" from left or right edge of board. South track 1.5" from top edge, Middle track 3" from top and North track 4.5" from top.

Figure 1: Track Layout on Standard Board

2. Operation

As stated, the project functioned autonomously once started. Therefore, operation was very straightforward. The necessary steps required are outlined below.

1. Place Train #1 on the main track (must use our train with a magnet attached).
2. Place the Amtrak trolley on the DC track (also had a magnet attached).
3. Power the DC track with its own +9 V supply through the appropriate circuit.
4. Set the Hornby speed to approximately 25% of full throttle.
5. Make sure that ONLY dip-switch 8 is set on the DC power supply circuit.
6. Turn power on and let everything begin.

The magnets on each of the trains was necessitated by our use of Hall sensors. Hornby speed is limited so as to enable time for on-the-fly track switching to occur. Dip-switch 8 yields the appropriate speed for the DC "cop" car. Table 1 outlines the codes seen on the microcomputer display during operation.

Code	Action Taking Place
23	Initialization
##	Number of Train Currently on North Track
BB	Robber Train Successfully Robbed Bank (arrived before Cop)
CC	Robber Train Caught at the Bank by the Cop

Table 1: Display Codes

3. Hardware

The hardware elements consisted of two railroad switches, three Hall sensors, one track kill, and one +9 V DC power supply. Each of these required one daughter board. In an attempt to minimized cross-talk and to aid in debugging, it was decided that each individual circuit should have its own board. These were attached to the main board at locations near the components which they were controlling. This helped in wiring and assembly. Circuit schematics are included in the appendix.

3.1 Microcomputer

The most essential piece of hardware was the microcomputer. Our computer varied from the standard one only in its address decode logic and buffering scheme. The basic computer layout is shown in Figure 2.

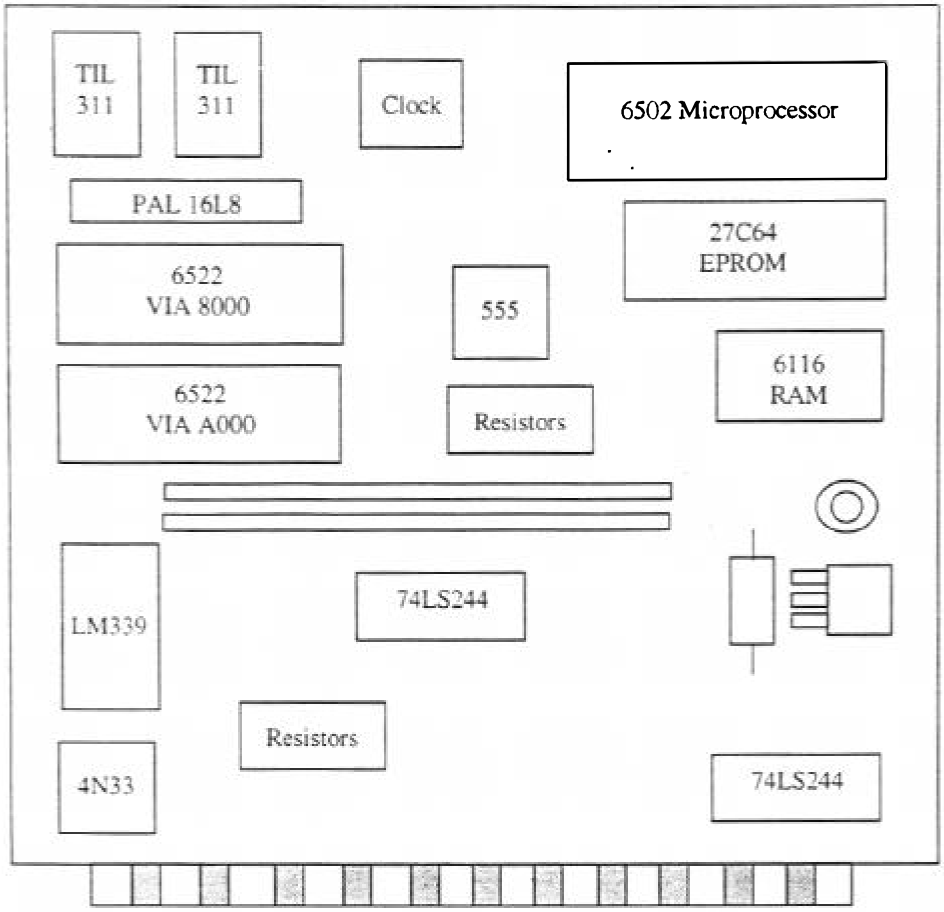


Figure 2: Microcomputer Layout

Address decode logic was performed using a Programmable Array Logic chip (PAL 16L8). Each device was differentiated from the others through the use of only two bits. This was accomplished with address lines 14 and 15. Address line 13 was used to control which VIA was accessed. Memory addresses were assigned specifically to facilitate this system. A memory map is included in the appendix. Below are the logic diagrams for each peripheral device (Figure 3) followed by the important lines of the PAL programming scheme (Table 2). The programming worksheet is included in the appendix.

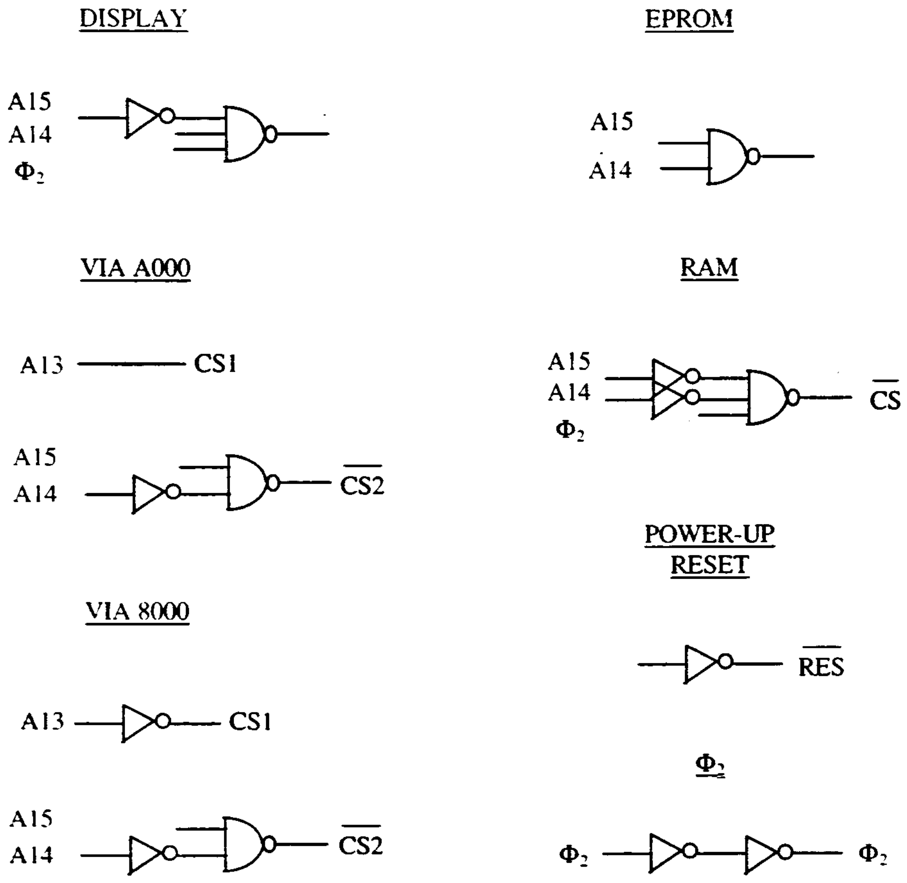


Figure 3: Address Decode Logic

For the PAL programming code, all lines other than those listed are entirely zeros except the line directly above each one listed which is all ones (i.e. L0031 is all 1's while L0033-L0286 are all 0's).

