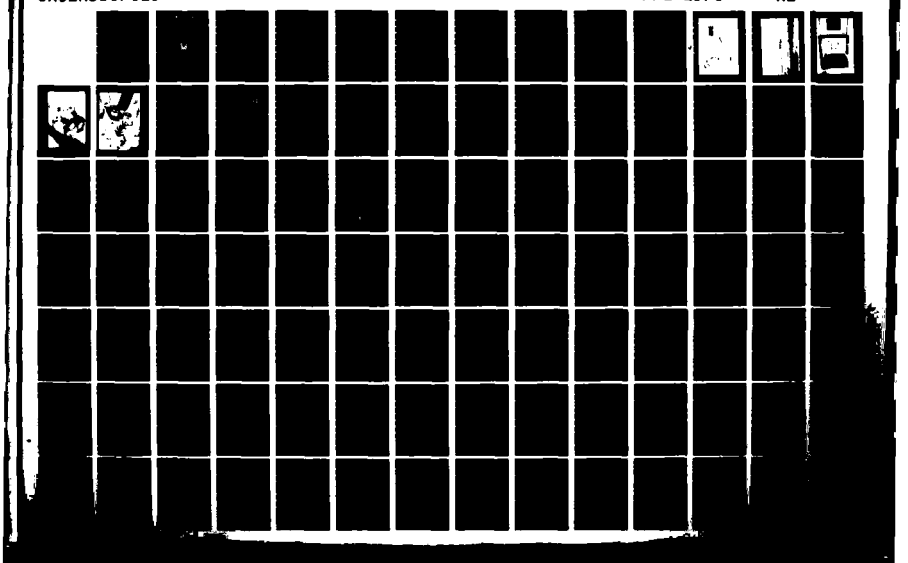
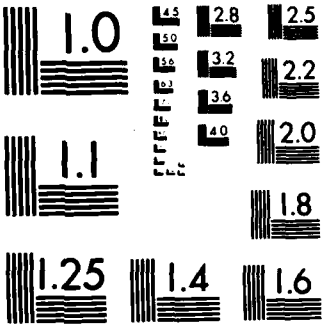
AD-A143 646 SIMULATED TANK ANTI-ARMOR GUNNERY SYSTEM (STAGS-TOW) 1/3
(U) NAVAL TRAINING EQUIPMENT CENTER ORLANDO FL
A MARSHALL ET AL. MAY 83 PMT-E-T-6605.83

UNCLASSIFIED F/G 19/5 NL





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

SIMULATED TANK ANTI-ARMOR GUNNERY SYSTEM (STAGS-T)

BY Albert Marshall
Dr. Herbert Towle
Bon Shaw
Gary Bond
Jeff Lohman
Ed Purvis

TOW TRAINER

PREPARED FOR

U.S. ARMY PROJECT MANAGER FOR TRAINING DEVICES
NAVAL TRAINING EQUIPMENT CENTER (NTEC)
ORLANDO, FLORIDA 32813

BY
NAVAL TRAINING EQUIPMENT CENTER
ADVANCED SIMULATION CONEPTS LABORATORY
SIMULATION TECHNOLOGY BRANCH
ORLANDO, FLA

MAY 1983

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

AD-A143646

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: PMT-E-T-6605.83	2. GOVT ACCESSION NO.: DA300399
3. RECIPIENT'S CATALOG NUMBER:	
4. TITLE (and Subtitle): Simulated Tank Anti-Armor Gunnery System (STAGS - TOW)	5. TYPE OF REPORT & PERIOD COVERED: Interim Report
	6. PERFORMING ORG. REPORT NUMBER:
7. AUTHOR(s): Albert Marshall, Bon Shaw, Dr. Herbert Towle, Gary Bond, Edward Purvis and Jeff Lohman	8. CONTRACT OR GRANT NUMBER(s): Project 1785
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Naval Training Equipment Center, N-73, Orlando, FL 32813	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: PE 62727A, Project 1X162727A230, Task Area A230
11. CONTROLLING OFFICE NAME AND ADDRESS: U.S. Army (DRCPM-TND-ET) (305/646-5881), Project Manager for Training Devices, Orlando, FL 32813	12. REPORT DATE: May 1983
	13. NUMBER OF PAGES:
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office):	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:

16. 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)

(U) Infantry	(U) Microprocessor	(U) Training Devices	(U) Simulation
(U) Anti-Armor	(U) TOW	(U) Target Acquisition	(U) Firepower
(U) Anti-Tank	(U) Multiprocessor	(U) Night Vision	(U) Battalion
(U) Missile	(U) Thermal Sight	(U) Concept Development	

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes a missile gunnery simulator research model developed to train TOW gunners. It is derived from a model previously developed for DRAGON. The system employs a terrain board with model enemy armored vehicles moving in a variety of attack scenarios. When the gunner fires the missile, he hears computer generated rocket sounds and experiences the smoke of the missile launch. When the smoke clears, he views both target and missile flare through his sight. The gunner's aiming error is measured using a

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S/N 0102-LF-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

microprocessor-controlled diode matrix array. The matrix detector senses an IR emitting light source which is located on the model target. The flight equations of motion for the missile are solved by a 16-bit microprocessor every 0.02 seconds. A second coordinated 16-bit processor controls a display that plots both vertical and horizontal aiming error for analysis of the gunner's performance by an instructor. Experienced TOW gunners have tested the system and attested to the realism and training potential.

The report also describes a method of simulating the TOW thermal sight for gunner training.

SUMMARY

The Simulated Tank Anti-Armor Gunnery System (STAGS) is a generic trainer developed to train TOW, DRAGON, STINGER/ILAW and other missile gunners at a reasonable cost.

This report describes the research-model/STAGS TOW. The Simulated Tank Anti-Armor Gunnery System (STAGS-T) employs multiple microprocessors to solve the TOW flight equations, generate graphics, generate weapon sounds, and control stepper motors that move three miniature models on a terrain board. A computer-generated voice and microprocessor-controlled sound system coordinate the training session by issuing the standard firing commands. When the trainee fires, he hears the weapon's ignition. While looking through the sight, he sees smoke at launch. When the smoke clears, he views the missile moving down range and the explosion at impact. The trainee can use either of the two simulated sights -- optical or thermal. Realistic thermal images are presented in the thermal sight. The instructor can view in real-time both a TV picture of the gunner's sight picture and a graphics display of gunner aiming error vs. range. The instructor can recall the missile's flight path vs. range.

This model has been evaluated by experienced TOW gunner teams from both the U.S. Marine Corps and U.S. Army. Assessment of these evaluators was that the task simulation is highly realistic and that the approach should yield a high level of training transfer.

A laboratory model was constructed by the Advanced Simulation Concepts Laboratory, Naval Training Equipment Center, Orlando, Florida, for the U.S. Army Project Manager for Training Devices (PM TRADE) and the U.S. Marine Corps.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS
(continued)

SECTION I

INTRODUCTION

This report describes a system using advanced electro-optics and microprocessor technology to enable training of TOW gunners at a reasonable cost. Anti-armor live-fire training is expensive. Each live round costs thousands of dollars. The user community finds that presently-fielded devices and simulators are inadequate for full gunnery training.

TOW is a crew-portable, heavy anti-tank weapon designed to attack and defeat armored vehicles and field fortifications. The missile is tube-launched, optically-tracked, wire command-link guided. The gunner guides the missile by tracking the target either through an optical telescope (day sight) or an infrared night vision sight (thermal sight). The thermal sight is a passive device that detects heat emissions (infrared energy) from a target area, converts the infrared energy to electrical signals and then to visible light and displays the visible light as a real-time scene for tracking by the gunner.

In the STAGS trainer, the targets are miniature scale model tanks on a terrain board. Movement of the tanks is accomplished using microprocessor-controlled stepper motors. A computer-generated voice and microprocessor-controlled sound system coordinate the training sessions by issuing the firing commands to the student; this subsystem also supports instructor-selectable real-time automated student coaching.

When the trainee fires the STAGS-T training device, he hears the gyro wind-up noise followed by the explosions of the rocket launch.

In the sight, he observes the smoke from launch and, when the smoke clears, he sees the missile flying down range. The visual explosion at the final missile impact is also inserted in the sight. In mission scenarios, one or more tank models maneuver over terrain and can move into cover. Multiple-threat scenarios have been designed to force students to select the greater threat or to practice target transfer.

After the mission is completed, the computer-generated voice unit tells the trainee where the missile impacted.

During missile flight, the instructor can view, in real-time, both a TV picture of the gunner sight picture and a graphics plot of gunner aiming error vs. range. The instructor can also recall the missile's flight path vs. range.

In summary, key features of STAGS-T are:

- Both the optical and thermal sights are simulated for all-situation training.
- Computer-controlled, computer-generated voice unit issues all firing commands to the trainee to initiate the firing sequence.

- Computer-controlled voice may both coach and debrief the trainee.
- Microprocessors solve TOW flight equations for realistic control training.
- Missile explosions as well as return fire are simulated.
- The student uses realistic missile reaction to in-flight gunnery commands.
- Highly-detailed models maneuver over a hilly terrain board, supporting training of both simple and complex tracking skills, including target transfer and -- for advanced gunnery -- "flying the missile" around obstructions.
- Cost of expensive targets and missile are not required.
- Video record and playback capability is available.
- Random target evasive maneuvers can be selected as a scenario choice.
- Smoke and obscurants may be inserted in the gunner's sight.
- Real-time feedback to the instructor include both the gunner's sight picture and gunner aiming error, GAE, vs. range.

Figures I-1 through I-5 show the STAGS-T laboratory model.

A similar system (STAGS-D) was developed for DRAGON. The DRAGON system was well accepted by the user community. STAGS-T also has been demonstrated to experienced gunners, with similar acceptance.

Figure I-1. STAGS TOW System.

Figure I-2. Terrain Board.

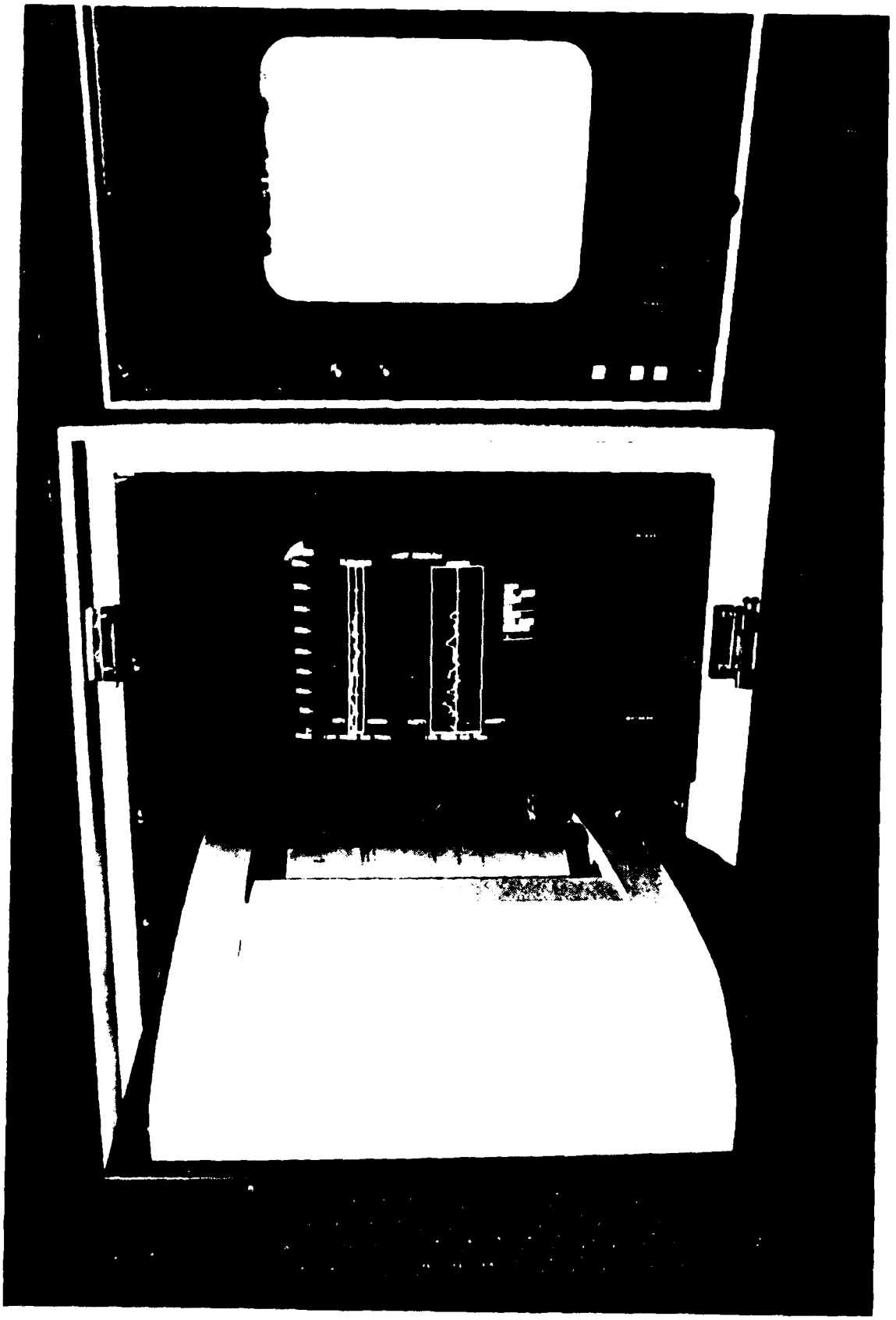


Figure I-3. Instructor's Console.

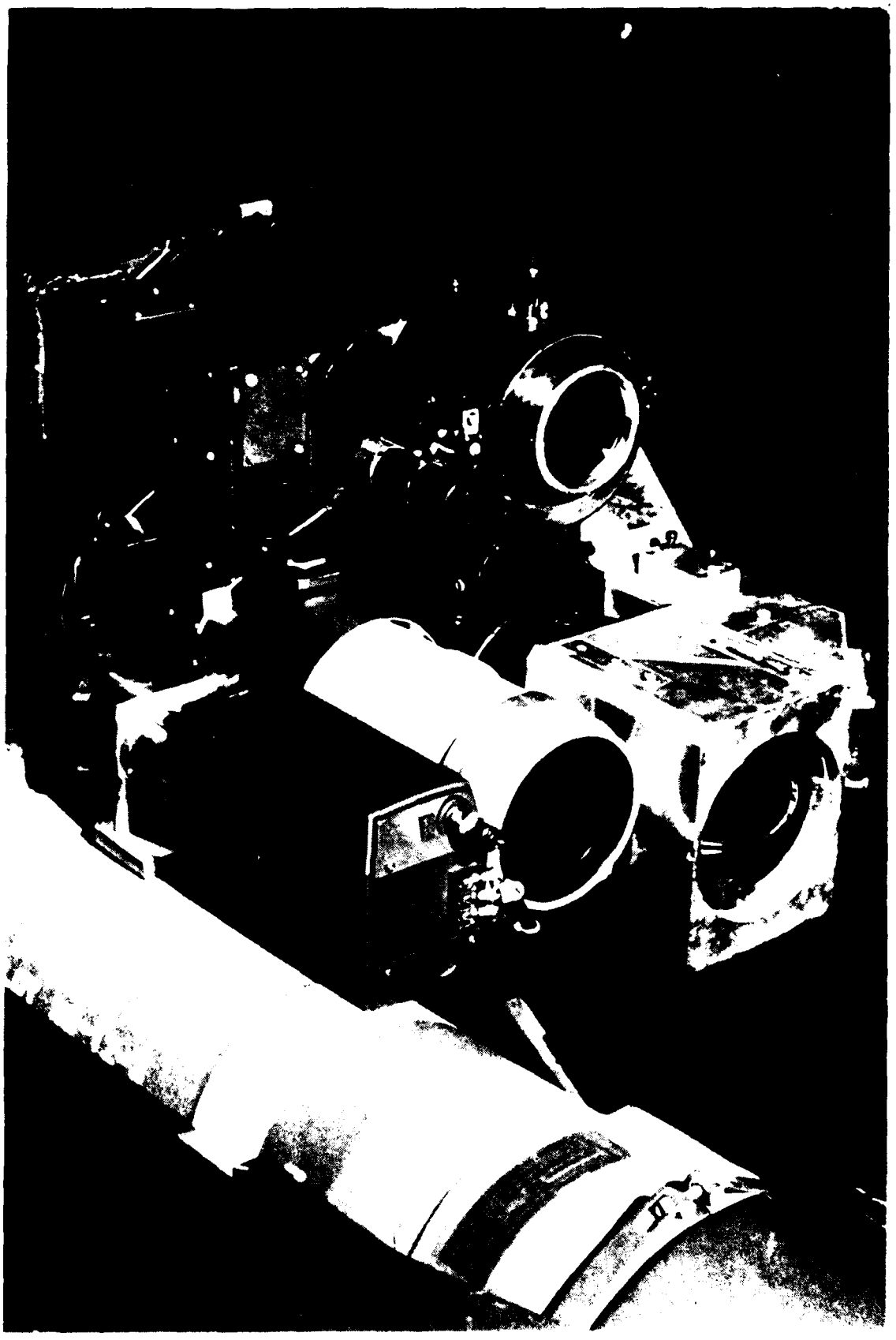


Figure 1-4. Day Sight and Thermal Sight (Front View).

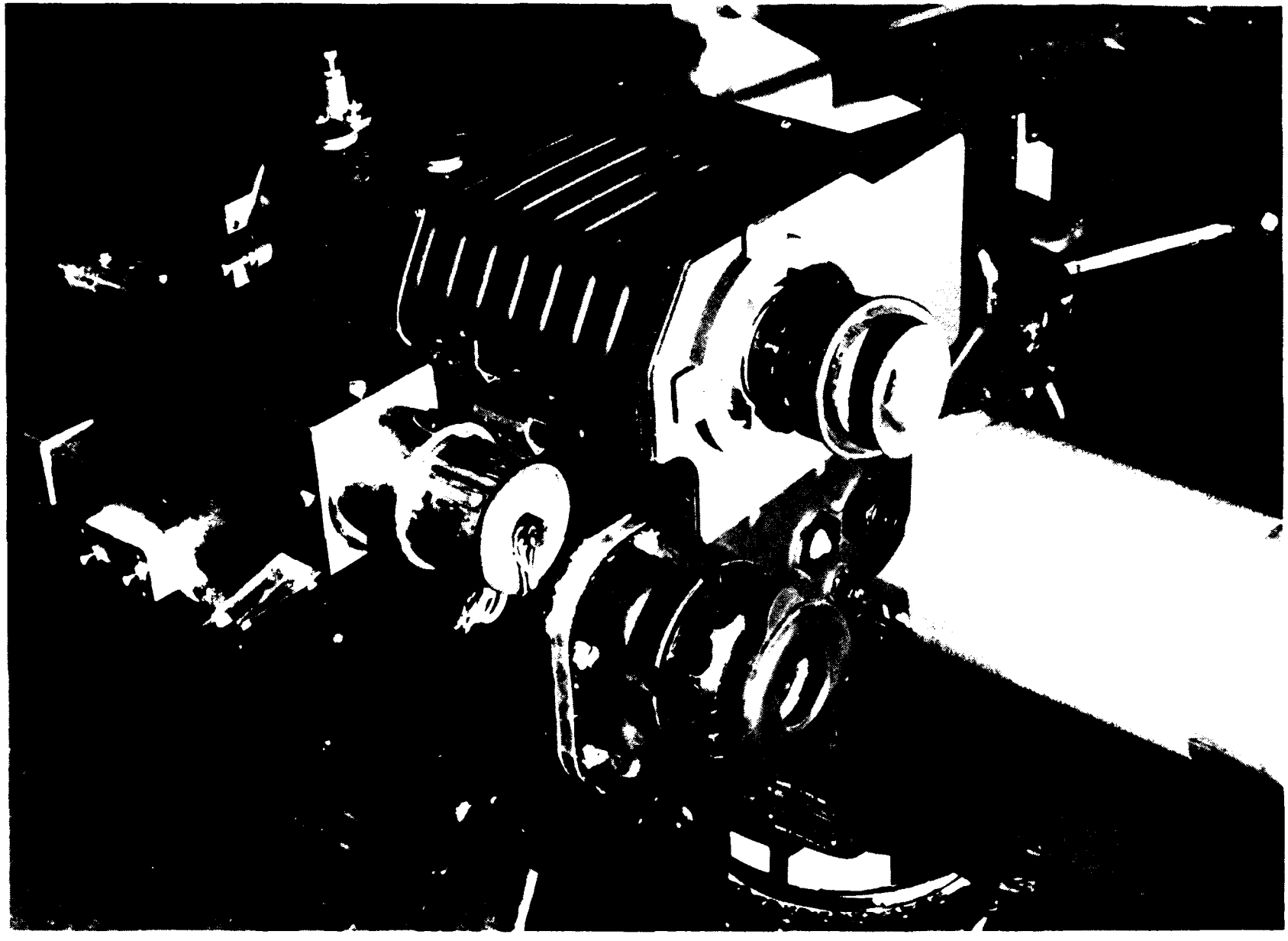


Figure I-5. Day Sight and Thermal Sight (Rear View).

SECTION II

SYSTEM DESCRIPTION

A system block diagram is shown in Figure II-1. The system consists of three basic components:

Terrain Board,
Student TOW Station, and
Instructor's Console.

The TOW utilizes a 13X sighting telescope and can engage targets beyond 3,000 meters. Computer-generated imagery, video-disc and movies were considered for the display system but they lacked the resolution required for a well-defined target at TOW ranges. To overcome such deficiencies, model-board presentation was selected. Targets consist of three (3) 1/285 scaled tank models. The models can rotate and simulate motion at oblique angles as well as toward the gunner or across his field of view. One model moves on a track as if on a hill; another model is behind the hill and moves in and out of the cover of the hill. The third model, controlled by a linear actuator, moves up and down so any degree of defilade can be simulated. This model can also "bounce" slowly up and down, giving an appearance of movement on a rough road. The scale models used in the lab model are commercially available "micro-armor" models and have excellent detail. The models are moved by intelligent positioning stepper motor controllers. The instructor selects engagement scenario and the Personnel Interface Processor (PIP) loads the program into an internal program buffer in the stepper motor controller. While moving the model, the controller functions independently of the PIP and executes the commands that were stored in the program buffer. Engagement scenarios are stored in the PIP and are selectable from the instructor's console by the input terminal. The velocity, direction and range of the tank targets is in the scenario program. A programmable output pin on the positioning stepper motor controller is used to synchronize the computer voice system which issues the firing orders to the trainee. Human-like speech firing commands are given the trainee by a National Semiconductor speech processor. Vocabulary words used in the system are stored in ROM of the speech synthesizer system. The operator can also select a "coach" function which coaches the trainee during the missile flight.

Gunner aiming errors (GAE) are determined using a 100 x 100 photodiode matrix camera located on the simulated TOW weapon and boresighted to the sight. The matrix camera views an infrared light source at the target; the output data is sent to a Missile Flight Simulator (MFS) processor to determine the GAE. Target location and velocity are also input to the missile flight system by monitoring the stepper motor pulses. The MFS processor solves the TOW flight equations and provides status to the PIP. The PIP controls the graphics units which inserts the smoke, missile and impact explosions graphics into the gunner's sight. This processor also controls the GAE display on the instructor's console. This display plots GAE vs. range in real-time. On a recall basis, missile position vs. range is plotted for debrief of the gunner.

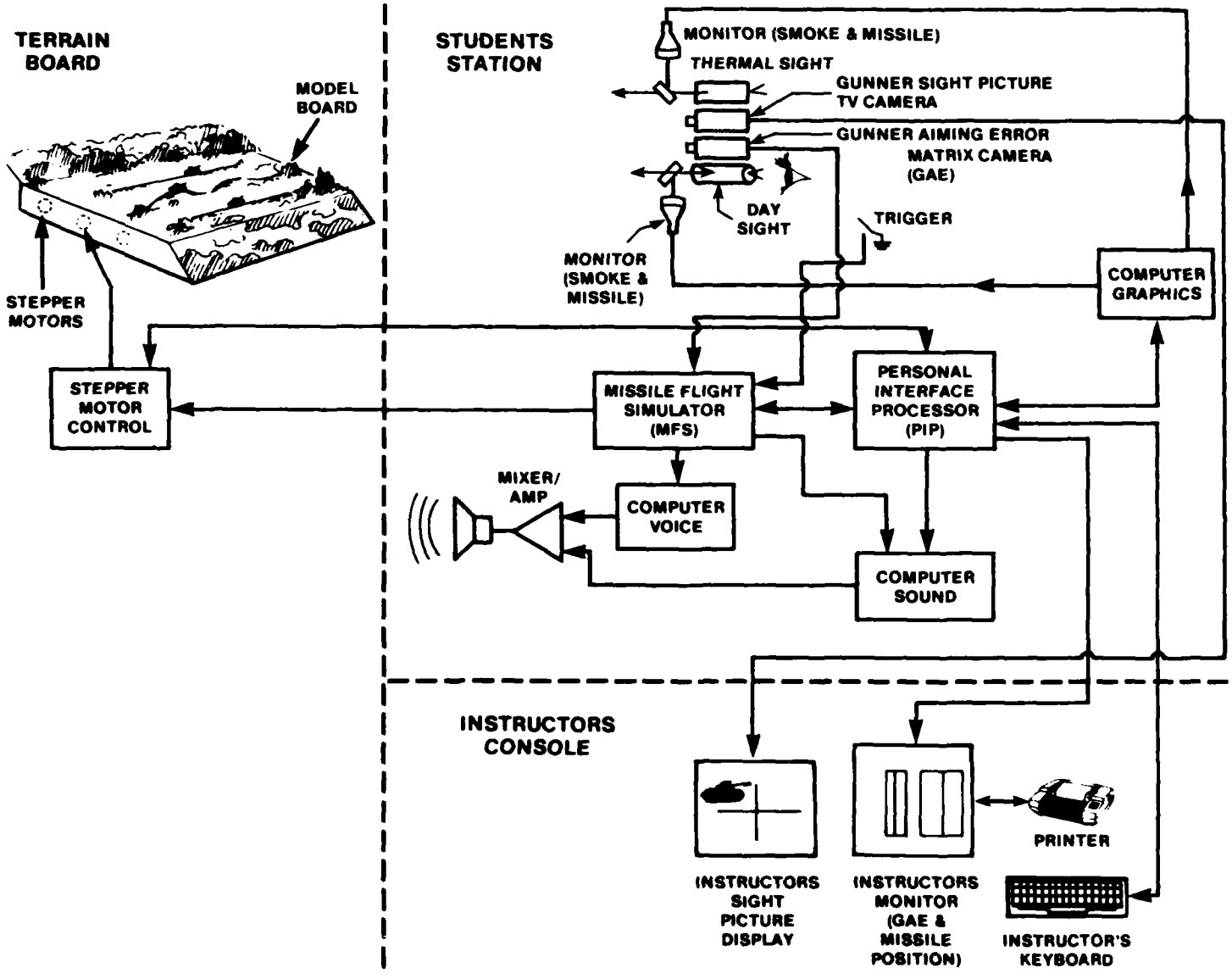


Figure II-1. System Block Diagram.

Both processors, the MFS and PIP, use the Intel 8086 microprocessor. The MFS also uses the 8087 math coprocessor. The 8087 coprocessor allows all flight equations to be solved using floating-point arithmetic in real-time and expands the graphics capability over the original STAGS-DRAGON.

The AN/TAS-4A night sight is also simulated so the gunner can be trained in the use of both day and thermal sights. Simulated thermal targets were developed and are discussed in Section III-F.

The MFS communicates with a General Instrument sound-producing integrated circuit which produces the gyro wind-up, rocket motor, and warhead explosions. Sounds are attenuated as a function of the distance of the missile from the trainee.

A TV screen on the instructor's console provides the instructor with the same view as seen through the gunner's sight. TV data are obtained from a CCTV camera located on and boresighted to the simulated TOW day sight.

Data printouts for both a hit and various types of misses are shown in Figures II-2 through II-13.

The instructor reviews, in real-time, gunner aiming error vs. range. He can recall, after the missile flight, the missile position vs. range. Various phenomena due to poor gunner performance are also simulated, i.e., lost guidance, etc.

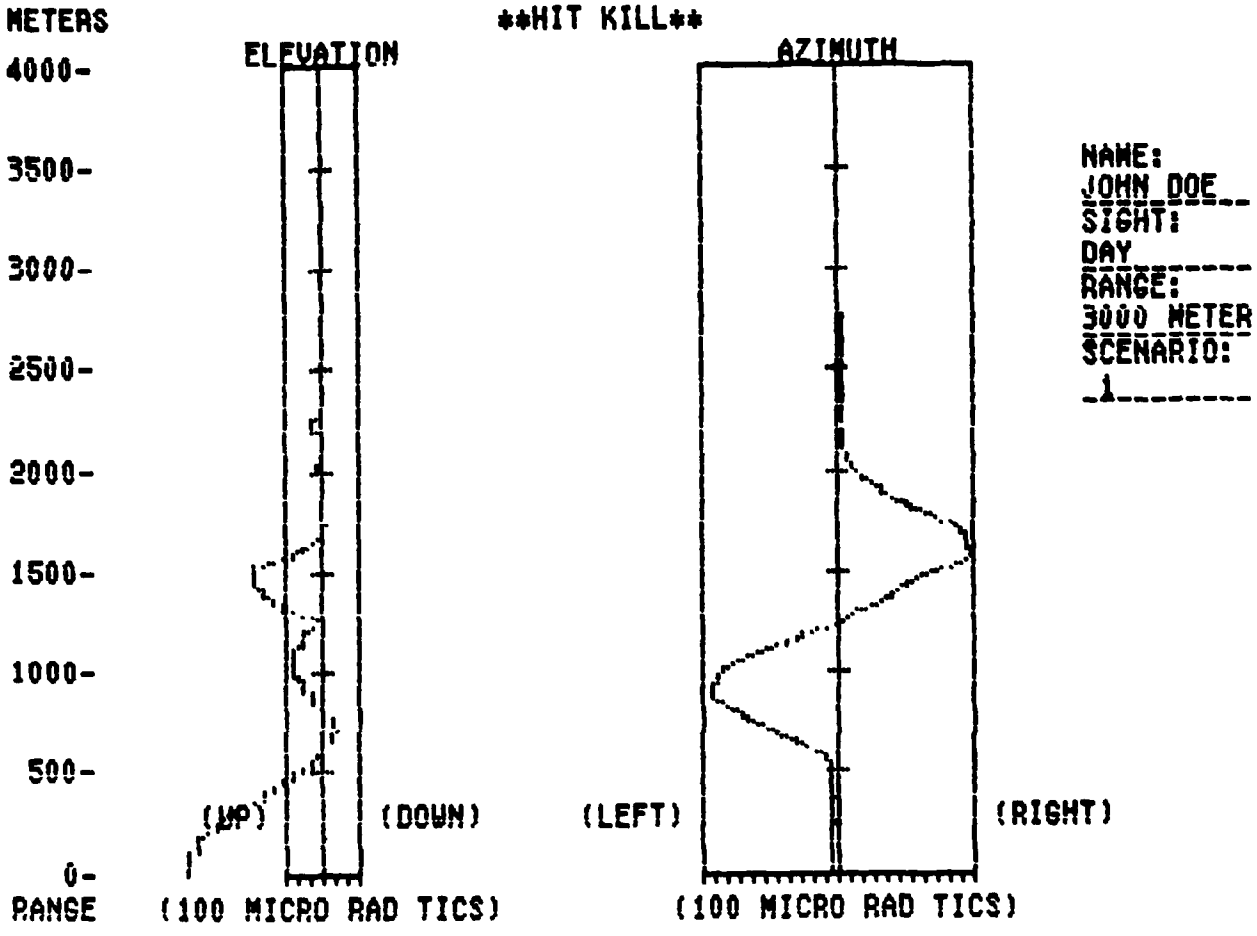


Figure II-2. Gunner Aiming Error vs. Range (Hit Kill).

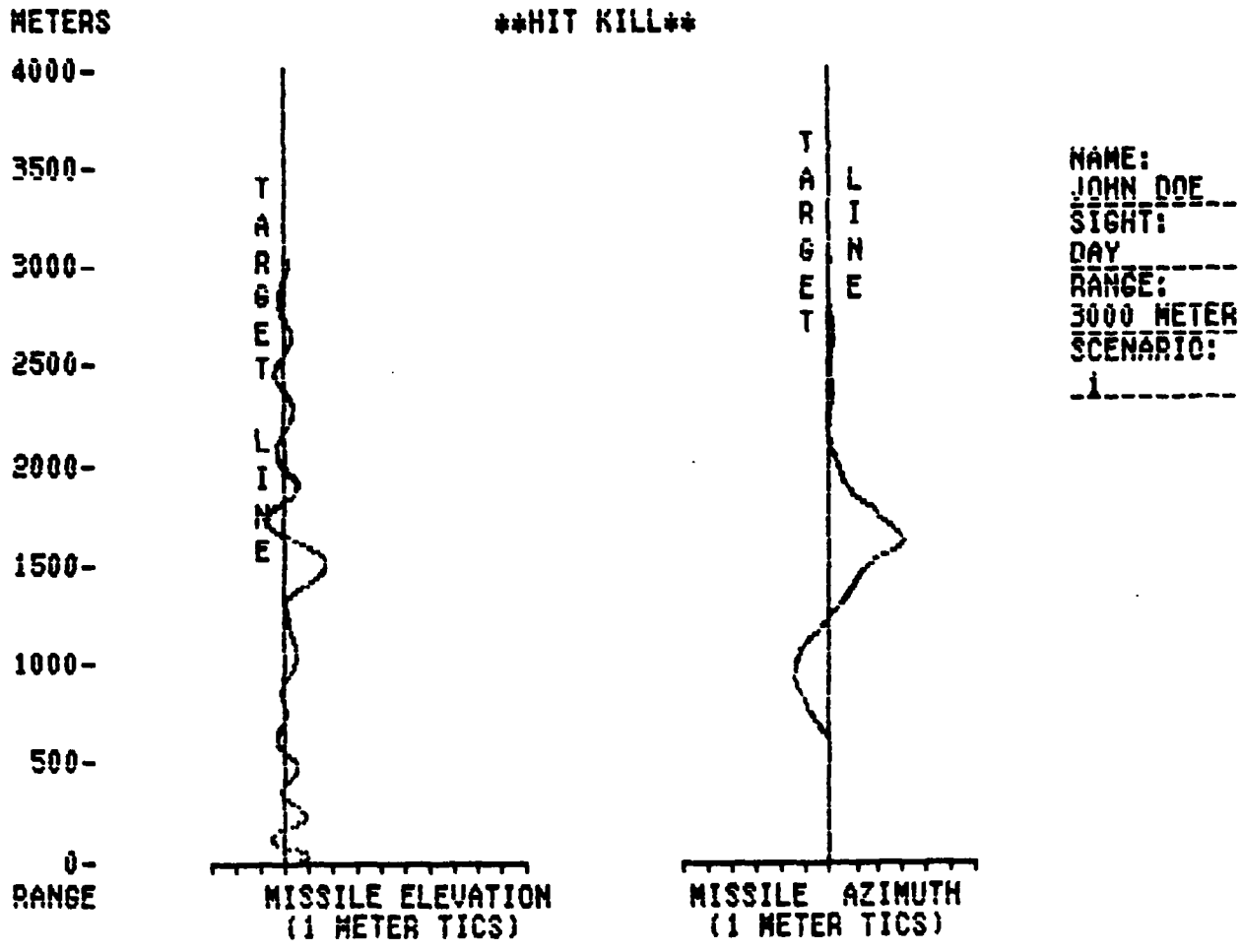


Figure II-3. Missile Position vs. Range (Hit Kill).

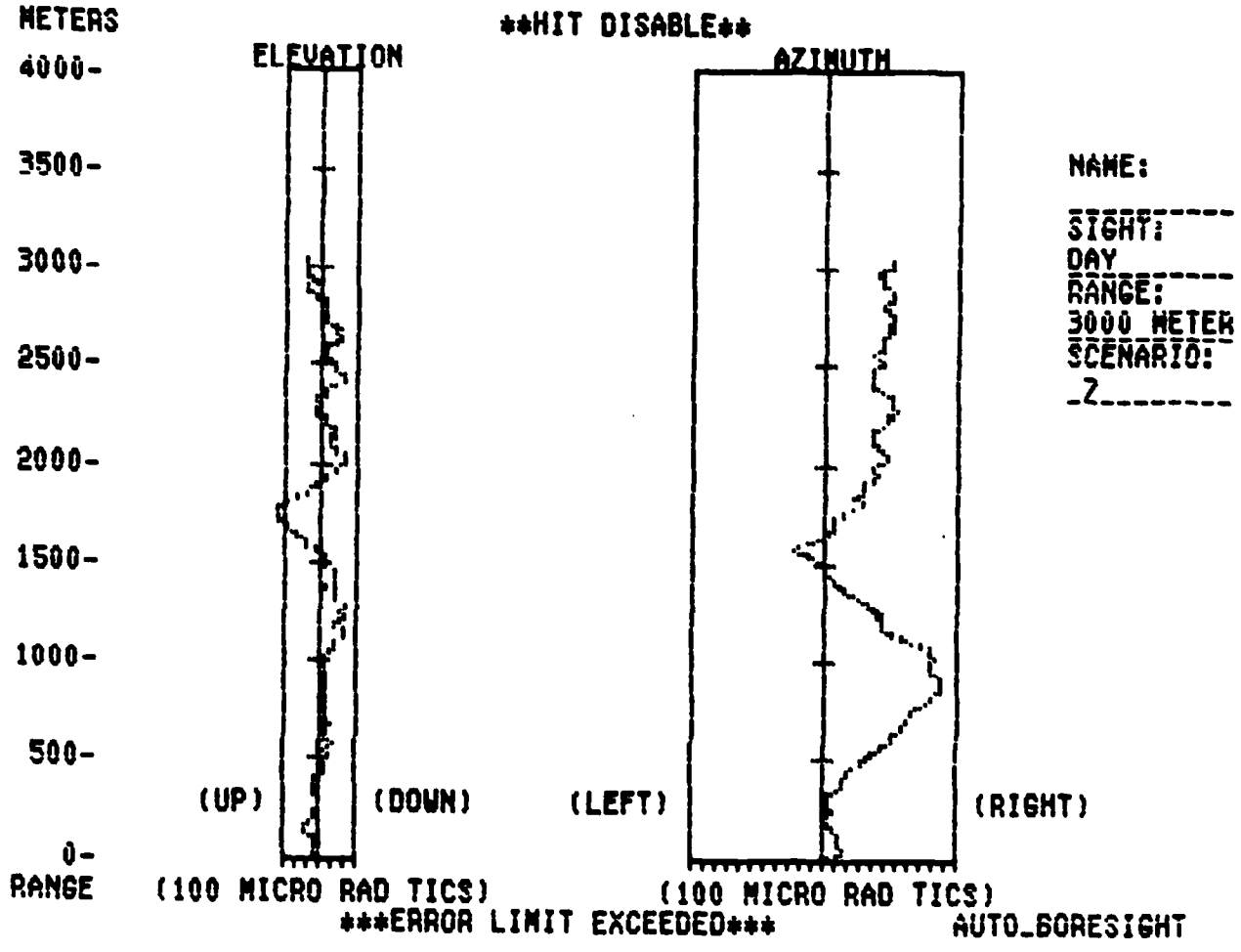


Figure II-4. Gunner Aiming Error vs. Range (Hit Disable).

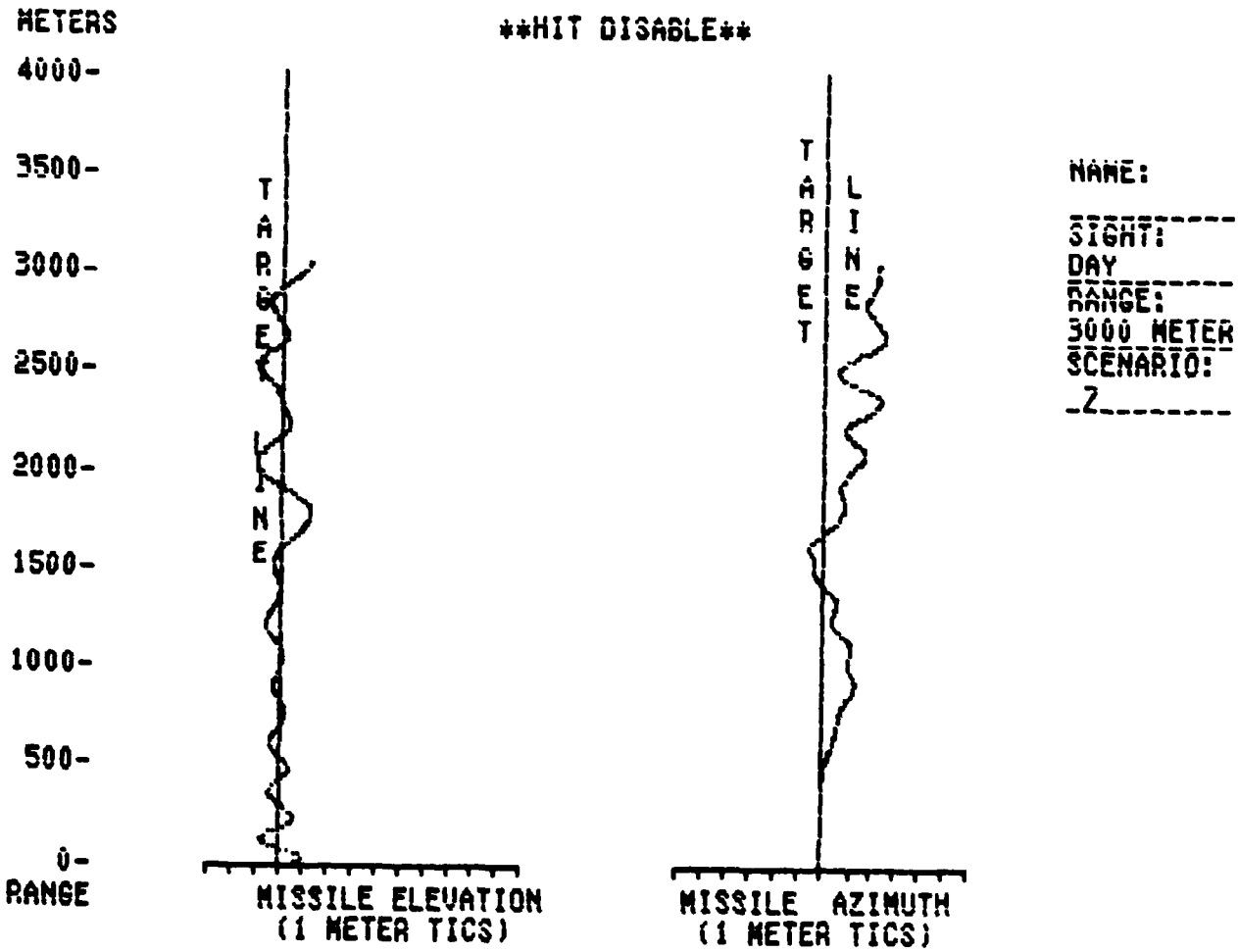


Figure II-5. Missile Position vs. Range (Hit Disable).

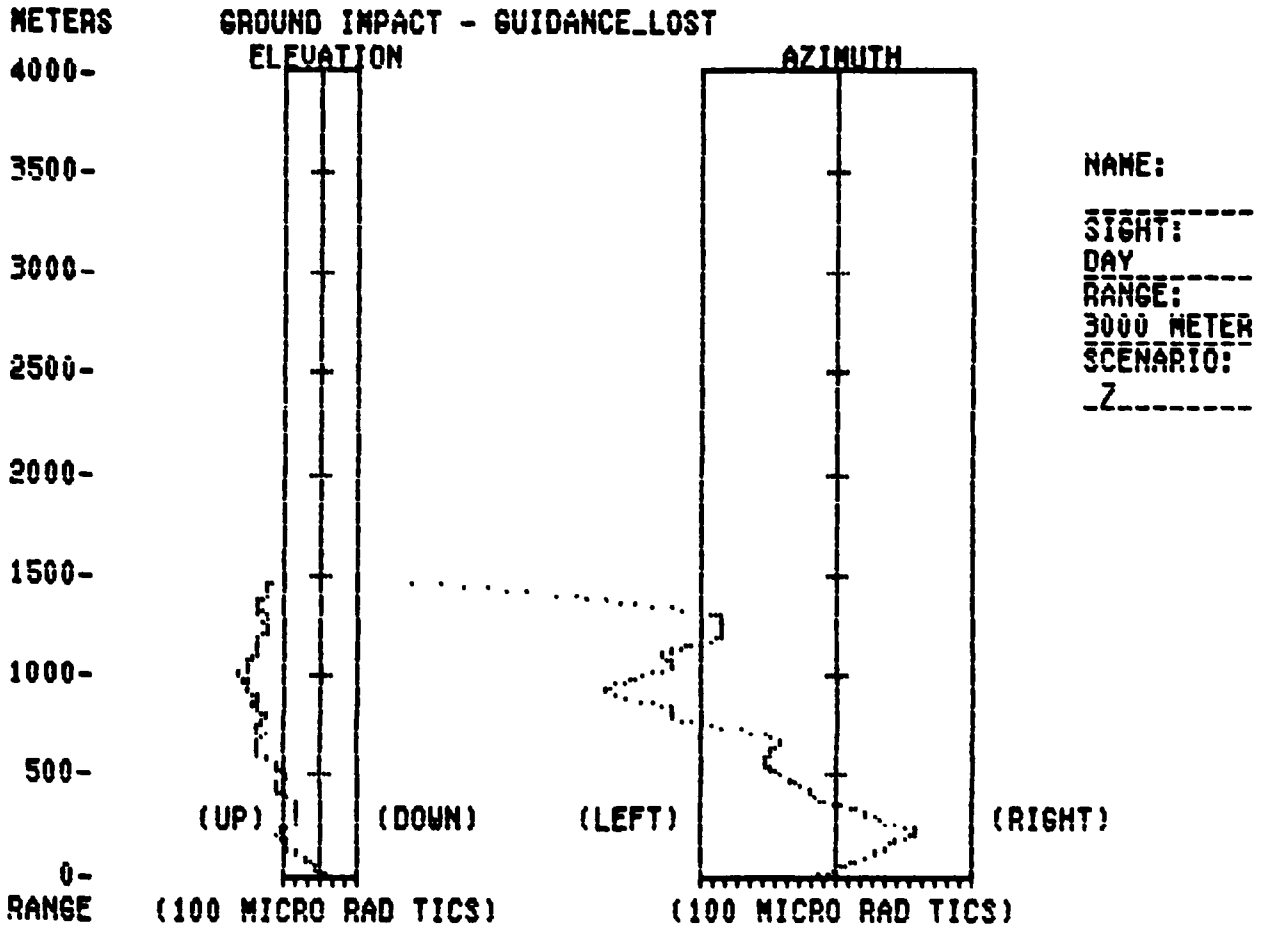


Figure II-6. Gunner Aiming Error vs. Range (Guidance Lost).

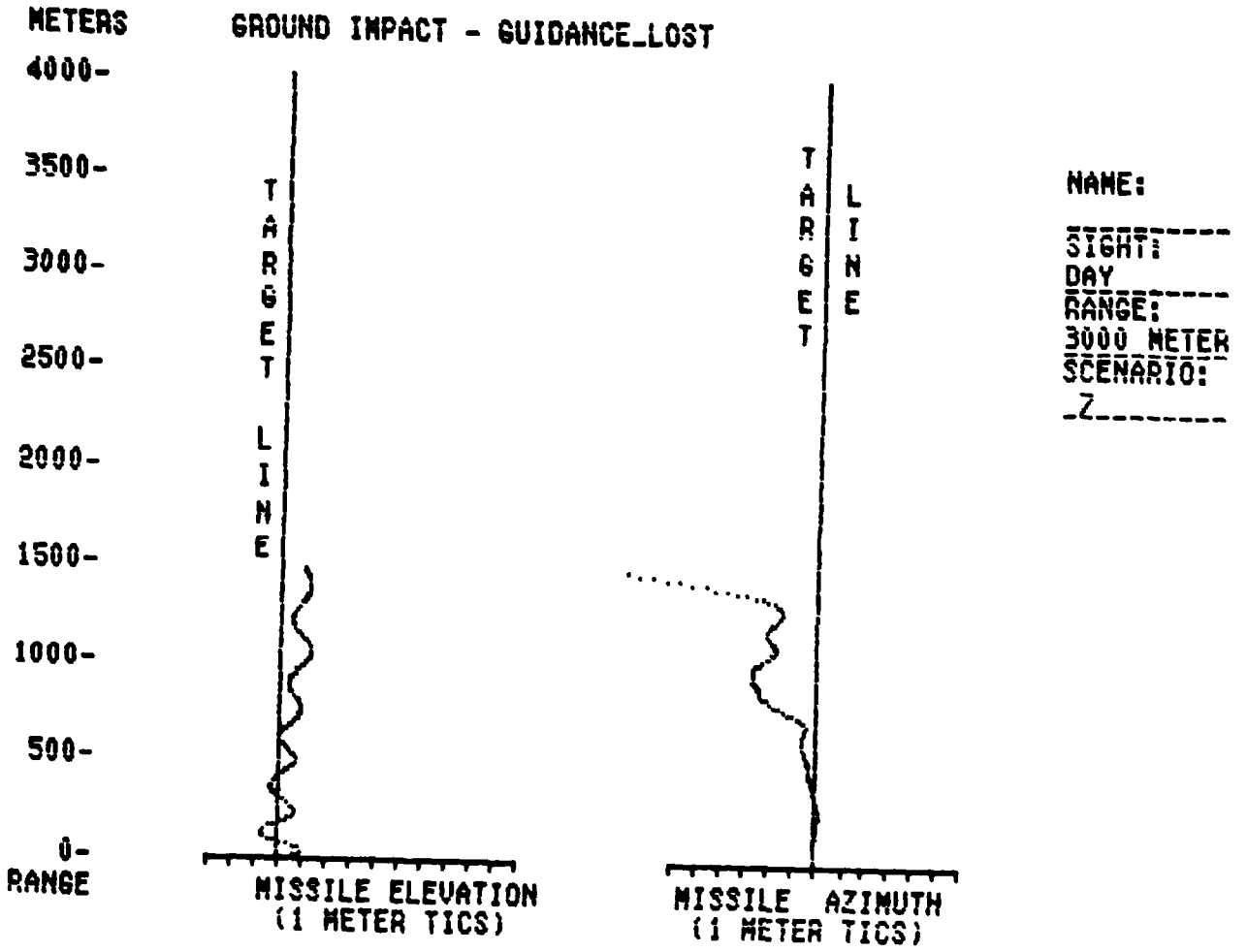


Figure II-7. Missile Position vs. Range (Guidance Lost).

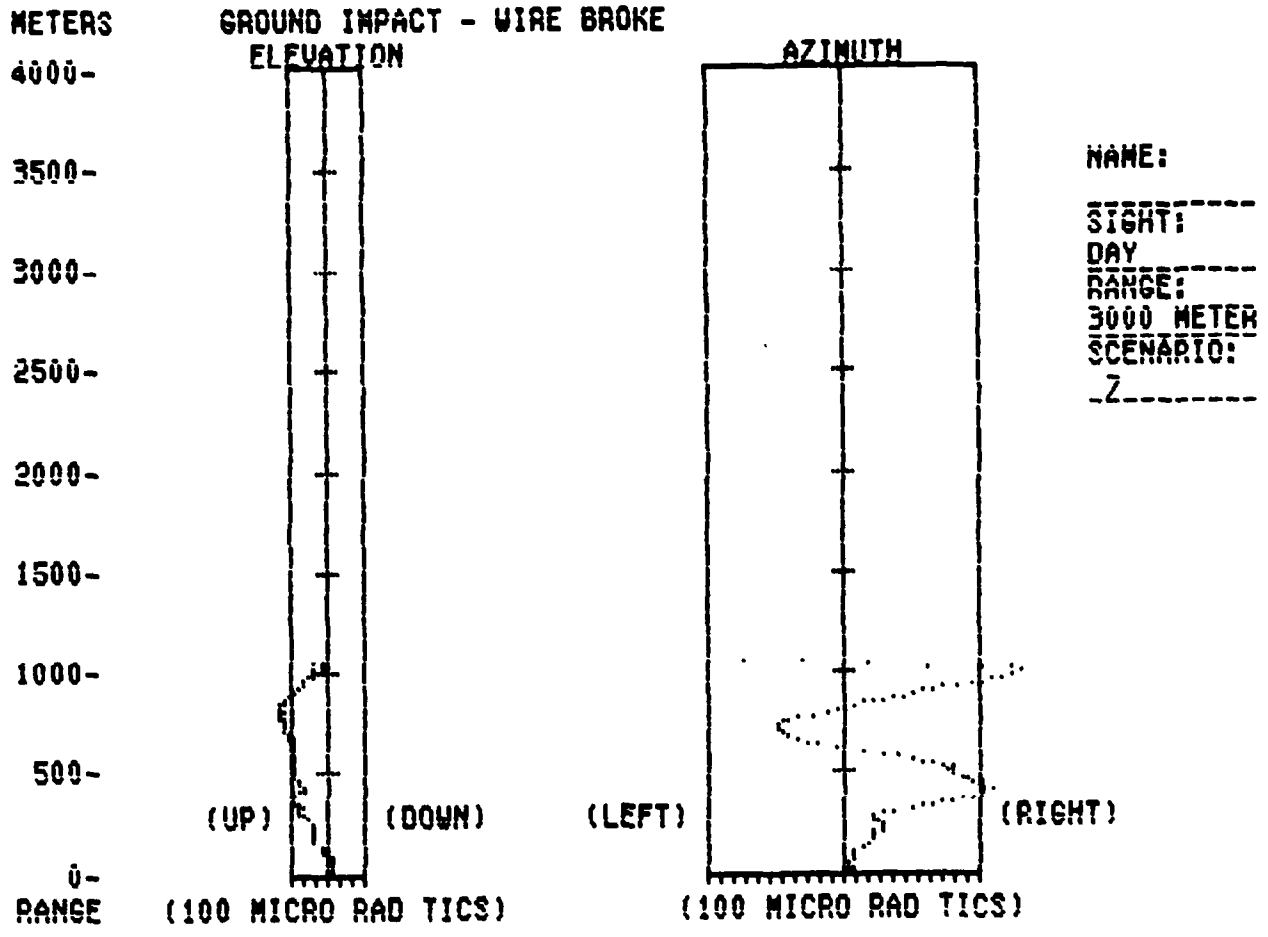


Figure II-8. Gunner Aiming Error vs. Range (Wire Broke).

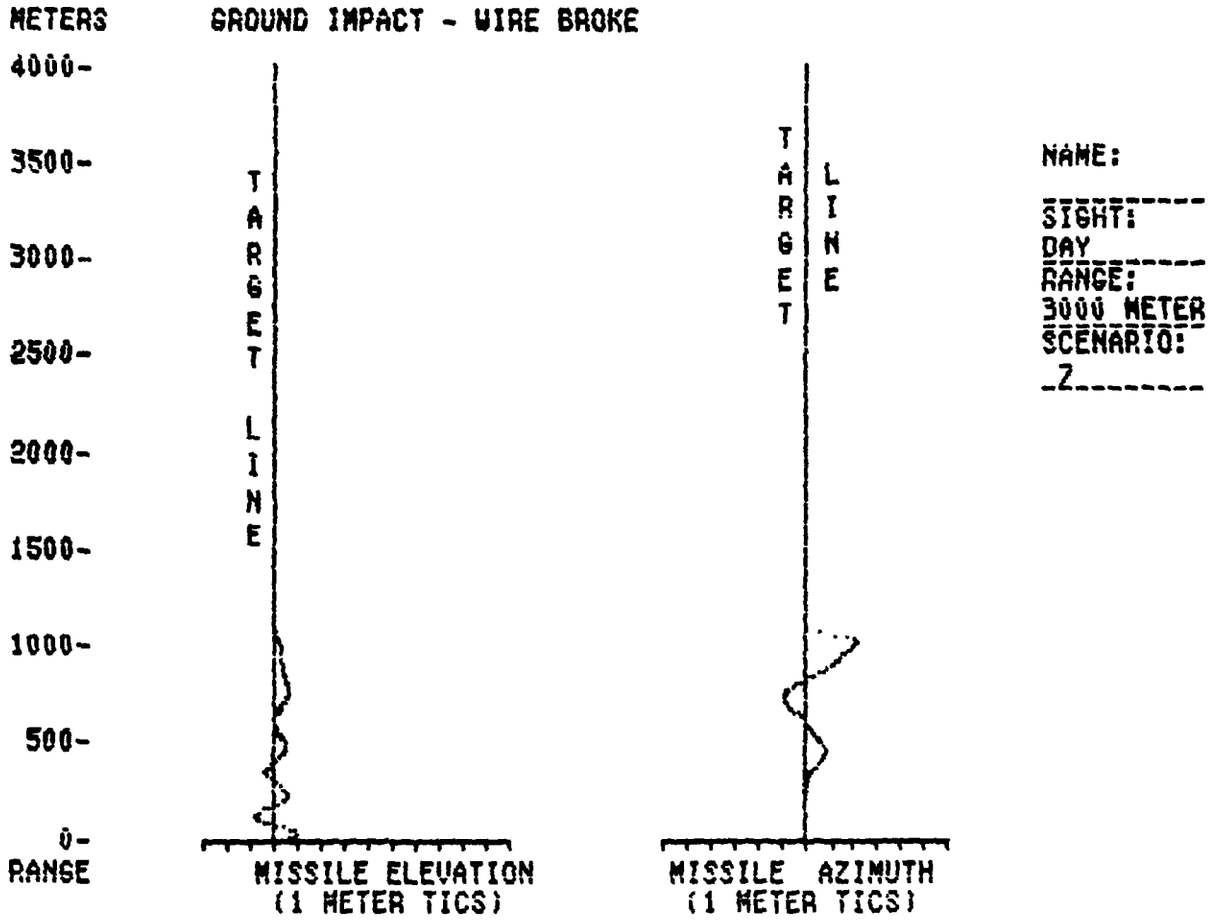


Figure II-9. Missile Position vs. Range (Wire Broke).

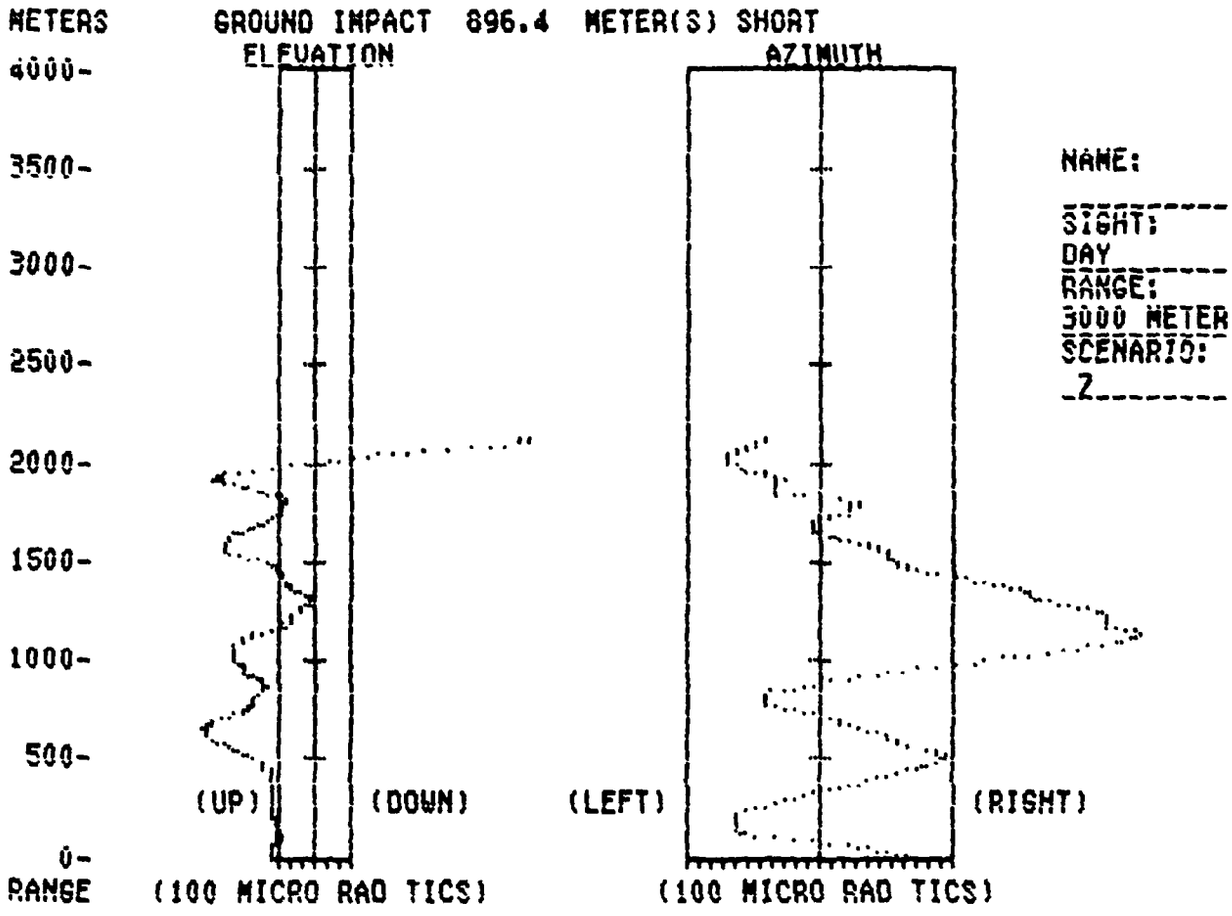


Figure II-10. Gunner Aiming Error vs. Range (Ground Impact).

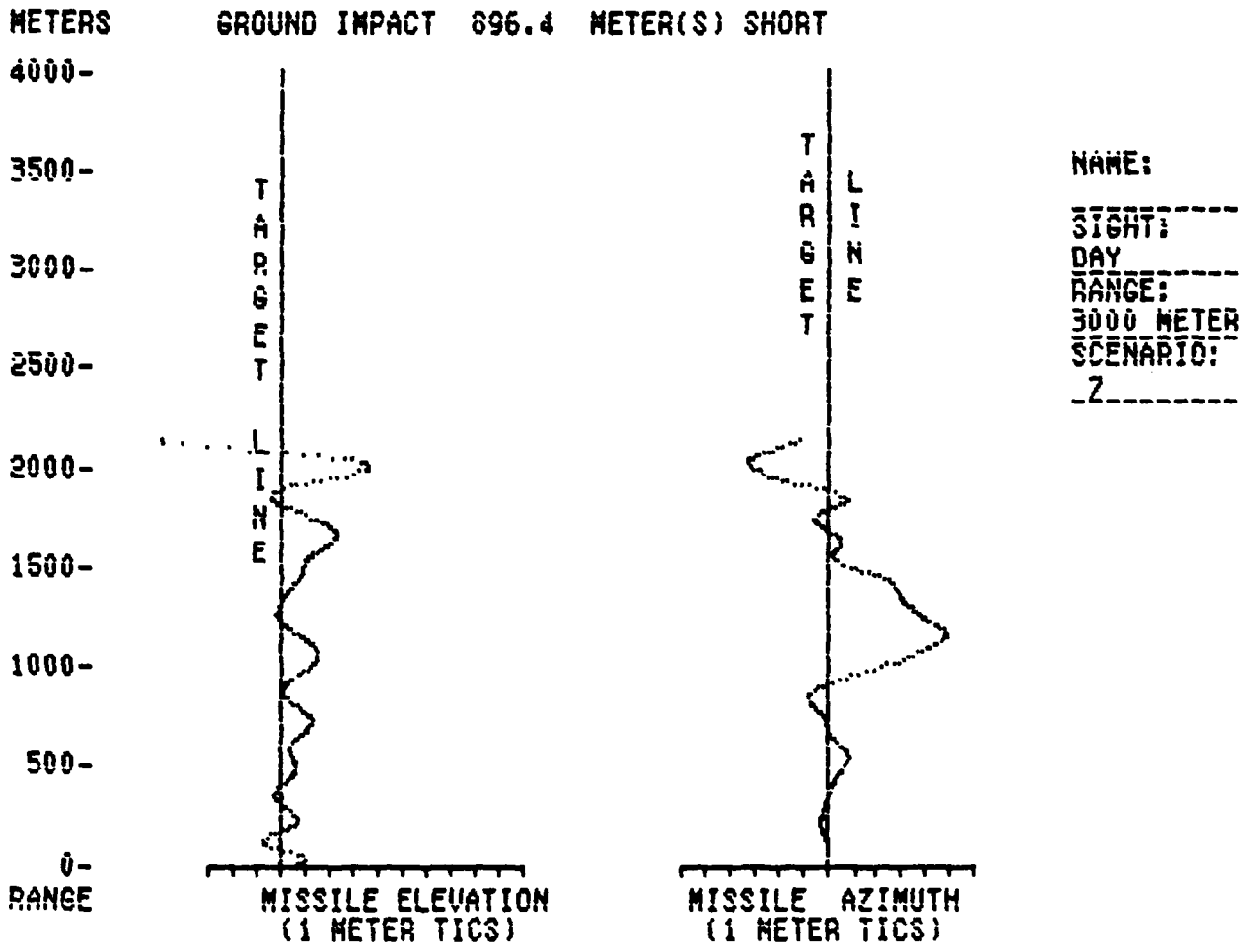


Figure II-11. Missile Position vs. Range (Ground Impact).

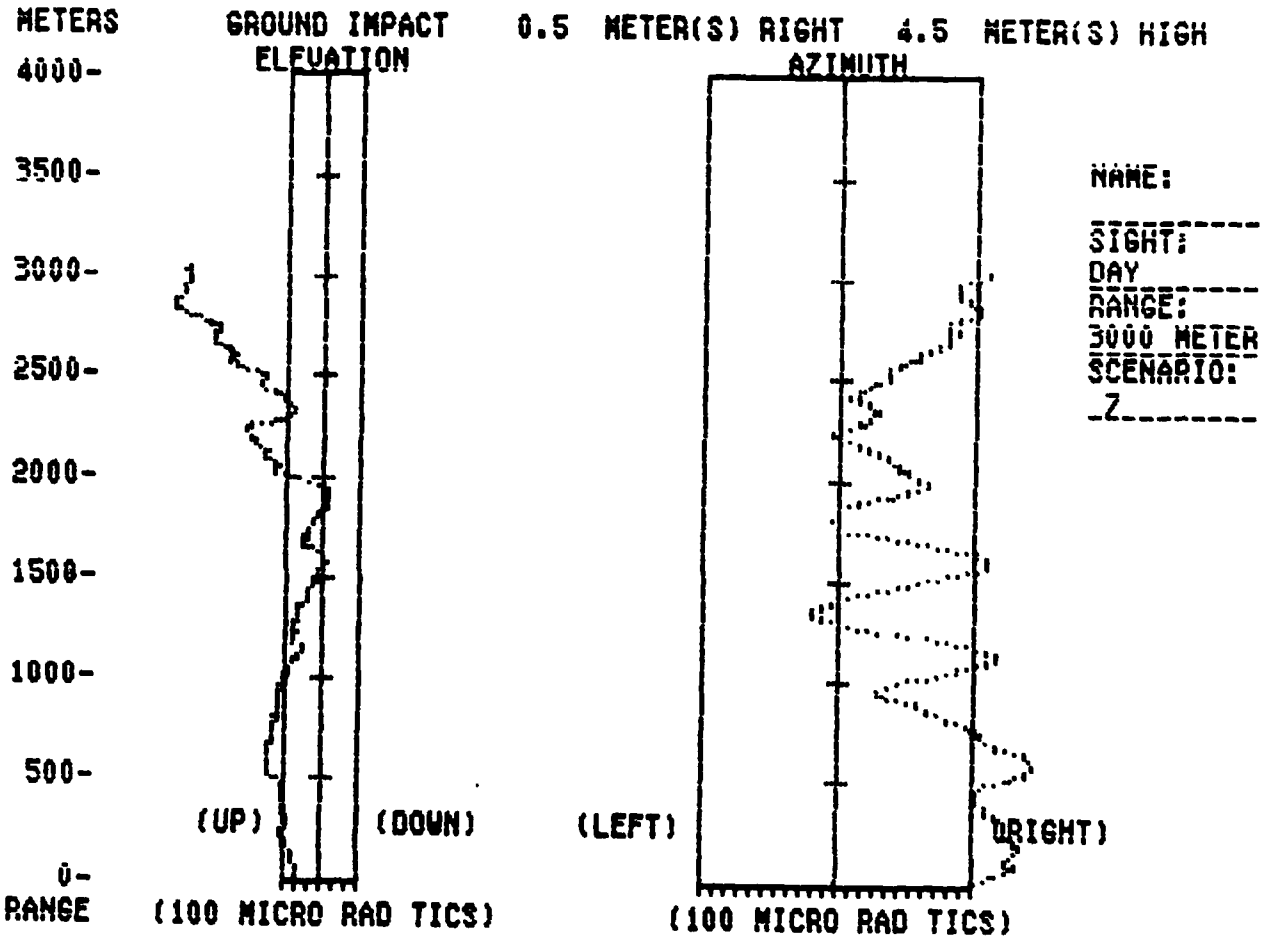


Figure II-12. Gunner Aiming Error vs. Range (Miss).

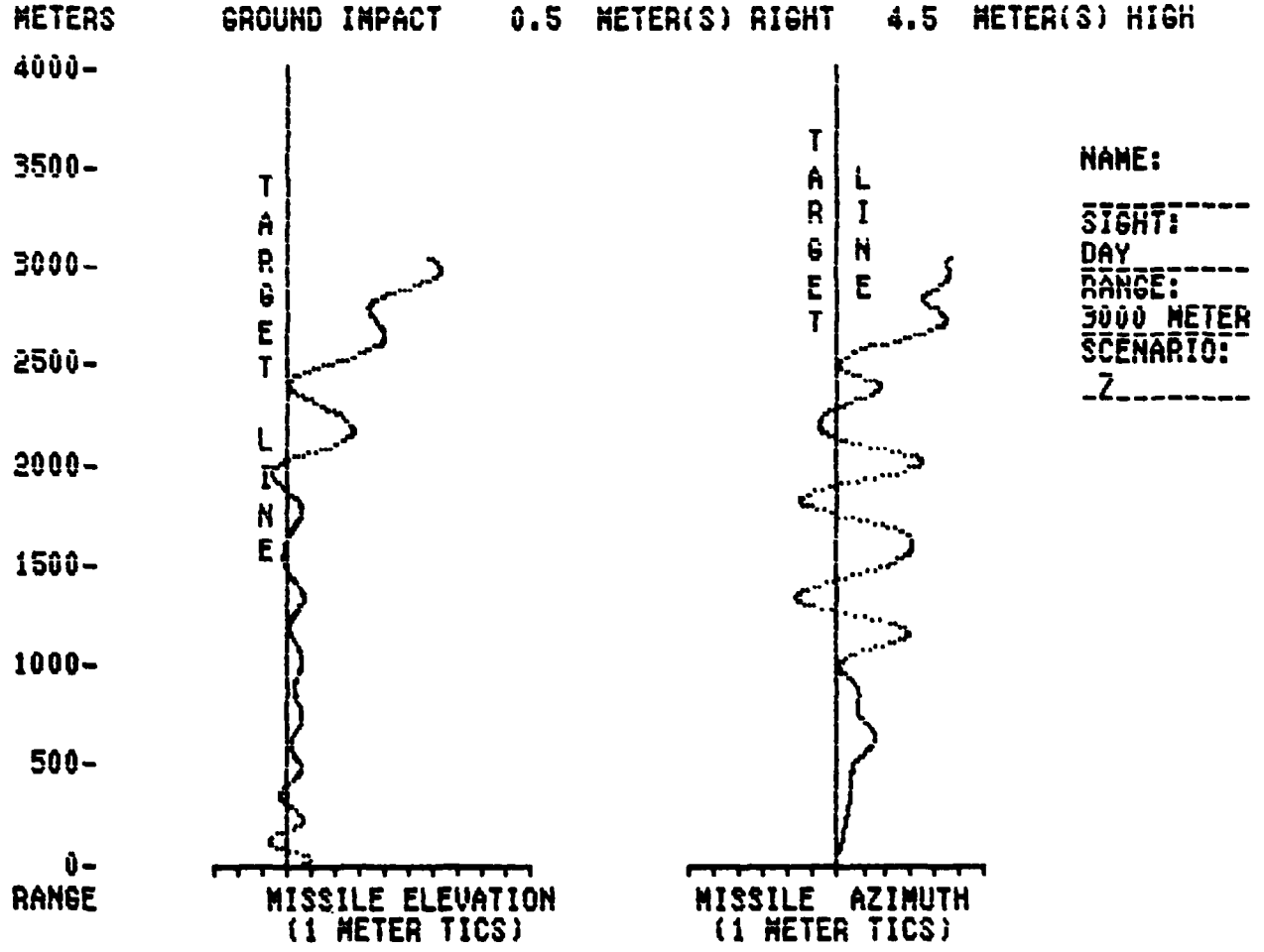


Figure II-13. Missile Position vs. Range (Miss).

SECTION III

SYSTEM DESIGN

A. ELECTRO-OPTICS SUBSYSTEM

The electro-optics subsystem is designed to support the following functions:

(1) Obtain data to measure the gunner aiming error.

(2) Simulate the day sight and insert smoke, TOW missile flare, impact explosions and atmospheric phenomena.

(3) Simulate the night sight and insert launch motor blooming, TOW missile flare and missile impact explosions.

A block diagram of the electro-optic subsystem is shown in Figure III-1.

In order to measure GAE with the required resolution on a target, a 508 mm focal length ruggedized Questar telescope was used. The 100 x 100 photodiode matrix camera has a square detector of 0.6 cm x 0.6 cm. The FOV of the system is 11.8 mr. To simulate a target at 3,000 meters, a 1/285 scaled model is used. The terrain board is located 34 ft. 6 in. from the trainee. At this range the matrix detector views a square, 4.89 in. x 4.89 in. The resolution across the 4.89 in. FOV is 0.0489 inches/pixel. In the real world, this is equivalent to ± 6.96 inches/pixel or ± 3.48 inches per one-half pixel on a real tank at a range of 3,000 meters.

The ruggedized Questar telescope lens is used to collect data for both the matrix camera and gunner's sight picture display.

An infrared source is located slightly above the target. The 100 x 100 photodiode camera is boresighted to the optical sight and is used to measure the GAE. The TV camera provides basic scenic video which, when mixed with computer-generated graphics, yields the sight picture shown on the console TV screen.

When using the "day sight," the trainee looks through a real TOW sight. The sight has been modified to insert generated graphics data from a miniature TV. Smoke, explosions, and the TOW missile are inserted.

When using the "thermal sight" or "night sight," the trainee looks at the terrain board in a darkened room. The terrain board is illuminated with both a dim red base light (to provide background) and a sight-mounted red "spotlight."

The models are selectively painted with retro-reflective paint. Paint is placed on only those areas of the target where the temperature is above ambient, i.e., the road wheels, engine compartment, etc.[1] A small collimated red light

---

[1] Thermal signature patterns are modeled after those shown in Palmer, John E., John D'Agostino and T. Jack Lillie; Infrared Target Recognition Handbook (IRTH); U.S. Army Night Vision and Electro-optics Laboratory, Ft. Belvoir, VA, 1982.

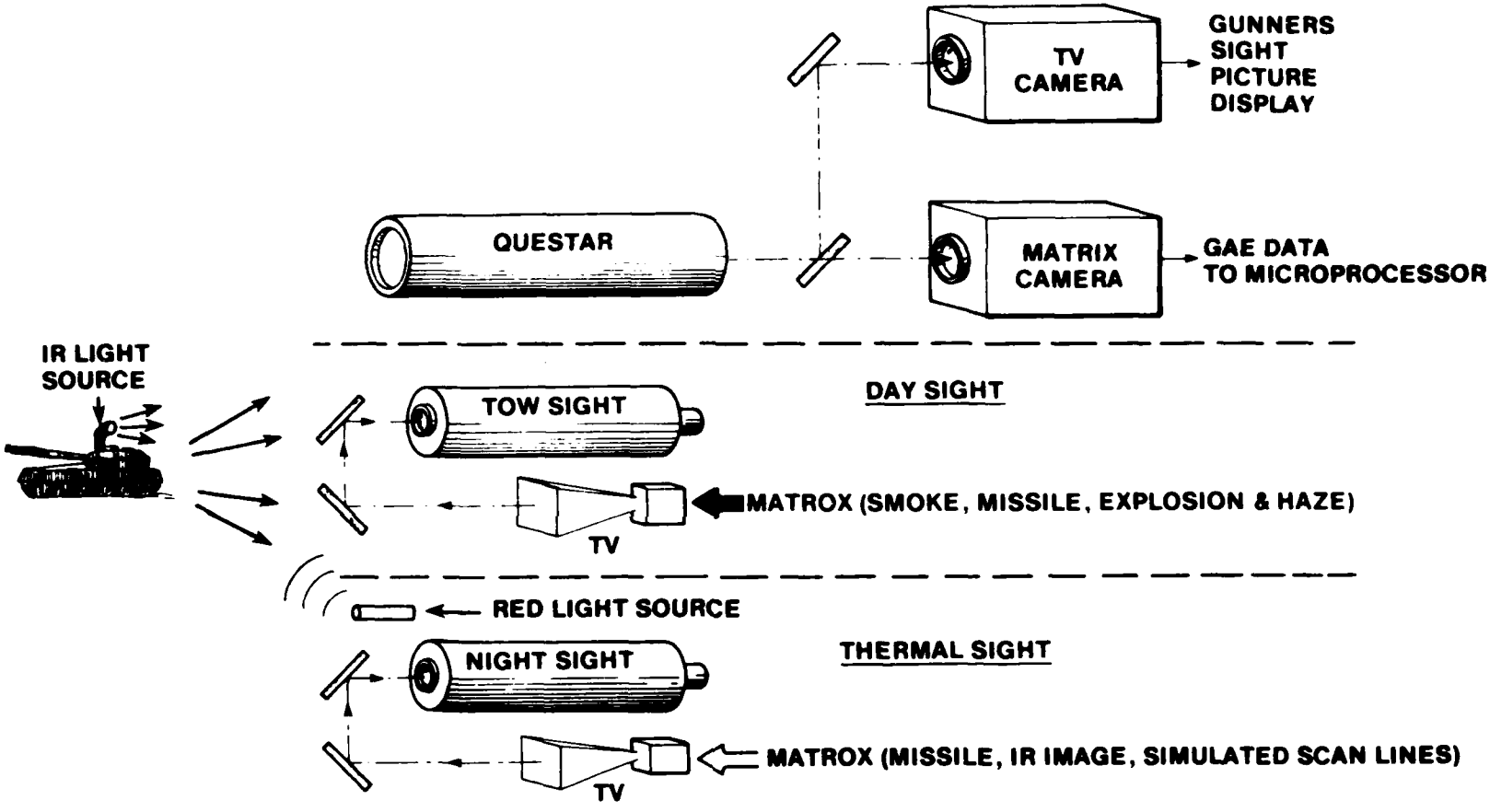


Figure III-1. Electro-optics and Thermal Sight Subsystem.

source (the "spotlight") is located directly above the simulated night sight telescope. The telescope has the same field of view as the operational thermal sight. The beam size from the small light source just overfills the simulated night sight FOV. The intensity of the light is dim enough that it cannot be observed by the trainee unless he is observing reflected light through the simulated night sight. When the gunner's sight is on the target, light from the light source is retro-reflected back from the target into the telescope, making the target appear as if seen through a night sight. Strong reflections occur from areas on the target painted with retro-reflective paint. These areas are the hot areas so the gunner sees a simulated thermal signature. Because the retro-reflected light is directional, only the person looking through the sight simulator can see the retro-reflected light from the target. The TV inserts both the missile flare and the simulated raster scan lines seen in the real night sight.

B. MICROPROCESSOR SUBSYSTEM

The STAGS-TOW trainer relies on both general purpose 8-bit single-chip microcomputers as well as 16-bit high speed single board computers containing state-of-the-art floating point math coprocessors. The system is shown in Figure III-2.

Single chip microcomputers are used to relieve the Missile Flight Simulator (MFS) Processor and the Personnel Interface Processor (PIP) from time-consuming tasks. The Cybernetics CY512 and Intel 8741 microcomputers are used as intelligent stepper motor controllers and a stepper motor coordinator. A National Semiconductor single-chip computer is dedicated to producing computer voice feedback and a Zilog Z-80 controls aspects of the computer graphics.

The MFS is the master processor for the STAGS-T system. This processor commands and interrogates all the sub-processors. The MFS board uses an Intel 86/12A microprocessor with an 8087 high speed math coprocessor. Upon completion of the flight calculations the MFS delivers flight information as well as gunner aiming information to the PIP. Target evasion control and computer voice control timing are provided by the MFS.

The PIP microcomputer is also an Intel SBC 86/12A single board computer. Grey scale computer graphics for visual simulation of launch, missile, smoke, haze and impact explosions originate from this processor. Instructor graphics are generated and instructor keyboard inputs obtained through the PIP.

Model board control is accomplished via a chain of command starting with the instructor's keyboard request for a particular scenario. The selection is routed from the PIP to the Intel 8741 stepper motor coordinator. The stepper motor coordinator then routes the scenario sequence to the appropriate stepper motor controller. Positional information is routed back to the MFS via the stepper motor coordinator when it receives a request.

TOW flight simulation is provided using point mass missile dynamics restricted to the dominant roots of the system characteristic equation. The damping and natural frequency coefficients conform to values presented in the TOW Weapons System Characteristics Document (T-24) (Rev. A.), Hughes Aircraft, 13 Nov 1981. Missile motion is referred to the gunner-target line which is a moving axis unless the target is stationary.

