



## **DABS Interrogation Modulator**

WILLIAM A. BARNWELL

*Identification Systems Branch  
Radar Division*

April 16, 1987

SECURITY CLASSIFICATION OF THIS PAGE

<b>REPORT DOCUMENTATION PAGE</b>				
1a. REPORT SECURITY CLASSIFICATION <b>UNCLASSIFIED</b>		1b. RESTRICTIVE MARKINGS		
2a. SECURITY CLASSIFICATION AUTHORITY		3. DISTRIBUTION / AVAILABILITY OF REPORT  Approved for public release; distribution unlimited.		
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE		4. PERFORMING ORGANIZATION REPORT NUMBER(S)  NRL Report 9021		
5. MONITORING ORGANIZATION REPORT NUMBER(S)		6a. NAME OF PERFORMING ORGANIZATION  Naval Research Laboratory		
6b. OFFICE SYMBOL (If applicable) Code 5351		7a. NAME OF MONITORING ORGANIZATION		
6c. ADDRESS (City, State, and ZIP Code)  Washington, DC 20375-5000		7b. ADDRESS (City, State, and ZIP Code)		
8a. NAME OF FUNDING / SPONSORING ORGANIZATION Federal Aviation Administration		8b. OFFICE SYMBOL (If applicable) ARD-242		9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)  Washington, DC 20590		10. SOURCE OF FUNDING NUMBERS		
		PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.
		WORK UNIT ACCESSION NO. DN980-188		
11. TITLE (Include Security Classification)  DABS Interrogation Modulator				
12. PERSONAL AUTHOR(S) Barnwell, William A.				
13a. TYPE OF REPORT Final		13b. TIME COVERED FROM 2/79 TO 10/81	14. DATE OF REPORT (Year, Month, Day) 1987 April 16	15. PAGE COUNT 43
16. SUPPLEMENTARY NOTATION				
17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)	
FIELD	GROUP	SUB-GROUP	Discrete Address Beacon System (DABS/Mode S)	
			IFF system test equipment	
			Microprocessor-controlled test equipment	
19. ABSTRACT (Continue on reverse if necessary and identify by block number)				
<p>This report describes the development of a user-friendly discrete address beacon system (DABS) interrogation modulator unit. The unit was designed to support an automatic testing program to define the degree of compatibility of the DABS (now Mode S) system with the Mark XII Mode 4 IFF system.</p> <p>The design philosophy, implementation, and software development of this microcomputer-controlled phase and amplitude waveform modulator is reviewed both for users of this equipment and as a guide for those developing similar specialized equipment. This interrogation modulator has proven to be flexible in use in connection with DABS as well as on other IFF programs. Appendix A includes helpful development and debugging aids and some lessons learned.</p>				
20. DISTRIBUTION / AVAILABILITY OF ABSTRACT <input checked="" type="checkbox"/> UNCLASSIFIED/UNLIMITED <input type="checkbox"/> SAME AS RPT. <input type="checkbox"/> DTIC USERS			21. ABSTRACT SECURITY CLASSIFICATION <b>UNCLASSIFIED</b>	
22a. NAME OF RESPONSIBLE INDIVIDUAL William A. Barnwell			22b. TELEPHONE (Include Area Code) (202) 767-2655	22c. OFFICE SYMBOL Code 5351

## CONTENTS

INTRODUCTION .....	1
DABS INTERROGATION MODES AND SIGNAL CHARACTERISTICS .....	1
HARDWARE OVERVIEW .....	2
RF SUBSECTION .....	4
DIGITAL SUBSECTION .....	4
CONTROL PANEL SUBSECTION .....	5
SOFTWARE OVERVIEW .....	6
MICROCOMPUTER FIRMWARE .....	9
APPLICATION SOFTWARE .....	9
IMPLEMENTATION AND OPERATION .....	13
CONCLUSIONS .....	17
REFERENCES .....	17
APPENDIX A — Lessons Learned .....	18
APPENDIX B — PIA I/O Assignments .....	20
APPENDIX C — NDABLB Program Listing .....	23
APPENDIX D — NDAB Program Listing .....	33

## DABS INTERROGATION MODULATOR

### INTRODUCTION

As a result of air traffic densities projected for the 1990s, the ensuing impact on the round reliability (% of replies/interrogation) of civilian air traffic control radar beacon system (ATCRBS) transponders, and on friendly replies unsynchronized in time (FRUIT) rates received by ATCRBS interrogators, the Federal Aviation Administration (FAA) is developing a replacement system, the discrete address beacon system (DABS), now known as Mode S. To control the interference resulting from the projected densities, DABS includes a significant change in both the transmitted signals-in-space format and operational use [1]. Because the civilian ATCRBS and military identification friend or foe (IFF) system (Mark XII) share frequencies, electromagnetic compatibility (EMC) and interference (EMI) issues arise. A DABS/MK XII Mode 4 compatibility program was established to resolve EMC/EMI issues between the joint operation of DABS and the MK XII Mode 4.

The Identification Systems Branch performs automated testing of IFF systems and techniques. Tests were performed to measure the interaction of the two systems during the DABS/MK XII Mode 4 compatibility program. The test results served to validate and supplement computer simulations assessing the degree of EMC. To support the automated testing, a variety of specialized equipments have been developed, one of which is the DABS interrogation modulator.

The discussion of the DABS interrogation modulator includes a brief description of the Mode S interrogation formats, a hardware review, a software review, and a summary of the current implementation and operation. Discussions of the lessons learned during the design and development of an embedded, microprocessor-controlled, smart, stand-alone instrument are covered in Appendix A.

### DABS INTERROGATION MODES AND SIGNAL CHARACTERISTICS

The DABS interrogations are transmitted on the same frequency channel as the current ATCRBS interrogations, but with a tighter frequency specification,  $1030 \pm 0.01$  MHz. The interrogations consist of a series of On-Off-Keyed (OOK) pulses, some of which have internal phase modulation. As in the civilian ATCRBS and military Mark XII interrogators, most of the DABS interrogator pulses are transmitted through the main antenna. The other pulses are transmitted through a separate control antenna, the sidelobe suppression (SLS) control antenna. The transponder uses a comparison of the effective radiated energy (ERP) of the pulses transmitted through each antenna to inhibit replying to interrogations received outside the main beam. The latter technique is termed interrogation sidelobe suppression (ISLS), and the pulses transmitted through the control antenna are ISLS pulses. The pulses that make up the ATCRBS, Mark XII, and DABS interrogations have been given designations "P1" to "P6," which allow easy functional grouping and discussion.

There are two DABS interrogation formats: the ATCRBS/DABS All-Call and the DABS itself. Each of these formats has two modes that only differ by either pulse spacing, in the case of Mode A and Mode C All-Call interrogations, or by pulse length, in the case of short and long DABS interrogations. Figure 1 shows the ATCRBS/DABS All-Call interrogation pulse sequences. The All-Call interrogations consist of three pulses: P<sub>1</sub>, P<sub>3</sub>, and P<sub>4</sub>. One or two control (ISLS) pulses, P<sub>2</sub> alone, or

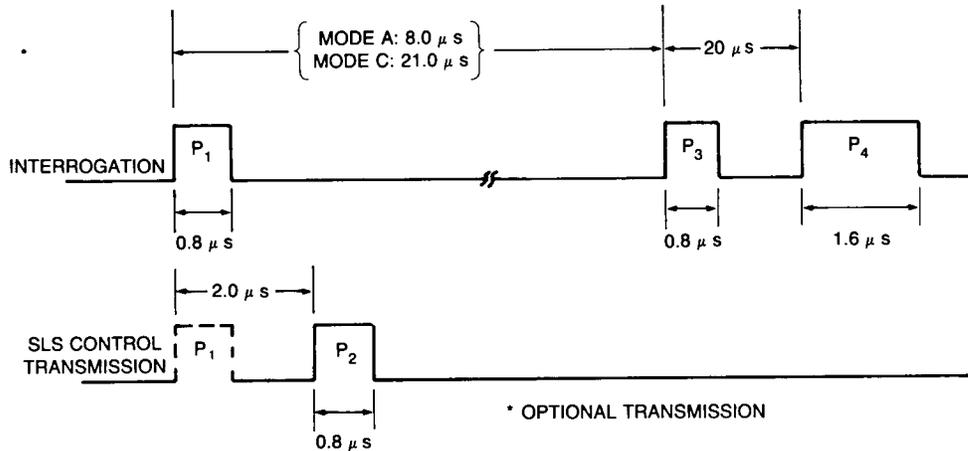


Fig. 1 — ATCRBS/DABS All-Call interrogation pulse sequence

$P_1$  and  $P_2$ , are transmitted through the SLS antenna.  $P_1$  through  $P_4$  are OOK pulses without phase modulation. (See Table 1 for pulse shapes specifications.) When only the  $P_2$  pulse is transmitted, the technique is termed *sequential ISLS*.

Table 1 — Pulse Shapes for DABS Interrogations

Pulse Designator	Pulse Duration ( $\mu\text{s}$ )	Duration Tolerance ( $\mu\text{s}$ )	Rise Time		Decay Time	
			Min ( $\mu\text{s}$ )	Max ( $\mu\text{s}$ )	Min ( $\mu\text{s}$ )	Max ( $\mu\text{s}$ )
$P_1, P_2, P_3, P_5$	0.8	$\pm 0.1$	0.05	0.1	0.05	0.2
$P_4$	1.6	$\pm 0.1$	0.05	0.1	0.05	0.2
$P_6$ (short)	16.25	$\pm 0.25$	0.05	0.1	0.05	0.2
$P_6$ (long)	30.25	$\pm 0.25$	0.05	0.1	0.05	0.2

Figure 2 shows the DABS interrogation modes. The DABS interrogations consist of three pulses:  $P_1$ ,  $P_2$ , and  $P_6$ . One control (ISLS) pulse,  $P_5$ , is transmitted through the SLS antenna at the same time as  $P_6$  is transmitted through the main antenna, simultaneous ISLS.  $P_1$ ,  $P_2$ , and  $P_5$  are each OOK pulses without phase modulation, but  $P_5$  differs in pulse width (see Table 1).  $P_6$ , however, is an OOK pulse with a complex structure of binary phase shift keying (BPSK) data modulation.  $P_6$  also has two pulse lengths corresponding to the short and long DABS interrogation modes which allow differing amounts of data to be sent—56 or 112 data bits. Within  $P_6$ , the synchronization phase shift occurs  $1.25 \mu\text{s}$  after the leading edge. Each  $180^\circ$  data phase shift, within  $P_6$ , occurs at a time  $(N + 0.25) \pm 0.02 \mu\text{s}$  ( $N \geq 2$ ) after synchronization phase reversal, equivalent to a 4 Mb/s data rate. The last phase reversal is greater than a  $0.5 \mu\text{s}$  from the trailing edge of  $P_6$  to eliminate the interference due to the effects of the RF pulse trailing edge.

### HARDWARE OVERVIEW

Figure 3 depicts the DABS interrogation modulator. Figure 4 is a simplified block diagram. The unit is a microcomputer-controlled, phase and amplitude waveform modulator. It has a control panel through which the operator selects one of eight interrogation options; seven operator-loaded fixed-data interrogations and one continuously random data interrogation. Each of the interrogations is formed by clocking binary data out of two independent control buffers (256 samples in length), one for phase

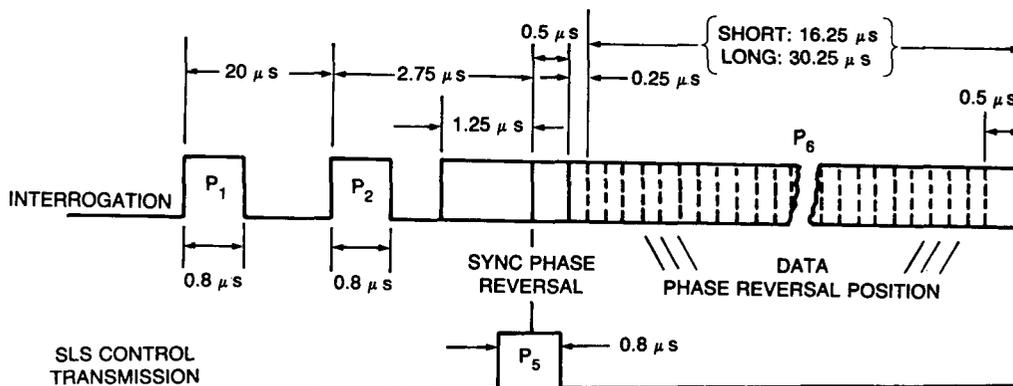
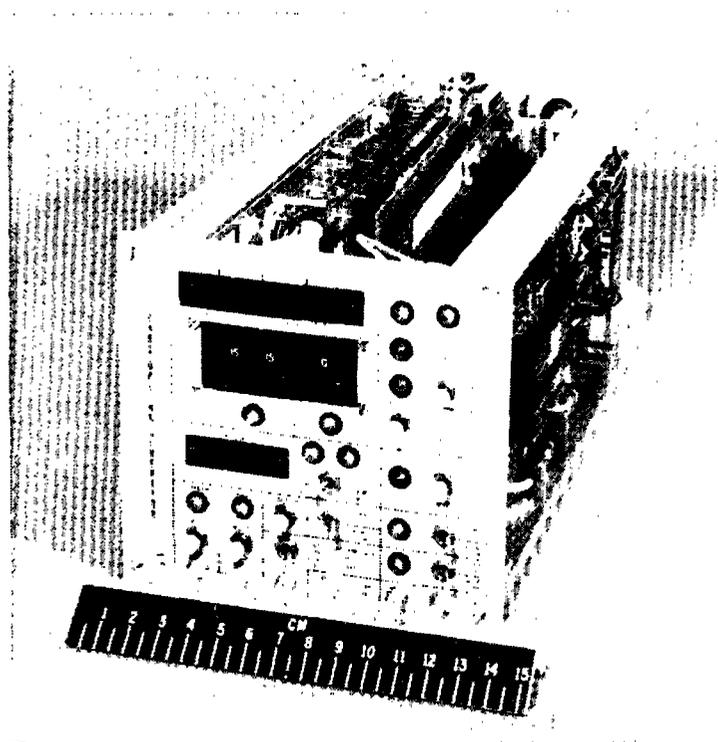


