STAND ALONE PRESSURE MEASUREMENT DEVICE (SAPMD) FOR THE SPACE SHUTTLE ORBITER

by

Bill Tomlinson

FINAL REPORT
NASA Contract NAS9-17601
SwRI Project 15-1062

for

National Aeronautics and Space Administration
Lyndon B. Johnson Space Center
Houston, Texas 77058

January 1989
INTRODUCTION

This document presents the final technical report for the development and delivery of a Stand Alone Pressure Measurement Device (SAPMD) and associated ground support equipment. This program was developed for the NASA/Johnson Space Center under NASA contract NAS9-17601 (SwRI Project 15-1062).

The data and documentation contained herein are the results of the development and successful completion of this contract.

Background

This program fulfilled the need to measure pressure at the surface of the thermal protective system tile on the space shuttle Orbiter during ascent, and in order to avoid the extensive impact associated with wiring the measurement into the Orbiter data system, the measurement device must be completely stand-alone and incorporate its own power supply and data recording facility. The device must be small enough to be mounted under the thermal protection system tiles and must be rugged enough to withstand the environments it will encounter at the bond line of the tiles throughout an Orbiter mission. It must be failsafe and data recorded during ascent must be recoverable after the mission without removal of the device.

Specifications

The SAPMD shall measure ambient pressure at the surface of the Orbiter TPS in the range of 0-15 pounds per square inch absolute (PSIA). Measurement will begin at solid rocket booster (SRB) ignition as sensed by appropriate vibration sensing elements in the SAPMD. Pressure and corresponding real-time data are to be recorded every one tenth second for 140 seconds and at the end of the recording period, the operation will be discontinued with the data preserved for interrogation subsequent to Orbiter re-entry and landing.

The type and size of the battery shall be such as to allow the vibration sensing elements and a real-time clock to be initialized a minimum of 30 days prior to launch and still provide power as necessary to perform the 140 second data recording period after SRB ignition. Battery installation shall be in such a manner as to allow battery replacement without removing the SAPMD from its position or removing more than one TPS tile.

The SAPMD must be mounted in specific locations under tiles of the Orbiter TPS. To accommodate such mounting, the absolute maximum physical dimensions must not exceed 6.0 inches in length, 1.5 inches in width and 0.4 inches in height, and the device shall be of such configuration that it can be bonded to the Orbiter skin at the joint line of two TPS tiles with the pressure sensing port at the surface of the tile. The SAPMD must remain operational in the temperature range of -40 to +85°C and survive storage temperatures of -55 to +125°C. The pressure port must withstand 934°C without causing damage to the TPS during entry and must remain functional at 262°C during ascent.

The accuracy of the pressure measurement must be plus or minus one-half pound per square inch absolute over a temperature range of 0 to +36°C.

Conclusion

All of the above specifications have been met and verified by prototype testing and is documented in the enclosed test data.

Four flight-qualified models were fabricated and of these, two have been delivered and successfully flown in the cargo bay of STS-26.

A contract modification changed the delivery of four flight models to two while modifying the remaining two for use in the nozzle bearing area of the SRB during a ground test at the Morton Thiokol site in Utah.
INDEX

CEI Specification
"Flight Hazard Evaluation of the Lithium Thionyl Chloride Cell"
SAPMD Schematics
Test Data, Prototype
SAPMD Assembly Drawing
Flight Software Summary
SGA Software Summary
HP.SAPMD CEI Specification
HP.SAPMD Electronic Schematics
HP.SAPMD Mechanical Schematics
HP.SAPMD GSE Software Listings
HP/SGA "Read Me"
SAPMD Acceptance Test Procedure
CEI SPECIFICATION
CONTRACT END ITEM SPECIFICATION

PERFORMANCE/DESIGN

AND

PROJECT CONFIGURATION

REQUIREMENTS

SE-176TA

STAND-ALONE PRESSURE MEASUREMENT DEVICE

FOR THE SPACE SHUTTLE ORBITER

CONTRACT NUMBER NAS 9-17601

Approved by: Rex R. Ritz
JSC Contracting Officer
Houston, Texas

Approved by: William C. Gibson
Southwest Research Institute
6220 Culebra Road
San Antonio, Texas

Approval date: 20 Oct 87
<table>
<thead>
<tr>
<th>REV</th>
<th>SCN. NUMBER</th>
<th>PAGES AFFECTED</th>
<th>PARAGRAPHS AFFECTED</th>
<th>DATE</th>
<th>APPROVAL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

11
<table>
<thead>
<tr>
<th>Paragraph</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>INTRODUCTION</td>
<td>1-1</td>
</tr>
<tr>
<td>1.1</td>
<td>Scope</td>
<td>1-1</td>
</tr>
<tr>
<td>1.2</td>
<td>Engineering Baseline</td>
<td>1-1</td>
</tr>
<tr>
<td>2.0</td>
<td>APPLICABLE DOCUMENTS</td>
<td>2-1</td>
</tr>
<tr>
<td>3.0</td>
<td>TECHNICAL REQUIREMENTS</td>
<td>3-1</td>
</tr>
<tr>
<td>3.1</td>
<td>Performance</td>
<td>3-1</td>
</tr>
<tr>
<td>3.2</td>
<td>Product Configuration</td>
<td>3-1</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Manufacturing Drawings</td>
<td>3-1</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Government Furnished Property List</td>
<td>3-1</td>
</tr>
<tr>
<td>3.2.3</td>
<td>Standards of Manufacturing, Manufacturing Processes, and Production</td>
<td>3-4</td>
</tr>
<tr>
<td>4.0</td>
<td>QUALITY ASSURANCE</td>
<td>4-1</td>
</tr>
<tr>
<td>4.1</td>
<td>Quality Requirements</td>
<td>4-1</td>
</tr>
<tr>
<td>4.1.1</td>
<td>Applicability of NHB 5300.4 (1D2)</td>
<td>4-1</td>
</tr>
<tr>
<td>4.1.2</td>
<td>Applicability of NHB 5300.4 (3A-1)</td>
<td>4-1</td>
</tr>
<tr>
<td>4.1.3</td>
<td>Drawing Compliance</td>
<td>4-1</td>
</tr>
<tr>
<td>4.1.4</td>
<td>Additional Requirements</td>
<td>4-1</td>
</tr>
<tr>
<td>4.2</td>
<td>Reliability Requirements</td>
<td>4-1</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Additional Requirements</td>
<td>4-1</td>
</tr>
<tr>
<td>4.3</td>
<td>Test Requirements</td>
<td>4-1</td>
</tr>
<tr>
<td>5.0</td>
<td>PREPARATION FOR DELIVERY</td>
<td>5-1</td>
</tr>
<tr>
<td>5.1</td>
<td>Containers</td>
<td>5-1</td>
</tr>
<tr>
<td>5.2</td>
<td>Marking</td>
<td>5-1</td>
</tr>
<tr>
<td>6.0</td>
<td>NOTES</td>
<td>6-1</td>
</tr>
<tr>
<td>6.1</td>
<td>Intended Use</td>
<td>6-1</td>
</tr>
<tr>
<td>6.2</td>
<td>Ordering Data</td>
<td>6-1</td>
</tr>
<tr>
<td>6.3</td>
<td>Definitions</td>
<td>6-1</td>
</tr>
</tbody>
</table>
1.0 INTRODUCTION

1.1 Scope

This specification establishes the requirements for complete identification and acceptance of a Stand-Alone Pressure Measurement Device (SAPMD) for the Space Shuttle Orbiter to be formally accepted by the Manned Spacecraft Center (MSC).

1.2 Engineering Baseline

The engineering baseline shall be established by a Critical Design Review (CDR) for this Contract End Item (CEI). All units of this CEI, regardless of intended use, shall be manufactured and accepted to the configuration defined by this specification and formally approved Engineering Change Proposals (ECP's)/Specification Change Notices (SCN's).
2.0  APPPLICABLE DOCUMENTS

The following documents of the exact issue shown form a part of this specification to the extent specified herein. In the event of conflict between this specification and documents referenced herein, this specification shall take precedence.

Specifications-JSC

NHB 5300.4 (3A-1)  Requirements for Soldering of Electrical Connections
NHB 5300.4 (1D2)  Safety, Reliability, Maintainability, and Quality Provisions for the Space Shuttle Program
NHB 8060.1B  Flammability, Odor, and Offgassing Requirements
NHB 5300.4 (IC)  Inspection System Provisions
JSC 07700, Vol. IV  Configuration Management
JSCM 8080  Criteria and Standards
JSC-09604B  JSC GFE Materials Selection List and Materials Documentation Procedures
JSC-SE-R-0006B  NASA/JSC Materials and Processes
JSC 17481  JSC Safety Guidelines Document for Space Shuttle GFE
JSC-SP-T-0023B  Specification, Environmental Acceptance Testing
JSC/MSC-SPEC-M-1A  Marking and Identification
JSC SW-E-0002  Space Shuttle Program GSE General Design Requirements

Specifications-Rockwell

MF-0004-002B  Electrical Design Requirements for Electrical Equipment Utilized on the Space Shuttle Vehicle

Standards-Military

MIL-STD-975E  NASA Standard (EEE) Parts List
3.0 TECHNICAL REQUIREMENTS

3.1 Performance

The Stand Alone Pressure Measure Device (SAPMD) shall measure ambient pressure at the surface of the Orbiter TPS. The measurement range shall be 0 - 15 psia. The measurements shall begin at solid rocket booster (SRB) ignition as sensed by appropriate vibration sensors located within the enclosure incorporating the battery and electronics. Upon sensing SRB ignition, the SAPMD will monitor and record pressure for 140 seconds to a solid state non-volatile memory storage device. At the end of the recording period, the operation will be discontinued with the data preserved for interrogation subsequent to Orbiter entry and landing.

The SAPMD shall have a means to accurately time tag the recorded data in units of 1/2 seconds since January 1. The timekeeping and vibration sensor circuit shall be initialized 30 days before launch. The battery capacity shall be such that this timekeeping can be continued for a minimum of 50 days.

