Homeward Bound

Jonathan Nichols & Michael Liebman

MAE 412

Professor M. Littman

May, 1995

This paper was written in accordance with University regulations.

Michael K. Liebman

Jon Nichols

326

Table of Contents

List of Figures and Tables

Homeward Bound

The Story Behind *Homeward Bound*

Homeward Bound is about a tired, little train that has traveled cross-country from California to Princeton, and needs to drop its three cars full of cargo at their appropriate home tracks. The conductor, Mr. Ned Lecky, knows that the train consists of an engine and three different cars, but he doesn't know the order that the cars are in. Luckily, California has recently decided to install barcodes on the sides of all their train cars. The conductor must drop each of the cars at the appropriate siding in Princeton, but he does not have time to stop the train and look at the order of the cars. Thus, he has radioed ahead (while traveling through the Midwest) and asked for a barcode reader, some optical sensors, and some decouplers to be installed in appropriate positions to ease his task. Conductor Lecky is now able to complete his job, and drop his cargo off in the correct places.

Introduction

Homeward Bound was designed to fulfill the aforementioned task, and to adhere to the constraints and goals imposed by MAE 412. Specifically, these goals are to sense, actuate and sequence using a synthesis of hardware and software. Some constraints include that the project must be assembled with N-gauge track, and that must use a computer driven by the 6502 microprocessor, interfaced with the Hornby Zero-One control system. The project also must contain the north and south tracks and a bi-directional track which connect to the side boards. Finally it must support the software developed by Ned Lecky, which implements the side board sensing and actuating capabilities, and obtains information from the Hornby controller.

The purpose of the project is for the engine to drop off three different cars equipped with barcodes, at their respective sidings, independent of the order of cars. The project begins with the full train, traveling clockwise around the outside main loop which consists of the two side board loops and the track on the main board connecting them. Once the train passes the barcode reader on the main board, the last car on the train is determined. The train then stops at the track kill section on the right side board, and then begins the sequence of dropping off the car that is last.

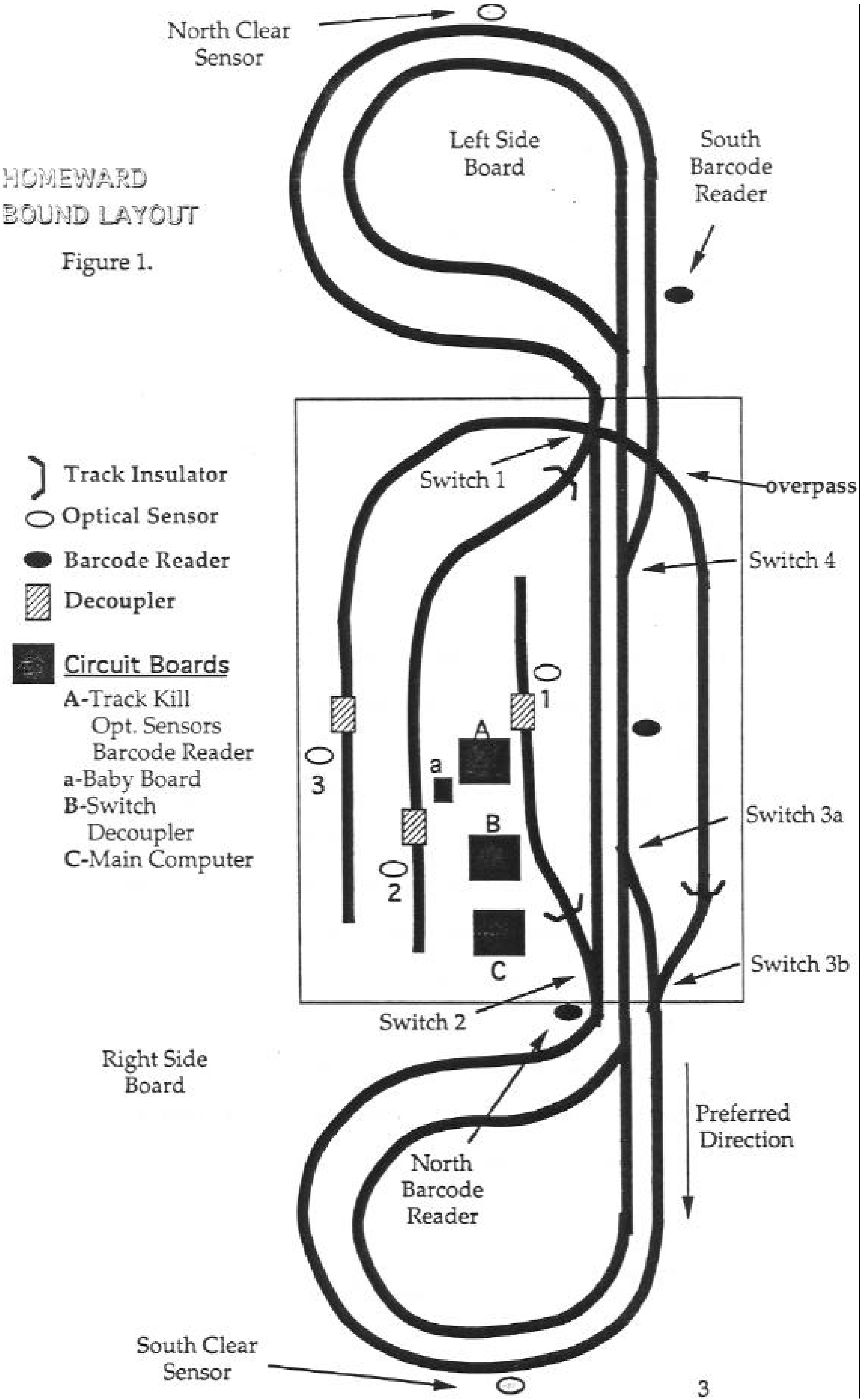
This sequence consists of stopping at one of the side board track kill sections, and then backing up into the correct siding. An optical sensor at the end of the siding senses the train, and then kills power to that section of track, stopping the train. Then, the train begins to move forward again until it is directly over the mechanical decoupler. The optical sensor senses the train again and activates the decoupler, dropping off the last car. The train then continues forward until it reaches its normal clockwise mode of travel, at which the barcode reader can once again determine the last car. This process continues until the last car is dropped off, and the project is completed.

Track Layout

Figure 1 shows a schematic of the main track board, including the track configuration, sensors, actuators, and circuit boards. The numbered sidings correspond the number of the car which is to be dropped off.

HOMEWARD BOUND LAYOUT

Figure 1.



The most notable feature that is not obvious from the figure is the overpass that is pointed out. This track section gradually increases in height until it is high enough so that a train can fit underneath, and then it levels out. Another noteworthy aspect of the layout is the fact that if car 1 is to be dropped off[1], the train must reverse itself, because the preferred direction of the track forces the train into its siding forwards, instead of backwards. Therefore we must send the train through a reversing loop in order to drop off car 1.

Operation

Homeward Bound requires extensive interaction between the user and the project. This is because a user prompt is required to reverse the direction of a train, or to change its speed. This will occur several times for each car's drop off sequence.

When the computer and Hornby are turned on, the computer displays **11** at the beginning of its initialization, and then **22** when initialization is complete. Then it displays **FA**, and waits for the user to setup the train. The train should be placed anywhere on the outer loops of the side boards facing in the proper direction with the barcodes facing in the correct direction so that the barcode readers can properly sense the barcodes. The user should then set the train to a moderate speed (halfway) using the Hornby by pressing **train-2-right** (if the engine used was train no. 2). Then the code **1-1-right** (entering 1 for computer #1) should be entered to tell the computer that we have set the train to **FA**st.

[1]By some silly oversight, switch 2 brings the train to siding 1 which drops off car #1, and switch 1 brings the train to siding 2 which drops of car #2. So, the car number corresponds to the siding number, not the switch number.

- ## Computer Codes

After staring the train as described above, the computer tells the user what it is doing, and what the user should do next through the following codes:

Computer Code	Explanation
11	Initialization
22	Initialization Complete
FA	Set train to Forward fAst. Press **1-1-Right.**
FO	Set train to Forward slOw. Press **1-2-Right.**
BA	Set train to BAckwards slow. Press **1-1-Left.**
2A	Waiting for the North track to be clear.
2B	Waiting for the North track to have a train
5A	Waiting for the South track to be clear.
5B	Waiting for the South track to have a train.
00	Barcode routine is running but no car has been detected.
01	Car 01 just passed by barcode reader.
02	Car 02 just passed by barcode reader.
03	Car 03 just passed by barcode reader.
ED	EnD. All cars have been dropped off.

Table 1.

- ## Notes

Before each barcode reading the train is stopped at the South relay, and FO is displayed requesting a Forward slOw speed. It is important that the train be going slowly in order for the barcode reader to correctly read the car numbers. This is because the cars sway a lot more than engines do, which causes the optical sensors' performance to deteriorate.

When backing into a siding, moderate speed should be used so that the momentum of the train carries it past the entire barcode of the last car.

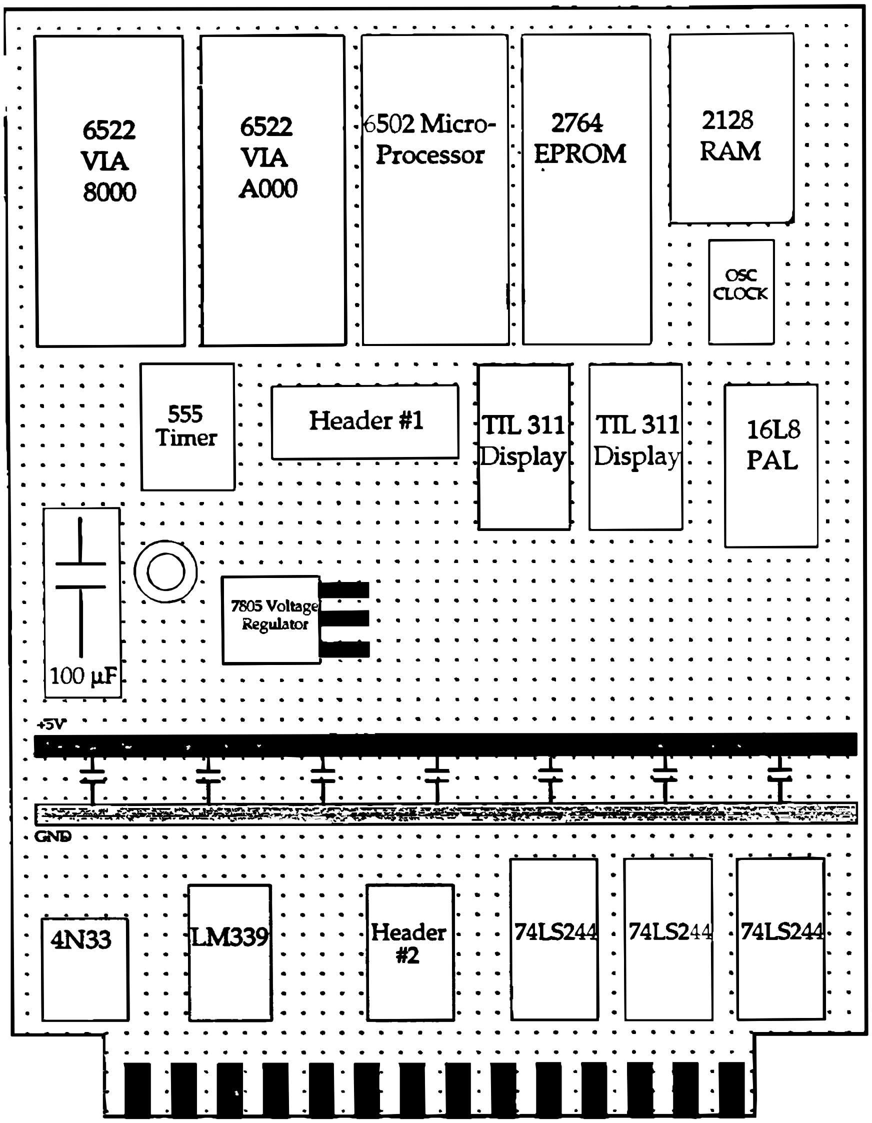
However, if too much speed is used, it is possible that the train will stop with the second-to-last car's barcode behind the sensor. This would result in the decoupling of the last two cars.