Fig. 2 — DABS interrogation, pulse sequence



80087(3)

Fig. 3 — DABS interrogation modulator

shift keying (PSK) and the other for pulse amplitude modulation (PAM) keying. Upon receiving a trigger after being loaded, the contents of the buffers are always synchronously clocked out at a 4 MHz rate for 136 samples. The latter corresponds to the DABS interrogation modulation rate and message duration of the long format interrogation. The generation of the short format interrogation is software controlled by zeroing the PAM buffer data contents after 79 samples, during the load operation. The unit contains a differential phase shift keying (DPSK) modulator and provides a PAM video output for driving an external PIN diode RF switch, such as the HP 8403A.

The modulator is designed to be interactive during initialization and operation. For example, the operator is guided through the initialize, load, and operate procedures by blinking LEDs. During the loading operation, memory errors are checked and LEDs signal if a problem has occurred. During operation, the microcomputer monitors various user inputs for changes; for example, it monitors

which interrogation to transmit, whether to hold the current random message for retransmission, or to reload. The number of options available, the desire to have a programmable unit, and the desire to make the unit "user friendly" drive the design of the interactive display and the control software.

With the preceding short hardware overview for reference, the remaining part of the hardware section focuses on the RF, digital, and control panel subsections. The microcomputer-related discussions are reserved for the software section of the report.

## RF SUBSECTION

The RF subsection is composed of both microwave circuits and the digital to analog interfaces. The PAM driver is the simplest, consisting of a shift register delay to compensate for the DPSK encoding delay, followed by a line driver that drives an external PIN diode RF modulator. The DPSK modulator is composed of a flip-flop differential encoder driving a bipolar line driver in a current loop configuration. The current loop line driver modulates a double balanced mixer's IF port (Anzac MD-150). With an external 1030 MHz signal generator feeding the DABS interrogation modulator at the RF port of the mixer, a DPSK-modulated carrier appears at the local oscillator (LO) port. The DPSK signal passes through an isolator and then to an output connector. The final interrogation signal is obtained by taking the DPSK output and passing it through the external PIN diode PAM modulator mentioned earlier. Figure 5 shows a typical DABS long format interrogation.

## DIGITAL SUBSECTION

The digital subsection includes three main functional blocks: a clock synthesizer, the control buffers, and the digital interface. The clock synthesizer and digital interface blocks are grouped as the timing and control block in Figure 4. The clock synthesizer generates the various clock and control signals from a 40 MHz source. The control of the clock synthesizer is maintained by the 6800-based microcomputer through the digital interface; microcomputer output lines drive override gates used to control the source of interrogation triggers (external, push-button, or internal microcomputer), or the microcomputer can assume complete control of each clock and control line during the control buffer load sequence. The 40 MHz reference limits the pulse-to-pulse jitter to be less than 50 ns from the external interrogate pulse.

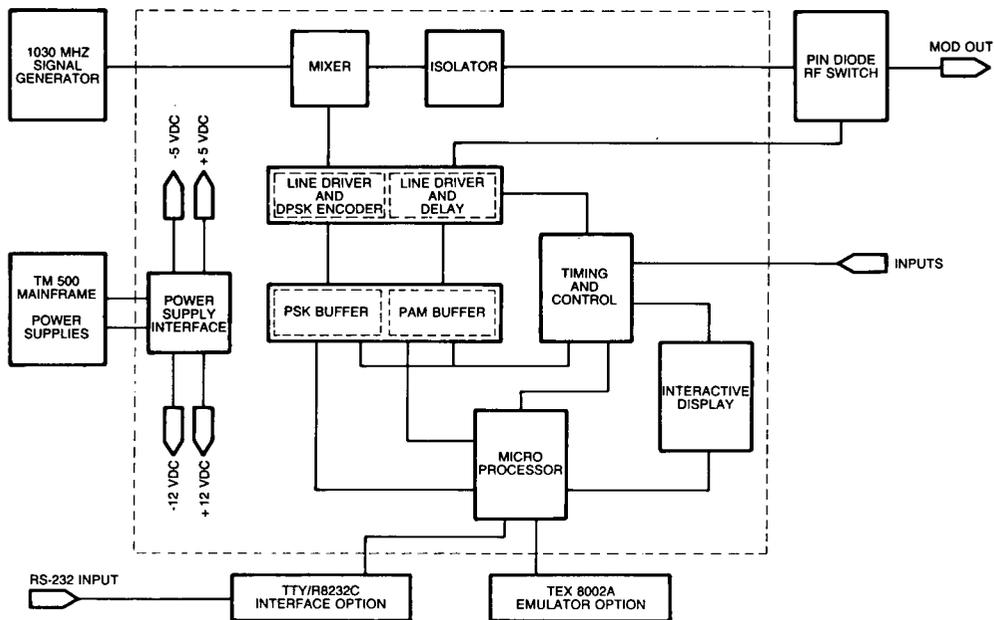


Fig. 4 — Timing and control block

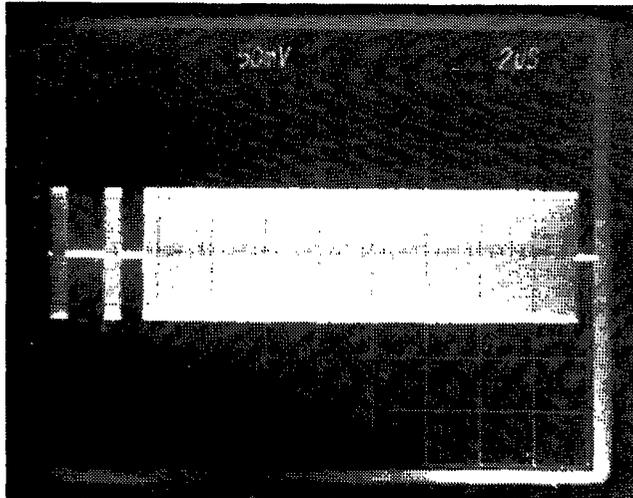


Fig. 5 — DABS long format interrogation

The control buffer block includes two independent RAM buffers, the address counters, and the parallel-to-serial output registers. Each buffer is composed of two multiplexed  $8 \times 128$ -bit RAMs, to satisfy the RAM access time requirements of loading and dumping during operation. The address counters and multiplex circuitry are contained within the block and can be controlled by the microcomputer during any load sequence, either during the initial load prior to operation, or during the random message reload. The two control buffers feed two synchronously clocked parallel-to-serial shift registers that are also composed of multiplexed registers. The multiplexed streams are combined to form two synchronously clocked binary data streams, the PSK control and the PAM control.

The RAM buffers are divided into eight segments that are accessed by the memory section select (MSS) thumbwheel switch during loading and under operation. The MSS thumbwheel controls the upper 3 bits of the RAM buffer address lines, while the remaining 8 bits of address are supplied by a synchronous counter. Of the 8 bits, the lowest order bit is used as the multiplex control. During the loading operation, the operator is cued to change the MSS until each section is loaded or the loading is terminated by setting the MSS to 15. During operation, the MSS allows the operator to change the transmitted interrogation (which corresponds to the upper three bits) and is sensed for detection of error values (7 to 15 or other unloaded segments) and requests for random message operation (0).

The digital interface block contains most of the microcomputer-generated control lines and the gating to assert control of the digital section by the microcomputer during loading and operation. Some of the signals are used directly, and the rest undergo gating to form the final system control lines and clocks; an example is the "external trigger input" that is gated by the "internal/external" switch and the computer override line.

### CONTROL PANEL SUBSECTION

Figure 6 shows the control panel display. It is composed of input and output connectors, LED indicators, and a variety of switches. Each of the LEDs are controlled by the microcomputer, and all of the switches are sensed by the microcomputer. The *random sequence hold*, *reset*, and *external interrogate pulse* inputs are latched for sensing by the microcomputer. An LED is associated with each of the switches (toggle, thumbwheel, or push-button) used to load or operate the unit. In two cases, the LED indicates a change of the label for a byte-wide or a multibyte-wide display: "Address of Data" or "Random Sequence Count" and "Envelope Data" or " $\Delta\phi$  Data Entry."

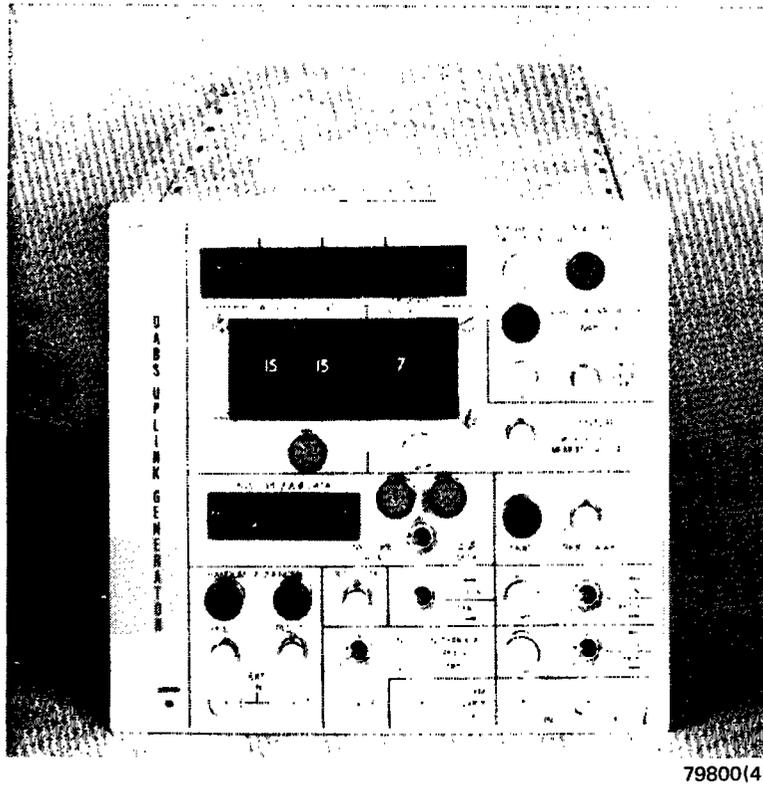


Fig. 6 — DABS interactive display

79800(4)

The design of the control panel and the use of an embedded microcomputer provide device flexibility with a minimum of operator confusion and frustration. From power on, the operator is guided through the initialization (loading) and operate process by the blinking LEDs. The configuration is designed to allow the operator the choice of loading ROM-stored messages (PSK) and waveforms (PAM) or to create any sequence of phase and amplitude changes within the  $34 \mu\text{s}$  of a long format or the  $20 \mu\text{s}$  of a short format interrogation. The choice is limited to 8 and controlled by the MSS switch. Setting the MSS to values from 0 to 7 implies that the operator enters ROM-stored interrogations for each value set, while setting MSS to values from 8 to 14 implies that the operator enters the PSK and PAM sequences using the data byte entry thumbwheels for each value set. In addition, the random message format can have either the ROM-stored DABS PAM sequence (MSS set to 0) or an operator-loaded PAM sequence (MSS set to 8). Table 2 summarizes the control/display definitions, and Table 3 summarizes the current preprogrammed interrogations.

