

AD-A127 278

DESIGN FABRICATION AND TEST OF AN INTEGRATED
BUILT-IN-TEST (BIT) CONTROL UNIT(U) SPERRY SYSTEMS
MANAGEMENT HUNTSVILLE AL T R HOOP ET AL. OCT 82

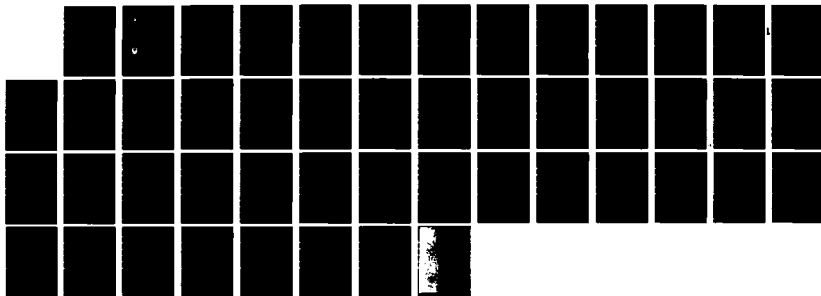
1/1

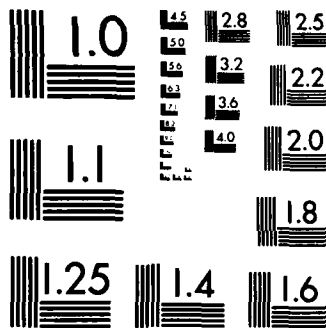
UNCLASSIFIED

DRSMI/RL-CR-83-1 DAAH01-81-D-A012

F/G 14/2

NL





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

750 376

EAS 750

(12)

AD 11 9 1 2 0 8



TECHNICAL REPORT RL-CR-83-1

DESIGN, FABRICATION AND TEST OF AN
INTEGRATED BUILT-IN-TEST(BIT) CONTROL
UNIT FINAL REPORT

Thomas R. Hoop and Edwin K. Thomas
Sperry Corporation
Sperry Systems Management
Huntsville, Alabama 35801

October 1982

Prepared for
Structures Directorate
US Army Missile Laboratory

412472



U.S. ARMY MISSILE COMMAND

Redstone Arsenal, Alabama 35809

Approved for public release; distribution unlimited.

DTC FILE COPY

DTC
APR 13 1983

S
A

DISPOSITION INSTRUCTIONS

DESTROY THIS REPORT WHEN IT IS NO LONGER NEEDED. DO NOT RETURN IT TO THE ORIGINATOR.

DISCLAIMER

THE FINDINGS IN THIS REPORT ARE NOT TO BE CONSTRUED AS AN OFFICIAL DEPARTMENT OF THE ARMY POSITION UNLESS SO DESIGNATED BY OTHER AUTHORIZED DOCUMENTS.

TRADE NAMES

USE OF TRADE NAMES OR MANUFACTURERS IN THIS REPORT DOES NOT CONSTITUTE AN OFFICIAL INDORSEMENT OR APPROVAL OF THE USE OF SUCH COMMERCIAL HARDWARE OR SOFTWARE.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

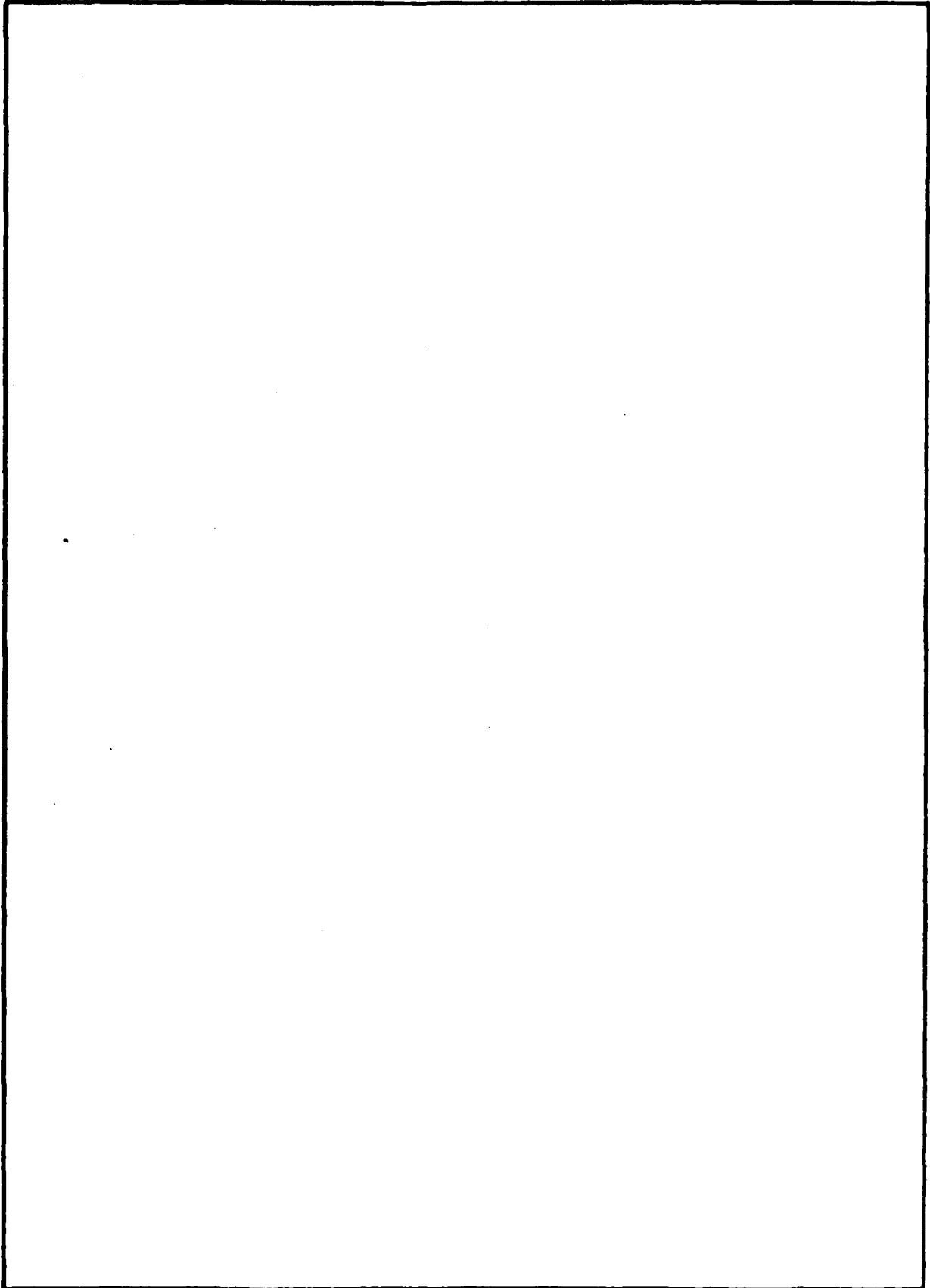
REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER RL-CR-83-1	2. GOVT ACCESSION NO. AD-A127 278	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Final Report; Design, Fabrication and Test of an Integrated Built-in-Test (BIT) Control Unit		5. TYPE OF REPORT & PERIOD COVERED
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Thomas R. Hoop and Edwin K. Thomas		8. CONTRACT OR GRANT NUMBER(s) DAAH01-81-D-A012
9. PERFORMING ORGANIZATION NAME AND ADDRESS Sperry Corp Sperry Systems Management Huntsville, Alabama 35801		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Commander US Army Missile Command ATTN: DRSMI-RPT Redstone Arsenal, AL 35898		12. REPORT DATE October 1982
		13. NUMBER OF PAGES 41
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Commander US Army Missile Command ATTN: DRSMI-RLD Redstone Arsenal, AL 35898		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) Fault Detection Peakhold Fault Isolation Module Built-in-test		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This report describes the design of an integrated Built-In-Test (BIT) control unit which may be interfaced with selected missile control subsystems. The control unit design provides the capability to perform tests of the subsystem operational integrity at time of turn-on and continually monitors system operation. In addition, the control unit design provides the capability to perform limited fault diagnosis and isolation to a replaceable assembly. The control unit design provides both a visual display and voice output to alert the operator of a detected malfunction.		

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

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)



SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

TABLE OF CONTENTS

<u>SECTION</u>		<u>PAGE</u>
1.0	SCOPE	1
2.0	DESCRIPTION OF TOW POWER CONDITIONER.....	1
3.0	TEST METHODS.....	1
3.1	Operational Test Modes.....	2
3.1.1	Mode 1 - Fault Detection.....	2
3.1.2	Mode 2 - Fault Isolation.....	2
4.0	BIT/BITE TEST EQUIPMENT.....	5
4.1	General Description.....	5
4.2	Operator Interface	5
4.3	Central Processing Unit.....	9
4.4	Analog Data Acquisition Board.....	9
4.5	Speech Synthesizer	9
4.6	Display Unit.....	11
4.7	Fault Generation Circuitry.....	11
5.0	SOFTWARE.....	11
6.0	SUMMARY.....	13
APPENDIX A		
	Program Flowchart.....	A-1 - A-12
APPENDIX B		
	Program Listing.....	B-1 - B-12

1.0 SCOPE

This report describes the design of an integrated Built-In-Test (BIT) control unit which may be interfaced with selected missile control subsystems. The control unit design provides the capability to perform tests of the subsystem operational integrity at time of turn-on and continually monitors system operation. In addition, the control unit design provides the capability to perform limited fault diagnosis and isolation to a replaceable assembly. The control unit design provides both a visual display and voice output to alert the operator of a detected malfunction.

The TOW Power Conditioner is the demonstration missile subsystem to be tested by the BIT control unit. The details of the control unit design as they are described in this report apply primarily to the TOW Power Conditioner.

2.0 DESCRIPTION OF TOW POWER CONDITIONER

The TOW Power Conditioner is a single, sealed package with two service connectors and a test point access panel (the test points available are only useful as a function/no-function type test). It converts a 24.5 Vdc input to three isolated output voltages: +24V, +50V, and -50V. These conversions are performed by eight modules which are connected to a motherboard by push-on terminals. All interconnections between modules A2 through A7 are provided by this motherboard.* To access these modules, a side panel and the front panel must be removed. Removal of the side panel provides access to test points on the motherboard.

3.0 TEST METHODS

The BIT control unit monitors the power conditioner input and outputs to determine its operational go/no-go status. In the event of a failure, the BIT unit identifies common failures and isolates them to replaceable modules or interface between modules of the power conditioner. Identified faults are announced by both a visual display and a voice output.

*Module A1 plugs into module A2 and module A8 is mounted to the sidewall.

The test equipment has two modes of operation: Mode 1 identifies the presence of a fault, and Mode 2 isolates the source of the fault to a replaceable module. In Mode 1, to detect a fault, the power conditioner's input and output voltages are monitored at the package interface. If a voltage falls outside a specified tolerance band, an alarm is activated to notify the operator. The operator is alerted by a display and/or by a synthesized voice output. In the event of a failure, the power conditioner package is removed and replaced.

Once a fault has been detected in the unit during Mode 1, if repairs are to be made, the package must be opened. A test connector is accessible after removal of the side panel.

3.1 Operational Test Modes

3.1.1 Mode 1 - Fault Detection

During Mode 1, the voltages present on output connector J2 are monitored to detect a fault. These outputs are isolated and examined with a differential voltmeter.

To be within tolerance, the 24V output may vary between 22.2 and 33V. This must be true for a supply voltage of 24 \pm 5.5V. The 48V output voltages may vary between 48 and 56V. For this task, the outputs are monitored under no load with a 24V input. The only cables in use are those tied to J1, the input power connector, and J2 the output connector.

3.1.2 Mode 2 -Fault Isolation

Once a fault has been detected in Mode 1, the power conditioner side panel is removed and a fault isolation connector attached for Mode 2 fault isolation. An ITT/Cannon MDB connector is used.

To isolate faults to a specific module, the test points listed in Table 1 are monitored. The theoretical waveforms to be monitored are shown in Figure 1. To examine the waveforms, conversion circuitry is required.

