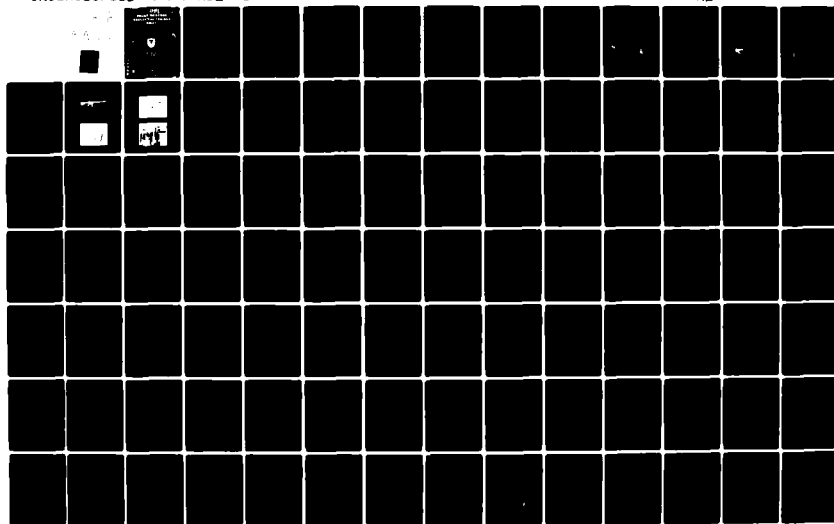
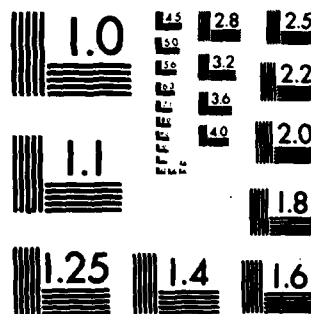


AD-A091 077 OFFICE OF THE PROJECT MANAGER FOR TRAINING DEVICES 0--ETC F/6 5/9
SQUAD WEAPONS ANALYTICAL TRAINER (SWAT) M-16 VERSION.(U)
JUL 80 A MARSHALL, B SHAW, H TOWLE

UNCLASSIFIED PM TRADE-RE-0005

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

DA091072

DISCLAIMER NOTICE

**THIS DOCUMENT IS BEST QUALITY
PRACTICABLE. THE COPY FURNISHED
TO DTIC CONTAINED A SIGNIFICANT
NUMBER OF PAGES WHICH DO NOT
REPRODUCE LEGIBLY.**

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER (14) PM TRADE — RE-0005	2. GOVT ACCESSION NO. DADA4998	3. RECIPIENT'S CATALOG NUMBER AD-A091 077
4. TITLE (and Subtitle) (6) Squad Weapon Analytical Trainer (SWAT), Final Report M-16 Version.		5. TYPE OF REPORT & PERIOD COVERED (9) Final Report
6. AUTHOR(s) (10) Albert Marshall, Bon Shaw, D. Herbert Towle, George Siragusa and Tom Riordan		7. PERFORMING ORG. REPORT NUMBER
8. PERFORMING ORGANIZATION NAME AND ADDRESS		9. CONTRACT OR GRANT NUMBER(s) Project 8781
11. CONTROLLING OFFICE NAME AND ADDRESS U.S. Army (DRCPM-TND-RE) Project Manager for Training Devices Orlando, Florida 32813		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
12. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) (12) 152		11. REPORT DATE (11) July 1988
		12. NUMBER OF PAGES
		13. SECURITY CLASS. (of this report) Unclassified
		14a. DECLASSIFICATION/DOWNGRADING SCHEDULE
15. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Trainer Tactical Infantry Weapons M-16		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) The Squad Weapons Analytical Trainer (SWAT) is an electro-optic based, microcomputer controlled, training device that enables tactical infantry weapons training with a M-16 rifle, under a simulated high stress battle field environment.		

DD FORM 1473 JAN 73

EDITION OF 1 NOV 68 IS OBSOLETE
GPO 5102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

409997

JP

SUMMARY

The Squad Weapons Analytical Trainer (SWAT), is an electro-optic based, microcomputer controlled, training device that enables tactical infantry weapons training with an M-16 rifle and 16mm motion picture projectors which simulate a high stress battlefield environment. In a short period of time a trainee can be subjected to a large variety of combat situations where each trainee's performance is analyzed in real-time and immediate feedback is given to both the trainees and instructor. Combat scenarios can be changed to fit any potential battlefield requirement.

Prototype models were constructed by the Research and Technology Department, NTEC, Orlando, Florida for both PM TRADE and the U.S. Marine Corps. These models were successfully tested by the U.S. Army Infantry Board (USAIB) for the Directorate of Training Developments, U.S. Army Infantry School, Fort Benning, Georgia and at Camp Lejeune, North Carolina by the U.S. Marine Corps. It was stated that the tests did give some evidence of the SWAT system's potential for training transfer (Ref. 9). Furthermore, enlisted men, snipers and a variety of General, Field and Company grade officers who fired and observed the SWAT stated that it was a valuable training tool (Ref. 8).

PM TRADE and USMC sponsored work is continuing on this program to develop the capability to add other weapons i.e., Dragon, LAW, M-60 machine gun, etc.

The PM TRADE project manager, was Dr. B. Rashis. The authors wish to thank him for the helpful assistance he gave during this program.

Accession For	
NTIS GRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	☐
By _____	
Distribution /	
Availability Codes	
For Author or	
Dist	Special
A-3 F-4	

TABLE OF CONTENTS

I. Introduction	1
II. System Description	3
III. System Design	9
A. Projectors	9
B. Rifle Electronics	11
C. Computer Voice and Audio Design	16
D. Bang and Recoil System	19
E. Distribution of Fire and Weapon Movement Monitoring	22
F. Rifle Mockup	26
G. Microcomputer Control System	26
1. Single Board Computer	27
2. The Interface Board	28
3. 8080 Program	61
4. Score Display and Worst Performance	64
5. Self Check	64
H. Film Animation	66
IV. Conclusions	68
Appendix	
A. Functional Description of Console Switches	A1
B. Test Equipment	B1
C. System Program	C1
D. UPI-41 Program	D1
E. Self Check Program	E1

ILLUSTRATIONS

I- 1	Artists Concept	1
II- 1	System Block Diagram	3
II- 2	Rifle Electronics Block Diagram	4
II- 3	M-16 Training Rifle	6
II- 4	Instructor's Console	6
II- 5	Trainees Firing at Screen	7
II- 6	Synchronized Visual and IR Projectors	7
III- 1	Transmittance of Hot and Cold Mirrors	9
III- 2	Target Present Decoder	10
III- 3	Photo Detector Spectral Response and Geometry	11
III- 4	Pre-Amplifier - Rifle Electronics - Board #1	12
III- 5	Bi-Quad Active Filter	13
III- 6	Universal Active Filters and Voltage Comparators - Board #2	14
III- 7	NAND Gates and J-K Flip Flops - Board #3	15
III- 8	One-Shot and Counters - Board #4	17
III- 9	NAND Gates, One-Shot and Timer - Board #5	18
III-10	Audio Amplifier - Board #6	20
III-11	Bang Simulation Generator - Board #8	21
III-12	Recoil Circuit - Board #7	23
III-13	Laser Signal Generation - Board #10	24
III-14	Laser Pulser	25
III-15	80/20-4 Main and Interface Boards	29
III-16	Interface Board Layout	30
III-17	Interface Board Schematic (1 of 3)	31
III-18	Interface Board Schematic (2 of 3)	32

III-19	Interface Board Schematic (3 of 3)	33
III-20	Typical Printout Format on Terminal	35
III-21	UPI-41 Intelligent Controller System Block Diagram	36
III-22	SBC 80/20 to Controller	38
III-23	SBC 80/20 to UPI-41 Data BYTE	38
III-24	SBC 80/20 Input Source Configuration	39
III-25	SBC 80/20 to Controller - Clock Connections	41
III-26	Serial Transmission Character	42
III-27	ADM-3 Screen Use	43
III-28	Character String Transmitted to ADM-3A	44
III-29	Control Switches to Control Interface	45
III-30	UPI-41 Single Chip Microcomputer Block Diagram	46
III-31	UPI-41 Control Program Memory Map	47
III-32	UPI-41 RAM Memory Map	48
III-33	UPI-41 Control Program Flowchart Processing Loop	50
III-34	UPI-41 Control Program Flowchart - Interrupt Service Routine	51
III-35	Interrupt Routine Flowchart	52
III-36	Fixed Base FIFO Operation	54
III-37	Moving Base FIFO Operation	55
III-38	Data Access Segment Flowchart	57
III-39	UPI-41 Instruction Cycle	59
III-40	UPI-41 Character Transmission Subroutine	60
III-41	8080 Program Strategy	62
III-42	Program Flowchart	63
A-1	Instructor's Console - Front Panel Control Locations	71
B-1	Laser Checker	74
B-2	Rifle Checker	75
B-3	Rifle Substitution Box	79

B-4	Boresight Box	80
-----	-------------------------	----

TABLES

III- 1	Register Bank 1 Map	56
III- 2	Register Bank 0 Map	58

SECTION I

INTRODUCTION

The Squad Weapons Analytical Trainer (SWAT) is an electro-optic based, microcomputer controlled, training device that enables tactical infantry weapons training with an M-16 rifle, under a simulated high stress battlefield environment. In a short period of time a trainee can be subjected to a large variety of combat situations where each trainee's performance is analyzed in realtime, and immediate feedback is given to both the trainees and instructor. Combat scenarios can be changed to fit any potential battlefield requirement. An artist's concept of the trainer is shown in Figure I-1.

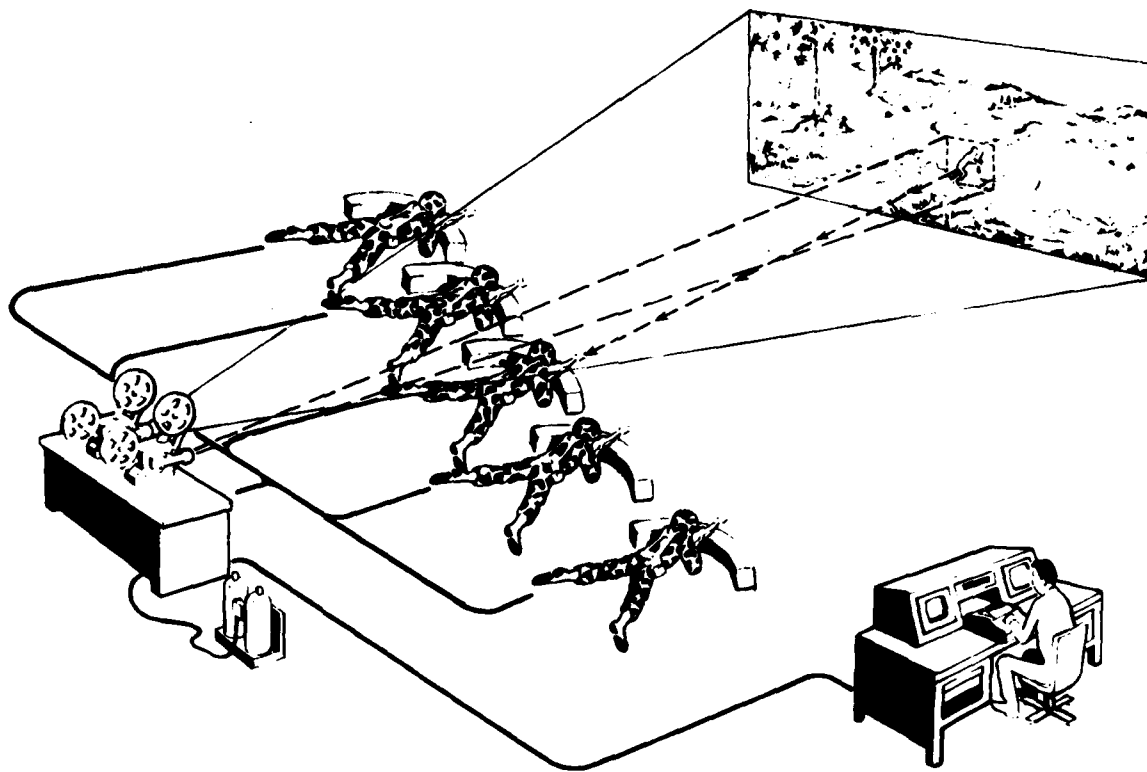


Figure I-1. Artist's Concept

This training device provides the trainees or instructor the following simulated weapons effects and feedback information:

- Weapon recoil
- Weapon bang
- Magazine action
- Automatic or single shot simulation

- Lead and elevation if applicable, is programmed in the system
- Real-time individual audio scoring feedback, using computer generated voice, via a headset
- Trainee feedback data displayed in columns on TV type monitor for instructor observation
- Reaction time
- Movement of weapon relative to correct kill zone is observed by instructor and recorded for playback.
- Lowest performer indicated to instructor
- Identification of trainee responsible for shooting with no target present
- Built-in self-check features
- Score determined
- Hardcopy of scoring results

SECTION II

SYSTEM DESCRIPTION

This section of the report describes the system. Details of the system design are included in Section III.

The system utilizes two motion picture projectors: a visual and an infrared (IR) target spot projector (see Figure II-1). The visual projector displays the battle scene including the visual targets. The infrared projector provides invisible infrared target areas at which the weapon must be aimed in order to score a hit. Lead is programmed into the infrared target film, which the weapon receiver detects, requiring the trainee to lead the target as necessary. Figure II-1 shows the visual target on the left and the infrared target on the right indicating that the target is moving to the right.

Each trainee has a simulated M-16 rifle with an attached infrared (IR) receiver. The IR detector is a four-quadrant photodiode. The four-quadrant target information and microcomputer logic determines kills, eight areas of near misses, and total misses. The regions of near miss include high, low, left, right, high right, high left, low left, and low right.

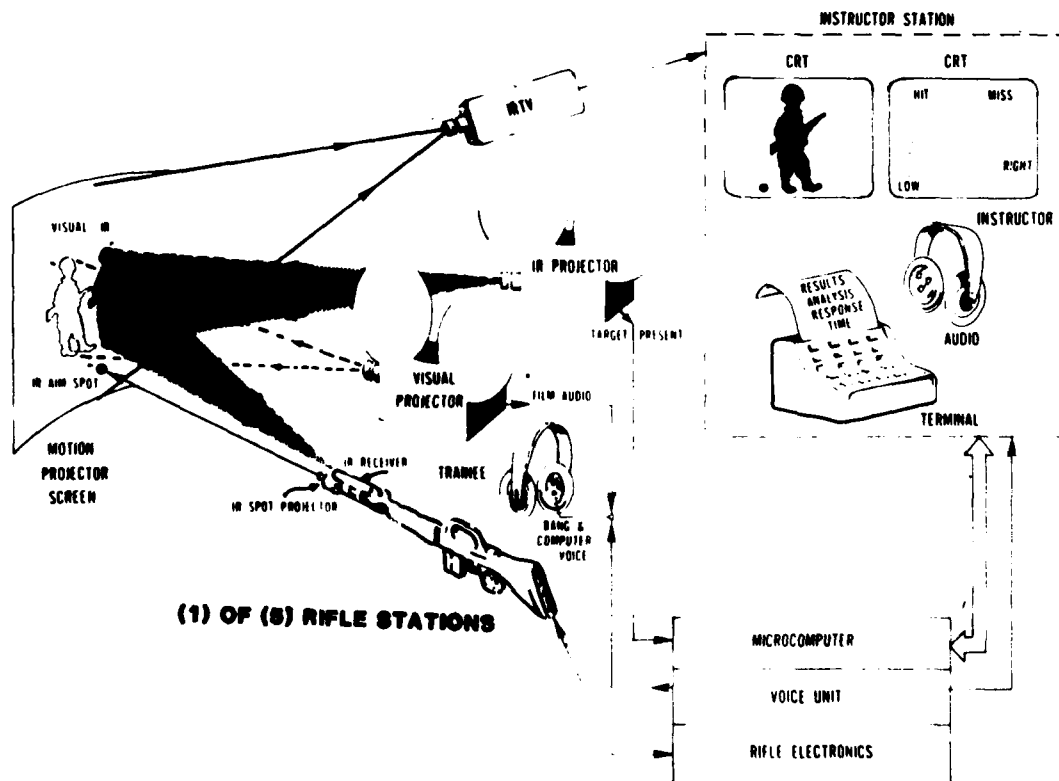


Figure II-1. System Block Diagram

When the trainee fires the weapon he hears a simulated bang and feels a recoil. Recoil is generated by a short pulse of air released near the front sight which drives the weapon high and to the right. An 8080 based microcomputer determines where the round would have hit using the detector's quadrant data and supplies this information to both a computer generated voice unit and a CRT display on the instructor's console. The computer voice unit drives both the trainee and instructor headsets. When a target appears on the screen, the IR projector outputs a target present signal from the magnetic audio stripe on the film. This signal starts a clock in the microcomputer which measures the time until the trainee fires, or effectively his reaction time. The target present signal is also used to determine the number of targets that appeared, targets ignored, targets shot at and if the trainee fired when no target was present. Trainee results are continuously displayed in columns on a CRT display on the instructor's station. At the completion of the exercise, the results, analyzes and response time are printed by a terminal at the instructor's station.

Distribution of fire can be monitored using a gallium arsenide laser infrared source located in the flash hider part of the rifle. The projected IR laser spot is invisible to the trainee but is detected by an infrared television camera and displayed by a CRT located on the instructor's console as shown in Figure II-1. When the rifle is fired the IR spot projector illuminates the screen with a small IR spot. If the instructor wants to continuously monitor rifle motion the IR aiming spot is left on continuously and the laser spot brightens when the trainee shoots. The TV camera data can also be recorded for playback during debrief.

Figure II-2 shows the rifle electronics and two projected targets. Discrimination of the infrared targets is enhanced by projecting the IR targets at frequencies different from the visual scene signals and amplifying the infrared targets. The motion picture projectors have also been modified to incorporate hot and cold mirrors, whose function will be described.

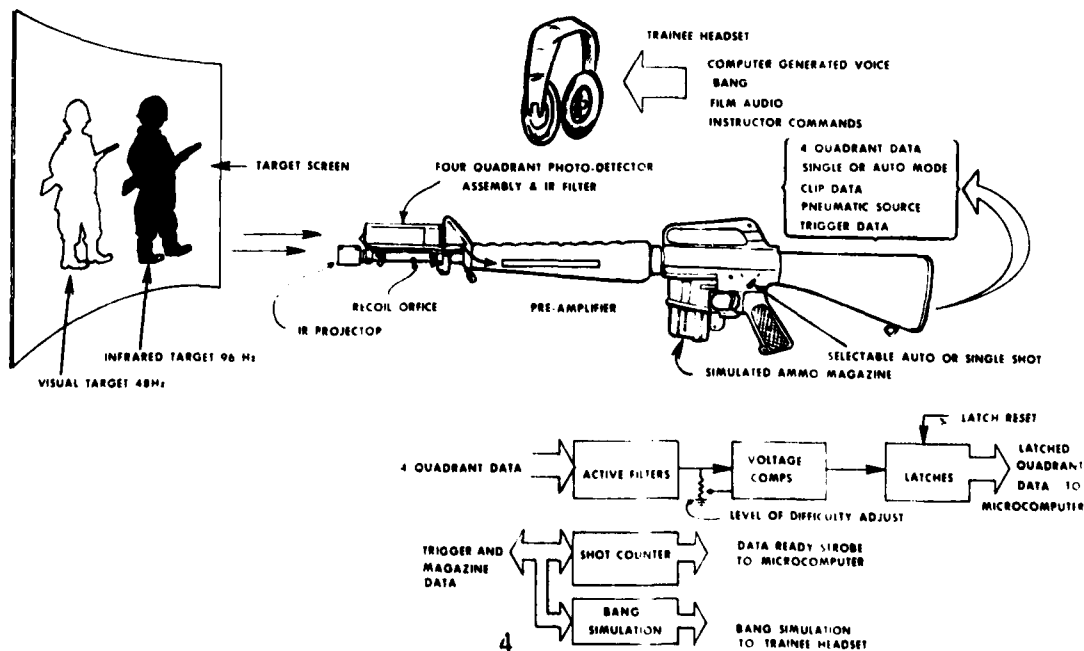


Figure II-2. Rifle Electronics Block Diagram

The visual projector contains a hot mirror. This multilayer dielectric mirror reflects or removes most of the infrared above 750 nanometer from the visual scene. The infrared projector contains a cold mirror. The cold mirror reflects the visual energy and passes the infrared energy above 750 nanometers. This allows a weapon equipped with an infrared receiver to ignore the visual data and obtain its target data from the infrared projector.

The S/N ratio of the system is further improved by using two different projector chopper frequencies. In the visual projector the chopper is a two bladed equally divided shutter. In the IR projector the chopper is a four bladed shutter. The visual scene is chopped or shuttered at a frequency of 48 Hz; the IR data is shuttered at a frequency of 96 Hz. By using two different chopping frequencies active filters in the weapons IR receiver can be tuned to detect the infrared target spot and ignore the visual battle scene. The projectors are frame locked together synchronously.

The rifle uses an IR detector consisting of a lens and a four-quadrant photo diode detector to detect infrared targets. An infrared filter is utilized in the weapon optical system to reduce the visual signal effect on the photo detector. The photo detector signals are amplified by two bi-FET operational amplifiers. A voltage comparator sets a threshold to establish a digital "one" or "zero". The voltage reference level of the comparator can be set to adjust the level of difficulty. The voltage comparator data is latched and delivered as input to the microcomputer system for data analysis, display and feedback.

The rifle can operate in either a single-shot or automatic mode and requires the trainee to reload after he has fired thirty rounds. The rifle's simulated magazine contains a capacitor. When the magazine is inserted into the rifle this internal capacitor is discharged, which resets a counter.

Bang simulation is achieved by filtering a noise source and then producing a noise envelope with a sharp rise time and exponential decay.

The training rifle is shown in Figure II-3. The four-quadrant detector is located on top of the barrel and the flash hider contains a gallium arsenide IR laser. The rifle is a replica but contains real sights that are adjustable. The plastic hose shown attached to the rifle, Figure II-3, is used to carry the air for recoil.

The instructor's console is shown in Figure II-4. The right hand CRT displays the verbal data transmitted to each trainee in five columns. The lowest score is automatically flagged by a LED under the applicable trainees column. This alerts the instructor so he can more closely observe that trainee. The left hand side of the console contains a CRT display used to monitor the weapon motion. Communication to the microcomputer is via a terminal shown in front of the instructor. See Appendix E for description of the switches on the console.



Figure II-3. M-16 Training Rifle



Figure II-4. Instructor's Console



Figure II-5. Trainees Firing at Screen

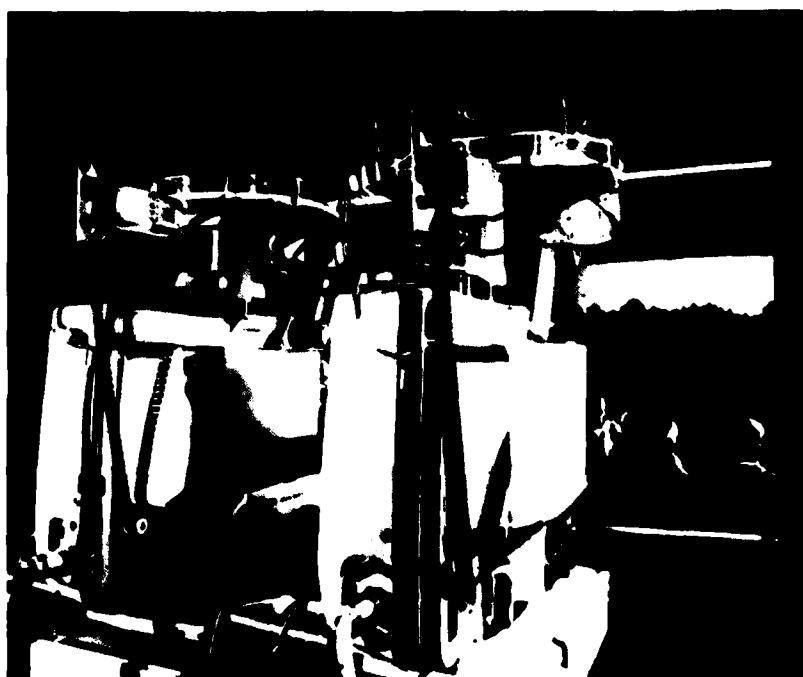


Figure II-6. Synchronized Visual and IR Projectors

Figure II-5 shows the trainees firing at the screen. Note each trainee wears a headset for individual feedback.

Figure II-6 shows the projectors. Loopers (a closed-loop film strip) are used so rewinding is not necessary. An auto-stop/auto-align feature is visible near the loopers.

The computer voice system is a solid state communications processor. It operates as a standard data terminal to the host 80/20 microcomputer system. The vocabulary has been digitized and stored in nonvolatile memory (PROM). The system contains thirty-two individually addressable words and five independent output channels. Thus, the computer voice system can talk to any or all of the five trainees while saying the same or different words or phrases. Each trainee wears a headset so he hears only the feedback applicable to his performance.

The system is controlled by a modified Intel 80/20 microcomputer system.

Section III, next, describes the system design.

SECTION III

SYSTEM DESIGN

A. PROJECTORS

The motion picture projectors are two Hokuskin, 16mm sound projectors equipped for frame-for-frame sync. The lamp is a 500 watt Xenon-arc, type KXL-500H. One projector is used as an IR target spot projector. The IR projector uses a cold mirror to remove the visual energy, Melles Griot, 03MHGD07. The transmittance of the hot and cold mirrors are shown in Figure III-1.

Loopers are utilized instead of reels to eliminate the necessity of rewinding the film.

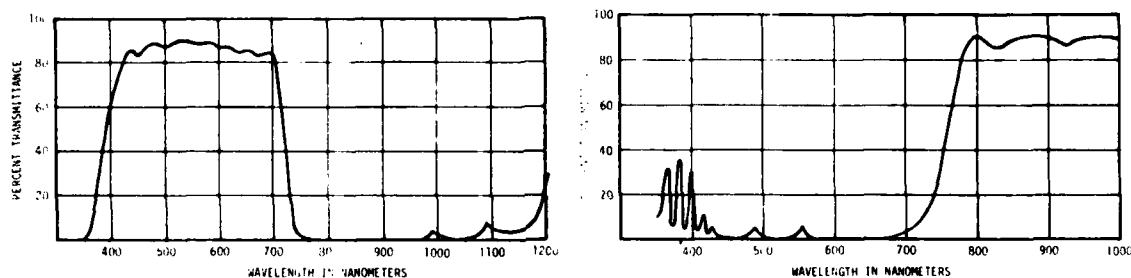


Figure III-1. Transmittance of Hot and Cold Mirrors

The projectors are equipped for either optical or magnetic sound reproduction. Sound for the battle scene is recorded for optical pickup on the visual projector.

Target present signals are recorded on the magnetic stripe of the IR film. The target present signal is a 1 KHz audio tone, which is decoded by an electronic tone decoder, Figure III-2.

The battle scene film was both taken and projected using a 25mm focal length lens to minimize perspective distortion.

The IR projector has a modified four bladed shutter which chops the IR data at a frequency of 96 Hz. The visual projector has a conventional two bladed chopper which chops the visual scene at 48 Hz.

The projectors are equipped with an auto-stop feature which allows the film to be stopped at any desired location by simply placing a foil metal strip on the desired stop location.

The screen is silver matte, 9 ft x 12 ft overall.

B. RIFLE ELECTRONICS

The rifle electronics detect the IR target spot, amplifies, discriminates and provides digital data to the 8080 based microcomputer.

The detector optics is a single element double convex lens, with a diameter of 29mm and focal length of 114mm.

The IR detector is a four-quadrant silicon photodiode. This device consists of four discrete elements on a single substrate with an active output lead from each element. When the weapon is aimed properly the infrared target spot is centered on the detector and the output current from each quadrant is equal. As the rifle is moved the currents change as a function of the location of the infrared target spot on the detector. Imbalance in the current indicates off-center position. The detector has an active area of 0.05" x 0.05" per element with a gap of 0.005" between elements. The detector physical geometry and spectral response is shown in Figure III-3.

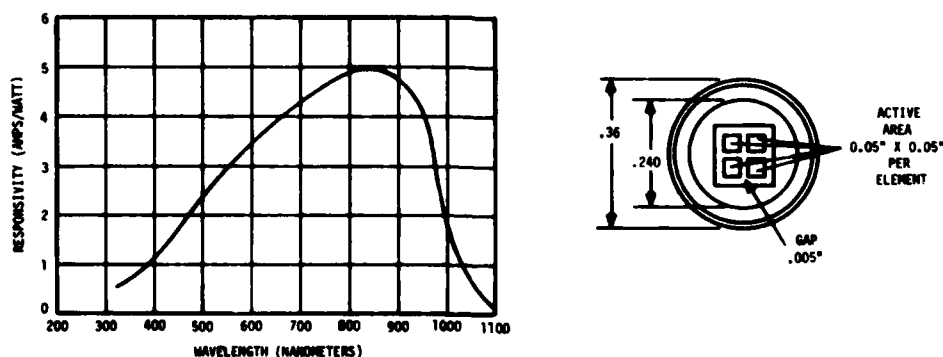
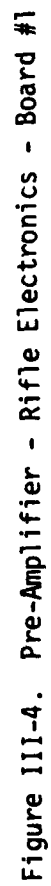


Figure III-3. Photo Detector Spectral Response and Geometry

The field of view of the IR detector is approximately seven inches on the screen.

The currents from the diode are input to an operational amplifier, TL082. The photo diode detector is basically a current source with an output impedance which is very large. The first stage of the current-to-voltage converter presents almost zero load impedance to ground because the inverting input appears as a virtual ground. The input current from the diode flows through the two Megohm feedback resistor, generating an output voltage.



$$\text{voltage out} = i_d R_f$$

where

$$\begin{aligned} R_f &= 2 \text{ Megohm} \\ i_d &= \text{detector current} \end{aligned}$$

A separate channel is used for each of the four quadrants. The output from the current-to-voltage amplifier goes to a noninverting amplifier with a gain of 23. This stage is also part of the TL082. The electronics described above is located on Board #1, Pre-Amp. (Figure III-4)

Input signals to the active filter are 48 Hz from the visual scene, 96 Hz from the IR target spot and any extraneous light. The active filter is used to pass and amplify the desired IR signal at 96 Hz and reject all other signals.

The UAF - 41 is a two pole active filter. It uses three operational amplifiers in a double integrator feedback loop to generate two conjugate poles. Location of the poles in the complex plane, and thus the natural frequency and Q are determined by external resistors.

The equivalent configuration of this band pass filter is shown in Figure III-5. The filter is designed for a 96 Hz center frequency with both a Q and gain of 50.

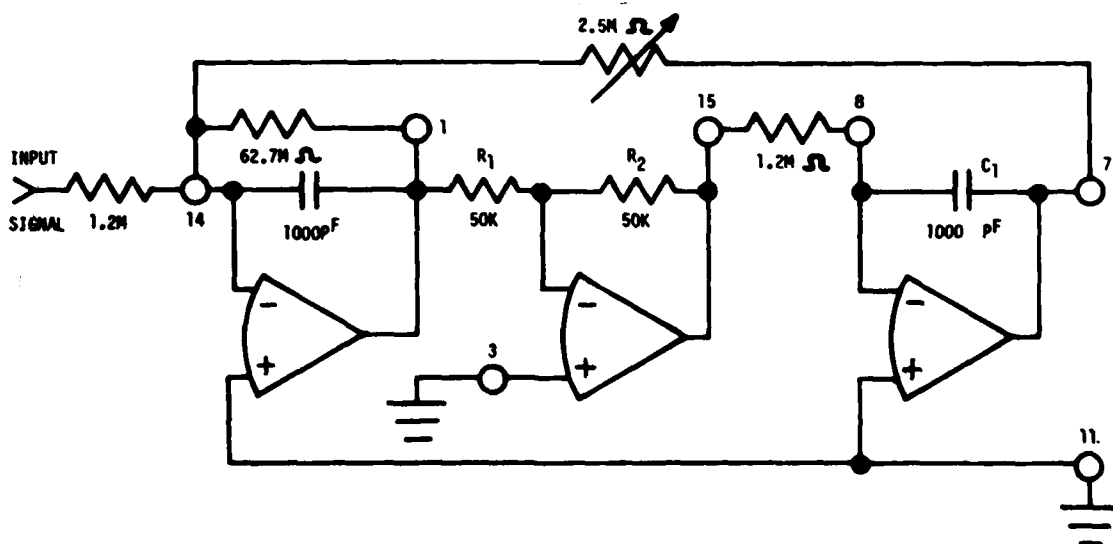


Figure III-5. Bi-Quad Active Filter

Both the active filters and voltage comparators are located on Board #2, Figure III-6. The output of the active filter is a sine wave with a frequency of 96 Hz. The output sine wave goes positive and negative about a zero volt reference level. This output is clamped and fed to a voltage comparator. The voltage comparator changes the analog detector signal to a digital signal. The input signal level for a one or zero is determined

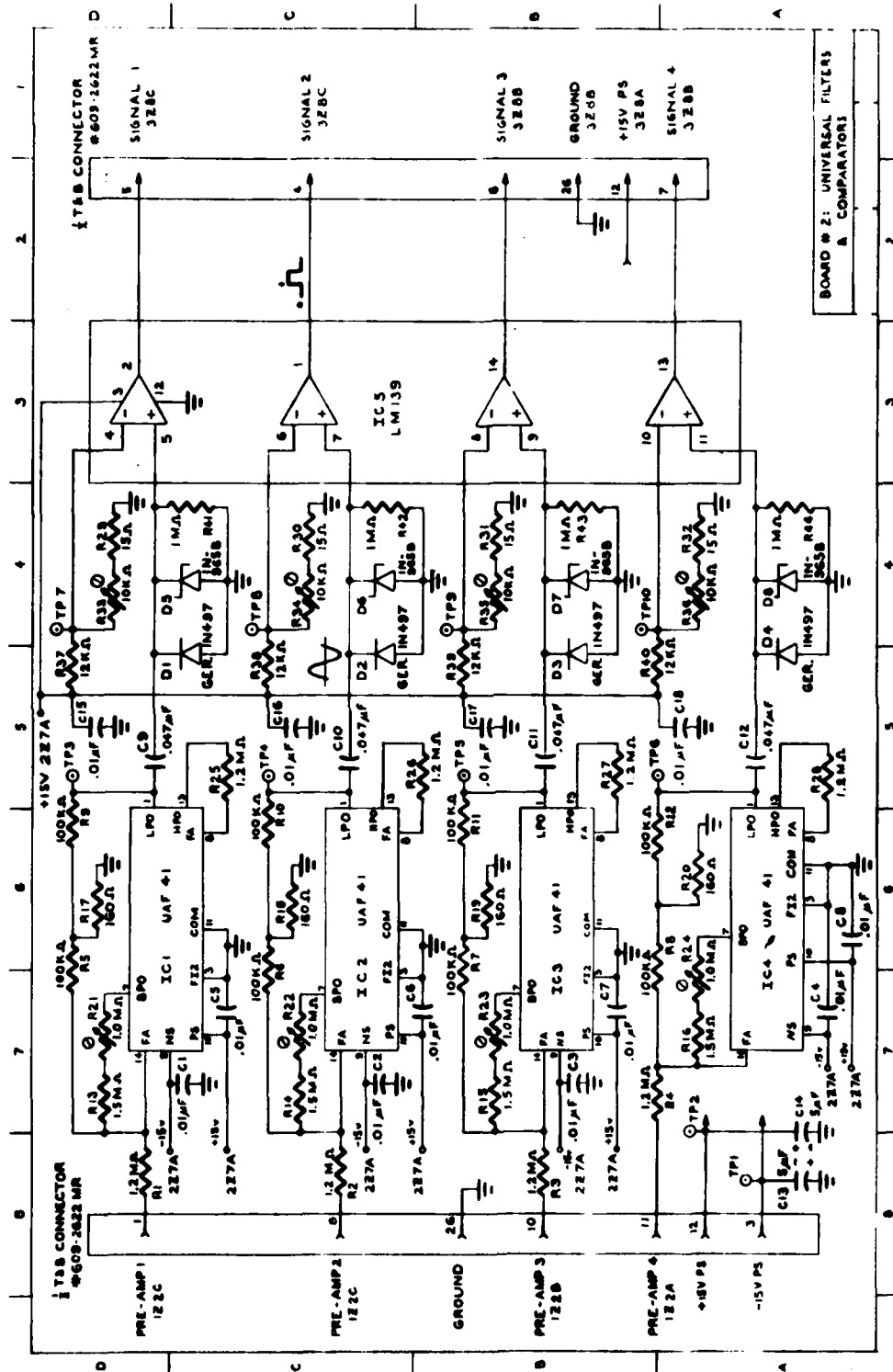


Figure III-6. Universal Active Filters and Voltage Comparators - Board #2

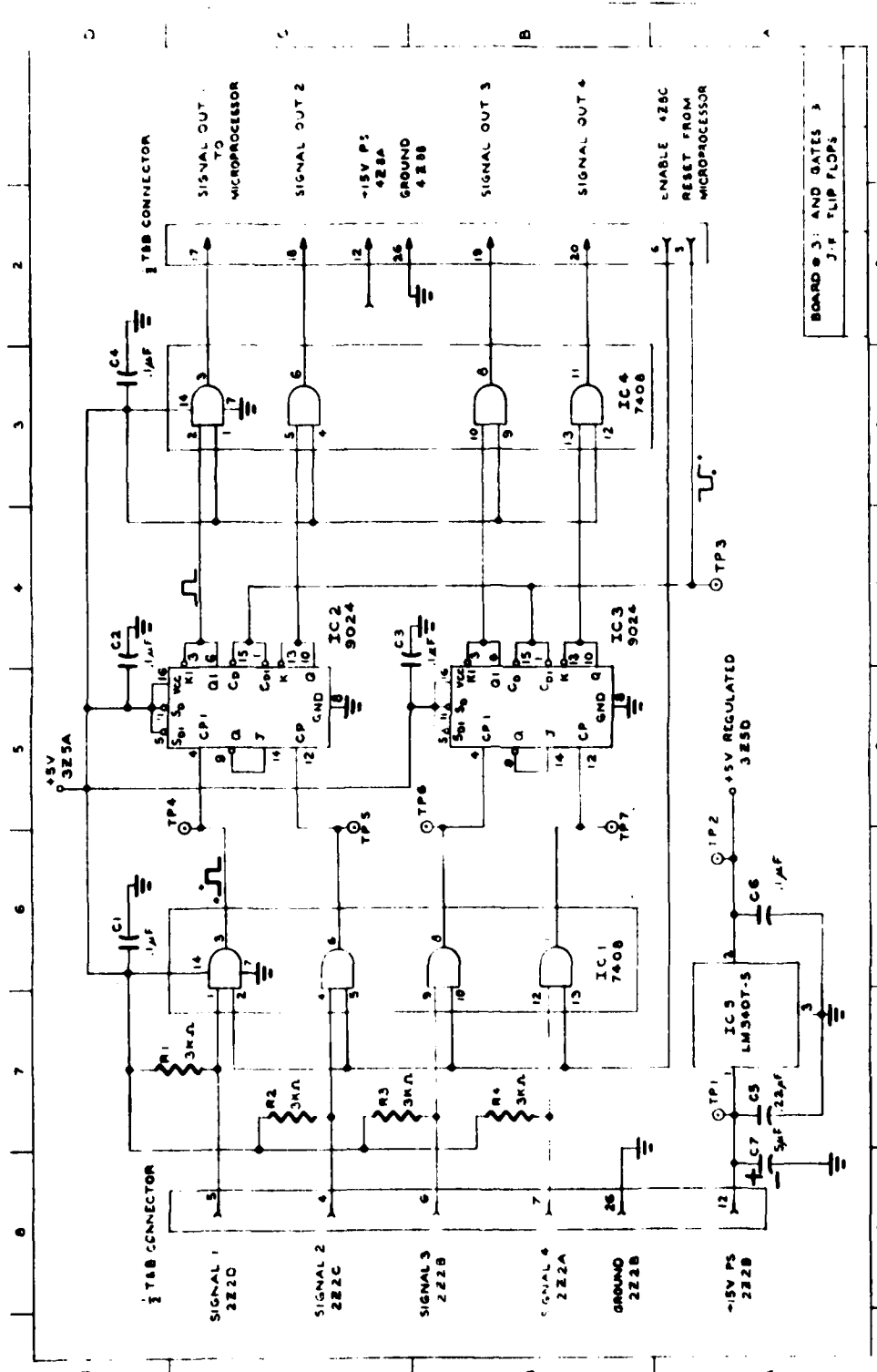


Figure III-7. NAND Gates and J-K Flip Flops - Board #3

by a resistor or reference voltage setting. Each of the four voltage comparator channels has its own reference voltage setting resistor, i.e., R33, R34, R35, and R36. The reference voltage setting controls the degree of difficulty in hitting a target. The detector signals next go to IC1, a 7408 AND gate, Board #3, Figure III-7. If the trainee pulls the rifle trigger and has rounds remaining in his magazine, the NAND gate is enabled by an input from Board #4. Board #4 is shown in Figure III-8. IC2 and IC3, Board #3 are 9024 JK flip flops configured as latches. Each 9024 has two latches. The 9024 is reset by the microprocessor after it has accepted the four-quadrant IR target spot data. IC4, Board #3 is a line driver.

Board #5, Figure III-9, is connected to the rifle trigger. IC1, a 5437, containing NAND gates, debounces the trigger and applies 5 volts to IC3. IC3, a timer, provides pulses of 12 Hz, which is the firing rate of the weapon. A one shot is also triggered and provides a single pulse. The output of Board #5 is determined by the setting of the single or auto fire switch on the simulated weapon. The setting of auto or single shot determines which gate on IC1 is active. If the trainee is in auto fire pulses at 12 Hz are provided Board #2. IC3 on Board #4 has a gate which will pass the signal if the counters IC1, IC2 on Board #4 indicate rounds are left. The counter enables IC1 on Board #3 and also enables the data ready pulse provided by IC4 on Board #4 to the microprocessor. IC4 is a one shot which generates a 10 μ sec data ready pulse for the microprocessor to indicate data is available. After the microprocessor has read the data it resets the latches; IC2 on Board #3.

The one shot IC6, Board #4, Figure III-8, is used to reset the counters. The dummy magazine contains a capacitor. In the "loaded" configuration the capacitor is charged to 5 volts. The magazines are easily loaded or charged by momentarily inserting them into a charging fixture.

When the dummy magazine or capacitor is inserted in the rifle it discharges through R4, providing the counter reset voltage. The magazine is reloaded by charging the capacitor in the magazine to 5 volts.

C. COMPUTER VOICE AND AUDIO SYSTEM

The Computer Voice System is a Business Communicator Model LVM-70 manufactured by VOTRAX, the Vocal Interface Division of Federal Screw Works, Troy, Michigan. The LVM-70 was designed specifically to be used as a concentrator for touch-tone based information systems.

Up to 32 words (16 seconds) are available with up to eight audio output channels. The trainer utilizes an output channel for each trainer. When a shot is fired by a trainee the host computer (80/20) decodes the incoming rifle data and then sends three bytes of serial data to the LVM-70 specifying a start word, a trainee identification word, and the appropriate voice response code. The voice output line for each trainee is routed to the trainees audio mixer/amplifier, Board #6, Figure III-10, as well as the instructor's control panel.

The LVM-70 voice communicator can be replaced in later models for roughly 1/4 the original cost, due to technological advances.

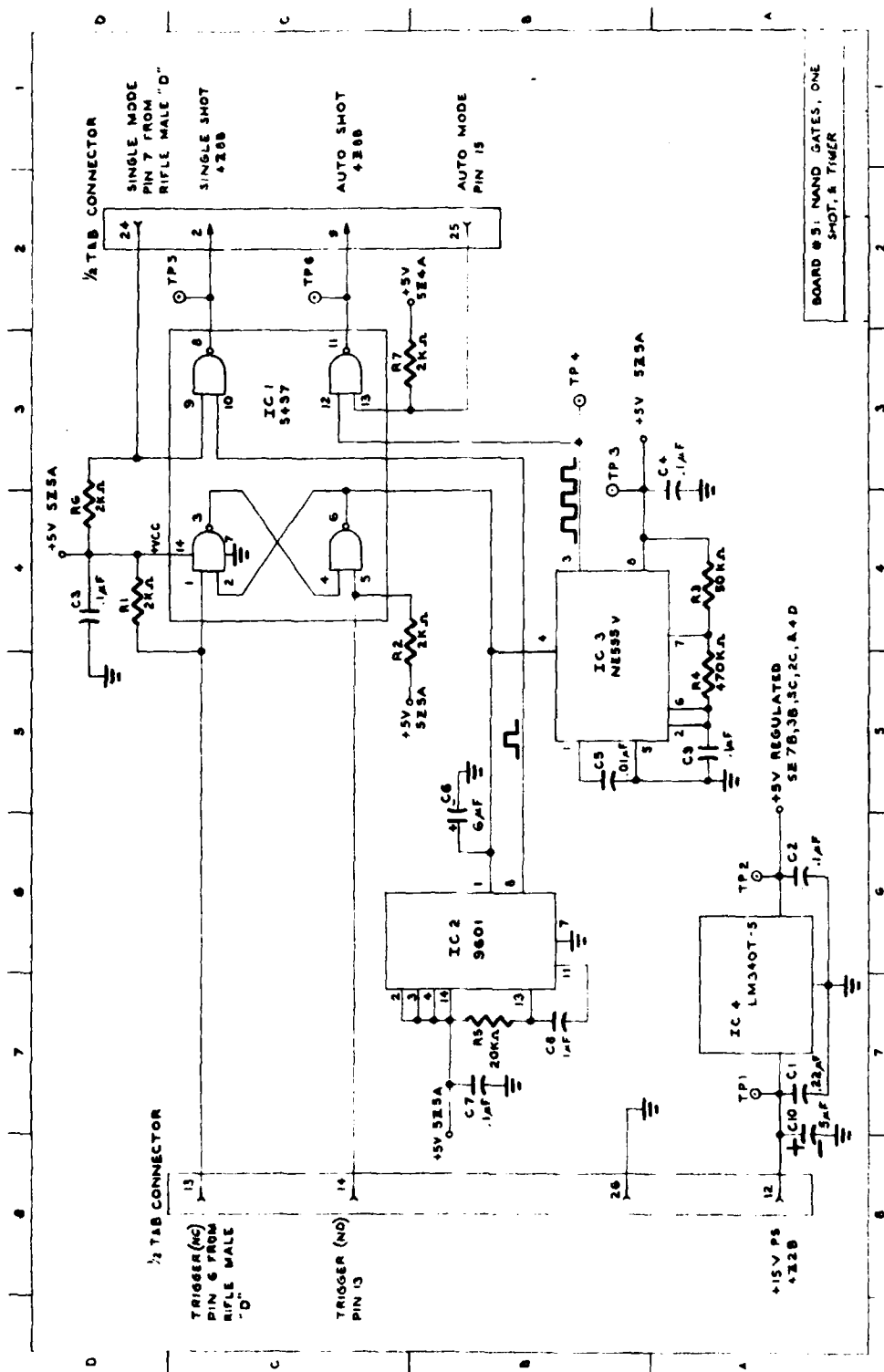


Figure III-9. NAND Gates, One-Shot and Timer - Board #5

Each trainee's audio system consists of two stages of audio amplification. A Texas Instrument's TL074 low noise, quad, dual operational amplifier, IC1, is used. Consequently, two trainees are handled by a single TL074 (Figure III-10). The first stage of amplification is primarily an audio mixer. Five independent channels are mixed into one. These five channels consist of the computer voice feedback system, the instructor communication line, the synthetic rifle bang, coordinated battlefield sounds and general battlefield environment sounds. The instructor uses an identical mixer/amplifier channel but his inputs consist of the various computer voice responses to the trainees. The instructor selects which trainee he desires to hear by pushing the appropriate switch on the instructor control panel.