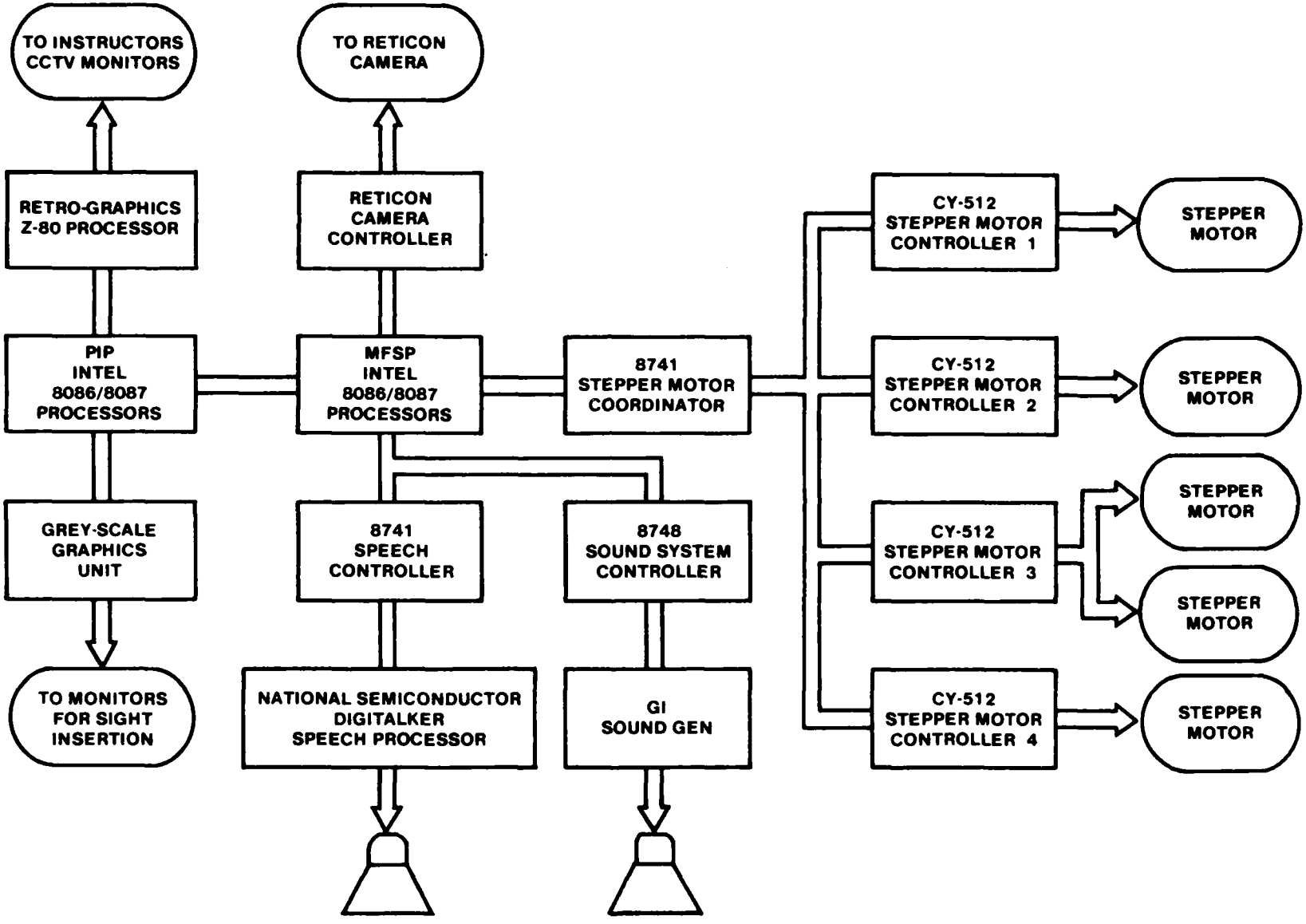


Figure III-2. Microprocessor Subsystem.

The second order system response characteristic of the dominant roots implies a "Type 1" closed loop control system. Such systems provide accurate steady-state response for a stationary target but develop errors for moving targets. The simulation effectively removes such errors by adding an offset to the line-of-sight, the offset being determined by the angular velocity of the line-of-sight.

The flight equations are discussed in detail in Appendix A.

The MFS software is divided into seven modules. The individual modules are described below. For program module listings, see the Appendices.

Main TOW Module

The Main TOW Module is the central module of the Missile Flight Simulator (MFS). Upon reset, it begins the simulation by invoking the procedure PROLOG in the TOW SPEECH MODULE. This causes Digitalker (computer voice subsystem) to issue several commands, including the fire command. The module then waits for a trigger pull. If the gunner has not fired in two seconds, the fire command is repeated. If the gunner does not fire within three additional seconds, the simulation is aborted. Upon a trigger pull, launch sounds are generated and the simulation variables are initialized. The module then enters the main loop, successively invoking YZCNTR, FLIGHT, and DODGE until the FLIGHT procedure sets the FINISHED flag. YZCNTR obtains the gunner aiming error (GAE) in half pixels. The FLIGHT procedure in the TOW FLIGHT MODULE then uses the GAE to simulate the next forty milliseconds of the flight. The DODGE procedure in the MAIN TOW MODULE then checks for the possibility of target evasion. If evasion is enabled, this procedure sends a program to the target motor controller with random steps, rate, and direction. The conditions for evasion are outlined in the DODGE procedure listing.

When the simulation is finished, several buffers are filled with comments that the PIP reads and displays on the instructor's monitor. If the missile flew past the target, the miss distances at the plane of the target are converted to ASCII and placed in the H-MIS-ASCII and V-MIS-ASCII buffers. If the missile fell short, the distance from the target is converted to ASCII and placed in the X-MIS-ASCII buffer. If the missile hit the hill, or left the field of view, the appropriate special comment is placed in the X-MIS-ASCII buffer as described in the module listing. After the comments have been generated, the module invokes the EPILOG procedure in the TOW SPEECH MODULE. The procedure executes verbal debriefing. Finally, the module enters an infinite loop waiting for a reprise request from the instructor.

TOW Flight Module

This module consists of three procedures. INITIATE-VAR initializes variables used by several of the modules. The FLIGHT procedure performs the actual flight simulation. It first converts the GAE obtained from YZCNTR to radians, then it invokes ACTIVE-TRACK in the TOW TARGET MODULE to determine the number of the target track. The TARGET-DATA procedure in the TOW TARGET MODULE is then called. This procedure updates HTARG and several variables used by other procedures in the target module. FLIGHT then checks for sensed excessive gunner control movement within 3 seconds of launch -- which symptomizes loss of control. If the "excessive movement" condition exists,

the FINISHED flag is set and a ground explosion sound is generated. FLIGHT then determines if the missile is within eight feet of the target. If it is, CURRENT-TRACK, which encodes the target track number, is used to invoke the appropriate TRACK procedure in the target module. These procedures ascertain whether the missile hit or missed the target as described below in the TOW TARGET MODULE. Once the missile is within eight feet of the target, the FINISHED flag is set, and a ground explosion sound is generated if the missile flew past the target. The FLIGHT procedure then updates the values of the missile parameters. These values are then used to calculate the new values for OFF-H and Z. FLIGHT then invokes the GROUNDED procedure in the TOW TARGET MODULE.

The GROUNDED procedure checks for missile ground impact. FLIGHT then examines the GAE to determine whether the missile left the field of view. FINISHED is set in this case. Finally, the missile data sample used by the H-REPRISE procedure is updated. H-REPRISE sends Z and OFF-H samples from the flight to the PIP which displays this data on the instructor's monitor.

This module incorporates code which allows the target to be located on any track in any position at the time of missile impact. Further, it permits target switching during the simulation. Because only one motor count can currently be maintained by the SMC (Stepper Motor Coordinator), these features are not presently required and have been disabled as outlined below.

The target module consists of six major procedures. Because target switching is not currently used, ACTIVE TRACK sets the FLIGHT procedure variable CURRENT TRACK to STARTING TRACK, since TARGET SWITCH is initialized to zero by INITIATE-VAR. STARTING TRACK is filled by the PIP when the instructor selects the scenario. TARGET DATA updates HTARG, and calculates the value of ALPHA, the rotation of the target on track 3 in radians. Because the count for motor 5 is not presently available, INITIATE-VAR sets ACNT, the counter for motor 5, to zero, unless scenario 9 is selected, in which case it is set to 90 degrees. Because the count for motor 4 is not currently available, INITIATE-VAR sets DCNT, the linear actuator count, to 550. Consequently, DEPRESSION DEPTH is always zero and the GAEZ and the vertical tank dimensions are not modified. If the target is on track 2, the slope of the hill and its height at the target position are obtained for use by the TRACK 2 procedure when the missile is within eight feet of the target. Finally, TARGET Z is calculated for use by the GROUNDED procedure. The UPDATE COUNTS procedure is used to acquire the target count. It is capable of obtaining any one of the five motor counts. However, this code is not currently used, since the SMC maintains only one count.

The GROUNDED procedure simply determines whether the missile hit the ground. The TRACK 1, TRACK 2, and TRACK 3 procedures determine whether the missile hits or misses the target on the respective track.

TOW Speech Module

This module consists of two major procedures, PROLOG and EPILOG. PROLOG waits until the scenario is selected, then it uses STARTING TRACK to turn on the target IR light. If the instructor selects the day sight, it causes Digitalker to say "Use Day Sight," otherwise it says "Use Night Sight." PROLOG then waits until the motors start before issuing "Alert! Tank." The direction is obtained from the EAST WEST flag set by the PIP. After saying either "East" or "West," it causes Digitalker to say "3,000 Meters," pause one second, and "At My Command." PROLOG then waits for a fire signal from the target motor controller. Upon receipt of the fire signal it causes Digitalker to say "Fire."

EPILOG causes Digitalker to say "Cease Tracking" followed by a one second delay. If the missile hit the target, it says "Hit." Otherwise, it uses the ASCII miss distances in H-MIS-ASCII and V-MIS-ASCII to derive the code required by Digitalker to say the miss distances. If the missile fell short, the X-MIS-ASCII buffer is used for the same purpose, unless a special miss comment is in the buffer, in which case Digitalker says nothing.

Tow Utility

This module contains four procedures, MISS COMMENT is used by the MAIN TOW MODULE to convert miss distances from real format to ASCII format. MISS COMMENT in turn uses the HX2AS procedure which converts an integer to ASCII format. The SOUND procedure used by several modules uses the WHT KIND parameter to place signals to the sound system on the parallel connector. The PPI SET procedure simply sets up the 8255 on the MFS 86/12 board.

TOW XF and TOW IR

TOW XF transfers line-by-line data provided by the photo-detector line array processor into a complete picture array. TOW IR analyzes the IR-spot data array provided by TOW XF to compute the horizontal and vertical gunner aiming error including auto-boresight. These are the only assembly language modules and were originally written for DRAGON.

C. COMPUTER GRAPHICS AND VIDEO SUBSYSTEMS

Computer generated real-time graphics are controlled by the Personnel Interface Processor. A computer graphics board, EIA sync generator and phase-locked-loop synchronization circuit coordinate the various raster scan CCTV monitors. Figure III-3 shows the complete graphics and video subsystem.

A mini-monitor is used to insert real-time video graphics for the gunner optical sight while a second mini-monitor is used in conjunction with the thermal sight to produce thermal video effects. These graphic effects include launch obscuration, haze, rocket motor burn, missile flare and impact explosion.

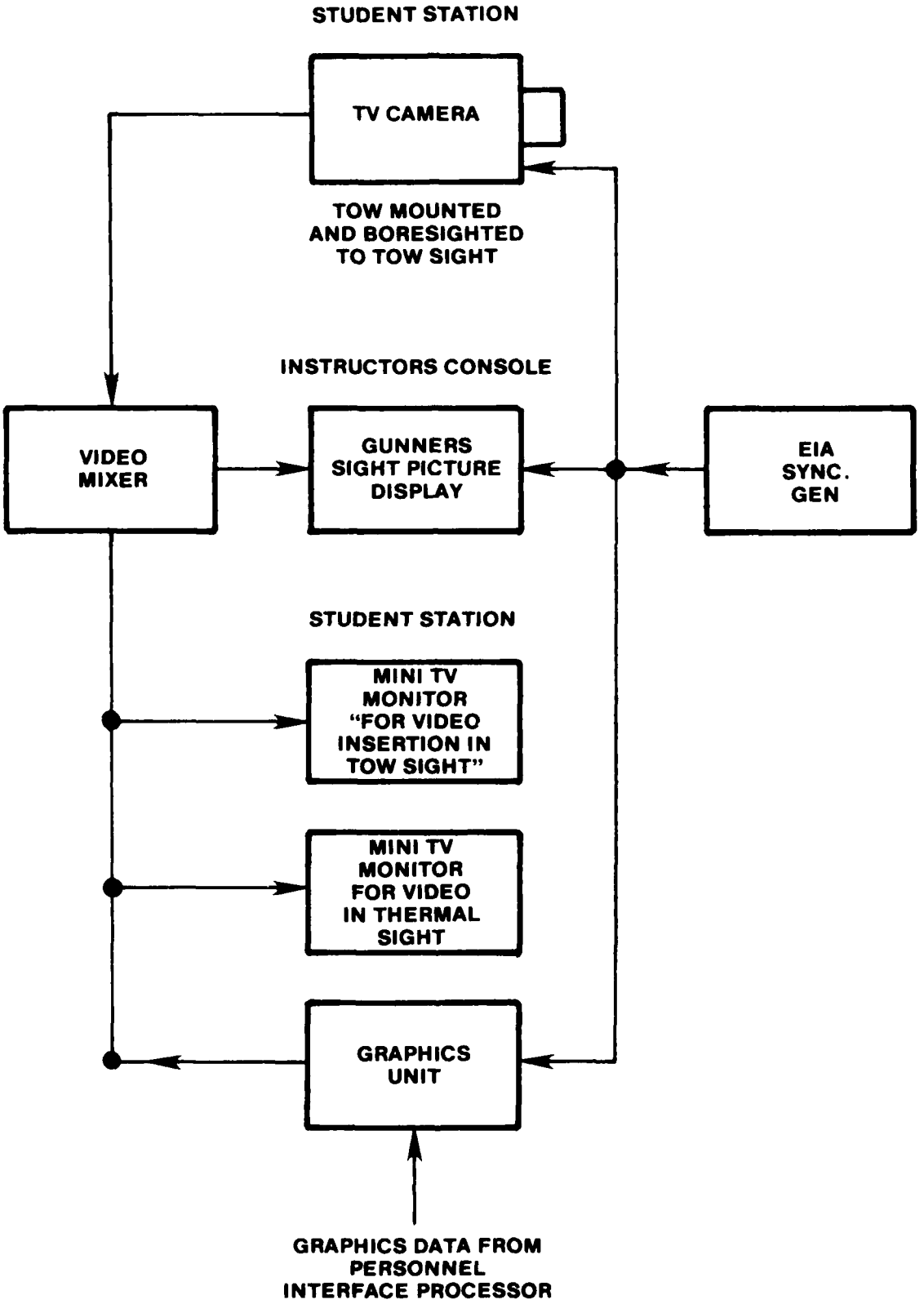


Figure III-3. Computer Graphics and Video Subsystem.

Real-time graphics are also generated for the instructor console. Vertical and horizontal gunner aiming error vs. distance, in 100 micro-radian units, is displayed. For follow-up analysis, graphics are presented showing missile position relative to an imaginary target line between the gunner and his target.

Real-time video graphics are generated by the Personnel Interface Processor (PIP). The PIP receives gunner aiming error information from the missile flight simulator processor (MFS) as well as missile angle from the line of sight of the gunner to the missile position. The gunner aiming error is used to position the final explosion (hit or miss) in the TOW sights.

Smoke and haze are simulated in the TOW optical sight and blooming in the thermal sight by modulating the background level, i.e., overall grey scale setting of the entire graphic video insertion in the gunner sight.

The final explosion of the missile and/or tank is simulated at the end of the TOW flight and inserted, via the RGB-256 graphics board, into the gunner sight. The explosion is a series of geometric star shapes indicating either a hit or miss. The PIP uses the missile-to-aim-point information to position the explosion wherever the missile was as it impacted the target or ground. A ground explosion is similar to a target explosion, however, it differs by only exploding in an upward sense. Thus, the TOW gunner has visual feedback through his sights indicating hit and miss. The computer generated graphics are passed directly to the gunner's sights through Hitachi VM154A, one-and-a-quarter-inch, closed circuit television (CCTV) monitors.

The TOW computer graphic visual presentation is prepared by the PIP. In addition to this processor a computer graphics board, a phase-locked-loop sync board, and an EIA composite sync generator are used.

Computer generated graphics provide three major functions:

a. Real-time video graphics are generated for the gunner day sight. These graphics include a simulated missile, marker flare, smoke, and a final explosion.

b. Real-time video graphics are generated for the gunner night sight. These graphics include a simulated missile, marker flare, image flare due to hot gasses, and final explosion.

c. Real-time graphics generated for the instructor indicate both vertical and horizontal gunner aiming error. Also, for follow-up analysis, graphics may be presented for gunner aiming error vs. distance and missile position vs. distance.

Gunner's sight real-time computer graphics are generated on a Matrox model RGB-256 (256 x 256 x 4) graphics board. The Matrox graphics board produces 256 pixels horizontally by 256 pixels vertically by sixteen levels of grey scale. The sixteen levels of grey scale provide for a range of visual intensity which allows for smoke generation which varies from fully transparent to completely opaque in sixteen discreet levels. The Matrox RGB-256 is a graphics imaging system in which a complete grey scale capability has been integrated onto a single printed circuit board. The card includes

built-in NTSC (American) and PAL (European) grey scale encoders which can provide up to sixteen shades of grey. The encoders permit the RGB-256 to directly drive standard low cost black and white TV monitors on a single 75 ohm cable. It features the industry-standard Intel Multibus which makes it directly bus-compatible with all Intel single-board computers.

Real-time video graphics are generated by the PIP. The PIP receives gunner aiming error information from the MFS as well as missile angle from the line of sight of the gunner to the missile position. The gunner aiming error is used to position the final explosion (hit or miss) in the TOW sight.

Gunner Graphics

TOW sight graphic missile simulation is accomplished by deriving the missile position. Second, the size of the missile flare is determined by the elapsed time since the missile launch. Third, the flare brilliance is determined by the elapsed time since launch.

The flare size shrinks from 10 pixels down to 1 pixel from launch to maximum range. The brilliance decays from a level of ten (with fifteen being most brilliant) to a level of zero at maximum range.

An octagon was selected as the simulated missile shape as this can be quickly calculated for real-time graphics. This shape appears mostly as a circular area to the TOW gunner.

Smoke and haze are simulated in the TOW sight by modulating the overall grey scale setting of the entire graphic video insertion in the gunner sight. It is possible to tell the graphics board to "erase" to any given grey scale level between zero and fifteen, with zero being black (transparent in the gunner's sight) and fifteen being white (opaque in the gunner's sight). The levels of background are modulated with time to effect a smoke and/or haze simulation. A typical smoke simulation might consist of starting from level zero rising to level fifteen, dropping to level eight, back up to fifteen, down to four, up to eight and down to zero during a period of one to two seconds.

The final explosion of the missile and/or tank is simulated at the end of the TOW flight and inserted, via the graphics board, into the shapes indicating either a hit or miss. The PIP uses the missile-to-aim-point data to superimpose the explosion at the last calculated missile position. A ground explosion differs from a target explosion by exploding only in an upward sense and for only half the duration of a target explosion. This difference provides the TOW gunner with visual feedback through his sight indicating hit or miss. The computer generated graphics are passed directly to the gunner's sight through an Hitachi VM154A one-and-a-quarter-inch closed circuit television (CCTV) monitor. The optical arrangement is shown in Figure III-1. The television screen appears at infinity along with the viewed scene through the day or thermal sight. One CCTV camera is mounted inside the TOW day sight and another in the thermal sight. Video support electronics are located as near as possible to the cathode ray tube for maximum high frequency response.

Instructor's Console Graphics

The instructor console graphics subsystem is composed of two units, (1) a video replica of the gunner sight picture and (2) a graphic plot of gunner aiming error vs. distance or missile position vs. distance.

The representation of the gunner's sight is accomplished by mixing the output of the gunner sight TV camera with the video graphics presented to the gunner's sight. This composite picture thus presents to the instructor an image of the gunner's view. In addition, the flight time of the simulated missile and the current date are notated at the bottom of the screen.

Scenic coverage of the model board is by a closed circuit television (CCTV) camera mounted outboard on the day sight housing. This camera is boresighted to the gunner's day and thermal sights. The camera used is an RCA TC-2021/N with a NUVICON camera tube. The CCTV video is combined with that of the Matrox RGB-256 graphic generator. The combination of CCTV video and computer graphics is then a representative visual image of the gunner sight picture except for the crosshairs. Crosshairs are added, to complete the instructor sight picture display, by passing the video presentation through a Colorado Video Model 260 electronic crosshair generator.

The graphical plots of the gunner aiming error (GAE) vs. distance, in 100 micro-radian units, for both azimuth and elevation error are presented in real-time during missile flight. The graphs show the actual gunner aiming errors as well as any loss of guidance during the flight. The trainee name, scenario number, range of target, and type of sight used for the scenario are notated on the margin for record keeping.

For a miss, the displayed results show the deviation from target in meters and tenths of meters where the missile passed the target plane. If the missile struck the ground before passing the target, a message is displayed stating "GROUND IMPACT" as well as the distance remaining to the target. If the target is hit, then either "HIT KILL" or "HIT DISABLE" is displayed. A "HIT KILL" is proclaimed if the missile has been calculated to have impacted within the center third of the target, likewise, "HIT DISABLE" is proclaimed if the missile impacted the target outside the center third of the target. (See Figures II-2 through II-5.)

After the missile impact a reprise of the flight may be called. The missile position vs. range in both azimuth and elevation is replayed on a real-time basis. This display as well as the gunner-aiming-error display may be printed out on a hard copy printer in less than twenty seconds time.

D. COMPUTER VOICE SYSTEM

Human-like speech firing commands are given the trainee by a speech processor. The computer-generated voice can also coach the student when selected, i.e., "high," "low," "right" and "left." The computer-generated voice unit also tells the trainee where the missile impacted and when the mission is completed. The computer-generated voice serves to coordinate the training session by issuing the firing commands at a specific point in the scenario, thus relieving the instructor of this task.

Firing commands consist of the following:

[Squad]
[Tanks]
[North]
[South]
[East]
[West]
[3,000]
[Meters]
[At my command...Fire]
[Cease tracking]

Debrief commands consist of:

[Hit]
[Miss]
[High]
[Low]
[Right]
[Left]

followed by distance in meters, i.e., [Miss - High - Ten Meters].

The computer voice is generated using a board containing a National Semiconductor "Digitalker" Speech Chip and Speech ROMS. The board also contains an Intel 8741 Programmable Peripheral Interface chip, random logic for address decoding, and analog components to filter the output of the "Digitalker." This system produces a natural sounding voice under the control of the Missile Flight Simulator (MFS) board.

Refer to the block diagram in Figure III-4 for the discussion in this section. The circuit schematic is shown in Figure III-5 through III-8.

The 8741 Programmable Peripheral Interface connects the National Semiconductor "Digitalker" to the Intel "Multibus" which this system uses for inter-board communication. The 8741 performs two functions. First, on power-up or system reset, it performs a routine which set up the speech chip in the proper configuration. Secondly, the 8741 translates between "Multibus" and "Digitalker" bus signals.

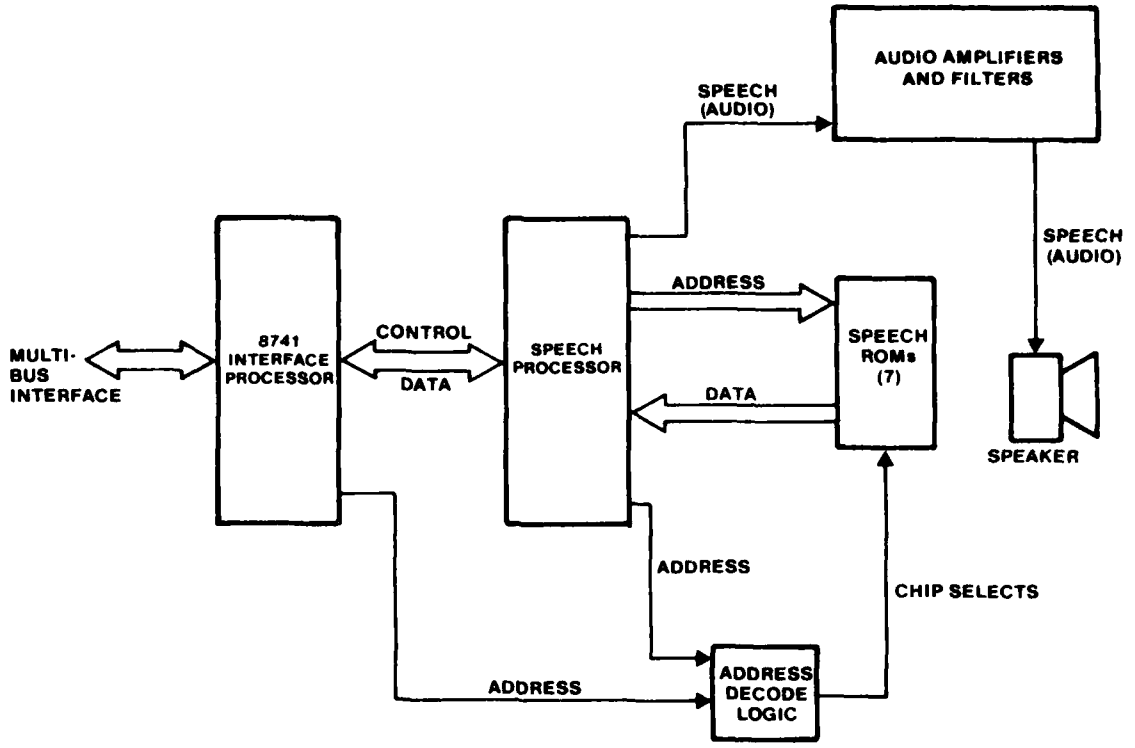


Figure III-4. Computer Synthesized Voice Subsystem.

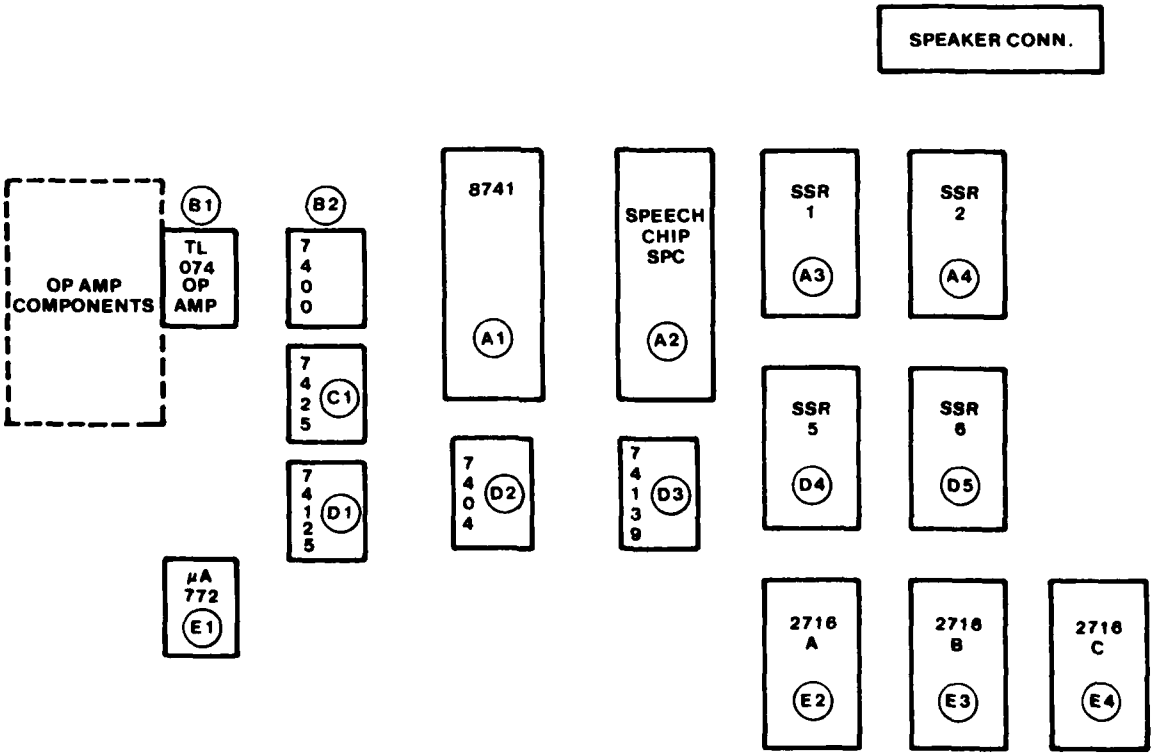


Figure III-5. Integrated Circuit Complement for Computer Synthesized Voice Subsystem.

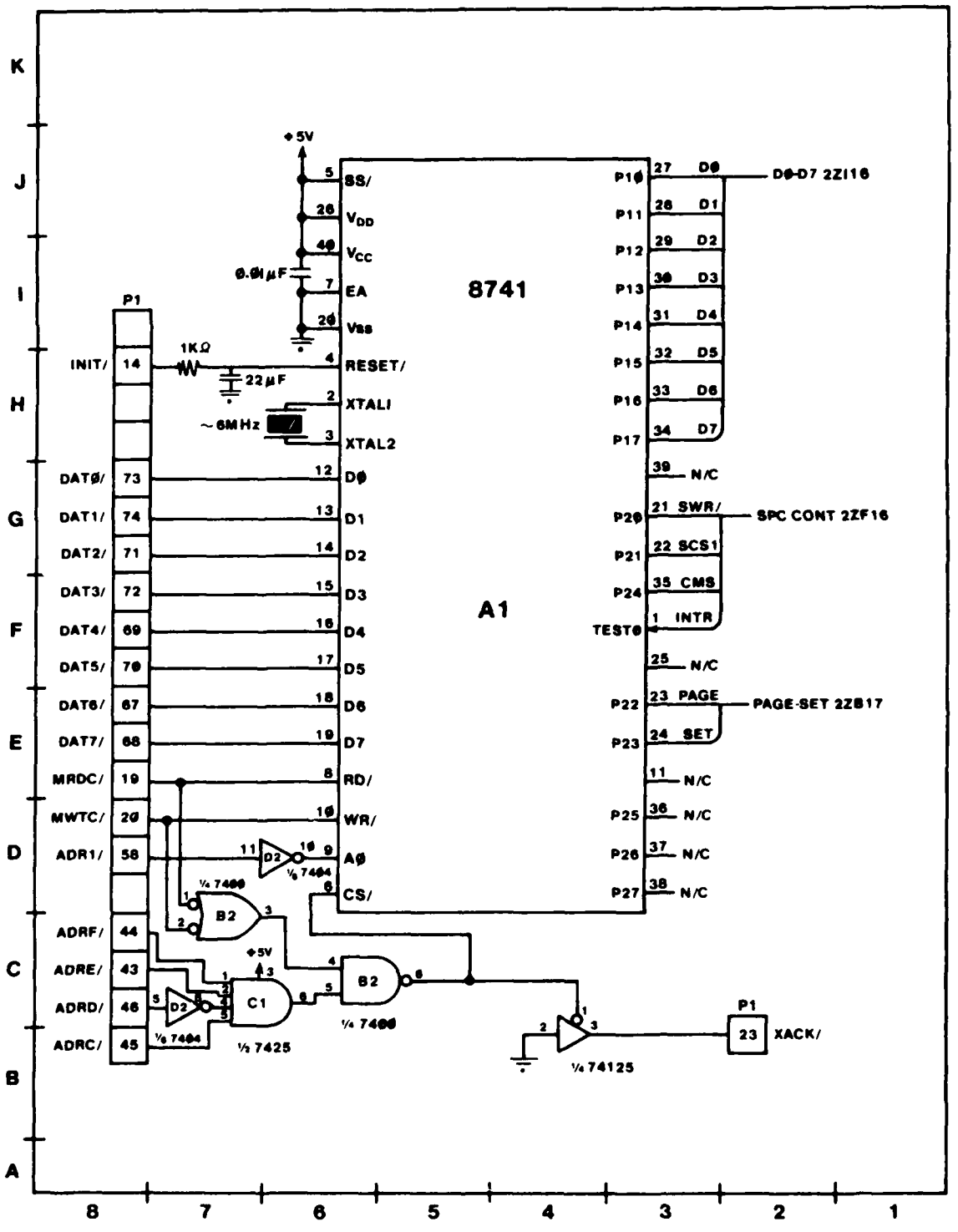


Figure III-6. Computer Synthesized Voice Subsystem Schematic (1 of 3).

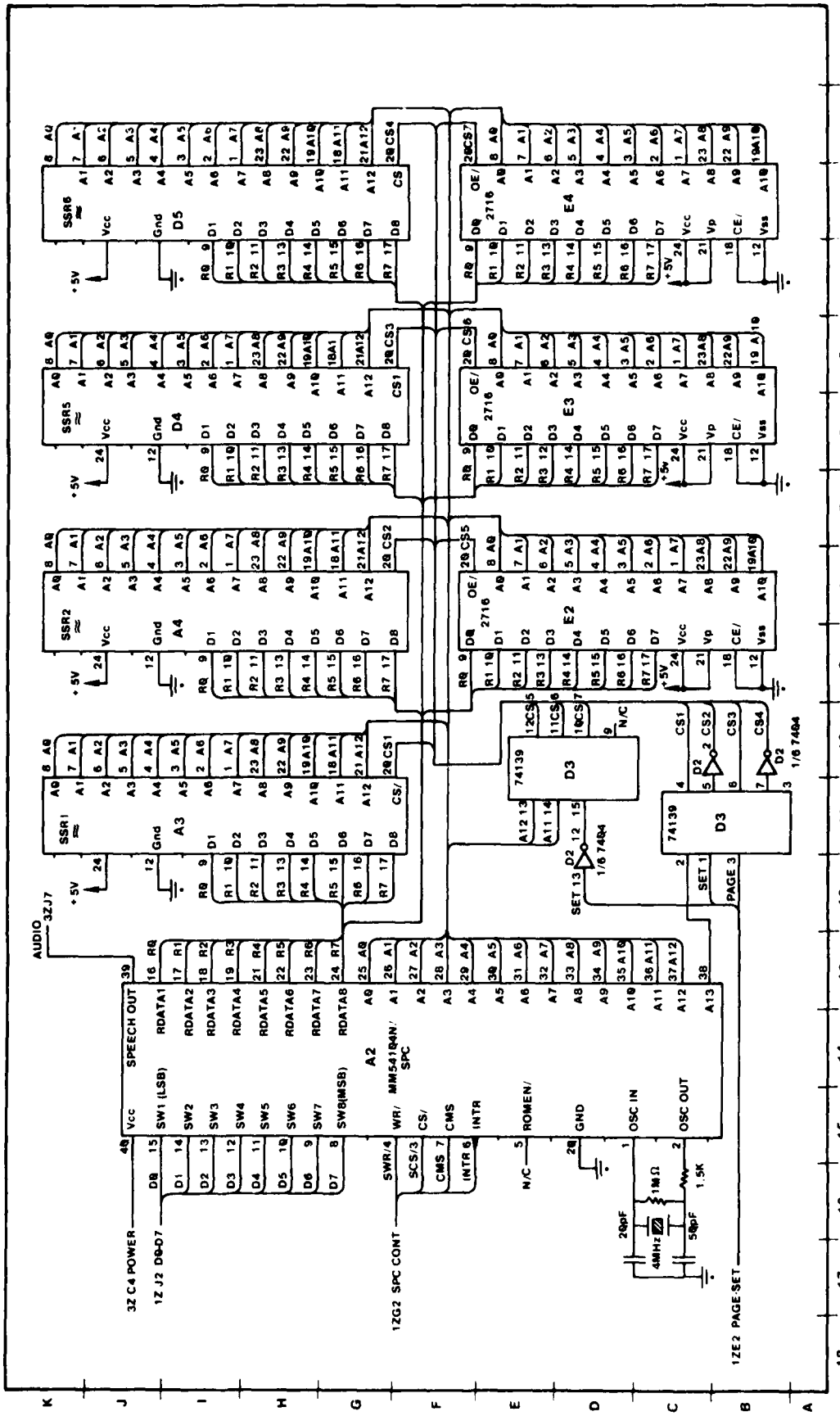


Figure III-7. Computer Synthesized Voice Subsystem Schematic (2 of 3).

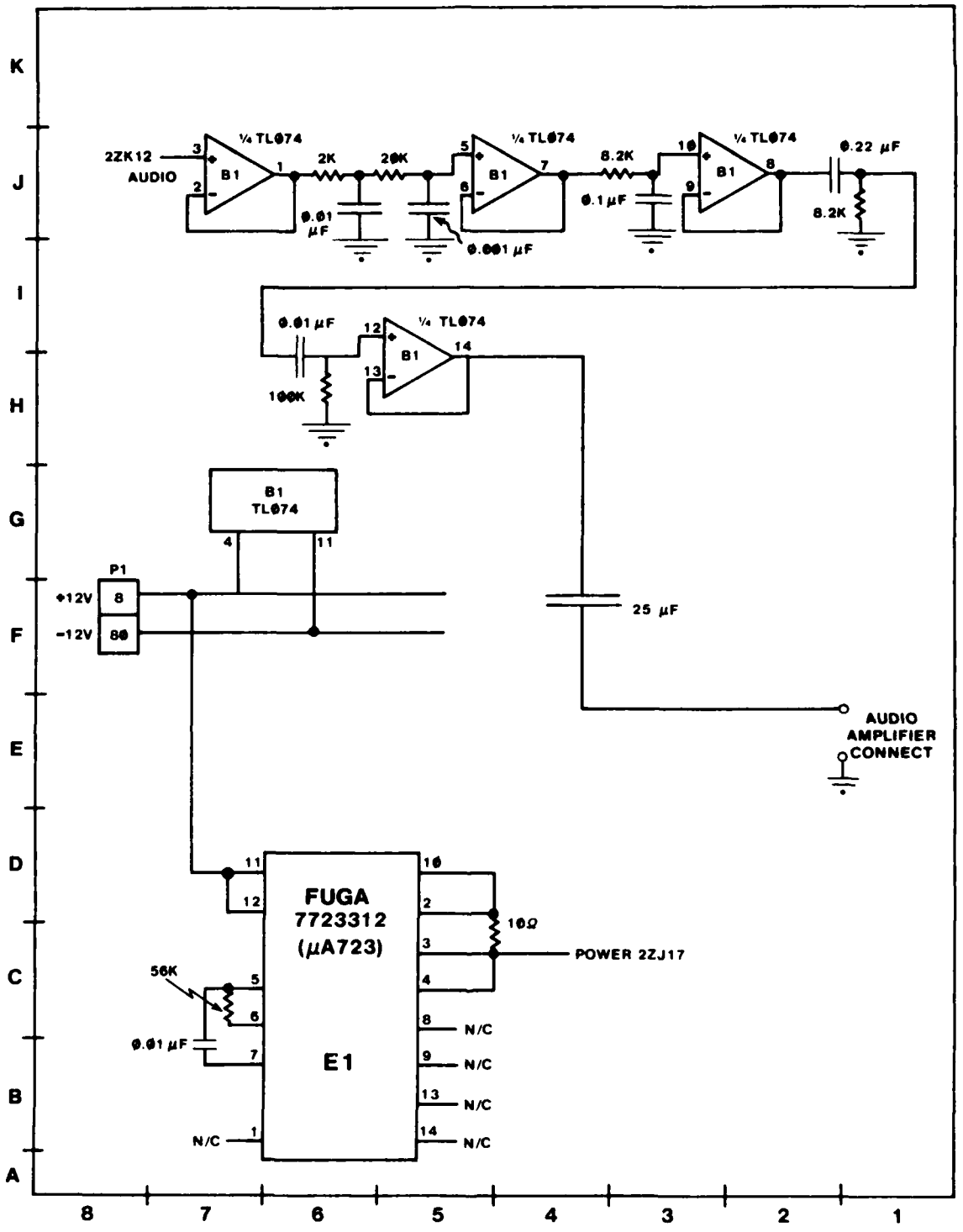


Figure III-8. Computer Synthesized Voice Subsystem Schematic (3 of 3).

The speech chip accepts data from the 8741 and processes it to produce an analog signal representing the desired word or phrase. The data are actually the beginning addresses of code for the desired words or phrases in the speech ROMS. When the speech processor receives a proper beginning address, it reads the data and converts the digital coding into an analog output. This output is the desired word or phrase plus a large number of high frequency harmonics. These harmonics must be filtered out in order to produce intelligible speech. That is the purpose of the analog filter on the board. The output of this filter can be amplified or used to drive a small speaker directly.

A UA723 quad op-amp chip with appropriate resistors and capacitors is used to implement a band-pass filter with the frequency response recommended in Figure 5 of National Semiconductor Application Note 252 (by Jim Smith and Dave Weinrich, December 1980).

Provision has been made to install a Burr-Brown PGA100AG Digitally-Controlled Programmable Gain/Multiplexed Input Operational Amplifier chip in the circuit between the filter and the speaker. This makes it possible to add software gain control of the speech by installing the chip on the board.

In order to use the computer voice boards, it must be plugged into the "Multibus." The address for data is 0D000H and the address for the 8741 status register is 0D002H. Before data are sent to the computer voice board, the controlling processor must first check to see if the 8741 is ready to receive data. This is done by reading the status register and checking the second bit (bit 1). If the bit is a logical one then the computer voice board is busy and the controlling processor (MFS board) must wait until it is not busy. The "Multibus" inverts all data and address information. The address decoding logic of the Computer Voice board re-inverts the address. Therefore, the address should be send "true." All data, however, must be sent inverted or inverted upon reception.

Once the controlling processor has determined that the computer voice board is ready to receive data, it writes a byte of data to the data address. The busy flag must be checked before each byte is sent.

For each word or phrase that is to be spoken, the computer voice board must receive three bytes of data. The first should be the page of memory where the data are located. The second is the address of the data within the page. The third is the volume desired. Since the volume control chip has not been installed, this byte has no effect on the speech. The byte must be sent, however, because the software in the 8741 expects to receive volume control information and the board will not function without it. The volume control byte may have any value; however, to be safe should the volume control chip ever be installed, this byte should be zero. This makes the software fully compatible with the volume control chip, whether or not it is present on the board.

The computer voice board will remain busy until it has completed saying a word or phrase unless it is reset by the system reset. Since this is in the millisecond range, it may cause problems for the controlling processor unless this is taken into account when software is written.

A flowchart of this software is provided in Figures III-9 through III-11. A source code listing is included in the Appendix of this report. The first section of code initializes the computer voice board. The gain control word is set to zero to produce unity gain. The 8741 contains a short routine which causes the speech chip to initialize itself. This is necessary because the speech chip has no hardware reset pin. It is therefore necessary to send a set of commands which cause the speech chip to cycle through its built-in routine and stop at the beginning of the routine.

After the initialization has been completed, the 8741 enables its data-bus interrupt and enters a software halt loop. The status register will now show that the 8741 is ready to receive data. Reception of the first data byte, which should be a page number, causes the 8741 to enter a loop which inputs the page number, the word address, and the volume control byte and stores them in its internal RAM. Once all three control bytes have been received, the 8741 proceeds to output the proper control signals to latch this information into speech ROM address decode logic, the speech chip, and the volume control chip. When this process is complete the 8741 reenters the halt loop. The purpose of receiving all three bytes before outputting anything is to minimize the time that the controlling processor must remain tied up sending the data.

A description of the controlling processor is contained in the TOW Speech Module. The controlling processor and its operating format are described in TOW Speech Module (Sec. III C, above).

The speech data in the speech ROMS are encoded to reduce the amount of storage space needed. This is done in a number of steps. First the waveform is phase-angle adjusted to produce mirror symmetry. Second, the low-level portions of the waveform are replaced with zeros. Delta modulation is used to encode the remaining waveform. As a result, only one fourth of the original waveform actually must be encoded. These data, along with information on periods of silence (zeroed section of the waveform) and mirroring are all that need be stored. The speech chip uses this stored information to reproduce an approximation of the original waveform that is close enough to the original that, with the proper filter, amplifier, and speaker, the original speaker can be recognized. For further information refer to National Semiconductor AN-252.

The computer voice board is used for three basic purposes which will be described in the following paragraphs. These are: (1) coaching the trainee during a missile simulation, (2) debriefing the trainee at the end of a simulation, and (3) assisting the squad leader by generating the repetitive commands specified by the TOW manuals.

Coaching is an optional function that uses words such as "low," "high," "left," "right," etc., to correct the trainee's aim. Coaching is selected by typing a control-C character (i.e., by holding down the key marked "CNTL" on the keyboard and pressing the letter "C"). A control-N will disable the coaching.

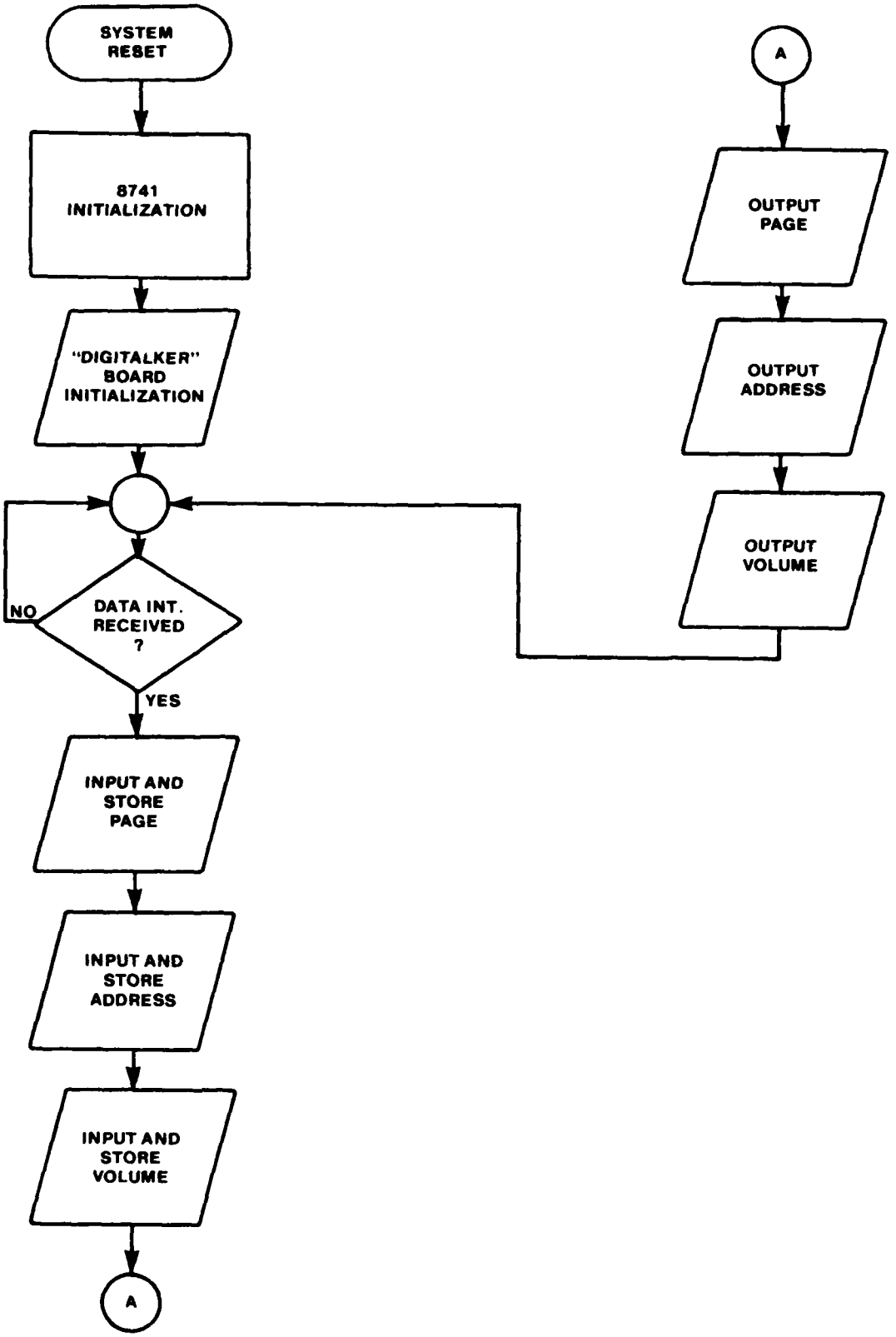


Figure III-9. Flowchart of UP 241 W, 019.

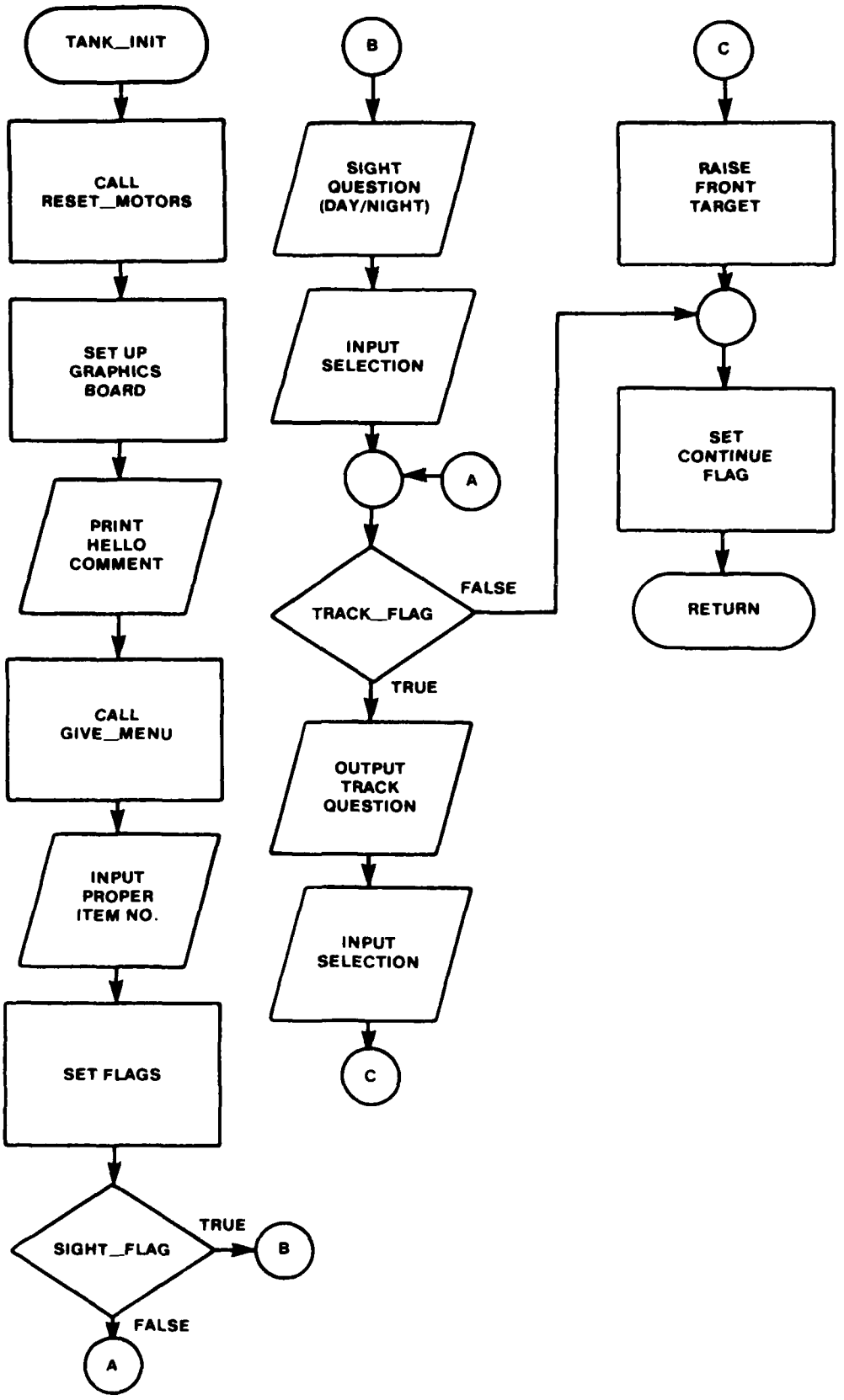


Figure III-10. Flowchart of Tank-Unit Procedure.

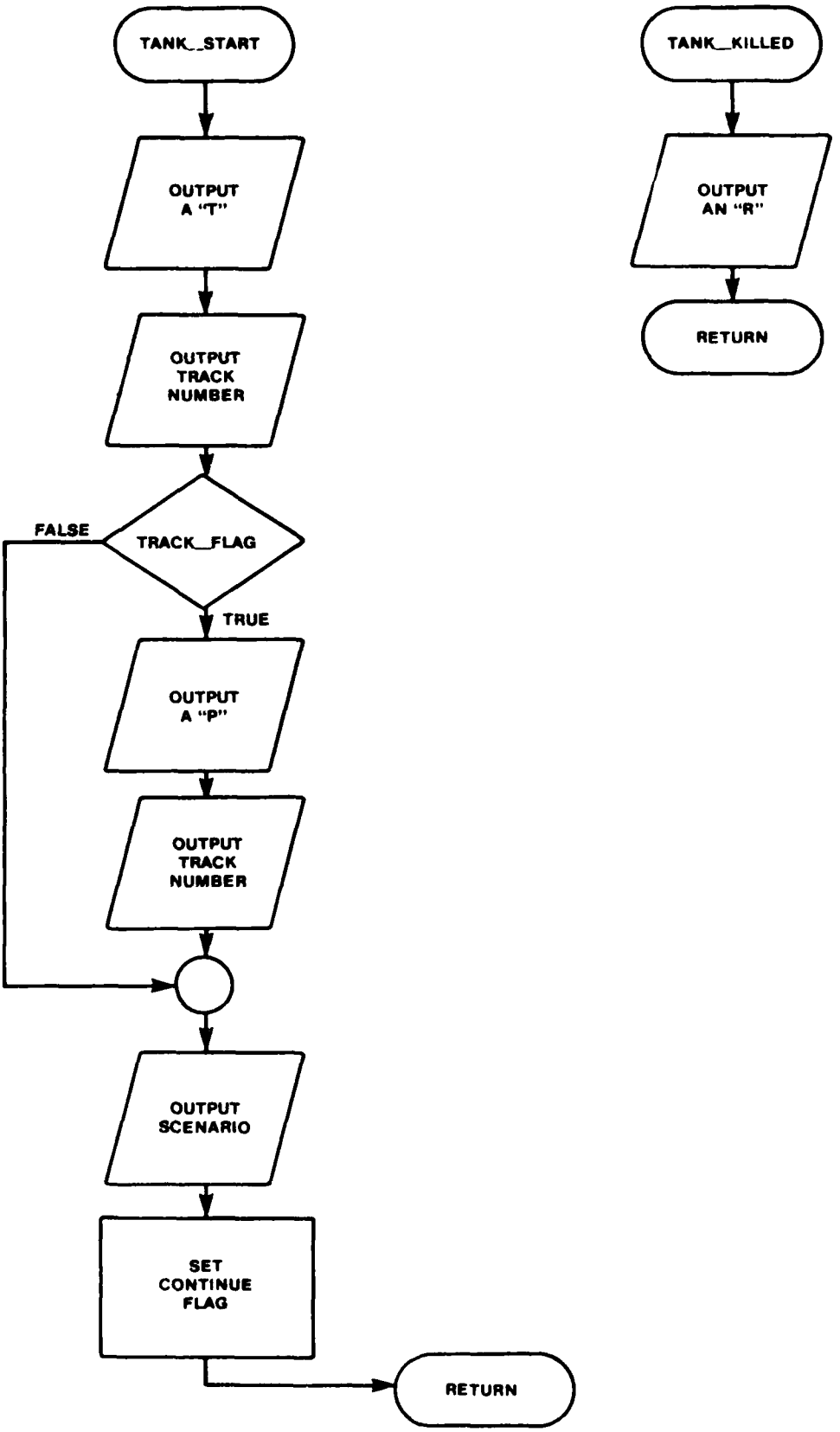


Figure III-11. Flowchart of Tank-Start and Tank-Killed Procedures.

At the end of a simulated missile flight, the system will announce "hit" or "miss" according to the student's performance. Further, if a miss occurred, the direction and distance of the miss will be announced.

The TOW manuals recommend a number of repetitive commands, such as "squad, tank, east, 3,000 meters," and "fire," which the system issues by using the computer voice board. These commands are used during each missile flight simulation as appropriate to the selected scenario.

The computer voice provides coaching, debriefing and the issuance of repetitive commands. Voice recognition offers the possibility for future system enhancement by allowing personnel to communicate verbally with the system controls.

E. COMPUTER GENERATED SOUND SYSTEM

Simulation of sounds produced during an actual TOW missile firing is accomplished by interfacing a pair of Intel 8748 microcomputers to three General Instruments AY-3-8910 programmable sound generators. These microcomputers are referred to as the TOW SOUND CONTROLLER (TSC) and the RETURN FIRE CONTROLLER (RFC) in Figure III-12. Also shown on the block diagram are the three programmable sound generators, PSG-A, PSG-B and PSG-C. Figure III-13 is a detailed schematic of the interconnections between these devices.

Data necessary for the PSGs to produce sound are stored in the permanent memory of the TSC and the RFC. During missile flight time, the MFS processor simply selects the sound to be made and communicates its choice over the 4-bit bus designated in Figure III-13 by lines INTERRUPT, DATA1, DATA0, and RETURN FIRE. This approach allows the MFS processor to handle sound-making decisions with minimum time taken from its primary functions. The table below describes the choice of sounds available to the MFS processor and the corresponding 4-bit bus value:

		INTERRUPT	/ DATA1	/ DATA0	/ RETURN FIRE
(1)	Gyro wind-up, Launch explosion, and Rocket motor burn	0	1	1	1
(2)	Target hit explosions	0	1	0	1
(3)	Ground impact explosion	0	0	1	1
(4)	Return fire sounds	1	X	X	0

Microcomputer RFC is dedicated to producing return fire sounds only. PSG-C is connected directly to the RFC and is dedicated to it. If the 4-bit word shown in (4) appears on the bus (DATA1 and DATA0 are "don't cares"), then a sound simulating an incoming round is produced. It is recommended that either an 8749- or an 8751-based sound system be implemented in future prototypes in order to provide a broader range of sounds within a single chip.

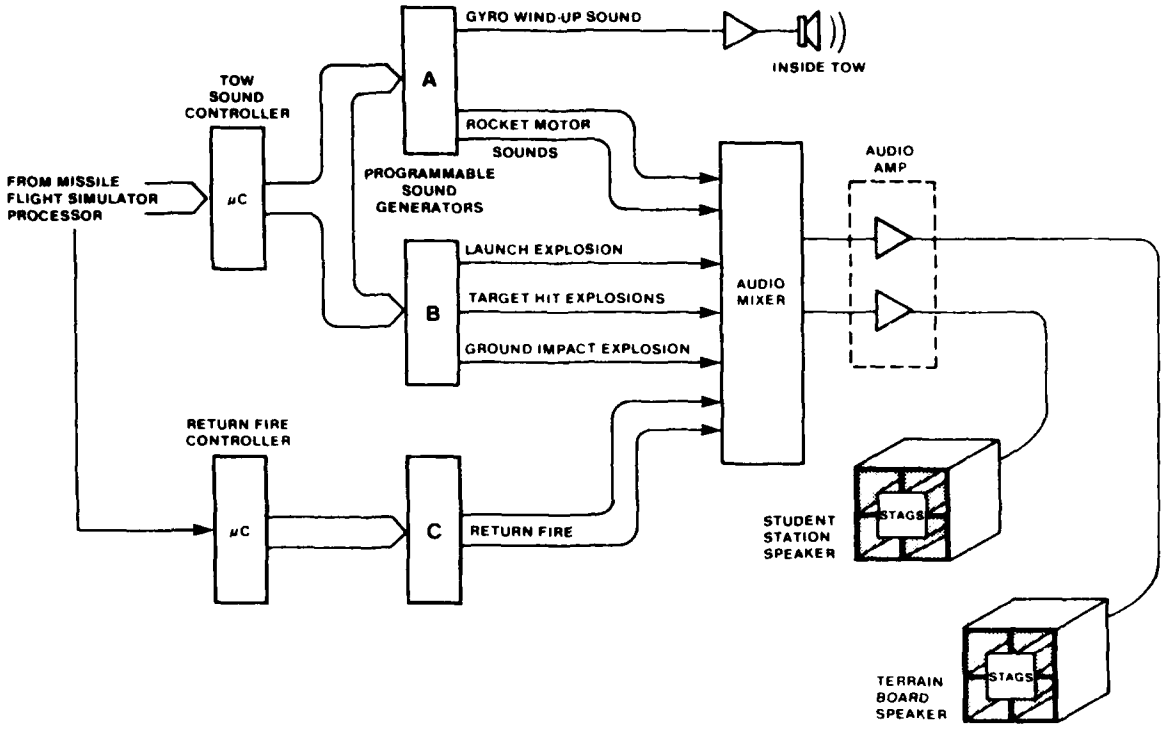


Figure III-12. Computer Generated Sound System.

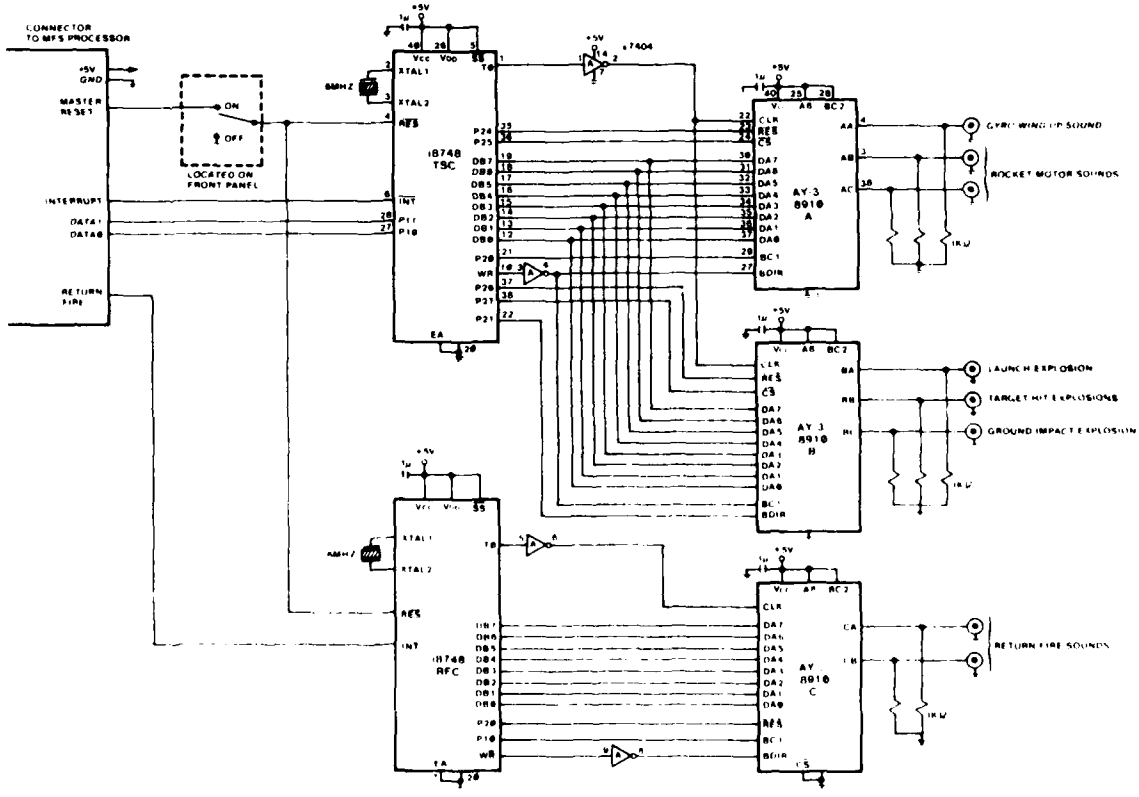


Figure III-13. Computer Generated Sound Schematic.

The MFS processor detects trigger pull and signals the TSC with the 4-bit word shown as (1) in the table above. No more than 15 microseconds are required by the TSC to capture the values of DATA1 and DATA0 on port 1. The TSC decodes this word to mean trigger pull and initiates a sequence which automatically produces a 1.5 second gyro wind-up sound, followed by a launch explosion, followed by a 0.5 second rocket motor burn, and ending with a sound which simulates the missile fading into the distance. Likewise, if the TSC decodes the word as shown in (2) then a series of 3 explosions, increasing in amplitude, are produced through PSG-B. A single ground impact explosion is produced through PSG-B when the word in (3) is decoded. Complete software listings for the TSC and the RFC are shown in Appendix D.

The General Instruments programmable sound generator is a 40-pin, 8-bit device with microprocessor compatibility. The device features three independent analog channels, each with access to its own tone generator. A 16-control register array communicates to the microcomputer through an 8-bit bi-directional port. Four lines are alloted for bus control logic (read, write, and chip select). Each tone generator looks to two registers within the array for a 12-bit tone period. A range of frequencies covering the full eight octaves of the equal-tempered chromatic scale is available.

Pseudo-random noise may be mixed to any or all channels from a noise generator with basic frequencies of 4 Khz to 125 Khz. Two modes of output control are available for each channel: fixed and variable amplitude. The fixed level amplitude mode selects an amplitude specified in the array by the microcomputer. For use in this system the variable amplitude mode is selected, forcing an envelope generator to control the shape and cycle of all outputs. Controlling the envelope generator is a 16-bit tone period within the array allowing for frequency ranges of 12 Hz to 8 Khz and a 5-bit shape/cycle control register. Three D/A converters supply 0 to 1 volt signals to the output channels.

F. MINIATURE TARGET BOARD

The STAGS-TOW lab model sight optics yield magnifications and fields-of-view replicating those of the operational sights of approximately 3000 m. This scale range was chosen so as to take advantage of ready availability of 1:285 scale models. These models are commonly used in war-gaming and come in all NATO and WARSAW PACT vehicles.

These low cost models are manufactured by "Micro Armor" and have excellent detail. The models are modified so that they ride securely along the track and are easily interchangeable.

In order to provide a variety of scenarios with which a trainee can engage a target, three tracks with unique characteristics have been mounted inside the terrain board. Targets on all three tracks run perpendicular to the trainee's line of sight and can move either left-to-right, or right-to-left. The chain mechanisms which carry the targets along the tracks are driven by Superior Electric "Slo-Syn" synchonous/stepping motors with 200 steps per revolution. The frontmost track (Track 1) has the added capability of vertical movement of the target. An Airpax linear actuator is mounted with the chain mechanism and raises or lowers the target to simulate moving over rough terrain, or driving into (out of) a ravine. The center target (Track 2) moves over hilly terrain and developes the trainee's skill at combined

elevation-and-azimuth control of the missile. The rearmost target (Track 3) moves in and out of cover provided by the hills of Track 2. This target can rotate about its axis while moving laterally which, through the sight, appears as a frontal or oblique target. Rotation is imparted through a North American Phillips stepping motor (7.5 degrees per step) mounted with the chain mechanism. All motors are driven from similar Darlington drive circuits, shown in Figure III-14. These drive circuits reside inside the terrain board. Four Darlington pairs exist for each phase of a stepper motor.

All targets move across 40-inch tracks. The stepper motors traverse this distance in 5,240 steps, which equates to a positional accuracy of .0076 inches at any point along the track. By using the 1/285 scale models, the position of a tank at a range of 3,000 meters in the real world is known to within 2.18 scale inches.

Each track has an infrared light source which moves with the target. The light is provided by an incandescent lamp filtered by an 87C Kodak Wratten Gelatin filter. A light pipe is used as a conduit and is bent to project the light toward the student station. The lamps operate independently and are controlled by the MFS processor. Scenarios are written in a manner which provides a single light source throughout the missile flight. In this way, multiple targets can be presented to the trainee so long as only one target is present when the missile reaches the target plane. This concept also provides for a transfer of targets during the first seconds of flight time. The driver circuit for the lamps is mounted inside the terrain board; the schematic is presented in Figure III-15.

A model-control board resides within the instructor's console. Five intelligent stepper motor controllers are mounted on this board, each dedicated to one of the five stepper motors driving the targets. Figure III-16 is a block diagram of the model-control board and Figure III-17 is a detailed schematic.

The controllers utilized are Cybernetic Microsystems CY512. The CY512 controller is a standard five-volt, 40-pin LSI device configured to control a 4-phase stepper motor. The CY512 interfaces to a microcomputer through a bi-directional 8-bit port. High level commands sent to the CY512 are stored in an internal program buffer. The CY512 then acts as a stand-alone device controlling the stepper motor in accordance with the program in the buffer.

Scenarios are selected at the instructor's console. These scenarios are a sequence of high level commands stored in the PIP. When the instructor initiates the scenario, the PIP passes these commands to the Stepper Motor Coordinator (MC). This device is an Intel 8741 Universal Peripheral Interface which resides on the model-control board (see Figure III-17). The MC decodes the data from the PIP and routes the high level commands to the proper Stepper Motor Controller (SMC-1 through SMC-5 in Figure III-17). Two input/output (I/O) expanders (Intel 8243s), shown as IOE-A and IOE-B on Figure III-17, provide the MC with the I/O necessary to monitor and control the flow of data to all five SMCs.

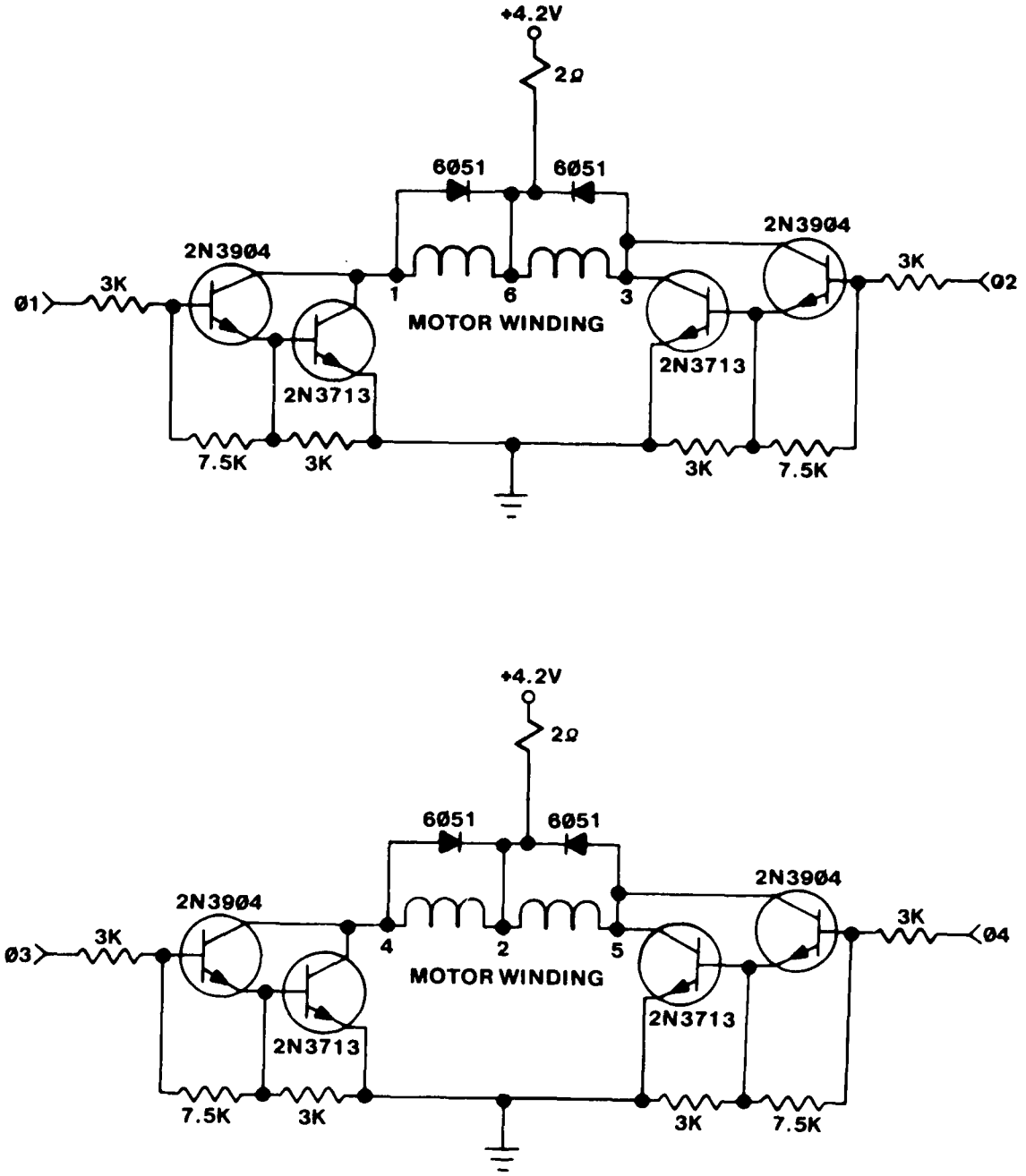
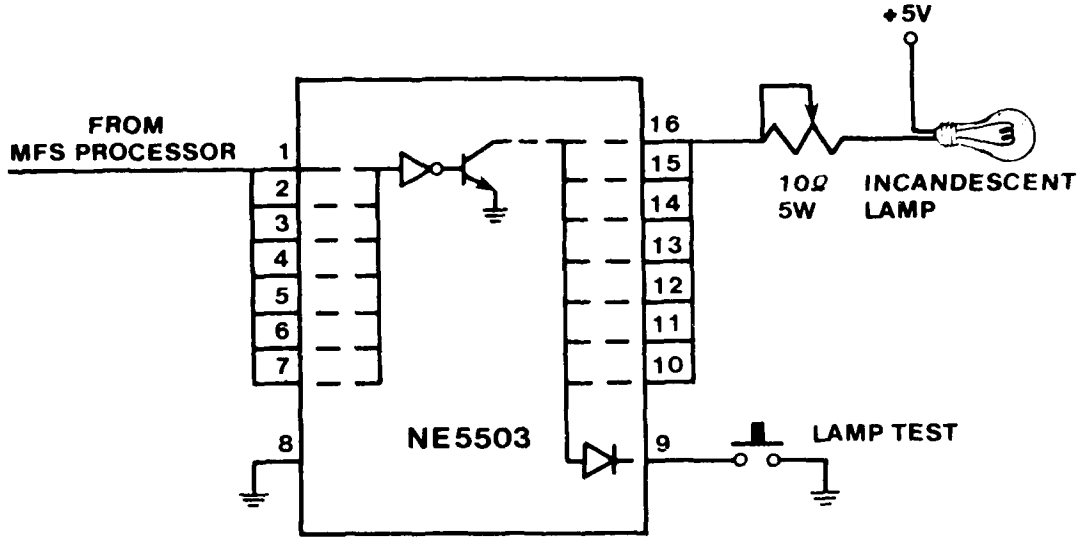


Figure III-14. Stepper Motor Drive Circuits.



TYPICAL CIRCUIT - ONE PER TRACK

Figure III-15. IR Source Driver.

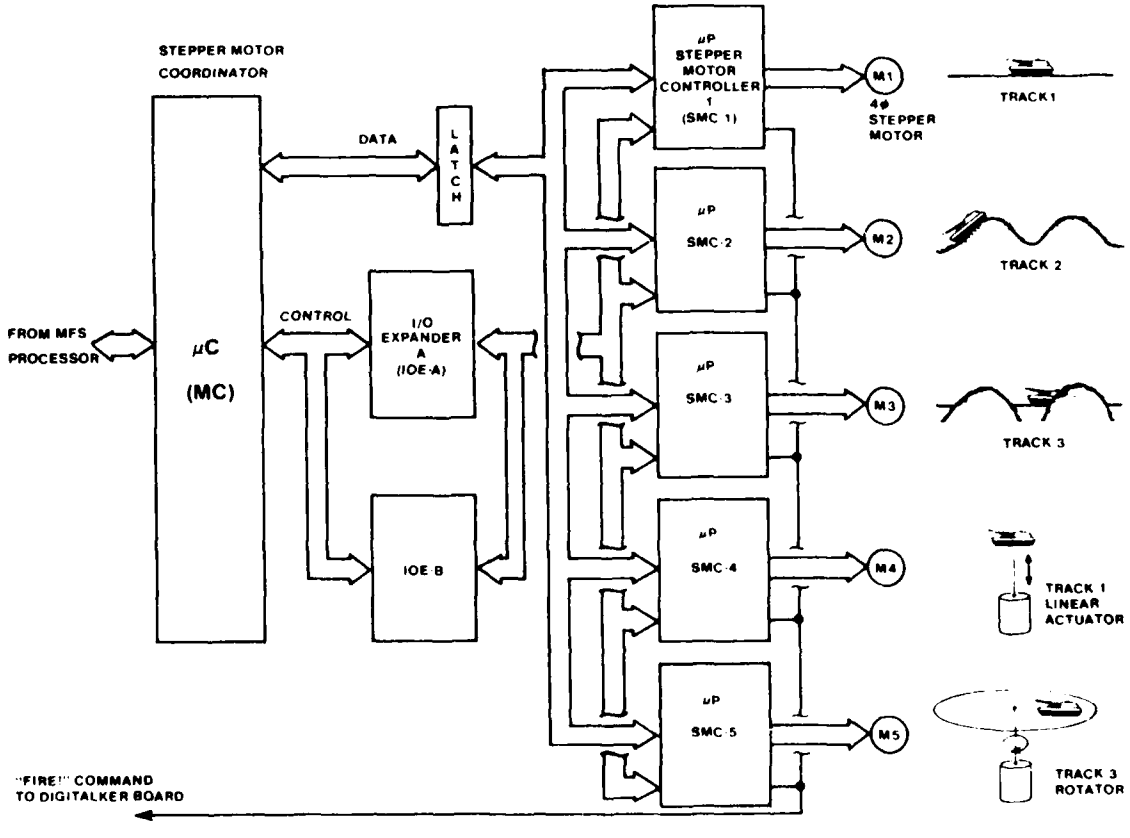


Figure III-16. Terrain Board Control System.

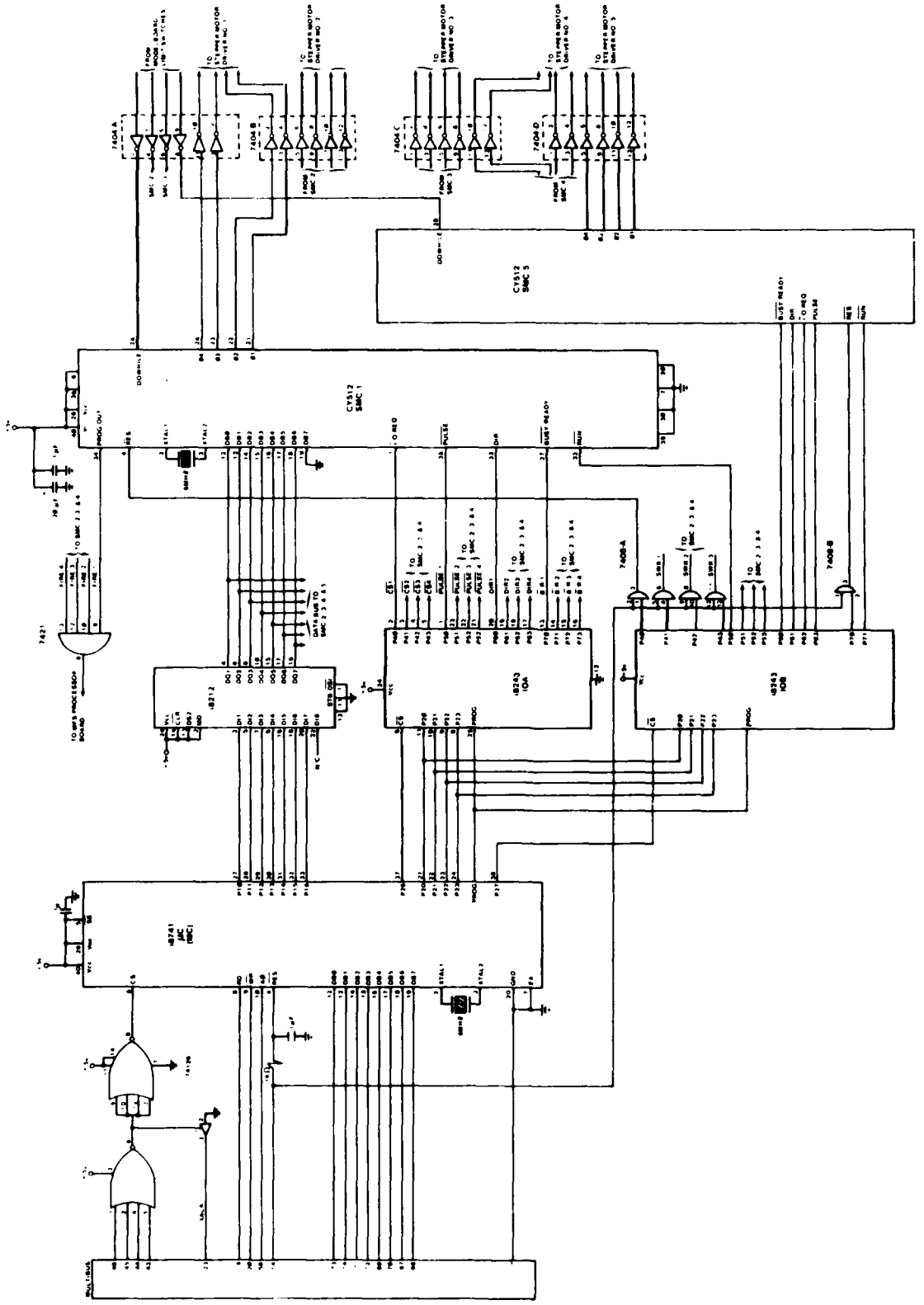


Figure III-17. Terrain Board Control System Schematic.

The MFS processor requires the position of the target be known at all times throughout the missile flight. Although scenarios may call for movement of two or more tanks, only one of them can be a hit-eligible target, for threat-vulnerability and target-transfer training considerations. The MC is instructed as to which track the target resides upon. By monitoring the PULSE lines of SMC-1, SMC-2, and SMC-3, the MC is able to keep an ongoing count of the target position and pass this count to the MFS on command.

SMC-1, -2, and -3 signal the MFS processor when to issue the vocal command "Fire." The scenario loaded into the controller instructs the controller as to the point in time when to strobe the programmable output pin shown as FIRE in Figure III-14. This signal travels directly from the model control board to the MFS processor.

A complete listing of the software contained within the MC appears in Appendix E.

G. TOW STATISTICAL PACKAGE

The TOW Statistical Package adds computational programs to supply GAE data in terms of cumulative mean and standard deviation statistics compiled from periodic sampling of the data shown in the GAE graphic displays.

Aiming errors in both elevation and azimuth are shown in Figure III-18. Figures III-19, III-20 and III-21 show the output of the STAGS-T Statistical Package for the complete flight and for a partitioning of the flight into time intervals of [0,4] and [4,14] seconds.

The TOW Statistical Package was added to STAGS-T in response to a request from the U.S. Army Human Engineering Laboratory dated 20 January 1983.

STAGS-T program additions to provide the statistical capability are detailed in Appendix G.