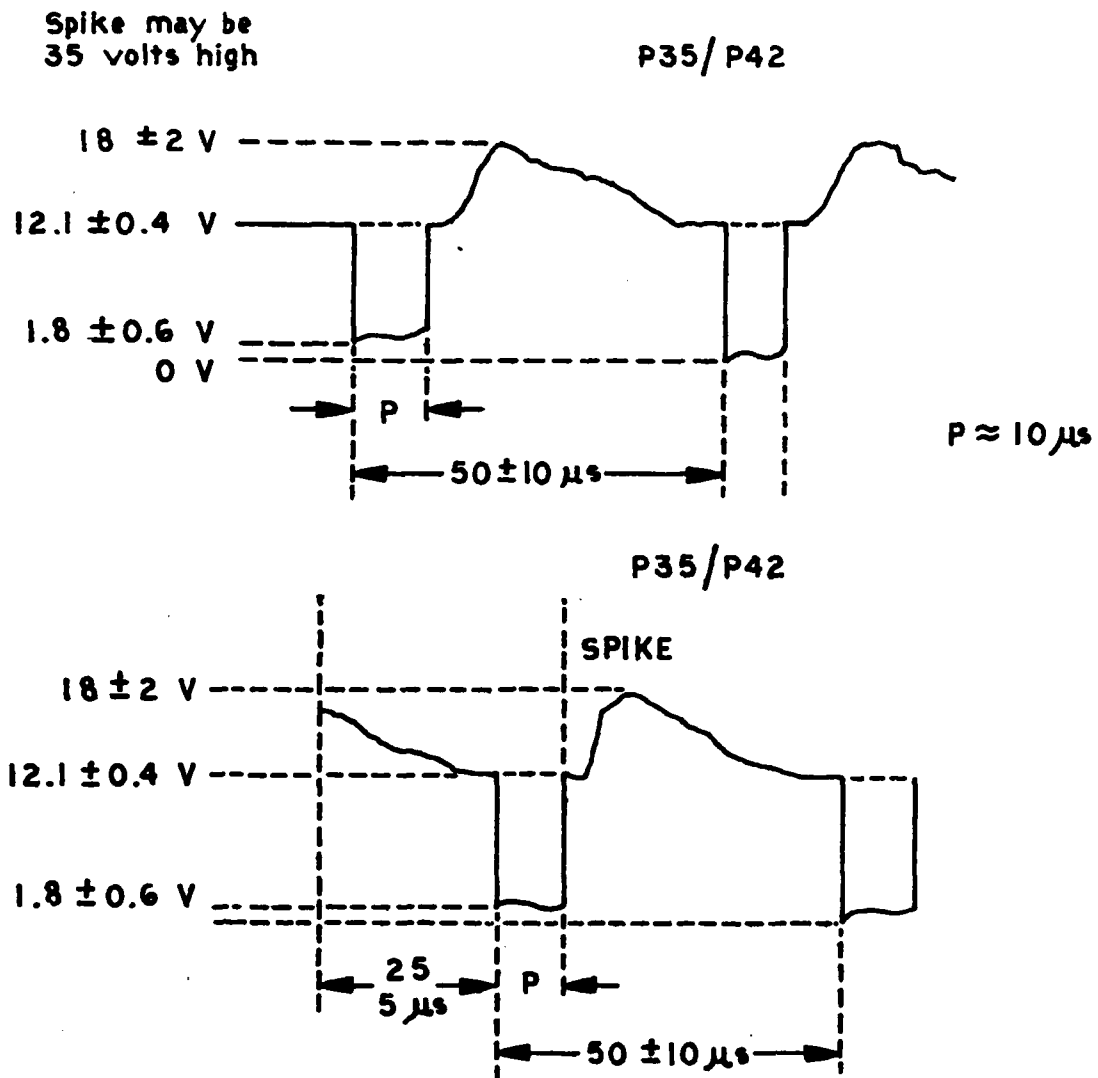


Figure 1. Output waveforms for Module A3.

To implement this requirement, the peak measurement circuits shown in Figure 2 are used. These acquire the maximum and minimum peaks of the waveform. If the output stays high, the output of the minimum peak circuit will be incorrect. If it stays low, the output of the maximum peak circuit will be incorrect. If a pulse is present and both levels are in an acceptable range, the waveform is acceptable. The A/D converter samples the peak outputs to determine if they are in the correct tolerance band.

The only other waveform to be examined is at the output of the A5 module. Three secondaries are rectified and deliver dc voltages. The 24V output, however, is rectified at the A6 module, so waveform peaks are examined and held by the circuit shown in Figure 3.

4.0 BIT/BITE TEST EQUIPMENT

4.1 General Description

To implement the tests described above the equipment of Figure 4 is used. The system is housed in a standard 19-inch rack mounted unit. An external, rack mounted power supply is necessary to provide the high power-up current of the power conditioner. Internally, the BIT control unit electronics are housed in a MULTIBUS * card cage and have a separate power supply. In addition to the display and function keys, the power conditioner's interface connectors are mounted on the front panel. The external supply voltage is routed through the BIT control unit and has an on/off switch on the front panel.

4.2 Operator Interface

The operator interface is accomplished by five switches/keys on the control unit front panel. The TESTER POWER switch applies power to all components of the control unit itself. The POWER CONDITIONER switch applies power to the power conditioner connector J1. Both power switches have an indicator lamp. The MODE key selects the fault detection or fault isolation mode. The RESET key reboots the system and starts program operation. The CONTINUE key restarts program operation at programmed stopping points.

*MULTIBUS is a trademark of Intel Corporation

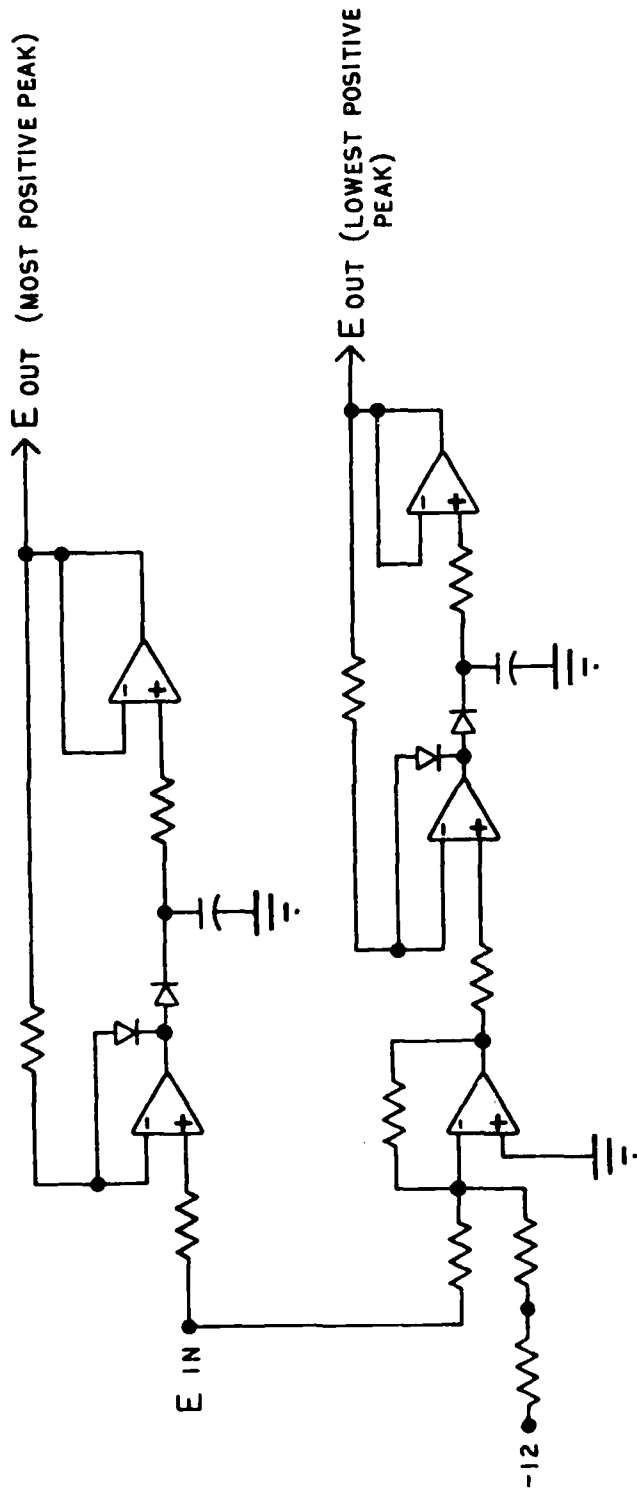
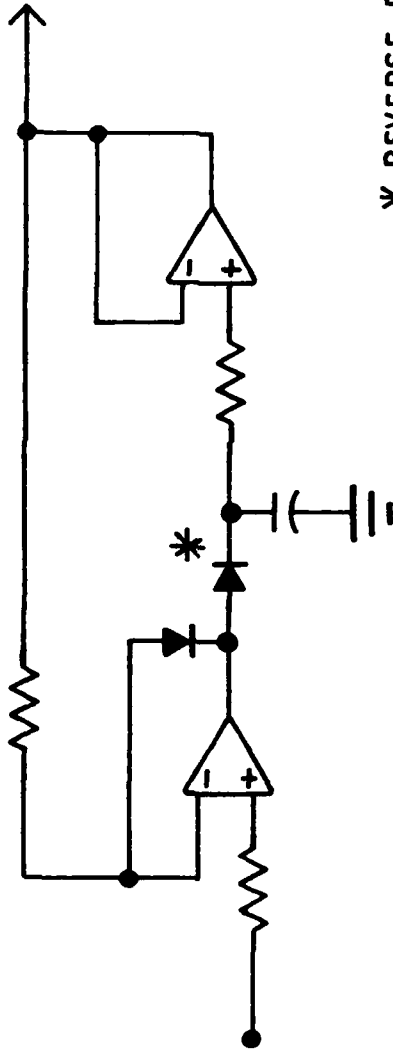


Figure 2. Peak measurement circuitry.



* REVERSE POLARITY FOR
NEGATIVE LEG

Figure 3. Peakhold circuitry for output of Module A5.

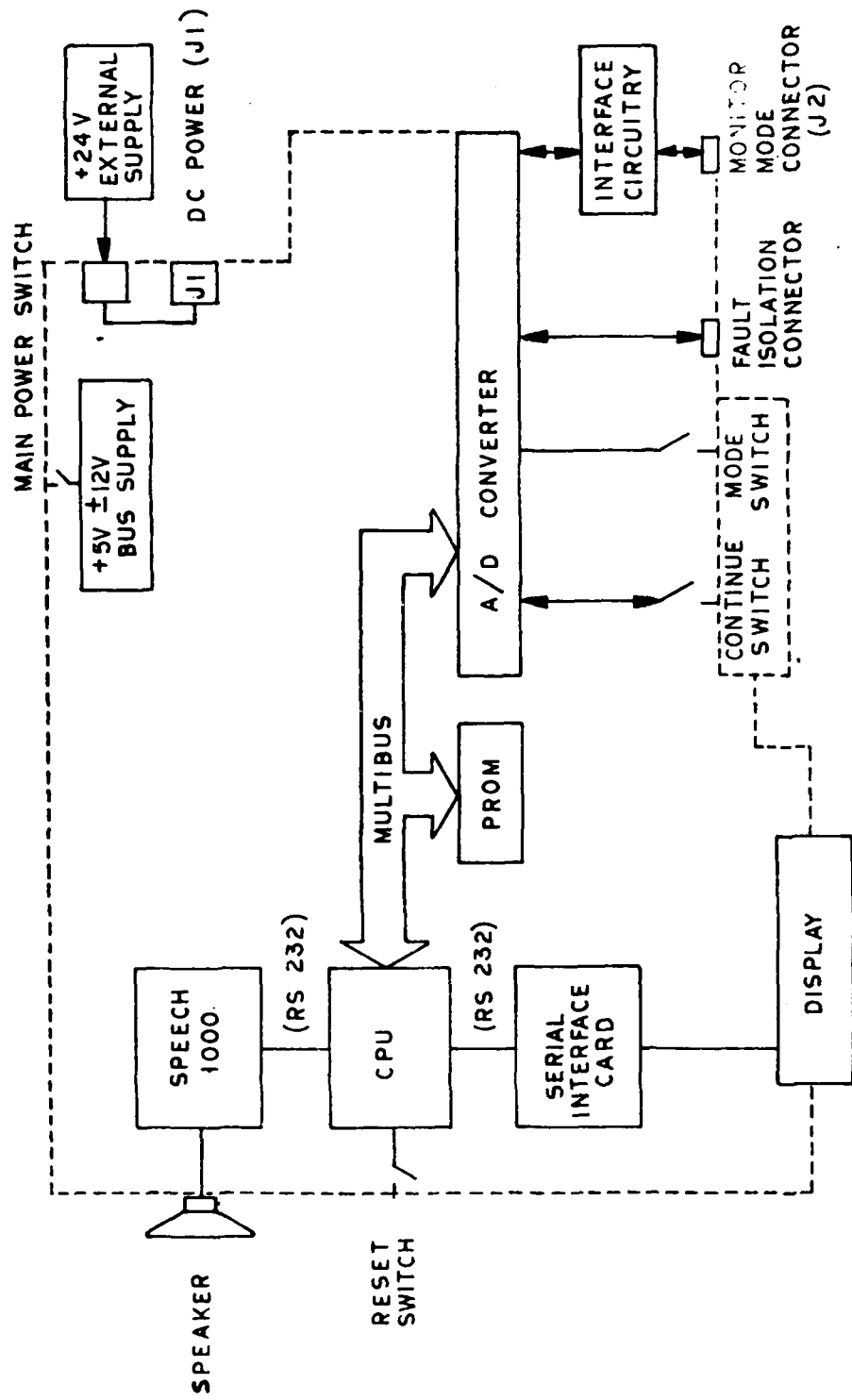


Figure 4. Block diagram of tester.