When pulling forward to decouple a car, the speed should be very, very slow. After the decoupler is finished, the computer kills the track and displays **FA,** telling the user to set the normal **FA**st speed.

When all the cars have been dropped off the barcode reader will not read any cars, and when the engine hits the south clear sensor, **ED** will be displayed meaning that the sequencing has EnDed.

Microcomputer

The computer used by *Homeward Bound* conforms to the standard MAE 412 design, with several individual facets. A diagram of the final board is shown in figure 2, and a schematic is shown in appendix A1.



To Edge
Connector

Figure 2.

- Input and Output

The design of the I/O system was unique to the project, and is drawn schematically on appendix A2. Table 2 gives a listing of the connections between the ports and edge connector pins.

Edge Pin	Computer Connection	Color	Description
A	4N33	Green	Track Ground
B	4N33	Yellow	Track Power
C-L	N/C		
M	PA0, VIA 8000	Green	State of Track Kill
N	PA1, VIA 8000	White	1st bit of barcode
P	PA2, VIA 8000	Blue	2nd bit of barcode
R	PA3, VIA 8000	Yellow	Clear pulse for shift reg.
S	PA4, VIA 8000	Red	Optical Sensor, Siding 1
T	PA5, VIA 8000	Grey	Optical Sensor, Siding 2
U	PA6, VIA 8000	White	Optical Sensor, Siding 3
V	Power, +5V	Red	Power & Gnd for
W	Ground	Black	Sensor Daughter Board
X-Z	N/C		
1-3	N/C		
4	Ground	Black	SSB Pin 15
5	PB0, VIA A000	White	SSB Pin 16
6	PB1, VIA A000	Yellow	SSB Pin 17
7	PB2, VIA A000	Blue	SSB Pin 1
8	PB3, VIA A000	Orange	SSB Pin 2
9	PB6, VIA A000	Green	SSB Pin 3
10	N/C		
11	PB4, VIA A000	Red	SSB Pin 4
12	PA0, VIA A000	Grey	Switch 1
13	PA1, VIA A000	Brown	Switch 2
14	PA2, VIA A000	Blue	Switch 3
15	PA3, VIA A000	Orange	Switch 4
16	PA4, VIA A000	Green	Relay for switches 1 & 2
17	PA5, VIA A000	White	Relay for switches 3 & 4
18	PA6, VIA A000	Yellow	Mechanical decoupler
19	Power, +5V	Red	Power & Gnd for
20	Ground	Black	Switch Daughter Board
21-22	N/C		

Table 2.

- **Address Decode Logic**

In order to reduce the amount of hardware on the main computer, the address decode logic and the buffering and inverting were encoded on a 16L8

Programmable Array Logic (PAL) chip. Appendix A3 shows the functions of the PAL, and appendix B shows the PAL logic diagram. The PAL takes as its input the address lines, A13, A14 and A15, in addition to the Φ_2 clock line and the signal from the power-up reset circuit. Its outputs are the chip enable lines for the RAM, EPROM, VIA A000, VIA 8000 and the display chips, as well as a buffered Φ_2 line and an inverted reset line.

HARDWARE COMPONENTS

- **Optical Sensors**

Standard OPB742 infrared optical sensors were used to detect reflective tape on the cars. Figure 3 shows where they should be placed with respect to their distance from the car on the track, and from the board.

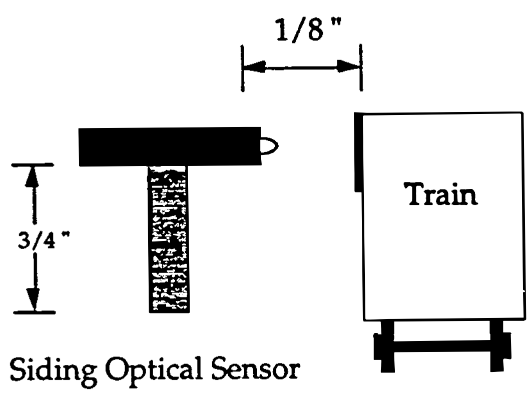


Figure 3.

There is one sensor for each siding, and it is at the same height as the clock (lower) level of the barcode on the cars. The optical sensor circuit is shown appendix A4. Originally, each optical sensor circuit was meant to have a Schmidt Trigger, but after the construction of the circuits, it was discovered that a Schmidt trigger only works on the noninverting lead of an analog comparator. Therefore, the circuit diagrams shown in appendix A4 show the actual constructed circuit, but the attempted Schmidt trigger has no effect. Since the Schmidt trigger is not crucial for the siding optical sensors, this posed no problem.

A steady 30 mA current runs through the diodes causing them to emit infra-red light. The photo-transistor in the optical detector is off if no light is reflected onto it, and the positive lead of the comparator is low. The threshold voltage is set for 2.5 volts. If reflected light hits the photo-transistor, it connects the noninverting input of the comparator to +5V, and the output of the comparator is also +5V. Thus, the optical sensors are normally low, and go high when reflective tape is 1/8" in front of them. The sensitivity to room lights was a small fraction of a volt, so the project works just as well in the dark.

- **Barcodes**

A barcode is constructed by spray painting a stiff piece of cardboard black, and placing a piece of reflective tape on it. Then appropriate sections of the tape must be removed to uncover the black sections underneath. When placing the barcodes on the cars, care must be taken to keep the barcode straight, and to make sure that it is at the right height and distance from the optical sensors.

The barcode scheme used for our cars is similar to that used by the Lecky software for engines. The barcode is split into two levels. The lower level is a clock pulse, and the upper level is the data. The barcodes were placed on the same side of the train as the normal barcodes used on the engines. However, the car barcodes were raised so that they would not trigger the clock on the side-board barcode readers. Thus, they do not interfere with the normal Lecky collision software.

Figure 4 shows what a typical car looks like, and exactly where the bar code should be placed. It also shows the appropriate bar codes for each car, 1-3.

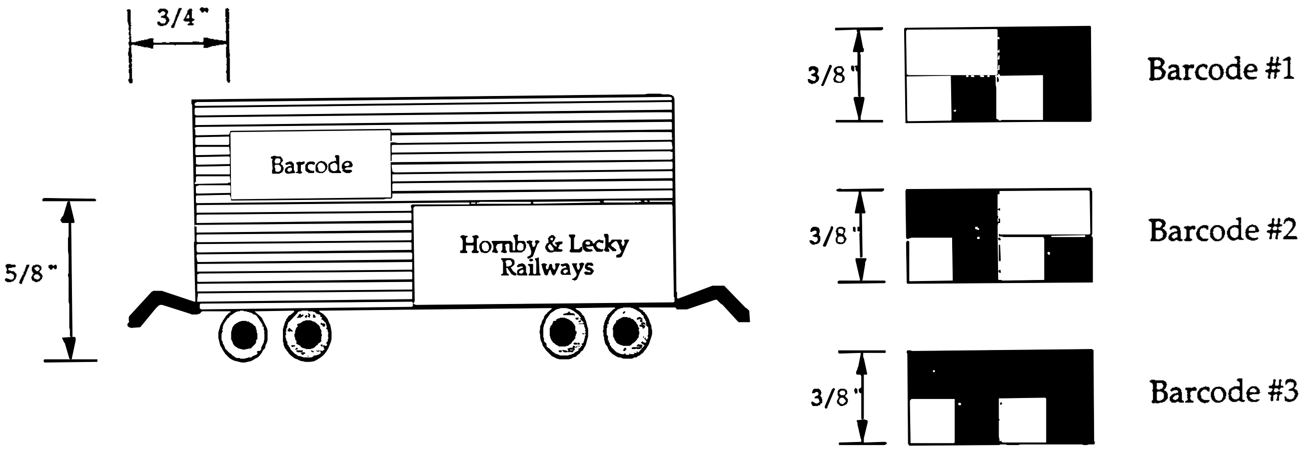


Figure 4.

- **Barcode Reader**

The barcode reader consists of two sensors on top of each other. Figure 5 shows how it should be placed relative to the car on the track.

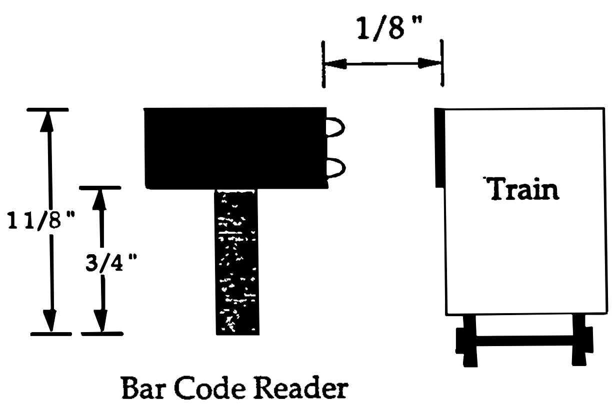


Figure 5.

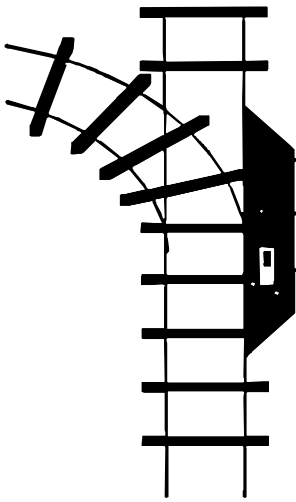
Its schematic is also shown in appendix A5. The data (top) optical sensor is wired just like the other optical sensors described above, and thus is high when reflective tape is in front of it. The clock (bottom) optical sensor is slightly different. It is important that this sensor has a Schmidt trigger on it because it is connected to the clock on a shift register. Without a Schmidt trigger, noisy behavior in the clock transition causes multiple transitions, so it turns out that all of the bits in the shift register are set to the same value, which 1 or 0 depending whether the data above the clock transition is either dark or light.

For the clock sensor, a working Schmidt trigger circuit is included on the noninverting input to the comparator. The two threshold voltages are 2.27V and 2.72V. Since the Schmidt trigger was changed from the inverting to the non-inverting input at the last minute, a baby board had to be added. Another consequence is that the clock is normally high, and goes low when reflective tape is in front of it which is opposite the other sensors.