Each of the five inputs to the mixer stage as well as the final output stage have their own volume control.

D. BANG AND RECOIL SYSTEM

1. BANG SYSTEM

An electronic bang is presented to the trainee via his headset when he has fired a shot. The bang board, Board #8, Figure III-11, produces the synthetic gunshot sound and passes this sound to the trainees audio mixer/amplifier Board #6, Figure III-10. The bang is produced by generating random noise, due to diode D1 being biased near its breakdown voltage, and then using the FET to generate an envelope for this random noise. This envelope consists of a sharp rise time and an exponential decay which corresponds closely to a gun shot noise envelope. Specifically, the diode D1 produces random noise which is amplified by 1/2 of IC1, a dual operational amplifier. This amplified random noise is presented to the drain of the FET. The FET does not pass this noise until its gate is presented the sharp rise and exponential decay envelop representing an actual rifle shot sound envelope. The sharp rise of voltage on the gate of the FET is produced by IC2 changing to a high state; 5 volts. When IC2 changes back to a low state, 0 volts, the diode D2 isolates the gate of the FET from being pulled down to an off state and allows the RC network consisting of R6, R12, and C7 to exponentially decay the residual voltage thus producing decaying gunshot envelope of noise. The source of the FET thus produces on demand random filtered noise within an envelope resembling a gunshot bang. The second half of IC1, an operational amplifier, produces final amplification of this sound before passing the output to the students audio mixer/amplifier.

2. RECOIL SYSTEM

The recoil system consists of three major parts: air hose, recoil board, and the air valve.

The air hose follows the electrical cable up to the rifle and into the butt of the weapon. The hose is a lightweight, nylon reinforced, dimensionally stable air line hose. After entering the butt of the rifle it runs forward and attaches to the rifle barrel. The barrel is plugged at the tip end and an outlet orifice has been drilled on the bottom of the barrel near the tip end. The orifice is pointing down and 30 degrees to the left which produces a thrust up and to the right when a shot is fired.

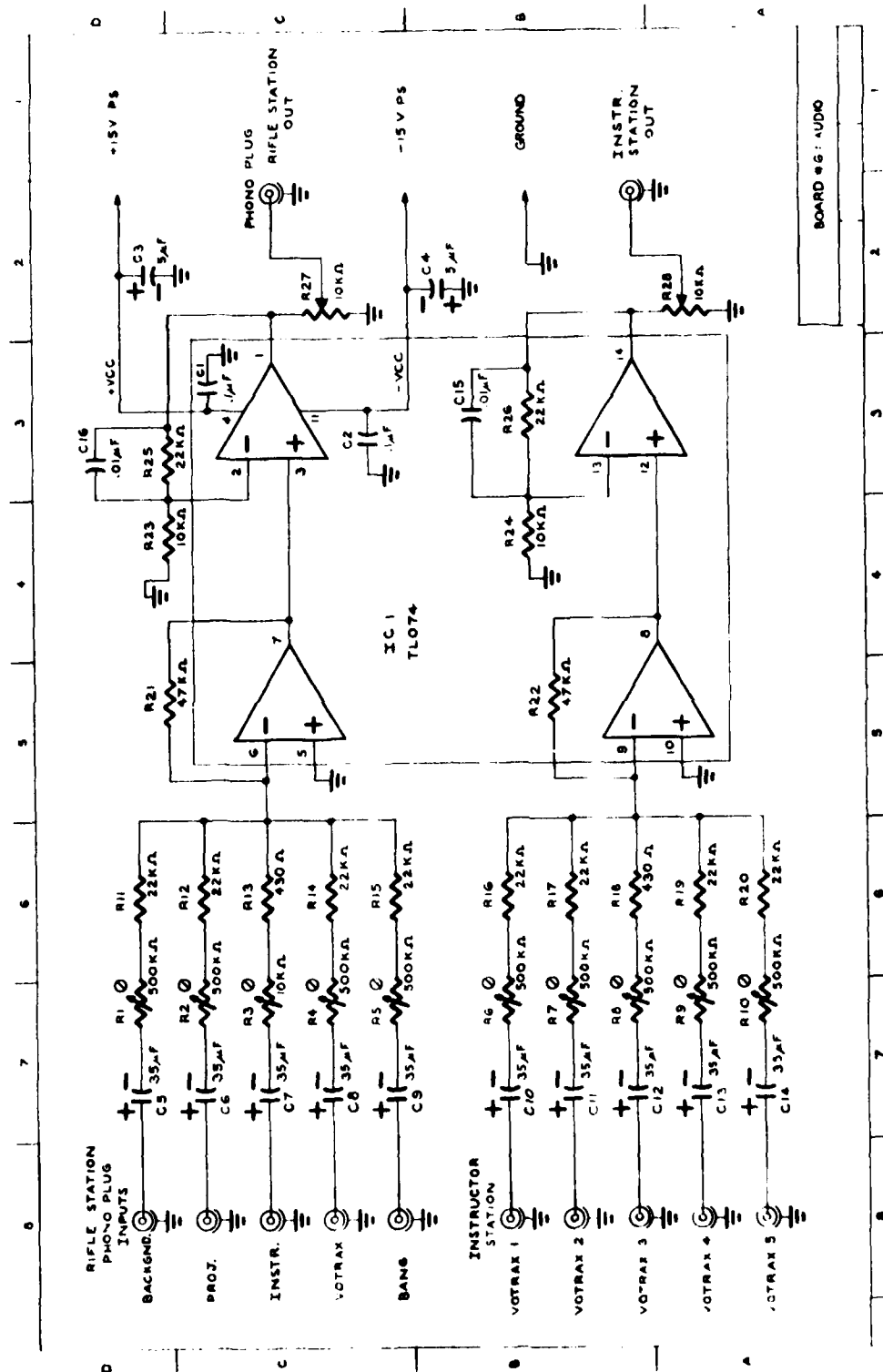


Figure III-10. Audio Amplifier - Board #6

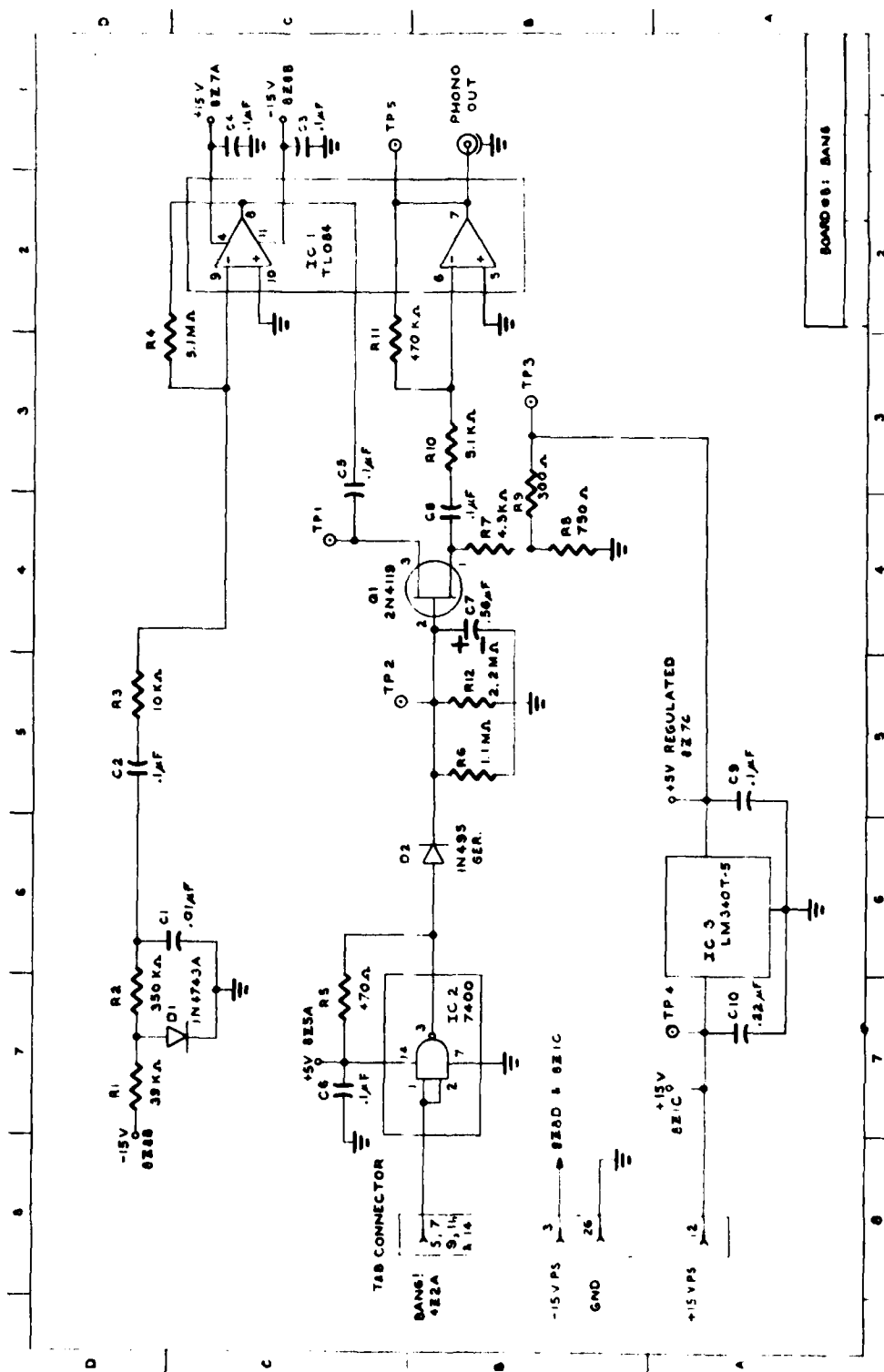


Figure III-11. Bang Simulation Generator - Board #8

The recoil circuit, Board #7, consists of a 555 integrated circuit timer, IC2, and a darlington pair transistor driver circuit for the recoil air valve. The 555 timer is set for a nominal 20-25 msec duration. The variable resistor R2 serves to regulate the timing duration (Figure III-12).

The recoil valve is a pilot operated solenoid valve. Because it is pilot operated, the on-off rise and fall times for actuation are very short and power consumption is only 8.5 watts.

E. DISTRIBUTION OF FIRE AND WEAPON MOVEMENT MONITORING

Distribution of fire and weapon movement can be monitored and recorded during a training exercise for playback. The system allows the instructor to view where the weapon is aimed relative to the IR target spot. This feature is completely independent of the basic system.

An IR light source is used on the weapon. The infrared light source used in the system is a semiconductor, gallium arsenide laser. The laser is collimated by a simple plano convex lens. The laser is attached where the weapon flash hider is located. If the instructor wishes to view the location of the trainee's weapon, he selects the laser he wants turned on and holds down a button on the instructor's console. The instructor is able to view both the projector IR target and laser spot from the selected trainee's rifle. This information is detected using an RCA TC 1005/H01 low bloom silicon target Vidicon and closed circuit video equipment. The TV display tube is located in the instructor's console and the TV camera near the motion picture projectors.

The laser spot brightness seen on the TV is a function of the pulse repetition frequency (prf) of the gallium arsenide laser. Two modes are available:

- Flash only
- Track plus flash

In the flash mode only, a single flash occurs when the trainee fires. In the track and flash mode, the instructor sees a point of laser light on the screen all the time, which moves as a function of where the trainee is pointing; when the trainee fires, a brighter flash occurs.

Laser energy reflected off the screen is eye safe. However, the trainee should not point his weapon in another trainee's eyes as eye damage can occur from looking directly into the laser beam.

The laser timing signals are generated using Board #10, Figure III-13. The laser pulser, Board #11, is shown in Figure III-14.

The laser pulser uses a SCR, GA201 to discharge capacitor C1. Q1 is used to allow rapid recharge of Q1. The laser is a 5 watt peak power laser with a nominal 50 nanosecond pulse width.

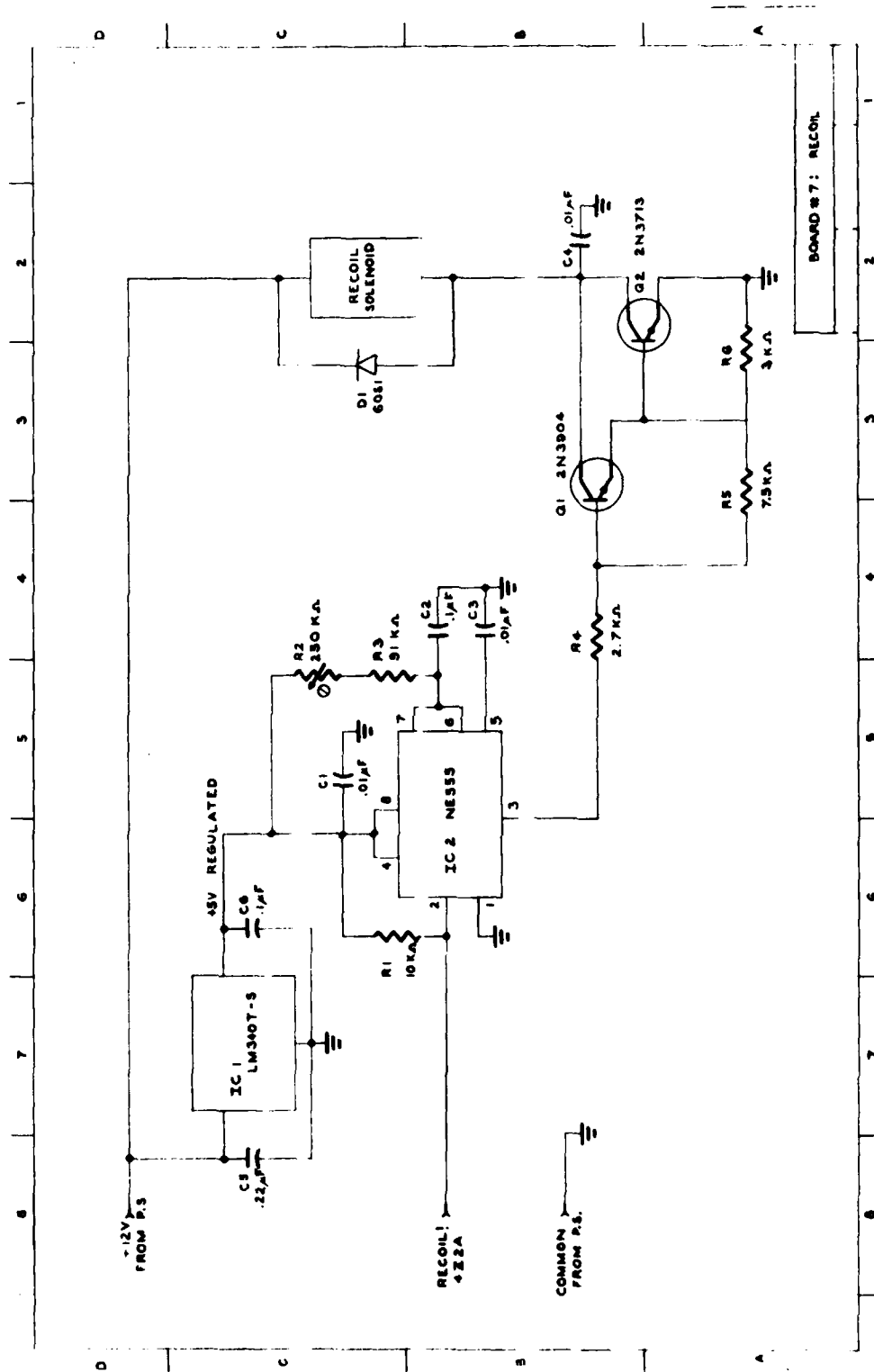


Figure III-12. Recoil Circuit - Board #7

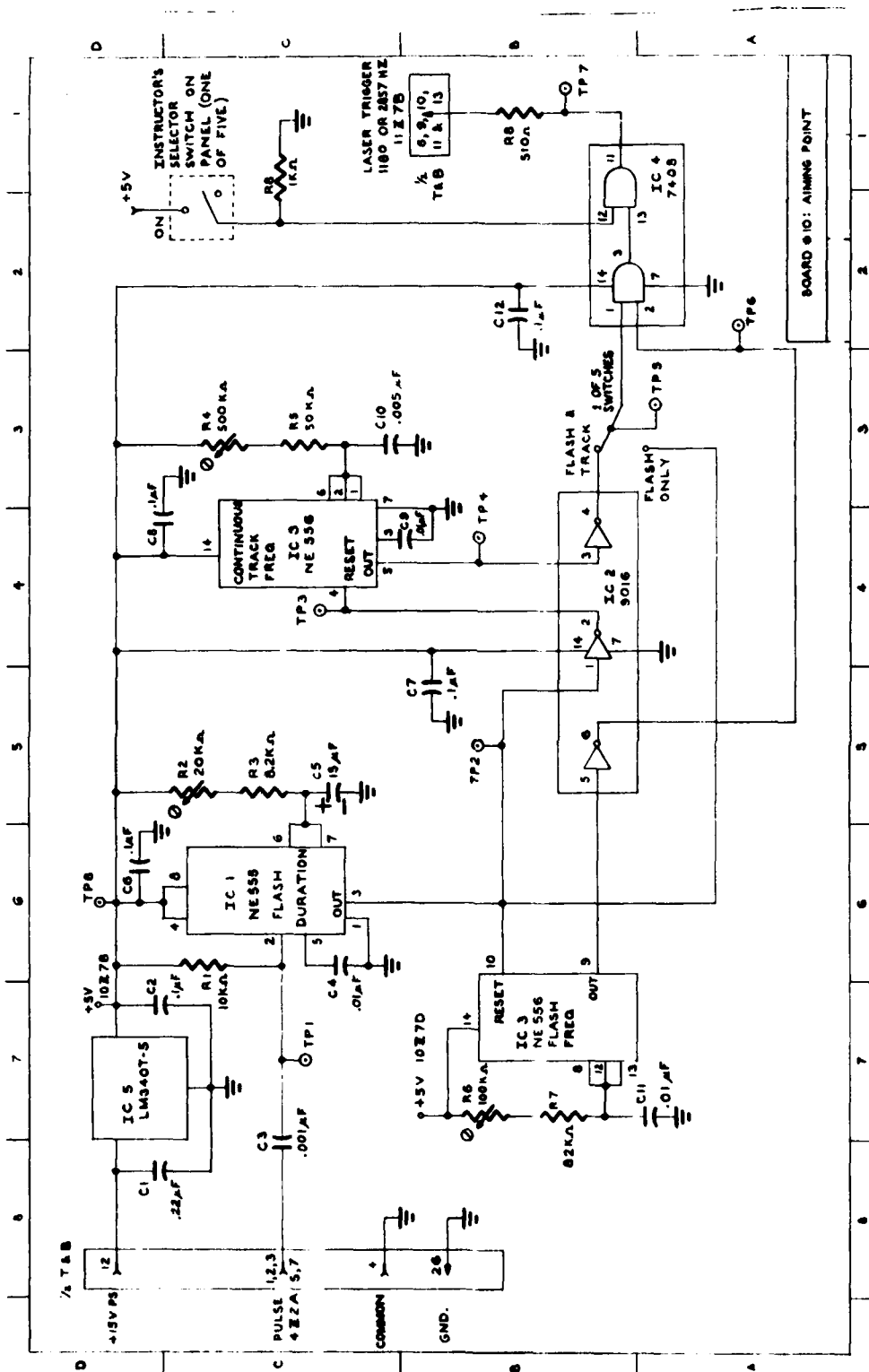


Figure III-13. Laser Signal Generation - Board #10

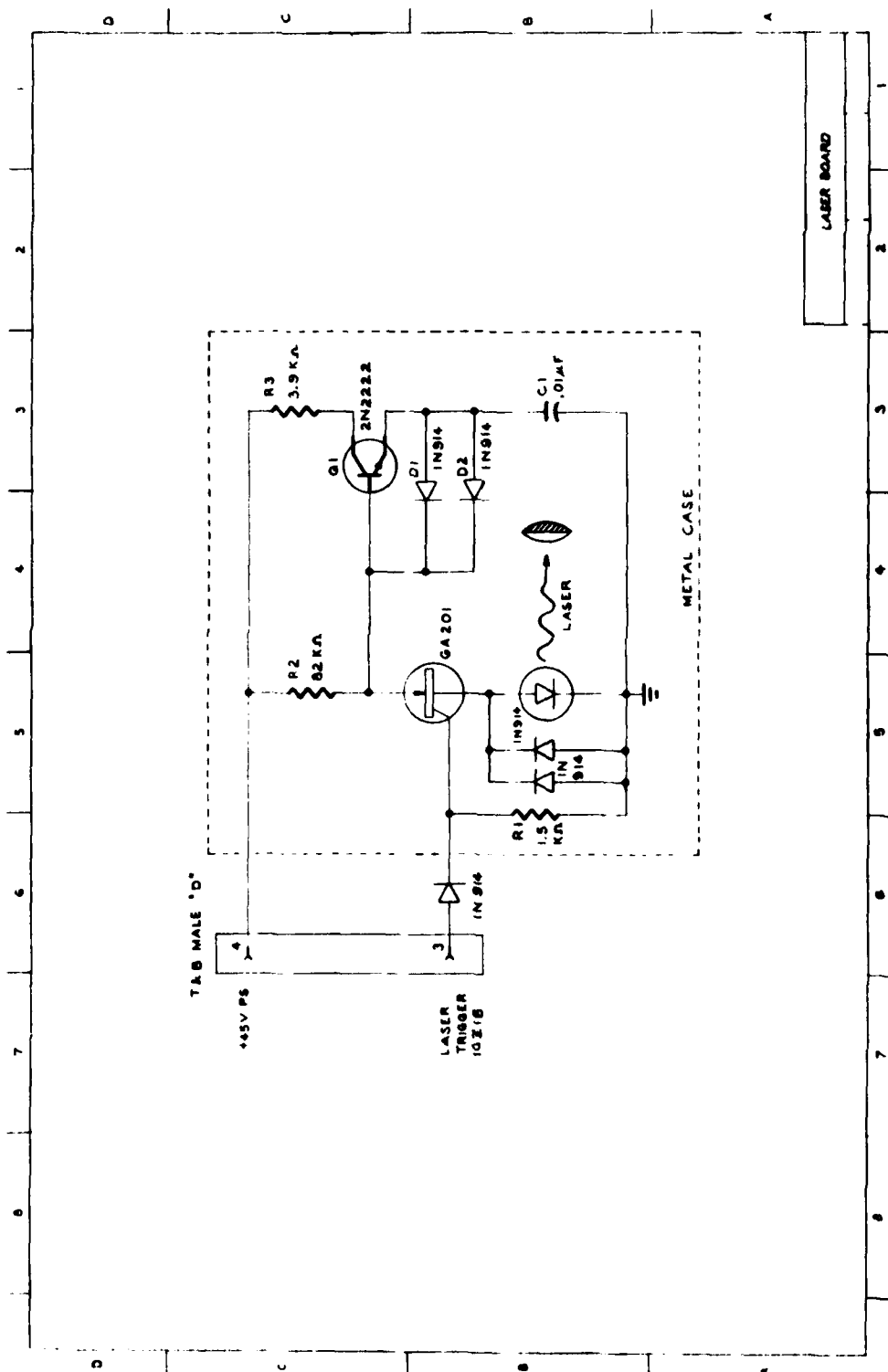


Figure III-14. Laser Pulser

F. RIFLE MOCKUP

The rifle mockup is manufactured by Replica Models, Inc. It is not designed to accept a round of ammunition and cannot be converted to accept ammunition. The original replicas received from Replica Models have been extensively modified to perform satisfactorily as a trainer. The original barrel plug was removed and moved to the front end of the barrel to accommodate the recoil. A recoil orifice was then machined and an electronic board was installed within the handguards. To accommodate boresighting, the original molded-on nonadjustable front and rear sights were replaced with adjustable front and rear sights. The mode selector switch was modified to reflect the real M16E1 mode positions; the trigger mechanism was modified for better performance; microswitches for the trigger and mode selector switch were installed; and magazine sensing contacts were installed for reloading simulation.

An optical four-quadrant detector and optics are mounted above the barrel and below the sights. A solid state laser and optics for point of aim information has been inserted in the flash hider position.

Air for the recoil and electronic wiring approach the rifle from the bottom rear of the butt of the rifle. The true weight of the M16E1 was restored by removing unused mechanism from the upper receiver. The true balance was maintained through equal weight additions, i.e., the detector/laser combination at the front end of the rifle offset the hose and electronic wire harness at the butt end of the rifle.

Special test equipment is included in Appendix B.

G. MICROCOMPUTER CONTROL SYSTEM

The 8080 Microprocessor Based Control System performs these functions:

- Interrogates the instructor for session parameters
- Stores session parameters for final hard copy
- Determines if self-check is desired, and reacts accordingly
- Initializes peripheral LSI chips and zeros memory storage
- Inputs rifle data, decodes and stores it
- Measures response time for first rifle shot at new target for each of four or five rifles
- Outputs shot results to audio feedback and instructor's CRT
- Identifies shooter making most errors and sends the identification to the instructor's console "LEDS"
- Updates shooter's results file

- Checks for session end and terminates the data collection mode upon the instructor's signal
- Computes trainee's overall score
- Prints trainee's results on the instructor's electronic data terminal

1. SINGLE BOARD COMPUTER

The SWAT System is controlled by a modified INTEL 80/20-4 Microcomputer System, Reference 6. This microcomputer system, which is based on the INTEL 8080 microprocessor, includes an enclosure with front panel controls, power supply, cooling fans, and a card cage in which is located the main 80/20-4 board as well as the interface board (IFB), which is described below.

a. 80/20-4 MODIFICATIONS

A number of modifications are required before the SBC 80/20-4 can be used in UIWT/SWAT Version 1.2. These are detailed below with page identifications to be found in Reference 6 unless otherwise noted.

(1) Pull-up resistor packs, SBC-902, page 2-5, must be inserted in socket A5 and A6 as input terminators for port 2 at address E6. These terminators were supplied with the 80/20-4 systems as Beckman part number 1899-747-0, 3000645-01.

(2) Insert inverting line drivers, either #7437 or #7400, in sockets A3, A4, A9 and A10 for output ports 3 and 6 at addresses E6 and EA. See pages 2-4 and 4-24.

(3) Solder a jumper between J3-8 and J3-10 using the solder points on the rear side of the board. This connects "Request to Send" to "Clear to Send". See Table 2-5, page 2-7.

(4) Wire wrap A jumper from pin 1, a 5 volt source, near J3 pin #25 and solder it to a through hole just below "C9" between A15 and A16 on the front of the board. This should put 5 volts onto J3-16 "REC LINE SIG DETECT" which goes to the "DATA CARRIER DETECT" of the 743 TI terminal. Otherwise the terminal will not function. See page 27 of Reference 1.

(5) Change the wire wrap jumper which exists between pins 141-142 just above A22, the 8253, to a jumper between pins 141-143. This is an option which connects the clock input for counter 1 of the 8253 to the output of counter 0. See page 4-21.

(6) Interconnect wire wrap pins 11 and 12 near the upper right hand corner of the board. These are the protect and signal grounds for the TI 743 terminal. See page 27 of Reference 1.

(7) Scratch through the line going from the transmit data (TXD), pin 19 of the 8251 USART, to the SN75188 (MC1488) line driver. Connect the 8251 side to J1-50 with a jumper wire and connect the driver side to J2-50. This allows the interface board to switch the serial output between the VOTRAX and the control console.

(8) Remove the jumper from wire wrap pins 52-53, located just below the left part of the leftmost 8255 and put a jumper between pins 51-52. This enables port 1 as an input. See Figure 5-2 (sheet 4 of 5). Check that a jumper exists between pins 71-72 to enable port 4 as an output.

(9) Connect the 80/20-4 front panel interrupt switch into the interrupt controller as interrupt #7. To do this, connect pins 36, 37, 38, and 39 together and also to pin #45.

(10) Make the required modifications to use 2716 2K byte EPROMS. These are given in Table 2-12, which is entitled "Jumper Changes For Optional 8K EPROM Installation". See page 2-15.

<u>REMOVE</u>	<u>INSTALL</u>	
W2, A-C	W2, A-B	Between A45 and A46
W4, B-D	W4, A-D	Above A78
W4, C-E	W4, B-E	Above A78
W7, A-B	W7, A-D	Below A79
W8, A-C	W8, A-B	Below A79
C35, 53 and 72		Above A37 and Below A64 and A79

2. THE INTERFACE BOARD

All input/output (I/O) operations of the SBC 80/20-4 microcomputer pass through the interface board (IFB). These operations can be divided into three categories.

- Rifle communications
- I/O through the 8751 "USART"
- Output through the UPI-41, 8741 Universal Peripheral Interface

Figure III-15 is a block diagram of these data paths and their associated control lines. More details are shown on Figure III-16 through III-19.

a. RIFLE COMMUNICATIONS

IR spot quadrant detector data are input from each rifle to a separate 8212 eight-bit input/output port chip on the IFB. A trigger-pull signal is also sent from each rifle to its associated 8212. Upon sensing a trigger signal, the quadrant data are latched into the 8212 buffer and an interrupt signal requesting service is output from the 8212 to the main board through input port 1. The service request lines from all five 8212s are "ORED" together onto a single line which also goes to port 1 to signal that at least one 8212 requires service. AS long as this ORED line indicates a service need, the microcomputer polls each 8212 service request line in turn. When one is detected that needs service, the address of the 8212 responsible for the request is output from port 3 on the main board to a 9311 one-of-sixteen decoder on the IFB. A data read signal is then output from port 6 to the 9311, which commands the 8212 to place the contents of its latched buffer on the common data bus.

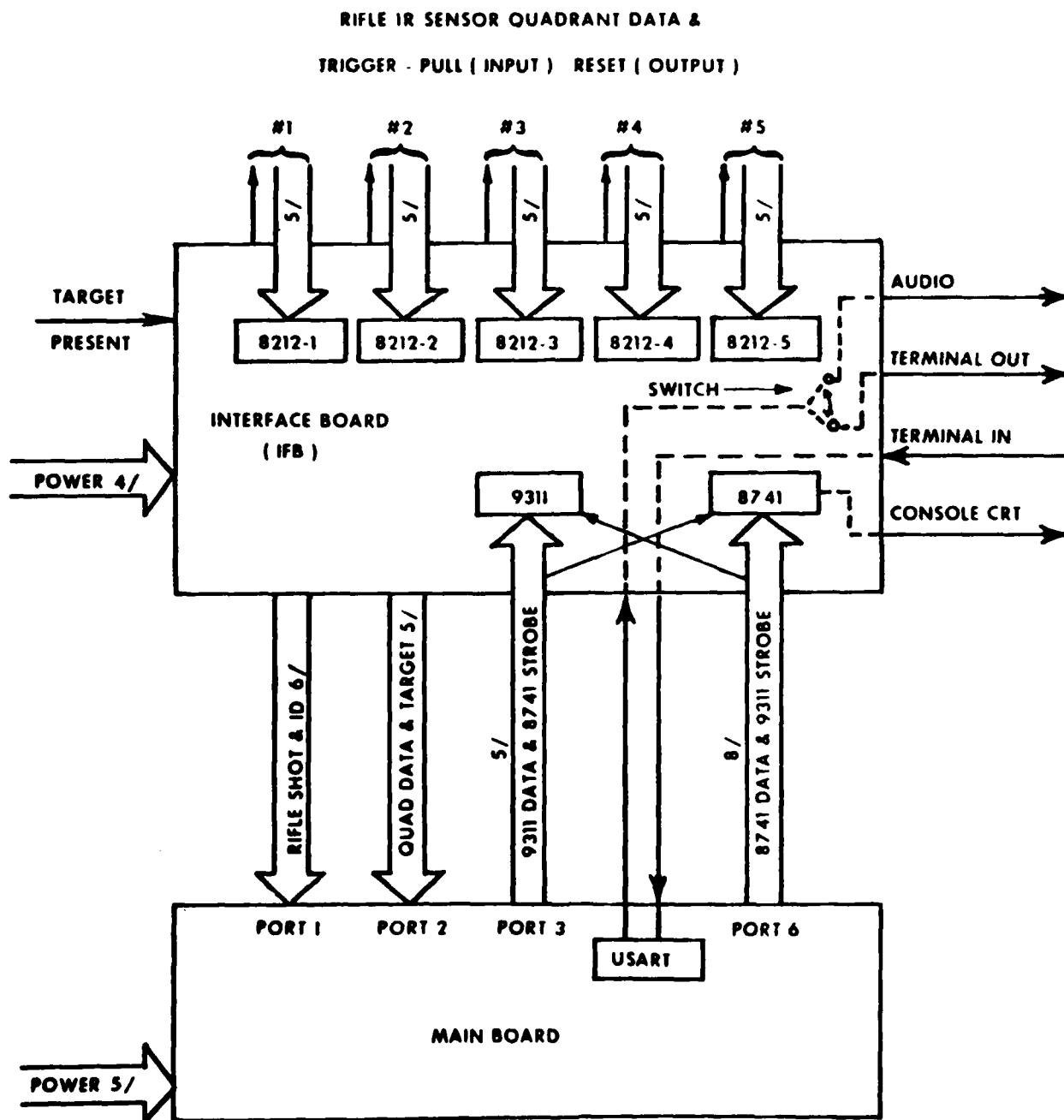


Figure III-15. 80/20-4 Main and Interface Boards

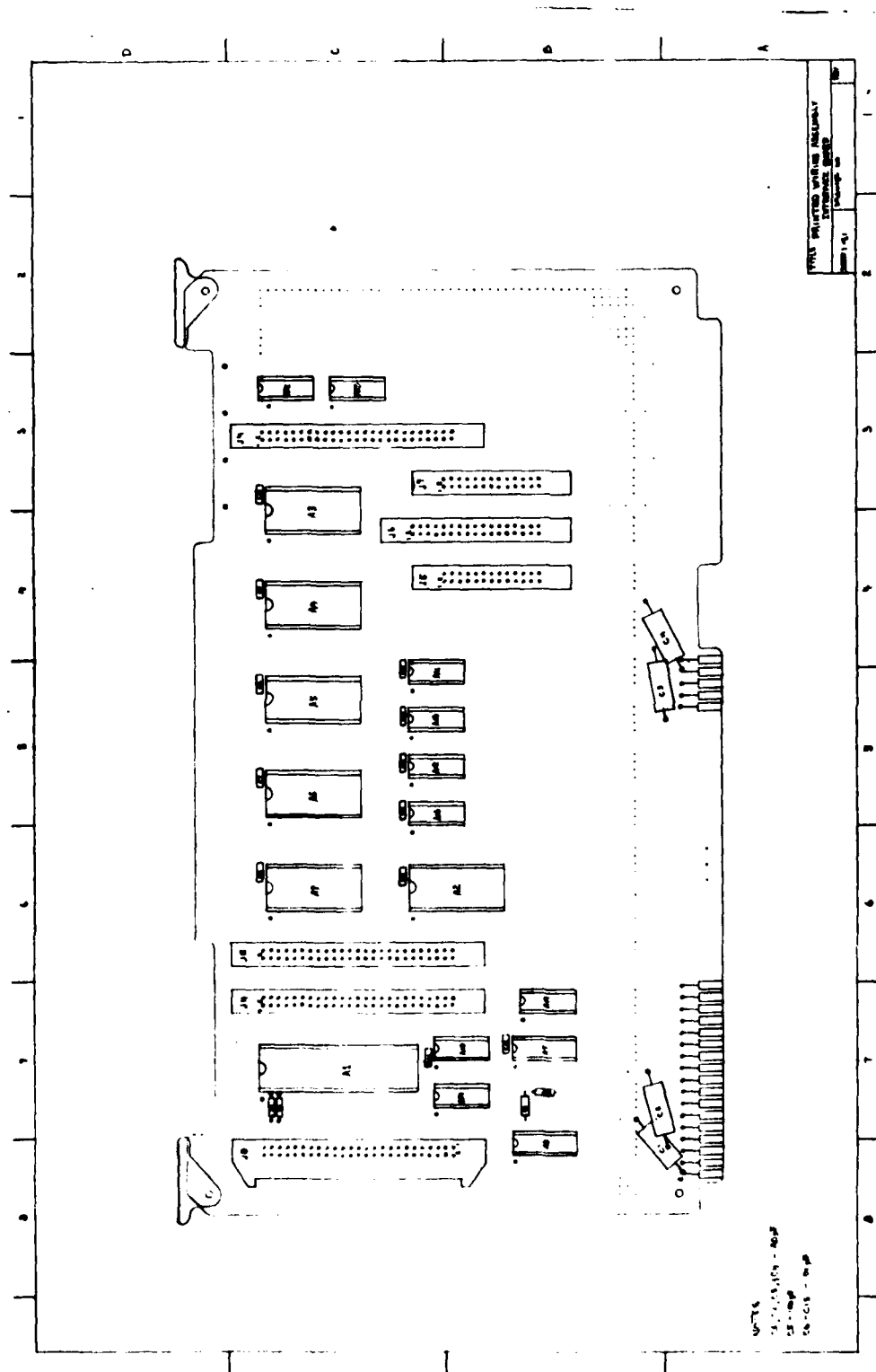


Figure III-16. Interface Board Layout

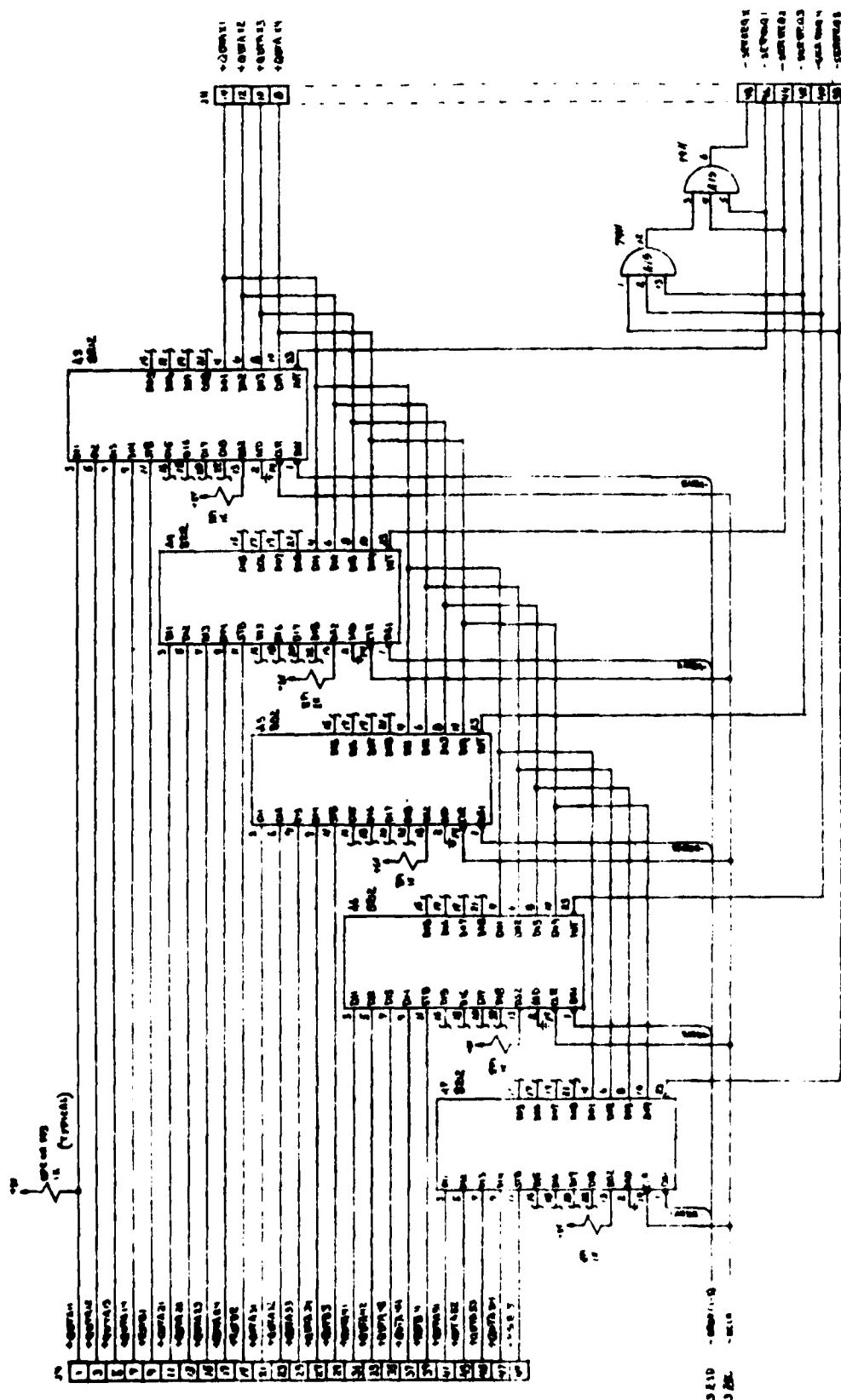


Figure III-17. Interface Board Schematic (1 of 3)

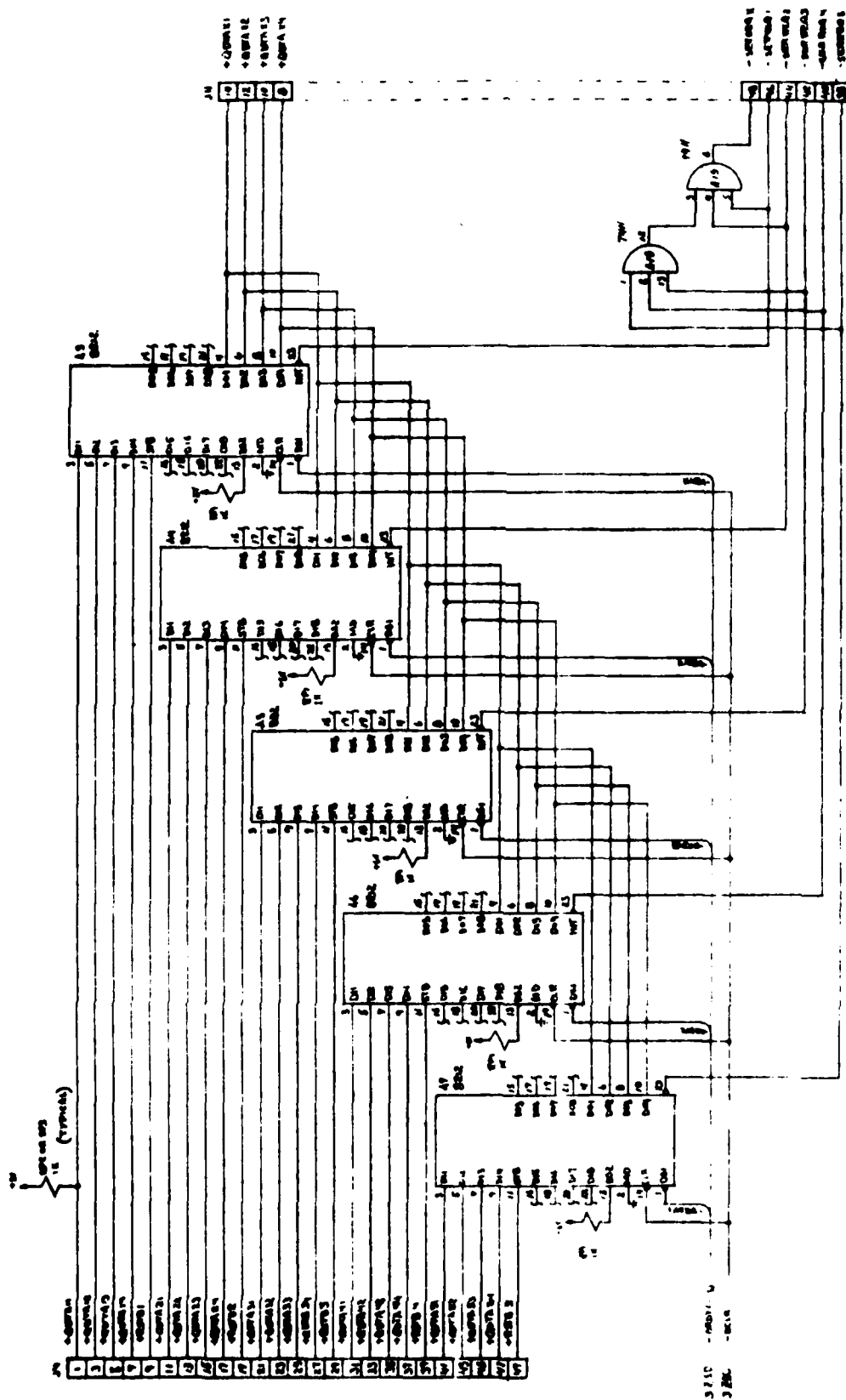


Figure III-18. Interface Board Schematic (2 of 3)

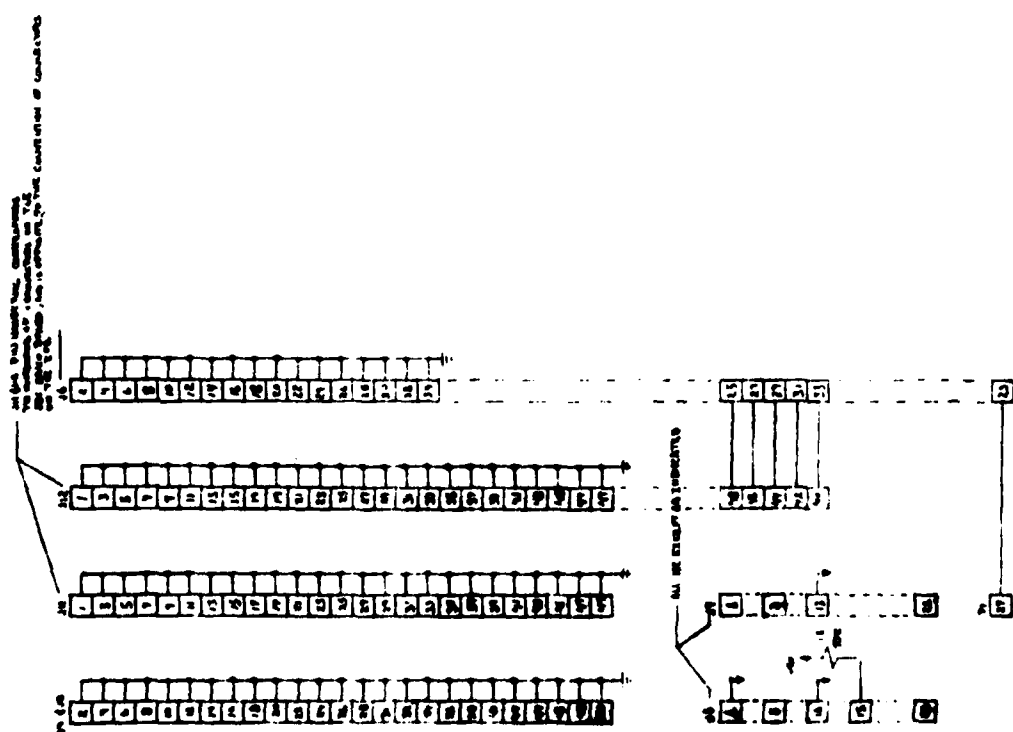
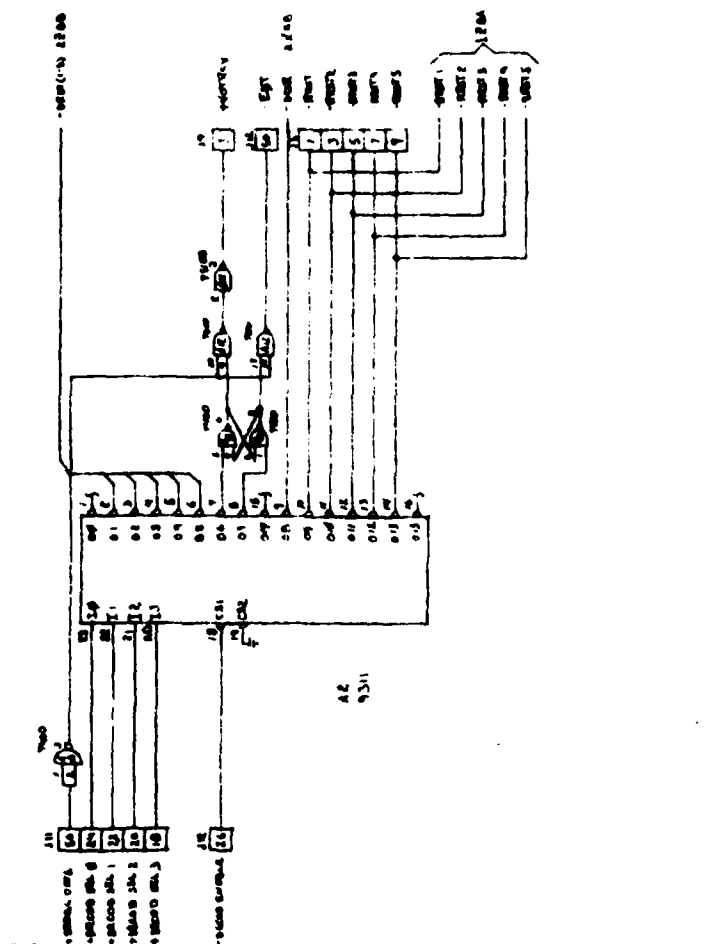


Figure III-19. Interface Board Schematic (3 of 3)

Each of the other four unaffected 8212 chips may also contain data, but are held temporarily in an inactive "three-state" and present a high impedance load to the bus. The only quadrant data available at the main board port 2, therefore, are from the 8212 being serviced. These data are read into memory and the data read signal to the 9311 is removed. This removes the read command to the 8212 and clears its interrupt service request. A pulse is then sent from port 6 to the 9311 which issues a reset signal from the IFB to the rifle electronics associated with the serviced 8212.