The block diagram shown in Figure 3-1 depicts the method in which the SAPMD shall process and record the pressure and time data. The heart of the system will be an INTEL 80C31, 8-bit CMOS processor with the program in electrically erasable programmable prom and the memory device shall be a 64K CMOS electrically erasable prom capable of 10-year data retention.

The battery supply shall be two each 600 mAH Lithium Thionyl Chloride batteries in a removable battery holder.

Data retrieval shall be accomplished with a battery-powered 80C88-based computer. Communication with the SAPMD shall be serial with additional connector pins to provide auxiliary power to the SAPMD.

The SAPMD shall be fabricated to meet the environmental conditions as specified in paragraphs 3.5.1 and 3.5.2 of the contract specification.

3.2 Product Configuration

Figure 3-2 Top Assembly Drawing.

3.2.1 Manufacturing Drawings

The configuration of the SAPMD shall be in accordance with drawing number 15-1062-457, and drawings and engineering data assembled thereunder, including all approved changes thereto. Class II changes to manufacturing drawings are allowable without NASA approval, however they are subject to classification review by NASA.

3.2.2 Government Furnished Property List

NONE
FIGURE 3-1. REVISED SAPMB BLOCK DIAGRAM
FIGURE 3-2 TOP ASSEMBLY DRAWING

- Battery Module
- Ceramic Substrate
- Pressure Tube
- Connector
- Pressure Transducer
- Sensor Electronics
- ADC0802
- Motion Transducer
- Power Switch and Time Keeping Electronics
- 28C64 EE PROM
- 54HC138
- 27C64 E PROM
- 54HC373
- 54HC11
- 80C31 CPU
3.2.3 Standards of Manufacturing, Manufacturing Processes, and Production

The applicability of the following publications to the SAPMD may be revised only by engineering changes having prior approval of NASA.

<table>
<thead>
<tr>
<th>MIL-STD-975F</th>
<th>NASA Standard Electrical, Electronic, and Electromechanical (EEE) Parts List</th>
</tr>
</thead>
</table>

Specifications—Military
None

Specifications—NASA

<table>
<thead>
<tr>
<th>JSC/MSFC-SPEC-M-1A</th>
<th>Marking and Identification</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSC-SE-R-0006B</td>
<td>NASA/JSC Requirements for Materials and Processes</td>
</tr>
<tr>
<td>JSC-SP-T-0023B</td>
<td>Specification, Environmental Acceptance Testing</td>
</tr>
<tr>
<td>JSC SW-E-0002, Rev. B</td>
<td>Space Shuttle Program GSE General Design Requirements</td>
</tr>
</tbody>
</table>

Documents—NASA

<table>
<thead>
<tr>
<th>JSC 07 700, Vol. IV Rev. B</th>
<th>Space Shuttle Program Configuration Management Requirements (with changes through No. 60)</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSCM 8080</td>
<td>Manned Spacecraft Criteria and Standards</td>
</tr>
<tr>
<td>JSC-09604B</td>
<td>JSC GFE Materials Selection List and Materials Documentation Procedures</td>
</tr>
<tr>
<td>JSC 17481A</td>
<td>Safety Requirements Document for JSC Space Shuttle Flight Equipment</td>
</tr>
<tr>
<td>NHB 5300.4(3A-1)</td>
<td>Requirements for Soldering Electrical Connections</td>
</tr>
<tr>
<td>NHB 5300.4(1D2)</td>
<td>Safety, Reliability, Maintainability, and Quality Provisions for the Space Shuttle Program</td>
</tr>
</tbody>
</table>
NHB 8060.1B

Flammability, Odor, and Offgassing Requirements and Test Procedures for Materials, in Environments that Support Combustion

NHB 5300.4(1C)

Inspection System Provisions for Aeronautical and Space System Materials, Parts, Components and Services

Other Standards/Documents

Rockwell

MF-0004-002B

Electrical Design Requirements for Electrical Equipment Utilized on the Space Shuttle Vehicle
4.0 QUALITY ASSURANCE

Southwest Research Institute is responsible for accomplishment of each verification required herein.

4.1 Quality Requirements

4.1.1 Applicability of NHB 5300.4 (ID2)
Paragraphs 1D200 and 1D301.6.

4.1.2 Applicability of NHB 5300.4 (3A-1)
A) Chapter 2, all paragraphs.
B) Chapter 3, all paragraphs.
C) Chapter 4, all but paragraphs 3A401, and 3A502.
D) Chapter 8, all paragraphs.

4.1.3 Drawing Compliance

Written verification that the SAPMD has been fabricated, inspected, and tested to the latest applicable drawings identified in 3.2.1 and has incorporated the GFP specified in 3.2.2 will be provided at each Acceptance Review.

4.1.4 Additional Requirements
Paragraph 5.1.3, JSC document 20793.

4.2 Reliability Requirements
A) Design per document JSCM 8080.
B) Design Review (PDR and CDR).
C) Limited life items identification per SwRI document 1062-LL-01.
D) EEE parts per Mil-Std-975F (where possible).
E) Derating per Mil-Std-975F, appendix A.

4.2.1 Additional Requirements
None

4.3 Test Requirements

Per contract NAS9-17601, latest revision.
5.0 PREPARATION FOR DELIVERY

5.1 Containers

Unless otherwise specified, the preservation, packaging, and packing shall be equivalent to the contractor's best commercial practice, provided that this practice will be sufficient to protect the SAPMD against damage during shipment. Exterior containers shall conform to Uniform Freight Classification Rules for rail shipment or National Motor Freight Classification Rules for truck shipment, as applicable.

5.2 Marking

Interior and exterior containers shall be marked in accordance with MIL-STD-129 "Marking for Shipment and Storage".
6.0 NOTES

6.1 Intended Use

The SAPMD, part number 15-1062-900-01, is intended for use in the measurement of ambient air pressure and the recording of that data in the vicinity of the Space Shuttle Orbiter exterior surfaces. Data thus acquired will be transferred to a portable computer system post flight for analysis and archiving.

6.2 Ordering Data

Procurement documents shall specify:


(b) Special precautions shall be applied to control of electrostatic discharge during all stages of parts procurement, storage, fabrication and test.

6.3 Definitions

A) SAPMD - Stand Alone Pressure Monitor Device

NOTICE: When MSC drawings, specification, or other data are used for any purpose other than in connection with a definitely related MSC procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever and the fact that MSC may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell, any patented invention that may be in any way related thereto.
FLIGHT HAZARD EVALUATION OF THE LITHIUM THIONYL CHLORIDE CELL
FLIGHT HAZARD EVALUATION
OF THE
LITHIUM THIONYL CHLORIDE CELL
PURPOSE OF EVALUATIONS

* Temperature Vacuum Test
  * Loss of Hermeticity of Package and Temperature at Which That Loss Occurred
* Electromechanical Failure
  * Degradation of the Cell's Ability to Supply Power and Temperature at Which That Degradation Occurred
  * Qualitative Rate of Failure Over Time
Temperature Vacuum Test Results

- Temperature Risk of Less Than 5°C/min. Never Caused Violent Rupture of Case
- Cell Continued to Produce Usable Power Even After Encapsulant Failure
Purpose of Evaluations

- Short Circuit Test
  - Time Rate of Case Temperature Change
  - Maximum Short Circuit Current
  - The Degradation of the Cell's Ability to Supply Power
Short Circuit Test Results

* Case Temperature Could Exceed 100°C With No Visible Damage To Case and No Loss of Encapsulant Integrity

* Output Current Could Exceed 1.0 Ampere and Cell Could Still Produce Usable Power After Test
SAPMD SCHEMATICS
TEST DATA, PROTOTYPE
PROTOTYPE SAPMD TEST RESULTS

Initial Tests Conducted 3 March 1987 @ NASA Dryden Flight Research Center

- Results Unacceptable
  - Very High Zero Drift w/Temperature
  - High Pressure Transducer Drift w/Temperature

Unit Returned to SwRI for Repair/Calibration

- Error Sources Analyzed
  - High Zero Drift w/Temperature from Pressure Transducer
  - Large Error Resulting in Temperature Drift of Voltage Regulator

Prototype Modified and Recalibrated

- Pressure Transducer Replaced w/Better Performing Unit
- Current Limit of Voltage Regulator Raised
- Thermistor Inserted in Series w/PRESSure Transducer
- Extensive Calibration Performed Prior to Return to JSC

Prototype Returned to JSC and Recalibrated on 6 July 1987

- SwRI Informed JSC Accepts Repaired Unit
### TOP TEMPERATURE

<table>
<thead>
<tr>
<th>TEMP</th>
<th>PRESS DECREASING</th>
<th>PRESS INCREASING</th>
</tr>
</thead>
<tbody>
<tr>
<td>14.7</td>
<td>13.71</td>
<td>13.00</td>
</tr>
<tr>
<td>14</td>
<td>11.01</td>
<td>11.01</td>
</tr>
<tr>
<td>10</td>
<td>9.02</td>
<td>8.95</td>
</tr>
<tr>
<td>8</td>
<td>7.05</td>
<td>6.96</td>
</tr>
<tr>
<td>6</td>
<td>5.05</td>
<td>5.05</td>
</tr>
<tr>
<td>4</td>
<td>3.06</td>
<td>3.06</td>
</tr>
<tr>
<td>2</td>
<td>1.14</td>
<td>1.14</td>
</tr>
<tr>
<td>0.03</td>
<td>0.00</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TEMP</th>
<th>PRESS DECREASING</th>
<th>PRESS INCREASING</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>12.2</td>
<td>12.7</td>
</tr>
<tr>
<td>10</td>
<td>10.16</td>
<td>10.16</td>
</tr>
<tr>
<td>8</td>
<td>8.17</td>
<td>8.17</td>
</tr>
<tr>
<td>6</td>
<td>6.11</td>
<td>6.04</td>
</tr>
<tr>
<td>4</td>
<td>4.05</td>
<td>3.98</td>
</tr>
<tr>
<td>2</td>
<td>2.13</td>
<td>1.92</td>
</tr>
<tr>
<td>0.03</td>
<td>0.00</td>
<td></td>
</tr>
</tbody>
</table>