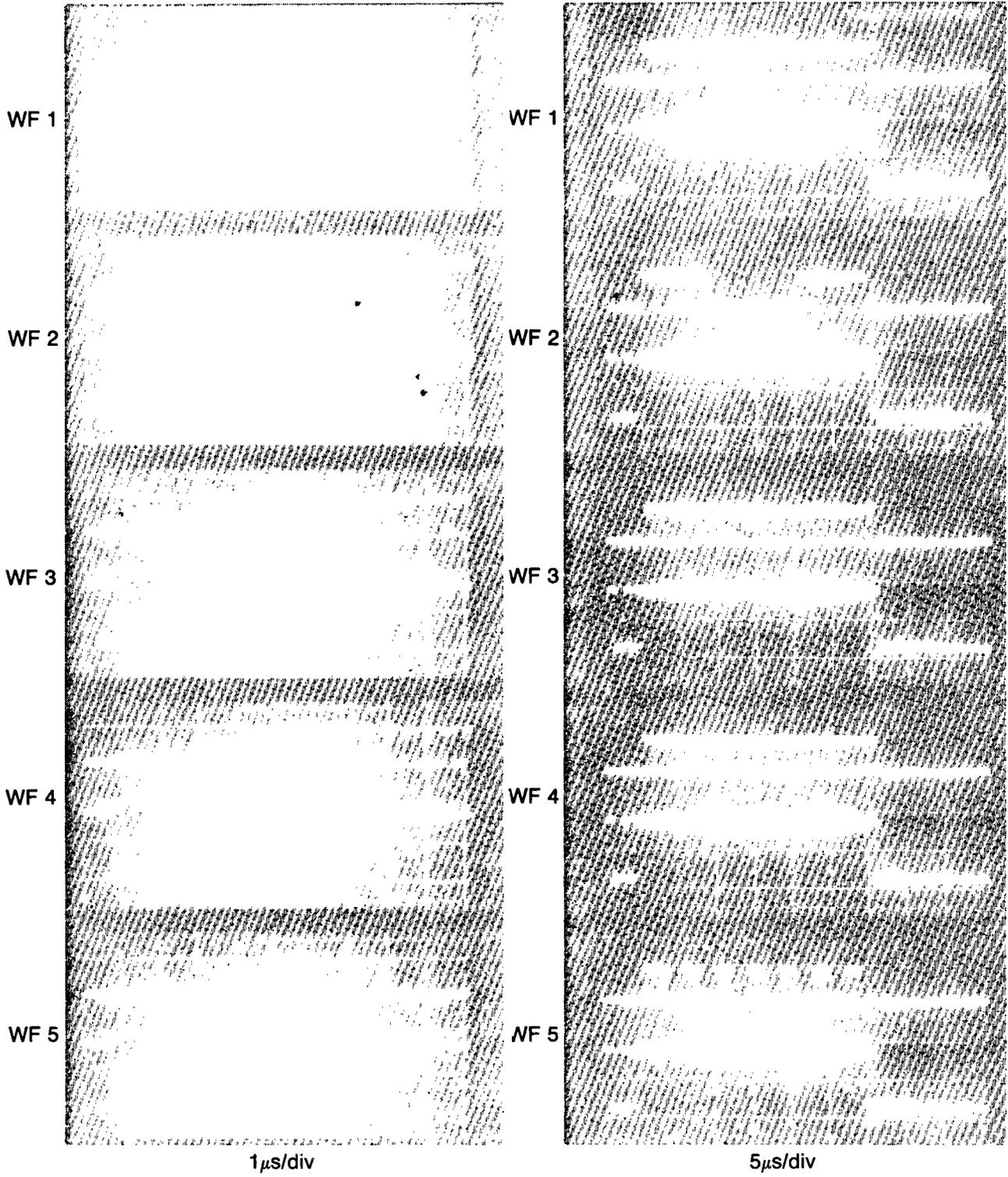
## SOFTWARE OVERVIEW

The DABS interrogation modulator contains an embedded microcomputer system (WINTEK, Inc.'s Motorola 6800 Micro Modules). The embedded microcomputer's primary function is to provide for the adaptive, *user friendly* loading and operation of the DABS interrogation modulator. From power up, the microcomputer asserts positive control of the loading and operation of the analog and digital subsections, while maintaining and sensing the control panel for operator interaction. The various tasks performed by the microcomputer range from guiding the operator through the load sequence, with blinking LEDs, to enabling each interrogation transmission after the operate-mode subtasks are complete, such as the calculation of a new random message and reloading of the PSK buffer before allowing the next transmission to occur. To accomplish the tasks, a 5 kB EPROM stored control program services ten Motorola 6821 PIA ports (over 100 input/output lines) in real time.

Table 2 — Control/Display Definitions

Name	Purpose	Type
Reset Address Counter	To reset the memory section address counter. LED indicates switch is active.	Push-button
Data Byte Entry	To enter 8 bits of data to either the envelope or the $\Delta 0$ data buffers. Byte-wide LED indicates what is loaded. Single LED shows switch is active.	2 hex digit thumbwheel
Memory Section Select (MSS)	To select the section of memory to be loaded or transmitted. For current case, 1 of 7 sections. Address being loaded is shown in multibyte-wide LED. Single LED shows switch is active.	1 hex digit thumbwheel
Enter DBE/MSS	To strobe either the DBE or MSS thumbwheel data into memory. LED indicates switch is active.	Push-button
Skip/Okay	To skip over a byte of data that is correct as is or to validate data that have been written to memory. LED shows switch active.	Push-button
Envelope/ $\Delta$ Data	To indicate whether envelope or $\Delta$ data buffer is being loaded. LED indicates switch is active.	Toggle
Load/Operate	To select desired mode of the DABS interrogation modulator. LED indicates switch active.	Toggle
Long/Short	To select whether a long or short DABS format is to be loaded. LED indicates switch active.	Toggle
Reset Random Sequence	To reset the random sequence generator to start of sequence. LED indicates switch active.	Push-button or external input.
Hold Random Sequence	To hold last sequence for retransmission. LED shows switch active.	Push-button or ext. input
Local/Remote	Used to select action of "Reset CPU" Local => restart application program, Remote => run debug program.	Toggle
Reset CPU	Maskable interrupt.	Push-button

Table 3 — Preprogrammed Interrogations



Top: PSK out; Bottom: PAM out; Long DABS Interrogation

To a large degree, the size of the control program and the large number of I/O lines are driven by the goal of providing a flexible stand-alone test fixture, easily altered by any operator during a DABS/Mark XII Mode 4 compatibility test sequence. The control program is a real-time task-oriented procedure with three primary tasks: initialize, load, and operate. Within each of the main tasks, sub-tasks exist. For example, the operate task breaks into transmit; check for a new interrogation selection; if new or random, generate and load the new interrogation; then transmit. The resultant control program is over 700 lines of structured source code with 24 procedures, 1 main and 23 sub-procedures, 2 of which are interrupt handling procedures. After using the NRL DEC-10 computer maintained PL/W compiler and object code linker, the source code is reduced to a 5 kB EPROM load module for eventual loading into the individual EPROMs of the ROM module.

The control program is written in PL/W, a vendor-supplied variation of a standard 8080 microprocessor high-level language, PL/I. The PL/W variation generates 6800 microprocessor load modules. With a generous use of comment statements and subprocedures coupled with a FORTRAN-like syntax, the source code is readable and quickly generated. However, a long and involved process is required to transfer the load module from the DEC-10 to the individual EPROMs, debug the software, debug the hardware, and detect errors in the DEC-10 PL/W compiler itself. Appendix A discusses the inefficiencies of the latter process.

With the preceding short software overview for reference, the remaining part of the software section focuses on two subsections, microcomputer firmware and application software. A discussion of software development, downloading the load module to EPROMs, and debugging is covered in Appendix A.

## **MICROCOMPUTER FIRMWARE**

The embedded microcomputer system consists of three modules (circuit cards); the control module, the ROM module, and the parallel I/O module. The modules are interconnected through a mother board and edge connectors. The system is then housed within the DABS interrogation modular package. Table 4 summarizes the general features of each module. The specific control module configuration used for this application is a complete single board microcomputer. The control module has a 6800-series microprocessor with 512 B of RAM, 512 B of EPROM, a RS-232 serial I/O port, and 40 parallel input or output lines (8 of which are control lines). The control module EPROM contains a modified FANTOM II (vendor-supplied debug program) for stand-alone maintenance. The ROM module contains 8 kB of EPROM storage of which 5 kB are used by the current application program. The parallel I/O module contains 8 parallel PIA ports each having 8 input or output lines and 2 control lines.

Because of a desire to maintain software compatibility between FANTOM II and the control program, and the number of RAM, ROM, and I/O ports, slight modifications are made to both the control module and the parallel I/O module. The modifications increase the address space decoding to resolve hardware conflicts. Table 5 shows the hardware memory map for the partial address decoding and control lines. The table is an extension of the vendor-supplied table assumed by the FANTOM II program. Appendix B details the parallel I/O pin assignments, by PIA port address, and shows each line's label and direction served.

## **APPLICATION SOFTWARE**

The control program for the DABS interrogator modulator is written in PL/W and is over 700 lines of source code in length. The program is composed of one main procedure (task scheduler) and 23 subprocedures. The main procedure is divided into 3 segments; initialize, load, and operate. In support of the main procedure, a library segment contains the 23 subprocedures, nearly all of which are hardware interface programs controlling one or more of the PIA port I/O lines. The control program is divided into two modules each of which is shown in Appendix C and D respectively. The division of the program into modules, which individually compile to less than 2 kB, is caused by

Table 4 — Wintek Module Features

Control Module	Parallel I/O Module
Features/Design Details	Features/Design Details
<b>CLOCK</b> 1MHz crystal controlled	<b>PORTS</b> 2, 4, 6 or 8 ports. Each port consists of 8 I/O lines plus 2 control lines. Each port can be programmed as an input port or an output port. All control lines are individually programmable.
<b>MPU</b> 6800 instruction set	<b>INPUT/OUTPUT DRIVE</b> PIA inputs/outputs are buffered by 74LS245 octal tri-state transceivers. Each output I/O line is capable of driving 15 TTL loads or 150 low power loads. Inputs are low power Schottky hysteresis inputs.
<b>RAM</b> 128, 256, 384, or 512 bytes, 1, 2, 3, or 4 6810s	<b>MEMORY SPACE</b> jumper selectable. In some cases Parallel I/O Modules with more than 4 ports can result in a memory space conflict with the serial I/O port or with some of the RAM on the Control Module. All conflicts are easily avoided by designing the system in accordance with WINTEK Application Note AN-0010 or by choosing any one of the following configurations: (1) Use Parallel I/O Modules with 4 or fewer ports; (2) Use a Control Module less ACIA, (3) Use a Control Module less RAM 3 and RAM 4.
<b>ROM</b> socket for 1/2K, 1K, 2K, or 4K ROM or EPROM. Shipped strapped for 2708/MCM68308, 2704, all 2716, 2732, MCM65317 types require restrap.	<b>CONNECTORS</b> each port terminates in a 20-pin male connector. The mate connector could be a 20-pin female per port or a 50-pin female per 2 adjacent ports. Male connectors are also suitable for wire wrapping.
<b>SERIAL I/O</b> programmable UART. The serial I/O option consists of 6850, baud rate clock and connector. The baud rate clock is set at the factory for 300 baud. A trim pot is provided for 110 to 1200 baud. For higher baud rates the cassette/RS232 Module is recommended. The connector is a 14-pin DIP socket with ACIA signals, bus voltages, and baud clock.	<b>POWER</b> +5V at 1.5 A max (8 ports)
<b>PARALLEL I/O</b> up to 32 TTL lines individually programmable as input or output plus 8 programmable control lines. The parallel I/O option consists of 1 or 2 PIAs and 50-pin male connector with PIA signals and +5V.	
<b>POWER (three voltage version)</b> +5V $\pm 5\%$ at 600 ma $\pm 12V \pm 5\%$ at 50 ma each	
<b>POWER (single voltage version)</b> +5V $\pm 5\%$ at 600 ma	
<b>ROM Module</b>	
<b>MEMORY</b> up to 32K bytes (2716 or 2732) or 16K bytes (2708). Module span is field modifiable to half these maximum values.	
<b>MEMORY SPACE</b> field selectable at starting address (hex) 0000, 4000, 8000, or C000.	
<b>POWER</b> +5V at 200 mA plus: $\pm 12V$ at 70 mA each 2708 or +5V at 100 mA each 2716/2516 +5V at 200 mA each 2732	

Table 5 — Hardware Memory Map and Control Lines

ADDRESS LINE	15 F	14 E	13 D	12 C	11 B	10 A	09 9	08 ..	07 ..	06 ..	05 ..	04 ..	03 ..	02 ..	01 ..	00 ..
ROM	1	1	1	1	1	1	A	A	A	A	A	A	A	A	A	A
RAM 1	X	X	X	0		L	0	0	0	A	A	A	A	A	A	A
RAM 2	X	X	X	0		L	0	0	1	A	A	A	A	A	A	A
RAM 3	X	X	X	0		L	0	1	0	A	A	A	A	A	A	A
RAM 4	X	X	X	0		H	0	1	1	A	A	A	A	A	A	A
ACIA	X	X	X	0		L	1	X	X	X	0	0	1	X	X	A
PIA20	X	X	X	0		L	1	X	X	X	0	1	0	X	X	A
PIA10	X	X	X	0		L	1	X	X	X	1	0	0	X	A	A
PIA 1	X	X	X	0		E	1	X	X	X	X	X	0	0	A	A
PIA 2	X	X	X	0		E	1	X	X	X	X	X	0	1	A	A
PIA 3	X	X	X	0		E	1	X	X	X	X	X	1	0	A	A
PIA 4	X	X	X	0		E	1	X	X	X	X	X	1	1	A	A
ROMEN 1	X	X	X	X		H	X	X	X	X	X	X	X	X	X	X
ROMEN 2	1	1	1	1	X	X	X	X	X	X	X	X	X	X	X	X
RAMSL	X	X	X	0	X	X	X	X	X	X	X	X	X	X	X	X
I/ODET	X	X	X	X		E	X	X	X	X	X	X	X	X	X	X

Key: A => All used.  
 L => Bits are alike.  
 E => Exclusive OR.  
 H => All High.  
 X => Don't care.

address space conflicts that force a 1 kB separation between the two modules. With the latter in mind, the library segment spans Appendix C and the first part of Appendix D. The main procedure follows the library segment and is contained within Appendix D.