The serviced 8212 is now in three state, its service request line is off and it is ready to latch in new data upon receiving the next trigger-pull signal. In the meantime, if other 8212 chips need service as indicated by assertion of the ORED line, the computer polls the next 8212 interrupt line. If it needs service, the process is repeated; if not, the next 8212 service line is polled in sequence. This continues until the ORED service line goes off and the computer moves ahead with the remainder of the program.

A "Target Present" signal from the IR spot projector is carried directly through the interface board to the main board through input port 2. The target present information is recorded and used during scoring to identify a valid target.

b. USART I/O

The control terminal is an electronic data terminal operating at a rate of 300 bits per second, Reference 1. At the initiation of each training session, the computer connects the output serial data stream from the 8251 programmable communication interface or Universal Synchronous/Asynchronous Receiver/Transmitter (USART) to the terminal. The computer, therefore, is able to carry on a two-way conversation with the squad leader in order to obtain "initialization" data as shown on Figure 6. The computer questions the squad leader and prompts for answers by issuing the character ">".

During the actual training session, the USART output is switched to the digitized word audio system. When the session is finished, the squad leader strikes/presses the start/print button on the instructor's console and USART output is again directed to the terminal which types out hard copy scores, as also shown on Figure III-20.

c. UPI-41 MICROCOMPUTER OUTPUT

During a training session, console CRT data are output in parallel from port 6 of the 80/20-4 single board computer to an 8741 Universal Peripheral Interface Slave Microcomputer (UPI-41) on the IFB. The UPI-41 decodes the parallel data and sends a 19,200 BAUD, 7 bit ASCII data stream to the console CRT. The console CRT translates the serial data stream into a score message and displays the message in the column reserved for the appropriate rifle. The UPI-41 also monitors the setting of 5 control switches, one for each rifle which allows the squad leader to inhibit the display of scores for any or all rifles.

WANT ID YES OR NO
NO
LET'S START

"INITIALIZE" PORTION OF TRAINING
SESSION

RIFLE: 1

YOUR RESULTS ARE:

TOTAL SHOTS: 99
HITS: 16
MISSES: 29
LOWS: 2
LOW RIGHTS:
RIGHTS: 6
HIGH RIGHTS: 3
HIGHS: 4
HIGH LEFTS: 8
LEFTS: 22
LOW LEFTS:
NO TARGET: 9
TARGETS IGNORED: 8
TARGETS SHOT AT: 30
AVERAGE TIME: 1.2 SECONDS

OH WELL: THERE'S HOPE IF YOU SPEED UP
YOUR OVERALL SCORE IS: 37

SESSION PROPER.
NO OUTPUT TO TERMINAL . OUTPUT
IS VIA VOTRAX DIGITIZED AUDIO
WORDS & CONSOLE CRT. THIS
PHASE IS TERMINATED BY AN
INTERRUPT FROM TERMINAL.

"PRESENTATION OF RESULTS"

Figure III-20. Typical Printout Format on Terminal
(Continues for all 5 rifles)

The UPI-41 system description is divided into four parts: part 1, a functional summary and a component interface description are presented. Performance criteria are also established in this section. Part 2 describes the facilities available within the UPI-41 and explains their use in the present application. Part 3 describes the UPI-41 control program and part 4 evaluates the system with respect to assumption validity, performance criteria, and maximum system capabilities. The source program is given in Appendix D.

d. UPI-41 MICROCOMPUTER OUTPUT II

(1) System Description

The function of the intelligent controller is to receive parallel data from the SBC 80/20, decode the data, and cause a message to appear on the ADM-3A screen based upon the content of the data received. The control switch settings also affect controller operation, but only secondarily.

A block diagram showing the system component relationships appears in Figure III-21.

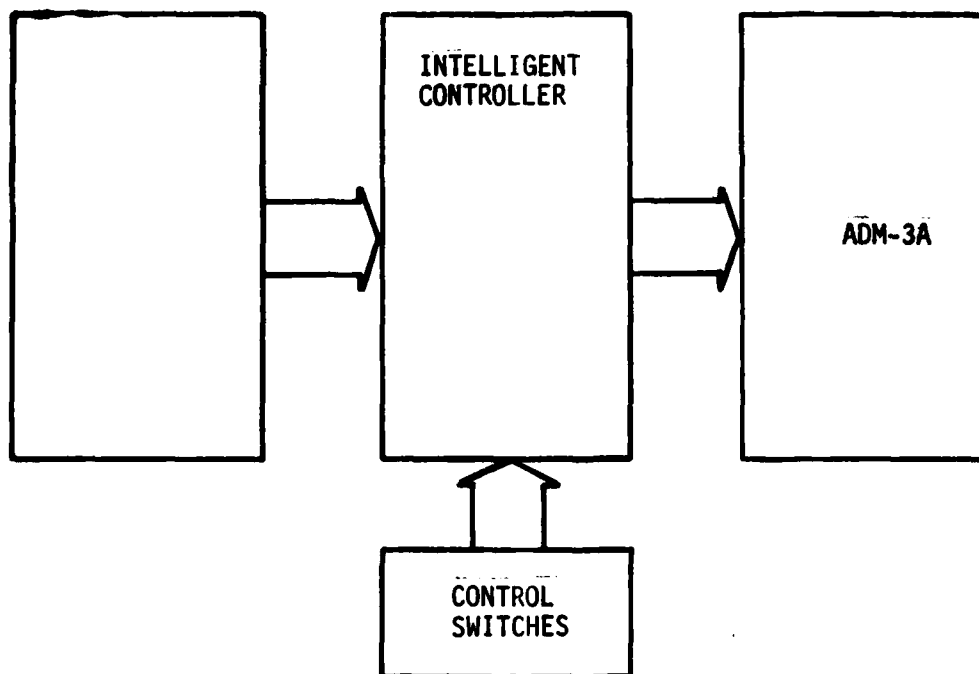


Figure III-21. UPI-41 Intelligent Controller System Block Diagram

The following describes the three component interfaces shown in the figures: SBC 80/20 to controller, controller's to ADM-3A, and control switches to controller.

(a) SBC 80/20 To Controller Interface

The SBC 80/20 to controller interface is comprised of three sets of connections. The first set, consisting of 8 data lines and 1 control line, are the data transfer connections. The second set consists of the clock connections while the third set consists of only one connection, the initialization connection.

1. Data Transfer Connections

The 8 data lines of the data transfer set connect an 8 bit output port on the SBC 80/20 to the 8 bit Interface Register of the UPI-41. There are six I/O ports on the SBC 80/20 numbered 1 through 6 (1). These ports are divided into the Group A ports, 1-3, and the Group B ports, 4-6. Each port group corresponds to a single 8255 Programmable Peripheral Interface, PPI. Port 4 of Group B is programmed as an output port and used for the SBC 80/20 to UPI-41 data connection.

To transmit data to the UPI-41, the SBC 80/20 places data on port 4 and sends a Data-Available pulse to the UPI-41 over the control line. The Data Available pulse is software generated and is transmitted through port 3 of Group A 8255. The length of the Data Available pulse is set by the time required to execute the instructions necessary to change the logic level of the control line twice, first from high to low, then from low to high. For the SBC 80/20 this results in a 10 microsecond pulse. The maximum pulse length to the UPI-41 is set at twice the instruction cycle length, or 6.5 microseconds; therefore, the 10 microsecond Data Available pulse is sent to the one shot within the controller where it is shortened to 1 microsecond. The 1 microsecond pulse from the one shot supplies the WR input to the UPI-41. On the rising edge of this pulse the data on the SBC 80/20 output port is latched into the UPI-41 Interface Register. SBC 80/20 to controller data transfer connections are illustrated in Figure III-22.

Each byte of data transferred from the SBC 80/20 to the UPI-41 contains two kinds of information encoded into separate fields within the byte. The three most significant bits contain a source identifier encoded in straight binary, and the four least significant bits contain a message identifier, also in straight binary, see Figure III-23. Bit four is not used.

The rate of data transfer from the SBC 80/20 to the controller can be characterized by three separate data transfer rates of which the last two will be of interest. The first two rates are determined by the SBC 80/20 input configuration, Figure III-24, while the third is determined by the input configuration in combination with the SBC 80/20 data processing rate.

The SBC 80/20 input configuration consists of 5 input sources, where each source contains a data latch and a service request line. When data is latched into one of the sources, the SBC 80/20 receives a service request signal from that source. For each service request that the SBC 80/20 responds to, a data byte will be sent to the controller.

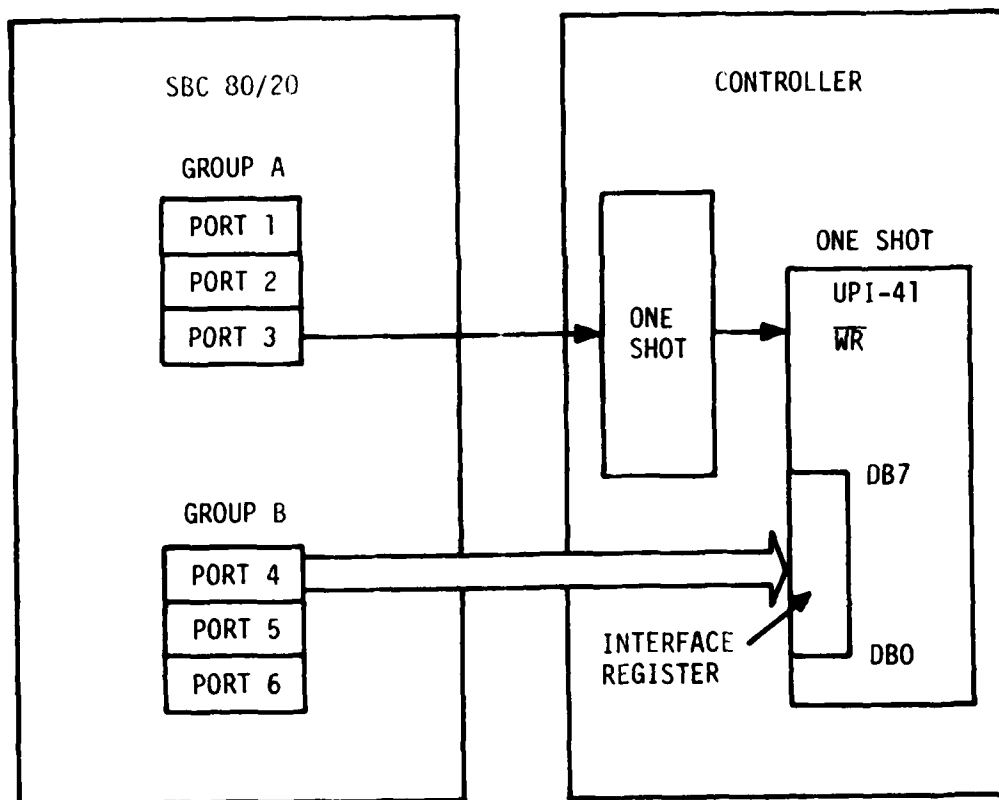


Figure III-22. SBC 80/20 to Controller

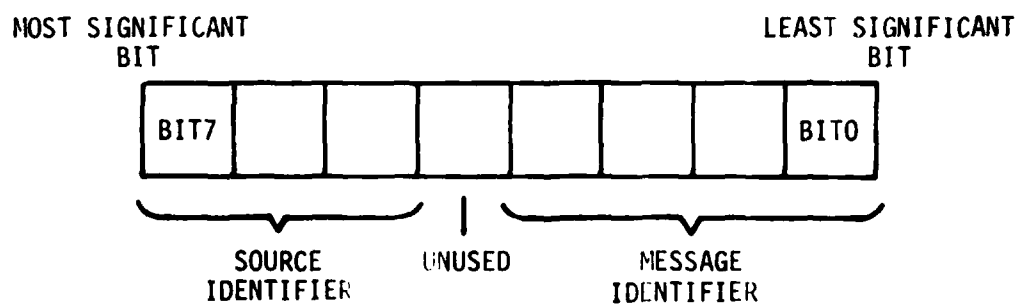


Figure III-23. SBC 80/20 to UPI-41 Data BYTE

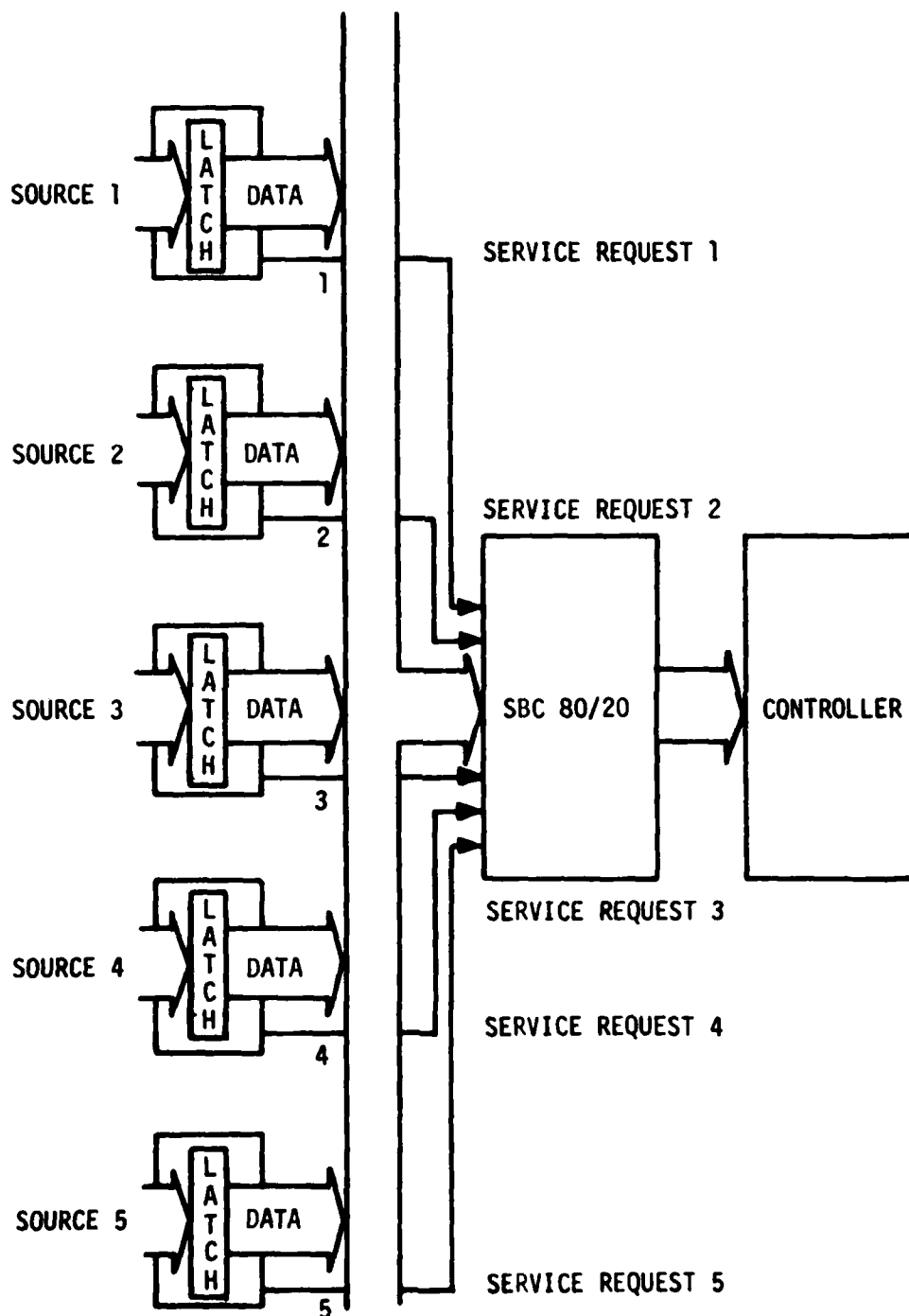


Figure III-24. SBC 80/20 Input Source Configuration

The first data transfer rate is the average transfer rate and occurs when the 5 sources are initiating service requests at their nominal rate. The second data transfer rate is a peak average rate, and occurs when all 5 sources are initiating service requests at their maximum rate of 12 per second. This condition results in a peak average rate of 12×5 , or 60 transfers per second. The third data transfer rate is the maximum rate, and occurs anytime there are simultaneous service requests to the SBC 80/20. This rate is determined by the processing rate of the SBC 80/20. Analysis using: (1) real-time emulation under control of Intel's In Circuit Emulator, ICE-80, (2) tabulation of instructions executed and their execution time and (3) experimental determination, indicates that the SBC 80/20 processing rate is approximately 200 inputs per second.

As indicated before, the peak average transfer rate of 60 transfers per second, and the maximum transfer rate of 200 transfers per second are the relevant quantities characterizing the data transfer interface.

To keep up with the SBC 80/20 over extended periods, the processing rate of the UPI-41 must equal or exceed the SBC 80/20 peak average transfer rate, and to keep up with the SBC 80/20 when simultaneous service requests have occurred, the reception rate of the UPI-41 must equal or exceed the SBC 80/20 maximum transfer rate.

The requirement on the UPI-41 processing rate will be used in the sequel to determine the baud rate used in the controller to ADM-3A interface, while the requirement on the UPI-41 reception rate will be used to establish the necessity of a data queue within the UPI-41.

One final point is that there are no provisions for the UPI-41 to indicate that it is ready to accept a data transfer from the SBC 80/20. Thus, the data queue mentioned above will be filled by an interrupt driven procedure. This technique will assure that a data byte has been removed from the Interface Register before an additional data transfer can occur.

2. Clock Connections

The clock connections supply the UPI-41 clock inputs, X1 and X2. A single line from the SBC 80/20 supplies the controller with a 9.216 megahertz clock which the SBC 80/20 makes available as the BCLK output. Within the controller, the BCLK frequency is divided in half by a 7474D flip flop. This division is necessary to bring the BCLK frequency within the 1 to 6 megahertz operating range of the UPI-41. The Q and Q₀ outputs of this flip flop supply the UPI-41 inputs, X1 and X2, with a 180° out of phase 4.608 megahertz clock. While the UPI-41 is capable of generating its own clock by connecting a crystal to the X1 and X2 inputs, the BCLK frequency is used since the standard asynchronous communication frequencies can be derived from it. The clock connections are shown in Figure III-25.

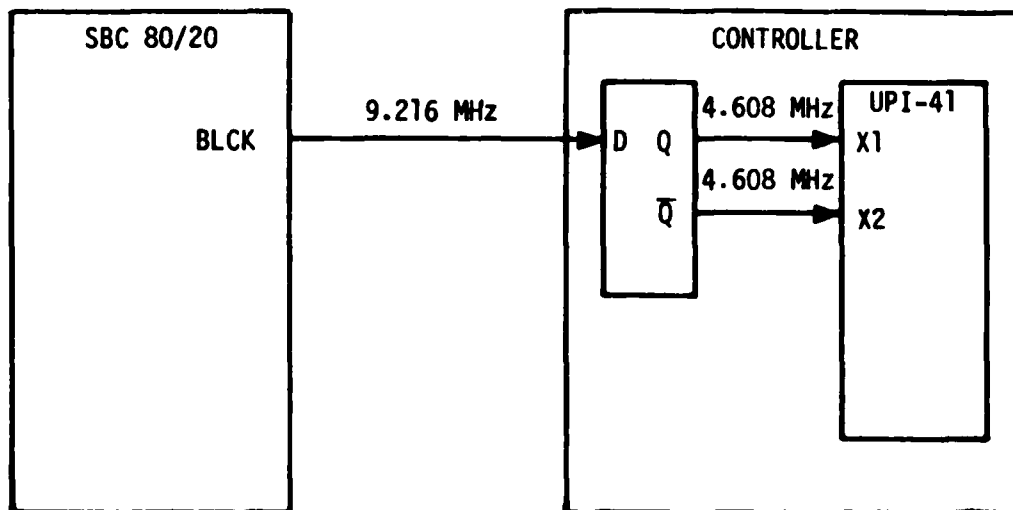


Figure III-25. SBC 80/20 to Controller - Clock Connections

3. Initialization Connection

The initialization connection is between INIT output of the SBC 80/20 and the RESET input of the UPI-41. A low going pulse on this line causes the control program of the UPI-41 to begin execution at location 0.

(b) Controller to ADM-3A Interface

The controller to ADM-3A interface consists of a single line which originates from line 0 to port 1 on the UPI-41, passes through the 75188 inverting line driver, and terminates on the Receive Data, RXD, input of the ADM-3A. The line driver converts the TTL output of port 1, 0 - 5 volts, into RS-232C logic levels of ± 12 volts.

Information is transmitted from the UPI-41 to the ADM-3A serially using 7 bit ASCII code under the RS-232C communication protocol. For this application, the number of bits per character has been minimized by using a single stop bit and no parity bit. For a given serial transmission rate this configuration will result in the fastest possible character transmission time. This time is an important consideration, as each parallel byte received by the controller from the SBC 80/20 will require a 22 character message to be transmitted. With the single start bit, the 9 bit serial character appears as shown in Figure III-26.

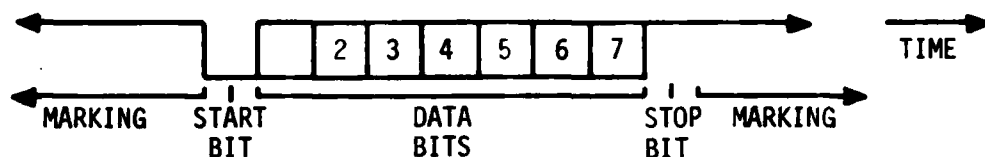


Figure III-26. Serial Transmission Character

Each data byte received by the UPI-41, except as noted in the next section causes a string of 9 bit characters to be sent from the UPI-41 to the ADM-3A, a 24 line by 80 character CRT display.

The function of the ADM-3A is to provide three kinds of information concerning the SBC 80/20 inputs to an observer. The ADM-3A displays a message, indicates the SBC 80/20 source corresponding to the message, and reflects the order of input occurrence. The message is indicated by the characters displayed on the screen. The source is indicated by dividing the ADM-3A screen into 5 columns of equal width, with the first column reserved for source 1 messages, the second column for source 2 messages, and so on for the five sources. The order of inputs is indicated by scrolling the display 1 line each time a message is displayed.

For a screen width of 80 characters, and not allowing an overlap of columns, the message field for each source is limited to the integer portion of $80/5$, or 16 characters. The ADM-3A screen use is illustrated in Figure III-27.

To implement the function of the ADM-3A as described above requires that 22 characters be sent to the ADM-3A for each SBC 80/20 to controller transfer. The 22 characters are sent in 3 groups: a cursor control group, a message group, and a display control group.

The first group sent, the cursor control group, contains four characters which cause the cursor to the ADM-3A to position itself at the beginning of one of the five message columns. The first two control characters "escape" and "equals", activate the ADM-3A cursor positioning logic, while the next two characters are interpreted as the X and Y coordinates of the new cursor position, respectively. The Y coordinate sent is always the same, 037H, and selects the bottom line of the display. The X coordinate is determined by the SBC 80/20 input source.

The second group sent, the message group, contains 16 characters. These characters will be printed on the screen of the ADM-3A in the message field whose beginning was established by the cursor positioning control group.

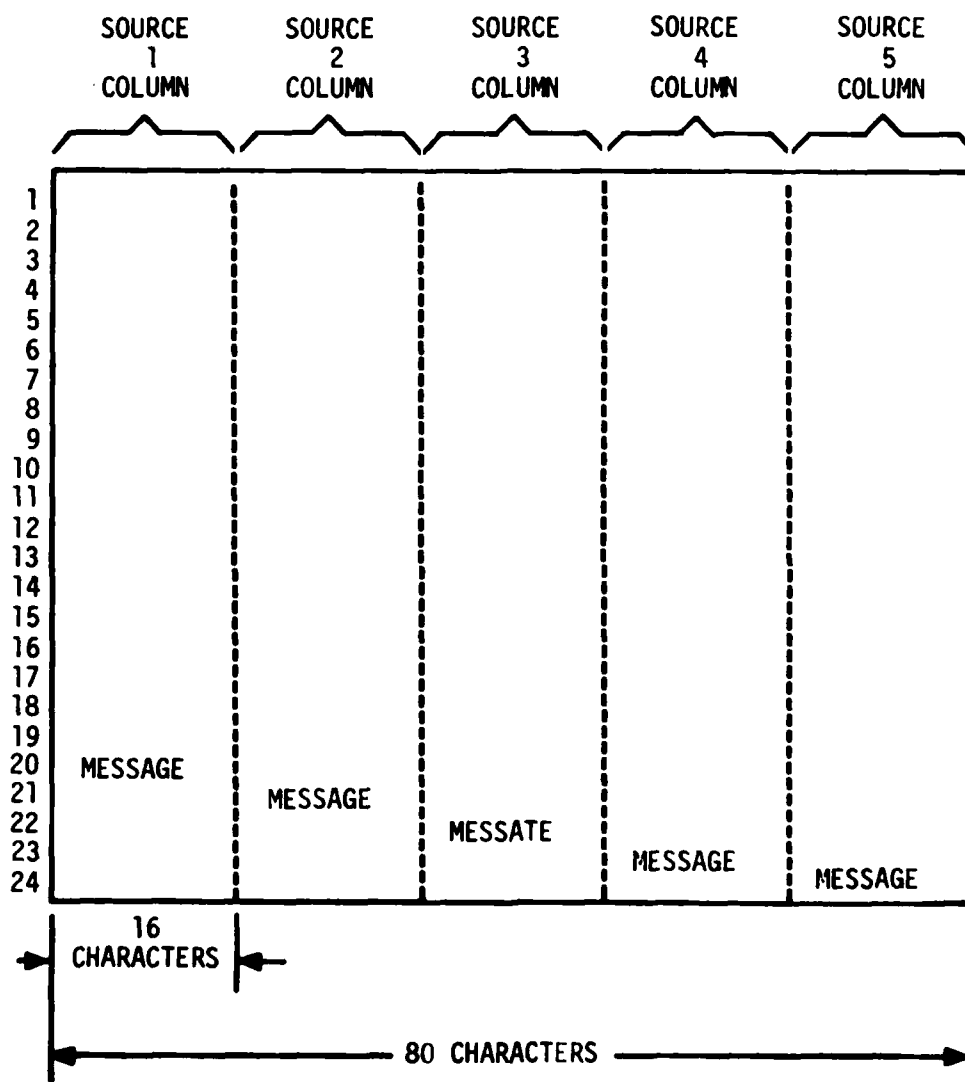


Figure III-27. ADM-3 Screen Use

The third group sent, the display control group, contains the remaining 2 characters. These characters, a carriage return and line feed, cause the display to scroll up one line in preparation for the next control group 1 sequence.

The complete 22 characters string appears as shown in Figure III-28.

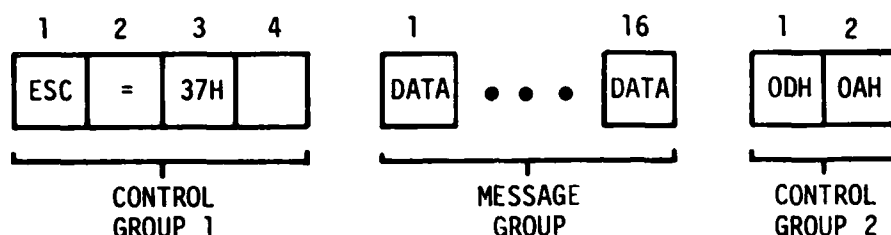


Figure III-28. Character String Transmitted to ADM-3A

The final aspect of the controller to ADM-3A interface is the serial transmission rate to be used. Having now established (1) the number of characters sent by the controller to the ADM-3A per SBC 80/20 input, (2) the number of serial bits per character, 9, and (3) the UPI-41 processing rate requirement, 60 transfers/sec, a minimum serial transmission, or baud, rate can be computed as:

$$\text{minimum baud rate} = \frac{22 \text{ characters/SBC 80/20 transfer} \times 9 \text{ bits/character} \times 60 \text{ SBC 80/20 transfers/second}}{1} \quad (1)$$

or 11,880 bits per second. The next highest, indeed the highest, baud rate at which the ADM-3A can receive data is 19,200 baud. This value must necessarily be chosen as the data transmission rate.

(c) Control Switches to Controller Interface

The control switches to controller interface is a 5 line connection between 5 control switch outputs and the 5 least significant inputs of port 2 on the UPI-41. The design of port 2 on the UPI-41 is such that if nothing is connected to a port line, the line will read as a logic one, whereas, if the line is grounded through a 1k resistor, the port will read a logic zero (3). The control switch to controller connections are shown in Figure III-29.

During the processing of a data byte by the UPI-41, the binary source identifier is translated into a linear select code which is then compared with the switch setting on port 2. If the switch corresponding to the source identifier is set in the abort position, a logic 0 is present and a message will not be sent. This is the exception referred to in the controller to ADM-3A interface description. If the switch is set in the display position

a logic true will be present and a message will be sent.

This concludes the overall system description. The next two sections will describe the principle device within the intelligent controller, the UPI-41 single chip microcomputer.

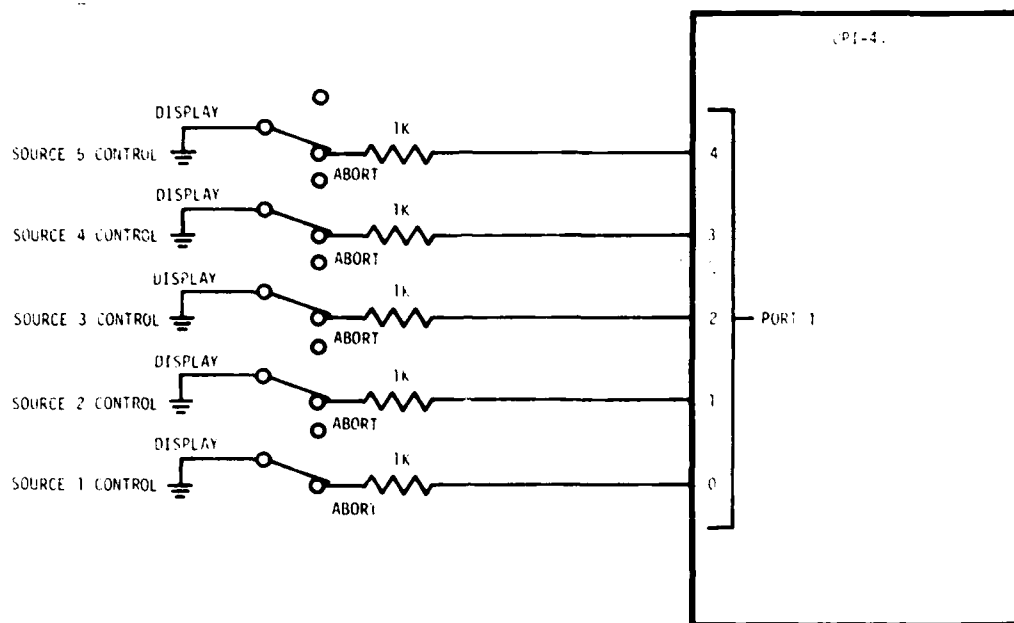


Figure III-29. Control Switches to Controller Interface

e. UPI-41 MICROCOMPUTER OUTPUT II

The UPI-41 single chip microcomputer provides the intelligence of the intelligent controller. The block diagram in Figure III-30 illustrates the facilities available within the UPI-41.

As described in the previous section, the interface register is used for communication with the SBC 80/20, port 1 is used for communication with the ADM-3A, and port 2 is used for communication with the control switches.

Program memory is divided into 4 pages of 256 bytes each. These pages are numbered 0 to 3. Page 0 contains the main loop of the control program, while page 1 contains the various subroutines called by the main loop. Page 3 has a special feature in that data bytes can be transferred from it to the accumulator using the current value of the accumulator as a pointer. This "table lookup" feature is used to access the message strings which are sent to the ADM-3A. 16 messages of 16 characters each are stored, using all 256 bytes within the page. Program memory configuration is shown in Figure III-31.

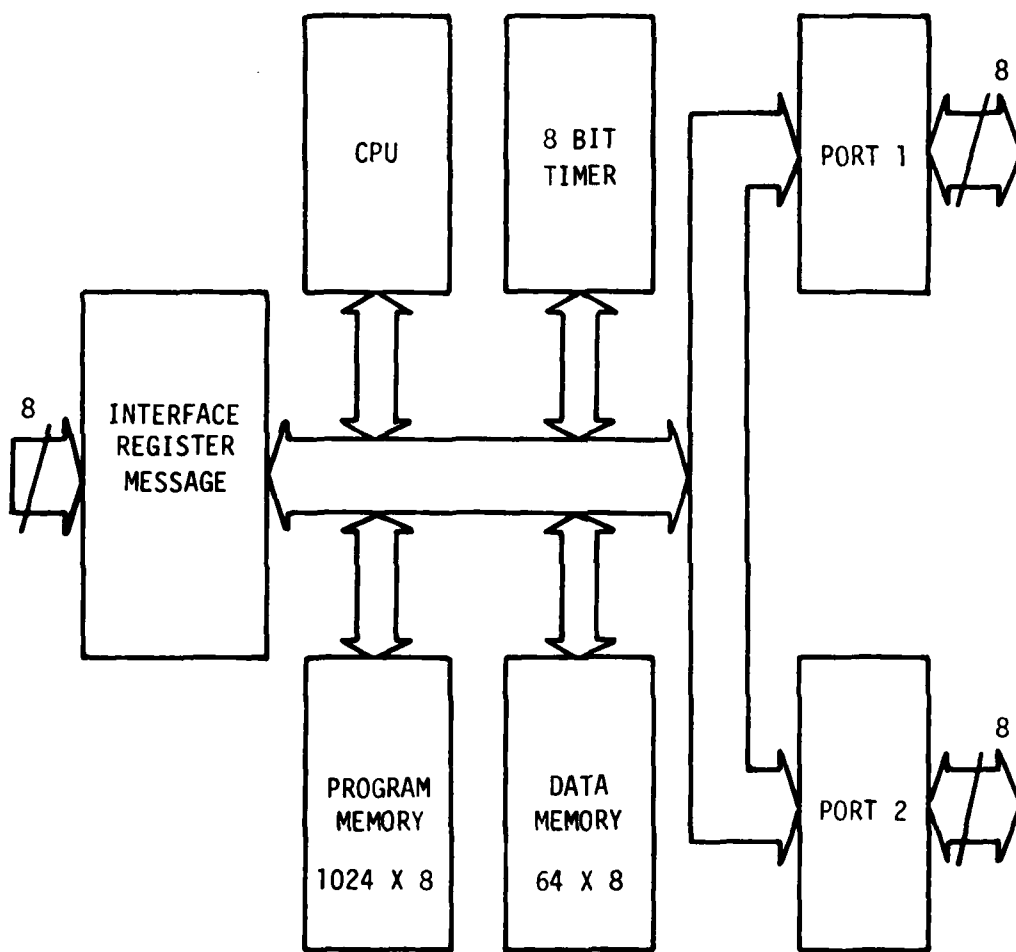


Figure III-30. UPI-41 Single Chip Microcomputer Block Diagram

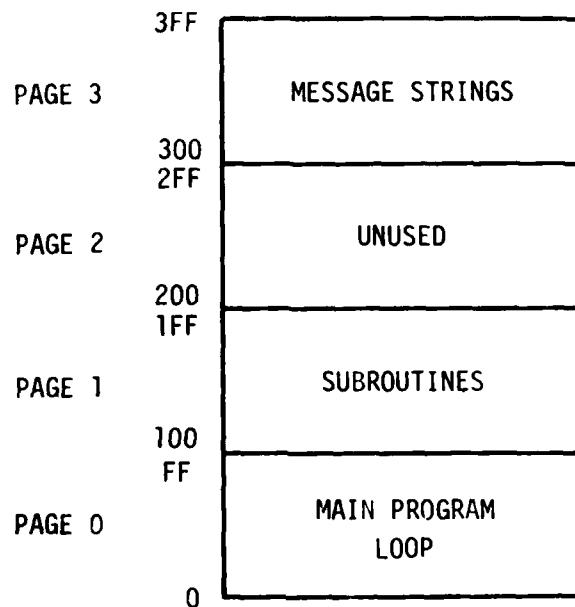


Figure III-31. UPI-41 Control Program Memory Map

RAM within the UPI-41 serves three purposes: it contains the registers, the subroutine and interrupt stack, and the variable data storage locations. The distribution of the 64 RAM locations between these three functions is shown in Figure III-32.

The registers in bank 0 are designated R0-R7, while those in bank 1 are designated R0'-R7'. Only one register bank at a time can be addressed. Bank selection is accomplished by executing a special select register bank X, SELRBX, instruction where X is either 0 or 1. The registers of bank 0 are used for data processing and message transmission, while those of bank 1 are used for queue control.

The UPI-41 contains a rather sophisticated timer which was evaluated for use as the bit interval generator for UPI-41 to ADM-3A serial transmission. As several difficulties were encountered, the use of the timer while representing a possible area for future research, was rejected in favor of a software timing approach. The software timing routine will be described, along with the rest of the UPI-41 control program, in the next section.

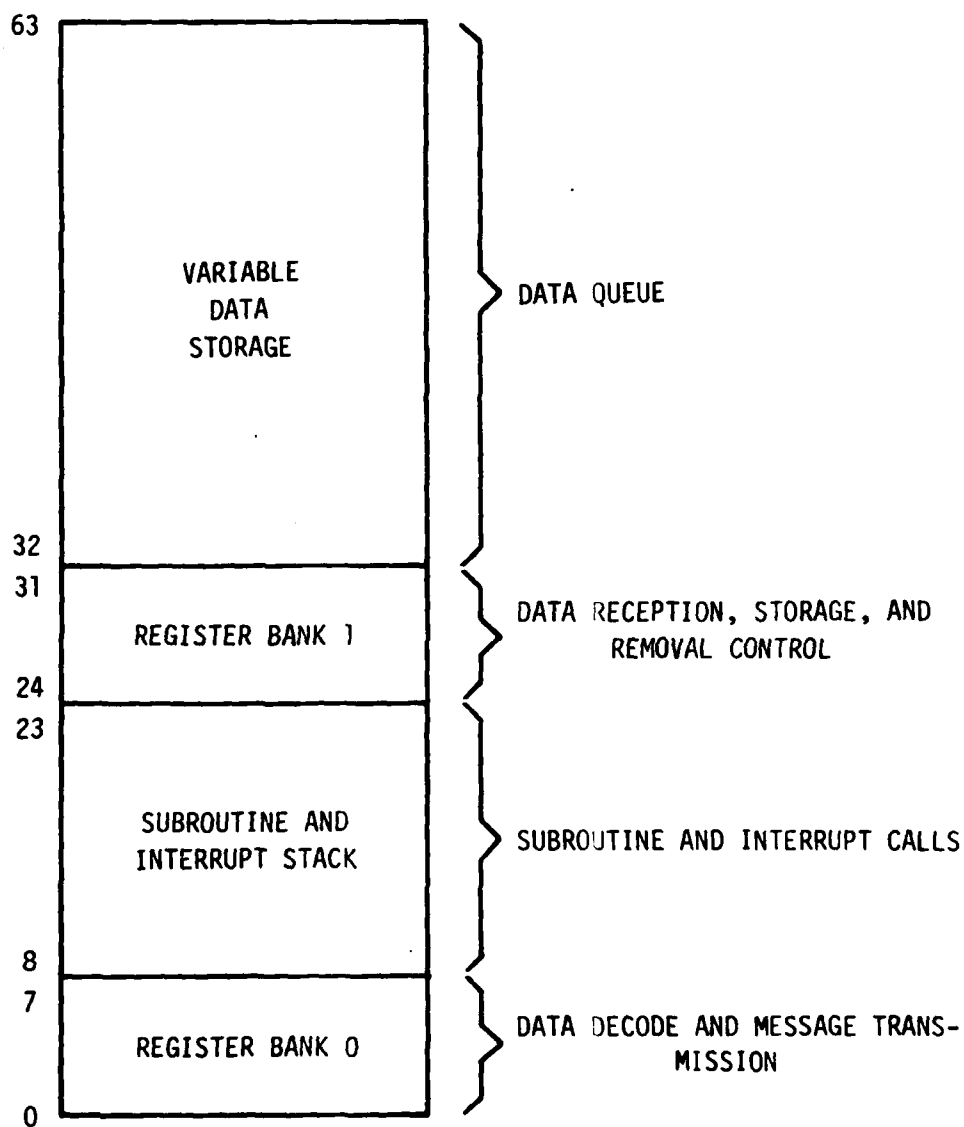


Figure III-32. UPI-41 RAM Memory Map

f. UPI-41 CONTROL PROGRAM IV

The UPI-41 program is written in MCS-48/UPI-41 assembly language. It was assembled using a cross assembler operating on an Intel Microcomputer Development System, MDS-800. The machine code was burned into the EPROM program memory of the UPI-41 using an Intel Universal Prom Programmer and the Universal Prom Mapper Software. The assembly of the program and the burning of the EPROM were done under control of the Intel System Implementation Supervisor, ISIS II, operating from an Intel Dual Floppy Disk Drive.