### TEMPERATURE INCREASING

<table>
<thead>
<tr>
<th>TEMP</th>
<th>PRESS DECREASING</th>
<th>PRESS INCREASING</th>
</tr>
</thead>
<tbody>
<tr>
<td>14.7</td>
<td>14.45</td>
<td>14.45</td>
</tr>
<tr>
<td>14</td>
<td>12.36</td>
<td>12.22</td>
</tr>
<tr>
<td>10</td>
<td>10.16</td>
<td>10.16</td>
</tr>
<tr>
<td>8</td>
<td>8.17</td>
<td>8.17</td>
</tr>
<tr>
<td>6</td>
<td>6.11</td>
<td>6.04</td>
</tr>
<tr>
<td>4</td>
<td>4.05</td>
<td>3.98</td>
</tr>
<tr>
<td>2</td>
<td>2.13</td>
<td>1.92</td>
</tr>
<tr>
<td>0.03</td>
<td>0.00</td>
<td></td>
</tr>
</tbody>
</table>

### INITIAL PRESSURE DECREASING

<table>
<thead>
<tr>
<th>TEMP</th>
<th>PRESS DECREASING</th>
<th>PRESS INCREASING</th>
</tr>
</thead>
<tbody>
<tr>
<td>14.7</td>
<td>15.49</td>
<td>14.57</td>
</tr>
<tr>
<td>14</td>
<td>14.37</td>
<td>14.37</td>
</tr>
<tr>
<td>12</td>
<td>12.44</td>
<td>12.51</td>
</tr>
<tr>
<td>10</td>
<td>10.45</td>
<td>10.37</td>
</tr>
<tr>
<td>8</td>
<td>8.31</td>
<td>8.31</td>
</tr>
<tr>
<td>6</td>
<td>6.25</td>
<td>6.32</td>
</tr>
<tr>
<td>4</td>
<td>4.12</td>
<td>4.12</td>
</tr>
<tr>
<td>2</td>
<td>2.06</td>
<td>2.06</td>
</tr>
<tr>
<td>0.03</td>
<td>0.00</td>
<td></td>
</tr>
</tbody>
</table>

### INITIAL PRESSURE INCREASING

<table>
<thead>
<tr>
<th>TEMP</th>
<th>PRESS DECREASING</th>
<th>PRESS INCREASING</th>
</tr>
</thead>
<tbody>
<tr>
<td>76.7</td>
<td>15.35</td>
<td>14.64</td>
</tr>
<tr>
<td>73.3</td>
<td>14.64</td>
<td>14.64</td>
</tr>
<tr>
<td>73.3</td>
<td>12.58</td>
<td>12.51</td>
</tr>
<tr>
<td>74.0</td>
<td>10.45</td>
<td>10.37</td>
</tr>
<tr>
<td>73.3</td>
<td>8.39</td>
<td>8.39</td>
</tr>
<tr>
<td>73.9</td>
<td>6.25</td>
<td>6.25</td>
</tr>
<tr>
<td>75.3</td>
<td>4.12</td>
<td>4.12</td>
</tr>
<tr>
<td>74.6</td>
<td>2.13</td>
<td>2.13</td>
</tr>
<tr>
<td>75.3</td>
<td>0.00</td>
<td></td>
</tr>
</tbody>
</table>

### TEMPERATURE DECREASING

<table>
<thead>
<tr>
<th>TEMP</th>
<th>PRESS DECREASING</th>
<th>PRESS INCREASING</th>
</tr>
</thead>
<tbody>
<tr>
<td>14.7</td>
<td>15.35</td>
<td>14.85</td>
</tr>
<tr>
<td>14</td>
<td>14.78</td>
<td>14.85</td>
</tr>
<tr>
<td>12</td>
<td>12.85</td>
<td>12.79</td>
</tr>
<tr>
<td>10</td>
<td>10.52</td>
<td>10.66</td>
</tr>
<tr>
<td>8</td>
<td>8.60</td>
<td>8.53</td>
</tr>
<tr>
<td>6</td>
<td>6.40</td>
<td>6.47</td>
</tr>
<tr>
<td>4</td>
<td>4.37</td>
<td>4.41</td>
</tr>
<tr>
<td>2</td>
<td>2.27</td>
<td>2.27</td>
</tr>
<tr>
<td>0.03</td>
<td>0.21</td>
<td></td>
</tr>
</tbody>
</table>

### TEMPERATURE INCREASING

<table>
<thead>
<tr>
<th>TEMP</th>
<th>PRESS DECREASING</th>
<th>PRESS INCREASING</th>
</tr>
</thead>
<tbody>
<tr>
<td>14.7</td>
<td>15.63</td>
<td>15.63</td>
</tr>
<tr>
<td>14</td>
<td>14.92</td>
<td>13.6</td>
</tr>
<tr>
<td>12</td>
<td>12.79</td>
<td>12.79</td>
</tr>
<tr>
<td>10</td>
<td>10.66</td>
<td>10.66</td>
</tr>
<tr>
<td>8</td>
<td>8.60</td>
<td>8.53</td>
</tr>
<tr>
<td>6</td>
<td>6.54</td>
<td>6.54</td>
</tr>
<tr>
<td>4</td>
<td>4.41</td>
<td>4.41</td>
</tr>
<tr>
<td>2</td>
<td>2.34</td>
<td>2.34</td>
</tr>
<tr>
<td>0.03</td>
<td>0.28</td>
<td></td>
</tr>
</tbody>
</table>

---

RAW DATA, SAPMOD MODIFIED PROTOTYPE PERFORMANCE AT JSC JULY 6-8, 1987
### 0°F TEMPERATURE INCREASING

<table>
<thead>
<tr>
<th>TEMP</th>
<th>PRESS DECREASING</th>
<th>TEMP</th>
<th>PRESS INCREASING</th>
<th>TEMP</th>
<th>PRESS DECREASING</th>
<th>TEMP</th>
<th>PRESS INCREASING</th>
</tr>
</thead>
<tbody>
<tr>
<td>14.7</td>
<td>1.5</td>
<td>14.7</td>
<td>-2.0</td>
<td>15.6</td>
<td>0</td>
<td>14.8</td>
<td>4</td>
</tr>
<tr>
<td>14</td>
<td>-1.9</td>
<td>14.7</td>
<td>-2.0</td>
<td>15.6</td>
<td>0</td>
<td>14.8</td>
<td>4</td>
</tr>
<tr>
<td>12</td>
<td>-1.9</td>
<td>12.7</td>
<td>-1.0</td>
<td>12.8</td>
<td>4</td>
<td>12.8</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>-0.3</td>
<td>10.6</td>
<td>-2.0</td>
<td>10.8</td>
<td>4</td>
<td>10.8</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>0.4</td>
<td>8.6</td>
<td>-2.0</td>
<td>8.7</td>
<td>4</td>
<td>8.7</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>-0.3</td>
<td>6.5</td>
<td>-1.0</td>
<td>6.6</td>
<td>0</td>
<td>6.6</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>-1.5</td>
<td>4.5</td>
<td>0</td>
<td>4.6</td>
<td>-2</td>
<td>4.6</td>
<td>-2</td>
</tr>
<tr>
<td>2</td>
<td>1.3</td>
<td>2.4</td>
<td>-0.6</td>
<td>2.6</td>
<td>-2</td>
<td>2.6</td>
<td>-2</td>
</tr>
<tr>
<td>0.03</td>
<td>1.0</td>
<td>0.43</td>
<td>-1.0</td>
<td>0.64</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 0°F BEFORE COLD SOAK

<table>
<thead>
<tr>
<th>TEMP</th>
<th>PRESS DECREASING</th>
<th>TEMP</th>
<th>PRESS INCREASING</th>
</tr>
</thead>
<tbody>
<tr>
<td>14.7</td>
<td>-41.2</td>
<td>15.1</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>-41.3</td>
<td>14.6</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>-41.5</td>
<td>12.5</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>-39.2</td>
<td>10.6</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>-38.5</td>
<td>8.6</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>-41.6</td>
<td>6.6</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>-40.0</td>
<td>4.6</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>-42.0</td>
<td>2.7</td>
<td></td>
</tr>
<tr>
<td>0.03</td>
<td>-40.0</td>
<td>0.64</td>
<td></td>
</tr>
</tbody>
</table>
SAPMD ASSEMBLY DRAWING
PRESSURE TRANSDUCER

- 6-32 UNF THREAD
- .37
- .30 MAX.
- .50
- .35

- ENTRAN EPI-080 TYPE PRESSURE TRANSDUCER
- CUSTOM PACKAGE
- INTERNAL COMPENSATION
- SILICON DIAPHRAM
- EXCITATION 5VDC
- STAINLESS STEEL HOUSING
PRESSURE TRANSDUCER LEADS OUT THIS Ø .040 HOLE

HOUSING 2024-T3 ALUM.

PRESSURE TRANSODUCER Ø .080

POTTED TRANSODUCER COMPENSATION ELECTRONICS

PRESSURE TUBE MOUNT 6-32 EXTERNAL THREADS 17-4 PH STAINLESS
FLIGHT SOFTWARE SUMMARY
SUMMARY OF
SAPMD FLIGHT SOFTWARE FUNCTIONS

* Detect Launch
* Log Pressure Data
* Hardware Self-Test
* Interact with GSE
SAPMD FLIGHT SOFTWARE MODULES

* INIT - Check for Power-On or Tick Reset, Arm Watchdog Timer
* TICK - Update Internal Clocks
* Launch - Detect Launch and Record Pressure Sample
* GSE - Communicate with SAPMD GSE
* TIMEOUT - Suspend Operation of SAPMD Pending Reset
* STATUS - Transmit Current SAPMD Status Over Serial Line
* HANG - Enter 80C51 Power-Down Mode
SAPMD FLIGHT SOFTWARE EXECUTION LEVELS

* Critical Functions (High Priority)

Must Execute to Completion; Cannot be Suspended and Resumed or Interrupted by Reset.