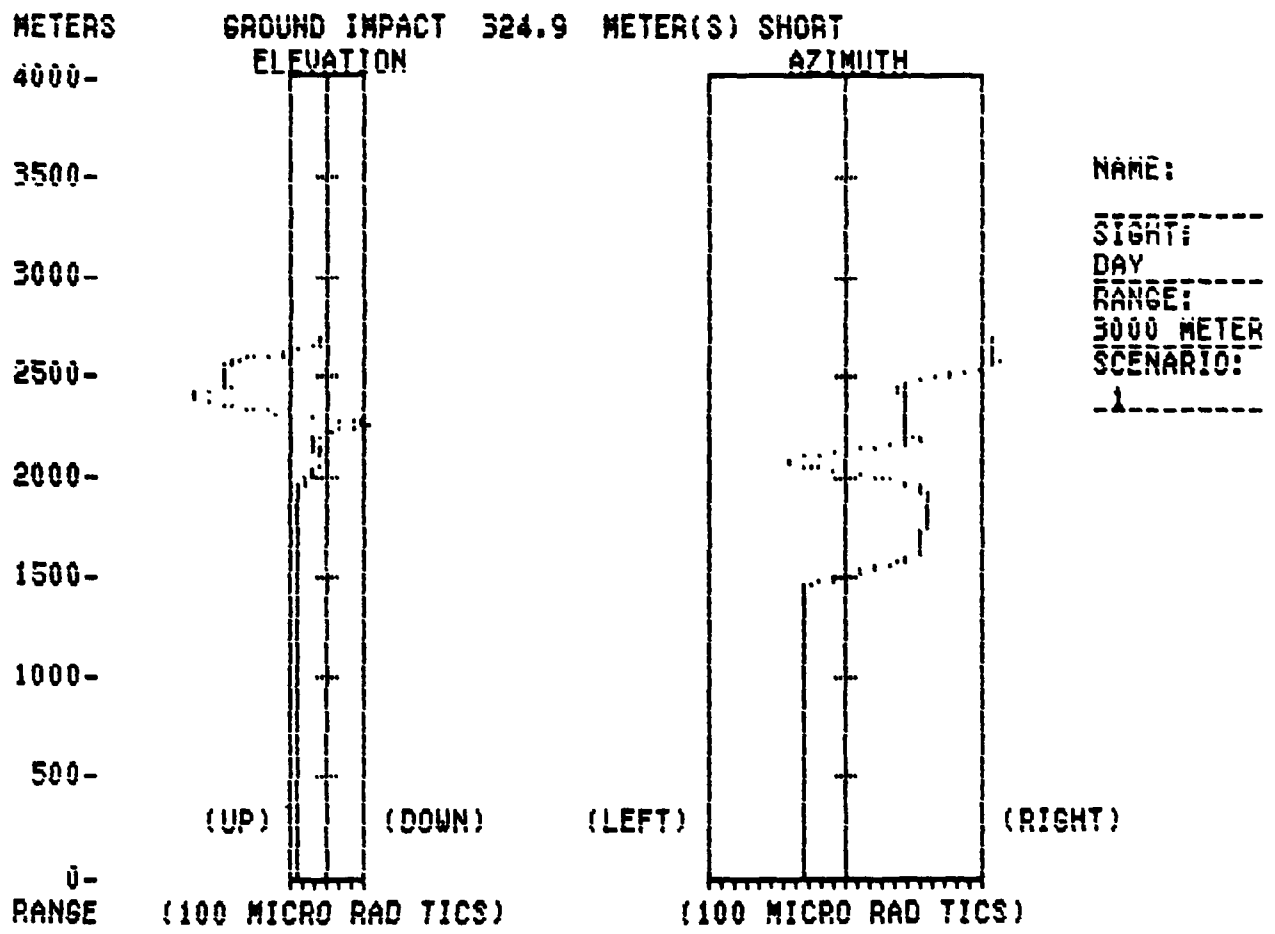


Figure III-18. Gunner Aiming Error in Elevation and Azimuth.

STAGS/T STATISTICAL PACKAGE
(MEAN AND STANDARD DEVIATION IN MICRORADIANS)

ENTER START TIME: 0

ENTER END TIME : 14

ELEVATION		AZIMUTH	
MEAN :	274	MEAN :	96
STANDARD DEVIATION:	238	STANDARD DEVIATION:	523
DIRECTION:	UP	DIRECTION:	RIGHT

Figure III-19. Statistics for Complete Flight.

STAGS/T STATISTICAL PACKAGE
(MEAN AND STANDARD DEVIATION IN MICRORADIANS)

ENTER START TIME: 0

ENTER END TIME : 4

ELEVATION		AZIMUTH	
MEAN :	236	MEAN :	354
STANDARD DEVIATION:	0	STANDARD DEVIATION:	0
DIRECTION:	UP	DIRECTION:	LEFT

Figure III-20. Interval [0,4] Statistics.

STAGS/T STATISTICAL PACKAGE
(MEAN AND STANDARD DEVIATION IN MICRORADIANS)

ENTER START TIME: 4

ENTER END TIME : 14

ELEVATION		AZIMUTH	
MEAN :	294	MEAN :	324
STANDARD DEVIATION:	290	STANDARD DEVIATION:	507
DIRECTION:	UP	DIRECTION:	RIGHT

Figure III-21. Interval [4,14] Statistics.

SECTION IV

CONCLUSIONS

This system has undergone preliminary evaluation by both U.S. Marine Corps and U.S. Army experienced TOW gunners. All gunners were favorably impressed with its realism and teaching attributes. The U.S. Army Infantry School has proposed Training Effectiveness Evaluation in 2FY84.

STAGS research models have been designed for both DRAGON and TOW.

The STAGS-DRAGON is currently under contract for Engineering Development models; STAGS-TOW is about to enter the same status.

Investigation is underway to seek out other weapon-systems for which exploitation of the STAGS technology could fulfill training needs now not adequately addressed.

APPENDIX A

TOW FLIGHT SIMULATION EQUATIONS

This appendix describes the mathematical model of TOW flight dynamics as implemented in STAGS-TOW.

The initial model was based on constant values for system damping and natural frequency as well as constant missile velocity. The final model is similar except for the inclusion of variable damping, natural frequency, and down-range velocity. These system characteristics are treated as quasi-stationary in the incremental solution to the flight equations.

Design considerations for the TOW missile components as well as the overall system, which is of primary importance for STAGS-T, are detailed in the TOW 2 Weapon System, Systems Characteristic Document (T-24) Revision A.[1] As stated on pages 3-58 and 3-63 of the Systems Characteristic Document (T-24), "The primary closed loop system response is characterized by a pair of underdamped poles." A design goal was a damping ratio between 0.4 to 0.5. The closed loop natural frequency ranges from 8 to 2 radians per second. The Laplace transformed system function for horizontal motion is therefore

$$(1) \qquad \frac{Y(s)}{Y_C(s)} = \frac{\omega^2}{s^2 + 2\xi\omega + \omega^2}$$

where s = Independent Laplace variable

Y = Horizontal distance of the missile from the initial launch line

Y_C = Commanded Y distance

ξ = Damping constant

ω = Undamped natural frequency, radians per second.

An identical expression holds for vertical, Z, displacements although the damping ratios and natural frequency parameters are slightly different.

The command displacement in the Y direction is influenced by:

(a) Target motion
(b) Gunner aiming error
(c) Compensation for missile horizontal acceleration as discussed later.

Vertical or Z displacement commands are similar except that possible evasive maneuvers are very limited with the result that the differential equation for this direction is somewhat simpler.

---

1. TOW 2 Weapon System, Systems Characteristic Document (T-24) Revision A, Contract: DAAH01-79-C-1360, 13 November 1981.

Ideally, the gunner aiming error will be zero and the missile should fly along the target line from launch to target impact. Thus, for this ideal situation

$$y_c(t) = \alpha(t) \cdot x(t) \tag{2}$$

where t = Time

x = Downrange distance

y_c = Commanded horizontal distance of missile from the launch line

α = Horizontal angular position of target relative to launch line.

Errors and Compensation

y_c, α, and x are functions of time; generally unspecified, but two special cases yield considerable insight into system behavior. Both special cases are characterized by constant downrange missile velocity, V_x, thus

$$x(t) = V_x \cdot t \tag{3}$$

Case 1 is further limited so that $\alpha(t) = \alpha_o$, a constant.

Case 2 is restricted to constant target velocity ω_T in radians per second. Thus $\alpha(t) = \omega_T t$.

For Case 1:

$$y_{c1}(t) = \alpha_o V_x t \tag{4}$$

and for Case 2:

$$y_{c2}(t) = \omega_T V_x t^2 \tag{5}$$

When considered as a classical closed-loop control system, Case 1 represents a ramp or constant velocity input for $0 < t$ while Case 2 represents a parabolic or constant acceleration input for $0 < t$.

All systems which can be described by equation (1) are said to be "Type 1 systems." Such systems respond to ramp and parabolic inputs in well-defined ways once initial transients have died-out and steady-state conditions are attained. Defining error, $\bar{e}$

$$\bar{e}(t) \triangleq y_c(t) - y(t) \tag{6}$$

and

$$\dot{\bar{e}}(t) \triangleq \frac{d\bar{e}(t)}{dt} \tag{7}$$

Table I may be shown to apply during steady-state for the two cases of interest.

Table I. Errors Caused by Specified Inputs.

	CASE 1	CASE 2
	constant velocity $\text{vel} = \alpha_o V_x$ $0 < t$	constant acc $\text{acc} = 2\,\omega_T V_x$ $0 < t$
$\bar{e}$ steady-state	$\frac{2\,\xi\omega}{\omega^2}(\alpha_o V_x)$	∞
$\dot{\bar{e}}$ steady-state	ϕ	$\frac{4\,\xi\omega}{\omega^2}(V_x\,\omega_T)$

From Table I:

(a) A steady state miss error occurs when the target is stationary at an offset angle, α_o, from the launch line.

(b) A uniform target cross range velocity, V_{TARG}, causes a uniform target angular velocity, $\omega_T = V_{TARG}/X_{TARG}$. This results in an unbounded error as time becomes infinite but the error has a finite rate of increase as given by the right hand bottom entry in Table I.

$$\dot{e}_{\text{steady-state}} = \frac{4\xi\omega}{\omega^2}\left[V_x\left(\frac{V_{TARG}}{X_{TARG}}\right)\right] \tag{8}$$

An error compensation for Case 2 can be made by adding a compensating lead term to the command input such that

$$y_{comp} = \dot{e}t = \frac{4\xi\omega}{\omega^2}\left[V_x t\left(\frac{V_{TARG}}{X_{TARG}}\right)\right] \tag{9}$$

defining

$$\bar{K} \triangleq \frac{4\xi\omega}{\omega^2} \tag{10}$$

and assuming constant V_x, so that

$$x = V_x t \tag{11}$$

the compensation term becomes

$$y_{COMP} = \bar{K}x\,(V_{TARG}/X_{TARG}) \tag{12}$$

$\bar{K}$ as defined in equation (10) is based on steady-state conditions. A first-order correction to $\bar{K}$ which accounts for a finite missile flight time may be shown to be

$$\frac{K}{\bar{K}} = \frac{t - (1 - 4\xi^2)/2\xi\omega}{t - \bar{K}/2} \tag{13}$$

This is plotted on Figure A-1. For a typical flight time of 11 seconds, the correction to $\overline{K}$ = 1.03 or 3 percent.

Including the correction from (13) in (9) and (3)

$$y_{COMP} = Kx\omega_T = Kx\left(\frac{V_{TARG}}{X_{TARG}}\right) \tag{14}$$

With the actual TOW system the target velocity may be inferred by the sight azimuth sweep rate, i.e., it may be considered that

$$\omega_{SIGHT} = \omega_T \tag{15}$$

ω_{SIGHT} provides a signal which is proportional to the target cross-track horizontal velocity provided no gunner aiming error is present. Even with some gunner aiming error it might be assumed that error averages to zero and, therefore, may be neglected. In STAGS-T target position and gunner aiming error are measured. Letting β represent gunner aiming error

$$\omega_{SIGHT} = \frac{d\alpha}{dt} + \frac{d\beta}{dt} \qquad \omega_{SIGHT} \tag{16}$$

Thus in simulating the TOW system with ω_{SIGHT} representing ω_{TARGET} we have from equation (14)

$$y_{COMP} = K\,x\frac{d}{dt}(\alpha + \beta) \tag{17}$$

The commanded $y_c(t)$, therefore, is

$$y_c(t) = \alpha(t)\,x(t) + \beta(t)\,x(t) + Kx(t)\,\frac{d}{dt}\left[\alpha(t) + \beta(t)\right] \tag{18}$$

Considering the right hand side of (18): the first term is due to target displacement from the initial launch line; the second term represents the effect of gunner aiming error, while the last term provides compensation.

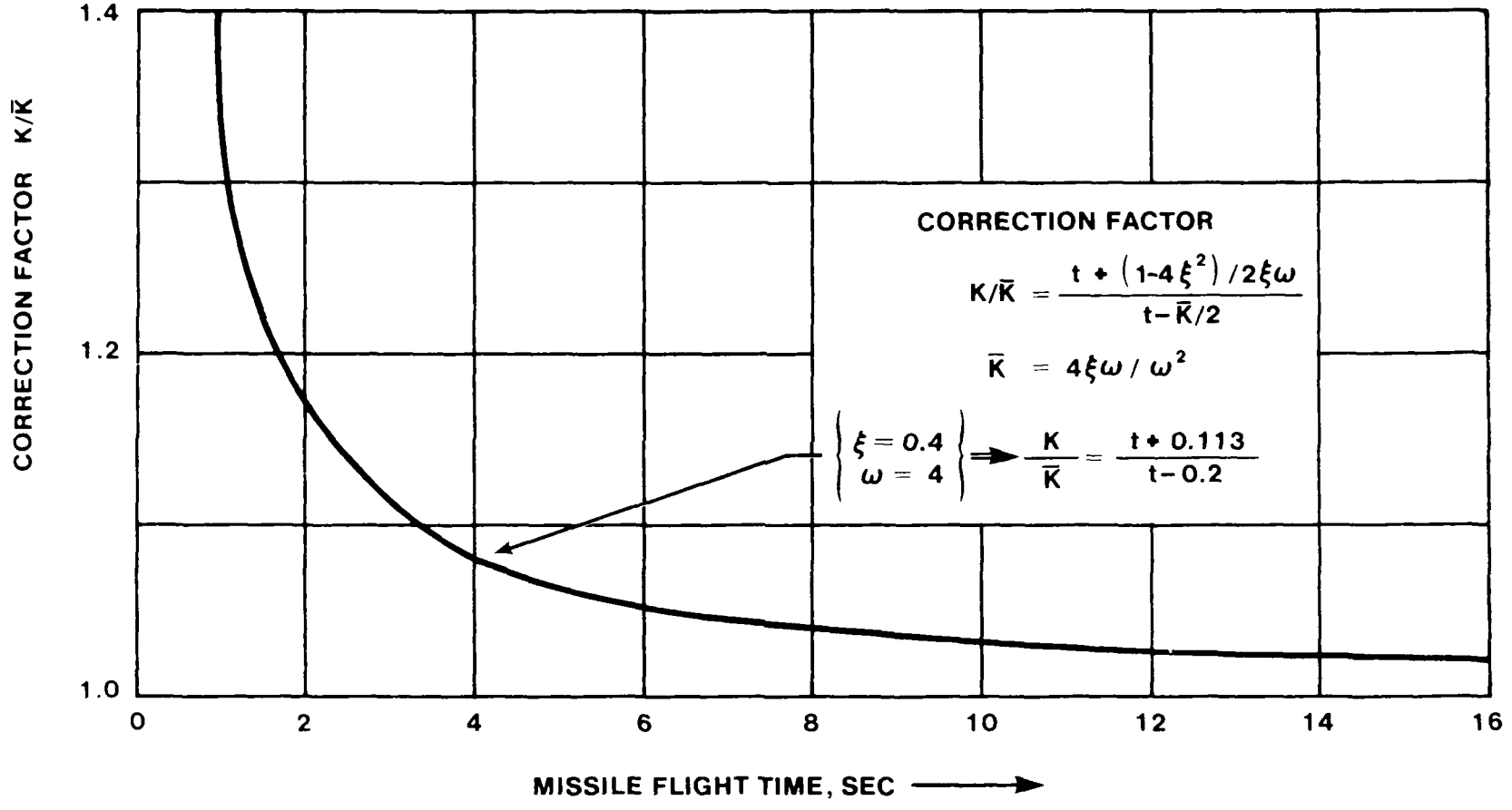

1.4
1.2
1.0
CORRECTION FACTOR K/K̄
0
2
4
6
8
10
12
14
16
MISSILE FLIGHT TIME, SEC
CORRECTION FACTOR
K/K̄ = (t + (1-4ξ²)/2ξω) / (t - K̄/2)
K̄ = 4ξω / ω²
ξ = 0.4
ω = 4
K/K̄ = (t + 0.113) / (t - 0.2)

Figure A-1.

Figures A-2 and A-3 are two equivalent versions of the system block diagram. Figure A-2 relates the input commanded horizontal position to the output horizontal position with an error, $\bar{E}$. Figure A-3 shows the error, E, as the difference between a point on the target line, αx, and the missile position. This figure also shows the difference between E and $\bar{E}$.

Differential Equations

Figure A-3 in the S domain is equivalent to time domain differential equations:

$$\frac{d^2 y(t)}{dt^2} + 2\xi\omega \frac{dy(t)}{dt} + \omega^2 y(t) = \omega^2 y_c(t) \tag{19}$$

$$e(t) = \alpha(t)\, x(t) - y(t) \tag{20}$$

$$y_c(t) = \alpha(t)\, x(t) + \beta(t)\, x(t) + Kx(t) \frac{d}{dt}\left[\alpha(t) + \beta(t)\right] \tag{21}$$

Using the identity

$$\frac{d}{dt}\left\{x(t)\left[\alpha(t) + \beta(t)\right]\right\}$$

$$= \left[\alpha(t) + \beta(t)\right] \frac{dx(t)}{dt} + x(t)\frac{d}{dt}\left[\alpha(t) + \beta(t)\right] \tag{22}$$

$$= \left[\alpha(t) + \beta(t)\right] \dot{x}(t) + x(t)\left[\dot{\alpha}(t) + \dot{\beta}(t)\right]$$

and eliminating y(t) from (19), (20), and (21) gives

$$\frac{d^2}{dt^2} e = \frac{d^2}{dt^2}(\alpha x) - \frac{d}{dt}\left[\omega^2 Kx(\alpha + \beta) - 2\xi\omega(\alpha x - e)\right] \tag{23}$$

$$- \omega^2\left[\beta x - K(\alpha + \beta)\dot{x} + e\right]$$

Integration of (23) for an initially relaxed system yields

$$e = \alpha x + \int\left\{\left[-\omega^2 Kx(\alpha + \beta) + 2\xi\omega(\alpha x - e)\right]\right. \tag{24}$$

$$\left. + \omega^2 \int\left[K(\alpha + \beta)\dot{x} - \beta x - e\right] dT\right\} dT$$

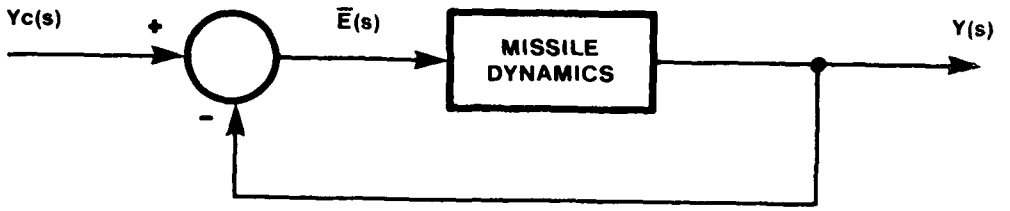


Figure A-2. Block Diagram of STAGS-T.

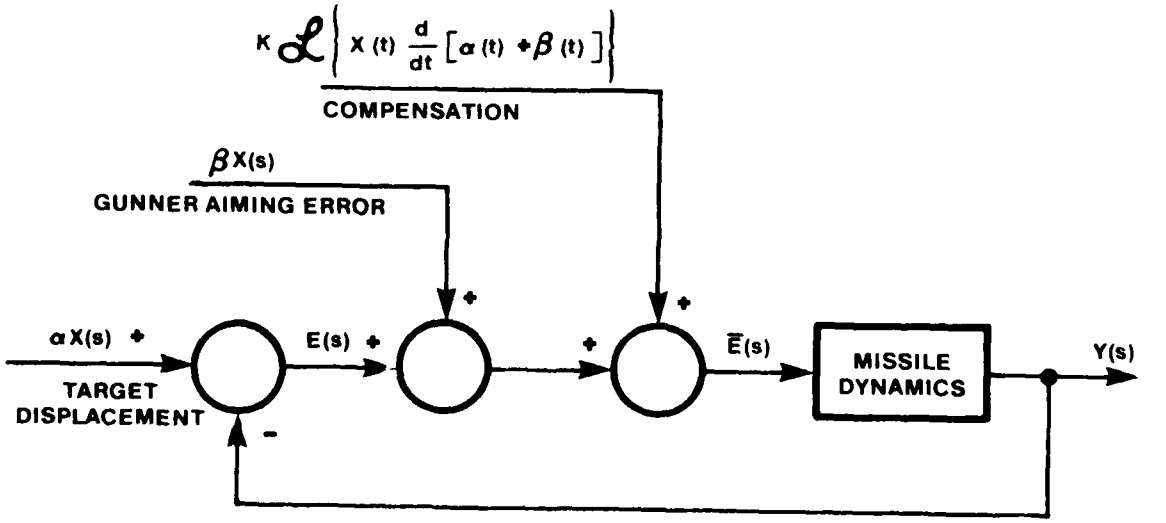


Figure A-3. Equivalent Block Diagram.

or with slight rework (24) becomes

$$e = \alpha x - \int \left\{ 2\xi\omega e + \left[\alpha(\omega^2 K - 2\xi\omega) + \omega^2 K\beta \right] x + \omega^2 \int \left[\beta x - K(\alpha + \beta)\dot{x} + e \right] dT \right\} dT \tag{25}$$

Equation (25) as written in PLM86 for STAGS-T is shown below:

```
ERR_COMP1 = DAMP*OFF_H;
ERR_COMP2 = (HTARG*COEF2 + COEF1*GAEY_F)*X;
INTEGRAND1 = (GAEY_F*X_KOEF*(HTARG + GAEY_F)*VX+OFF_H)/25;
INTEGRAL1 = INTEGRAL1 + INTEGRAND1;
ERR_COMP3 = OMEGA_SQ*INTEGRAL1;
INTEGRAND2 = (ERR_COMP1 + ERR_COMP2 + ERR_COMP3)/25;
INTEGRAL2 = INTEGRAL 2 + INTEGRAND2;
OFF_H _ HTARG*X - INTEGRAL2.
```

The damping and natural frequency of the missile in both pitch and yaw is shown on page 3-72 of the Systems Characteristic Document (T-24). The approximations used in STAGS-T for these coefficients are shown in Figure A-4. Figure A-5 gives the downrange velocity and position.

Equation (25) and the PLM/86 incremental expressions are unchanged. System parameters, however, are updated on each pass through the program. This update occurs 25 times per second. The value of K as used in the compensation term required modification because of the variable parameters. The necessary factor was found by computer simulation to be 0.835, i.e., the quasistatic equations use

$$K = 0.835 \left[\frac{2 \cdot (2\xi\omega)}{\omega^2} \right] \tag{26}$$

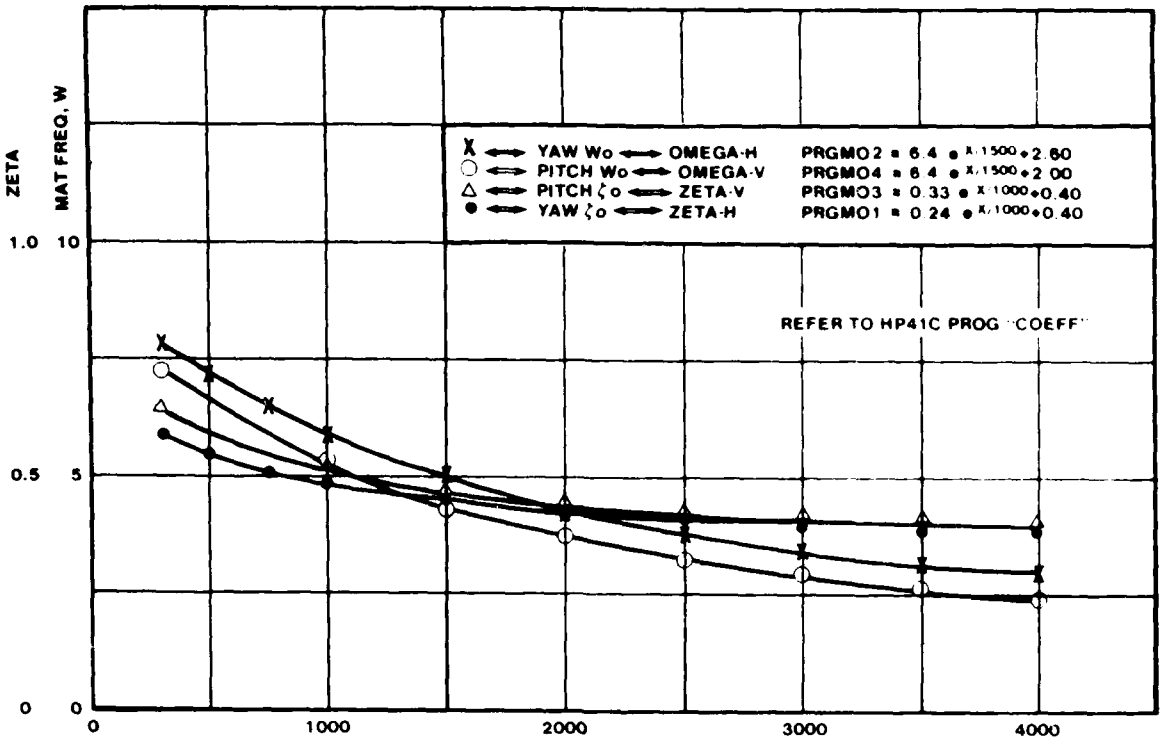


Figure A-4. Coefficient Approximations.

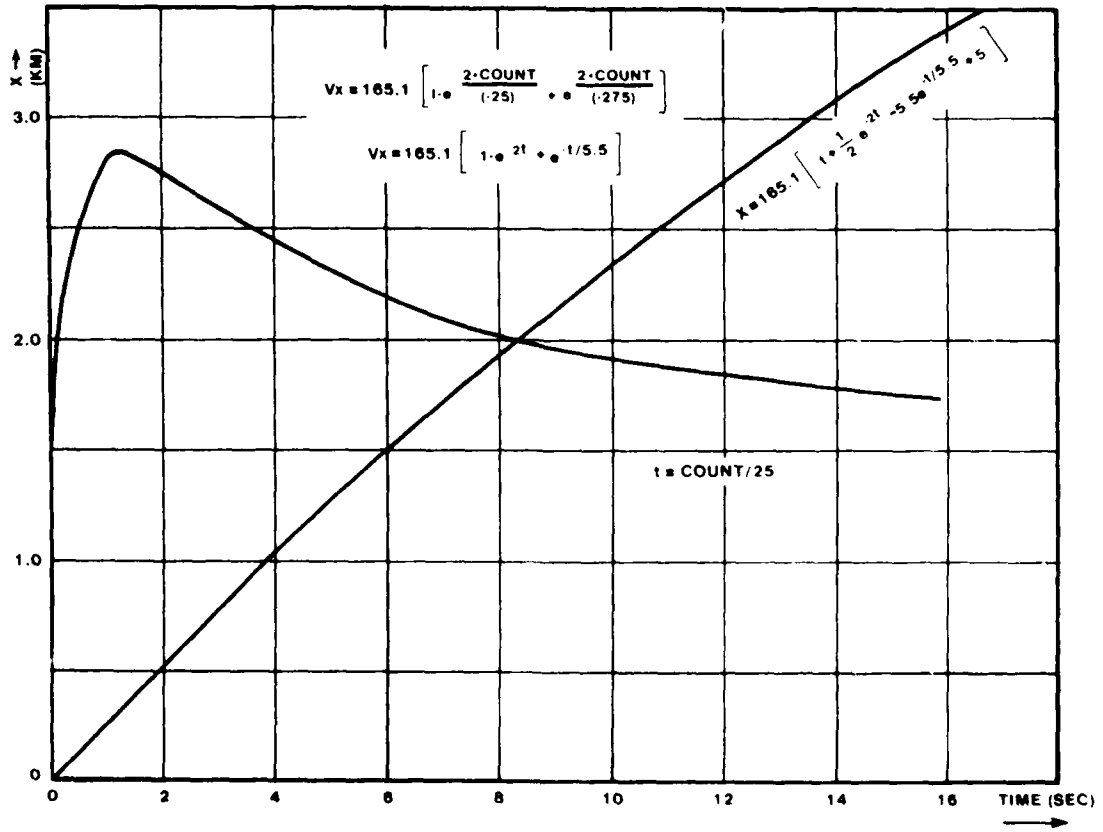


Figure A-5. Downrange Velocity and Position vs. Time.

APPENDIX B

MULTI-PROCESSOR MAIN PROGRAMS

```
21  2               DECLARE ASCII_PTR POINTER;
22  2               DECLARE LAST_ELEMENT WORD;
23  2               DECLARE (ASCII_BUFFER BASED ASCII_PTR) (1) BYTE;
24  2               DECLARE M WORD;
25  2               DO M = 0 TO LAST_ELEMENT;
26  3                  CALL SEND(ASCII_BUFFER(M));
27  3                  END;
28  2               END STRING_OUT;

                /* CONV$SEND CONVERTS A INTEGER TO ASCII CHARACTERS AND SENDS
                   THE CHARACTERS TO THE SMCB */

29  1       CONV$SEND: PROCEDURE(HEX);
30  2               DECLARE ASCII_STRING(4) BYTE;
31  2               DECLARE (REMAINDER,HEX,N) INTEGER;
32  2               HEX = IABS(HEX);
33  2               DO N = 3 TO 0 BY - 1;
34  3                  REMAINDER = HEX MOD 10 + 30H;
35  3                  ASCII_STRING(N) = LOW(UNSIGN(REMAINDER));
36  3                  HEX = HEX/10;
37  3                  END;
38  2               CALL STRING_OUT(@ASCII_STRING,3);
39  2               END CONV$SEND;

                /* NAMES FOR DIRECTION PARAMETERS USED IN CALLING MISS_COMMENT */

40  1       DECLARE MISS_RT LITERALLY '0', MISS_LT LITERALLY '1',
                MISS_UP LITERALLY '2', MISS_SH LITERALLY '4';

41  1       DECLARE FOREVER LITERALLY 'WHILE 1', NO_TRIG_PUL LITERALLY
                'INPUT(PORT_B) AND 1', ACTION BYTE;

42  1       DECLARE SOUND_TRIG_PULL LITERALLY '0';  /* USED TO CALL SOUND */

43  1       H_REPRISE: PROCEDURE EXTERNAL;          /* THE REPRISE PROCEDURE */
44  2         END H_REPRISE;

45  1       SETRET: PROCEDURE EXTERNAL;             /* SETS UP RETICON BOARD */
46  2         END SETRET;

                /* SOUND PROVIDES SIGNALS FOR THE SOUND SUBSYSTEM */

47  1       SOUND: PROCEDURE(KIND) EXTERNAL;
48  2         DECLARE KIND BYTE;
49  2         END SOUND;

50  1       DECLARE FINISHED BYTE EXTERNAL;   /* FLAG SET TO MARK THE END OF FLIGHT */

51  1       DECLARE PORT_B LITERALLY '0CAH';  /* 8255 PARALLEL PORT */

52  1       DECLARE TWO_SEC_INTERVAL LITERALLY '20000', THREE_SEC_INTERVAL LITERALLY '30000';

53  1       DECLARE J INTEGER;

54  1       DECLARE SECOND_TIME BYTE;         /* SECOND FIRE COMMAND FLAG */

                /* COMPONENTS OF THE CY512 PROGRAM  PASSED TO THE SMCB DURING EVASION */
```

```
55  1     DECLARE PROG_START (*) BYTE DATA ('H',0DH,'S 1',0DH,'R ');
56  1     DECLARE PROG_MID (*) BYTE DATA (0DH,'F 1',0DH,'B',0DH,'E',0DH);
57  1     DECLARE PROG_END (*) BYTE DATA (0DH,'G',0DH,'0',0DH,'Q!*');
58  1     DECLARE TID_BIT (*) BYTE DATA (0DH,'N ');

59  1     DECLARE (RATE,STEPS) INTEGER;   /* VARIABLE PARAMETERS OF MOTOR FOR EVASION */
60  1     DECLARE DIR BYTE;

61  1     PROLOG: PROCEDURE EXTERNAL;     /* CAUSES VOICE SUBSYSTEM TO ISSUE INITIAL COMMANDS */
62  2             END PROLOG;

63  1     FIRE_AGAIN: PROCEDURE EXTERNAL; /* CAUSES VOICE SUBSYSTEM TO REPEAT FIRE COMMAND */
64  2                 END FIRE_AGAIN;

65  1     QUIT: PROCEDURE EXTERNAL;       /* CAUSES VOICE SUBSYSTEM TO ISSUE ABORT COMMAND */
66  2           END QUIT;

          /* EPILOG CAUSES COMPUTER VOICE TO ISSUE HIT/MISS COMMENTS */

67  1     EPILOG: PROCEDURE EXTERNAL;
68  2             END EPILOG;

69  1     YZCNTR: PROCEDURE EXTERNAL;     /* OBTAINS GUNNER AIMING ERROR (GAE) */
70  2      END YZCNTR;

71  1     PPI_SET: PROCEDURE EXTERNAL;    /* SETS UP 8255 PARALLEL PORTS_A & _C OUTPUT, PORT_B INPUT */
72  2      END PPI_SET;

73  1     FLIGHT: PROCEDURE EXTERNAL;     /* MISSILE FLIGHT SIMULATION PROCEDURE */
74  2      END FLIGHT;

75  1     INITIATE$VAR: PROCEDURE EXTERNAL;
76  2      END INITIATE$VAR;

77  1     CY512_RESET: PROCEDURE;          /* RESETS ALL STEPPER MOTORS PRIOR TO EVASION */
78  2                  CALL SEND('R');
79  2                  CALL SEND('T');
80  2                  CALL SEND(CURRENT_TRACK AND 03H);
81  2                  CALL SEND('P');
82  2                  CALL SEND(CURRENT_TRACK AND 03H);
83  2                  END CY512_RESET;

84  1     DODGE: PROCEDURE;                /* CONTROLS AND EXECUTES EVASION */

85  2            DECLARE OLD_TCNT INTEGER;    /* STORES PREVIOUS TCNT */
86  2            DECLARE TANK_STOPPED LITERALLY '(TCNT - OLD_TCNT) = 0'; /* NO MOVEMENT BETWEEN PASSES */

87  2            DECLARE GAEY_LIMIT LITERALLY '1.0E-3';      /* AIMING AT TANK WHEN WITHIN LIMITS */
88  2            DECLARE GAEZ_LIMIT LITERALLY '4.0E-4';
89  2            DECLARE ON_TARGET LITERALLY 'ABS(GAEY_F) < GAEY_LIMIT AND
                                           ABS(GAEZ_F) < GAEZ_LIMIT';
90  2            DECLARE PERCENT_FACTOR LITERALLY '(LOW_RANDOM AND 2) = 2'; /* EVASION PROBABILITY IS 50% */

91  2            DECLARE TIMES_UP LITERALLY 'HIGH_RANDOM = 0';  /* RANDOM DELAY */
92  2            DECLARE GOOD_GUNNER LITERALLY 'GUNNER_RATING > 0'; /* NO EVASION FOR RATING OF ZERO */
93  2            DECLARE NOT_TRACK_3 LITERALLY '((CURRENT_TRACK) AND 03H) <> 3'; /* NO EVASION ON TRACK 3 */
```

```
 94   2          DECLARE EVADE_ENABLED LITERALLY 'TIMES_UP AND TANK_STOPPED AND
                                    GOOD_GUNNER AND PERCENT_FACTOR AND
                                    ON_TARGET AND NOT_TRACK_3';
 95   2          IF EVADE_ENABLED
                 THEN DO;

                      /* GET RATE, STEPS, AND DIRECTION */

 97   3               RATE = INT((LOW_RANDOM AND 60H) + 100);
 98   3               IF LOW_RANDOM THEN DIR = 2DH; ELSE DIR = 2BH;
101   3               IF (5240 - TCNT) > TCNT  /* MAXIMUM MOVE IS DISTANCE TO NEAREST EDGEOF MODEL BOARD */
                      THEN STEPS = TCNT/INT((SHR(LOW_RANDOM,2) AND 07H) + 1);
103   3               ELSE STEPS = (5240 - TCNT)/INT((SHR(LOW_RANDOM,1) AND 07H) + 1);

                      /* RESET CY512s, GET CURRENT TRACK AND SEND IT */

104   3               CALL CY512_RESET;

                      /* SEND PROGRAM WITH VARIABLE RATE, STEPS, AND DIRECTION */

105   3               CALL STRING_OUT(@PROG_START,LAST(PROG_START));
106   3               CALL CONV$SEND(RATE);
107   3               CALL STRING_OUT(@PROG_MID,LAST(PROG_MID));
108   3            CALL STRING_OUT(@DIR,0);
109   3               CALL STRING_OUT(@TID_BIT,LAST(TID_BIT));
110   3               CALL CONV$SEND(STEPS);
111   3               CALL STRING_OUT(@PROG_END,LAST(PROG_END));

                      /* DISABLE ALL BUT GUNNER RATING OF 2 */

112   3               GUNNER_RATING = GUNNER_RATING - 1;

                      /* GET RANDOM NUMBERS FOR GUNNER_RATING OF 2 */
                      /* BY DOING A BYTE SWAP, KEEP DELAY BETWEEN MOVES UNDER 1 SEC */

113   3               HIGH_RANDOM = (LOW(UNSIGN(J)) AND 1FH); /* MAX DELAY = 0.6 SEC */
114   3               LOW_RANDOM = HIGH(UNSIGN(J)) AND 1111$1101B;
115   3               END;
116   2          OLD_TCNT = TCNT;                /* SAVE TCNT */
117   2          HIGH_RANDOM = HIGH_RANDOM - 1;   /* RANDOM DELAY COUNTER */
118   2          END DODGE;
          /*****************************************************************
                              PROGRAM STARTS
          *****************************************************************/
119   1     DECLARE START_UP LABEL PUBLIC;
120   1     DECLARE (LOW_RANDOM, HIGH_RANDOM) BYTE;

121   1     CALL TIME(15000);              /* 1.5 SEC TO ALLOW UPI-41'S TO RESET */

122   1     START_UP:
        1       CALL PPI_SET;

123   1     DO I = 0 TO 100;
124   2       H_MIS_ASCII(I) = ' ';       /* CLEAR BUFFERS USED IN REPRISE */
125   2     END;
```

```
126  1     CONTINUE = 0;                /* ZERO FLAG USED BY PROLOG */

127  1     CALL PROLOG;                 /* ISSUE INITIAL COMMANDS */
128  1     J = TWO_SEC_INTERVAL;        /* J USED AS INTERVAL TIMER & RANDOM NUMBER BY DODGE */
129  1     SECOND_TIME = 0;             /* SET IF FIRE COMMAND WAS REPEATED */

130  1       DO WHILE NO_TRIG_PUL;
131  2        CALL TIME(1);             /* 0.1 MSEC DELAY */
132  2        J = J - 1;
133  2        IF (J = 0) AND (SECOND_TIME = 0)    /* REPEAT FIRE COMMAND IF TRAINEE */
               THEN DO;                          /* DID NOT FIRE IN TWO SECONDS */
135  3            CALL FIRE_AGAIN;
136  3            J = THREE_SEC_INTERVAL;
137  3            SECOND_TIME = 1;
138  3            END;
139  2        ELSE IF (J = 0) AND (SECOND_TIME = 1) /* ISSUE ABORT COMMAND IF TRAINEE */
               THEN DO;                           /* HASN'T FIRED IN THREE SECONDS */
141  3            CALL QUIT;                     /* AFTER REPEAT FIRE COMMAND */
142  3            HALT;                          /* STOP PROCESSOR */
143  3            END;
             END;
145  1     CALL SOUND(SOUND_TRIG_PULL);          /* LAUNCH EXPLOSION */

146  1     CALL INITIATE$VAR;              /* FOR TOW FLIGHT */

147  1     LOW_RANDOM = LOW(UNSIGN(J));          /* GET RANDOM NUMBER */
148  1     HIGH_RANDOM = SHL(HIGH(UNSIGN(J)),1); /* RANDOM DELAY FROM 0 TO 9.4 SEC */

149  1     CALL TIME(15000);                     /* SIMULATED LAUNCH DELAY */

150  1     CALL SETRET;                          /* SET UP RETICON CAMERA */

151  1     TOW_FLYS: DO WHILE NOT FINISHED;      /* MAIN LOOP */
152  2       CALL YZCNTR;                        /* GET GAE */
153  2       CALL FLIGHT;                        /* SIMULATE NEXT 40 MILLISECONDS OF FLIGHT */
154  2       CALL DODGE;                         /* EXECUTE POSSIBLE EVASION */
155  2     END TOW_FLYS;

156  1     IF ((WIRE_BROKE = 1) OR (HILL_IMPACT = 1) OR (GUIDANCE_LOST = 1))

            /* SPECIAL MISS COMMENTS VIA X_MIS_ASCII BUFFER */
           THEN DO;
158  2          IF WIRE_BROKE = 1 THEN CALL MOVB(@WIRE_COMMENT,@X_MIS_ASCII,12);
160  2          IF HILL_IMPACT = 1 THEN CALL MOVB(@HILL_COMMENT,@X_MIS_ASCII,10);
162  2          IF GUIDANCE_LOST = 1 THEN CALL MOVB(@GUIDANCE_COMMENT,@X_MIS_ASCII,15);
164  2          END;

            /* OTHERWISE PRINT MISS DISTANCES */

165  1     ELSE DO;
166  2          IF RIGHT > 0.
               THEN CALL MISS_COMMENT(@RIGHT,@H_MIS_ASCII,MISS_RT);
168  2          IF LEFT > 0.
```

```
                THEN CALL MISS_COMMENT(@LEFT,@H_MIS_ASCII,MISS_LT);
170   2         IF UP <> 0.
                THEN CALL MISS_COMMENT(@UP,@V_MIS_ASCII,MISS_UP);
172   2         END;

                /* PRINT SHORT DISTANCE ONLY IF NO SPECIAL MISSES */

173   1       IF SHORT < 0. AND ABS(SHORT) > 2.6416 THEN
174   1         DROP_SHORT: DO;
175   2           IF (WIRE_BROKE = 0 AND HILL_IMPACT = 0 AND GUIDANCE_LOST = 0) THEN
176   2           CALL MISS_COMMENT(@SHORT,@X_MIS_ASCII,MISS_SH);
177   2          FELL_SHORT = 1;
178   2         END DROP_SHORT;
179   1       ELSE FELL_SHORT = 0;

180   1       CALL EPILOG;              /* VERBAL MISS COMMENTS */

181   1       ACTION_WAIT;              /* WAIT FOR  REPRISE */
                DO FOREVER;
182   2           ACTION = NOT(INPUT(PORT_B)) AND 04H;
183   2           IF ACTION = 4 THEN CALL H_REPRISE;
185   2         END ACTION_WAIT;

186   1       END MAIN_TOW_MODULE;
```

CROSS-REFERENCE LISTING

DEFN	ADDR	SIZE	NAME, ATTRIBUTES, AND REFERENCES
			ABS. . . . . BUILTIN 95 173
41	0013H	1	ACTION . . . . . BYTE 182 183
181	0319H		ACTION_WAIT. . . . LABEL
23	0000H	1	ASCII_BUFFER . . . BYTE BASED(ASCII_PTR) ARRAY(1) 26
20	0006H	4	ASCII_PTR. . . . . POINTER PARAMETER AUTOMATIC 21 23 26
30	000FH	4	ASCII_STRING . . . BYTE ARRAY(4) 35 38
3	A01FH	1	CONTINUE . . . . . BYTE AT ABSOLUTE 126
29	0367H	96	CONVSEND . . . . . PROCEDURE STACK=0014H 106 110
12	0000H	1	CURRENT_TRACK. . . BYTE EXTERNAL(8) 80 82 95
77	03C7H	55	CY512_RESET. . . . PROCEDURE STACK=0008H 104
17	0000H	4	DECADR . . . . . POINTER PARAMETER 18
60	0015H	1	DIR. . . . . BYTE 99 100 108
17	0000H	1	DIRECTION. . . . . BYTE PARAMETER 18
84	03FEH	372	DODGE. . . . . PROCEDURE STACK=0018H 154
174	02CDH		DROP_SHORT . . . . LABEL
67	0000H		EPILOG . . . . . PROCEDURE EXTERNAL(21) STACK=0000H 180
94			EVADE_ENABLED. . . LITERALLY 95
2	A01EH	1	FELL_SHORT . . . . BYTE AT ABSOLUTE 177 179
50	0000H	1	FINISHED . . . . . BYTE EXTERNAL(17) 151
63	0000H		FIRE_AGAIN . . . . PROCEDURE EXTERNAL(19) STACK=0000H 135
73	0000H		FLIGHT . . . . . PROCEDURE EXTERNAL(24) STACK=0000H 153
41			FOREVER. . . . . LITERALLY 181
11	0000H	4	GAEY_F . . . . . REAL EXTERNAL(6) 95
87			GAEY_LIMIT . . . . LITERALLY 95
11	0000H	4	GAEZ_F . . . . . REAL EXTERNAL(7) 95
88			GAEZ_LIMIT . . . . LITERALLY 95
92			GOOD_GUNNER. . . . LITERALLY 95
16	0026H	15	GUIDANCE_COMMENT . BYTE ARRAY(15) DATA 163
13	0000H	1	GUIDANCE_LOST. . . BYTE EXTERNAL(12) 156 162 175
4	A080H	1	GUNNER_RATING. . . BYTE AT ABSOLUTE 95 112
29	0004H	2	HEX. . . . . INTEGER PARAMETER AUTOMATIC 31 32 34 36
			HIGH . . . . . BUILTIN 114 148
120	0017H	1	HIGH_RANDOM. . . . BYTE 95 113 117 148
14	0010H	10	HILL_COMMENT . . . BYTE ARRAY(10) DATA 161
13	0000H	1	HILL_IMPACT. . . . BYTE EXTERNAL(10) 156 160 175
2	A022H	22	H_MIS_ASCII. . . . BYTE ARRAY(22) AT ABSOLUTE 124 167 169
43	0000H		H_REPRISE. . . . . PROCEDURE EXTERNAL(14) STACK=0000H 184
8	000EH	1	I. . . . . BYTE 123 124
			IABS . . . . . BUILTIN 32
75	0000H		INITIATEVAR. . . . PROCEDURE EXTERNAL(25) STACK=0000H 146
			INPUT. . . . . BUILTIN 130 182
			INT. . . . . BUILTIN 97 102 103
53	0006H	2	J. . . . . INTEGER 113 114 128 132 133 136 139 147 148
47	0000H	1	KIND . . . . . BYTE PARAMETER 98
			LAST . . . . . BUILTIN 105 107 109 111
20	0004H	2	LAST_ELEMENT . . . WORD PARAMETER AUTOMATIC 22 25
9	0000H	4	LEFT . . . . . REAL EXTERNAL(2) 168 169
			LOW. . . . . BUILTIN 35 113 147
120	0016H	1	LOW_RANDOM . . . . BYTE 95 97 98 102 103 114 147

```
24  0000H     2  M. . . . . .     WORD        25   26
 1  006CH   714  MAIN_TOW_MODULE.  PROCEDURE STACK=001AH
17  0000H        MISS_COMMENT . . .  PROCEDURE EXTERNAL(13) STACK=0000H      167  169  171  176
40               MISS_LT. . . . . .  LITERALLY       169
40               MISS_RT. . . . . .  LITERALLY       167
40               MISS_SH. . . . . .  LITERALLY       176
40               MISS_UP. . . . . .  LITERALLY       171
                 MOVE . . . . . .  BUILTIN         159  161  163
31  0004H     2  N. . . . . .     INTEGER          33   35
93               NOT_TRACK_3. . . .  LITERALLY       95
41               NO_TRIG_PUL. . . .  LITERALLY       130
85  000CH     2  OLD_TCNT . . . . .  INTEGER          95  116
89               ON_TARGET. . . . .  LITERALLY       95
 5  0000H     1  OUTDATA. . . . . .  BYTE PARAMETER         6
90               PERCENT_FACTOR . .  LITERALLY       95
51               PORT_B . . . . . .  LITERALLY       130  182
71  0000H        PPI_SET. . . . . .  PROCEDURE EXTERNAL(23) STACK=0000H      122
57  0046H     8  PROG_END . . . . .  BYTE ARRAY(8) DATA        111
56  003DH     9  PROG_MID . . . . .  BYTE ARRAY(9) DATA        107
55  0035H     8  PROG_START . . . .  BYTE ARRAY(8) DATA        105
61  0000H        PROLOG . . . . . .  PROCEDURE EXTERNAL(18) STACK=0000H      127
65  0000H        QUIT . . . . . .  PROCEDURE EXTERNAL(20) STACK=0000H      141
59  0008H     2  RATE . . . . . .  INTEGER          97  106
17  0000H     4  REALADR. . . . . .  POINTER PARAMETER          18
31  0002H     2  REMAINDER. . . . .  INTEGER          34   35
 9  0000H     4  RIGHT. . . . . .  REAL EXTERNAL(1)          166  167
54  0014H     1  SECOND_TIME. . . .  BYTE        129  133  137  139
 5  0000H        SEND . . . . . .  PROCEDURE EXTERNAL(0) STACK=0000H       26   78   79   80   81   82
45  0000H        SETRET . . . . . .  PROCEDURE EXTERNAL(15) STACK=0000H      150
                 SHL. . . . . .  BUILTIN         148
 9  0000H     4  SHORT. . . . . .  REAL EXTERNAL(4)          173  176
                 SHR. . . . . .  BUILTIN         102  103
47  0000H        SOUND. . . . . .  PROCEDURE EXTERNAL(16) STACK=0000H      145
42               SOUND_TRIG_PULL. .  LITERALLY       145
119 0089H        START_UP . . . . .  LABEL PUBLIC        122
59  000AH     2  STEPS. . . . . .  INTEGER         102  103  110
20  0336H    49  STRING_OUT . . . .  PROCEDURE STACK=000EH          38  105  107  108  109  111
86               TANK_STOPPED . . .  LITERALLY       95
10  0000H     2  TCNT . . . . . .  INTEGER EXTERNAL(5)        95  101  102  103  116
12  0000H     1  THREE_SEC_FLAG . .  BYTE EXTERNAL(9)
52               THREE_SEC_INTERVAL  LITERALLY       136
58  004EH     3  TID_BIT. . . . . .  BYTE ARRAY(3) DATA        109
                 TIME . . . . . .  BUILTIN         121  131  149
91               TIMES_UP . . . . .  LITERALLY       95
151 0174H        TOW_FLYS . . . . .  LABEL
52               TWO_SEC_INTERVAL .  LITERALLY       128
                 UNSIGN . . . . . .  BUILTIN          35  113  114  147  148
 9  0000H     4  UP . . . . . .  REAL EXTERNAL(3)          170  171
 2  A038H    22  V_MIS_ASCII. . . .  BYTE ARRAY(22) AT ABSOLUTE          171
13  0000H     1  WIRE_BROKE . . . .  BYTE EXTERNAL(11)         156  158  175
15  001AH    12  WIRE_COMMENT . . .  BYTE ARRAY(12) DATA       159
 2  A04EH    22  X_MIS_ASCII. . . .  BYTE ARRAY(22) AT ABSOLUTE          159  161  163  176
69  0000H        YZCNTR . . . . . .  PROCEDURE EXTERNAL(22) STACK=0000H      152

MODULE INFORMATION:
```

```
      CODE AREA SIZE     = 0572H   1394D
      CONSTANT AREA SIZE = 0000H      0D
      VARIABLE AREA SIZE = 0018H     24D
      MAXIMUM STACK SIZE = 001AH     26D
      303 LINES READ
      0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION
```

```
ISIS-II PL/M-86 V2.1 COMPILATION OF MODULE TOWFLIGHTMODULE
OBJECT MODULE PLACED IN :F2:TOWFLP.OBJ
COMPILER INVOKED BY:  PLM86 :F2:TOWFLP.019 DEBUG ROM MEDIUM XREF IXREF WORKFILES(:F2:,:F2:) DATE(1/24/83)

 1          TOW$FLIGHT$MODULE: DO;

            /*****************************************************************************
                                  OFF-BOARD ABSOLUTE ADDRESSES
            *****************************************************************************/

 2   1      DECLARE DISTANCE WORD AT (0A00AH);          /* FOR PIP */
 3   1      DECLARE TARGET_SWITCH BYTE AT (0A00DH);     /* NOT CURRENTLY USED */
 4   1      DECLARE (YANG_BIRD, ZANG_BIRD) INTEGER AT (0A010H), (BIRD_DT_RDY,
                COACH_ON, BIRD_HITS, BIRD_MISSES, H_REF_RQ, H_REF_GO, V_REF_RQ,
                V_REF_GO,GRND_BIRD,END_REFRISE) BYTE AT (0A014H);
 5   1      DECLARE (HIT_KILL,HIT_DISABLE) BYTE AT (0A020H);
 6   1      DECLARE TURNED BYTE AT (0A081H);            /* SIGNALS TANK HAS TURNED */

            /************************ END OFF-BOARD ADDRESSES ****************************/

 7   1      DECLARE (GUIDANCE_LOST,THREE_SEC_FLAG) BYTE PUBLIC;

 8   1      DECLARE COACH_COUNT BYTE;                   /* 1 SEC INTERVAL TIMER FOR COACHING */

                /* VARAIBLES FOR WIRE BREAK */

 9   1      DECLARE (ALPHA,BETA,GAMMA,DELTA,JERK_Y,JERK_Z) REAL;
10   1      DECLARE STORE_HTARG (4) REAL, STORE_GAEY(4) REAL, STORE_GAEZ(4) REAL;
11   1      DECLARE L INTEGER, WIRE_BROKE BYTE PUBLIC;

12   1      DECLARE (SPEECH_HIT,SPEECH_MISS) BYTE EXTERNAL; /* HIT/MISS FLAGS FOR EPILOG */

13   1      DECLARE FINISHED BYTE PUBLIC;

14   1      DECLARE (GAEY_F, GAEZ_F, HTARG, SHORT, OFF_H, Z, X, MISSILE_Z) REAL PUBLIC;

15   1      DECLARE (RIGHT,LEFT,UP) REAL EXTERNAL;

16   1      DECLARE (B_Y,B_Z,DATA_RDY1,HILL_IMPACT,CURRENT_TRACK) BYTE EXTERNAL;

17   1      DECLARE (ACNT, DCNT) INTEGER EXTERNAL;  /* ANGULAR & DEPRESSION MOTOR COUNTS */
                                                    /* NOT CURRENTLY USED */
18   1      DECLARE (GAEY, GAEZ) WORD EXTERNAL;     /* GAE IN HALF PIXELS FROM YZCNTR */

19   1      DECLARE TEMP INTEGER;

20   1      DECLARE COUNT INTEGER PUBLIC;           /* 40 MILLISECOND INCREMENTS */

21   1      DECLARE (MAX_RANGE, VX, INTEGRAL, INTEGRAND, DEL_X) REAL;

22   1      DECLARE (VZ, FTIME) REAL;

23   1      DECLARE (ZETA_H, ZETA_V, OMEGA_H, OMEGA_V) REAL;

24   1      DECLARE (OMEGA_SQ_H, OMEGA_SQ_V, DAMP_H, DAMP_V) REAL; /* MISSILE DAMPING & NATURAL FREQUENCY */
```

```
ISIS-II PL/M-86 V2.1 COMPILATION OF MODULE MAIN_TOW_MODULE
OBJECT MODULE PLACED IN :F2:TOWMN.OBJ
COMPILER INVOKED BY:  PLM86 :F2:TOWMN.020 DEBUG ROM MEDIUM XREF IXREF WORKFILES(:F2:,:F2:) DAT

 1          MAIN_TOW_MODULE: DO;

            /*************************************************************************
                                    OFF-BOARD ADSOLUTE ADDRESSES
            *************************************************************************/

                 /* BUFFERS FOR MISS COMMENTS DURING REPRISE */

 2   1      DECLARE (H_MIS_ASCII,V_MIS_ASCII,X_MIS_ASCII) (22) BYTE AT (0A022H),
                    FELL_SHORT BYTE AT (0A01EH);

 3   1      DECLARE CONTINUE BYTE AT (0A01FH);

 4   1      DECLARE GUNNER_RATING BYTE AT (0A080H);

            /********************** END OFF-BOARD ABSOLUTE ADDRESSES **********************/

                 /* SEND TRANSMITS A BYTE TO UPI-41 ON THE STEPPER MOTOR CONTROLLER
                    BOARD (SMCB) */

 5   1      SEND: PROCEDURE (OUTDATA) EXTERNAL;
 6   2           DECLARE OUTDATA BYTE;
 7   2           END SEND;

 8   1      DECLARE I BYTE;

 9   1      DECLARE (RIGHT,LEFT,UP,SHORT) REAL EXTERNAL;  /* MISS DISTANCES */

10   1      DECLARE TCNT INTEGER EXTERNAL;                /* TARGET POSITION COUNTER */

11   1      DECLARE (GAEY_F, GAEZ_F) REAL EXTERNAL;       /* AIMING ERRORS */

12   1      DECLARE (CURRENT_TRACK, THREE_SEC_FLAG) BYTE EXTERNAL;

13   1      DECLARE (HILL_IMPACT, WIRE_BROKE, GUIDANCE_LOST) BYTE EXTERNAL;

14   1      DECLARE HILL_COMMENT(10) BYTE DATA('- HIT HILL');

15   1      DECLARE WIRE_COMMENT(12) BYTE DATA('- WIRE BROKE');

16   1      DECLARE GUIDANCE_COMMENT(15) BYTE DATA('- GUIDANCE_LOST');

17   1      MISS_COMMENT: PROCEDURE(REAL$ADR, DEC$ADR, DIRECTION) EXTERNAL;
18   2           DECLARE (REAL$ADR, DEC$ADR) POINTER, DIRECTION BYTE;
19   2       END MISS_COMMENT;

                 /* STRING_OUT SENDS A STRING OF INSTRUCTIONS TO THE SMCB USING
                    THE SEND PROCEDURE GIVEN A POINTER TO THE STRING AND THE
                    SUBSCRIPT OF THE LAST ELEMENT */

20   1      STRING_OUT: PROCEDURE(ASCII_PTR,LAST_ELEMENT);
```

```
25  1     DECLARE (KOEF, COEF1, COEF2) REAL;      /* FLIGHT DYNAMIC COEFFICIENT */

26  1     DECLARE (SUMG, INTEGRAL1, INTEGRAND1, ERR_COMP2, ERR_COMP3, ERR_COMP1, XTARG) REAL;

27  1     DECLARE (VTARG, HANG, VANG, DEL$VZ) REAL; /* HANG,VANG GIVE MISSILE POSITION RELATIVE TO AIM
        -   DIANS */

28  1     DECLARE EXPONENTIAL REAL;

29  1     DECLARE RESULTS(800) STRUCTURE(S_X WORD,S_Y REAL,S_Z REAL,
          S_GAEY REAL, S_GAEZ REAL) AT (2000H), I INTEGER;

30  1     DECLARE GROUND_EXP  LITERALLY '2',     /* PARAMETERS FOR SOUND */
              HIT_TARGET  LITERALLY '1',
              FACTOR      LITERALLY '0.835'; /* CORRECTION FACTOR FOR LEAD ANGLE CONSTANT */

31  1     DECLARE INIT_COACH_COUNT LITERALLY '25';  /* COACH EVERY SEC IF ENABLED */

            /* 8087.LIB PROCEDURES */

32  1     INIT87: PROCEDURE EXTERNAL;
33  2           END INIT87;
34  1     ■qerEXP: PROCEDURE (X) REAL EXTERNAL;
35  2           DECLARE X REAL;
36  2           END ■qerEXP;

            /* COACH USES GAE TO ISSUE COACHING COMMANDS IF AIMING ERROR
               EXCESSIVE */

37  1     COACH: PROCEDURE (GAEH, GAEV) EXTERNAL;
38  2           DECLARE (GAEH,GAEV) REAL;
39  2           END COACH;

40  1     SOUND: PROCEDURE (WHAT_KIND) EXTERNAL;
41  2           DECLARE WHAT_KIND BYTE;
42  2           END SOUND;

            /* TARGET DATA OBTAINS HTARG, TARGET_Y, TARGET_Z, TCNT AND
               EFFECTIVE TANK DIMENSIONS */

43  1     TARGET_DATA: PROCEDURE EXTERNAL;
44  2                  END TARGET_DATA;

            /* ACTIVE TRACK OBTAINS THE NUMBER OF THE TARGET TRACK */

45  1     ACTIVE_TRACK: PROCEDURE EXTERNAL;
46  2                   END ACTIVE_TRACK;

            /* UPDATE COUNTS IS USED TO OBTAIN THE INITIAL TARGET COUNT */

47  1     UPDATE_COUNTS: PROCEDURE (FIRST_PASS) EXTERNAL;
48  2                    DECLARE FIRST_PASS BYTE;
49  2                    END UPDATE_COUNTS;

            /* PROCEDURES THAT DETERMINE HIT OR MISS, ONE FOR EACH TRACK */
```

```
50  1     TRACK_1: PROCEDURE EXTERNAL;
51  2              END TRACK_1;
52  1     TRACK_2: PROCEDURE EXTERNAL;
53  2              END TRACK_2;
54  1     TRACK_3: PROCEDURE EXTERNAL;
55  2              END TRACK_3;

                 /* GROUNDED CHECKS FOR MISSILE GROUND IMPACT */

56  1     GROUNDED: PROCEDURE EXTERNAL;
57  2               END GROUNDED;

58  1     H_REPRISE: PROCEDURE PUBLIC;
59  2                H_REP_RQ = 1;            /* SIGNALS PIP TO SET-UP FOR H_REPRISE */
60  2                DO WHILE NOT H_REP_GO;   /* WAIT FOR PIP TO COMPLETE SET-UP */
61  3                END;
62  2                H_REP_GO = 0;
63  2                DO I = 1 TO COUNT;
64  3                  CALL TIME(380);
65  3                  TEMP = FIX(100.0 + RESULTS(I).S_Z/0.1);
66  3                  IF TEMP < 0
                       THEN B_Z = 0;
68  3                  ELSE B_Z = LOW(UNSIGN(TEMP));

                 /* PASS VERTICAL DISTANCE OF MISSILE FROM TARGET LINE TO PIP BOARD
                    IN DECIMETER INCREMENTS.                                      */

69  3                  TEMP = FIX(100.0 - RESULTS(I).S_Y/0.1);
70  3                  IF TEMP < 0
                       THEN B_Y = 0;
72  3                  ELSE B_Y = LOW(UNSIGN(TEMP));

                 /* PASS HORIZONTAL DISTANCE OF MISSILE FROM TARGET LINE TO PIP BOARD
                    IN DECIMETER INCREMENTS.                                      */

73  3                  DISTANCE = RESULTS(I).S_X;

74  3                  DATA_RDY1 = 1;
75  3                  BIRD_DT_RDY = 1;
76  3                  END;
77  2                CALL TIME(380);
78  2                END_REPRISE = 1;
79  2                END H_REPRISE;
               /*****************************************************************
                         PROGRAM VARIABLE INITIALIZATION
                *****************************************************************/
80  1     INITIATE$VAR: PROCEDURE PUBLIC;

81  2         CALL INIT87;
82  2         COACH_COUNT = INIT_COACH_COUNT;
83  2         TARGET_SWITCH = 0;            /* TEMPORARILY DISABLE TARGET SWITCHING */
84  2         CALL ACTIVE_TRACK;            /* GET CURRENT TRACK */
85  2         CALL UPDATE_COUNTS(1);        /* GET INITIAL COUNTS */
86  2         IF TURNED = 1
```

```
                THEN ACNT = 24;                 /* 90 DEGREE TURN */
 88   2         ELSE ACNT = 0;                  /* NO TURN */
 89   2         DCNT = 550;                     /* ZERO DEPRESSION COUNT */
 90   2         VZ = 10.160;
 91   2         DO L = 0 TO 3;                  /* ZERO WIRE BREAK STACKS */
 92   3            STORE_HTARG(L), STORE_GAEY(L), STORE_GAEZ(L) = 0.;
 93   3            END;
 94   2         WIRE_BROKE = 0;
 95   2         MISSILE_Z, X, Z, VTARG, VANG, HANG, OFF_H = 0.;
 96   2         GAEY_F, GAEZ_F, INTEGRAL2, SUMG, INTEGRAL1 = 0.;
 97   2         RIGHT,LEFT,UP,SHORT = 0.;
 98   2         GUIDANCE_LOST,THREE_SEC_FLAG = 0;

 99   2         XTARG = 3000.0;
100   2         MAX_RANGE = 3500.0;       /* RANGE FOR FINAL GROUND IMPACT */
101   2         SPEECH_HIT, SPEECH_MISS = 0;
102   2         HILL_IMPACT,FINISHED,BIRD_HITS,BIRD_MISSES,GRND_BIRD = 0;
103   2         H_REP_RQ,H_REP_GO,V_REP_RQ,V_REP_GO,END_REPRISE = 0;
104   2         HIT_KILL, HIT_DISABLE = 0;
105   2         I = 0;
106   2         COUNT = 0;
107   2       END INITIATE$VAR;
                /*****************************************************************************
                                        MFS MAIN PROCEDURE
                *****************************************************************************/
108   1       FLIGHT: PROCEDURE PUBLIC;

109   2       COUNT = COUNT + 1;
110   2       I=I+1;
111   2       IF COUNT >= 75 THEN THREE_SEC_FLAG = 1;

                /*****************************************************************************
                                GET TARGET POSITION & GUNNER AIMING ERROR
                *****************************************************************************/

                /*  TARGET POSITION IS MEASURED FROM THE INITIAL TARGET LOCATION  */

113   2       GAEY_F = 5.906E-5 * FLOAT(SIGNED(GAEY)); /* .6 CM DETECTOR HEIGHT DIVIDED BY */
114   2       GAEZ_F = 5.906E-5 * FLOAT(SIGNED(GAEZ)); /* (200 HALF PIXELS * 76.2 CM FL) */

115   2       CALL ACTIVE_TRACK;              /* GET CURRENT TRACK */
116   2       CALL TARGET_DATA;               /* GET TARGET PARAMETERS & CORRECT RAW GUNNER AIMING ERROR */

                /*****************************************************************************
                                    TRANSFER BIRD POSITIONS TO 'PIP'
                *****************************************************************************/

117   2       ZANG_BIRD = FIX(VANG/0.0001); /* ANGULAR POSITION OF MISSILE RELATIVE */
118   2       YANG_BIRD = FIX(HANG/0.0001); /* TO AIMING AXIS IN HALF PIXELS */
119   2       BIRD_DT_RDY = 1;

                /*****************************************************************************
                                        CHECK IF WIRE BROKE
                *****************************************************************************/
```

```
120  2      ALPHA = STORE_GAEY(3) - GAEY_F;          /* CHANGE IN AIMING ANGLE */
121  2      GAMMA = STORE_HTARG(3) - HTARG;          /* CHANGE IN TARGET ANGLE */
122  2      DELTA = ALPHA - GAMMA;
123  2      BETA = STORE_GAEZ(3) - GAEZ_F;
124  2      JERK_Y = (DELTA * DELTA) * 25.0 / 4.0;
125  2      JERK_Z = (BETA * BETA) * 25.0 / 4.0;

             /* A JERK IS A RATE OF 1 MRAD. IN 1/25 SEC. FOR AN ANGLE OF 2.5 MRAD. */

126  2      IF (COUNT > 5) AND ((JERK_Y > 62.5E-6) OR (JERK_Z > 62.5E-6))
             THEN DO;
128  3          FINISHED,WIRE_BROKE,GRND_BIRD = 1;
129  3          CALL SOUND(GROUND_EXP);
130  3          END;
131  2      ELSE DO;
132  3          DO L = 3 TO 1 BY -1;
133  4           STORE_GAEY(L) = STORE_GAEY(L - 1);
134  4           STORE_GAEZ(L) = STORE_GAEZ(L - 1);
135  4           STORE_HTARG(L) = STORE_HTARG(L - 1);
136  4           END;
137  3          STORE_GAEY(0) = GAEY_F;
138  3          STORE_GAEZ(0) = GAEZ_F;
139  3          STORE_HTARG(0) = HTARG;
140  3          END;
             /************************** COACHING **************************/

141  2      IF COACH_ON = 1
             THEN DO;
143  3          COACH_COUNT = COACH_COUNT - 1;
144  3          IF COACH_COUNT = 0
             THEN DO;
146  4              CALL COACH(GAEY_F,GAEZ_F);
147  4              COACH_COUNT = INIT_COACH_COUNT;
148  4              END;
149  3          END;
             /*************************************************************************
                                  CHECK FOR POSIBLE HIT
             *************************************************************************/

150  2      SHORT = X - XTARG;
151  2      IF ABS(SHORT) < 4.9 THEN        /* + OR - 8 FT FROM RANGE OF TARGET */
152  2       AT_TARGET: DO;
153  3            DO CASE CURRENT_TRACK;
154  4                        ;                    /* NULL STATEMENT */
155  4                        CALL TRACK_1;
156  4                        CALL TRACK_2;
157  4                        CALL TRACK_3;
158  4                        END;
159  3         FINISHED = 1;
160  3            IF SPEECH_HIT = 1
             THEN DO;
162  4                 GRND_BIRD = 1;
163  4                 CALL SOUND(GROUND_EXP);
164  4                 END;
165  3         END AT_TARGET;
```

```
                /*****************************************************************************
                     CALCULATE MISSILE DAMPING, NATURAL FREQUENCY, AND VELOCITY.
                *****************************************************************************/
                  /*************************************************************
                       SEE TOW REPORT APPENDIX FOR THE DEFINITION OF THE
                       FOLLOWING VARIABLES AND EQUATIONS.
                   *************************************************************/
166   2       EXPONENTIAL = @qerEXP(-X/1000.0);

167   2       ZETA_H = (0.24 * EXPONENTIAL + 0.40)/4.0;   /* QUICK FIX 11/09/82 */
168   2       ZETA_V = (0.33 * EXPONENTIAL + 0.40)/4.0;

169   2       EXPONENTIAL = 6.4 * @qerEXP(-X/1500.0);

170   2       OMEGA_H = EXPONENTIAL + 2.60;
171   2       OMEGA_V = EXPONENTIAL + 2.00;

172   2       OMEGA_SQ_H = OMEGA_H * OMEGA_H;
173   2       OMEGA_SQ_V = OMEGA_V * OMEGA_V;
174   2       DAMP_H = 2.0 * ZETA_H * OMEGA_H;
175   2       DAMP_V = 2.0 * ZETA_V * OMEGA_V;
176   2       KOEF = FACTOR * (2.0 * DAMP_H / OMEGA_SQ_H);
177   2       COEF1 = OMEGA_SQ_H * KOEF;
178   2       COEF2 = COEF1 - DAMP_H;

179   2       FTIME = 0.04 * FLOAT(COUNT);        /* RETICON FREQUENCY NOW 25 HERTZ */

180   2       VX = 165.10 * (1.0 - @qerEXP(-2.0*FTIME) + @qerEXP(-FTIME/5.5));

                /*****************************************************************************
                                        FIND MISSILE POSITION
                 *****************************************************************************/

181   2       DEL_X = VX/25.0;
182   2       X = X + DEL_X;
183   2       DISTANCE = UNSIGN(FIX(X));


184   2       DEL$VZ = (OMEGA_SQ_V*GAEZ_F*X - DAMP_V*VZ - OMEGA_SQ_V*Z)/25.0;
185   2       VZ = VZ + DEL$VZ/2.0;
186   2       Z = Z + VZ/25.0;
187   2       MISSILE_Z = Z + 1.25;
188   2       VZ = VZ + DEL$VZ/2.0;

189   2       ERR_COMP1 = DAMP_H*OFF_H;
190   2       ERR_COMP2 = (HTARG*COEF2 + COEF1*GAEY_F)*X;
191   2       INTEGRAND1 = (GAEY_F*X-KOEF*(HTARG + GAEY_F)*VX+OFF_H)/25.0;
192   2       INTEGRAL1 = INTEGRAL1 + INTEGRAND1;
193   2       ERR_COMP3 = OMEGA_SQ_H*INTEGRAL1;
194   2       INTEGRAND2 = (ERR_COMP1 + ERR_COMP2 + ERR_COMP3)/25.0;
195   2       INTEGRAL2 = INTEGRAL2 + INTEGRAND2;
196   2       OFF_H = HTARG*X - INTEGRAL2;

                /*****************************************************************************
```

```
                                        DEFINING ANGLES
                    ***************************************************************************/

197   2       IF COUNT > 0 THEN  /* TO AVOID A 'DIVIDE BY ZERO' WHEN CALCULATING 'HMISS.' */
198   2       ANGLES: DO;
199   3         HANG = -OFF_H/X - GAEY_F;
200   3         VANG = Z/X - VTARG - GAEZ_F;
201   3          END ANGLES;

                    /***************************************************************************
                                              CRASHED?
                    ***************************************************************************/

202   2       CALL GROUNDED;

                    /***************************************************************************
                                           OUT OF FOV?
                     ***************************************************************************/

203   2       IF (THREE_SEC_FLAG = 1) AND ((ABS(GAEY_F) > 5.0E-3) OR (ABS(GAEZ_F) > 5.0E-3))
                THEN DO;
205   3           FINISHED,GRND_BIRD,GUIDANCE_LOST = 1;  /* EXACT NUMBER IS 5.906E-3 */
206   3           CALL SOUND(GROUND_EXP);
207   3           END;

                    /***************************************************************************
                              MISSILE DATA SAMPLE FOR POSSIBILE REPRISE
                    ***************************************************************************/

208   2       RESULTS(I).S_X = DISTANCE;
209   2       RESULTS(I).S_Y = - OFF_H;
210   2       IF OFF_H > 12.7
                 THEN RESULTS(I).S_Y = -12.7;
212   2       IF OFF_H < 0. AND ABS(OFF_H) > 12.7
                 THEN RESULTS(I).S_Y = 12.7;
214   2       RESULTS(I).S_Z = Z;
215   2       IF Z > 15.0
                 THEN RESULTS(I).S_Z = 15.0;
217   2       IF Z < 0. AND ABS(Z) > 5.00
                 THEN RESULTS(I).S_Z = -5.00;

219   2       RESULTS(I).S_GAEY=GAEY_F;
220   2       RESULTS(I).S_GAEZ=GAEZ_F;

221   2       END FLIGHT;

222   1       END TOW$FLIGHT$MODULE;
```

CROSS-REFERENCE LISTING

DEFN	ADDR	SIZE	NAME, ATTRIBUTES, AND REFERENCES
			ABS. . . . . BUILTIN 151 203 212 217
17	0000H	2	ACNT . . . . . INTEGER EXTERNAL(10) 87 88
45	0000H		ACTIVE_TRACK . . . PROCEDURE EXTERNAL(19) STACK=0000H 84 115
9	0000H	4	ALPHA. . . . . REAL 120 122
198	08DFH		ANGLES . . . . . LABEL
152	059EH		AT_TARGET. . . . . LABEL
9	0004H	4	BETA . . . . . REAL 123 125
4	A014H	1	BIRD_DT_RDY. . . . BYTE AT ABSOLUTE 75 119
4	A016H	1	BIRD_HITS. . . . . BYTE AT ABSOLUTE 102
4	A017H	1	BIRD_MISSES. . . . BYTE AT ABSOLUTE 102
16	0000H	1	B_Y. . . . . BYTE EXTERNAL(5) 71 72
16	0000H	1	B_Z. . . . . BYTE EXTERNAL(6) 67 68
37	0000H		COACH. . . . . PROCEDURE EXTERNAL(16) STACK=0000H 146
8	00EAH	1	COACH_COUNT. . . . BYTE 82 143 144 147
4	A015H	1	COACH_ON . . . . . BYTE AT ABSOLUTE 141
25	00AEH	4	COEF1. . . . . REAL 177 178 190
25	00B2H	4	COEF2. . . . . REAL 178 190
20	006CH	2	COUNT. . . . . INTEGER PUBLIC 63 106 109 111 126 179 197
16	0000H	1	CURRENT_TRACK. . . BYTE EXTERNAL(9) 153
24	00A2H	4	DAMP_H . . . . . REAL 174 176 178 189
24	00A6H	4	DAMP_V . . . . . REAL 175 184
16	0000H	1	DATA_RDY1. . . . . BYTE EXTERNAL(7) 74
17	0000H	2	DCNT . . . . . INTEGER EXTERNAL(11) 89
9	000CH	4	DELTA. . . . . REAL 122 124
27	00DEH	4	DELVZ. . . . . REAL 184 185 188
21	007EH	4	DEL_X. . . . . REAL 181 182
2	A00AH	2	DISTANCE . . . . . WORD AT ABSOLUTE 73 183 208
4	A01DH	1	END_REPRISE. . . . BYTE AT ABSOLUTE 78 103
26	00CAH	4	ERR_COMP1. . . . . REAL 189 194
26	00C2H	4	ERR_COMP2. . . . . REAL 190 194
26	00C6H	4	ERR_COMP3. . . . . REAL 193 194
28	00E2H	4	EXPONENTIAL. . . . REAL 166 167 168 169 170 171
30			FACTOR . . . . . LITERALLY 176
13	00ECH	1	FINISHED . . . . . BYTE PUBLIC 102 128 159 205
47	0000H	1	FIRST_PASS . . . . BYTE PARAMETER 48
			FIX. . . . . BUILTIN 65 69 117 118 183
108	0375H	1955	FLIGHT . . . . . PROCEDURE PUBLIC STACK=0008H
			FLOAT. . . . . BUILTIN 113 114 179
22	0086H	4	FTIME. . . . . REAL 179 180
37	0000H	4	GAEH . . . . . REAL PARAMETER 38
37	0000H	4	GAEV . . . . . REAL PARAMETER 38
18	0000H	2	GAEY . . . . . WORD EXTERNAL(12) 113
14	004AH	4	GAEY_F . . . . . REAL PUBLIC 96 113 120 137 146 190 191 199 203 219
18	0000H	2	GAEZ . . . . . WORD EXTERNAL(13) 114
14	004EH	4	GAEZ_F . . . . . REAL PUBLIC 96 114 123 138 146 184 200 203 220
9	0008H	4	GAMMA. . . . . REAL 121 122
4	A01CH	1	GRND_BIRD. . . . . BYTE AT ABSOLUTE 102 128 162 205
56	0000H		GROUNDED . . . . . PROCEDURE EXTERNAL(24) STACK=0000H 202
30			GROUND_EXP . . . . LITERALLY 129 163 206

```
 7  00E8H     1  GUIDANCE_LOST. . .   BYTE PUBLIC          98  205
27  00D6H     4  HANG . . . . . .    REAL        95  118  199
16  0000H     1  HILL_IMPACT. . . .  BYTE EXTERNAL(8)          102
 5  A021H     1  HIT_DISABLE. . . .  BYTE AT ABSOLUTE          104
 5  A020H     1  HIT_KILL . . . . .  BYTE AT ABSOLUTE          104
30                HIT_TARGET . . . .  LITERALLY
14  0052H     4  HTARG. . . . . .    REAL PUBLIC         121  139  190  191  196
58  00C0H   290  H_REPRISE. . . . .  PROCEDURE PUBLIC STACK=0004H
 4  A019H     1  H_REP_GO . . . . .  BYTE AT ABSOLUTE          60   62  103
 4  A018H     1  H_REP_RQ . . . . .  BYTE AT ABSOLUTE          59  103
29  00E6H     2  I. . . . . .        INTEGER         63   65   69   73  105  110  208  209  211  213  214
                                      216  218  219  220
32  0000H       INIT87 . . . . . .  PROCEDURE EXTERNAL(14) STACK=0000H       81
80  01E2H   403  INITIATEVAR. . . .  PROCEDURE PUBLIC STACK=0008H
31                INIT_COACH_COUNT .  LITERALLY       82  147
26  00BAH     4  INTEGRAL1. . . . .  REAL        96  192  193
21  0076H     4  INTEGRAL2. . . . .  REAL        96  195  196
26  00BEH     4  INTEGRAND1 . . . .  REAL       191  192
21  007AH     4  INTEGRAND2 . . . .  REAL       194  195
 9  0010H     4  JERK_Y . . . . . .  REAL       124  126
 9  0014H     4  JERK_Z . . . . . .  REAL       125  126
25  00AAH     4  KDEF . . . . . .    REAL       176  177  191
11  0048H     2  L. . . . . .        INTEGER         91   92  132  133  134  135
15  0000H     4  LEFT . . . . . .    REAL EXTERNAL(3)          97
                 LOW. . . . . .      BUILTIN         68   72
21  006EH     4  MAX_RANGE. . . . .  REAL       100
14  0066H     4  MISSILE_Z. . . . .  REAL PUBLIC          95  187
34  0000H       MQEREXP. . . . . .  PROCEDURE REAL EXTERNAL(15) STACK=0000H       166  169  180
14  005AH     4  OFF_H. . . . . .    REAL PUBLIC          95  189  191  196  199  209  210  212
23  0092H     4  OMEGA_H. . . . . .  REAL       170  172  174
24  009AH     4  OMEGA_SQ_H . . . .  REAL       172  176  177  193
24  009EH     4  OMEGA_SQ_V . . . .  REAL       173  184
23  0096H     4  OMEGA_V. . . . . .  REAL       171  173  175
29  2000H 14400  RESULTS. . . . . .  STRUCTURE ARRAY(800) AT ABSOLUTE          65   69   73  208  209  211
                                      213  214  216  218  219  220
15  0000H     4  RIGHT. . . . . .    REAL EXTERNAL(2)          97
14  0056H     4  SHORT. . . . . .    REAL PUBLIC          97  150  151
                 SIGNED . . . . . .  BUILTIN        113  114
40  0000H       SOUND. . . . . .    PROCEDURE EXTERNAL(17) STACK=0000H       129  163  206
12  0000H     1  SPEECH_HIT . . . .  BYTE EXTERNAL(0)          101  160
12  0000H     1  SPEECH_MISS. . . .  BYTE EXTERNAL(1)          101
10  0028H    16  STORE_GAEY . . . .  REAL ARRAY(4)         92  120  133  137
10  0038H    16  STORE_GAEZ . . . .  REAL ARRAY(4)         92  123  134  138
10  0018H    16  STORE_HTARG. . . .  REAL ARRAY(4)         92  121  135  139
26  00B6H     4  SUMQ . . . . . .    REAL        96
29  000AH     4  S_GAEY . . . . . .  REAL MEMBER(RESULTS)      219
29  000EH     4  S_GAEZ . . . . . .  REAL MEMBER(RESULTS)      220
29  0000H     2  S_X. . . . . .      WORD MEMBER(RESULTS)       73  208
29  0002H     4  S_Y. . . . . .      REAL MEMBER(RESULTS)       69  209  211  213
29  0006H     4  S_Z. . . . . .      REAL MEMBER(RESULTS)       65  214  216  218
43  0000H       TARGET_DATA. . . .  PROCEDURE EXTERNAL(18) STACK=0000H       116
 3  A00DH     1  TARGET_SWITCH. . .  BYTE AT ABSOLUTE          83
19  006AH     2  TEMP . . . . . .    INTEGER         65   66   68   69   70   72
 7  00E9H     1  THREE_SEC_FLAG . .  BYTE PUBLIC          98  112  203
                 TIME . . . . . .    BUILTIN         64   77
 1  00C0H       TOWFLIGHTMODULE. .  PROCEDURE STACK=0000H
50  0000H       TRACK_1. . . . . .  PROCEDURE EXTERNAL(21) STACK=0000H       155
```

```
 52  0000H        TRACK_2. . . . . .   PROCEDURE EXTERNAL(22) STACK=0000H       156
 54  0000H        TRACK_3. . . . . .   PROCEDURE EXTERNAL(23) STACK=0000H       157
  6  A081H     1  TURNED . . . . . .   BYTE AT ABSOLUTE          86
                  UNSIGN . . . . . .   BUILTIN          68   72  183
 15  0000H     4  UP . . . . . . . .   REAL EXTERNAL(4)         97
 47  0000H        UPDATE_COUNTS. . .   PROCEDURE EXTERNAL(20) STACK=0000H       85
 27  00DAH     4  VANG . . . . . . .   REAL         95  117  200
 27  00D2H     4  VTARG. . . . . . .   REAL         95  200
 21  0072H     4  VX . . . . . . . .   REAL        180  181  191
 22  0082H     4  VZ . . . . . . . .   REAL         90  184  185  186  188
  4  A01BH     1  V_REP_GO . . . . .   BYTE AT ABSOLUTE         103
  4  A01AH     1  V_REP_RQ . . . . .   BYTE AT ABSOLUTE         103
 40  0000H     1  WHAT_KIND. . . . .   BYTE PARAMETER       41
 11  00E8H     1  WIRE_BROKE . . . .   BYTE PUBLIC          94  128
 14  0062H     4  X. . . . . . . . .   REAL PUBLIC          95  150  166  169  182  183  184  190  191  196
                                        199  200
 34  0000H     4  X. . . . . . . . .   REAL PARAMETER       35
 26  00CEH     4  XTARG. . . . . . .   REAL         99  150
  4  A010H     2  YANG_BIRD. . . . .   INTEGER AT ABSOLUTE      118
 14  005EH     4  Z. . . . . . . . .   REAL PUBLIC          95  184  186  187  200  214  215  217
  4  A012H     2  ZANG_BIRD. . . . .   INTEGER AT ABSOLUTE      117
 23  008AH     4  ZETA_H . . . . . .   REAL        167  174
 23  008EH     4  ZETA_V . . . . . .   REAL        168  175
```

```
MODULE INFORMATION:

     CODE AREA SIZE     = 0818H   2840D
     CONSTANT AREA SIZE = 0000H      0D
     VARIABLE AREA SIZE = 00EDH    237D
     MAXIMUM STACK SIZE = 0008H      8D
     382 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION
```

```
ISIS-II PL/M-86 V2.1 COMPILATION OF MODULE TOW_TARGET_MODULE
OBJECT MODULE PLACED IN :F2:TOWTAR.OBJ
COMPILER INVOKED BY:  PLM86 :F2:TOWTAR.005 DEBUG ROM MEDIUM XREF IXREF WORKFILES(:F2:,:F2:) DATE(1/13/83)

   1          TOW_TARGET_MODULE: DO:

              /*****************************************************************
                              OFF-BOARD ABSOLUTE ADDRESSES
               *****************************************************************/

   2   1      DECLARE (BIRD_HITS, BIRD_MISSES) BYTE AT (0A016H);
   3   1      DECLARE GRND_BIRD BYTE AT (0A01CH);
   4   1      DECLARE (STARTING_TRACK, TARGET_SWITCH, FINAL_TRACK) BYTE AT (0A00CH);
   5   1      DECLARE (HIT_KILL,HIT_DISABLE) BYTE AT (0A020H);

              /********************* GLOBAL VARIABLES **********************/

   6   1      DECLARE (SPEECH_HIT, SPEECH_MISS) BYTE EXTERNAL;

   7   1      DECLARE PORT_C LITERALLY '0CCH';       /* 8255 PARALLEL PORT C */

   8   1      DECLARE FINISHED BYTE EXTERNAL;

                   /* MOTOR COUNTS, TCOUNTO IS INITIAL TARGET COUNT */

   9   1      DECLARE (TCNT, ACNT, DCNT, TCOUNTO) INTEGER PUBLIC;

  10   1      DECLARE (RIGHT,LEFT,UP) REAL PUBLIC;

  11   1      DECLARE (HILL_IMPACT, CURRENT_TRACK) BYTE PUBLIC;

  12   1      DECLARE (GAEZ_F, X, Z, MISSILE_Z, HTARG, OFF_H) REAL EXTERNAL;

              /******************** EXCLUSIVE VARIABLES ********************/

                  /* ALPHA IS ROTATION OF TANK IN RADIANS, DEPRESSION DEPTH IS
                     DEPTH OF FRONT TANK IN METERS */

  13   1      DECLARE (ALPHA,DEPRESSION_DEPTH) REAL;

  14   1      DECLARE (MISSILE_Y,TARGET_Y, EFFECTIVE_TANK_LENGTH, EFF_HALF_TANK_LENGTH) REAL;

  15   1      DECLARE (HILL_Z, TARGET_Z, BETA, MAGNITUDE, ANGLE) REAL;

  16   1      DECLARE (TEMP, EFF_HALF_TANK_HEIGHT, MISSILE_Y_PRIME, MISSILE_Z_PRIME) REAL;

  17   1      DECLARE I BYTE;

                  /* TABLE OF HILL SLOPE AND HEIGHT AS A FUNCTION OF TARGET COUNT */

  18   1      DECLARE MODEL_COUNT(32) INTEGER DATA
                       (104,242,380,518,656,794,932,1070,1208,1346,1485,
                        1623,1761,1899,2037,2175,2313,2451,2589,2727,2866,
                        3004,3142,3280,3418,3556,3694,3832,3970,4108,4247,
                        4385);
```