The program description is divided into three parts:

- Initialization procedures
- Data reception and storage
- Data decode and message transmission

The program listing is located in Appendix "D", flowcharts appear in Figures III-33 through III-35.

(1) Initialization Procedures

The first section of the UPI-41 program performs functions which are necessary prior to data reception. These functions are the initialization of registers and the initialization of the ADM-3A screen. The values placed in the various registers will be explained as they are encountered within the program. The screen initialization procedure consists of clearing the screen and positioning the cursor in the bottom left hand corner. The screen is cleared by transmitting a special character, 01AH, to the ADM-3A, while the cursor is positioned using the 4 character cursor positioning sequence described previously in the controller to ADM-3A interface section.

As the final step in the initialization procedures, the UPI-41 enables itself to data reception by outputting a logic zero to port 2 line 7. This port line is connected to the UPI-41 chip select, CS, input. Since all port lines are in the logic high state following a system reset, UPI-41 input is disabled until the output instruction is executed.

(2) Data Reception and Storage

When data is written into the UPI-41 interface register by the SBC 80/20, an interrupt request is generated. Upon recognition of the interrupt, the interrupt vector jump at locations 3 and 4 in program memory is executed, and the interrupt service routine, lines 118 through 134 in Appendix "D", is entered. The interrupt routine inputs the data from the interface register and places the data in a queue. A flowchart of the interrupt service routine appears in Figure III-34.

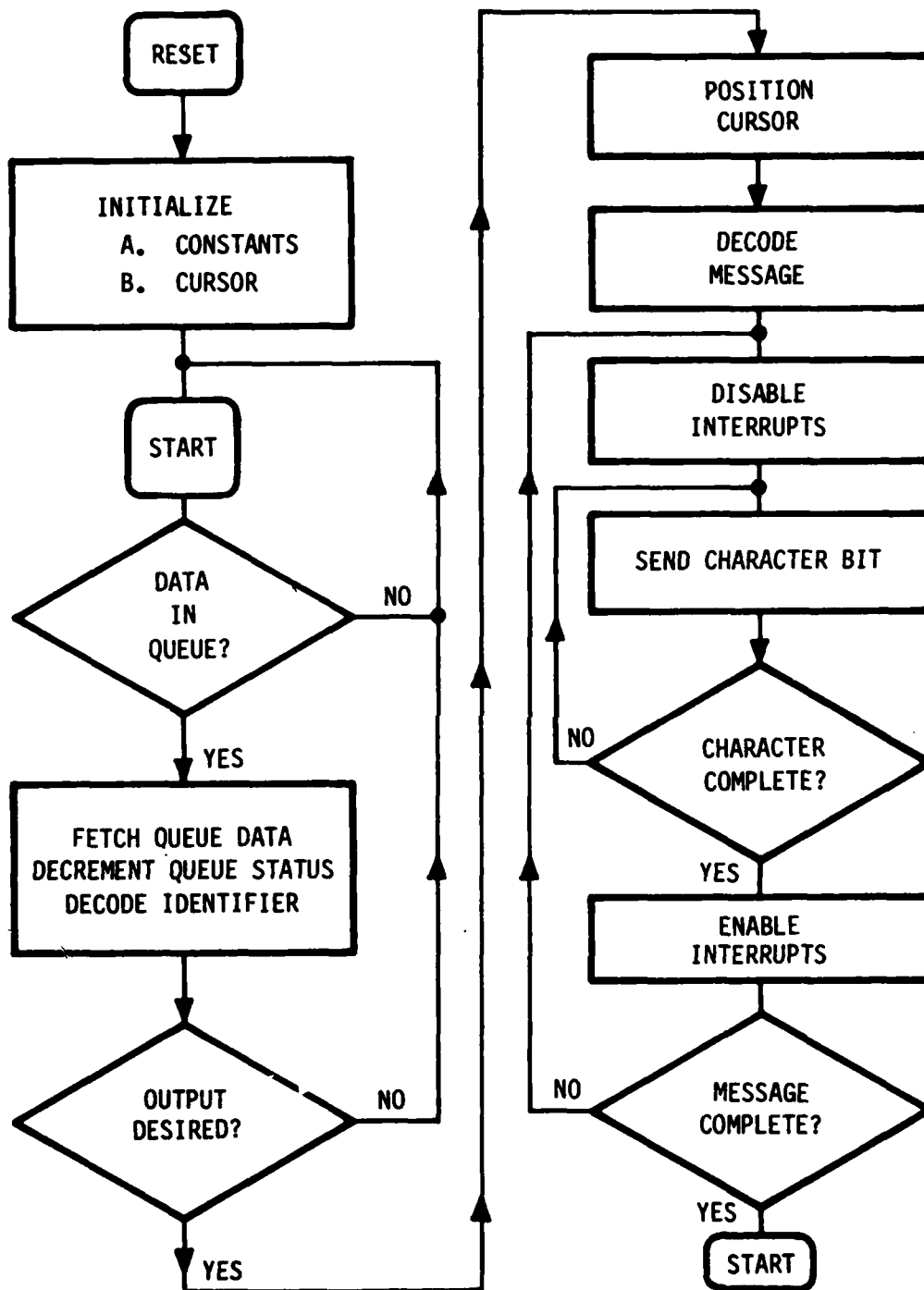


Figure III-33. UPI-41 Control Program Flowchart Processing Loop

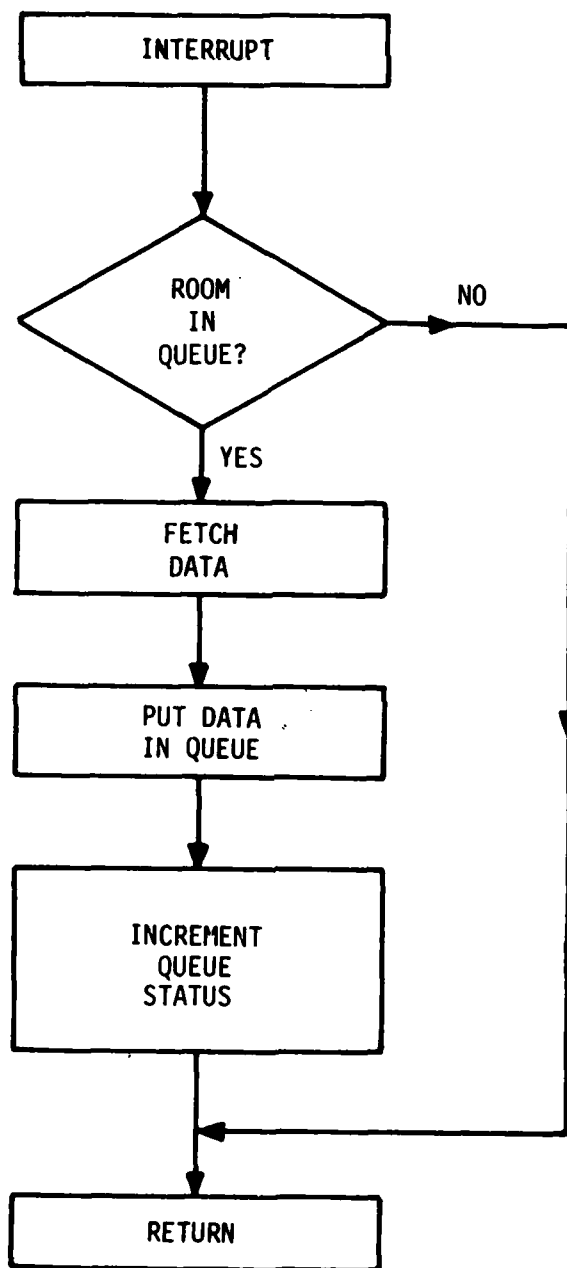


Figure III-34. UPI-41 Control Program Flowchart - Interrupt Service Routine

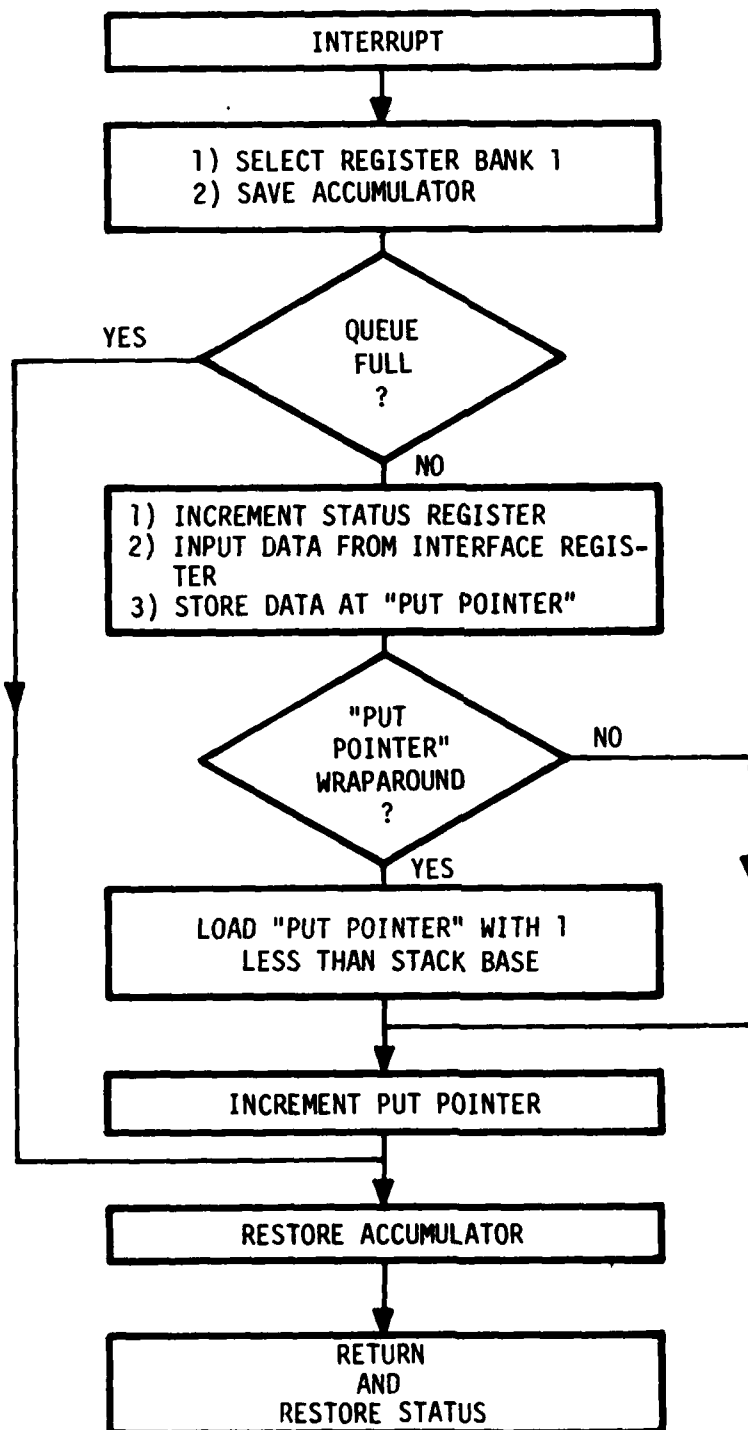


Figure III-35. Interrupt Routine Flowchart

It was pointed out in the section describing the SBC 80/20 controller interface that the UPI-41 reception rate requirement would necessitate the data queue. The necessity for the queue can be shown as follows:

Unless the 19,200 baud rate can meet the UPI-41 reception rate requirement as well as the processing rate requirement, it is necessary to provide a data queue to prevent data from being overwritten in the interface register. For this condition to be met, the 19,200 baud rate must be proportionately greater than the 11,800 minimum baud rate by at least the proportion of the reception rate requirement to the processing rate requirement, or

$$\frac{19,200}{11,800} \stackrel{M}{=} \frac{200}{60} \quad (2)$$

as this is not true, a queue must be maintained.

To meet the storage requirements a First In First Out, or FIFO, stack is implemented in the variable data storage area of the RAM memory. See Figure III-36. A FIFO stack allows data to be retrieved so that order of entry is preserved. The operation of a FIFO stack can be conceptualized by considering a storage mechanism where data inputs are stacked one on top of the other as they arrive, and where data removal is accomplished by pulling from the bottom. As an entry is removed, all remaining entries move down one location. This operation is illustrated in Figure III-36.

The problem with this implementation is in moving the remaining data entries down. For N remaining inputs, the operation requires 2N memory accesses and 5N program steps as shown below:

- (a) Increment pointer
- (b) Load data byte - first memory access
- (c) Decrement pointer
- (d) Store data byte - second memory access
- (e) Increment pointer

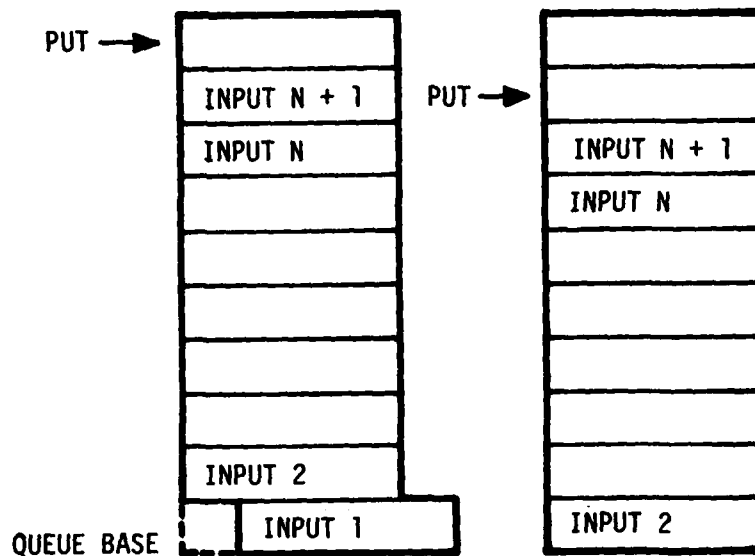


Figure III-36. Fixed Base FIFO Operation

A more efficient algorithm uses a "get data" pointer as well as the "put data" pointer used in the implementation above. The get data pointer allows the "bottom" of the stack to move upward as data is removed from the stack. This eliminates the necessity of moving each of the remaining inputs down. Instead, the get data pointer is incremented once each time data is removed. The put data pointer always identifies the next location available for data storage and the get data pointer identifies the location of the next value to be removed. The only problem with this implementation is that unless data memory is infinitely long, storage locations will run out at some point. This condition being unacceptable, a "top-of-stack" must be defined, and as the pointers reach the top they must be wraparound. In this application the top-of-stack has been made coincident with the top of RAM, making the last location address 63 and giving a stack size of $(63-32) + 1$, or 32 locations. As each pointer reaches location 63, it is returned to location 32 instead of being incremented further. Implemented in this manner, the number of steps required for a data removal is independent of N and, for the UPI-41, has a maximum value of 5 as indicated by lines 85 through 89 of the program listing.

For either implementation, some way of determining when the stack is full must be available. For the two pointer implementation, the queue full condition is easily detected by maintaining a queue status value which indicates how many entries are presently on the stack. If a check of the queue status register indicates that the queue is full, additional data must be rejected to avoid overwriting of the earliest entry with the newest entry. Since the UPI-41 has been designed to meet the processing rate requirement, it follows that the maximum stack usage must be less than or equal to the number of SBC 80/20 input sources, or 5; therefore, the queue full condition can never occur in this application. Use of the queue status value in this application, then, is limited to determining when data is available on the stack. Figure III-37 illustrates the operation of the moving base FIFO stack.

The put and get data pointers, the queue status, and the constants used to determine the pointer wraparound and queue full conditions are located in register bank 1. Also, since the data reception routine is entered in response to an interrupt, another bank 1 register is allocated for accumulator storage. Finally, one register is used for temporary data byte storage during computations.

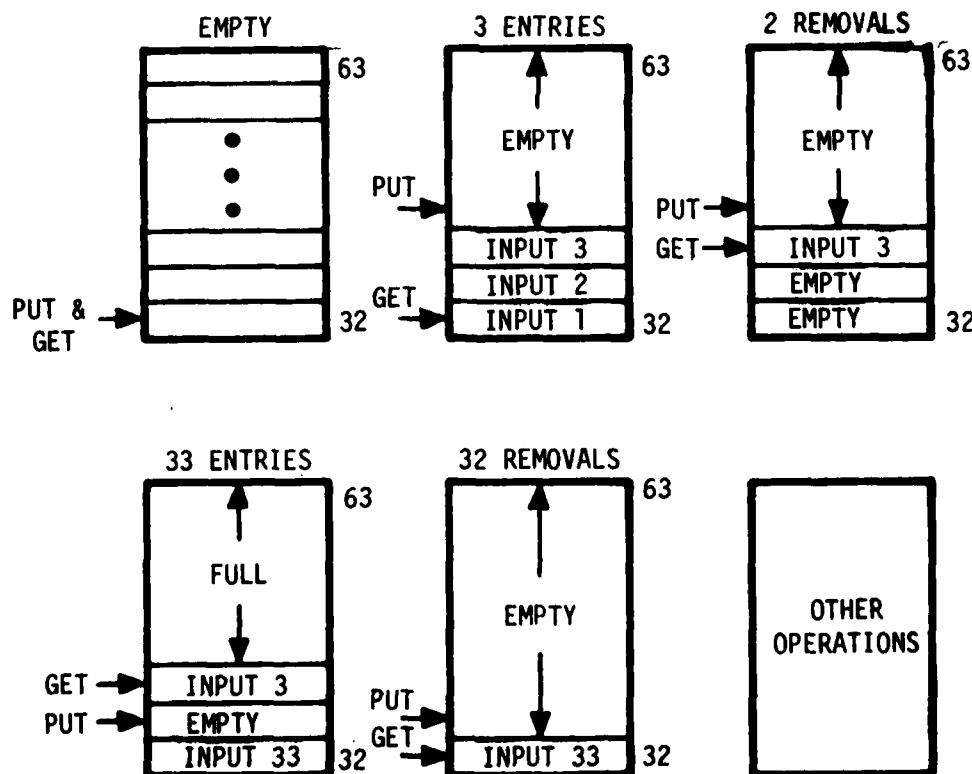


Figure III-37. Moving Base FIFO Operation

Registers 0 and 1 are the only locations which can serve as pointers into the variable data storage area; therefore, the get and put data pointers are defined as the contents of registers 0 and 1 respectively, the other locations are assigned arbitrarily as per Table III-1.

TABLE III-1. REGISTER BANK 1 MAP

Register 7'	Temporary Storage
Register 6'	Queue Status Con. = 224
Register 5'	Wraparound Constant = 193
Register 4'	Unused
Register 3'	Accumulator Storage
Register 2'	Queue Status
Register 1'	Put Data Pointer
Register 0'	Get Data Pointer

(3) Data Decode and Message Transmission

Once data is placed in the queue by the interrupt service routine, a check of the queue status register, lines 80 and 81 of the program listing, will indicate that data is available for processing. The program will then enter the main program loop, line 82, where the data decode and message transmission function begins.

This section of the program can be divided into 3 segments:

- (a) Data access
- (b) Source processing
- (c) Message processing

1. Data Access

The function of the data access segment is to remove a data byte from the queue and perform the transition between register bank 1 operation and register bank 0 operation. The data removal steps are reminiscent of the steps performed in the interrupt routine, while the bank transition is accomplished by placing the data in the accumulator and then selecting the new register bank. A flow-chart is shown in Figure III-38.

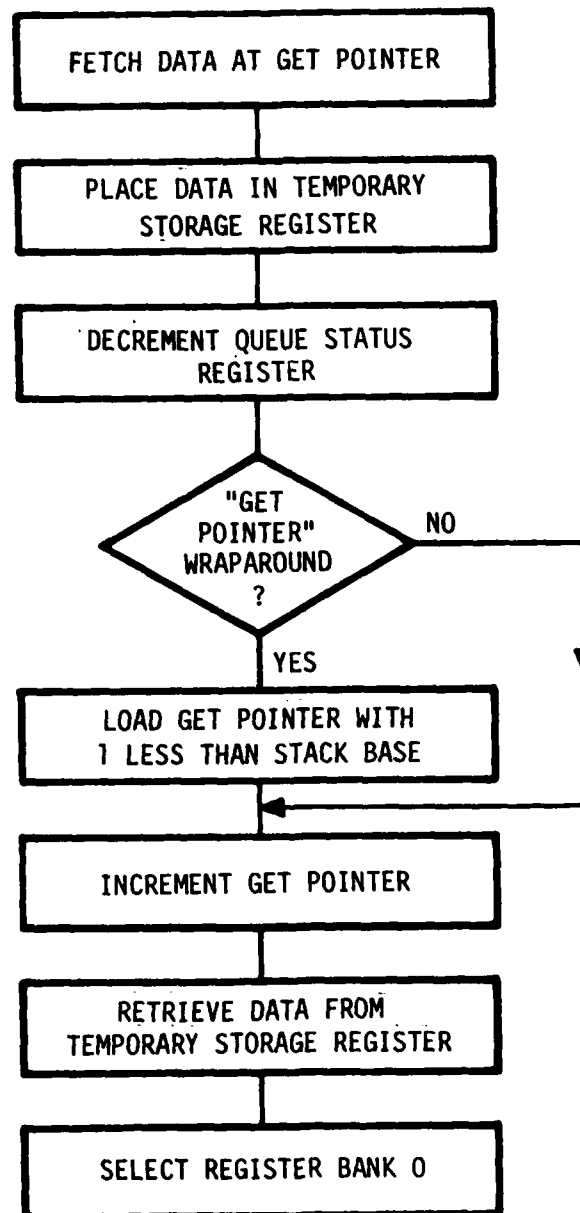


Figure III-38. Data Access Segment Flowchart

Register bank 0 is used for the remainder of the program. All locations within this bank are assigned arbitrarily as shown in Table III-2.

TABLE III-2. REGISTER 0 MAP

Register 7	Data Byte
Register 6	Message Length Constant
Register 5	Binary Source Identifier
Register 4	Linear Select Source Identifier
Register 3	Message Identifier
Register 2	Relay Counter
Register 1	Unused
Register 0	Serial Transmission Counter

2. Source Processing

The function of the source processing segment, lines 93 through 98, is to use the source identifier portion of the data byte to (1) determine whether a message transmission is desired and (2) position the cursor at the proper place on the ADM-3A screen. The source processing segment calls three subroutines; MASK, LOCSET, and TAB.

Subroutine MASK, lines 144 through 156, converts the binary source identifier into the linear select identifier through the use of the lookup table located at MSKDAT, line 143. The subroutine then performs the comparison with the port 2 control switch lines and sets a flag according to the result.

Subroutine LOCSET, lines 161 through 168, sends the characters which activate the cursor control logic and the Y coordinate value to the ADM-3A.

Subroutine TAB, lines 157 through 160, converts the binary source identifier into the proper X coordinate value and completes the cursor positioning sequence by transmitting the coordinate value to the ADM-3A.

3. Message Processing

The function of the message processing segment, lines 99 through 112, is to convert the message identifier portion of the data byte into the page 3 address of the message string, output the message string, scroll the ADM-3A display one line, and return to the queue status checking loop.

The page 3 address of the message string is produced by multiplying the binary message identifier by 16. Thus, the message identifier is converted into the starting address of a 16 character string which makes up the message. The multiplication is accomplished by swapping the high and low order nybbles of the data byte and then masking out the low order nybble. This operation is equivalent to four left shifts and, therefore, multiplies the source identifier by 2^4 , or 16.

Subroutine STROUT, lines 169 through 175, uses the message address produced by the preceding multiplication and the string length constant contained in register 6 to control the transmission of the 16 character message string to the ADM-3A.

The CRLF procedure, lines 107 through 110, cause the scroll of the ADM-3A display by sending the carriage return line feed combination.

Finally, register bank 1 is selected so that when the jump at line 112 occurs the register bank containing the queue status value, R2', will be addressed by the WAIT loop.

This completes the description of the control program except for the subroutine which controls character transmission. This function is accomplished by the OUTPUT subroutine, lines 176 through 191.

It was noted in the description of the clock connection, section II, that the 4.608 megahertz clock input to the UPI-41 would be used to generate the proper communication frequency. The following discussion explains this process and the operation of the OUTPUT subroutine,

Each instruction in the UPI-41 instruction set consists of either 1 or 2 instruction cycles. Each instruction cycle consists of 5 machine states and each state consists of 3 clock periods. See Figure III-39.

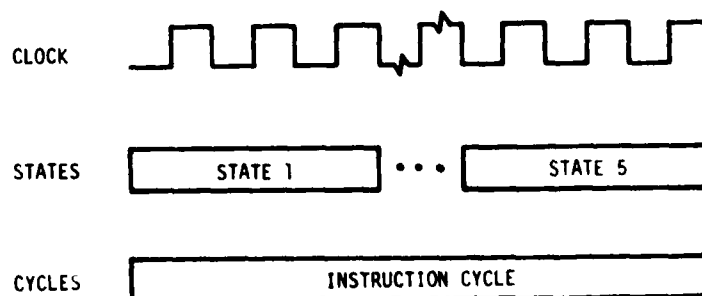


Figure III-39. UPI-41 Instruction Cycle

The instruction cycle execution rate, then, is 1/15 of the input clock rate or 307,200 instruction cycles per second. The instruction cycle execution rate divided by 16 produces the serial transmission rate of 19,200 baud. Therefore, a bit interval, i.e., the time a serial bit should be present on port 1 during transmission, is exactly 16 instruction cycles. A 9 bit character can be transmitted by constructing a loop which places a new serial bit on the port 1 transmission line every 16 instruction cycles.

The OUTPUT subroutine, Figure III-40, expects the 7 least significant accumulator bits to hold the 7 bit ASCII representation of the character to be sent. As 9 bits are required to send a complete character, including the start and stop bits, the 8 bit accumulator and the carry bit are catenated to form a 9 bit register. The accumulators most significant bit and the carry bit serve as the stop and start bits respectively. Once the 9 bit register is set up with the character, the bits are sent by successively rotating the bits into the least significant bit position of the accumulator and then outputting the accumulator to port 1.

Instructions 1, 2, and 3 set up the character, the transmission loop begins at line 180. Note that the number of instruction cycles required for each instruction in the transmission loop is shown to the right of the instructions.

For the first eight bits transmitted, program execution proceeds through the steps indicated 1 through 8. As can be verified by the reader, 16 instruction cycles are executed between bit changes.

```

OUTPUT: DIS      I
             MOV   R0, #07H ; SERIAL BIT COUNTER
             MOV   A, R1  ; GET ASCII CHARACTER TO BE OUTPUT
             ANL   P1, #00H ; PUT OUT START BIT
             MOV   R2, #04H ; SET UP DELAY LOOP LENGTH
             CALL  DELAY
LOOP1:  OUTL   P1, A  ; OUTPUT CURRENT BIT OF SERIAL CODE
             RR    A    ; GET NEXT BIT OF ASCII CODE
             NOP     ; WAIT 1 INSTRUCTION CYCLE TO COMPENSATE
                   ; FOR RR BEING A SINGLE CYCLE OPERATION
             MOV   R2, #02H ; SET UP DELAY LOOP LENGTH
             CALL  DELAY
             DJNZ  R0, LOOP1 ; TEST FOR 7 BITS OUTPUT
             ORL   P1, #01H ; PUT OUT STOP BIT
             MOV   R2, #03H ; SET UP DELAY LOOP LENGTH
             CALL  DELAY
             JF0   NOTINEN ; IF IN SETUP SEGMENT DONT ENABLE INTERRUPTS
             EN    I
NOTINEN: RET      ; RETURN FROM SUBROUTINE

```

Figure III-40. UPI-41 Character Transmission Subroutine

Program flow for the final bit proceeds through the steps indicated A through E. While this sequence requires only 9 instruction cycles, analysis of the complete program shows that for any set of conditions a minimum of 8 additional instruction cycles will be required to reach the initial output instruction for a new character. Thus, a minimum of $9 + 8$, or 17, cycles will be executed exceeding the minimum of 16 by 1 cycle. But, as there is no maximum length for the stop bit since its level corresponds to the nonactive, or "marking" state, the value 17 is acceptable.

The instruction executed just prior to entry into the bit transmission loop disables interrupts, while the instruction just before the return reenables them. Interrupts must be disabled during transmission of a character since the occurrence of an interrupt service routine would insert extra instruction cycles, thereby destroying the integrity of the software timing loop.

As a concluding remark on the UPI-41 control program, it is noted that starting on page 51 of the listing, a sample set of message strings is shown.

The program listing referred to throughout this section is the assembly listing produced during the assembly of the UPI-41 control program source file. This version of the program was used for the system evaluation to be presented in the next section.

3. 80/80 PROGRAM

Operation of the 80/20-4 Microcomputer is directed by program code in three 2716 2KX8 EPROMS. The code was compiled from a program written in PL/M-80 language. Reference 3 gives a number of PL/M-80 examples. While reference 5 provides the language syntax and other definitions.

The overall program strategy is shown on Figure III-41, with more detail given on Figure III-42. The program listing is given in Appendix A.

After power has been turned on, the program starts when the 80/20-4 "RESET" button is pushed.

During "initialize" the program issues a series of questions to the system console and prompts for answers as shown on Figure III-41. If desired, the date, training session number, and trainee names are obtained and stored for future reference. When all identification data have been collected and other housekeeping details completed, the program issues "LET'S START" and the main training session loop is entered. This loop may be executed along the three different paths indicated on the flowchart, Figure III-42.

If there is no target present on the screen and no rifle trigger is pulled, i.e., no "action", then path 3 will be selected by program logic. No data comments are generated during early passes around the path 3 loop. Subsequent "action" will cause flags to be set and a return to path 3 may result in a comment of either "NO TARGET" or "YOU FROZE" being sent to the earphones of any erring trainee. Corresponding error data are filled in RAM memory for the identified trainee which will lower his score printed out after the session ends.

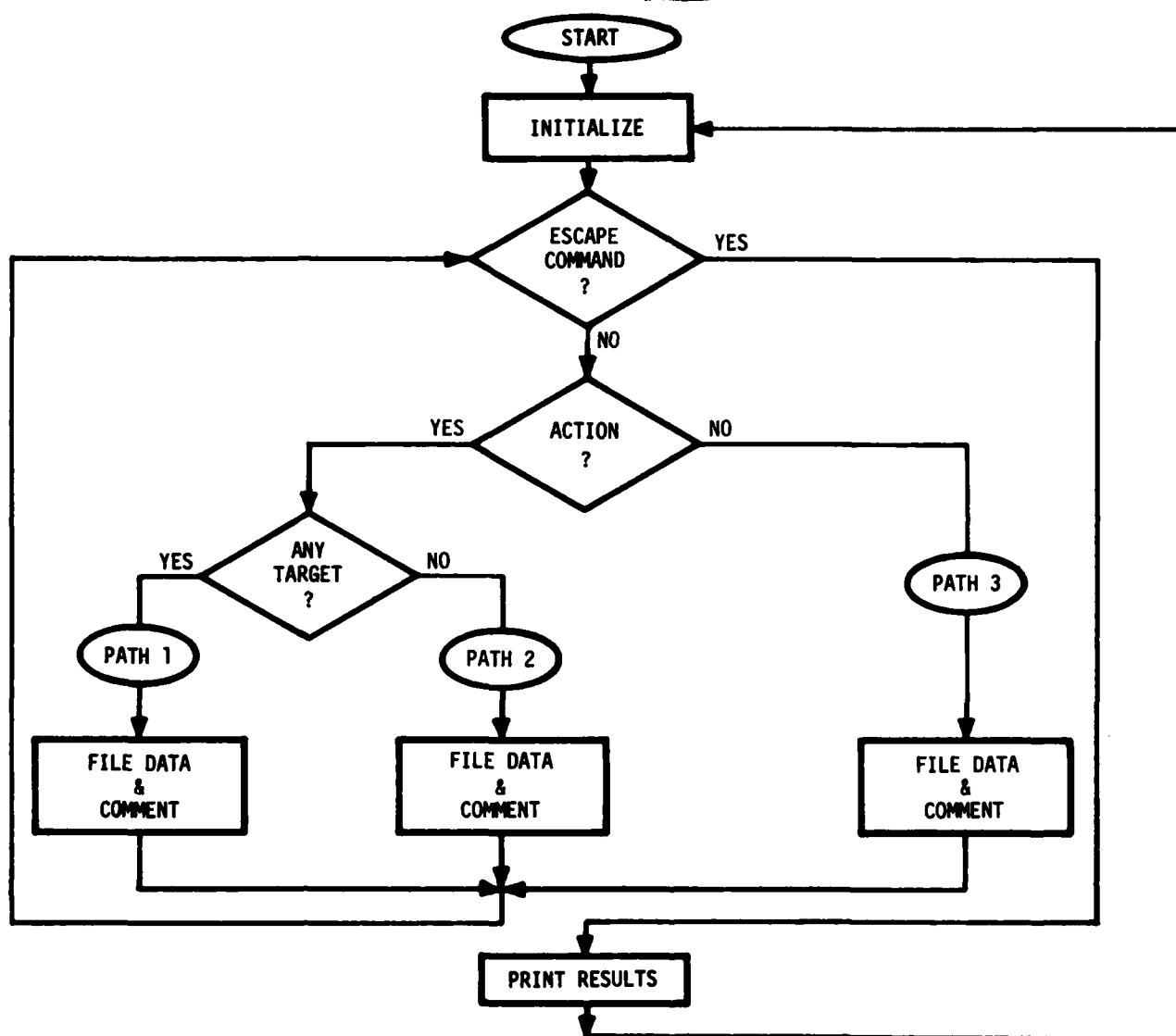


Figure III-41. 8080 Program Strategy

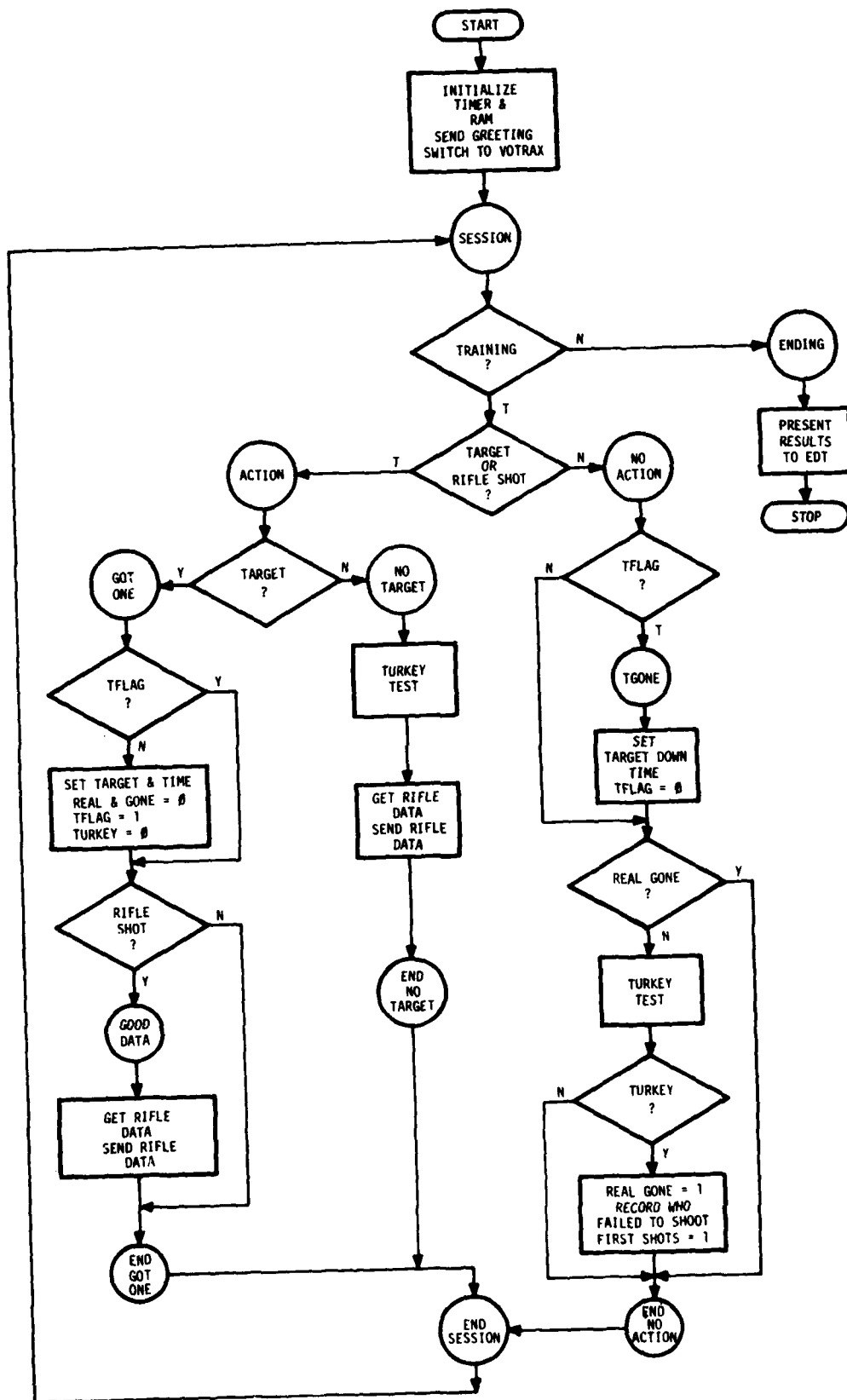


Figure III-42. Program Flowchart

"ACTION" is true after a target becomes available and/or a rifle is "FIRED". The session loop will now pass through either path 1 or 2 as dictated by program logic. This logic also determines the proper data to be filed and comment to be sent to the trainee's earphones. For example, if the target disappears but the trainee persists in shooting for more than one second, then a "NO TARGET" comment will be sent to the rifleman, and a negative score is placed in his data file.

Operation remains locked in the session loop until an escape is signaled by the squad leader's pushing the start/print button. This causes a system interrupt and control passes to "PRINT RESULTS" which produces output as shown on Figure III-41. The program then returns to the "INITIALIZE" block and data records are initialized in preparation for the next training session.

4. SCORE DISPLAY AND WORST PERFORMANCE

The final score for each trainee is calculated by adding the following items:

- 100 x (Hits per shot)
- 60 x (Near misses per shot)
- 30 if average reaction time = 0.5 seconds, or less, or
- 20 if average reaction time > 0.5 but ≤ 0.9 seconds, or
- 10 if average reaction time > 0.9 but ≤ 1.3 seconds, or
- 0 if average reaction time > 1.3 seconds.
- 2 x (Number of targets ignored)

When the light emitting diode or "LED" is lit under a trainee's CRT column, he is the "WORST SHOOTER" inasmuch as he has the highest total of these items:

- Misses
- Shots with no target present
- Targets ignored - PP the worst shooter is recomputed each time a target is not present on the screen.

5. SELF CHECK

The SWAT self check has two parts: (1) an SBC 80/20 check, and (2) the interface board (IFB) check. These procedures are described separately.

a. SBC 80/20 CHECK

The 80/20 single board computer checkout requires the insertion of an INTEL SBC 416 ROM extender board on which is located the test program driver and a duplicate of the SWAT version 1.2 program. The duplicate program is contained in five 2708 ROMs. These ROMs are located at addresses through 0C000H, see Appendix E.

The 80/20 check verifies proper operation of the SBC memory, I/O ports, timer and USART. A complete check requires about a second. To run the check, the normal 50 pin connectors to "J1" and "J2" of the 80/20 must be removed and replaced with a special test strap which connects input terminals of J1 to output terminals on J2 and vice versa. The SBC 416 board must also be inserted into the computer card cage. While not necessary, it may be convenient to remove the interface board to avoid damage to the wire-wrap pins. The SWAT program is started with a reset in the normal way. The first thing that the SWAT program does is to determine if the SBC 416 board is in the card position on the SBC 416. If the SBC 416 is in place, the program finds a "1" located at address 0C000H. If it is not in this location, no transfer acknowledge signal will be returned. After a millisecond, therefore, the attempt is aborted and a "00" is read into the 8080's accumulator. The 00 occurs due to the 1K pullups on the inputs of inverting line drivers. With a 1 found at 0C000H, the test program is conducted, while if a 00 is returned, the normal SWAT program starts.

The 80/20 test signals its completion by turning on a "LED" located near the top right corner of the 80/20 barod. If no flashes are noted, the test was completed in a satisfactory manner. If flashes occur, some trouble was uncovered, the nature of which is indicated by the number of quick flashes grouped together. The trouble code is:

1 FLASH	==	RAM FAILURE
2 FLASHES	==	ROM FAILURE
3 FLASHES	==	I/O FAILURE
4 FLASHES	==	TIMER TOO SLOW
5 FLASHES	==	TIMER TOO FAST
6 FLASHES	==	USART FAILURE

A TIMER TOO FAST
6 FLASHES == USART FAILURE

A flash group is sent for each failure detected, so it is possible that more than one trouble code may be detected during a single 80/20 test. The test may be terminated only by turning off the power. This should be done, of course, before any physical changes are made to the computer.

b. IFB SELF TEST

To test the interface board, the 80/20 test SBC 416 board may be removed from the card cage and the IFB reinserted. The normal 50 pin rifle input strap connector and the 34 pin strap connector must be removed. A special strap must be connected between the 50 pin rifle test simulator output and the 50 pin rifle input connector.

The CRT and Electronic Data Terminal, EDT, are needed for the test. The VOTRAX unit may optionally be disconnected or left connected.

To conduct the test, the SWAT system is started as usual by a start or reset. When the query "WANT ID YES OR NO?" is presented on the EDT, a control-T for "TEST", should be entered. The system should respond with:

RIFLE SIMULATOR STRAP IN PLACE?

If no errors are found, the test requires 18 to 19 seconds to run, and the system responds with:

TEST COMPLETE.

Hitting any key on the data terminal will result in the standard output being typed out, as shown in the attached listing.

If trouble is detected, the rifle "ID" number will be typed with "F" for failure and a coded diagnostic: RES, INT or DAT.

RES == The rifle did not receive a reset pulse.

INT == The UPI-41 did not receive the data strobe that it originally sent.

DAT == Improper data was received by the UPI-41.

H. FILM ANIMATION

This section describes the process by which a second projection film is produced to enable the electro-optic rifle receiver to sense targets on the movie screen and score the trainee's performance.

A black-and-white animated companion film is prepared for simultaneous synchronized projection on an infrared projector with a full color battle scenario on a second projector. The black-and-white film is animated frame-by-frame to produce a clear target zone surrounded by an opaque field. If more than one target is present then more than one clear target zone is animated within a frame.

Infrared light is projected through the clear target area to produce a target zone for the rifle electro-optic sensors. This infrared target zone is usually animated to directly overlay the visual target being projected by the full color battle scenario but may also be super-elevated and/or lead the target as required.

Film animation has been accomplished by inspecting a battle scenario frame-by-frame on a Movieola (a type of laboratory film analyzer). Each frame is inspected for the total number of targets, and target location. Each frame is catalogued and then compared to each other for target range and rate of transverse motion measured. Lead and superelevation calculations are computed from this data for final animation.

Selected target shapes and sizes are prepared for use in the animation process. These shapes have usually been silhouette, oval, and circles. The target shape is realized as an opaque shape on a clear strip of acetate.

The battle scenario is again viewed frame-by-frame in the animation process. The animator locates targets according to the script and overlays on appropriate target size and shape on a rear projection screen. Lead and superelevation corrections are applied if necessary and then the battle scenario is removed while a single frame of the animated film is exposed.

Upon completion the entire animated film is developed using a reversal process. This process causes the opaque target shapes to become transparent and the background to be opaque. The contrast is adjusted for a $D^* = 2.5$ or better. A D^* of 2.0 has been used successfully.

Copies are made of this master animated film and both battle scene and the animated films are edge numbered for easy identification and editing. A final coat of laquer is then added to protect the emulsions and extend the life of the films.

The resolution of man targets beyond 300 meters is difficult and is a limitation of the system using standard 16mm film.

No research was done on automating the production of the IR target film. It is our opinion that research in this area could reduce both the time and expense of producing the infrared target film.

SECTION IV

CONCLUSIONS

The SWAT was tested successfully by both the U.S. Army and also by the U.S. Marine Corps but under the name UIWT (Universal Infantry Weapon Trainer). The two systems are virtually identical except for the number of trainee firing positions, the SWAT having five while the UIWT consists of four firing positions.

The U.S. Army test was conducted by the U.S. Army Infantry Board (USAIB) for the Directorate of Training Developments, U.S. Army Infantry School (USAIS), Fort Benning, Georgia (See Reference 9). The test was conducted 22 January through 5 March 1980, employing test soldiers from the U.S. Army Marksmanship Unit and TOE units. Testing included use of the SWAT as a vehicle for training riflemen in the technique of engaging moving personnel targets. A comparison of record fire scores, achieved by personnel trained on the SWAT, on the Infantry Remoted Target System, Defense Test Range, and a no-training control group, was conducted to address training potential. Figure IV-1 indicates the results of firers interviews concerning the SWAT weapon and SWAT targets. Some of the major findings were that test soldiers improved their firing performance on three iterations of SWAT firing, but not on three iterations of DTR firing. It is stated that "this test does give some evidence of the SWAT system's potential for training transfer. However, a final estimate of the system's training value cannot be made until a training program is developed which optimizes SWAT performance".