Line Number	Bit Sequence
L0032	0111 0111 1111 1111 1111 1111 1111 1111
L0288	0111 1011 1111 1111 1111 1111 1111 1111
L0544	1011 0111 0111 1111 1111 1111 1111 1111
L0800	1011 1011 0111 1111 1111 1111 1111 1111
L1056	1111 1111 1111 0111 1111 1111 1111 1111
L1312	1111 1111 1011 1111 1111 1111 1111 1111
L1568	1111 1111 1111 1111 0111 1111 1111 1111

Table 2: PAL Program Code

3.2 Daughter Board/Computer Communication

All lines into and out of the computer interfaced with one another through the 44-pin edge connector. Below is a table that describes the path that all of the input and output VIA lines traveled as they went through the buffer, 44-pin edge connector, and to the daughter boards. Only the lines unique to our project are shown.

Edge Pin	Computer	Buffer Direction	Device
3	Ground	Output	Local Ground
10	VIA 8000 B0	Input	Cop Clear/Present Sensor Line from FF
12	VIA 8000 A2	Output	Left Switch Turn
13	VIA 8000 A1	Output	Right Switch Straight
14	VIA 8000 A0	Output	Right Switch Turn
15	VIA 8000 B1	Input	Track Kill Sensor Line from FF
16	VIA 8000 A3	Output	Left Switch Straight
17	VIA 8000 A5	Output	Track Kill Sensor Set (S)
18	VIA 8000 A4	Output	Track Kill Relay
19	+5 V	None	Daughter Board Power
20	Ground	None	Daughter Board Ground
21	+5 V	None	Daughter Board Power
22	Ground	None	Daughter Board Ground

Table 3: Port and Pin Assignments

3.3 Railroad Switch Circuit

Two switches were used in this project--one left turnout and one right turnout. We encountered a great deal of difficulty with this aspect of the work. Originally, the switches chattered and sometimes did not function at all. They were extremely sensitive to track power lines. For this reason a number of circuits were tried and the following was the one finally decided upon.

Each daughter board contained two circuits, one to signal the switch inside and one to move it back out. The switches were activated from the track power. Pulses were taken from the ± 20 V power and sent to the switch coil. Prolonged current flow through the switch coil would have caused it to burn out. The track power and computer power were separated from one another by means of the 4N33 opto-isolator. This was used to ensure that components on the computer never came in direct contact with the higher voltage signals.

A TTL low logic signal was used to allow current to flow through the low voltage side of the opto-isolator. This activated a light emitting diode which in turn triggered the base in a transistor on the high voltage side and allowed current to flow; all of this took place within the 4N33 chip. The current ran from the track high power to the gate of a Silicon Controlled Rectifier (SCR). By giving a current glitch, the SCR acted as a diode for this brief moment and power was provided to the coil, thus causing it to pull the switch. Resistors were used as a current divider to ensure that the gate of the SCR received the correct level. Diodes were used in one case to ensure that current ran only one way through the 'track' side of the opto-isolator and in the other case to guard against interference from the back EMF generated across the switch coil.

Switching was not allowed to occur during track data cycles due to the possibility of data corruption. This corresponded to drawing power during frames 1 and 3 (frame 2 was negative voltage). Figure 4 shows the timing diagram for this operation.

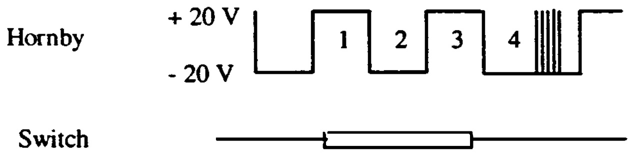


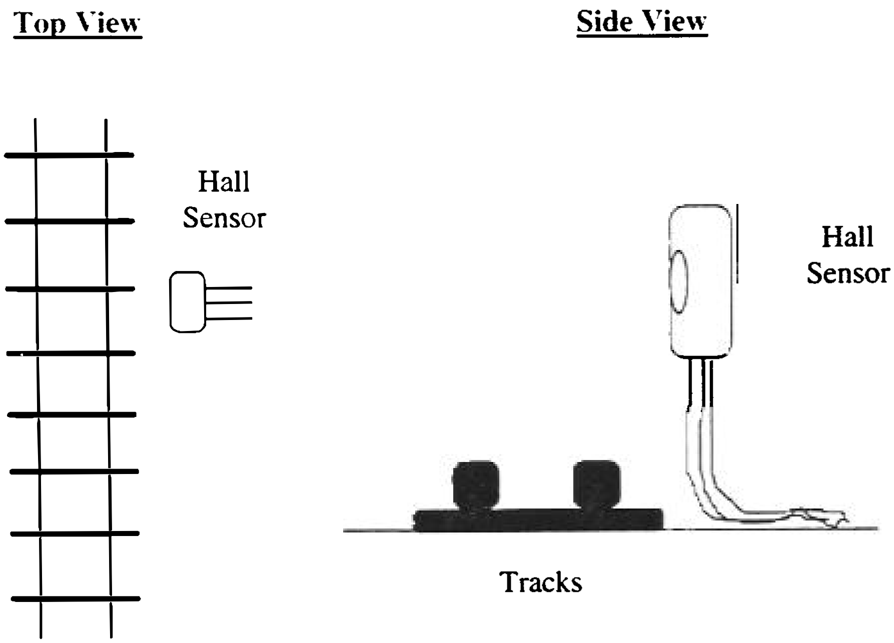
Figure 4: Switch Timing

3.4 Track Kill Circuit

The track kill circuit made use of a double-pole double-throw (DPDT) relay and a PNP transistor. A TTL low signal to the sourcing transistor's base turned it on and allowed current to flow from the collector (+5 V) through the emitter and across the relay coil to ground. This triggered the relay to switch the pin receiving track power from a state of contact with the isolated track to a no-connection. The result of this action was that the isolated section of track lost its power source and went dead. A diode was used once again to deal with the problem of back EMF.

3.5 Sensors

In an effort to maximize stability and minimize hardware, Hall sensors were chosen over optical sensors. Hall sensors have proven extremely robust in the past and require very little in the way of circuitry and assembly. Rare earth magnets were mounted on the side of each of the cars used in the project. The sensors themselves were securely fastened to the board with the sensor face approximately one inch above the board and to the side of the track (Figure 5). We considered placing the sensors under the tracks and attaching magnets to the bottom of the cars, but due to the design of the DC trolley this was not possible.


Top View

Side View

Figure 5: Sensor Placement

3.6 Cop Car Position Sensors

Two Hall sensors were employed to determine the position of the DC cop car. The outputs from these sensors were connected to an R-S flip flop which was constructed out of two NAND gates. The output from the cop clear sensor acted as the reset (R) while the output from the cop present sensor acted as set (S). Both of these lines were tied to pull-up resistors and remained high unless a magnet was in front of the sensor. When a magnet passed the clear sensor, the R line momentarily went low and this set the flip flop in a low state. On the other hand, when the present sensor line pulsed low, this caused the flip flop output to return to high. Our project required that we preset the original state of the flip flop. Although at power up both R and S lines are high and the flip flop is in a quiescent state, we used a capacitor between the opposite output and ground to ensure that the correct output always started in a high state. This flip flop output line went to the VIA as an input.

	Reset (R)	Set (S)	Input to VIA (Q)
both start high	1	1	quiescent
clear	0	1	0
set	1	0	1

Table 4: R-S Flip Flop States

3.7 Track Kill Sensor

A similar scheme to the one above was utilized for the track kill sensor. Everything was the same except the S line of the flip flop. Instead of having a Hall sensor determining the state of that line, we instead used a VIA output signal. In this case a magnet in front of the sensor set the flip flop in its low state and then a high-low-high pulse from the VIA returned the flip flop to its original high state. The output once again acted as a VIA input.