- INIT: Reset Test (Check Memory Bit Pattern)
- TICK: Update Clocks
- TIMEOUT: Suspend Operation

* Control Functions (Low Priority)

May be Interrupted by TIMEOUT, Suspended and Resumed

- LAUNCH: Detect Launch, Record Pressure Sample
- GSE: Communicate with GSE
- STATUS: Transmit SAPMD Status
CONFIGURING SAPMD FLIGHT SOFTWARE

* SAPMD Operating Parameters in RAM/EEPROM may be Altered Using GSE

<table>
<thead>
<tr>
<th>EEPROM Resident Parameters</th>
<th>RAM Resident Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td># Pressure Samples Recorded</td>
<td>Branch Points - Bail Out for S/W</td>
</tr>
<tr>
<td>First Free EEPROM Address</td>
<td>Clocks</td>
</tr>
<tr>
<td>EEPROM Size</td>
<td>Timeout Duration</td>
</tr>
<tr>
<td>SAPMD Serial Number</td>
<td>EEPROM Power Switch</td>
</tr>
<tr>
<td></td>
<td>80C51 Special Function Resistor (SFR) Images</td>
</tr>
<tr>
<td></td>
<td>Launch Detect Counter</td>
</tr>
</tbody>
</table>
SAPMD SELF-TESTS

• EEPROM Self-Test
  • Address Test
  • ALL 55H
  • ALL AAH
  • ALL FFH
  • ALL 00H

• Power System Test

• A/D Converter Test

• Pressure Transducer Test

• 80C51 RAM Test - Forces Power-On Reset

• 80C51 ROM Test
  • Compute and Compare with Recorded Checksum
PRESSURE DATA RECORDING FORMAT

* Up to 3 Acquisition Cycles in 2K EEPROM
* Each Sample Numbered
* Pressure File Format:

<table>
<thead>
<tr>
<th>Byte</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-3</td>
<td>GMT at Launch Detect</td>
</tr>
<tr>
<td>4-563</td>
<td>Pressure Samples</td>
</tr>
</tbody>
</table>

* Pressure Sample Format:

<table>
<thead>
<tr>
<th>Byte</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Sequence # Mod 256</td>
</tr>
<tr>
<td>1</td>
<td>Pressure Transducer Reading</td>
</tr>
</tbody>
</table>
SAPMD STATUS INFORMATION

* Transmitted Every Tick

* Status Information:
  * EEPROM Powered
  * Self-Test in Progress
  * GSE Transaction in Progress
  * Data Acquisition in Progress
  * Acquisition Complete
  * Error

* RAM Dump:
  * RAM Block may be Dumped with Status to Monitor SAPMD Operation

* Status Information Displayed on GSE (GRID)
SGA SOFTWARE SUMMARY
SUMMARY OF
SAPMD GSE SHUTTLE GAUGE ACCESS (SGA) SOFTWARE

* Menu Driven
* Main Menu
* 2 Branch Menus
* Allows Interactive Access to SAPMD
* Provides SAPMD Calibration Offsets
SGA MAIN MENU

1. COMMAND/INTERROGATE SAPMD
2. SAPMD SELF-TEST
3. RECOVER PRESSURE DATA FILENAME
4. DISPLAY PRESSURE DATA FILENAME
5. PRINT PRESSURE DATA FILENAME

SELECT OPTION:

* Options 1 and 2 use Branch Menus
OPTION 1: COMMAND/INTERROGATE SAPMD MENU

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SG ddd/hh:mm:ss</td>
<td>Set GMT</td>
</tr>
<tr>
<td>SM ddd/hh:mm:ss</td>
<td>Set MET</td>
</tr>
<tr>
<td>TM</td>
<td>READ Clocks</td>
</tr>
<tr>
<td>DR xx[,yy]</td>
<td>DUMP 80C51 RAM From xx for yy Bytes</td>
</tr>
<tr>
<td>DE xxxx[,yy]</td>
<td>DUMP 80C51 External Memory xxxx for yy</td>
</tr>
<tr>
<td>DS xx</td>
<td>DUMP 80C51 SFR</td>
</tr>
<tr>
<td>ER xx</td>
<td>ENTER 80C51 RAM at xx</td>
</tr>
<tr>
<td>EE xxxx</td>
<td>ENTER 80C51 External Memory at xxxx</td>
</tr>
<tr>
<td>ES xx</td>
<td>ENTER 80C51 SFR</td>
</tr>
<tr>
<td>P Filename</td>
<td>PROGRAM Filename into EEPROM</td>
</tr>
<tr>
<td>MON</td>
<td>TOGGLE Monitor Data Window Display</td>
</tr>
</tbody>
</table>

> *

Commands are Entered Following "->" and Scroll Beneath Menu
OPTION 2: SAPMD SELF-TEST MENU

1. PERFORM ALL TESTS
2. EEPROM TEST
3. POWER SYSTEM TEST
4. A/D CONVERTER TEST
5. PRESSURE TRANSDUCER TEST
6. 80C51 RAM TEST
7. 80C51 ROM TEST

SELECT OPTION:

- Test Executed and Completion Status Displayed by SGA
SAPMD STATUS DISPLAY

- Status Line:
  - EEPROM-On Self-Test GSE Acquisition Complete Error
  - Displayed on Top Line of All Menu Screens
  - Active Functions Indicated by Flashing Reverse Video
  - Dumped RAM Displayed on Dedicated Window Which Replaces Command/Interrogate Menu
  - "MON" Command Alternately Displays Monitored RAM Window or Command Menu
SAPMD<-->SGA DIALOG

• All Commands and Responses ASCII
• Command Character Reception Acknowledged by Echo
• SGA or SAPMD May Abort Command at Any Time
• All Commands Generate Completion Response
• Status and Dumped RAM Data are Binary, Distinguished by Set High Order Bit of First Message Byte
SAPMD CALIBRATION

- Calibration Information
  - SAPMD Serial Number
  - Pressure Transducer Zero Bias
    - Number of Counts to Subtract from Each Sample to Adjust for Transducer Error
  - PSI/Count for Transducer Samples
- File is "SAPMD.CAL"
HP.SAPMD CEI SPECIFICATION
1.0 INTRODUCTION

This specification establishes the requirements specified in NAS9-17601, Request for Engineering Change Proposal dated April 19, 1988 for the design, development, fabrication, testing and delivery of two (2) Stand-Alone Pressure Measurement Devices (SAPMD) based on the design of the existing SAPMDs for the Shuttle Orbiter. The revised design will be capable of operation in a 1000 psi, 187°F environment for use in the Solid Rocket Booster (SRB) tests. The main tasks of this new work is the design of a high-pressure housing, substitution of a high-pressure sensor and design of a longer life battery supply.

1.1 Background

The present design of the SAPMD incorporates a microprocessor system implemented with hybrid module techniques using low power CMOS units which are contained in a 0.4-in. thick metal housing. The system is designed to operate installed under selected heat shield tiles on the Shuttle Orbiter and to thus survive and operate in a pressure regime from near zero psi to atmospheric pressure (14.7 psi). The unit operates from self-contained lithium batteries which provide an operating lifetime in the sleep mode of 1200 hours. The system is awakened by sensing the vibration of launch and takes pressure data for a period of 140 seconds with data readings every 100 ms. On-board real-time clock data are recorded with the pressure data. The data are recorded in EEPROMs which are capable of retaining the data indefinitely at temperatures up to 257°F. After retrieval of the module, data are supplied to GSE equipment for further use.

1.2 Specification Changes

In order to meet the requirements imposed by the SRB tests, the SAPMD must be redesigned to accommodate the new test environment. First, the housing of the SAPMD must be redesigned to survive a pressure of 1000 psi with at least a 50 percent overpressure capability. Protection of the internal electronic circuits from any mechanical stress is important for both reliability and accuracy. Second, a new pressure sensor must be selected which can measure pressures from 14.7 to 1000 psi with an accuracy of one percent FS over an ambient temperature range of 100 to 300°F. Third, the battery power supply must be modified to provide a lifetime of 2400 hours (100 days) of power-down operation. Forth, a new circuit must be added to allow an external hard-wired control line to activate the system prior to ignition. This circuit replaces the vibration sensor used to detect launch.
2.0 TECHNICAL APPROACH

The following paragraphs describe the technical work on the three tasks required to modify the SAPMD design for high pressure measurements.

2.1 Mechanical Design

The housing of the modified SAPMD shall be designed for operation at 1000 PSIA and 185°F. The housing of the monitor as shown in Figure 1 will contain the electronic circuits, batteries, and pressure transducer. The bottom cover of the housing is removable to provide access to the batteries and the data connector. The cover is sealed with an O-ring to ensure that the internal pressure does not rise above 50 PSIA. The internal pressure must be limited since the batteries are sealed units and cannot withstand the high external ambient pressure. The housing structure shall be designed to withstand a maximum of pressure of 2000 PSIA to provide a 100% overpressure safety factor. If higher over-pressures are expected, the housing design can be modified either by the use of higher strength materials or by an increase in housing dimensions. The proposed design incorporates 300 series stainless steel as the housing material. The proposed design has no mounting holes assuming the unit will either be clamped or bonded in position. The design may be readily changed to provide attachment points if so required.

2.2 Electronic Design Changes

The changes in the electronic design of the SAPMD will be primarily in the start and stop command circuitry and, if necessary, to the bias circuit of the high-pressure transducer. The external start command will require additional circuitry included on the new battery board which will contain four (4) model LTC-7PN lithium batteries.

The start command will be controlled by a pair of wires connected to the high-pressure feed-thru of the SAPMD and routed outside the nozzle opening to a relay. This relay operation will be under control of the local firing-range officials and will be operated prior to ignition.