Three cables provide the interface between the B17 control unit and the power conditioner. The first cable interfaces with power conditioner connector J1 and supplies 24V input power. The second cable interfaces with the power conditioner connector J2 to monitor output voltages. The third cable is used only during Mode 2 for fault isolation, and connects to the test points listed in Table 1 through the fault generation box.

4.3 Central Processing Unit

The control unit CPU card is a Forward Technology FT-68M, which has a Motorola M68000 CPU, 256K bytes of RAM, 32 Kbytes of PROM, two TS-232C interfaces, and is MULTIBUS compatible.

4.4 Analog Data Acquisition Card

Analog data acquisition is performed with a MULTIBUS compatible A/D converter card, the National Semiconductor 8737 analog I/O card with memory. It has 32 single-ended/16 differential channels and a 12 bit A/D converter. For this application, analog inputs are automatically sampled in a sequential repetitive mode. The on-card RAM holds the current value for each channel. Divider networks provide suitable input voltage levels.

4.5 Speech Synthesizer

To notify the operator of a system failure in Mode 1 or to direct operator actions for fault locating in Mode 2, a Telesensory speech synthesizer is used. The Speech 1000 card is controlled by the host computer through the use of commands and word pointers. Word pointers specify which utterance should be spoken. The commands and word pointers are downloaded in RAM buffers on the Speech 1000 card. For this project, an American male vocabulary has been selected. A list of available words for sentence construction is provided in Table 2. The on-card amplifier supplies up to 2 watts of speech output into the 8-ohm speaker. Volume can be controlled by an on-board potentiometer.

Table 2. Speech Synthesizer Vocabulary

A	HUNDRED	PLUS	VOLTS
AMPS	IN	PLUS	WARNING
AT	IS	POINT	ZERO
ATTENTION	KILO-	POWER	100 Hz TONE (.2 sec)
C	LIGHT	RANGE	400 Hz TONE (1 sec)
*CENTI-	LOADING	REPLACE	
CENTIGRADE	LOW	SECONDS	
CONDITIONER	M	SEVEN	
D	MEGA-	SEVENTEEN	
DEGREES	*METERS	SEVENTY	
*DEGREES	MICRO-	SIX	
EIGHT	MILLI-	SIXTEEN	
EIGHTEEN	MINUS	SIXTY	
EIGHTY	*MINUTES	SUPPLY	
ELEVEN	NANO-	SWITCH	
EQUALS	NINE	TEMPERATURE	
*ERROR	NINETEEN	TEN	
*F	NINETY	THE	
FAILURE	O	THIRTEEN	
FARENHEIGHT	OF	THIRTY	
FAULT	OFF	THOUSAND	
FIFTEEN	OHMS	THREE	
FIFTY	ON	TIMES	
FIVE	ONE	TOLERANCE	
FORTY	OUT	TOO	
FOUR	OUTPUT	TURN	
FOURTEEN	OVER	TWELVE	
HERTZ	P	TWENTY	
HIGH	*PERCENT	TWO	
*HOUR	PIN	UNIT	

4.6 Display Unit

To display commands and fault analysis, an IEE Model 3600-06-240 vacuum fluorescent display is used. This is a 6 line by 40 character display with 5 by 7 dot matrix characters to provide full alpha-numeric capability. All control, refresh, and display functions are executed by a dedicated on-board processor. An accessory serial data card provides an RS-232 control port for the display.

4.7 Fault Generation Circuitry

A fault generation box permits the artificial insertion of faults in the TOW power conditioner. By interrupting or modifying the signal between the TOW fault isolation cable and the BIT control unit, the degraded signals result in fault messages for the appropriate modules. A schematic is shown in Figure 5.

5.0 SOFTWARE

The program flow chart is shown in Appendix A. The program listing is shown in Appendix B. This program is written in the programming language "C". It is stored in PROM on the Forward Technology 68000 CPU Board. This program is written in a structured manner. The program, in main, tests the mode switch and branches to the mode 1 or mode 2 function. Each mode function calls a variety of functions that test an individual module. For example, "altest" tests the A1 module, "a2test" tests the A2 module, "f15 test" tests the output filter F15, etc.

To interface with the A/D card, display, and speech synthesizer, functions "atod", "disp", and "spk" are called. To display test the function "dissix" pulls six lines of text from the array of text and displays them using "disp". An array of pointers, text (), points to the text. Likewise, words (), points at the code for each spoken sentence. These arrays of pointers are defined at the start of the program. To provide for easy modification, define statements assign names to parameter ranges tested during the program. For example, "#define-J1LOW" assigns an A to D count value of 2867 counts as the lowest acceptable range for the input voltage. (This corresponds to 20 volts).

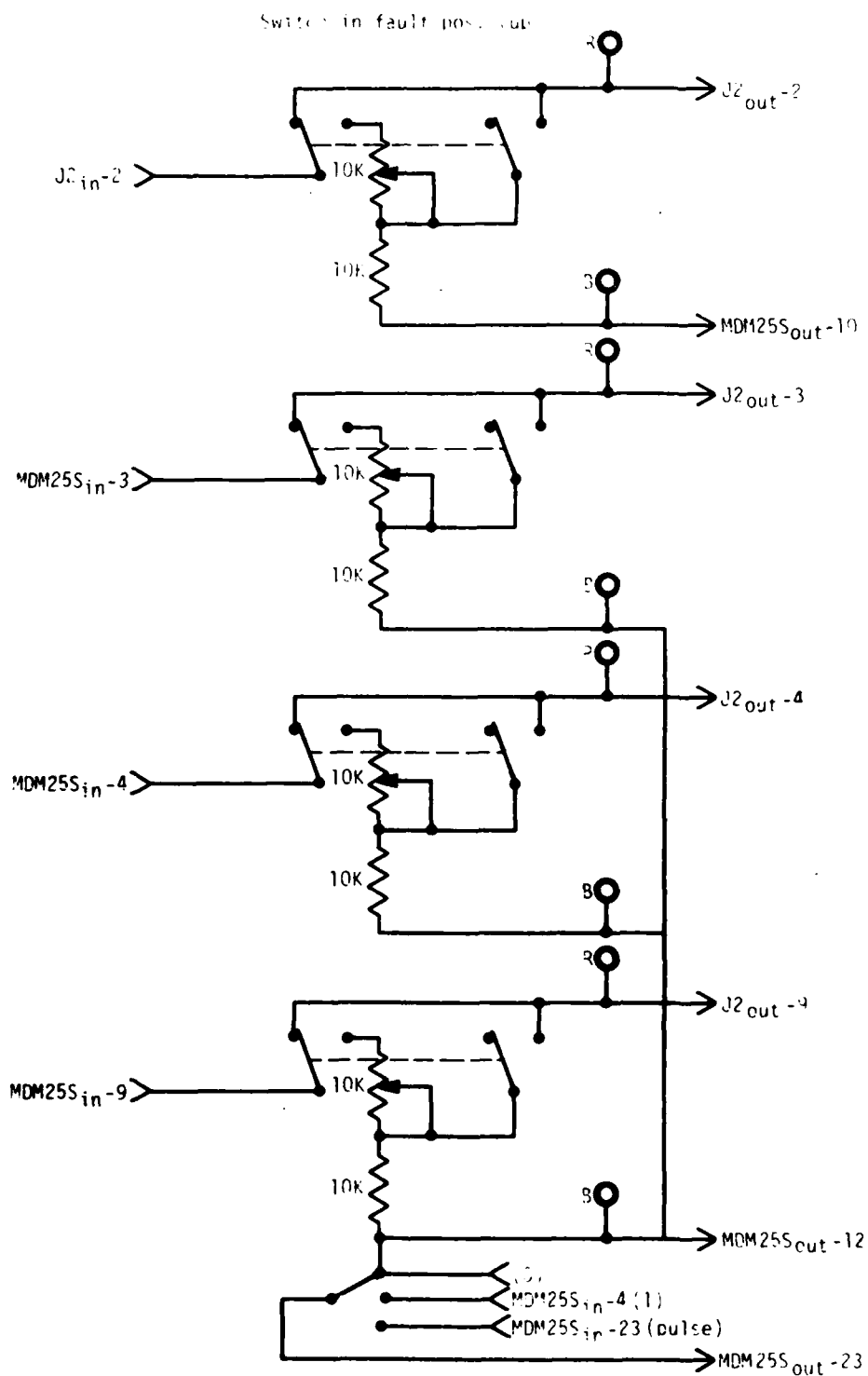


Figure 5. Fault generation circuitry.

The function "scalamod" acquires the a to d count value, converts it to a decimal number, removes the most significant digits for display, and modifies the displayable text to include the voltage value. The function "deadend" maintains the last displayed data and waits for a new instruction from the front panel. The function "cont" suspends activity until the continue switch is pressed. The function "fault" inserts a fault message into displayable text. The function "itoc" converts a 16 bit integer into an 8 bit character constant.