A shift register was used so that the computer would not miss part of the barcode when it jumped to the interrupt service routine. The clock line of the shift register is connected to the clock optical sensor, and is activated on a low-to-high transition. This is equivalent to a REFLECTIVE-to-BLACK transition in the barcode clock line. Both data lines of the shift register are connected to the data optical sensor. The 1st and 2nd bits of the shift register, and the clear line are fed to the computer through the I/O ports so that software can manipulate them. Since there are only 3 cars, only the first 2 bits of the shift register are important.

- Switches

Mechanical switches were used to route the train off of the main line and to the reversing loop. Figure 6 shows the left convention of the switch, and the lines that connect to it.



Left Turn Switch

Figure 6.

It apparent from figure 1 that switches 1, 3b and 4 are right switches and switches 2, and 3a are left switches. The switches' normal mode corresponds to the combination of positions which sends the train around the main loop, and does not let it go to any sidings or reversing loops. For this to occur, switches 1, 2 and 3b are forward, and switches 3a and 4 are turned. Because the project never requires that switches 3a and 3b be both forward or both turned, they are connected to the same circuit, but in the opposite sense. Switches 1, 2 and 3b send the train to the appropriate sidings, and switches 3a, and 4 take the train on and off of the center, bi-directional track and into the reversing loops (only the reversing loop on the left side board is used in the project).

A schematic for each switching circuit is shown in appendices A6 and A7. The switches' direction is controlled by a double pole, double throw relay, which itself is able to handle two switches at a time. The relay is activated by sending a TTL pulse from the I/O chips on the computer into the base of a PNP transistor, which sends another signal to the base of a NPN transistor, which sends a current of approximately 80 mA through the relay loop. When the relay gets thrown in a certain direction, only one of the coils of the switch is activated (only the red, or the green is connected to the black), so only one switch circuit is needed for each switch. Therefore, only 4 switch circuits, and 2 relay circuits are required.

The switch coils are powered by sending a TTL signal into an opto-isolator, which then sends a +5 V pulse from the track to the gate of a TIC106D SCR, activating its forward bias. In order that power from the track not be drawn during a data frame, and to insure that a positive voltage pulse is sent to the SCR for both Hornby polarities, and to insure that the switch is not powered for more than a few microseconds, the TTL signal to the opto-isolator is sent only for FRANUM=1-3. After FRANUM=3, the gate voltage will no longer be +5 V, so when the Hornby pulse goes negative, the SCR shuts itself off, and the coil is no longer powered.

- **Mechanical Decoupler**

These devices are used to disconnect the last car from the rest of the train. They are placed strategically close to the optical sensors at each siding. Figure 7 shows how they are oriented.

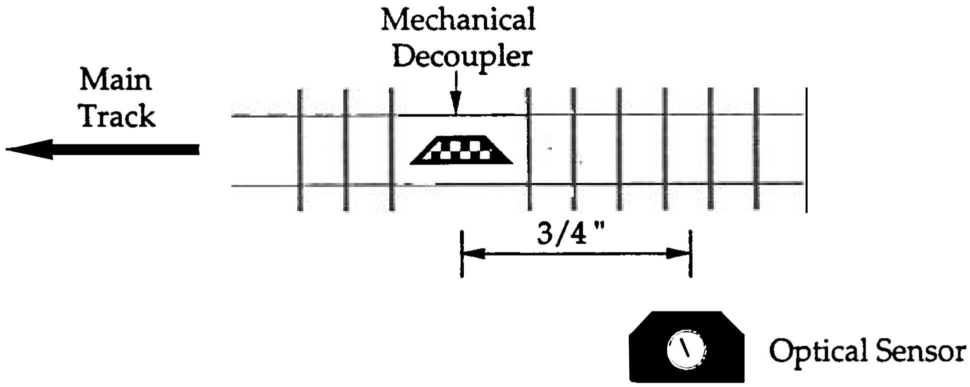


Figure 7.

Note that the distance from the mechanical decoupler to the optical sensor is identical to the distance from the train coupler to the bar code. This was done so that when the optical sensor senses the train on its way out, the car's coupler is directly over the mechanical decoupler. This will detach the car without much trouble.

The circuitry is shown in appendix A7, and is essentially identical to the switch circuits excluding the relay. Since the nature of the project allows all three decouplers to be thrown at the same time, there is only one decoupler circuit, and all three decouplers are connected in parallel. Once again, track power must not be drawn from the track during a data frame, so the decoupler is only activated for FRANUM=1-4. However, since the decoupler must be on much longer than the switch, this cycle of powering during frames 1-4 and cutting power during the data frame is repeated thousands of times, keeping the decouplers on for a few seconds.

- ## Track Kill Circuit

Once again, in order to minimize hardware, all three isolated track sections are connected together. A relay circuit is used to cut the line between the powered section of track and the isolated track when the TTL signal from the I/O ports goes low. It is only necessary to cut one of the track lines. Cutting either line will disturb the signal enough to stop the train. A schematic of this circuit is shown in appendix A6.

Software

Homeward Bound's software program, DROP5.ASM interacts extensively with the program, LECKY3_0.ASM. Other than the aforementioned utilization of the FRANUM mailbox, DROP5.ASM uses the NOVRD, SOVRD, NTRAIN and STRAIN (north and south override, and north and south train) mailboxes as well. NOVRD and SOVRD[2] initially have the value 0. However, if #$FF is stored in NOVRD or SOVRD, the corresponding north or south track kill relay will activate. This was useful to the project because it allowed for stopping and backing up into the sidings for car drop off.

NTRAIN and STRAIN were useful because they determined where the train was on the outside boards. For example, the north barcode reader will read the barcode from the engine and store the value into the NTRAIN mailbox. The NTRAIN mailbox will have this value until it the train passes by the north clear sensor, after which the NTRAIN value will be set to zero. Therefore, if we want to stop the train at the south track kill section, we can wait until the NTRAIN goes to zero, and then override the south relay by storing #$FF into SOVRD. Repowering the track again is as easy as storing #$00 back into the mailbox.

---

[2]The override ability is an extra feature added into LECKY3_0.ASM

Setting output voltages to VIA ports is done carefully using masks listed at the beginning of the program. The variable, PAA, is used as a mirror of the A Port of the VIA at A000.

The main sequence of events is as follows. The last car's barcode is read. Then that car is dropped off. This process is repeated until there are no more cars. The flow-chart is shown in appendix C, as is the complete software listing with comments and a short description of all of the relevant subroutines.

The BARCODE subroutine reads the last car's barcode and saves the value into the LCAR mailbox. It ends when the train reaches the south clear sensor. The LCAR mailbox determines which subroutine to follow, DO1, DO2 or DO3.

DO1 begins by overriding the north relay[3], and prompting the user to send it backwards. Switch 4 gets thrown forward which bring the train to the reversing loop, and switch 2 gets thrown to turn, which will bring car 1 to its siding. The train then backs up through the reversing loop on the left side board and into the siding until the optical sensor at siding 1 senses it. It then goes through the previously mentioned sequence of decoupling the cars. The next task is to throw the switches back into the normal mode, and to send the train back into its normal clockwise rotation. After the car is decoupled, the train moves forward again, back through the reversing loop and then stops at the north relay. The computer prompts the user to send the train forward again, after which it switches switch 2 back to its normal mode. When the engine reaches the north clear sensor, switch 4 is thrown to its normal mode as well, and the program continues.

---

[3]A noteworthy point to make here is that everytime a train stops at a side board track kill section and subsequently backs up into the project again, the other side board's relay is overridden as well, to prevent other trains from coming onto the track.

DO2 begins by stopping at the north track kill section, and prompting the user to send it faster, since the barcode routine requires a low speed. It then waits until a north train is entered (before this point, NTRAIN was zero, so we must wait until NTRAIN is non-zero and then zero again before we switch 1 to turn), and then until it gets cleared. It stops the train at the south track kill section, and then prompts the user to send it backwards. At this point, it also switches 1 to turn, so the train can back into its siding. It then performs the decoupling sequence, and then moves forward again. When a train is registered at the south barcode reader, switch 1 is returned to normal mode and the program goes on.

DO3 is similar to DO2. It stops at the north track kill section after the BARCODE routine. It then prompts the user to back it up. When the correct value is entered, it switches switch 3b to turn and backs up. It performs the decoupling sequence, and then proceeds forward again. When a train is registered on the north track, switch 3 gets thrown to its normal mode, and the BARCODE routine is run again.

Conclusions

We achieved our goals of sensing, actuating and sequencing, and also conformed to the constraints that were placed upon us. And in the process, we essentially came up with something new, and were able to solve every problem that faced us in a feasible fashion. Placing barcodes on the cars seemed difficult at first, but then we realized that we only had to look at one last train at a time, rather than storing a sequence of trains. This realization definitely prevented much pain and suffering. Furthermore, we only had to store one bit at a time into the shift register, and extract the last two bits when the train is finished, rather than storing two bits at a time for each car, which would require knowing when one car stopped and the next one started. Thus, the shift register acts essentially like a stack, of which we are only interested in the top two elements.

We were able to make the mechanical decouplers work quite well, quite to our surprise. Most likely, this is due to the fact that we were running the trains at minimum speed during decoupling. This is a robustness problem that is out of our hands, unless we had implemented the electro-magnetic decouplers which would work better at high speeds. Yet, in the real world, you don't see too many trains being detatched at high speed.

The biggest problem we encountered was the robustness of the barcode readers. They only work well when the train is going really slowly. This is most likely due to the fact that the cars we were using swayed back and forth more markedly as the speed increased. Further work on this project will involve using cars with more rigidity.

We learned a great deal during this design process, and we are very proud of what we have created. Not too many college kids get to design and build a fully operational N-Trak train system and get credit for it.