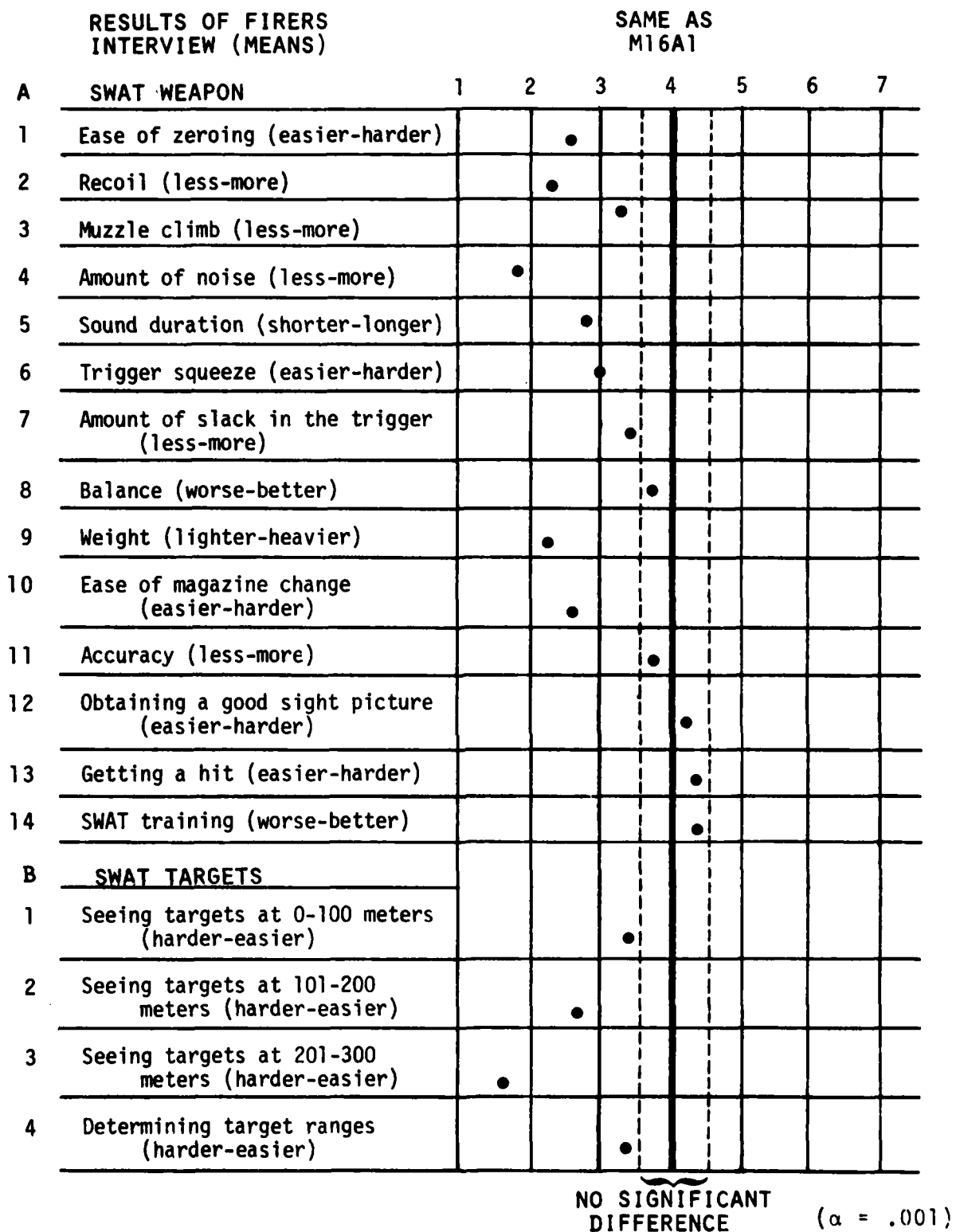
An evaluation of the UIWT's training effectiveness and potential was performed in November 1979 at Camp Lejeune, North Carolina by the U.S. Marine Corps (See Reference 8). The evaluation was conducted by three members of Code N-241, Naval Training Equipment Center. Three different groups of Marines acted as test subjects or provided expert opinion. They are as follows:

(a) 120 enlisted men took part in a quasi-experiment designed to determine the benefits of the UIWT versus a more traditional training method,

(b) Eight highly experienced snipers evaluated the UIWT for its training capabilities. These Marines also serve as marksmanship instructors,

(c) A variety of General, Field and Company grade officers fired the simulator and gave opinions concerning its usefulness.

Generally, the infantrymen were very positive about their experience with the UIWT. They would like to see deployment of the device into actual training situations. An overwhelming number stated that they would rather train in the UIWT than on the pop-up range. The elements of realism and immediate feedback were the main reasons for infantrymen satisfaction with the UIWT.



Note: In all cases the SWAT rifle and targets are being compared to the M16A1 and the DTR range. If there was no difference between them, then a 4 was chosen. The farther away the mean is from 4, the more difference there is between the SWAT and the M16A1.

Figure IV-1. Results of Firers' Interview (Means)

Without exception the officers who fired and observed the UIWT stated that it was a valuable training tool. Some officers went so far as to request that the UIWT prototype remain at Camp Lejeune so that they could start training Marines. A number of officers expressed concern about UIWT maintainability and reliability. They felt that if the UIWT was to be used for large numbers of trainees it would have to have rigid specifications for reliability.

The Marines suggested that there were no special features of the UIWT system (i.e., feedback, recoil, instructor console, infrared monitor, etc.) which should be deleted. All features seemed acceptable and desirable to those who evaluated the system.

The UIWT functioned well throughout the study. Breakdowns and malfunctions occurred on few occasions. This performance record is even more impressive when it is considered that this version of the UIWT is a prototype. The maintainability and reliability of the UIWT, based upon this evaluation, must be considered as good. Future iterations in the production format should only serve to increase these two characteristics.

Despite test limitations, the UIWT evaluation was considered to be successful by those participating in the evaluation. The overwhelming enthusiasm for the training device, exhibited by the Marine personnel who fired it and observed it, gave evidence of its potential usefulness.

The evaluation of the UIWT at Camp Lejeune produced positive findings. Every characteristic of the UIWT met with approval. An evaluation of trial scores on the UIWT provided empirical evidence of the UIWT's effectiveness.

No formalized Program of Instruction (POI) exists for team firing training. Consequently, none was administered with the UIWT evaluation. This area of training should be given consideration in the future. Presently the instructor merely gives informal directions on how a fire team should function. The same informal procedure is followed for marksman-ship instruction. Observation of the UIWT evaluation identified a number of factors which should be formally addressed in instruction for team fire.

- (a) How large is a fire sector?
- (b) Enemy tactics (i.e., Warsaw Pact, Vietnamese)
- (c) Ammunition rationing
- (d) Change in fire team tactics if a member is made inoperable (i.e., gun jams or casualty)

There is presently no training which accomplishes the objectives that the UIWT addresses (e.g., fire team training, on board ship practice and training, and providing realistic combat scenarios which require application of proper aiming techniques). Since the Marine Corps deems these objectives to be important, it is recommended that the UIWT effort be funded and preparation be made for contractor production of the system.

In addition, development of the UIWT's capability to simulate other infantry weapons such as the TOW, DRAGON, motars, etc., should continue. Also, any formalized instruction should include techniques for firing at moving targets (including leading, firing into brush cover, and firing through smoke). It is possible for an instructor to forget to cover many of these important points when it is presented in an informal manner.

The UIWT has a counter which allows the number of rounds fired during a session to be accurately assessed. This counter showed that over 40,000 electronic rounds were used for the evaluation. Presently, costs for live M-16 ammunition are approximately nine cents per round. At this rate, the UIWT accomplished an ammunition cost savings of nearly \$4,000. This dollar figure if applied to a year's worth of training, would be a significant savings. At this rate the UIWT would pay for itself in a year. If the transportation fuel costs were added to the ammunition cost the savings would be even more impressive.

REFERENCES

1. CT 8101 Printing Terminal User's Instruction Manual, Tektronix, Inc., Beaverton, Oregon, May 1977. A manual covering the Texas Instruments Model 743 Electronic Data Terminal.
2. Marshall, Albert H., Herbert C. Towle, Bon F. Shaw, "Electro-Optic Infantry Weapons Trainer," 10th Annual Electro-Optic/Laser 78 Conference and Exposition, September 1978 Sponsored by the Laser Institute of America (LIA).
3. McCracken, D. D., "A Guide to P./M Programming For Microcomputer Applications," Addison-Wesley Publishing Company, Reading, Mass, 1978.
4. MCS-48 Microcomputer User's Manual, Intel Order Number 9800270D, July 1978.
5. PL/M-80 Programming Manual, Intel Order Number 98-268, Intel Corporation, Santa Clara, 1976.
6. System 80/20-4 Microcomputer Hardware Reference Manual, Intel Order Number 980484A, Intel Corporation, Santa Clara, 1977.
7. UPI-41 User's Manual (Preliminary), Intel Order Number 9800504A.
8. Universal Infantry Weapons Trainer Evaluation, Report No. N-2411P11-A/2-79, N-241 Instructional Concepts Development Branch.
9. Maj. J. Fagersten, Maj. G. Bishop, Mr. M. Wasson and Ms. E. Redden, Concept Evaluation Test Squad Weapons Analytical Trainer, Final Report, U.S. Army Infantry School, Ft. Benning, GA.

APPENDIX A

FUNCTIONAL DESCRIPTION OF CONSOLE SWITCHES

Figure A-1 displays the Instructor's Console. The right side display screen presents printed columns showing trainees results each time they fire, and indicates by a red light which trainee is doing the poorest. The miniature toggle switches must be in the up (or on) position for any trainee participating in the exercise. The left side display screen presents positions of movie targets and indicates where trainee is aiming and where his round has been fired in relation to where the target appeared.

On the main, center panel, the instructor operated switches are arranged and labeled in groups by function. They are:

1. Main power (single switch)
2. Microcomputer (three switches)
3. Computer Voice (two switches)
4. Projector (two switches)
5. Motor (three position rotary switch)
6. Boresite (single switch)
7. Recoil (single switch)
8. Score display (single switch)
9. Audio Communications (eleven switches)
10. Weapon motion system (eleven switches)

The operation of these switches follows:

MAIN POWER - Applies the ac power to entire console.

MICROCOMPUTER - "On" applies operating voltages to computer.

"Reset" clears all previous data from computer and asks if individual identification of each trainee is required. If none is required, depress any key except Y (For "yes"). Computer will signify system is ready by printing out "let's start".

"Start print" starts the hard copy printout process printing trainees results and analysis.

If individual scoring identification is required, depress the letter Y on typewriter. Computer will ask you for today's date, session number and to enter in name of first trainee, then instruct you to identify the others in same manner. NOTE: Names of trainees must be limited to those containing no more than six letters.

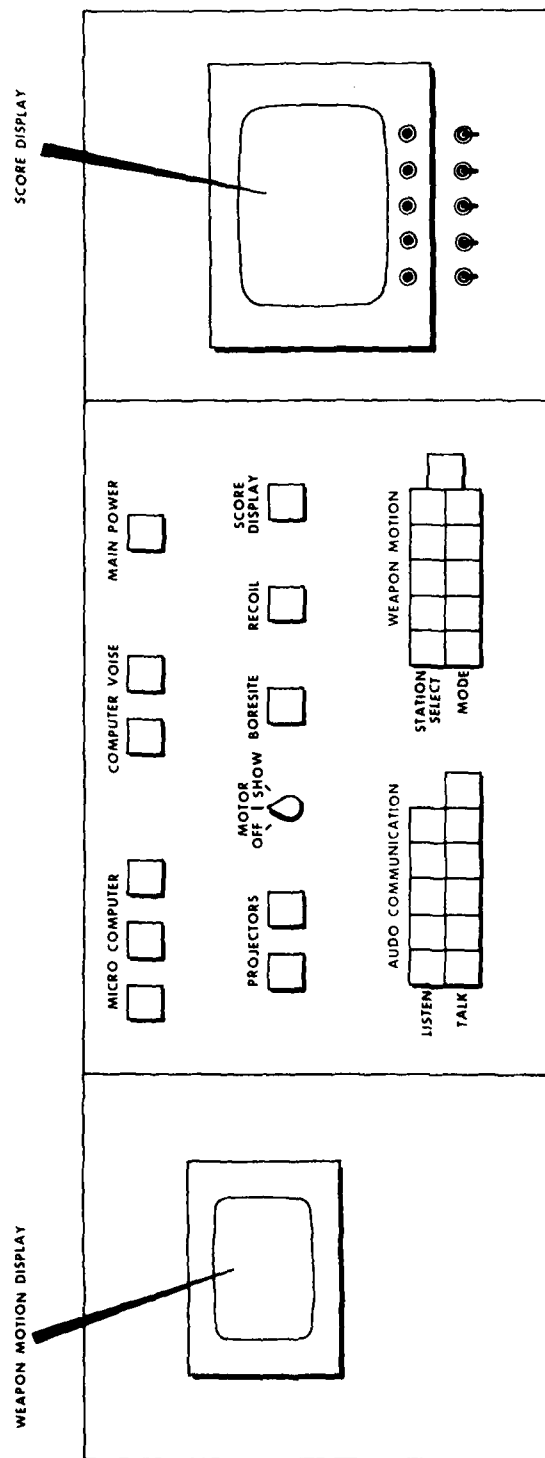


Figure A-1. Instructor's Console - Front Panel Control Locations

COMPUTER VOICE - Permits instructor to activate the computer voice carried to trainee headphones. The voice reset button should be pushed whenever the computer voice is turned on.

PROJECTORS - "Locked-Unlocked" switch either locks or unlocks synchronous motors of IR and visual projectors together so that target frame and visual frame coincide, or permits instructor to operate projectors independently during alignment. For normal operation the switch should remain locked.

"Start Movie" physically starts the movie if motor switch is not in off position.

MOTOR - This three position rotary switch performs the function of applying ac power first to projector motor and then to projector lamps (this procedure is conventional in all movie projectors to conserve lamp life).

BORESIGHT - This switch allows instructor to override the "target present" signal delivered to computer by the projector, so that trainees can fire at the target box instead of the movie screen. When lower half of this switch is illuminated the system is in normal (movie screen) position.

RECOIL - This switch gives instructor the option to conduct exercise with or without rifle recoil.

SCORE DISPLAY - This switch clears the right side screen display.

AUDIO COMMUNICATIONS - This bank of eleven switches permits the instructor to listen in on the headphone of any trainee, and talk to any one or all trainees.

WEAPON ACTION SYSTEM - This bank of eleven switches allows the instructor to view, on the left side display screen, exactly where any, or all, trainees are aiming and firing in relation to where targets appeared on movie screen. The upper bank of switches selects the trainee to be observed. The lower bank of switches either energizes a trainee's laser continually or only when he had fired his rifle. These switches supply operating power to the laser attached to each rifle, and must be held pushed in for the laser to operate.

APPENDIX B

TEST EQUIPMENT

1. LASER CHECKER

The laser checker test box, Figure B-1, allows instructor to make a go-on-go check of the SWAT tracking laser which is attached to the simulated rifle. Referring to the schematic, Q1 is a silicon photodiode which responds to IR energy from the rifle laser. When the rifle trigger is pulled, with the laser positioned a few inches from the detector, Q1 detects the infra red pulse, and delivers an output signal to Q2.

Q2 is a high gain amplifier whose output is a sharply rising positive pulse that provides the gating signal required to turn on silicon controlled rectifier Q3, placing its anode at ground potential, and allowing capacitor C3 to begin charging up toward Vcc through resistor R5, and energizes the Sonalert alarm producing an audio tone of approximately 2 KHz.

Q4 is a unijunction transistor oscillator which is enabled whenever capacitor C3 is returned to ground. As capacitor C3 starts to charge toward Vcc it produces an exponentially rising DC voltage at the emitter junction of Q4. When this voltage reaches the breakdown point for this particular unijunction, Q4, conducts heavily, shorting emitter to ground thereby allowing the charge on C3 to dissipate, lifting the anode of Q3 off of ground which turns off the SCR, disabling the audio alarm, and the entire circuit is again ready for recylcing.

Power for the laser checker is externally provided via binding posts. Any battery voltage from six to thirty volts may be used. The audio alarm sound level varies with battery voltage. Fifteen volts provides more than adequate sound level for an average room.

2. RIFLE CHECKER

By using an ordinary penlite flashlight as a source of light energy, and directing the light into the lens of the rifle receiver, the rifle checker test box allows the instructor to confirm that the four quadrant detector in the rifle is functioning, the "Full Clip" feature of the SWAT system is operating correctly, that the rifle mode select switch, and rifle trigger switch, are also functioning properly. In addition, the SWAT laser can be tested using the laser checker in conjunction with the rifle checker.

Detector Test: Referring to the schematic of the rifle checker, Figure B-2, connections to the rifle are made via a T&B connector identical to that with the SWAT system. Connector pin numbers are identified on the schematic. IC1 is a quad comparator which establishes the signal strength reference level of the IR energy received. Each of the four comparators parameters are identical. The desired reference level is established on the non-inverting inputs of IC1. The rifle detector outputs appear as signal at the inverting inputs

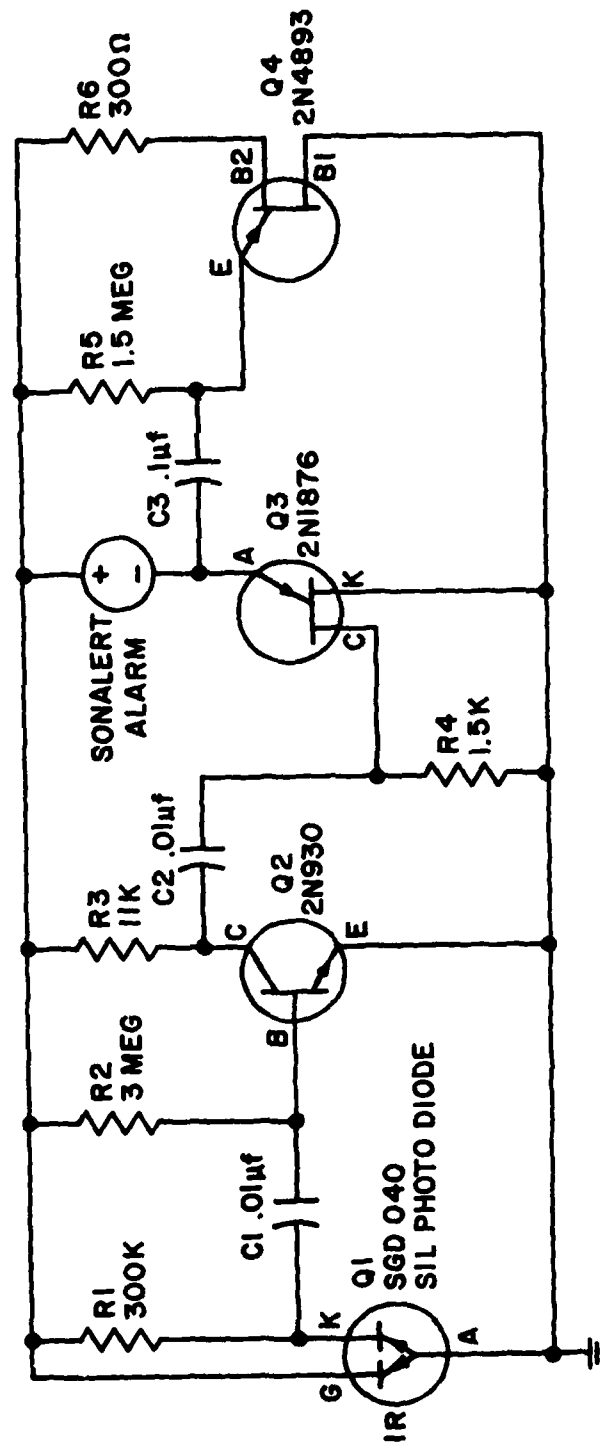
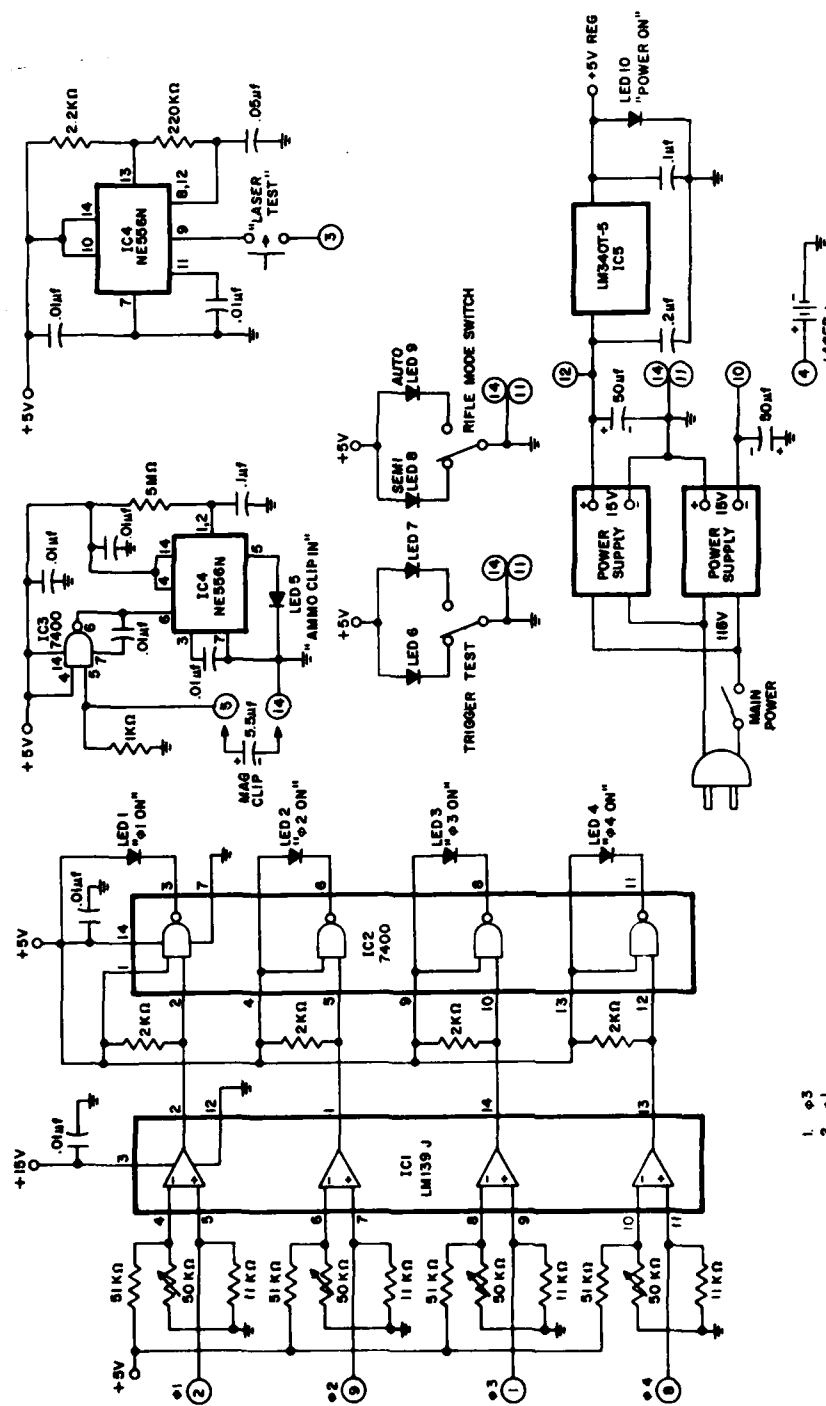


Figure B-1. Laser Defector Box



1. $\phi 3$
 2. $\phi 1$
 3. LASER TEST OUTPUT
 4. +45V (INTERNAL)
 5. CLIP
 6. TRIGGER N.C.
 7. MODE "SEMI"
 8. $\phi 4$
 9. $\phi 2$
 10. -15 VOLTS
 11. GND & CLIP COMMON
 12. +15 VOLTS
 13. TRIGGER N.O.
 14. COMMON TRIG. CLIP, MODE
 15. MODE "AUTOMATIC"

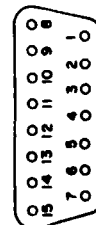


Figure B-2. Rifle Checker

of IC1. When this input signal is of an amplitude equal to the reference level established via the variable resistor, the outputs of IC1 rise to a logical "HIGH". This output "HIGH" signal appears on one input to IC2 which is a quad NAND gate. Since the remaining NAND inputs are permanently "HIGH", IC2 produces a logical "LOW" at its output, providing a path to ground for that particular LED, allowing it to light up, indicating that the SWAT rifle detector is operational. By pointing the flashlight beam into the rifle receiver lens and moving it about, it is possible to observe that all four quadrants of SWAT's rifle detector are indeed functioning.

Ammo Magazine Test: One half of IC4 is a timer arranged to produce one pulse of approximately five second duration whenever it receives an input, thereby testing the "full ammo magazine" feature of SWAT. A "full magazine" is simulated by one whose internally-mounted capacitor has been charged. A separate, partitioned-holding/charging tray which accommodates up to 30 magazines is provided with the instructor's console. In future modifications, this holding/charging tray can be incorporated into the console. Referring to the schematic, when a (charged capacitor) full magazine is inserted in the rifle, the capacitor discharges into one gate of IC3, a NAND, providing a negative going signal as input to timer IC4. When IC4 turns on its output goes "HIGH" allowing LED 5 to light for approximately five seconds. If an "empty magazine" (i.e. one which has expended its full thirty rounds) is inserted into the SWAT rifle, it cannot energize the circuit and LED 5 will not illuminate.

Laser Test: This test is performed by one half of IC4. The circuit is a free-running oscillator delivering sharp positive pulses of approximately one microsecond duration at a PRR of 5 KHz. This output is directly connected to the laser input via the normal connecting cable. When the "laser test" button on the rifle checker box is depressed, the output of oscillator IC4 is allowed to trigger the laser, and the laser will emit IR energy.

By setting up the laser checker* box several inches from the laser, an audio alarm tone indicates correct operation of the laser each time the test button is depressed. *Circuit operation is described in Paragraph 1 of this Appendix B.

Rifle Trigger and Mode Switch Test: Correct physical functioning of the SWAT rifle trigger is verified by observing the test lights LED 6 and LED 7. LED 6 will normally be on. When the rifle trigger is pulled LED 6 goes off for a fraction of time, LED 7 comes on during that instant, and as the trigger is pulled all the way LED 6 comes back on and remains on.

Rifle Mode Switch Test: The SWAT rifle mode select switch is tested by observing the operation of LED 8 and LED 9. In the semi-automatic mode, LED 8 is on and in the automatic mode LED 9 is on.

An internal dual fifteen volt power supply provides operating voltages for the SWAT as well as for the pre-amp receiver in the SWAT rifle. A single miniature dry cell forty-five volt battery within the test box supplies operating power for the SWAT laser.

3. RIFLE SUBSTITUTION BOX

Using the rifle substitution box, it is possible to simulate the operation of a SWAT rifle on the instructor's console. It can simulate a rifle being fired either in automatic or semi-automatic mode, and can simulate the target information normally received by the rifle's quadrant detector from the projected movie image.

The rifle substitution box is connected to the instructor's console via a T&B connector identical to that on a SWAT rifle. Power for the test box is derived from the console. Operation of the trigger switch and the mode switch are obvious. Capacitor C1 simulates the capacitor which is internally mounted inside of each magazine. It remains in the charged state via switch SW4 as long as the test box power switch is in the "on" position. When the counting circuit in the SWAT electronics indicates to the computer that thirty rounds have been expended by allowing no more shots, the insertion of a new magazine is simulated by momentarily engaging switch SW1.

Target information is simulated by the status of four switches, SW5, SW6, SW7, and SW8. IC1 is a free-running oscillator delivering a square wave at a PRR of 96 Hz (which is the rate at which the projected IR Target energy is chopped). The output of the oscillator is delivered to the console via switches SW5 through 8 as simulated rifle detector signals.

Referring to the schematic, Figure B-3, the adopted convention for the four quadrant rifle detector is shown. Switches may be independently thrown to either ground (low) or to the 96 Hz positive pulse output of IC1, constituting either a true or a false logic signal to the SWAT console. Any one of ten possible combinations of hit or near misses can be duplicated by these four switches. For example: Switches SW3, SW4 "Low" and switches SW1, SW2 "High" will be recorded by the computer as a "low right" signal from the simulated rifle. Similarly, SW1, SW4, SW2 "High" and SW3 "Low" would simulate a SWAT rifle detector condition where IR energy is centered on those quadrants. This information would be interpreted by the computer as a trainee having fired "Low" at the target.

4. BORESIGHT BOX

The boresight box, Figure B-4, allows the instructor to initially check the closeness of SWAT rifle boresight alignment. A free running oscillator IC1 with a PRR of 96 Hz delivers input pulses to a Darlington amplifier which consists of Q1 and Q2. The Darlington amplifier pulses a high intensity incandescent lamp at 96 Hz. The visible portion of light is filtered out so the rifle is aimed at the black-outlined aiming pip on the box, fires the weapon, receives audio

feedback results via his headphones, and adjusts rifle sights for accurate alignment of front and rear sights.

Power for the boresight box is supplied externally via one small, 12 volt rechargeable sealed-gel battery. The boresight box can also be utilized as a marksmanship target. The instructor simply enables the front panel "target present" switch allowing the SWAT console to disregard target information from the movie screen, and instead, to accept target information from the boresight box.



Figure B-3. Rifle Substitution Box

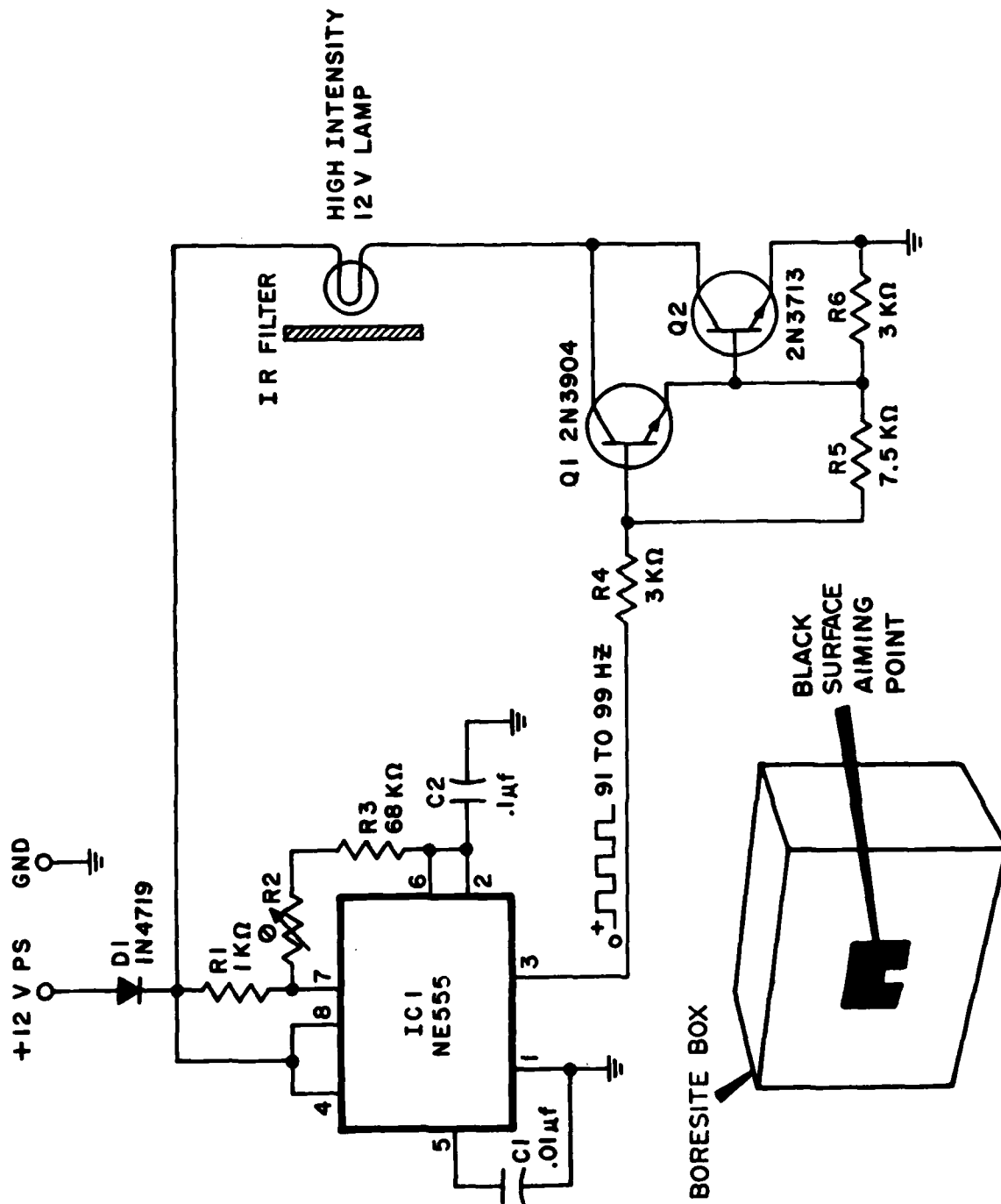


Figure B-4. Boresight Box

-LOCATE F1 UIWT.TMP TO F1 UIWT SYMBOLS LINES MAP PRINTC F1 UIWT.PRT

SYMBOL TABLE OF MODULE UIWT
READ FROM FILE F1 UIWT.TMP
WRITTEN TO FILE F1 UIWT

VALUE TYPE SYMBOL

	MOD	MAINUIWTMODULE
391FH	SYM	MEMORY
0000H	SYM	CLEARACCUMULATOR
3800H	SYM	TRAINING
3801H	SYM	JK
380AH	SYM	TRAIN
00BEH	SYM	TURKEYTEST
3802H	SYM	TARGETDOWNTIME
3804H	SYM	TARGETEXPECTED
3805H	SYM	TFLAG
3806H	SYM	REALGONE
3807H	SYM	TURKEY
3808H	SYM	TARGETLINE
0004H	SYM	TESTEDHARDCHECK
000EH	SYM	UIWTPROGRAM
000EH	SYM	INITIALIZE
0027H	SYM	SESSION
003CH	SYM	ACTION
0043H	SYM	GOTONE
004AH	SYM	NEWONE
0064H	SYM	GOODLATH
006AH	SYM	NOTHARGET
0073H	SYM	NOACTION
007AH	SYM	TGONE
008CH	SYM	WAITONE
0096H	SYM	RECORDGONE
00AFH	SYM	ENDING
00BEH	LIN	41
00BEH	LIN	42
00CAH	LIN	43
00D9H	LIN	44
00EBH	LIN	45
00F6H	LIN	46
00FEH	LIN	47
0103H	LIN	48
0001H	LIN	55
0004H	LIN	56
000BH	LIN	57
000EH	LIN	59
000EH	LIN	60
0011H	LIN	61
0014H	LIN	62
0017H	LIN	63
001AH	LIN	64
001DH	LIN	65
0027H	LIN	66
002EH	LIN	67
003CH	LIN	68
003CH	LIN	69
0043H	LIN	70
0043H	LIN	71
004AH	LIN	72
004AH	LIN	73
0050H	LIN	74
005AH	LIN	75
0058H	LIN	76
005CH	LIN	77

0060H	LIN	78
0064H	LIN	79
0067H	LIN	80
006AH	LIN	81
006AH	LIN	82
006DH	LIN	83
0070H	LIN	84
0070H	LIN	85
0073H	LIN	86
0073H	LIN	87
007AH	LIN	88
007AH	LIN	89
0080H	LIN	90
0085H	LIN	91
0085H	LIN	92
008CH	LIN	93
008CH	LIN	94
008FH	LIN	95
0096H	LIN	96
0096H	LIN	97
009BH	LIN	98
009EH	LIN	99
00A9H	LIN	100
00ACH	LIN	101
00ACH	LIN	102
00ACH	LIN	103
00ACH	LIN	104
00AFH	LIN	105
00B2H	LIN	106
00B5H	LIN	107
00B9H	LIN	108
00BCH	LIN	109
	MOD	STARTUPMODULE
391FH	SYM	MEMORY
13FCH	SYM	INTERRUPTCALL
380AH	SYM	HISTORY
380FH	SYM	NAME
383CH	SYM	DATE
3848H	SYM	IDNUMBER
0104H	SYM	WHEN
0114H	SYM	UIWID
0129H	SYM	IDENT
013CH	SYM	QUERY
0152H	SYM	TRAINEES
016CH	SYM	ONRIFLE
384FH	SYM	X
3850H	SYM	I
3851H	SYM	J
3852H	SYM	K
3853H	SYM	JK
3854H	SYM	IDFLAG
0176H	SYM	PORTSET
017FH	SYM	SETDATA
3855H	SYM	POINTER
3857H	SYM	LENGTH
3858H	SYM	VALUE
3859H	SYM	FINAL
0139H	SYM	LOOP
01B7H	SYM	SIMULATERIFLES
385BH	SYM	RUNTYPE
01BFH	SYM	STARTUP
024CH	SYM	SETINT
02A2H	SYM	GETID
02A7H	SYM	CLRDATE
02C2H	SYM	GETDATE
032AH	SYM	GETIDNUMBER

LOC	SYM	LINE NAME
0176H	LIN	22
0176H	LIN	23
017AH	LIN	24
017EH	LIN	25
017FH	LIN	29
018CH	LIN	31
0199H	LIN	32
01A5H	LIN	33
01ACH	LIN	34
01B3H	LIN	35
01B6H	LIN	36
01B7H	LIN	41
01B7H	LIN	42
01BBH	LIN	43
01BEH	LIN	44
01BFH	LIN	46
01BFH	LIN	47
01CAH	LIN	48
01D5H	LIN	49
01E0H	LIN	50
01EBH	LIN	51
01F6H	LIN	52
01FBH	LIN	53
0200H	LIN	54
0205H	LIN	55
020AH	LIN	56
020FH	LIN	57
0214H	LIN	58
0217H	LIN	59
021BH	LIN	60
021FH	LIN	61
0223H	LIN	62
0231H	LIN	63
0239H	LIN	64
023DH	LIN	65
0241H	LIN	66
0248H	LIN	67
0249H	LIN	68
024CH	LIN	69
0250H	LIN	70
0254H	LIN	71
0258H	LIN	72
025UH	LIN	73
0262H	LIN	74
0267H	LIN	75
026DH	LIN	76
0272H	LIN	77
0278H	LIN	79
027EH	LIN	80
0284H	LIN	81
028BH	LIN	82
0290H	LIN	83
0295H	LIN	84
029HH	LIN	85
02A2H	LIN	86
02A2H	LIN	87
02A7H	LIN	88
02B2H	LIN	89
02B8H	LIN	90
02B9H	LIN	91
02C2H	LIN	92
02D0H	LIN	93
02D6H	LIN	94
02D9H	LIN	95

02E3H	LIN	96
02EAH	LIN	97
02F7H	LIN	98
02FEH	LIN	99
0303H	LIN	100
0308H	LIN	101
030DH	LIN	102
0318H	LIN	103
031EH	LIN	104
0323H	LIN	105
0326H	LIN	106
032BH	LIN	107
0339H	LIN	108
033FH	LIN	109
0347H	LIN	110
034CH	LIN	111
0353H	LIN	112
0360H	LIN	113
0367H	LIN	114
036CH	LIN	115
0371H	LIN	116
0376H	LIN	117
0381H	LIN	118
0387H	LIN	119
0395H	LIN	120
039BH	LIN	121
03A4H	LIN	122
03A9H	LIN	123
03AEH	LIN	124
03B3H	LIN	125
03B6H	LIN	126
03C7H	LIN	127
03D5H	LIN	128
03DBH	LIN	129
03E3H	LIN	130
03E8H	LIN	131
0403H	LIN	132
040AH	LIN	133
0411H	LIN	134
0416H	LIN	135
042BH	LIN	136
0430H	LIN	137
0437H	LIN	138
0437H	LIN	139
043FH	LIN	140
0442H	LIN	141
	MOD	TIMERMODULE
391FH	SYM	MEMORY
385CH	SYM	LSTIMEBYTE
385DH	SYM	MSTIMEBYTE
385EH	SYM	TIME
385EH	SYM	LOWTIMEBYTE
385FH	SYM	HIGHTIMEBYTE
0443H	SYM	TIMERSTART
045CH	SYM	TTYTIMER
0469H	SYM	VOTRAXTIMER
0476H	SYM	CLOCKREAD
0443H	LIN	8
0443H	LIN	9
0447H	LIN	10
044BH	LIN	11
044FH	LIN	12
0453H	LIN	13
0457H	LIN	14
045BH	LIN	15
045BH	LIN	16

0450H LIN	17
0460H LIN	18
0464H LIN	19
0468H LIN	20
0469H LIN	21
0469H LIN	22
046DH LIN	23
0471H LIN	24
0475H LIN	25
0476H LIN	26
0476H LIN	27
047AH LIN	28
047FH LIN	29
0484H LIN	30
0488H LIN	31
MOD	RIFLEDATAMODULE
391FH SYM	MEMORY
3860H SYM	MESSAGE
0498H SYM	SCORIT
3861H SYM	SHOTLOCATION
3862H SYM	FILE
0488H SYM	DECODE
3863H SYM	FIRSTSHOT
3868H SYM	SCORE
3868H SYM	ADDER
38A9H SYM	SPEED
38B8H SYM	TOTALTIME
38BAH SYM	DELTATIME
38BCH SYM	FILE
38BDH SYM	RIFLE
38BEH SYM	FPTR
38BFH SYM	SHOTFLAG
38C0H SYM	RIFLEID
04B7H SYM	UPISTROBE
04C0H SYM	TARGETAVAILABLE
38C1H SYM	TARGET
04C6H SYM	RIFLESHOT
04CCH SYM	GETRIFLEDATA
38C2H SYM	SHOTDATA
04CCH SYM	UNRESOLVED
04DAH SYM	WHOSHOT
0521H SYM	HOWQUICK
05D4H SYM	OKDATA
05FDH SYM	VOTRAXMESSAGES
068CH SYM	TURKEYDATA
06F2H SYM	WHOFILEDTOSHOOT
076AH SYM	SHOWWURST
38C3H SYM	BADWORD
38C4H SYM	BADNEWS
076AH SYM	HOWBAD
07D1H SYM	HUNTWORST
0498H LIN	6
049EH LIN	8
04B6H LIN	9
04B7H LIN	31
04B7H LIN	32
04BBH LIN	33
04BFH LIN	34
04C0H LIN	35
04C0H LIN	37
04C5H LIN	38
04C6H LIN	39
04C6H LIN	40
04C6H LIN	41
04CBH LIN	42
04CTH	11

AD-A091 077 OFFICE OF THE PROJECT MANAGER FOR TRAINING DEVICES 0--ETC F/6 5/9
SQUAD WEAPONS ANALYTICAL TRAINER (SWAT) M-16 VERSION.(U)

UNCLASSIFIED JUL 80 A MARSHALL, B SHAW, H TOWLE
PM TRADE-RE-0005

NL

2.1

A A

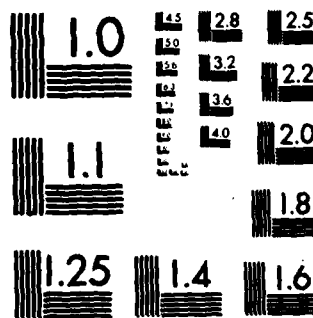
END

DATE

THRU

1-8

DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

0400H LIN	44
040CH LIN	46
0405H LIN	47
040AH LIN	48
04E1H LIN	49
04F3H LIN	50
04F6H LIN	51
0500H LIN	52
0503H LIN	53
050CH LIN	54
0513H LIN	56
0521H LIN	57
0521H LIN	58
052DH LIN	59
053CH LIN	60
054EH LIN	61
056DH LIN	62
057DH LIN	63
0588H LIN	64
0588H LIN	65
0588H LIN	66
0593H LIN	67
059BH LIN	68
059FH LIN	69
05A4H LIN	70
05A8H LIN	71
05B2H LIN	72
05B6H LIN	73
05BAH LIN	74
05BFH LIN	75
05C8H LIN	76
05D4H LIN	77
05D4H LIN	78
05EEH LIN	79
05FAH LIN	80
05FDH LIN	81
060DH LIN	82
0615H LIN	83
061DH LIN	84
0625H LIN	85
0625H LIN	86
062AH LIN	87
062FH LIN	88
0632H LIN	89
063AH LIN	90
063AH LIN	91
063FH LIN	92
0644H LIN	93
0647H LIN	94
064FH LIN	95
064FH LIN	96
0654H LIN	97
0659H LIN	98
065CH LIN	99
0664H LIN	100
0664H LIN	101
0669H LIN	102
066EH LIN	103
0671H LIN	104
0685H LIN	105
0685H LIN	106
068CH LIN	108
06A0H LIN	109
06AAH LIN	110
06ADH LIN	111
06B2H LIN	112

06B7H	LIN	112
06B7H	LIN	114
06C7H	LIN	116
06D7H	LIN	117
06E1H	LIN	118
06E4H	LIN	119
06E9H	LIN	120
06E9H	LIN	121
06EEH	LIN	122
06F1H	LIN	123
06F2H	LIN	124
06F2H	LIN	125
0700H	LIN	126
070AH	LIN	127
0719H	LIN	129
072DH	LIN	130
0737H	LIN	131
073AH	LIN	132
073FH	LIN	133
0748H	LIN	134
074DH	LIN	135
0752H	LIN	136
0757H	LIN	137
0757H	LIN	138
075EH	LIN	139
0769H	LIN	140
076AH	LIN	141
076AH	LIN	144
0778H	LIN	145
0783H	LIN	146
0798H	LIN	147
07C0H	LIN	148
07C7H	LIN	149
07CCH	LIN	150
07D1H	LIN	151
07DFH	LIN	152
07F8H	LIN	153
0808H	LIN	154
0821H	LIN	156
0829H	LIN	157
0832H	LIN	158
0832H	LIN	159
0839H	LIN	160
083EH	LIN	161
	MOD	CONSOLEMODULE
391FH	SYM	MEMORY
083FH	SYM	HELLO
38C9H	SYM	I
38CAH	SYM	Y
38CBH	SYM	GOS
0851H	SYM	TTYSET
0869H	SYM	TTYRES
0871H	SYM	VOTRAXSET
087AH	SYM	VOTRES
0891H	SYM	CIN
0891H	SYM	RXRDY
08A1H	SYM	TXRDY
08MFH	SYM	COUT
38CCH	SYM	ITEM
08C2H	SYM	BITDUMP
08C8H	SYM	PRNTNUM
0912H	SYM	PRINT
38CDH	SYM	POINTER
38CFH	SYM	FINAL
0925H	SYM	LOOP
0943H	SYM	GREETING