3.8 DC Power Supply Circuit

In order to supply DC power to the cop car's tracks, we used a +9 V unit that plugged into the wall socket ("wall wart"). In use it was discovered that the supply actually provided over 11 V so a circuit was designed which would give us flexibility to limit the voltage which we actually placed across the tracks. A PNP sourcing power transistor was used to limit the supply current drawn out of the power supply. When current flowed through the base, this enabled a multiplied amount of current to pass through from the emitter to the collector and on to the tracks.

Resistors were placed between the transistor base and ground so as to limit the base current and in doing so also limit the collector current. The 2N2955 has a β of approximately 10 so this enabled us to limit the collector current to less than 1 A through the use of 4-510 ohm resistors in parallel. Assuming a voltage of 11.5 V from the power pack, this restricted the base current to 90 mA and thus the collector current to 0.9 A. The resistors were placed in parallel and 0.5 W resistors were used to deal with the problem of excessive power dissipation. All four of them combined needed to handle approximately 1 W so this scheme was adequate.

In order to limit the voltage across the tracks, a number of different techniques were tried. The easiest solution would have been to place a variable resistor between the collector and track so as to provide a voltage drop before the tracks. The choice of a variable resistor would also have been advantageous because they allow for high power dissipation. When this solution was tried, it proved unsuccessful because a variable transistor in the appropriate range was not available. Due to the fact that the cop car resistance was only 30 ohms, a small variable resistor on the order of 10-15 ohms was needed. Unfortunately this was not available.

An alternative solution to this problem which we eventually utilized was to use a set of 8 switches in a dip switch package. This allowed us to easily switch between the sets of resistors that were placed in series with the collector and track. Depending upon which switch was turned on, a different set of four resistors in parallel dropped the voltage. Once again four resistors were used in parallel to prevent the resistors from overheating. Although inelegant, this solution proved very effective as we were able to vary the track voltage from anywhere between 7.5 V and 11.5 V.

4. Software

The code was written in assembly language for the 6502 microprocessor, and compiled with the XASM65 cross-assembler. The compiled hex version of the code was then burned into a 27C64 EPROM (Erasable Programmable Read Only Memory) chip using GTEK hardware and software.

The software was written to be as simple and straightforward as possible. This was done through a main program which calls major subroutines in sequence. The flowchart in the appendix shows the organization of the main program and subroutines, and the reasoning behind the code will also be explained briefly below.

4.1 Initialization

The Lecky routine, which automatically initializes the $A000 Virtual Interface Adapter (VIA), was incorporated into the code via a %include statement. The Lecky routine is initialized so that the robber train speed and direction can be controlled by the Hornby control panel. In addition, the $8000 VIA ports directions were set so that port B read input data and port A sent output data. Port A was also preset to high, or FF, essentially presetting all bits to daughter boards high. This enabled the use of pulses to the daughter boards that go low and then high.

4.2 Main Sequence ("Sequencing")

Essentially, the main sequence called major subroutines and then repeated itself, allowing the train to run almost indefinitely. First, it called a subroutine that preset the switches to main track. Then the program entered a loop called WAIT1 where it repeatedly read the mailbox NTRAIN, a Lecky program variable that contains the bar code last read by the bar code sensor, until the number in NTRAIN matched the number in ROBID (the robber train bar code number). In other words, the program stayed in this waiting loop indefinitely until the train that last passed the bar code sensor was the robber train.

The program then called the subroutine COPCH, which set the zero-bit according to the state of the cop position sensors. If the cop was nearby, i.e. the zero-bit was low, the train traversed the main north track without any switches being thrown. The program simply waited in another loop, CLCHK, until the value of NTRAIN showed that the North track was cleared before starting all over again at WAIT1.

However, if the cop was not nearby, i.e. if the zero-bit was high, the program jumped to the marker GO within the main sequence. The first step in this sequence was to set the switches to the inside track. It then called WAIT2, another loop, in which the program waited for the train to reach the track kill sensor.

In the next subroutine, ROBRY, the program essentially checked if the cop had crossed into the near half of the independently powered track while the robber was on his way to the bank, *after* the switch was thrown. If the cop was not "nearby" at this time, this part of the sequence ends quickly, and the track power is restored with imperceptible pause in the robber train's movement from the track kill in the subroutine ENDRB. If the cop was nearby, though, the robber train waited on the killed track until the cop had passed into the far half of the track again. This was done in the subroutine ENDRB with a loop that waited for the cop to clear by repeatedly calling COPCH. The track power was then restored. In both cases, ENDRB was called, but the results were still different because the timing was different. It was done this way instead of completely separate programs for the two scenarios because it was simpler and more elegant a solution. The only other difference was in the TIL display, which indicated a "BB" if the robber was successful (former scenario) and a "CC" if the robber had been "caught" (latter scenario with visibly killed track). Finally, for both scenarios, after track power was restored and the robber train on its way again, the program waited in the loop CLCHK before proceeding to reset the switches and track kill sensor.

The program then jumped back to the start of the loop WAIT1 to start all over again. It does not need to start before this point because the track kill sensor and switches were already reset in ENDRB.

4.3 Subroutines

The following is a list and description of the subroutines. Included where necessary are tables showing the bit mask or bit map as well as the operation performed in that routine. An 'x' in a bit position indicates a value that can be either 0 or 1. When referring to the VIA $8000 port lines, bit 0 is the least significant bit while bit 7 is the most significant.

COPCH: Checks if the cop is in the half of the independently powered track near the north track. This is done by reading the output of the RS flip-flop connected to the cop position sensors. Essentially, if the output is high, the flip flop has been "set" by the cop position sensor preceding the near half of the independently powered track. If the output is low, the flip flop has been "reset" by the cop position sensor preceding the far half of the track. To read the output, though, the cop state bit, bit 0, must be isolated from the rest of the bits loaded into VIA $8000 port B. Bit 0 is then compared to hex number #01, in order to set the zero-bit. This is useful for using commands like BNE and BEQ.

VIA 8000 Input	xxxx xxxx
AND	0000 0001
Result	0000 000x
Compare	0000 0001

WAIT2: Waits for train to reach track kill sensor. This is done by a loop that repeatedly reads bit of the input to VIA $8000 port B (in a similar way that bit 0 is read and compared in COPCH, except bit 1 is compared to the hexadecimal number #02) until the first-bit is set low.

VIA 8000 Input	xxxx xxxx
AND	0000 0010
Result	0000 00x0
Compare	0000 0010

ROBRY: This subroutine first calls COPCH, and if the zero-bit is set low, i.e. the cop is near, then the program jumps to the marker CAUGHT, and "CC" is loaded into the display as a signal that the robber has been caught. If the zero-bit is high, then the program continues and "BB" is loaded into the displays a signal that the robber has been successful in the robbery.

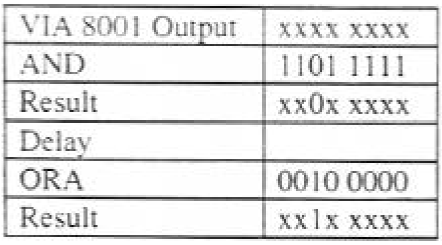
TPOW: Returns power to the isolated portion of the track. The track power bit, bit 5, is set high by taking the OR of VIA $8000 port A output and the hexadecimal number #10.

VIA 8001 Output	xxxx xxxx
ORA	0001 0000
Result	xxx1 xxxx

TKILL: Kills power to track. The track power bit, bit 5, is set low by taking the AND of VIA S8000 port A output and the hexadecimal number #EF.

VIA 8001 Output	xxxx xxxx
AND	1110 1111
Result	xxx0 xxxx

TKSON: Sets track kill sensor ready by setting the track kill sensor bit (bit 6) on VIA S8000 port A output low and then high, in a pulse. Like in TPOW and TKILL, the bit is set low and high while preserving other bit values in port A. A delay subroutine was used before the pulse was returned high, because it was found that otherwise the pulse was of too short duration to be detected by the track kill sensor.