APPENDICES

Appendix A: Schematics

- A1-Main Computer Board
- A2-I/O Schematic
- A3-PAL Function Diagrams
- A4-Optical Sensor Diagrams
- A5-Barcode Reader Circuit Diagram
- A6-Switches 1 & 2 and Track Kill Circuits

A7-Switches 3 & 4 and Mechanical Decoupler Circuit

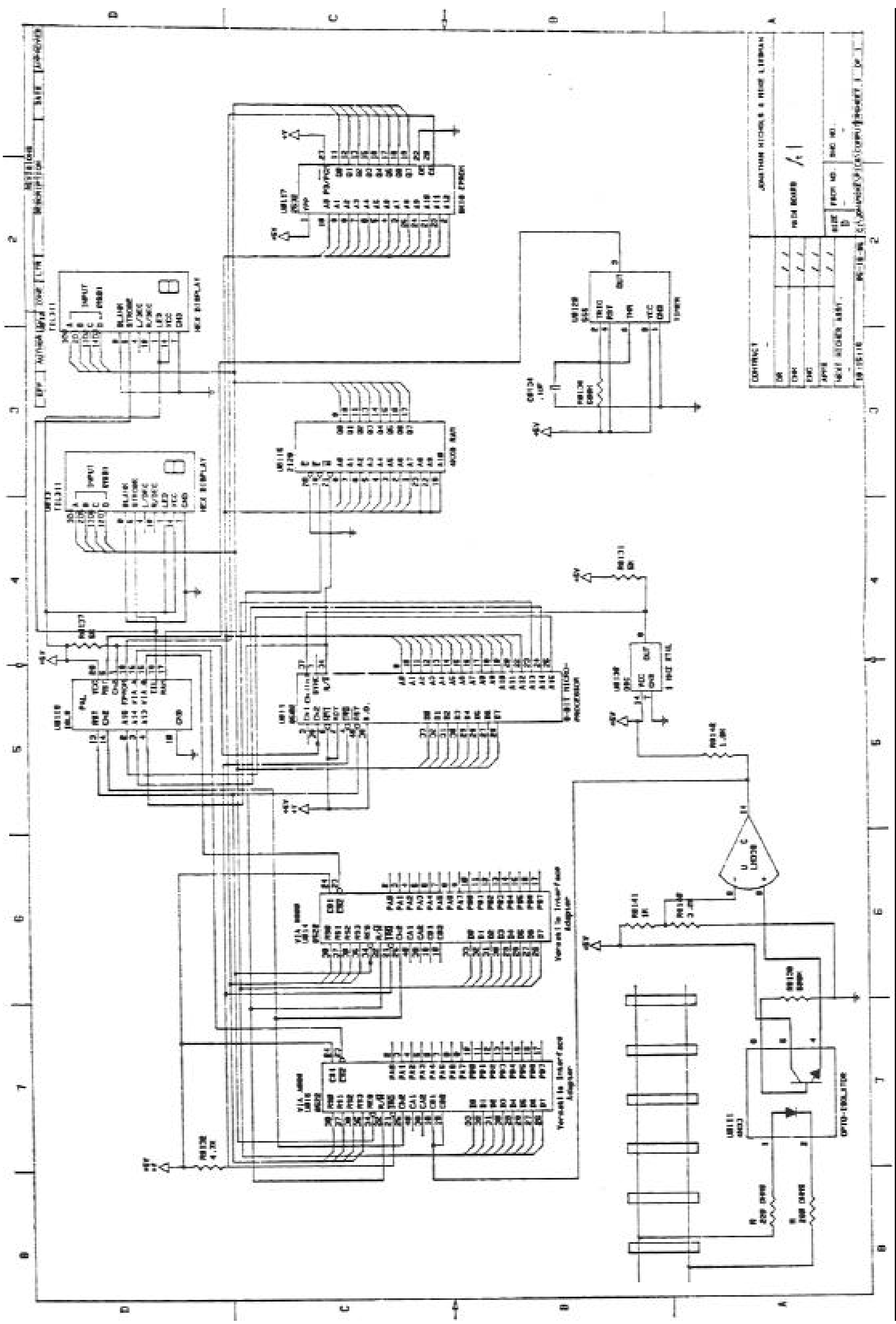
Appendix B: PAL Logic Diagram

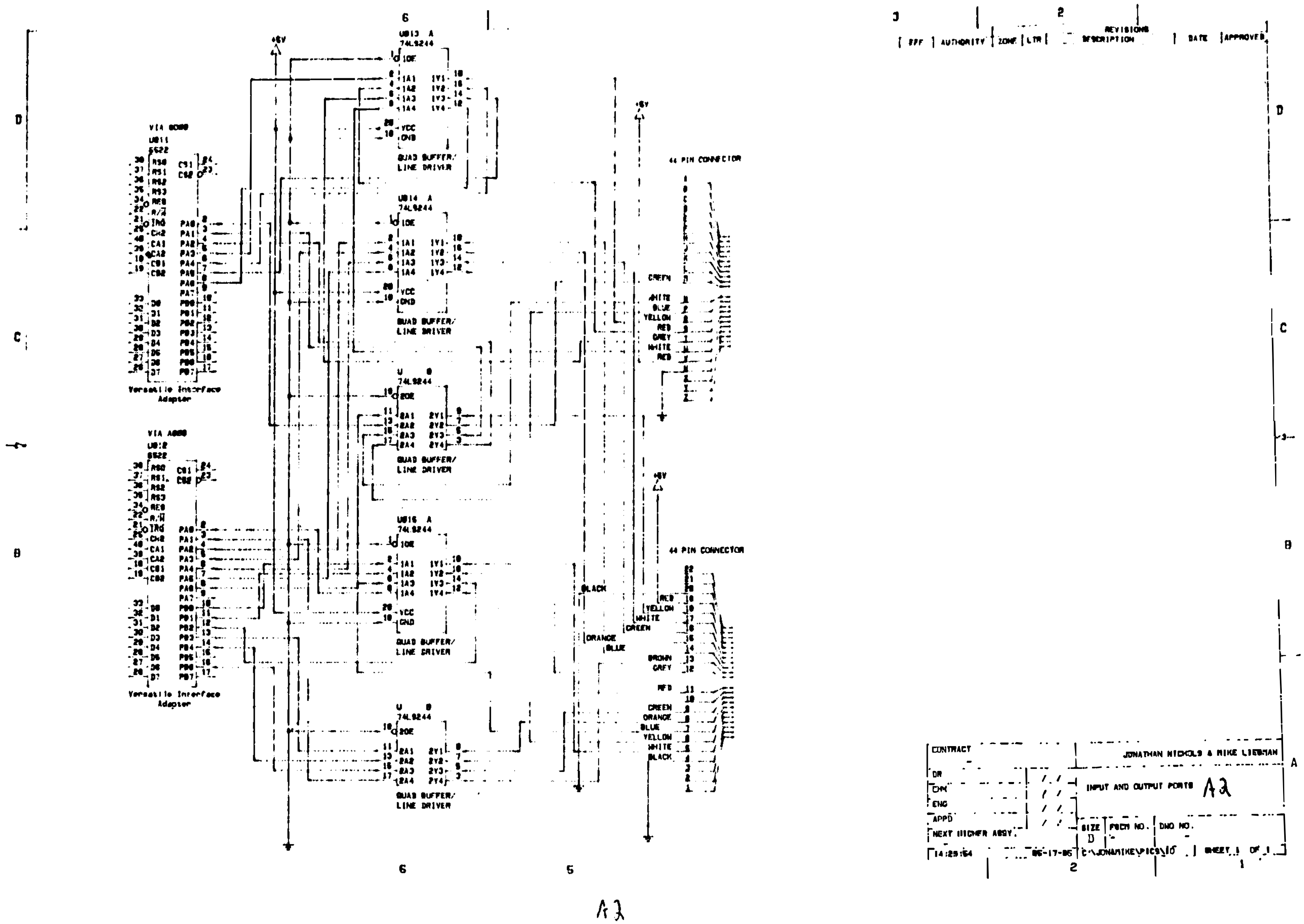
Appendix C: Software

C1-Main Program Flow Chart

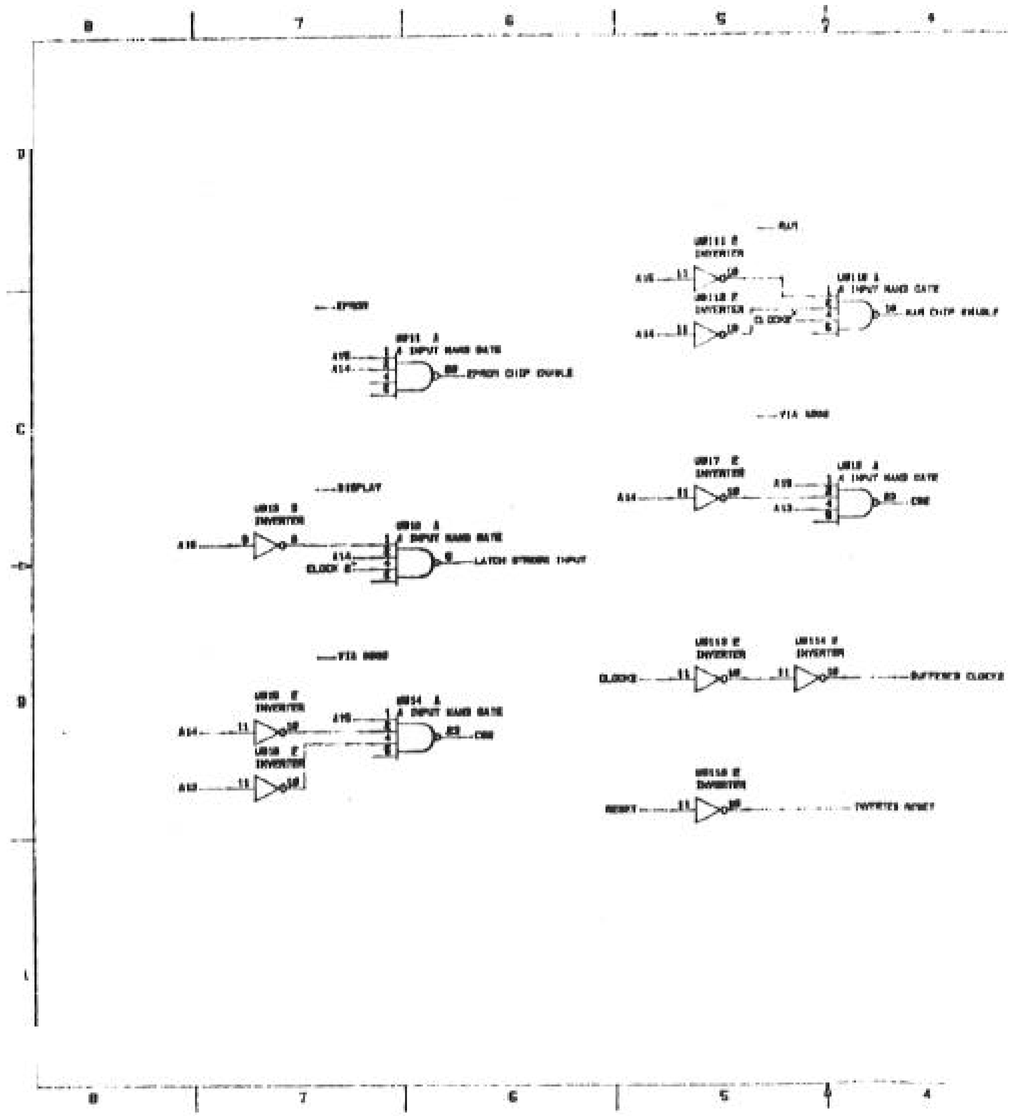
C2-Subroutine Description

C3-DROP5.ASM


HEX DISPLAY
HEX DISPLAY
8-BIT MICRO-PROCESSOR
Versatile Interface Adapter
Versatile Interface Adapter
OPTO-ISOLATOR
TIMER
JONATHAN NICHOLS & MIKE LIEBMAN
MAIN BOARD