0051H LIN	2
0051H LIN	10
0053H LIN	11
0059H LIN	12
005DH LIN	13
0060H LIN	14
0064H LIN	15
0068H LIN	16
0069H LIN	17
0069H LIN	18
006DH LIN	19
0070H LIN	20
0071H LIN	21
0071H LIN	22
0075H LIN	23
0079H LIN	24
007AH LIN	25
007AH LIN	26
007EH LIN	27
0082H LIN	28
0086H LIN	29
008AH LIN	30
008DH LIN	31
0090H LIN	32
0091H LIN	33
0091H LIN	34
0099H LIN	35
009CH LIN	36
00A1H LIN	37
00A1H LIN	38
00A1H LIN	39
00ABH LIN	40
00AEH LIN	41
00AFH LIN	42
00B3H LIN	44
00B9H LIN	45
00BCH LIN	46
00C1H LIN	47
00C2H LIN	48
00C2H LIN	49
00C7H LIN	50
00C8H LIN	51
00C8H LIN	52
00CDH LIN	53
00DBH LIN	54
00EDH LIN	55
00F2H LIN	56
00F9H LIN	57
0900H LIN	58
0907H LIN	59
090CH LIN	60
0911H LIN	61
0912H LIN	62
0918H LIN	64
0925H LIN	65
0931H LIN	66
0938H LIN	67
093FH LIN	68
0942H LIN	69
0943H LIN	70
0943H LIN	71
0949H LIN	72
094CH LIN	73
094DH LIN	74
MOD	RESULTSMODULE
391FH SYM	MEMORY

3801H	SYM	LEADTIME
3804H	SYM	I
3805H	SYM	Z
3806H	SYM	SUMSHOTS
3807H	SYM	NEARMISSES
3808H	SYM	AVGTIME
0A24H	SYM	CONVRT
380AH	SYM	HEX
094EH	SYM	RIFLEID
0956H	SYM	TOTALSHOTS
0964H	SYM	RIFLEHIT
096BH	SYM	RIFLEMISS
0974H	SYM	RIFLELOW
097BH	SYM	RIFLELOWRIGHT
0988H	SYM	RIFLERIGHT
0991H	SYM	RIFLEHIGHRIGHT
099FH	SYM	RIFLEHIGH
09A7H	SYM	RIFLEHIGHLEFT
09B4H	SYM	RIFLELEFT
09BCH	SYM	RIFLELOWLEFT
09C8H	SYM	RIFLETURKEY
09D4H	SYM	RIFLETARGETIGNORED
09E6H	SYM	BLANK
09E9H	SYM	HOWMANYSHOTS
09FBH	SYM	AVERAGETIME
0A0AH	SYM	UNITS
0A12H	SYM	YOURSCORE
0A6EH	SYM	PRESENTRESULTS
0A9DH	SYM	ONERIFLERESULTS
0AC2H	SYM	TYPEIT
0B24H	SYM	SUM
0B5CH	SYM	SUM2
0A24H	LIN	24
0A28H	LIN	26
0A36H	LIN	27
0A56H	LIN	28
0A66H	LIN	29
0A6DH	LIN	30
0A6EH	LIN	51
0A6EH	LIN	52
0A6FH	LIN	53
0A72H	LIN	54
0A78H	LIN	55
0A7EH	LIN	56
0A85H	LIN	58
0A8EH	LIN	59
0A91H	LIN	60
0A91H	LIN	61
0A97H	LIN	62
0A9DH	LIN	63
0AABH	LIN	64
0AC2H	LIN	65
0AC2H	LIN	66
0AC8H	LIN	67
0ACEH	LIN	68
0AD7H	LIN	69
0ADDH	LIN	70
0AE3H	LIN	71
0AFAH	LIN	72
0B00H	LIN	73
0B06H	LIN	74
0B0CH	LIN	75
0B12H	LIN	76
0B19H	LIN	77
0B1FH	LIN	78
0B24H	LIN	79

0B32H	LIN	80
0B50H	LIN	81
0B57H	LIN	82
0B5CH	LIN	83
0B6AH	LIN	84
0B88H	LIN	85
0B8FH	LIN	86
0B96H	LIN	87
0B99H	LIN	88
0B9FH	LIN	89
0BB6H	LIN	90
0BB9H	LIN	91
0BBFH	LIN	92
0BD2H	LIN	93
0BD5H	LIN	94
0BD8H	LIN	95
0BF2H	LIN	96
0BF5H	LIN	97
0BF8H	LIN	98
0C12H	LIN	99
0C15H	LIN	100
0C18H	LIN	101
0C32H	LIN	102
0C35H	LIN	103
0C38H	LIN	104
0C52H	LIN	105
0C55H	LIN	106
0C58H	LIN	107
0C72H	LIN	108
0C75H	LIN	109
0C78H	LIN	110
0C92H	LIN	111
0C95H	LIN	112
0C98H	LIN	113
0CB2H	LIN	114
0CB5H	LIN	115
0CB8H	LIN	116
0CD2H	LIN	117
0CD5H	LIN	118
0CD8H	LIN	119
0CF2H	LIN	120
0CF5H	LIN	121
0CF8H	LIN	122
0D12H	LIN	123
0D15H	LIN	124
0D18H	LIN	125
0D2EH	LIN	126
0D31H	LIN	127
0D37H	LIN	128
0D4FH	LIN	129
0D54H	LIN	130
0D7CH	LIN	131
0D84H	LIN	132
0D8FH	LIN	133
0D96H	LIN	134
0D9DH	LIN	135
0DA2H	LIN	136
0DA9H	LIN	137
0DAEH	LIN	138
0DB4H	LIN	139
0DB9H	LIN	140
0DBEH	LIN	141
0DC4H	LIN	142
0DC7H	LIN	143
0DCDH	LIN	144
0DDAH	LIN	145

0006H	LIN	146
0006H	LIN	147
0000H	LIN	148
00DEH	LIN	149
	MOD	FINALMODULE
391FH	SYM	MEMORY
38DBH	SYM	W
00DFH	SYM	FAST
0E09H	SYM	GOOD
0E22H	SYM	FAIR
0E48H	SYM	POOR
38DCH	SYM	TIMECREDIT
38DDH	SYM	N
0E7EH	SYM	COMMENT
0ED5H	SYM	HX2AS
38DEH	SYM	HEXADR
38E0H	SYM	DECADR
38E2H	SYM	N
38E3H	SYM	M
0F62H	SYM	COMPOSITE
0E66H	SYM	COMP
38E4H	SYM	OVERALL
38E6H	SYM	DECNUM
0E7EH	LIN	9
0E7EH	LIN	10
0E89H	LIN	12
0E8FH	LIN	13
0E94H	LIN	14
0E97H	LIN	15
0EA2H	LIN	17
0EA8H	LIN	18
0EADH	LIN	19
0EB0H	LIN	20
0EBBH	LIN	22
0EC1H	LIN	23
0EC6H	LIN	24
0EC9H	LIN	26
0ECFH	LIN	27
0ED4H	LIN	28
0ED4H	LIN	29
0ED5H	LIN	30
0EDFH	LIN	32
0EEDH	LIN	33
0EF5H	LIN	34
0F12H	LIN	35
0F24H	LIN	36
0F2BH	LIN	37
0F30H	LIN	38
0F4EH	LIN	39
0F5AH	LIN	40
0F5EH	LIN	41
0F61H	LIN	42
0F62H	LIN	43
0F62H	LIN	45
0FBBH	LIN	46
0FC1H	LIN	47
0FCDH	LIN	49
0FD6H	LIN	50
0FE4H	LIN	51
0FF1H	LIN	52
0FF8H	LIN	53
0FF8H	LIN	54
0FFDH	LIN	55
1002H	LIN	56
1007H	LIN	57
	MOD	INTERRUPT7

```

1FH SYM MEMORY
000H SYM INTERRUPTROUTINE
1000H LIN 3
100CH LIN 4
1011H LIN 5
1015H LIN 6

```

```

MEMORY MAP OF MODULE UIWT
READ FROM FILE :F1:UIWT.TMP
WRITTEN TO FILE :F1:UIWT
MODULE START ADDRESS 0001H

```

START	STOP	LENGTH	REL	NAME
0H	0H	1H	A	ABSOLUTE
1H	10AFH	10AFH	A	ABSOLUTE
13FCH	13FEH	3H	A	ABSOLUTE

ISIS-II IXREF, V1.1

INVOKED BY:

-IXREF :F1:*.IXI TITLE('UIWT VERSION 1.2 INTER-MODULE CROSS-REFERENCE') &

PRINT(:F1:UIWT.REF)

INTER-MODULE CROSS-REFERENCE LISTING

NAME ATTRIBUTES; MODULE NAMES

ADDER	STRUCTURE(5); RIFLEDATAMODULE RESULTSMODULE
AVGTIME	ADDRESS; RESULTSMODULE FINALMODULE
BITDUMP	PROCEDURE; CONSOLEMODULE STARTUPMODULE
CIN	PROCEDURE BYTE; CONSOLEMODULE TESTPROCMODULE STARTUPMODULE
CLOCKREAD	PROCEDURE ADDRESS; TIMERMODULE RIFLEDATAMODULE TESTPROCMODULE MAINUIMODULE
COMMENT	PROCEDURE; FINALMODULE RESULTSMODULE
COMPOSITE	PROCEDURE; FINALMODULE RESULTSMODULE
COUT	PROCEDURE; CONSOLEMODULE RIFLEDATAMODULE TESTPROCMODULE MAINUIMODULE RESULTSMODULE STARTUPMODULE FINALMODULE
DATE	BYTE(12); STARTUPMODULE RESULTSMODULE
DECIMAL	BYTE(3); RESULTSMODULE CONSOLEMODULE STARTUPMODULE
DELTATIME	ADDRESS; RIFLEDATAMODULE MAINUIMODULE
DONE	PROCEDURE; TESTPROCMODULE TESTMODULE
FAIL	PROCEDURE; TESTPROCMODULE
FILE	BYTE; RIFLEDATAMODULE RESULTSMODULE STARTUPMODULE FINALMODULE
FIRSTSHOT	BYTE(5); RIFLEDATAMODULE MAINUIMODULE STARTUPMODULE
FPTR	BYTE; RIFLEDATAMODULE
GETRIFLEDATA	PROCEDURE; RIFLEDATAMODULE MAINUIMODULE
GREETING	PROCEDURE; CONSOLEMODULE MAINUIMODULE
HISTORY	BYTE(5); STARTUPMODULE RIFLEDATAMODULE
H2AS	PROCEDURE; FINALMODULE
IDFLAG	BYTE; STARTUPMODULE RESULTSMODULE
IDNUMBER	BYTE(7); STARTUPMODULE RESULTSMODULE
INTERRUPTROUTINE	PROCEDURE; INTERRUPT7 STARTUPMODULE
IOTEST	PROCEDURE; TESTPROCMODULE TESTMODULE
LSTIMEBYTE	BYTE; TIMERMODULE
MSTIMEBYTE	BYTE; TIMERMODULE
NAME	STRUCTURE(5); STARTUPMODULE RESULTSMODULE
NEARMISSSES	BYTE; RESULTSMODULE FINALMODULE
PORTSET	PROCEDURE; STARTUPMODULE TESTPROCMODULE
PRESENTRESULTS	PROCEDURE; RESULTSMODULE MAINUIMODULE
PRINT	PROCEDURE; CONSOLEMODULE TESTPROCMODULE RESULTSMODULE STARTUPMODULE FINALMODULE
PRNTNUM	PROCEDURE; CONSOLEMODULE TESTPROCMODULE RESULTSMODULE
RANTST	PROCEDURE; ** UNRESOLVED ** TESTMODULE
REALGONE	BYTE; MAINUIMODULE STARTUPMODULE
RIFLE	BYTE; RIFLEDATAMODULE RESULTSMODULE FINALMODULE
RIFLESHOT	PROCEDURE BYTE; RIFLEDATAMODULE MAINUIMODULE
RONTST	PROCEDURE; ** UNRESOLVED ** TESTMODULE
SBCIN	PROCEDURE; ** UNRESOLVED ** TESTPROCMODULE
SCORE	STRUCTURE(5); RIFLEDATAMODULE RESULTSMODULE STARTUPMODULE FINALMODULE
SETDATA	PROCEDURE; STARTUPMODULE RIFLEDATAMODULE MAINUIMODULE
SHOTFLAG	BYTE; RIFLEDATAMODULE STARTUPMODULE
SHOWMORT	PROCEDURE; RIFLEDATAMODULE MAINUIMODULE
SPEED	STRUCTURE(5); RIFLEDATAMODULE RESULTSMODULE STARTUPMODULE FINALMODULE

STARTUP	PROCEDURE; STARTUPMODULE MAINUIMTMODULE
SUMSHOTS	BYTE; RESULTSMODULE FINALMODULE
TARGETAVAILABLE	PROCEDURE; BYTE; RIFLEDATAMODULE MAINUIMTMODULE
TARGETDOWNTIME	ADDRESS; MAINUIMTMODULE
TARGETEXPECTED	BYTE; MAINUIMTMODULE
TARGETTIME	ADDRESS; MAINUIMTMODULE RIFLEDATAMODULE
TEST	PROCEDURE; TESTMODULE MAINUIMTMODULE
TFLAG	BYTE; MAINUIMTMODULE RIFLEDATAMODULE STARTUPMODULE
TIME	ADDRESS; TIMERMODULE
TIMERSTART	PROCEDURE; TIMERMODULE TESTPROCMODULE MAINUIMTMODULE
TIMERTEST	PROCEDURE; TESTPROCMODULE TESTMODULE
TRAIN	BYTE; MAINUIMTMODULE STARTUPMODULE INTERRUPT7
TSTCHECK	BYTE; TESTPROCMODULE MAINUIMTMODULE
TTYRES	PROCEDURE; CONSOLEMODULE TESTPROCMODULE RESULTSMODULE
TTYSET	PROCEDURE; CONSOLEMODULE TESTPROCMODULE STARTUPMODULE
TTYTIMER	PROCEDURE; TIMERMODULE TESTPROCMODULE CONSOLEMODULE
TURKEY	BYTE; MAINUIMTMODULE RIFLEDATAMODULE STARTUPMODULE
TWRDY	PROCEDURE; CONSOLEMODULE TESTPROCMODULE MAINUIMTMODULE
UPISTROBE	PROCEDURE; RIFLEDATAMODULE TESTPROCMODULE STARTUPMODULE
USARTTEST	PROCEDURE; TESTPROCMODULE TESTMODULE
VOTRAXSET	PROCEDURE; CONSOLEMODULE TESTPROCMODULE
VOTRAXTIMER	PROCEDURE; TIMERMODULE TESTPROCMODULE CONSOLEMODULE
VOTRES	PROCEDURE; CONSOLEMODULE TESTPROCMODULE MAINUIMTMODULE
WHOFFAILEDTO SHOOT	PROCEDURE; RIFLEDATAMODULE MAINUIMTMODULE

MODULE DIRECTORY

MODULE NAME	FILE NAME	DISKETTE NAME
-------------	-----------	---------------

CONSOLEMODULE	CONSOL.PLM	UIMT.TST
FINALMODULE	FINAL.PLM	UIMT1.3
INTERRUPT7	INTER.PLM	UIMT1.3
MAINUIMTMODULE	MAINUI.PLM	UIMT1.2
RESULTSMODULE	RESULT.PLM	UIMT1.3
RIFLEDATAMODULE	RIFLE.PLM	UIMT1.3
STARTUPMODULE	START.PLM	UIMT1.3
TESTMODULE	TESTER.PLM	UIMT.TST
TESTPROCMODULE	TSTPRC.PLM	UIMT.TST
TIMERMODULE	TIMER.PLM	UIMT.TST

ISIS-II PL/M-88 V3.1 COMPILATION OF MODULE MAININTMODULE
OBJECT MODULE PLACED IN :F1:MAINUI.OBJ
COMPILER INVOKED BY: PLM88 :F1:MAINUI.PLM 1XREF DEBUG DATE (1 FEB 79)

/* THIS PROGRAM WAS WRITTEN BY H.C. TOMLE IN THE WINTER AND SPRING OF 1978 */
/* IT ASSUMES THE SYSTEM HAS BEEN RESET PRIOR TO RUNNING */
/* THIS MODIFICATION, WHICH RESULTS IN A CHANGE TO UIMT VERSION 1.2
RESULTS FROM THE REQUIREMENT TO SHOW THE WORST SHOOTER. */

```
1      MAININTMODULE:
      DO;

2  1      SHOWWORST: PROCEDURE EXTERNAL;
3  2      END SHOWWORST;

4  1      DECLARE CLEARACCUMULATOR BYTE AT (0) DATA(0AFH); /* AFH=>EXCLUSIVE OR
                                     ACCUMULATOR WITH ITSELF*/
5  1      TXYDY: PROCEDURE EXTERNAL; /* CHECKS USART TRANSMIT BUFFER FOR "EMPTY" */
6  2      END TXYDY;

7  1      VOTRES: PROCEDURE EXTERNAL; /* VOTRAX RESET SEE CONSOL MODULE */
8  2      END VOTRES;

9  1      COUT: PROCEDURE (ITEM) EXTERNAL; /* SENDS A BIT OUT THROUGH USART */
10 2      DECLARE ITEM BYTE;
11 2      END COUT;

12 1      DECLARE USARTCONTROL LITERALLY '0EDH'; /* ADDRESS OF USART CONTROL P. 3-32 */
          USARTARESET LITERALLY '40H'; /* RETURNS 8251 TO MODE INST. FORMAT
          REF. PAGE 3-43 */

13 1      STARTUP: PROCEDURE EXTERNAL; /* SET I/O FOR RIFLE ETC. */
14 2      END;

15 1      TARGET$AVAILABLE: PROCEDURE BYTE EXTERNAL; /* CHECKS FOR AN IR SPOT
          IT WILL RETURN 1 IF IR SPOT IS FOUND. */
16 2      END;

17 1      TIMER$START: PROCEDURE EXTERNAL; /* START 8253 REGISTERS */
18 2      END;

19 1      CLOCK$READ: PROCEDURE ADDRESS EXTERNAL; /* READS CLOCK FOR TARGET & SHOTS */
20 2      END;

21 1      RIFLE$SHOT: PROCEDURE BYTE EXTERNAL; /* WAITS FOR ANY RIFLE SHOT */
22 2      END;

23 1      GET$RIFLE$DATA: PROCEDURE EXTERNAL; /* READ & RECORD INPUT BYTE */
24 2      END;

25 1      SET$DATA: PROCEDURE (POINTER, LENGTH, VALUE) EXTERNAL;
26 2      DECLARE POINTER ADDRESS, (LENGTH, VALUE) BYTE;
27 2      END SET$DATA.
```

```

28 1      PRESENT#RESULTS: PROCEDURE EXTERNAL; /* OUTPUTS DATA TO CONSOLE */
29 2      END;

30 1      DECLARE TSTCHECK BYTE EXTERNAL;

31 1      TEST: PROCEDURE EXTERNAL;
32 2      END TEST;

33 1      DECLARE FOREVER LITERALLY 'WHILE 1';
34 1      DECLARE COUNTER#SET LITERALLY 'BERSFH';
35 1      DECLARE (TRAINING, JK) BYTE;
36 1      DECLARE TRAIN BYTE PUBLIC AT (. TRAINING);
37 1      MNO#FAILED#TO#SHOOT: PROCEDURE EXTERNAL; /* IDENTIFIES WHICH RIFLE MISSED
                                                    A TARGET SHOT OPPORTUNITY */

38 2      END;
39 1      GREETING: PROCEDURE EXTERNAL; /* PRINTS GREETING TO CONSOLE */
40 2      END;

41 1      TURKEY#TEST: PROCEDURE; /* CHECKS FOR A 1 SECOND TIME ELAPSE SINCE TARGET DOWN */
42 2      IF TARGET#DOWN#TIME >= CLOCK#READ THEN
43 2      DELTA#TIME = TARGET#DOWN#TIME - CLOCK#READ;
44 2      ELSE
45 2      DELTA#TIME = 1 + COUNTER#SET + TARGET#DOWN#TIME - CLOCK#READ;
46 2      IF DELTA#TIME > 200 THEN
47 2      TURKEY = 1;
48 2      ELSE TURKEY = 0;
49 2      END TURKEY#TEST;

49 1      DECLARE DELTA#TIME ADDRESS EXTERNAL;
50 1      DECLARE FIRST#SHOT(5) BYTE EXTERNAL;
51 1      DECLARE TARGET#DOWN#TIME ADDRESS PUBLIC;
52 1      DECLARE (TARGET#EXPECTED, TFLAG, REAL#ONE) BYTE PUBLIC;
53 1      DECLARE TURKEY BYTE PUBLIC;
54 1      DECLARE TARGET#TIME ADDRESS PUBLIC;

/* END OF DECLARATIONS */

/***** PROGRAM STARTS *****/

55 1      TEST#BOARD#CHECK: DO;
56 2      IF TSTCHECK THEN /* IF ROM EXTENDER BOARD IS NOT PRESENT, THE 00/20 TIMES
57 2      OUT AND WILL READ 00 INTO THE ACCUMULATOR */
58 2      CALL TEST; /*PERFORM 00/20 BOARD CHECK*/

59 2      END TEST#BOARD#CHECK;

60 1      UINT#PROGRAM: DO FOREVER;
61 2      INITIALIZE:
62 2      CALL TIMER#START;

63 2      CALL STARTUP;

64 2      CALL GREETING;

65 2      CALL TARDY; /* INSURES "GREETING" HAS BEEN COMPLETED */

```

```

64 2    CALL VOTRES; /* SWITCH TO VOTRAX LINE & CHANGE TO 9600 BAUD */
65 2    TARGET#DOWN#TIME = CLOCK#READ + 200; /* FOR TURKEY TEST */
66 2    SESSION:
        DO WHILE TRAINING; /* WE WILL END WITH AN "ESCAPE" FROM CONSOLE */
67 3        IF TARGET#AVAILABLE OR RIFLE#SHOT THEN
68 3        ACTION: DO; /* EITHER A RIFLE SHOT OR A TARGET AVAILABLE */
69 4            IF TARGET#AVAILABLE THEN
70 4            GOT#ONE: DO;
71 5                IF NOT TFLAG THEN
72 5                NEW#ONE: DO; /* A NEW TARGET HAS APPEARED */
73 6                    TARGET#TIME = CLOCK#READ;
74 6                    REAL#GONE = 0;
75 6                    TFLAG=1;
76 6                    TURKEY = 0;
77 6                    END NEW#ONE;
78 5                IF RIFLE#SHOT THEN
79 5                GOOD#DATA: CALL GET#RIFLE#DATA;
80 5                    END GOT#ONE;
81 4                ELSE /* ELSE THERE IS NO TARGET, BUT A SHOT WAS FIRED */
82 4                NOSTARGET: DO;
83 5                    CALL TURKEY#TEST;
84 5                    CALL GET#RIFLE#DATA;
85 4                    END NOSTARGET;
86 3                END ACTION;
87 4                ELSE /* ELSE THERE IS NEITHER A TARGET NOR SHOT */
88 4                NOACTION: DO;
89 5                    IF TFLAG THEN /* THAT IS, THERE WAS A TARGET ON LAST PASS */
90 5                    TGONE: DO;
91 6                        TARGET#DOWN#TIME = CLOCK#READ;
92 6                        TFLAG = 0;
93 5                        END TGONE;
94 4                        IF NOT REAL#GONE THEN
95 4                        WAIT#ONE: DO;
96 5                            CALL TURKEY#TEST; /* WE WILL HOLD REAL#GONE=0 FOR ONE SECOND */
97 5                            IF TURKEY THEN /* WE SCORE "TARGET IGNORED" ONLY IF REAL#GONE=1 */
98 5                            RECORD#GONE: DO;
99 6                                REAL#GONE = 1;
100 6                                CALL WHO#FAILED#TO#SHOOT;
101 6                                CALL SET#DATA( FIRST$SHOT,5,1); /* NEXT SHOT AT A TARGET WILL BE TIMED */
102 6                                CALL SHOW#WORST;
103 5                                END RECORD#GONE;
104 4                                END WAIT#ONE;
105 3                                END NOACTION;
106 2                                END SESSION;
107 2    ENDING:
108 2    CALL PRESENT#RESULTS;
109 2    CALL TXRDY;
110 2    OUTPUT(USART#CONTROL)=USART#RESET;
111 2    END UINT#PROGRAM;

```

189 1 END MAINMODULE;

MODULE INFORMATION:

CODE AREA SIZE = 0103H 259D
VARIABLE AREA SIZE = 000AH 10D
MAXIMUM STACK SIZE = 0006H 6D
167 LINES READ
0 PROGRAM ERROR(S)

END OF PL/M-88 COMPILATION

ISIS-11 PL/M-88 V3.1 COMPILATION OF MODULE STARTUPMODULE
 OBJECT MODULE PLACED IN :F1:START.OBJ
 COMPILER INVOKED BY: PLM88 :F1:START.PLM INREF DEBUG DATE (25 JUN 79)

```

#NOINTVECTOR

1      STARTUPMODULE: DO;

2 1    BIT$DUMP: PROCEDURE EXTERNAL; /* IN CONSOLEMODULE */
3 2      END BIT$DUMP;

4 1      DECLARE INTERRUPT7 LITERALLY '13FH',
          CALL$IT LITERALLY '0C3H';

5 1      DECLARE INTERRUPT$CALL STRUCTURE (JUMP BYTE, WHERE$TO ADDRESS) AT (INTERRUPT7)
          DATA (CALL$IT, INTERRUPT$ROUTINE);

6 1      DECLARE LIT LITERALLY 'LITERALLY',

          RESET$8212 LIT '0F8H',
          ENABLE$9311 LIT '18H',
          PORT3 LIT '0E6H',
          PORT6 LIT '0E6H',

          GROUP1 LIT '0E7H', /* 8255 CONTROL. REF. PAGE 3-68 OF 88/28 MANUAL */
          WORD1 LIT '92H', /* PORTS 1 & 2 INPUT, 3 OUTPUT ALL MODE 0. PAGE 4-13 -- */
          GROUP2 LIT '0EBH', /* PAGE 3-68 */
          WORD2 LIT '88H', /* PORTS 4,5 & 6 ALL OUTPUTS MODE 0 */
          USART$CONTROL LIT '0EDH', /* PAGE 3-48 */
          USART$RESET LIT '48H', /* INTERNAL RESET. PAGE 3-43 */
          MODE$SET LIT '0CEH', /* SETS 2 STOP BITS, 8 BITS, 16X. PAGE 3-38, -41 & 4-48 */
          /* USE 4FH FOR 1 STOP BIT, 8 BIT WORD & 64X */
          COMMAND$WORD LIT '27H', /* SETS TRANSMIT/RECEIVER READY,
          DATA TERMINAL READY, REQUEST TO SEND. */
          /* SEE PAGE 3-43 AND STEP 48 OF MDS MONITOR SHADOW PROM */

7 1      DECLARE DECIMAL(3) BYTE EXTERNAL;
8 1      DECLARE HISTORY(5) BYTE PUBLIC; /* THIS WILL
          IDENTIFY WHICH RIFLE WAS SHOT AT A PARTICULAR TARGET */
9 1      DECLARE SCORE(5) STRUCTURE (MISS BYTE, HIT BYTE, LOW BYTE, LOWRIGHT BYTE,
          RIGHT BYTE, HIGHRIGHT BYTE, HIGH BYTE, HIGHLEFT BYTE,
          LEFT BYTE, LOWLEFT BYTE, ERROR BYTE, TURKEY BYTE,
          TARGET$IGNORED BYTE) EXTERNAL;

10 1     DECLARE (SHOT$FLAG, TURKEY, FILE, TFLAG, REAL$NONE) BYTE EXTERNAL,
          FIRST$SHOT(5) BYTE EXTERNAL,
          SPEED(5) STRUCTURE (SHOTS BYTE, TIME$SUM ADDRESS) EXTERNAL;

11 1     DECLARE TRAIN BYTE EXTERNAL;
12 1     DECLARE ADICM1 LITERALLY '0D8H', /* REF PAGE -100 */
          ADICM2 LITERALLY '0D9H';
13 1     DECLARE OCM1 LITERALLY '7FH';

14 1     DECLARE CRLF LIT '0DH,0AH',
          NAME (5) STRUCTURE (LETTER(9) BYTE) PUBLIC,
          DATE (12) BYTE PUBLIC.
  
```

```

IDNUMBER (7) BYTE PUBLIC;
PROMPT LIT '3EH';
WHEN (16) BYTE DATA (15, 'TODAY'S DATE?', CRLF);
UIMT$ID (21) BYTE DATA (20, 'UIMT/SMART VER. 1.3', CRLF), /* CHANGED 6/25/79 */
/* TIMES FOR "FAST", "SLOW", ETC IN "FINAL.PLM" WERE INCREASED */
/* AT AL MARSHALL'S INSTRUCTIONS. A CHANGE IS ALSO BEING MADE */
/* IN THE "SHOMMORST" PROCEDURE IN RIFLE.PLM TO AVOID THE */
/* PROBLEM WHERE ONLY FOUR RIFLES ARE SHOOTING ==> RIFLE #5 IS */
/* THE WORST BECAUSE OF ALL THE TARGET IGNOREDS & THERE IS NO */
/* RIFLE #5 "WORST-RIFLE" LIGHT. */

```

```

IDENT (19) BYTE DATA (18, 'EXERCISE NUMBER?', CRLF);
QUERY (22) BYTE DATA (21, 'WANT ID? YES OR NO.', CRLF);
TRAINEES (26) BYTE DATA (25, 'ENTER NAMES OF TRAINEES', CRLF);
ONRIFLE (18) BYTE DATA (9, 'ON RIFLE '),
(X, I, J, K, JK) BYTE;
ID$FLAG BYTE PUBLIC;

```

```

15 1  TTY$SET: PROCEDURE EXTERNAL;
16 2  END TTY$SET; /* USED ONLY WHEN USART IS IN RESET CONDITION */

17 1  CIN: PROCEDURE BYTE EXTERNAL;
18 2  END CIN;

19 1  COUT: PROCEDURE (ITEM) EXTERNAL;
20 2  DECLARE ITEM BYTE;
21 2  END COUT;
22 1  PORT$SET: PROCEDURE PUBLIC;
23 2  OUTPUT(GROUP1)=WORD1; /*SET CONTROL WORD INTO GROUP 1 I/O PORTS */
24 2  OUTPUT(GROUP2)=WORD2; /*SET CONTROL WORD INTO GROUP 2 I/O PORTS */
25 2  END PORT$SET;

26 1  PRINT: PROCEDURE (PTR) EXTERNAL;
27 2  DECLARE PTR ADDRESS;
28 2  END PRINT;

29 1  SET$DATA: PROCEDURE (POINTER, LENGTH, VALUE) PUBLIC;
30 2  DECLARE (POINTER, FINAL) ADDRESS,
    (LENGTH, VALUE, SET BASED POINTER) BYTE;
31 2  FINAL = POINTER + LENGTH - 1;
32 2  LOOP: DO WHILE POINTER <= FINAL;
33 3  SET = VALUE;
34 3  POINTER = POINTER+1;
35 3  END LOOP;
36 2  END SET$DATA;

37 1  INTERRUPT$ROUTINE: PROCEDURE INTERRUPT 7 EXTERNAL;
38 2  END INTERRUPT$ROUTINE;

39 1  UPI$STROBE: PROCEDURE EXTERNAL;
40 2  END UPI$STROBE;

41 1  SIMULATE$RIFLES: PROCEDURE;
42 2  OUTPUT(PORT6)=00H; /* TELL UPI-41 TO SIMULATE RIFLE DATA */
43 2  CALL UPI$STROBE;
44 2  END SIMULATE$RIFLES;

```

```

45 1  DECLARE CTALT LIT '14H', RUNTYPE BYTE;

      /***** END OF DECLARATIONS *****/

46 1  STARTUP: PROCEDURE PUBLIC;

47 2  CALL SET#DATA( SCORE, 63, 0);
48 2  CALL SET#DATA( HISTORY, 5, 0);
49 2  CALL SET#DATA( FIRST$SHOT, 5, 1);
50 2  CALL SET#DATA( SPEED, 15, 0);
51 2  CALL SET#DATA( DECIMAL, 3, 0);
52 2  TURKEY = 0;
53 2  TRAIN = 1;
54 2  TFLAG = 0;
55 2  SHOT$FLAG = 0;
56 2  FILE = 4; /* WHO$SHOT FIRST INCREMENTS FILE, MODS (DIVIDES) BY 5,
                I.E. 5/5=1 WITH REMAINDER = 0. NOW, RIFLE=FILE+1,
                SO WE START WITH RIFLE 01 !!! */

57 2  REAL$DONE = 1; /* TARGET HAS NOT BEEN AVAILABLE FOR OVER 1 SEC. */

58 2  CALL PORT$SET;

59 2  OUTPUT(PORT3) = NOT RESET#8212; /* WILL CLEAR ALL 8212 DATA
                                     LATCHES FOLLOWING STROBE */
60 2  OUTPUT(PORT6) = ENABLE#9311; /* THE LEADING EDGE OF THE STROBE */
61 2  OUTPUT(PORT6) = 0; /* THE TRAILING EDGE */

62 2  DO JK = 9 TO 13; /* CLEARS THE JK FF DATA LATCHES */
63 3  OUTPUT(PORT3) = NOT JK AND 0FH;
64 3  OUTPUT(PORT6) = ENABLE#9311;
65 3  OUTPUT(PORT6) = 0;
66 3  END;

67 2  DISABLE;
68 2  CALL TTY$SET;

69 2  SETINT: OUTPUT(ADICM1)=(LOW(INTERRUPT7) AND 000H) + 1FH;
70 2  OUTPUT(ADICM2)=HIGH(INTERRUPT7);
71 2  OUTPUT(ADICM2)=000H; /* MASK ALL BUT INTERRUPT 7 */

      /* WE NOW OBTAIN IDENTIFICATION DATA FOR THE SESSION. IF NEEDED */

72 2  CALL COUT(00H); /* CR */
73 2  CALL COUT(0AH); /* LF */
74 2  CALL COUT(0AH);

75 2  CALL PRINT( UIMTSID);
76 2  CALL COUT(0AH);
77 2  CALL PRINT( QUERY);
78 2  CALL OUTPUT (PROMPT);
79 2  X=CIN;
80 2  RUNTYPE = X; /* SAVE INPUT FOR IDENTIFICATION OF CONTROL T */
81 2  CALL COUT(X);
  
```

```
82 2    CALL COUT(8DH);
83 2    CALL COUT(8AH); /* CR, LF */
84 2    ID#FLAG=0;      /* CONTROL FOR ID PRINT-OUT AT END OF SESSION */

85 2    IF X = 'Y' THEN
86 2    GET#ID: DO;
87 3    ID#FLAG=1;

88 3    CLR#DATE: CALL SET#DATA(, DATE, 12, 8DH); /* CLEARS "DATE" */

89 3    CALL PRINT(, WHEN);
90 3    CALL COUT(PROMPT);

91 3    DATE(0)=11;

92 3    GET#DATE:
93 4    DO I=1 TO 9;
94 4    X=CIN;
95 4    IF X=8DH THEN I=9;
96 4    CALL COUT(X);
97 4    DATE(I) = X;
98 4    END GET#DATE;

99 3    CALL COUT(8DH);
100 3    DATE(11) = 8AH; /* LF */
101 3    CALL COUT(8AH);

102 3    CALL SET#DATA(, ID#NUMBER, 7, 8DH); /* FILLS "ID NUMBER" WITH CR */

103 3    CALL PRINT(, IDENT);
104 3    CALL COUT(PROMPT);

105 3    CALL BIT#DUMP; /* CLEAN OUT USART INPUT BUFFER */

106 3    ID#NUMBER(0)=6;

107 3    GET#ID#NUMBER: DO I=1 TO 4; /* A 4 PLACE "ID" */
108 4    X=CIN;
109 4    IF X=8DH THEN I=4;
110 4    CALL COUT(X);
111 4    ID#NUMBER(I)=X;
112 4    END GET#ID#NUMBER;

113 4    END GET#ID#NUMBER;

114 3    CALL COUT(8DH);
115 3    ID#NUMBER(6)=8AH;
116 3    CALL COUT(8AH);

117 3    CALL SET#DATA(, NAME, 45, 8DH); /* FILLS NAME MATRIX WITH CR */

118 3    CALL PRINT(, TRAINEES);

119 3    GET#NAMES:
120 4    DO I=0 TO 4; /* FOR FIVE TRAINEES */
121 4    CALL PRINT(, ON#I#FILE);
122 4    CALL COUT(31H+1);
```

```

122 4      CALL COUT(ODH);
123 4      CALL COUT(OAH);
124 4      CALL COUT(PROMPT);

125 4      CALL BIT$OUP;

126 4      NAME(I). LETTER(0)=0;

127 4      ONE$NAME:
        DO J=1 TO 6;          /* GETS NAMES UP TO 6 LETTERS LONG */
128 5          X=CIN;
129 5          IF X=ODH THEN J=6;
131 5          NAME(I). LETTER(J)=X;
132 5          CALL COUT(X);
133 5      END ONE$NAME;
134 4      CALL COUT(ODH);      /* CR */
135 4      NAME(I). LETTER(0) = OAH; /* LF */
136 4      CALL COUT (OAH);
137 4      END GET$NAMES;
138 3      END GET$ID;

139 2      IF RUN$TYPE = CTRL$T THEN
140 2          CALL SIMULATE$RIFLES;

141 2          END START$UP;
142 1      END START$UP$MODULE;

```

MODULE INFORMATION:

```

CODE AREA SIZE      = 033FH      831D
VARIABLE AREA SIZE = 0032H      82D
MAXIMUM STACK SIZE = 0004H      4D
248 LINES READ
0 PROGRAM ERROR(S)

```

END OF PL/M-88 COMPILATION

ISIS-11 PL/M-88 V3.1 COMPILATION OF MODULE TIMERM00LE

OBJECT MODULE PLACED IN :F1:TIMER.OBJ

COMPILER INVOKED BY: PLM80 :F1:TIMER.PLM DEBUG IXREF DATE (23 OCT 78)

```

1      TIMERMODULE: DO;

/* THIS MODULE SETS THE 8253 MODES AND READS REGISTERS. NOTE THAT ALL
   THREE 8253 GATES MUST BE HIGH! ALL PAGE REFERENCES ARE TO THE 88/20
   REFERENCE MANUAL 98-317C */
2      1      DECLARE LIT LITERALLY 'LITERALLY',
           COUNTER0 LIT '0DCH', COUNTER1 LIT '0D0H',
           COUNTER2 LIT '0DEH', CONTROL LIT '0DFH'; /* SEE PAGE 2-7 */
3      1      DECLARE CNTR0MODE LIT '34H', /* 2 BYTES, MODE 2 PAGE 3-76 */
           CNTR1MODE LIT '74H', /* 2 BYTES, MODE 2 */
           CNTR2MODE LIT '0B6H'; /* 2 BYTES, MODE 3 */

/*THE FOLLOWING 2-BYTE WORDS ARE THE "BAUD RATE FACTORS" TABLE 4-34 P 4-48 */

4      1      DECLARE LOW0 LIT '0', /* COUNTER 0 PERIOD IS 5 MILLESECONDS */
           HIGH0 LIT '15H',
           LOW1 LIT '5FH', /* COUNTER 1 PERIOD IS 5 MINUTES */
           HIGH1 LIT '0EAH', /* = 5 MIN * 60 (SEC/MIN) / 0.005 SEC -1 IN HEX */
           /******SET UP FOR 300 BAUD *****/
           LOW2 LIT '0E0H', /* COUNTER 2 FREQUENCY IS 4.8 KHZ */
           HIGH2 LIT '00', /* SEE PAGE 3-38. AND NOTE THAT THE 8251 IS SET FOR 16X */

           LOW0VOTRAX LIT '07H',
           HIGH0VOTRAX LIT '00H'; /* SETS VOTRAX OUTPUT TO 9600 BAUD */

5      1      DECLARE TIME0LATCH LIT '40H', /* A COUNTER 1 LATCH. PAGE 3-84. */
           (LS0TIME0BYTE, MS0TIME0BYTE) BYTE PUBLIC;
6      1      DECLARE TIME ADDRESS PUBLIC;
7      1      DECLARE LOW0TIME0BYTE BYTE AT (.TIME), HIGH0TIME0BYTE BYTE AT (.TIME + 1);

8      1      TIMER$START: PROCEDURE PUBLIC;
9      2      OUTPUT(CONTROL)=CNTR0MODE; /* SET COUNTERS 0 & 1 MODES */
10     2      OUTPUT(CONTROL)=CNTR1MODE;
11     2      OUTPUT(COUNTER0)=LOW0; /* INITIALIZE COUNTERS */
12     2      OUTPUT(COUNTER0)=HIGH0;
13     2      OUTPUT(COUNTER1)=LOW1;
14     2      OUTPUT(COUNTER1)=HIGH1;
15     2      END TIMER$START;

16     1      TTY$TIMER: PROCEDURE PUBLIC;
17     2      OUTPUT(CONTROL)=CNTR2MODE;
18     2      OUTPUT(COUNTER2)=LOW2; /* WORDS FOR 300 BAUD */
19     2      OUTPUT(COUNTER2)=HIGH2;
20     2      END TTY$TIMER;

21     1      VOTRAX$TIMER: PROCEDURE PUBLIC; /* SET VOTRAX TO 9600 BAUD */
22     2      OUTPUT(CONTROL)=CNTR2MODE;
23     2      OUTPUT(COUNTER2)=LOW0VOTRAX;
24     2      OUTPUT(COUNTER2)=HIGH0VOTRAX;
25     2      END VOTRAX$TIMER;

```

```

26 1  CLOCK$READ: PROCEDURE ADDRESS PUBLIC; /* GETS THE CONTENTS OF COUNTER 1 */
27 2  OUTPUT(CONTROL)=TIME$LATCH;
28 2  LOW$TIME$BYTE=INPUT(COUNTER1);
29 2  HIGH$TIME$BYTE=INPUT(COUNTER1);
30 2  RETURN TIME;
31 2  END CLOCK$READ;

32 1  END TIMER$MODULE;

```

MODULE INFORMATION:

```

CODE AREA SIZE      = 0045H    690
VARIABLE AREA SIZE = 0000H     40
MAXIMUM STACK SIZE = 0000H     00
59 LINES READ
0 PROGRAM ERROR(S)

```

END OF PL/M-80 COMPILATION

ISIS-11 PL/M-88 V3.1 COMPILATION OF MODULE RIFLEDATANMODULE
 OBJECT MODULE PLACED IN :F1:RIFLE.OBJ
 COMPILER INVOKED BY: PL/M88 :F1:RIFLE.PL/M INREF DEBUG DATE (25 JUN 79)

```

1      RIFLE#DATAMODULE:DO;

      /* INCLUDES VARIOUS PROCEDURES FOR RIFLE DATA I/O */

2  1    DECLARE MESSAGE BYTE;

3  1    COUT: PROCEDURE (ITEM) EXTERNAL;
4  2    DECLARE ITEM BYTE;
5  2    END COUT;

6  1    SCORIT: PROCEDURE (SHOT#LOCATION#FILE);
7  2    DECLARE (SHOT#LOCATION#FILE) BYTE;
8  2    ADDER(FILE) J(SHOT#LOCATION)=ADDER(FILE) J(SHOT#LOCATION)+1;
9  2    END SCORIT;

      /* DECODE TRANSLATES QUADRANT DATA INTO SHOT POSITION. NOTE THAT
      ERRORS ARE TREATED A MISSES. THUS A "HIGH-LOW" OR A "LEFT-RIGHT"
      WHICH SHOULD NEVER OCCUR IS A MISS. TO SCORE AS ERRORS CHANGE THE
      SECOND AND THIRD 0 OF DECODE INTO "10" */

10 1    DECLARE DECODE(16) BYTE DATA(0,2,4,3,6,0,5,4,8,9,0,2,7,8,6,1);

11 1    CLOCK#READ: PROCEDURE ADDRESS EXTERNAL;
12 2    END CLOCK#READ;

13 1    SET#DATA: PROCEDURE (PTR,LENGTH,VALUE) EXTERNAL;
14 2    DECLARE (LENGTH,VALUE) BYTE, PTR ADDRESS;
15 2    END SET#DATA;

16 1    DECLARE FIRST#SHOT(5) BYTE PUBLIC;
17 1    DECLARE LIT LITERALLY 'LITERALLY',
      HIGH#D LIT '0EH', LOW#D LIT '1', RIGHT#D LIT '2', LEFT#D LIT '3',
      HIT#D LIT '4', MISS#D LIT '5', NOM#D LIT '6', TARGET#D LIT '7',
      SOLDIER#D LIT '8', YOUNG#D LIT '9', ARM#D LIT '0AH',
      WPS#D LIT '0BH', MARINE#D LIT '0CH', ENEMY#D LIT '0DH',
      AVAILABLE#D LIT '32', FROZEN#D LIT '0FH',
      PORT1 LIT '0E4H', /* PAGE 3-68 OF 88/20 MANUAL */
      PORT2 LIT '0E5H',
      PORT3 LIT '0E6H',
      PORT4 LIT '0E8H',
      PORT6 LIT '0EAH',
      ENABLE1 LIT '10H';

18 1    DECLARE COUNTER#SET LIT '0E5FH'; /* INITIAL VALUE 5 MIN COUNTER */
19 1    DECLARE HISTORY(5) BYTE EXTERNAL;
20 1    DECLARE (TURKEY, TFLAG) BYTE EXTERNAL;
21 1    DECLARE SCORE(5) STRUCTURE (MISS BYTE, HIT BYTE, LOW BYTE, LOW#RIGHT BYTE,
      RIGHT BYTE, HIGH#RIGHT BYTE, HIGH BYTE, HIGH#LEFT BYTE,
      LEFT BYTE, LOW#LEFT BYTE, ERROR BYTE, TURKEY BYTE,
      TARGET#IGNORED BYTE) PUBLIC;

      /* WE WILL USE "ADDER" TO COMPUTE TOTAL SHOTS */
22 1    DECLARE ADDER(5) STRUCTURE ( J(13) BYTE) PUBLIC AT ( SCORE);
  