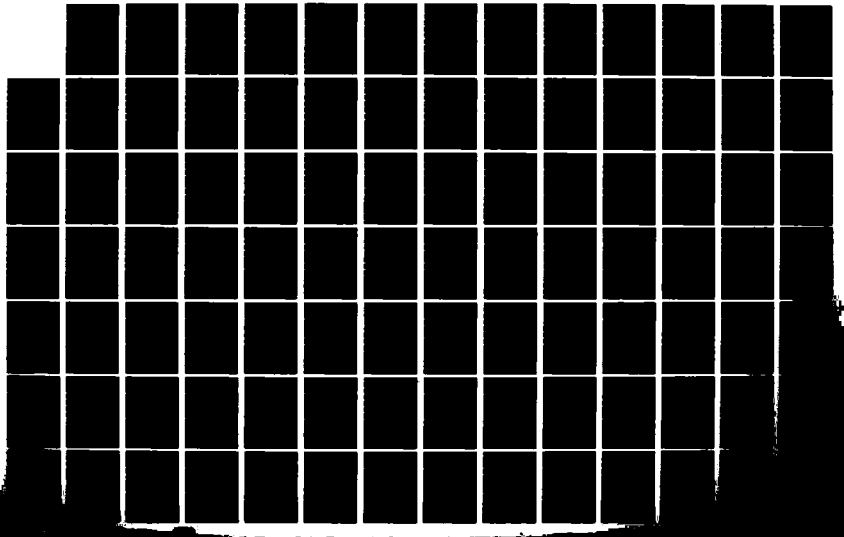
```
19  1     DECLARE HILL_SLOPE(32) REAL DATA
                  (0.,0.1571,0.,-0.1309,0.,0.0698,0.1396,0.2182,0.2531,
                   0.2094,-0.0349,-0.2182,-0.3491,-0.0785,0.1222,0.1134,
                   -0.2443,-0.2531,-0.0524,-0.0175,0.0436,0.1047,0.1920,
                   0.0785,0.1745,0.2880,0.1396,-0.1134,-0.3491,-0.2880,
                   -0.2268,0.);
20  1     DECLARE HILL_HEIGHT(32) REAL DATA
                  (0.,0.2898,1.2318,0.9420,0.,0.0725,0.7971,2.1013,
                   3.8404,7.3909,8.1155,7.3185,5.2896,3.4781,3.4781,
                   4.3476,3.7679,1.5217,0.2898,0.1449,0.2174,0.9420,
                   1.9564,3.1158,3.8404,5.5070,7.5358,7.6808,5.7243,
                   3.2607,1.1594,0.);

                     /* ALL DIMENSIONS IN METERS */

21  1     DECLARE TANK_LENGTH LITERALLY '8.3',
                  TANK_WIDTH  LITERALLY '3.9',
                  TANK_HEIGHT LITERALLY '2.9',
                  KILL_BOX    LITERALLY '1.45',
                  HALF_TANK_HEIGHT LITERALLY '1.45',
                  LAUNCH_HEIGHT LITERALLY '0.9144';

                     /* PARAMETERS FOR SOUND */

22  1     DECLARE HIT_TARGET LITERALLY '1',
                  GROUND_EXP LITERALLY '2';

                     /* TYPE OF HIT CONTROLS REPRISE COMMENT:

                           REGULAR     => 'HIT'
                           HIT KILL    => 'HIT KILL'
                           HIT DISABLE => 'HIT DISABLE'   */

23  1     DECLARE REGULAR LITERALLY '0',
                  DISABLE_ LITERALLY '1',
                  KILL LITERALLY '2';

                  /********************* EXTERNAL PROCEDURES ************************/

                     /* 8087 PROCEDURES IN CEL87.LIB */

24  1     $qerAT2: PROCEDURE (Y,X) REAL EXTERNAL;
25  2              DECLARE (Y,X) REAL;
26  2              END $qerAT2;
27  1     $qerCOS: PROCEDURE (THETA) REAL EXTERNAL;
28  2              DECLARE THETA REAL;
29  2              END $qerCOS;
30  1     $qerSIN: PROCEDURE (THETA) REAL EXTERNAL;
31  2              DECLARE THETA REAL;
32  2              END $qerSIN;
33  1     $qerY2X: PROCEDURE (Y,X) REAL EXTERNAL;
34  2              DECLARE (Y,X) REAL;
35  2              END $qerY2X;

36  1     SOUND: PROCEDURE (WHAT_KIND) EXTERNAL;
37  2            DECLARE WHAT_KIND BYTE;
38  2            END SOUND;
```

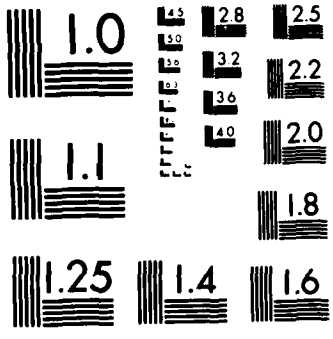
```
                /********** MODULE PROCEDURES **********/

                /* RECEIVE GETS A BYTE FROM THE UPI-41 MOTOR CONTROLLER */

39   1          RECEIVE: PROCEDURE BYTE PUBLIC;
40   2                   DECLARE STATUS BYTE AT (0F000H), INDATA BYTE AT (0F002H);
41   2                   DO WHILE STATUS;              /* WAIT UNTIL OBF = 1 */
42   3                      END;
43   2                   RETURN NOT(INDATA);           /* MULTIBUS INVERTS */
44   2                   END RECEIVE;

                /* SEND OUTPUTS A BYTE TO THE UPI-41 MOTOR CONTROLLER */

45   1          SEND: PROCEDURE (OUTDATA) PUBLIC;
46   2                DECLARE OUTDATA BYTE , STAT_COM BYTE AT (0F000H),
                             P_DATA BYTE AT (0F002H);
47   2                DO WHILE NOT SHR(STAT_COM,1);    /* WAIT UNTIL UPI41 IBF = 0 */
48   3                   END;
49   2                P_DATA = NOT OUTDATA;            /* MULTIBUS INVERTS */
50   2                END SEND;

                /* GET_COUNT OBTAINS A SPECIFIED MOTOR'S ABSOLUTE POSITION FROM
                   THE UPI-41 */

51   1          GET_COUNT: PROCEDURE (MOTOR_NUMBER) INTEGER;
52   2                     DECLARE MOTOR_NUMBER BYTE;
53   2                     DECLARE (LSB,MSB) BYTE;
54   2                     DECLARE COUNT INTEGER AT (@LSB);
55   2                     CALL SEND('C');          /* UPI-41 PROMPT FOR SENDING */
56   2                     MSB = RECEIVE;           /* TARGET COUNT TO 86/12 */
57   2                     LSB = RECEIVE;
58   2                     RETURN COUNT;
59   2                     END GET_COUNT;

                /* UPDATE_COUNTS UPDATES THE MOTOR COUNTS ON THE CURRENT TRACK.
                   WHEN FIRST PASS IS SET, THE TCNT ON THE STARTING TRACK IS
                   RECORDED IN TCOUNT0. THIS IS DONE FROM INITIATE VARIABLES.
                   FOR ALL OTHER CALLS, FIRST_PASS IS RESET, AND THE CURRENT
                   TRACK'S MOTOR COUNTS ARE UPDATED. */

60   1          UPDATE_COUNTS: PROCEDURE (FIRST_PASS) PUBLIC;
61   2                         DECLARE FIRST_PASS BYTE;
62   2                         DO CASE (CURRENT_TRACK AND 03H);
63   3                            ;
64   3                            DO;
65   4                              IF FIRST_PASS = 1
                                    THEN TCOUNT0 = GET_COUNT(1);
67   4                               ELSE TCNT = GET_COUNT(1);
68   4                              END;
69   3                            DO;
70   4                             IF FIRST_PASS = 1
                                   THEN TCOUNT0 = GET_COUNT(2);
72   4                              ELSE TCNT = GET_COUNT(2);
73   4                              END;
74   3                            DO;
75   4                             IF FIRST_PASS = 1
```

```
                        THEN TCOUNT0 = GET_COUNT(3);
 77   4                     ELSE TCNT = GET_COUNT(3);
 78   4                     END;
 79   3                   END;
 80   2                 END UPDATE_COUNTS;

 81   1     TARGET_DATA: PROCEDURE PUBLIC;

           /* FIRST, GET HTARG FOR FLIGHT, THEN GET TARGET_Y IN CASE TANK
              IS ON TRACK_3, THEN GET EFFECTIVE TANK LENGTHS AND HEIGHTS,
              THEN GET HILL SLOPE AND HILL Z IN CASE TANK IS ON TRACK 2,
              IN ANY CASE, GET TARGET Z FOR GROUNDED PROCEDURE.          */

 82   2          CALL UPDATE_COUNTS(0);
 83   2          HTARG = 1.694E-5 * FLOAT(TCNT - TCOUNT0); /* 1.694E-5 RAD/COUNT */
 84   2          TARGET_Y = 0.05247 * FLOAT(TCNT);  /* 0.05247 METERS/COUNT */
 85   2          ALPHA = FLOAT(ACNT)/15.278875;     /* 15.278875 COUNTS/RAD. */
 86   2          IF DCNT > 200
             THEN DEPRESSION_DEPTH = TANK_HEIGHT * FLOAT(550 - DCNT)/350.; /* 350 COUNTS IN A TANK_HEIGHT *
 88   2          ELSE DEPRESSION_DEPTH = TANK_HEIGHT; /* TANK IS HIDDEN */

           /* CORRECT GUNNER AIMING ERROR Z ON TRACK 1, NOTE DCNT = 550 ON ALL OTHER TRACKS*/

 89   2          GAEZ_F = GAEZ_F + TANK_HEIGHT * FLOAT(550-DCNT)/1.050E6;
      2          EFFECTIVE_TANK_LENGTH = TANK_LENGTH * ABS(@qerCOS(ALPHA)) +
                                         TANK_WIDTH * ABS(@qerSIN(ALPHA));
 91   2          EFF_HALF_TANK_LENGTH = EFFECTIVE_TANK_LENGTH/2.0;
 92   2          EFF_HALF_TANK_HEIGHT = (TANK_HEIGHT - DEPRESSION_DEPTH)/2.0;

 93   2          IF CURRENT_TRACK = 2
             THEN HILL_DATA: DO;

           /********** GET HILL SLOPE AND HILL Z AT TARGET **************/

           /* GET ENTRY JUST BEYOND TARGET Y */

 95   3     I = 0;
 96   3     DO WHILE TCNT >= MODEL_COUNT(I);
 97   4        I = I + 1;
 98   4        END;

           /* IF OUT OF TABLE, THEN EVERYTHING IS ZERO */

 99   3     IF (I = 0) OR (I >= 32 )
           THEN BETA, HILL_Z = 0.;

           /* IF IN TABLE, INTERPOLATE */

101   3     ELSE DO;
102   4          BETA = HILL_SLOPE(I) - FLOAT((MODEL_COUNT(I) - TCNT) 138)*
                                        (HILL_SLOPE(I) - HILL_SLOPE(I-1));
103   4          HILL_Z = HILL_HEIGHT(I) - FLOAT((MODEL_COUNT(I) - TCNT) 138)*
                                           (HILL_HEIGHT(I) - HILL_HEIGHT(I-1));
104   4          END;

105   3     END HILL_DATA;
```

AD-A143 646 SIMULATED TANK ANTI-ARMOR GUNNERY SYSTEM (STAGS-TOW) 2/3
(U) NAVAL TRAINING EQUIPMENT CENTER ORLANDO FL
A MARSHALL ET AL. MAY 83 PMT-E-T-6605.83

UNCLASSIFIED F/G 19/5 NL





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

```
106  2     DO CASE (CURRENT_TRACK AND 03H);
107  3        ;
108  3        TARGET_Z = HALF_TANK_HEIGHT - DEPRESSION_DEPTH;
109  3        TARGET_Z = HILL_Z + HALF_TANK_HEIGHT;
110  3        TARGET_Z = HALF_TANK_HEIGHT;
111  3        END;

112  2     END TARGET_DATA;

113  1     ACTIVE_TRACK: PROCEDURE PUBLIC;
114  2                   IF TARGET_SWITCH = 0
                         THEN CURRENT_TRACK = STARTING_TRACK;
116  2                   ELSE CURRENT_TRACK = FINAL_TRACK;
117  2                   END ACTIVE_TRACK;

118  1     HIT: PROCEDURE(TYPE);
119  2          DECLARE TYPE BYTE;
120  2          CALL SOUND(HIT_TARGET);
121  2          SPEECH_HIT = 1;
122  2          BIRD_MISSES,SPEECH_MISS = 0;
123  2          FINISHED = 1;
124  2          DO CASE TYPE;
125  3             BIRD_HITS = 1;
126  3             HIT_DISABLE = 1;
127  3             HIT_KILL = 1;
128  3             ;
129  3             END;
130  2          END HIT;
131  1     HILL: PROCEDURE;
132  2           BIRD_MISSES,SPEECH_MISS,HILL_IMPACT = 1;
133  2           END HILL;

134  1     GROUNDED: PROCEDURE PUBLIC;
135  2               DECLARE HEIGHT REAL;
136  2               HEIGHT = ((TARGET_Z - LAUNCH_HEIGHT)* X / 3000.) + LAUNCH_HEIGHT;
137  2               IF (HEIGHT + Z) <= 0.
                     THEN DO;
139  3                    FINISHED, GRND_BIRD = 1;
140  3                    CALL SOUND(GROUND_EXP);
141  3                    END;
142  2               END GROUNDED;

                    /* LEVEL DETERMINES HIT OR MISS FOR TARGET ON LEVEL TERRAIN. */

143  1     LEVEL: PROCEDURE (HMISS,VMISS);
144  2            DECLARE (HMISS,VMISS) REAL;

145  2            IF ABS(VMISS) < EFF_HALF_TANK_HEIGHT AND
                     ABS(HMISS) < EFF_HALF_TANK_LENGTH
                  THEN DO;
147  3                    IF ABS(HMISS) < KILL_BOX THEN CALL HIT(KILL);
149  3                    ELSE CALL HIT(DISABLE_);
150  3                    END;
151  2            ELSE FLY_PAST:  DO;
152  3                            BIRD_MISSES,SPEECH_MISS = 1;
153  3                            IF HMISS > EFF_HALF_TANK_LENGTH
```

```
                              THEN LEFT = HMISS - EFF_HALF_TANK_LENGTH;
155   3                       IF HMISS < 0. AND ABS(HMISS) > EFF_HALF_TANK_LENGTH
                              THEN RIGHT = HMISS + EFF_HALF_TANK_LENGTH;
157   3                       IF VMISS > EFF_HALF_TANK_HEIGHT
                              THEN UP = VMISS - EFF_HALF_TANK_HEIGHT;
159   3                       IF VMISS < 0. AND ABS(VMISS) > EFF_HALF_TANK_HEIGHT
                              THEN CALL HILL;
161   3                       END FLY_PAST;
162   2             END LEVEL;
                 /* FALLING SLOPE DETERMINES HIT OR MISS FOR TARGET PARTIALLY
                    OBSCURED BY A HILL WITH SLOPE FALLING FROM RIGHT TO LEFT */

163   1     FALLING_SLOPE: PROCEDURE (SLOPE, INTERCEPT);
164   2                    DECLARE (SLOPE, INTERCEPT) REAL;
165   2                    DECLARE (HILL_Y, LEFT_TANK_EDGE, RIGHT_TANK_EDGE) REAL;
166   2                    HILL_Y = (MISSILE_Z - INTERCEPT)/SLOPE;
167   2                    LEFT_TANK_EDGE = TARGET_Y + EFF_HALF_TANK_LENGTH;
168   2                    RIGHT_TANK_EDGE = TARGET_Y - EFF_HALF_TANK_LENGTH;
169   2                    IF (MISSILE_Z > 0.) AND (MISSILE_Z < TANK_HEIGHT) AND
                              (MISSILE_Y < LEFT_TANK_EDGE AND MISSILE_Y > HILL_Y) AND
                              (MISSILE_Y > RIGHT_TANK_EDGE)
                         THEN CALL HIT(REGULAR);
171   2                    ELSE MISS: DO;
172   3                              BIRD_MISSES,SPEECH_MISS = 1;
173   3                              IF MISSILE_Y < HILL_Y
                                     THEN CALL HILL;
175   3                              ELSE FLY_PAST: DO;
176   4                                   IF MISSILE_Y > LEFT_TANK_EDGE
                                          THEN LEFT = MISSILE_Y - LEFT_TANK_EDGE;
178   4                                   IF MISSILE_Y < RIGHT_TANK_EDGE
                                          THEN RIGHT = RIGHT_TANK_EDGE - MISSILE_Y;
180   4                                   IF MISSILE_Z > TANK_HEIGHT
                                          THEN UP = MISSILE_Z - TANK_HEIGHT;
182   4                                   END FLY_PAST;
183   3                              END MISS;
184   2                  END FALLING_SLOPE;
                 /* RISING SLOPE DETERMINES HIT OR MISS FOR TARGET PARTIALLY
                    OBSCURED BY A HILL WITH SLOPE RISING FROM RIGHT TO LEFT. */

185   1     RISING_SLOPE: PROCEDURE (SLOPE, INTERCEPT);
186   2                   DECLARE (SLOPE, INTERCEPT) REAL;
187   2                   DECLARE (HILL_Y, RIGHT_TANK_EDGE, LEFT_TANK_EDGE) REAL;
188   2                   HILL_Y = (MISSILE_Z - INTERCEPT)/SLOPE;
189   2                   RIGHT_TANK_EDGE = TARGET_Y - EFF_HALF_TANK_LENGTH;
190   2                   LEFT_TANK_EDGE = TARGET_Y + EFF_HALF_TANK_LENGTH;
191   2                   IF (MISSILE_Z > 0.) AND (MISSILE_Z < TANK_HEIGHT) AND
                             (MISSILE_Y > RIGHT_TANK_EDGE AND MISSILE_Y < HILL_Y) AND
                             (MISSILE_Y < LEFT_TANK_EDGE)
                        THEN CALL HIT(REGULAR);
193   2                   ELSE MISS: DO;
194   3                              BIRD_MISSES,SPEECH_MISS = 1;
195   3                              IF MISSILE_Y > HILL_Y
                                     THEN CALL HILL;
197   3                              ELSE FLY_PAST: DO;
198   4                                   IF MISSILE_Y < RIGHT_TANK_EDGE
```

```
                                    THEN RIGHT = RIGHT_TANK_EDGE - MISSILE_Y;
200   4                             IF MISSILE_Y > LEFT_TANK_EDGE
                                    THEN LEFT = MISSILE_Y - LEFT_TANK_EDGE;
202   4                             IF MISSILE_Z > TANK_HEIGHT
                                    THEN UP = MISSILE_Z - TANK_HEIGHT;
204   4                             END FLY_PAST;
205   3                       END MISS;
206   2             END RISING_SLOPE;
                 /* TRACK 1 ADJUSTS MISSILE Z FOR POSSIBLE DEPRESSION AND CALLS
                    LEVEL TO DETERMINE HIT OR MISS */

207   1      TRACK_1: PROCEDURE PUBLIC;
208   2               TEMP = MISSILE_Z - DEPRESSION_DEPTH - EFF_HALF_TANK_HEIGHT;
209   2               CALL LEVEL(OFF_H, TEMP);
210   2               END TRACK_1;

                 /* TRACK 2 ROTATES OFF_H AND Z INTO COORDINATE SYSTEM CENTERED
                    ON THE TARGET AND PARALLEL TO THE SLOPE OF THE HILL. THE
                    MISSILE COORDINATES IN THIS SYSTEM ARE THEN USED TO CALL LEVEL */
211   1      TRACK_2: PROCEDURE PUBLIC;

212   2      MAGNITUDE = mqerY2X((mqerY2X(OFF_H,2,0) + mqerY2X(Z,2,0)),0.5);
213   2      ANGLE = mqerAT2(OFF_H,Z);
214   2      MISSILE_Y_PRIME = MAGNITUDE * mqerSIN(ANGLE + BETA);
215   2      MISSILE_Z_PRIME = MAGNITUDE * mqerCOS(ANGLE + BETA);

               /******* WITH NEW COORDINATES, CALL LEVEL FOR HIT OR MISS *******/

216   2      CALL LEVEL(MISSILE_Y_PRIME, MISSILE_Z_PRIME);

217   2      END TRACK_2;

                 /* TRACK 3 IS BROKEN UP INTO REGIONS OF LEVEL TERRAIN, FALLING
                    SLOPE OBSCURING TANK, RISING SLOPE OBSCURING TANK, AND TOTALLY
                    AND TOTALLY OBSCURED. TCNT IS USED TO PLACE THE TARGET IN THE
                    CORRECT REGION, AND THEN THE APPROPRIATE PROCEDURE IS CALLED. */

218   1      TRACK_3: PROCEDURE PUBLIC;

                 /* BREAKPOINTS OF REGIONS */

219   2             DECLARE Y_A LITERALLY  '52.52',
                              Y_B LITERALLY  '65.22',
                              Y_C LITERALLY '120.00',
                              Y_D LITERALLY '141.70',
                              Y_E LITERALLY '165.30',
                              Y_F LITERALLY '180.70',
                              Y_G LITERALLY '222.80',
                              Y_H LITERALLY '233.20';

                 /* PARAMETERS OF HILL IN FALLING AND RISING SLOPE REGIONS */

220   2             DECLARE B_SLOPE LITERALLY '0.2105',
                              B_INCPT LITERALLY '-11.06',
                              D_SLOPE LITERALLY '-0.1323',
```

```
                        D_INCPT LITERALLY '18.75',
                        F_SLOPE LITERALLY '0.1705',
                        F_INCPT LITERALLY '-28.17',
                        H_SLOPE LITERALLY '-0.2416',
                        H_INCPT LITERALLY '56.36';

                /* TANK IS LOCATED IN CORRECT REGION, AND APPROPRIATE RULES
                   OF HIT OR MISS APPLIED */

221   2           MISSILE_Y = TARGET_Y + OFF_H;
222   2           IF TARGET_Y < (Y_A - EFFECTIVE_TANK_LENGTH)
                  THEN CALL LEVEL(OFF_H, Z);
224   2           ELSE IF TARGET_Y < Y_B
                   THEN CALL RISING_SLOPE(B_SLOPE,B_INCPT);
226   2            ELSE IF TARGET_Y < Y_C
                    THEN CALL HILL;
228   2             ELSE IF TARGET_Y < (Y_D + EFFECTIVE_TANK_LENGTH)
                      THEN CALL FALLING_SLOPE(D_SLOPE,D_INCPT);
230   2              ELSE IF TARGET_Y < (Y_E - EFFECTIVE_TANK_LENGTH)
                       THEN CALL LEVEL(OFF_H,Z);
232   2               ELSE IF TARGET_Y < Y_F
                        THEN CALL RISING_SLOPE(F_SLOPE,F_INCPT);
234   2                ELSE IF TARGET_Y < Y_G
                         THEN CALL HILL;
236   2                 ELSE IF TARGET_Y < (Y_H + EFFECTIVE_TANK_LENGTH)
                          THEN CALL FALLING_SLOPE(H_SLOPE,H_INCPT);
238   2                 ELSE CALL LEVEL(OFF_H,Z);
239   2           END TRACK_3;

240   1     END TOW_TARGET_MODULE;
```

CROSS-REFERENCE LISTING

DEFN	ADDR	SIZE	NAME, ATTRIBUTES, AND REFERENCES
			ABS. . . . . BUILTIN 90 145 147 155 159
9	0002H	2	ACNT . . . . . INTEGER PUBLIC 85
113	04FEH	46	ACTIVE_TRACK . . . . . PROCEDURE PUBLIC STACK=0002H
13	0014H	4	ALPHA. . . . . REAL 85 90
15	003CH	4	ANGLE. . . . . REAL 213 214 215
15	0034H	4	BETA . . . . . REAL 100 102 214 215
2	A016H	1	BIRD_HITS. . . . . BYTE AT ABSOLUTE 125
2	A017H	1	BIRD_MISSES. . . . . BYTE AT ABSOLUTE 122 132 152 172 194
220			B_INCPT. . . . . LITERALLY 225
220			B_SLOPE. . . . . LITERALLY 225
54	006FH	2	COUNT. . . . . INTEGER AT 58
11	006DH	1	CURRENT_TRACK. . . . . BYTE PUBLIC 62 93 106 115 116
9	0004H	2	DCNT . . . . . INTEGER PUBLIC 86 87 89
13	0018H	4	DEPRESSION_DEPTH . . . REAL 87 88 92 108 208
23			DISABLE_ . . . . . LITERALLY 149
220			D_INCPT. . . . . LITERALLY 229
220			D_SLOPE. . . . . LITERALLY 229
14	0024H	4	EFFECTIVE_TANK_LENGTH. REAL 90 91 222 228 230 236
16	0044H	4	EFF_HALF_TANK_HEIGHT . REAL 92 145 157 158 159 208
14	0028H	4	EFF_HALF_TANK_LENGTH . REAL 91 145 153 154 155 156 167 168 189 190
163	0750H	380	FALLING_SLOPE. . . . . PROCEDURE STACK=0020H 229 237
4	A00EH	1	FINAL_TRACK. . . . . BYTE AT ABSOLUTE 116
8	0000H	1	FINISHED . . . . . BYTE EXTERNAL(2) 123 139
60	0006H	1	FIRST_PASS . . . . . BYTE PARAMETER AUTOMATIC 61 65 70 75
			FLOAT. . . . . BUILTIN 83 84 85 87 89 102 103
151	067AH		FLY_PAST . . . . . LABEL
175	0851H		FLY_PAST . . . . . LABEL
197	09CDH		FLY_PAST . . . . . LABEL
220			F_INCPT. . . . . LITERALLY 233
220			F_SLOPE. . . . . LITERALLY 233
12	0000H	4	GAEZ_F . . . . . REAL EXTERNAL(3) 89
51	0235H	37	GET_COUNT. . . . . PROCEDURE INTEGER STACK=000CH 66 67 71 72 76 77
3	A01CH	1	GRND_BIRD. . . . . BYTE AT ABSOLUTE 139
134	05A5H	92	GROUNDED . . . . . PROCEDURE PUBLIC STACK=000AH
22			GROUND_EXP . . . . . LITERALLY 140
21			HALF_TANK_HEIGHT . . . LITERALLY 108 109 110
135	0050H	4	HEIGHT . . . . . REAL 136 137
131	058EH	23	HILL . . . . . PROCEDURE STACK=0002H 160 174 196 227 235
94	03EAH		HILL_DATA. . . . . LABEL
20	00C0H	128	HILL_HEIGHT. . . . . REAL ARRAY(32) DATA 103
11	006CH	1	HILL_IMPACT. . . . . BYTE PUBLIC 132
19	0040H	128	HILL_SLOPE . . . . . REAL ARRAY(32) DATA 102
165	0054H	4	HILL_Y . . . . . REAL 166 169 173
187	0060H	4	HILL_Y . . . . . REAL 188 191 195
15	002CH	4	HILL_Z . . . . . REAL 100 103 109
118	052CH	98	HIT. . . . . PROCEDURE STACK=000AH 148 149 170 192
5	A021H	1	HIT_DISABLE. . . . . BYTE AT ABSOLUTE 126
5	A020H	1	HIT_KILL . . . . . BYTE AT ABSOLUTE 127
22			HIT_TARGET . . . . . LITERALLY 120

```
143  FFFCH    4  MMISS. . . . . .  REAL PARAMETER AUTOMATIC      144  145  147  153  154  155  156
 12  0000H    4  MTARG. . . . . .  REAL EXTERNAL(7)          83
220              M_INCPT. . . . . .  LITERALLY       237
220              M_SLOPE. . . . . .  LITERALLY       237
 17  006EH    1  I. . . . . .  BYTE        95   96   97   99  102  103
 40  F002H    1  INDATA . . . . . .  BYTE AT ABSOLUTE          43
185  FFF8H    4  INTERCEPT. . . . . .  REAL PARAMETER AUTOMATIC      186  188
163  FFF8H    4  INTERCEPT. . . . . .  REAL PARAMETER AUTOMATIC      164  166
 23              KILL . . . . . .  LITERALLY       148
 21              KILL_BOX . . . . . .  LITERALLY       147
 21              LAUNCH_HEIGHT. . . . .  LITERALLY       136
 10  000CH    4  LEFT . . . . . .  REAL PUBLIC          154  177  201
165  0058H    4  LEFT_TANK_EDGE . . . .  REAL        167  169  176  177
187  0068H    4  LEFT_TANK_EDGE . . . .  REAL        190  191  200  201
143  0601H  335  LEVEL. . . . . .  PROCEDURE STACK=001CH          209  216  223  231  238
 53  006FH    1  LSB. . . . . .  BYTE        54   57
 15  0038H    4  MAGNITUDE. . . . . .  REAL        212  214  215
193  09A0H       MISS . . . . . .  LABEL
171  0824H       MISS . . . . . .  LABEL
 14  001CH    4  MISSILE_Y. . . . . .  REAL        169  173  176  177  178  179  191  195  198  199  200  201
                                        221
 16  0048H    4  MISSILE_Y_PRIME. . . .  REAL        214  216
 12  0000H    4  MISSILE_Z. . . . . .  REAL EXTERNAL(6)          166  169  180  181  188  191  202  203  209
 16  004CH    4  MISSILE_Z_PRIME. . . .  REAL        215  216
 18  0000H   64  MODEL_COUNT. . . . . .  INTEGER ARRAY(32) DATA          96  102  103
 51  0004H    1  MOTOR_NUMBER . . . . .  BYTE PARAMETER AUTOMATIC        52
 24  0000H       MQERAT2. . . . . .  PROCEDURE REAL EXTERNAL(9) STACK=0000H        213
 27  0000H       MQERCOS. . . . . .  PROCEDURE REAL EXTERNAL(10) STACK=0000H        90  215
 30  0000H       MQERSIN. . . . . .  PROCEDURE REAL EXTERNAL(11) STACK=0000H        90  214
 33  0000H       MQERY2X. . . . . .  PROCEDURE REAL EXTERNAL(12) STACK=0000H       212
 53  0070H    1  MSB. . . . . .  BYTE        56
 12  0000H    4  OFF_H. . . . . .  REAL EXTERNAL(8)          209  212  213  221  223  231  238
 45  0006H    1  OUTDATA. . . . . .  BYTE PARAMETER AUTOMATIC        46   49
  7              PORT_C . . . . . .  LITERALLY
 46  F002H    1  P_DATA . . . . . .  BYTE AT ABSOLUTE          49
 39  01E8H   33  RECEIVE. . . . . .  PROCEDURE BYTE PUBLIC STACK=0002H         56   57
 23              REGULAR. . . . . .  LITERALLY       170  192
 10  0008H    4  RIGHT. . . . . .  REAL PUBLIC          156  179  199
187  0064H    4  RIGHT_TANK_EDGE. . . .  REAL        189  191  198  199
165  005CH    4  RIGHT_TANK_EDGE. . . .  REAL        168  169  178  179
185  08CCH  380  RISING_SLOPE . . . . .  PROCEDURE STACK=0020H          225  233
 45  0209H   44  SEND . . . . . .  PROCEDURE PUBLIC STACK=0004H         55
                 SHR. . . . . .  BUILTIN         47
185  FFFCH    4  SLOPE. . . . . .  REAL PARAMETER AUTOMATIC      186  188
163  FFFCH    4  SLOPE. . . . . .  REAL PARAMETER AUTOMATIC      164  166
 36  0000H       SOUND. . . . . .  PROCEDURE EXTERNAL(13) STACK=0000H       120  140
  6  0000H    1  SPEECH_HIT . . . . . .  BYTE EXTERNAL(0)          121
  6  0000H    1  SPEECH_MISS. . . . . .  BYTE EXTERNAL(1)          122  132  152  172  194
  4  A00CH    1  STARTING_TRACK . . . .  BYTE AT ABSOLUTE          115
 40  F000H    1  STATUS . . . . . .  BYTE AT ABSOLUTE           41
 46  F000H    1  STAT_COM . . . . . .  BYTE AT ABSOLUTE           47
 21              TANK_HEIGHT. . . . . .  LITERALLY       87   88   89   92  169  180  181  191  202  203
 21              TANK_LENGTH. . . . . .  LITERALLY       90
 21              TANK_WIDTH . . . . . .  LITERALLY       90
 81  02E5H  537  TARGET_DATA. . . . . .  PROCEDURE PUBLIC STACK=0018H
  4  A00DH    1  TARGET_SWITCH. . . . .  BYTE AT ABSOLUTE          114
 14  0020H    4  TARGET_Y . . . . . .  REAL        84  167  168  189  190  221  222  224  226  228  230  232
```

```
                                                  234  236
    15  0030H     4  TARGET_Z . . . . . .   REAL        108  109  110  136
     9  0000H     2  TCNT . . . . . . . .   INTEGER PUBLIC         67   72   77   83   84   96  102  103
     9  0006H     2  TCOUNTO. . . . . . .   INTEGER PUBLIC         66   71   76   83
    16  0040H     4  TEMP . . . . . . . .   REAL        208  209
    30  0000H     4  THETA. . . . . . . .   REAL PARAMETER          31
    27  0000H     4  THETA. . . . . . . .   REAL PARAMETER          28
     1  01E8H        TOW_TARGET_MODULE. . .  PROCEDURE STACK=0000H
   207  0A48H    39  TRACK_1. . . . . . .   PROCEDURE PUBLIC STACK=0020H
   211  0A6FH   143  TRACK_2. . . . . . .   PROCEDURE PUBLIC STACK=0020H
   218  0AFEH   402  TRACK_3. . . . . . .   PROCEDURE PUBLIC STACK=0026H
   118  0004H     1  TYPE . . . . . . . .   BYTE PARAMETER AUTOMATIC        119  124
    10  0010H     4  UP . . . . . . . . .   REAL PUBLIC           158  181  203
    60  025AH   139  UPDATE_COUNTS. . . .   PROCEDURE PUBLIC STACK=0012H         82
   143  FFF8H     4  VMISS. . . . . . . .   REAL PARAMETER AUTOMATIC        144  145  157  158  159
    36  0000H     1  WHAT_KIND. . . . . .   BYTE PARAMETER          37
    24  0000H     4  X. . . . . . . . . .   REAL PARAMETER          25
    12  0000H     4  X. . . . . . . . . .   REAL EXTERNAL(4)             136
    33  0000H     4  X. . . . . . . . . .   REAL PARAMETER          34
    24  0000H     4  Y. . . . . . . . . .   REAL PARAMETER          25
    33  0000H     4  Y. . . . . . . . . .   REAL PARAMETER          34
   219                Y_A. . . . . . . . .   LITERALLY       222
   219                Y_B. . . . . . . . .   LITERALLY       224
   219                Y_C. . . . . . . . .   LITERALLY       226
   219                Y_D. . . . . . . . .   LITERALLY       228
   219                Y_E. . . . . . . . .   LITERALLY       230
   219                Y_F. . . . . . . . .   LITERALLY       232
   219                Y_G. . . . . . . . .   LITERALLY       234
   219                Y_H. . . . . . . . .   LITERALLY       236
    12  0000H     4  Z. . . . . . . . . .   REAL EXTERNAL(5)             137  212  213  223  231  238


MODULE INFORMATION:

     CODE AREA SIZE     = 0C90H   3216D
     CONSTANT AREA SIZE = 0000H      0D
     VARIABLE AREA SIZE = 0071H    113D
     MAXIMUM STACK SIZE = 0026H     38D
     423 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION
```

```
ISIS-II PL/M-86 V2.1 COMPILATION OF MODULE TOW_SPEECH_MODULE
OBJECT MODULE PLACED IN :F2:TOWSPC.OBJ
COMPILER INVOKED BY:  PLM86 :F2:TOWSPC.010 DEBUG ROM MEDIUM XREF IXREF WORKFILES(:F2:,:F2:) DATE(1/17/83)
 1          TOW_SPEECH_MODULE: DO;

            /********************* DECLARE ABSOLUTE ADDRESSES *********************/

 2  1       DECLARE DAY_SIGHT BYTE AT (0A007H), CONTINUE BYTE AT (0A01FH),
                    EAST_WEST BYTE AT (0A00FH), STARTING_TRACK BYTE AT (0A00CH);

 3  1       DECLARE (H_MIS_ASCII,V_MIS_ASCII,X_MIS_ASCII) (22) BYTE AT (0A022H);

            /********************* DECLARE GLOBAL VARIABLES *********************/

 4  1       DECLARE (GUIDANCE_LOST,HILL_IMPACT, WIRE_BROKE) BYTE EXTERNAL;

 5  1       DECLARE (RIGHT,LEFT,UP,SHORT) REAL EXTERNAL;

            /********************* DECLARE EXCLUSIVE VARIABLES *********************/

 6  1       DECLARE ALREADY_OUT BYTE;          /* FLAG USED BY COACHING */

 7  1       DECLARE (SPEECH_HIT, SPEECH_MISS) BYTE PUBLIC;

 8  1       DECLARE I WORD, K BYTE;

 9  1       DECLARE PORT_B LITERALLY '0CAH', PORT_C LITERALLY '0CCH';

               /* FIRE SIGNAL FROM DIGITALKER BOARD */

10  1       DECLARE DONT_FIRE LITERALLY 'SHR(INPUT(PORT_B),1)';

               /* GAE LIMITS WHICH CAUSE COACHING WHEN EXCEEDED */

11  1       DECLARE GAEH_LIMIT LITERALLY '600.0E-6', GAEV_LIMIT LITERALLY '150.0E-6';

            /************************ DECLARE WORD CODES ************************/

               /* VOICE SUBSYSTEM REQUIRES A SEQUENCE OF THREE BYTES TO
                  SPECIFY THE WORD OR PAUSE TO BE SPOKEN. THE FORMAT IS
                  WORD PAGE NUMBER, WORD ADDRESS ON PAGE, VOLUME OF WORD */

12  1       DECLARE USE             LITERALLY '02H,04H,02H',
                    DAY             LITERALLY '02H,02H,02H',
                    NIGHT           LITERALLY '02H,08H,02H',
                    SIGHT           LITERALLY '02H,11H,02H',
                    SQUAD           LITERALLY '02H,18H,03H',
                    TANK            LITERALLY '02H,00H,03H',
                    EAST            LITERALLY '02H,0BH,02H',
                    WEST            LITERALLY '02H,0CH,02H',
                    THREE           LITERALLY '00H,03H,02H',
                    THOUSAND        LITERALLY '00H,1DH,02H',
                    METER           LITERALLY '00H,6AH,02H',
                    SS              LITERALLY '00H,81H,02H',
```

```
                AT_             LITERALLY '02H,05H,02H',
                MY              LITERALLY '02H,15H,02H',
                COMMAND         LITERALLY '02H,03H,02H',
                FIRE            LITERALLY '02H,06H,03H',
                FIRE_STUPID     LITERALLY '02H,06H,03H',
                ABORT           LITERALLY '01H,00H,02H',
                CEASE           LITERALLY '02H,0EH,02H',
                TRACKING        LITERALLY '02H,10H,02H',
                HIT             LITERALLY '02H,13H,02H',
                MISS            LITERALLY '02H,12H,02H',
                RIGHT_          LITERALLY '00H,80H,02H',
                LEFT_           LITERALLY '00H,63H,02H',
                HIGH_           LITERALLY '00H,5BH,02H',
                LOW_            LITERALLY '00H,67H,02H',
                POINT           LITERALLY '00H,7AH,02H',
                FROM            LITERALLY '01H,34H,02H',
                TARGET          LITERALLY '02H,16H,02H',
                HUNDRED         LITERALLY '00H,1CH,02H',
                ON              LITERALLY '00H,73H,02H',
                CORRECT         LITERALLY '01H,16H,02H',
                VERY_SHORT_PAUSE LITERALLY '00H,43H,00H',
                SHORT_PAUSE     LITERALLY '00H,44H,00H',
                MEDIUM_PAUSE    LITERALLY '00H,45H,00H',
                LONG_PAUSE      LITERALLY '00H,46H,00H',
                VERY_LONG_PAUSE LITERALLY '00H,47H,00H';

        /*********************** DECLARE PHRASES ***********************/

            /* PHRASES ARE STORED IN ROM AS CONSTANT STRINGS OF THE
               WORD CODES DEFINED ABOVE */

13  1       DECLARE PHRASE_1 (*) BYTE DATA
                        (USE,VERY_SHORT_PAUSE,DAY,VERY_SHORT_PAUSE,SIGHT);
14  1       DECLARE PHRASE_2 (*) BYTE DATA
                        (USE,VERY_SHORT_PAUSE,NIGHT,VERY_SHORT_PAUSE,SIGHT);
15  1       DECLARE PHRASE_3 (*) BYTE DATA
                        (SQUAD,VERY_LONG_PAUSE,TANK,MEDIUM_PAUSE);
16  1       DECLARE PHRASE_4 (*) BYTE DATA
                        (EAST,MEDIUM_PAUSE);
17  1       DECLARE PHRASE_5 (*) BYTE DATA
                        (WEST,MEDIUM_PAUSE);
18  1       DECLARE PHRASE_6 (*) BYTE DATA
                        (THREE,VERY_SHORT_PAUSE,THOUSAND,SHORT_PAUSE,METER,SS);
19  1       DECLARE PHRASE_7 (*) BYTE DATA
                        (AT_,VERY_SHORT_PAUSE,MY,VERY_SHORT_PAUSE,COMMAND);
20  1       DECLARE PHRASE_8 (*) BYTE DATA
                        (CEASE,SHORT_PAUSE,TRACKING);
21  1       DECLARE PHRASE_9 (*) BYTE DATA
                        (HIT);
22  1       DECLARE PHRASE_10 (*) BYTE DATA
                         (MISS,VERY_LONG_PAUSE);
23  1       DECLARE PHRASE_11 (*) BYTE DATA
                         (HIGH_,VERY_LONG_PAUSE);
24  1       DECLARE PHRASE_12 (*) BYTE DATA
                         (RIGHT_);
25  1       DECLARE PHRASE_13 (*) BYTE DATA
                         (LEFT_);
```

```
26  1     DECLARE PHRASE_14 (*) BYTE DATA
                              (METER);
27  1     DECLARE PHRASE_15 (*) BYTE DATA
                              (SS);
28  1     DECLARE PHRASE_16 (*) BYTE DATA
                              (SHORT_PAUSE);
29  1     DECLARE PHRASE_17 (*) BYTE DATA
                              (POINT);
30  1     DECLARE PHRASE_18 (*) BYTE DATA
                              (LONG_PAUSE);
31  1     DECLARE PHRASE_19 (*) BYTE DATA
                              (FIRE);
32  1     DECLARE PHRASE_20 (*) BYTE DATA
                              (FIRE_STUPID);
33  1     DECLARE PHRASE_21 (*) BYTE DATA
                              (ABORT);
34  1     DECLARE PHRASE_22 (*) BYTE DATA
                              (FROM,SHORT_PAUSE,TARGET);
35  1     DECLARE PHRASE_23 (*) BYTE DATA
                              (THOUSAND);
36  1     DECLARE PHRASE_24 (*) BYTE DATA
                              (HUNDRED);
37  1     DECLARE PHRASE_25 (*) BYTE DATA
                              (LOW_);
38  1     DECLARE PHRASE_26 (*) BYTE DATA
                              (HIGH_);
39  1     DECLARE PHRASE_27 (*) BYTE DATA
                              (CORRECT);

       /*********************** MODULE PROCEDURES ***********************/

       /* THE SEND PROCEDURE WRITES A BYTE TO THE UPI41 ON THE
          DIGITALKER BOARD WHEN IT IS READY */

40  1  SEND: PROCEDURE (OUTDATA);
41  2        DECLARE OUTDATA BYTE, STAT_COM BYTE AT (0D002H),
                    P_DATA BYTE AT (0D000H);
42  2        DO WHILE NOT SHR(STAT_COM,1);      /* WAIT UNTIL UPI41 IBF = 0 */
43  3           END;
44  2        P_DATA = NOT OUTDATA;              /* MULTIBUS INVERTS */
45  2        END SEND;

       /* XMIT WRITES AN ENTIRE PHRASE TO THE UPI41 ON THE DIGITALKER
          BOARD */

46  1  XMIT: PROCEDURE (PHRASE_PTR, PHRASE_LENGTH);
47  2        DECLARE PHRASE_PTR POINTER, PHRASE_LENGTH WORD;
48  2        DECLARE (ITEM BASED PHRASE_PTR) (1) BYTE;
49  2        I = 0;
50  2        DO WHILE I < PHRASE_LENGTH;
51  3           CALL SEND (ITEM(I));
52  3           I = I + 1;
53  3           END;
54  2        END XMIT;

       /*****************************************************************
                        IN THE FOLLOWING, 'WORD' MEANS THAT THE WORD
```

```
                    IS SPOKEN.
 ***************************************************************************/

            /* SAY_AMOUNT HAS DIGITALKER SAY THE NUMBER OF METERS THAT MISSILE
               MISSED BY. IT IS FOLLOWED BY A SHORT PAUSE.  */

 55  1      SAY_AMOUNT: PROCEDURE (BUFF_PTR);
 56  2                  DECLARE BUFF_PTR POINTER;
 57  2                  DECLARE (ASCII_NUMBER BASED BUFF_PTR) (4) BYTE;
 58  2                  IF ASCII_NUMBER(0) <> 20H                    /* NO THOUSANDS DIGIT */
                     THEN DO;
 60  3                       CALL SEND(00H);                         /* PAGE ZERO */
 61  3                       CALL SEND(ASCII_NUMBER(0) - 30H);       /* 'THOUSANDS DIGIT' */
 62  3                       CALL SEND(02H);                         /* VOLUME TWO */
 63  3                       CALL XMIT(@PHRASE_23,SIZE(PHRASE_23)); /*'THOUSAND' */
 64  3                       END;
 65  2                  IF (ASCII_NUMBER(1) <> 20H) AND (ASCII_NUMBER(1) <> 30H)
                                                                     /* NO HUNDREDS DIGIT */
                     THEN DO;
 67  3                       CALL SEND(00H);                         /* PAGE ZERO */
 68  3                       CALL SEND(ASCII_NUMBER(1) - 30H);       /* 'HUNDREDS DIGIT' */
 69  3                       CALL SEND(02H);                         /* VOLUME TWO */
 70  3                       CALL XMIT(@PHRASE_24,SIZE(PHRASE_24)); /* 'HUNDRED' */
 71  3                       END;

 72  2                  CALL SEND(00H);                              /* PAGE 0 */
 73  2                  IF (ASCII_NUMBER(2) = 20H)                   /* NO TENS DIGIT */
                     THEN DO;
 75  3                       IF ASCII_NUMBER(3) = 30H                /* ZERO HAS SPECIAL ADDRESS */
                          THEN CALL SEND(1FH);                       /* ADDRESS OF ZERO */
 77  3                       ELSE CALL SEND(ASCII_NUMBER(3) - 30H); /* OTHER ADDRESSES */
 78  3                       END;
 79  2                  ELSE IF ASCII_NUMBER(2) = 31H                /* TENS DIGIT IS ONE */
                          THEN CALL SEND(ASCII_NUMBER(3) - 30H + 0AH); /* SEND ADDRESS */
 81  2                       ELSE IF ASCII_NUMBER(2) <> 30H
                          THEN DO;                  /* TENS DIGIT IS NOT ZERO OR ONE */
 83  3                            CALL SEND(ASCII_NUMBER(2) - 1EH); /* SEND TENS ADDR. */
 84  3                            IF ASCII_NUMBER(3) <> 30H     /* UNLESS ONES IS ZERO */
                               THEN DO;
 86  4                                 CALL SEND(02H);          /* SEND VOLUME FOR TENS */
 87  4                                 CALL SEND(00H);                  /* PAGE ZERO */
 88  4                                 CALL SEND(ASCII_NUMBER(3) - 30H); /* ONES ADDR. */
 89  4                                 END;
 90  3                            END;
                      CALL SEND(02H);                    /* SEND VOLUME FOR LAST WORD */
 92  2                   IF ASCII_NUMBER(5) <> 30H          /* UNLESS NO TENTHS */
                      THEN DO;
 94  3                       CALL XMIT(@PHRASE_17,SIZE(PHRASE_17)); /* SAY POINT */
 95  3                       CALL SEND(00H);                         /* PAGE ZERO */
 96  3                       CALL SEND(ASCII_NUMBER(5) - 30H);       /* TENTHS ADDR. */
 97  3                       CALL SEND(02H);                         /* VOLUME */
 98  3                       END;
 99  2                   CALL XMIT(@PHRASE_16,SIZE(PHRASE_16));  /* SHORT PAUSE */
100  2                   END SAY_AMOUNT;

            /* MISS_DISTANCE HAS DIGITALKER SAY THE MISS_DISTANCE AND METER OR
               METERS FOLLOWED BY A LONG PAUSE */
```

```
101   1     MISS_DISTANCE: PROCEDURE (DISTANCE, ASCII_PTR);
102   2                    DECLARE DISTANCE REAL, ASCII_PTR POINTER;
103   2                    CALL SAY_AMOUNT(ASCII_PTR);              /* SAY THE AMOUNT */
104   2                    CALL XMIT(@PHRASE_14,SIZE(PHRASE_14));  /* SAY METERS UNLESS */
105   2                    IF FIX(10.0 * DISTANCE) <> 10            /* DISTANCE = 1.0 */
                           THEN CALL XMIT(@PHRASE_15,SIZE(PHRASE_15)); /* THEN SAY METER */
107   2                    CALL XMIT(@PHRASE_18,SIZE(PHRASE_18));      /* LONG PAUSE */
108   2                    END MISS_DISTANCE;

                     /* PROLOG IS CALLED FROM TOWMN BEFORE A TRIGGER PULL. */

109   1     PROLOG: PROCEDURE PUBLIC;
110   2             DO WHILE NOT CONTINUE;              /* WAIT TILL SCENARIO SELECTED */
111   3                END;
112   2             CONTINUE = 0;

                         /* TURN ON TARGET LIGHT */

113   2             DO CASE (STARTING_TRACK AND 03H);
114   3                ;
115   3                OUTPUT(PORT_C) = 0FEH;           /* PORT_C HAS INVERTED OUTPUTS */
116   3                OUTPUT(PORT_C) = 0FDH;
117   3                OUTPUT(PORT_C) = 0FBH;
118   3                END;

119   2             IF DAY_SIGHT = 1          /* 1 = 'USE DAY SIGHT', 0 = 'USE NIGHT SIGHT' */
                      THEN CALL XMIT(@PHRASE_1,SIZE(PHRASE_1));
121   2             ELSE CALL XMIT(@PHRASE_2,SIZE(PHRASE_2));
122   2             DO WHILE NOT CONTINUE;          /* WAIT TILL READY AND MOTORS START */
123   3                END;
124   2             CONTINUE = 0;
125   2             CALL XMIT(@PHRASE_3,SIZE(PHRASE_3));               /* 'ALERT! TANK' */
126   2             IF EAST_WEST = 1                              /* 1 = 'EAST', 0 = 'WEST' */
                      THEN CALL XMIT(@PHRASE_4,SIZE(PHRASE_4));
128   2             ELSE CALL XMIT(@PHRASE_5,SIZE(PHRASE_5));
129   2             CALL XMIT(@PHRASE_6,SIZE(PHRASE_6));               /* '3000 METERS' */
130   2             CALL TIME(10000);                                  /* 1 SEC DELAY */
131   2             CALL XMIT(@PHRASE_7,SIZE(PHRASE_7));              /* 'AT MY COMMAND' */
132   2             CALL TIME(10000);
133   2             DO WHILE DONT_FIRE;                    /* WAIT TILL CY512 SAYS FIRE*/
134   3                END;                             /* AND OUTPUT TO BIT1 ON PORT_B */
135   2             CALL XMIT(@PHRASE_19,SIZE(PHRASE_19));             /* 'FIRE' */
136   2             ALREADY_OUT = 0;                    /* ZERO A COACHING FLAG */
137   2             END PROLOG;

                     /* FIRE AGAIN MAKES DIGITALKER SAY FIRE LOUDER. CALLED FROM TOWMN IF
                        TRAINEE DOESN'T FIRE WITHIN TWO SECONDS AFTER FIRST COMMANDED. */

138   1     FIRE_AGAIN: PROCEDURE PUBLIC;
139   2                 CALL XMIT(@PHRASE_20,SIZE(PHRASE_20));   /* FIRE STUPID */
140   2                 END FIRE_AGAIN;

                     /* QUIT MAKES DIGITALKER SAY CEASE TRACKING THEN ABORT. CALLED FROM
                        TOWMN IF TRAINEE DOESN'T FIRE THE SECOND TIME. */

141   1     QUIT: PROCEDURE PUBLIC;
```

```
142  2          CALL XMIT(@PHRASE_8,SIZE(PHRASE_8));          /* CEASE TRACKING */
143  2          CALL XMIT(@PHRASE_18,SIZE(PHRASE_18));        /* LONG PAUSE */
144  2          CALL XMIT(@PHRASE_21,SIZE(PHRASE_21));        /* ABORT */
145  2          END QUIT;

          /* COACH CALLED FROM TOWFLP; FLIGHT PROCEDURE. VERTICAL COACHING
             OVERIDES HORIZONTAL COACHING, SINCE MORE CRITICAL. THIS IS TO AVOID
             CONFUSING THE GUNNER BY COACHING IN BOTH DIRECTIONS SIMULTANEOUSLY */

146  1     COACH: PROCEDURE (GAEH, GAEV) PUBLIC;
147  2          DECLARE (GAEH, GAEV) REAL;
148  2          IF (ABS(GAEH) < GAEH_LIMIT) AND (ABS(GAEV) < GAEV_LIMIT)
                   AND (ALREADY_OUT = 1)

                    /* WAS OFF TARGET, BACK ON TARGET */

                THEN DO;
150  3                CALL XMIT(@PHRASE_27,SIZE(PHRASE_27));    /* 'ON TARGET' */
151  3                ALREADY_OUT = 0;
152  3                END;
153  2          IF ABS(GAEV) > GAEV_LIMIT

                    /* HIGH OR LOW */

                THEN DO;
155  3                IF GAEV > 0. THEN CALL XMIT(@PHRASE_26,SIZE(PHRASE_26)); /* 'HIGH' */
157  3                ELSE CALL XMIT(@PHRASE_25,SIZE(PHRASE_25));              /* 'LOW ' */
158  3                ALREADY_OUT = 1;                                 /* OFF TARGET */
159  3                END;

                      /* RIGHT OR LEFT, BUT NOT HIGH OR LOW */

160  2          ELSE IF ABS(GAEH) > GAEH_LIMIT
                     THEN DO;
162  3                IF GAEH > 0. THEN CALL XMIT(@PHRASE_12,SIZE(PHRASE_12)); /* 'RIGHT ' */
164  3                ELSE CALL XMIT(@PHRASE_13,SIZE(PHRASE_13));   /* 'LEFT' */
165  3                ALREADY_OUT = 1;                               /* OFF TARGET */
166  3                END;
                END COACH;

             /* NOTE THAT EPILOG USES THE ASCII BUFFERS FOR MISS DISTANCE TO
                CAUSE DIGITALKER TO ALSO SAY THE MISS DISTANCES */

168  1     EPILOG: PROCEDURE PUBLIC;
169  2           CALL XMIT(@PHRASE_8,SIZE(PHRASE_8));            /* 'CEASE TRACKING' */
170  2           CALL TIME(10000);                               /* 1 SEC DELAY */
171  2           IF SPEECH_HIT = 1
                 THEN CALL XMIT(@PHRASE_9,SIZE(PHRASE_9));                /* 'HIT' */
173  2           IF SPEECH_MISS = 1
                 THEN DO;
175  3                CALL XMIT(@PHRASE_10,SIZE(PHRASE_10));             /* 'MISS' */
176  3                IF UP <> 0.
                      THEN DO;
178  4                     CALL MISS_DISTANCE(UP,@V_MIS_ASCII(0));     /* DISTANCE */
179  4                     CALL XMIT(@PHRASE_11,SIZE(PHRASE_11));         /* 'HIGH' */
180  4                     END;
181  3                IF RIGHT <> 0.
```

```
                    THEN DO;
183   4                 CALL MISS_DISTANCE(RIGHT,@H_MIS_ASCII(0));   /* DISTANCE */
184   4                 CALL XMIT(@PHRASE_12,SIZE(PHRASE_12));         /* 'RIGHT' */
185   4                 END;
186   3               IF LEFT <> 0.
                    THEN DO;
188   4                 CALL MISS_DISTANCE(LEFT,@H_MIS_ASCII(0));    /* DISTANCE */
189   4                 CALL XMIT(@PHRASE_13,SIZE(PHRASE_13));          /* 'LEFT' */
190   4                 END;
191   3               END;

                   /* NOTE THAT X_MIS_ASCII USED FOR SPECIAL COMMENTS IN
                      SPECIAL MISS SITUATIONS. DIGITALKER IS PREVENTED FROM
                      ATTEMPTING TO USE THE BUFFER IN THESE SITUATIONS */

192   2          ELSE IF (SHORT < 0. AND ABS(SHORT) > 2.6416) AND (WIRE_BROKE = 0)
                    AND (HILL_IMPACT = 0) AND (GUIDANCE_LOST = 0)
                    THEN DO;
194   3                 CALL XMIT(@PHRASE_10,SIZE(PHRASE_10));         /* 'MISS' */
195   3                 CALL MISS_DISTANCE(SHORT,@X_MIS_ASCII(0));    /* DISTANCE */
196   3                 CALL XMIT(@PHRASE_22,SIZE(PHRASE_22));     /* 'FROM TARGET' */
197   3                 END;
                  END EPILOG;

199   1     END TOW_SPEECH_MODULE;
```