The initialize segment is entered on power up or if the *reset* pushbutton is pushed. Within this segment the interrupts are disabled, PIA port direction registers are set, initial output data are loaded to the respective PIA ports, and an image of the PIA ports is saved in RAM (both input and output). The impact of this process on the hardware is to initialize the buffers, displays, and control lines. The flow to the next task (load) is controlled by the setting of the *load/operate* toggle switch. If set to *load*, the process continues; if set to *operate*, the blinking LED announcement to the operator is started.

The load task is entered by the operator setting the *load/operate* switch to *load*. Without going into the exact detail, the operator is guided through the succeeding steps with blinking LEDs being extinguished by operator action on the particular switch involved. First, a control buffer memory section is selected for loading by the value of the MSS switch. Second, a *long/short* interrogation format is selected. Next, based on the value of the MSS (see Table 6), the PAM and the PSK image buffers are loaded followed, respectively, by the loading of the hardware control buffers. Then, the cycle is repeated until each of the allowed sections is filled. At that time, the *memory section loading complete* LED is lit and the *load/operate* LED is blinked. The blinking continues until the operator switches to *operate*.

The task segment is entered with the unit having been loaded and the *load/operate* toggle switched to *operate*. At that time the MSS value is read (the interrogation desired for transmission), loaded to the ADDBUS which selects the proper segment of the hardware control buffers, and the transmission begins upon receipt of a trigger pulse (either internal or external). If the interrogation is a random data type, the random sequence counter and display are initiated. After transmission, the

Table 6 — Memory Section Select Switch Settings

MSS	During Loading		
	PSK		PAM
0	Random message seq.		DABS block PAM Long or short
1	ROM stored PSK seq. #1		DABS block PAM Long or short
2	ROM stored PSK Seq. #2		DABS block PAM Long or short
3	ROM stored PSK Seq. #3		DABS block PAM Long or short
4	ROM stored PSK Seq. #4		DABS block PAM Long or short
5	ROM stored PSK Seq. #5'		DABS block PAM Long or short
6	ROM stored PSK Seq. #6		DABS block PAM Long or short
7	Not available because of address ambiguity with 15.		
8	Random message seq.		Operator selected PAM seq #0
9	Operator spec PSK seq #1		Operator selected PAM seq #1
10	Operator spec PSK seq #2		Operator selected PAM seq #2
11	Operator spec PSK seq #3		Operator selected PAM seq #3
12	Operator spec PSK seq #4		Operator selected PAM seq #4
13	Operator spec PSK seq #5		Operator selected PAM seq #5
14	Operator spec PSK seq #6		Operator selected PAM seq #6
15	Use indicates desire to terminate loading process early.		

During Operation
MSS
0 — 6 Available
7 — 15 Not available

new random sequence is calculated, loaded to the PSK buffer image, the "0" PSK hardware control buffer is loaded, and the random sequence counter is incremented and displayed. Then, for both fixed and random cases, the MSS is read and passed out to the ADDBUS. Finally, the *load/operate* toggle is read before the next transmission. Figure 7 shows the binary pseudo random data bit generator implemented in software. The generator is a maximal linear sequence shift register of length 13, with a 20033 octal tap position and a 00001 octal seed [3]. The binary data out is pseudo noise with period 8191.

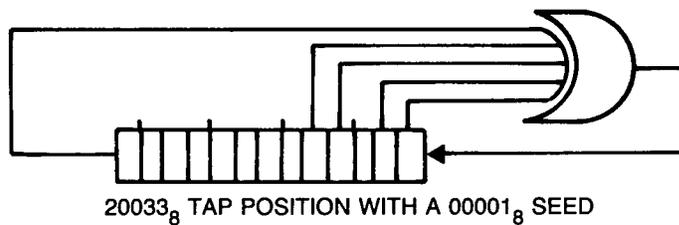


Fig. 7 — Random data-bit generator

Table 7 summarizes the library segment by a tabulation of the short titles of the subprocedures without elaboration, due to the number. Table 8 shows the software memory map assignments for each of the ROM, RAM, and I/O functions.

## IMPLEMENTATION AND OPERATION

Throughout this report, current implementation and operational features are discussed, especially in the application software section for the operation sequence; however, especially significant items are covered in this section. In particular, the *operator loaded* formats are not part of the current EPROM software load module, although the source code is written for a future software update and the hardware is configured to support the feature. Likewise, the random interrogation *Hold* and *Reset* features are not available in the software module. For the current control program, the sustained internal interrogation rates are 200 and 45 Hz for the fixed data maximum and random data interrogations, respectively. The external rates must be less than or equal to the internal rates. Typical input and output power levels of the 1030 MHz signal generator and the PSK modulated signal are +10 and 0 dBm, respectively.

The operation of the unit is reliable and has fulfilled the design goals of *user friendly* operation. The unit is used routinely as part of an instrumentation package in a series of flight tests on the NRL P-3 aircraft. For the current implementation, Fig. 8 shows a typical DABS long format interrogation produced by the unit. Figure 9 focuses on the initial 5  $\mu$ s, which includes the ATCRBS suppression pulses, P<sub>1</sub> and P<sub>2</sub>, and the first 1.25  $\mu$ s DPSK reference period. Figure 10 shows the detailed phase transition of the 1030 MHz carrier. Figures 11 and 12 are typical emission spectrums for random data long message format interrogation (differing MHz/division units), while Figs. 13 and 14 are for one of the fixed constant patterns (MSS=4).

Table 7 — Library Short Titles

Label	Procedure	Description
ANNLD:	PROCEDURE EXTERNAL;	/* --- Announce Load Operation --- */
LDACT:	PROCEDURE EXTERNAL;	/* --- Load Address Counter --- */
EADBSR:	PROCEDURE EXTERNAL;	/* --- Enable ADDBUS and Read --- */
ANNMSS:	PROCEDURE EXTERNAL;	/* --- Announce MSS Operation --- */
ANNLSM:	PROCEDURE EXTERNAL;	/* --- Announce Long/Short Operation --- */
NPAMLD:	PROCEDURE EXTERNAL;	/* --- Normal PAM Load --- */
PKPRLD:	PROCEDURE EXTERNAL;	/* --- PSK Preamble Load --- */
CKADCT:	PROCEDURE EXTERNAL;	/* --- Clock the Address Counter --- */
WBUF:	PROCEDURE EXTERNAL	(TEMP1, TEMP2, I, K);
	---- Write to hardware buffer with a software trick ----	*/
RBUF:	PROCEDURE EXTERNAL	(ATMP1, I, K);
	---- Read from hardware buffer with a software trick ----	*/
LDBUF:	PROCEDURE EXTERNAL (K);	--- Load the Hardware Buffers --- */
PSKLD:	PROCEDURE EXTERNAL;	--- Load the PSK Image Array --- */
RNSQLD:	PROCEDURE EXTERNAL;	--- Random Data Sequence Load --- */
PAMLD:	PROCEDURE EXTERNAL;	--- PAM Image Array Load by Operator
ZRND:	PROCEDURE EXTERNAL;	--- Zeroize Random Operation --- */
MSCOMP:	PROCEDURE EXTERNAL;	--- Memory Selection Complete --- */
ITDIS:	PROCEDURE EXTERNAL;	
START:	PROCEDURE INTERRUPT	0; /* --- Hardware Reset --- */
IPULSE:	PROCEDURE INTERRUPT	3; /* --- Interrogate Pulse --- */
IPINT:	PROCEDURE;	
ENRCNT:	PROCEDURE;	
MSTEST:	PROCEDURE;	/* --- Memory Section Switch Test --- */

Table 8 — Software Memory Map

Device	Address	Function
ROM	5000 — 5FFF	Control program
ROM	7000 — 7FFF	Control program
ROM	FC00 — FFFF	Modified FANTOM II
RAM 1	0000 — 007F	Reserved for PL/W
RAM 2	0080 — 00FF	PL/W and user pointers (0C — FF)
RAM 3	0100 — 01FF	User program
RAM 4	ED80 — EDFD	User and debug program
ACIA	EE08 EE99	Status/Control Register Data In/Out Register
PIA20	EE10 EE11 EE12 EE13	Data Register A Data Register B Control Register A Control Register B
PIA10	EE20 EE21 EE22 EE23	Data Register A Data Register B Control Register A Control Register B
PIA 1	EA00 EA01 EA02 EA03	Control Register A Control Register B Data Register A Data Register B
PIA 2	EA04 EA05 EA06 EA07	Control Register A Control Register B Data Register A Data Register B
PAI 3	EA08 EA09 EA0A EA0B	Control Register A Control Register B Data Register A Data Register B
PIA 4	EA0C EA0D EA0E EA0F	Control Register A Control Register B Data Register A Data Register B

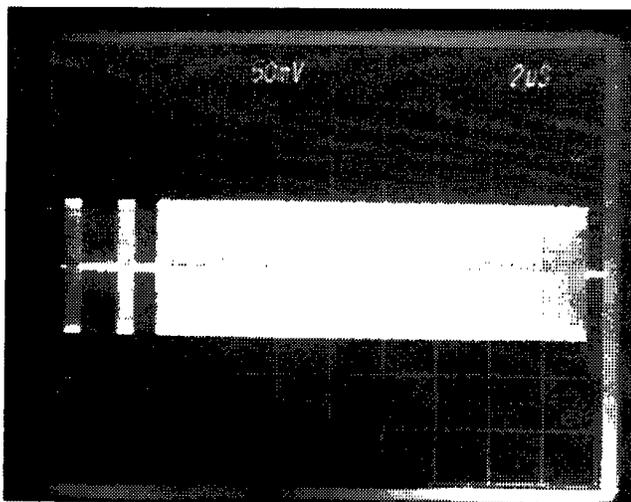


Fig. 8 — Typical DABS long format interrogation

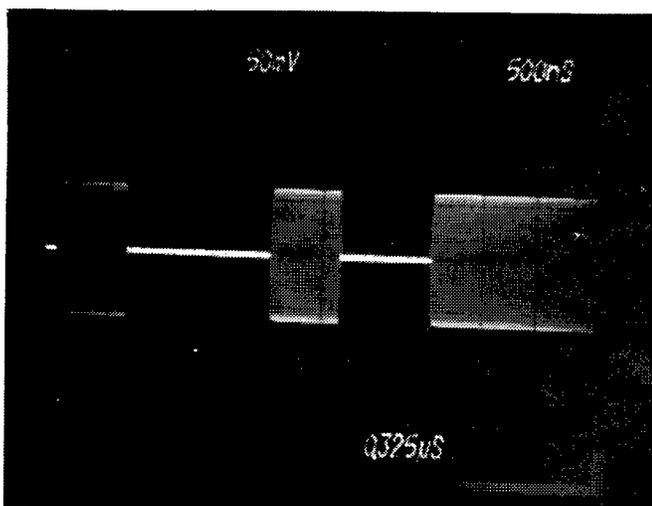


Fig. 9 — First 5  $\mu$ s of Fig. 8

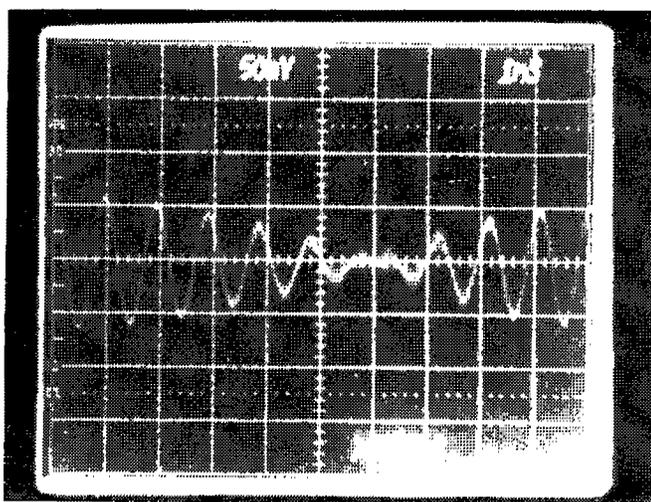


Fig. 10 — Phase transition detail

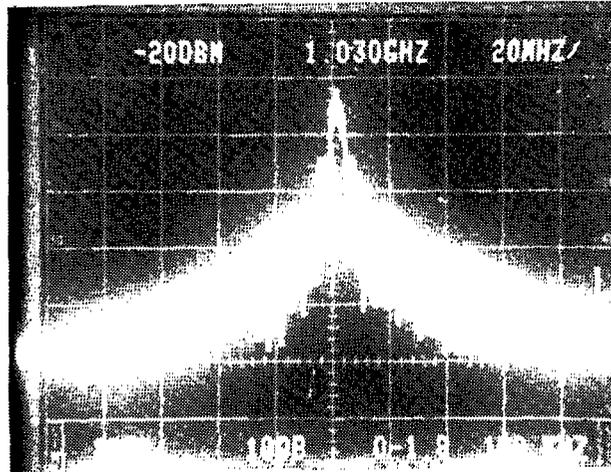


Fig. 11 — Transmission frequency spectrum data, random data, long format, 20 MHz/div