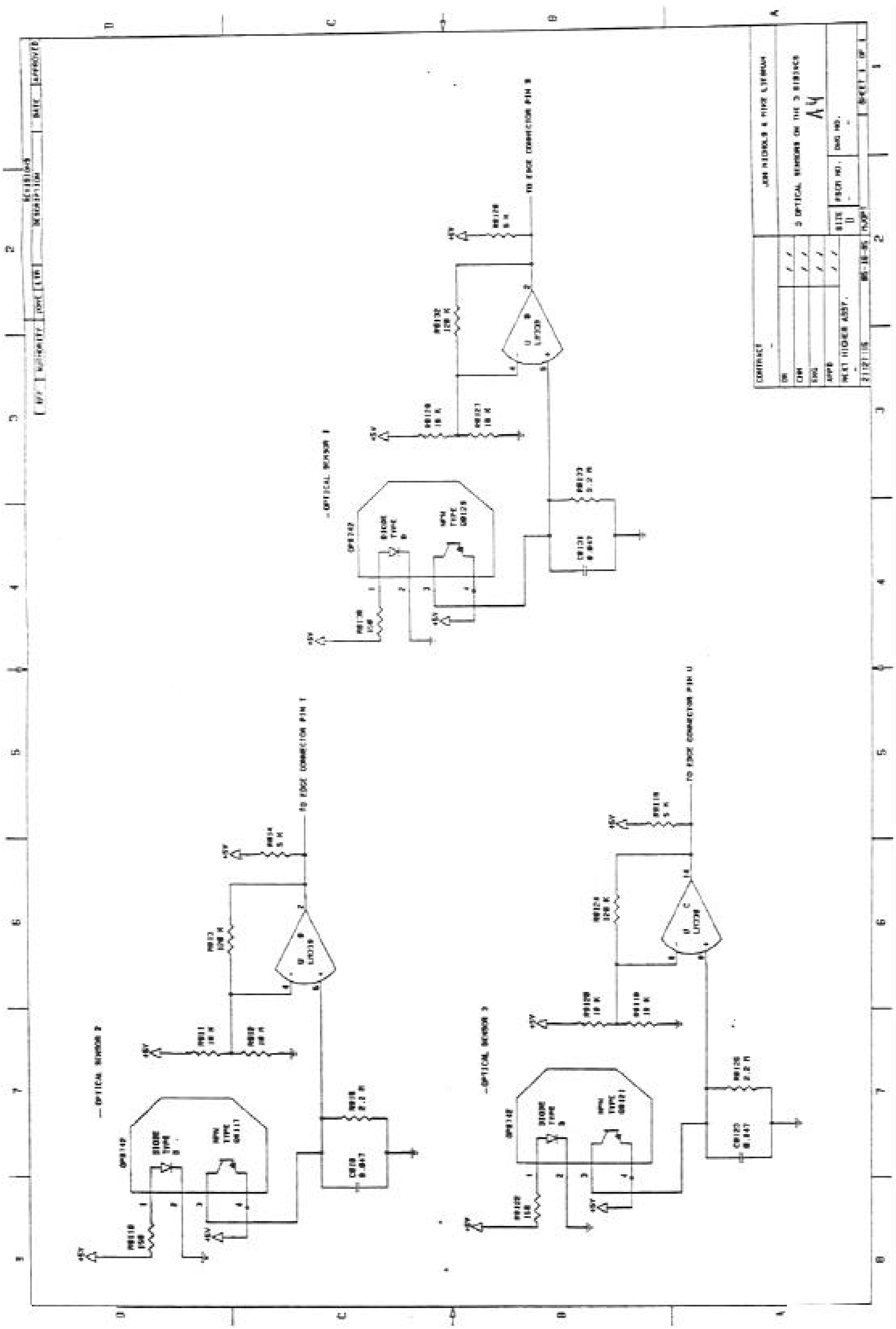
REVISIONS
EFF
AUTHORITY
ZONE
LTR
DESCRIPTION
DATE
APPROVED
VIA 8000
U811
6522
Versatile Interface Adapter
VIA A000
U812
6522
Versatile Interface Adapter
U813 A
74LS244
U814 A
74LS244
U816 A
74LS244
QUAD BUFFER/
LINE DRIVER
44 PIN CONNECTOR
+5V
GREEN
WHITE
BLUE
YELLOW
RED
GREY
BLACK
ORANGE
BROWN
CONTRACT
DR
CHK
ENG
APPD
NEXT HIGHER ASSY.
JONATHAN NICHOLS & MIKE LIEBMAN
INPUT AND OUTPUT PORTS
A2
SIZE
D
FSCM NO.
DWG NO.
SHEET 1 OF 1
14:29:54
06-17-86
A3



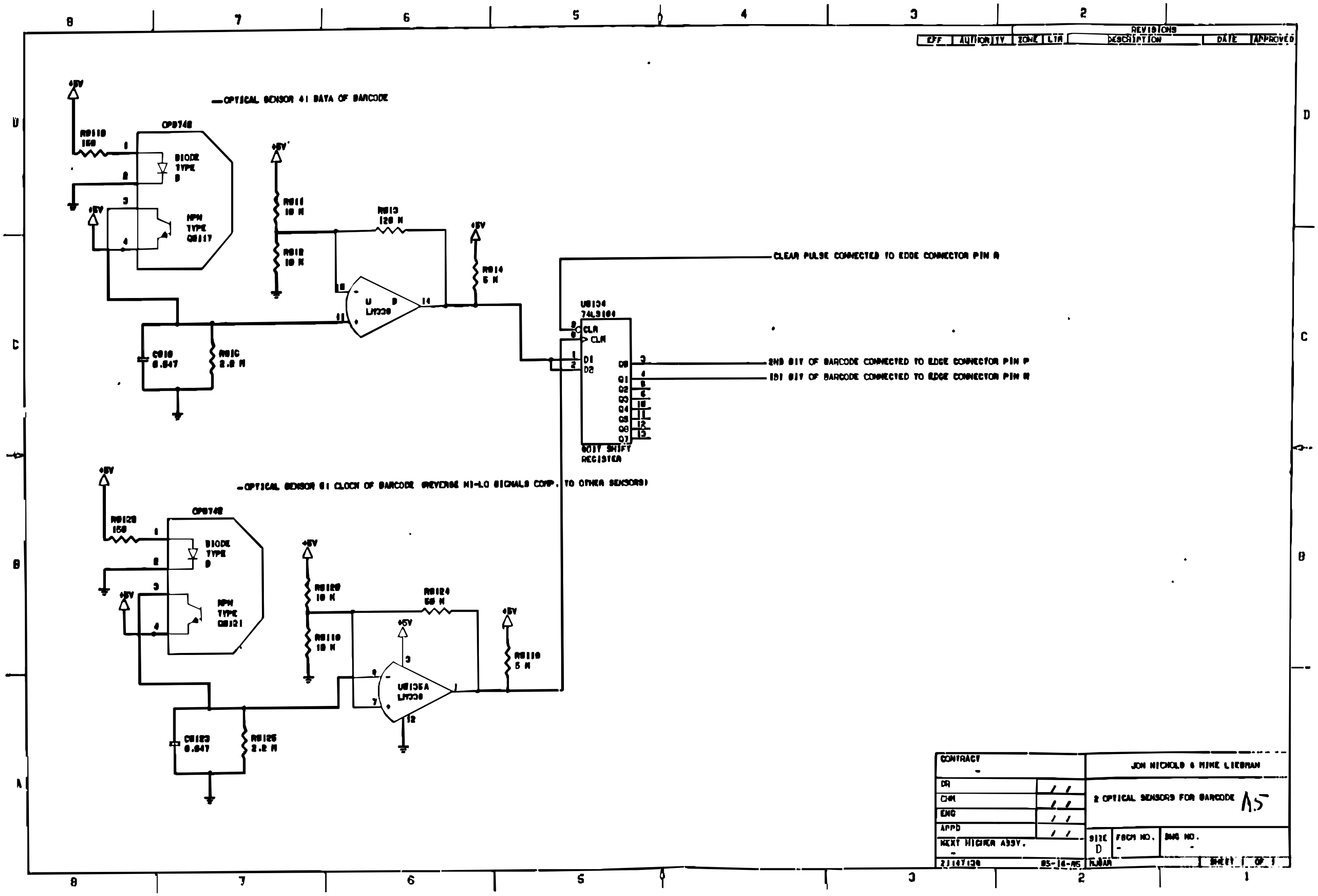
CONTRACT	JONATHAN NICHOLS & MIKE LIEBMAN
DR / CHK / ENG / APPD	SCHEMATICS FOR PAL FUNCTIONS A3
NEXT HIGHER ASSY.	SIZE D · FSCM NO. · DWG NO.