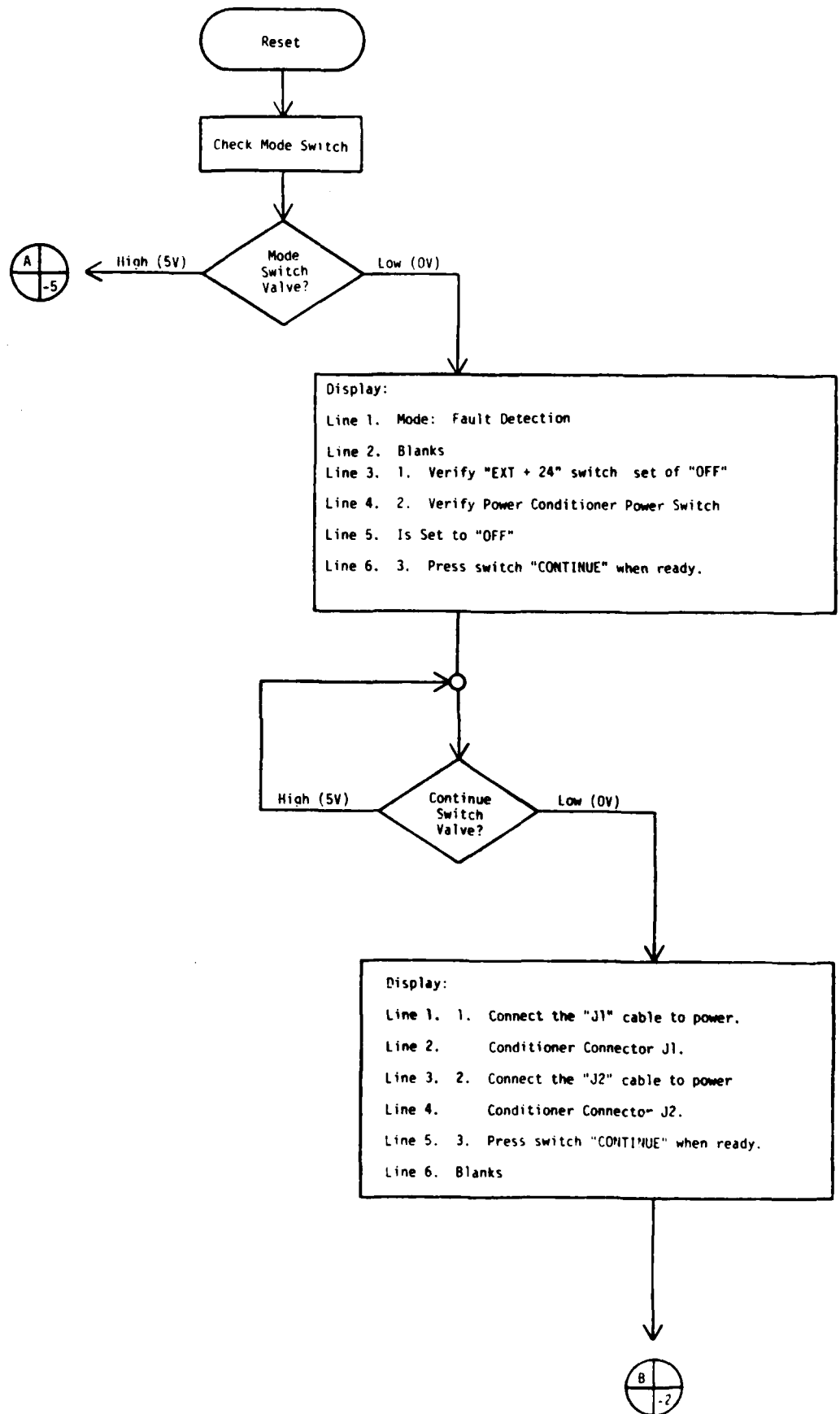
6.0 SUMMARY

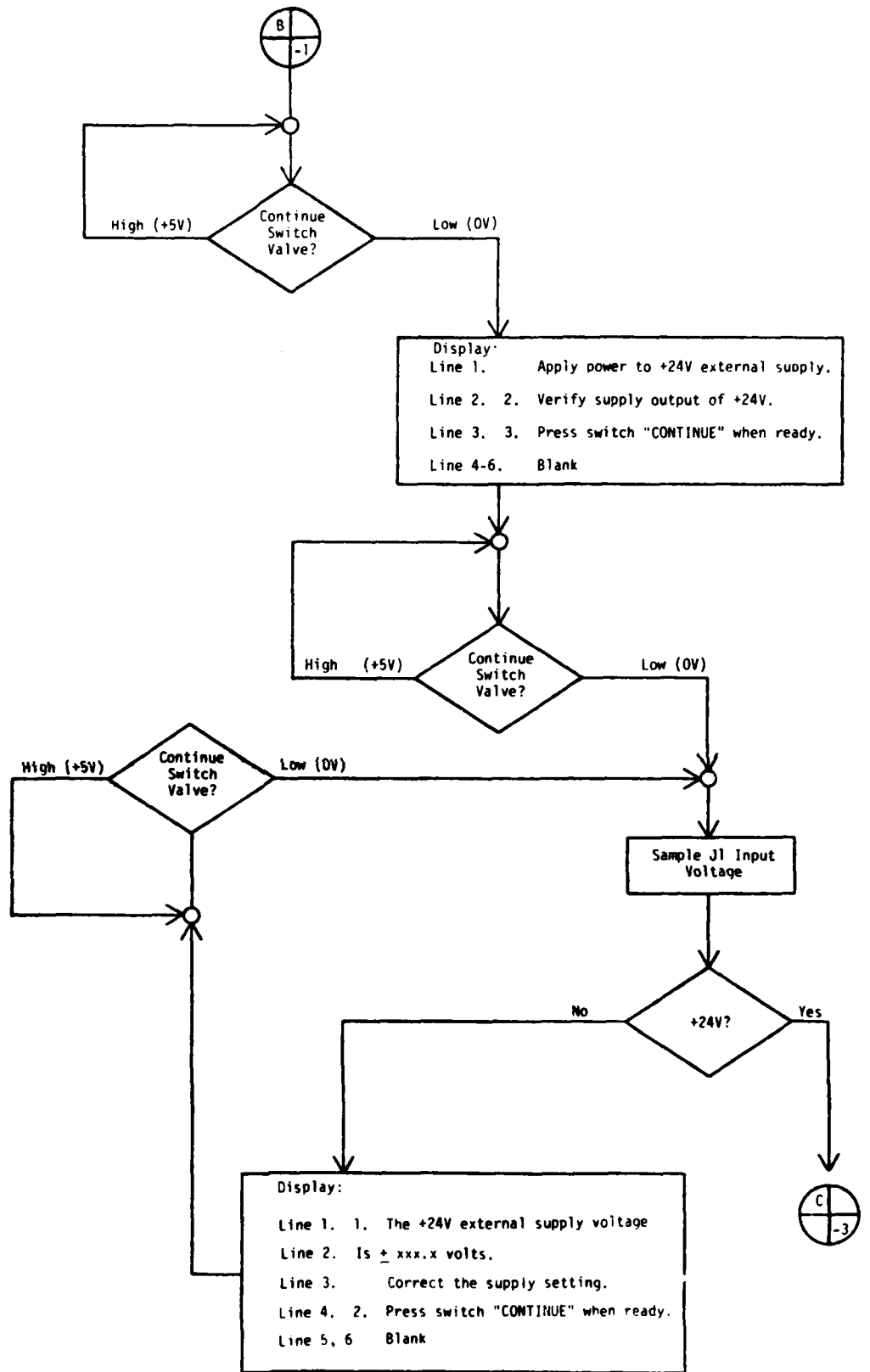
The BIT control system described above provides the necessary capability to adequately detect failures and isolate faults in the TOW Power Conditioner. The architecture of the test system design is adaptable to a wide variety of test applications by simply adding additional stimulus, measurement, and switching modules. Since the system is based on the MULTIBUS architecture, such modules are readily available off-the-shelf and may be added with little or no impact on the system hardware design described above.

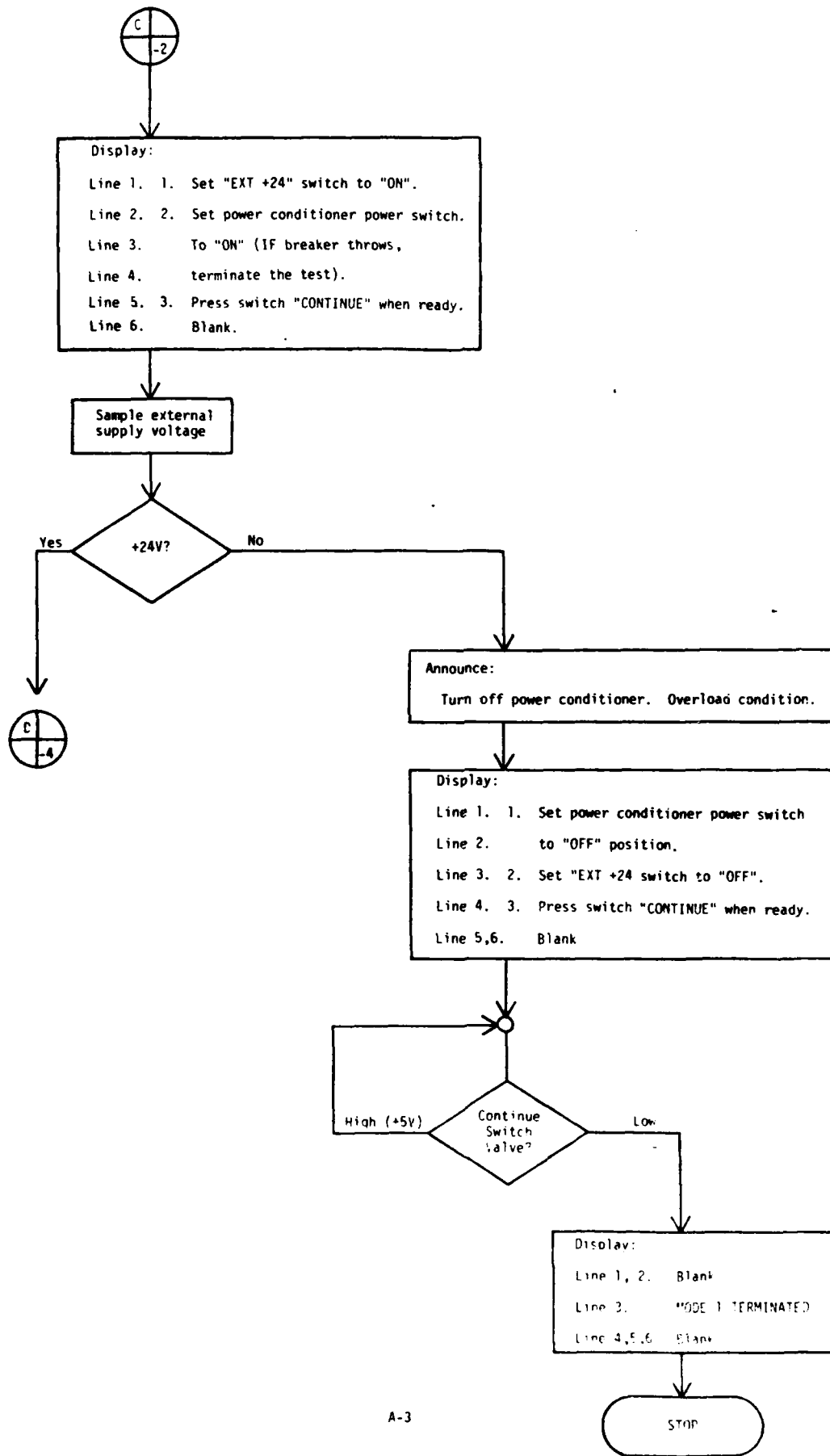
The general test methodology discussed above may be applied to a wide variety of hardware systems. However, for each application, dedicated software must be developed to perform the specific tests required for fault detection of the selected test item.

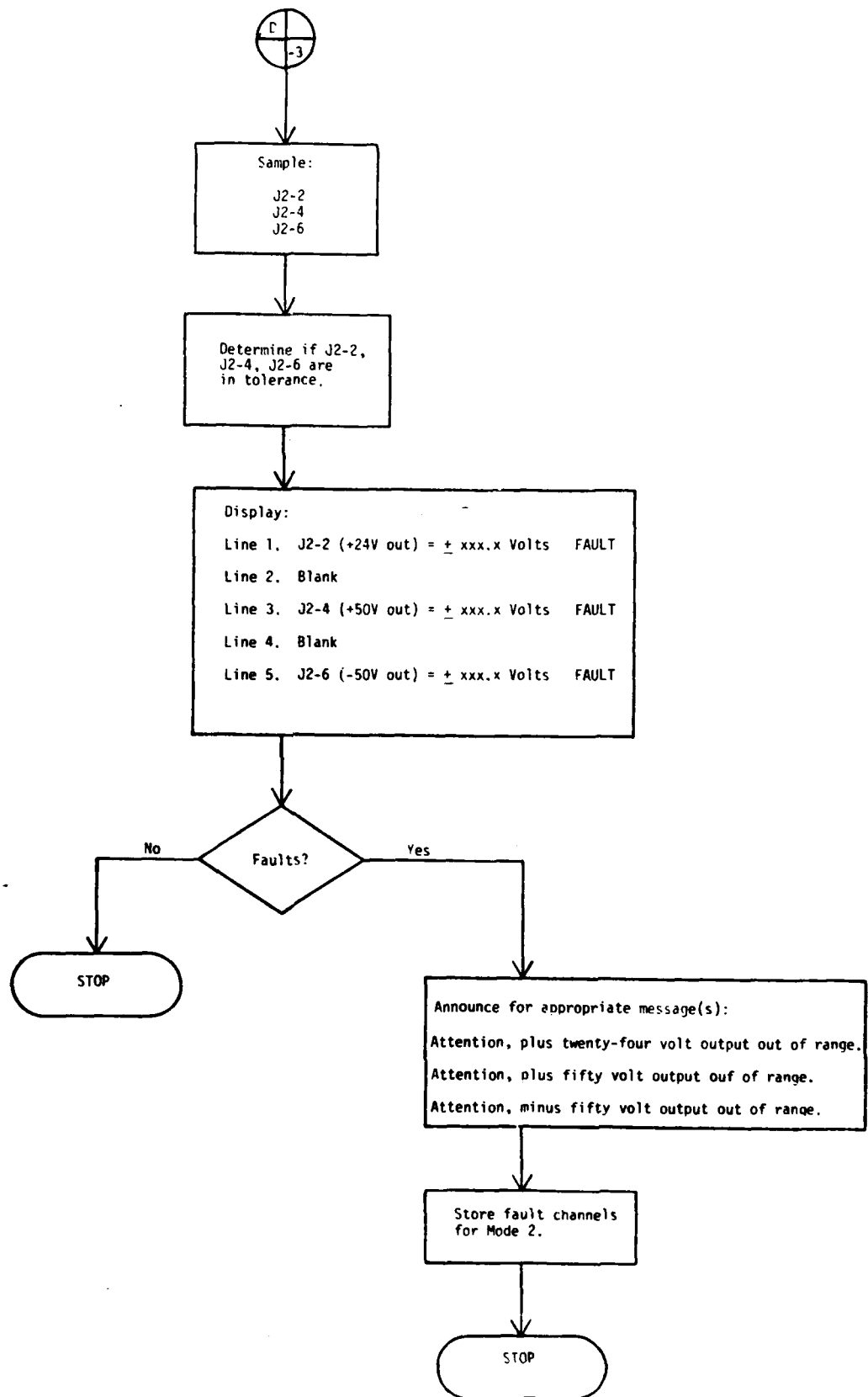
The hardware architecture described above is well suited for expansion to include a test bus and signal switching modules. Through such expansion, the BIT control unit may be integrated into a complete missile system to provide overall system integrity tests and fault isolation.