VIA 8001 Output	xxxx xxxx
AND	1101 1111
Result	xx0x xxxx
Delay	
ORA	0010 0000
Result	xx1x xxxx

CLCHK: Waiting loop for train to pass clear sensor. This loop repeatedly reads the value of NTRAIN, the bar code number of the last train read by the bar code reader, until NTRAIN is 0.

SWIN: Sets both switches to inside track, which crosses over the independently powered loop that the cop train is on. This is done by first calling GETF1, which is a wait loop for power frame 1. The switch bits, bits 0 and 2, are set low by taking the AND of VIA $8000 port A and hex number #FA. Like in TKILL, this preserves other bit values in port A. To end the pulse, first GETF3 is called, and then the switch bits are set high by taking the OR of VIA $8000 port A and hex number #0F. This automatically sets all bits related to switching (bits 0-3) high again, while preserving the other bits (bits 4-7). Actual switching can only be done during a power frame (frames 1-3), because otherwise transmission during the data frame (frame 4) might disrupt communication in the Hornby system. We allowed for time of one frame in between the two because the switches, like the track kill sensor, require a pulse of two frames to function.

VIA 8001 Output	xxxx 1111
AND	1111 1010
Result	xxxx 1010
Draw Power (FRANUM 1-3)	
ORA	0000 1111
Result	xxxx 1111

SWOUT: Sets both switches to main north track. This is done in a way similar to that described in SWIN, but with bits 1 and 3 being set low with hex number #F5.

VIA 8001 Output	xxxx 1111
AND	1111 0101
Result	xxxx 0101
Draw Power (FRANUM 1-3)	
ORA	0000 1111
Result	xxxx 1111

GETF1: Waits for power frame 1. Repeatedly checks Hornby mailbox FRANUM until equal with the number #01

GETF3: Waits for power frame 3. Repeatedly checks Hornby mailbox FRANUM until equal with the number #03.

ENDRB: Waits until cop has cleared the near half of the independently powered track, restores power to track, waits for train to reach clear sensor, resets switches to the main track and track kill sensor ready again. This is done with a loop that calls COPCH repeatedly until the zero-bit is set high. Then calls TPOW, CLCHK, SWOUT, and TKSON.

DELAY: Runs a delay loop by counting down from FF to 00. DEX decrements X by 1, while BNE repeats the loop until the zero-bit is set, when 00 is finally reached. Similarly, DEY decrements Y until 00 is reached.

5. Conclusion

Overall this project was a success. The goals as originally stated were accomplished in their entirety. Our project operated without glitches, and thus successfully accomplished the requirements of sensing, actuating, and sequencing. However, due to its simplicity, it might be desired to add other aspects of sensing, actuating, and sequencing to the current system.

One suggestion for improvement would be the addition of a new track to the inner track for the robber train to be sent to if it gets "caught" by the cop train. This track could have, at the end of it, a jail with a door that is lowered once the robber train has entered the jail. Such a system would require a hall sensor and track kill at the entrance to the jail, a DC motor and cable attached to the jail door, and a switch at some point on the inner track between the bank and left switch. The track in the vicinity of the jail would need to be electrically isolated from the rest of the track. In addition, a separate power supply would need to be sent to the DC motor. This system would also require a new subroutine to run the DC motor to lower the cable and jail door to the exact position.

Light Emitting Diodes (LED's) could be placed in the vicinity of the bank, and controlled by the main computer so that the LED's are turned on if the robbers have successfully robbed the bank. This circuit might also need to draw a separate power supply modulated by transistors if the LED's require more than +5 V to operate.

Several LED's could also be placed around the independently powered track in conjunction with a microprocessor timer to predict the position of the cop train with greater precision. In particular, when the train is in a certain segment of the track, the LED nearest to that segment would be lit. Because there would be several LED's, it would be more efficient to assign addresses to them and use address code logic with a few bits and a PAL, rather than assign bits from the VIA $8000 ports to every LED. The program would also need to be modified to check the timer and change the states of the LED's regularly, but not so often that it disrupts the smooth functioning of the sequencing part of the program. In addition, if there is error, the program could act as a feedback process and somehow adjust the timer or the LED being lit when the train passes the hall sensors.

Appendix

A.1 Memory Map

The following is a map of the memory for the microcomputer. Hexadecimal addresses are on the left along with their binary counterparts on the right. One can easily observe how the two highest order bits were used to specify the device. The VIA's are distinguished by the third highest order bit.

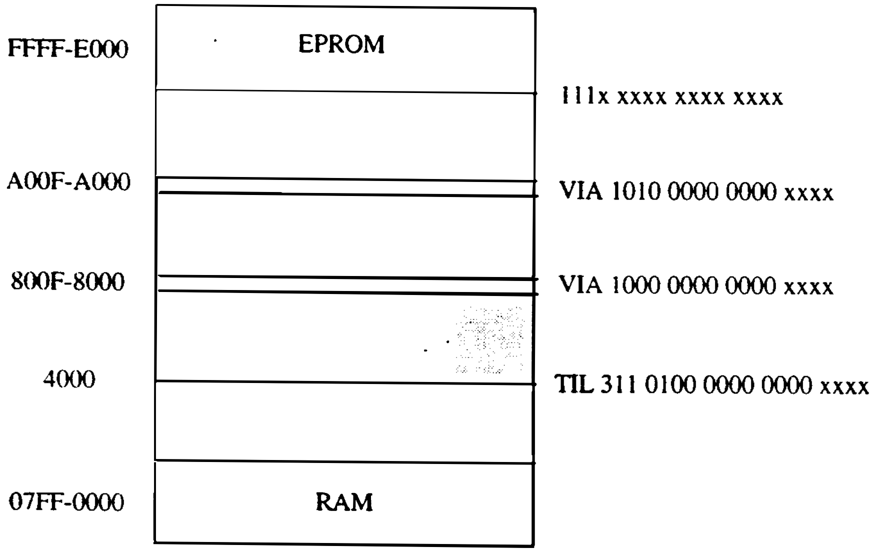
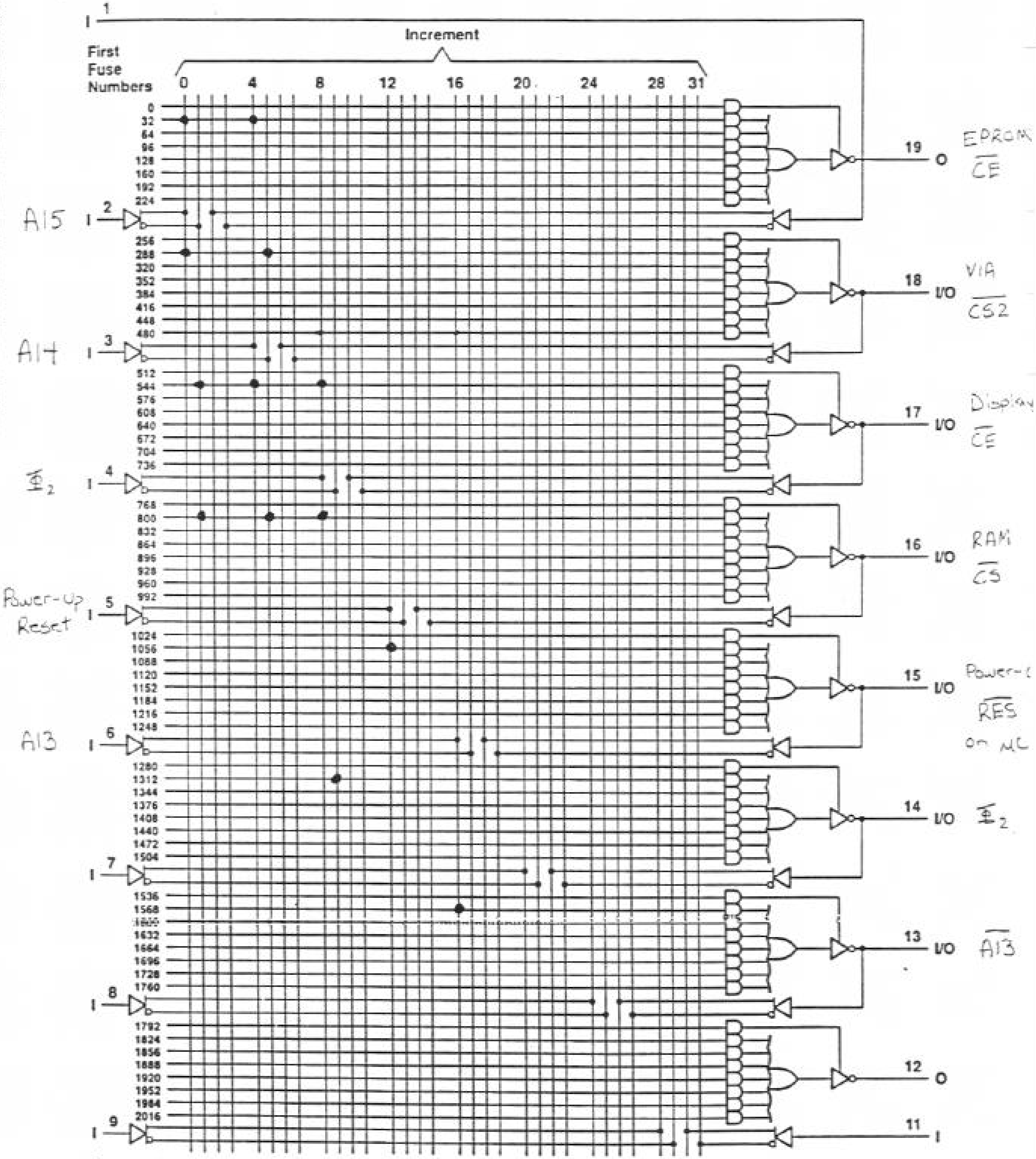