A3

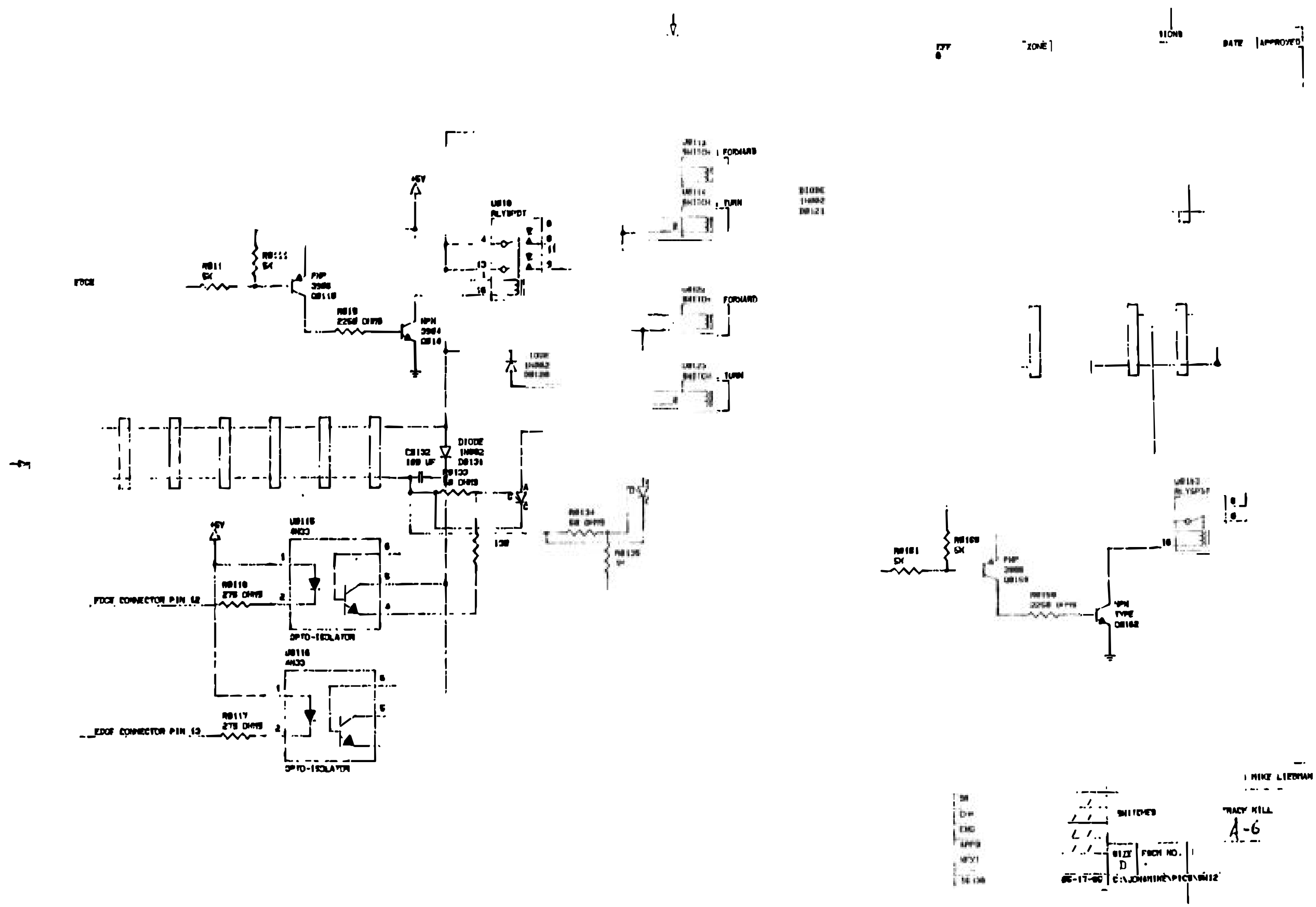
352


OPTICAL SENSOR 1
OPTICAL SENSOR 2
OPTICAL SENSOR 3
CONTRACT
NEXT HIGHER ASSY.
A4

A4

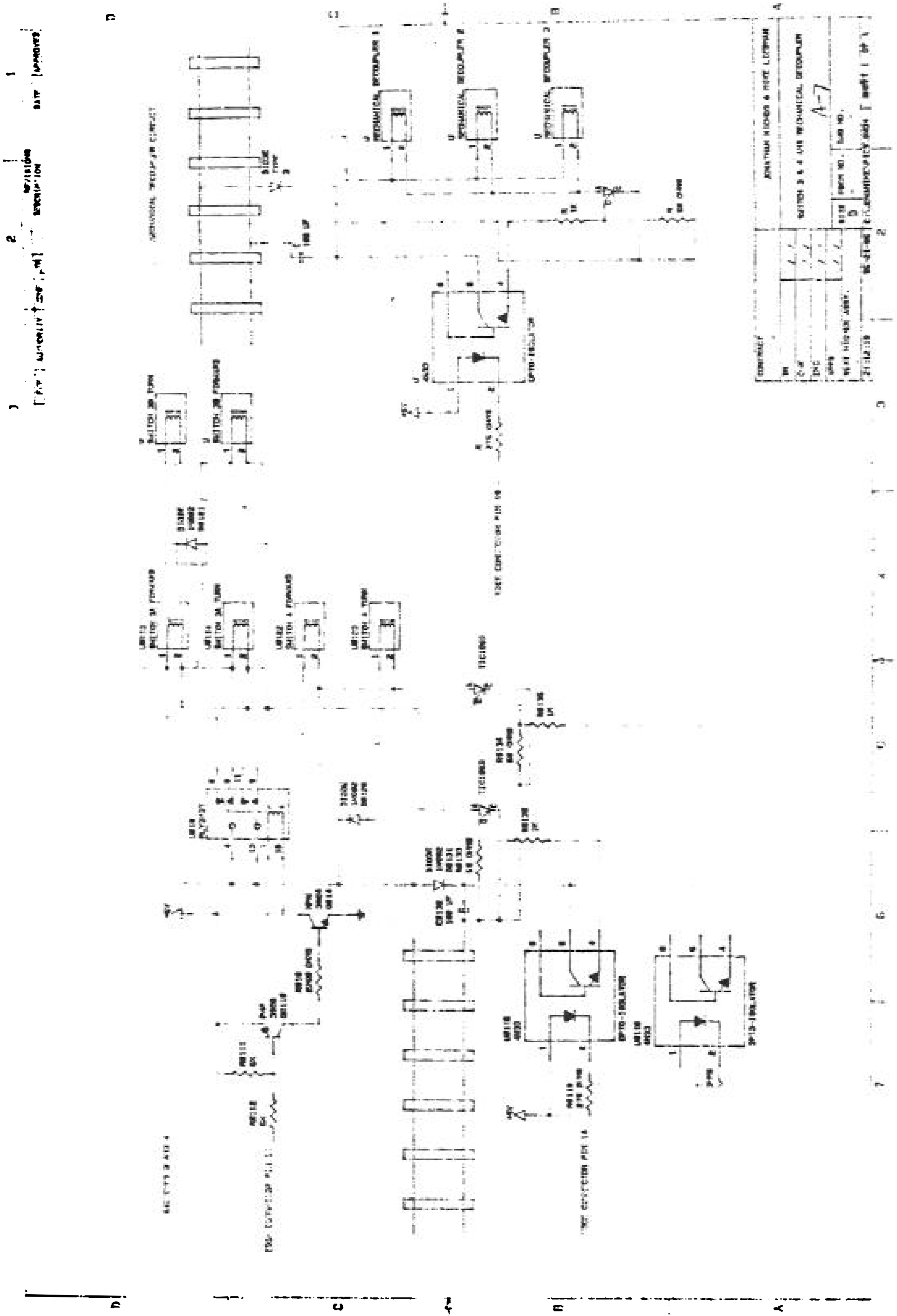

REVISIONS
EFF | AUTHORITY | ZONE | LTR | DESCRIPTION | DATE | APPROVED
OPTICAL SENSOR 41 DATA OF BARCODE
OP8742
DIODE TYPE D
NPN TYPE QB117
R8118 150
R811 10 K
R812 10 K
R813 120 K
R814 5 K
C810 0.047
R81C 2.2 M
U B LM339
+5V
CLEAR PULSE CONNECTED TO EDGE CONNECTOR PIN R
U8134 74LS164
CLR
CLK
D1
D2
Q0 Q1 Q2 Q3 Q4 Q5 Q6 Q7
8BIT SHIFT REGISTER
2ND BIT OF BARCODE CONNECTED TO EDGE CONNECTOR PIN P
1ST BIT OF BARCODE CONNECTED TO EDGE CONNECTOR PIN N
OPTICAL SENSOR 61 CLOCK OF BARCODE (REVERSE HI-LO SIGNALS COMP. TO OTHER SENSORS)
OP8742
R8128 150
DIODE TYPE D
NPN TYPE QB121
R8120 10 K
R8110 10 K
R8124 68 K
R8118 5 K
U8135A LM339
C8123 0.047
R8125 2.2 M
CONTRACT
DR
CHK
ENG
APPD
NEXT HIGHER ASSY.
2J147139
05-16-85
JON NICHOLS & MIKE LIEBMAN
2 OPTICAL SENSORS FOR BARCODE
A5
SIZE D
FSCM NO.
DWG NO.
SHEET 1 OF 1

A-5


EDGE
RB11 5K
RB111 5K
PNP 3906 QB110
RB19 2250 OHMS
NPN 3904 QB14
+5V
UB10 RLYSPDT
DIODE 1N902 DB121
UB113 SWITCH FORWARD
UB114 SWITCH TURN
UB124 SWITCH FORWARD
UB125 SWITCH TURN
DIODE 1N902 DB131
CB132 100 UF
RB133 50 OHMS
RB134 50 OHMS
RB135 5K
UB115 4N33
RB116 270 OHMS
EDGE CONNECTOR PIN 12
OPTO-ISOLATOR
UB116 4N33
RB117 270 OHMS
EDGE CONNECTOR PIN 13
UB162 RLYSPST
RB161 5K
RB160 5K
PNP 3906 UB159
RB158 2250 OHMS
NPN TYPE QB162
ZONE
SIGNS
DATE
APPROVED
MIKE LIEBMAN
TRACY KILL
A-6
SWITCHES
SIZE D
FSCM NO.
C:\JOHNMIKE\PICS\B12

355

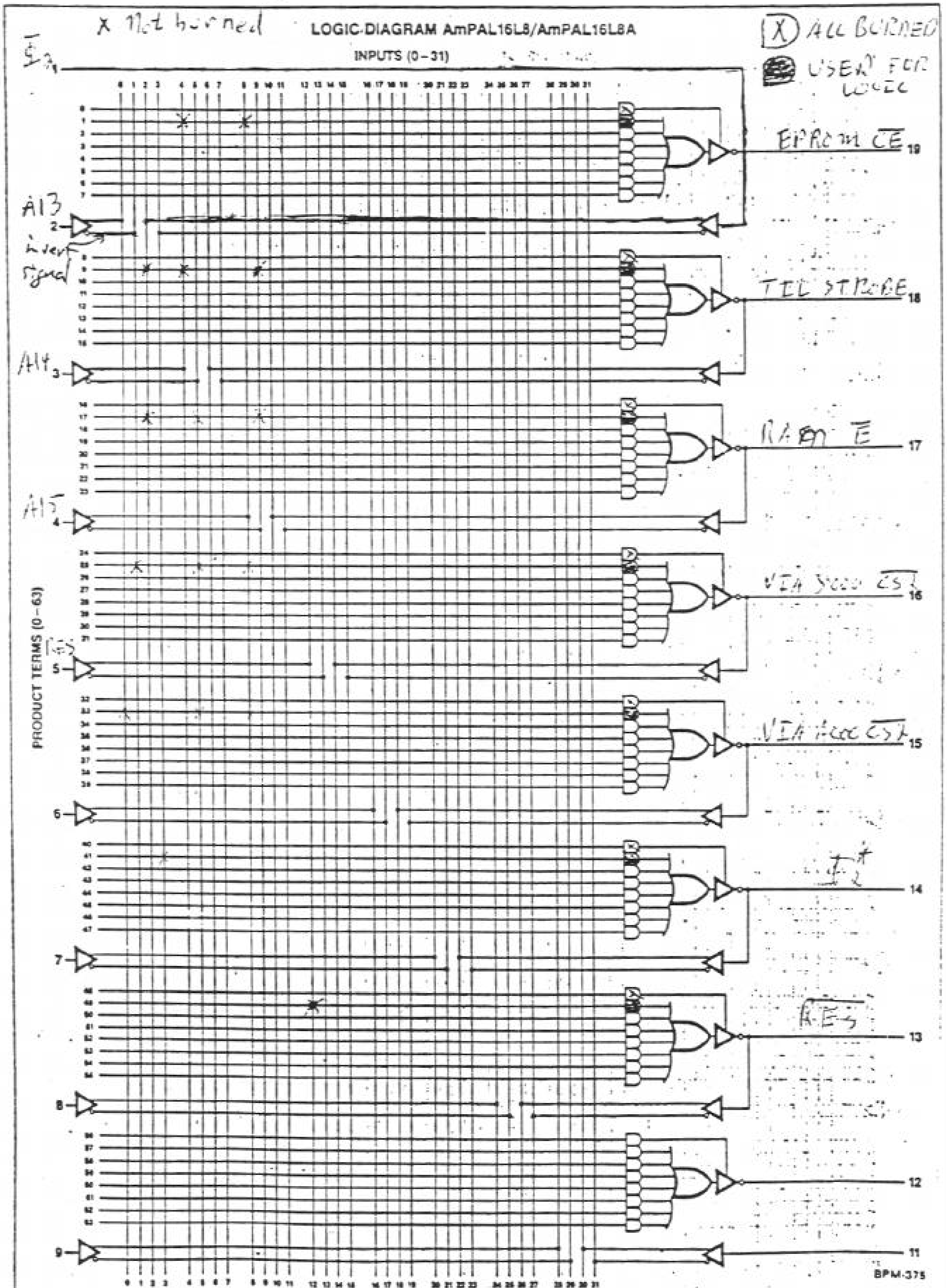
[redacted]