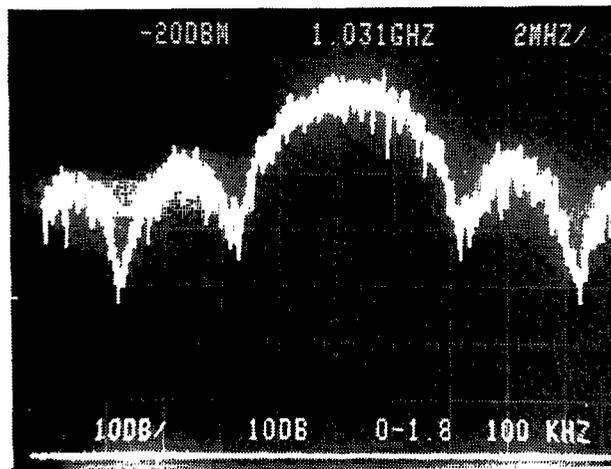


Fig. 12 — Transmission frequency spectrum data random message, long format, 2 MHz/div

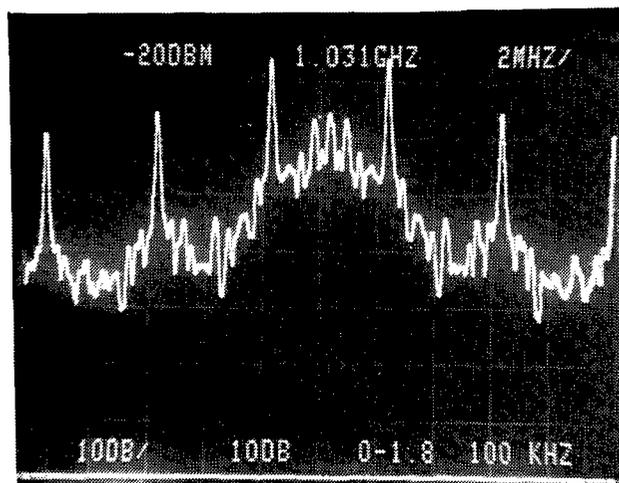


Fig. 13 — Transmission frequency spectrum data, constant message data, long format, 20 MHz/div

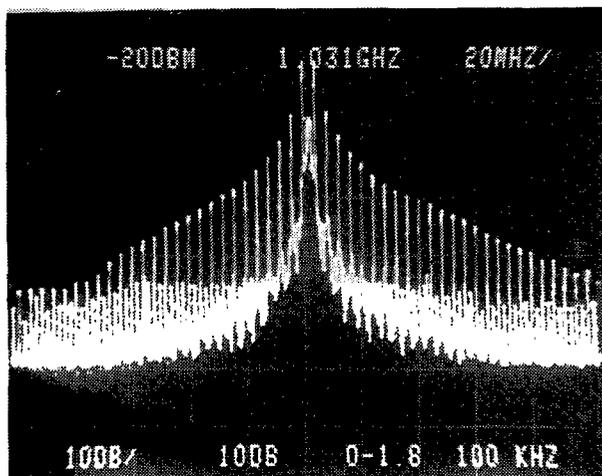


Fig. 14 — Transmission frequency spectrum data,  
constant message data, long format, 2 MH/div

## CONCLUSIONS

The operation of the unit is reliable and has fulfilled the design goals of *user friendly* operation. The unit is used routinely as part of an instrumentation package in a series of flight tests on the NRL P-3 aircraft.

## REFERENCES

1. U. A. Orlando and P. R. Drouilhet, "Discrete Address Beacon System," ATC-42, Revision A, MIT Lincoln Laboratory, April 1980.
2. *Properties of Linear Binary Encoding Sequences*, by Robert Gold Associates, 1 Aug 1971. Lecture Notes.

## **Appendix A**

### **LESSONS LEARNED**

#### **DEVELOPMENT AND DEBUGGING AIDS**

Any embedded microprocessor system development project is dependent on the availability and use of software and firmware development tools. For this case, a variety of development tools are used and spread between various places. The PL/W high-level compiler and the object code linker are maintained and supported on the NRL DEC-10 system. NRL also maintains a TEK 8002A microprocessor development system. The TEK 8002A includes in-circuit debugging and emulation, mainframe file transfer of load modules, and EPROM programming features.

The lack of an integrated microprocessor development system and software package with its own microprocessor high-level compiler, hardware emulation probe, and EPROM programming capability, severely impacts the time required for development. In this case, the software source is generated at a smart terminal and is uploaded to the DEC-10 for compilation and linking. The HEX format file is then downloaded to the TEK-8002A microprocessor emulator system. The file is manipulated to strip the extraneous filler lines and to format for TEK 8001 compatibility. The final debugged load module is then manipulated to format for the vendor EPROM programmer module. Finally, the load modules are downloaded to a vendor-supplied control module and EPROM programmer module configuration for the programming of individual EPROMs. As discussed in the next section of this appendix, one quarter of the development time is required to integrate dispersed development tools.

The debugging of both the hardware and software occurred at all levels: DEC-10, TEK 8002, and stand alone. Though not intuitively obvious, hardware errors were induced by the vendor-supplied, PL/W compiler errors maintained on the DEC-10. Software debugging process starts at the compiler, then at the linker, then continues with the execution of the load module in the TEK-8002A emulator. For this setup, complete software emulation is not possible because of the number of I/O involved. Thus the hybrid emulation mode (the emulator 6800 probe plugged into the control module, but using the emulator memory for program storage and execution) is the next level of software and hardware debugging operation. The next to the last step in the process is operating the DABS interrogation modulator with the control program loaded in the EPROM module and the emulator 6800 probe acting as a 6800 microprocessor. During this step, the emulator monitors the actions. The final step is operating stand alone with scopes and logic analyzer to monitor the actions and using the EPROM-stored FANTOM II debugger program to fine tune the software and hardware. FANTOM II is initiated by a hardware interrupt triggered by a front panel switch and is used because the TEK-8002A cannot execute 6800 instructions at the normal clock rate of 1 MHz. Instead it runs at one tenth the rate, thus making the detection of real-time errors difficult. Using the FANTOM II on the control module finishes the process of isolating the software and hardware errors.

#### **DESIGN AND DEVELOPMENT CYCLES EXAMPLE**

Finally, a significant lesson learned during the effort that other designers and managers should note is the percentage of time required for each critical design and development task. From a cold start, concept of unit determined, no experience with the microprocessor emulator system, forgotten experience with the DEC-10, forgotten experience with 6800 microprocessors, and no parts ordered, the process and allocation of time is:

- hardware design including selection of parts—2 months;
- model shop layout, parts delivery, and construction—5 months;
- in parallel with ESD, software control program source generated—2 months;
- learning curve for DEC-10/TEK 8002A—1 month; PL/W, compiler and linker—1 month;
- PL/W, compiler, linker, and compiler error detection—3 months;
- three iterations of load module and hardware/software debugging—2 months;
- final test—1 month

The project took 11 months to complete because 3 of those months were used to compensate for the lack of an integrated microprocessor development system and high-level language software package. However, without the software and hardware tools that were available and used, it is very hard to estimate the time and effort required to produce the unit, if at all.

**Appendix B**  
**PIA I/O ASSIGNMENTS**

PIA	PIN	LABEL	DIRECTION
PI(02H) EA02	S1A-01	LSB, LSD OF ADD.; LSB, LSD OF RAN. SEQ. CNT.	I/O
	-03	ADD.; RAN. SEQ. CNT.	I/O
	-05	ADD.; RAN. SEQ. CNT.	I/O
	-07	ADD.; RAN. SEQ. CNT.	I/O
	-09	ADD.; RAN. SEQ. CNT.	I/O
	-11	ADD.; RAN. SEQ. CNT.	I/O
	-13	ADD.; RAN. SEQ. CNT.	I/O
	-15	MSB, MSD OF ADD.; RAN. SEQ. CNT.	I/O
S1A-19	CA2	0	
PI(03H) EA03	S1B-01	LSB, MSD OF RANDOM SEQUENCE COUNT	0
	-03	RANDOM SEQUENCE COUNT	0
	-05	RANDOM SEQUENCE COUNT	0
	-07	MSB, MSD OF RANDOM SEQUENCE COUNT	0
	-09	<u>MSB OF RANDOM SEQUENCE COUNT</u>	0
	-11	<u>EXTERNAL INTERROGATE PULSE OVERRIDE (EXTOR)</u>	0
-13			
S1B-15			
PI(06H) EA06	S2A-01	LSB, LSD OF DATA BYTE ENTRY (DBE)	I
	-03	DATA BYTE ENTRY (DBE)	I
	-05	DATA BYTE ENTRY (DBE)	I
	-07	DATA BYTE ENTRY (DBE)	I
	-09	DATA BYTE ENTRY (DBE)	I
	-11	DATA BYTE ENTRY (DBE)	I
	-13	DATA BYTE ENTRY (DBE)	I
S2A-15	MSB, MSD OF DATA BYTE ENTRY (DBE)	I	
PI(07H) EA07	S2B-01	<u>LSB, LSD OF DATA BYTE DISPLAY (DBD)</u>	0
	-03	<u>DATA BYTE DISPLAY (DBD)</u>	0
	-05	<u>DATA BYTE DISPLAY (DBD)</u>	0
	-07	<u>DATA BYTE DISPLAY (DBD)</u>	0
	-09	<u>DATA BYTE DISPLAY (DBD)</u>	0
	-11	<u>DATA BYTE DISPLAY (DBD)</u>	0
	-13	<u>DATA BYTE DISPLAY (DBD)</u>	0
S2B-15	MSB, MSD OF <u>DATA BYTE DISPLAY (DBD)</u>	0	

PIA	PIN	LABEL	DIRECTION
	S3A-01	LSB <u>MEMORY SECTION SELECT (MSS)</u>	I
	-03	<u>MEMORY SECTION SELECT (MSS)</u>	I
	-05	<u>MEMORY SECTION SELECT (MSS)</u>	I
	-07	MSB <u>MEMORY SECTION SELECT (MSS)</u>	I
PI(OAH)	-09	ENVELOPE DATA / PSK DATA	I
EAOA	-11	LOAD / OPERATE	I
	-13	LONG / SHORT	I
	-15	INTERNAL / EXTERNAL Interrogate Pulse	I
	-17	CA1 ENTER MSS / DBE interrupt	I
	S3A-19	CA2 SKIP / OK interrupt	I
	S3B-01	ENTER DBE LED	0
	-03	ENVELOPE DATA LED	0
	-05	PSK DATA LED	0
	-07	RESET ADDRESS COUNTER LED	0
PI(OBH)	-09	LOAD of MEMORY COMPLETE LED	0
EAOB	-11	LOAD / OPERATE LED	0
	-13	LONG / SHORT LED	0
	S3B-15	SKIP / OK LED	0
	S4A-01		I
	-03		I
	-05		I
	-07		I
PI(OEH)	-09		I
EAOE	-11		I
	-13		I
	S4A-15		I
	S4B-01	RESET RANDOM SEQUENCE LED	0
	-03	HOLD RANDOM SEQUENCE LED	0
	-05	LSB MEMORY SECTION SELECT to ADD. CNT. CNTBUS	0
	-07	MEMORY SECTION SELECT to ADD. CNT. CNTBUS	0
PI(OFH)	-09	MSB MEMORY SECTION SELECT to ADD. CNT. CNTBUS	0
EAOF	-11	ENTER MSS LED	0
	-13	RANDOM SEQUENCE COUNT LED	0
	-15	ADDRESS of DATA LED	0
	-17		
	S4B-19		