CROSS-REFERENCE LISTING

DEFN	ADDR	SIZE	NAME, ATTRIBUTES, AND REFERENCES
12			ABORT. . . . . LITERALLY 33
			ABS. . . . . BUILTIN 148 153 160 192
6	0002H	1	ALREADY_OUT. . . . BYTE 136 148 151 158 165
57	0000H	4	ASCII_NUMBER . . . BYTE BASED(BUFF_PTR) ARRAY(4) 58 61 65 68 73 75 77 79 80 81 83 84 88 92 96
101	0004H	4	ASCII_PTR. . . . . POINTER PARAMETER AUTOMATIC 102 103
12			AT_. . . . . LITERALLY 19
55	0004H	4	BUFF_PTR . . . . . POINTER PARAMETER AUTOMATIC 56 57 58 61 65 68 73 75 77 79 80 81 83 84 88 92 96
12			CEASE. . . . . LITERALLY 20
146	0461H	293	COACH. . . . . PROCEDURE PUBLIC STACK=001EH
12			COMMAND. . . . . LITERALLY 19
2	A01FH	1	CONTINUE . . . . . BYTE AT ABSOLUTE 110 112 122 124
12			CORRECT. . . . . LITERALLY 39
12			DAY. . . . . LITERALLY 13
2	A007H	1	DAY_SIGHT. . . . . BYTE AT ABSOLUTE 119
101	FFFCH	4	DISTANCE . . . . . REAL PARAMETER AUTOMATIC 102 105
10			DONT_FIRE. . . . . LITERALLY 133
12			EAST . . . . . LITERALLY 16
2	A00FH	1	EAST_WEST. . . . . BYTE AT ABSOLUTE 126
168	0586H	379	EPILOG . . . . . PROCEDURE PUBLIC STACK=0028H
12			FIRE . . . . . LITERALLY 31
138	041FH	19	FIRE_AGAIN . . . . PROCEDURE PUBLIC STACK=0012H
12			FIRE_STUPID. . . . LITERALLY 32
			FIX. . . . . BUILTIN 105
12			FROM . . . . . LITERALLY 34
146	FFFCH	4	GAEH . . . . . REAL PARAMETER AUTOMATIC 147 148 160 162
11			GAEH_LIMIT . . . . LITERALLY 148 160
146	FFF8H	4	GAEV . . . . . REAL PARAMETER AUTOMATIC 147 148 153 155
11			GAEV_LIMIT . . . . LITERALLY 148 153
4	0000H	1	GUIDANCE_LOST. . . BYTE EXTERNAL(0) 192
12			HIGH_. . . . . LITERALLY 23 38
4	0000H	1	HILL_IMPACT. . . . BYTE EXTERNAL(1) 192
12			HIT. . . . . LITERALLY 21
12			HUNDRED. . . . . LITERALLY 36
3	A022H	22	H_MIS_ASCII. . . . BYTE ARRAY(22) AT ABSOLUTE 183 188
8	0000H	2	I. . . . . WORD 49 50 51 52
			INPUT. . . . . BUILTIN 133
48	0000H	1	ITEM . . . . . BYTE BASED(PHRASE_PTR) ARRAY(1) 51
8	0005H	1	K. . . . . BYTE
5	0000H	4	LEFT . . . . . REAL EXTERNAL(4) 186 188
12			LEFT_. . . . . LITERALLY 25
12			LONG_PAUSE . . . . LITERALLY 30
12			LOW_ . . . . . LITERALLY 37
12			MEDIUM_PAUSE . . . LITERALLY 15 16 17
12			METER. . . . . LITERALLY 18 26
12			MISS . . . . . LITERALLY 22
101	028BH	92	MISS_DISTANCE. . . PROCEDURE STACK=0022H 178 183 188 195
12			MY . . . . . LITERALLY 19

```
12                    NIGHT. . . . . LITERALLY       14
12                    ON . . . . . LITERALLY
40  0004H     1       OUTDATA. . . . . BYTE PARAMETER AUTOMATIC       41   44
                      OUTPUT . . . . . BUILTIN       115  116  117
13  0014H    15       PHRASE_1 . . . . . BYTE ARRAY(15) DATA       120
22  0077H     6       PHRASE_10. . . . . BYTE ARRAY(6) DATA        175  194
23  007DH     6       PHRASE_11. . . . . BYTE ARRAY(6) DATA        179
24  0083H     3       PHRASE_12. . . . . BYTE ARRAY(3) DATA        163  184
25  0086H     3       PHRASE_13. . . . . BYTE ARRAY(3) DATA        164  189
26  0089H     3       PHRASE_14. . . . . BYTE ARRAY(3) DATA        104
27  008CH     3       PHRASE_15. . . . . BYTE ARRAY(3) DATA        106
28  008FH     3       PHRASE_16. . . . . BYTE ARRAY(3) DATA         99
29  0092H     3       PHRASE_17. . . . . BYTE ARRAY(3) DATA         94
30  0095H     3       PHRASE_18. . . . . BYTE ARRAY(3) DATA        107  143
31  0098H     3       PHRASE_19. . . . . BYTE ARRAY(3) DATA        135
14  0023H    15       PHRASE_2 . . . . . BYTE ARRAY(15) DATA       121
32  009BH     3       PHRASE_20. . . . . BYTE ARRAY(3) DATA        139
33  009EH     3       PHRASE_21. . . . . BYTE ARRAY(3) DATA        144
34  00A1H     9       PHRASE_22. . . . . BYTE ARRAY(9) DATA        196
35  00AAH     3       PHRASE_23. . . . . BYTE ARRAY(3) DATA         63
36  00ADH     3       PHRASE_24. . . . . BYTE ARRAY(3) DATA         70
37  00B0H     3       PHRASE_25. . . . . BYTE ARRAY(3) DATA        157
38  00B3H     3       PHRASE_26. . . . . BYTE ARRAY(3) DATA        156
39  00B6H     3       PHRASE_27. . . . . BYTE ARRAY(3) DATA        150
15  0032H    12       PHRASE_3 . . . . . BYTE ARRAY(12) DATA       125
16  003EH     6       PHRASE_4 . . . . . BYTE ARRAY(6) DATA        127
17  0044H     6       PHRASE_5 . . . . . BYTE ARRAY(6) DATA        128
18  004AH    18       PHRASE_6 . . . . . BYTE ARRAY(18) DATA       129
19  005CH    15       PHRASE_7 . . . . . BYTE ARRAY(15) DATA       131
20  006BH     9       PHRASE_8 . . . . . BYTE ARRAY(9) DATA        142  169
21  0074H     3       PHRASE_9 . . . . . BYTE ARRAY(3) DATA        172
46  0004H     2       PHRASE_LENGTH. . . WORD PARAMETER AUTOMATIC       47   50
46  0006H     4       PHRASE_PTR . . . . POINTER PARAMETER AUTOMATIC        47   48   51
12                    POINT. . . . . LITERALLY       29
 9                    PORT_B . . . . . LITERALLY     133
 9                    PORT_C . . . . . LITERALLY     115  116  117
109 02E7H   312       PROLOG . . . . . PROCEDURE PUBLIC STACK=0012H
41  D000H     1       P_DATA . . . . . BYTE AT ABSOLUTE          44
141 0432H    47       QUIT . . . . . . PROCEDURE PUBLIC STACK=0012H
 5  0000H     4       RIGHT. . . . . . REAL EXTERNAL(3)         181  183
12                    RIGHT_ . . . . . LITERALLY       24
55  0135H   342       SAY_AMOUNT . . . . PROCEDURE STACK=0016H          103
40  00DEH    44       SEND . . . . . . PROCEDURE STACK=0004H           51   60   61   62   67   68   69   72
                       76   77   80   83   86   87   88   91   95   96   97
 5  0000H     4       SHORT. . . . . . REAL EXTERNAL(6)         192  195
12                    SHORT_PAUSE. . . . LITERALLY       18   20   28   34
                      SHR. . . . . . . BUILTIN       42  133
12                    SIGHT. . . . . . LITERALLY       13   14
                      SIZE . . . . . . BUILTIN       63   70   94   99  104  106  107  120  121  125  127
                       128  129  131  135  139  142  143  144  150  156  157  163  164  169
                       172  175  179  184  189  194  196
 7  0003H     1       SPEECH_HIT . . . . BYTE PUBLIC         171
 7  0004H     1       SPEECH_MISS. . . . BYTE PUBLIC         173
12                    SQUAD. . . . . . LITERALLY       15
12                    SS . . . . . . . LITERALLY       18   27
 2  A00CH     1       STARTING_TRACK . . BYTE AT ABSOLUTE         113
41  D002H     1       STAT_COM . . . . . BYTE AT ABSOLUTE          42
```

```
 12                TANK . . . . . .   LITERALLY        15
 12                TARGET . . . . . .   LITERALLY        34
 12                THOUSAND . . . . .   LITERALLY        18   35
 12                THREE. . . . . .   LITERALLY        18
                   TIME . . . . . .   BUILTIN        130  132  170
  1  00DEH         TON_SPEECH_MODULE.   PROCEDURE STACK=0000H
 12                TRACKING . . . . .   LITERALLY        20
  5  0000H      4  UP . . . . . .   REAL EXTERNAL(5)          176  178
 12                USE. . . . . .   LITERALLY        13   14
 12                VERY_LONG_PAUSE. .   LITERALLY        15   22   23
 12                VERY_SHORT_PAUSE .   LITERALLY        13   14   18   19
  3  A038H     22  V_MIS_ASCII. . . .   BYTE ARRAY(22) AT ABSOLUTE          178
 12                WEST . . . . . .   LITERALLY        17
  4  0000H      1  WIRE_BROKE . . . .   BYTE EXTERNAL(2)          192
 46  010AH     43  XMIT . . . . . .   PROCEDURE STACK=000EH          63   70   94   99  104  106  107  120
                                        121  125  127  128  129  131  135  139  142  143  144  150  156  157
                                        163  164  169  172  175  179  184  189  194  196
  3  A04EH     22  X_MIS_ASCII. . . .   BYTE ARRAY(22) AT ABSOLUTE          195


MODULE INFORMATION:

     CODE AREA SIZE     = 0701H   1793D
     CONSTANT AREA SIZE = 0000H      0D
     VARIABLE AREA SIZE = 0006H      6D
     MAXIMUM STACK SIZE = 0028H     40D
     361 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION
```

```
ISIS-II PL/M-86 V2.1 COMPILATION OF MODULE TOW_UTILITY
OBJECT MODULE PLACED IN :F2:TOWUT.OBJ
COMPILER INVOKED BY:  PLM86 :F2:TOWUT.006 DEBUG ROM MEDIUM XREF IXREF WORKFILES(:F2:,:F2:) DATE(1/13/83)

   1          TOW_UTILITY: DO;

              /* TOWUT.006 HAS NEW SOUND PROCEDURE, CORRECTED MISS_COMMENT PROCEDURE,
                 AND DELETION OF ALL REFERENCES TO TIMER.                          */

   2   1      DECLARE PPI_CONTROL LITERALLY '0CEH';

              /* HX2AS CONVERTS AN INTEGER TO ASCII CHARACTERS WITH THE
                 LEAST SIGNIFICANT DIGIT IN THE TENTHS POSITION */

   3   1      HX2AS: PROCEDURE (HEX,ASCII_ADR) PUBLIC;
   4   2        DECLARE ASCII_ADR POINTER, HEX INTEGER,
                        ASCII BASED ASCII_ADR (6) BYTE, N INTEGER , REMAINDER INTEGER;
   5   2        HEX = IABS(HEX);
   6   2        DO N = 4 TO 0 BY - 1;
   7   3          REMAINDER = HEX MOD 10 + 30H;
   8   3          ASCII(N) = LOW(UNSIGN(REMAINDER));
   9   3          HEX = HEX/10;
  10   3         END;
  11   2        ASCII(5) = ASCII(4);
  12   2        ASCII(4) = '.';

  13   2        N=0;

  14   2        DO WHILE (ASCII(N) = 30H) AND (N < 3);  /* REPLACE LEADING ZEROES WITH BLANKS */
  15   3          ASCII(N) = 20H;
  16   3          N = N + 1;
  17   3         END;
  18   2       END HX2AS;

              /* CONVERTS A REAL NUMBER TO ASCII CHARACTERS WITH APROPRIATE DIRECTION */

  19   1      MISS_COMMENT: PROCEDURE(REAL$ADR, DEC$ADR, DIRECTION) PUBLIC;
  20   2      DECLARE(REAL$ADR, DEC$ADR) POINTER, TEMP1 BASED REAL$ADR REAL;
  21   2      DECLARE PHRASE BASED DEC$ADR (22) BYTE, (N,DIRECTION) BYTE;
  22   2      DECLARE TEMP2 INTEGER;

  23   2      TEMP2 = FIX(10.0 * TEMP1);        /* GET HEX MISS DISTANCE IN DECIMETERS */
  24   2      IF TEMP2 <> 0                     /* DO NOTHING IF TEMP2 = 0.0 */
                 THEN DO;
  26   3                CALL HX2AS(TEMP2,DEC$ADR);
  27   3                PHRASE(7)  = ' ';
  28   3                PHRASE(8)  = 'M';
  29   3                PHRASE(9)  = 'E';
  30   3                PHRASE(10) = 'T';
  31   3                PHRASE(11) = 'E';
  32   3                PHRASE(12) = 'R';
  33   3                PHRASE(13) = '(';
  34   3                PHRASE(14) = 'S';
  35   3                PHRASE(15) = ')';
  36   3                PHRASE(16) = ' ';
```

```
37  3           DO CASE DIRECTION;
38  4               DO;
39  5                 PHRASE(17) = 'R';
40  5                 PHRASE(18) = 'I';
41  5                 PHRASE(19) = 'G';
42  5                 PHRASE(20) = 'H';
43  5                 PHRASE(21) = 'T';
44  5                 END;
45  4               DO;
46  5                 PHRASE(17) = 'L';
47  5                 PHRASE(18) = 'E';
48  5                 PHRASE(19) = 'F';
49  5                 PHRASE(20) = 'T';
50  5                 END;
51  4               DO;
52  5                 PHRASE(17) = 'M';
53  5                 PHRASE(18) = 'I';
54  5                 PHRASE(19) = 'S';
55  5                 PHRASE(20) = 'S';
56  5                 END;
57  4               ;
58  4               DO;
59  5                 PHRASE(17) = 'S';
60  5                 PHRASE(18) = 'H';
61  5                 PHRASE(19) = 'O';
62  5                 PHRASE(20) = 'R';
63  5                 PHRASE(21) = 'T';
64  5                 END;
65  4               END;
66  3             END;
67  2     END MISS_COMMENT;

68  1     SOUND: PROCEDURE (WHAT_KIND) PUBLIC;
69  2           DECLARE WHAT_KIND BYTE;
70  2           DECLARE PORT_A LITERALLY '0C8H';

           /* PORT_A BIT2 IS INT, AN ACTIVE HIGH STROBE FOR DATA ON PORT_A BITS
              0 & 1. THE SOUND SIGNALS ARE ENCODED ON PORT_A BITS 0 & 1. THIS DATA
              IS VALID ONLY WHILE BIT 2 IS HIGH. NOTE THAT PORT_A OUTPUTS ARE
              INVERTED BEFORE THEY REACH THE CONNECTOR. BIT 2 IS HIGH FOR 100
              MICROSECONDS. THE SOUND SIGNALS ARE ENCODED AS FOLLOWS:

                        WHAT_KIND = 0  => TRIGGER PULL SIGNAL (A1 = 0, A0 = 0)
                        WHAT_KIND = 1  => IMPACT HIT SIGNAL   (A1 = 0, A0 = 1)
                        WHAT_KIND = 2  => IMPACT MISS SIGNAL  (A1 = 1, A0 = 0)
                                                                          */

71  2           OUTPUT(PORT_A) = 04H OR (WHAT_KIND AND 03H);
72  2           CALL TIME(1);
73  2           OUTPUT(PORT_A) = 00H;
74  2           END SOUND;

75  1     PPI_SET: PROCEDURE PUBLIC;
76  2       DECLARE PPI_MODE LITERALLY '82H';     /* PORTS A & C OUTPUT, B INPUT, */
                              /* REF PAGES 2-10 & 3-15 */
77  2       OUTPUT(PPI_CONTROL) = PPI_MODE;   /* ALL PPI OUTPUTS GO LOW */
```

```
78  2       END PPI_SET;

79  1     END TOW_UTILITY;
```

```
CROSS-REFERENCE LISTING
-----------------------
 DEFN  ADDR   SIZE  NAME, ATTRIBUTES, AND REFERENCES
-----  ------ -----  --------------------------------

 4  0000H     6  ASCII. . . . . .   BYTE BASED(ASCII_ADR) ARRAY(6)       8  11  12  14  15
 3  0006H     4  ASCII_ADR. . . . .  POINTER PARAMETER AUTOMATIC         4   8  11  12  14  15
19  0008H     4  DECADR . . . . . .  POINTER PARAMETER AUTOMATIC        20  21  26  27  28  29  30
                                       31  32  33  34  35  36  39  40  41  42  43  46  47  48
                                       49  52  53  54  55  59  60  61  62  63
19  0006H     1  DIRECTION. . . . .  BYTE PARAMETER AUTOMATIC          21  37
               FIX. . . . . .      BUILTIN          23
 3  000AH     2  HEX. . . . . .     INTEGER PARAMETER AUTOMATIC        4   5   7   9
 3  0004H   157  HX2AS. . . . . .    PROCEDURE PUBLIC STACK=000AH       26
               IABS . . . . . .    BUILTIN           5
               LOW. . . . . .      BUILTIN           8
19  00A1H   312  MISS_COMMENT . . .  PROCEDURE PUBLIC STACK=001AH
 4  0000H     2  N. . . . . .      INTEGER           6   8  13  14  15  16
21  0006H     1  N. . . . . .      BYTE
               OUTPUT . . . . . .  BUILTIN          71  73  77
21  0000H    22  PHRASE . . . . . .  BYTE BASED(DECADR) ARRAY(22)       27  28  29  30  31  32  33
                                       34  35  36  39  40  41  42  43  46  47  48  49  52  53
                                       54  55  59  60  61  62  63
70             PORT_A . . . . . .  LITERALLY        71  73
 2             PPI_CONTROL. . . .  LITERALLY        77
76             PPI_MODE . . . . .  LITERALLY        77
75  01FDH     9  PPI_SET. . . . . .  PROCEDURE PUBLIC STACK=0002H
19  000CH     4  REALADR. . . . . .  POINTER PARAMETER AUTOMATIC       20  23
 4  0002H     2  REMAINDER. . . . .  INTEGER           7   8
68  01D9H    36  SOUND. . . . . .    PROCEDURE PUBLIC STACK=0004H
20  0000H     4  TEMP1. . . . . .    REAL BASED(REALADR)             23
22  0004H     2  TEMP2. . . . . .    INTEGER          23  24  26
               TIME . . . . . .    BUILTIN          72
 1  0004H        TOW_UTILITY. . . .  PROCEDURE STACK=0000H
               UNSIGN . . . . . .  BUILTIN           8
68  0006H     1  WHAT_KIND. . . . .  BYTE PARAMETER AUTOMATIC       69  71

MODULE INFORMATION:

     CODE AREA SIZE     = 0206H    518D
     CONSTANT AREA SIZE = 0000H      0D
     VARIABLE AREA SIZE = 0007H      7D
     MAXIMUM STACK SIZE = 001AH     26D
     111 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION
```

```
ISIS-II MCS-86 MACRO ASSEMBLER V2.1 ASSEMBLY OF MODULE RDRSB
OBJECT MODULE PLACED IN :F2:TOWXF.OBJ
ASSEMBLER INVOKED BY:  ASM86 :F2:TOWXF.SRC DEBUG XREF DATE(1/17/83)

LOC  OBJ                  LINE     SOURCE

                            1     ;THIS PROGRAM, STARTED SEPT 5, 1979, READS DATA FROM THE RETICON RSB6020
                            2     ; INTERFACE BOARD INTO THE 86/12 MEMORY: REFERENCES ARE:
                            3     ;       1. RSB6020 OPERATING INSTRUCTIONS, MARCH 9, 1979, EG&G RETICON
                            4     ;          SUNNYVALE, CALIFORNIA.
                            5     ;       2. MCS-86 ASSEMBLE LANGUAGE REFERENCE MANUAL, #9800640A,
                            6     ;          INTEL CORP, SANTA CLARA, CALIFORNIA
                            7
                            8     ;IT IS BEING CLEANED UP A BIT FEB 20, 1981
                            9
                           10     ;EQUATES AT TOP OF PROGRAM PER P. 8-1, REF 2
                           11
                           12             NAME    RDRSB
                           13     DGROUP  GROUP   SBC_REGS, RSB_REGS
                           14     CGROUP  GROUP   CODE
                           15
  0001                     16     INMSK   EQU     01      ;SET UP FOR CAMERA 1 ONLY. SEE P. 19, REF 1
  0064                     17     LINES   EQU     100
  0010                     18     ENDFR   EQU     10H     ;MASK FOR THE 2-TO-THE-4TH BIT, P. 45, REF 1
                           19
                           20     ASSUME SS:DGROUP, CS:CGROUP, DS:DGROUP, ES:DGROUP
                           21
----                       22     SBC_REGS SEGMENT COMMON         ;NOTE THAT 'COMMON' FILES MUST BE
                           23                                     ;COMMON IN ALL MODULES, I.E. CAN'T BE
0000 (800                  24     SBCREG  DB      800 DUP (?)     ;'AT' IN ONE AND 'COMMON' IN ANOTHER.
     ??
     )
                           25                                     ;THEY DO NOT, HOWEVER, HAVE TO BE POINTED
                           26                                     ;TO BY THE SAME SEGMENT REGISTER IN BOTH
                           27                                     ;MODULES, NOR DO THEY HAVE TO
                           28                                     ;OF THE SAME LENGTH.
0320 (6                    29     PARTLY_OFF      DB      6 DUP (?)       ; INITIALEZE = 0
     ??
     )
0326 (6                    30     LOCATIONS       DB      6 DUP (?)       ; INITIALIZE = 1
     ??
     )
----                       31     SBC_REGS ENDS
                           32
----                       33     RSB_REGS SEGMENT AT 0E00H       ;BASE ADDRESS OF RETICON BOARD IS 0E000H
0000 (512                  34     RSBDTA  DB      200H DUP (?)
     ??
     )
0200 (1                    35     STAT1   DB      1 DUP (?)       ;THIS FORM IS NECESSARY TO AVOID LOADING ERRORS
     ??
     )
0201 (11                   36     STAT2   DB      0BH DUP (?)
     ??
     )
020C (2                    37     RESET   DB      2 DUP(?)
     ??
```

```
LOC  OBJ                  LINE     SOURCE
     )
020E (1                     38     CNFG15  DB      1 DUP (?)
     ??
     )
020F (1                     39     PROCOM  DB      1 DUP (?)
     ??
     )
----                        40     RSB_REGS ENDS
                            41
----                        42     STACK   SEGMENT STACK   'STACK'
0000 (10                    43             DW      10 DUP (?)
     ????
     )
0014                        44     STKTOP  LABEL   WORD
----                        45     STACK   ENDS
                            46
                            47
----                        48     CODE    SEGMENT PUBLIC  'CODE'
                            49
                            50             PUBLIC  INIT1
0000                        51     INIT1   PROC    NEAR            ;INITIALIZATION OF RSB 6020 INTERFACE BOARD
0000 A20C02          R      52     RTINIT: MOV     RESET,AL        ;RESET IS A 'DUMMY' REGISTER, ALL IT NEEDS
0003 C6060E0201      R      53             MOV     CNFG15,INMSK    ;IS THE 'MWTC/' PULSE FROM P1 #20
0008 A20F02          R      54             MOV     PROCOM,AL       ;PROCOM IS ALSO A 'DUMMY' REGISTER.
000B C3                     55             RET
                            56     INIT1   ENDP
                            57
                            58             PUBLIC  RD_RAST
000C                        59     RD_RAST PROC    NEAR
000C A20F02          R      60     LAST:   MOV     PROCOM,AL       ;WILL WAIT FOR LAST RASTER LINE
000F A00002          R      61     WTLP0:  MOV     AL,STAT1        ;FROM HERE TO CHECK IS ST'D NUCMCY
0012 D0E0                   62             SHL     AL,1
0014 7303                   63             JNB     WTLP3
0016 A20F02          R      64     PSPR0:  MOV     PROCOM,AL
0019 A00002          R      65     WTLP3:  MOV     AL,STAT1
001C D0E0                   66             SHL     AL,1
001E 7206                   67             JB      CHECK
0020 D0E0                   68             SHL     AL,1
0022 72F5                   69             JB      WTLP3
0024 EBF0                   70             JMP     PSPR0
0026 A00300          R      71     CHECK:  MOV     AL,RSBDTA[3]
0029 2410                   72             AND     AL,ENDFR
002B 74DF                   73             JZ      LAST
                            74
                            75     ;HAVING FOUND THE LAST LINE OF THE FRAME, WE WILL TRANSFER THE RETICON
                            76     ;DATA, LINE-BY-LINE THE THE INTEL 86/12 BOARD. EACH LINE TRANSFERRED STARTS
                            77     ;WITH A 'NEW COMMAND CYCLE' AS PER PAGE 40 OF REFERENCE 1.^
                            78
002D B364                   79             MOV     BL,LINES                ;WILL DECREMENT FROM 100 TO ZERO
002F BF0000          R      80             MOV     DI,OFFSET DGROUP:SBCREG
                            81
0032 A20F02          R      82     NUCMCY: MOV     PROCOM,AL       ;AGAIN, PROCOM IS A DUMMY
0035 A00002          R      83     WTLP1:  MOV     AL,STAT1
0038 D0E0                   84             SHL     AL,1
003A 7303                   85             JNB     WTLP
```

```
LOC  OBJ                  LINE     SOURCE

003C A20F02         R       86     PSPR:   MOV     PROCOM,AL
003F A00002         R       87     WTLP:   MOV     AL,STAT1
0042 D0E0                   88             SHL     AL,1
0044 7206                   89             JB      OTLP
0046 D0E0                   90             SHL     AL,1
0048 72F5                   91             JB      WTLP
004A EBF0                   92             JMP     PSPR
                            93
                            94     ;NOW TRANSFER A SINGLE LINE OF RETICON RSB 6020 DATA TO THE INTEL 86 BOARD.
                            95
004C A00000         R       96     OTLP:   MOV     AL,RSBDTA               ;TRANSITIONS IN THE LINE
004F 32E4                   97             XOR     AH,AH
0051 40                     98             INC     AX                      ;NUMBER OF WORDS TO XFER
0052 D1E0                   99             SHL     AX,1                    ;X2 NUMBER OF BYTES TO XFER
0054 8BC8                  100             MOV     CX,AX
0056 BE0000         R      101             MOV     SI,OFFSET DGROUP:RSBDTA
0059 F3                    102     REP     MOVS    BYTE PTR SBCREG [DI], BYTE PTR RSBDTA [SI]
005A A4
005B FECB                  103             DEC     BL
005D 75D3                  104             JNZ     NUCMCY
                           105
005F C3                    106             RET
                           107
                           108     RD_RAST ENDP
                           109
----                       110     CODE    ENDS
                           111
                           112             END
```

```
XREF SYMBOL TABLE LISTING
---- ------ ----- -------


NAME         TYPE     VALUE  ATTRIBUTES, XREFS

??SEG . . . SEGMENT         SIZE=0000H PARA PUBLIC
CGROUP. . . GROUP           CODE   14# 20
CHECK . . . L NEAR   0026H  CODE  67 71#
CNFG15. . . V BYTE   020EH  RSB_REGS  38# 53
CODE. . . . SEGMENT         SIZE=0060H PARA PUBLIC 'CODE'  14# 48 110
DGROUP. . . GROUP           SBC_REGS RSB_REGS   13# 20 20 20 80 101
ENDFR . . . NUMBER   0010H    18# 72
INIT1 . . . L NEAR   0000H  CODE PUBLIC  50 51# 56
INMSK . . . NUMBER   0001H    16# 53
LAST. . . . L NEAR   000CH  CODE  60# 73
LINES . . . NUMBER   0064H    17# 79
LOCATIONS . V BYTE   0326H  SBC_REGS  30#
NUCMCY. . . L NEAR   0032H  CODE  82# 104
OTLP. . . . L NEAR   004CH  CODE  89 96#
PARTLY_OFF. V BYTE   0320H  SBC_REGS  29#
PROCOM. . . V BYTE   020FH  RSB_REGS  39# 54 60 64 82 86
PSPR. . . . L NEAR   003CH  CODE  86# 92
PSPRO . . . L NEAR   0016H  CODE  64# 70
RD_RAST . . L NEAR   000CH  CODE PUBLIC  58 59# 108
RESET . . . V BYTE   020CH  RSB_REGS  37# 52
RSB_REGS. . SEGMENT         SIZE=0210H PARA ABS   13# 33 40
RSBDTA. . . V BYTE   0000H  RSB_REGS  34# 71 96 101 102
RTINIT. . . L NEAR   0000H  CODE  52#
SBC_REGS. . SEGMENT         SIZE=032CH PARA COMMON   13# 22 31
SBCREG. . . V BYTE   0000H  SBC_REGS  24# 80 102
STACK . . . SEGMENT         SIZE=0014H PARA STACK 'STACK'
STAT1 . . . V BYTE   0200H  RSB_REGS  35# 61 65 83 87
STAT2 . . . V BYTE   0201H  RSB_REGS  36#
STKTOP. . . V WORD   0014H  STACK  44#
WTLP. . . . L NEAR   003FH  CODE  85 87# 91
WTLP0 . . . L NEAR   000FH  CODE  61#
WTLP1 . . . L NEAR   0035H  CODE  83#
WTLP3 . . . L NEAR   0019H  CODE  63 65# 69


ASSEMBLY COMPLETE, NO ERRORS FOUND
```

```
 ISIS-II MCS-86 MACRO ASSEMBLER V2.1 ASSEMBLY OF MODULE IR_CENTER
 OBJECT MODULE PLACED IN :F2:TOWIR.OBJ
 ASSEMBLER INVOKED BY:  ASM86 :F2:TOWIR.003 DEBUG XREF DATE (01/14/83)
LOC  OBJ                  LINE     SOURCE

                            1     NAME    IR_CENTER
                            2
                            3     EXTRN   START_UP: FAR
                            4
                            5     ;THIS IS A DRIVER PROGRAM FOR THE GUNNER AIMING ERROR MATRIX CAMERA, RETICON
                            6     ;RSB 6020 USING THE 86/12A BOARD. COMMENTS ARE UPDATED 9/23/81.
                            7
                            8     DGROUP  GROUP   STACK, SBC_REGS, XFER_SEG
                            9     CGROUP  GROUP   CODE
                           10
                           11     ASSUME  SS:DGROUP, CS:CGROUP, DS:DGROUP, ES:DGROUP
                           12
----                       13     STACK   SEGMENT STACK   'STACK'
0000 (64                   14             DW      64 DUP(?)
     ????
     )
0080                       15     TOP_STK LABEL   WORD
----                       16     STACK   ENDS
                           17
                           18     ;THIS PROGRAM WILL RESIDE ON THE SBC 'DFS'. IT WILL WRITE Y, Z(CENTER)
                           19     ;DATA TO THE SBC 'PIP' VIA THE MULTIBUS. THE 'PIP' HAS BEEN 'JUMPERED'
                           20     ;SO AS TO ALLOW THE MULTIBUS TO ACCESS 8K OF ITS RAM STARTING AT LOCATION
                           21     ;A000H. THE 'DFS' JUMPERS ALLOW MULTIBUS ACCESS TO 8K OF RAM STARTING
                           22     ;AT 8000H. THE ON-BOARD LOCATION OF THESE AVAILABLE 8K-S START AT 6000H
                           23     ;ON BOTH BOARDS. THE BOARDS (REF FIG 2-1 86/12) ARE JUMPERED AS FOLLOWS:
                           24     ;
                           25     ;  SBC DFS JUMPERS MULTIBUS ACCESS :  SBC PIP JUMPERS MULTIBUS ACCESS
                           26     ;    -------------------------     :    -------------------------
                           27     ;       127-128 ==> X = 0          :          127-128 ==> X = 0
                           28     ;    S1 6-11 CLOSED                :       S1 6-11 CLOSED
                           29     ;    S1 5-12   "    ==> 8K         :       S1 5-12   "    ==> 8K
                           30     ;    S1 1-16   "                   :       S1 1-16   "
                           31     ;    S1 2-15 OPEN                  :       S1 2-15 OPEN
                           32     ;    S1 3-14 CLOSED                :       S1 3-14 CLOSED
                           33     ;    S1 4-13   "    ==> 8000H      :       S1 4-13 OPEN  ==> A000H
                           34     ;
                           35
----                       36     XFER_SEG        SEGMENT AT 0A00H        ;ON SBC 'PIP' AS NOTED ABOVE
                           37                                             ;WILL PASS DATA FOR YCNTR,ZCNTR.
                           38
0000 (1                    39     START_BIT       DB      1 DUP(?)
     ??
     )
                           40             PUBLIC  B_Y,B_Z,DATA_RDY1
0001 (1                    41     B_Y             DB      1 DUP(?)
     ??
     )
0002 (1                    42     B_Z             DB      1 DUP(?)
     ??
     )
```

```
LOC  OBJ       LINE     SOURCE

0003 (1        43       DATA_RDY1       DB      1 DUP(?)
     ??
     )
0004 (1        44       BAD_MISS        DB      1 DUP(?)
     ??
     )
0005 (1        45       OFFSET_Y        DB      1 DUP(?)
     ??
     )
0006 (1        46       OFFSET_Z        DB      1 DUP(?)
     ??
     )
----           47       XFER_SEG        ENDS
               48
----           49       SBC_REGS        SEGMENT COMMON
0000 (800      50       SBCREG  DB      800 DUP(?)
     ??
     )
0320 (1        51       MISS    DB      1 DUP(?)              ;THE FOLLOWING 6 BYTES ARE NAMED "PARTLY_OFF"
                         IN RT86XF
     ??
     )
               52
0321 (1        53       RIGHT   DB      1 DUP(?)
     ??
     )
0322 (1        54       LEFT    DB      1 DUP(?)
     ??
     )
0323 (1        55       UP      DB      1 DUP(?)
     ??
     )
0324 (1        56       DOWN    DB      1 DUP(?)
     ??
     )
0325 (1        57                DB      1 DUP(?)        ;DUMMY BYTE TO MAKE 3 WORDS OF "PARTLY_OFF" IN RT86XF
     ??
     )
0326 (1        58       YCNTR   DB      1 DUP(?)        ;THE FOLLOWING 6 BYTES ARE IN "LOCATIONS" IN RT86XF
     ??
     )
0327 (1        59       ZCNTR   DB      1 DUP(?)
     ??
     )
0328 (1        60       YMAX    DB      1 DUP(?)
     ??
     )
0329 (1        61       YMIN    DB      1 DUP(?)
     ??
     )
032A (1        62       ZMAX    DB      1 DUP(?)
     ??
     )
032B (1        63       ZMIN    DB      1 DUP(?)
     ??
```

```
LOC  OBJ                  LINE     SOURCE

     )
                           64
032C (1                    65     GAEY    DW      1 DUP(?)
     ????
     )
032E (1                    66     GAEZ    DW      1 DUP(?)
     ????
     )
0330 (1                    67     SAVGAEY DW      1 DUP(?)
     ????
     )
0332 (1                    68     SAVGAEZ DW      1 DUP(?)
     ????
     )
                           69
                           70     PUBLIC  GAEY,GAEZ
                           71
----                       72     SBC_REGS        ENDS
                           73
                           74
                           75     EXTRN   RD_RAST: NEAR
  0064                     76     SIZ     EQU     64H
----                       77     CODE    SEGMENT PUBLIC  'CODE'
                           78
                           79             PUBLIC  SETRET
0000                       80     SETRET  PROC    FAR
                           81             EXTRN   INIT1:  NEAR
0000 55                    82             PUSH    BP
0001 1E                    83             PUSH    DS
0002 B8----        R       84             MOV     AX,DGROUP
0005 8ED8                  85             MOV     DS,AX
0007 8EC0                  86             MOV     ES,AX
0009 E80000        E       87             CALL    INIT1
000C A00000        R       88     WAIT0:  MOV     AL,START_BIT  ;THIS WAIT LOOP HOLDS UP MATRIX CAMERA DATA UNTIL
                           89                                   ; THE RETRO-GRAPHICS BOARD IS SET UP AND THE
                           90                                   ; MATROX BOARD IS READY AND WAITING.
000F 3C01                  91             CMP     AL,1
0011 75F9                  92             JNE     WAIT0
0013 1F                    93             POP     DS
0014 5D                    94             POP     BP
0015 CB                    95             RET
                           96     SETRET  ENDP
                           97
                           98             PUBLIC  YZCNTR
0016                       99     YZCNTR  PROC    FAR
0016 55                   100             PUSH    BP
0017 1E                   101             PUSH    DS
0018 B8----        R      102             MOV     AX,DGROUP
001B 8ED8                 103             MOV     DS,AX
001D 8EC0                 104             MOV     ES,AX
001F E80000        E      105             CALL    RD_RAST
                          106
                          107     ;WE NOW LOOK FOR THOSE TRANSITIONS WHICH CAN BE ASSOCIATED WITH A SINGLE
                          108     ;BRIGHT SPOT ON THE RETICON CAMERA 100X100 FIELD OF VIEW. THE SPOT SOURCE
                          109     ;IS LOCATED AT A FIXED POSITION RELATIVE TO THE CENTER OF THE TARGET.
```

```
LOC  OBJ                  LINE     SOURCE

                          110      ;THE RETICON CAMERA IS MOUNTED ON THE CENTER LINE OF THE SIMULATED WEAPON
                          111      ;SO THAT THE OFFSET OF THE SPOT FROM THE CENTER OF THE 100X100 FIELD OF VIEW,
                          112      ;WHEN CORRECTED BY AUTOBORESIGHT, WILL MEASURE THE LEAD AND ELEVATION
                          113      ;OF THE WEAPON AT TRIGGER-PULL TIME. A SAMPLE RETICON DATA LINE CONTAINING A
                          114      ;SINGLE BRIGHT SPOT IS:
                          115      ;
                          116      ;      03 03 1C 60 1E E0 64 60
                          117      ;
                          118      ;THIS INDICATES THREE TRANSITIONS IN THE LINE BUT ONLY THE FIRST TWO ARE
                          119      ;SIGNIFICANT. AT 1CH THERE IS A DARK-TO-LIGHT TRANSITION, AS INDICATED BY THE
                          120      ;HIGHEST ORDER BIT (HOB) = 0 IN DATA BYTE #3 (60). THE FOLLOWING TRANSITION
                          121      ;IS AT 1EH AND IS A LIGHT-TO-DARK, AS INDICATED BY HOB=1 IN THE FOLLOWING
                          122      ;BYTE #5 (E0). EVERY LINE HAS A FORCED TRANSITION AT THE END-OF-LINE.
                          123      ;LOCATION 64H=100D. THE PROGRAM STARTS FROM LINE #0 LOOKING FOR 03 AS THE
                          124      ;INITIAL DATA BYTE. IF 03 IS NOT FOUND, THE NEXT LINE IS EXAMINED. THIS IS
                          125      ;CONTINUED UNTIL 100 LINES HAVE BEEN EXAMINED FOR THE PROPER NUMBER OF
                          126      ;TRANSITIONS.
                          127
                          128      ;RESISTER USAGE IN 'CENTER'
                          129      ;       (AL) = NUMBER OF TRANSITIONS IN DATA LINE
                          130      ;       (AH) = BYTES TO ADD TO DATA LINE POINTER
                          131      ;       (BX) = DATA LINE POINTER, POINTS TO START OF DATA LINE IN 'SBCREG'
                          132      ;       (CL) = LINE NUMBER
                          133      ;       (CH) = 64H = 100 ==> THE LAST DATA LINE
                          134
0022 B90165               135      CENTER: MOV     CX,6501H        ;(CH)=65H,1 + LAST LINE, (CL)=1;--> FIRST LINE
0025 BBFEFF               136              MOV     BX,-2           ;INITIAL VALUE OF DATA LINE POINTER
0028 B402                 137              MOV     AH,2            ;INITIAL DATA LINE POINTER INCREMENT
002A C606290364     R     138              MOV     YMIN,SIZ        ;SET INITIAL VALUE AT 64H
002F C6062B0364     R     139              MOV     ZMIN,SIZ        ;DITTO FOR ZMIN
0034 C6062B0301     R     140              MOV     YMAX,1
0039 C6062A0301     R     141              MOV     ZMAX,1
                          142
003E 3AE9                 143      DUMP:   CMP     CH,CL           ;HAVE WE FINISHED WITH LAST LINE?
0040 7503                 144              JNE     OVER            ;NEED 'OVER' BECAUSE CONDITIONAL JUMPS MUST BE
                          145                                      ;LESS THAN +127 BYTES AWAY
0042 E98500               146              JMP     DONE
0045 02DC                 147      OVER:   ADD     BL,AH           ;UPDATE DATA LINE POINTER, NOW BECAUSE WE CANNOT
0047 80D700               148              ADC     BH,0            ;ADD A SINGLE BYTE TO BX, WE DO IT IN TWO STEPS
                          149                                      ;USING THE CARRY FLAG, 'CY'. N.B. (BX)=0 ON
                          150                                      ;THE FIRST PASS THROUGH 'DUMP.'
                          151
004A 8A870000       R     152              MOV     AL,SBCREG[BX]   ;FIRST DATA BYTE ==> TRANSITIONS IN DATA LINE
                          153
004E 8AE0                 154              MOV     AH,AL           ;WILL FORM DATA LINE POINTER INCREMENT IN AH
0050 FEC4                 155              INC     AH
0052 D0E4                 156              SHL     AH,1            ;(AH)=2(AL+1), THE DATA LINE POINTER INCREMENT
                          157
0054 8A870200       R     158              MOV     AL,SBCREG[BX+2]
0058 3C64                 159              CMP     AL,SIZ
005A 7405                 160              JE      SKIP            ;IF NO SPOT, THEN GO TO NEXT DATA LINE
005C E80A00               161              CALL    GOODLN          ;WILL UPDATE SPOT INFORMATION
005F EBD0                 162              JMP     DUMP            ;GO TO NEXT DATA LINE
                          163
0061 3ACD                 164      SKIP:   CMP     CL,CH           ;THE LAST LINE?
```

```
LOC  OBJ                  LINE    SOURCE

0063 7465                 165             JZ      DONE            ;YES! SO WE JUMP TO THE FINAL CLEAN-UP
0065 FEC1                 166             INC     CL              ;NO! SO WE RETURN TO "DUMP" AND
0067 EB05                 167             JMP     DUMP            ;EXAMINE THE NEXT LINE
                          168
0069                      169     GOODLN  PROC    NEAR
0069 80BF000002      R    170             CMP     SBCREG[BX],2    ;ONLY TWO TRANSITIONS?
006E 7419                 171             JE      TWOX            ;IF SO, SPOT IS ON RIGHT EDGE
0070 3A062903        R    172             CMP     AL,YMIN         ;IF NOT, GET NORMAL CENTER. RECALL THAT
                          173                                     ;AL CONTAINS SBCREG[BX+2]
0074 7703                 174             JA      N1              ;JUMP IF (AL) IS ABOVE YMIN, IE. CY FLAG = 0
                          175
                          176     ;RECALL THAT A CMP OPERATION SUBTRACTS THE SOURCE OR 2ND OPERAND FROM THE
                          177     ;DESTINATION OR 1ST OPERAND. IT DOES THIS BY ADDING THE TWOS COMPLEMENT OF THE
                          178     ;SOURCE OPERAND TO THE DESTINATION OPERAND AND A CARRY-OUT FROM THE HIGH ORDER
                          179     ;BIT CAUSES THE CY FLAG TO SET TO 0, BECAUSE OF THE SUBTRACTION OPERATION.
                          180     ;AT LEAST THIS IS WHAT THE 8080 DOES, AND THIS FLAG IS THE SAME. IN THE 8086
                          181     ;THEY SAY THAT AFTER A SUBTRACT OPERATION "CY" IS SET UPON A CARRY INTO(!)
                          182     ;THE HOB OF THE RESULT!
                          183
0076 A22903          R    184             MOV     YMIN,AL         ;IF NO JUMP THEN UPDATE VALUE OF YMIN
0079 8A870400        R    185     N1:     MOV     AL,SBCREG[BX+4] ;RIGHT EDGE OF SPOT
007D 3A062803        R    186             CMP     AL,YMAX
0081 7230                 187             JB      N4              ;JUMP IF (AL) IS BELOW YMAX, OR CY = 1
0083 A22803          R    188             MOV     YMAX,AL         ;IF NO JUMP THEN UPDATE YMAX
0086 EB2B90               189             JMP     N4              ;JUMP AROUND "TWOX"
0089 8A870200        R    190     TWOX:   MOV     AL,SBCREG[BX+2]
008D 80BF030080      R    191             CMP     SBCREG[BX+3],80H
0092 7211                 192             JB      REDGE           ;HOB CLEAR ==> DARK TO LIGHT TRANSITION, R EDGE
0094 C606290301      R    193             MOV     YMIN, 1         ;HOB SET ==> LIGHT TO DARK TRANSITION, L EDGE
0099 3A062803        R    194             CMP     AL, YMAX
009D 7214                 195             JB      N4
009F A22803          R    196             MOV     YMAX, AL
00A2 EB0F90               197             JMP     N4
00A5 C606280364      R    198     REDGE:  MOV     YMAX,64H        ;WILL BE ON RIGHT EDGE
00AA 3A062903        R    199             CMP     AL,YMIN
00AE 7703                 200             JA      N4
00B0 A22903          R    201             MOV     YMIN,AL
00B3 3A0E2A03        R    202     N4:     CMP     CL,ZMAX         ;AT THIS POINT Z IS MEASURED DOWNWARD
00B7 7204                 203             JB      N3              ;AND WE MUST COMPLEMENT Z AT END
00B9 880E2A03        R    204             MOV     ZMAX,CL         ;UPDATE ZMAX
00BD 3A0E2B03        R    205     N3:     CMP     CL,ZMIN
00C1 7704                 206             JA      N9
00C3 880E2B03        R    207             MOV     ZMIN,CL         ;UPDATE ZMIN
00C7 FEC1                 208     N9:     INC     CL
00C9 C3                   209             RET
                          210     GOODLN  ENDP
                          211
00CA C606200300      R    212     DONE:   MOV     MISS,0
00CF 803E2B0364      R    213             CMP     ZMIN,64H
00D4 7500                 214             JNE     N5
00D6 C606200301      R    215             MOV     MISS,1          ;ZMIN = 64H ==> NO SPOT, SO SHOT WAS A MISS!
00DB C606040001      R    216             MOV     BAD_MISS,1      ;WE ARE "REALLY FINISHED"
00E0 E98300               217             JMP     READ
00E3 803E2B0301      R    218     N5:     CMP     ZMIN,1
00E8 7505                 219             JNE     N6
```

```
LOC  OBJ                  LINE     SOURCE

00EA C606230301     R      220              MOV     UP,1            ;SPOT INCLUDED FIRST LINE ==> SHOT WAS HIGH!
00EF 803E2A0364     R      221      N6:     CMP     ZMAX,64H
00F4 7505                  222              JNE     N7
00F6 C606240301     R      223              MOV     DOWN,1          ;SPOT INCLUDED LAST LINE ==> SHOT WAS LOW
00FB 803E290301     R      224      N7:     CMP     YMIN,1
0100 7505                  225              JNE     N8
0102 C606220301     R      226              MOV     LEFT,1
0107 803E2B0364     R      227      N8:     CMP     YMAX,64H
010C 7505                  228              JNE     FINI
010E C606210301     R      229              MOV     RIGHT,1
                           230
0113 B065                  231      FINI:   MOV     AL,65H
0115 2A062A03       R      232              SUB     AL,ZMAX
0119 B465                  233              MOV     AH,65H
011B 2A262B03       R      234              SUB     AH,ZMIN
011F A22B03         R      235              MOV     ZMIN,AL
0122 88262A03       R      236              MOV     ZMAX,AH
0126 A02B03         R      237              MOV     AL,YMAX
0129 02062903       R      238              ADD     AL,YMIN
012D 8A1E0500       R      239              MOV     BL,OFFSET_Y     ;HORIZONTAL BORESIGHT OFFSET
0131 02C3                  240              ADD     AL,BL
0133 A22603         R      241              MOV     YCNTR,AL        ;YCNTR IN HALF-PIXELS FROM LEFT SIDE OF SCREEN
0136 A20100         R      242              MOV     B_Y,AL
0139 B464                  243              MOV     AH,100
013B 2AE0                  244              SUB     AH,AL
013D 8AC4                  245              MOV     AL,AH
013F 98                    246              CBW
0140 A32C03         R      247              MOV     GAEY,AX
0143 A33003         R      248              MOV     SAVGAEY,AX
0146 A02A03         R      249              MOV     AL,ZMAX
0149 02062B03       R      250              ADD     AL,ZMIN
014D 8A1E0600       R      251              MOV     BL,OFFSET_Z     ;VERTICAL BORESIGHT OFFSET
0151 02C3                  252              ADD     AL,BL
0153 A22703         R      253              MOV     ZCNTR,AL        ;ZCNTR IN HALF-PIXELS FROM BOTTOM OF SCREEN
0156 A20200         R      254              MOV     B_Z,AL
0159 B464                  255              MOV     AH,100
015B 2AE0                  256              SUB     AH,AL
015D 8AC4                  257              MOV     AL,AH
015F 98                    258              CBW
0160 A32E03         R      259              MOV     GAEZ,AX
0163 A33203         R      260              MOV     SAVGAEZ,AX
0166 803E200301     R      261      READ:   CMP     MISS,1
016B 7518                  262              JNE     NEXT1
016D A13003         R      263              MOV     AX,SAVGAEY
0170 A32C03         R      264              MOV     GAEY,AX
0173 056400                265              ADD     AX,100
0176 A20100         R      266              MOV     B_Y,AL
0179 A13203         R      267              MOV     AX,SAVGAEZ
017C A32E03         R      268              MOV     GAEZ,AX
017F 056400                269              ADD     AX,100
0182 A20200         R      270              MOV     B_Z,AL
0185 B001                  271      NEXT1:  MOV     AL,1
0187 A20300         R      272              MOV     DATA_RDY1,AL    ;THIS TELLS THE SLAVE PROCESSOR THAT NEW
                           273                                      ;DATA ARE READY.
018A 1F                    274              POP     DS
```

```
LOC  OBJ                  LINE     SOURCE

018B 50                    275              POP     BP
018C CB                    276              RET
                           277
                           278     YZCNTR   ENDP
                           279
----                       280     CODE     ENDS
                           281
                           282              END
```

XREF SYMBOL TABLE LISTING
---- ------ ----- -------

NAME	TYPE	VALUE	ATTRIBUTES, XREFS
??SEG . .	SEGMENT		SIZE=0000H PARA PUBLIC
B_Y . . .	V BYTE	0001H	XFER_SEG PUBLIC 40 41# 242 266
B_Z . . .	V BYTE	0002H	XFER_SEG PUBLIC 40 42# 254 270
BAD_MISS.	V BYTE	0004H	XFER_SEG 44# 216
CENTER. .	L NEAR	0022H	CODE 135#
CGROUP. .	GROUP		CODE 9# 11
CODE. . .	SEGMENT		SIZE=018DH PARA PUBLIC 'CODE' 9# 77 280
DATA_RDY1	V BYTE	0003H	XFER_SEG PUBLIC 40 43# 272
DGROUP. .	GROUP		STACK SBC_REGS XFER_SEG 8# 11 11 11 84 102
DONE. . .	L NEAR	00CAH	CODE 146 165 212#
DOWN. . .	V BYTE	0324H	SBC_REGS 56# 223
DUMP. . .	L NEAR	003EH	CODE 143# 162 167
FINI. . .	L NEAR	0113H	CODE 228 231#
GAEY. . .	V WORD	032CH	SBC_REGS PUBLIC 65# 70 247 264
GAEZ. . .	V WORD	032EH	SBC_REGS PUBLIC 66# 70 259 268
GOODLN. .	L NEAR	0069H	CODE 161 169# 210
INIT1 . .	L NEAR	0000H	EXTRN 81# 87
LEFT. . .	V BYTE	0322H	SBC_REGS 54# 226
MISS. . .	V BYTE	0320H	SBC_REGS 51# 212 215 261
N1. . . .	L NEAR	0079H	CODE 174 185#
N3. . . .	L NEAR	008DH	CODE 203 205#
N4. . . .	L NEAR	00B3H	CODE 187 189 195 197 200 202#
N5. . . .	L NEAR	00E3H	CODE 214 218#
N6. . . .	L NEAR	00EFH	CODE 219 221#
N7. . . .	L NEAR	00FBH	CODE 222 224#
N8. . . .	L NEAR	0107H	CODE 225 227#
N9. . . .	L NEAR	00C7H	CODE 206 208#
NEXT1 . .	L NEAR	0185H	CODE 262 271#
OFFSET_Y.	V BYTE	0005H	XFER_SEG 45# 239
OFFSET_Z.	V BYTE	0006H	XFER_SEG 46# 251
OVER. . .	L NEAR	0045H	CODE 144 147#
RD_RAST .	L NEAR	0000H	EXTRN 75# 105
READ. . .	L NEAR	0166H	CODE 217 261#
REDGE . .	L NEAR	00A5H	CODE 192 198#
RIGHT . .	V BYTE	0321H	SBC_REGS 53# 229
SAVGAEY .	V WORD	0330H	SBC_REGS 67# 248 263
SAVGAEZ .	V WORD	0332H	SBC_REGS 68# 260 267
SBC_REGS.	SEGMENT		SIZE=0334H PARA COMMON 8# 49 72
SBCREG. .	V BYTE	0000H	SBC_REGS 50# 152 158 170 185 190 191
SETRET. .	L FAR	0000H	CODE PUBLIC 79 80# 96
SIZ . . .	NUMBER	0064H	76# 138 139 159
SKIP. . .	L NEAR	0061H	CODE 160 164#
STACK . .	SEGMENT		SIZE=0080H PARA STACK 'STACK'
START_BIT	V BYTE	0000H	XFER_SEG 39# 88
START_UP.	L FAR	0000H	EXTRN 3#
TOP_STK .	V WORD	0080H	STACK 15#
TWOX. . .	L NEAR	0089H	CODE 171 190#
UP. . . .	V BYTE	0323H	SBC_REGS 55# 220
WAIT0 . .	L NEAR	000CH	CODE 88# 92
XFER_SEG.	SEGMENT		SIZE=0007H PARA ABS 8# 36 47
YCNTR . .	V BYTE	0326H	SBC_REGS 58# 241

```
NAME        TYPE     VALUE  ATTRIBUTES, XREFS

YMAX. . .  V BYTE   0328H  SBC_REGS  604 140 186 188 194 196 198 227 237
YMIN. . .  V BYTE   0329H  SBC_REGS  610 138 172 184 193 199 201 224 238
YZCNTR. .  L FAR    0016H  CODE PUBLIC  98 994 278
ZCNTR . .  V BYTE   0327H  SBC_REGS  594 253
ZMAX. . .  V BYTE   032AH  SBC_REGS  620 141 202 204 221 232 236 249
ZMIN. . .  V BYTE   0328H  SBC_REGS  630 139 205 207 213 218 234 235 250


ASSEMBLY COMPLETE, NO ERRORS FOUND
```

```
ISIS-II IXREF, V1.2
INVOKED BY:
-IXREF :F2:TOWMN.IXI, :F2:TOWFLP.IXI, :F2:TOWTAR.IXI, :F2:TOWSPC.IXI, &
:F2:TOWUT.IXI PRINT(:F2:TOWIX.REF) &
TITLE('MFS INTER-MODULE CROSS REFERENCE')

INTER-MODULE CROSS-REFERENCE LISTING
------------------------------------

NAME . . . . . ATTRIBUTES; MODULE NAMES
-----------------------------------------

ACNT . . . . . INTEGER; TOW_TARGET_MODULE  TOWFLIGHTMODULE
ACTIVE_TRACK . . . PROCEDURE; TOW_TARGET_MODULE  TOWFLIGHTMODULE
B_Y. . . . . BYTE; ** UNRESOLVED **  TOWFLIGHTMODULE
B_Z. . . . . BYTE; ** UNRESOLVED **  TOWFLIGHTMODULE
COACH. . . . . PROCEDURE; TOW_SPEECH_MODULE  TOWFLIGHTMODULE
COUNT. . . . . INTEGER; TOWFLIGHTMODULE
CURRENT_TRACK. . . BYTE; TOW_TARGET_MODULE  TOWFLIGHTMODULE  MAIN_TOW_MODULE
DATA_RDY1. . . . . BYTE; ** UNRESOLVED **  TOWFLIGHTMODULE
DCNT . . . . . INTEGER; TOW_TARGET_MODULE  TOWFLIGHTMODULE
EPILOG . . . . . PROCEDURE; TOW_SPEECH_MODULE  MAIN_TOW_MODULE
FINISHED . . . . . BYTE; TOWFLIGHTMODULE  TOW_TARGET_MODULE  MAIN_TOW_MODULE
FIRE_AGAIN . . . . PROCEDURE; TOW_SPEECH_MODULE  MAIN_TOW_MODULE
FLIGHT . . . . . PROCEDURE; TOWFLIGHTMODULE  MAIN_TOW_MODULE
GAEY . . . . . WORD; ** UNRESOLVED **  TOWFLIGHTMODULE
GAEY_F . . . . . REAL; TOWFLIGHTMODULE  MAIN_TOW_MODULE
GAEZ . . . . . WORD; ** UNRESOLVED **  TOWFLIGHTMODULE
GAEZ_F . . . . . REAL; TOWFLIGHTMODULE  TOW_TARGET_MODULE  MAIN_TOW_MODULE
GROUNDED . . . . . PROCEDURE; TOW_TARGET_MODULE  TOWFLIGHTMODULE
GUIDANCE_LOST. . . BYTE; TOWFLIGHTMODULE  TOW_SPEECH_MODULE  MAIN_TOW_MODULE
HILL_IMPACT. . . . BYTE; TOW_TARGET_MODULE  TOW_SPEECH_MODULE  TOWFLIGHTMODULE  MAIN_TOW_MODULE
HTARG. . . . . REAL; TOWFLIGHTMODULE  TOW_TARGET_MODULE
HX2AS. . . . . PROCEDURE; TOW_UTILITY
H_REPRISE. . . . . PROCEDURE; TOWFLIGHTMODULE  MAIN_TOW_MODULE
INIT87 . . . . . PROCEDURE; *** UNRESOLVED ***  TOWFLIGHTMODULE
INITIATEVAR. . . . PROCEDURE; TOWFLIGHTMODULE  MAIN_TOW_MODULE
INIT_STEPPER . . . PROCEDURE; TOW_UTILITY
LEFT . . . . . REAL; TOW_TARGET_MODULE  TOW_SPEECH_MODULE  TOWFLIGHTMODULE  MAIN_TOW_MODULE
MISSILE_Z. . . . . REAL; TOWFLIGHTMODULE  TOW_TARGET_MODULE
MISS_COMMENT . . . PROCEDURE; TOW_UTILITY  MAIN_TOW_MODULE
MQERAT2. . . . . PROCEDURE; *** UNRESOLVED ***  TOW_TARGET_MODULE
MQERCOS. . . . . PROCEDURE; *** UNRESOLVED ***  TOW_TARGET_MODULE
MQEREXP. . . . . PROCEDURE; *** UNRESOLVED ***  TOWFLIGHTMODULE
MQERSIN. . . . . PROCEDURE; *** UNRESOLVED ***  TOW_TARGET_MODULE
MQERY2X. . . . . PROCEDURE; *** UNRESOLVED ***  TOW_TARGET_MODULE
OFF_H. . . . . REAL; TOWFLIGHTMODULE  TOW_TARGET_MODULE
PPI_SET. . . . . PROCEDURE; TOW_UTILITY  MAIN_TOW_MODULE
PROLOG . . . . . PROCEDURE; TOW_SPEECH_MODULE  MAIN_TOW_MODULE
QUIT . . . . . PROCEDURE; TOW_SPEECH_MODULE  MAIN_TOW_MODULE
RECEIVE. . . . . PROCEDURE  BYTE; TOW_TARGET_MODULE
RIGHT. . . . . REAL; TOW_TARGET_MODULE  TOW_SPEECH_MODULE  TOWFLIGHTMODULE  MAIN_TOW_MODULE
SEND . . . . . PROCEDURE; TOW_TARGET_MODULE  MAIN_TOW_MODULE
SETRET . . . . . PROCEDURE; ** UNRESOLVED **  MAIN_TOW_MODULE
SHORT. . . . . REAL; TOWFLIGHTMODULE  TOW_SPEECH_MODULE  MAIN_TOW_MODULE
```

```
   SOUND. . . . . PROCEDURE;  TOW_UTILITY  TOW_TARGET_MODULE  TOWFLIGHTMODULE  MAIN_TOW_MODULE
   SPEECH_HIT . . . . BYTE;  TOW_SPEECH_MODULE  TOW_TARGET_MODULE  TOWFLIGHTMODULE
   SPEECH_MISS. . . . BYTE;  TOW_SPEECH_MODULE  TOW_TARGET_MODULE  TOWFLIGHTMODULE
   START_UP . . . . . LABEL;  MAIN_TOW_MODULE
   TARGET_DATA. . . . PROCEDURE;  TOW_TARGET_MODULE  TOWFLIGHTMODULE
   TARGET_LOC . . . . PROCEDURE;  TOW_UTILITY
   TCNT . . . . . INTEGER;  TOW_TARGET_MODULE  MAIN_TOW_MODULE
   TCOUNTO. . . . . INTEGER;  TOW_TARGET_MODULE
   THREE_SEC_FLAG . . BYTE;  TOWFLIGHTMODULE  MAIN_TOW_MODULE
   TRACK_1. . . . . PROCEDURE;  TOW_TARGET_MODULE  TOWFLIGHTMODULE
   TRACK_2. . . . . PROCEDURE;  TOW_TARGET_MODULE  TOWFLIGHTMODULE
   TRACK_3. . . . . PROCEDURE;  TOW_TARGET_MODULE  TOWFLIGHTMODULE
   UP . . . . . REAL;  TOW_TARGET_MODULE  TOW_SPEECH_MODULE  TOWFLIGHTMODULE  MAIN_TOW_MODULE
   UPDATE_COUNTS. . . PROCEDURE;  TOW_TARGET_MODULE  TOWFLIGHTMODULE
   WIRE_BROKE . . . . BYTE;  TOWFLIGHTMODULE  TOW_SPEECH_MODULE  MAIN_TOW_MODULE
   X. . . . . REAL;  TOWFLIGHTMODULE  TOW_TARGET_MODULE
   YZCNTR . . . . . PROCEDURE;  ** UNRESOLVED **  MAIN_TOW_MODULE
   Z. . . . . REAL;  TOWFLIGHTMODULE  TOW_TARGET_MODULE


**UNRESOLVED**   : IN ASSEMBLY LANGUAGE CODE, EITHER TOWXF OR TOWIR
***UNRESOLVED*** : FLOATING POINT LIBRARY PROCEDURES IN 8087.LIB AND
                   CEL87.LIB


MODULE DIRECTORY
----------------


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

   MAIN_TOW_MODULE. . TOWMN.019   TOW10
   TOWFLIGHTMODULE. . TOWFLP.018  TOW10
   TOW_SPEECH_MODULE. TOWSPC.009  TOW10
   TOW_TARGET_MODULE. TOWTAR.005  TOW10
   TOW_UTILITY. . . . TOWUT.006   TOW10
```

APPENDIX C

COMPUTER GRAPHICS AND VIDEO SUBSYSTEM PROGRAMS

```
ISIS-II PL/M-86 V2.1 COMPILATION OF MODULE TOW_START_UP_MODULE
OBJECT MODULE PLACED IN :F1:TOWST.OBJ
COMPILER INVOKED BY:  PLM86 :F1:TOWST.024 DEBUG ROM XREF OPTIMIZE(3) LARGE &
                      TITLE('0930 5 NOVEMBER 1982')

   1          TOW_START_UP_MODULE: DO;

   2   1      DECLARE SCENE_COUNT LITERALLY '06D',MAX_MENU_NO LITERALLY '01D';

   3   1      DECLARE CARR_RET LITERALLY '0DH', SPACE LITERALLY '20H',BELL LITERALLY '07H';

   4   1      DECLARE IODATA LITERALLY '0D8H',IOSTATUS LITERALLY '0DAH',MASK LITERALLY '7FH';

   5   1      DECLARE VECTOR_MODE LITERALLY '35Q',ALPHA_401Q_MODE LITERALLY '15Q',
                ADM3A_MODE LITERALLY '30Q',GRAPHICS_CLEAR LITERALLY '31Q',
                ADM3A_CLEAR LITERALLY '32Q',CLEAR_ALL LITERALLY '33Q',
                HOME_CURSOR LITERALLY '14Q';

   6   1      DECLARE COUNTER_2 LITERALLY '0D4H',CONTROL LITERALLY '0D6H',
                CNTR2MODE LITERALLY '096H',SETCOUNT LITERALLY '04H';
                /* FOR TIMER SETUP */

   7   1      DECLARE USART_CONTROL LITERALLY '0DAH',USART_MODE LITERALLY '4EH',
                USART_COMMAND LITERALLY '37H';

   8   1      DECLARE LINE_FEED LITERALLY '0AH';

   9   1      DECLARE (DAY_SIGHT,STARTING_TRACK,TARGET_SWITCH,FINAL_TRACK,
                EAST_WEST,CONTINUE) BYTE EXTERNAL;

  10   1      DECLARE (RESPONSE,GO_NOW,MENU_NO,MENU_DONE,SCENARIO,I,PREVIOUS_RESPONSE) BYTE;

  11   1      DECLARE (OK1,OK2,TOTALY_OK) BYTE;

  12   1      DECLARE SCENARIO_BUFFER (9) BYTE EXTERNAL;

  13   1      DECLARE (RESP_1_ASCII,RESP_2_ASCII,RESP_1_NUM,RESP_2_NUM) BYTE;

  14   1      DECLARE (SIGHT_FLAG,TRACK_FLAG,DONE,OK,SAME) BYTE;



                /*****************************************************************************
                * THE SCENARIOS ARE LISTED BELOW.                                            *
                *****************************************************************************/


  15   1      DECLARE SCENE_0 (*) BYTE DATA (CARR_RET);

  16   1      DECLARE SCENE_1 (*) BYTE DATA ('A',CARR_RET,'H',CARR_RET,'B',CARR_RET,
                'S 1',CARR_RET,'R 225',CARR_RET,'F 1',CARR_RET,'E',CARR_RET,
                'X 1500',CARR_RET,'P 2620',CARR_RET,'X 1000',CARR_RET,'C',CARR_RET,
                'B',CARR_RET,'I',CARR_RET,'Q','!','*');

  17   1      DECLARE SCENE_2 (*) BYTE DATA ('P',01H,'A',CARR_RET,'H',CARR_RET,
```

```
            'B',CARR_RET,'S 1',CARR_RET,'R 173',CARR_RET,'F 1',CARR_RET,'E',CARR_RET,
            'X 764',CARR_RET,'P 5240',CARR_RET,'0',CARR_RET,'Q','*','P',02H,'A',
            CARR_RET,'H',CARR_RET,'B',CARR_RET,'S 1',CARR_RET,'R 173',CARR_RET,'F 1',
            CARR_RET,'E',CARR_RET,'X 382',CARR_RET,'P 1650',CARR_RET,'0',CARR_RET,'Q',
            '*','P',03H,'A',CARR_RET,'H',CARR_RET,'B',CARR_RET,'S 1',CARR_RET,'R 173',
            CARR_RET,'F 1',CARR_RET,'E',CARR_RET,'P 1650',CARR_RET,'0',CARR_RET,'Q','*',
            'P',04H,'A',CARR_RET,'H',CARR_RET,'B',CARR_RET,'S 1',CARR_RET,'R 1',CARR_RET,
            'F 1',CARR_RET,'E',CARR_RET,'X 4364',CARR_RET,'C',CARR_RET,'B',CARR_RET,
            'X 1000',CARR_RET,'C',CARR_RET,'B',CARR_RET,'0',CARR_RET,'Q','*',
            'P',05H,'A',CARR_RET,'H',CARR_RET,'B',CARR_RET,'S 1',CARR_RET,'R 80',
            CARR_RET,'F 1',CARR_RET,'E',CARR_RET,'X 764',CARR_RET,'P 650',CARR_RET,'0',
            CARR_RET,'Q','!','*');

18  1     DECLARE SCENE_3 (*) BYTE DATA ('P',01H,'A',CARR_RET,'H',CARR_RET,'B',
            CARR_RET,'S 1',CARR_RET,'R 173',CARR_RET,'F 1',CARR_RET,'E',CARR_RET,
            'X 764',CARR_RET,'P 5240',CARR_RET,'0',CARR_RET,'Q','*','P',02H,'A',
            CARR_RET,'H',CARR_RET,'B',CARR_RET,'S 1',CARR_RET,'R 173',CARR_RET,'F 1',
            CARR_RET,'E',CARR_RET,'X 382',CARR_RET,'P 1650',CARR_RET,'X 1091',CARR_RET,
            'P 5240',CARR_RET,'0',CARR_RET,'Q','*','P',03H,'A',CARR_RET,'H',CARR_RET,
            'B',CARR_RET,'S 1',CARR_RET,'R 173',CARR_RET,'F 1',CARR_RET,'E',CARR_RET,
            'P 500',CARR_RET,'X 400',CARR_RET,'P 0',CARR_RET,'0',CARR_RET,'Q','*','P',
            04H,'A',CARR_RET,'H',CARR_RET,'B',CARR_RET,'S 1',CARR_RET,'R 1',CARR_RET,
            'F 1',CARR_RET,'E',CARR_RET,'X 4364',CARR_RET,'C',CARR_RET,'B',CARR_RET,
            'X 1500',CARR_RET,'C',CARR_RET,'B',CARR_RET,'0',CARR_RET,'Q','*','P',05H,
            'A',CARR_RET,'H',CARR_RET,'B',CARR_RET,'S 1',CARR_RET,'R 80',CARR_RET,'F 1',
            CARR_RET,'E',CARR_RET,'X 384',CARR_RET,'P 650',CARR_RET,'X 1091',CARR_RET,
            'P 250',CARR_RET,'0',CARR_RET,'Q','!','*');

19  1     DECLARE SCENE_4 (*) BYTE DATA ('P',01H,'A',CARR_RET,'H',CARR_RET,'B',
            CARR_RET,'S 1',CARR_RET,'R 173',CARR_RET,'F 1',CARR_RET,'E',CARR_RET,
            'X 764',CARR_RET,'P 5240',CARR_RET,'0',CARR_RET,'Q','*','P',02H,'A',
            CARR_RET,'H',CARR_RET,'B',CARR_RET,'S 1',CARR_RET,'R 173',CARR_RET,'F 1',
            CARR_RET,'E',CARR_RET,'X 382',CARR_RET,'P 1650',CARR_RET,'0',CARR_RET,
            'Q','*','P',03H,'A',CARR_RET,'H',CARR_RET,'B',CARR_RET,'S 1',CARR_RET,
            'R 173',CARR_RET,'F 1',CARR_RET,'E',CARR_RET,'P 1650',CARR_RET,'0',CARR_RET,
            'Q','*','P',04H,'A',CARR_RET,'H',CARR_RET,'B',CARR_RET,'S 1',CARR_RET,'R 1',
            CARR_RET,'F 1',CARR_RET,'E',CARR_RET,'X 5200',CARR_RET,'C',CARR_RET,'B',
            CARR_RET,'0',CARR_RET,'Q','*','P',05H,'A',CARR_RET,'H',CARR_RET,'B',CARR_RET,
            'S 1',CARR_RET,'R 245',CARR_RET,'F 1',CARR_RET,'E',CARR_RET,'P 150',CARR_RET,
            'X 6000',CARR_RET,'R 125',CARR_RET,'F 1',CARR_RET,'P 650',CARR_RET,'0',
            CARR_RET,'Q','!','*');

20  1     DECLARE SCENE_5 (*) BYTE DATA ('P',01H,'A',CARR_RET,'H',CARR_RET,'B',CARR_RET,
            'S 1',CARR_RET,'R 156',CARR_RET,'F 1',CARR_RET,'E',CARR_RET,'P 5240',
            CARR_RET,'0',CARR_RET,'Q','*','P',02H,'A',CARR_RET,'H',CARR_RET,'B',CARR_RET,
            'S 1',CARR_RET,'R 156',CARR_RET,'F 1',CARR_RET,'E',CARR_RET,'P 500',CARR_RET,
            'X 400',CARR_RET,'P 0',CARR_RET,'0',CARR_RET,'Q','*','P',03H,'A',CARR_RET,
            'H',CARR_RET,'B',CARR_RET,'S 1',CARR_RET,'R 156',CARR_RET,'F 1',CARR_RET,'E',
            CARR_RET,'X 350',CARR_RET,'P 575',CARR_RET,'X 425',CARR_RET,'P 0',CARR_RET,
            '0',CARR_RET,'Q','*','P',04H,'A',CARR_RET,'H',CARR_RET,'B',CARR_RET,'S 1',
            CARR_RET,'R 1',CARR_RET,'F 1',CARR_RET,'E',CARR_RET,'X 4364',CARR_RET,'C',
            CARR_RET,'B',CARR_RET,'X 1091',CARR_RET,'C',CARR_RET,'B',CARR_RET,'0',
            CARR_RET,'Q','*','P',05H,'A',CARR_RET,'H',CARR_RET,'B',CARR_RET,'S 1',
            CARR_RET,'R 100',CARR_RET,'F 1',CARR_RET,'E',CARR_RET,'X 367',CARR_RET,
            'P 650',CARR_RET,'X 545',CARR_RET,'P 450',CARR_RET,'X 545',CARR_RET,'L 3,3',
            CARR_RET,'P 650',CARR_RET,'0',CARR_RET,'Q','!','*');
```

```
21  1     DECLARE SCENE_6 (*) BYTE DATA ('P',01H,'A',CARR_RET,'H',CARR_RET,'B',CARR_RET,
            'S 1',CARR_RET,'R 173',CARR_RET,'F 1',CARR_RET,'E',CARR_RET,'X 764',CARR_RET,
            'P 5240',CARR_RET,'O',CARR_RET,'Q','*','P',02H,'A',CARR_RET,'H',CARR_RET,
            'B',CARR_RET,'S 1',CARR_RET,'R 173',CARR_RET,'F 1',CARR_RET,'E',CARR_RET,
            'P 2200',CARR_RET,'X 382',CARR_RET,'P 1650',CARR_RET,'O',CARR_RET,'Q','*',
            'P',03H,'A',CARR_RET,'H',CARR_RET,'B',CARR_RET,'S 1',CARR_RET,'R 173',
            CARR_RET,'F 1',CARR_RET,'E',CARR_RET,'X 382',CARR_RET,'P 2100',CARR_RET,
            'X 425',CARR_RET,'P 1650',CARR_RET,'O',CARR_RET,'Q','*','P',04H,'A',CARR_RET,
            'H',CARR_RET,'B',CARR_RET,'S 1',CARR_RET,'R 1',CARR_RET,'F 1',CARR_RET,'E',
            CARR_RET,'X 4364',CARR_RET,'C',CARR_RET,'B',CARR_RET,'X 820',CARR_RET,'C',
            CARR_RET,'B',CARR_RET,'O',CARR_RET,'Q','*','P',05H,'A',CARR_RET,'H',CARR_RET,
            'B',CARR_RET,'S 1',CARR_RET,'R 245',CARR_RET,'F 1',CARR_RET,'E',CARR_RET,
            'P 650',CARR_RET,'R 235',CARR_RET,'F 1',CARR_RET,'P 625',CARR_RET,'X 273',
            CARR_RET,'P 650',CARR_RET,'X 273',CARR_RET,'L 15,7',CARR_RET,'P 650',
            CARR_RET,'O',CARR_RET,'Q','!','*');

22  1     DECLARE GO_HOME (*) BYTE DATA ('H',CARR_RET,'N 1',CARR_RET,'-',CARR_RET,
            'R 245',CARR_RET,'E',CARR_RET,'G',CARR_RET,'T',CARR_RET,'I',CARR_RET,
            'QD',CARR_RET,'*');

23  1     DECLARE RAISE_MOTOR (*) BYTE DATA ('P',05H,'A',CARR_RET,'H',CARR_RET,
            'S 1',CARR_RET,'R 225',CARR_RET,'F 1',CARR_RET,'E',CARR_RET,
            'P 650',CARR_RET,'I',CARR_RET,'O',CARR_RET,'Q','*');

24  1     DECLARE LOWER_MOTOR (*) BYTE DATA ('H',CARR_RET,'S 1',CARR_RET,
            'R 150',CARR_RET,'F 1',CARR_RET,'E',CARR_RET,'-',CARR_RET,'N 700',CARR_RET,
            'G',CARR_RET,'I',CARR_RET,'O',CARR_RET,'QD',CARR_RET,'*');



          /*****************************************************************************
          * COMMENTS PRINTED ON THE SCREEN BY THE PRINT PROCEDURE ARE LISTED BELOW.    *
          *****************************************************************************/


25  1     DECLARE HELLO (*) BYTE DATA (CARR_RET,LINE_FEED,LINE_FEED,LINE_FEED,LINE_FEED,
            'STACS-T TRAINER VERSION 1.0',CARR_RET,LINE_FEED,'*');

26  1     DECLARE FOO_BAH (*) BYTE DATA (' I TOLD YOU THIS !$^&%$#@ TRAINER CAN''T ',
                                        'DO THAT!!!',CARR_RET,LINE_FEED,'*',);

27  1     DECLARE HELLO_CON (*) BYTE DATA (CARR_RET,LINE_FEED,'NEED A MENU? (Y OR N)',
            CARR_RET,LINE_FEED,'*');

28  1     DECLARE ITEM_1 (*) BYTE DATA ('ITEM 1: TANK MOVES TO THE CENTER OF THE',
            ' TRACK AND STOPS',CARR_RET,LINE_FEED,'*');

29  1     DECLARE ITEM_2 (*) BYTE DATA ('ITEM 2: FRONT TANK RISES FROM TRENCH, MOVING',
            ' WEST, AND BECOMES THE TARGET',CARR_RET,LINE_FEED,'        CENTER AND ',
            'REAR TANKS MOVE FROM EAST TO WEST INTO COVER',CARR_RET,LINE_FEED,'*');

30  1     DECLARE ITEM_3 (*) BYTE DATA ('ITEM 3: FRONT TANK RISES FROM TRENCH MOVING ',
            'WEST, THEN SINKS AGAIN',CARR_RET,LINE_FEED,'        CENTER TANK MOVES ',
            'WEST INTO COVER; THEN REAPPEARS, MOVING EAST',CARR_RET,LINE_FEED,'        ',
            'AND BECOMES THE TARGET.  REAR TANK MOVES OUT, THEN RETREATS TO ',
            CARR_RET,LINE_FEED,'        ITS REAR INTO COVER',CARR_RET,LINE_FEED,'*');
```

```
31  1     DECLARE ITEM_4 (*) BYTE DATA ('ITEM 4: CENTER AND REAR TANKS MOVE EAST TO ',
            'WEST INTO COVER',CARR_RET,LINE_FEED,'        FRONT TANK RISES FROM TRENCH',
            ' AND BECOMES TARGET AS CENTER AND',CARR_RET,LINE_FEED,'         REAR TANKS',
            ' DISAPPEAR',CARR_RET,LINE_FEED,'*');

32  1     DECLARE ITEM_5 (*) BYTE DATA ('ITEM 5: FRONT TANK RISES FROM TRENCH AND ',
            'TRAVERSES HILLY TERRAIN',CARR_RET,LINE_FEED,'        CENTER AND REAR ',
            'TANKS APPEAR, THEN RETREAT INTO COVER',CARR_RET,LINE_FEED,'*');

33  1     DECLARE ITEM_6 (*) BYTE DATA ('ITEM 6: FRONT TANK MOVES WEST OVER ROUGH ',
            'TERRAIN',CARR_RET,LINE_FEED,'        CENTER AND REAR TANKS MOVE WEST ',
            'THROUGH COVER AND THEN RETREAT',CARR_RET,LINE_FEED,'        BACK INTO IT',
            CARR_RET,LINE_FEED,'*');

34  1     DECLARE PAGE_COMMENT (*) BYTE DATA (CARR_RET,LINE_FEED,LINE_FEED,LINE_FEED,
            LINE_FEED,LINE_FEED,'PAGE *');

35  1     DECLARE SINGLE_PAGE (*) BYTE DATA (LINE_FEED,'PRESS "E" TO EXIT MENU.',
            CARR_RET,LINE_FEED,LINE_FEED,'*');

36  1     DECLARE FIRST_PAGE (*) BYTE DATA (LINE_FEED,'PRESS "N" TO SEE THE NEXT PAGE,',
            ' "E" TO EXIT MENU',LINE_FEED,CARR_RET,LINE_FEED,'*');

37  1     DECLARE CENTER_PAGE (*) BYTE DATA (LINE_FEED,'PRESS "N" TO SEE THE NEXT PAGE,',
            ' "P" TO SEE THE PREVIOUS PAGE, "E" TO EXIT MENU',LINE_FEED,CARR_RET,
            LINE_FEED,'*');

38  1     DECLARE LAST_PAGE (*) BYTE DATA (LINE_FEED,'PRESS "P" TO SEE THE PREVIOUS ',
            'PAGE, "E" TO EXIT MENU',LINE_FEED,CARR_RET,LINE_FEED,'*');

39  1     DECLARE FOO (*) BYTE DATA (' THIS STAGS-T TRAINER CAN''T DO THAT!!',
            CARR_RET,LINE_FEED,'*');

40  1     DECLARE REQUEST (*) BYTE DATA (CARR_RET,LINE_FEED,LINE_FEED,LINE_FEED
            ,LINE_FEED,'WHICH ITEM ?? *');

41  1     DECLARE SIGHT_Q (*) BYTE DATA ('DO YOU WISH TO USE THE DAYSIGHT (D) OR THE ',
            'NIGHTSIGHT (N)? ','*');

42  1     DECLARE TRACK_Q (*) BYTE DATA ('WHICH TRACK DO YOU WISH TO RUN THIS SCENARIO',
            ' ON',CARR_RET,LINE_FEED,'    (1 = FRONT, 2 = CENTER, AND 3 = REAR)? ','*');

43  1     DECLARE ITEM_PTRS_1 (*) POINTER DATA(@ITEM_1,@ITEM_2,@ITEM_3,@ITEM_4,@ITEM_5,
            @ITEM_6);


      /*****************************************************************************
      * THE FOLLOWING SUBROUTINE INPUTS A BYTE OF ASCII DATA FROM THE TERMINAL     *
      *****************************************************************************/

44  1     CIN:PROCEDURE BYTE;
45  2       DO WHILE NOT SHR(INPUT(IOSTATUS),1);
46  3       END;
47  2     RETURN MASK AND INPUT(IODATA);
48  2       END CIN;
```

```
                /*****************************************************************************
                * THE FOLLOWING SUBROUTINE OUTPUTS A BYTE OF ASCII DATA TO THE TERMINAL      *
                *****************************************************************************/

49   1          COUT: PROCEDURE (ITEM);
50   2            DECLARE ITEM BYTE,
51   2              DO WHILE NOT(INPUT(IOSTATUS));
52   3                END;
53   2            OUTPUT(IODATA)=ITEM;
54   2            CALL TIME(15);
55   2              END COUT;

                /*****************************************************************************/
                /* THE FOLLOWING ROUTINE OUTPUTS A CHARACTER TO THE 8741 WHEN CALLED FROM   */
                /* THE MAIN PROGRAM                                                         */
                /*****************************************************************************/

56   1          OUTPT: PROCEDURE (OUTDATA);
57   2            DECLARE OUTDATA BYTE, STAT_COM BYTE AT (0F000H), P_DATA BYTE AT (0F002H);
58   2            DO WHILE NOT SHR(STAT_COM,1);     /* WAIT UNTIL UPI41 IBF = 0 */
59   3              END;
60   2            P_DATA = NOT OUTDATA;         /* NOT  BECAUSE MULTIBUS INVERTS */
61   2            END OUTPT;



62   1          PRINT: PROCEDURE(PNTR);  /* PROMPTS THE CONSOLE */
63   2           DECLARE I WORD;
64   2           DECLARE PNTR POINTER,
                       CHAR BASED PNTR (1) BYTE;/* CHAR MUST BE AN ARRAY TO KEEP PLM HAPPY */
65   2           I = 0;
66   2           LOOP: DO WHILE CHAR(I) <> '*';
67   3            CALL COUT(CHAR(I));
68   3            I = I + 1;
69   3           END LOOP;
70   2          END PRINT;

                /*****************************************************************************
                * THIS PROCEDURE SENDS AN ENTIRE PROGRAM (OR 'SCENE'), POINTED TO BY        *
                * PROG_PTR, TO THE CY512                                                    *
                *****************************************************************************/

71   1          TANK_PROG: PROCEDURE(PROG_PTR,LENGT);
72   2            DECLARE PROG_PTR POINTER, LENGT WORD, (ITEM BASED PROG_PTR) (1) BYTE,
                     C WORD;
73   2            C = 0;
74   2            DO WHILE C < LENGT;
75   3              CALL OUTPT(ITEM(C));
76   3              C = C + 1;
77   3            END;
78   2          END TANK_PROG;


                /*****************************************************************************
                * THIS PROCEDURE SENDS ALL THE MOTORS TO THEIR HOME POSITIONS.              *
                *****************************************************************************/
```

```
 79  1     RESET_MOTORS: PROCEDURE;

 80  2       I = 0;
 81  2       DO WHILE I < 4;
 82  3         I = I + 1;
 83  3         CALL OUTPT('P');
 84  3         CALL OUTPT(I);
 85  3         CALL TANK_PROG(@GO_HOME,SIZE(GO_HOME));
 86  3         END;

 87  2       CALL OUTPT('P');
 88  2       CALL OUTPT(05H);
 89  2       CALL TANK_PROG(@LOWER_MOTOR,SIZE(LOWER_MOTOR));

 90  2       END RESET_MOTORS;


             /* OUTPUTS PAGE 1 OF THE MENU */

 91  1     MENU_1: PROCEDURE;
 92  2       RESPONSE = 0;
 93  2       CALL PRINT(@PAGE_COMMENT);
 94  2       CALL COUT('1');
 95  2       CALL COUT(CARR_RET);
 96  2       CALL COUT(LINE_FEED);
 97  2       CALL COUT(LINE_FEED);

 98  2       I = 0;
 99  2       DO WHILE I < LENGTH(ITEM_PTRS_1);
100  3         CALL PRINT(ITEM_PTRS_1(I));
101  3         I = I + 1;
102  3         END;

103  2       CALL PRINT(@SINGLE_PAGE);
104  2       END MENU_1;


             /* OUTPUTS PAGE 2 OF THE MENU */

105  1     MENU_2: PROCEDURE;
106  2       RESPONSE = 0;
107  2       END MENU_2;


             /* OUTPUTS PAGE 3 OF THE MENU */

108  1     MENU_3: PROCEDURE;
109  2       RESPONSE = 0;
110  2       END MENU_3;


             /* OUTPUTS PAGE 4 OF THE MENU */

111  1     MENU_4: PROCEDURE;
112  2       RESPONSE = 0;
113  2       END MENU_4;
```

```
                /* OUTPUTS PAGE 5 OF THE MENU */

114   1         MENU_5: PROCEDURE;
115   2           RESPONSE = 0;
116   2           END MENU_5;


117   1         GIVE_MENU: PROCEDURE;

118   2         CALL PRINT(@HELLO_CON);

119   2         RESPONSE = 0;
120   2         RESPONSE = CIN;
121   2          CALL COUT(RESPONSE);  /* ECHO PRINT */

122   2          IF (RESPONSE = 59H) OR (RESPONSE = 79H) THEN   /* UPPER OR LOWER CASE 'Y' */
123   2          MENU: DO;
124   3           CALL COUT(CARR_RET);
125   3           CALL COUT(LINE_FEED);
126   3           MENU_DONE = 0;
127   3           MENU_NO = 1;
128   3           CALL COUT(VECTOR_MODE);
129   3           CALL COUT(CLEAR_ALL);
130   3           CALL COUT(HOME_CURSOR);
131   3           CALL TIME(2000);
132   3           CALL MENU_1;
133   3           DO WHILE NOT MENU_DONE;
134   4             SAME = 0;
135   4             OK = 0;
136   4             RESPONSE = CIN;

137   4             IF (RESPONSE = 4EH) OR (RESPONSE = 6EH) THEN DO;    /*  UC OR LC 'N' */
139   5               IF MENU_NO < MAX_MENU_NO THEN DO;
141   6                 MENU_NO = MENU_NO + 1;
142   6                 OK = 1;
143   6                 END;
144   5               ELSE DO;
145   6               SAME = 1;
146   6               CALL COUT(BELL);
147   6               END;
148   5                 END;

149   4             IF (RESPONSE = 50H) OR (RESPONSE = 70H) THEN DO;    /* UC OR LC 'P' */
151   5               IF MENU_NO > 1 THEN DO;
153   6                 MENU_NO = MENU_NO - 1;
154   6                 OK = 1;
155   6                 END;
156   5               ELSE DO;
157   6               SAME = 1;
158   6               CALL COUT(BELL);
159   6               END;
160   5                 END;

161   4             IF (RESPONSE = 45H) OR (RESPONSE = 65H) THEN DO;
163   5               MENU_DONE = 1; /* UC/LC 'E' */
```

```
164  5            OK = 1;
165  5            END;

166  4          IF NOT MENU_DONE THEN IF NOT SAME THEN IF OK THEN DO;
170  5            CALL COUT(CLEAR_ALL);
171  5            CALL COUT(HOME_CURSOR);
172  5            CALL TIME(2000);
173  5            DO CASE (MENU_NO - 1);
174  6              CALL MENU_1;
175  6              CALL MENU_2;
176  6              CALL MENU_3;
177  6              CALL MENU_4;
178  6              CALL MENU_5;
179  6              END;
180  5            END;

181  4          END;

182  3        END MENU;
183  2        ELSE DO;
184  3         CALL COUT(CARR_RET);
185  3         CALL COUT(LINE_FEED);
186  3         END;

187  2        CALL COUT(CLEAR_ALL);
188  2        CALL COUT(HOME_CURSOR);
189  2        CALL TIME(2000);

190  2      END GIVE_MENU;

191  1      TANK_INIT: PROCEDURE PUBLIC;

              /*****************************************************************************
                                  TANK_INIT PROCEDURE BEGINS HERE
              *****************************************************************************/

              /*****************************************************************************
              * RESET ALL MOTORS TO STARTING POSITIONS.                                    *
              *****************************************************************************/

192  2       CALL RESET_MOTORS;


193  2       CALL COUT(VECTOR_MODE);
194  2       CALL COUT(CLEAR_ALL);
195  2       CALL COUT(HOME_CURSOR);
196  2       CALL TIME(2000);

197  2       COMMENT1:
              CALL PRINT(@HELLO);

198  2       CALL GIVE_MENU;


199  2       OK1,OK2,TOTALY_OK,GO_NOW,RESPONSE,PREVIOUS_RESPONSE = 0;
```

```
200   2      GET_ITEM: DO WHILE NOT TOTALY_OK;
                 /* WAIT TILL A PROPER MENU ITEM IS ENTERED */
201   3       CALL PRINT (@REQUEST);

202   3      OK_1: DO WHILE NOT OK1;
203   4      RESP_1_ASCII = CIN;
204   4      RESP_1_NUM = RESP_1_ASCII-30H;
205   4      IF (RESP_1_ASCII < 3AH) AND (RESP_1_ASCII > 30H) THEN /* IS BETWEEN 1 & 9 */
206   4      DO;
207   5          CALL COUT (RESP_1_ASCII);
208   5          OK1 = 1;
209   5      END;
210   4      ELSE CALL COUT(BELL);
211   4      END OK_1;

212   3      OK_2: DO WHILE NOT OK2;
213   4      RESP_2_ASCII = CIN;
214   4      RESP_2_NUM = RESP_2_ASCII - 30H;
215   4      IF ((RESP_2_ASCII < 3AH) AND (RESP_2_ASCII > 29H)) OR (RESP_2_ASCII = CARR_RET) THEN
216   4      DO;
217   5      IF RESP_2_ASCII <> CARR_RET THEN CALL COUT (RESP_2_ASCII);
219   5      OK2 = 1;
220   5      END;
221   4      ELSE CALL COUT(BELL);
222   4      END OK_2;

223   3      IF RESP_2_ASCII <> CARR_RET THEN
224   3      DO;
225   4      RESP_1_NUM = RESP_1_NUM * 10D;
226   4      RESPONSE = RESP_1_NUM + RESP_2_NUM;
227   4      SCENARIO_BUFFER (6) = RESP_1_ASCII;
228   4      SCENARIO_BUFFER (7) = RESP_2_ASCII;
229   4      GO_NOW = 0;
230   4      END;

231   3      ELSE DO;
232   4      RESPONSE = RESP_1_NUM;
233   4      SCENARIO_BUFFER (6) = SPACE;
234   4      SCENARIO_BUFFER (7) = RESP_1_ASCII;
235   4      GO_NOW = CARR_RET;
236   4      END;

237   3      IF RESPONSE <= SCENE_COUNT THEN TOTALY_OK = 1;

239   3      ELSE DO;
240   4      OK1,OK2 = 0;
241   4      IF RESPONSE <> PREVIOUS_RESPONSE THEN CALL PRINT(@FOO);
243   4      ELSE CALL PRINT(@FOO_BAH);
244   4      CALL COUT(CARR_RET);
245   4      CALL COUT(LINE_FEED);
246   4      CALL COUT(BELL);
247   4      PREVIOUS_RESPONSE = RESPONSE;
248   4      CALL GIVE_MENU;
249   4      END;
250   3      END;

251   2       SCENARIO = RESPONSE;
```

```
252  2      WAIT_GO:
               DO WHILE GO_NOW <> CARR_RET;    /* WAIT FOR CARR_RET */
253  3           GO_NOW = CIN;
254  3           IF GO_NOW <> CARR_RET THEN CALL COUT(BELL);
256  3           END WAIT_GO;

257  2      CALL COUT(CARR_RET);
258  2      CALL COUT(LINE_FEED);


               /*****************************************************************************
               * SET FLAGS:                                                                 *
               *   SIGHT_FLAG = 1 ==> INSTRUCTOR HAS A CHOICE OF DAY OR NIGHT SIGHT         *
               *   TRACK_FLAG = 1 ==> INSTRUCTOR HAS A CHOICE OF WHICH TRACK TO USE         *
               *   EAST_WEST = 1 ==> TARGET WILL START FROM THE EAST                        *
               *   STARTING_TRACK ==> THE TRACK THE SCENARIO WILL START ON                  *
               *   TARGET_SWITCH = 0 ==> THERE WILL BE NO TARGET SWITCH                     *
               *   FINAL_TRACK ==> TRACK TO SWITCH TO IF THERE IS A SWITCH                  *
               *   CONTINUE ==> SYNCHRONIZES DIGITALKER                                     *
               *****************************************************************************/

259  2      SIGHT_FLAG,EAST_WEST = 1;
260  2      TRACK_FLAG,TARGET_SWITCH,FINAL_TRACK = 0;

261  2      FLAG_SET:DO CASE (SCENARIO);

262  3        FLAG_SET_0:DO;    /* DO CASE EXPECTS 0, BUT THERE IS NEVER SCENARIO 0 */
263  4          END;

264  3        FLAG_SET_1:DO;
265  4          TRACK_FLAG = 1;
266  4          END;

267  3        FLAG_SET_2:DO;
268  4          STARTING_TRACK = 1;
269  4          END;

270  3        FLAG_SET_3:DO;
271  4          STARTING_TRACK = 2;
272  4          END;

273  3        FLAG_SET_4:DO;
274  4          STARTING_TRACK = 1;
275  4          END;

276  3        FLAG_SET_5:DO;
277  4          STARTING_TRACK = 1;
278  4          END;

279  3        FLAG_SET_6:DO;
280  4          STARTING_TRACK = 1;
281  4          END;

282  3        END FLAG_SET;
```

```
283  2     DONE = 0;

284  2     IF SIGHT_FLAG THEN DO WHILE NOT DONE; /* WE HAVE A CHOICE OF DAY/NIGHT SIGHT */

286  3       RESPONSE = 0;
287  3       CALL PRINT(@SIGHT_Q);
288  3       RESPONSE = CIN;
289  3       CALL COUT(RESPONSE);
290  3       CALL COUT(CARR_RET);
291  3       CALL COUT(LINE_FEED);
292  3       CALL COUT(LINE_FEED);
293  3       IF (RESPONSE = 44H) OR (RESPONSE = 64H) THEN DO;  /* UC OR LC 'D' */
295  4         DONE = 1;
296  4         DAY_SIGHT = 1;  /* USE DAY SIGHT */
297  4         END;
298  3       IF (RESPONSE = 4EH) OR (RESPONSE = 6EH) THEN DO;  /* UC OR LC 'N' */
300  4         DONE = 1;
301  4         DAY_SIGHT = 0;  /* USE NIGHT SIGHT */
302  4         END;

303  3       IF NOT DONE THEN CALL COUT(BELL);

305  3       END;


306  2     DONE = 0;

307  2     IF TRACK_FLAG THEN DO WHILE NOT DONE;   /* WE HAVE A CHOICE OF TRACK */

309  3       RESPONSE = 0;
310  3       CALL PRINT(@TRACK_Q);
311  3       RESPONSE = CIN;
312  3       CALL COUT(RESPONSE);
313  3       CALL COUT(CARR_RET);
314  3       CALL COUT(LINE_FEED);
315  3       CALL COUT(LINE_FEED);
316  3       IF (RESPONSE > 30H) AND (RESPONSE < 34H) THEN DO;
318  4         STARTING_TRACK = RESPONSE - 30H;        /* SELECT STARTING TRACK */
319  4         DONE = 1;
320  4         END;
321  3       ELSE CALL COUT(BELL);

322  3       CALL TANK_PROG(@RAISE_MOTOR,SIZE(RAISE_MOTOR));

323  3       END;

324  2     CONTINUE = 1;   /* USED TO SYNCHRONIZE OPERATIONS WITH 'DIGITALKER' */

325  2     END TANK_INIT;



326  1     TANK_START: PROCEDURE PUBLIC;   /* CALL AFTER SCREEN PRESENTATION COMPLETE */

327  2       CALL OUTPT('T');
328  2       CALL OUTPT(STARTING_TRACK);
```

```
329  2        IF TRACK_FLAG THEN DO;
331  3          CALL OUTPT('P');
332  3          CALL OUTPT(STARTING_TRACK);
333  3          END;

334  2      DO CASE (SCENARIO);
335  3        CALL TANK_PROG(@SCENE_0,SIZE(SCENE_0));
336  3        CALL TANK_PROG(@SCENE_1,SIZE(SCENE_1));
337  3        CALL TANK_PROG(@SCENE_2,SIZE(SCENE_2));
338  3        CALL TANK_PROG(@SCENE_3,SIZE(SCENE_3));
339  3        CALL TANK_PROG(@SCENE_4,SIZE(SCENE_4));
340  3        CALL TANK_PROG(@SCENE_5,SIZE(SCENE_5));
341  3        CALL TANK_PROG(@SCENE_6,SIZE(SCENE_6));
342  3      END;

343  2      CONTINUE = 1;

344  2      END TANK_START;


        /*****************************************************************************
        * WE NOW WISH TO STOP THE TANKS IMMEDIATELY AND WAIT FOR RESET.              *
        *****************************************************************************/

345  1      TANK_KILLED: PROCEDURE PUBLIC;

346  2        CALL OUTPT('R');          /* RESET ALL CY512S */

347  2        END TANK_KILLED;

348  1        END TOW_START_UP_MODULE;
```