The contact closure activates a latch circuit inside the SAPMD which in turn enables a micro-powered voltage regulator to supply +5V to the SAPMD and commences recording pressure data. After a 160-second sampling period, the SAPMD returns to a power-down condition where it remains until turned back on by a command to the latch. This allows the system to be restarted in case of a misfire or delay.

The GSE will be used for interrogation, only, and will operate in the same manner as before, including supplying power to the SAPMD during interrogation.

While the SAPMD is in the power-down mode, the power drain will be less than 20 microamps of quiescent current of the regulator and latch circuitry.
2.3 Pressure Transducer

The pressure transducer selected is a standard Kulite XT-190 series ruggedized integrated sensor type absolute pressure transducer. The transducer is available with a maximum operating temperature of 350°F with a temperature compensated range of 100 to 300°F. The maximum change in sensitivity over the 100 to 300°F range is +/- 4.0 % with a repeatability of 0.05% of full scale with a 10 Vcc excitation voltage; the nominal output of the sensor is 100 mV full scale. The 8-bit resolution of the existing SAPMD A/D converter provides a theoretical resolution of 1000 psi/256 counts or 3.9 psi for the system. Actual measured performance of the present system indicates that a noise level of +/-3 LSB can be expected which gives a +/-11.7 psi noise level for the pressure measurements which combined with +/-3 psi non-linearity and hysteresis and 0.5 psi repeatability gives an error factor of +/-15.2 psi for single point measurements. The temperature coefficient of the transducer gives an expected error of +/-40 psi. The frequency response of this transducer will allow it to track the sum of the average pressure and the instantaneous acoustic pressure.

2.4 Battery Design

The required longer operational lifetime of the SAPMD and some additional power requirements by the pressure sensor will require more available battery power. The present battery power supply consists of two 3.4V lithium cells connected in series to supply a nominal 6.8V to a linear regulator which reduces the voltage to 4.5V for the electronic circuits. To increase the battery lifetime, four cells will be used in a series-parallel arrangement to supply 6.8V at twice the amp-hr rating of the existing supply and will be regulated down to 5V for the supply of the electronic circuits by a micro-powered regulator on the new battery board.
HP.SAPMD ELECTRONIC SCHEMATICS
ORIGINAL PAGE IS OF POOR QUALITY
HP.SAPMD MECHANICAL SCHEMATICS
HP.SAPMD GSE SOFTWARE LISTINGS
The following files are included with this distribution:

<table>
<thead>
<tr>
<th>File</th>
<th>Type</th>
<th>Size</th>
<th>Date/Time</th>
<th>Time</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HPSGA</td>
<td>EXE</td>
<td>72520</td>
<td>1-03-89</td>
<td>5:06p</td>
<td>Executable Program</td>
</tr>
<tr>
<td>README</td>
<td>HP</td>
<td>2765</td>
<td>1-03-89</td>
<td>5:53p</td>
<td>Program Notes</td>
</tr>
<tr>
<td>PACKING</td>
<td>LST</td>
<td>2198</td>
<td>1-03-89</td>
<td>6:23p</td>
<td>This file</td>
</tr>
<tr>
<td>HPSGA</td>
<td></td>
<td>1284</td>
<td>1-03-89</td>
<td>3:47p</td>
<td>MAKE file for HPSGA</td>
</tr>
<tr>
<td>HPSGALNK</td>
<td>LNK</td>
<td>121</td>
<td>1-03-89</td>
<td>3:48p</td>
<td>Link command file for HPSGA</td>
</tr>
<tr>
<td>CMDINT</td>
<td>C</td>
<td>30491</td>
<td>12-19-88</td>
<td>6:09p</td>
<td>C Source Files</td>
</tr>
<tr>
<td>DBCMD</td>
<td>C</td>
<td>9472</td>
<td>1-01-80</td>
<td>12:06a</td>
<td></td>
</tr>
<tr>
<td>DEBUG</td>
<td>C</td>
<td>3468</td>
<td>1-03-89</td>
<td>3:50p</td>
<td></td>
</tr>
<tr>
<td>DIPRESS</td>
<td>C</td>
<td>6561</td>
<td>1-03-89</td>
<td>4:00p</td>
<td></td>
</tr>
<tr>
<td>ERROR</td>
<td>C</td>
<td>2745</td>
<td>12-28-88</td>
<td>12:32p</td>
<td></td>
</tr>
<tr>
<td>HPSGA</td>
<td>C</td>
<td>264</td>
<td>12-28-88</td>
<td>1:59p</td>
<td></td>
</tr>
<tr>
<td>MENU</td>
<td>C</td>
<td>3860</td>
<td>1-03-89</td>
<td>5:05p</td>
<td></td>
</tr>
<tr>
<td>PRPRESS</td>
<td>C</td>
<td>2800</td>
<td>1-03-89</td>
<td>4:07p</td>
<td></td>
</tr>
<tr>
<td>RECOVER</td>
<td>C</td>
<td>4122</td>
<td>12-29-88</td>
<td>4:29p</td>
<td></td>
</tr>
<tr>
<td>SELFTEST</td>
<td>C</td>
<td>4332</td>
<td>1-03-89</td>
<td>5:03p</td>
<td></td>
</tr>
<tr>
<td>STATUS</td>
<td>C</td>
<td>2359</td>
<td>1-03-89</td>
<td>3:51p</td>
<td></td>
</tr>
<tr>
<td>SUPGLOB</td>
<td>C</td>
<td>9874</td>
<td>1-03-89</td>
<td>4:04p</td>
<td></td>
</tr>
<tr>
<td>WINDOW</td>
<td>C</td>
<td>11136</td>
<td>1-01-80</td>
<td>4:05p</td>
<td></td>
</tr>
<tr>
<td>CONIO</td>
<td>ASM</td>
<td>2304</td>
<td>1-01-80</td>
<td>10:36a</td>
<td>Assembly Source Files</td>
</tr>
<tr>
<td>FIO</td>
<td>ASM</td>
<td>8320</td>
<td>1-01-80</td>
<td>3:17p</td>
<td></td>
</tr>
<tr>
<td>_WINDOW</td>
<td>ASM</td>
<td>7061</td>
<td>12-19-88</td>
<td>6:25p</td>
<td></td>
</tr>
<tr>
<td>CONIO</td>
<td>LST</td>
<td>5501</td>
<td>12-19-88</td>
<td>6:26p</td>
<td>Assembly List Files</td>
</tr>
<tr>
<td>FIO</td>
<td>LST</td>
<td>17655</td>
<td>12-19-88</td>
<td>6:26p</td>
<td></td>
</tr>
<tr>
<td>_WINDOW</td>
<td>LST</td>
<td>16591</td>
<td>12-19-88</td>
<td>6:25p</td>
<td></td>
</tr>
<tr>
<td>CONIO</td>
<td>CRF</td>
<td>749</td>
<td>12-19-88</td>
<td>6:26p</td>
<td>Assembly Cross Reference Files</td>
</tr>
<tr>
<td>FIO</td>
<td>CRF</td>
<td>2937</td>
<td>12-19-88</td>
<td>6:26p</td>
<td></td>
</tr>
<tr>
<td>_WINDOW</td>
<td>CRF</td>
<td>2378</td>
<td>12-19-88</td>
<td>6:25p</td>
<td></td>
</tr>
<tr>
<td>HPSGA</td>
<td>MAP</td>
<td>17359</td>
<td>1-03-89</td>
<td>5:06p</td>
<td>Link map for HPSGA</td>
</tr>
<tr>
<td>CMDINT</td>
<td>OBJ</td>
<td>5872</td>
<td>1-03-89</td>
<td>4:09p</td>
<td>Object Modules</td>
</tr>
<tr>
<td>CONIO</td>
<td>OBJ</td>
<td>167</td>
<td>12-19-88</td>
<td>6:26p</td>
<td></td>
</tr>
<tr>
<td>DBCMD</td>
<td>OBJ</td>
<td>2242</td>
<td>1-03-89</td>
<td>4:06p</td>
<td></td>
</tr>
<tr>
<td>DEBUG</td>
<td>OBJ</td>
<td>1475</td>
<td>1-03-89</td>
<td>4:09p</td>
<td></td>
</tr>
<tr>
<td>DIPRESS</td>
<td>OBJ</td>
<td>2297</td>
<td>1-03-89</td>
<td>4:07p</td>
<td></td>
</tr>
<tr>
<td>ERROR</td>
<td>OBJ</td>
<td>1261</td>
<td>1-03-89</td>
<td>4:05p</td>
<td></td>
</tr>
<tr>
<td>FIO</td>
<td>OBJ</td>
<td>790</td>
<td>12-19-88</td>
<td>6:26p</td>
<td></td>
</tr>
<tr>
<td>HPSGA</td>
<td>OBJ</td>
<td>965</td>
<td>1-03-89</td>
<td>4:05p</td>
<td></td>
</tr>
<tr>
<td>MENU</td>
<td>OBJ</td>
<td>2036</td>
<td>1-03-89</td>
<td>5:05p</td>
<td></td>
</tr>
<tr>
<td>PRPRESS</td>
<td>OBJ</td>
<td>1636</td>
<td>1-03-89</td>
<td>4:07p</td>
<td></td>
</tr>
<tr>
<td>RECOVER</td>
<td>OBJ</td>
<td>1203</td>
<td>1-03-89</td>
<td>4:07p</td>
<td></td>
</tr>
<tr>
<td>SELFTEST</td>
<td>OBJ</td>
<td>1654</td>
<td>1-03-89</td>
<td>5:05p</td>
<td></td>
</tr>
<tr>
<td>STATUS</td>
<td>OBJ</td>
<td>1194</td>
<td>1-03-89</td>
<td>4:07p</td>
<td></td>
</tr>
<tr>
<td>WINDOW</td>
<td>OBJ</td>
<td>2261</td>
<td>1-03-89</td>
<td>4:05p</td>
<td></td>
</tr>
<tr>
<td>_WINDOW</td>
<td>OBJ</td>
<td>639</td>
<td>12-19-88</td>
<td>6:25p</td>
<td></td>
</tr>
</tbody>
</table>
# make file for hpsga