WILLIAM A. BARNWELL

PIA	PIN		LABEL		DIRECTION
	P2-12	CA2			
	-08	CA1			
	-49	PA0			
	-47	PA1			
	-45	PA2			
PI10(0)	-43	PA3	DATA BUS A		I/O
EE20	-41	PA4			
	-39	PA5			
	-37	PA6			
	P2-35	PA7			
	P2-15	CB2	IPFIN	interrupt	I
	-17	CB			
	-33	PB0			
	-31	PB1			
	-29	PB2			
PI10(1)	-27	PB3	DATA BUS B		I/O
EE21	-25	PB4			
	-23	PB5			
	-21	PB6			
	P2-19	PB7			
	P2-14	CA2	Reset Address Counter	interrupt	I
	-10	CA1	Reset Random Sequence	interrupt	I
	-50	PA0	<u>PPMASL</u>		0
	-48	PA1	<u>PQASL</u>		0
	-46	PA2	PR/WA		0
PI20(0)	-44	PA3	<u>PENDA</u>		0
EE10	-42	PA4	<u>PENRA</u>		0
	-40	PA5	<u>PENABLE</u>		0
	-38	PA6	PCLKAD		0
	P2-36	PA7	PLDAD		0
	P2-16	CB2	Interrogate Pulse	interrupt	I
	-18	CB1	Hold Random Sequence	interrupt	I
	-34	PB0	<u>PPMBSL</u>		0
	-32	PB1	<u>PQBSL</u>		0
	-30	PB2	PR/WB		0
PI20(1)	-28	PB3	<u>PENDB</u>		0
EE11	-26	PB4	<u>PENRB</u>		0
	-24	PB5	<u>PCLRAD</u>		0
	-22	PB6	PENADCT		0
	P2-20	PB7	PIP/OR		0

## Appendix C

### NDABL B PROGRAM LISTING

```

/*$+A$+G$-I$+L$+P*/
/*
/* NDABLIB MODULE VERS 1.3, ORIG DATE 1/27/81 , VERS DATE 4/22/81
/*
/*
TIME:    EXTERNAL(TEMP3) ADDRESS;    /* IDENTIFY TIME TO LINKER
/*
/*
DECLARE ( ACI AT OEE08H, PI10 AT OEE20H ) BYTE;
DECLARE ( PI20 AT OEE10H, PI AT OEAOOH ) BYTE;
DECLARE INITAC BYTE PUBLIC;
DECLARE (BLINK,MSLDMA,MSLDMK,RANCNT,SEED) ADDRESS PUBLIC;
DECLARE (ATMP1,ATMP2,ATMP3) ADDRESS PUBLIC;
DECLARE ( TEMP1,TEMP2,TEMP3,TEMP4,TEMP5 ) BYTE PUBLIC;
DECLARE (PAMIM,PSKIM,CODEIM) (18) BYTE PUBLIC;
DECLARE (IPCOM,ERR,MSSVAL,ADDIMA) BYTE PUBLIC;
DECLARE (CNTBUS,LASTMS,LSMSG) BYTE PUBLIC;
DECLARE ZRO LIT '00H',BIT0 LIT '01H',BIT1 LIT '02H';
DECLARE BIT2 LIT '04H',BIT3 LIT '08H',BIT4 LIT '10H';
DECLARE BIT5 LIT '20H',BIT6 LIT '40H',BIT7 LIT '80H';
DECLARE ( PIA1A,PIA1B,PIA2A,PIA2B,PIA3A,PIA3B ) BYTE PUBLIC;
DECLARE (PIA4A,PIA4B,PIA10A,PIA10B,PIA20A,PIA20B) BYTE PUBLIC;
DECLARE ( K,I,ACDDR,CLDDR,DIRO,DIRI ) BYTE PUBLIC;
DECLARE IPFLG BYTE PUBLIC;
/*
/*      ----- Start of Library Segment -----
/*
/*
ANNLD:  PROCEDURE EXTERNAL; /* --- Announce Load Operation ---
/*
/*
/*
DECLARE (ANL1,ANL2) LABEL;
BLINK = 4000;
ANL2:   TEMP1 = PI(OAH);           /* READ PIA3A
IF (BIT5 AND TEMP1) = 00H THEN GO TO ANL1; /* LD=0, OP=1
TEMP2 = PIA3B;                   /* READ PIA3B OUTPUT IMAGE
TEMP2 = ( TEMP2 AND OFFH ) OR BIT5; /* LOAD LED ON
PI(OBH) = TEMP2;                 /* WRITE TO PIA3B
PIA3B = TEMP2;                   /* WRITE IMAGE OF PIA3B
ATMP1 = BLINK;
CALL TIME( ATMP1 );              /* DELAY .4 SEC.
TEMP1 = PI(OAH);
IF (BIT5 AND TEMP1) = 00H THEN GO TO ANL1; /* LD=0, OP=1
TEMP2 = ( ( NOT BIT5 ) AND TEMP2 );
PI(OBH) = TEMP2;                 /* LOAD LED OFF
PIA3B = TEMP2;                   /* WRITE IMAGE OF PIA3B
ATMP1 = BLINK;
CALL TIME( ATMP1 );              /* DELAY .4 SEC.
GO TO ANL2;

```

WILLIAM A. BARNWELL

```
ANL1:  TEMP2 = PIA3B;
        TEMP2 = ( TEMP2 AND 0FFH ) OR BIT5;
        PI(0BH) = TEMP2;          /* LEAVE LOAD LED ON      */
        PIA3B = TEMP2;          /* WRITE IMAGE OF PIA3B  */
```

```

LDADCT: PROCEDURE EXTERNAL; /* ----- Load Address Counter ----- */
/* */
/* */
/* */
/* */
TEMP1 = CNTBUS*4;
TEMP2 = PIA4B; /* READ IMAGE OF PIA4B */
TEMP2 = (PIA4B AND 11100011B) OR TEMP1;
PIA4B = TEMP2;
PI(0FH) = TEMP2;
TEMP4=PIA20A OR BIT7; /* READ IMAGE OF PIA20A */
PI20(0)=TEMP4; /* SET PLDAD HIGH */
TEMP4=(TEMP4 AND (NOT BIT6)); /* LOAD HIGH AND CLK LOW */
PI20(0)=TEMP4; /* PCLKAD SET LOW */
TEMP4=(TEMP4 AND (NOT BIT7)); /* PLDAD-LOW, PCLKAD-LOW */
PI20(0)=TEMP4;
TEMP4=(TEMP4 OR BIT6); /* PCLKAD SET HIGH */
PI20(0)=TEMP4;
PIA20A=TEMP4; /* WRITE IMAGE OF PI20A */
RETURN;
END;
/* */
/* */
EADBSR: PROCEDURE EXTERNAL; /* --- Enable ADBBUS and Read --- */
/* */
/* */
TEMP4=PI(0); /* READ PIA1 CONTROL REG A */
TEMP4=(TEMP4 AND (NOT BIT2)); /* ACCESS DDR */
PI(0)=TEMP4;
PI(2)=0; /* SET AS INPUTS */
PIA1A = 0;
TEMP4=(TEMP4 OR BIT2 ); /* SET OUTPUT REG */
PI(0)=(TEMP4 OR BIT3); /* ENABLE ADBBUS */
PI(3)=00H; /* WRITE 0 TO HIGH ORDER ADD/RAN DISPLAY */
PIA1B = 00H;
TEMP4 = PIA4B; /* READ IMAGE OF PIA4B DATA REG. */
TEMP4 = ( TEMP4 AND 0BFH ) OR BIT7;
PI(0FH) = TEMP4;
PIA4B = TEMP4;
RETURN;
END;
/* */
/* */
ANNMSS: PROCEDURE EXTERNAL; /* --- Announce MSS Operation --- */
/* */
/* */
DECLARE (ANL3,ANL4) LABEL;
TEMP1=PI(0AH); /* PIA3A DATA REGISTER */
TEMP4=PI(08H); /* PIA3A CONTROL REGISTER */
TEMP4=(TEMP4 OR BIT0); /* ENABLE CA1 */
PI(08H)=TEMP4;
ANL4: TEMP2=PIA4B; /* READ PIA4B OUTPUT IMAGE */
TEMP2 = (TEMP2 AND 0FFH) OR BIT5;
PI(0FH)=TEMP2; /* SET MSSLED ON */
ATMP1=BLINK;
CALL TIME (ATMP1); /* DELAY 0.4 SECOND */

```

WILLIAM A. BARNWELL

```

TEMP5=PI(08H);          /* READ PIA3A CONTROL REGISTER */
IF (TEMP5 AND BIT7)=80H THEN GO TO ANL3;
PIA4B=TEMP2;           /* WRITE IMAGE OF PIA4B      */

TEMP2=((NOT BIT5) AND TEMP2);
PI(0FH)=TEMP2;        /* MSSLED OFF                */
ATMP1=BLINK;
CALL TIME (ATMP1);
PIA4B=TEMP2;          /* WRITE IMAGE                */
TEMP5=PI(08H);        /* READ PIA3A CONTROL REG    */
IF (TEMP5 AND BIT7)=80H THEN GO TO ANL3;
GO TO ANL4;

ANL3:  TEMP5=(TEMP5 AND 3EH); /* DISABLE CA1,CA2          */
MSSVAL=(NOT(PI(0AH))) AND 0FH; /* CLEAR IRQA1 AND READ MSS VALUE */
PI(08H)=TEMP5;
TEMP2=(PIA4B AND (NOT BIT5));
PI(0FH)=TEMP2;        /* SET MSSLED OFF            */
PIA4B=TEMP2;
ADDIMA=(MSSVAL AND 07H)*32; /* FORM MSS IMAGE            */
CNTBUS = MSSVAL AND 07H;
CALL LDADCT;
CALL EADBSR;
TEMP1=PI(2);
ERR=0;
IF TEMP1<> ADDIMA THEN ERR=0FFH;
RETURN;
END;
/*
/*
ANNLSM: PROCEDURE EXTERNAL; /* -- Announce Long/Short Operation -- */
/*
/*
DECLARE (ANL1,ANL2,ANL3) LABEL;
TEMP2=PI(0AH);
TEMP4=PI(08H) OR BIT3; /* CA2 ENABLED WITH BIT3 SET */
PI(08H)=TEMP4;        /* ENABLE SKIP/OK INTERRUPT */
ATMP1=BLINK;
ANL2:  TEMP1=PIA3B OR BIT6 OR BIT7;
PI(0BH)=TEMP1;        /* SET LED'S ON              */
CALL TIME (ATMP1);    /* DELAY .4 SECOND           */
PIA3B=TEMP1;          /* WRITE IMAGE OF PI(0BH)    */
IF (PI(08H)AND BIT6)=BIT6 THEN GO TO ANL1;
TEMP1=PIA3B AND (NOT(BIT6 OR BIT7));
PI(0BH)=TEMP1;        /* LED'S OFF                  */
CALL TIME (ATMP1);
PIA3B=TEMP1;          /* UPDATE PIA3B IMAGE        */
IF (PI(08H)AND BIT6)=BIT6 THEN GO TO ANL1;
GO TO ANL2;
ANL1:  LSMSG=PI(0AH) AND BIT6; /* LSMSG=0 => LONG          */
PI(08H)=PI(08H) AND (NOT BIT3); /* DISABLE SKIP/OK          */
IF (PIA3B AND (NOT(BIT6 OR BIT7)))=0 THEN GO TO ANL3;
PI(0BH)=PI(0BH) AND 3FH; /* SET LED'S OFF            */
PIA3B=PIA3B AND 3FH; /* WRITE IMAGE                */
ANL3:  RETURN;
END;
/*

```

```

NPAMLD: PROCEDURE EXTERNAL; /* ----- Normal PAM Load ----- */
/* */
/* */
/* */
/*
  DECLARE (ANL1, ANL2) LABEL;
  I = 0;
  TEMP1 = LSMSG;
  PAMIM(0) = 00000111B;
  PAMIM(1) = 11000111B;
  PAMIM(2) = 11111111B;
  I = I + 3;
  IF (TEMP1=BIT6) THEN GO TO ANL1;
  DO WHILE I<= (17-2);
/*---  RECALL '0' INDEX AND LAST BYTE SET OUTSIDE OF DO LOOP  ----*/
  PAMIM(I) = 11111111B;
  I = I + 1;
  END;
  PAMIM(16) = 01111111B;
  GO TO ANL2;
ANL1:  DO WHILE I<=(10-2);
  PAMIM(I) = 11111111B;
  I = I + 1;
  END;
  PAMIM(9) = 01111111B;
  I = I + 1;
  DO WHILE I<= 16;
  PAMIM(I) = 0;
  I = I + 1;
  END;
ANL2:  RETURN;
  END;
/* */
/* */
PKPRLD: PROCEDURE EXTERNAL; /* ----- PSK Preamble Load ----- */
/* */
/* */
/* */
  PSKIM(0) = 00000000B;
  PSKIM(1) = 00000000B;
  PSKIM(2) = (( PSKIM(2) AND 0EOH) OR 08H);
  RETURN;
  END;
/* */
/* */
CKADCT: PROCEDURE EXTERNAL; /* --- Clock the Address Counter --- */
/* */
/* */
/* */
  DECLARE (ANL1, ANL2) LABEL;
  IF ((PIA20A AND BIT6) = 0 ) THEN GO TO ANL1;
  GO TO ANL2;
ANL1:  PIA20A = PIA20A OR BIT6; /* RETURN 'PCLKAD' HIGH */
  PI20(0) = PIA20A;
ANL2:  PIA20A = (PIA20A AND (NOT BIT6)); /* 'PCLKAD' LOW */
  PI20(0) = PIA20A;
  PIA20A = PIA20A OR BIT6; /* RETURN 'PCLKAD' HIGH */
  PI20(0) = PIA20A;