Figure A-1: Memory Map

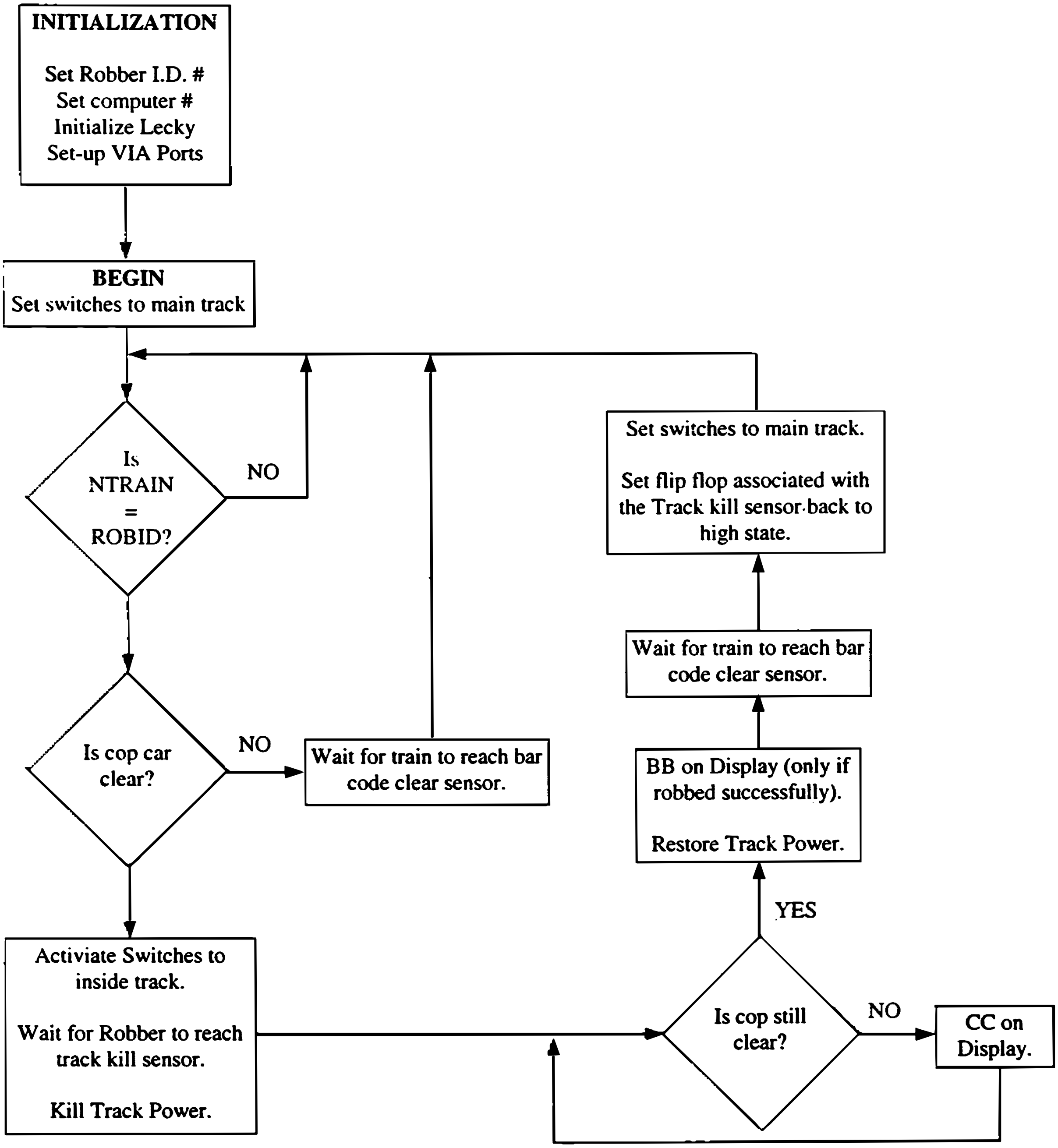
A.2 PAL Worksheet

logic diagram (positive logic)



Fuse number = First fuse number + Increment

A.3 Program Flow Chart


INITIALIZATION
Set Robber I.D. #
Set computer #
Initialize Lecky
Set-up VIA Ports
BEGIN
Set switches to main track
Is NTRAIN = ROBID?
NO
Is cop car clear?
NO
Wait for train to reach bar code clear sensor.
Activiate Switches to inside track.
Wait for Robber to reach track kill sensor.
Kill Track Power.
Is cop still clear?
NO
CC on Display.
YES
BB on Display (only if robbed successfully).
Restore Track Power.
Wait for train to reach bar code clear sensor.
Set switches to main track.
Set flip flop associated with the Track kill sensor back to high state.