APPENDIX A
PROGRAM FLOWCHART



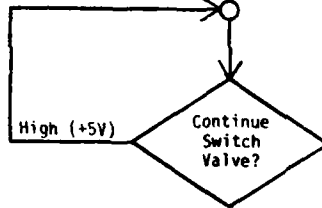




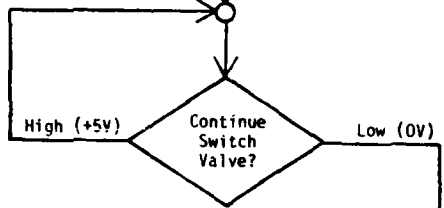




Display:
Line 1. Mode: Fault Isolation
Line 2. Blank
Line 3. 1. Verify "EXT +24" switch set to "OFF".
Line 4. 2. Verify power conditioner power switch
Line 5. is set to "OFF".
Line 6. 3. Press switch "CONTINUE" when ready.



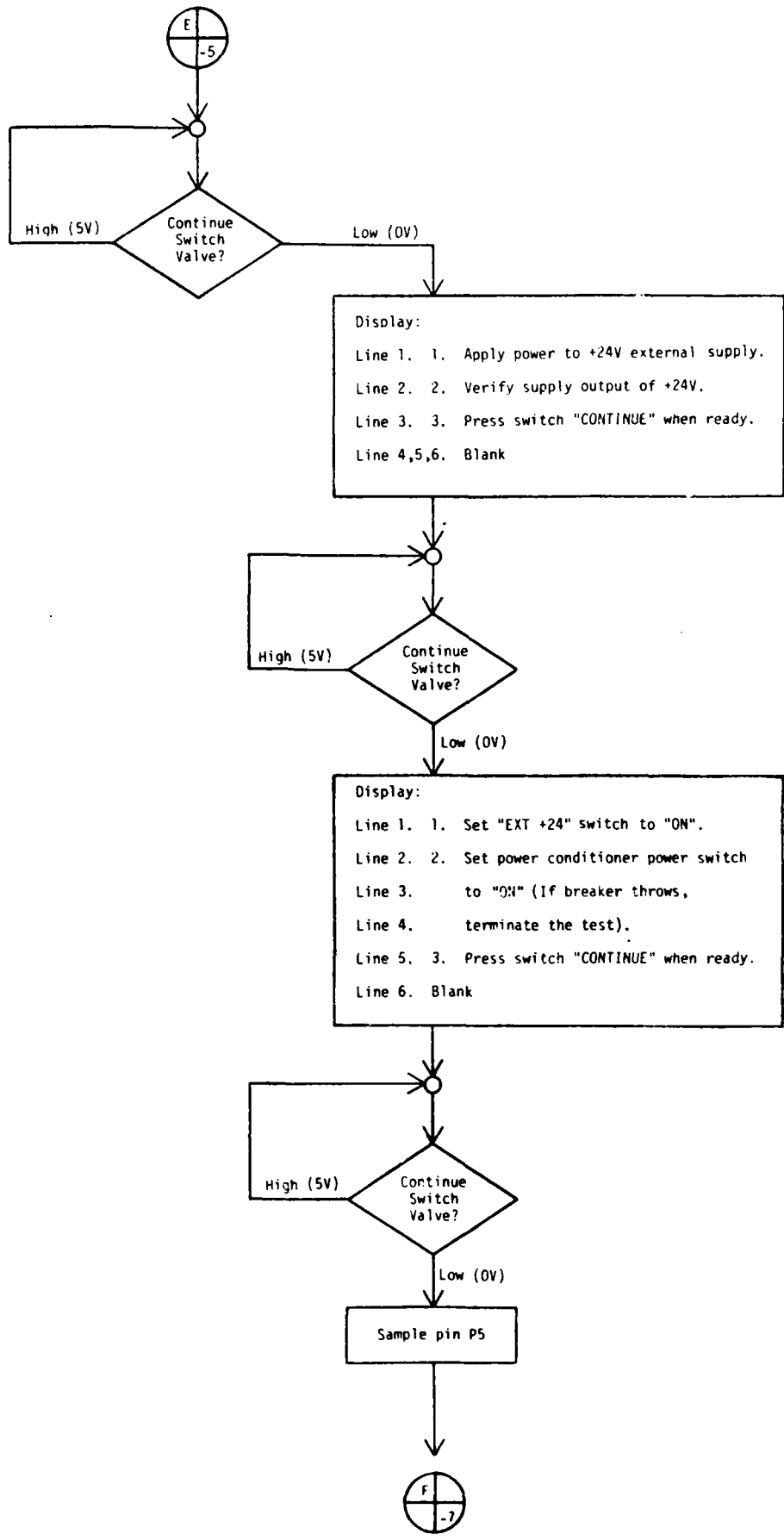
Display:
Line 1. 1. Connect the "J1" cable to power
Line 2. conditioner connector J1.
Line 3. 2. Connect the "J2" cable to power
Line 4. conditioner connector J2.
Line 5. 3. Press switch "CONTINUE" when ready.
Line 6. Blank