CROSS-REFERENCE LISTING

DEFN	ADDR	SIZE	NAME, ATTRIBUTES, AND REFERENCES
5			ADM3A_CLEAR. . . . . LITERALLY
5			ADM3A_MODE . . . . . LITERALLY
5			ALPHA_4010_MODE. . . LITERALLY
3			BELL . . . . . LITERALLY 146 158 210 221 246 255 304 321
72	0002H	2	C. . . . . WORD 73 74 75 76
3			CARR_RET . . . . . LITERALLY 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 42 95 124 184 215 217 223 235 244 252 254 257 290 313
37	0987H	83	CENTER_PAGE. . . . . BYTE ARRAY(83) DATA
64	0000H	1	CHAR . . . . . BYTE BASED(PNTR) ARRAY(1) 66 67
44	0B0EH	21	CIN. . . . . PROCEDURE BYTE STACK=0002H 120 136 203 213 253 288 311
5			CLEAR_ALL. . . . . LITERALLY 129 170 187 194
6			CNTR2MODE. . . . . LITERALLY
197	0E2CH		COMMENT1 . . . . . LABEL
9	0000H	1	CONTINUE . . . . . BYTE EXTERNAL(5) 324 343
6			CONTROL. . . . . LITERALLY
6			COUNTER_2. . . . . LITERALLY
49	0B23H	34	COUT . . . . . PROCEDURE STACK=0004H 67 94 95 96 97 121 124 125 128 129 130 146 158 170 171 184 185 187 188 193 194 195 207 210 218 221 244 245 246 255 257 258 289 290 291 292 304 312 313 314 315 321
9	0000H	1	DAY_SIGHT. . . . . BYTE EXTERNAL(0) 296 301
14	0014H	1	DONE . . . . . BYTE 283 285 295 300 303 306 308 319
9	0000H	1	EAST_WEST. . . . . BYTE EXTERNAL(4) 259
9	0000H	1	FINAL_TRACK. . . . . BYTE EXTERNAL(3) 260
36	0952H	53	FIRST_PAGE . . . . . BYTE ARRAY(53) DATA
261	0FDAH		FLAG_SET . . . . . LABEL
262	0FE7H		FLAG_SET_0 . . . . . LABEL
264	0FE7H		FLAG_SET_1 . . . . . LABEL
267	0FEEH		FLAG_SET_2 . . . . . LABEL
270	0FEEH		FLAG_SET_3 . . . . . LABEL
273	0FF9H		FLAG_SET_4 . . . . . LABEL
276	0FF9H		FLAG_SET_5 . . . . . LABEL
279	0FF9H		FLAG_SET_6 . . . . . LABEL
39	0A13H	40	FOO. . . . . BYTE ARRAY(40) DATA 242
26	056AH	54	FOO_BAH. . . . . BYTE ARRAY(54) DATA 243
200	0E4CH		GET_ITEM . . . . . LABEL
117	0C81H	382	GIVE_MENU. . . . . PROCEDURE STACK=0014H 198 248
22	04E7H	26	GO_HOME. . . . . BYTE ARRAY(26) DATA 85
10	0005H	1	GO_NOW . . . . . BYTE 199 229 235 252 253 254
5			GRAPHICS_CLEAR . . . LITERALLY
25	0547H	35	HELLO. . . . . BYTE ARRAY(35) DATA 197
27	05A0H	26	HELLO_CON. . . . . BYTE ARRAY(26) DATA 118
5			HOME_CURSOR. . . . . LITERALLY 130 171 188 195
10	0009H	1	I. . . . . BYTE 80 81 82 84 98 99 100 101
63	0000H	2	I. . . . . WORD 65 66 67 68
			INPUT. . . . . BUILTIN 45 47 51
4			IODATA . . . . . LITERALLY 47 53

```
  4                 IOSTATUS . . . . . .  LITERALLY        45   51
 72  0000H     1    ITEM . . . . . .      BYTE BASED(PROG_PTR) ARRAY(1)        75
 49  0004H     1    ITEM . . . . . .      BYTE PARAMETER AUTOMATIC        50   53
 28  05BAH    58    ITEM_1 . . . . . .    BYTE ARRAY(58) DATA         43
 29  05F4H   141    ITEM_2 . . . . . .    BYTE ARRAY(141) DATA        43
 30  0681H   243    ITEM_3 . . . . . .    BYTE ARRAY(243) DATA        43
 31  0774H   163    ITEM_4 . . . . . .    BYTE ARRAY(163) DATA        43
 32  0817H   130    ITEM_5 . . . . . .    BYTE ARRAY(130) DATA        43
 33  0899H   145    ITEM_6 . . . . . .    BYTE ARRAY(145) DATA        43
 43  0000H    24    ITEM_PTRS_1. . . . .  POINTER ARRAY(6) DATA          99  100
 38  09DAH    57    LAST_PAGE. . . . . .  BYTE ARRAY(57) DATA
 71  0004H     2    LENGT. . . . . .      WORD PARAMETER AUTOMATIC        72   74
                    LENGTH . . . . . .    BUILTIN          99
  8                 LINE_FEED. . . . . .  LITERALLY        25   26   27   28   29   30   31   32   33   34   35
                                            36   37   38   39   40   42   96   97  125  185  245  258  291  292
                                           314  315
 66  0B74H         LOOP . . . . . .      LABEL
 24  0523H    36   LOWER_MOTOR. . . . .  BYTE ARRAY(36) DATA         89
  4                 MASK . . . . . .      LITERALLY        47
  2                 MAX_MENU_NO. . . . .  LITERALLY       139
123  0CB0H         MENU . . . . . .      LABEL
 91  0C01H    88   MENU_1 . . . . . .    PROCEDURE STACK=0010H          132  174
105  0C59H    10   MENU_2 . . . . . .    PROCEDURE STACK=0002H          175
108  0C63H    10   MENU_3 . . . . . .    PROCEDURE STACK=0002H          176
111  0C6DH    10   MENU_4 . . . . . .    PROCEDURE STACK=0002H          177
114  0C77H    10   MENU_5 . . . . . .    PROCEDURE STACK=0002H          178
 10  0007H     1   MENU_DONE. . . . . .  BYTE       126  133  163  166
 10  0006H     1   MENU_NO. . . . . .    BYTE       127  139  141  151  153  173
 14  0015H     1   OK . . . . . .        BYTE       135  142  154  164  168
 11  000BH     1   OK1. . . . . .        BYTE       199  202  208  240
 11  000CH     1   OK2. . . . . .        BYTE       199  212  219  240
202  0E61H         OK_1 . . . . . .      LABEL
212  0E96H         OK_2 . . . . . .      LABEL
 56  0004H     1   OUTDATA. . . . . .    BYTE PARAMETER AUTOMATIC        57   60
 56  0B45H    38   OUTPT. . . . . .      PROCEDURE STACK=0004H          75   83   84   87   88  327  328  331
                                           332  346
                    OUTPUT . . . . . .    BUILTIN          53
 34  092AH    12   PAGE_COMMENT . . . .  BYTE ARRAY(12) DATA         93
 62  0004H     4   PNTR . . . . . .      POINTER PARAMETER AUTOMATIC          64   66   67
 10  000AH     1   PREVIOUS_RESPONSE. .  BYTE       199  241  247
 62  0B6BH    38   PRINT. . . . . .      PROCEDURE STACK=000CH          93  100  103  118  197  201  242  243
                                           287  310
 71  0006H     4   PROG_PTR . . . . . .  POINTER PARAMETER AUTOMATIC          72   75
 57  F002H     1   P_DATA . . . . . .    BYTE AT ABSOLUTE           60
 23  0501H    34   RAISE_MOTOR. . . . .  BYTE ARRAY(34) DATA        322
 40  0A3BH    20   REQUEST. . . . . .    BYTE ARRAY(20) DATA        201
 79  0BB7H    74   RESET_MOTORS . . . .  PROCEDURE STACK=0012H          192
 10  0004H     1   RESPONSE . . . . . .  BYTE        92  106  109  112  115  119  120  121  122  136  137  149
                                          161  199  226  232  237  241  247  251  286  288  289  293  298  309
                                          311  312  316  318
 13  000EH     1   RESP_1_ASCII . . . .  BYTE       203  204  205  207  227  234
 13  0010H     1   RESP_1_NUM . . . . .  BYTE       204  225  226  232
 13  000FH     1   RESP_2_ASCII . . . .  BYTE       213  214  215  217  218  223  228
 13  0011H     1   RESP_2_NUM . . . . .  BYTE       214  226
 14  0016H     1   SAME . . . . . .      BYTE       134  145  157  167
 10  0008H     1   SCENARIO . . . . . .  BYTE       251  261  334
 12  0000H     9   SCENARIO_BUFFER. . .  BYTE ARRAY(9) EXTERNAL(6)      227  228  233  234
```

```
    15  0018H     1  SCENE_0. . . . . .   BYTE ARRAY(1) DATA        335
    16  0019H    52  SCENE_1. . . . . .   BYTE ARRAY(52) DATA       336
    17  004DH   205  SCENE_2. . . . . .   BYTE ARRAY(205) DATA      337
    18  011AH   241  SCENE_3. . . . . .   BYTE ARRAY(241) DATA      338
    19  020BH   212  SCENE_4. . . . . .   BYTE ARRAY(212) DATA      339
    20  02DFH   248  SCENE_5. . . . . .   BYTE ARRAY(248) DATA      340
    21  03D7H   272  SCENE_6. . . . . .   BYTE ARRAY(272) DATA      341
     2               SCENE_COUNT. . . .   LITERALLY       237
     6               SETCOUNT . . . . .   LITERALLY
                     SHR. . . . . . . .   BUILTIN          45   58
    14  0012H     1  SIGHT_FLAG . . . .   BYTE       259  284
    41  0A4FH    60  SIGHT_Q. . . . . .   BYTE ARRAY(60) DATA       287
    35  0936H    28  SINGLE_PAGE. . . .   BYTE ARRAY(28) DATA       103
                     SIZE . . . . . . .   BUILTIN          85   89  322  335  336  337  338  339  340  341
     3               SPACE. . . . . . .   LITERALLY       233
     9  0000H     1  STARTING_TRACK . .   BYTE EXTERNAL(1)          268  271  274  277  280  318  328  332
    57  F000H     1  STAT_COM . . . . .   BYTE AT ABSOLUTE          58
   191  0DFFH   797  TANK_INIT. . . . .   PROCEDURE PUBLIC STACK=0018H
   345  11C6H    18  TANK_KILLED. . . .   PROCEDURE PUBLIC STACK=0008H
    71  0B91H    38  TANK_PROC. . . . .   PROCEDURE STACK=000EH          85   89  322  335  336  337  338  339
                                           340  341
   326  111CH   170  TANK_START . . . .   PROCEDURE PUBLIC STACK=0012H
     9  0000H     1  TARGET_SWITCH. . .   BYTE EXTERNAL(2)          260
                     TIME . . . . . . .   BUILTIN          54  131  172  189  196
    11  000DH     1  TOTALY_OK. . . . .   BYTE       199  200  238
     1  0B0EH        TOW_START_UP_MODULE. PROCEDURE STACK=0000H
    14  0013H     1  TRACK_FLAG . . . .   BYTE       260  265  307  329
    42  0A88H    93  TRACK_Q. . . . . .   BYTE ARRAY(93) DATA       310
     7               USART_COMMAND. . .   LITERALLY
     7               USART_CONTROL. . .   LITERALLY
     7               USART_MODE . . . .   LITERALLY
     5               VECTOR_MODE. . . .   LITERALLY       128  193
   252  0F8FH        WAIT_GO. . . . . .   LABEL


MODULE INFORMATION:

     CODE AREA SIZE     = 11D8H   4568D
     CONSTANT AREA SIZE = 0000H      0D
     VARIABLE AREA SIZE = 0017H     23D
     MAXIMUM STACK SIZE = 0018H     24D
     652 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION
```

```
SIS-II PL/M-86 V2.1 COMPILATION OF MODULE MAT_GRAPH
OBJECT MODULE PLACED IN :F2:GRAPH.OBJ
COMPILER INVOKED BY:  PLM86 :F2:GRAPH.PLM DEBUG  LARGE OPTIMIZE(3) ROM WORKFILES (:F2:,:F2:)

 1          MAT_GRAPH: DO;
 2    1     DECLARE XREG LITERALLY '14H',
                  YREG LITERALLY '16H',
                  GSCALE LITERALLY '10H',
                  SCROLL LITERALLY '12H',
                  FLAGS LITERALLY '12H',
                  ERASE LITERALLY '14H';
 3    1     DECLARE (XREG_DATA, YREG_DATA, GSCALE_VAL) BYTE PUBLIC;
 4    1     DECLARE (YANG_BYTE,ZANG_BYTE) BYTE PUBLIC;
 5    1     DECLARE BIT_BUCKET BYTE;
 6    1     DECLARE (XMIN,YMIN,XMIN_TMP,XMAX_TMP,YMIN_TMP) INTEGER;
 7    1     DECLARE (SIZ,HALF_SIZ,YANG_SCALED,ZANG_SCALED,YMAX,XMAX,TMP,
                   TMP_WRD,ONE_THIRD,TWO_THIRDS) INTEGER PUBLIC;
 8    1     DECLARE (YANG_BIRD,ZANG_BIRD) INTEGER EXTERNAL;

           /*****************************************************************
                  MAT_OUT OUTPUTS AN X,Y,GSCALE_VAL TO THE
                  MATROX GRAPHICS UNIT.
           *****************************************************************/

 9    1     MAT_OUT: PROCEDURE PUBLIC;
10    2     OUTPUT (XREG) = LOW(UNSIGN(XMIN_TMP));
11    2     OUTPUT (YREG) = LOW(UNSIGN(YMIN));
12    2     OUTPUT (GSCALE) = GSCALE_VAL;
13    2     END MAT_OUT;

14    1     LINE: PROCEDURE PUBLIC;
15    2      IF YMIN > 0 AND YMIN <= 255D THEN DO;
17    3         DO WHILE XMAX_TMP >= XMIN_TMP;
18    4         IF XMIN_TMP >= 0 AND XMIN_TMP <= 255 THEN CALL MAT_OUT;
20    4         XMIN_TMP = XMIN_TMP + 1;
21    4         END;

22    3     END;

23    2     END LINE;

24    1     MATROX_START_UP: PROCEDURE PUBLIC;


25    2     OUTPUT (SCROLL) = 0;
26    2     OUTPUT (GSCALE) = 0;
27    2     BIT_BUCKET = INPUT (ERASE); /*INITIATE ERASE CYCLE*/
28    2     DO WHILE (NOT(INPUT(FLAGS))); /* WAIT TILL MATROX IS FINISHED ERASING */
29    3     END;
30    2     END MATROX_START_UP;


31    1     MATROX_DRAW: PROCEDURE PUBLIC;
32    2     YANG_SCALED =YANG_BIRD;    /* WAS YANG_BIRD / 4 */
33    2     ZANG_SCALED =ZANG_BIRD;    /* WAS ZANG_BIRD / 4 */
```

```
34   2      YANG_BYTE = LOW(UNSIGN(YANG_SCALED + 127D));
35   2      ZANG_BYTE = LOW(UNSIGN(ZANG_SCALED + 127D));

36   2      HALF_SIZ = SIZ/2;

37   2      XMIN = (127D + YANG_SCALED) - HALF_SIZ;
38   2      YMIN = (127D + ZANG_SCALED) - HALF_SIZ;
39   2      XMAX = (127D + YANG_SCALED) + HALF_SIZ;
40   2      YMAX = (127D + ZANG_SCALED) + HALF_SIZ;
41   2      TMP = SIZ/3;
42   2      ONE_THIRD = YMIN + TMP;
43   2      TWO_THIRDS = YMIN + (TMP * 2);

44   2      DO WHILE YMIN <= ONE_THIRD;
45   3          TMP_WRD = ONE_THIRD - YMIN;
46   3          XMIN_TMP = TMP_WRD + XMIN;
47   3          XMAX_TMP = XMAX - TMP_WRD;
48   3          CALL LINE;
49   3          YMIN = YMIN + 1;
50   3          END;

51   2      DO WHILE YMIN <= TWO_THIRDS;
52   3          XMIN_TMP = XMIN;
53   3          XMAX_TMP = XMAX;
54   3          CALL LINE;
55   3          YMIN = YMIN + 1;
56   3          END;

57   2      DO WHILE YMIN <= YMAX;
58   3          TMP_WRD = YMIN - TWO_THIRDS;
59   3          XMIN_TMP = TMP_WRD + XMIN;
60   3          XMAX_TMP = XMAX - TMP_WRD;
61   3          CALL LINE;
62   3          YMIN = YMIN + 1;
63   3          END;

64   2      END MATROX_DRAW;
65   1      END;



MODULE INFORMATION:

     CODE AREA SIZE     = 0175H    373D
     CONSTANT AREA SIZE = 0000H      0D
     VARIABLE AREA SIZE = 0024H     36D
     MAXIMUM STACK SIZE = 0012H     18D
     89 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION
```

```
ISIS-II PL/M-86 V2.1 COMPILATION OF MODULE SMOKE
OBJECT MODULE PLACED IN :F2:SMOKE.OBJ
COMPILER INVOKED BY:  PLM86 :F2:SMOKE.PLM DEBUG  LARGE OPTIMIZE(3) ROM WORKFILES (:F2:,:F2:)


  1          SMOKE:  DO;

  2   1      DECLARE (SERIES_NO, DONE, BACKGROUND, PATH) BYTE PUBLIC;

  3   1      DECLARE FLAGS LITERALLY '12H';
  4   1      DECLARE GSCALE LITERALLY '10H';
  5   1      DECLARE (GSCALE_VAL,DATA_RDY1,BIRD_DT_RDY,V_REP_FLAG,H_REP_FLAG) BYTE EXTERNAL;
  6   1      DECLARE (SAVE_YANG_BIRD, SAVE_ZANG_BIRD) INTEGER PUBLIC;
  7   1      DECLARE SIZ INTEGER EXTERNAL;
  8   1      DECLARE SAVE_SIZ INTEGER;
  9   1      DECLARE (SAVE_NEW_YANG_BIRD,SAVE_NEW_ZANG_BIRD,SAVE_NEW_SIZ) INTEGER;
 10   1      DECLARE (DISTANCE) WORD EXTERNAL;
 11   1      DECLARE (YANG_BIRD,ZANG_BIRD) INTEGER EXTERNAL;
 12   1      DECLARE ERASE LITERALLY '14H';
 13   1      DECLARE COUNT_DOWN BYTE;
 14   1      DECLARE COUNT_DOWN_INIT_VAL LITERALLY '5';

 15   1      MATROX_DRAW: PROCEDURE EXTERNAL;
 16   2      END MATROX_DRAW;


 17   1      OUTPUT_SMOKE: PROCEDURE;
 18   2      DECLARE BIT_BUCKET BYTE;
 19   2      OUTPUT (GSCALE) = BACKGROUND;
 20   2      BIT_BUCKET= INPUT (ERASE);
 21   2      END OUTPUT_SMOKE;


 22   1      GET_GV_AND_SIZ: PROCEDURE PUBLIC;

 23   2      IF DISTANCE <= 300D THEN DO;  /* WAS 500 */
 25   3      SIZ = 10D;
 26   3      GSCALE_VAL = 15D;
 27   3      GOTO SKIP;
 28   3      END;

 29   2      IF DISTANCE > 300D AND DISTANCE <= 400D THEN DO;
 31   3      SIZ = 9D;
 32   3      GSCALE_VAL = 15D;
 33   3      GOTO SKIP;
 34   3      END;

 35   2      IF DISTANCE > 400D AND DISTANCE <= 500D THEN DO;
 37   3      SIZ = 8D;
 38   3      GSCALE_VAL = 15D;
 39   3      GOTO SKIP;
 40   3      END;

 41   2      IF DISTANCE > 500D AND DISTANCE <= 600D THEN DO;
 43   3      SIZ = 7D;
 44   3      GSCALE_VAL = 15D;
```

```
45  3     GOTO SKIP;
46  3     END;

47  2     IF DISTANCE > 600D AND DISTANCE <= 700D THEN DO;
49  3     SIZ = 6D;
50  3     GSCALE_VAL = 15D;
51  3     GOTO SKIP;
52  3     END;

53  2     IF DISTANCE > 700D AND DISTANCE <= 900D THEN DO;
55  3     SIZ = 5D;
56  3     GSCALE_VAL = 15D;
57  3     GOTO SKIP;
58  3     END;

59  2     IF DISTANCE > 900D AND DISTANCE <= 1000D THEN DO;
61  3     SIZ = 4D;
62  3     GSCALE_VAL = 15D;
63  3     GOTO SKIP;
64  3     END;

65  2     IF DISTANCE > 1000D AND DISTANCE <= 1100D THEN DO;
67  3     SIZ =3D;
68  3     GSCALE_VAL = 15D;
69  3     GOTO SKIP;
70  3     END;

71  2     IF DISTANCE > 1100D AND DISTANCE <= 1200D THEN DO;
73  3     SIZ = 2D;
74  3     GSCALE_VAL = 15D;
75  3     GOTO SKIP;
76  3     END;

77  2     IF DISTANCE > 1200D AND DISTANCE <= 1300D THEN DO;
79  3     SIZ = 1D;
80  3     GSCALE_VAL = 15D;
81  3     GOTO SKIP;
82  3     END;

83  2     IF DISTANCE > 1300D THEN GSCALE_VAL = 15D;

           /*****************************************************************
           IF DISTANCE > 1400D AND DISTANCE <= 1500D THEN GSCALE_VAL = 10D;
           IF DISTANCE > 1500D AND DISTANCE <= 1600D THEN GSCALE_VAL = 9;
           IF DISTANCE > 1600D AND DISTANCE <= 1700D THEN GSCALE_VAL = 8;
           IF DISTANCE > 1700D AND DISTANCE <= 1800D THEN GSCALE_VAL = 7;
           IF DISTANCE > 1800D AND DISTANCE <= 1900D THEN GSCALE_VAL = 6;
           IF DISTANCE > 1900D AND DISTANCE <= 2000D THEN GSCALE_VAL = 5;
           IF DISTANCE > 2000D AND DISTANCE <= 2100D THEN GSCALE_VAL = 4;
           IF DISTANCE > 2100D AND DISTANCE <= 2200D THEN GSCALE_VAL = 3;
           IF DISTANCE > 2200D AND DISTANCE <= 2300D THEN GSCALE_VAL = 2;
           IF DISTANCE > 2300D AND DISTANCE <= 2400D THEN GSCALE_VAL = 1;
           IF DISTANCE > 2400D AND DISTANCE <= 2500D THEN GSCALE_VAL = 0;
           *****************************************************************/
85  2     SKIP; END;
```

```
86  1     SMOKE_INIT: PROCEDURE PUBLIC;
87  2     SERIES_NO = 0;
88  2     DONE = 0;
89  2     BACKGROUND = 0;
90  2     COUNT_DOWN = COUNT_DOWN_INIT_VAL;
91  2     END SMOKE_INIT;


92  1     SMOKE : PROCEDURE PUBLIC;


93  2     DO CASE SERIES_NO;

94  3     SERIES_0:  DO;
95  4     BACKGROUND = BACKGROUND + 1;
96  4     IF BACKGROUND = 150 THEN SERIES_NO = SERIES_NO + 1;
98  4               END;

99  3     SERIES_1:   DO;
100 4     BACKGROUND = BACKGROUND - 1;
101 4     IF BACKGROUND = 8 THEN SERIES_NO = SERIES_NO + 1;
103 4               END;

104 3     SERIES_2:   DO;
105 4     BACKGROUND = BACKGROUND + 1;
106 4     IF BACKGROUND = 150 THEN SERIES_NO = SERIES_NO + 1;
108 4               END;

109 3     SERIES_3:   DO;
110 4     BACKGROUND = BACKGROUND - 1;
111 4     IF BACKGROUND = 4 THEN SERIES_NO = SERIES_NO + 1;
113 4               END;

114 3     SERIES_4:   DO;
115 4     BACKGROUND = BACKGROUND + 1;
116 4     IF BACKGROUND = 8 THEN SERIES_NO = SERIES_NO + 1;
118 4               END;
119 3     SERIES_5:   DO;
120 4     BACKGROUND = BACKGROUND -1;
121 4     IF BACKGROUND = 0 THEN SERIES_NO = SERIES_NO +1;
123 4               END;
124 3     SERIES_6:  DO;
125 4     BACKGROUND = 0;
126 4     DONE = 1;
127 4               END;


128 3     END;
129 2     END;


130 1     OCTAGON_DRIVER: PROCEDURE PUBLIC;

131 2     IF NOT (V_REP_FLAG OR H_REP_FLAG) THEN DO;
133 3     COUNT_DOWN = COUNT_DOWN -1;
134 3     IF (NOT DONE) OR (COUNT_DOWN = 0) THEN DO;
```

```
136   4       IF DONE THEN
137   4               DO;
138   5       GSCALE_VAL = BACKGROUND;
139   5       PATH = 0;
140   5       SAVE_NEW_YANG_BIRD = YANG_BIRD;
141   5       SAVE_NEW_ZANG_BIRD = ZANG_BIRD;
142   5       SAVE_NEW_SIZ = SIZ;
143   5       YANG_BIRD = SAVE_YANG_BIRD;
144   5       ZANG_BIRD = SAVE_ZANG_BIRD;
145   5       SIZ  = SAVE_SIZ;
146   5       IF (INPUT(FLAGS)) THEN CALL MATROX_DRAW;
148   5       YANG_BIRD = SAVE_NEW_YANG_BIRD;
149   5       ZANG_BIRD = SAVE_NEW_ZANG_BIRD;
150   5       SIZ = SAVE_NEW_SIZ;
151   5               END;

152   4           IF DISTANCE < 100D THEN PATH = 1;
154   4            IF PATH = 1 THEN
155   4              DO;
156   5                  CALL SMOKE;
157   5              CALL SMOKE;
158   5              IF (INPUT(FLAGS)) THEN CALL OUTPUT_SMOKE;
160   5              PATH = 0;
161   5              END;
                   ELSE
162   4          DO;
163   5            CALL GET_GV_AND_SIZ;
164   5              IF BACKGROUND >= GSCALE_VAL THEN GSCALE_VAL = BACKGROUND;


166   5              DO WHILE NOT BIRD_DT_RDY;
167   6              IF DATA_RDY1 THEN RETURN;
169   6              END;

170   5              SAVE_YANG_BIRD = YANG_BIRD;
171   5              SAVE_ZANG_BIRD = ZANG_BIRD;
172   5              SAVE_SIZ  = SIZ;
173   5              IF (INPUT(FLAGS)) THEN CALL MATROX_DRAW;
175   5              PATH = 1;

176   5             END;
177   4             IF COUNT_DOWN = 0 THEN COUNT_DOWN = COUNT_DOWN_INIT_VAL;
179   4             END;
180   3             END;
181   2             END;
182   1             END;



MODULE INFORMATION:

     CODE AREA SIZE     = 036FH    879D
     CONSTANT AREA SIZE = 0000H      0D
     VARIABLE AREA SIZE = 0012H     18D
     MAXIMUM STACK SIZE = 000AH     10D
     212 LINES READ
```

```
PL/M-86 COMPILER    SMOKE

      0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION
```

```
ISIS-II PL/M-86 V2.1 COMPILATION OF MODULE GAE
OBJECT MODULE PLACED IN :F1:GAE.OBJ
COMPILER INVOKED BY:  PLM86 :F1:GAE.PLM DEBUG  LARGE OPTIMIZE(3) ROM WORKFILES (:F1:,:F1:)


  1          GAE:    DO;
  2    1     DECLARE (START_BIT,B_Y,B_Z, DATA_RDY1,BAD_MISS, OFFSET_Y, OFFSET_Z,
                  V_REP_FLAG,H_REP_FLAG,DAY_SIGHT)BYTE EXTERNAL,
                 (YANG_BIRD,ZANG_BIRD) INTEGER EXTERNAL,
                 (BIRD_DT_RDY, BIRD_HITS, BIRD_MISSES, H_REP_RQ,
                 H_REP_GO,V_REP_RQ,V_REP_GO, GRND_BIRD, END_REPRISE, FELL_SHORT)
                 BYTE EXTERNAL,
                 (H_MISS_ASCII,V_MISS_ASCII,X_MISS_ASCII) (22) BYTE EXTERNAL,
                 AUTO_BORESIGHT BYTE EXTERNAL,
                 DISTANCE WORD EXTERNAL,
                 FAR_TARGET BYTE EXTERNAL;

  3    1     DECLARE STARTING_TRACK BYTE EXTERNAL;
  4    1     DECLARE (OFFSET_Y1,OFFSET_Y2,OFFSET_Y3,OFFSET_Z1,OFFSET_Z2,OFFSET_Z3)
             BYTE PUBLIC;
  5    1     DECLARE (NOFFSET_Y1,NOFFSET_Y2,NOFFSET_Y3,NOFFSET_Z1,NOFFSET_Z2,NOFFSET_Z3)
             BYTE PUBLIC;

  6    1     DECLARE ALPHA_MODE (2) BYTE DATA (350,370);
  7    1     DECLARE POINT_MODE (2) BYTE DATA (350,340);
  8    1     DECLARE VECTOR_MODE (1) BYTE DATA (350);
  9    1     DECLARE ADM3_MODE  (3) BYTE DATA (350,370,300);
 10    1     DECLARE BUFFER (10) BYTE;
 11    1     DECLARE (YCNT,ZCNT) BYTE PUBLIC;
 12    1     DECLARE (GAE_OFFSET,H_X_GRAPHIC_POINT, H_Y_GRAPHIC_POINT) WORD;
 13    1     DECLARE Y_SCALE_FACTOR LITERALLY '82D';
 14    1     DECLARE PASS BYTE;
 15    1     DECLARE FAST BYTE PUBLIC;
 16    1     DECLARE ERR_MSG (*) BYTE DATA (350,400,1400,500,1230,370,
                                    'xxxERROR LIMIT EXCEEDEDxxx',350);


 17    1     PRINT: PROCEDURE (MESSAGE, MSG_LENGTH) EXTERNAL;
 18    2           DECLARE MESSAGE POINTER;
 19    2           DECLARE MSG_LENGTH WORD;
 20    2     END PRINT;

 21    1     GAE_INIT: PROCEDURE PUBLIC;

 22    2     BUFFER (0) = 360;
 23    2     BUFFER (1) = 360;
 24    2     BUFFER (2) = 360;
 25    2     BUFFER (3) = 360;
 26    2     BUFFER (4) = 360;
 27    2     BUFFER (5) = 360;
 28    2     BUFFER (6) = 360;
 29    2     BUFFER (7) = 360;
 30    2     BUFFER (8) = 360;
 31    2     BUFFER (9) = 360;
 32    2     FAST = 0;
 33    2     DISTANCE = 0;
```

```
34  2     H_X_GRAPHIC_POINT = 0;
35  2     H_Y_GRAPHIC_POINT = 0;
36  2     BAD_MISS = 0;
37  2     AUTO_BORESIGHT = 0;
38  2     END GAE_INIT;

39  1     GRAPH_GAE:  PROCEDURE PUBLIC;

40  2     DECLARE TEMP WORD;

41  2     IF AUTO_BORESIGHT = 1
          THEN DO;
43  3          IF DAY_SIGHT THEN DO CASE (STARTING_TRACK AND 03H);
45  4          ;
46  4          DO;
47  5            OFFSET_Y, OFFSET_Y1 = OFFSET_Y1 + (100 - B_Y);
48  5            OFFSET_Z, OFFSET_Z1 = OFFSET_Z1 + (100 - B_Z);
49  5            END;
50  4          DO;
51  5            OFFSET_Y, OFFSET_Y2 = OFFSET_Y2 + (100 - B_Y);
52  5            OFFSET_Z, OFFSET_Z2 = OFFSET_Z2 + (100 - B_Z);
53  5            END;
54  4          DO;
55  5            OFFSET_Y, OFFSET_Y3 = OFFSET_Y3 + (100 - B_Y);
56  5            OFFSET_Z, OFFSET_Z3 = OFFSET_Z3 + (100 - B_Z);
57  5            END;
58  4          END; /* DAY_SIGHT DO CASE */


59  3          IF NOT DAY_SIGHT THEN DO CASE (STARTING_TRACK AND 03H);
61  4          ;
62  4          DO;
63  5            OFFSET_Y, NOFFSET_Y1 = NOFFSET_Y1 + (100 - B_Y);
64  5            OFFSET_Z, NOFFSET_Z1 = NOFFSET_Z1 + (100 - B_Z);
65  5            END;
66  4          DO;
67  5            OFFSET_Y, NOFFSET_Y2 = NOFFSET_Y2 + (100 - B_Y);
68  5            OFFSET_Z, NOFFSET_Z2 = NOFFSET_Z2 + (100 - B_Z);
69  5            END;
70  4          DO;
71  5            OFFSET_Y, NOFFSET_Y3 = NOFFSET_Y3 + (100 - B_Y);
72  5            OFFSET_Z, NOFFSET_Z3 = NOFFSET_Z3 + (100 - B_Z);
73  5            END;
74  4          END; /* NIGHT_SIGHT DO CASE */

75  3          IF OFFSET_Y >= 20 THEN CALL PRINT(@ERR_MSG, LENGTH(ERR_MSG));
77  3          IF OFFSET_Z >= 20 THEN CALL PRINT(@ERR_MSG, LENGTH(ERR_MSG));

79  3          AUTO_BORESIGHT = 0;

80  3          END; /* IF AUTO_BORESIGHT */

81  2     IF NOT BAD_MISS THEN
82  2        DO;
83  3         YCNT = B_Y;
```

```
 84  3          ZCNT = B_Z;
 85  3        END;


 86  2      FAST = 1;


 87  2      IF PASS = 0 THEN DO;

 89  3      TEMP = (YCNT * 79D) / 20D;

 90  3      IF YCNT < 100D THEN

 91  3      DO;
 92  4      H_X_GRAPHIC_POINT = (395D - TEMP);
 93  4      IF H_REP_FLAG OR V_REP_FLAG THEN H_X_GRAPHIC_POINT = H_X_GRAPHIC_POINT/2;
 95  4      H_X_GRAPHIC_POINT = H_X_GRAPHIC_POINT + GAE_OFFSET;
 96  4      END;

 97  3      ELSE DO;
 98  4      H_X_GRAPHIC_POINT = (TEMP - 395D);
 99  4      IF H_REP_FLAG OR V_REP_FLAG THEN H_X_GRAPHIC_POINT = H_X_GRAPHIC_POINT/2;
101  4      H_X_GRAPHIC_POINT = GAE_OFFSET - H_X_GRAPHIC_POINT;
102  4      END;

103  3      IF H_X_GRAPHIC_POINT >1000D THEN H_X_GRAPHIC_POINT = 1000D;
105  3      IF H_X_GRAPHIC_POINT < 0 THEN H_X_GRAPHIC_POINT = 0;


107  3      IF FAR_TARGET THEN H_Y_GRAPHIC_POINT = (DISTANCE + 360D) / 6;
109  3      ELSE H_Y_GRAPHIC_POINT = (DISTANCE / 3) + 60D;

110  3      IF H_Y_GRAPHIC_POINT > 779D THEN H_Y_GRAPHIC_POINT = 779D;
112  3      END;



113  2      ELSE DO;
114  3      TEMP = (ZCNT * 79D) / 20D;

115  3      IF ZCNT < 100D THEN

116  3      DO;
117  4      H_X_GRAPHIC_POINT = (395D - TEMP);
118  4      IF H_REP_FLAG OR V_REP_FLAG THEN H_X_GRAPHIC_POINT = H_X_GRAPHIC_POINT/2;
120  4      H_X_GRAPHIC_POINT = GAE_OFFSET - H_X_GRAPHIC_POINT;
121  4      END;

122  3      ELSE DO;
123  4      H_X_GRAPHIC_POINT = (TEMP - 395D);
124  4      IF H_REP_FLAG OR V_REP_FLAG THEN H_X_GRAPHIC_POINT = H_X_GRAPHIC_POINT/2;
126  4      H_X_GRAPHIC_POINT = GAE_OFFSET + H_X_GRAPHIC_POINT;
127  4      END;

128  3      IF H_X_GRAPHIC_POINT >1000D THEN H_X_GRAPHIC_POINT = 1000D;
130  3      IF H_X_GRAPHIC_POINT < 0 THEN H_X_GRAPHIC_POINT = 0;
```

```
132  3     IF FAR_TARGET THEN H_Y_GRAPHIC_POINT = (DISTANCE + 360D) / 6;
134  3     ELSE H_Y_GRAPHIC_POINT = (DISTANCE / 3) + 60D;
135  3     IF H_Y_GRAPHIC_POINT > 779D THEN H_Y_GRAPHIC_POINT = 779D;
137  3     END;



138  2     OUTPUT: DO;
139  3     BUFFER (0) = 340; /* GO TO POINT_MODE*/
140  3     BUFFER (1) = (SHR (H_Y_GRAPHIC_POINT,5) AND 00011111B) OR 00100000B;
141  3     BUFFER (2) = (H_Y_GRAPHIC_POINT AND 00011111B) OR 01100000B;
142  3     BUFFER (3) = (SHR (H_X_GRAPHIC_POINT,5) AND 00011111B) OR 00100000B;
143  3     BUFFER (4) = (H_X_GRAPHIC_POINT AND 00011111B) OR 01000000B;
144  3     IF BAD_MISS THEN
145  3        DO;
146  4     BUFFER (5) = 370;   /* GO TO ALPHA MODE    */
147  4     BUFFER (6) = 520;   /* SEND ASTERISK       */
148  4     CALL    PRINT (@BUFFER, 7);
149  4        END;
150  3           ELSE DO;
151  4     CALL    PRINT (@BUFFER,5);
152  4        END;
153  3     END;
154  2     END GRAPH_GAE;


155  1     GAE_DRIVER: PROCEDURE PUBLIC;

156  2     DO;
157  3     IF V_REP_FLAG = 1 THEN DO;
159  4     GAE_OFFSET = 675D;
160  4     PASS = 0;
161  4     CALL GRAPH_GAE;
162  4     GAE_OFFSET = 300D;
163  4     PASS = 1;
164  4     CALL GRAPH_GAE;
165  4     BAD_MISS = 0;
166  4     END;

167  3     IF H_REP_FLAG = 1 THEN DO;
169  4     GAE_OFFSET = 675D;   /* THERE ARE 20 GU'S PER METER DURING REPRISE */
170  4     PASS = 0;            /* IN THE HORIZONTAL MISSILE PLOT (PASS 0).  */
171  4     CALL GRAPH_GAE;
172  4     GAE_OFFSET = 225D;   /* THERE ARE 20 GU'S PER METER DURING REPRISE */
173  4     PASS = 1;            /* IN THE VERTICAL MISSILE PLOT (PASS 1).    */
174  4     CALL GRAPH_GAE;
175  4     BAD_MISS = 0;
176  4     END;

177  3     IF NOT (H_REP_FLAG OR V_REP_FLAG) THEN DO;
179  4     GAE_OFFSET = 675D;
180  4     PASS = 0;
181  4     CALL GRAPH_GAE;
182  4     GAE_OFFSET = 255D;
183  4     PASS = 1;
184  4     CALL GRAPH_GAE;
```

```
185   4       BAD_MISS = 0;
186   4       END;
187   3       END;
188   2       END GAE_DRIVER;
189   1       END;



MODULE INFORMATION:

     CODE AREA SIZE     = 0550H   1360D
     CONSTANT AREA SIZE = 0000H      0D
     VARIABLE AREA SIZE = 0022H     34D
     MAXIMUM STACK SIZE = 0016H     22D
     227 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION
```

```
ISIS-II MCS-86 MACRO ASSEMBLER V2.1 ASSEMBLY OF MODULE EXPLOSION
OBJECT MODULE PLACED IN :F1:EXP.OBJ
ASSEMBLER INVOKED BY:  ASM86 :F1:EXP.SRC DEBUG


LOC  OBJ                  LINE     SOURCE

                            1      NAME    EXPLOSION
                            2
                            3      PUBLIC  HIT_EXPLOSION,GROUND_EXPLOSION,SET_FLG,RESET_FLG
                            4
                            5      CGROUP  GROUP   CODE
                            6
                            7      DATA_GROUP      GROUP   DGROUP,DATA_FOR_EXP
                            8      ASSUME CS:CGROUP,DS:DATA_GROUP
                            9
----                        10     DGROUP  SEGMENT PUBLIC  'DATA'
                            11     EXTRN START_BIT:BYTE, DATA_RDY1:BYTE, YANG_BYTE:BYTE, ZANG_BYTE:BYTE
----                        12     DGROUP  ENDS
                            13
                            14
                            15
----                        16     DATA_FOR_EXP    SEGMENT PUBLIC 'DATA'
  0014                      17     XREG            EQU     14H
  0016                      18     YREG            EQU     16H
  0010                      19     GSCALE          EQU     10H
  0012                      20     SCROLL          EQU     12H
  0012                      21     FLAGS           EQU     12H
  0014                      22     ERASE           EQU     14H
                            23
0000 ??                     24     Y_LOC           DB      ?
0001 ??                     25     Z_LOC           DB      ?
0002 ??                     26     Y_LOC_TMP       DB      ?
0003 ??                     27     Z_LOC_TMP       DB      ?
0004 ??                     28     Y_DOT_TMP       DB      ?
0005 ??                     29     Z_DOT_TMP       DB      ?
0006 ??                     30     OFFSET1         DB      ?
0007 ??                     31     OFFSET2         DB      ?
0008 ??                     32     FIRST_TIME      DB      ?
0009 ??                     33     GSCALE_VALUE    DB      ?
000A ??                     34     GRND_EXP        DB      ?
000B ??                     35     FRMGRB_FLG      DB      ?
----                        36     DATA_FOR_EXP    ENDS
                            37
                            38
                            39
----                        40     CODE    SEGMENT PUBLIC  'CODE'
                            41
0000                        42     SET_FLG                 PROC FAR
0000 1E                     43                             PUSH    DS
0001 B8----          R      44                             MOV     AX,DGROUP
0004 8ED8                   45                             MOV     DS,AX
0006 C6060B0001      R      46                             MOV     FRMGRB_FLG,1
000B 1F                     47                             POP     DS
000C CB                     48                             RET
                            49     SET_FLG                 ENDP
                            50
```

```
LOC  OBJ                  LINE    SOURCE

000D                        51    RESET_FLG           PROC FAR
000D 1E                     52                        PUSH    DS
000E B8----         R       53                        MOV     AX,DGROUP
0011 8ED8                   54                        MOV     DS,AX
0013 C6060E0000     R       55                        MOV     FRMGRB_FLG,0
0018 1F                     56                        POP     DS
0019 CB                     57                        RET
                            58    RESET_FLG           ENDP
                            59
001A                        60    GROUND_EXPLOSION    PROC    FAR
001A 1E                     61                        PUSH    DS
001B B8----         R       62                        MOV     AX,DGROUP
001E 8ED8                   63                        MOV     DS,AX
0020 C6060A0001     R       64                        MOV     GRND_EXP,1
0025 E81200                 65                        CALL    HIT_EXP
0028 C6060A0000     R       66                        MOV     GRND_EXP,0
002D 1F                     67                        POP     DS
002E CB                     68                        RET
                            69    GROUND_EXPLOSION    ENDP
                            70
002F                        71    HIT_EXPLOSION       PROC    FAR
002F 1E                     72                        PUSH    DS
0030 B8----         R       73                        MOV     AX,DGROUP
0033 8ED8                   74                        MOV     DS,AX
0035 E80200                 75                        CALL    HIT_EXP
0038 1F                     76                        POP     DS
0039 CB                     77                        RET
                            78    HIT_EXPLOSION       ENDP
                            79
003A                        80    HIT_EXP         PROC    NEAR
003A C606080001     R       81                    MOV     FIRST_TIME,1
003F C606060001     R       82                    MOV     OFFSET1,1
0044 C606070001     R       83                    MOV     OFFSET2,1
0049 C60609000F     R       84                    MOV     GSCALE_VALUE,15D
004E E86A00                 85                    CALL    GET_X_Y
0051 E88000                 86                    CALL    DRAW
0054 803E0E0001     R       87                    CMP     FRMGRB_FLG,1        ;IF ONE THEN WE ARE DONE
0059 745E                   88                    JE      DONE_WITH_EXP       ;WITH FRMGRB EXPLOSION
005B E85D00                 89                    CALL    GET_X_Y
005E 8006020003     R       90                    ADD     Y_LOC_TMP,3
0063 8006030000     R       91                    ADD     Z_LOC_TMP,0
0068 E86900                 92                    CALL    DRAW
006B E84D00                 93                    CALL    GET_X_Y
006E 802E020003     R       94                    SUB     Y_LOC_TMP,3
0073 8006030000     R       95                    ADD     Z_LOC_TMP,0
0078 E85900                 96                    CALL    DRAW
007B 803E0A0001     R       97                    CMP     GRND_EXP,1
0080 7437                   98                    JE      DONE_WITH_EXP
0082 E83600                 99                    CALL    GET_X_Y
0085 8006020005     R      100                    ADD     Y_LOC_TMP,5
008A 8006030005     R      101                    ADD     Z_LOC_TMP,5
008F E84200                102                    CALL    DRAW
0092 E82600                103                    CALL    GET_X_Y
0095 802E020005     R      104                    SUB     Y_LOC_TMP,5
009A 802E030005     R      105                    SUB     Z_LOC_TMP,5
```

```
LOC  OBJ                  LINE     SOURCE

009F E83200               106                 CALL    DRAW
00A2 E81600               107                 CALL    GET_X_Y
00A5 8006020000     R     108                 ADD     Y_LOC_TMP,0
00AA 8006030008     R     109                 ADD     Z_LOC_TMP,8
00AF E82200               110                 CALL    DRAW
00B2 803E0A0001     R     111                 CMP     GRND_EXP,1
00B7 7400                 112                 JE      DONE_WITH_EXP
00B9 90                   113     DONE_WITH_EXP:  NOP
00BA C3                   114                 RET
                          115     HIT_EXP     ENDP
                          116
                          117
00BB                      118     GET_X_Y     PROC    NEAR
00BB 8A360000       E     119                 MOV     DH,YANG_BYTE
00BF 88360300       R     120                 MOV     Z_LOC_TMP,DH
00C3 88360500       R     121                 MOV     Z_DOT_TMP,DH
00C7 8A160000       E     122                 MOV     DL,ZANG_BYTE
00CB 88160200       R     123                 MOV     Y_LOC_TMP,DL
00CF 88160400       R     124                 MOV     Y_DOT_TMP,DL
00D3 C3                   125                 RET
                          126     GET_X_Y     ENDP
                          127
00D4                      128     DRAW        PROC    NEAR
00D4 8A1E0300       R     129                 MOV     BL,Z_LOC_TMP
00D8 881E0500       R     130                 MOV     Z_DOT_TMP,BL
00DC A00200         R     131                 MOV     AL,Y_LOC_TMP
00DF A20400         R     132                 MOV     Y_DOT_TMP,AL
00E2 E88101               133                 CALL    OUTPUT_EXP
00E5 E89301               134                 CALL    DELAY
                          135
00E8 8A1E0500       R     136     DRAW_W_OFFSET1: MOV BL,Z_DOT_TMP
00EC A00400         R     137                 MOV     AL,Y_DOT_TMP
00EF 021E0600       R     138                 ADD     BL,OFFSET1
00F3 E87001               139                 CALL    OUTPUT_EXP      ;(X,Y+OFFSET1)
                          140
00F6 803E0A0001     R     141                 CMP     GRND_EXP,1      ;SKIP THIS IF GROUND EXPLOSION
00FB 740E                 142                 JE      NOT_HIT
00FD 8A1E0500       R     143                 MOV     BL,Z_DOT_TMP
0101 A00400         R     144                 MOV     AL,Y_DOT_TMP
0104 2A1E0600       R     145                 SUB     BL,OFFSET1
0108 E85B01               146                 CALL    OUTPUT_EXP      ;(X,Y-OFFSET1)
                          147
010B 8A1E0500       R     148     NOT_HIT:    MOV     BL,Z_DOT_TMP
010F A00400         R     149                 MOV     AL,Y_DOT_TMP
0112 02060600       R     150                 ADD     AL,OFFSET1
0116 E84D01               151                 CALL    OUTPUT_EXP      ;(X+OFFSET1,Y)
                          152
0119 A00400         R     153                 MOV     AL,Y_DOT_TMP
011C 8A1E0500       R     154                 MOV     BL,Z_DOT_TMP
0120 2A060600       R     155                 SUB     AL,OFFSET1
0124 E83F01               156                 CALL    OUTPUT_EXP      ;(X-OFFSET1,Y)
                          157
0127 A00400         R     158                 MOV     AL,Y_DOT_TMP
012A 8A1E0500       R     159                 MOV     BL,Z_DOT_TMP
012E 02060600       R     160                 ADD     AL,OFFSET1
```

```
LOC  OBJ                  LINE     SOURCE

0132 021E0600        R    161                    ADD     BL,OFFSET1
0136 E82D01               162                    CALL    OUTPUT_EXP      ;(X+OFFSET1,Y+OFFSET1)
                          163
0139 803E0A0001      R    164                    CMP     GRND_EXP,1
013E 7424                 165                    JE      NOT_HIT2        ;SKIP IF GROUND EXPLOSION
0140 8A1E0500        R    166                    MOV     BL,Z_DOT_TMP
0144 A00400          R    167                    MOV     AL,Y_DOT_TMP
0147 2A1E0600        R    168                    SUB     BL,OFFSET1
014B 02060600        R    169                    ADD     AL,OFFSET1
014F E81401               170                    CALL    OUTPUT_EXP      ;(X+OFFSET1,Y-OFFSET1)
                          171
0152 A00400          R    172                    MOV     AL,Y_DOT_TMP
0155 8A1E0500        R    173                    MOV     BL,Z_DOT_TMP
0159 2A060600        R    174                    SUB     AL,OFFSET1
015D 2A1E0600        R    175                    SUB     BL,OFFSET1
0161 E80201               176                    CALL    OUTPUT_EXP      ;(X-OFFSET1,Y-OFFSET1)
                          177
0164 8A1E0500        R    178    NOT_HIT2:       MOV     BL,Z_DOT_TMP
0168 A00400          R    179                    MOV     AL,Y_DOT_TMP
016B 021E0600        R    180                    ADD     BL,OFFSET1
016F 2A060600        R    181                    SUB     AL,OFFSET1
0173 E8F000               182                    CALL    OUTPUT_EXP      ;(X-OFFSET1,Y+OFFSET1)
                          183
0176 E80201               184                    CALL    DELAY
0179 FE060600        R    185                    INC     OFFSET1
017D 803E06000A      R    186                    CMP     OFFSET1,10D
0182 7624                 187                    JBE     S_DRAW_W_OFFSET1
0184 803E0B0001      R    188                    CMP     FRMGRB_FLG,1    ;CHECK TO SEE IF WE'RE IN THE
                          189                                            ;FRAME GRABBING MODE
0189 741A                 190                    JE      S_GO_BACK
018B C606080001      R    191                    MOV     FIRST_TIME,1
0190 C606090000      R    192                    MOV     GSCALE_VALUE,0 ;SET DATA LEVEL TO BLACK
0195 A00400          R    193                    MOV     AL,Y_DOT_TMP
0198 8A1E050C        R    194                    MOV     BL,Z_DOT_TMP
019C E8C700               195                    CALL    OUTPUT_EXP
019F E8D900               196                    CALL    DELAY
01A2 EB0790               197                    JMP     DRAW_W_OFFSET2
01A5 E9B000               198    S_GO_BACK:      JMP     GO_BACK
01A8 E93DFF               199    S_DRAW_W_OFFSET1:  JMP  DRAW_W_OFFSET1
                          200
01AB 8A1E0500        R    201    DRAW_W_OFFSET2: MOV     BL,Z_DOT_TMP
01AF A00400          R    202                    MOV     AL,Y_DOT_TMP
01B2 021E0700        R    203                    ADD     BL,OFFSET2
01B6 A00400          R    204                    MOV     AL,Y_DOT_TMP
01B9 E8AA00               205                    CALL    OUTPUT_EXP      ;(X,Y+OFFSET2)
01BC 8A1E0500        R    206                    MOV     BL,Z_DOT_TMP
01C0 A00400          R    207                    MOV     AL,Y_DOT_TMP
01C3 2A1E0700        R    208                    SUB     BL,OFFSET2
01C7 E89C00               209                    CALL    OUTPUT_EXP      ;(X,Y-OFFSET2)
01CA 8A1E0500        R    210                    MOV     BL,Z_DOT_TMP
01CE A00400          R    211                    MOV     AL,Y_DOT_TMP
01D1 02060700        R    212                    ADD     AL,OFFSET2
01D5 E88E00               213                    CALL    OUTPUT_EXP      ;(X+OFFSET2,Y)
01D8 A00400          R    214                    MOV     AL,Y_DOT_TMP
01DB 8A1E0500        R    215                    MOV     BL,Z_DOT_TMP
```

```
LOC  OBJ                  LINE     SOURCE

01DF 2A060700       R     216                 SUB     AL,OFFSET2
01E3 E88000               217                 CALL    OUTPUT_EXP      ;(X-OFFSET2,Y)
01E6 A00400         R     218                 MOV     AL,Y_DOT_TMP
01E9 8A1E0500       R     219                 MOV     BL,Z_DOT_TMP
01ED 02060700       R     220                 ADD     AL,OFFSET2
01F1 021E0700       R     221                 ADD     BL,OFFSET2
01F5 E86E00               222                 CALL    OUTPUT_EXP      ;(X+OFFSET2,Y+OFFSET2)
01F8 8A1E0500       R     223                 MOV     BL,Z_DOT_TMP
01FC A00400         R     224                 MOV     AL,Y_DOT_TMP
01FF 2A1E0700       R     225                 SUB     BL,OFFSET2
0203 02060700       R     226                 ADD     AL,OFFSET2
0207 E85000               227                 CALL    OUTPUT_EXP      ;(X+OFFSET2,Y-OFFSET2)
020A A00400         R     228                 MOV     AL,Y_DOT_TMP
020D 8A1E0500       R     229                 MOV     BL,Z_DOT_TMP
0211 2A060700       R     230                 SUB     AL,OFFSET2
0215 2A1E0700       R     231                 SUB     BL,OFFSET2
0219 E84A00               232                 CALL    OUTPUT_EXP      ;(X-OFFSET2,Y-OFFSET2)
021C 8A1E0500       R     233                 MOV     BL,Z_DOT_TMP
0220 A00400         R     234                 MOV     AL,Y_DOT_TMP
0223 021E0700       R     235                 ADD     BL,OFFSET2
0227 2A060700       R     236                 SUB     AL,OFFSET2
022B E83800               237                 CALL    OUTPUT_EXP      ;(X-OFFSET2,Y+OFFSET2)
022E E84A00               238                 CALL    DELAY
0231 FE060700       R     239                 INC     OFFSET2
0235 803E080001     R     240                 CMP     FIRST_TIME,1
023A 7508                 241                 JNE     GO_ON
023C C606080000     R     242                 MOV     FIRST_TIME,0
0241 E967FF               243                 JMP     DRAW_W_OFFSET2
0244 C606080001     R     244     GO_ON:      MOV     FIRST_TIME,1
0249 C60609000F     R     245                 MOV     GSCALE_VALUE,15D  ;SETS DATA LEVEL TO WHITE
024E 803E070014     R     246                 CMP     OFFSET2,20D
0253 7703                 247                 JA      GO_BACK
0255 EB0C90               248                 JMP     DRAW_W_OFFSET1_S
0258 C606060001     R     249     GO_BACK:    MOV     OFFSET1,1
025D C606070001     R     250                 MOV     OFFSET2,1
0262 C3                   251                 RET
0263 E982FE               252     DRAW_W_OFFSET1_S:  JMP  DRAW_W_OFFSET1
                          253     DRAW        ENDP
                          254
0266                      255     OUTPUT_EXP  PROC    NEAR
0266 B1FF                 256                 MOV     CL,255D
0268 2ACB                 257                 SUB     CL,BL
026A 8AD9                 258                 MOV     BL,CL
026C BAD800               259                 MOV     DX,00D8H
026F E614                 260                 OUT     XREG,AL
0271 8AC3                 261                 MOV     AL,BL
0273 E616                 262                 OUT     YREG,AL
0275 A00900         R     263                 MOV     AL,GSCALE_VALUE
0278 E610                 264                 OUT     GSCALE,AL
027A C3                   265                 RET
                          266     OUTPUT_EXP  ENDP
                          267
                          268
0278                      269     DELAY       PROC    NEAR
027B B9E803               270                 MOV     CX,1000D
```

```
LOC  OBJ                  LINE     SOURCE

027E E2FE                  271     AGAIN:          LOOP    AGAIN
0280 C3                    272                     RET
                           273     DELAY           ENDP
                           274
                           275
----                       276     CODE    ENDS
                           277     END

ASSEMBLY COMPLETE, NO ERRORS FOUND
```

```
ISIS-II PL/M-86 V2.1 COMPILATION OF MODULE TOW_ALSO
OBJECT MODULE PLACED IN :F1:RETROP.OBJ
COMPILER INVOKED BY:  PLM86 :F1:RETROP.PLM DEBUG LARGE OPTIMIZE(2) ROM



            $TITLE ('PLM86 PROCEDURE TO PRINT GRAPHS')

  1         TOW_ALSO:     DO;

  2   1     DECLARE USARTDATA LITERALLY '00DCH';
  3   1     DECLARE USARTSTAT LITERALLY '00DEH';
  4   1     DECLARE FAST BYTE EXTERNAL;

  5   1     CO: PROCEDURE (CHAR);
  6   2         DECLARE CHAR BYTE;
  7   2         DO WHILE NOT INPUT (USARTSTAT);
  8   3         END;
  9   2         OUTPUT (USARTDATA) = CHAR;
 10   2     END CO;

 11   1     PRINT:  PROCEDURE (MESSAGEPTR,LENGTH) PUBLIC;
 12   2         DECLARE MESSAGEPTR POINTER, LENGTH WORD, I WORD,
                       OUTARRAY BASED MESSAGEPTR (1) BYTE;

 13   2         DO I= 0 TO LENGTH -1;
 14   3         CALL CO(OUTARRAY(I));
 15   3         IF NOT FAST THEN CALL TIME (15);
 17   3         END;

 18   2     END PRINT;
 19   1     END;



MODULE INFORMATION:

     CODE AREA SIZE     = 0068H    104D
     CONSTANT AREA SIZE = 0000H      0D
     VARIABLE AREA SIZE = 0002H      2D
     MAXIMUM STACK SIZE = 000EH     14D
     26 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION
```

```
ISIS-II PL/M-86 V2.1 COMPILATION OF MODULE KEYBOARD_IO
OBJECT MODULE PLACED IN :F2:MATROX.OBJ
COMPILER INVOKED BY:  PLM86 :F2:MATROX.PLM DEBUG  LARGE OPTIMIZE(3) ROM WORKFILES (:F2:,:F2:)


  1          KEYBOARD_IO:    DO;

  2    1     DECLARE (START_BIT, B_Y,B_Z,DATA_RDY1,BAD_MISS,OFFSET_Y,OFFSET_Z)
                   BYTE PUBLIC AT (06000H);
  3    1     DECLARE (DAY_SIGHT,FAR_TARGET,AUTO_BORESIGHT) BYTE PUBLIC AT (06007H),
                 DISTANCE WORD PUBLIC AT (0600AH),
                 (STARTING_TRACK,TARGET_SWITCH,FINAL_TRACK,EAST_WEST) BYTE PUBLIC AT (600CH);
  4    1     DECLARE YANG_BIRD INTEGER PUBLIC AT (06010H);
  5    1     DECLARE ZANG_BIRD INTEGER PUBLIC AT (06012H);
  6    1     DECLARE (BIRD_DT_RDY, COACH_ON, BIRD_HITS, BIRD_MISSES, H_REF_RQ,
                 H_REF_GO, V_REF_RQ, V_REF_GO, GRND_BIRD, END_REPRISE,FELL_SHORT,
                 CONTINUE, HIT_KILL, HIT_DISABLE) BYTE PUBLIC AT (06014H);
  7    1     DECLARE (H_MISS_ASCII,V_MISS_ASCII,X_MISS_ASCII) (22) BYTE PUBLIC AT(06022H);
  8    1     DECLARE USART_STAT LITERALLY '00DEH',
                   USART_DATA LITERALLY '00DCH';
  9    1     DECLARE CTRL_B LITERALLY '020';
 10    1     DECLARE CTRL_C LITERALLY '030';
 11    1     DECLARE CTRL_N LITERALLY '160';
 12    1     DECLARE NAME_LET_1 (*) BYTE DATA (350,630,1420,730,1130,370);
 13    1     DECLARE (CHAR,DUMMY) BYTE;
 14    1     DECLARE RUB_OUT LITERALLY '7FH';
 15    1     DECLARE CARRIAGE_RETURN LITERALLY '0DH';
 16    1     DECLARE LINE_FEED LITERALLY '0AH';
 17    1     DECLARE NAME_BUF (11) BYTE;
 18    1     DECLARE ERASE_MODE (2) BYTE DATA (330,1770);
 19    1     DECLARE WRITE_MODE (2) BYTE DATA (330,1410);
 20    1     DECLARE ERASE_BUFFER (13) BYTE;
 21    1     DECLARE T_SIGHT_BUFFER (*) BYTE PUBLIC DATA (350,610,1540,730,1130,370,
                                                  1160,1110,1070,1100,1240,1230,
                                                  1110,1070,1100,1240,350);
 22    1     DECLARE D_SIGHT_BUFFER (*) BYTE PUBLIC DATA (350,610,1540,730,1130,370,1040,
                                                  1010,1310,350);
 23    1     DECLARE S_RANGE_BUFFER (*) BYTE PUBLIC DATA (350,570,1660,730,1130,370,610,
                                                  620,650,600,400,1150,1050,1240,
                                                  1050,1220,350);
 24    1     DECLARE L_RANGE_BUFFER (*) BYTE PUBLIC DATA (350,570,1660,730,1130,370,630,
                                                  600,600,600,400,1150,1050,1240,
                                                  1050,1220,350);
 25    1     DECLARE SCENARIO_BUFFER (9) BYTE PUBLIC;
 26    1     DECLARE AUTO_BORESIGHT_MSG (*) BYTE DATA (350,400,1400,700,1210,370,1010,
                                                     1250,1240,1170,1370,1020,1170,
                                                     1220,1050,1230,1110,1070,1100,
                                                     1240,350);


 27    1     DECLARE (OFFSET_Y1,OFFSET_Y2,OFFSET_Y3,OFFSET_Z1,OFFSET_Z2,OFFSET_Z3)
             BYTE EXTERNAL;

 28    1     DECLARE (NOFFSET_Y1,NOFFSET_Y2,NOFFSET_Y3,NOFFSET_Z1,NOFFSET_Z2,NOFFSET_Z3)
             BYTE EXTERNAL;
```

```
29  1     PRINT:  PROCEDURE (MESSAGE, MSG_LENGTH) EXTERNAL;
30  2         DECLARE MESSAGE POINTER,
                  MSG_LENGTH WORD;
31  2     END PRINT;

32  1     TANK_START: PROCEDURE EXTERNAL;
33  2     END TANK_START;

34  1     INGEST: PROCEDURE PUBLIC;

35  2     DECLARE (I,J,K) BYTE; /* I IS THE CURSOR POINTER*/
36  2     ERASE_BUFFER (0) = 400;
37  2     ERASE_BUFFER (1) = 400;
38  2     ERASE_BUFFER (2) = 400;
39  2     ERASE_BUFFER (3) = 400;
40  2     ERASE_BUFFER (4) = 400;
41  2     ERASE_BUFFER (5) = 400;
42  2     ERASE_BUFFER (6) = 400;
43  2     ERASE_BUFFER (7) = 400;
44  2     ERASE_BUFFER (8) = 400;
45  2     ERASE_BUFFER (9) = 400;
46  2     ERASE_BUFFER (10) = 330;
47  2     ERASE_BUFFER (11) = 1410;
48  2     ERASE_BUFFER (12) = 350;

49  2     SCENARIO_BUFFER (0) = 350;
50  2     SCENARIO_BUFFER (1) = 560;
51  2     SCENARIO_BUFFER (2) = 1400;
52  2     SCENARIO_BUFFER (3) = 730;
53  2     SCENARIO_BUFFER (4) = 1130;
54  2     SCENARIO_BUFFER (5) = 370;
55  2     SCENARIO_BUFFER (8) = 350;

56  2     NAME_BUF (0) = 400;
57  2     NAME_BUF (1) = 400;
58  2     NAME_BUF (2) = 400;
59  2     NAME_BUF (3) = 400;
60  2     NAME_BUF (4) = 400;
61  2     NAME_BUF (5) = 400;
62  2     NAME_BUF (6) = 400;
63  2     NAME_BUF (7) = 400;
64  2     NAME_BUF (8) = 400;
65  2     NAME_BUF (9) = 400;
66  2     NAME_BUF (10) = 350;

67  2     I=0;
68  2     FAR_TARGET = 1;


69  2     IF DAY_SIGHT = 1 THEN CALL PRINT (@D_SIGHT_BUFFER, LENGTH (D_SIGHT_BUFFER));
71  2                      ELSE CALL PRINT (@T_SIGHT_BUFFER, LENGTH (T_SIGHT_BUFFER));
72  2     CALL    TIME (10000);
73  2     IF FAR_TARGET = 0   THEN CALL PRINT (@S_RANGE_BUFFER, LENGTH (S_RANGE_BUFFER));
75  2                      ELSE CALL PRINT (@L_RANGE_BUFFER, LENGTH (L_RANGE_BUFFER));
76  2     CALL    TIME (10000);
77  2     CALL PRINT (@SCENARIO_BUFFER, LENGTH (SCENARIO_BUFFER));
```

```
 78   2      DO K = ^ TO (LENGTH (ERASE_BUFFER) -4);
 79   3         ERASE_BUFFER (K) = 400;
 80   3      END;

 81   2      DO K = 0 TO (LENGTH (NAME_BUF) -2);
 82   3         NAME_BUF    (K) = 400;
 83   3      END;

             /*****************************************************************************

                      THIS STARTS THE NAME INPUT ROUTINE.

             *****************************************************************************/
 84   2      DUNNN = 0;

 85   2      IF DAY_SIGHT THEN DO CASE (STARTING_TRACK AND 03H);
 87   3      ;
 88   3         DO;
 89   4           OFFSET_Y = OFFSET_Y1;
 90   4           OFFSET_Z = OFFSET_Z1;
 91   4           END;
 92   3         DO;
 93   4           OFFSET_Y = OFFSET_Y2;
 94   4           OFFSET_Z = OFFSET_Z2;
 95   4           END;
 96   3         DO;
 97   4           OFFSET_Y = OFFSET_Y3;
 98   4           OFFSET_Z = OFFSET_Z3;
 99   4           END;
100   3         END;

101   2      ELSE DO CASE (STARTING_TRACK AND 03H);
102   3      ;
103   3         DO;
104   4           OFFSET_Y = NOFFSET_Y1;
105   4           OFFSET_Z = NOFFSET_Z1;
106   4           END;
107   3         DO;
108   4           OFFSET_Y = NOFFSET_Y2;
109   4           OFFSET_Z = NOFFSET_Z2;
110   4           END;
111   3         DO;
112   4           OFFSET_Y = NOFFSET_Y3;
113   4           OFFSET_Z = NOFFSET_Z3;
114   4           END;
115   3         END;

116   2      START_BIT = 1;

117   2      CI: DO WHILE NOT DATA_RDY1;
118   3          DO WHILE SHR (INPUT(USART_STAT),1);
119   4          CHAR = (INPUT(USART_DATA) AND 07FH);

120   4          IF CHAR = CTRL_C THEN COACH_ON = 1;

122   4          IF CHAR = CTRL_N THEN COACH_ON = 0;
```

```
124  4          IF CHAR = CTRL_B THEN
125  4          DO;
126  5          AUTO_BORESIGHT = 1;
127  5          CALL    PRINT(@AUTO_BORESIGHT_MSG, LENGTH (AUTO_BORESIGHT_MSG));
128  5          END;

129  4      ELSE IF NOT DUNNN THEN DO;

131  5          IF CHAR = RUB_OUT
               THEN
132  5          DO; IF I > 0 THEN I = I-1;
135  6              ERASE_BUFFER (I) = NAME_BUF (I);
136  6              NAME_BUF (I) = 400;
137  6              CALL    PRINT (@ERASE_MODE, LENGTH (ERASE_MODE));
138  6              CALL    PRINT (@NAME_LET_1, LENGTH (NAME_LET_1));
139  6              CALL    PRINT (@ERASE_BUFFER, LENGTH(ERASE_BUFFER));
140  6              CALL    PRINT (@WRITE_MODE, LENGTH (WRITE_MODE));

141  6              DO J = 0 TO 09D;
142  7              ERASE_BUFFER (J) = 400;
143  7              END;

144  6          END;
145  5          ELSE DO;
146  6              IF CHAR = LINE_FEED THEN GOTO SKIP;
148  6              IF CHAR = CARRIAGE_RETURN THEN

149  6              DO;
150  7                  CALL TANK_START;
151  7                  DUNNN = 1;
152  7                  GOTO SKIP_IT;
153  7              END;

154  6          IF I < 10D THEN
155  6                  DO;
156  7                  NAME_BUF(I) = CHAR;
157  7                  I=I+1;
158  7                  CALL    PRINT (@NAME_LET_1, LENGTH (NAME_LET_1));
159  7                  CALL    TIME (200D);
160  7                  CALL    PRINT (@NAME_BUF, LENGTH(NAME_BUF));
161  7                  END;
162  6           SKIP: END;
163  5            END;
                 END;
165  3          END;
166  2      SKIP_IT: END INGEST;

167  1      REP_INFO: PROCEDURE PUBLIC;

168  2      IF DAY_SIGHT = 1 THEN CALL PRINT (@D_SIGHT_BUFFER, LENGTH (D_SIGHT_BUFFER));
170  2                       ELSE CALL PRINT (@T_SIGHT_BUFFER, LENGTH (T_SIGHT_BUFFER));
171  2      CALL    TIME (1000D);
172  2      IF FAR_TARGET = 0   THEN CALL PRINT (@S_RANGE_BUFFER, LENGTH (S_RANGE_BUFFER));
174  2                       ELSE CALL PRINT (@L_RANGE_BUFFER, LENGTH (L_RANGE_BUFFER));
175  2      CALL    TIME (1000D);
176  2      CALL PRINT (@SCENARIO_BUFFER, LENGTH (SCENARIO_BUFFER));
```

```
 177   2       CALL    TIME (1000D);
 178   2       CALL    PRINT (@NAME_LET_1, LENGTH (NAME_LET_1));
 179   2       CALL PRINT (@NAME_BUF, LENGTH (NAME_BUF));
 180   2       END REP_INFO;


 181   1       END KEYBOARD_IO;



MODULE INFORMATION:

     CODE AREA SIZE     = 04C7H   1223D
     CONSTANT AREA SIZE = 0000H      0D
     VARIABLE AREA SIZE = 0026H     38D
     MAXIMUM STACK SIZE = 000EH     14D
     229 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION
```

```
ISIS-II PL/M-86 V2.1 COMPILATION OF MODULE MAIN_MODULE
OBJECT MODULE PLACED IN :F2:RETRO.OBJ
COMPILER INVOKED BY:  PLM86 :F2:RETRO.PLM DEBUG  LARGE OPTIMIZE(3) ROM WORKFILES (:F2:,:F2:)


   1          MAIN_MODULE:    DO;
              /*----------------DECLARATIONS------------------*/
   2    1     DECLARE (DATA_RDY1,START_BIT,OFFSET_Y,OFFSET_Z,AUTO_BORESIGHT) BYTE EXTERNAL;
   3    1     DECLARE DISTANCE WORD EXTERNAL;
   4    1     DECLARE (FRESH_START,BIT_BUCKET,PGM_WAIT) BYTE;
   5    1     DECLARE (HIT_PARADE_MEMORY,HIT_SHORT_MEMORY,DEAD_BIRD_MEMORY,
                    V_REP_FLAG,H_REP_FLAG) BYTE PUBLIC;
   6    1     DECLARE (V_REP_RQ,V_REP_GO,H_REP_RQ,H_REP_GO,FAST,BIRD_DT_RDY) BYTE EXTERNAL;
   7    1     DECLARE (H_MISS_ASCII,V_MISS_ASCII,X_MISS_ASCII) (22) BYTE EXTERNAL;
   8    1     DECLARE (GRND_BIRD,BAD_MISS,BIRD_MISSES,BIRD_HITS,
                END_REPRISE,FELL_SHORT, HIT_KILL, HIT_DISABLE, COACH_ON) BYTE EXTERNAL;
   9    1     DECLARE LIT LITERALLY 'LITERALLY';
  10    1     DECLARE CR LIT '0DH';
  11    1     DECLARE LF LIT '0AH';
  12    1     DECLARE FOREVER LITERALLY 'WHILE 1';
  13    1     DECLARE FAR_TARGET BYTE EXTERNAL;
  14    1     DECLARE USART_PORT LIT '0D8H',
                USART_CNTRL LIT '0DAH',
                USART_RESET LIT '040H',
                USART_CMMND LIT '037H',
                USART_MODE LIT  '04EH',
                USART_STATUS LIT '0DEH',
                TIMER_CNTRL LIT  '0D6H',
                TIMER_CNTRL2 LIT '0D4H',
                CNTR2_MODE  LIT  '0B6H',
                LOW_ADM    LIT   '004H',
                HIGH_ADM LIT     '000H';


  15    1     DECLARE (OFFSET_Y1,OFFSET_Y2,OFFSET_Y3,OFFSET_Z1,OFFSET_Z2,
                OFFSET_Z3,NOFFSET_Y1,NOFFSET_Y2,NOFFSET_Y3,NOFFSET_Z1,
                NOFFSET_Z2,NOFFSET_Z3) BYTE EXTERNAL;


  16    1     DECLARE ALPHA_MODE_HOME (*) BYTE DATA (330,140);

  17    1     DECLARE CLEAR_SCREEN BYTE DATA (310);

  18    1     DECLARE V_REP_GRAPH (*) BYTE PUBLIC DATA
                (310,350,400,1400,400,1000,370,'V',350);

  19    1     DECLARE H_REP_GRAPH (*) BYTE PUBLIC DATA

              /*********************************'RANGE METERS'************************************/
              (310,350,400,1710,400,1000,370,1220,1010,1160,1070,1050,350,670,1560,400,1000,
              370,1150,1050,1240,1050,1220,1230,

              /************'NAME SIGHT RANGE SCENARIO'************/
              350,630,1750,730,1130,370,1160,1010,1150,1050,720,350,620,1750,730,1130,370,
              1370,1370,1370,1370,1370,1370,1370,1370,1370,1370,350,620,1470,730,1130,370,
              1230,1110,1070,1100,1240,720,350,610,1470,730,1130,370,1370,1370,1370,1370,
```

```
1370,1370,1370,1370,1370,1370,350,600,1610,730,1130,370,1220,1010,1160,1070,
1050,720,350,570,1610,730,1130,370,1370,1370,1370,1370,1370,1370,1370,1370,
1370,1370,350,560,1730,730,1130,370,1230,1030,1050,1160,1010,1220,1110,1170,
720,350,550,1730,730,1130,370,1370,1370,1370,1370,1370,1370,1370,1370,1370,
1370,

/*************************VERTICAL MISSILE ENVELOPE****************/
350,410,1740,550,1110,410,1740,470,1010,660,1570,470,1010,

/************************'MISSILE ELEVATION'**********************/
350,400,1710,460,1200,370,'MISSILE ELEVATION',

/*************************'(1 METER TICS)'*************************/
350,400,1400,470,1010,370,'(1 METER TICS)',

/************************HORIZONTAL MISSILE ENVELOPE***************/
350,410,1740,610,1130,410,1740,700,1330,350,660,1570,650,1030,410,1740,
650,1030,

/************************'MISSILE AZIMUTH'*************************/
350,400,1710,610,1200,370,'MISSILE  AZIMUTH',

/*************************'(1 METER TICS)'*************************/
350,400,1400,610,1370,370,'(1 METER TICS)');

20  1   DECLARE MORE_HREF_GRAPH (*) BYTE PUBLIC DATA

/************************'TARGET LINE'(FOR VERT MISSILE POSITION)******

(350,630,1440,460,1100,370,1240,350,620,1460,460,1100,370,1010,350,610,
1500,460,1100,370,1220,350,600,
1520,460,1100,370,1070,350,570,1540,460,1100,370,1050,350,560,1560,460,
1100,370,1240,350,540,1620,460,1100,370,1140,350,530,1640,460,1100,370,
1110,350,520,1660,460,1100,370,1160,350,510,1700,460,1100,370,1050,350,
500,1720,460,1100,370,0400,

/************************'TARGET LINE'(FOR HORIZONTAL MISSILE POS.)*****/

350,640,1520,640,1120,370,1240,350,630,1540,640,1120,370,1010,350,620,
1600,640,1120,370,1220,350,610,1600,640,1120,370,1070,350,600,1620,
640,1120,370,1050,350,570,1640,640,1120,370,1240,350,630,1540,650,1220,370,
1140,350,620,1600,650,1220,370,1110,350,610,1600,650,1220,370,1160,
350,600,1620,650,1220,370,1050,

/********************** REPRISE ELEVATION TICS **********************/

340,410,1670,470,1010,410,1650,470,1010,410,1670,470,1250,410,1650,470,1250,
410,1670,500,1110,410,1650,500,1110,410,1670,500,1350,410,1650,500,1350,410,
1670,510,1210,410,1650,510,1210,410,1670,520,1050,410,1650,520,1050,410,1670,
520,1310,410,1650,520,1310,410,1670,530,1150,410,1650,530,1150,410,1670,540,
1010,410,1650,540,1010,410,1670,540,1250,410,1650,540,1250,410,1670,550,1110,
410,1650,550,1110,
/********************** REPRISE AZIMUTH TICS ************************

340,410,1670,610,1130,410,1650,610,1130,410,1670,610,1370,410,1650,610,1370,
410,1670,620,1230,410,1650,620,1230,410,1670,630,1070,410,1650,630,1070,410,
1670,630,1230,410,1650,630,1230,410,1670,640,1170,410,1650,640,1170,410,1670,
```

```
            650,1030,410,1650,650,1030,410,1670,650,1320,410,1650,650,1320,410,1670,660,
            1130,410,1650,660,1130,410,1670,660,1370,410,1650,660,1370,410,1670,670,1230,
            410,1650,670,1230,410,1670,700,1070,410,1650,700,1070,410,1670,700,1330,410,
            1650,700,1330,

            /********************** EXTENTION FOR VER. REF. ORDINATES **************/

            350,410,1740,450,1050,410,1740,470,1010,340,410,1670,450,1050,410,1650,
            450,1050,410,1670,450,1310,410,1650,450,1310,410,1670,460,1150,410,1650,
            460,1150);

21  1       DECLARE FAR_RANGE_DATA (*) BYTE PUBLIC DATA

            (350,410,1650,400,1000,370,400,400,400,600,550,
            350,440,1500,400,1000,370,400,650,600,600,550,
            350,460,1720,400,1000,370,610,600,600,600,550,
            350,510,1530,400,1000,370,610,650,600,600,550,
            350,530,1770,400,1000,370,620,600,600,600,550,
            350,560,1610,400,1000,370,620,650,600,600,550,
            350,610,1430,400,1000,370,630,600,600,600,550,
            350,630,1650,400,1000,370,630,650,600,600,550,
            350,660,1460,400,1000,370,640,600,600,600,550);

22  1       DECLARE NEAR_RANGE_DATA (*) BYTE PUBLIC DATA
            (350,410,1650,400,1000,370,400,400,400,600,550,
            350,440,1500,400,1000,370,400,620,650,600,550,
            350,460,1720,400,1000,370,400,650,600,600,550,
            350,510,1530,400,1000,370,600,670,650,600,550,
            350,530,1770,400,1000,370,610,600,600,600,550,
            350,560,1610,400,1000,370,610,620,650,600,550,
            350,610,1430,400,1000,370,610,650,600,600,550,
            350,630,1650,400,1000,370,610,670,650,600,550,
            350,660,1460,400,1000,370,620,600,600,600,550);

23  1       DECLARE GRAPH_DATA (*) BYTE PUBLIC DATA

            /**************AZIMUTH ENVELOPE********************/
            (350,410,1740,610,1250,660,1570,610,1250,660,
            1570,700,1210,410,1740,700,1210,410,1740,610,
            1250,350,410,1740,650,1030,660,1570,650,1030,

            /**************ELEVATION ENVELOPE********************/
            350,410,1740,470,1010,660,1570,470,1010,660,1570,500,1350,410,1740,500,1350,
            410,1740,470,1010,350,410,1740,470,1370,660,1570,470,1370,

            /**************AZIMUTH RANGE TICS********************/
            340,440,1560,640,1360,440,1560,640,1320,440,1560,650,1060,440,1560,650,1130,
            470,1410,640,1360,470,1410,640,1320,470,1410,650,1060,470,1410,650,1130,510,
            1630,640,1360,510,1630,640,1320,510,1630,650,1060,510,1630,650,1130,540,1450,
            640,1360,540,1450,640,1320,540,1450,650,1060,540,1450,650,1130,560,1700,640,
            1360,560,1700,640,1320,560,1700,650,1060,560,1700,650,1130,610,1520,640,1360,
            610,1520,640,1320,610,1520,650,1060,610,1520,650,1130,630,1750,640,1360,630,
            1750,640,1320,630,1750,650,1060,630,1750,650,1130,

            /**************ELEVATION RANGE TICS********************/
            340,440,1560,470,1320,440,1560,470,1260,440,1560,500,1020,440,1560,500,1070,
            470,1410,470,1320,470,1410,470,1260,470,1410,500,1020,470,1410,500,1070,510,
```

```
1630,470,1320,510,1630,470,1260,510,1630,500,1020,510,1630,500,1070,540,1450,
470,1320,540,1450,470,1260,540,1450,500,1020,540,1450,500,1070,560,1700,470,
1320,560,1700,470,1260,560,1700,500,1020,560,1700,500,1070,610,1520,470,1320,
610,1520,470,1260,610,1520,500,1020,610,1520,500,1070,630,1750,470,1320,630,
1750,470,1260,630,1750,500,1020,630,1750,500,1070,

/xxxxxxxxxxxxxAZIMUTH MIL TICSxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx/
340,410,1670,610,1250,410,1650,610,1250,410,1670,610,1370,410,1650,610,1370,
410,1670,620,1110,410,1650,620,1110,410,1670,620,1230,410,1650,620,1230,410,
1670,620,1350,410,1650,620,1350,410,1670,630,1070,410,1650,630,1070,410,1670,
630,1210,410,1650,630,1210,410,1670,630,1330,410,1650,630,1330,410,1670,640,
1050,410,1650,640,1050,410,1670,640,1170,410,1650,640,1170,410,1670,640,1310,
410,1650,640,1310,410,1670,650,1030,410,1650,650,1030,410,1670,650,1150,410,
1650,650,1150,410,1670,650,1270,410,1650,650,1270,410,1670,660,1010,410,1650,
660,1010,410,1670,660,1130,410,1650,660,1130,410,1670,660,1250,410,1650,660,
1250,410,1670,660,1370,410,1650,660,1370,410,1670,670,1110,410,1650,670,1110,
410,1670,670,1230,410,1650,670,1230,410,1670,670,1350,410,1650,670,1350,410,
1670,700,1070,410,1650,700,1070,410,1670,700,1210,410,1650,700,1210);

24  1       DECLARE MORE_GRAPH_DATA (x) BYTE PUBLIC DATA

/xxxxxxxxxxxxELEVATION MIL TICSxxxxxxxxxxxxxxxxxx/
(340,410,1670,470,1010,410,1650,470,1010,410,1670,500,1350,410,1650,500,1350,
410,1670,470,1370,410,1650,470,1370,410,1670,470,1130,410,1650,470,1130,410,
1670,500,1230,410,1650,500,1230,410,1670,470,1250,410,1650,470,1250,410,1670,
500,1110,410,1650,500,1110,

/xxxxxxxxxxx'RIGHT LEFT UP DOWN'xxxxxxxxxxxxxxxxx/
350,430,1440,710,1000,370,500,1220,1110,1070,1100,1240,510,350,430,1440,560,
1210,370,500,1140,1050,1060,1240,510,350,430,1440,510,1140,370,500,1040,1170,
1270,1160,510,350,430,1440,440,1330,370,500,1250,1200,510,

/xxxxxxxxxxxxx'(100 MICRO RAD TICS)' FOR AZIMUTH AND 'AZIMUTH'xxxxxx/
350,660,1630,630,1240,370,1010,1320,1110,1150,1250,1240,1100,350,400,1710,
600,1350,370,500,610,600,600,400,1150,1110,1030,1220,1170,400,1220,1010,
1040,400,1240,1110,1030,1230,510,

/xxxxxxxxxxxxx'(100 MICRO RAD TICS)' FOR ELEV. AND 'ELEVATION'xxxxxxx/
350,660,1630,460,1030,370,1050,1140,1050,1260,1010,1240,1110,1170,1160,
350,400,1710,430,1310,370,500,610,600,600,400,1150,1110,1030,1220,1170,
400,1220,1010,1040,400,1240,1110,1030,1230,510,

/xxxxxxxxxxxx'NAME SIGHT RANGE SCENARIO'xxxxxxxxxxx
350,630,1750,730,1130,370,1160,1010,1150,1050,720,350,620,1750,730,1130,370,
1370,1370,1370,1370,1370,1370,1370,1370,1370,1370,350,620,1470,730,1130,370,
1230,1110,1070,1100,1240,720,350,610,1470,730,1130,370,1370,1370,1370,1370,
1370,1370,1370,1370,1370,1370,350,600,1610,730,1130,370,1220,1010,1160,1070,
1050,720,350,570,1610,730,1130,370,1370,1370,1370,1370,1370,1370,1370,1370,
1370,1370,350,560,1730,730,1130,370,1230,1030,1050,1160,1010,1220,1110,1170,
720,350,550,1730,730,1130,370,1370,1370,1370,1370,1370,1370,1370,1370,1370,
1370,

/xxxxxxxxxxxxxxxxxxxxxxxxxxx'RANGE METERS'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx/
350,400,1710,400,1000,370,1220,1010,1160,1070,1050,350,670,1560,400,1000,370,
1150,1050,1240,1050,1270,1230);
```

```
 25   1      PRINT:  PROCEDURE (MESSAGE, MSG_LENGTH) EXTERNAL;
 26   2          DECLARE MESSAGE POINTER,
                     MSG_LENGTH  WORD;
 27   2      END PRINT;

 28   1      HIT_EXPLOSION: PROCEDURE EXTERNAL;
 29   2      END HIT_EXPLOSION;

 30   1      GROUND_EXPLOSION: PROCEDURE EXTERNAL;
 31   2      END GROUND_EXPLOSION;

 32   1      GAE_INIT: PROCEDURE EXTERNAL;
 33   2      END GAE_INIT;

 34   1      TANK_KILLED: PROCEDURE EXTERNAL;
 35   2      END TANK_KILLED;

 36   1      TANK_INIT: PROCEDURE EXTERNAL;
 37   2      END TANK_INIT;

 38   1      GAE_DRIVER: PROCEDURE EXTERNAL;
 39   2      END GAE_DRIVER;

 40   1      SMOKE_INIT: PROCEDURE EXTERNAL;
 41   2      END SMOKE_INIT;

 42   1      OCTAGON_DRIVER: PROCEDURE EXTERNAL;
 43   2      END OCTAGON_DRIVER;

 44   1      REP_INFO: PROCEDURE EXTERNAL;
 45   2      END REP_INFO;

 46   1      SET_FLG: PROCEDURE EXTERNAL;
 47   2      END SET_FLG;

 48   1      RESET_FLG: PROCEDURE EXTERNAL;
 49   2      END RESET_FLG;

 50   1      FRAME_GRAB: PROCEDURE;
 51   2      OUTPUT(0CH) = 0;
 52   2      BIT_BUCKET = INPUT(10H);
 53   2      BIT_BUCKET = INPUT(14H);
 54   2      CALL TIME (400D);
 55   2      END FRAME_GRAB;

 56   1      DECLARE POINT_MODE (2) BYTE DATA (350,340);
 57   1      DECLARE VECTOR_MODE (1) BYTE DATA (350);

 58   1      DECLARE AUTO_BORESIGHT_MESG (*) BYTE DATA
                    (350,400,1400,700,1210,370,1010,
                     1250,1240,1170,1370,1020,1170,
                     1220,1050,1230,1110,1070,1100,
                     1240,350);
 59   1      DECLARE HIT_KILL_MSG (*) BYTE DATA
                    (350,670,1560,540,1200,370,'**HIT KILL**',350);
                                        /* HIT MSG IS AT (400,750)  */
 60   1      DECLARE HIT_DISABLE_MSG (*) BYTE DATA
```

```
                    (350,670,1560,540,1200,370,'**HIT DISABLE**',350);
                                        /* HIT MSG IS AT (400,750)  */
61   1      DECLARE HIT_MSG (*) BYTE DATA (350,670,1560,540,1200,370,'**HIT**',350);
                                        /* HIT MSG IS AT (400,750)  */

            /*DECLARE DIST_FELL_SHORT_MSG (*) BYTE DATA (350,670,1560,460,1100,370);  */
                                        /* DIST_FELL_SHORT MSG AT (200,750) */

62   1      DECLARE MISS_DISTANCE_MSG (*) BYTE DATA (350,670,1560,450,1120,370,
                                                'GROUND IMPACT ');
                                        /*  MISS_DISTANCE_MSG AT (170,750) */

63   1      DECLARE PPI_CONTROL LITERALLY 'OCEH';
64   1      DECLARE PPI_MODE LITERALLY '10000000B';
65   1      DECLARE SET_BIT_C0 LITERALLY '0';    /* 0000 0000 */
66   1      DECLARE RESET_BIT_C0 LITERALLY '1';  /* 0000 0001 */
67   1      DECLARE PPI_PORT_C LITERALLY 'OCCH';
68   1      DECLARE (HIT_KILL_MEM, HIT_DISABLE_MEM, BIRD_HITS_MEM) BYTE PUBLIC;


69   1      HIT_PARADE: PROCEDURE PUBLIC;
70   2      HIT_PARADE_MEMORY = 1;
71   2      DO WHILE NOT (INPUT(12H));   /* DO WHILE MATRO* IS BUSY */
72   3      END;
73   2      OUTPUT(10H) = 0;     /* SET GSCALE_VAL TO 0 */
74   2      BIT_BUCKET = INPUT(14H); /* START ERASE CYCLE */
75   2        DO WHILE NOT (INPUT(12H));  /*  WAIT TILL MATRO* IS THRU WITH ERASE */
76   3        END;
77   2      IF NOT (H_REF_FLAG OR V_REF_FLAG) THEN
78   2        DO;
79   3      CALL HIT_EXPLOSION;
80   3      CALL TANK_KILLED;
81   3        END;

            /**************************************************
              DO;
                CALL SET_FLG;
                CALL HIT_EXPLOSION;
                CALL RESET_FLG;
                CALL TIME (2500D);
                OUTPUT(PPI_PORT_C) = SET_BIT_C0;   TURN OFF VIDEO TO GUNNER
                CALL FRAME_GRAB;
                CALL SET_FLG;
                CALL HIT_EXPLOSION;
                CALL RESET_FLG;
              END;**************************************************/

82   2      FAST = 0;
83   2      IF BIRD_HITS_MEM THEN CALL PRINT (@HIT_MSG,LENGTH(HIT_MSG));
85   2      IF HIT_KILL_MEM THEN CALL PRINT  (@HIT_KILL_MSG,LENGTH(HIT_KILL_MSG));
87   2      IF HIT_DISABLE_MEM THEN CALL PRINT (@HIT_DISABLE_MSG,LENGTH(HIT_DISABLE_MSG));
89   2      CALL PRINT (@VECTOR_MODE,LENGTH(VECTOR_MODE));
90   2      FAST = 1;
91   2      END HIT_PARADE;


92   1      DEAD_BIRD: PROCEDURE PUBLIC;    * THE BIRD HAS FLOWN BY THE TARGET
```

```
                                                AND HAS STRUCK THE GROUND.    */
 93   2      BIRD_MISSES,GRND_BIRD = 0;
 94   2      DEAD_BIRD_MEMORY = 1;
 95   2      DO WHILE NOT (INPUT(12H));   /* DO WHILE MATROX IS BUSY*/
 96   3      END;
 97   2      OUTPUT(10H) = 0;    /* SET GSCALE_VAL TO 0 */
 98   2      BIT_BUCKET = INPUT(14H); /* START ERASE CYCLE */
 99   2        DO WHILE NOT (INPUT(12H));  /*  WAIT TIL MATROX IS THRU WITH ERASE */
100   3        END;
101   2      IF NOT (H_REP_FLAG OR V_REP_FLAG) THEN CALL GROUND_EXPLOSION;
              /**************************************************************
                DO;

                  CALL SET_FLG;
                  CALL HIT_EXPLOSION;
                  CALL RESET_FLG;
                  CALL TIME (2500D);
                  OUTPUT(PPI_PORT_C) = SET_BIT_C0;  *TURN OFF VIDEO TO GUNNER *
                  CALL FRAME_GRAB;
                  CALL SET_FLG;
                  CALL HIT_EXPLOSION;
                  CALL RESET_FLG;
                END;
              **************************************************************/

103   2      FAST = 0;
104   2      CALL PRINT (@MISS_DISTANCE_MSG,LENGTH(MISS_DISTANCE_MSG));
105   2      CALL PRINT (@H_MISS_ASCII,LENGTH (H_MISS_ASCII));
106   2      CALL PRINT (@V_MISS_ASCII,LENGTH (V_MISS_ASCII));
107   2      CALL PRINT (@VECTOR_MODE,LENGTH (VECTOR_MODE));
108   2      FAST = 1;
109   2      END DEAD_BIRD;



110   1      HIT_SHORT: PROCEDURE PUBLIC;    /*  THE BIRD HAS HIT SHORT OF THE
                                                  TARGET.                     */
111   2      FELL_SHORT = 0;
112   2      HIT_SHORT_MEMORY = 1;
113   2      DO WHILE NOT (INPUT(12H));   /* DO WHILE MATROX IS BUSY*/
114   3      END;
115   2      OUTPUT(10H) = 0;    /* SET GSCALE_VAL TO 0 */
116   2      BIT_BUCKET = INPUT(14H); /* START ERASE CYCLE */
117   2        DO WHILE NOT (INPUT(12H));  /*  WAIT TIL MATROX IS THRU WITH ERASE */
118   3        END;
119   2      IF NOT (H_REP_FLAG OR V_REP_FLAG) THEN CALL GROUND_EXPLOSION;
121   2      FAST = 0;
122   2      CALL PRINT (@MISS_DISTANCE_MSG,LENGTH(MISS_DISTANCE_MSG));
              /*CALL PRINT (@DIST_FELL_SHORT_MSG,LENGTH(DIST_FELL_SHORT_MSG));  */
        2     CALL PRINT (@X_MISS_ASCII, LENGTH (X_MISS_ASCII));
        2     CALL PRINT (@VECTOR_MODE,LENGTH (VECTOR_MODE));
              FAST = 1;
              END HIT_SHORT;

               ***************************
              /**TIAL_REP_RQ: PROCEDURE PUBLIC;
              ***ANCE = 0;
```

```
                FAST=0;
                CALL PRINT (@V_REP_GRAPH, LENGTH(V_REP_GRAPH));
                FAST = 1;
                V_REP_GO = 1;
                V_REP_RQ = 0;
                V_REP_FLAG = 1;
                END VERTICAL_REP_RQ;
                ***********************************/

127   1         HORIZONTAL_REP_RQ: PROCEDURE PUBLIC;
128   2         START_BIT = 0;
129   2         DISTANCE = 0;
130   2         FAST = 0;
131   2         CALL PRINT (@H_REP_GRAPH, LENGTH (H_REP_GRAPH));
132   2         CALL PRINT (@MORE_HREP_GRAPH, LENGTH (MORE_HREP_GRAPH));
133   2         CALL REP_INFO;

134   2         IF FAR_TARGET THEN CALL PRINT (@FAR_RANGE_DATA, LENGTH (FAR_RANGE_DATA));
136   2         ELSE CALL PRINT (@NEAR_RANGE_DATA, LENGTH (NEAR_RANGE_DATA));

137   2         FAST = 1;
138   2         H_REP_RQ = 0;
139   2         H_REP_FLAG = 1;
140   2         START_BIT = 1;
141   2         H_REP_GO = 1;
142   2         END HORIZONTAL_REP_RQ;

143   1         RESULTS: PROCEDURE PUBLIC;
144   2         FAST = 0;
145   2         IF HIT_PARADE_MEMORY THEN CALL HIT_PARADE;
147   2         IF HIT_SHORT_MEMORY THEN CALL HIT_SHORT;
149   2         IF DEAD_BIRD_MEMORY THEN CALL DEAD_BIRD;
151   2         FAST = 1;
152   2         V_REP_FLAG,H_REP_FLAG = 0;
153   2         END_REPRISE = 0;
154   2         END RESULTS;


155   1         CHECK_FOR_AUTO_BORE: PROCEDURE;

156   2         DECLARE CHARACTER BYTE;

157   2         IF SHR (INPUT(0DEH),1) THEN DO;
159   3             CHARACTER = INPUT (0DCH);
160   3             IF CHARACTER = 2 THEN DO;
162   4                 CALL PRINT (@AUTO_BORESIGHT_MESG, LENGTH(AUTO_BORESIGHT_MESG));
163   4                 AUTO_BORESIGHT = 1;
164   4                 END;
165   3             IF CHARACTER = 3 THEN COACH_ON = 1;
167   3             IF CHARACTER = 160 THEN COACH_ON = 0;
169   3             IF CHARACTER = 100 THEN PGM_WAIT = 1;
171   3             IF CHARACTER = 70  THEN PGM_WAIT = 0;
173   3                 END;
                /***************************************************************
                  0DEH IS USART_STATUS, 0DCH IS USART DATA, 2 IS CTRL_B,
                ***************************************************************/
174   2         END CHECK_FOR_AUTO_BORE;
```

```
175   1     SET_USART: PROCEDURE;
176   2     USART_SET_UP;
            OUTPUT(TIMER_CNTRL) = CNTR2_MODE;
177   2     CALL    TIME(1);
178   2     OUTPUT(TIMER_CNTRL2)= LOW_ADM;
179   2     CALL    TIME(1);
180   2     OUTPUT (TIMER_CNTRL2) = HIGH_ADM;
181   2     CALL    TIME(1);
182   2     OUTPUT (USART_CNTRL) = USART_RESET;
183   2     CALL    TIME(1);
184   2     OUTPUT(USART_CNTRL) = USART_MODE;
185   2     CALL    TIME(1);
186   2     OUTPUT (USART_CNTRL) = USART_CMMND;
187   2     CALL TIME(1);
188   2     OUTPUT (USART_CNTRL) = USART_RESET;
189   2     CALL    TIME(1);
190   2     OUTPUT (USART_CNTRL) = USART_MODE;
191   2     CALL    TIME(1);
192   2     OUTPUT (USART_CNTRL) = USART_CMMND;
193   2     END SET_USART;

194   1     INGEST: PROCEDURE EXTERNAL;
195   2     END INGEST;

196   1     MATROX_START_UP: PROCEDURE EXTERNAL;
197   2     END MATROX_START_UP;

198   1     MATROX_DRAW: PROCEDURE EXTERNAL;
199   2     END MATROX_DRAW;


            /*--------------EXECUTABLE STATEMENTS-------------*/

200   1     IF FRESH_START <> 0DCH THEN START_UP:DO;
202   2     START_BIT = 0;
203   2     OFFSET_Y, OFFSET_Y1, OFFSET_Y2, OFFSET_Y3 = 0;
204   2     OFFSET_Z, OFFSET_Z1, OFFSET_Z2, OFFSET_Z3 = 0;
205   2     NOFFSET_Y1, NOFFSET_Y2, NOFFSET_Y3 = 0;
206   2     NOFFSET_Z1, NOFFSET_Z2, NOFFSET_Z3 = 0;


207   2     COACH_ON = 0;
208   2     FRESH_START = 0DCH;
209   2     END START_UP;

210   1     OUTPUT(PPI_CONTROL) = PPI_MODE;
211   1     OUTPUT(PPI_PORT_C)  = RESET_BIT_C0; /* SET PORT C BIT 0 TO ZERO */
212   1     CALL TIME (15000);
213   1     START_BIT = 0;
214   1     FAST, HIT_KILL, HIT_KILL_MEM, HIT_DISABLE, HIT_DISABLE_MEM = 0;
215   1     BIRD_HITS,BIRD_HITS_MEM,PGM_WAIT = 0;
216   1     CALL SET_USART;
217   1     CALL SET_USART;

218   1     OUTPUT(10H) = 0;    /* SET GSCALE_VAL TO 0 */
219   1     BIT_BUCKET = INPUT(14H); /* START ERASE CYCLE */
```

```
220  1        DO WHILE NOT (INPUT(12H));  /* WAIT TILL MATROX IS THRU WITH ERASE */
221  2        END;

         /**************************CLEAR THE RETRO_GRAPHICS SCREEN***********/

222  1        OUTPUT (USART_PORT) = CLEAR_SCREEN;
223  1        CALL    TIME(1700D);  /* DELAY UNTIL FINISHED (>165 MILLISECONDS)*/
224  1        CALL PRINT (@ALPHA_MODE_HOME, LENGTH (ALPHA_MODE_HOME));
225  1        CALL TANK_INIT;  /* GET SCENARIO INFO FROM CONSOLE  */
226  1        OUTPUT (USART_PORT) = CLEAR_SCREEN;
227  1        CALL TIME (1700D);

         /***************** DRAW THE GRAPH ***************************************/
228  1        FAST = 0;
229  1        START_BIT = 0;
230  1        FAR_TARGET = 1;

231  1        CALL    PRINT (@GRAPH_DATA, LENGTH (GRAPH_DATA));
232  1        CALL    PRINT (@MORE_GRAPH_DATA, LENGTH (MORE_GRAPH_DATA));

233  1        IF FAR_TARGET = 1 THEN
234  1            CALL PRINT (@FAR_RANGE_DATA, LENGTH (FAR_RANGE_DATA));
235  1        ELSE    CALL PRINT (@NEAR_RANGE_DATA, LENGTH (NEAR_RANGE_DATA));

236  1        CALL SMOKE_INIT;
237  1        CALL    GAE_INIT;
238  1        CALL    MATROX_START_UP;
239  1        HIT_PARADE_MEMORY = 0;
240  1        HIT_SHORT_MEMORY = 0;
241  1        DEAD_BIRD_MEMORY = 0;
242  1        H_REP_RQ,V_REP_RQ,H_REP_GO,V_REP_GO,END_REPRISE,BIRD_DT_RDY,BIRD_HITS,
         FELL_SHORT,BIRD_MISSES,BAD_MISS,DATA_RDY1,V_REP_FLAG,H_REP_FLAG,
         START_BIT = 1;

243  1        CALL    INGEST;

244  1        DO FOREVER;

245  2        DO WHILE DATA_RDY1 <> 1;    /* WAIT TILL TOW BOARD HAS TRANSFERED DATA
                                            TO THE FIFO */

246  3        IF BIRD_HITS THEN DO;
248  4            BIRD_HITS_MEM = 1;
249  4            BIRD_HITS = 0;
250  4            CALL HIT_PARADE;
251  4            END;

257  3        IF HIT_KILL THEN DO;
254  4            HIT_KILL_MEM = 1;
255  4            HIT_KILL = 0;
256  4            CALL HIT_PARADE;
257  4            END;

258  3        IF HIT_DISABLE THEN DO;
260  4            HIT_DISABLE_MEM = 1;
261  4            HIT_DISABLE = 0;
262  4            CALL HIT_PARADE;
```

```
263   4          END;

264   3       IF BIRD_MISSES AND GRND_BIRD THEN CALL DEAD_BIRD;
266   3       IF FELL_SHORT THEN CALL HIT_SHORT;
268   3       IF H_REP_RQ THEN CALL HORIZONTAL_REP_RQ;
270   3       IF END_REPRISE THEN CALL RESULTS;
272   3       CALL CHECK_FOR_AUTO_BORE;

273   3       DO WHILE PGM_WAIT;
274   4       CALL CHECK_FOR_AUTO_BORE;
275   4       END;

276   3       END;


277   2       DATA_RDY1 = 0;              /*   RESET TILL SET AGAIN            */
278   2       CALL    PRINT(@POINT_MODE, LENGTH (POINT_MODE));
279   2       CALL GAE_DRIVER;
280   2       CALL OCTAGON_DRIVER;

281   2       END;


282   1       END MAIN_MODULE;



MODULE INFORMATION:

     CODE AREA SIZE     = 0E27H   3623D
     CONSTANT AREA SIZE = 0000H      0D
     VARIABLE AREA SIZE = 000CH     12D
     MAXIMUM STACK SIZE = 001CH     28D
     588 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION
```