```

```

23 1  DECLARE SPEED(S) STRUCTURE(SHOTS BYTE, TIME$SUM ADDRESS) PUBLIC;
24 1  DECLARE TOTAL$TIME ADDRESS;
25 1  DECLARE DELTA$TIME ADDRESS PUBLIC;
26 1  DECLARE (FILE, RIFLE, FPTR) BYTE PUBLIC;
27 1  DECLARE SHOT$FLAG BYTE PUBLIC;
28 1  DECLARE TARGET$TIME ADDRESS EXTERNAL;
29 1  DECLARE RIFLE$ID BYTE;
30 1  DECLARE CRT$STROBE LIT '000100000';

31 1  UPI$STROBE: PROCEDURE PUBLIC;
32 2  OUTPUT(PORT3) = CRT$STROBE;
33 2  OUTPUT(PORT3) = 0;
34 2  END UPI$STROBE;

35 1  TARGET$AVAILABLE: PROCEDURE BYTE PUBLIC; /* GIVES "TRUE" IF TARGET PRESENT */
36 2  DECLARE TARGET BYTE;
37 2  TARGET=INPUT(PORT2);
38 2  RETURN TARGET; /* PORT 2 BIT NUMBER 0 WILL BE HIGH IF TARGET PRESENT */
39 2  END TARGET$AVAILABLE;

40 1  RIFLE$SHOT: PROCEDURE BYTE PUBLIC; /* RETURNS "TRUE" IF SHOT FIRED */
41 2  SHOT$FLAG=INPUT(PORT1); /* "SHOT$FLAG" IS TRUE WHEN SHOT FIRED */
42 2  RETURN SHOT$FLAG;
43 2  END RIFLE$SHOT;

44 1  GET$RIFLE$DATA: PROCEDURE PUBLIC;
45 2  DECLARE SHOT$DATA BYTE;
46 2  UNRESOLVED: DO WHILE SHOT$FLAG=INPUT(PORT1);
47 3  SHOT$DATA=0; /* NEEDED TO ENTER FOLLOWING "WHO$SHOT" ROUTINE */
48 3  WHO$SHOT: DO WHILE NOT SHOT$DATA;
49 4  FILE = (FILE+1) MOD 5; /* FILE 0 CONTAINS RIFLE NUMBER 1 DATA,
                           FILE 1 ==> RIFLE 02, ETC. */
50 4  RIFLE=FILE+1; /*"FILE" IS INITIALIZED ONLY ONCE. THUS
                  WE START CHECKING WHERE WE LEFT OFF */
51 4  SHOT$DATA=SHR(SHOT$FLAG, RIFLE);
52 4  END WHO$SHOT; /* "RIFLE" EQUALS SHOOTING RIFLE NUMBER */

53 3  RIFLE$ID = ROR(RIFLE, 3); /* GET RIFLE ID INTO HIGH ORDER BITS */

54 3  IF NOT TURKEY THEN
55 3  DO;
56 4  IF FIRST$SHOT(FILE) THEN
57 4  HONQUICK: DO;
58 5  IF TARGET$TIME >= CLOCK$READ THEN
59 5  DELTA$TIME = TARGET$TIME - CLOCK$READ;
60 5  ELSE
61 5  DELTA$TIME = COUNTER$SET + 1 + TARGET$TIME - CLOCK$READ;
62 5  SPEED(FILE). TIME$SUM=SPEED(FILE). TIME$SUM + DELTA$TIME;
63 5  SPEED(FILE). SHOTS = SPEED(FILE). SHOTS + 1;
64 5  FIRST$SHOT(FILE) = 0;
65 4  END HONQUICK;
66 3  END;

66 3  HISTORY(FILE) = 1;

67 3  OUTPUT(PORT3)=NOT RIFLE AND 0FH; /* SETS 9311 ADDRESS OF /DS1 LINE 0 RIFLE */
68 3  OUTPUT(PORT6) = ENABLE1; /* LATCHES SHOT$DATA FROM RIFLE ONTO BUS */

```

```

69 3  SHOT#DATA=INPUT(PORT2); /* READS QUADRANT DATA */
70 3  OUTPUT(PORT6) = 0; /* RETURNS /DS1 0 RIFLE HIGH AND DROPS 8212 FROM RUS */
71 3  OUTPUT(PORT3)=NOT(00H OR RIFLE) AND 0FH; /* ADDRESSES 9311 /GOT DATA 0 RIFLE */
72 3  OUTPUT(PORT6) = ENABLE1;
73 3  OUTPUT(PORT6) = 0;
74 3  CALL COUT(0FH); /* OUTPUT "SVN" MESSAGE TO VOTRAX */
75 3  CALL COUT(000H + FILE); /* RIFLE ADDRESSED */
76 3  IF (NOT TURKEY) AND TFLAG THEN
77 3  OK#DATA: DO; /* ADD IN NEW SCORE DATA. ASSUME FOR PORT 2 THAT
                        BIT 1 = LOW, BIT 2 = RIGHT
                        BIT 3 = HIGH, BIT 4 = LEFT */
78 4  CALL SCORIT(MESSAGE:=DECODE(SHR(SHOT#DATA,1) AND 0FH),FILE);
79 4  OUTPUT(PORT6) = NOT(RIFLE#ID + MESSAGE) AND 11101111B;
80 4  CALL UPI#STROBE;
81 4  VOTRAX#MESSAGES: DO CASE MESSAGE;
82 5  CALL COUT(MISS#ID);
83 5  CALL COUT(HIT#ID);
84 5  CALL COUT(LOW#ID);
85 5  DO;
86 6  CALL COUT(LOW#ID);
87 6  CALL COUT(RIGHT#ID);
88 6  END;
89 5  CALL COUT(RIGHT#ID);
90 5  DO;
91 6  CALL COUT(HIGH#ID);
92 6  CALL COUT(RIGHT#ID);
93 6  END;
94 5  CALL COUT(HIGH#ID);
95 5  DO;
96 6  CALL COUT(HIGH#ID);
97 6  CALL COUT(LEFT#ID);
98 6  END;
99 5  CALL COUT(LEFT#ID);
100 5  DO;
101 6  CALL COUT(LOW#ID);
102 6  CALL COUT(LEFT#ID);
103 6  END;
104 5  END VOTRAX#MESSAGES;
105 4  END OK#DATA;
106 3  IF TURKEY THEN
107 3  DO;
108 4  TURKEY#DATA: SCORE(FILE), TUR#EV = SCORE(FILE), TURKEY#1.
109 4  OUTPUT(PORT6) = NOT(RIFLE#ID + 00H) AND 11101111B;
110 4  CALL UPI#STROBE; /* SENDS TURKEY TO CONSOLE CRT */
111 4  CALL COUT(N0#ID);
112 4  CALL COUT(TARGET#ID);

```

```

113 4      END;

114 3      IF (NOT TFLAG) AND (NOT TURKEY) THEN
115 3      DO;
116 4          SCORE(FILE).MISS = SCORE(FILE).MISS + 1;
117 4          OUTPUT(PORT6) = NOT(RIFLE$ID + 0AH) AND 11101111B;
118 4          CALL UP1$STROBE; /* SENDS "MISS, LATE" MESSAGE TO CONSOLE CRT */
119 4          CALL COUT(MISS$MD);
120 4          END;

121 3      CALL COUT(0FFH); /* VOTRAX SIGN-OFF */

122 3      END UNRESOLVED;
123 2      END GET$RIFLE$DATA;

124 1      WHO$FAILED$TO$SHOOT: PROCEDURE PUBLIC;
125 2      DO FPTR=0 TO 4;
126 3          RIFLE$ID = ROR(FPTR+1,3);
127 3          IF HISTORY(FPTR)=0 THEN
128 3          DO;
129 4              SCORE(FPTR).TARGET$IGNORED=SCORE(FPTR).TARGET$IGNORED+1;
130 4              OUTPUT(PORT6) = NOT(RIFLE$ID + 0CH) AND 11101111B;
131 4              CALL UP1$STROBE; /* SENDS TARGET IGNORED TO CONSOLE CRT */
132 4              CALL COUT(0FFH); /* VOTRAX "INT" MESSAGE */
133 4              CALL COUT(0C0H + FPTR); /* ADDRESS RIFLE NOT SHOOTING */
134 4              CALL COUT(YOUN$MD);
135 4              CALL COUT(FROZE$MD);
136 4              CALL COUT(0FFH); /* END OF VOTRAX MESSAGE */
137 4              END;
138 3          END;
139 2          CALL SET$DATA(.HISTORY,5,0);
140 2      END WHO$FAILED$TO$SHOOT;

141 1      SHOW$WORST: PROCEDURE PUBLIC;
142 2      DECLARE BAD$WORD BYTE;
143 2      DECLARE BAD$NEWS(5) BYTE;
144 2      HOW$BAD: DO FPTR = 0 TO 4;
145 3          BAD$NEWS(FPTR) = 0;
146 3          IF SPEED(FPTR).SHOTS<0 THEN /* AVOIDS "WORST-SHOOTER" IF NO RIFLE AT FPTR */
147 3          BAD$NEWS(FPTR) = SCORE(FPTR).MISS + SCORE(FPTR).TURKEY +
              SCORE(FPTR).TARGET$IGNORED;

148 3          END HOW$BAD;
149 2          BAD$WORD = 1;
150 2          RIFLE = 1;
151 2          HUNT$WORST: DO FPTR = 0 TO 3;
152 3              IF BAD$NEWS(FPTR+1) = BAD$NEWS(RIFLE-1) THEN
153 3              BAD$WORD = BAD$WORD OR ROL(00H,(FPTR + 2));
154 3              IF BAD$NEWS(FPTR+1) > BAD$NEWS(RIFLE-1) THEN
155 3              DO;
156 4                  RIFLE = FPTR + 2;
157 4                  BAD$WORD = (BAD$WORD AND 0) OR ROL(00H,RIFLE);
158 4                  END;
159 3          END HUNT$WORST; /* RIFLE HAS VALUE OF WORST SHOOTER */
160 2          OUTPUT(PORT4) = BAD$WORD;
161 2          END SHOW$WORST;

162 1      END RIFLE$DATA$MODULE;

```

MODULE INFORMATION:

CODE AREA SIZE = 0387H 951D
VARIABLE AREA SIZE = 0069H 105D
MAXIMUM STACK SIZE = 0004H 4D
222 LINES READ
0 PROGRAM ERROR(S)

END OF PL/M-88 COMPILATION

1515-II PL/M-88 V3.1 COMPILATION OF MODULE CONSOLEMODULE
 OBJECT MODULE PLACED IN :F1:CONSOL.OBJ
 COMPILER INVOKED BY: PL/M88 :F1:CONSOL.PLM DEBUG IXREF DATE (12 OCT 78)

```

1      CONSOLE$MODULE: DO;
      /* THIS MODULE CONTAINS CONSOLE I/O ROUTINES */

2 1    DECLARE USART$DATA LITERALLY '0ECH', /* PAGE 3-48 OF 88/20 MANUAL */
      USART$STATUS LITERALLY '0EDH', /* PAGE 3-44 */
      ESC LITERALLY '1BH', /* ASCII "ESCAPE" */
      MASK LITERALLY '7FH',
      ZERO LITERALLY '30H',
      CR LITERALLY '0DH', LF LITERALLY '0AH',
      ENABLE$9311 LITERALLY '10H',
      PORT3 LITERALLY '06H',
      PORT6 LITERALLY '06AH',
      TTY$LINE LITERALLY '06H',
      VOTRAX$LINE LITERALLY '09H',
      USART$CONTROL LITERALLY '0EDH',

      /******THE SILENT 700 IS HEREAFTER DEFINED AS A TTY *****/
      TTY$MODE LITERALLY '04EH', /* 1 STOP BITS, 8 BIT, 16X, P. 3-38, 41, 4-48 */
      VOTRAX$MODE LITERALLY '4EH', /* DITTO EXCEPT 1 STOP BIT */
      USART$COMMAND LITERALLY '27H', /* DATA TERMINAL READY, ETC */
      USART$RESET LITERALLY '40H',
      HELLO (18) BYTE DATA (17, CR, LF, LF, LF, 'LET'S START', CR, LF);

3 1    DECLARE DECIMAL(3) BYTE EXTERNAL;
4 1    DECLARE (I,V,GOS) BYTE;

5 1    TTY$TIMER: PROCEDURE EXTERNAL;
6 2    END;

7 1    VOTRAX$TIMER: PROCEDURE EXTERNAL;
8 2    END;

9 1    TTY$SET: PROCEDURE PUBLIC;
10 2    OUTPUT(PORT3) = TTY$LINE; /* WILL DIRECT USART OUTPUT TO TTY */
11 2    OUTPUT(PORT6) = ENABLE$9311;
12 2    OUTPUT(PORT6) = 0;

13 2    CALL TTY$TIMER; /* SETS UP FOR 110 BAUD */

14 2    OUTPUT(USART$CONTROL) = TTY$MODE;
15 2    OUTPUT(USART$CONTROL) = USART$COMMAND;
16 2    END TTY$SET;

17 1    TTY$RES: PROCEDURE PUBLIC; /* TTY RESET */
18 2    OUTPUT(USART$CONTROL) = USART$RESET;
19 2    CALL TTY$SET;
20 2    END TTY$RES;

21 1    VOTRAX$SET: PROCEDURE PUBLIC;
22 2    OUTPUT(USART$CONTROL) = VOTRAX$MODE;
23 2    OUTPUT(USART$CONTROL) = USART$COMMAND;
24 2    END VOTRAX$SET;

```

```

25 1  VOTRES: PROCEDURE PUBLIC; /* VOTRAX RESET */
26 2  OUTPUT(USART$CONTROL) = USART$RESET;
27 2  OUTPUT(PORT3) = VOTRAX$LINE;
28 2  OUTPUT(PORT6) = ENABLE$9311;
29 2  OUTPUT(PORT6) = 0;

30 2  CALL VOTRAX$TIMER;
31 2  CALL VOTRAX$SET;

32 2  END VOTRES;

33 1  CIN: PROCEDURE BYTE PUBLIC; /* GETS A BYTE FROM THE CONSOLE */
34 2  RRDY: DO WHILE NOT SHR(INPUT(USART$STATUS), 1); /* I.E. WHILE INPUT BUFFER
                                                    IS NOT READY */
35 3  END RRDY;
36 2  RETURN MASK AND INPUT(USART$DATA);
37 2  END CIN;

38 1  TXRDY: PROCEDURE PUBLIC;
39 2  DO WHILE NOT SHR(INPUT(USART$STATUS), 2);
40 3  END;
41 2  END TXRDY;

42 1  COUT: PROCEDURE (ITEM) PUBLIC; /* OUTPUTS "ITEM" */
43 2  DECLARE ITEM BYTE;
44 2  DO WHILE NOT(INPUT(USART$STATUS));
45 3  END;
46 2  OUTPUT(USART$DATA)=ITEM;
47 2  END COUT;

48 1  BIT$DUMP: PROCEDURE PUBLIC;
49 2  Y = INPUT(USART$DATA); /* "Y" HERE IS A BIT-BUCKET */
50 2  END BIT$DUMP;

51 1  PRNTNUM: PROCEDURE PUBLIC;
52 2  GOS=0;
53 2  DO I=0 TO 2;
54 3  IF (Y=DECIMAL(I)) <> ZERO THEN
55 3  GOS=1;
56 3  IF GOS THEN CALL COUT(Y);
57 3  END;
58 3  END;
59 2  CALL COUT(CR);
60 2  CALL COUT(LF);
61 2  END PRNTNUM;

62 1  PRINT: PROCEDURE (POINTER) PUBLIC;
63 2  DECLARE (POINTER, FINAL) ADDRESS;
        CHAR BASED POINTER BYTE;
64 2  FINAL=POINTER+CHAR; /* FIRST CHAR IS CHARACTER COUNT */
65 2  LOOP: DO WHILE POINTER < FINAL;
66 3  POINTER=POINTER+1;
67 3  CALL COUT(CHAR);
68 3  END LOOP;
69 2  END PRINT;

70 1  GREETING: PROCEDURE PUBLIC;

```

```

71 2    CALL PRINT(HELLO);
72 2    CALL BIT$DUMP;
73 2    ENABLE;
74 2    END GREETING;

```

```

75 1    END CONSOLE$MODULE;

```

MODULE INFORMATION:

```

CODE AREA SIZE      = 010FH    271D
VARIABLE AREA SIZE = 0000H     8D
MAXIMUM STACK SIZE = 0004H     4D
115 LINES READ
0 PROGRAM ERROR(S)

```

END OF PL/M-80 COMPILATION

ISIS-II PL/M-88 V3.1 COMPILATION OF MODULE RESULTSMODULE

OBJECT MODULE PLACED IN :F1:RESULT.OBJ

COMPILER INVOKED BY: PLM88 :F1:RESULT.PL M IXREF DEBUG DATE (3 OCT 78)

```

1      RESULTS#MODULE: DO;

2  1    TTVRES: PROCEDURE EXTERNAL;
3  2      END TTVRES;

4  1    COMPOSITE: PROCEDURE EXTERNAL;
5  2      END COMPOSITE;

6  1    COMMENT: PROCEDURE EXTERNAL;
7  2      END COMMENT;

8  1    COUT: PROCEDURE (LTR) EXTERNAL;
9  2      DECLARE LTR BYTE;
10 2      END COUT;

11 1    PRINT: PROCEDURE (POINTER) EXTERNAL;
12 2      DECLARE POINTER ADDRESS;
13 2      END PRINT;

14 1    PRNTNUM: PROCEDURE EXTERNAL;
15 2      END PRNTNUM;

16 1    DECLARE (RIFLE, FILE) BYTE EXTERNAL;
17 1    DECLARE OR LITERALLY '00H', LF LITERALLY '0AH';
18 1    DECLARE SCORE (5) STRUCTURE(MISS BYTE, HIT BYTE, LOW BYTE, LOWRIGHT BYTE,
    RIGHT BYTE, HIGHRIGHT BYTE, HIGH BYTE, HIGHLEFT BYTE,
    LEFT BYTE, LOWLEFT BYTE, ERROR BYTE, TURKEY BYTE,
    TARGET#IGNORED BYTE) EXTERNAL;

19 1    DECLARE DECIMAL(3) BYTE PUBLIC;
20 1    DECLARE (1,2) BYTE,
    (SUM#SHOTS, NEAR#MISSES) BYTE PUBLIC;

21 1    DECLARE ADDER(5) STRUCTURE(J(13) BYTE) EXTERNAL;
22 1    DECLARE SPEED(5) STRUCTURE(SHOTS BYTE, TIME#SUM ADDRESS) EXTERNAL;
23 1    DECLARE AVG#TIME ADDRESS PUBLIC;

24 1    CONVRT: PROCEDURE(HEX);
25 2      DECLARE HEX BYTE;
26 2      DO I=0 TO 2;
27 3      DECIMAL(2-I)= HEX MOD 10 + 30H;
28 3      HEX = HEX / 10;
29 3      END;
30 2      END CONVRT;

31 1    DECLARE RIFLE#ID(8) BYTE DATA(7, 'RIFLE: ');
32 1    DECLARE TOTAL#SHOTS(14) BYTE DATA(13, 'TOTAL SHOTS: ');
33 1    DECLARE RIFLE#HIT(7) BYTE DATA(6, 'HITS: ');
34 1    DECLARE RIFLE#MISS(9) BYTE DATA(8, 'MISSES: ');
35 1    DECLARE RIFLE#LOW(7) BYTE DATA(6, 'LOWS: ');
36 1    DECLARE RIFLE#LOWRIGHT(13) BYTE DATA(12, 'LOW RIGHTS ');

```

```

37 1  DECLARE RIFLE$RIGHT(9) BYTE DATA(8, 'RIGHTS: ');
38 1  DECLARE RIFLE$HIGH$RIGHT(14) BYTE DATA(13, 'HIGH RIGHTS: ');
39 1  DECLARE RIFLE$HIGH(8) BYTE DATA(7, 'HIGHS: ');
40 1  DECLARE RIFLE$HIGH$LEFT(13) BYTE DATA(12, 'HIGH LEFTS: ');
41 1  DECLARE RIFLE$LEFT(8) BYTE DATA(7, 'LEFTS: ');
42 1  DECLARE RIFLE$LOW$LEFT(12) BYTE DATA(11, 'LOW LEFTS: ');
43 1  DECLARE RIFLE$TURKEY(12) BYTE DATA(11, 'NO TARGET: ');
44 1  DECLARE RIFLE$TARGET$IGNORED(18) BYTE DATA(17, 'TARGETS IGNORED: ');
45 1  DECLARE BLANK(3) BYTE DATA(2, CR, LF);
46 1  DECLARE HOWMANY$SHOTS(18) BYTE DATA(17, 'TARGETS SHOT AT: ');
47 1  DECLARE AVERAGE$TIME(15) BYTE DATA(14, 'AVERAGE TIME: ');
48 1  DECLARE UNITS(8) BYTE DATA(7, 'SECONDS');
49 1  DECLARE YOUR$SCORE (18) BYTE DATA (17, 'YOUR RESULTS ARE:');

50 1  DECLARE NAME(5) STRUCTURE(LETTER(9) BYTE) EXTERNAL,
      DATE (11) BYTE EXTERNAL,
      ID$NUMBER (6) BYTE EXTERNAL,
      ID$FLAG BYTE EXTERNAL;

51 1  PRESENT$RESULTS: PROCEDURE PUBLIC;

52 2  DISABLE;

53 2  CALL TTYRES; /* RESET FOR TTY OUTPUT SEE CONSOL MODULE */

54 2  CALL PRINT( BLANK);
55 2  CALL PRINT( BLANK);

56 2  IF ID$FLAG THEN DO;
58 3  CALL PRINT( DATE);
59 3  CALL PRINT( ID$NUMBER);
60 3  END;

61 2  CALL PRINT( BLANK);
62 2  CALL PRINT( BLANK);

63 2  ONE$RIFLE$RESULTS: DO RIFLE=1 TO 5;

64 3  IF SPEED(RIFLE-1) SHOTS <> 0 THEN
65 3  TYPE$IT: DO;

66 4  CALL PRINT( BLANK);

67 4  CALL PRINT( RIFLE$ID);
68 4  CALL COUT(RIFLE+30H);

69 4  CALL PRINT( BLANK);
70 4  CALL PRINT( BLANK);

71 4  IF ID$FLAG THEN
72 4  CALL PRINT( NAME(RIFLE-1));
73 4  CALL PRINT( YOUR$SCORE);

74 4  CALL PRINT( BLANK);
75 4  CALL PRINT( BLANK);

76 4  FILE=RIFLE-1;

```

```
77 4    CALL PRINT(. TOTAL#SHOTS);

78 4    SUM#SHOTS = 0;
79 4    SUM: DO Z=0 TO 11;
80 5    SUM#SHOTS = SUM#SHOTS + ADDER(FILE). J(Z);
81 5    END SUM;

82 4    NEAR#MISSES = 0;
83 4    SUM2: DO Z = 2 TO 9;
84 5    NEAR#MISSES = NEAR#MISSES + ADDER(FILE). J(Z);
85 5    END SUM2;

86 4    CALL CONVRT(SUM#SHOTS);
87 4    CALL PRNTNUM;

88 4    CALL PRINT(. RIFLE#HIT);
89 4    CALL CONVRT(SCORE(FILE). HIT);
90 4    CALL PRNTNUM;

91 4    CALL PRINT(. RIFLE#MISS);
92 4    CALL CONVRT(SCORE(FILE). MISS);
93 4    CALL PRNTNUM;

94 4    CALL PRINT(. RIFLE#LOW);
95 4    CALL CONVRT(SCORE(FILE). LOW);
96 4    CALL PRNTNUM;

97 4    CALL PRINT(. RIFLE#LOW#RIGHT);
98 4    CALL CONVRT(SCORE(FILE). LOW#RIGHT);
99 4    CALL PRNTNUM;

100 4    CALL PRINT(. RIFLE#RIGHT);
101 4    CALL CONVRT(SCORE(FILE). RIGHT);
102 4    CALL PRNTNUM;

103 4    CALL PRINT(. RIFLE#HIGH#RIGHT);
104 4    CALL CONVRT(SCORE(FILE). HIGH#RIGHT);
105 4    CALL PRNTNUM;

106 4    CALL PRINT(. RIFLE#HIGH);
107 4    CALL CONVRT(SCORE(FILE). HIGH);
108 4    CALL PRNTNUM;

109 4    CALL PRINT(. RIFLE#HIGH#LEFT);
110 4    CALL CONVRT(SCORE(FILE). HIGH#LEFT);
111 4    CALL PRNTNUM;

112 4    CALL PRINT(. RIFLE#LEFT);
113 4    CALL CONVRT(SCORE(FILE). LEFT);
114 4    CALL PRNTNUM;

115 4    CALL PRINT(. RIFLE#LOW#LEFT);
116 4    CALL CONVRT(SCORE(FILE). LOW#LEFT);
117 4    CALL PRNTNUM;

118 4    CALL PRINT(. RIFLE#TURKEY);
```

```

119 4    CALL CONVRT(SCORE(FILE), TURKEY);
120 4    CALL PRNTNUM;

121 4    CALL PRINT(. RIFLE$TARGET$IGNORED);
122 4    CALL CONVRT(SCORE(FILE), TARGET$IGNORED);
123 4    CALL PRNTNUM;

124 4    CALL PRINT(. HONK$M$SHOTS);
125 4    CALL CONVRT(SPEED(FILE), SHOTS);
126 4    CALL PRNTNUM;

127 4    CALL PRINT(. AVERAGE$TIME);
128 4    IF (Z:=SPEED(FILE), SHOTS)=0 THEN Z=1;
129 4    AVG$TIME = (SPEED(FILE), TIME$SUM/20)/Z;
130 4    CALL CONVRT(LOW(AVG$TIME));
131 4    IF (Z:=DECIMAL(0)) < 30H THEN CALL COUT(Z);
132 4    CALL COUT(DECIMAL(1));
133 4    CALL COUT(2EH); /* A PERIOD OR DECIMAL POINT */
134 4    CALL COUT(DECIMAL(2));
135 4    CALL COUT(20H);
136 4    CALL PRINT(. UNITS);
137 4    CALL COUT(CR);
138 4    CALL COUT(LF);

141 4    CALL PRINT(. BLANK);
142 4    CALL COMMENT; /* COMMENT AS TO REACTION TIME */

143 4    CALL PRINT(. BLANK);
/*****

COMPOSITE IS EASY TO CHANGE. IT'S IN MODULE "FINAL"

*****/

144 4    CALL COMPOSITE; /* THE COMPOSITE SCORE IS INITIALLY: =
                        100*(HITS/SHOTS) + 60*(NEAR MISSES/SHOTS) +
                        10*(TIME CREDIT FROM PROCEDURE "COMMENT", ABOVE)
                        - 2*(NUMBER OF TARGETS IGNORED) */

145 4    CALL PRINT(. BLANK);

146 4    END TYPE$IT;

147 3    END ONE$RIFLE$RESULTS;

148 2    ENABE;

149 2    END PRESENT$RESULTS;

150 1    END RESULTS$MODULE;

```

MODULE INFORMATION:

CODE AREA SIZE = 0491H 1169D
 VARIABLE AREA SIZE = 000AH 10D

3 OCT 78 PAGE 5

MAXIMUM STACK SIZE = 0006H 6D
215 LINES READ
0 PROGRAM ERROR(S)

END OF PL/1-80 COMPILATION

1515-11 PL/M-88 V3.1 COMPILATION OF MODULE FINALMODULE

OBJECT MODULE PLACED IN :F1.FINAL.OBJ

COMPILER INVOKED BY: PLM88 :F1.FINAL.PLM INREF DEBUG DATE (25 JUN 79)

```

1      FINALMODULE: DO;
2 1    DECLARE SPEED(5) STRUCTURE(SHOTS BYTE, TIME$SUM ADDRESS) EXTERNAL,
      (FILE, RIFLE, SUNSHOTS, NEARMISSES) BYTE EXTERNAL,
      W BYTE, AVGTIME ADDRESS EXTERNAL,
      SCORE(5) STRUCTURE(MISS BYTE, HIT BYTE, LOW BYTE, LOWRIGHT BYTE,
      RIGHT BYTE, HIGHRIGHT BYTE, HIGH BYTE,
      HIGHLEFT BYTE, LEFT BYTE, LOWLEFT BYTE,
      ERROR BYTE, TURKEY BYTE, TARGET$IGNORED BYTE)
      EXTERNAL,
      FAST (42) BYTE DATA (41, 'HAW!! YOU'RE THE FASTEST SHOT IN THE WEST'),
      GOOD (25) BYTE DATA (24, 'HEY! YOU'RE PRETTY QUICK'),
      FAIR (38) BYTE DATA (37, 'OH WELL! THERE'S HOPE IF YOU SPEED UP'),
      POOR (30) BYTE DATA (29, 'SORRY, BUT YOU'RE PRETTY SLOW'),
      OR LITERALLY 'OH', LF LITERALLY 'OH',
      (TIME$CREDIT, N) BYTE;

3 1    COUT: PROCEDURE (LTR) EXTERNAL;
4 2    DECLARE LTR BYTE;
5 2    END COUT;

6 1    PRINT: PROCEDURE (POINTER) EXTERNAL;
7 2    DECLARE POINTER ADDRESS;
8 2    END PRINT;

9 1    COMMENT: PROCEDURE PUBLIC;
10 2    IF AVGTIME <= 5 THEN
11 2    DO;
12 3    CALL PRINT( FAST);
13 3    TIME$CREDIT = 3;
14 3    END;
15 2    ELSE IF AVGTIME <= 9 THEN
16 2    DO;
17 3    CALL PRINT( GOOD);
18 3    TIME$CREDIT = 2;
19 3    END;
20 2    ELSE IF AVGTIME <= 13 THEN
21 2    DO;
22 3    CALL PRINT( FAIR);
23 3    TIME$CREDIT = 1;
24 3    END;
25 2    ELSE DO;
26 3    CALL PRINT( POOR);
27 3    TIME$CREDIT = 0;
28 3    END;
29 2    END COMMENT;

30 1    HX2AS: PROCEDURE(HEX$ADR, DEC$ADR) PUBLIC;
31 2    DECLARE(HEX$ADR, DEC$ADR) ADDRESS,
      HEX BASED HEX$ADR ADDRESS,
      DECIMAL BASED DEC$ADR (5) BYTE,
      (N,N) BYTE;

```

```

32 2    DO N = 0 TO 4;
33 3      N=N;
34 3      DECIMAL(N) = HEX MOD 10 + 30H;
35 3      HEX = HEX/10;
36 3      END;
37 2    N=0;
38 2    DO WHILE DECIMAL(N) = 30H AND N<5;
39 3      DECIMAL(N) = 20H; /* REPLACE LEADING ZEROS WITH SPACES */
40 3      N = N + 1;
41 3      END;
42 2    END HK2AS;

43 1    COMPOSITE: PROCEDURE PUBLIC;
44 2    DECLARE COMP(24) BYTE DATA(23, 'YOUR OVERALL SCORE IS: '),
      OVERALL ADDRESS,
      DECNUM (5) BYTE;
45 2    OVERALL = 100*(SCORE(FILE). HIT)/SUMSHOTS + 60*MISS/MISSSES/SUMSHOTS
      + 10*TIME/CREDIT - 2*(SCORE(FILE). TARGET*100/RED);
46 2    CALL PRINT( COMP);

47 2    IF OVERALL < 0F00H THEN DO; /* I. E. CHECK FOR NEGATIVE SCORE */
49 3      CALL HK2AS( OVERALL, DECNUM);
50 3      DO N = 0 TO 4;
51 4        CALL COUT(DECNUM(N));
52 4      END;
53 3    END;

54 2    CALL COUT(OR);
55 2    CALL COUT(LF);
56 2    CALL COUT(LF);
57 2    END COMPOSITE;

58 1    END FINALMODULE;

```

MODULE INFORMATION:

```

CODE AREA SIZE      = 0229H    3530
VARIABLE AREA SIZE = 0010H    160
MAXIMUM STACK SIZE = 0000H     80
85 LINES READ
0 PROGRAM ERROR(S)

```

END OF PL/M-88 COMPILATION

ISIS-11 PL/M-88 V3.1 COMPILATION OF MODULE INTERRUPT7

OBJECT MODULE PLACED IN :F1:INTER.OBJ

COMPILER INVOKED BY: PL/M88 :F1:INTER.PLM [XREF DEBUG DATE (3 OCT 78)]

#NOINTVECTOR

```

1      INTERRUPT7: DO;
2  1    DECLARE TRAIN BYTE EXTERNAL,
        ADDR2 LITERALLY '0D0H', /* ADDRESS TO WHICH WE SEND THE FOLLOWING
                                NON-SPECIFIC END OF INTERRUPT */
        OCK2E LITERALLY '20H'; /* THE NON-SPECIFIC EOI, SEE PAGE 3-100
                                AND PAGE 3-100 */

3  1    INTERRUPT$ROUTINE: PROCEDURE INTERRUPT 7 PUBLIC;
4  2      TRAIN=0; /* PROGRAM WILL CALL FOR RESULTS TO BE TYPED OUT */
5  2      OUTPUT(ADDR2) = OCK2E;
6  2      END INTERRUPT$ROUTINE;

7  1    END INTERRUPT7;
    
```

MODULE INFORMATION:

```

CODE AREA SIZE      = 0013H      190
VARIABLE AREA SIZE = 0000H      00
MAXIMUM STACK SIZE = 0000H      00
17 LINES READ
0 PROGRAM ERROR(S)
    
```

END OF PL/M-88 COMPILATION

LOC	OBJ	SEQ	SOURCE STATEMENT
		1 ;	ASSEMBLY LANGUAGE PROGRAM WRITTEN FOR THE UPI-41
		2 ;	(UNIVERSAL PERIPHERAL INTERFACE-41) DURING THE
		3 ;	SUMMER TERM OF ACADEMIC YEAR 77-78 BY THOMAS J
		4 ;	RIORDAN WHILE WORKING AS A GRADUATE ASSISTANT
		5 ;	FOR DR. HERBERT C. TOMLE AT THE NAVAL TRAINING
		6 ;	EQUIPMENT CENTER (NTEC) IN ORLANDO FLORIDA.
		7 ;	
		8 ;	
		9 ;	THE PROGRAM ACCEPTS A PARALLEL DATA TRANSFER FROM
		10 ;	AN OUTPUT PORT (8255) OF AN INTEL SBC-80/20-4
		11 ;	SINGLE BOARD COMPUTER SYSTEM. THE DATA WORD IS
		12 ;	DECODED TO OBTAIN A REFERENCE COLUMN ON THE FACE
		13 ;	OF AN ADM CRT. THE CRT CURSOR IS THEN POSITIONED
		14 ;	IN THAT COLUMN. THE DATA WORD IS FURTHER DECODED
		15 ;	TO OBTAIN THE ADDRESS IN ROM OF A TEXT STRING WHICH
		16 ;	IS THEN SHIFTED OUT SERIALY THROUGH AN I/O PORT
		17 ;	LINE OF THE UPI-41 AT 19200 BAUD. THE PROGRAM
		18 ;	IS INTERRUPT DRIVEN AND UTILIZES A FIFO STACK TO
		19 ;	BALANCE OUT DISPARITIES BETWEEN THE RATE AT WHICH IT
		20 ;	CAN SHIFT OUT SERIAL DATA AS COMPARED TO THE HIGHEST
		21 ;	POSSIBLE RATE AT WHICH IT MUST ACCEPT PARALLEL DATA.
		22 ;	
		23 ;	THE PUNTIME CONFIGURATION OF THE UPI-41 IS AS FOLLOWS:
		24 ;	
		25 ;	REGISTER BANK 0
		26 ;	
		27 ;	
		28 ;	REGISTER 0(R0) 7 BIT ASCII CODE COUNTER
		29 ;	REGISTER 1(R1) ASCII CHAR TO BE OUTPUT
		30 ;	REGISTER 2(R2) COUNT FOR VARIABLE DELAY
		31 ;	REGISTER 3(R3) OUTPUT STRING ADDRESS
		32 ;	REGISTER 4(R4) MASK VALUE FROM LOOKUP TABLE
		33 ;	REGISTER 5(R5) BINARY CODE FOR CRT COLUMN POSITION
		34 ;	REGISTER 6(R6) COUNTER FOR STRING OUTPUT
		35 ;	REGISTER 7(R7) PARALLEL DATA TRANSFER
		36 ;	
		37 ;	REGISTER BANK 1
		38 ;	
		39 ;	
		40 ;	REGISTER 0(R0) CURRENT DATA POINTER
		41 ;	REGISTER 1(R1) FINAL DATA POINTER
		42 ;	REGISTER 2(R2) QUEUE STATUS
		43 ;	REGISTER 3(R3) ACCUMULATOR STORAGE
		44 ;	REGISTER 4(R4) UNUSED
		45 ;	REGISTER 5(R5) CONSTANT=193D
		46 ;	REGISTER 6(R6) CONSTANT=224D
		47 ;	REGISTER 7(R7) TEMPORARY DATA WORD STORAGE
		48 ;	
		49 ;	
		50 ;	PORT 1 SERIAL TRANSMISSION ON BIT 0
		51 ;	PORT 2 LINES 0-4 USED AS A MASK INPUT TO INHIBIT TEXT
		52 ;	STRING OUTPUT. LINE 7 USED TO ENABLE CHIP SELECT

18 JAN 79

LOC	OBJ	SEQ	SOURCE STATEMENT
		53	;
		54	;
		55	;
		56	;
0000		57	ORG 0
0000 0400		58	JMP INIT ; PRESERVE INTERRUPT VECTORS
0003		59	ORG 30 ; EXTERNAL INTERRUPT VECTOR
0003 0450		60	EXTINT: JMP INROUT ; JUMP TO INTERRUPT ROUTINE
0007		61	ORG 70 ; TIMER INTERRUPT VECTOR
0007 0460		62	TIMINT: JMP TIINRT ; TIMER INTERRUPT ROUTINE
000A		63	ORG 100
000A 95		64	INIT: CPL F0 ; SET FLAG SO INTERRUPTS NOT ENABLED
		65	; DURING INITIALIZATION ROUTINE
000B 9A00		66	ANL P2, 00E0H ; NUMBER WILL DRIVE LINES 0-4 TO GROUND IN CASE
		67	; THE SIMULATION PGM IS GOING TO BE RUN I. E.
		68	; THIS WILL KEEP EACH RIFLE FROM PICKING UP
		69	; AN EXTRANEIOUS SHOT DUE TO THE OUTPUT LINES
		70	; COMING UP HIGH--00/20 0212 CLEAR WILL NOT
		71	; HAVE BEEN DONE AT THIS POINT IN TIME--
		72	; LINES 5 & 6 WILL BOTH BE HIGH AS REQUIRED
		73	; TO LET AN EXTERNAL SIGNAL CONTROL THE TAR PRES FLAG.
		74	; BUT WILL NOT ENABLE THE CHIP SELECT WHICH
		75	; IS TIED TO LINE 7
000D 0910		76	MOV R1, 01AH ; ASCII CHAR TO CLEAR CRT SCREEN
000F 3430		77	CALL OUTPUT
0011 3450		78	CALL DELAY1
0013 3410		79	CALL LOCSET ; SET UP CRT TO ACCEPT X COORD VALUE
0015 0920		80	MOV R1, 020H ; X VALUE FOR COLUMN 1
0017 3430		81	CALL OUTPUT ; ROUTINE TO SEND ASCII CHARACTER
0019 05		82	SEL R01
001A 0020		83	MOV R0, 0320 ; INITIAL VALUE FOR READ MEMORY POINTER
001C 0920		84	MOV R1, 0320 ; INITIAL VALUE FOR WRITE MEMORY POINTER
001E 0A00		85	MOV R2, 00 ; CLEAR QUEUE STATUS REGISTER
0020 0EE0		86	MOV R6, 02240 ; 224 + 32 AVAILABLE LOCATIONS IN RAM
		87	; = 256 => OVERFLOW
0022 00C1		88	MOV R5, 01930 ; 193 + 63(LAST RAM ADDRESS) = 256 =>
		89	; OVERFLOW
0024 9A60		90	ANL P2, 060H ; ENABLE CHIP SELECT
0026 05		91	EN I ; ENABLE EXTERNAL INTERRUPTS
0027 FA		92	WAIT: MOV A, R2 ; GET QUEUE STATUS
0028 0627		93	JZ WAIT ; IF QUEUE EMPTY NO ACTION
		94	;
		95	;
		96	;
		97	;
		98	;
002A F0		99	START: MOV A, 000 ; GET DATA FROM RAM LOCATION
002B AF		100	MOV R7, A ; STORE DATA
002C 0A		101	DEC R2 ; DECREMENT QUEUE STATUS REGISTER
002D FD		102	MOV A, R5 ; 193 DECIMAL
002E 68		103	ADD A, R0 ; CHECK FOR LAST ACCESS BEING @ TOP OF RAM
002F 9673		104	JNZ CONT
0031 001F		105	MOV R0, 0310 ; ONE LESS THAN BOTTOM OF RAM
0033 18		106	CONT: INC R0 ; NEXT RAM ACCESS LOCATION
0034 FF		107	MOV A, R7 ; RETRIEVE DATA

LOC	OBJ	SEQ	SOURCE STATEMENT
0035	C5	108	SEL R80
0036	AF	109	MOV R7, A ; STORE DATA
0037	4310	110	ORL A, 010H ; SET BIT WHICH IS HARD WIRED LOW
0039	37	111	CPL A
003A	963E	112	JNZ CONTM
003C	4410	113	JMP RIFSIM ; IF CODE FOR RIFLE SIMULATION ROUTINE
		114	; WAS SENT JUMP TO IT
003E	0A1F	115	CONTM: ORL P2, 01FH ; IF SIMULATION PGM NOT BEING RUN
		116	; THEN PORT 2 0-4 MUST BE INPUTS.
0040	FF	117	MOV A, R7 ; RETRIEVE DATA
0041	3405	118	CALL MASK ; CHECK TO SEE IF OUTPUT DESIRED
0043	0655	119	JFO ESCAPE ; IF FLAG SET WAIT FOR NEW DATA
0045	341A	120	CALL LOCSET ; SET UP ADM TO ACCEPT X-COORD VALUE
0047	FD	121	MOV A, R5 ; GET RIFLE ID FROM PROCEDURE MASK STORAGE
		122	; LOCATION
0048	3414	123	CALL TAB ; TAB OVER TO LOCATION CORRES TO RIFLE #
004A	FF	124	MOV A, R7 ; RETRIEVE 00/20 DATA
004B	47	125	SWAP A ; PUT CODE FOR TYPE OF SHOT
		126	; IN UPPER 4 BITS TO ALLOW ACCESS TO
		127	; 16 MEMORY LOCATIONS PER SHOT TYPE
004C	53F0	128	ANL A, 00F0H ; MASK OUT LOW ORDER BITS
004E	AB	129	MOV R3, A ; STORE RELATIVE ADDRESS OF CHAR STRING
004F	E3	130	MOV R3, A ; GET STRING LENGTH
0050	AE	131	MOV R6, A ; STORE COUNTER VALUE
0051	3427	132	CALL STROUT ; PROCEDURE TO OUTPUT ASCII STR
0053	345D	133	CALL CALF
0055	05	134	ESCAPE: SEL R81 ; RETURN TO CORRECT REG BANK FOR WAIT LOOP
0056	0427	135	JMP WAIT
		136 ;	
		137 ;	
		138 ;	
		139 ;	
		140 ;	
0058	05	141	INROUT: SEL R81 ; INTERRUPT REG BANK
0059	AB	142	MOV R3, A ; SAVE ACCUMULATOR
005A	FE	143	MOV A, R6 ; 2240
005B	6A	144	ADD A, R2
005C	0668	145	JZ QUEFUL ; CHECK FOR QUEUE FULL
005E	1A	146	INC R2 ; INCREMENT QUEUE STATUS REGISTER
005F	22	147	IN A, DBB ; INPUT DATA
		148	; FROM SR INTERRUPT STORE FF
0060	A1	149	MOV ARL, A ; STORE NEW DATA
0061	FD	150	MOV A, R5 ; 1930
0062	69	151	ADD A, R1 ; CHECK TO SEE IF STORE WAS
		152	; IN LAST AVAILABLE RAM LOCATION
0063	9667	153	JNZ CONT1 ; IF NOT THEN CONTINUE
0065	091F	154	MOV R1, 031D ; BOTTOM OF QUEUE
0067	19	155	CONT1: INC P1 ; INCREMENT WRITE POINTER REG
0068	FB	156	QUEFUL: MOV A, R3 ; RESTORE ACCUMULATOR
0069	93	157	RETR ; RETURN FROM INTERRUPT
006A	65	158	TIINRT: STOP TCNT ; PREVENT FURTHER TIMER OVERFLOW
006B	C7	159	MOV A, PSW ; THIS SEQUENCE OF OPERATIONS ALLOWS
006C	07	160	DEC A ; THE RETURN ADDRESS WHICH WAS STORED
006D	5307	161	ANL A, 007H ; ON THE STACK TO BE ALTERED SO THAT
006F	E7	162	RL A ; ON RETURN IT WILL NOT CONTINUE IN