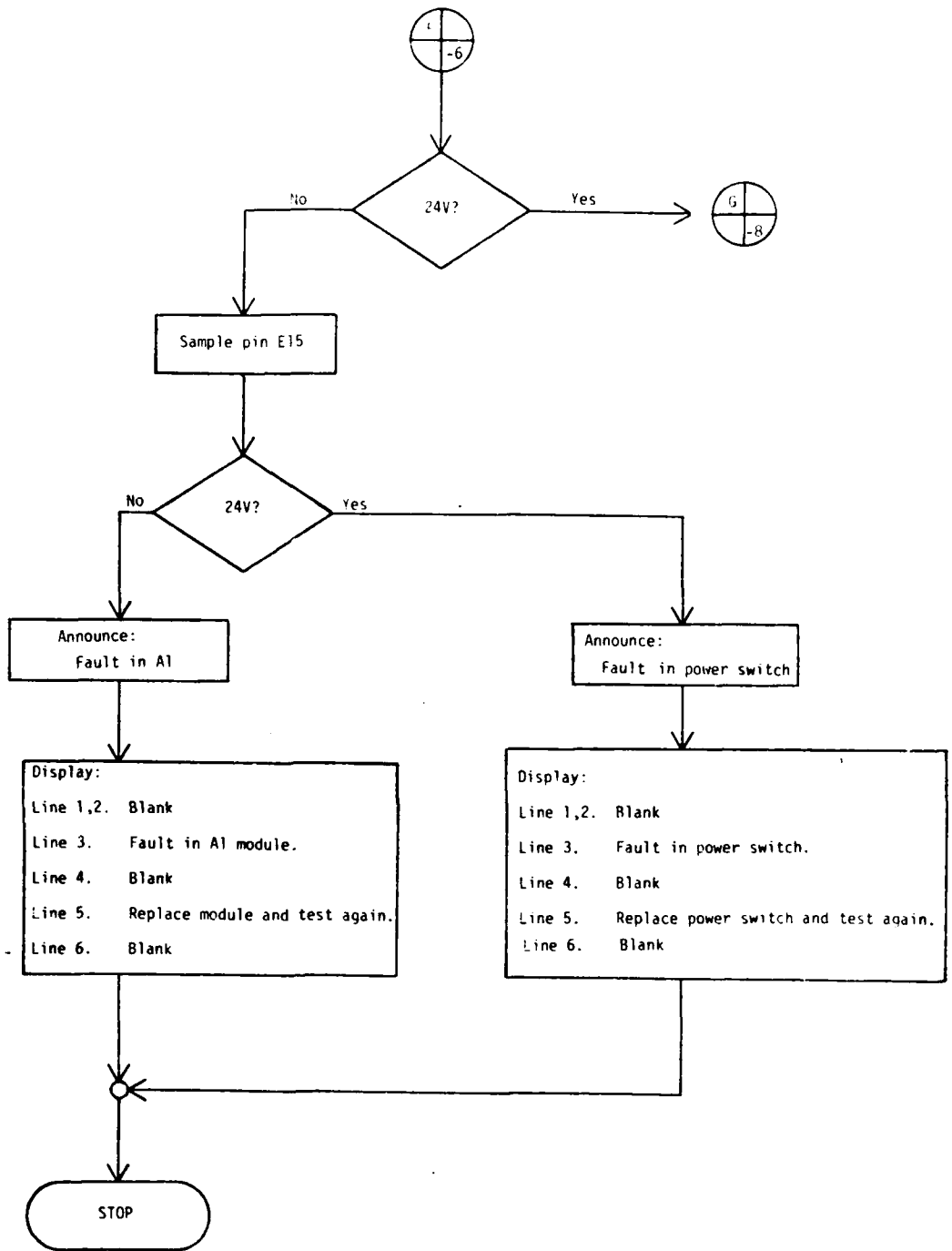


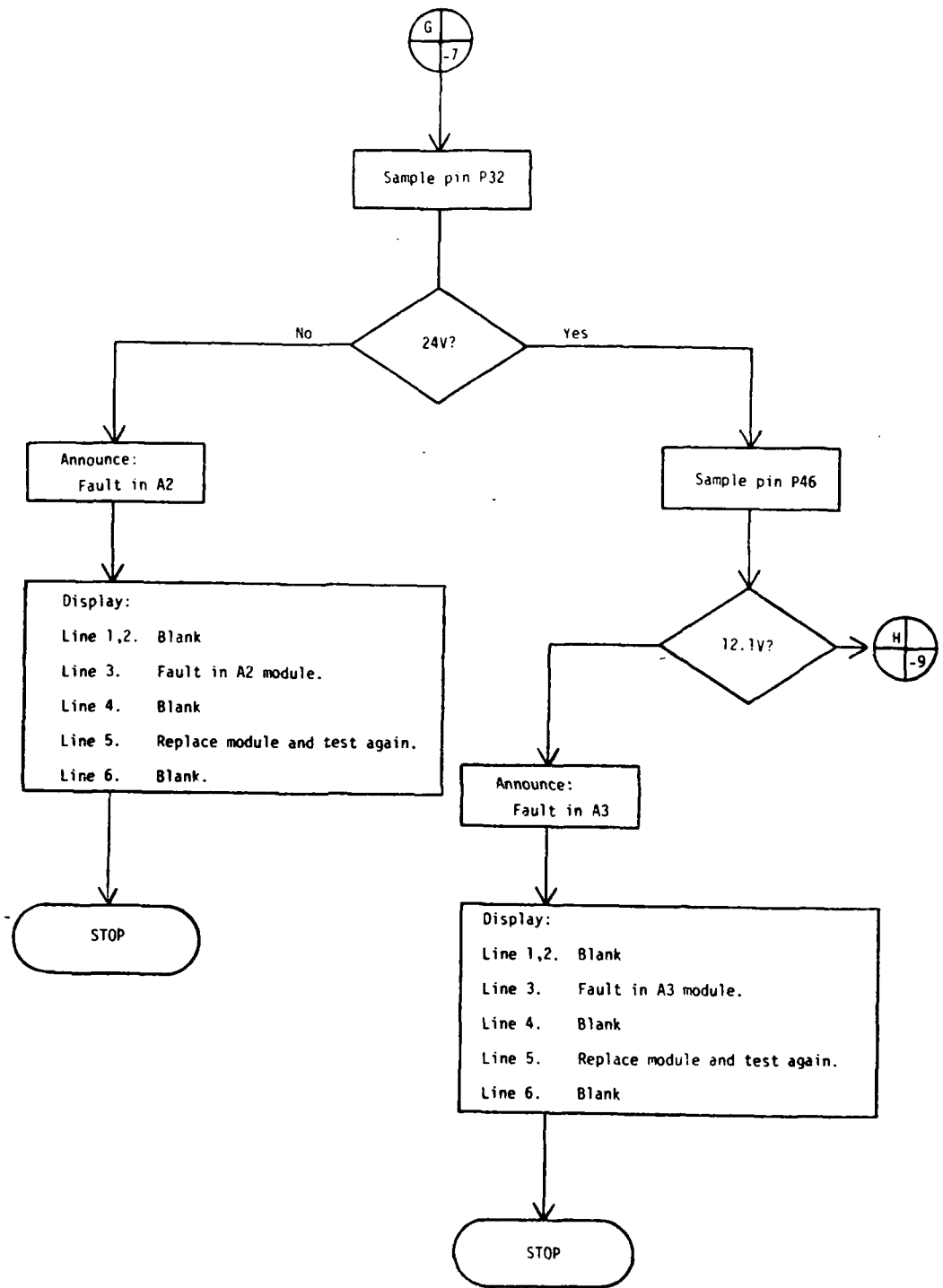
Display:
Line 1. 1. Remove the power conditioner side
Line 2. panel.
Line 3. 2. Connect the power conditioner test
Line 4. cable.
Line 5. 3. Press switch "CONTINUE" when ready.
Line 6. Blank