APPENDIX D

COMPUTER GENERATED SOUND SYSTEM PROGRAMS

```
LOC  OBJ        SEQ        SOURCE STATEMENT

                  1 ;              17 SEPTEMBER 1982
                  2
                  3
                  4 ;        PROGRAM PRODUCES SOUNDS FOR THE FLIGHT OF A TOW MISSILE
                  5
                  6
                  7
0000              8          ORG     0
0000 040B         9          JMP     0BH
                 10
0003             11          ORG     3
0003 4482        12          JMP     INTERR
                 13
0007             14          ORG     7
0007 1609        15          JTF     TEMAR
0009 693A        16 TEMAR:   JMP     TIMER
                 17
000B             18          ORG     0BH
000B 23FF        19 RESET:   MOV     A,#0FFH
000D 39          20          OUTL    P1,A
000E 3A          21          OUTL    P2,A            ; INITIALIZE PORTS
000F 9AFC        22          ANL     P2,#0FCH        ; SET UP BC1'S  1111/1100
0011 05          23          EN      I               ; ENABLE EXTERNAL INTERRUPT ALLOWING
                 24 ;                                  COMMUNICATION WITH THE HOSE COMPUTER
0012 7422        25          CALL    RESGAA          ; RESET THE SOUND GENERATORS
0014 7427        26          CALL    RESGBB
                 27
0016 B802        28 CLEAR:   MOV     R0,#2           ; RAM POINTER
0018 B93E        29          MOV     R1,#62D         ; RAM COUNTER
001A 27          30 CHASE:   CLR     A
001B A0          31          MOV     @R0,A           ; ERASE ALL RAM LOCATIONS
001C 18          32          INC     R0
001D E91A        33          DJNZ    R1,CHASE
                 34
001F 75          35          ENTO    CLK             ; ENABLE THE 2 MHZ CLOCK FOR THE PSG'S
0020 2348        36          MOV     A,#72D
0022 62          37          MOV     T,A
0023 25          38          EN      TCNTI
                 39
                 40 ;##########################################################################
                 41
                 42 ;       MAIN LOOP ROUTINE WHICH LOOKS AT RB0 REGISTERS R5,R6, AND R7
                 43 ;       AND DETERMINES WHETHER TO MAKE THE FOLLOWING SOUNDS:
                 44 ;              (1)  TRIGGER PULL (R5=FFH)
                 45 ;              (2)  IMPACT-HIT   (R6=FFH)
                 46 ;              (3)  IMPACT-MISS  (R7=FFH)
                 47
0024 FD          48 MALOOP:  MOV     A,R5
0025 C62B        49          JZ      CHKHIT          ; CHECK FOR TRIGGER PULL.  JUMP IF NOT THERE
0027 143F        50          CALL    TRIGER          ; ROUTINE TO MAKE LAUNCH SEQUENCE SOUNDS
0029 BD00        51          MOV     R5,#0           ; CLEAR FLAG REGISTER
002B FE          52 CHKHIT:  MOV     A,R6
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

002C C638         53        JZ      CHKMIS          ; CHECK FOR AN IMPACT-HIT. JUMP IF NOT THERE
002E 5400         54        CALL    HITSOU          ; ROUTINE TO MAKE IMPACT-HIT SOUNDS
0030 BD00         55 DOLOOP: MOV    R5,#0           ; CLEAR ALL FLAG REGISTERS SINCE FLIGHT OVER
0032 BE00         56        MOV     R6,#0
0034 BF00         57        MOV     R7,#0
0036 0424         58        JMP     MALOOP
0038 FF           59 CHKMIS: MOV    A,R7
0039 C624         60        JZ      MALOOP          ; CHECK FOR AN IMPACT-MISS.  JUMP IF NOT THERE
003B 5458         61        CALL    MISSOU          ; ROUTINE TO MAKE IMPACT-MISS SOUNDS
003D 0430         62        JMP     DOLOOP
                  63
                  64 ;*****************************************************************************
                  65
                  66 ;       THIS ROUTINE PROUDUCES THE LAUNCH SEQUENCE SOUNDS IN THIS ORDER
                  67 ;               (1)  GYRO WIND-UP FOR 1.5 SECONDS
                  68 ;               (2)  LAUNCH MOTOR EXPLOSION
                  69 ;               (3)  ROCKET MOTOR BURN OF 1 SECOND
                  70 ;               (4)  FADE AWAY SOUND EFFECT
                  71
                  72 ;----------GYRO SOUND----------
                  73
003F B820         74 TRIGER: MOV    R0,#020H        ; FINE TUNE
0041 B000         75        MOV     @R0,#0
0043 7411         76        CALL    SENDA           ; THIS IS AN ATTEMPT TO DO A "POP"
0045 18           77        INC     R0
0046 B001         78        MOV     @R0,#1          ; COURSE TUNE
0048 7411         79        CALL    SENDA
004A B827         80        MOV     R0,#027H        ; ENABLE
004C B0F6         81        MOV     @R0,#366Q       ; CHANNEL A
004E 7411         82        CALL    SENDA
0050 18           83        INC     R0              ; AMPLITUDE
0051 B00F         84        MOV     @R0,#0FH        ; MAX
0053 7411         85        CALL    SENDA
                  86
0055 BA64         87        MOV     R2,#100D        ; DELAY 100 MSEC
0057 742C         88        CALL    DELAY
                  89
0059 BB64         90        MOV     R3,#100D        ; LOOP COUNTER (DECREMENTS ONCE PER LOOP)
005B B82E         91        MOV     R0,#02EH        ; LOCATION OF FINE TUNE FOR FREQ #1
005D B04C         92        MOV     @R0,#04CH       ; FINE TUNE PERIOD FOR #1 (MUST BE EVEN)
005F 18           93        INC     R0              ; LOCATION OF COURSE TUNE FOR FREQ #1
0060 B001         94        MOV     @R0,#1          ; COURSE TUNE PERIOD FOR #1
0062 B83E         95        MOV     R0,#03EH        ; LOCATION OF FINE TUNE FOR FREQ #2
0064 B0FE         96        MOV     @R0,#0FEH       ; FINE TUNE PERIOD FOR #2 (MUST BE EVEN)
0066 18           97        INC     R0              ; LOCATION OF COURSE TUNE FOR FREQ #2
0067 B000         98        MOV     @R0,#0          ; COURSE TUNE PERIOD FOR #2
                  99
0069 B82E        100        MOV     R0,#02EH        ; INITIALIZE AT FREQ #1
                 101
006B B920        102 GYSET:  MOV     R1,#020H
006D F0          103        MOV     A,@R0           ; GET FINE TUNE
006E A1          104        MOV     @R1,A           ; 20H GETS FINE TUNE VALUE
006F 19          105        INC     R1              ; R1 GETS 21H
0070 18          106        INC     R0
0071 F0          107        MOV     A,@R0           ; GET COURSE TUNE
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

0072 A1         108        MOV     @R1,A          ; 21H GETS COURSE TUNE VALUE
0073 B820       109        MOV     R0,#020H       ; FINE TUNE LOCATION
0075 7411       110        CALL    SENDA
0077 18         111        INC     R0             ; COURSE TUNE LOCATION
0078 7411       112        CALL    SENDA
007A B826       113        MOV     R0,#026H       ; NOISE LOCATION
007C B00F       114        MOV     @R0,#0FH       ; NOISE TO MIDRANGE
007E 7411       115        CALL    SENDA
0080 18         116        INC     R0             ; ENABLE LOCATION
0081 B0F6       117        MOV     @R0,#366Q      ; 11/110/110
0083 7411       118        CALL    SENDA
0085 18         119        INC     R0             ; AMP LOCATION 28H FOR CHANNEL A
0086 B00D       120        MOV     @R0,#0DH       ; MAX AMP
0088 7411       121        CALL    SENDA
                122
008A BA0E       123        MOV     R2,#14D        ; CALL 14 msec DELAY PER LOOP
008C 742C       124        CALL    DELAY          ; TOTAL LOOP TIME = .014 x (R3) = 1.4 SEC
                125
008E EB94       126        DJNZ    R3,GYDOC       ; JUMP TO CHANGE FREQUENCIES
0090 7422       127        CALL    RESGAA         ; OTHERWISE SILENCE THE PSG AND
0092 2400       128        JMP     LEXPLO         ; JUMP TO LAUNCH EXPLOSION
                129
0094 FB         130 GYDOC: MOV     A,R3
0095 129D       131        JB0     GYRAA          ; CHANGE FREQUENCIES
0097 B82E       132        MOV     R0,#02EH
0099 14A3       133        CALL    GYRBB
009B 046B       134        JMP     GYSET
009D B83E       135 GYRAA: MOV     R0,#03EH
009F 14A3       136        CALL    GYRBB
00A1 046B       137        JMP     GYSET
                138
00A3 F0         139 GYRBB: MOV     A,@R0          ; NEXT TIME THRU LOOP FREQ'S #1 & #2 WILL
00A4 96AC       140        JNZ     GYRCC          ; SOUND HIGHER
00A6 18         141        INC     R0
00A7 F0         142        MOV     A,@R0          ; !!!! FREQ PERIODS MUST BE GREATER THAN
00A8 07         143        DEC     A              ; 200 OR THE SOUND WILL MESS UP WHEN OVERFLOW
00A9 A0         144        MOV     @R0,A          ; OCCURS HERE!!!!
00AA C8         145        DEC     R0
00AB F0         146        MOV     A,@R0
00AC 07         147 GYRCC: DEC     A
00AD 07         148        DEC     A
00AE A0         149        MOV     @R0,A          ; DECREMENT EACH FREQUENCY BY 2
00AF 93         150        RETR
                151
                152 ;**********************************************************************************
                153
0100            154        ORG     100H
                155
                156 ;*****************************
                157
                158 ;      LAUNCH EXPLOSION FOR -----  PSG-B , CHANNEL-A
                159
0100 7427       160 LEXPLO: CALL   RESGBB         ; CLEAR OUT PSG-B
                161
0102 80         162        MOVX    A,@R0          ; START PULSE TO TIMER
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

                163
0103 B826       164        MOV     R0,#026H        ; NOISE LOCATION
0105 B01F       165        MOV     @R0,#01FH       ; NOISE PERIOD VALUE
0107 7400       166        CALL    SENDB
0109 18         167        INC     R0              ; ENABLE LOCATION
010A B0F7       168        MOV     @R0,#367Q       ; 11/110/111
010C 7400       169        CALL    SENDB
010E 18         170        INC     R0              ; AMP LOCATION FOR CHANNEL A
010F B010       171        MOV     @R0,#10H        ; ENABLE ENVELOPE
0111 7400       172        CALL    SENDB
0113 B82B       173        MOV     R0,#02BH        ; FINE TUNE ENVELOPE LOCATION
0115 B000       174        MOV     @R0,#0          ; FINE TUNE ENV VALUE
0117 7400       175        CALL    SENDB
0119 18         176        INC     R0              ; COURSE TUNE ENVELOPE AT 2CH
011A B020       177        MOV     @R0,#020H       ; COURSE TUNE VALUE
011C 7400       178        CALL    SENDB
011E 18         179        INC     R0              ; ENVELOPE SHAPE/CYCLE
011F B000       180        MOV     @R0,#0          ; DECAY
0121 7400       181        CALL    SENDB
                182
0123 55         183        STRT    T
                184
0124 B907       185        MOV     R1,#7D          ; DELAY 0.8 SEC TILL BURN
0126 7433       186        CALL    DALAY
                187
                188 ;**********************************************************************
                189
                190 ;       ROCKET BURN ROUTINE ----  PSG-A , CHANNEL-B
                191
0128 7427       192        CALL    RESGBB          ; SILENCE B
012A 7422       193        CALL    RESGAA          ; CLEAR OUT PSG-A
                194
012C BBA0       195        MOV     R3,#160D        ; LOOP COUNTER
012E B829       196        MOV     R0,#029H        ; AMP LOCATION FOR B
0130 B00F       197        MOV     @R0,#0FH        ; INITIAL VALUE
0132 A5         198        CLR     F1              ; AMP FLAG
                199
0133 B826       200 BURN:  MOV     R0,#026H        ; NOISE LOCATION
0135 B01F       201        MOV     @R0,#1FH        ; NOISE PERIOD VALUE
0137 7411       202        CALL    SENDA
0139 18         203        INC     R0              ; ENABLE LOCATION 27H
013A B0EF       204        MOV     @R0,#357Q       ; 11/101/111
013C 7411       205        CALL    SENDA
013E B829       206        MOV     R0,#029H        ; AMP LOCATION 29H FOR CHANNEL B
0140 FB         207        MOV     A,R3
0141 924D       208        JB4     NATT
0143 764E       209        JF1     CHAF
0145 F0         210        MOV     A,@R0
0146 C64E       211        JZ      CHAF
0148 07         212        DEC     A
0149 A0         213        MOV     @R0,A
014A B5         214        CPL     F1
014B 244E       215        JMP     CHAF
014D A5         216 NATT:  CLR     F1
014E 7411       217 CHAF:  CALL    SENDA
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

                 218
0150 BA0A        219        MOV     R2,#10D         ; CALL A 10 MSEC DELAY PER LOOP
0152 742C        220        CALL    DELAY
                 221
0154 EB33        222        DJNZ    R3,BURN
                 223
                 224 ;*************************************************************************
                 225
                 226 ;      FADE AWAY ROUTINE ON PSG-B CHANNEL B
                 227
0156 7422        228        CALL    RESGAA
0158 7422        229        CALL    RESGAA
015A A5          230        CLR     F1              ; AMP CONTROL FLAG
015B BBFE        231        MOV     R3,#0FEH        ; LOOP COUNTER
015D B822        232        MOV     R0,#022H        ; LOCATION OF FINE TUNE
015F B000        233        MOV     @R0,#0          ; INITIAL VALUE
0161 B829        234        MOV     R0,#029H        ; LOCATION OF AMP CHANNEL B
0163 B00F        235        MOV     @R0,#0FH        ; INITIAL AMP TO MAX
                 236
0165 B822        237 WHISTL: MOV     R0,#022H        ; FINE TUNE LOCATION
0167 10          238        INC     @R0             ; INCREMENT FINE TUNE PER LOOP
0168 7400        239        CALL    SENDB
016A 18          240        INC     R0              ; COURSE TUNE LOCATION
016B B001        241        MOV     @R0,#1          ; COURSE TUNE VALUE
016D 7400        242        CALL    SENDB
016F B826        243        MOV     R0,#026H        ; NOISE PERIOD LOCATION
0171 B00F        244        MOV     @R0,#0FH        ; MIDRANGE
0173 7400        245        CALL    SENDB
0175 18          246        INC     R0              ; ENABLE REGISTER LOCATION
0176 B0ED        247        MOV     @R0,#355Q       ; 11/101/101
0178 7400        248        CALL    SENDB
                 249
017A B829        250        MOV     R0,#029H        ; AMP LOCATION FOR CHANNEL A
017C FB          251        MOV     A,R3            ; DECREMENT AMP FOR EACH CHANGE IN B4
017D 9289        252        JB4     NOCH
017F 768A        253        JF1     CHAN
0181 F0          254        MOV     A,@R0           ; ENSURE ITS NOT ZERO
0182 C68A        255        JZ      CHAN
0184 07          256        DEC     A
0185 A0          257        MOV     @R0,A
0186 B5          258        CPL     F1              ; FLAG NOT TO DECREMENT
0187 248A        259        JMP     CHAN
0189 A5          260 NOCH:  CLR     F1
018A 7400        261 CHAN:  CALL    SENDB
                 262
018C BA0A        263        MOV     R2,#10D
018E 742C        264        CALL    DELAY           ; DELAY PER LOOP = 10 MSEC
                 265
0190 EB97        266        DJNZ    R3,WHAFFL
0192 7422        267        CALL    RESGAA          ; CLEAR BOTH PSG'S
0194 7427        268        CALL    RESGBB
0196 93          269        RETR
                 270
0197 2465        271 WHAFFL: JMP     WHISTL
                 272
```

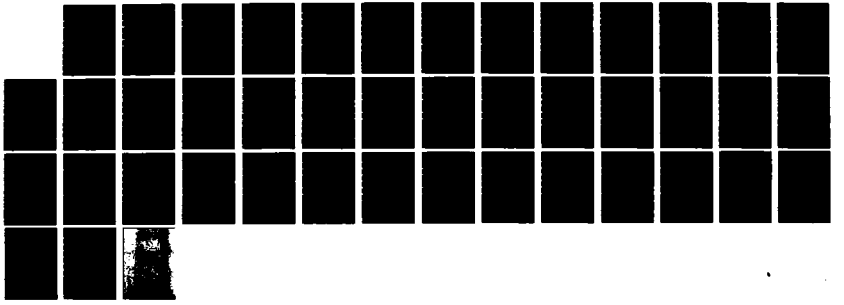
```
LOC  OBJ           SOURCE STATEMENT

          ;********************************************************************
0200                     ORG     200H

          ;************************
0200 80        HITSOU:  MOVX    A,@R0           ; STOP PULSE TO TIMER

0201 B826               MOV     R0,#026H        ; NOISE LOCATION
0203 B01F               MOV     @R0,#01FH       ; VALUE
0205 7411               CALL    SENDA
0207 18                 INC     R0              ; ENABLE LOCATION
0208 B00F               MOV     @R0,#337Q       ; 11/011/111
020A 7411               CALL    SENDA
020C B82A               MOV     R0,#02AH        ; CHANNEL C AMP LOCATION
020E B004               MOV     @R0,#4          ; FIRST EXPLOSION AMP
0210 7411               CALL    SENDA

0212 BAFA               MOV     R2,#250D        ; 250 MSEC DELAY
0214 742C               CALL    DELAY

0216 7422               CALL    RESGAA

0218 B826               MOV     R0,#026H        ; NOISE
021A B01F               MOV     @R0,#01FH
021C 7411               CALL    SENDA
021E 18                 INC     R0              ; ENABLE
021F B00F               MOV     @R0,#337Q       ; 11/011/111
0221 7411               CALL    SENDA
0223 B82A               MOV     R0,#02AH        ; AMP FOR C
0225 B00A               MOV     @R0,#0AH
0227 7411               CALL    SENDA

0229 BAFA               MOV     R2,#250D        ; 250 MSEC DELAY
022B 742C               CALL    DELAY

022D 7422               CALL    RESGAA

022F B826               MOV     R0,#026H        ; NOISE
0231 B01F               MOV     @R0,#01FH
0233 7411               CALL    SENDA
0235 18                 INC     R0              ; ENABLE
0236 B00F               MOV     @R0,#337Q       ; 11/011/111
0238 7411               CALL    SENDA
023A B82A               MOV     R0,#02AH        ; AMP FOR C
023C B010               MOV     @R0,#010H       ; ENABLE ENVELOPE
023E 7411               CALL    SENDA
0240 18                 INC     R0              ; FINE TUNE ENV
0241 B000               MOV     @R0,#0
0243 7411               CALL    SENDA
0245 18                 INC     R0              ; COURSE TUNE ENV
0246 B040               MOV     @R0,#040H
0248 7411               CALL    SENDA
024A 18                 INC     R0              ; SHAPE/CYCLE
024B B000               MOV     @R0,#0
```

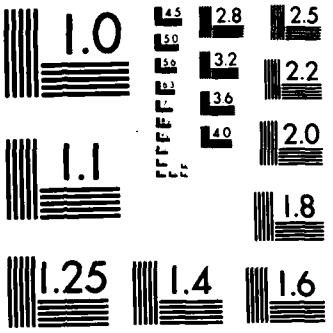
```
LOC  OBJ        SEQ        SOURCE STATEMENT

024D 7411        328        CALL    SENDA
                 329
024F B91E        330        MOV     R1,#30D         ; CALL A 3 SEC DELAY
0251 7433        331        CALL    DALAY
                 332
0253 7422        333        CALL    RESGAA
0255 7427        334        CALL    RESGBB
0257 93          335        RETR
                 336
                 337 ;*****************************************************************************
                 338
                 339 ;       SOUND TO INDICATE A GROUND IMPACT OR MISS
                 340
0258 80          341 MISSOU: MOVX    A,@R0           ; STOP PULSE TO TIMER
                 342
0259 B826        343        MOV     R0,#026H        ; EXPLOSION TO APPEAR ON PSG-E CHANNEL-C
025B B01F        344        MOV     @R0,#01FH
025D 7400        345        CALL    SENDB
025F 18          346        INC     R0              ; ENABLE LOCATION
0260 B00F        347        MOV     @R0,#3370       ; 11/011/111
0262 7400        348        CALL    SENDB
0264 B82A        349        MOV     R0,#02AH        ; AMP LOCATION FOR C
0266 B010        350        MOV     @R0,#010H       ; ENABLE ENVELOPE
0268 7400        351        CALL    SENDB
026A 18          352        INC     R0              ; ENVELOPE FINE TUNE LOCATION
026B B000        353        MOV     @R0,#0          ; VALUE
026D 7400        354        CALL    SENDB
026F 18          355        INC     R0              ; ENVELOPE COURSE TUNE LOCATION
0270 B040        356        MOV     @R0,#040H       ; VALUE
0272 7400        357        CALL    SENDB
0274 18          358        INC     R0              ; ENVELOPE SHAPE/CYCLE LOCATION
0275 B000        359        MOV     @R0,#0
0277 7400        360        CALL    SENDB
                 361
0279 B91E        362        MOV     R1,#30D         ; CALL A 2.0 SECOND DELAY
027B 7433        363        CALL    DALAY
                 364
027D 7422        365        CALL    RESGAA
027F 7427        366        CALL    RESGBB
0281 93          367        RETR
                 368
                 369 ;*****************************************************************************
                 370
                 371 ;       INTERRUPT ROUTINE TO SET REGISTERS RB0 R5,R6,R7
                 372
                 373 ;       FOR ACC = 11 ---> RB0 R5 GETS FFH       DESIGNATES TRIGGER PULL
                 374 ;         "   "  = 10 --->  "  R6  "  FFH         "      IMPACT-HIT
                 375 ;         "   "  = 01 --->  "  R7  "  FFH         "      IMPACT-MISS
                 376
                 377 ;                         RB0 R4 RESERVED HERE
                 378
0282 C5          379 INTERR: SEL     RB0     ; ENSURE REGISTER BANK 0
0283 AC          380        MOV     R4,A    ; TEMPORARY STORAGE
0284 09          381        IN      A,P1    ; GRAB DATA
0285 328E        382        JB1     INCONT
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

0287 1297        383        JB0     IMPMIS  ; ACC = 01
0289 4499        384        JMP     INTRET
028B 128F        385 INCONT: JB0     INTRIG  ; ACC = 11
028D 4493        386        JMP     IMPHIT  ; ACC = 10
                 387
028F BDFF        388 INTRIG: MOV     R5,#0FFH        ; FOR TRIGGER PULL
0291 4499        389        JMP     INTRET
0293 BEFF        390 IMPHIT: MOV     R6,#0FFH        ; FOR IMPACT-HIT
0295 4499        391        JMP     INTRET
0297 BFFF        392 IMPMIS: MOV     R7,#0FFH        ; FOR IMPACT-MISS
                 393
0299 FC          394 INTRET: MOV     A,R4            ; RESTORE ACCUMULATOR
029A BC14        395 ILOOPA: MOV     R4,#20D         ; DELAY 100 usec TO ALLOW INTERRUPT LINE
029C EC9C        396 INLOOP: DJNZ    R4,INLOOP       ; TO RETURN HIGH
029E 869A        397        JNI     ILOOPA          ; KEEP LOOPING TILL INTERRUPT RETURNS HIGH
02A0 93          398        RETR
                 399
                 400 ;*****************************************************************************
                 401
0300             402        ORG     300H
                 403
                 404 ;       SUBROUTINE TO SEND DATA TO PSG-B
                 405
0300 9A7F        406 SENDB:  ANL     P2,#07FH        ; CHIP SELECT PSG-B  (0111/1111)
                 407
0302 F0          408        MOV     A,@R0
0303 A9          409        MOV     R1,A            ; R1 GETS DATA
0304 F8          410        MOV     A,R0
0305 530F        411        ANL     A,#0FH          ; ACC GETS ADDRESS OF PSG REGISTER
                 412
0307 8A02        413        ORL     P2,#2           ; PIN BC1 SETS PSG TO 'READ'
0309 90          414        MOVX    @R0,A           ; 'LATCH ADDRESS' TOGGLED BY WRITE PIN ON 8
030A 9AFD        415        ANL     P2,#0FDH        ; PIN BC1 SETS PSG TO 'INACTIVE'
030C F9          416        MOV     A,R1            ; GET DATA
030D 90          417        MOVX    @R0,A           ; 'WRITE' TO PSG
                 418
030E 8A80        419        ORL     P2,#080H        ; DESELECT PSG-B
0310 93          420        RETR
                 421
                 422 ;       SUBROUTINE TO SEND DATA TO PSG-A
                 423
0311 9ADF        424 SENDA:  ANL     P2,#0DFH        ; CHIP SELECT (1101/1111)
                 425
0313 F0          426 SANAA:  MOV     A,@R0
0314 A9          427        MOV     R1,A            ; R1 GETS DATA
0315 F8          428        MOV     A,R0
0316 530F        429        ANL     A,#0FH          ; ACC GETS ADDRESS
                 430
0318 8A01        431        ORL     P2,#1           ; PSG TO 'READ' STATE
031A 90          432        MOVX    @R0,A           ; PSG TO 'LATCH ADDRESS' STATE
031B 9AFE        433        ANL     P2,#0FEH        ; BACK TO 'INACTIVE' STATE
031D F9          434        MOV     A,R1
031E 90          435        MOVX    @R0,A           ; TOGGLE PSG THRU 'WRITE' CYCLE
                 436
031F 8A20        437        ORL     P2,#020H        ; DESELECT PSG-B
```

AD-A143 646 SIMULATED TANK ANTI-ARMOR GUNNERY SYSTEM (STAGS-TOW) 3/3
(U) NAVAL TRAINING EQUIPMENT CENTER ORLANDO FL
A MARSHALL ET AL. MAY 83 PMT-E-T-6605.83

UNCLASSIFIED F/G 19/5 NL





```
LOC  OBJ          SEQ     SOURCE STATEMENT

0321 93           438           RETR
                  439
                  440 ;*****************************************************************************
                  441
                  442 ;       ROUTINE TO RESET SOUND GENERATOR "A"                    1110/1111
                  443
0322 9AEF         444 RESGAA: ANL     P2,#0EFH        ; RESET PSG-A                 0001/0000
0324 8A10         445         ORL     P2,#010H        ; COMPLETE RESET CYCLE
0326 93           446         RETR
                  447
                  448 ;       ROUTINE TO RESET SOUND GENERATOR "B"                    1011/1111
                  449
0327 9ABF         450 RESGBB: ANL     P2,#0BFH        ; RESET PSG-B                 0100/0000
0329 8A40         451         ORL     P2,#040H        ; COMPLETE RESET CYCLE
032B 93           452         RETR
                  453
                  454 ;       DELAY ROUTINE  --  DELAY = R2 X .001 SECONDS
                  455
032C B8C8         456 DELAY:  MOV     R0,#0C8H        ; R0<--200D
032E E82E         457 DLOOP:  DJNZ    R0,DLOOP        ; 200 X 5 MICRO-SEC PER INSTRUCTION
0330 EA2C         458         DJNZ    R2,DELAY        ; R2 X 1 MICRO-SEC
0332 93           459         RETR
                  460
                  461 ;       DELAY ROUTINE  --  DELAY = R1 X .100 SECONDS
                  462
0333 BA64         463 DALAY:  MOV     R2,#100D        ; CALL UP A 100 MILSEC DELAY
0335 742C         464         CALL    DELAY
0337 E933         465         DJNZ    R1,DALAY
0339 93           466         RETR
                  467
                  468 ;*****************************************************************************
                  469
                  470 TIMER:  SEL     RB1             ; TEMP STORE
033A D5           471         MOV     R2,A
033B AA           472
                  473         MOV     A,#72D          ; REINITIALIZE TIMER
033C 2348         474         MOV     T,A
033E 62           475
                  476         INC     R0
033F 18           477         MOV     A,R0            ; INCREMENT UPPER BYTE OF TIMER
0340 F8           478         JZ      TIMINC          ; CHECK FOR 20 SEC
0341 C649         479         MOV     A,R1
0343 F9           480         JB2     TIMOUT
0344 524C         481 TIMRET: MOV     A,R2
0346 FA           482         SEL     RB0
0347 C5           483         RETR
0348 93           484                                 ; INCREMENT UPPER BYTE OF TIMER
                  485 TIMINC: INC     R1
0349 19           486         JMP     TIMRET
034A 6446         487
                  488 TIMOUT: MOVX    A,@R0           ; STOP TIMER
034C 80           489 INFINL: JMP     INFINL          ; INFIN LOOP TILL RESET
034D 644D         490
                  491
                  492         END
```

```
BURN   0133   CHAN   018A   CHAF   014E   CHASE  001A   CHKHIT 002B   CHKMIS 0038   CLEAR  0016   DALAY  0333
DELAY  032C   DLOOP  032E   DOLOOF 0030   GYDOC  0094   GYRAA  009D   GYRBB  00A3   GYRCC  00AC   GYSET  006B
HITSOU 0200   ILOOPA 029A   IMPHIT 0293   IMPMIS 0297   INCONT 02BB   INFINL 034D   INLOOP 029C   INTERR 0282
INTRET 0299   INTRIG 028F   LEXPLO 0100   MALOOP 0024   MISSOU 0258   NATT   014D   NOCH   0189   RESET  0008
RESGAA 0322   RESGBB 0327   SANAA  0313   SENDA  0311   SENDB  0300   TEMAR  0009   TIMER  033A   TIMINC 0349
TIMOUT 034C   TIMRET 0346   TRIGER 003F   WHAFFL 0197   WHISTL 0165

ASSEMBLY COMPLETE, NO ERRORS
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

                  1 ;                    5 NOV 1982
                  2
                  3
                  4 ;       RETURN FIRE ROUTINE FOR STAGS-TOW
                  5
                  6
0000              7          ORG     0
0000 040B         8          JMP     0BH
                  9
                 10
0003             11          ORG     3
0003 0427        12          JMP     REFIRE
                 13
                 14
0007             15          ORG     7
0007 1609        16          JTF     TEMAP
0009 0427        17 TEMAP:   JMP     REFIRE
                 18
                 19
000B             20          ORG     0BH
000B 23FF        21          MOV     A,#0FFH
000D 39          22          OUTL    P1,A            ; INITIALIZE THE PORTS
000E 3A          23          OUTL    P2,A
                 24
000F 99FE        25          ANL     P1,#0FEH        ; SET BC1 LOW
0011 9AFE        26          ANL     P2,#0FEH        ; RESET THE PSG
0013 8A01        27          ORL     P2,#1
0015 75          28          ENTO    CLK             ; ENABLE CLOCK FOR PSG
                 29
0016 23FF        30          MOV     A,#0FFH
0018 62          31          MOV     T,A             ; ENABLE TIMER/COUNTER INTERRUPT
0019 25          32          EN      TCNTI
001A 45          33          STRT    CNT
001B 05          34          EN      I
                 35
001C B802        36          MOV     R0,#2           ; CLEAR OUT ALL RAM LOCATIONS
001E B93E        37          MOV     R1,#62D
0020 27          38 CLEAR:   CLR     A
0021 A0          39          MOV     @R0,A
0022 18          40          INC     R0
0023 E920        41          DJNZ    R1,CLEAR
                 42
0025 0425        43 LOOK:    JMP     LOOK            ; LOOP FOR RETURN FIRE CALLED
                 44
0027 A5          45 REFIRE:  CLR     F1              ; AMPLITUDE FLAG
0028 BFFE        46          MOV     R7,#0FEH        ; LOOP COUNTER
002A BE01        47          MOV     R6,#1           ; INITIAL FINE TUNE VALUE FOR WHISTLE
002C BD0F        48          MOV     R5,#0FH         ; INITIAL AMPLITUDE FOR WHISTLE
                 49
002E 8901        50 WHISTL:  ORL     P1,#1           ; BC1 SETS PSG TO 'READ'
0030 27          51          CLR     A
0031 90          52          MOVX    @R0,A           ; LATCH FINE TUNE ADDRESS
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

0032 99FE          53          ANL     P1,#0FEH
0034 FE            54          MOV     A,R6
0035 90            55          MOVX    @R0,A       ; FINE TUNE VALUE
                   56
0036 8901          57          ORL     P1,#1
0038 2301          58          MOV     A,#1
003A 90            59          MOVX    @R0,A       ; LATCH COURSE TUNE ADDRESS
003B 99FE          60          ANL     P1,#0FEH
003D 2301          61          MOV     A,#1
003F 90            62          MOVX    @R0,A       ; DATA FOR COURSE TUNE
                   63
0040 8901          64          ORL     P1,#1
0042 2306          65          MOV     A,#6
0044 90            66          MOVX    @R0,A       ; NOISE PERIOD ADDRESS
0045 99FE          67          ANL     P1,#0FEH
0047 230F          68          MOV     A,#0FH      ; MIDRANGE
0049 90            69          MOVX    @R0,A
                   70
004A 8901          71          ORL     P1,#1
004C 2307          72          MOV     A,#7
004E 90            73          MOVX    @R0,A       ; ENABLE REGISTER ADDRESS
004F 99FE          74          ANL     P1,#0FEH
0051 23F6          75          MOV     A,#366Q
0053 90            76          MOVX    @R0,A       ; CHANNEL A ONLY
                   77
0054 FF            78          MOV     A,R7
0055 9260          79          JB4     NOCH        ; JUMP FOR EACH 1/16 OF TOTAL LOOP
0057 7661          80          JF1     CHAN        ; PREVENTS OVER DECREMENTING
0059 FD            81          MOV     A,R5        ; GET AMP
005A C661          82          JZ      CHAN        ; CHECK FOR ZERO
005C C0            83          DEC     R5          ; OTHERWISE DECREMENT AMPLITUDE
005D B5            84          CPL     F1
005E 0461          85          JMP     CHAN
0060 A5            86 NOCH:    CLR     F1
                   87
0061 8901          88 CHAN:    ORL     P1,#1
0063 2308          89          MOV     A,#8
0065 90            90          MOVX    @R0,A       ; AMPLITUDE LOCATION
0066 99FE          91          ANL     P1,#0FEH
0068 FD            92          MOV     A,R5
0069 90            93          MOVX    @R0,A
                   94
006A BA05          95          MOV     R2,#5       ; CALL A 5msec DELAY PER LOOP
006C 1485          96          CALL    DELAY
                   97
006E 1E            98          INC     R6          ; INCREMENT FINE TUNE
006F EF2E          99          DJNZ    R7,WHISTL
                  100
0071 9AFE         101 EXPLO:   ANL     P2,#0FEH    ; ERASE ALL LOCATIONS
0073 8A01         102          ORL     P2,#1
                  103
0075 8901         104          ORL     P1,#1
0077 2306         105          MOV     A,#6
0079 90           106          MOVX    @R0,A       ; LATCH NOISE ADDRESS
007A 99FE         107          ANL     P1,#0FEH
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT
007C 231F          108          MOV     A,#01FH
007E 90            109          MOVX    @R0,A
                   110
007F 8901          111          ORL     P1,#1
0081 2307          112          MOV     A,#7
0083 90            113          MOVX    @R0,A          ; ENABLE CHANNEL B ONLY
0084 99FE          114          ANL     P1,#0FEH
0086 23ED          115          MOV     A,#355Q        ; 11/101/101
0088 90            116          MOVX    @R0,A
                   117
0089 8901          118          ORL     P1,#1
008B 2309          119          MOV     A,#9           ; CHANNEL B
008D 90            120          MOVX    @R0,A
008E 99FE          121          ANL     P1,#0FEH
0090 2310          122          MOV     A,#010H        ; ENABLE ENVELOPE
0092 90            123          MOVX    @R0,A
                   124
0093 8901          125          ORL     P1,#1
0095 230B          126          MOV     A,#0BH
0097 90            127          MOVX    @R0,A          ; FINE TUNE ADDRESS
0098 99FE          128          ANL     P1,#0FEH
009A 27            129          CLR     A
009B 90            130          MOVX    @R0,A
                   131
009C 8901          132          ORL     P1,#1
009E 230C          133          MOV     A,#0CH
00A0 90            134          MOVX    @R0,A          ; COURSE TUNE ADDRESS
00A1 99FE          135          ANL     P1,#0FEH
00A3 2340          136          MOV     A,#040H
00A5 90            137          MOVX    @R0,A
                   138
00A6 8901          139          ORL     P1,#1
00A8 230D          140          MOV     A,#0DH
00AA 90            141          MOVX    @R0,A          ; ENVELOPE SHAPE/CYCLE
00AB 99FE          142          ANL     P1,#0FEH
00AD 27            143          CLR     A
00AE 90            144          MOVX    @R0,A
                   145
00AF BAFF          146          MOV     R2,#0FFH
00B1 14B5          147          CALL    DELAY          ; DELAY 255 MSEC
                   148
00B3 04B3          149 OLOOP:   JMP     OLOOP
                   150
00B5 B8C8          151 DELAY:   MOV     R0,#0C8H       ; DELAY 1 msec PER R2 BIT
00B7 E8B7          152 DLOOP:   DJNZ    R0,DLOOP
00B9 EAB5          153          DJNZ    R2,DELAY
00BB 93            154          RETR
                   155
                   156
                   157          END
USER SYMBOLS
CHAN   0061    CLEAR  0020    DELAY  00B5    DLOOP  00B7    EXPLO  0071    LOOK   0025    MOCH   0060    OLOOP  00B3
REFIRE 0027    TEMAR  0009    WHISTL 002E

ASSEMBLY COMPLETE,  NO ERRORS
```

APPENDIX E

TERRAIN BOARD MODEL CONTROL PROGRAMS

```
LOC  OBJ          SEQ        SOURCE STATEMENT

                    1
                    2 ;             DECEMBER 15, 1982
                    3
                    4
                    5 ;       THIS PROGRAM IS DESIGNED TO PASS DATA FROM A HOST COMPUTER TO 5
                    6 ;       STEPPER MOTOR CONTROLLERS (CY512-1 THRU -5).  IN TURN, THE 8741
                    7 ;       uC PASSES POSITIONAL INFORMATION IN THE FORM OF A 16 BIT COUNT TO
                    8 ;       THE HOST FOR EACH CONTROLLER WHEN REQUESTED.
                    9
                   10 ;       PORT1: (OUTPUT)
                   11 ;              P10-P16         DATA BUS TO CY512'S
                   12 ;              P17             N/C
                   13
                   14 ;       PORT2: (CONTROL)
                   15 ;              P20-P23         BUS TO 8243 I/O EXPANDER'S A & B
                   16 ;              P24-P25         N/C
                   17 ;              P26             CS/ FOR 8243-A
                   18 ;              P27             CS/ FOR 8243-B
                   19
                   20 ;       8243-A PORT FUNCTIONS ARE AS FOLLOWS:
                   21
                   22 ;       P40 <-- CHIP SELECT CY512-1        P50 <-- PULSE OUTPUT FROM CY512-1
                   23 ;       P41 <--  '    '   CY512-2        P51 <--   '    '    '   CY512-2
                   24 ;       P42 <--  '    '   CY512-3        P52 <--   '    '    '   CY512-3
                   25 ;       P43 <--  '    '   CY512-4        P53 <--   '    '    '   CY512-4
                   26
                   27 ;       P60 <-- DIRECTION FROM CY512-1     P70 <-- BUSY/READY FROM CY512-1
                   28 ;       P61 <--     '     CY512-2        P71 <--  '  /  '    '   CY512-2
                   29 ;       P62 <--     '     CY512-3        P72 <--  '  /  '    '   CY512-3
                   30 ;       P63 <--     '     CY512-4        P73 <--  '  /  '    '   CY512-4
                   31
                   32 ;       8243-B PORT FUNCTIONS ARE AS FOLLOWS:
                   33
                   34 ;       P40 <-- SOFTWARE RESET CY512-1     P50 <-- RUN SIGNAL FROM CY512-1
                   35 ;       P41 <--    '       CY512-2        P52 <--  '    '    '   CY512-2
                   36 ;       P42 <--    '       CY512-3        P53 <--  '    '    '   CY512-3
                   37 ;       P43 <--    '       CY512-4        P54 <--  '    '    '   CY512-4
                   38
                   39 ;       P60 <-- BUSY/READY FROM CY512-5    P70 <-- SOFTWARE RESET CY512-5
                   40 ;       P61 <-- DIRECTION FROM CY512-5     P71 <-- RUN SIGNAL FROM CY512-5
                   41 ;       P62 <-- CHIP SELECT CY512-5        P72 <-- N/C
                   42 ;       P63 <-- PULSE OUTPUT FROM CY512-5  P73 <-- N/C
                   43
                   44 ;       TEST PINS:  T1 USED AS INTERRUPT TO COUNT THE POSITION OF THE CY512
                   45
                   46 ;*******************************************************************************
                   47
0000               48         ORG     0                       ;RESERVE INTERRUPT LOCATIONS
0000 0408          49         JMP     0BH
                   50
0003               51         ORG     3
0003 047C          52         JMP     COUNT           ; COUNT ROUTINE
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

                 53
000B             54        ORG     0BH
000B 23FF        55        MOV     A,#0FFH         ; INITIALIZE PORTS 1 & 2
000D 39          56        OUTL    P1,A
000E 3A          57        OUTL    P2,A
                 58
000F 9ABF        59        ANL     P2,#0BFH        ; SELECT 8243-A  1011/1111
0011 230F        60        MOV     A,#0FH
0013 3C          61        MOVD    P4,A            ; WRITE F TO OUTPUT PORT FOR CHIP SELECT
0014 3D          62        MOVD    P5,A            ; INPUT PORT FOR PULSE
0015 3E          63        MOVD    P6,A            ; INPUT PORT FOR DIRECTION
0016 3F          64        MOVD    P7,A            ; INPUT PORT FOR BUSY/READY
0017 8A40        65        ORL     P2,#040H        ; DESELECT 8243-A  0100/0000
0019 9A7F        66        ANL     P2,#07FH        ; SELECT 8243-B    0111/1111
001B 230F        67        MOV     A,#0FH
001D 3C          68        MOVD    P4,A            ; WRITE F TO SOFTWARE RESET LINES
001E 3D          69        MOVD    P5,A            ; INPUT PORT FOR RUN
001F 3E          70        MOVD    P6,A            ; INPUT PORT FOR CY512-5
0020 3F          71        MOVD    P7,A            ; OUTPUT PORT FOR CY512-5
0021 8A80        72        ORL     P2,#080H        ; DESELECT 8243-B  1000/0000
                 73
0023 B802        74        MOV     R0,#2           ; RAM LOCATION POINTER
0025 B93E        75        MOV     R1,#62D         ; COUNTER
0027 27          76 CLEAR: CLR     A
0028 A0          77        MOV     @R0,A           ; CLEAR OUT ALL RAM LOCATIONS
0029 18          78        INC     R0
002A E927        79        DJNZ    R1,CLEAR
002C A8          80        MOV     R0,A
002D A9          81        MOV     R1,A
                 82
002E 34C4        83        CALL    RESET           ; RESET ALL THE CY512'S
                 84
                 85 ;*************************************************************************************
                 86
0030 D630        87 START: JNIBF   START           ; MAIN LOOP ROUTINE
0032 22          88        IN      A,DBB
0033 AA          89        MOV     R2,A            ; TEMP STORAGE OF DATA
0034 D354        90        XRL     A,#054H         ; CHECK FOR A 'T' TO GET PRIORITY TRACK
0036 C63A        91        JZ      PRIOR
0038 2400        92        JMP     ISERV           ; JUMP IF TRACK NUMBER OR 'P'
                 93
003A D63A        94 PRIOR: JNIBF   PRIOR           ; LOOP TILL PRIORITY TRACK NUMBER APPEARS
003C 22          95        IN      A,DBB
003D 5418        96        CALL    XLATE           ; GET MASK IN R6
003F FE          97        MOV     A,R6
0040 D5          98        SEL     RB1
0041 AD          99        MOV     R5,A            ; SET MASK IN R5
0042 C5         100        SEL     RB0
0043 0430       101        JMP     START
                102
                103 ;*************************************************************************************
                104
                105 ;       THIS ROUTINE KEEPS COUNT OF THE PRIORITY TRACK AND SENDS THIS COUNT
                106 ;       WHEN CALLED FOR (MSB FIRST).  IF A 'P' IS SENT THEN THE MOTORS STOP.
                107
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

0045 9ABF       108 TEST:  ANL    P2,#0BFH      ; SELECT 8243-A
0047 D5         109        SEL    RB1
0048 0D         110 PALOOP: MOVD  A,P5          ; GET PULSE DATA
0049 5D         111        ANL    A,R5          ; PRIORITY MASK OFF
004A C662       112        JZ     PAGOON        ; JUMP OUT OF LOOP WHEN PULSE GOES LOW
004C D648       113        JNIBF  PALOOP        ; LOOP IF NO INTERR AND PULSE IS HI
                114
004E 22         115        IN     A,DBB
004F D343       116        XRL    A,#043H       ; CHECK FOR AN 'R' TO STOP MOTORS
0051 965D       117        JNZ    MOTOSP
0053 FE         118        MOV    A,R6          ; OUTPUT UPPER BYTE OF COUNTER
0054 02         119        OUT    DBB,A
0055 8655       120 EELEE: JOBF   EELEE
0057 FC         121        MOV    A,R4          ; OUTPUT LOWER BYTE OF COUNTER
0058 02         122        OUT    DBB,A
0059 8659       123 QQLQQ: JOBF   QQLQQ
005B 0448       124        JMP    PALOOP
                125
005D 34C4       126 MOTOSP: CALL  RESET         ; ROUTINE TO STOP MOTORS
005F C5         127        SEL    RB0
0060 0430       128        JMP    START
                129
0062 0E         130 PAGOON: MOVD  A,P6          ; GET DIRECTION
0063 5D         131        ANL    A,R5
0064 9670       132        JNZ    CLOCKW        ; JUMP IF IN CLOCKWISE DIRECTION
0066 FC         133 CONCLY: MOV   A,R4
0067 966D       134        JNZ    CONDEC        ; JUMP TO DECREMENT LOWER BYTE ONLY
0069 FE         135        MOV    A,R6          ; GET UPPER BYTE
006A C675       136        JZ     COURET        ; JUMP TO PREVENT ROLLOVER OF COUNTER
006C CE         137        DEC    R6            ; DECREMENT UPPER BYTE
006D CC         138 CONDEC: DEC   R4
006E 0475       139        JMP    COURET
                140
0070 1C         141 CLOCKW: INC   R4            ; INCREMENT LOWER BYTE
0071 FC         142        MOV    A,R4          ; TEST LOWER BYTE FOR ROLLOVER
0072 9675       143        JNZ    COURET
0074 1E         144        INC    R6            ; INCREMENT UPPER BYTE
                145
0075 0D         146 COURET: MOVD  A,P5          ; LOOP UNTIL PULSE GOES HIGH
0076 5D         147        ANL    A,R5
0077 9648       148        JNZ    PALOOP        ; JUMP WHEN PULSE GOES HI
0079 D675       149        JNIBF  COURET        ; CHECK INTERR ONLY WHILE PULSE LOW
007B AA         150        MOV    R2,A          ; STORE CURRENT PORT READING
                151
007C 22         152 COUNT: IN     A,DBB         ; GET WORD
007D D343       153        XRL    A,#043H       ; CHECK FOR 'C' TO SEND COUNT
007F 9695       154        JNZ    MOSTOP        ; OTHERWISE A 'R' WAS SENT TO STOP MOTORS
0081 FE         155        MOV    A,R6          ; GET UPPER BYTE OF COUNTER
0082 02         156        OUT    DBB,A         ; SEND IT
                157
0083 0D         158 OOLOO: MOVD   A,P5          ; CHECK TO SEE IF A PULSE HAS COME ALONG
0084 4A         159        ORL    A,R2
0085 AA         160        MOV    R2,A
0086 8683       161        JOBF   OOLOO         ; JUMP TILL HOST ACCEPTS IT
0088 FC         162        MOV    A,R4          ; GET LOWER BYTE OF COUNTER
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

0089 02          163        OUT     DBB,A          ; SEND IT
                 164
008A 0D          165 AALAA: MOVD    A,P5
008B 4A          166        ORL     A,R2           ; CHECK AGAIN FOR PULSE
008C AA          167        MOV     R2,A
008D B68A        168        JOBF    AALAA
008F FA          169        MOV     A,R2
0090 5D          170        ANL     A,R5
0091 9648        171        JNZ     PALOOP
0093 0475        172        JMP     COURET
                 173
0095 34C9        174 MOSTOP: CALL   RESET          ; JUMP TO RESET MOTORS
0097 0475        175        JMP     COURET
                 176
                 177 ;*********************************************************************************
                 178
0100             179        ORG     100H
                 180
                 181 ;*********************************************************************************
                 182
                 183 ;       INTERRUPT ROUTINE TO PASS DATA TO THE PROPER CY512
                 184
0100 FA          185 ISERV: MOV     A,R2           ; GET TRACK NUMBER
0101 5418        186        CALL    XLATE
0103 FE          187        MOV     A,R6
0104 C620        188        JZ      START1         ; DEFAULT ON ERROR OR "P" OR "C" OR WHATEVER.
0106 922F        189        JB4     LOAD5
0108 5439        190        CALL    CHECK
010A 34D8        191        CALL    RESET1         ; RESETS THE CY-512 MASKED IN R6
                 192
010C D60C        193 LOAD:  JNIBF   LOAD
010E 22          194        IN      A,DBB
010F AA          195        MOV     R2,A
0110 D32A        196        XRL     A,#2AH         ; CHECK FOR AN "*" TO EXIT ISERV ROUTINE
0112 C629        197        JZ      JUMQ1
0114 FA          198        MOV     A,R2           ; CHECK FOR AN "!" TO JUMP TO ALLGO ROUTINE
0115 D321        199        XRL     A,#021H
0117 C62B        200        JZ      ALLQ1
0119 FA          201        MOV     A,R2           ; TO RESET THE COUNTER LOOK FOR AN "A"
011A D341        202        XRL     A,#041H        ; OR "AT HOME" COMMAND
011C 9625        203        JNZ     LOCON          ; JUMP AROUND IF NOT "A"
011E B920        204        MOV     R1,#020H       ; LOCATION OF 16 BIT COUNTER
0120 B100        205        MOV     @R1,#0
0122 19          206        INC     R1             ; UPPER BYTE
0123 B100        207        MOV     @R1,#0
0125 5400        208 LOCON: CALL    OUTPT
0127 240C        209        JMP     LOAD
                 210
0129 2488        211 JUMQ1: JMP     Q1
012B 2469        212 ALLQ1: JMP     ALLGO
012D 0430        213 START1: JMP    START
                 214
                 215 ;*********************************************************************************
                 216
                 217 ;       ROUTINE TO PASS DATA TO CY512-5     NO COUNT IS NEEDED BY THE HOST
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

                218
012F 9A7F       219 LOAD5:  ANL     P2,#07FH        ;SEL 8243-B
0131 230E       220         MOV     A,#0EH
0133 9F         221         ANLD    P7,A            ;RESET CY512-5
0134 BF16       222         MOV     R7,#22D
0136 EF36       223 CY5L00: DJNZ    R7,CY5L00
0138 2301       224         MOV     A,#1
013A 8F         225         ORLD    P7,A            ;REMOVE RESET
013B 8A80       226         ORL     P2,#080H        ;DESEL 8243-B
                227
013D D63D       228 LP2:    JNIBF   LP2
013F 22         229         IN      A,DBB
0140 AA         230         MOV     R2,A            ; TEMP STORE
0141 D32A       231         XRL     A,#2AH          ; SEARCH FOR AN ASTERISK ('*')
0143 C62D       232         JZ      START1
0145 FA         233         MOV     A,R2
0146 D321       234         XRL     A,#21H
0148 C62B       235         JZ      ALLQ1
014A 9A7F       236         ANL     P2,#07FH        ; SEL 8243-B
014C 0E         237 CHECK1: MOVD    A,P6
014D 5304       238         ANL     A,#4            ; CHECK TO ENSURE CONTROLLER IS NOT RUNNING
014F C64C       239         JZ      CHECK1          ; JUMP IF RUNNING
0151 0E         240 LP1:    MOVD    A,P6
0152 5301       241         ANL     A,#1            ; CY512 READY?
0154 C651       242         JZ      LP1             ; JUMP IF BUSY
0156 FA         243         MOV     A,R2
0157 4380       244         ORL     A,#080H         ; P17 LATCHED HIGH FOR DISPLAY ROUTINE
0159 39         245         OUTL    P1,A            ; MOVE DATA TO PORT 1
015A 230D       246         MOV     A,#0DH          ; 1101
015C 9F         247         ANLD    P7,A            ; SEL CY512-5
015D 0E         248 LP3:    MOVD    A,P6
015E 5301       249         ANL     A,#1            ; CY512 BUSY?
0160 965D       250         JNZ     LP3             ; LOOP UNTIL CONTROLLER BUSY
0162 2302       251         MOV     A,#2
0164 8F         252         ORLD    P7,A            ; DESEL CY512-5
0165 8A80       253         ORL     P2,#080H        ; DESEL 8243-B
0167 243D       254         JMP     LP2
                255
                256 ;*******************************************************************************
                257
                258 ;       THIS ROUTINE SENDS A 'D' & 'CARRIAGE RETURN' TO ALL CONTROLLERS
                259
0169 BA44       260 ALLGO:  MOV     R2,#044H        ; SEND D TO ALL CONTROLLERS
016B 3477       261         CALL    GOGETT
016D BA0D       262         MOV     R2,#0DH         ; SEND CARR RET TO ALL CONTROLLERS
016F 3477       263         CALL    GOGETT
0171 B930       264         MOV     R1,#030H        ; SET FLAG TO JUMP TO TEST ROUTINE
0173 B1FF       265         MOV     @R1,#0FFH
0175 240C       266         JMP     LOAD            ; JUMP TO ROUTINE WHICH LOOKS FOR COUNT
                267
0177 BE01       268 GOGETT: MOV     R6,#1           ; SEND D & CR TO CONTROLLER #1
0179 348A       269         CALL    GOUT
017B BE02       270         MOV     R6,#2           ; SAME FOR #2
017D 348A       271         CALL    GOUT
017F BE04       272         MOV     R6,#4           ; SAME FOR #3
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

0181 348A        273        CALL    GOUT
0183 BE08        274        MOV     R6,#8          ; SAME FOR #4
0185 348A        275        CALL    GOUT
0187 34A2        276        CALL    GOUT5          ; SAME FOR #5
0189 83          277        RET
                 278
018A 5439        279 GOUT:  CALL    CHECK          ; ENSURE MOTOR IS NOT MOVING
018C 9ABF        280        ANL     P2,#0BFH       ; SELECT 8243-A
018E 0F          281 GTLOOP: MOVD   A,P7           ; CHECK READY/BUSY LINES
018F 5E          282        ANL     A,R6
0190 C68E        283        JZ      GTLOOP         ; JUMP IF CONTROLLER BUSY
                 284
0192 FA          285        MOV     A,R2
0193 4380        286        ORL     A,#080H        ; TO KEEP P17 HIGH
0195 39          287        OUTL    P1,A           ; SEND TO CONTROLLERS
                 288
0196 FE          289        MOV     A,R6           ; SELECT THE CONTROLLER
0197 37          290        CPL     A
0198 9C          291        ANLD    P4,A
                 292
0199 0F          293 GELOOP: MOVD   A,P7           ; CHECK READY/BUSY LINES
019A 5E          294        ANL     A,R6           ; LOOP UNTIL EACH CONTROLLER IS BUSY
019B 9499        295        JNZ     GELOOP
                 296
019D FE          297        MOV     A,R6
019E 8C          298        ORLD    P4,A           ; DESELECT CONTROLLER
019F 8A40        299        ORL     P2,#040H       ; DESELECT 8243-A
01A1 83          300        RET
                 301
01A2 9A7F        302 GOUT5: ANL     P2,#07FH       ; SEL 8243-B
01A4 0E          303 LP1A:  MOVD    A,P6
01A5 5301        304        ANL     A,#1           ; CY512 READY?
01A7 C6A4        305        JZ      LP1A           ; LOOP UNTIL CONTROLLER READY
01A9 FA          306        MOV     A,R2
01AA 4380        307        ORL     A,#080H        ; P17 TO REMAIN HIGH TILL DISPLAY ROUTINE
01AC 39          308        OUTL    P1,A           ; MOVE DATA TO PORT 1
01AD 230D        309        MOV     A,#0DH
01AF 9F          310        ANLD    P7,A           ; SEL CY512-5
01B0 0E          311 LP3A:  MOVD    A,P6
01B1 5301        312        ANL     A,#1           ; CY512 BUSY?
01B3 96B0        313        JNZ     LP3A           ; LOOP UNTIL CONTROLLER READY
01B5 2302        314        MOV     A,#2
01B7 8F          315        ORLD    P7,A           ;DESEL CY512-5
01B8 8A80        316        ORL     P2,#080H       ;DESEL 8243-B
                 317
01BA 83          318        RET
                 319
                 320 ;*****************************************************************************
                 321
01BB 8930        322 Q1:    MOV     R1,#030H       ; JUMP IF FLAGS HAVE BEEN SET BY A '1'
01BD F1          323        MOV     A,@R1
01BE 96C2        324        JNZ     TAAT
01C0 0430        325        JMP     START          ; MASTER RETURN
                 326
01C2 0445        327 TAAT:  JMP     TEST
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

                328
                329 ;*******************************************************************************
                330
                331 ;       SOFTWARE RESET FOR ALL CONTROLLERS
                332
01C4 9A7F       333 RESET:  ANL     P2,#07FH        ; SELECT 8243-B
01C6 27         334         CLR     A
01C7 3C         335         MOVD    P4,A            ; RESET ALL CY512S (4 ONLY)
01C8 230E       336         MOV     A,#0EH          ; 0000/1110 RESET FOR CY512-5
01CA 9F         337         ANLD    P7,A
01CB BF1e       338         MOV     R7,#22D         ; THIS IS FOR 110 MICROSECOND DELAY
01CD EFCD       339 LOOP:   DJNZ    R7,LOOP
01CF 230F       340         MOV     A,#0FH
01D1 3C         341         MOVD    P4,A            ; REMOVE SW RESET
01D2 2301       342         MOV     A,#1            ; REMOVE SW RESET FOR CY512-5
01D4 8F         343         ORLD    P7,A
01D5 8A80       344         ORL     P2,#080H        ; DESELECT 8243-B
01D7 93         345         RETR
                346
                347 ;*******************************************************************************
                348
                349 ;       SOFTWARE RESET FOR CONTROLLER MASKED IN BY R6
                350
01D8 9A7F       351 RESET1: ANL     P2,#07FH        ; SELECT 8243-B
01DA FE         352         MOV     A,R6
01DB 37         353         CPL     A
01DC 9C         354         ANLD    P4,A            ; RESET PROPER CY512
01DD BF16       355         MOV     R7,#22D         ; THIS IS FOR 110 MICROSECOND DELAY
01DF EFDF       356 LOOP1:  DJNZ    R7,LOOP1
01E1 FE         357         MOV     A,R6
01E2 8C         358         ORLD    P4,A            ; REMOVE SW RESET FROM SELECTED CY ONLY
01E3 8A80       359         ORL     P2,#080H        ; DESELECT 8243-B
01E5 83         360         RET
                361
                362 ;***************************
                363
0200            364         ORG     200H
                365
                366 ;*******************************************************************************
                367
                368 ;       OUTPUT ROUTINE WHICH PASSES DATA TO THE CONTROLLER MASKED IN BY R6
                369
0200 5439       370 OUTPT:  CALL    CHECK               ; MAKE SURE CY512 IS FINISHED MOVING
0202 9ABF       371         ANL     P2,#0BFH            ; SELECT 8243-A
0204 0F         372 LOOP3:  MOVD    A,P7                ; CHECK TO SEE IF CY512 IS READY
0205 5E         373         ANL     A,R6
0206 C604       374         JZ      LOOP3               ; IF NOT, CHECK AGAIN
0208 FA         375         MOV     A,R2                ; MOVE DATA TO ACC
0209 4380       376         ORL     A,#080H         ; P17 TO REMAIN HIGH TILL DISPLAY ROUTINE
020B 39         377         OUTL    P1,A                ; OUTPUT TO DATA LINES
020C FE         378         MOV     A,R6
020D 37         379         CPL     A                   ; COMPLEMENT MASK
020E 9C         380         ANLD    P4,A                ; SELECT PROPER CY512
020F 0F         381 LOOP4:  MOVD    A,P7                ; LOOP UNTIL CHOSEN CY512 IS BUSY
0210 5E         382         ANL     A,R6
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

0211 960F        383        JNZ     LOOP4                   ; JUMP IF NOT YET BUSY
0213 FE          384        MOV     A,R6
0214 8C          385        ORLD    P4,A                    ; DESELECT PROPER CY512
0215 8A40        386        ORL     P2,#040H                ; DESELECT 8243-A
0217 83          387        RET
                 388
                 389 ;*************************************************************************
                 390
                 391 ;      THIS TRANSLATE ROUTINE TAKES A MOTOR NUMBER FROM THE HOST AND
                 392 ;      PRODUCES THE PROPER MASK IN R6.
                 393
0218 07          394 XLATE: DEC     A
0219 C636        395        JZ      R6IS1           ; A "1" SPECIFYING MOTOR 1 WAS SENT
021B 07          396        DEC     A
021C C633        397        JZ      R6IS2           ; MOTOR "2"
021E 07          398        DEC     A
021F C630        399        JZ      R6IS3           ; MOTOR "3"
0221 07          400        DEC     A
0222 C62D        401        JZ      R6IS4           ; MOTOR "4"
0224 07          402        DEC     A
0225 C62A        403        JZ      R6IS5           ; MOTOR "5"
0227 BE00        404        MOV     R6,#0           ; DEFAULT ON ERROR
0229 83          405        RET
                 406
022A BE10        407 R6IS5: MOV     R6,#010H        ; 0001/0000
022C 83          408        RET
022D BE08        409 R6IS4: MOV     R6,#8           ; 0000/1000
022F 83          410        RET
0230 BE04        411 R6IS3: MOV     R6,#4           ; 0000/0100
0232 83          412        RET
0233 BE02        413 R6IS2: MOV     R6,#2           ; 0000/0010
0235 83          414        RET
0236 BE01        415 R6IS1: MOV     R6,#1           ; 0000/0001
0238 83          416        RET
                 417
                 418 ;*************************************************************************
                 419
                 420 ;      CHECK TO INSURE THE MASKED IN CONTROLLER IS NOT EXECUTING A PROGRAM
                 421
0239 9A7F        422 CHECK: ANL     P2,#07FH                ;SELECT 8243-B
023B 0D          423 LOOPQ: MOVD    A,P5                    ;MOVE P5 TO A
023C 5E          424        ANL     A,R6                    ;HAS CY512 FINISHED ITS RUN?
023D C63B        425        JZ      LOOPQ                   ;IF NOT, THEN CHECK AGAIN
023F 8A80        426        ORL     P2,#080H                ;DESELECT 8243-B
0241 83          427        RET
                 428
                 429
                 430        END
```