```

WILLIAM A. BARNWELL

```

END;
/*                                                                    */
    RETURN;

/*                                                                    */
WBUF:  PROCEDURE EXTERNAL (TEMP1, TEMP2, I, K);
/*                                                                    */
/* ----- Write to Hardware Buffers -----                          */
/*                with software trick                                  */
/*                                                                    */
    DECLARE ( TEMP1, TEMP2,K,I) BYTE;
    DECLARE (ANL1, ANL2, ANL3, ANL4) LABEL;
    K=K AND 01H;                /* K=0 ==> PAM ; K=1 ==> PSK */
    I = I AND 00000001B;        /* INDEX OF 0 OR 1 ALLOWED */
    PIA20A(I) = (PIA20A(I) AND 0E0H) OR 17H;
    IF K=0 THEN PIA20A(I)=PIA20A(I) AND 0EFH;
    PI20(I) = PIA20A(I);        /* DISABLES PAM_ & PSK_ */
    IF (TEMP2 = 0FFH) THEN GO TO ANL1;
ANL2:  PIA10A(I) = TEMP1;
    PI10(I) = TEMP1;           /* DATA TO DATA BUS */
    IF K=1 THEN PIA20A(I)=(PIA20A(I) AND 11111001B);
        ELSE PIA20A(I)=(PIA20A(I) AND 11111010B);
    PI20(I) = PIA20A(I);
    PIA20A(I) = PIA20A(I) OR 00000100B;
    PI20(I) = PIA20A(I);
    IF (TEMP2 = 0FFH) THEN GO TO ANL3;
    PIA20A(I) = PIA20A(I) AND 11110110B;    /* PAM_ IS SELECTED */
    PI20(I) = PIA20A(I);
ANL4:  RETURN;
ANL1:  PI10(2+I) = PI10(2+I) AND 1111011B;
/*--- DATA DIRECTION SELECTION ---
    PI10(I) = 0FFH;           /* CHANGE TO OUTPUTS */
    PI10(I+2)=PI10(I+2) OR 00000100B;     /* OUTPUT REG SEL */
    GO TO ANL2;
ANL3:  PI10(2+I) = PI10(2+I) AND 1111011B;
/*--- DATA DIRECTION SELECTION ---
    PI10(I) = 0;              /* CHANGE TO INPUTS */
    PI10(2+I) = PI10(2+I) OR 00000100B; /* OUTPUT REGISTER SELECT */
    IF K=0 THEN PIA20A(I)=(PIA20A(I) OR 18H);
    IF K=0 THEN PI20(I)=PIA20A(I);
    PIA2B=PI10(I);
    IF (PIA2B<>TEMP1) THEN ERR=0FFH;
    PI(07H)=PIA2B;
    PI20(I)=PIA20A(I) AND 0F4H;
    PI20(I) = PIA20A(I);
    GO TO ANL4;
END;
/*                                                                    */
/*                                                                    */
RBUF:  PROCEDURE EXTERNAL (ATMP1,I,K);
/*                                                                    */
/* ----- Read From Hardware Buffers -----                          */
/*                with a software trick                                  */
/*                                                                    */
    DECLARE ATMP1 ADDRESS;
    DECLARE (TEMP1 BASED ATMP1,K,I) BYTE;

```

```

I = I AND 01H;
PIA20A(I) = (PIA20A(I) AND 0EOH) OR 14H;
K=K AND 01H;                               /* K=0 ==> PAM; K=1 ==> PSK */

IF K=1 THEN PIA20A(I)=PIA20A(I) OR 01H;
ELSE PIA20A(I)=PIA20A(I) OR 00AH;
PI20(I) = PIA20A(I);
TEMP1 = PI10(I);
/*--- ISOLATE BUS, SELECT PSK AND PAM --- */
PIA20A(I) = (PIA20A(I) AND 0EOH) OR 14H;
PI20(I) = PIA20A(I);
RETURN;
END;
/* */
/* */
LDBUF:   PROCEDURE EXTERNAL(K);
/* */
/* ----- Load the Hardware Buffers ----- */
/* */
DECLARE (K,I,TEMP1,TEMP2) BYTE;
DECLARE ANL1 LABEL;
K=K AND 01H;
CNTBUS=MSSVAL AND 07H;
ERR=0;
CALL LDADCT;
CALL EADBSR;
I=0+(18*K);
TEMP1=PAMIM(I);
TEMP2=OFFH;
CALL WBUF(TEMP1,TEMP2,I,K);
IF ERR<>0 THEN GO TO ANL1;
I=I+1;
CALL CKADCT;
DO WHILE I<=(16+(K*18));
  TEMP1=PAMIM(I);
  CALL WBUF(TEMP1,TEMP2,I,K);
  I=I+1;
  CALL CKADCT;
  IF ERR<>0 THEN GO TO ANL1;
END;
ANL1:   RETURN;
END;
/* */
/* */
PSKLD:  PROCEDURE EXTERNAL;
/* */
/* ----- Load the PSK Image Array ----- */
/* */
I=3;
DO CASE (MSSVAL AND 07H);
DO;    /* --- MSS Set to 0 Case Start --- */
  I=3;
  PSKIM(2)=08H;
  DO WHILE I<=16;
    PSKIM(I)=00H;
    I=I+1;

```

WILLIAM A. BARNWELL

```

RANCNT=0;
SEED=1;
END;

END;
DO;      /* --- MSS Set to 1 Case Start --- */
  I=3;
  PSKIM(2)=88H;
  DO WHILE I<=16;
    PSKIM(I)=99H;
    I=I+1;
  END;
END;
DO;      /* --- MSS Set to 2 Case Start --- */
  I=2;
  DO WHILE I<=5;
    PSKIM(I)=82H;
    PSKIM(I+5)=00H;
    PSKIM(I+10)=82H;
    I=I+1;
  END;
  PSKIM(6)=00H;
  PSKIM(11)=00H;
  PSKIM(16)=00H;
END;
DO;      /* --- MSS Set to 3 Case Start --- */
  I=3;
  PSKIM(2)=68H;
  DO WHILE I<=16;
    PSKIM(I)=0DBH;
    PSKIM(I+1)=0B6H;
    PSKIM(I+2)=06DH;
    I=I+3;
  END;
  PSKIM(16)=036H;
END;
DO;      /* --- MSS Set to 4 Case Start --- */
  I=3;
  PSKIM(2)=0E8H;
  DO WHILE I<=16;
    PSKIM(I)=OFFH;
    I=I+1;
  END;
END;
DO;      /* --- MSS Set to 5 Case Start --- */
  I=3;
  PSKIM(2)=0E8H;
  DO WHILE I<=16;
    PSKIM(I)=OFFH;
    PSKIM(I+1)=00H;
    I=I+2;
  END;
END;
;      /* --- MSS Set to 6 Case Start --- */
;      /* --- MSS Set to 7 Case Start --- */
END;      /* END OF DO CASE */

```

```

IF (LSMSG=BIT6) THEN PSKIM(9)=PSKIM(9) AND 1FH;
RETURN;
PSKIM(16)=PSKIM(16) AND 1FH;

END;
/*
/*
RNSQLD: PROCEDURE EXTERNAL;
/*
/* ----- Random Data Sequence Load -----
/*
/*
PSKIM(2)=PSKIM(2) AND 0COH;
PSKIM(16)=PSKIM(16) AND 0FH;
ATMP2=SEED AND 1FFFH; /* 13 BIT SEED USED. SEED <> 0 TO START */
TEMP3=(HIGH(ATMP2))/16;
TEMP2=(ATMP2 AND BIT0) + TEMP3;
TEMP2=(SHR(ATMP2,2) AND BIT0) + (SHR(ATMP2,3) AND BIT0) + TEMP2;
ATMP3= TEMP2 AND BIT0;
SEED=(SEED*2) + ATMP3;
ATMP3=SHL(ATMP3,6); /* ADJUSTS FOR INITIAL DATA OFFSET */
I=2;
IF (LSMSG=BIT6) THEN TEMP4=9;
ELSE TEMP4=16;
DO WHILE I<=TEMP4;
ATMP2=DOUBLE(PSKIM(I));
ATMP2=(ATMP2*2) + ATMP3;
PSKIM(I)=LOW(ATMP2);
ATMP3=HIGH(ATMP2);
I=I+1;
END;
RANCNT=RANCNT + 1;
PSKIM(2)=PSKIM(2) OR 08H;
PSKIM(16)=PSKIM(16) AND 1FH;
IF (LSMSG=BIT6) THEN PSKIM(9)=PSKIM(9) AND 1FH;
RETURN;
END;
/*
/*
PAMLD: PROCEDURE EXTERNAL;
/*
/* ----- PAM Image Array Load by Operator -----
/*
/* Not implemented in this version
/*
/*
CALL NPAMLD; /* ---- Normal PAM Load ----
RETURN;
END;
/*
/*
ZRND: PROCEDURE EXTERNAL; /* ---- Zeroize Random Operation ----
/*
/* Zeroize Initial Random Data and Counter
/*
/*
DO I=0 TO 35;
PSKIM(I)=0;
END;
SEED=1;

```

WILLIAM A. BARNWELL

```

PI(03H)=PIA1B;
RANCNT=0;
PIA1B=PIA1B AND (NOT BIT5);

RETURN;
END;
/* */
/* */
MSCOMP: PROCEDURE EXTERNAL; /* ---- Memory Selection Complete ---- */
/* */
/* */
CNTBUS=(NOT(PI(0AH))) AND 07H;
MSSVAL=CNTBUS;
CALL LDADCT;
CALL EADBSR;
PIA4B=(PIA4B AND 9CH) OR 80H;
PI(0FH)=PIA4B;
PIA3B=(PIA3B AND 70H) OR 10H;
PI(0BH)=PIA3B;
RETURN;
END;
/* */
/* */
ITDIS: PROCEDURE EXTERNAL;
/* */
/* */
/* */
I=0;
DO WHILE I<=15;
  DECLARE ANL1 LABEL;
  PI(I)=PI(I) AND 0FCH;
  PI(I+1)=PI(I+1) AND 0FCH;
  ATMP2=PI(I+2) + PI(I+3);
  IF (I>4) THEN GO TO ANL1;
  PI10(2)=PI10(2) AND 0FCH;
  PI10(3)=PI10(3) AND 0FCH;
  PI20(2)=PI20(2) AND 0FCH;
  PI20(3)=PI20(3) AND 0FCH;
  ATMP2=PI10(0) + PI10(1);
  ATMP2=PI20(0) + PI20(1);
ANL1: I=I+4;
END;
RETURN;
END;
/* */
EOF

```

## Appendix D

### NDAB PROGRAM LISTING

```

/*$+A$+G$-I$+L$+P*/
/*
/* NDAB MODULE VERS 1.6, ORIG DATE 1/27/81, VERS DATE 4/16/81 */
/* ( Append NDAB to NDABL B for Compilation ) */
TIME: EXTERNAL(TEMP3) ADDRESS; /* IDENTIFY TIME TO LINKER */
/*
ANNLD: EXTERNAL;
LDADCT: EXTERNAL;
EADBSR: EXTERNAL;
ANNMSS: EXTERNAL;
ANNLSM: EXTERNAL;
NPAMLD: EXTERNAL;
PKPRLD: EXTERNAL;
CKADCT: EXTERNAL;
WBUF: EXTERNAL (TEMP1, TEMP2, I, K);
RBUF: EXTERNAL (ATMP1, I, K);
LDBUF: EXTERNAL (K);
PSKLD: EXTERNAL;
RNSQLD: EXTERNAL;
PAMLD: EXTERNAL;
ZRND: EXTERNAL;
MSCOMP: EXTERNAL;
ITDIS: EXTERNAL;
    DECLARE MAIN LABEL;
    DECLARE ( ACI AT OEE08H, PI10 AT OEE20H ) BYTE;
    DECLARE ( PI20 AT OEE10H, PI AT OEA00H ) BYTE;
    DECLARE INITAC BYTE EXTERNAL;
    DECLARE (BLINK,MSLDMA,MSLDMK,RANCNT,SEED) ADDRESS EXTERNAL;
    DECLARE (ATMP1,ATMP2,ATMP3 ) ADDRESS EXTERNAL;
    DECLARE ( TEMP1,TEMP2,TEMP3,TEMP4,TEMP5 ) BYTE EXTERNAL;
    DECLARE (PAMIM,PSKIM,CODEIM) (18) BYTE EXTERNAL;
    DECLARE (IPCOM,ERR,MSSVAL,ADDIMA) BYTE EXTERNAL;
    DECLARE (CNTBUS,LASTMS,LSMSG) BYTE EXTERNAL;
    DECLARE ZRO LIT '00H',BIT0 LIT '01H',BIT1 LIT '02H';
    DECLARE BIT2 LIT '04H',BIT3 LIT '08H',BIT4 LIT '10H';
    DECLARE BIT5 LIT '20H',BIT6 LIT '40H',BIT7 LIT '80H';
    DECLARE ( PIA1A,PIA1B,PIA2A,PIA2B,PIA3A,PIA3B ) BYTE EXTERNAL;
    DECLARE ( PIA4A,PIA4B,PIA10A,PIA10B,PIA20A,PIA20B )BYTE EXTERNAL;
    DECLARE ( K,I,ACDDR,CLDDR,DIRO,DIRI ) BYTE EXTERNAL;
    DECLARE IPFLG BYTE EXTERNAL;
    DECLARE (ANL1,ANL2,ANL3,ANL4,ANL5,ANL6,ANL7,ANL8,ANL9) LABEL;
    GO TO MAIN;
/*
/*