supglob.c : \include\stdio.h \include\process.h \include\stdlib.h
m supglob.c #allow change to supglob so that all else will compile

hpsga.obj : hpsga.c supglob.c
c l /I\sapmd hpsga.c /c

window.obj : window.c supglob.c
c l /I\sapmd window.c /c

error.obj : error.c supglob.c
c l /I\sapmd error.c /c

dbcmd.obj : dbcmd.c supglob.c
c l /I\sapmd dbcmd.c /c

menu.obj : menu.c supglob.c
c l /I\sapmd menu.c /c

prpress.obj : prpress.c supglob.c
c l /I\sapmd prpress.c /c

status.obj : status.c supglob.c
c l /I\sapmd status.c /c

dipress.obj : dipress.c supglob.c
c l /I\sapmd dipress.c /c

recover.obj : recover.c supglob.c
c l /I\sapmd recover.c /c

selftest.obj : selftest.c supglob.c
c l /I\sapmd selftest.c /c

cmdint.obj : cmdint.c supglob.c
c l /I\sapmd cmdint.c /c

debug.obj : debug.c supglob.c
c l /I\sapmd debug.c /c

_window.obj : _window.asm
   masm _window.asm,

conio.obj : conio.asm
   masm conio.asm,

fio.obj : fio.asm
   masm fio.asm,

hpsga.exe : hpsga.obj window.obj _window.obj error.obj dbcmd.obj conio.obj fio.obj menu.obj prpress.obj status.obj dipress.obj recover.obj selftest.obj cmdint.obj debug.obj
   link @hpsgallnk.lnk
#include <supglob.c>

CMDINT

Command and interrogate the SAPMD.

#include <supglob.c>

cmdint()
{
    int day,
    hh,
    mmss[2],
    cmd,
    i;
    unsigned char adrcnt[3];
    unsigned char *tptr;
    unsigned long ticks;
    screen.lines++;
    screen.cury++;
    menu(2,1);
    while (1)
    {
        wchs(CR);
        prompt(">");
        if ((i=rcln())==LEFT || i==HOME)
        {
            screen.lines--;
            screen.cury--;
            return(0);
        }
        switch (scan())
        {
            case SG:
            {
                cmd=SETGMT;
                goto ii;
            }
            case SM:
            {
                cmd=SETMET;
            }
            ll:
            {
                if (scan()==NUMBER && (day=acc)<=365) /* get day ... */
                    if (scan()==SLASH)
                        /* eat '/' */
                        if (scan()==NUMBER && (hh=acc)<24) /* get hour ... */
                            [for (i=0;i<2;i++) /* get minutes and seconds */
                                if (scan()==COLON) /* get ':' */
                                    if (scan()==NUMBER && (mmss[i]=acc)<60)
                                        continue; /* next iteration */
                                    break; /* exit loop on error */
                            if (i>=2) /* :hh:ss present? */
                                if (scan()==EOL) /* good terminator? */
                                    [ticks=day*0x2a3001+(unsigned long)hh*7200+
                                     mmss[0]*120+mmss[1]*2; /* 1/2 secs*/
                                        if (!sacmd(cmd,&ticks,4) || rdsg()!=ACK)
                                            p_error(BADSAPMD); /* bad response */
                                    break;]; /* next command */
                                error(BADCMD);
                                break;
                        /* strange command */
                    /* next */
            }
        }
    }
}
case TM:
    if (scan()==EOL)
        [adrct[0]=GMTADR; /* read GMT */
         adrct[1]=8; /* check for good command */
         if (!sacmd(DUMPRAM,adrct,2) || versg(RAMDATA)) /* issue */
            [p_error(BADSAPMD); /* strange response */
            break;]; /* next command */
         rdtime();});/* read and display times */
        break; /* next */

/*******************************/
/* DR: Dump 80C51 ram */
/*******************************/
    case DR:
        schex(); /* dump 80C51 ram */
        adrc[0]=acc; /* get address */
        if (acc<=0x7f) /* save parameter */
            [adrct[1]=16; /* check range */
             if (scan()==COMMA) /* default byte count */
                [if (scan()==NUMBER) /* check for count */
                    [adrct[1]=acc & 0xff; /* a number there? */
                    scan();]
                else /* make good number */
                    [error(BADCMD); /* get EOL */
                    break;]);
            else /* no number */
                [error(BADCMD); /* strange command */
                break;]
        if (token==EOL) /* next */
            [if (!sacmd(DUMPRAM,adrct,2) || versg(RAMDATA)) /* send error */
            p_error(BADSAPMD); /* command ok */
            else /* display SFR contents */
                [dump(adrct[0],adrct[1]);/* read and print resp. */
                break;]);
        error(BADCMD); /* strange command */
        break; /* next */

/*******************************/
/* DS: Dump 80C51 SFR */
/*******************************/
    case DS:
        if (schex()==NUMBER && (adrct[0]=acc)>0x7f && acc<=255) /* dump SFR */
            if (scan()==EOL) /* good command? */
                [if (!sacmd(DUMPSFR,adrct,1) || versg(SFRDATA)) /* send error */
                p_error(BADSAPMD); /* command ok */
                else /* display SFR contents */
                    [adrct[1]=1; /* fake byte count */
                    dump(adrct[0],adrct[1]); /* display SFR contents */
                    break;]);
            else /* strange command */
                [error(BADCMD); /* next */
                break;]

/*******************************/
/* DE: Dump external memory */
/*******************************/
    case DE:
        schex(); /* dump 80C51 ram */
        adrc[0]=acc; /* get address */
        adrc[1]=acc>>8; /* save parameter */
        [.../* ... */
adrct[2]=16; /* default byte count */
if (scan()==COMMA) /* check for count */
  [if (scan()==NUMBER) /* a number there? */
    [adrct[2]=acc & 0xff; /* make good number */
      scan();]
  else /* no number */
    [error(BADCMD); /* strange command */
      break;];
if (token==EOL) /* check for garbage */
  [if (!sacmd(DUMPEXT,adrct,3) || versg(EXTDATA))
    p_error(BADSAPMD); /* print error */
  else /* command ok */
    dump(adrct[1]<<8|adrct[0],adrct[2]); /* read, print */
    break;]; /* next */
error(BADCMD); /* strange command */
break; /* next */

/***********************
*/
/*
*/
/* ER: Enter 80C51 ram
*/
/***********************
*/
case ER: /* enter ram */
  schex(); /* get address */
  i=acc; /* save address */
  if (i<=127) /* check for good address */
    if (scan()==EOL) /* check for good command */
      [enter(DUMPRAM,LOADRAM,i,0x7f,1,RAMDATA); /* ram */
        break;]; /* next */
  error(BADCMD); /* strange command */
  break; /* next */

/***********************
*/
/*
*/
/* ES: Enter 80C51 SFR
*/
/***********************
*/
case ES: /* enter ram */
  schex(); /* get address */
  i=acc; /* save address */
  if (i<=255 && i>=128) /* check for good address */
    if (scan()==EOL) /* check for good command */
      [enter(DUMPSFR,LOADSFR,i,0x7f,1,SFRDATA); /* load SFR */
        break;]; /* next */
  error(BADCMD); /* strange command */
  break; /* next */

/***********************
*/
/*
*/
/* EE: Enter EEPROM
*/
/***********************
*/
case EE: /* enter ram */
  schex(); /* get address */
  i=acc; /* save address */
  if (scan()==EOL) /* check for good command */
    [enter(DUMPEXT,LOADEE,i,0xffff,2,EXTDATA); /* load EEPROM */
      break;]; /* next */
  error(BADCMD); /* strange command */
  break; /* next */

/***********************
*/
P: Program file into EEPROM

```c
/* P: Program file into EEPROM */

case P:
    if (i=prog()) error(i);
    break;

MON: Toggle ram display window

```
/* RBYTE */
/* Read and display byte. Check for errors. */

rpbyte()
{unsigned char rch;
 int i;
 for (i=0;i<2;i++)
  {if ((rch=rdsig())==ABORT)
   {p_error(BADSAPMD);
    return(0);};
   wchs(rch);}
 return(1);}

/* rdtime */
/* Read and display GMT and MET. */

rdtime()
{static char *gmetxt[]="GMT: ","MET: ");
 int i,
  j,
  k,
  jtime[5];
 union {unsigned char byt[4];
     unsigned long ticks;} time;
 unsigned char timtxt[15];
 for (i=0;i<2;i++)
  {wchs(CR);
   stype(gmetxt[i]);
   for (j=0;j<4;j++)
    {for (k=0;k<2;k++)
     if ((timtxt[k]=rdsg())==ABORT)
      {p_error(BADSAPMD);
       return(0);};
     timtxt[2]='\0';
     time.byt[j]=bhex(timtxt);}
    dcdtime(jtime,time);
    printf("%3d/%2d:%2d:%2d.%1d",jtime[0],jtime[1],jtime[2],jtime[3],jtime[4]);}

/* dcdtime */
/* Convert binary time to ascii string. */

dcdtime(btime,atime)
{int btime[];
 unsigned long atime;
 [static unsigned long cnv[]={0x2a3001,72001,1201,21}; /* conversion const*/
 int i;
 for (i=0;i<4;i++)
  {btime[i]=atime/cnv[i];
   atime-=btime[i]*cnv[i];}
```c
12:    break;        /* error exit
    next */

/* */
/* */
EOL
/* */

/*****************************/
/* */
EOL
/* */
/*****************************/
/* */
case EOL:    /* EOL
    ignore blank line */
/* */
/* */
QUIT
/* */

/*****************************/
/* */
case Q:      /* quit
    scrup(0,0,24,79,0);    /* clear screen
    i=inp(0x21);          /* read 8259 interrupt mask
    outp(0x21,i|0x10);   /* stop serial interrupts
    exit(0);            /* stop.
    default:            /* otherwise
        error(BADCMD);}}]  /* unrecognized command */

/*****************************/
/* */
HEXW
/* */
/* Print the passed word in hex. */
/* */
/*****************************/
/* */
hexw(id,x)    /* display hex word
    struct window *id;    /* window id
    int x;            /* data
    [hex(id,x>>8);
        hex(id,x);]    /* display high ...
    */
/* */
/* */
HEXC
/* */
/* Convert the passed nibble to hex ascii. */
/* */
/*****************************/
/* */
char hexc(x)    /* convert to hex ascii
    int x;          /* nibble
    [x&=0xf;
        return((x<=9) ?x+'0' :x-10+'A' );]    /* get nibble
    */
/* */
/* */
HEX
/* */
/* Print the specified byte at the current cursor position on the */
/* specified window. */
/* */
/*****************************/
/* */
hex(id,x)    /* print byte in hex
    struct window *id;    /* window id
    int x;            /* data
    [wchw(id,hexc(x)>>4));    /* print high nibble
    */
```
purge();
if (sgch(cmd)) return(0);
for (i=0;i<plen;i++)
    {chex(*par++,hxcmd);
     for (j=0;j<2;j++)
        if (sgch(hxcmd[j]))
            return(0);}
if (sgch(CR)) return(0);
return(l);

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

S G C H

Send character to SAPMD and get response byte.