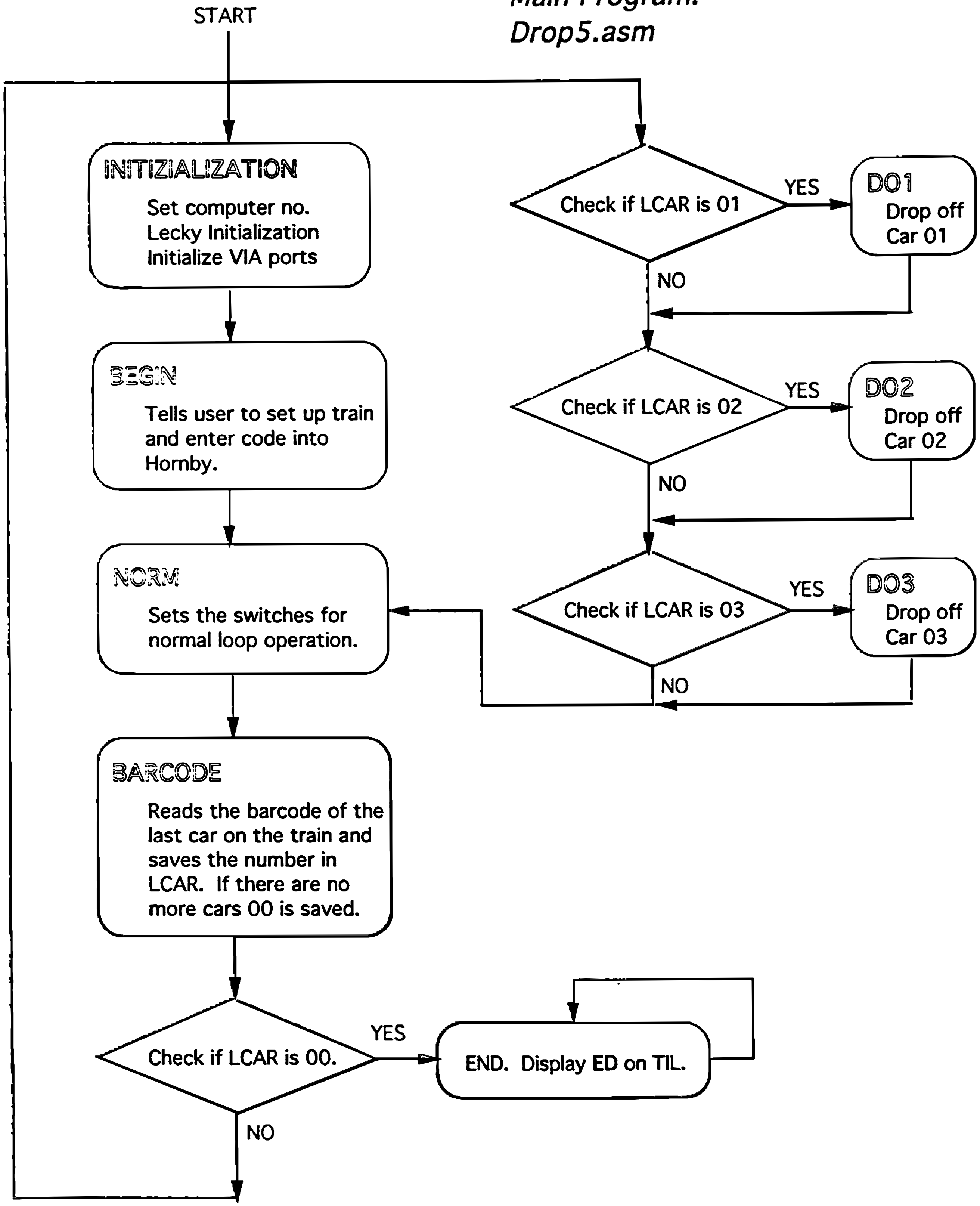
MECHANICAL DECOUPLER 1
MECHANICAL DECOUPLER 2
MECHANICAL DECOUPLER 3
OPTO-ISOLATOR
OPTO-ISOLATOR
OPTO-ISOLATOR
JONATHAN KOCHER & MOYE LITMAN

A-7

APPENDIX B


LOGIC DIAGRAM AmPAL16L8/AmPAL16L8A
INPUTS (0 – 31)
PRODUCT TERMS (0 – 63)
x Not burned
X ALL BURNED
USED FOR LOGIC
A13
invert signal
A14
A15
EPROM CE
TEL STROBE
RAM E
VIA 3000 CS1
VIA 4000 CS1
RES
BPM-375