```

WILLIAM A. BARNWELL

```
START:  PROCEDURE INTERRUPT 0; /* ----- Hardware Reset ----- */
/*
/*   THIS PROCEDURE IS THE 'RESET' INTERRUPT ROUTINE, IT
/* SETS MSLDMA FLAG TO ZERO TO SIGNAL NEED TO REINITIALIZE
/* ALL OF THE BUFFERS.
/*
      MSLDMA=0;
      GO TO MAIN;
END;
```

```

/*
/*
IPULSE: PROCEDURE INTERRUPT 3; /* ----- Interrogate Pulse -----
/*
/*
/* THIS INTERRUPT PROCEDURE HANDLES THE 'IP' AND THE 'IPFIN'
/* INTERRUPTS AT THIS TIME. THE RANDOM SEQUENCE HOLD AND THE
/* RANDOM SEQUENCE RESET INTERRUPTS WILL BE IMPLEMENTED
/* LATER.
/*
/* THE ROUTINE SETS THE EXTOR, DISABLES INTERRUPTS, SETS
/* IPFLG, CLEARS THE 'IP' INTERRUPT, DISABLES IT, AND THEN
/* WAITS FOR 'IPFIN'. THEN THE ROUTINE CLEARS THE 'IPFIN'
/* INTERRUPT, DISABLES IT, SETS IPCOM AND RETURNS.
/*
DECLARE ANL2 LABEL;
DECLARE TEMP1 BYTE;
DISABLE;
PIA1B=PIA1B AND (NOT BIT5);
PI(03H)=PIA1B;
IPFLG=OFFH;
PI20(03H)=PI20(03H) AND OF4H;
TEMP1=PI20(01H);
ANL2: IPCOM=0;
TEMP1=PI10(03H);
PI10(03H)=PI10(03H) AND OF4H;
TEMP1=PI10(01H);
IPCOM=OFFH;
RETURN;
END;
/*
/*
IPINT: PROCEDURE;
/*
/*
/* THIS ROUTINE TESTS THE 'INTERNAL/EXTERNAL INTERROGATE
/* PULSE' TOGGLE. IF THE POSITION IS SET FOR EXTERNAL IP, THE
/* 'IP' INTERRUPT IS CLEARED, SET UP, IPFLG AND IPCOM ARE
/* CLEARED, INTERRUPTS ARE ENABLED, AND FINALLY THE EXTERNAL
/* OVERRIDE (EXTOR) IS CLEARED (NEG. TRUE). IF THE POSITION IS
/* SET FOR INTERNAL IP, THEN THE IP INTERRUPT IS CLEARED, SET-
/* UP, IPFLG AND IPCOM ARE CLEARED, EXTOR IS SET (0), INTER-
/* RUPTS ARE ENABLED, AND THEN 'PIP/OR' IS CYCLED.
/*
DECLARE ANL1 LABEL;
TEMP1=PI20(0)+PI20(1)+PI10(0)+PI10(1); /* CLEAR IP AND IPFIN */
PI20(03H)=(PI20(03H) AND 1FH) OR 1CH;
PI10(03H)=(PI10(03H) AND OF4H) OR 18H;
IPFLG=0;
IPCOM=0;
ENABLE;
PIA20B=OF4H;
PI20(1)=PIA20B;
PIA20A=054H;
PI20(0)=PIA20A;

```

WILLIAM A. BARNWELL

```

    IF ((PI(OAH) AND BIT7)=0) THEN GO TO ANL1;
/*--- INTERNAL INTERROGATE PULSE SECTION --- */
    PIA3B=PIA3B AND 1FH;
    PI(03H)=PIA3B;
/*--- CYCLE PIP/OR --- */
    PI20(1)=PI20(1) AND 7FH;
    PI20(1)=PI20(1) OR 80H;
    RETURN;
ANL1:  PIA3B=PIA3B OR 20H;
    PI(03H)=PIA3B;
    RETURN;
END;
/* */
/* */
ENRCNT: PROCEDURE;
/* */
/* ----- Enable Random Message Counter ----- */
/* */
/* */
    DECLARE ATMP1 ADDRESS;
    PI(0H)=PI(0) AND 33H;
    PI(2)=OFFH;
    PI(0)=34H;
    ATMP1=RANCNT;
    PIA1A=LOW(ATMP1);
    PI(2)=PIA1A;
    PIA1B=((HIGH(ATMP1)) AND 01FH) OR 20H;
    PI(3)=PIA1B;
    PIA4B=(PI(0FH) AND 3FH) OR 40H;
    PI(0FH)=PIA4B;
    RETURN;
END;
/* */
/* */
MSTEST: PROCEDURE;
/* */
/* ----- Memory Section Switch Test ----- */
/* and Execute Selection */
/* */
/* */
    DECLARE ANL1 LABEL;
    TEMP1=((NOT(PI(OAH))) AND 07H);
    TEMP2=SHL(BITO,TEMP1);
    IF ((MSLDMA AND TEMP2)=0) THEN GO TO ANL1;
    IF CNTBUS=TEMP1 THEN GO TO ANL1;
    IF (TEMP1=0) THEN CALL ZRND; /* NEW MSS= RANDOM SEQ. */
    IF ((TEMP1<>0) AND (CNTBUS=0)) THEN CALL EADBSR;
    CNTBUS=TEMP1;
    MSSVAL=TEMP1;
    TEMP2=PIA4B;
    TEMP2=(TEMP2 AND 11100011B) OR (TEMP1*4);
    PIA4B=TEMP2;
    PI(0FH)=TEMP2;
ANL1:  RETURN;
END;
/* */
/* ----- End of Library ----- */
/* */

```

```

/*                                                                    */
MAIN:  STACKPTR=01BCH;                                               */
/* ----- Start of Initialization -----                            */
/*                                                                    */
  DISABLE;                                                           /* DISABLE ALL MASKABLE INTERRUPTS */
  INITAC = 01000001B;                                               /* 0$10$000$01B                    */
  ACI(0) = INITAC;                                                 /* WRITE TO ACIA CONTROL REGISTER  */
  I = 0;
  ACDDR = 0;                                                         /* ACCESS DATA DIRECTION REGISTER; B2=0 */
  DIRO = OFFH;                                                      /* ALL OUTPUTS                       */
  DIRI = 0;                                                          /* ALL INPUTS                         */
/* ----- Set PIA's direction -----                                */
DO WHILE I<=0CH;
  PI(I) = ACDDR;
  PI(I+1) = ACDDR;
  IF (I+2) = 2 THEN PI(I+2) = DIRO;
  ELSE PI(I+2) = DIRI;
  IF (I=0) THEN PI(I)=34H;
  ELSE PI(I)=3CH;
  PI(I+3) = DIRO;
  PI(I+1) = 34H;
  I = I+4;
END;
I = 0;
DO WHILE I <= 1;
  PI10(I+2) = ACDDR;
  PI20(I+2) = ACDDR;
  PI10(I) = DIRI;
  PI20(I) = DIRO;
  PI10(I+2) = 04H;
  PI20(I+2) = 04H;
  I = I+1;
END;
/* ----- Set Initial Output Data -----                            */
PI(8) = 04H;
PI(2) = 10001000B;                                                 /* ADDRESS OF DATA/RAN. SEQ.COUNT */
PI(3) = 00001000B;                                                 /* ADD OF DATA/RAN. SEQ. COUNT    */
PI(7) = 10001000B;                                                 /* DATA DISPLAY                    */
PI(0BH) = 0;                                                       /* LED'S                             */
PI(0FH) = 00100100B;                                               /* MODULE CONTROL SIGNALS          */
PI20(0) = 01010111B;                                               /* CONTROL SIGNALS                  */
PI20(1) = 01110111B;                                               /* CONTROL SIGNALS                  */
/* ----- Store Image of PIA's -----                                */
PIA1A = 88H;
PIA1B = 08H;
PIA2A = PI(6);
PIA2B = 88H;
PIA3A = PI(0AH);
PIA3B = 00H;
PIA4A = PI(0EH);
PIA4B = 24H;
PIA10A = PI10(0);
PIA10B = PI10(1);
PIA20A = 57H;
PIA20B = 77H;

```

WILLIAM A. BARNWELL

```

MSLDMK=03FH;
MSLDMA=00H;
LASTMS=0;
/* ----- End of Initialization ----- */
/* */
/* ----- Announce to Operator to Perform Load Operation ----- */
/* */
CALL ANNLD;
/* */
/* ----- Announce Memory Section Selection Required ----- */
/* */
/* */
ANL5: CALL ANNMSS;
IF (MSSVAL=06H)OR(MSSVAL=07H)OR(MSSVAL=0EH) THEN GO TO ANL5;
IF (ERR=OFFH) THEN GO TO ANL5;
IF (MSSVAL=0FH) THEN GO TO ANL9; /* AND SET LASTMS=OFFH */
TEMP1=SHL(BIT0,CNTBUS);
MSLDMA=(MSLDMA OR TEMP1) AND MSLDMK;
/* */
/* ----- Announce Long/Short Message Selection Required ----- */
/* */
/* */
CALL ANNLSM;
ANL7: IF (MSSVAL <8) THEN CALL NPAMLD;
IF (MSSVAL <8) THEN GO TO ANL6;
/* ----- Load PAM Image Array ----- */
CALL PAMLD;
ANL6: K=0; /* K is a switch to allow LDBUF to load PAM and PSK */
/* ----- Load PAM Hardware Buffer ----- */
CALL LDBUF(K);
IF (ERR=OFFH) THEN GO TO ANL7;
/* ----- Load the PSK Preamble of the PSK Image Array ----- */
CALL PKPRLD;
K=1;
/* ----- Load the PSK Image Array ----- */
CALL PSKLD;
/* ----- Load the PSK Hardware Buffer ----- */
ANL8: CALL LDBUF(K);
IF (ERR=OFFH) THEN GO TO ANL7;
/* ----- Check for Memory Section Loading Complete ----- */
IF (MSLDMA<>MSLDMK) THEN GO TO ANL5; /* Are all sections loaded? */
ANL9: LASTMS=OFFH;
CALL ZRND;
CALL ITDIS;
/* ----- Announce Memory Section Loading Complete ----- */
CALL MSCOMP;
ANL4: CALL TIME(BLINK);
PIA20A=074H;
PIA20B=074H;
PI20(0)=074H;
PI20(1)=074H;
/* ----- Check for Load/Operate to go to Operate ----- */
IF (PI(OAH) AND BIT5)=00H THEN GO TO ANL4;
/* */
/* ----- Read MSS for Initial selection for Transmission ----- */
CALL MSTEST;
PIA3B=PIA3B AND 0DFH;

```

```

PI(OBH)=PIA3B;
CALL TIME(BLINK);
IF (CNTBUS<>0) THEN GO TO ANL3; /* CNTBUS=0 ==> Random Data Req. */
/* --- For Random Data from Xmission to Xmission MSS Set to 0 --- */
CALL ENRCNT;
/*
/* ----- Start of First Xmission ----- */
ANL3:  CALL ITDIS;
      CALL IPINT;
      IF (CNTBUS<>0) THEN GO TO ANL2;
      CALL RNSQLD; /* Random Sequence Load */
ANL2:  IF (IPCOM=0) THEN GO TO ANL2; /* Test for IP Interrupt */
      IF (CNTBUS<>0) THEN GO TO ANL1;
/* ----- Load Next Random Data into PSK Hardware Buffer ----- */
PIA20A=PIA20A OR BIT5;
PI20(0)=PIA20A;
PIA3B=PI(OBH) AND OEFH;
PI(OBH)=PIA3B;
K=1;
CALL LDBUF(K);
IF (ERR<>0) THEN GO TO ANL2;
PIA3B=PIA3B OR 10H;
PI(OBH)=PIA3B;
RANCNT=RANCNT + 1;
CALL ENRCNT;
/* ----- Check for Next Selection for Next Xmission ----- */
ANL1:  CALL MSTEST;
/* ----- Check for Load Request before Xmission ----- */
IF (PI(OAH) AND BIT5)=20H THEN GO TO ANL3; /* Test Load/Operate */
GO TO MAIN;
EOF

```