A.4 Computer Code BLTRAIN2.ASM

```
;CODE FOR TRAIN ROBBERY ROUTINE
;BY BRADLEY MENDELSON AND LINA JIN
;MAE 412 TRAIN TRACK THAT SENSES, ACTUATES, AND SEQUENCES

%INCLUDE LECKY2_0.ASM

;VIA $8000 ADDRESSES
V8BI     EQU $8000      ;DATA REGISTER B INPUT
V8BD     EQU $8002      ;DATA REGISTER B DIRECTION
V8AO     EQU $8001      ;DATA REGISTER A OUTPUT
V8AD     EQU $8003      ;DATA REGISTER A DIRECTION

;LOCAL VARIABLES
DISP     EQU $4000      ;TIL DISPLAY
ROBID    EQU $0200      ;ROBBER TRAIN ID

 INITIALIZATION
         ORG $E000
         SEI
         CLD
         LDX #$FF       ;LOAD ALL 1'S INTO STACK
         TXS
         LDA #$01       ;LOAD ROBBER TRAIN ID
         STA ROBID      ;STORE ROBBER TRAIN ID IN MEMORY
         LDA #$02       ;LOAD BLOCK COMPUTER ID
         STA BLKID      ;STORE BLOCK COMPUTER ID IN HORNBY
         JSR INIT       ;INITIALIZE LECKY ROUTINE
         LDA #$00
         STA V8BD       ;SET DATA REGISTER B TO READ INPUT
         LDA #$FF
         STA V8AD       ;SET DATA REGISTER A TO SEND OUTPUT
         STA V8AO       ;PRESET DATA REGISTER A ALL 1'S

;SEQUENCING
BEGIN    LDA #$23
         STA DISP       ;"23" ON DISPLAY--SEQUENCING BEGUN
         JSR SWOUT      ;PRESET SWITCHES TO MAIN TRACK
WAIT1    LDA NTRAIN     ;READ TRAIN ON N TRACK
         STA DISP       ;DISPLAY
         CMP ROBID      ;CHECK IF TRAIN IS ROBBER
         BNE WAIT1      ;IF NOT, WAIT AGAIN FOR TRAIN
         JSR COPCH      ;ELSE CHECK FOR COP
         BNE GO         ;IF NO COP, HEAD FOR INSIDE TRACK
         JSR CLCHK      ;ELSE DO NOTHING UNTIL N TRACK CLEAR
         JMP WAIT1      ;THEN WAIT AGAIN FOR TRAIN
GO       JSR SWIN       ;SET SWITCHES TO INSIDE TRACK
         JSR WAIT2      ;WAIT FOR TRAIN TO REACH TRACK KILL
```

```
                        ;SENSOR
         JSR TKILL      ;CUT OFF POWER TO INSIDE TRACK
         JSR ROBRY      ;IF NO COP, PROCEED TO ROB BANK (BB)
         JSR ENDRB      ;WAIT FOR TRAIN TO CLEAR N TRACK,
                        ;RESET ALL
         JMP WAIT1      ;START OVER

;SUBROUTINES

;READ STATE OF COP, I.E. CHECK IF COP NEARBY
COPCH    LDA V8BI       ;READ VIA INPUT
         AND #$01       ;ISOLATE COP STATE BIT
         CMP #$01       ;CHECK IF COP NEARBY (BIT IS HIGH)
         RTS

;WAIT FOR TRAIN TO REACH TRACK KILL SENSOR
WAIT2    LDA V8BI       ;READ VIA INPUT
         AND #$02       ;ISOLATE TRACK KILL SENSOR STATE BIT
         CMP #$02       ;CHECK IF TRAIN PASSED SENSOR (BIT
                        ;IS LOW)
         BEQ WAIT2      ;IF NOT, READ VIA AGAIN
         RTS

;ROBBERY (IF COP THERE, WAIT THEN LEAVE, ELSE ROB BANK THEN
;LEAVE)
ROBRY    JSR COPCH      ;CHECK IF COP NEARBY
         BEQ CAUGHT     ;IF SO, YOU'RE CAUGHT, END SEQUENCE
         LDA #$BB
         STA DISP       ;ELSE "BB" ON DISPLAY INDICATES
                        ;ROBBERY BEGUN
         JMP DONE       ;ROBBERY SUCCESSFUL, GO TO END
CAUGHT   LDA #$CC
         STA DISP       ;"CC" INDICATES YOU'RE CAUGHT
DONE     RTS            ;PROCEED TO END ROBBERY

;TRACK RETURN POWER (CLEAR TRACK KILL)
TPOW     LDA #$10
         ORA V8AO       ;SET TRACK KILL BIT HIGH
         STA V8AO       ;VIA OUTPUT (RETURN POWER)
         RTS

;KILL POWER TO TRACK
TKILL    LDA #$EF
         AND V8AO       ;SET TRACK KILL BIT LOW
         STA V8AO       ;VIA OUTPUT (KILL POWER TO TRACK)
         RTS

;SET TRACK KILL SENSOR READY
TKSON    LDA V8AO       ;READ VIA DATA A OUTPUT
         AND #$DF       ;SET TRACK KILL SENSOR BIT LOW
         STA V8AO       ;VIA OUTPUT (START PULSE LOW)
         JSR DELAY
         ORA #$20       ;SET TRACK KILL SENSOR BIT HIGH
         STA V8AO       ;VIA OUTPUT (END PULSE)
         RTS

;CHECK CLEAR SENSOR
CLCHK    LDA NTRAIN      ;READ N TRACK TRAIN ID FROM HORNBY
         CMP ROBID       ;CHECK IF ROBBER TRAIN STILL INSIDE
         BEQ CLCHK       ;IF SO, REPEAT CHECK
         RTS

;SET SWITCHES TO INSIDE TRACK
SWIN     JSR GETF1       ;WAIT FOR FRAME 1
         LDA #$FA        ;LOAD BITMAP FOR SWITCH-IN LINES LOW
         AND V8AO        ;LEAVE ALL BITS SAME EXCEPT SWITCH
                         ;ONES
         STA V8AO        ;VIA OUTPUT (START PULSE LOW)
         JSR GETF3       ;WAIT FOR FRAME 3
         LDA #$0F        ;LOAD BITMAP FOR ALL SWITCH BITS
                         ;HIGH
         ORA V8AO        ;RESET SWITCH OUTPUTS HIGH
         STA V8AO        ;VIA OUTPUT (END PULSE)
         RTS

;SET SWITCHES TO STRAIGHT TRACK
SWOUT    JSR GETF1       ;WAIT FOR FRAME 1
         LDA #$F5        ;LOAD BITMAP FOR SWITCH-OUT LINES
                         ;LOW
         AND V8AO        ;LEAVE ALL BITS SAME EXCEPT SWITCH
                         ;ONES
         STA V8AO        ;VIA OUTPUT (START PULSE LOW)
         JSR GETF3       ;WAIT FOR FRAME 3
         LDA #$0F        ;LOAD BITMAP FOR ALL SWITCH BITS
                         ;HIGH
         ORA V8AO        ;RESET SWITCH OUTPUTS HIGH
         STA V8AO        ;VIA OUTPUT (END PULSE)
         RTS


;CHECK FRAME NUMBER
GETF1    LDA #$01
         CMP FRANUM      ;CHECK HORNBY MAILBOX IF FRAME
         BNE GETF1       ;IF NOT, REPEAT CHECK
         RTS

GETF3    LDA #$03
         CMP FRANUM      ;CHECK IF FRAME 3
         BNE GETF3       ;IF NOT, REPEAT CHECK
         RTS

;WAIT FOR TRAIN TO CLEAR N TRACK, THEN RESET SWITCHES AND
;SENSOR
ENDRB    JSR COPCH       ;CHECK IF COP OUT OF WAY
         BEQ ENDRB       ;LOOP UNTIL COAST IS CLEAR
         JSR TPOW        ;CLEAR TRACK KILL
         JSR CLCHK       ;WAIT FOR TRAIN TO REACH CLEAR
                         ;SENSOR
```