**************************************************************************

sgch(cmdb)
int cmdb;
{int rch;
wrsg(cmdb);
return(versg(cmdb));}

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

V E R S G

Verify response from SAPMD

**************************************************************************

versg(cmdb)
int cmdb;
{int rch;
 rch=rdsg();
 if (rch==cmdb) return(0);
 while (rch!=ABORT)
    {wrsg(ILNK);
     rch=rdsg();};
return(l);}
wchw(id,hexc(x));} /* print low nibble */

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

CHEX

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

C H E X

/* Convert byte to ascii hex string. */
/*****************************************************************************/
/* */
/* */
/* */
/* */
/* */
/* */
/* */

chex(byt,str)
  unsigned char byt;
  char *str;
  {*str=hexc(byt>>4);
    str++;
    *str=hexc(byt);}

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

BHEX

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

bhex(hstr)
  char *hstr;
  [int a;
    a=0;
    while (*hstr!='\0')
      [a=a*16+(*hstr<='9' ? *hstr-'0' : *hstr-'A'+10);
        hstr++;]
    return(a);]

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

ENTER

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

enter(cmd,cme,adr,maxadr,alen,dtype)
  int cmd,
    cme,
    adr,
    maxadr,
    alen,
    dtype;
  [int i,
    c,
    cflag;
    union [char bpar[3];
      int wpar;] cadr;
    while (1)
      [wchs(CR);
        hexw(&screen,adr);
        wchs(':');
        wchs(' ');
        for (i=0;i<8;i++)
          [cadr.wpar=adr++;
            cadr.bpar[alen]=1;
          if (!sacmd(cmd,&cadr,cmd!=DUMPSFR?alen+1:l)||versg(dtype))
            [p_error(BADSAPMD);
              return(0);]
        if (!rpbyte()) return(0);]*/} */
wchs('.');          /* terminate data */
wchs(' ');           /* space */
acc=0;               /* clear accumulator */
c=2;                 /* count characters to go */
cflag=0;             /* flag no change */
while (c>0)
    if ((ch=toupper(rdch()))!=(char)0xff) /* character avail. */
        switch (ch)             /* check for activation char */
            {case '0':          /* hex digits ... */
               case '1':
               case '2':
               case '3':
               case '4':
               case '5':
               case '6':
               case '7':
               case '8':
               case '9':
               case 'A':
               case 'B':
               case 'C':
               case 'D':
               case 'E':
               case 'F':
                if (c)        /* 2 characters yet? */
                    {wchs(ch);    /* write character */
                    c--;         /* count 1 character */
                    acc=acc*16+(ch<='9'?ch-'0':ch-'A'+10); /* flag byte changed */
                    cflag++;}
                continue;   /* next */
            case ' ':       /* space */
                for (;c>-l;c--) wchs(' '); /* space to next col. */
            if (cflag)       /* any change? */
                [cadr.bpar[alen]=acc; /* plant value */
                if (!sacmd(cme,&cadr,alen+l) || rdsg() != ACK)
                    {p_error(BADSAPMD); /* display message */
                    return(0);}]; /* --> return */
                break;       /* next */
            case BACKSPACE: /* oops */
                cflag=0;       /* no change */
            case CR:        /* carriage return */
                if (cflag)      /* any change? */
                    [cadr.bpar[alen]=acc; /* plant value */
                if (!sacmd(cme,&cadr,alen+l) || rdsg() != ACK)
                    [p_error(BADSAPMD); /* display message */
                    return(0)];/* --> return */
                return(1);/* return */
            default:        /* invalid character */
                ;}}}        /* ignore */
/* **************************************************
 /**************************************************
 SACMD

 Issue command to SAPMD.

 **************************************************

 sacmd(cmd,par,plen)        /* issue GSE command */
   int cmd,                   /* command byte */
       plen;                   /* number of parameter bytes */
       unsigned char *par;    /* parameter bytes */
       [unsigned char sch,    /* response character */
               hxcmd[2];
       int i,                     /* hex byte */
               j;                      /* iteration variable */
btime[4]=atime>0?5:0; /* plant halfsec */

*******************************************************************************
*/
*
*/
**
** PROG
**
** Program hex file into EEPROM.
**
*******************************************************************************
/
/
/
/
/
prog()
{
    int i,
    cks,
    bct,
    off;
    union [char a[3];
        int b;) addr;
    char *ip,
        *ipt;
    FILE *hxf;
    char hxln[133];
    skbl();
    ip=ipt;
    line[sizeof(line)-1]=CR;
    off=0;
    while (*ipt!=CR && *ipt!=',') ipt++;
    if (scan()==COMMA)
        [schex();
        off=acc;
        if (scan()!=EOL) return(BADCMD);];
    *ipt='\0';
    if ((hxf=fopen(ip,"rb"))==NULL) return(NOFILE);
    wchs(CR);
    stype("programming ... ");
    while (1)
    {for (i=0;i<sizeof(hxln);i++)
        [hxln[i]=fgetc(hxf);
        if (feof(hxf))
            [fclose(hxf);
            return(BADFILE);];
        if (hxln[i]==LF) break;];
    if (hxln[0]!={'')
        [fclose(hxf);
        return(BADFILE);];
    switch (bhx(&hxln[7]))
    {case 0:
        bct=bhx(&hxln[1]);
        addr.a[1]=bhx(&hxln[3]);
        addr.a[0]=bhx(&hxln[5]);
        ip=&hxln[9];
        cks=bct+addr.a[1]+addr.a[0];
        addr.b+=off;
        for (i=0;i<bct;i++)
            [cks+=bhx(ip);
            ip+=2;];
        if ((-cks&0xff)!=bhx(ip))
            [fclose(hxf);
            return(BADFILE);];
        ip=&hxln[9];
        for (i=0;i<bct;i++)
            [addr.a[2]=bhx(ip);
            wchs(CR);
            hexw(&screen,addr.b);
            wchs(':');
            wchs(' ');]
        return(BADFILE);];
    }
}

*******************************************************************************
*/
*/
*/
*/
*/
/* program file */
/* iteration variable */
/* checksum */
/* byte count */
/* offset */
/* EEPROM address */
/*
**
** filename pointer */
**
** ... */
**
** hex file pointer */
**
** hex file record */
**
** skip to filename */
**
** save filename start */
**
** terminate command for sure */
**
** default offset */
**
** look for end-of-filemark */
**
** offset present? */
**
** get hex address */
**
** change offset */
**
** good command? */
**
** terminate filename */
**
** new line */
**
** acknowledge */
**
** loop forever */
**
** read hex file record */
**
** read byte */
**
** early EOF? */
**
** close file */
**
** send message */
**
** EOR? */
**
** good record? */
**
** early EOF? */
**
** good record? */
**
** check type */
**
** data record */
**
** get byte count */
**
** get high address ... */
**
** ... and low */
**
** point to data */
**
** compute checksum */
**
** adjust for offset */
**
** convert data to binary */
**
** ... for checksum */
**
** next byte */
**
** checksums match? */
**
** close file */
**
** return error */
**
** point at data */
**
** program data */
**
** plant data */
**
** new line ... */
**
** print address ... */
**
** separate data */
**

wchs(' '); /* print data */
hex(&screen, addr.a[2]); /* close file */
if (!sacmd(LOADDEE, addr.a, 3)) /* return no error */
  {fclose(hxf);
   p_error(BADSAPMD);
   return(0);}
  ip++; /* next byte */
addr.b++;,/* next address */
break;
case 1:
  return(0);
default:
  return(BADFILE);}}

EXCFILE /* Open command file. */

EXCFILE /* Open command file */
excfie() /* iteration variable */
{int i;
  skbl(); /* skip blanks to filename */
  if (*iptr==CR) return(NOFILE); /* null line? */
  for (i=0;i<sizeof(line);i++) /* stomp EOL */
    if (line[i]==CR) line[i]=0;
  if ((cfile=fopen(iptr,"rb"))==NULL) /* ... */
    return(NOFILE);
  else /* file exist? */
    return(0); /* return error */
  cmdfile = l; /* file opened */
  return(0); /* flag command file open */
}

/* Convert ascii hex byte to int. */
bhx(hstr) /* good file */
char *hstr;
{int a,i;
  a=0;
  for (i=0;i<2;i++) /* end */
    {a=a*16+(*hstr<='9'?*hstr-'0':*hstr-'A'+10); /* accumulate */
     hstr++;}
  return(a);}

/* Print error and purge SAPMD response buffer. */
p_error(msg) /* error and purge */
{int msg;
  [error(msg);
   purge();] /* message number */
  /* signal error */
  /* empty response buffer */
  /* */
*/
SCAN

SCAN acquires user commands and turns them into lexical units for processing by callers. The integer returned is the token id, which is also placed in the global variable 'token'.