Homeward Bound
Main Program:
Drop5.asm



APPENDIX C2

NORTH0 -- Waits for the North track to be clear.
NORTH1-- Waits for the North track to have a train.
SOUTH0-- Waits for the South track to be clear.
SOUTH1-- Waits for the North track to have a train.
WAIT-- Waits for a half a second.
GETHF1-- Waits for 1-1-Right input into Hornby.
GETHF2-- Waits for 1-2-Right input into Hornby.
GETHB-- Waits for 1-1-Left input into Hornby.
FRAM1-- Waits for Frame 1.
NEXT-- Waits for the next frame.
DECOUP--Triggers the mechanical decoupler.
NORM--Switches all switches to their normal mode.
TRACKK--Kills the isolated track section on the main board.
TRACKP--Powers the isolated track section on the main board.
OPTIC1,OPTIC2,OPTIC3--Waits until an optical signal is read from the corresponding sensors.
SW1F,SW1T--Switches switch 1 to forward and turn mode, respectively.
SW2F,SW2T--etc.
SW3F,SW3T--etc.
SW4F,SW4T--etc.
PUTAA--Changes 1 bit on the output ports for the switches, given an accumulator input of 1 non-zero bit.
PUTAA2--Changes 1 bit on the output ports for the relays, given an accumulatorinput of 1 non-zero bit.
SWITCH1,SWITCH2,SWITCH3,SWITCH4--Called by SW1F ..... Calls FRAM1 and NEXT and PUTAA

```
$INCLUDE LECKY3_0.ASM
; DROP5.ASM
; HOMEWARD BOUND
; by Mike Liebman & Jon Nichols
; May, 1995

;**************************************
;BIT CODES
;**************************************
REL1    EQU     $10     ;0001 0000 FIRST RELAY
REL2    EQU     $20     ;0010 0000 SECOND RELAY

SW1     EQU     $01     ;0000 0001 FIRST SWITCH
SW2     EQU     $02     ;0000 0010 SECOND SWITCH
SW3     EQU     $04     ;0000 0100 THIRD SWITCH
SW4     EQU     $08     ;0000 1000 FOURTH SWITCH

MECH    EQU     $40     ;0100 0000 MECHANICAL DECOUPLERS
TKILL   EQU     $01     ;0000 0001 TRACK KILL
OP1     EQU     $10     ;0001 0000 FIRST OPTICAL SENSOR
OP2     EQU     $20     ;0010 0000 SECOND OPTICAL SENSOR
OP3     EQU     $40     ;0100 0000
BBITS   EQU     $03     ;0000 0011 BOTTOM BITS OF SHIFT REGISTER
CLEAR   EQU     $08     ;0000 1000 CLEAR PULSE FOR SHIFT REGISTER

;**************************************
;VARIABLES  (OUR MAILBOXES)
;**************************************

PAA     EQU     $0610   ;VIA A000 PORT A mirror variable

        ;USED IN BARCODE SUBROUTINE:
VAR3    EQU     $0611   ;VARIABLE
CURRENT EQU     $0612   ;CURRENT NUMBER READ FROM BARCODE
CODE    EQU     $0613   ;CODE READ LAST TIME
COUNT   EQU     $0614   ;COUNT
LCAR    EQU     $0615   ;BARCODE ON THE LAST CAR OF THE TRAIN

;**************************************
;INITIALIZATION
;**************************************
        ORG $E000
        SEI             ;set interrupt disable bit
        CLD             ;clear decimal mode
        LDX #$FF
        TXS             ;set up stack

        LDA #$01        ;We are Bench Number 1
        STA $03FC       ;Set computer ID to 1

        LDA #$00
        STA $03FF       ;Set DFLAG to 0
        JSR INIT        ;Initialize echo routine

        LDA #$00
        STA NOVRD
        STA SOVRD       ;Initialize overide mailboxes

        LDA #$11
        STA $4000       ;Display initalization code 11
        JSR WAIT

        LDA #$FF
        STA $A001       ;Intialize Port A to all high
        LDA #$7F
        STA $A003       ;Set Port A on VIA at A000 to output
        LDA #$FF
        STA $0610       ;Initialize Mailbox for Port A to all high

        LDA #$FF
        STA $8001       ;Initialize Port A of 8000 VIA to all high
        LDA #$09        ;0000 1001
        STA $8003       ;SET THE DIRECTION OF 8000 VIA PORT A
        LDA #$FF
        STA $8001

        LDA #$22
        STA $4000       ;Diplay initialization complete code 22
        JSR WAIT

;**************************************
;MAIN PROGRAM
;**************************************
        JSR BEGIN       ;TELLS USER TO PLACE TRAIN ON TRACK IN CORRECT
                        ;POSITION, POWER TRAIN FORWARD AND ENTER 1 1
                        ;RIGHT INTO HORNBY

MAIN    JSR NORM        ;PLACE SWITCHES INTO NORMAL MODE
        JSR BARCODE     ;READ BARCODES UNTIL TRAIN REACHES SOUTH CLEAR
        LDA LCAR        ;READ LAST CAR ON TRAIN
        STA $4000       ;SEND TO TIL

        CMP #$00
        BNE N1
        JSR EN1         ;IF THERE ARE NO CARS LEFT, END AND DISPLAY ED

N1      CMP #$01        ;IF CAR 1 IS LAST,
        BNE N2
        JSR DO1         ;DROP OFF CAR 1

N2      CMP #$02        ;IF CAR 2 IS LAST,
        BNE N3
        JSR DO2         ;DROP OFF CAR 2

N3      CMP #$03        ;IF CAR 3 IS LAST,
        BNE N4
        JSR DO3         ;DROP OFF CAR

N4      JMP MAIN

BEGIN   LDA #$FA
        STA $4000
        JSR GETHF1
        RTS

NORTH0  LDA #$2A
        STA $4000
        LDA NTRAIN      ;WAITS UNTIL NORTH TRACK IS CLEARED
        BNE NORTH0
        RTS

SOUTH0  LDA #$5A        ;WAIT FOR SOUTH TRACK CLEAR
        STA $4000
        LDA STRAIN
        BNE SOUTH0
        RTS

NORTH1  LDA #$2B        ;WAIT FOR NORTH TRACK TO HAVE A TRAIN
```

```
        STA $4000
        LDA NTRAIN
        BEQ NORTH1
        RTS

SOUTH1  LDA #$5B        ;WAIT FOR SOUTH TRACK TO HAVE A TRAIN
        STA $4000
        LDA STRAIN
        BEQ SOUTH1
        RTS

GETHB   LDA RDATA
        CMP #$81        ;RETURNS WHEN RDATA IS 81, USED FOR BACKWARDS PROMPT
        BNE GETHB
        RTS

GETHF1  LDA RDATA       ;RETURNS WHEN RDATA IS 01, USED FOR FORWARDS PROMPT
        CMP #$01
        BNE GETHF1
        RTS

GETHF2  LDA RDATA       ;RETURNS WHEN RDATA IS 02 USED FOR SLOW SPEED
        CMP #$02
        BNE GETHF2
        RTS

DO1     JSR SOUTH0
        LDA #$FF
        STA NOVRD       ;OVERRIDES NORTH RELAY TO STOP TRAIN
        STA SOVRD       ;OVERRIDES SOUTH RELAY TO PREVEN TRAINS FROM
                        ;BUMPING INTO EACH OTHER
        LDA #$BA
        STA $4000       ;TELLS USER TO REVERSE TRAIN AND SEND 1 1 LEFT
                        ;TO HORNBY
        JSR GETHB       ;WAITS FOR BACKWARDS PROMPT
        JSR SW4F
        JSR SW2T        ;TURNS SWITCHES FOR REVERSING LOOP AND BACKING UP
        LDA #$00
        STA NOVRD       ;UNOVERRIDES NORTH RELAY AND STARTS TRAIN
        JSR OPTIC1      ;WAITS FOR OPTICAL SENSOR TO TRIGGER
        JSR TRACKK      ;KILLS TRACK
        LDA #$F0        ;TELLS USER TO SEND TRAIN FORWARD AND ENTER
                        ;1 2 RIGHT
        STA $4000
        JSR GETHF2      ;WAITS UNTIL STUFF IS ENTERED
        JSR TRACKP      ;POWERS TRACK
        JSR OPTIC1      ;WAITS UNTIL OPTICAL SENSOR SENSES
        JSR DECOUP      ;DECOUPLES
        JSR TRACKK      ;KILLS TRACK
        LDA #$FA
        STA $4000       ;TELLS USER TO SPEED UP TRAIN
        JSR GETHF1      ;WAITS UNTIL 1 1 RIGHT IS ENTERED
        JSR TRACKP      ;POWERS TRACK
        LDA #$FF        ;UNOVERRIDES NORTH RELAY
        STA NOVRD
        JSR GETHF1
        LDA #$00
        STA NOVRD

        JSR SW2F        ;SWITCHES SWITCH 2 TO NORMAL MODE
        JSR NORTH1      ;WAITS UNTIL TRAIN IS REGISTERED ON NORTH TRACK
        JSR SW4T        ;TURNS SWITCH 4 BACK TO NORMAL MODE
        LDA #00
        STA SOVRD       ;UNOVERRIDES SOUTH RELAY TO LET OTHER CARS IN
        RTS

DO2     LDA #$FF
        STA NOVRD       ;KILL NORTH RELAY TRACK
        LDA #$FA
        STA $4000       ;DISPLAY 'FA' FOR FAST
        JSR GETHF1      ;WAIT FOR 1 1 RIGHT
        LDA #$00
        STA NOVRD       ;OVERRIDES NORTH RELAY
        JSR NORTH1      ;WAITS UNTIL NORTH TRAIN IS REGISTERED
        JSR NORTH0      ;WAITS UNTIL NORTH TRAIN IS CLEARED
        LDA #$FF
        STA SOVRD       ;OVERRIDES THE SOUTH RELAY AND STOPS TRAIN
        STA NOVRD       ;OVERRIDES THE NORTH RELAY TO PREVENT OTHER CARS
                        ;FROM COMING ON
        LDA #$BA        ;PROMPTS THE USER TO REVERSE THE TRAIN
        STA $4000       ;AND ENTER 1 1 LEFT INTO THE HORNBY
        JSR GETHB       ;WAITS UNTIL 1 1 LEFT IS ENTERED INTO HORNBY
        JSR SW1T        ;SWITCHES SWITCH 1 TO TURN
        LDA #$00
        STA SOVRD       ;UNOVERRIDES THE SOUTH RELAY AND STARTS TRAIN
        JSR OPTIC2      ;WAITS UNTIL OPTICAL PULSE IS FELT
        JSR TRACKK      ;KILLS TRACK
        LDA #$F0        ;TELLS USER TO SEND TRAIN FORWARD AND ENTER
        STA $4000       ;1 2 RIGHT INTO HORNBY
        JSR GETHF2      ;WAITS UNTIL 1 2 RIGHT IS ENTERED INTO HORNBY
        JSR TRACKP      ;POWERS TRACK
        JSR OPTIC2
        JSR DECOUP      ;DECOUPLES LAST TRAIN
        JSR TRACKK      ;KILLS TRACK
        LDA #$FA        ;
        STA $4000
        JSR GETHF1      ;WAITS UNTIL 1 1 RIGHT IS ENTERED
        JSR TRACKP
        JSR SOUTH1      ;WAITS UNTIL SOUTH TRAIN IS REGISTERED TO SWITCH
        JSR SW1F        ;SWITCHES SWITCH 1 TO NORMAL MODE
        LDA #00
        STA NOVRD       ;UNOVERRIDES NORTH RELAY TO ALLOW OTHER CARS ON
        RTS

DO3     JSR SOUTH0      ;WAITS UNTIL SOUTH TRAIN IS CLEARED
        LDA #$FF
        STA NOVRD       ;OVERRIDES THE NORTH RELAY AND STOPS TRAIN
        STA SOVRD       ;OVERRIDES THE SOUTH RELAY TO PREVENT OTHER CARS
                        ;FROM COMING ON
        LDA #$BA        ;PROMPTS THE USER TO REVERSE THE TRAIN
        STA $4000       ;AND ENTER 1 1 LEFT INTO THE HORNBY
        JSR GETHB       ;WAITS UNTIL 1 1 LEFT IS ENTERED INTO HORNBY
        JSR SW3F        ;SWITCHES SWITCH 3 TO FORWARD
        LDA #$00
        STA NOVRD       ;UNOVERRIDES THE NORTH RELAY AND STARTS TRAIN
        JSR OPTIC3      ;WAITS UNTIL OPTICAL PULSE IS FELT
        JSR TRACKK      ;KILLS TRACK
        LDA #$F0        ;TELLS USER TO SEND TRAIN FORWARD AND ENTER
        STA $4000       ;1 2 RIGHT INTO HORNBY
        JSR GETHF2      ;WAITS UNTIL 1 2 RIGHT IS ENTERED INTO HORNBY
        JSR TRACKP      ;POWERS TRACK
        JSR OPTIC3      ;WAITS UNTIL OPTICAL PULSE IS FELT BY #3
        JSR DECOUP      ;DECOUPLES LAST TRAIN
        JSR TRACKK      ;KILLS TRACK
        LDA #$FA
        STA $4000       ;TELLS USER TO SEND TRAIN FAST
        JSR GETHF1      ;WAITS FOR USER TO ENTER 1-1 RIGHT
        JSR TRACKP      ;POWERS TRACK
```

```
        JSR NORTH1      ;WAITS UNTIL NORTH TRAIN IS REGISTERED TO SWITCH
        JSR SW3T        ;SWITCHES SWITCH3 TO NORMAL MODE
        LDA #00
        STA SOVRD       ;UNOVERRIDES SOUTH RELAY TO ALLOW OTHER CARS ON
        RTS

EN1     JSR WAIT        ;ENDS PROJECT AND DISPLAYS ED
        JSR WAIT
        JSR WAIT
        LDA #$ED
        STA $4000
EN2     JMP EN2         ;INFINITE LOOP

;***********************
;NORM: SETS SWITCHES FOR NORMAL CW LOOP OPERATION
;***********************
NORM    JSR SW1F
        JSR SW2F
        JSR SW3T
        JSR SW4T
        RTS

;****************************************
;FRAME NO. SUBROUTINES: FRAM1 & NEXT
;****************************************
; Waits for frame number 1
FRAM1   LDA #$01        ;
        CMP FRANUM      ;Sets Z flag to 1 if equal, 0 otherwise
        BNE FRAM1       ;Branch if Z equals 0, so wait for frame 1
        RTS

; Waits for next frame
NEXT    LDA FRANUM      ;Get current framenumber
NEXT2   CMP FRANUM      ;See if new framenumber is different
        BEQ NEXT2       ;If they are the same keep waiting
        RTS


;******************************
;WAIT  ABOUT 1/2 SECOND
;******************************
WAIT    LDX #$FF
        LDY #$FF
LOP1    DEX
        BNE LOP1
        DEY
        BNE LOP1
        RTS

;*****************************************
;TRACK KILL SUBROUTINES: TRACKK & TRACKP
;*****************************************
TRACKK  LDA #TKILL      ;CHANGE VIA LINE TO ZERO
        EOR #$FF        ;"NOT"
        STA $8001
        RTS

TRACKP  LDA $FF         ;CHANGE VIA LINE TO 1
        STA $8001
        RTS

;***********************
;DECOUPLE SUBROUTINE
;***********************
DECOUP  LDX #$4F        ;SET INTIAL VALUE OF COUNTING REGISTER
CCC     JSR FRAM1
        LDA #MECH       ;TURN DECOUPLER ON
        JSR PUTAA       ;SEND TO VIA
        JSR NEXT        ;FRAME 2
        JSR NEXT        ;FRAME 3
        JSR NEXT        ;FRAME 4
        LDA #MECH
        JSR PUTAA
        DEX             ;DECREMENT X REGISTER
        BNE CCC         ;LOOPS SO IT DECOUPLES FOR A WHILE
        RTS

;************************
;OPTICAL SENSOR SUBROUTINES
;************************
OPTIC1  LDA $8001       ;THESE ROUTINES WAIT UNTIL AN OPTICAL PULSE IS FELT
        AND #OP1        ;BY LOOKING AT THE INPUT PORTS ON THE VIA
        BEQ OPTIC1
        RTS

OPTIC2  LDA $8001
        AND #OP2
        BEQ OPTIC2
        RTS

OPTIC3  LDA $8001
        AND #OP3
        BEQ OPTIC3
        RTS

;****************************************
;SUBROUTINE SWITCH
;****************************************
SW1F    LDA $0610       ;LOAD A PORT OF VIA AT A000 MAILBOX
        ORA #REL1       ;OR WITH RELAY BITS
        JSR PUTAA2      ;PLACES ACCUMULATOR INTO VIA AND MAILBOX
        JSR SWITCH1     ;PULSES SWITCH
        RTS

SW1T    LDA #REL1
        EOR #$FF        ;"NOT" THE RELAY
        AND $0610       ;AND WITH A PORT OF VIA AT A000 MAILBOX
        JSR PUTAA2
        JSR SWITCH1
        RTS

SW2F    LDA $0610       ;LOAD A PORT OF VIA AT A000 MAILBOX
        ORA #REL1       ;OR WITH RELAY BITS
        JSR PUTAA2      ;PLACES ACCUMULATOR INTO VIA AND MAILBOX
        JSR SWITCH2     ;PULSES SWITCH
        RTS

SW2T    LDA #REL1
        EOR #$FF        ;"NOT"
        AND $0610       ;AND WITH A PORT OF VIA AT A000 MAILBOX
        JSR PUTAA2
        JSR SWITCH2
        RTS

SW3F    LDA $0610       ;LOAD A PORT OF VIA AT A000 MAILBOX
        ORA #REL2       ;OR WITH RELAY BITS
        JSR PUTAA2      ;PLACES ACCUMULATOR INTO VIA AND MAILBOX
```