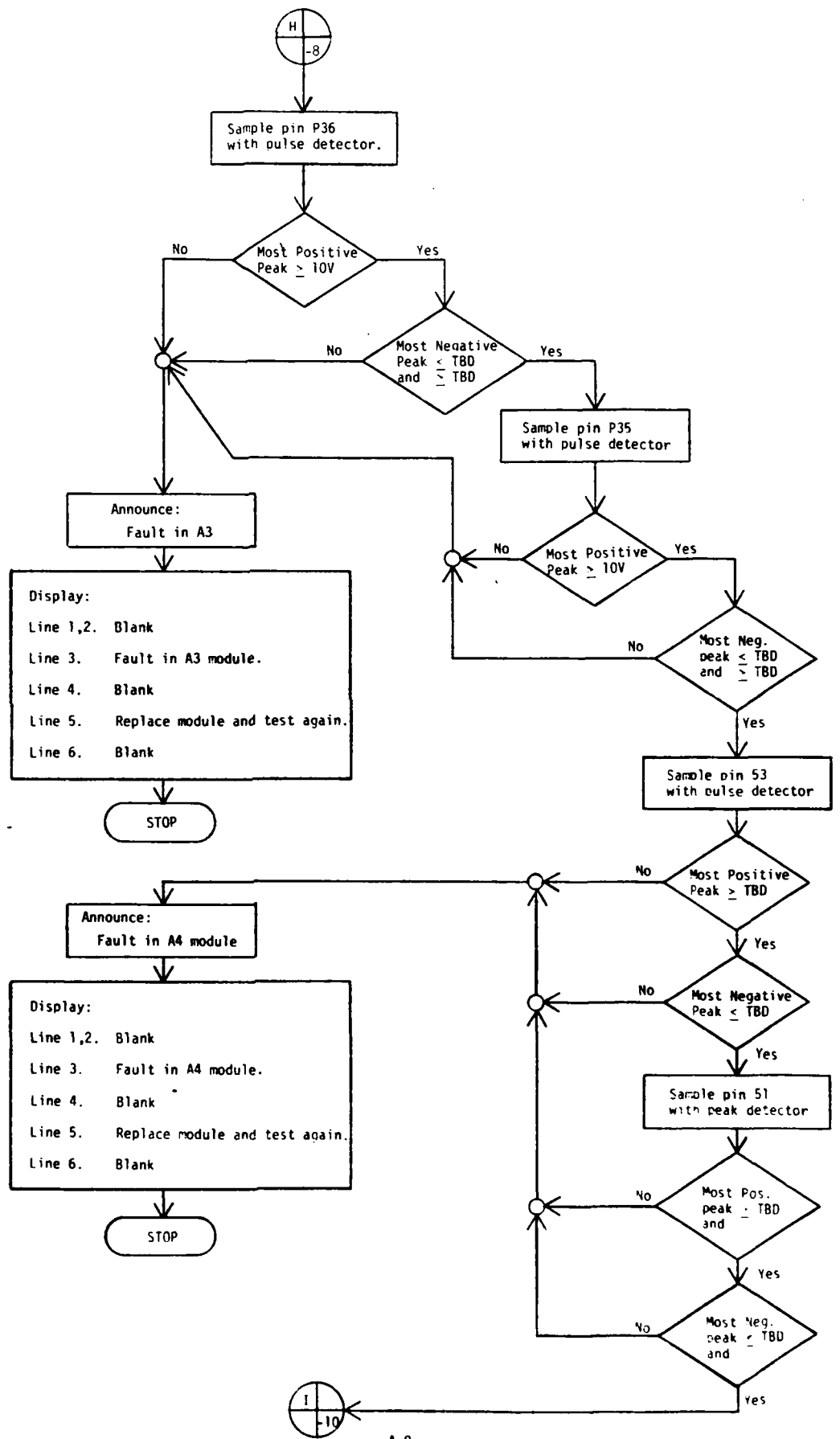
A-5

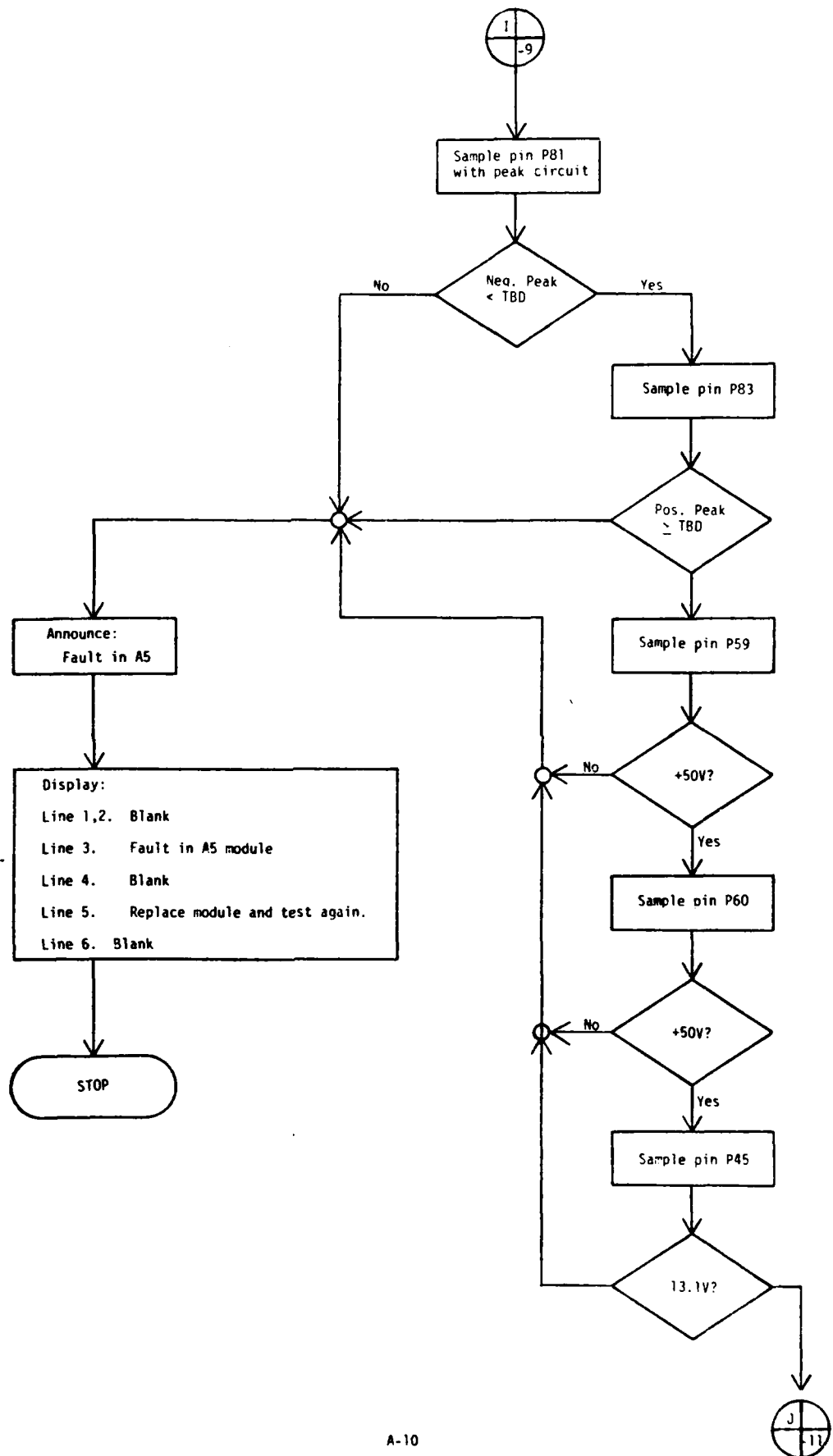


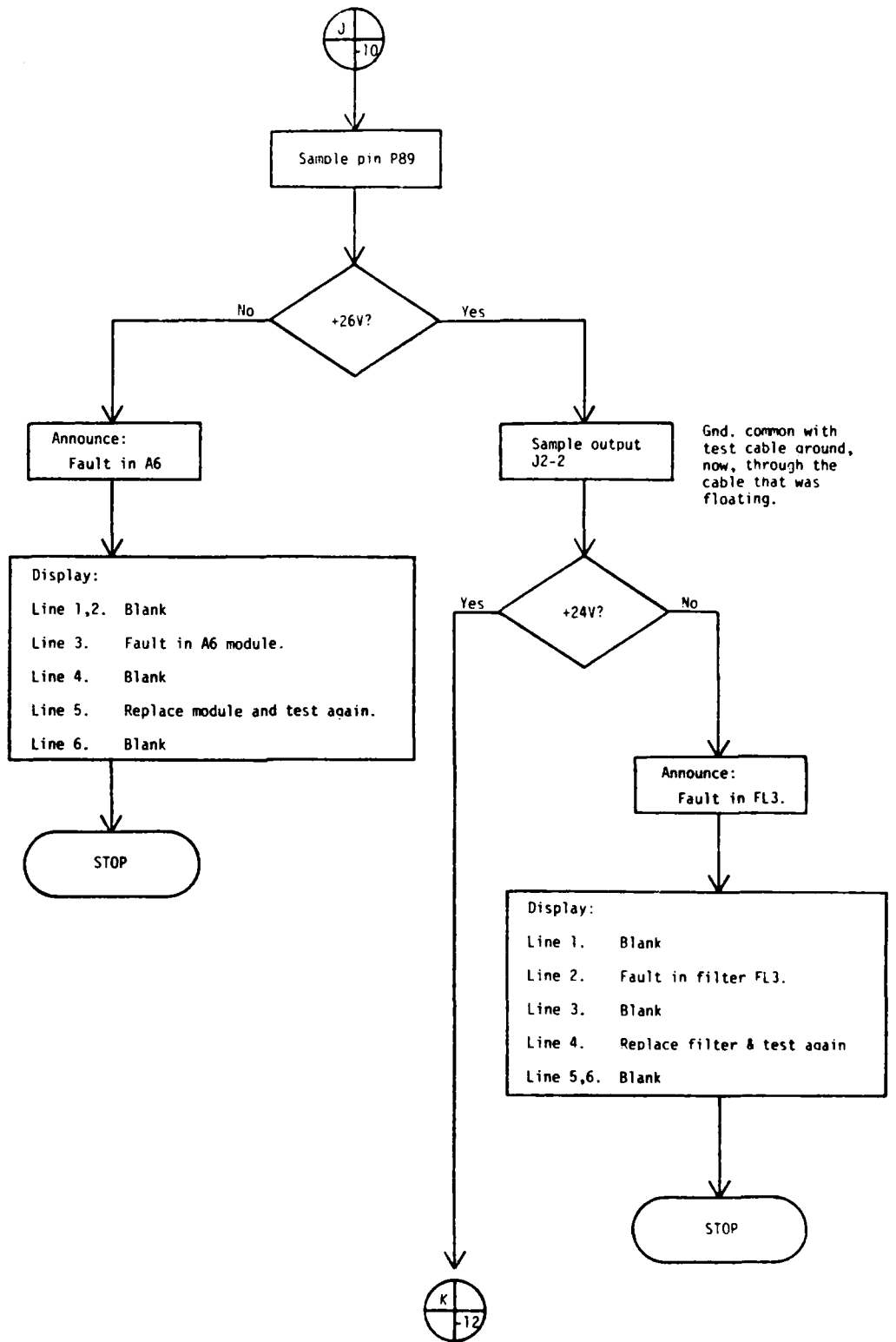


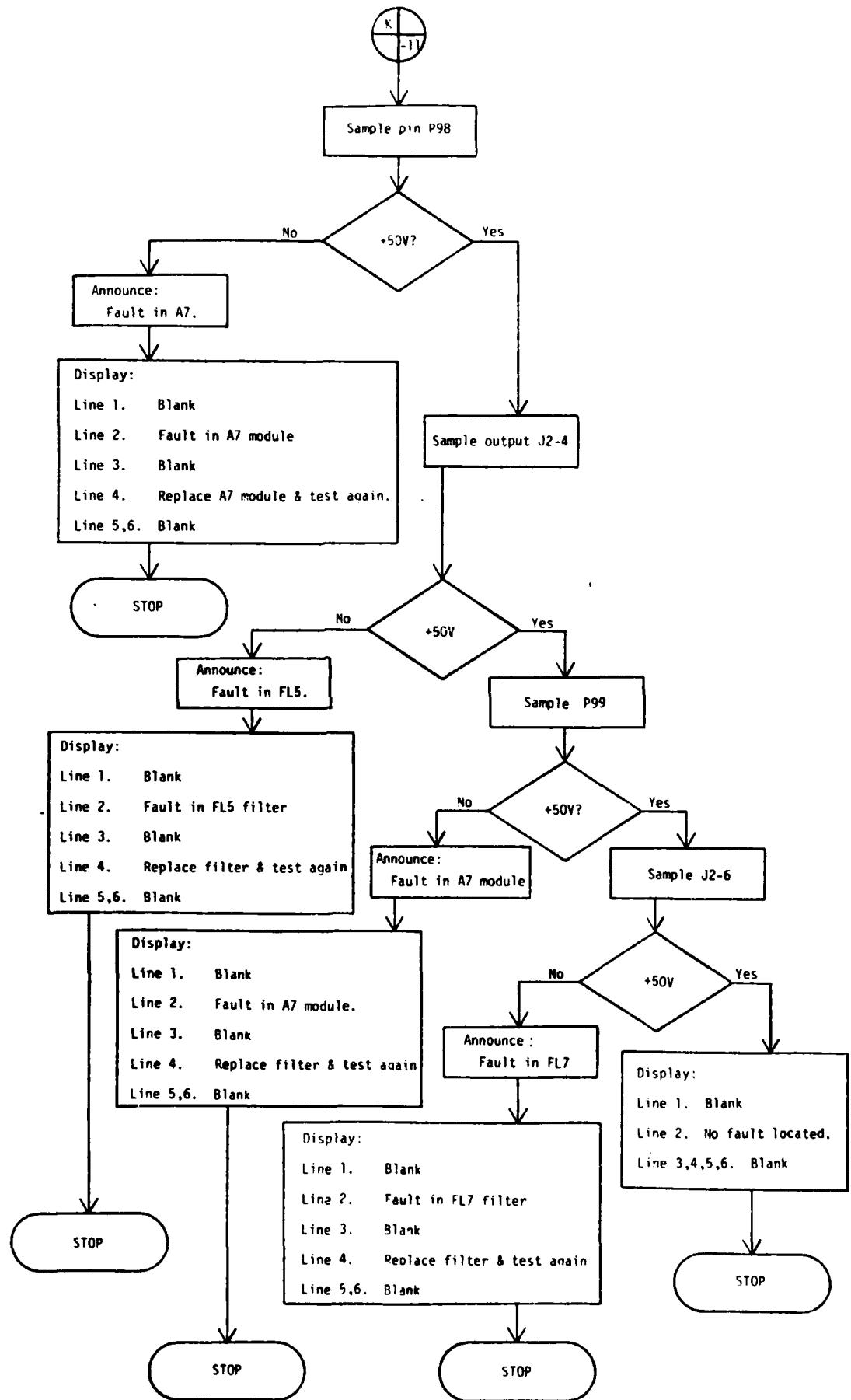












APPENDIX B
PROGRAM LISTING

```

#define          CONTSWT          2457      /* 2v */
#define          J1LOW            2867      /* 20v */
#define          J1HIGH           3726      /* 30v */
#define          J22LOW           2916      /* 21.2 */
#define          J22HIGH          3399      /* 33 */
#define          J24LOW           3030      /* 48 */
#define          J24HIGH          3194      /* 56 */
#define          J26LOW           3030      /* 48 */
#define          J26HIGH          3194      /* 56 */
#define          P32LOW           2867      /* 20 */
#define          P32HI            3723      /* 30 */
#define          E15LOW           2867      /* 20 */
#define          E15HI            3726      /* 30 */
#define          P40LOW           3071      /* 10 */
#define          P40HI            3481      /* 14 */
#define          P36BLOW          102       /* 1 */
#define          P36BHI           2303      /* 2.5 */
#define          P36TLOW          3071      /* 10 */
#define          P36THI           4095      /* 35 */
#define          P35BLOW          102       /* 1 */
#define          P35BHI           2303      /* 2.5 */
#define          P35TLOW          3071      /* 10 */
#define          P35THI           4095      /* 35 */
#define          P53BLOW          1945      /* -5 */
#define          P53BHI           2150      /* 5 */
#define          P53TLOW          2867      /* 40 */
#define          P53THI           3276      /* 60 */
#define          P51BLOW          1945      /* -5 */
#define          P51BHI           2150      /* 5 */
#define          P51TLOW          2867      /* 40 */
#define          P51THI           3276      /* 60 */
#define          P81LOW           2764      /* 70 */
#define          P83LOW           2764      /* 70 */
#define          P59LOW           3030      /* 48 */
#define          P59HI            3194      /* 56 */
#define          P60LOW           3030      /* 48 */
#define          P60HI            3194      /* 56 */
#define          P89LOW           2916      /* 21.2 */
#define          P89HI            3399      /* 33 */
#define          P98LOW           3030      /* 48 */
#define          P98HI            3194      /* 56 */
#define          P99LOW           3030      /* 48 */
#define          P99HI            3194      /* 56 */
static char *text[] = {
    "\024",
    "Mode: Fault Detection\015\012",
    " \015\012",
    "1. Verify \"EXT +24\" switch set to \"OFF\"\015\012",
    "2. Verify power conditioner power switch",
    " is set to \"OFF\".\015\012",

```

```

"3. Press switch \"CONTINUE\" when ready.",
"1. Connect the \"J1\" cable to power\015\012",
"  conditioner connector J1.\015\012",
"2. Connect the \"J2\" cable to power\015\012",
"  conditioner connector J2.\015\012",
"3. Press switch \"CONTINUE\" when ready.\015\012",
" ",
"1. Apply power to +24V external supply.\015\012",
" \015\012",
"2. Verify supply output of +24V. \015\012",
" \015\012",
"3. Press switch \"CONTINUE\" when ready.\015\012",
" ",
"1. The +24V external supply voltage\015\012",
"  is sabc.d volts.\015\012",
"  Correct the supply setting.\015\012",
"2. Press switch \"CONTINUE\" when ready.\015\012",
" \015\012",
" ",
"1. Set \"EXT +24\" switch to \"ON\".\015\012",
"2. Set power conditioner power switch\015\012",
"  to \"ON\" (if breaker throws,\015\012",
"  terminate the test).\015\012",
"3. Press switch \"CONTINUE\" when ready.\015\012",
" ",
"1. Set power conditioner power switch\015\012",
"  to \"OFF\".\015\012",
"2. Set \"EXT +24\" switch to \"OFF\".\015\012",
"3. Press switch \"CONTINUE\" when ready.\015\012",
" \015\012",
" ",
" \015\012",
" \015\012",
"MODE 1 TERMINATED\015\012",
" \015\012",
" \015\012",
" ",
"J2-2 (+24V out) = sabc.d Volts          \015\012",
" \015\012",
"J2-4 (+50V out) = sabc.d Volts        \015\012",
" \015\012",
"J2-6 (-50V out) = sabc.d Volts       \015\012",
" ",
"Mode: Fault Isolation\015\012",
" \015\012",
"1. Verify \"EXT +24\" switch set to \"OFF\".",
"2. Verify power conditioner power switch",
"  is set to \"OFF\".\015\012",
"3. Press switch \"CONTINUE\" when ready.",
"1. Connect the \"J1\" cable to power\015\012",

```

```

" conditioner connector J1.\015\012",
"2. Connect the \"J2\" cable to power\015\012",
" conditioner connector J2.\015\012",
"3. Press switch \"CONTINUE\" when ready.\015\012",
" ",
"1. Apply power to +24V external supply.\015\012",
"2. Verify supply output of +24V.\015\012",
"3. Press switch \"CONTINUE\" when ready.\015\012",
" \015\012",
" \015\012",
" ",
"1. Set \"EXT +24\" switch to \"ON\".\015\012",
"2. Set power conditioner power switch\015\012",
" to \"ON\" (If breaker throws,\015\012",
" terminate the test).\015\012",
"3. Press switch \"CONTINUE\" when ready.\015\012",
" ",
" \015\012",
"Fault in A1 module.\015\012",
" \015\012",
"Replace module and test again.\015\012",
" \015\012",
" ",
" \015\012",
" \015\012",
"Fault in power switch.\015\012",
" \015\012",
"Replace switch and test again\015\012",
" ",
" \015\012",
" \015\012",
"Fault in A2 module.\015\012",
" \015\012",
"Replace module and test again.\015\012",
" ",
" \015\012",
" \015\012",
"Fault in A3 module.\015\012",
" \015\012",
"Replace module and test again.\015\012",
" ",
" \015\012",
" \015\012",
"Fault in A3 module.\015\012",
" \015\012",
"Replace module and test again.\015\012",
" ",
" \015\012",
" \015\012",
"Fault in A4 module.\015\012",

```

```

" \015\012",
"Replace module and test again.\015\012",
" ",
" \015\012",
" \015\012",
"Fault in A5 module.\015\012",
" \015\012",
"Replace module and test again.\015\012",
" ",
" \015\012",
"Fault in filter FL3.\015\012",
"\015\012",
"Replace module and test again.\015\012",
" \015\012",
" ",
" \015\012",
"Fault in filter FL4.\015\012",
" \015\012",
"Replace filter and test again.\015\012",
" \015\012",
" ",
" \015\012",
"Fault in A7 module.\015\012",
" \015\012",
"Replace module and test again.\015\012",
" \015\012",
" ",
" \015\012",
"Fault in FL5 filter.\015\012",
" \015\012",
"Replace filter and test again.\015\012",
" \015\012",
" ",
" \015\012",
"Fault in FL7 filter.\015\012",
" \015\012",
"Replace filter and test again.\015\012",
" \015\012",
" ",
" \015\012",
"No faults located.\015\012",
" \015\012",
" \015\012",
" \015\012",
" ",
"1. Remove the power conditioner side\015\012",
" panel.\015\012",
"2. Connect the power conditioner test\015\012",
" cable.\015\012",
"3. Press switch \"CONTINUE\" when ready\015\012",