#include <supglob.c>

scan()
{acc=0;
 skbl();
 if (*iptr='0' && *iptr<='9')
 {while (*iptr='0' && *iptr<='9')
     acc=acc*10+*iptr++-'0';
 return(token=NUMBER);}
 else
 return(fid());}

skbl()
{while (*iptr==' ' ) iptr++;
}

fid()
{int i;
 struct name [char *ntxt;
     int tkn;];
 char *nm,
 *npt;
 static struct name idnt[]={"DE",DE,
     "DR",DR,
     "DS",DS,
     "EE",EE,
     "ER",ER,
     "ES",ES,
     "LA",LA,
     "MON",MON,
     "P",P,
     "SG",SG,
     "SM",SM,
     "TM",TM,
     "Q",Q,
     ",",COMMA,
     ":",COLON,
     "/",SLASH,
     "@",CMD,
     "="EQU};
 for (i=0;i<sizeof(idnt)/sizeof(struct name);i++) /* search for match */
SCHEX attempts to read hex numbers.

schex()

[acc=0;]
while (*iptr==' ') iptr++;
token=NUL;
while (1)
  switch (*iptr)
    [case '0':
      case '1':
      case '2':
      case '3':
      case '4':
      case '5':
      case '6':
      case '7':
      case '8':
      case '9':
        acc=acc*16+*iptr++-'0';
        token=NUMBER;
        break;
    case 'A':
    case 'B':
    case 'C':
    case 'D':
    case 'E':
    case 'F':
      acc=acc*16+*iptr++-'A'+10;
      token=NUMBER;
      break;
    default:
        return(token);};}

PROMPT

Prompt for user keyboard input.

prompt(prmpt)
char *prmpt;
{scrup(24,0,24,79,0);
movcurs(24,0);
for (;*prmpt!='\0';prmpt++) wch(*prmpt);};

RD LN

Read keyboard input until an activation character is encountered.
rdln()
  [int j;
  static char splch[]={'H','P','M','K','Q','I','R','S','G','O','<','>'};
  static char spltkn[]={UP,DOWN,RIGHT,LEFT,
                         PGDN,PGUP,INS,DEL,
                         HOME,ND,CTA,CTR};
  iptr=line;
  while (1)
    if ((ch=rdch())!=(char)0xff)
      switch (ch)
      {
        case CTRLC:
          j=inp(0x21);
          outp(0x21,j|0x10);
          popcurs();
          exit(0);
        case CR:
          *iptr=CR;
          iptr=line;
          return(EOL);
        case SPL:
          ch=rdch();
          for (j=0;j<sizeof(splch);j++)
            if (splch[j]==ch)
              [if (spltkn[j]==DEL)
                [ch=BACKSPACE;]
                goto 11;]}
          *iptr=CR;
          iptr=line;
          return(spltkn[j]);
        break;
      }
      case BACKSPACE:
      {
        if (line!=iptr)
          [wrch(ch);
           wrch(' ');]
        --iptr;}
      case LF:
      break;
      default:
      {[if (echo)
         [*iptr++=toupper(ch);
          wrch(ch);]]}
  }

R D C H

Read keyboard input until a character is encountered. Poll for
SAPMD status.

rdch()
  [unsigned char ch;
  if (!polst())
    [pshcurs();
     status();
     popcurs();];
  if (cmdfile)
    [ch=fgetc(cfile);]
if (!(ch==CEOF) && (!feof(cfile))) return(ch); /* EOF? */
fclose(cfile); /* close file */
cmdfile=0;}; /* mark command file closed */
return(rdc());} /* read keyboard */

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

R D S G

Read SAPMD input until a character is detected. Poll for
SAPMD status.

/************************************************************
rdsg()
[while (polsg())
  if (!polst())
    {pshcurs();
     status();
     popcurs();}
return(rds());]

/*******************************/
#include <supglob.c>

debug(argc,argv)
    int argc;
    char *argv[];

lint i,
    j;

static char        
['EEPROM-ON
*statxt[]
SELF-TEST GSE
ACQUISITION COMPLETE ERROR",0};

comio(0,0xc3);

for (i=0;i<sizeof(sapmd)/sizeof(struct cal)
    j++)/* SAPMD cal. coefs.*
if ((cfile=fopen("sapmd.cal","rt"))!=NULL)/* calibration file present? */
    for (j=0;j<sizeof(sapmd)/sizeof(struct cal);j++)/* SAPMD cal. coefs.*
if ((fscanf(cfile, "%d%d%f%\n",&sapmd[j].serial,&sapmd[j].offset,  
    &sapmd[j].coef))!=3)
    break;

    j=feof(cfile);
    fclose(cfile);};

show(m4);

sat(0x0f);

pmenu(m4,statxt,0,0);

sat(0x07);

while (1)
    {
        menu(1,1);
        if (!j)
            [error(BADCAL);
             j++];
        while (1)
            [prompt("SELECT OPTION: ");
             rdln();
             if (scan()==NUMBER)
                 [i=acc;
                 if (scan()==EOL)
                     [switch (i)
                         [case 1:
                             cmdint();
                             break;
                         case 2:
                             selftest();
                             break;
                         case 3:
                             recover();
                             continue;
                         case 4:
                             if (dipress())
                                 [error(NOFILE);
                                 display error;
                                 don't redraw;
                                 continue;];
                             break;
                         case 5:
                             if (prpress()) error (NOFILE); /* print data */
                             continue;
                         default:
                             error(BADCMD);
                             continue;];
                     break];}
        if (token==Q)
            [scrup(O,O,24,79,0);
             i=inp(0x21); /* read 8259 mask */
             break;]}}; /* clean up */

    /* close cal file */
    /* check for cal file error */
    /* flag error */
    /* remove error */
    /* not quite forever ... */
    /* prompt for user input */
    /* get input, act. char. */
    /* check for option */
    /* save option */
    /* check for number only */
    /* number, process option */
    /* COMMAND/INTERROGATE */
    /* go poke SAPMD */
    /* next */
    /* SAPMD SELF-TEST */
    /* exercise SAPMD */
    /* next */
    /* RECOVER PRESSURE DATA */
    /* go dump EEPROM log */
    /* next */
    /* DISPLAY PRESSURE DATA */
    /* display data */
    /* display error */
    /* don't redraw */
    /* next */
    /* PRINT PRESSURE DATA */
    /* bad option */
    /* send message */
    /* next iteration */
    /* next iteration */
    /* quit? */
    /* clear screen */
    /* read 8259 mask */

}
outp(0x21,i|0x10); /* stop serial interrupts */
exit(0); /* stop. */
if (token==CMD) /* command file? */
    [if (i=excfile()) error(i);] /* open command file */
else /* not a command file? */
    if (token!=EOL) error(BADCMD);]} /* null line? */

/*****************************/
#include <supglob.c>

dipress()
{
    int i, pnum;
    if (!getfile()) return(1);
    screen.lines++;
pshcurs();
    menu(5,1);
    prpage(prsam);
    pnum=0;
echo=0;
    while (1)
        switch (rdln())
            [case PGUP:
                if (pnum>0)
                    {pnum-=40;
                     prpage(&prsam[pnum]);}
                break;
            case PGDN:
                if (&prsam[pnum+40]<samptr)
                    {pnum+=40;
                     prpage(&prsam[pnum]);}
                break;
            case HOME:
            case LEFT:
                screen.lines--;
echo++;
popcurs();
return(0);
default:
    ;}
}*/

prpage(sm)
struct sam huge *sm;
{static char *headr =
    "SAMPLE PRESSURE"
int i, j, k;
clear(m6);
m6->cury=1;
m6->curx=6;
wtype(m6,headr);
for (i=8;i<=48;i+=40)
    {m6->cury=2;
m6->curx=32;
cursor(m6);*/
printf("SAPMD SERIAL %5d",sm->serial); /* print serial # */
for (j=0;j<20;j++)
if (sm<samptr)
{
   m6->curx=i;
   m6->cury++;
   cursor(m6);
   printf("%3d
sm++;}

movcurs(24,1);
setcurs(0x3000);}

/**********************************************************
*/
*/
*/
*/
GET FILE
*/
*/
*/
Get pressure data file and process.
*/
/**********************************************************
*/
*/
getfile()
{FILE *pfile;
 unsigned char pdata[8192],
 *pdptr;
 union [unsigned char byt[4];
 unsigned long tm;} tick;
char ah[3];
union [int i;
 unsigned char b[2];} s,
 serial;
int i,
j,k,p,m,
 adjust,
pnum;
float coef;
coef=.058594;
adjust=0;
prompt("ENTER FILENAME: ");
if ((i=rdln())_HOME II i==LEFT) return(0); /* read filename
skbl();
if (*iptr==CR) return(0);
for (i=0;i<sizeof(line);i++)
   if (line[i]==CR) line[i]=0;
if ((pfile=fopen(iptr,"rb"))==NULL)
   return(0);
elself
   for (p=0;p++)
   {
      ah[0]=fgetc(pfile);
      ah[1]=fgetc(pfile);
      ah[2]="0";
      if (feof(pfile)) break;
pdata[p]=bhex(ah);};
fclose(pfile);
pdptr=pdata;
samptr=prsam;
serial.b[1]=*pdptr++;
serial.b[0]=*pdptr++;
for (j=0;j<sizeof(sapmd)/sizeof(struct cal);j++)/* cal. coef.
   if (sapmd[j].serial==serial.i) /* is this one calibrated?
      [coef=sapmd[j].coef;
         adjust=sapmd[j].offset;
         break;};
for (j=0;j<4;j++) tick.byt[j]=*pdptr++; /* get GMT
	tick.tm*=51; /* convert to 100 msec units */
m = ( p - 6 ) / 2;
for (k=0;k<m;k++)
    s.i=k;
    s.b[0]=*pdptr++;
    samptr->sample=s.i;
    j=*pdptr++;
    samptr->press=(j<adjust?0:j-adjust)*coef;
    samptr->serial=serial.i;
    samptr++;
return(1);