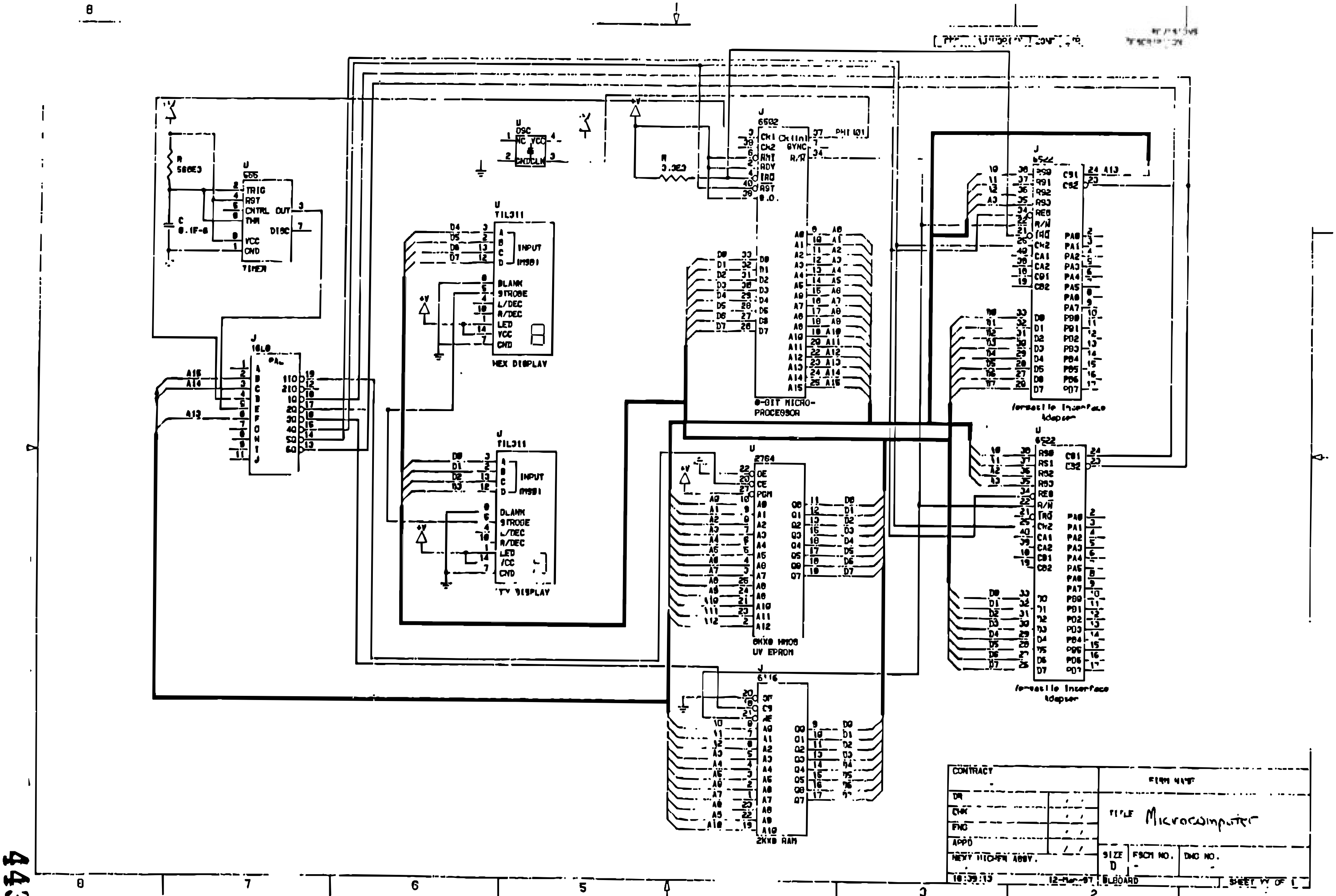
```
          JSR SWOUT       ;SET SWITCHES TO MAIN TRACK
          JSR TKSON       ;SET TRACK KILL SENSOR READY AGAIN
          RTS

;DELAY SUBROUTINE
DELAY     DEX
          BNE DELAY
          DEY
          BNE DELAY
          RTS

          END
```

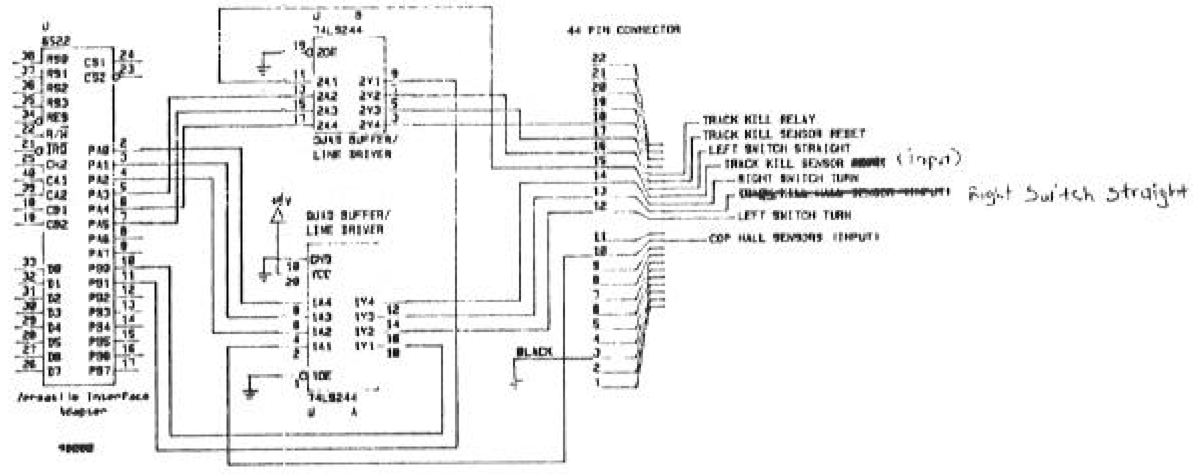
A.5 Circuit Diagrams

1. Microcomputer
2. VIA $8000 to Buffer to 44-Pin Edge Connector
3. Switch Circuit
4. Track Kill Circuit
5. Cop Car Position Circuit and Sensors
6. Track Kill Sensor
7. DC Power Supply Circuit

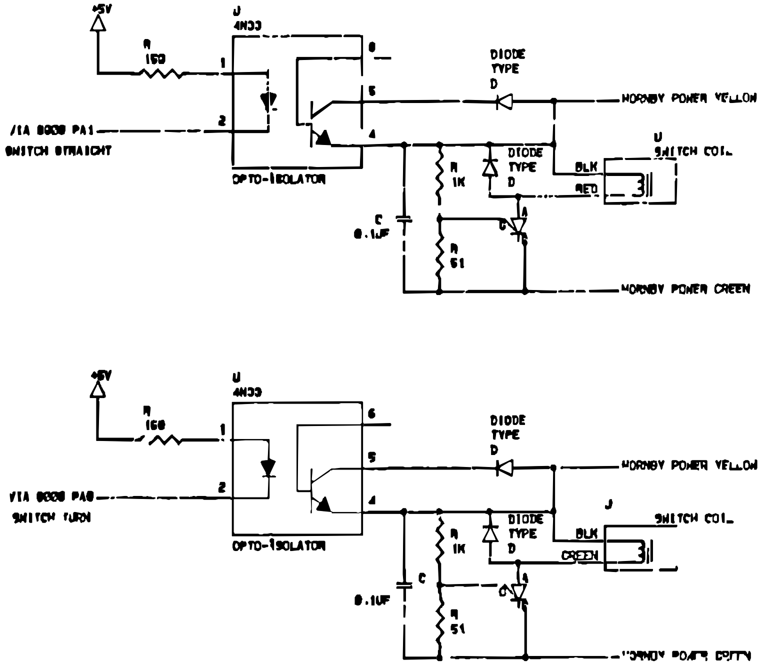

6502
8-BIT MICRO-PROCESSOR
6522
Versatile Interface Adapter
6522
Versatile Interface Adapter
2764
8KX8 NMOS UV EPROM
6116
2KX8 RAM
TIL311
HEX DISPLAY
TIL311
DISPLAY
555
TIMER
16L8
PAL
OSC
R 3.3E3
R 560E3
C 0.1F-6
INPUT
BLANK
STROBE
L/DEC
R/DEC
LED
VCC
GND
A15
A14
A13
PHI 01
CONTRACT
DR
CHK
ENG
APPD
NEXT HIGHER ASSY.
FIRM NAME
TITLE Microcomputer
SIZE D
FSCM NO.
DWG NO.
10:39:13
12-Mar-87
SBBOARD
SHEET OF

REVISIONS

LTR	ZONE	DESCRIPTION	DATE	APPROVED



CONTRACT -		FIRM NAME	
DR	/ /	TITLE VIA 8000 - Buffer - 44 pin Edge Connector	
CHK	/ /		
ENG	/ /		
APPD	/ /		
NEXT HIGHER ASSY. -		SIZE D	FSCM NO. -	DWG NO. -
18:49:26	21-May-97	EDGE	SHEET XX OF 1



REVISIONS

REV | ZONE | LTR | DESCRIPTION | DATE | APPROVED

CONTRACT -

FIRM NAME

DN

CHK

ENG

APPD

TITLE Switch Circuit

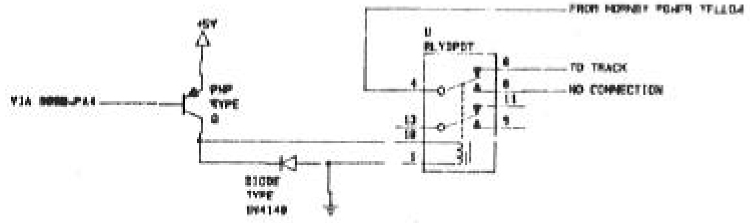
NEXT HIGHER ASSY.