USER SYMBOLS

AALAA	008A	ALLGO	0169	ALLQ1	012B	CHECK	0239	CHECK1	014C	CLEAR	0027	CLOCKW	0070
CONCLK	0066	CONDEC	006D	COUNT	007C	COURET	0075	CY5LOO	0136	EELEE	0055	GELOOP	0199
GOGETT	0177	GOUT	018A	GOUT5	01A2	GTLOOP	018E	ISERV	0100	JUMQ1	0129	LOAD	010C
LOAD5	012F	LOCON	0125	LOOP	01CD	LOOP1	01DF	LOOP3	0204	LOOP4	02[illegible]	LOOPQ	023B
LP1	0151	LP1A	01A9	LP2	013D	LP3	015D	LP3A	01B0	MOSTOP	0095	MOTOSP	005D
OOLOO	0083	OUTPT	0200	PAGOON	0062	PALOOP	0048	PRIOR	003A				

```
01      0188    QQLQQ  0059    R6IS1  0236    R6IS2  0233    R6IS3  0230    R6IS4  022D    R6IS5  022A    RESET  01C9
RESET1  01D8    START  0030    START1 012D    TAAT   01C2    TEST   0045    XLATE  0218

ASSEMBLY COMPLETE, NO ERRORS
```

APPENDIX F

COMPUTER GENERATED VOICE PROGRAM

```
LOC  OBJ        SEQ        SOURCE STATEMENT

                  1
                  2
                  3 ;UPI41 PORT 1 (P1) WILL OUTPUT DATA TO SPEECH CHIP (SPC)
                  4 ;UPI41 PORT 2 (P2) WILL FURNISH CONTROL SIGNALS
                  5
0000              6         ORG     0                     ;RESERVE INTERRUPT LOCATIONS
0000 0409         7         JMP     9
                  8
0003              9         ORG     3
0003 042A        10         JMP     ISERV                 ;TO PROCESS PHRASE
                 11
                 12
                 13 ;       PORT 2 BITS:---
                 14 ;       BIT 0 ==> SPC WR/
                 15 ;       BIT 1 ==> SPC CS/
                 16 ;       BIT 2 ==> PAGE1.  HIGH ==> ROM BANK '1'
                 17 ;       BIT 3 ==> SET.  THE ROM SET TO BE USED.
                 18 ;       BIT 4 ==> CMS
                 19 ;       BIT 5 ==> GAIN (LSB)
                 20 ;       BIT 6 ==> GAIN (MSB)
                 21 ;       BIT 7 ==> GAIN CONTROL CLOCK
                 22
                 23 ;       TO (PIN 1 ON THE 8741) = SPC 'INTR.' HIGH ==> DONE
                 24
                 25
                 26 ;*************************************************************************
                 27 ;                          MAIN TEST PROGRAM LOOP
                 28 ;*************************************************************************
                 29
                 30
0009             31         ORG     9
                 32
0009 23FF        33         MOV     A,#0FFH
0008 39          34         OUTL    P1,A
000C 3A          35         OUTL    P2,A
                 36
                 37
                 38 ;***** SET GAIN TO 1 FOR TEST
0000 9A7F        39         ANL     P2,#01111111B         ;LOWER CLOCK LINE
000F 9A9F        40         ANL     P2,#10011111B         ;SET GAIN TO 1
0011 8A80        41         ORL     P2,#10000000B         ;RAISE CLOCK LINE
0013 9A7F        42         ANL     P2,#01111111B         ;LOWER CLOCK LINE
                 43
                 44
                 45 ;** WILL FORCE SPC TO STOP PROCESSING ANY STARTING GARBAGE AND RESET
                 46 ;** SPC INTR ( = 0). THIS MAKES SPC APPEAR BUSY ****
                 47
0015 2311        48         MOV     A,#11H                ;SPC (WR/=1;CS/=0;CMS=1)
0017 3A          49         OUTL    P2,A
0018 9AFE        50         ANL     P2,#11111110B         ;SPC WR/=0
001A 8A01        51         ORL     P2,#00000001B         ;SPC WR/=1
001C 8A02        52         ORL     P2,#00000010B         ;SPC CS/=1
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

                 53
                 54 ;********* WILL NOW FORCE A VALID COMMAND TO THE SPC TO SET SPC INTR **********
                 55
001E 2347        56         MOV     A,#47H                  ;320MS SILENCE
0020 39          57         OUTL    P1,A                    ;P1 DATA OUT TO SPC
                 58
                 59                                         ; P24  P23  P22  P21  P20
                 60                                 ;SPC      CMS  SET   P1  CS/   WR/
                 61
                 62 ;       INITIAL P2                      ;  1    0    0    1    1
                 63
0021 9AE5        64         ANL     P2,#11100101B           ;  0    0    0    0    1
0023 9AFE        65         ANL     P2,#11111110B           ;  0    0    0    0    0
0025 8A03        66         ORL     P2,#00000011B           ;  0    0    0    1    1
                 67
                 68
                 69 ;*************************************************************************
                 70 ;                                TEST LOOP
                 71 ;*************************************************************************
                 72
0027 05          73         EN      I
0028 0428        74 BACK:   JMP     BACK                    ;WAIT FOR DATA READY INTERRUPT
                 75
002A 00          76 ISERV:  NOP
002B D62B        77 LOOPI:  JNIBF   LOOPI
002D 22          78         IN      A,DBB                   ;INPUT PAGE TO A
002E AE          79         MOV     R6,A                    ;MOVE A TO R6
002F D62F        80 LOOPP:  JNIBF   LOOPP
0031 22          81         IN      A,DBB                   ;INPUT WORD TO A
0032 AF          82         MOV     R7,A                    ;MOVE A TO R7
0033 D633        83 LOOPG:  JNIBF   LOOPG
0035 22          84         IN      A,DBB                   ;INPUT GAIN TO A
0036 5303        85         ANL     A,#00000011B            ;MASK OFF UPPER PORTION
0038 AD          86         MOV     R5,A                    ;MOVE A TO R5
0039 144A        87         CALL    SETT                    ;CALL SETT
003B 85          88         CLR     F0                      ;CLEAR F0 FLAG
003C FE          89         MOV     A,R6                    ;MOVE R6 TO A
003D C640        90         JZ      CONT                    ;IF PAGE=0 JUMP TO CONT
003F 95          91         CPL     F0                      ;IF PAGE=1 COMPLEMENT F0  F0 == 1
0040 FF          92 CONT:   MOV     A,R7                    ;MOVE R7 TO A
0041 B647        93         JF0     CONT1                   ;IF F0=1 JUMP TO CONT1
0043 1459        94         CALL    WRITE0                  ;ELSE CALL WRITE0
0045 0449        95         JMP     LEAVE
0047 145E        96 CONT1:  CALL    WRITE1
0049 93          97 LEAVE:  RETR
                 98
                 99
004A FE         100 SETT:   MOV     A,R6                    ;MOVE PAGE TO A
004B 324F       101         JB1     SETT1                   ;IF SETT = 1, GOTO SETT1
004D 0454       102         JMP     NOSETT                  ;ELSE JUMP TO NOSETT
004F 8A08       103 SETT1:  ORL     P2,#00001000B           ;SETT = 1
0051 27         104         CLR     A                       ;CLEAR A
0052 0458       105         JMP     ENDSTT                  ;JUMP TO ENDSTT
0054 9AF7       106 NOSETT: ANL     P2,#11110111B           ;SETT = 0
0056 53F7       107         ANL     A,#11110111B            ;MASK OFF SETT
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

0058 83          108 ENDSTT: RET
                 109
                 110
0059 9AFB        111 WRITE0: ANL     P2,#0FBH            ;P22 = 0 ==> PAGE = 0
005B 1463        112         CALL    OUTPUT
005D 83          113         RET
                 114
                 115
005E 8A04        116 WRITE1: ORL     P2,#04H             ;P22 = 1 ==> PAGE = 1
0060 1463        117         CALL    OUTPUT
0062 83          118         RET
                 119
                 120
0063 2663        121 OUTPUT: JNTO    OUTPUT              ;MUST INSURE SPC IS READY
0065 39          122         OUTL    P1,A                ;P1 DATA OUT TO SPC
                 123
0066 FD          124         MOV     A,R5                ;MOVE GAIN TO A
0067 9A7F        125         ANL     P2,#01111111B       ;LOWER CLOCK LINE
0069 9A9F        126         ANL     P2,#10011111B       ;MASK OFF OLD GAIN
006B 126F        127         JB0     SET5                ;IF BIT 0 IS 1 THEN JUMP TO SET5
006D 0471        128         JMP     TST1                ;ELSE JUMP TO TST1
006F 8A20        129 SET5:   ORL     P2,#00100000B       ;SET BIT 5 OF PORT 2
0071 3275        130 TST1:   JB1     SET6                ;IF BIT 1 IS 1 THEN JUMP TO SET6
0073 0477        131         JMP     NSET6               ;ELSE JUMP TO NSET6
0075 8A40        132 SET6:   ORL     P2,#01000000B       ;SET BIT 6 OF PORT 2
0077 00          133 NSET6:  NOP                         ;CONTINUE
0078 8A80        134         ORL     P2,#10000000B       ;RAISE CLOCK LINE
007A 9A7F        135         ANL     P2,#01111111B       ;LOWER CLOCK LINE
                 136
                 137                                     ; P24  P23  P22  P21  P20
                 138                             ;SPC      CMS  SET   P1  CS/   WR/
                 139
                 140 ;       INITIAL P2                  ;  0    ?   0/1   1     1
                 141
007C 9AED        142         ANL     P2,#11101101B       ;  0    ?   0/1   0     1
007E 9AFE        143         ANL     P2,#11111110B       ;  0    ?   0/1   0     0
0080 8A03        144         ORL     P2,#00000011B       ;  0    ?   0/1   1     1
                 145
0082 2682        146 LOOP1:  JNTO    LOOP1               ;WAIT UNTIL SPC IS NOT BUSY
0084 9A7F        147         ANL     P2,#01111111B       ;LOWER GAIN CLOCK
0086 9A9F        148         ANL     P2,#10011111B       ;SET GAIN TO 0
0088 8A80        149         ORL     P2,#10000000B       ;RAISE CLOCK LINE
008A 9A7F        150         ANL     P2,#01111111B       ;LOWER CLOCK LINE
                 151
008C 83          152         RET
                 153
                 154         END

USER SYMBOLS
BACK   0028   CONT   0040   CONT1  0047   ENDSTT 0058   ISERV  002A   LEAVE  0049   LOOP1  0082   LOOPI  002B
LOOPP  002F   LOOPQ  0033   NOSETT 0054   NSET6  0077   OUTPUT 0063   SET5   006F   SET6   0075   SETT   004A
SETT1  004F   TST1   0071   WRITE0 0059   WRITE1 005E

ASSEMBLY COMPLETE,  NO ERRORS
```

```
ISIS-II PL/M-86 V2.1 COMPILATION OF MODULE TOW_START_UP_MODULE
OBJECT MODULE PLACED IN :F1:TOWST.OBJ
COMPILER INVOKED BY:  PLM86 :F1:TOWST.042 DEBUG ROM XREF LARGE OPTIMIZE(3) &
                      TITLE('0945 17 DECEMBER 1982')

   1          TOW_START_UP_MODULE: DO;

   2   1      DECLARE SCENE_COUNT LITERALLY '09D',MAX_MENU_NO LITERALLY '01D';

   3   1      DECLARE CARR_RET LITERALLY '0DH', SPACE LITERALLY '20H',BELL LITERALLY '07H';

   4   1      DECLARE IODATA LITERALLY '0D8H',IOSTATUS LITERALLY '0DAH',MASK LITERALLY '7FH';

   5   1      DECLARE VECTOR_MODE LITERALLY '35Q',ALPHA_4010_MODE LITERALLY '15Q',
                ADM3A_MODE LITERALLY '30Q',GRAPHICS_CLEAR LITERALLY '31Q',
                ADM3A_CLEAR LITERALLY '32Q',CLEAR_ALL LITERALLY '33Q',
                HOME_CURSOR LITERALLY '14Q';

   6   1      DECLARE COUNTER_2 LITERALLY '0D4H',CONTROL LITERALLY '0D6H',
                CNTR2MODE LITERALLY '096H',SETCOUNT LITERALLY '04H';
                /* FOR TIMER SETUP */

   7   1      DECLARE USART_CONTROL LITERALLY '0DAH',USART_MODE LITERALLY '4EH',
                USART_COMMAND LITERALLY '37H';

   8   1      DECLARE LINE_FEED LITERALLY '0AH';

   9   1      DECLARE (DAY_SIGHT,STARTING_TRACK,TARGET_SWITCH,FINAL_TRACK,
                EAST_WEST,CONTINUE) BYTE EXTERNAL;

  10   1      DECLARE GUNNER_RATING BYTE AT (6080H),TURNED BYTE AT (6081H);

  11   1      DECLARE (RESPONSE,GO_NOW,MENU_NO,MENU_DONE,SCENARIO,I,PREVIOUS_RESPONSE) BYTE;

  12   1      DECLARE (OK1,OK2,TOTALY_OK) BYTE;

  13   1      DECLARE SCENARIO_BUFFER (9) BYTE EXTERNAL;

  14   1      DECLARE (RESP_1_ASCII,RESP_2_ASCII,RESP_1_NUM,RESP_2_NUM) BYTE;

  15   1      DECLARE (SIGHT_FLAG,TRACK_FLAG,DONE,OK,SAME) BYTE;

                /*****************************************************************************
                * THE SCENARIOS ARE LISTED BELOW.                                            *
                *****************************************************************************/


  16   1      DECLARE SCENE_0 (*) BYTE DATA (CARR_RET);

  17   1      DECLARE SCENE_1 (*) BYTE DATA ('A',CARR_RET,'H',CARR_RET,'B',CARR_RET,
                'S 1',CARR_RET,'R 225',CARR_RET,'F 1',CARR_RET,'E',CARR_RET,
                'X 1500',CARR_RET,'P 2950',CARR_RET,'X 1000',CARR_RET,'C',CARR_RET,
                'B',CARR_RET,'I',CARR_RET,'Q','!','*');
```

```
18   1      DECLARE SCENE_2 (*) BYTE DATA ('P',01H,'A',CARR_RET,'H',CARR_RET,
              'B',CARR_RET,'S 1',CARR_RET,'R 173',CARR_RET,'F 1',CARR_RET,'E',CARR_RET,
              'X 764',CARR_RET,'P 1600',CARR_RET,'C',CARR_RET,'B',CARR_RET,'P 5240',
              CARR_RET,'O',CARR_RET,'Q','*','P',02H,'A',
              CARR_RET,'H',CARR_RET,'B',CARR_RET,'S 1',CARR_RET,'R 173',CARR_RET,'F 1',
              CARR_RET,'E',CARR_RET,'X 382',CARR_RET,'P 1650',CARR_RET,'O',CARR_RET,'Q',
              '*','P',03H,'A',CARR_RET,'H',CARR_RET,'B',CARR_RET,'S 1',CARR_RET,'R 173',
              CARR_RET,'F 1',CARR_RET,'E',CARR_RET,'P 1650',CARR_RET,'O',CARR_RET,'Q','*',
              'P',04H,'A',CARR_RET,'H',CARR_RET,'B',CARR_RET,'S 1',CARR_RET,'R 1',CARR_RET,
              'F 1',CARR_RET,'E',CARR_RET,'X 4364',CARR_RET,'C',CARR_RET,'B',CARR_RET,
              'X 1000',CARR_RET,'C',CARR_RET,'B',CARR_RET,'O',CARR_RET,'Q','*',
              'P',05H,'A',CARR_RET,'H',CARR_RET,'B',CARR_RET,'S 1',CARR_RET,'R 80',
              CARR_RET,'F 1',CARR_RET,'E',CARR_RET,'X 764',CARR_RET,'P 650',CARR_RET,'O',
              CARR_RET,'Q','!','*');

19   1      DECLARE SCENE_3 (*) BYTE DATA ('P',01H,'A',CARR_RET,'H',CARR_RET,'B',
              CARR_RET,'S 1',CARR_RET,'R 173',CARR_RET,'F 1',CARR_RET,'E',CARR_RET,
              'X 764',CARR_RET,'P 2150',CARR_RET,'C',CARR_RET,'B',CARR_RET,'P 5240',
              CARR_RET,'O',CARR_RET,'Q','*','P',02H,'A',
              CARR_RET,'H',CARR_RET,'B',CARR_RET,'S 1',CARR_RET,'R 173',CARR_RET,'F 1',
              CARR_RET,'E',CARR_RET,'X 382',CARR_RET,'P 1650',CARR_RET,'X 1091',CARR_RET,
              'P 5240',CARR_RET,'O',CARR_RET,'Q','*','P',03H,'A',CARR_RET,'H',CARR_RET,
              'B',CARR_RET,'S 1',CARR_RET,'R 173',CARR_RET,'F 1',CARR_RET,'E',CARR_RET,
              'P 500',CARR_RET,'X 400',CARR_RET,'P 0',CARR_RET,'O',CARR_RET,'Q','*','P',
              04H,'A',CARR_RET,'H',CARR_RET,'B',CARR_RET,'S 1',CARR_RET,'R 1',CARR_RET,
              'F 1',CARR_RET,'E',CARR_RET,'X 4364',CARR_RET,'C',CARR_RET,'B',CARR_RET,
              'X 1500',CARR_RET,'C',CARR_RET,'B',CARR_RET,'O',CARR_RET,'Q','*','P',05H,
              'A',CARR_RET,'H',CARR_RET,'B',CARR_RET,'S 1',CARR_RET,'R 80',CARR_RET,'F 1',
              CARR_RET,'E',CARR_RET,'X 384',CARR_RET,'P 650',CARR_RET,'X 1091',CARR_RET,
              'P 250',CARR_RET,'O',CARR_RET,'Q','!','*');

20   1      DECLARE SCENE_4 (*) BYTE DATA ('P',01H,'A',CARR_RET,'H',CARR_RET,'B',
              CARR_RET,'S 1',CARR_RET,'R 173',CARR_RET,'F 1',CARR_RET,'E',CARR_RET,
              'X 764',CARR_RET,'P 2300',CARR_RET,'C',CARR_RET,'B',CARR_RET,'P 5240',
              CARR_RET,'O',CARR_RET,'Q','*','P',02H,'A',
              CARR_RET,'H',CARR_RET,'B',CARR_RET,'S 1',CARR_RET,'R 173',CARR_RET,'F 1',
              CARR_RET,'E',CARR_RET,'X 382',CARR_RET,'P 1650',CARR_RET,'O',CARR_RET,
              'Q','*','P',03H,'A',CARR_RET,'H',CARR_RET,'B',CARR_RET,'S 1',CARR_RET,
              'R 173',CARR_RET,'F 1',CARR_RET,'E',CARR_RET,'P 1650',CARR_RET,'O',CARR_RET,
              'Q','*','P',04H,'A',CARR_RET,'H',CARR_RET,'B',CARR_RET,'S 1',CARR_RET,'R 1',
              CARR_RET,'F 1',CARR_RET,'E',CARR_RET,'X 5200',CARR_RET,'C',CARR_RET,'B',
              CARR_RET,'O',CARR_RET,'Q','*','P',05H,'A',CARR_RET,'H',CARR_RET,'B',CARR_RET,
              'S 1',CARR_RET,'R 225',CARR_RET,'F 1',CARR_RET,'E',CARR_RET,'P 150',CARR_RET,
              'X 3250',CARR_RET,'R 125',CARR_RET,'F 1',CARR_RET,'P 650',CARR_RET,'O',
              CARR_RET,'Q','!','*');

21   1      DECLARE SCENE_5 (*) BYTE DATA ('P',01H,'A',CARR_RET,'H',CARR_RET,'B',CARR_RET,
              'S 1',CARR_RET,'R 156',CARR_RET,'F 1',CARR_RET,'E',CARR_RET,'P 1700',
              CARR_RET,'C',CARR_RET,'B',CARR_RET,'P 5240',
              CARR_RET,'O',CARR_RET,'Q','*','P',02H,'A',CARR_RET,'H',CARR_RET,'B',CARR_RET,
              'S 1',CARR_RET,'R 156',CARR_RET,'F 1',CARR_RET,'E',CARR_RET,'P 500',CARR_RET,
              'X 400',CARR_RET,'P 0',CARR_RET,'O',CARR_RET,'Q','*','P',03H,'A',CARR_RET,
              'H',CARR_RET,'B',CARR_RET,'S 1',CARR_RET,'R 156',CARR_RET,'F 1',CARR_RET,'E',
              CARR_RET,'X 350',CARR_RET,'P 575',CARR_RET,'X 425',CARR_RET,'P 0',CARR_RET,
              'O',CARR_RET,'Q','*','P',04H,'A',CARR_RET,'H',CARR_RET,'B',CARR_RET,'S 1',
              CARR_RET,'P 1',CARR_RET,'F 1',CARR_RET,'E',CARR_RET,'X 4364',CARR_RET,'C',
```

```
          CARR_RET,'B',CARR_RET,'X 1091',CARR_RET,'C',CARR_RET,'B',CARR_RET,'0',
          CARR_RET,'Q','*','P',05H,'A',CARR_RET,'H',CARR_RET,'B',CARR_RET,'S 1',
          CARR_RET,'R 100',CARR_RET,'F 1',CARR_RET,'E',CARR_RET,'X 367',CARR_RET,
          'P 650',CARR_RET,'X 545',CARR_RET,'P 500',CARR_RET,'X 545',CARR_RET,'L 2,3',
          CARR_RET,'P 650',CARR_RET,'0',CARR_RET,'Q','!','*');

22  1     DECLARE SCENE_6 (*) BYTE DATA ('P',01H,'A',CARR_RET,'H',CARR_RET,'B',CARR_RET,
          'S 1',CARR_RET,'R 173',CARR_RET,'F 1',CARR_RET,'E',CARR_RET,'X 1582',
          CARR_RET,'P 1650',CARR_RET,'C',CARR_RET,'B',CARR_RET,
          'P 5240',CARR_RET,'0',CARR_RET,'Q','*','P',02H,'A',CARR_RET,'H',CARR_RET,
          'B',CARR_RET,'S 1',CARR_RET,'R 173',CARR_RET,'F 1',CARR_RET,'E',CARR_RET,
          'P 2200',CARR_RET,'X 382',CARR_RET,'P 1650',CARR_RET,'0',CARR_RET,'Q','*',
          'P',03H,'A',CARR_RET,'H',CARR_RET,'B',CARR_RET,'S 1',CARR_RET,'R 173',
          CARR_RET,'F 1',CARR_RET,'E',CARR_RET,'X 382',CARR_RET,'P 2100',CARR_RET,
          'X 425',CARR_RET,'P 1650',CARR_RET,'0',CARR_RET,'Q','*','P',04H,'A',CARR_RET,
          'H',CARR_RET,'B',CARR_RET,'S 1',CARR_RET,'R 1',CARR_RET,'F 1',CARR_RET,'E',
          CARR_RET,'X 4364',CARR_RET,'C',CARR_RET,'B',CARR_RET,'X 820',CARR_RET,'C',
          CARR_RET,'B',CARR_RET,'0',CARR_RET,'Q','*','P',05H,'A',CARR_RET,'H',CARR_RET,
          'B',CARR_RET,'S 1',CARR_RET,'R 225',CARR_RET,'F 1',CARR_RET,'E',CARR_RET,
          'P 650',CARR_RET,'R 235',CARR_RET,'F 1',CARR_RET,'P 625',CARR_RET,'X 273',
          CARR_RET,'P 650',CARR_RET,'X 273',CARR_RET,'L 15,7',CARR_RET,'P 650',
          CARR_RET,'0',CARR_RET,'Q','!','*');

23  1     DECLARE SCENE_7 (*)BYTE DATA ('A',CARR_RET,'H',CARR_RET,'B',CARR_RET,'S 1',
          CARR_RET,'R 156',CARR_RET,'F 1',CARR_RET,'E',CARR_RET,'X 382',CARR_RET,
          'P 2475',CARR_RET,'C',CARR_RET,'B',CARR_RET,'P 5240',CARR_RET,'0',CARR_RET,
          'Q','*','P',04H,'A',CARR_RET,'H',CARR_RET,
          'B',CARR_RET,'S 1',CARR_RET,'R 1',CARR_RET,'F 1',CARR_RET,'E',CARR_RET,
          'X 3500',CARR_RET,'C',CARR_RET,'B',CARR_RET,'X 1227',CARR_RET,'C',CARR_RET,
          'B',CARR_RET,'0',CARR_RET,'Q','!','*');

24  1     DECLARE SCENE_8 (*) BYTE DATA ('P',01H,'A',CARR_RET,'H',CARR_RET,'B',CARR_RET,
          'S 1',CARR_RET,'R 225',CARR_RET,'F 1',CARR_RET,'E',CARR_RET,'P 5240',
          CARR_RET,'R 180',CARR_RET,'F 1',CARR_RET,'U',CARR_RET,'P 4800',CARR_RET,'C',
          CARR_RET,'B',CARR_RET,'P 0',CARR_RET,'0',CARR_RET,'Q','*','P',05,'A',
          CARR_RET,'H',CARR_RET,'B',CARR_RET,'S 1',CARR_RET,'R 200',CARR_RET,'F 1',
          CARR_RET,'E',CARR_RET,'X 10',CARR_RET,'W',CARR_RET,'U',CARR_RET,'P 650',
          CARR_RET,'X 545',CARR_RET,'P 500',CARR_RET,'X 545',CARR_RET,'L 2,5',CARR_RET,
          'P 650',CARR_RET,'0',CARR_RET,'Q','!','*');

25  1     DECLARE SCENE_9 (*) BYTE DATA ('P',03H,'A',CARR_RET,'H',CARR_RET,'B',CARR_RET,
          'S 1',CARR_RET,'R 225',CARR_RET,'F 1',CARR_RET,'E',CARR_RET,'X 1000',
          CARR_RET,'P 2950',CARR_RET,'R 50',CARR_RET,'F 1',CARR_RET,'C',CARR_RET,'B',
          CARR_RET,'U',CARR_RET,'P 2800',CARR_RET,'U',CARR_RET,'L 10,3',CARR_RET,
          'P 2950',CARR_RET,'0',CARR_RET,'Q','*','P',04H,'A',CARR_RET,'H',
          CARR_RET,'B',CARR_RET,'S 1',CARR_RET,'R 1',CARR_RET,'F 255',CARR_RET,'E',
          CARR_RET,'X 100',CARR_RET,'W',CARR_RET,'U',CARR_RET,'N 1',CARR_RET,'G',
          CARR_RET,'X 5',CARR_RET,'L 26,5',CARR_RET,'X 5',CARR_RET,'U',CARR_RET,'P 24',
          CARR_RET,'X 5',CARR_RET,'P 22',CARR_RET,'X 5',CARR_RET,'U',CARR_RET,'P 24',
          CARR_RET,'X 5',CARR_RET,'P 26',CARR_RET,'X 5',CARR_RET,'L 10,15',CARR_RET,
          'P 24',CARR_RET,'0',CARR_RET,'Q','!','*');

26  1     DECLARE GO_HOME (*) BYTE DATA ('H',CARR_RET,'N 1',CARR_RET,'-',CARR_RET,
          'R 245',CARR_RET,'E',CARR_RET,'G',CARR_RET,'T',CARR_RET,'I',CARR_RET,
          'DD',CARR_RET,'*');

27  1     DECLARE RAISE_MOTOR (*) BYTE DATA ('P',05H,'A',CARR_RET,'H',CARR_RET,
```

```
          'S 1',CARR_RET,'R 225',CARR_RET,'F 1',CARR_RET,'E',CARR_RET,
          'P 650',CARR_RET,'I',CARR_RET,'O',CARR_RET,'Q','*');

28   1    DECLARE LOWER_MOTOR (*) BYTE DATA ('H',CARR_RET,'S 1',CARR_RET,
          'R 225',CARR_RET,'F 1',CARR_RET,'E',CARR_RET,'-',CARR_RET,'N 1200',CARR_RET,
          'G',CARR_RET,'I',CARR_RET,'O',CARR_RET,'QD',CARR_RET,'*');

          /***************************************************************************
          * COMMENTS PRINTED ON THE SCREEN BY THE PRINT PROCEDURE ARE LISTED BELOW.   *
          ***************************************************************************/


29   1    DECLARE HELLO (*) BYTE DATA (CARR_RET,LINE_FEED,LINE_FEED,LINE_FEED,LINE_FEED,
          'STAGS-T TRAINER VERSION 1.0',CARR_RET,LINE_FEED,'*');

30   1    DECLARE FOO_BAH (*) BYTE DATA (' I TOLD YOU THIS !$"&%$#@ TRAINER CAN''T ',
                                          'DO THAT!!!',CARR_RET,LINE_FEED,'*',);

31   1    DECLARE HELLO_CON (*) BYTE DATA (CARR_RET,LINE_FEED,'NEED A MENU? (Y OR N)',
          CARR_RET,LINE_FEED,'*');

32   1    DECLARE ITEM_1 (*) BYTE DATA ('ITEM 1: TANK MOVES TO THE CENTER OF THE',
          ' TRACK AND STOPS',CARR_RET,LINE_FEED,'*');

33   1    DECLARE ITEM_2 (*) BYTE DATA ('ITEM 2: FRONT TANK RISES FROM TRENCH, MOVING',
          ' WEST, AND BECOMES THE TARGET',CARR_RET,LINE_FEED,'        CENTER AND ',
          'REAR TANKS MOVE FROM EAST TO WEST INTO COVER',CARR_RET,LINE_FEED,'*');

34   1    DECLARE ITEM_3 (*) BYTE DATA ('ITEM 3: FRONT TANK RISES FROM TRENCH MOVING ',
          'WEST, THEN SINKS AGAIN',CARR_RET,LINE_FEED,'        CENTER TANK MOVES ',
          'WEST INTO COVER; THEN REAPPEARS, MOVING EAST',CARR_RET,LINE_FEED,'        ',
          'AND BECOMES THE TARGET.  REAR TANK MOVES OUT, THEN RETREATS TO ',
          CARR_RET,LINE_FEED,'        ITS REAR INTO COVER',CARR_RET,LINE_FEED,'*');

35   1    DECLARE ITEM_4 (*) BYTE DATA ('ITEM 4: CENTER AND REAR TANKS MOVE EAST TO ',
          'WEST INTO COVER',CARR_RET,LINE_FEED,'        FRONT TANK RISES FROM TRENCH',
          ' AND BECOMES TARGET AS CENTER AND',CARR_RET,LINE_FEED,'        REAR TANKS',
          ' DISAPPEAR',CARR_RET,LINE_FEED,'*');

36   1    DECLARE ITEM_5 (*) BYTE DATA ('ITEM 5: FRONT TANK RISES FROM TRENCH AND ',
          'TRAVERSES HILLY TERRAIN',CARR_RET,LINE_FEED,'        CENTER AND REAR ',
          'TANKS APPEAR, THEN RETREAT INTO COVER',CARR_RET,LINE_FEED,'*');

37   1    DECLARE ITEM_6 (*) BYTE DATA ('ITEM 6: FRONT TANK MOVES WEST OVER ROUGH ',
          'TERRAIN',CARR_RET,LINE_FEED,'        CENTER AND REAR TANKS MOVE WEST ',
          'THROUGH COVER AND THEN RETREAT',CARR_RET,LINE_FEED,'        BACK INTO IT',
          CARR_RET,LINE_FEED,'*');

38   1    DECLARE ITEM_7 (*) BYTE DATA ('ITEM 7: TANK MOVES EAST TO WEST ON ANY TRACK',
          CARR_RET,LINE_FEED,'*');

39   1    DECLARE ITEM_8 (*) BYTE DATA ('ITEM 8: FRONT TANK MOVES WEST TO EAST AS ',
          'TARGET',CARR_RET,LINE_FEED,'*');

40   1    DECLARE ITEM_9 (*) BYTE DATA ('ITEM9: REAR TANK TRAVERSES OUT MOVES TOWARD ',
```

```
            'GUNNER',CARR_RET,LINE_FEED,'$');

41  1       DECLARE PAGE_COMMENT (*) BYTE DATA (CARR_RET,LINE_FEED,LINE_FEED,LINE_FEED,
              LINE_FEED,LINE_FEED,'PAGE $');

42  1       DECLARE SINGLE_PAGE (*) BYTE DATA (LINE_FEED,'PRESS "E" TO EXIT MENU.',
              CARR_RET,LINE_FEED,LINE_FEED,'$');

43  1       DECLARE FIRST_PAGE (*) BYTE DATA (LINE_FEED,'PRESS "N" TO SEE THE NEXT PAGE,',
              ' "E" TO EXIT MENU',LINE_FEED,CARR_RET,LINE_FEED,'$');

44  1       DECLARE CENTER_PAGE (*) BYTE DATA (LINE_FEED,'PRESS "N" TO SEE THE NEXT PAGE,',
              ' "P" TO SEE THE PREVIOUS PAGE, "E" TO EXIT MENU',LINE_FEED,CARR_RET,
              LINE_FEED,'$');

45  1       DECLARE LAST_PAGE (*) BYTE DATA (LINE_FEED,'PRESS "P" TO SEE THE PREVIOUS ',
              'PAGE, "E" TO EXIT MENU',LINE_FEED,CARR_RET,LINE_FEED,'$');

46  1       DECLARE FOO (*) BYTE DATA (' THIS STAGS-T TRAINER CAN''T DO THAT!!',
              CARR_RET,LINE_FEED,'$');

47  1       DECLARE REQUEST (*) BYTE DATA (CARR_RET,LINE_FEED,LINE_FEED,LINE_FEED
              ,LINE_FEED,'WHICH ITEM ?? $');

48  1       DECLARE SIGHT_Q (*) BYTE DATA ('DO YOU WISH TO USE THE DAYSIGHT (D) OR THE ',
              'NIGHTSIGHT (N)? ','$');

49  1       DECLARE TRACK_Q (*) BYTE DATA ('WHICH TRACK DO YOU WISH TO RUN THIS SCENARIO',
              ' ON',CARR_RET,LINE_FEED,'    (1 = FRONT, 2 = CENTER, AND 3 = REAR)? ','$');

50  1       DECLARE RATING_Q (*) BYTE DATA ('ENTER GUNNER RATING (0, 1, OR 2).  $');

51  1       DECLARE ITEM_PTRS_1 (*) POINTER DATA(@ITEM_1,@ITEM_2,@ITEM_3,@ITEM_4,@ITEM_5,
              @ITEM_6,@ITEM_7,@ITEM_8,@ITEM_9);

            /*****************************************************************************
            * THE FOLLOWING SUBROUTINE INPUTS A BYTE OF ASCII DATA FROM THE TERMINAL     *
            *****************************************************************************/

52  1       CIN:PROCEDURE BYTE;
53  2         DO WHILE NOT SHR(INPUT(IOSTATUS),1);
54  3         END;
55  2       RETURN MASK AND INPUT(IODATA);
56  2         END CIN;

            /*****************************************************************************
            * THE FOLLOWING SUBROUTINE OUTPUTS A BYTE OF ASCII DATA TO THE TERMINAL      *
            *****************************************************************************/

57  1       COUT: PROCEDURE (ITEM);
58  2         DECLARE ITEM BYTE;
59  2           DO WHILE NOT(INPUT(IOSTATUS));
60  3             END;
61  2         OUTPUT(IODATA)=ITEM;
```

```
62  2       CALL TIME(15);
63  2         END COUT;

            /*****************************************************************************/
            /* THE FOLLOWING ROUTINE OUTPUTS A CHARACTER TO THE 8741 WHEN CALLED FROM   */
            /* THE MAIN PROGRAM                                                          */
            /*****************************************************************************/

64  1       OUTPT: PROCEDURE (OUTDATA);
65  2         DECLARE OUTDATA BYTE, STAT_COM BYTE AT (0F000H), P_DATA BYTE AT (0F002H);
66  2         DO WHILE NOT SHR(STAT_COM,1);     /* WAIT UNTIL UPI41 IBF = 0 */
67  3           END;
68  2         P_DATA = NOT OUTDATA;          /* NOT  BECAUSE MULTIBUS INVERTS */
69  2         END OUTPT;



70  1       PRINT: PROCEDURE(PNTR);  /* PROMPTS THE CONSOLE */
71  2        DECLARE I WORD;
72  2        DECLARE PNTR POINTER,
                 CHAR BASED PNTR (1) BYTE;/* CHAR MUST BE AN ARRAY TO KEEP PLM HAPPY */
73  2        I = 0;
74  2        LOOP: DO WHILE CHAR(I) <> '*';
75  3         CALL COUT(CHAR(I));
76  3         I = I + 1;
77  3        END LOOP;
78  2       END PRINT;

            /*****************************************************************************
            * THIS PROCEDURE SENDS AN ENTIRE PROGRAM (OR 'SCENE'), POINTED TO BY        *
            * PROG_PTR, TO THE CY512                                                    *
            *****************************************************************************/

79  1       TANK_PROG: PROCEDURE(PROG_PTR,LENGT);
80  2         DECLARE PROG_PTR POINTER, LENGT WORD, (ITEM BASED PROG_PTR) (1) BYTE,
              C WORD;
81  2         C = 0;
82  2         DO WHILE C < LENGT;
83  3           CALL OUTPT(ITEM(C));
84  3           C = C + 1;
85  3         END;
86  2       END TANK_PROG;


            /*****************************************************************************
            * THIS PROCEDURE SENDS ALL THE MOTORS TO THEIR HOME POSITIONS.              *
            *****************************************************************************/

87  1       RESET_MOTORS: PROCEDURE;

88  2         I = 0;
89  2         DO WHILE I < 4;
90  3           I = I + 1;
91  3           CALL OUTPT('P');
92  3           CALL OUTPT(I);
93  3           CALL TANK_PROG(@GO_HOME,SIZE(GO_HOME));
94  3           END;
```

```
 95  2       CALL OUTPT('P');
 96  2       CALL OUTPT(05H);
 97  2       CALL TANK_PROG(@LOWER_MOTOR,SIZE(LOWER_MOTOR));

 98  2       END RESET_MOTORS;


            /* OUTPUTS PAGE 1 OF THE MENU */

 99  1      MENU_1: PROCEDURE;
100  2        RESPONSE = 0;
101  2        CALL PRINT(@PAGE_COMMENT);
102  2        CALL COUT('1');
103  2        CALL COUT(CARR_RET);
104  2        CALL COUT(LINE_FEED);
105  2        CALL COUT(LINE_FEED);

106  2        I = 0;
107  2        DO WHILE I < LENGTH(ITEM_PTRS_1);
108  3          CALL PRINT(ITEM_PTRS_1(I));
109  3          I = I + 1;
110  3          END;

111  2        CALL PRINT(@SINGLE_PAGE);
112  2        END MENU_1;


            /* OUTPUTS PAGE 2 OF THE MENU */

113  1      MENU_2: PROCEDURE;
114  2        RESPONSE = 0;
115  2        END MENU_2;


            /* OUTPUTS PAGE 3 OF THE MENU */

116  1      MENU_3: PROCEDURE;
117  2        RESPONSE = 0;
118  2        END MENU_3;


            /* OUTPUTS PAGE 4 OF THE MENU */

119  1      MENU_4: PROCEDURE;
120  2        RESPONSE = 0;
121  2        END MENU_4;


            /* OUTPUTS PAGE 5 OF THE MENU */

122  1      MENU_5: PROCEDURE;
123  2        RESPONSE = 0;
124  2        END MENU_5;


125  1      GIVE_MENU: PROCEDURE;
```

```
126  2     CALL PRINT(@HELLO_CON);

127  2     RESPONSE = 0;
128  2     RESPONSE = CIN;
129  2      CALL COUT(RESPONSE);  /* ECHO PRINT */

130  2      IF (RESPONSE = 59H) OR (RESPONSE = 79H) THEN   /* UPPER OR LOWER CASE 'Y' */
131  2      MENU: DO;
132  3       CALL COUT(CARR_RET);
133  3       CALL COUT(LINE_FEED);
134  3       MENU_DONE = 0;
135  3       MENU_NO = 1;
136  3       CALL COUT(VECTOR_MODE);
137  3       CALL COUT(CLEAR_ALL);
138  3       CALL COUT(HOME_CURSOR);
139  3       CALL TIME(2000);
140  3       CALL MENU_1;
141  3       DO WHILE NOT MENU_DONE;
142  4         SAME = 0;
143  4         OK = 0;
144  4         RESPONSE = CIN;

145  4         IF (RESPONSE = 4EH) OR (RESPONSE = 6EH) THEN DO;    /*  UC OR LC 'N' */
147  5           IF MENU_NO < MAX_MENU_NO THEN DO;
149  6             MENU_NO = MENU_NO + 1;
150  6             OK = 1;
151  6             END;
152  5           ELSE DO;
153  6         SAME = 1;
154  6         CALL COUT(BELL);
155  6         END;
156  5           END;

157  4         IF (RESPONSE = 50H) OR (RESPONSE = 70H) THEN DO;    /* UC OR LC 'P' */
159  5           IF MENU_NO > 1 THEN DO;
161  6             MENU_NO = MENU_NO - 1;
162  6             OK = 1;
163  6             END;
164  5           ELSE DO;
165  6         SAME = 1;
166  6         CALL COUT(BELL);
167  6         END;
168  5           END;

169  4         IF (RESPONSE = 45H) OR (RESPONSE = 65H) THEN DO;
171  5           MENU_DONE = 1; /* UC/LC 'E' */
172  5           OK = 1;
173  5           END;

174  4         IF NOT MENU_DONE THEN IF NOT SAME THEN IF OK THEN DO;
178  5           CALL COUT(CLEAR_ALL);
179  5           CALL COUT(HOME_CURSOR);
180  5           CALL TIME(2000);
181  5           DO CASE (MENU_NO - 1);
182  6             CALL MENU_1;
183  6             CALL MENU_2;
```

```
184  6            CALL MENU_3;
185  6            CALL MENU_4;
186  6            CALL MENU_5;
187  6            END;
188  5          END;

189  4        END;

190  3      END MENU;
191  2      ELSE DO;
192  3       CALL COUT(CARR_RET);
193  3       CALL COUT(LINE_FEED);
194  3       END;

195  2      CALL COUT(CLEAR_ALL);
196  2      CALL COUT(HOME_CURSOR);
197  2      CALL TIME(2000);

198  2     END GIVE_MENU;


199  1     TANK_INIT: PROCEDURE PUBLIC;

           /*****************************************************************************
                              TANK_INIT PROCEDURE BEGINS HERE
           *****************************************************************************/


           /*****************************************************************************
           * RESET ALL MOTORS TO STARTING POSITIONS.                                    *
           *****************************************************************************/

200  2     CALL RESET_MOTORS;


201  2     CALL COUT(VECTOR_MODE);
202  2     CALL COUT(CLEAR_ALL);
203  2     CALL COUT(HOME_CURSOR);
204  2     CALL TIME(2000);

205  2     COMMENT1:
            CALL PRINT(@HELLO);

206  2     CALL GIVE_MENU;


207  2     OK1,OK2,TOTALY_OK,GO_NOW,RESPONSE,PREVIOUS_RESPONSE = 0;
208  2     GET_ITEM: DO WHILE NOT TOTALY_OK;
              /* WAIT TILL A PROPER MENU ITEM IS ENTERED */
209  3      CALL PRINT (@REQUEST);

210  3     OK_1: DO WHILE NOT OK1;
211  4     RESP_1_ASCII = CIN;
212  4     RESP_1_NUM = RESP_1_ASCII-30H;
213  4     IF (RESP_1_ASCII < 3AH) AND (RESP_1_ASCII > 30H) THEN /* IS BETWEEN 1 & 9 */
214  4     DO;
215  5         CALL COUT (RESP_1_ASCII);
```

```
216  5          OK1 = 1;
217  5       END;
218  4       ELSE CALL COUT(BELL);
219  4       END OK_1;

220  3       OK_2: DO WHILE NOT OK2;
221  4       RESP_2_ASCII = CIN;
222  4       RESP_2_NUM = RESP_2_ASCII - 30H;
223  4       IF ((RESP_2_ASCII < 3AH) AND (RESP_2_ASCII > 29H)) OR (RESP_2_ASCII = CARR_RET) THEN
224  4       DO;
225  5       IF RESP_2_ASCII <> CARR_RET THEN CALL COUT (RESP_2_ASCII);
227  5       OK2 = 1;
228  5       END;
229  4       ELSE CALL COUT(BELL);
230  4       END OK_2;

231  3       IF RESP_2_ASCII <> CARR_RET THEN
232  3       DO;
233  4       RESP_1_NUM = RESP_1_NUM * 10D;
234  4       RESPONSE = RESP_1_NUM + RESP_2_NUM;
235  4       SCENARIO_BUFFER (6) = RESP_1_ASCII;
236  4       SCENARIO_BUFFER (7) = RESP_2_ASCII;
237  4       GO_NOW = 0;
238  4       END;

239  3       ELSE DO;
240  4       RESPONSE = RESP_1_NUM;
241  4       SCENARIO_BUFFER (6) = SPACE;
242  4       SCENARIO_BUFFER (7) = RESP_1_ASCII;
243  4       GO_NOW = CARR_RET;
244  4       END;

245  3       IF RESPONSE <= SCENE_COUNT THEN TOTALY_OK = 1;

247  3       ELSE DO;
248  4       OK1,OK2 = 0;
249  4       IF RESPONSE <> PREVIOUS_RESPONSE THEN CALL PRINT(@F00);
251  4       ELSE CALL PRINT(@F00_BAH);
252  4       CALL COUT(CARR_RET);
253  4       CALL COUT(LINE_FEED);
254  4       CALL COUT(BELL);
255  4       PREVIOUS_RESPONSE = RESPONSE;
256  4       CALL GIVE_MENU;
257  4       END;
258  3       END;

259  2        SCENARIO = RESPONSE;

260  2        WAIT_GO:
                DO WHILE GO_NOW <> CARR_RET;    /* WAIT FOR CARR_RET */
261  3            GO_NOW = CIN;
262  3            IF GO_NOW <> CARR_RET THEN CALL COUT(BELL);
264  3            END WAIT_GO;

265  2       CALL COUT(CARR_RET);
266  2       CALL COUT(LINE_FEED);
```

```
            /*****************************************************************************
            * SET FLAGS:                                                                 *
            *   SIGHT_FLAG = 1 ==> INSTRUCTOR HAS A CHOICE OF DAY OR NIGHT SIGHT         *
            *   TRACK_FLAG = 1 ==> INSTRUCTOR HAS A CHOICE OF WHICH TRACK TO USE         *
            *   EAST_WEST = 1 ==> TARGET WILL START FROM THE EAST                        *
            *   STARTING_TRACK ==> THE TRACK THE SCENARIO WILL START ON                  *
            *   TARGET_SWITCH = 0 ==> THERE WILL BE NO TARGET SWITCH                     *
            *   FINAL_TRACK ==> TRACK TO SWITCH TO IF THERE IS A SWITCH                  *
            *   CONTINUE ==> SYNCHRONIZES DIGITALKER                                     *
            *       TURNED ==> INDICATES WHETHER TRACK 3 IS ROTATED 1=ROTATED            *
            *****************************************************************************/

267   2     SIGHT_FLAG,EAST_WEST = 1;
268   2     TRACK_FLAG,TARGET_SWITCH,FINAL_TRACK,TURNED = 0;

269   2     FLAG_SET:DO CASE (SCENARIO);

270   3       FLAG_SET_0:DO;    /* DO CASE EXPECTS 0, BUT THERE IS NEVER SCENARIO 0 */
271   4         END;

272   3       FLAG_SET_1:DO;
273   4         TRACK_FLAG = 1;
274   4         END;

275   3       FLAG_SET_2:DO;
276   4         STARTING_TRACK = 1;
277   4         END;

278   3       FLAG_SET_3:DO;
279   4         STARTING_TRACK = 2;
280   4         END;

281   3       FLAG_SET_4:DO;
282   4         STARTING_TRACK = 1;
283   4         END;

284   3       FLAG_SET_5:DO;
285   4         STARTING_TRACK = 1;
286   4         END;

287   3       FLAG_SET_6:DO;
288   4         STARTING_TRACK = 1;
289   4         END;

290   3       FLAG_SET_7:DO;
291   4         TRACK_FLAG = 1;
292   4         END;

293   3       FLAG_SET_8:DO;
294   4         EAST_WEST = 0;
295   4         STARTING_TRACK = 1;
296   4         END;

297   3       FLAG_SET_9:DO;
298   4         STARTING_TRACK = 3;
299   4         TURNED = 1;
```

```
300  4          END;

301  3        END FLAG_SET;

302  2      DONE = 0;

303  2      IF SIGHT_FLAG THEN DO WHILE NOT DONE; /* WE HAVE A CHOICE OF DAY/NIGHT SIGHT */

305  3        RESPONSE = 0;
306  3        CALL PRINT(@SIGHT_Q);
307  3        RESPONSE = CIN;
308  3        CALL COUT(RESPONSE);
309  3        CALL COUT(CARR_RET);
310  3        CALL COUT(LINE_FEED);
311  3        CALL COUT(LINE_FEED);
312  3        IF (RESPONSE = 44H) OR (RESPONSE = 64H) THEN DO;  /* UC OR LC 'D' */
314  4          DONE = 1;
315  4          DAY_SIGHT = 1;  /* USE DAY SIGHT */
316  4          END;
317  3        IF (RESPONSE = 4EH) OR (RESPONSE = 6EH) THEN DO;  /* UC OR LC 'N' */
319  4          DONE = 1;
320  4          DAY_SIGHT = 0;  /* USE NIGHT SIGHT */
321  4          END;

322  3        IF NOT DONE THEN CALL COUT(BELL);

324  3        END;

325  2      DONE = 0;

326  2      IF TRACK_FLAG THEN DO WHILE NOT DONE;   /* WE HAVE A CHOICE OF TRACK */

328  3        RESPONSE = 0;
329  3        CALL PRINT(@TRACK_Q);
330  3        RESPONSE = CIN;
331  3        CALL COUT(RESPONSE);
332  3        CALL COUT(CARR_RET);
333  3        CALL COUT(LINE_FEED);
334  3        CALL COUT(LINE_FEED);
335  3        IF (RESPONSE > 30H) AND (RESPONSE < 34H) THEN DO;
337  4          STARTING_TRACK = RESPONSE - 30H;        /* SELECT STARTING TRACK */
338  4          DONE = 1;
339  4          END;
340  3        ELSE CALL COUT(BELL);

341  3        CALL TANK_PROG(@RAISE_MOTOR,SIZE(RAISE_MOTOR));

342  3        END;

343  2        RESPONSE,DONE = 0;
344  2        DO WHILE NOT DONE;
345  3          CALL PRINT(@RATING_Q);
346  3          RESPONSE = CIN;
347  3          CALL COUT(RESPONSE);
348  3          CALL COUT(CARR_RET);
```

```
349  3          CALL COUT(LINE_FEED);
350  3          RESPONSE = RESPONSE - 30H;
351  3          GUNNER_RATING = RESPONSE;
352  3          IF ((RESPONSE >= 0) AND (RESPONSE <= 2)) THEN DONE = 1;
354  3          ELSE CALL COUT(BELL);
355  3          END;

356  2      CONTINUE = 1;   /* USED TO SYNCHRONIZE OPERATIONS WITH 'DIGITALKER' */

357  2      END TANK_INIT;



358  1      TANK_START: PROCEDURE PUBLIC;   /* CALL AFTER SCREEN PRESENTATION COMPLETE */

359  2        CALL OUTPT('T');
360  2        CALL OUTPT(STARTING_TRACK);

361  2        IF TRACK_FLAG THEN DO;
363  3          CALL OUTPT('P');
364  3          CALL OUTPT(STARTING_TRACK);
365  3          END;

366  2      DO CASE (SCENARIO);
367  3        CALL TANK_PROG(@SCENE_0,SIZE(SCENE_0));
368  3        CALL TANK_PROG(@SCENE_1,SIZE(SCENE_1));
369  3        CALL TANK_PROG(@SCENE_2,SIZE(SCENE_2));
370  3        CALL TANK_PROG(@SCENE_3,SIZE(SCENE_3));
371  3        CALL TANK_PROG(@SCENE_4,SIZE(SCENE_4));
372  3        CALL TANK_PROG(@SCENE_5,SIZE(SCENE_5));
373  3        CALL TANK_PROG(@SCENE_6,SIZE(SCENE_6));
374  3        CALL TANK_PROG(@SCENE_7,SIZE(SCENE_7));
375  3        CALL TANK_PROG(@SCENE_8,SIZE(SCENE_8));
376  3        CALL TANK_PROG(@SCENE_9,SIZE(SCENE_9));
377  3      END;

378  2      CONTINUE = 1;

379  2      END TANK_START;


        /*****************************************************************************
        * WE NOW WISH TO STOP THE TANKS IMMEDIATELY AND WAIT FOR RESET.              *
        *****************************************************************************/

380  1      TANK_KILLED: PROCEDURE PUBLIC;

381  2        CALL OUTPT('R');          /* RESET ALL CY512S */

382  2        END TANK_KILLED;

383  1        END TOW_START_UP_MODULE;
```

CROSS-REFERENCE LISTING

DEFN	ADDR	SIZE	NAME, ATTRIBUTES, AND REFERENCES
5			ADM3A_CLEAR. . . . . LITERALLY
5			ADM3A_MODE . . . . . LITERALLY
5			ALPHA_4010_MODE. . . LITERALLY
3			BELL . . . . . LITERALLY 154 166 218 229 254 263 323 340 354
80	0002H	2	C. . . . . WORD 81 82 83 84
3			CARR_RET . . . . . LITERALLY 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 49 103 132 192 223 225 231 243 252 260 262 265 309 332 348
44	0C0EH	83	CENTER_PAGE. . . . . BYTE ARRAY(83) DATA
72	0000H	1	CHAR . . . . . BYTE BASED(PNTR) ARRAY(1) 74 75
52	0DC2H	21	CIN. . . . . PROCEDURE BYTE STACK=0002H 128 144 211 221 261 307 330 346
5			CLEAR_ALL. . . . . LITERALLY 137 178 195 202
6			CNTR2MODE. . . . . LITERALLY
205	10E0H		COMMENT1 . . . . . LABEL
9	0000H	1	CONTINUE . . . . . BYTE EXTERNAL(5) 356 378
6			CONTROL. . . . . LITERALLY
6			COUNTER_2. . . . . LITERALLY
57	0DD7H	34	COUT . . . . . PROCEDURE STACK=0004H 75 102 103 104 105 129 132 133 136 137 138 154 166 178 179 192 193 195 196 201 202 203 215 218 226 229 252 253 254 263 265 266 308 309 310 311 323 331 332 333 334 340 347 348 349 354
9	0000H	1	DAY_SIGHT. . . . . BYTE EXTERNAL(0) 315 320
15	0014H	1	DONE . . . . . BYTE 302 304 314 319 322 325 327 338 343 344 353
9	0000H	1	EAST_WEST. . . . . BYTE EXTERNAL(4) 267 294
9	0000H	1	FINAL_TRACK. . . . . BYTE EXTERNAL(3) 268
43	0BD9H	53	FIRST_PAGE . . . . . BYTE ARRAY(53) DATA
269	1296H		FLAG_SET . . . . . LABEL
270	12A3H		FLAG_SET_0 . . . . . LABEL
272	12A3H		FLAG_SET_1 . . . . . LABEL
275	12A3H		FLAG_SET_2 . . . . . LABEL
278	12A3H		FLAG_SET_3 . . . . . LABEL
281	12AEH		FLAG_SET_4 . . . . . LABEL
284	12AEH		FLAG_SET_5 . . . . . LABEL
287	12AEH		FLAG_SET_6 . . . . . LABEL
290	12AEH		FLAG_SET_7 . . . . . LABEL
293	12B5H		FLAG_SET_8 . . . . . LABEL
297	12C9H		FLAG_SET_9 . . . . . LABEL
46	0C9AH	40	FOO. . . . . BYTE ARRAY(40) DATA 250
30	075BH	54	FOO_BAH. . . . . BYTE ARRAY(54) DATA 251
208	1100H		GET_ITEM . . . . . LABEL
125	0F35H	382	GIVE_MENU. . . . . PROCEDURE STACK=0014H 206 256
26	06D7H	26	GO_HOME. . . . . BYTE ARRAY(26) DATA 93
11	0005H	1	GO_NOW . . . . . BYTE 207 237 243 260 261 262
5			GRAPHICS_CLEAR . . . LITERALLY
10	6080H	1	GUNNER_RATING. . . . BYTE AT ABSOLUTE 351
29	0738H	35	HELLO. . . . . BYTE ARRAY(35) DATA 205
31	0791H	26	HELLO_CON. . . . . BYTE ARRAY(26) DATA 126

```
 5                HOME_CURSOR. . . . .  LITERALLY       138  179  196  203
11  0009H     1  I. . . . . .  BYTE        88   89   90   92  106  107  108  109
71  0000H     2  I. . . . . .  WORD        73   74   75   76
                 INPUT. . . . . .  BUILTIN          53   55   59
 4               IODATA . . . . . .  LITERALLY       55   61
 4               IOSTATUS . . . . . .  LITERALLY     53   59
80  0000H     1  ITEM . . . . . .  BYTE BASED(PROG_PTR) ARRAY(1)        83
57  0004H     1  ITEM . . . . . .  BYTE PARAMETER AUTOMATIC        58   61
32  07ABH    58  ITEM_1 . . . . . .  BYTE ARRAY(58) DATA        51
33  07E5H   141  ITEM_2 . . . . . .  BYTE ARRAY(141) DATA       51
34  0872H   243  ITEM_3 . . . . . .  BYTE ARRAY(243) DATA       51
35  0965H   163  ITEM_4 . . . . . .  BYTE ARRAY(163) DATA       51
36  0A08H   130  ITEM_5 . . . . . .  BYTE ARRAY(130) DATA       51
37  0A8AH   145  ITEM_6 . . . . . .  BYTE ARRAY(145) DATA       51
38  0B1BH    47  ITEM_7 . . . . . .  BYTE ARRAY(47) DATA        51
39  0B4AH    50  ITEM_8 . . . . . .  BYTE ARRAY(50) DATA        51
40  0B7CH    53  ITEM_9 . . . . . .  BYTE ARRAY(53) DATA        51
51  0000H    36  ITEM_PTRS_1. . . . .  POINTER ARRAY(9) DATA          107  108
45  0C61H    57  LAST_PAGE. . . . . .  BYTE ARRAY(57) DATA
79  0004H     2  LENGT. . . . . .  WORD PARAMETER AUTOMATIC        80   82
                 LENGTH . . . . . .  BUILTIN         107
 8               LINE_FEED. . . . . .  LITERALLY       29   30   31   32   33   34   35   36   37   38   39
                                        40   41   42   43   44   45   46   47   49  104  105  133  193  253
                                       266  310  311  333  334  349
74  0E28H        LOOP . . . . . .  LABEL
28  0713H    37  LOWER_MOTOR. . . . .  BYTE ARRAY(37) DATA        97
 4               MASK . . . . . .  LITERALLY       55
 2               MAX_MENU_NO. . . . .  LITERALLY    147
131  0F64H       MENU . . . . . .  LABEL
99  0EB5H    88  MENU_1 . . . . . .  PROCEDURE STACK=0010H          140  182
113  0F0DH    10  MENU_2 . . . . . .  PROCEDURE STACK=0002H         183
116  0F17H    10  MENU_3 . . . . . .  PROCEDURE STACK=0002H         184
119  0F21H    10  MENU_4 . . . . . .  PROCEDURE STACK=0002H         185
122  0F2BH    10  MENU_5 . . . . . .  PROCEDURE STACK=0002H         186
11  0007H     1  MENU_DONE. . . . . .  BYTE       134  141  171  174
11  0006H     1  MENU_NO. . . . . .  BYTE        135  147  149  159  161  181
15  0015H     1  OK . . . . . .  BYTE        143  150  162  172  176
12  000BH     1  OK1. . . . . .  BYTE        207  210  216  248
12  000CH     1  OK2. . . . . .  BYTE        207  220  227  248
210  1115H       OK_1 . . . . . .  LABEL
220  114AH       OK_2 . . . . . .  LABEL
64  0004H     1  OUTDATA. . . . . .  BYTE PARAMETER AUTOMATIC        65   68
64  0DF9H    38  OUTPT. . . . . .  PROCEDURE STACK=0004H          83   91   92   95   96  359  360  363
                                       364  381
                 OUTPUT . . . . . .  BUILTIN         61
41  0BB1H    12  PAGE_COMMENT . . . .  BYTE ARRAY(12) DATA       101
70  0004H     4  PNTR . . . . . .  POINTER PARAMETER AUTOMATIC        72   74   75
11  000AH     1  PREVIOUS_RESPONSE. .  BYTE       207  249  255
70  0E1FH    38  PRINT. . . . . .  PROCEDURE STACK=000CH          101  108  111  126  205  209  250  251
                                       306  329  345
79  0006H     4  PROG_PTR . . . . . .  POINTER PARAMETER AUTOMATIC        80   83
65  F002H     1  P_DATA . . . . . .  BYTE AT ABSOLUTE          68
27  06F1H    34  RAISE_MOTOR. . . . .  BYTE ARRAY(34) DATA       341
50  0D6FH    36  RATING_Q . . . . . .  BYTE ARRAY(36) DATA       345
47  0CC2H    20  REQUEST. . . . . .  BYTE ARRAY(20) DATA        209
87  0E6BH    74  RESET_MOTORS . . . .  PROCEDURE STACK=0012H         200
11  0004H     1  RESPONSE . . . . . .  BYTE       100  114  117  120  123  127  128  129  130  144  145  157
```

```
                                                 169  207  234  240  245  249  255  259  305  307  308  312  317  328
                                                 330  331  335  337  343  346  347  350  351  352
  14  000EH     1  RESP_1_ASCII . . . .   BYTE        211  212  213  215  235  242
  14  0010H     1  RESP_1_NUM . . . . .   BYTE        212  233  234  240
  14  000FH     1  RESP_2_ASCII . . . .   BYTE        221  222  223  225  226  231  236
  14  0011H     1  RESP_2_NUM . . . . .   BYTE        222  234
  15  0016H     1  SAME . . . . . .       BYTE        142  153  165  175
  11  0008H     1  SCENARIO . . . . . .   BYTE        259  269  366
  13  0000H     9  SCENARIO_BUFFER. . .   BYTE ARRAY(9) EXTERNAL(6)      235  236  241  242
  16  0024H     1  SCENE_0. . . . . .     BYTE ARRAY(1) DATA        367
  17  0025H    52  SCENE_1. . . . . .     BYTE ARRAY(52) DATA       368
  18  0059H   216  SCENE_2. . . . . .     BYTE ARRAY(216) DATA      369
  19  0131H   252  SCENE_3. . . . . .     BYTE ARRAY(252) DATA      370
  20  022DH   223  SCENE_4. . . . . .     BYTE ARRAY(223) DATA      371
  21  030CH   259  SCENE_5. . . . . .     BYTE ARRAY(259) DATA      372
  22  040FH   284  SCENE_6. . . . . .     BYTE ARRAY(284) DATA      373
  23  052BH    99  SCENE_7. . . . . .     BYTE ARRAY(99) DATA       374
  24  058EH   136  SCENE_8. . . . . .     BYTE ARRAY(136) DATA      375
  25  0616H   193  SCENE_9. . . . . .     BYTE ARRAY(193) DATA      376
   2               SCENE_COUNT. . . . .   LITERALLY       245
   6               SETCOUNT . . . . . .   LITERALLY
                   SHR. . . . . . . . .   BUILTIN          53   66
  15  0012H     1  SIGHT_FLAG . . . . .   BYTE       267  303
  48  0CD6H    60  SIGHT_Q. . . . . .     BYTE ARRAY(60) DATA       306
  42  0BBDH    28  SINGLE_PAGE. . . . .   BYTE ARRAY(28) DATA       111
                   SIZE . . . . . . . .   BUILTIN          93   97  341  367  368  369  370  371  372  373  374
                                           375  376
   3               SPACE. . . . . . . .   LITERALLY       241
   9  0000H     1  STARTING_TRACK . . .   BYTE EXTERNAL(1)          276  279  282  285  288  295  298  337  360
                                           364
  65  F000H     1  STAT_COM . . . . . .   BYTE AT ABSOLUTE           66
 199  10B3H   932  TANK_INIT. . . . . .   PROCEDURE PUBLIC STACK=0018H
 380  1528H    18  TANK_KILLED. . . . .   PROCEDURE PUBLIC STACK=0008H
  79  0E45H    38  TANK_PROG. . . . . .   PROCEDURE STACK=000EH          93   97  341  367  368  369  370  371
                                           372  373  374  375  376
 358  1457H   209  TANK_START . . . . .   PROCEDURE PUBLIC STACK=0012H
   9  0000H     1  TARGET_SWITCH. . . .   BYTE EXTERNAL(2)          268
                   TIME . . . . . . . .   BUILTIN          62  139  180  197  204
  12  000DH     1  TOTALY_OK. . . . . .   BYTE       207  208  246
   1  0DC2H        TOW_START_UP_MODULE.   PROCEDURE STACK=0000H
  15  0013H     1  TRACK_FLAG . . . . .   BYTE       268  273  291  326  361
  49  0D12H    93  TRACK_Q. . . . . .     BYTE ARRAY(93) DATA       329
  10  6081H     1  TURNED . . . . . . .   BYTE AT ABSOLUTE          268  299
   7               USART_COMMAND. . . .   LITERALLY
   7               USART_CONTROL. . . .   LITERALLY
   7               USART_MODE . . . . .   LITERALLY
   5               VECTOR_MODE. . . . .   LITERALLY       136  201
 260  1243H        WAIT_GO. . . . . . .   LABEL
```

MODULE INFORMATION:

```
CODE AREA SIZE     = 153AH   5434D
CONSTANT AREA SIZE = 0000H     0D
VARIABLE AREA SIZE = 0017H    23D
MAXIMUM STACK SIZE = 0018H    24D
```

```
    730 LINES READ
    0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION
```

APPENDIX G

TOW STATISTICAL PACKAGE

Four program modules were modified to incorporate the TOW Statistical Package. The modifications were relatively simple because all the necessary information is collected in real-time during simulated missile flight with statistical analysis being a subsequent operation.

Figure G-1 shows the PLM/86 addition to the TOW Flight Module which calculates gunner aiming error statistics in elevation and azimuth. These include mean and unbiased (M-1) standard deviation in both axes for selected time intervals.

Figure G-2 shows the revised TOW Utility Module procedure "HX2AS" which processes both statistics and miss information.

Figure G-3 shows the revised TOW Main Module procedure "Action-Wait" which waits for a command for either statistics or a reprise presentation.

Figure G-4 (A, B and C) shows program additions to the PIP Keyboard-IO Module.

```
228  1    STATISTICS: PROCEDURE PUBLIC;
229  2     DECLARE H_MEAN (6) BYTE AT(0A0A2H), H_DEV (6) BYTE AT(0A0A8H);
230  2     DECLARE MEAN_RIGHT BYTE AT(0A0AEH);
231  2     DECLARE V_MEAN (6) BYTE AT(0A0B2H), V_DEV (6) BYTE AT(0A0B8H);
232  2     DECLARE MEAN_UP BYTE AT(0A0BEH);
233  2     DECLARE DO_STATS BYTE AT(0A090H);
234  2     DECLARE STATS_READY BYTE AT(0A091H);
235  2     DECLARE START_TIME BYTE AT(0A092H);
236  2     DECLARE END_TIME BYTE AT(0A093H);
237  2     DECLARE END_AT_TARGET BYTE AT(0A094H);
238  2     DECLARE (START_COUNT, END_COUNT, I) WORD, (SIGMA_Y, SIGMA_Z,
              SIGMA_Y_SQ, SIGMA_Z_SQ, F_COUNT) REAL;
239  2     DECLARE (VAR_Y, VAR_Z, MEAN_GAEY, MEAN_GAEZ,STD_DEV_Y, STD_DEV_Z) REAL;
240  2    SIGMA_Y, SIGMA_Z, SIGMA_Y_SQ, SIGMA_Z_SQ = 0.0;
241  2    DO_STATS = 0;
242  2    START_COUNT = 25 * START_TIME;
243  2    END_COUNT = 25 * END_TIME;
244  2    IF END_COUNT > UNSIGN(COUNT) THEN BEYOND: DO;
246  3        END_COUNT = UNSIGN(COUNT);
247  3        END_AT_TARGET = 1;
248  3       END BEYOND;
249  2    F_COUNT = FLOAT(INT(END_COUNT - START_COUNT));
250  2    SIGMA:
           DO I = (START_COUNT +1) TO END_COUNT;
251  3        SIGMA_Y = SIGMA_Y + RESULTS(I).S_GAEY;
252  3        SIGMA_Z = SIGMA_Z + RESULTS(I).S_GAEZ;
253  3        SIGMA_Y_SQ = SIGMA_Y_SQ + RESULTS(I).S_GAEY * RESULTS(I).S_GAEY;
254  3        SIGMA_Z_SQ = SIGMA_Z_SQ + RESULTS(I).S_GAEZ * RESULTS(I).S_GAEZ;
255  3       END SIGMA;
256  2    MEAN_GAEY = SIGMA_Y/F_COUNT;
257  2    VAR_Y = (SIGMA_Y_SQ - SIGMA_Y * MEAN_GAEY)/(F_COUNT - 1.);
258  2    IF VAR_Y < 1.0E-12 THEN VAR_Y = 0.0;     /* SMALLEST REPRESENTABLE SDEV. */
260  2    STD_DEV_Y = SQRT(VAR_Y, 0.5);
261  2    MEAN_GAEZ = SIGMA_Z/F_COUNT;
262  2    VAR_Z = (SIGMA_Z_SQ - SIGMA_Z * MEAN_GAEZ)/(F_COUNT - 1.);
263  2    IF VAR_Z   1.0E-12 THEN VAR_Z = 0.0;     /* SMALLEST REPRESENTABLE SDEV. */
265  2    STD_DEV_Z = SQRT(VAR_, 0.5);
266  2    IF MEAN_GAEY < 0. THEN MEAN_RIGHT = 0; ELSE MEAN_RIGHT = 1;
269  2    IF MEAN_GAEZ   0. THEN MEAN__UP = 0; ELSE MEAN_UP = 1;
272  2    CALL HX2AS(FIX(MEAN_GAEY * 1.E6),@H_MEAN);
273  2    CALL HX2AS(FIX(STD_DEV_Y * 1.E6),@H_DEV);
274  2    CALL HX2AS(FIX(MEAN_GAEZ * 1.E6),@V_MEAN);
275  2    CALL HX2AS(FIX(STD_DEV_Z * 1.E6),@V_DEV);
276  2    STATS_READY = 1;
277  2    END STATISTICS;
```

Figure G-1. Addition to TOW Flight Module.

```
                /* HX2AS CONVERTS AN INTEGER TO ASCII CHARACTERS WITH THE
                   LEAST SIGNIFICANT DIGIT IN THE TENTHS POSITION */
  4   1     HX2AS: PROCEDURE (IHEX,ASCII_ADR) PUBLIC;
  5   2       DECLARE ASCII_ADR POINTER, IHEX INTEGER, HEX WORD,
                      ASCII BASED ASCII_ADR (6) BYTE, N INTEGER , REMAINDER WORD;
  6   2       IF IHEX < 0 THEN HEX = UNSIGN(-IHEX);
  8   2                   ELSE HEX = UNSIGN(IHEX);
  9   2       DO N = 4 TO 0 BY - 1;
 10   3         REMAINDER = HEX MOD 10 + 30H;
 11   3         ASCII(N) = LOW(REMAINDER);
 12   3         HEX = HEX/10;
 13   3        END;
 14   2     IF NO_TENTHS = 0  /* NO_TENTHS = 1 DURING REPRISE */
 15   2     THEN DO;
 16   3          ASCII(5) = ASCII(4);
 17   3          ASCII(4) = '.';
 18   3          END;
 19   2       N=0;
 20   2       DO WHILE (ASCII(N) = 30H) AND (N < SIGNED(3 + NO_TENTHS));  /* REPLACE LEADING ZEROES WITH BLANKS */
 21   3         ASCII(N) = 20H;
 22   3         N = N + 1;
 23   3        END;
 24   2      END HX2AS;
```

Figure G-2. Revised "HX2AS" Procedure.

```
187   1     ACTION_WAIT:              /* WAIT FOR  REPRISE */
              DO FOREVER;
188   2         ACTION = NOT(INPUT(PORT_B)) AND 04H;
189   2         IF ACTION = 4 THEN CALL H_REPRISE;
191   2         IF DO_STATS THEN CALL STATISTICS;
193   2        END ACTION_WAIT;
```

Figure G-3. Revised "Action-Wait" Procedure.

```
43  1     DECLARE STATS_MSG_1 (*) BYTE DATA ('ENTER START TIME: ');
44  1     DECLARE STATS_MSG_2 (*) BYTE DATA ('ENTER END TIME  : ');
45  1     DECLARE ALPHA_MODE_HOME (*) BYTE DATA (33Q,14Q);
46  1     DECLARE STATS_MSG_0 (*) BYTE DATA (33Q,14Q,35Q,62Q,170Q,52Q,112Q,37Q,'STAGS/T',
                                ' STATISTICAL PACKAGE',35Q,62Q,142Q,46Q,114Q,
                                37Q,'(MEAN AND STANDARD DEVIATION IN MICRORADIANS)',
                                CR,LF,LF,LF,
                                   LF);
47  1     DECLARE STATS_MSG_3 (*) BYTE DATA (35Q,52Q,160Q,45Q,131Q,37Q,'  ELEVATION',
                                           35Q,52Q,160Q,66Q,100Q,37Q,'  AZIMUTH',CR,LF,
                                           LF,LF,
            '          MEAN     :                          MEAN     :',CR,LF,LF,LF,
            '          STANDARD                            STANDARD  ',CR,LF,
            '          DEVIATION:                          DEVIATION:',CR,LF,LF,LF,
            '          DIRECTION:                          DIRECTION:',35Q);

48  1     DECLARE STATS_MEAN_V    (*) BYTE DATA (35Q,50Q,156Q,51Q,106Q,37Q);
49  1     DECLARE STATS_STD_DEV_V (*) BYTE DATA (35Q,45Q,166Q,51Q,106Q,37Q);
50  1     DECLARE STATS_DIR_V     (*) BYTE DATA (35Q,43Q,164Q,52Q,102Q,37Q);
51  1     DECLARE STATS_MEAN_H    (*) BYTE DATA (35Q,50Q,156Q,70Q,136Q,37Q);
52  1     DECLARE STATS_STD_DEV_H (*) BYTE DATA (35Q,45Q,166Q,70Q,136Q,37Q);
53  1     DECLARE STATS_DIR_H     (*) BYTE DATA (35Q,43Q,164Q,71Q,132Q,37Q);



54  1     DECLARE RIGHT       (*) BYTE DATA ('RIGHT');
55  1     DECLARE LEFT        (*) BYTE DATA ('LEFT' );
56  1     DECLARE UP          (*) BYTE DATA ('UP'   );
57  1     DECLARE DOWN        (*) BYTE DATA ('DOWN' );
```

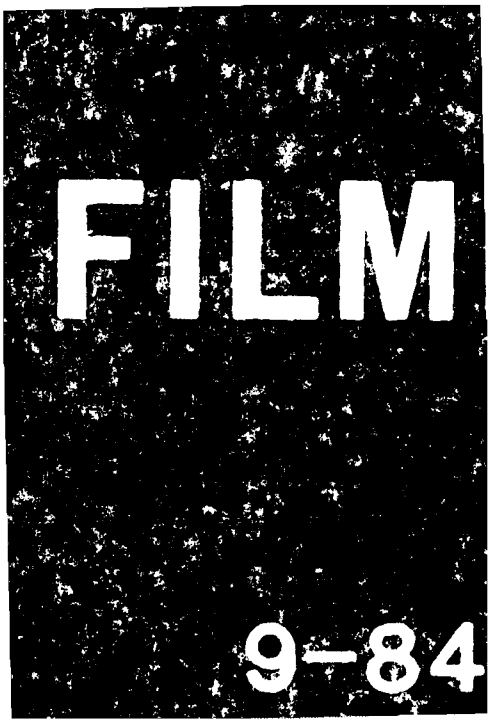
Figure G-4(A). Addition to "Keyboard-IO" Module.

```
                /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
                *    THE FOLLOWING PROCEDURE IS FOR GENERATING THE STATISTICAL DATA AFTER A  *
                *    FLIGHT.  THE USER MAY INPUT THE BEGINING TIME AND AN ENDING TIME IF LESS*
                *    THAN THE TOTAL FLIGHT TIME IS DESIRED.                                  *
                * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

236   1      STATS_ROUTINE: PROCEDURE PUBLIC;

237   2      RE_ENTRY: CALL PRINT (@ALPHA_MODE_HOME, LENGTH(ALPHA_MODE_HOME));
238   2      CALL    TIME(1700D);  /* DELAY UNTIL FINISHED (>165 MILLISECONDS)*/
239   2      FAST,I,DO_STATS,STATS_RDY,END_AT_TARGET,STATS_REQ = 0;
240   2      CALL PRINT (@STATS_MSG_0, LENGTH (STATS_MSG_0));

                /* WE ARE NOW READY TO QUERRY THE USER:
                   (1)  WE DISPLAY 'STATISTICAL ANALYSIS PROGRAM'
                   (2)  PRINT 'ENTER START TIME: ' (INGEST UP TO TWO DIGITS)
                   (3)  PRINT 'ENTER END TIME  : ' (INGEST UP TO TWO DIGITS)
                   (4)  UPON ENTERING THE SECOND NUMBER OR A CR THEN CLEAR SCREEN
                   (5)  IF A BEGINNING TIME WAS NOT ENTERED THEN SET START_TIME TO 0
                   (6)  IF AN ENDING TIME WAS NOT ENTERED THEN SET  END_TIME TO 16
                   (7)  ANNOUNCE TO THE MFS THAT THE LIMITS ARE SET (DO_STATS = 1)
                   (8)  WHEN THE DATA IS READY (STATS_RDY = 1) THEN PROCEDE TO DISPLAY IT */

                /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
241   2          TEMP_BUFFER(0),TEMP_BUFFER(1) = 30H; /* SET THE BUFFER = 00 */

242   2          CALL PRINT (@STATS_MSG_1, LENGTH(STATS_MSG_1));
243   2          CALL CI;
244   2          IF CHAR <> CR THEN TEMP_BUFFER(0) = CHAR;
246   2          ELSE GOTO SKIP_IT;  /* JUMP IF A CR IS RECEIVED  */
247   2          CALL CI;
248   2          IF CHAR <> CR THEN DO;
250   3                       CALL CO (CARRIAGE_RETURN);
251   3                       TEMP_BUFFER(1) = CHAR;
252   3                      END;
253   2          IF CHAR = CR THEN  DO;
255   3                       TEMP_BUFFER(1) = TEMP_BUFFER(0);
256   3                       TEMP_BUFFER(0) = 30H;
257   3                      END;
258   2      SKIP_IT: CALL CO (LF);
259   2               CALL CO (LF);
260   2               START_TIME = ASCII_TO_HEX (@TEMP_BUFFER(0),2);
261   2               IF START_TIME > 16 THEN START_TIME = 16;
                /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
263   2          TEMP_BUFFER(0),TEMP_BUFFER(1) = 30H;
264   2          CALL PRINT (@STATS_MSG_2, LENGTH(STATS_MSG_2));
265   2          CALL CI;

266   2          IF CHAR <> CR THEN TEMP_BUFFER(0) = CHAR;
268   2          ELSE DO;              /* IF CR THEN SET END_TIME TO 16 AND BAIL OUT */
269   3          END_TIME = 16;
270   3          GOTO SKIP_IT2;
271   3          END;

272   2          CALL CI;
273   2          IF CHAR <> CR THEN DO;
275   3                       CALL CO (CARRIAGE_RETURN);
276   3                       TEMP_BUFFER(1) = CHAR;
277   3                      END;
```

Figure G-4(B). Addition to "Keyboard-IO" Module.

```
278  2          IF CHAR = CR THEN  DO;
280  3                      TEMP_BUFFER(1) = TEMP_BUFFER(0);
281  3                      TEMP_BUFFER(0) = 30H;
282  3                    END;
283  2     SKIP_IT2: CALL CO (LF);
284  2               CALL CO (LF);
285  2               CALL CO (LF);
286  2               END_TIME = ASCII_TO_HEX (@TEMP_BUFFER(0),2);
287  2               IF END_TIME > 16 THEN END_TIME = 16; /* WE NOW HAVE BOTH VALUES */


289  2     IF END_TIME <= START_TIME THEN GOTO RE_ENTRY;

291  2     CALL PRINT (@STATS_MSG_3, LENGTH (STATS_MSG_3));

292  2     DO_STATS = 1;                /* ALERT THE MFS */

293  2         DO WHILE STATS_RDY = 0;      /* WAIT TILL THE DATA IS READY */
294  3         END;

295  2     CALL PRINT (@STATS_MEAN_H, LENGTH (STATS_MEAN_H));
296  2     HOR_MEAN: DO I = 0 TO 5;
297  3               CALL CO (H_MEAN(I));
298  3               END;


299  2     CALL PRINT (@STATS_STD_DEV_H, LENGTH (STATS_STD_DEV_H));
300  2     HOR_DEV:  DO I = 0 TO 5;
301  3               CALL CO (H_DEV(I));
302  3               END;

303  2     CALL PRINT (@STATS_DIR_H, LENGTH(STATS_DIR_H));
304  2     IF MEAN_RIGHT THEN CALL PRINT (@RIGHT, LENGTH (RIGHT));
306  2     ELSE CALL PRINT (@LEFT, LENGTH (LEFT));


307  2     CALL PRINT (@STATS_MEAN_V, LENGTH (STATS_MEAN_V));
308  2     VER_MEAN: DO I = 0 TO 5;
309  3               CALL CO (V_MEAN(I));
310  3               END;

311  2     CALL PRINT (@STATS_STD_DEV_V, LENGTH (STATS_STD_DEV_V));
312  2     VER_DEV:  DO I = 0 TO 5;
313  3               CALL CO (V_DEV(I));
314  3               END;

315  2     CALL PRINT (@STATS_DIR_V, LENGTH(STATS_DIR_V));
316  2     IF MEAN_UP THEN CALL PRINT (@UP, LENGTH(UP));
318  2     ELSE CALL PRINT (@DOWN, LENGTH (DOWN));

319  2     CALL CO(350);  /* SHIFT BACK TO VECTOR MODE */

320  2     END STATS_ROUTINE;

321  1     END KEYBOARD_IO;
```

Figure G-4(C). Addition to "Keyboard-IO" Module.


FILM
9-84