```

```

    " ",
    " \015\012",
    " \015\012",
    "Fault in A6 module.\015\012",
    " \015\012",
    "Replace module and test again.\015\012",
    " ",
    "\016"
};
static char *words[] = {
    "\140\121\130\103\104\340\062\125\126\103\340\177",
    "\101\340\062\044\024\004\142\124\123\120\131\340\177",
    "\101\340\062\044\027\142\124\123\120\131\340\177",
    "\101\340\062\045\027\142\124\123\120\131\340\177",
    "\107\111\241\036\240\001\340\177",
    "\107\111\130\134\340\177",
    "\107\111\036\002\340\177",
    "\107\111\036\003\340\177",
    "\107\111\036\004\340\177",
    "\107\111\036\005\340\177",
    "\107\111\036\006\340\177",
    "\107\111\073\114\003\340\177",
    "\107\111\073\114\005\340\177",
    "\107\111\036\007\340\177",
    "\107\111\073\114\006\340\177",
    "\107\111\073\114\007\340\177"
};
main ()
{
    initatod();
    ramcopy();
    if(atod(1)>2457)
        model();
    else
        mode2();
}
scalamod (count,divider,line,posinln)
int count,divider,line,posinln;
{
    int d;
    char itoc();
    long vol,temp,templ,const,con,i;
    if (divider == 4 ) const = 1954;
        else if (divider == 5 ) const = 2442;
        else if (divider == 10) const = 4884;
        else if (divider == 200) const = 97680;
        else const = 0;
    vol = 0;
    templ = (count - 2049);
    for (i=1;i<const;i++)

```

```

        vol = vol + temp1;
if (count < 2049) {
    *(text[line] + posinln - 0x3EF800) = '-';
    vol = - vol;
}
else
    *(text[line] + posinln - 0x3EF800) = '+';
d = 0;
con = 10000000;
temp = con;
while (vol >= temp) {
    d = d + 1;
    temp = temp + con;
}
vol = vol - (temp - con);
*(text[line] + (posinln + 1) - 0x3EF800) = itoc(d);
d = 0;
con = 1000000;
temp = con;
while (vol >= temp) {
    d = d + 1;
    temp = temp + con;
}
vol = vol - (temp - con);
*(text[line] + (posinln + 2) - 0x3EF800) = itoc(d);
d = 0;
con = 100000;
temp = con;
while (vol >= temp) {
    d = d + 1;
    temp = temp + con;
}
vol = vol - (temp - con);
*(text[line] + (posinln + 3) - 0x3EF800) = itoc(d);
d = 0;
con = 10000;
temp = con;
while (vol >= temp) {
    d = d + 1;
    temp = temp + con;
}
*(text[line] + (posinln + 5) - 0x3EF800) = itoc(d);
return;
}
altest ()
{
    int count;
    count = atod(5);
    if((count < P32LOW) || (count > P32HI)){
        count = atod(6);
    }
}

```

```

        if((count<E15LOW)|| (count>E15HI)){
            spk(words[4]);
            dissix(79);
            deadend();
        }
        spk(words[5]);
        dissix(73);
        deadend();
    }
    return;
}
a2test ()
{
    int count;
    count = atod(7);
    if((count<P32LOW)|| (count>P32HI)){
        spk(words[6]);
        dissix(85);
        deadend();
    }
    return;
}
a3test ()
{
    int count,skip;
    skip = 1;
    count = atod(8);
    if((count<P40LOW)|| (count>P40HI)) skip = 0;
    count = atod(9);
    if((count<P36BLOW)|| (count>P36BHI)) skip = 0;
    count = atod(10);
    if((count<P36TLOW)|| (count>P36THI)) skip = 0;
    count = atod(11);
    if((count<P35BLOW)|| (count>P35BHI)) skip = 0;
    count = atod(12);
    if((count<P35TLOW)|| (count>P35THI)) skip = 0;
    if(skip == 0){
        spk(words[7]);
        dissix(97);
        deadend();
    }
    return;
}
a4test ()
{
    int count,skip;
    skip = 1;
    count = atod(13);
    if((count<P53BLOW)|| (count>P53BHI)) skip = 0;
    count = atod(14);

```

```

        if((count<P53TLOW)|| (count>P53THI)) skip = 0;
        count = atod(15);
        if((count<P51BLOW)|| (count>P51BHI)) skip = 0;
        count = atod(16);
        if((count<P51TLOW)|| (count>P51THI)) skip = 0;
        if(skip == 0){
            spk(words[8]);
            dissix(103);
            deadend();
        }
        return;
    }
a5test ()
{
    int count,skip;
    skip = 1;
    count = atod(17);
    if(count<P81LOW) skip = 0;
    count = atod(18);
    if(count<P83LOW) skip = 0;
    count = atod(19);
    if((count<P59LOW)|| (count>P59HI)) skip = 0;
    count = atod(20);
    if((count<P60LOW)|| (count>P60HI)) skip = 0;
    if(skip == 0){
        spk(words[9]);
        dissix(109);
        deadend();
    }
    return;
}
a6test ()
{
    int count;
    count = atod(21);
    if((count<P89LOW)|| (count>P89HI)){
        spk(words[10]);
        dissix(157);
        deadend();
    }
    return;
}
a7test ()
{
    int count,skip;
    skip = 1;
    count = atod(22);
    if((count<P98LOW)|| (count>P98HI)) skip = 0;
    count = atod(23);
    if((count<P98LOW)|| (count>P98HI)) skip = 0;

```

```

        if(skip == 0){
            spk(words[13]);
            dissix(127);
            deadend();
        }
        return;
    }
    fl3test ()
    {
        int count;
        count = atod(2);
        if((count<J22LOW) || (count>J22HIGH)){
            spk(words[11]);
            dissix(115);
            deadend();
        }
        return;
    }
    fl5test ()
    {
        int count;
        count = atod(3);
        if((count<J24LOW) || (count>J24HIGH)){
            spk(words[12]);
            dissix(133);
            deadend();
        }
        return;
    }
    fl7test ()
    {
        int count;
        count = atod(4);
        if((count<J26LOW) || (count>J26HIGH)){
            spk(words[15]);
            dissix(139);
            deadend();
        }
        return;
    }
}
atod (ch)
int ch;
{
    register char *chadr;
    register unsigned badr,lsb,msb;
    badr = 0x47200;
    ch = ch << 1;
    chadr = badr | ch;
    lsb = 0;
    msb = 0;
}

```

```

        msb = *chadr++;
        lsb = *chadr;
        msb = msb << 8;
        msb = msb & 0x0000FFFF;
        lsb = lsb & 0x000000FF;
        msb = msb | lsb;
        return (msb);
    }
cont ()
{
    while((atod(0))>CONTSWT);
    return;
}
deadend ()
{
    int a;
    a = 2;
    while (a!=1);
    return;
}
dissix (firstln)
int firstln;
{
    int i, line;
    disp(text[0]);
    for(i=0;i<16000;i++)
    ;
    disp(text[163]);
    for(i=0;i<6;i++){
        line = (firstln +i);
        disp ((text[line] - 0x3EF800));
    }
    return;
}
disp(s)
register char *s;
{
    register char *pdispl,*pdisp2;
    pdispl = 0x600006;
    pdisp2 = 0x600004;
    while (*s != '\0') {
        while ((~(*pdispl))&0x0004)
        ;
        *pdisp2 = *s++;
    }
    return;
}
fault (line)
int(line);
{

```

```

        *(text[line]+33 - 0x3EF800)='F';
        *(text[line]+34 - 0x3EF800)='A';
        *(text[line]+35 - 0x3EF800)='U';
        *(text[line]+36 - 0x3EF800)='L';
        *(text[line]+37 - 0x3EF800)='T';
        return;
    }
char itoc(temp2)
int temp2;
{
    char c;
    if (temp2 == 0) c = '0';
    else if (temp2 == 1) c = '1';
    else if (temp2 == 2) c = '2';
    else if (temp2 == 3) c = '3';
    else if (temp2 == 4) c = '4';
    else if (temp2 == 5) c = '5';
    else if (temp2 == 6) c = '6';
    else if (temp2 == 7) c = '7';
    else if (temp2 == 8) c = '8';
    else if (temp2 == 9) c = '9';
    else c = 'E';
    return(c);
}
ivtest ()
{
    int count;
    count = atod(24);
    if ((count < J1LOW) || (count > J1HIGH))
    {
        spk(words[0]);
        dissix(31);
        cont();
        dissix(37);
        deadend();
    }
    return;
}
J1test ()
{
    int count;
    while (((count=atod(24))<J1LOW)||((count>J1HIGH))){
        scalamod(count,5,20,6);
        dissix(19);
        cont();
    }
    return;
}
mode2 ()
{

```

```

        dissix(49);
        cont();
        dissix(55);
        cont();
        dissix(151);
        cont();
        dissix(61);
        cont();
        dissix(67);
        cont();
        altest();
        a2test();
        a3test();
        a4test();
        a5test();
        a6test();
        fl3test();
        a7test();
        fl5test();
        fl7test();
        dissix(145);
        deadend();
        return;
    }
model ()
{
    dissix(1);
    cont();
    dissix(7);
    cont();
    dissix(13);
    cont();
    J1test();
    dissix(25);
    cont();
    ivtest();
    ovtest();
    return;
}
ovtest ()
{
    int count;
    count = atod(2);
    scalamod(count,5,43,18);
    if((count<J22LOW) || (count>J22HIGH)){
        spk(words[1]);
        fault(43);
    }
    count = atod(3);
    scalamod(count,10,45,18);
}

```

```

        if((count<J24LOW)|| (count>J24HIGH)) {
            spk(words[2]);
            fault(45);
        }
        count = atod(4);
        scalamod(count,10,47,18);
        if((count<J26LOW)|| (count>J26HIGH)) {
            spk(words[3]);
            fault(47);
        }
        dissix(43);
        deadend();
        return;
    }
    spk(s)
    register char s[];
    {
        register char *pspk;
        pspk = 0x40080;
        while (*s != '\0') {
            while ((*pspk)&0x0002)
                ;
            *pspk = *s++;
        }
        return;
    }
    ramcopy()
    {
        int i;
        long j,k,l;
        for (i=0;i<164;i++){
            j=0;
            do {
                k = j - 0x3EF800;
                *(text[i] + k) = *(text[i] + j);
                l = j;
                j = j + 1;
            }
            while ((*text[i] + l) != '\0');
        }
        return;
    }
    initatod()
    {
        char *ptr;
        int i;
        ptr = 0x47200;
        for (i= 0;i<64;i++)
            *ptr+=0x03;
    }
    return;
}

```

DISTRIBUTION LIST

	<u>No. of Copies</u>
Sperry Corporation Sperry System Management 1112 Church St Huntsville, AL 35801	4
DRCPM-CF	1
DRCPM-HD	1
DRCPM-PE	1
DRCPM-DT	1
DRCPM-VI	1
DRCPM-ROL	1
DRCPM-HEL	1
DRCPM-RS	1
DRCPM-HA	1
DRCPM-LC	1
DRCPM-MP	1
DRSMI-SNX	2
DRSMI-RLD	1
DRSMI-RL	20
DRSMI-RO	1
DRSMI-RA	1
DRSMI-RH	1
DRSMI-RX	1
DRSMI-RR	1
DRSMI-RK	1
DRSMI-RT	1
DRSMI-R	1
DRSMI-RD	1
DRSMI-RE	1
DRSMI-RS	1
DRSMI-RG	1
DDC	1
DRSMI-RPT	1
DRSMI-RPR	15

DISTRIBUTION LIST

	<u>No. of Copies</u>
Sperry Corporation Sperry System Management 1112 Church St Huntsville, AL 35801	4
DRCPM-CF	1
DRCPM-HD	1
DRCPM-PE	1
DRCPM-DT	1
DRCPM-VI	1
DRCPM-ROL	1
DRCPM-HEL	1
DRCPM-RS	1
DRCPM-HA	1
DRCPM-LC	1
DRCPM-MP	1
DRSMI-SNX	2
DRSMI-RLD	1
DRSMI-RL	20
DRSMI-RO	1
DRSMI-RA	1
DRSMI-RH	1
DRSMI-RX	1
DRSMI-RR	1
DRSMI-RK	1
DRSMI-RT	1
DRSMI-R	1
DRSMI-RD	1
DRSMI-RE	1
DRSMI-RS	1
DRSMI-RG	1
DDC	1
DRSMI-RPT	1
DRSMI-RPR	15

END

FILMED

5-83

DTIC