LOC	OBJ	SEQ	SOURCE STATEMENT
0070	0308	163	ADD A, 00 ; THE SAME LOOP IT WAS IN WHICH FORCED
0072	AB	164	MOV R0, A ; THE INTERRUPT CALL IN THE FIRST PLACE
0073	10	165	INC 000 ; PROCEDURE CAN BE FIGURED OUT BY REF
0074	10	166	INC 000 ; UPT-41 MANUAL PP. 2-8,9.
0075	2314	167	MOV A, 0LOW FAIL
0077	54AD	168	CALL FAIL
0079	93	169	RETR
007A	23F0	170	FINIS: MOV A, 0 LOW DONE ; TEST COMPLETE MESSAGE
007C	AB	171	MOV R3, A
007D	E3	172	MOV R3, A
007E	AE	173	MOV R6, A
007F	3427	174	CALL STROUT
0081	345D	175	CALL CRLF
0083	0483	176	HERE: JMP HERE ; WAIT FOR RESET(EXTERNAL)
		177	
		178	
		179	SUBROUTINES IN FIRST PAGE OF MEMORY
		180	
		181	
		182	
0100		183	ORG 2560
0100	01	184	MSKDAT: DB 10, 20, 40, 80, 160
0101	02		
0102	04		
0103	08		
0104	10		
0105	53E0	185	MASK: ANL A, 00E0H ; MASK OUT 5 LOW ORDER BITS
0107	47	186	SWAP A ; RIFLE ID IN BITS 1, 2, 3
0108	77	187	RR A ; IN BITS 0, 1, 2
0109	AD	188	MOV R5, A ; STORE RIFLE CODE
010A	07	189	DEC A ; RIFLE CODE(FILE) 0-4
010B	A3	190	MOV R4, A ; GET LOOKUP VALUE FOR CURRENT RIFLE
010C	AC	191	MOV R4, A ; STORE VALUE
010D	0A	192	IN A, P2 ; GET RIFLE MASK
010E	5C	193	ANL A, R4
010F	85	194	CLR F0
0110	5613	195	JNZ CONT2 ; JUMP IF RIFLE NOT MASKED
0111	93	196	CPL F0 ; SET FLAG INDICATING MASK
0112	53	197	CONT2: RET ; RETURN FROM SUBROUTINE
0114	17	198	TAB: INC A ; CREATE CORRECT DIGIT FOR HIGH BYTE
0115	47	199	SWAP A ; PUT IN HIGH BYTE
0116	A9	200	MOV R1, A
0117	3430	201	CALL OUTPUT
0118	83	202	RET ; RETURN FROM SUBROUTINE
011A	6918	203	LOCSET: MOV R1, 018H ; ASCII ESCAPE (REF ADM MANUAL
		204	RE CURSOR POSITIONING)
011C	3430	205	CALL OUTPUT
011E	893D	206	MOV R1, 030H ; ASCII EQUALS
0120	3430	207	CALL OUTPUT
0122	8937	208	MOV R1, 037H ; ROW 24 OF ADM TERMINAL
0124	3430	209	CALL OUTPUT
0126	83	210	RET ; RETURN FROM SUBROUTINE
0127	1B	211	STROUT: INC R3 ; GET TO LOCATION OF TEXT BEGINNING
0128	FB	212	MOV A, R3 ; RETRIEVE PAGE 3 ADDRESS (F
		213	ASCII STRING

LOC	OBJ	SEQ	SOURCE STATEMENT
0129	E3	214	MOVP3 A,0A ;GET ASCII CHARACTER
012A	A9	215	MOV R1,A
012B	3430	216	CALL OUTPUT
012D	EE27	217	DJNZ R6,STROUT ;HAVE ALL CHAR BEEN OUTPUT
012F	83	218	RET ;RETURN FROM SUBROUTINE
0130	15	219	OUTPUT: DIS I
0131	8007	220	MOV R0,007H ;SERIAL BIT COUNTER
0133	F9	221	MOV A,R1 ;GET ASCII CHARACTER TO BE OUTPUT
0134	9900	222	ANL P1,000H ;PUT OUT START BIT
0136	0A04	223	MOV R2,004H ;SET UP DELAY LOOP LENGTH
0138	344D	224	CALL DELAY
013A	39	225	LOOP1: OUTL P1,A ;OUTPUT CURRENT BIT OF SERIAL CODE
013B	77	226	RR A ;GET NEXT BIT OF ASCII CODE
013C	00	227	NOP ;WAIT 1 INSTRUCTION CYCLE TO COMPENSATE
		228	;FOR RR BEING A SINGLE CYCLE OPERATION
013D	0A02	229	MOV R2,002H ;SET UP DELAY LOOP LENGTH
013F	344D	230	CALL DELAY
0141	E83A	231	DJNZ R0,LOOP1 ;TEST FOR 7 BITS OUTPUT
0143	8901	232	ORL P1,001H ;PUT OUT STOP BIT
0145	0A03	233	MOV R2,003H ;SET UP DELAY LOOP LENGTH
0147	344D	234	CALL DELAY
0149	064C	235	JF0 NOINEN ;IF IN SETUP SEGMENT DONT ENABLE INTERRUPTS
014B	05	236	EN I
014C	83	237	NOINEN: RET ;RETURN FROM SUBROUTINE
014D	EA4D	238	DELAY: DJNZ R2,DELAY ;VARIABLE DELAY DEPENDING ON R2
014F	83	239	RET ;RETURN FROM SUBROUTINE
0150	0996	240	DELAY1: MOV R1,0150H ;NESTED DELAY LOOP, APPROX 1/4 SEC
0152	0AFF	241	DLOOP: MOV R2,00FFH ;
0154	344D	242	CALL DELAY
0156	E952	243	DJNZ R1,DLOOP
0158	83	244	RET
0159	FE	245	PULBIT: MOV A,R6 ;CURRENT RIFLE NUMBER
015A	07	246	DEC A ;CREATE POINTER FOR LOOKUP TABLE
015B	A3	247	MOVP A,0A ;GET BYTE WITH CORRECT PULSE BIT SET
015C	83	248	RET
015D	090A	249	CRLF: MOV R1,00AH ;LINE FEED
015F	3430	250	CALL OUTPUT
0161	0900	251	MOV R1,000H ;CARRIAGE RETURN
0163	3430	252	CALL OUTPUT
0165	83	253	RET
0166	27	254	CHECK: CLR A ;TIMER STARTING COUNT FOR TIMEOUT
		255	;THE PROCESSOR WILL BE INTERRUPTED
		256	;IF THE 00/20 DOESN'T RESPOND WITHIN
		257	;A SPECIFIED TIME.
0167	62	258	MOV T,A ;LOAD TIMER
0168	55	259	STRT T ;START TIMER
0169	5669	260	LOOK: JTI LOOK ;UPI WILL LOOK FOR RESET UNTIL
		261	;TIMEOUT HAS OCCURED.
016B	65	262	STOP TONT ;INTERRUPT MUST NOT OCCUR
016C	090A	263	MOV R1,0100 ;GIVE 00/20 SUFFICIENT TIME TO
016E	0AFF	264	LOOP2: MOV R2,00FFH ;SEND OUT VOTRAX INITIALIZATION WORDS
0170	344D	265	CALL DELAY
0172	E96E	266	DJNZ R1,LOOP2
0174	D600	267	NOINTR ;RESPONSE FROM 00/20?
0176	22	268	IN A,00 ;BRING IN 00/20 DATA

LOC	OBJ	SEQ	SOURCE STATEMENT
0177	DC	269	XRL A,R4 ; IF IDENTICAL RESULT IS ZERO
0178	C684	270	JZ NEXT1 ; THEN GO ON WITH TEST
017A	231A	271	MOV A,# LOW FAIL2
017C	54AD	272	CALL FAIL
017E	2484	273	JMP NEXT1
0180	2324	274	NOINTR: MOV A,# LOW FAIL3
0182	54AD	275	CALL FAIL ; INDICATE FAILURE
0184	B93A	276	NEXT1: MOV RL,#180 ; GIVE 88/20 TIME TO SEND REMAINING
		277	PORTION OF MESSAGE
0186	BAFF	278	LOOPV: MOV R2,#0FFH
0188	344D	279	CALL DELAY
018A	E986	280	DJNZ R1,LOOPV
018C	83	281	RET
018D	FE	282	RIFLOP: MOV A,R6 ; RETRIEVE RIFLE NUMBER
018E	E7	283	RL A ; CODE IN BITS 1,2,&3
018F	4381	284	ORL A,#01H ; NO START BIT ON SERIAL OUT LINE
0191	39	285	OUTL P1,A ; SET UP MUX
0192	FE	286	MOV A,R6 ; GET CURRENT RIFLE
0193	47	287	SWAP A ; PUT IN HIGH BYTE
0194	E7	288	RL A ; UPPER 3 BITS
0195	4D	289	ORL A,R5 ; CREATE CORRECT RETURN CODE
0196	AC	290	MOV R4,A ; TEMP STORE
0197	3459	291	CALL PULBIT ; GET BYTE WITH CORRECT BIT SET
		292	FOR RIFLE TRIGGER
0199	4340	293	ORL A,#40H ; KEEP TARGET PRESENT DOWN
019B	3A	294	OUTL P2,A ; RISING EDGE OF TRIGGER PULSE
019C	9A40	295	ANL P2,#40H ; FALLING EDGE OF PULSE
019E	3466	296	CALL CHECK
01A0	83	297	RET
		298 ;	
		299 ;	DRIVER FOR RIFLE SIMULATION AND ITS MESSAGES LOCATED
		300 ;	IN FOURTH PAGE OF MEMORY
		301 ;	
		302 ;	
		303 ;	THIS SEGMENT OF THE UPI-PROGRAM PROVIDES SIMULATED
		304 ;	RIFLE DATA INPUT TO THE 88/20 COMPUTER. IT CHECKS
		305 ;	FOR THE PROPER RETURN BYTE TO THE UPI-41 FOR THE
		306 ;	SIMULATED SHOT AND INDICATES FAILURES BY A MESSAGE
		307 ;	TO THE CONSOLE. INITIALLY IT SIGNS ON AND PROMPTS
		308 ;	THE USER FOR THE HARDWARE MODIFICATIONS NECESSARY
		309 ;	IF A FAILURE OCCURS THE TEST WILL CONTINUE AND OUT-
		310 ;	PUTS 1 FAILURE MESSAGE FOR EACH FAILURE OCCURENCE
		311 ;	WHEN THE TEST IS COMPLETE IT PROMPTS THE USER TO
		312 ;	ASK THE 88/20 FOR ITS OUTPUT.
		313 ;	
		314 ;	
		315 ;	THE PROGRAM IS NON-INTERRUPT DRIVEN AND INSTEAD USES
		316 ;	THE INTERRUPT FLAG TO DETERMINE WHEN VALID DATA IS PRESENT
		317 ;	ON THE 88/20 DATA LINES. UPON ENTERING THE ROUTINE
		318 ;	FLAG 0 IS SET SO THAT INTERRUPTS WILL NOT BE REENABLED
		319 ;	WHEN THE OUTPUT ROUTINE IS CALLED IF THE 88/20 DOES
		320 ;	NOT RESPOND AT ALL TO AN INPUT BY THE UPI-41(INDICATED
		321 ;	BY THE INTERRUPT FLAG NEVER BEING SET) THE 41 TIMES
		322 ;	OUT AND JUDGES THIS AS A FAILURE AND CONTINUES THE TEST
		323 ;	

LOC	OBJ	SEQ	SOURCE STATEMENT
		324 ;	
		325 ;	REGISTER BANK 1 IS USED FOR THE ROUTINE AND IS REDefined
		326 ;	AS FOLLOWS:
		327 ;	
		328 ;	REGISTER 0 UNUSED
		329 ;	REGISTER 1 OUTER LOOP OF DELAY COUNTER
		330 ;	REGISTER 2 INNER LOOP OF DELAY COUNTER
		331 ;	REGISTER 3 DELAY COUNTER
		332 ;	REGISTER 4 EXPECTED RETURN DATA FROM 00/20
		333 ;	REGISTER 5 TEMP STORAGE
		334 ;	REGISTER 6 5 RIFLE LOOP COUNTER AND CURRENT RIFLE
		335 ;	REGISTER 7 16 SHOT POSSIBILITIES LOOP COUNTER
		336 ;	AND CURRENT SHOT TYPE
		337 ;	
		338 ;	THE BOTTOM 5 LINES OF PORT TWO FUNCTION AS THE RIFLE
		339 ;	TRIGGERS INSTEAD OF AS THE MASK INPUTS.
		340 ;	
		341 ;	LINE SIX OF PORT TWO IS THE TARGET PRESENT SIGNAL
		342 ;	
		343 ;	PORT 1 LINES 4-7 SERVE AS THE SHOT TYPE INPUT LINES FOR
		344 ;	THE 00/20.
		345 ;	
		346 ;	
0200		347	ORG 5120 ; PAGE 2
		348 ;	
0200 00		349	SHTCOD: DB 0, 2, 4, 3, 6, 0, 5, 4
0201 02			
0202 04			
0203 03			
0204 06			
0205 00			
0206 05			
0207 04			
0208 00	350	DB	8, 9, 0, 2, 7, 8, 6, 1
0209 09			
020A 00			
020B 02			
020C 07			
020D 08			
020E 06			
020F 01			
		351 ;	
		352 ;	
0210 05	353	RIFSIN: CLR	F0 ; WHILE F0 IS ALREADY SET AT THIS POINT
	354		; THIS ADDS A LITTLE CLARITY, THE POINT
	355		; IS THAT INTERRUPTS CANNOT BE REENABLED
0211 95	356	CPL	F0 ; WHEN THE SERIAL OUTPUT ROUTINE IS ENTERED
	357		
0212 23 0	358	MOV	A, 00C0H ;
0214 3A	359	OUTL	P2, A ; DISALLOW FURTHER INTERRUPT REQUESTS
	360		; OR INTERRUPT FLAG SETS BY DeselectING
	361		; THE CHIP. TARGET FLAG DOWN AND TRIGGERS DOWN.
	362		; AND TAR PRES CONTROL SET FOR UP1-41 CONTROL
0215 15	363	DIS	1 ; WHEN CHIP RESELECTED INTERRUPTS WILL
	364		; BE CHECKED THROUGH THE INT. FLAG.

LOC	OBJ	SEQ	SOURCE STATEMENT	
0216	25	365	EN	TONT1
		366		
0217	891A	367	MOV	R1, 01AH
0219	3430	368	CALL	OUTPUT
0218	3450	369	CALL	DELAY1
021D	3450	370	CALL	CRLF
021F	3450	371	CALL	CRLF
0221	3450	372	CALL	CRLF
0223	2300	373	MOV	R, 0 LOW SIGNON
0225	AB	374	MOV	R3, A
0226	E3	375	MOV	R6, A
0227	AE	376	MOV	R6, A
0228	3427	377	CALL	STRUT
022A	3450	378	CALL	CRLF
022C	3450	379	CALL	CRLF
022E	23E0	380	MOV	R, 0 LOW PROMPT
0230	AB	381	MOV	R3, A
0231	E3	382	MOV	R6, A
0232	AE	383	MOV	R6, A
0233	3427	384	CALL	STRUT
0235	3450	385	CALL	CRLF
0237	3450	386	CALL	DELAY1
0239	3450	387	CALL	DELAY1
023B	3450	388	CALL	DELAY1
023D	D5	389	TEST:	SEL
023E	9A00	390	AND	P2, 000H
		391		
0240	BE05	392	RLOOP5: MOV	R6, 005H
		393		
0242	BF10	394	RLOOP1: MOV	R7, 016
		395		
0244	54C0	396	RLOOP2: CALL	SALOP
0246	EF44	397	DJNZ	R7, RLOOP2
0248	EE42	398	DJNZ	R6, RLOOP1
024A	8A40	399	ORL	P2, 040H
024C	8B05	400	MOV	R3, 050
024E	3450	401	LOOPX: CALL	DELAY1
0250	EB4E	402	DJNZ	R3, LOOPX
0252	9A00	403	TOLATE: AND	P2, 000H
0254	8AFF	404	MOV	R2, 0AFFH
0256	3440	405	CALL	DELAY
0258	8A40	406	ORL	P2, 040H
025A	8AFF	407	MOV	R2, 0AFFH
025C	3440	408	CALL	DELAY
025E	BE05	409	MOV	R6, 05
0260	800A	410	MOV	R5, 00AH
0262	3480	411	LOOPC: CALL	RIFLOP
		412		
		413		
		414		
0264	EE62	415	DJNZ	R6, LOOPC
0266	8B05	416	MOV	R3, 050
0268	3450	417	LOOPL: CALL	DELAY1
026A	EB68	418	DJNZ	R3, LOOPL
026C	BE05	419	TARIGN: MOV	R6, 050

LOC	OBJ	SEQ	SOURCE STATEMENT	
026E	3450	420	LOOPD: CALL	DELAY1
0270	9A00	421	ANL	P2, 000H
0272	0A0F	422	MOV	R2, 00FFH
0274	3440	423	CALL	DELAY
0276	3459	424	CALL	PULBIT
0278	43E0	425	ORL	A, 01100000B
027A	37	426	CPL	A
027B	3A	427	OUTL	P2, A
027C	9901	428	ANL	P1, 001H
027E	9A00	429	ANL	P2, 000
0280	3450	430	CALL	DELAY1
0282	3450	431	CALL	DELAY1
0284	22	432	IN	A, 00B
		433		
0285	0A40	434	ORL	P2, 0A0H
0287	0006	435	MOV	R3, 00D
0289	3450	436	LOOPK: CALL	DELAY1
028B	EB09	437	DJNZ	R3, LOOPK
028D	D69D	438	JNIBF	NOINT
028F	FE	439	MOV	A, R6
0290	E7	440	RL	A
0291	47	441	SWAP	A
0292	430C	442	ORL	A, 00CH
0294	AC	443	MOV	R4, A
0295	22	444	IN	A, 00B
0296	DC	445	XRL	A, R4
0297	C6A1	446	JZ	CONTA
0299	231A	447	MOV	A, 0 LOW FAIL2
029B	449F	448	JMP	CONTB
029D	2324	449	NOINT: MOV	A, 0 LOW FAIL3
029F	54AD	450	CONTB: CALL	FAIL
02A1	EE6E	451	CONTA: DJNZ	R6, LOOPD
02A3	BE05	452	MINOTR: MOV	R6, 05
02A5	0000	453	MOV	R5, 000H
02A7	348D	454	LOOPJ: CALL	RIFLOP
02A9	EEA7	455	DJNZ	R6, LOOPJ
02AB	047A	456	JMP	FINIS
		457		
		458		
		459		
		460		
		461		
		462		
		463		
		464		
02AD	C5	465	FAIL: SEL	R00
02AE	AF	466	MOV	R7, A
02AF	D5	467	SEL	R01
02B0	FE	468	MOV	A, R6
02B1	C5	469	SEL	R00
02B2	E7	470	RL	A
02B3	43F0	471	ORL	A, 00F0H
02B5	AD	472	MOV	R5, A
02B6	A3	473	MOV	A, 0A
02B7	A9	474	MOV	R1, A

; TARGET PRESENT UP
; GIVE 00/20 TIME TO RESPOND
; TOP 3 BITS MUST BE ZERO AFTER CPL
; ALL BUT ONE RIFLE WILL SHOOT
; RISING EDGE OF TRIGGERS
; HAVE DATA SET FOR HITS, NO START BIT
; FALLING EDGE
; GIVE 00/20 TIME TO SEND ALL MESSAGES
; CLEAR IBF FLAG WHICH THE 00/20 SENT 4
; PULSES TO.
; TARGET PRESENT DOWN
; DELAY OVER 1 SECOND SO THAT UP1 WILL BE
; CERTAIN TO HAVE RECEIVED INTER PULSE
; IF NO INT PULSE INDICATE A FAILURE
; GET CURRENT RIFLE
; DATA IN BITS 3,2,1.
; IN BITS 7,6,5.
; C IS ADDRESS OF "TARGET IGNORED."
; STORE
; INPUT DATA
; IF RESULT IS ZERO THEN BYTES IDENTICAL
; PREPARE TO CALL DATA FAILURE ROUT.
; 5 RIFLES
; CODE FOR NO TARGET
; SEND RIFLE IDENTIFIER TO CRT
; STORE FAILURE TYPE
; GET CURRENT RIFLE
; MUL BY TWO TO ACCESS TWO LOCATIONS AT A TIME
; ACCESS EXH
; TEMP STORE
; GET RIFLE IDENTIFIER

LOC	OBJ	SEQ	SOURCE STATEMENT
02B9	3430	475	CALL OUTPUT
02BA	FD	476	MOV R,R5 ;RETRIEVE POINTER
02BB	17	477	INC R ;ACCESS NEXT LOCATION
02BC	R3	478	MOV R,R0 ;GET REST OF IDENTIFIER
02BD	R9	479	MOV R,R1
02BE	3430	480	CALL OUTPUT
02C0	B920	481	MOV R,R1 ;ASCII SPACE
02C2	3430	482	CALL OUTPUT
02C4	FF	483	MOV R,R7 ;RETRIEVE FAILURE TYPE
02C5	R8	484	MOV R,R3 ;NOW SEND OUT FAILURE TYPE TO CRT
02C6	E3	485	MOV R,R0
02C7	FE	486	MOV R,R6
02C8	3427	487	CALL STROUT
02C9	345D	488	CALL CRLF
02CD	93	489	RETR
02CD	FF	490	SMLOOP: MOV R,R7 ;GET SHOT TYPE
02CE	07	491	DEC R
02CF	37	492	CPL R
02D0	530F	493	ANL R,R0FH ;SHOT DATA LINES HAVE INVERTING DRIVERS
02D2	47	494	SWAP R ;SHOT TYPE DATA LINES ARE P1 4-7
02D3	ND	495	MOV R,R5 ;TEMP STORE
02D4	FE	496	MOV R,R6 ;GET CURRENT RIFLE
02D5	E7	497	RL R ;PUT CODE IN BITS 1,2,43
02D6	4D	498	ORL R,R5 ;OR SHOT TYPE & RIFLE # TOGETHER
02D7	4301	499	ORL R,R01H ;DON'T SEND OUT A START BIT
02D9	39	500	OUTL P1,R ;SET UP SHOT DATA LINES AND
		501	;INPUTS TO THE MULTIPLEXER.
02DA	FF	502	MOV R,R7 ;RETRIEVE SHOT TYPE
02DB	07	503	DEC R
02DC	R3	504	MOV R,R0 ;GET CORRECT RETURN CODE FOR COMPARISON
02DD	AD	505	MOV R,R5 ;STORE
02DE	FE	506	MOV R,R6 ;GET CURRENT RIFLE
02DF	47	507	SWAP R ;PUT CODE IN UPPER 4 BITS
02E0	E7	508	RL R ;UPPER 3 BITS
02E1	4D	509	ORL R,R5 ;CREATE EXPECTED RETURN CODE
02E2	AC	510	MOV R,R4 ;STORE
02E3	3459	511	CALL PULBIT ;GET FROM PAGE 1 A BYTE WHICH WILL
		512	;HAVE THE CORRECT BIT SET FOR A
		513	;TRIGGER PULL BY THIS RIFLE
02E5	3A	514	OUTL P2,R ;RISING EDGE OF PULSE
02E6	27	515	CLR R
02E7	3A	516	OUTL P2,R ;FALLING EDGE OF PULSE. NOTE: THIS
		517	;MAY OR MAY NOT BE THE CASE IN REALITY
		518	;FOR IF THE MASK SWITCH FOR THE CURRENT
		519	;RIFLE IS ON, THEN THE ACTUAL PULSE
		520	;WILL ONLY BE THE MUCH SHORTER PULSE(500
		521	;NSEC) THAT THE UPI PROVIDES AS A FAST
		522	;TURN ON FOR THE PORT BEFORE THE 50K
		523	;PULLUP TAKES EFFECT. HERE THE 50K
		524	;PULLUP WILL NOT PROVIDE A HIGH OUTPUT
		525	;DUE TO THE 5K PULDOWN AND THE OUTL
		526	;INSTRUCTION WOULD NOT BE NEEDED
		527	;THE LOGIC 'HIGH' WILL ACTUALLY BE
		528	;ABOUT 2.5V WHICH IS ACCEPTABLE TO THE 41
02E8	3466	529	CALL CHECK

LOC	OBJ	SEQ	SOURCE STATEMENT
02EA	03	530	RET
		531	
		532	
02F2		533	ORG 2F2H
		534	
02F2	5231	535	DB 'R1','R2','R3','R4','R5'
02F4	5232		
02F6	5233		
02F8	5234		
02FA	5235		
		536	
		537	
		538	
		539	
		540	
		541	
		542	
		543	STRING DATA LOCATED IN THIRD PAGE OF MEMORY
		544	
		545	
0300		546	ORG 7680
0300	04	547	DB 4H 'MISS'
0301	40495353		
0305	20202020		
0309	20202020		
030D	202020		
0310	03	548	DB 3H 'HIT'
0311	404954		
0314	05	549 FAL1:	DB 05H 'F RES'
0315	46205245		
0319	53		
031A	05	550 FAL2:	DB 05H 'F DAT'
031B	46204441		
031F	54		
0320	03	551	DB 3H 'LOW'
0321	404F57		
0324	05	552 FAL3:	DB 05H 'F INT'
0325	4620494E		
0329	54202020		
032D	202020		
0330	09	553	DB 9H 'LOW RIGHT'
0331	404F5720		
0335	52494748		
0339	54202020		
033D	202020		
0340	05	554	DB 5H 'RIGHT'
0341	52494748		
0345	54202020		
0349	20202020		
034D	202020		
0350	0A	555	DB 10H 'HIGH RIGHT'
0351	40494748		
0355	20524947		
0359	40542020		
035D	202020		

LOC	OBJ	SEQ	SOURCE STATEMENT
0360	04	556	DB 4M 'HIGH
0361	40494748		
0365	20202020		
0369	20202020		
0360	202020		
0370	09	557	DB 9M 'HIGH LEFT
0371	40494748		
0375	204C4546		
0379	54202020		
0370	202020		
0380	04	558	DB 4M 'LEFT
0381	4C454654		
0385	20202020		
0389	20202020		
0380	202020		
0390	08	559	DB 8M 'LOW LEFT
0391	4C4F5720		
0395	4C454654		
0399	20202020		
0390	202020		
03A0	0D	560	DB 130 'MISS-TOO LATE
03A1	40495353		
03A5	2D544F4F		
03A9	204C4154		
03A0	452020		
03B0	09	561	DB 9M 'NO TARGET
03B1	4E4F2054		
03B5	41524745		
03B9	54202020		
03B0	202020		
03C0	0E	562	DB 140 'TARGET IGNORED
03C1	54415247		
03C5	45542049		
03C9	474E4F52		
03CD	454420		
03D0	0F	563	SIGNON: DB 150 'RIFLE SIMULATOR'
03D1	5249464C		
03D5	45205349		
03D9	40554C41		
03D0	544F52		
03E0	0F	564	PROMPT: DB 150 'STRAP IN PLACE?'
03E1	53545241		
03E5	5020494E		
03E9	20504C41		
03ED	43453F		
03F0	0F	565	DONE: DB 150 ' TEST COMPLETE'
03F1	20205445		
03F5	53542043		
03F9	4F40504C		
03FD	455445		
		566	END

USER SYMBOLS

CHECK	0166	CONT	0033	CONT1	0067	CONT2	0113	CONTA	02A1	CONTB	025F	CONTH	003E	CRLF	015D
DELAY	014D	DELAY1	0150	LOOP	0152	DONE	03F0	ESCAPE	0055	EXTINT	0003	FAIL	02A0	FAIL	0314

1515-11 MCS-40/UP1-41 MACRO ASSEMBLER, V2.0
18 JAN 79

PAGE 13

FAL2 031A	FAL3 0324	FINIS 007A	HERE 0083	INIT 000A	INROUT 0050	LOCSET 011A	LOOK 0169
LOOP1 013A	LOOPC 0262	LOOPD 026E	LOOPJ 02A7	LOOPK 0289	LOOPL 0268	LOOPX 024E	LOOPY 0186
LOOP2 016E	NRSK 0105	MINOTR 02A3	MSKDAT 0100	NEXT1 0184	NOIMEN 014C	.NOINT 029D	NOINTR 0180
OUTPUT 0130	PROMPT 03E0	PULBIT 0159	QUEFUL 0068	RIFLOP 018D	RIFSIN 0210	RLOOP1 0242	RLOOP2 0244
RLOOP5 0240	SHTCOD 0200	SIGNON 0300	SALoop 02CD	START 002A	STROUT 0127	TAB 0114	TARIGN 026C
TEST 023D	TIINRT 006A	TIMINT 0007	TOLATE 0252	WAIT 0027			

ASSEMBLY COMPLETE, NO ERRORS

1515-11 LOCATER VOLTAGE MONITOR
 -LOCATE :F1:TST.TMP TO F1:TST SYMBOLS LINES MAP PRINT(:F1:TST PPT)

SYMBOL TABLE OF MODULE TST
 READ FROM FILE :F1:TST.TMP
 WRITTEN TO FILE :F1:TST

VALUE TYPE SYMBOL

	MOD	TESTMODULE
C1BFH	SYM	MEMORY
C001H	SYM	TEST
C001H	LIN	14
C001H	LIN	15
C004H	LIN	16
C007H	LIN	17
C00AH	LIN	18
C00DH	LIN	19
C010H	LIN	20
C013H	LIN	21
	MOD	TESTPROCMODULE
C1BFH	SYM	MEMORY
C000H	SYM	TSTCHECK
3900H	SYM	N
C018H	SYM	DONE
C018H	SYM	LEDON
3901H	SYM	K
C020H	SYM	FAIL
3902H	SYM	J
3903H	SYM	IODATA
C08DH	SYM	IOTEST
C014H	SYM	LOWLIMIT
C016H	SYM	HIGHLIMIT
3904H	SYM	INITIALTIME
3906H	SYM	FINALTIME
3908H	SYM	ELAPSEDTIME
390AH	SYM	I
C00DH	SYM	TIMERTEST
C138H	SYM	USARTTEST
C018H	LIN	41
C018H	LIN	42
C018H	LIN	43
C01CH	LIN	44
C01FH	LIN	45
C020H	LIN	47
C024H	LIN	49
C033H	LIN	50
C041H	LIN	51
C045H	LIN	52
C04AH	LIN	53
C051H	LIN	54
C05FH	LIN	55
C064H	LIN	56
C068H	LIN	57
C072H	LIN	58
C080H	LIN	59
C085H	LIN	60
C08CH	LIN	61
C08DH	LIN	65
C08DH	LIN	67
C090H	LIN	68
C094H	LIN	69
C099H	LIN	70
C09EH	LIN	71
C0A3H	LIN	72
C0A7H	LIN	73

C080H	LIN	74
C081H	LIN	75
C086H	LIN	76
C08AH	LIN	77
C0BFH	LIN	78
C0C4H	LIN	79
C0C9H	LIN	80
C0CDH	LIN	81
C0D2H	LIN	82
C0D7H	LIN	83
C0DCH	LIN	85
C0DDH	LIN	90
C0DDH	LIN	91
C0E0H	LIN	92
C0E5H	LIN	93
C0EBH	LIN	94
C0F9H	LIN	95
C0FEH	LIN	96
C105H	LIN	97
C10BH	LIN	98
C117H	LIN	99
C121H	LIN	100
C126H	LIN	101
C132H	LIN	102
C137H	LIN	103
C138H	LIN	106
C138H	LIN	107
C13BH	LIN	108
C140H	LIN	109
C143H	LIN	110
C148H	LIN	111
C14CH	LIN	112
C151H	LIN	113
C15BH	LIN	114
C160H	LIN	115
MOD	RAMTST	
C168H	SYM	LOOP
C184H	SYM	LOOPI
C17FH	SYM	RAMFAL
C161H	SYM	RAMTST
MOD	ROMTST	
C1A9H	SYM	CONTZ
C194H	SYM	LOOPI
C18EH	SYM	ROMTST
MOD	SBCTIM	
C1B6H	SYM	LOOPI
C1B5H	SYM	LOOPB
C1B3H	SYM	SBCTIM

MEMORY MAP OF MODULE TST
 READ FROM FILE :F1:TST.TMP
 WRITTEN TO FILE :F1:TST
 MODULE START ADDRESS 0026H

START STOP LENGTH REL NAME

0000H	C08CH	8DH	A	ABSOLUTE
0080H	C1BEH	132H	A	ABSOLUTE

PL/M-88 COMPILER

1515-11 PL/M-88 V3.1 COMPILATION OF MODULE TESTMODULE

OBJECT MODULE PLACED IN :F1:TESTER.OBJ

COMPILER INVOKED BY: PLM88 :F1:TESTER.PLM DEBUG 1XREF DATE (12 01T 78)

/* THIS TEST PROGRAM WAS WRITTEN BY TOM RIORDAN ITS FUNCTION
IS TO ACT AS THE DRIVER FOR THE TEST PROCEDURES AS CALLED */

```

1      TESTMODULE: DO;
2  1    RANTST: PROCEDURE EXTERNAL;
3  2    END RANTST;
4  1    RONTST: PROCEDURE EXTERNAL;
5  2    END RONTST;
6  1    IOSTEST: PROCEDURE EXTERNAL;
7  2    END IOSTEST;
8  1    TIMERSTEST: PROCEDURE EXTERNAL;
9  2    END TIMERSTEST;
10 1    USARTSTEST: PROCEDURE EXTERNAL;
11 2    END USARTSTEST;
12 1    DONE: PROCEDURE EXTERNAL;
13 2    END DONE;

14 1    TEST: PROCEDURE PUBLIC;
15 2      CALL RANTST;
16 2      CALL RONTST;
17 2      CALL IOSTEST;
18 2      CALL TIMERSTEST;
19 2      CALL USARTSTEST;
20 2      CALL DONE;
21 2    END TEST;

22 1    END TESTMODULE;

```

MODULE INFORMATION:

```

CODE AREA SIZE      = 0013H      190
VARIABLE AREA SIZE = 0000H       00
MAXIMUM STACK SIZE = 0002H       20
29 LINES READ
0 PROGRAM ERROR(S)

```

END OF PL/M-88 COMPILATION

1515-11 PL/M-80 V3.1 COMPILATION OF MODULE TESTPROCMODULE
 OBJECT MODULE PLACED IN :F1:TSTPRC.OBJ
 COMPILER INVOKED BY: PLM80 :F1:TSTPRC.PLN IXREF DEBUG DATE (5 JUL 79)

```

1      TESTPROCMODULE: DO;

2  1    DECLARE TSCHECK BYTE PUBLIC AT (0C00H) DATA(1);
      #NLIST

38  1    DECLARE N BYTE;

39  1    DECLARE WORD LITERALLY 'ADDRESS';

40  1    DECLARE DONTCARE LITERALLY '00H', FOREVER LITERALLY 'WHILE 1',
      DIAGNOSTICLED LITERALLY '006H';

41  1    DONE: PROCEDURE PUBLIC;
42  2      LED#ON;
      DO FOREVER;
43  3          OUTPUT(DIAGNOSTICLED)=DONTCARE;
44  3          END LED#ON;
45  2      END DONE;

46  1    DECLARE K BYTE;
47  1    FAIL: PROCEDURE(J) PUBLIC;
48  2      DECLARE J BYTE;
49  2      DO K=1 TO J;
50  3          DO N=1 TO 10;
51  4              OUTPUT(DIAGNOSTICLED)=DONTCARE;
52  4              CALL SBCTIM(10);
53  4          END;
54  3          DO N=1 TO 40;
55  4              CALL SBCTIM(125);
56  4          END;
57  3      END;
58  2      DO N=1 TO 80;
59  3          CALL SBCTIM(250);      /* WAIT 2 SECONDS THEN GO ON WITH TEST */
60  3      END;
61  2      END FAIL;

62  1    DECLARE IOFAIL LITERALLY '3';      /* 3 FLASHES FOR AN I/O FAILURE */
63  1    DECLARE PORT1 LITERALLY '0E4H', PORT2 LITERALLY '0E5H',
      PORT3 LITERALLY '0E6H', PORT6 LITERALLY '0E7H';

64  1    DECLARE IODATA BYTE;
65  1    IO$TEST: PROCEDURE PUBLIC;
66  2      DO;
67  3          CALL PORT$SET; /* SET UP IO PORTS 1&2 AS INPUTS 3&6 AS OUTPUTS */

68  3          OUTPUT(PORT3)=00H; /* PORT 3 WILL INVERT OUTPUT THEN PORT 1 WILL REINVERT IT */
69  3          IODATA=INPUT(PORT1);
70  3          IF IODATA<00H THEN
71  3              CALL FAIL(IOFAIL);

72  3          OUTPUT(PORT3)=0FFH;
73  3          IODATA=INPUT(PORT1);
74  3          IF IODATA<0FFH THEN
  
```

```

75 3      CALL FAIL(I0$FAIL);

76 3      OUTPUT(PORT6)=00H; /* PORT 6 INVERTS OUTPUT BIT PORT 2 WILL NOT REINVERT */
77 3      I0DATA=INPUT(PORT2);
78 3      IF I0DATA<0FFH THEN
79 3          CALL FAIL(I0$FAIL);

80 3      OUTPUT(PORT6)=0FFH;
81 3      I0DATA=INPUT(PORT2);
82 3      IF I0DATA<00H THEN
83 3          CALL FAIL(I0$FAIL);
84 3      END;
85 2      END I0$TEST;

86 1      DECLARE LOWLIMIT WORD DATA(100), HIGHLIMIT WORD DATA(300);
87 1      DECLARE TIMER$FAIL$LOW LITERALLY '4', TIMER$FAIL$HIGH LITERALLY '5';
88 1      DECLARE (INITIALTIME, FINALTIME, ELAPSEDTIME) WORD;
89 1      DECLARE I BYTE;
90 1      TIMER$TEST: PROCEDURE PUBLIC;
91 2          CALL TIMER$START; /* START TIMERS 0 AND 1 */
92 2          CALL SBCTIM(250); /* GIVE TIMER TIME TO BEGIN FUNCTIONING */
93 2          INITIALTIME = CLOCKREAD;
94 2          DO I=1 TO 40; /* WAIT FOR ONE SECOND */
95 3              CALL SBCTIM(250);
96 3              END;
97 2          FINALTIME=CLOCKREAD;

98 2          ELAPSEDTIME = INITIALTIME - FINALTIME; /* COUNTERS ARE DOWN COUNTERS */
99 2          IF ELAPSEDTIME < LOWLIMIT THEN
100 2              CALL FAIL(TIMER$FAIL$LOW);
101 2          IF ELAPSEDTIME > HIGHLIMIT THEN
102 2              CALL FAIL(TIMER$FAIL$HIGH);

103 2      END TIMER$TEST;

104 1      DECLARE USART$FAIL LITERALLY '6';
105 1      DECLARE USART$STATUS LITERALLY '0EDH', USART$DATA LITERALLY '0EDH';
106 1      USART$TEST: PROCEDURE PUBLIC;
107 2          CALL VOTRAX$TIMER;
108 2          CALL SBCTIM(100); /* MAKE CERTAIN TIMER HAS STARTED */
109 2          CALL VOTRAX$SET; /* SET BAUD RATE AND BIT PATTERN */
110 2          CALL SBCTIM(100); /* MAKE CERTAIN USART HAS COMPLETED INTERNAL SETUP */
111 2          OUTPUT(USART$DATA)=101010100; /* SEND OUT TEST PATTERN */
112 2          CALL SBCTIM(20); /* WAIT APPROX 1.04 MSEC=USART SHOULD BE DONE */
113 2          /* N.B THIS MUST BE LONG ENOUGH EVEN WITHOUT WAIT STATES */
114 2          IF NOT SHR(INPUT(USART$STATUS), 2) THEN
115 2              CALL FAIL(USART$FAIL);
116 2      END USART$TEST;

116 1      END TEST$PROC$MODULE;

```

MODULE INFORMATION

CODE AREA SIZE = 0140H 3330

VARIABLE AREA SIZE = 0000H 110
MAXIMUM STACK SIZE = 0004H 40
138 LINES READ
0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION

11 OCT 78

LOC	OBJ	SEQ	SOURCE STATEMENT
		1	NAME RANTST
		2	STKLN 00H
		3	EXTRN DONE
		4	PUBLIC RANTST
		5	
		6	CSEG
0000	D1	7	RANTST: POP D ; GET RETURN ADDRESS THAT WAS PUSHED
		8	; BY THE CALL AND SAVE IT IN THE D0E
		9	; REG PAIR. IT WILL BE VALID IF RAM
		10	; IS OKAY, AND UNUSED OTHERWISE.
0001	0100F8	11	LXI B, 0F000H ; INCREMENTING THIS VALUE AND CHECKING
		12	; FOR OVERFLOW WILL INDICATE WHEN TEST
		13	; IS FINISHED.
0004	210038	14	LXI H, 3000H ; START OF RAM
0007	AF	15	LOOP: XRA A
0008	77	16	MOV M, A ; STORE 00H AT LOCATION
0009	7E	17	MOV A, M ; READ 00 FROM SAME LOCATION?
000A	B7	18	ORA A ; SET FLAGS
000B	C21E00	19	JNZ RANFAL ; IF 00 NOT READ BACK JUMP TO FAILURE ROUTINE
000E	2F	20	CMA ; IF PASSES THEN ACCUM=FFH
000F	77	21	MOV M, A ; STORE FFH AT LOCATION
0010	7E	22	MOV A, M ; READ FFH FROM SAME LOCATION?
0011	3C	23	INR A ; IF FFH READ BACK ACCUM=00
0012	C21E00	24	JNZ RANFAL ; IF FFH NOT READ BACK JUMP TO FAIL ROUT
0015	23	25	INX H ; ADDRESS NEXT MEMORY LOCATION
0016	03	26	INX B ; CHECK FOR TEST COMPLETE
0017	78	27	MOV A, B
0018	B7	28	ORA A
0019	C20700	29	JNZ LOOP
001C	D5	30	PUSH D ; PUT RETURN ADDRESS BACK ON STACK.
001D	C9	31	RET
001E	D3D6	32	RANFAL: OUT 006H ; FLASH LED 1 TIME TO INDICATE RAM FAILURE
0020	01E8FD	33	LXI B, 6500H ; DELAY APPROXIMATELY 1 SEC THEN JUMP TO
0023	00	34	LOOPA: DCX B ; DONE ROUTINE. THIS IS DONE BECAUSE THE
0024	00	35	; REST OF THE TESTS CANNOT BE RUN RELIABLY
0025	78	36	MOV A, E ; UNLESS THE RAM IS WORKING PROPERLY.
0026	B7	37	ORA A ;
0027	C22300	38	JNZ LOOPA ;
002A	C30000	39	JMP DONE
		40	END

PUBLIC SYMBOLS

RANTST C 0000

EXTERNAL SYMBOLS

NAME E 0000

UNDEF SYMBOLS

NAME E 0000 LOOP C 0007 LOOPA C 0023 RANFAL C 001E RANTST C 0000

ASSEMBLY COMPLETE. NO ERRORS

ASMB0 :F1:RONTST.SRC DEBUG MACROFILE TITLE('18 JUN 79')

IS15-11 0000/0005 MACRO ASSEMBLER, V2 0
18 JUN 79

RONTST PAGE 1

LOC	OBJ	SEQ	SOURCE STATEMENT
		1	NAME RONTST
		2	STKLN ON
		3	EXTRN FAIL
		4	PUBLIC RONTST
		5	
		6	CSEG
		7	
0000	210000	8	RONTST: LXI H, 0H ; START OF PROGRAM ROM
0003	110000	9	LXI D, 01000H ; START OF TEST ROM
0006	0E02	10	LOOPA MVI C, 2 ; 2 FLASHES INDICATES ROM FAILURE
0008	7E	11	MOV A, H ; READ PROGRAM ROM
0009	47	12	MOV B, A ; SAVE BYTE
000A	EB	13	XCHG
000B	7E	14	MOV A, H ; READ TEST ROM
000C	B8	15	CMP B ; A-B
000D	CA1000	16	JZ CONTZ ; IF A<0B THEN FAILURE HAS OCCURRED
0010	F5	17	PUSH PSH
0011	C5	18	PUSH B
0012	D5	19	PUSH D
0013	E5	20	PUSH H
0014	CD0000	21	CALL FAIL
0017	E1	22	POP H
0018	D1	23	POP D
0019	C1	24	POP B
001A	F1	25	POP PSH
001B	12	26	CONTZ INX D ; NEXT ROM LOCATIONS
001C	22	27	INX H
001D	7A	28	MOV A, D
001E	EB	29	XCHG
001F	FE14	30	CPI 14H ; IS TEST COMPLETE?
0021	CD0000	31	JNZ LOOPA
0024	C9	32	RET
		33	END

FUNCTION SYMBOLS
RONTST C 0000

EXTERNAL SYMBOLS
FAIL E 0000

USER SYMBOLS
INVT C 001B FAIL E 0000 LOOPA C 0004 RONTST C 0000

ASSEMBLY COMPLETE, NO ERRORS

ASM80 F1:SBCTIM.SRC DEBUG MACROFILE TITLE('23 OCT 78')

ISIS-II 0000/0005 MACRO ASSEMBLER V2.0
23 OCT 78

SBCTIM PAGE 1

LOC	OBJ	SEQ	SOURCE STATEMENT
		1	NAME SBCTIM
		2	STKLN 0H
		3	PUBLIC SBCTIM
		4	
		5	CSEG
0000	060A	6	SBCTIM: MVI B,10
0002	78	7	LOOPB: MOV A,B
0003	3D	8	LOOPA: DCR A
0004	C20300 C	9	JNZ LOOPA
0007	8D	10	DCR C
0008	C20200 C	11	JNZ LOOPB
000B	C9	12	RET
		13	END

PUBLIC SYMBOLS
SBCTIM C 0000

EXTERNAL SYMBOLS

USER SYMBOLS
LOOPA C 0003, LOOPB C 0002 SBCTIM C 0000

ASSEMBLY COMPLETE, NO ERRORS