SIZE D | FSCM NO. - | DWG NO. -

23:37:18 22-May-97 | TRSN | SHEET XX OF 1

REVISIONS

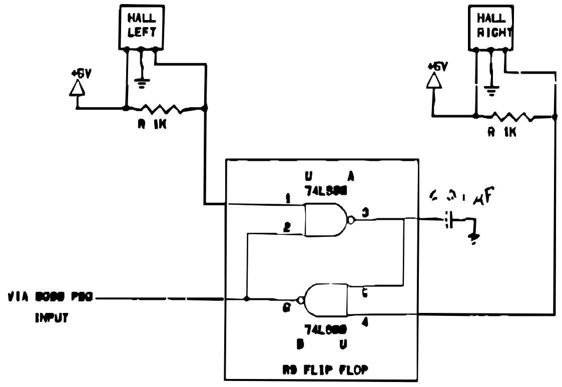
REV	[illegible]	ZONE	[illegible]	DESCRIPTION	DATE	APPROVED



CONTRACT -		FIRM NAME			
DR	/ /	TITLE Track Kill Circuit			
CHK	/ /				
ENG	/ /				
APPD	/ /				
NEXT HIGHER ASSY. -		SIZE D	FSCM NO. -	DWG NO. -	
23:38:12	22-May-87	TKILL			SHEET 1 OF 1

DPSS.DRW

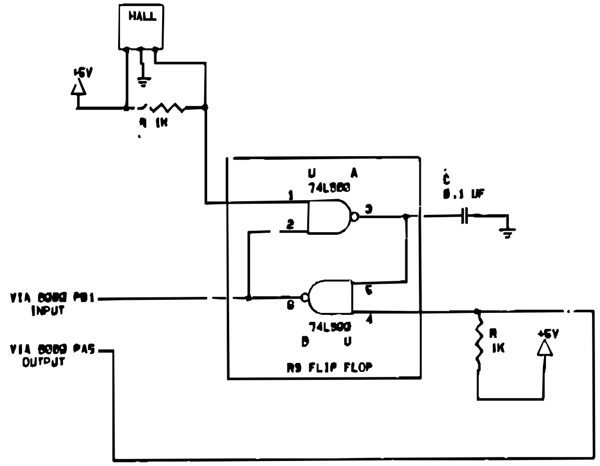
REVISIONS					
TYP	AUTHORITY	ZONE	LTR	DESCRIPTION	DATE	APPROVED



CONTRACT -		FIRM NAME		
DR	/ /	TITLE Cap Clear/Present Circuit		
CHK	/ /			
ENG	/ /			
APPD	/ /			
NEXT HIGHER ASSY. -		SIZE D	FSCM NO. -	DWG NO. -
23:25:00	22-May-87	COP88		SHEET XX OF

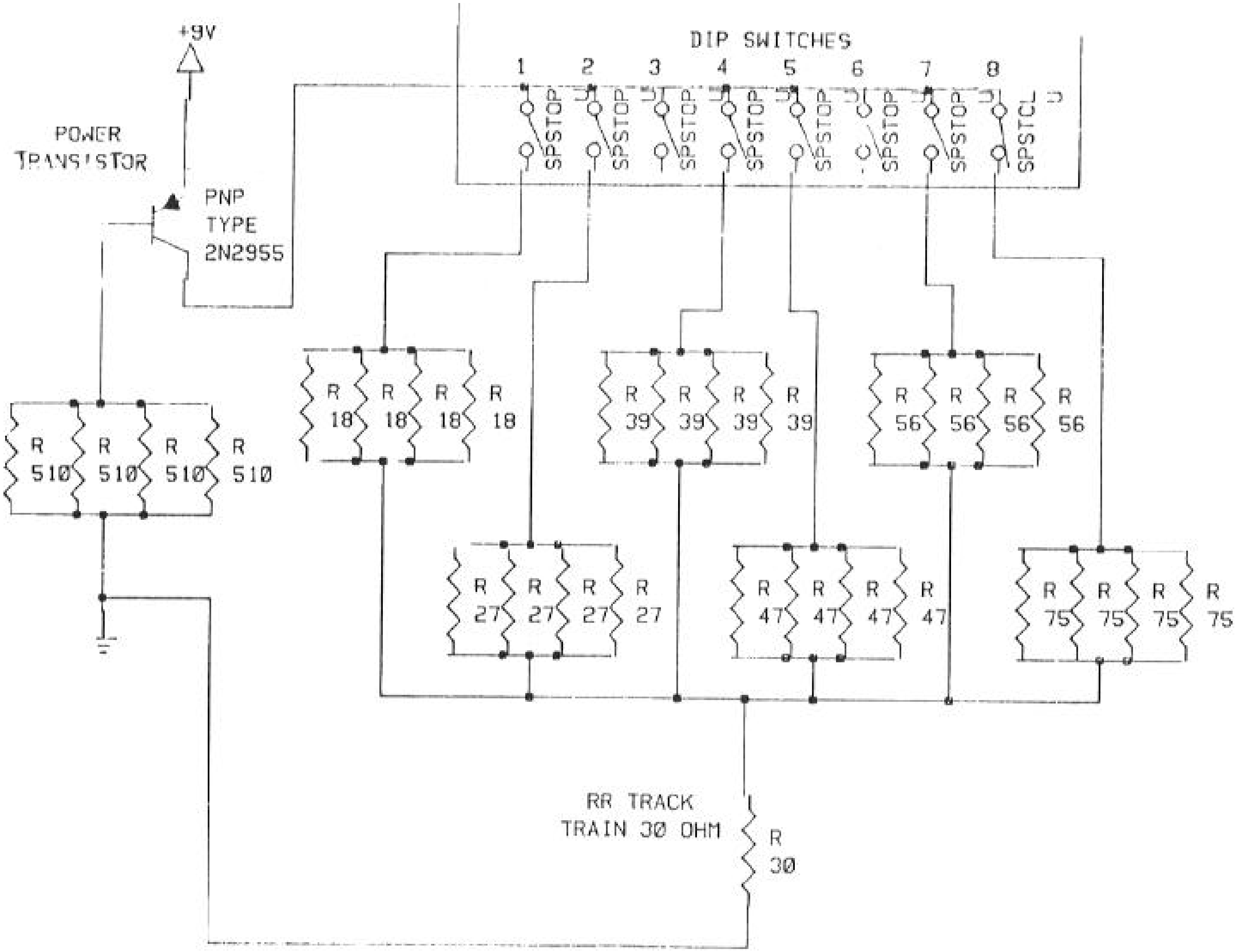
REVISIONS

EFF	AUTHORITY	ZONE	LTR	DESCRIPTION	DATE	APPROVED



CONTRACT -	FIRM NAME
DR / /	TITLE Truck Kill Sensor Circuit
CHK / /	
ENG / /	
APPD / /	SIZE D · FSCM NO. - · DWG NO. -
NEXT HIGHER ASSY. -	
23:42:31 2?-May-97	TRSB · SHEET XX OF 1

RN



DC Power Supply
Circuit