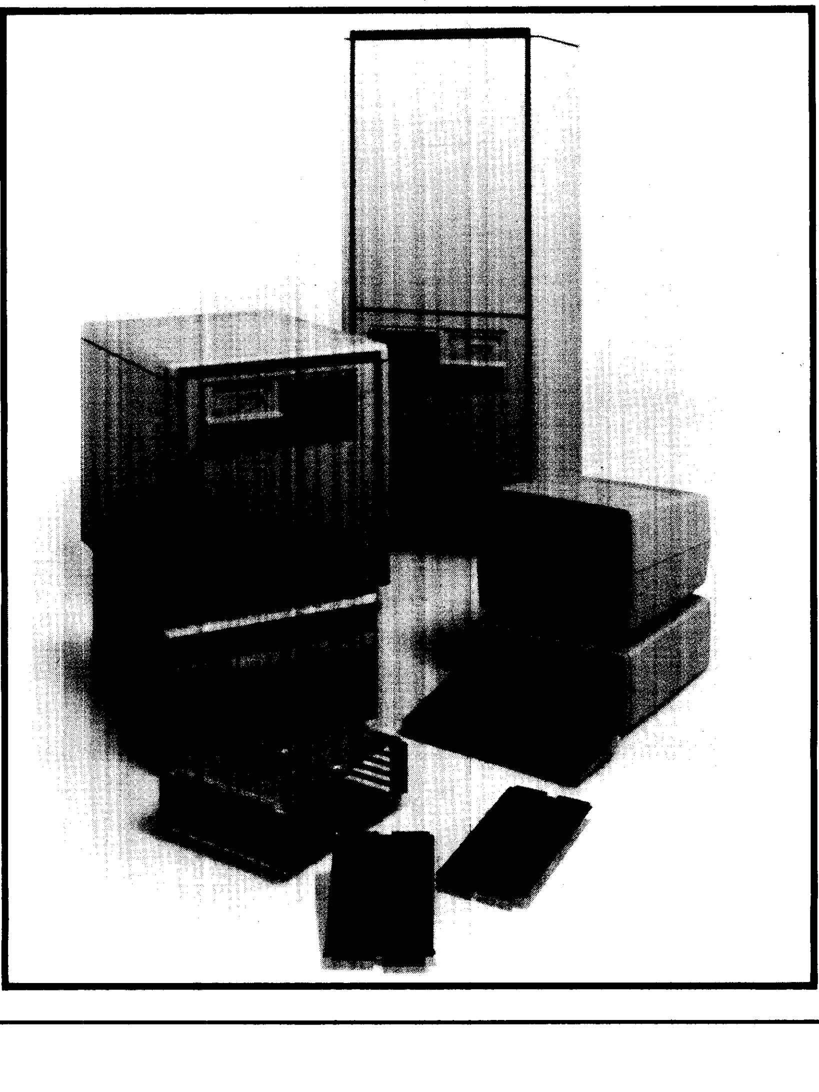
HEWLETT PACKARD

HP 1000 A600 Computer

Engineering and Reference Documentation — Vol. 2

HP 1000 A-Series



HP 1000 A600 Computer

Engineering and Reference Documentation

Volume 2

HEWLETT PACKARD

HEWLETT-PACKARD COMPANY
Data Systems Division
11000 Wolfe Road
Cupertino, California 95014

Vol. 2
MANUAL PART NO. 02156-90003
Printed in U.S.A. March 1983
E0383

Table of Contents

LIST OF ILLUSTRATIONS (Volume 2)

```
+----------------------------------------------------+------------------+
|                                                    |                  |
|   APPLICATION INFORMATION FOR 25 kHz POWER         |   APPENDIX  A    |
|                                                    |                  |
+----------------------------------------------------+------------------+
```

This appendix provides application information for the 25 kHz sine-wave output of the HP 12035A Power Supply and the optional HP 12158A 25-kHz Power Module that may be added to the 0950-0873 or 0950-0893 power supply in the 2156A or 2196A/B computers.

Introduction

HP 1000 L-Series Computers and Systems utilize the Model 12035A Power Module as their power supply, whereas in the A-Series the 12035A Power Module is used only with the 12030A Card Cage. An important design factor in the 12035A Power Module and in the 12158A 25 kHz Power Module option of 2156A or 2196A/B Computers is the inversion of 50/60 Hz ac power to a regulated 25 kHz sine wave that is stepped down and rectified to provide the outputs shown in Figure 1. A bonus of this design for the OEM or end user with unique power requirements that are not met by the standard dc voltages is the availability of 25 kHz ac power at the backplane of A/L-Series card cages, computers, and systems and at a connector on the front of the power module. At the 25 kHz frequency, power transformers and filtering components (capacitors and chokes) can be small and lightweight enough to make possible on-interface power supplies.

Uses of 25 kHz backplane power

25 kHz backplane power can be used when designing special interfaces on the 12010A Breadboard Interface to provide ac input power for compact, lightweight on-interface dc power supplies to meet any of the following requirements:

1. Provision of dc voltages in addition to those supplied by the 12035A Power Module.
2. Provision of dc supplies whose analog grounds are isolated from the computer ground.
3. Provision of multichannel isolated power to digital communication circuits to eliminate ground noise paths and maximize the reliability of serial data transfers.
4. Low voltage, high current power for supplying large arrays of integrated circuits.

Use of 25 kHz power from the power module front connector

25 kHz power is conveniently available from the power module front connector for powering circuits that are separate from the computer or system backplane. Uses might include signal conditioning power to external sensors (such as strain gauges) or power for logic circuits external to the computer backplane. Use of the power module's 25 kHz ac output can eliminate the need for separate, 50/60 Hz power supplies where external power requirements are small, minimizing costs, space requirements, and weight.

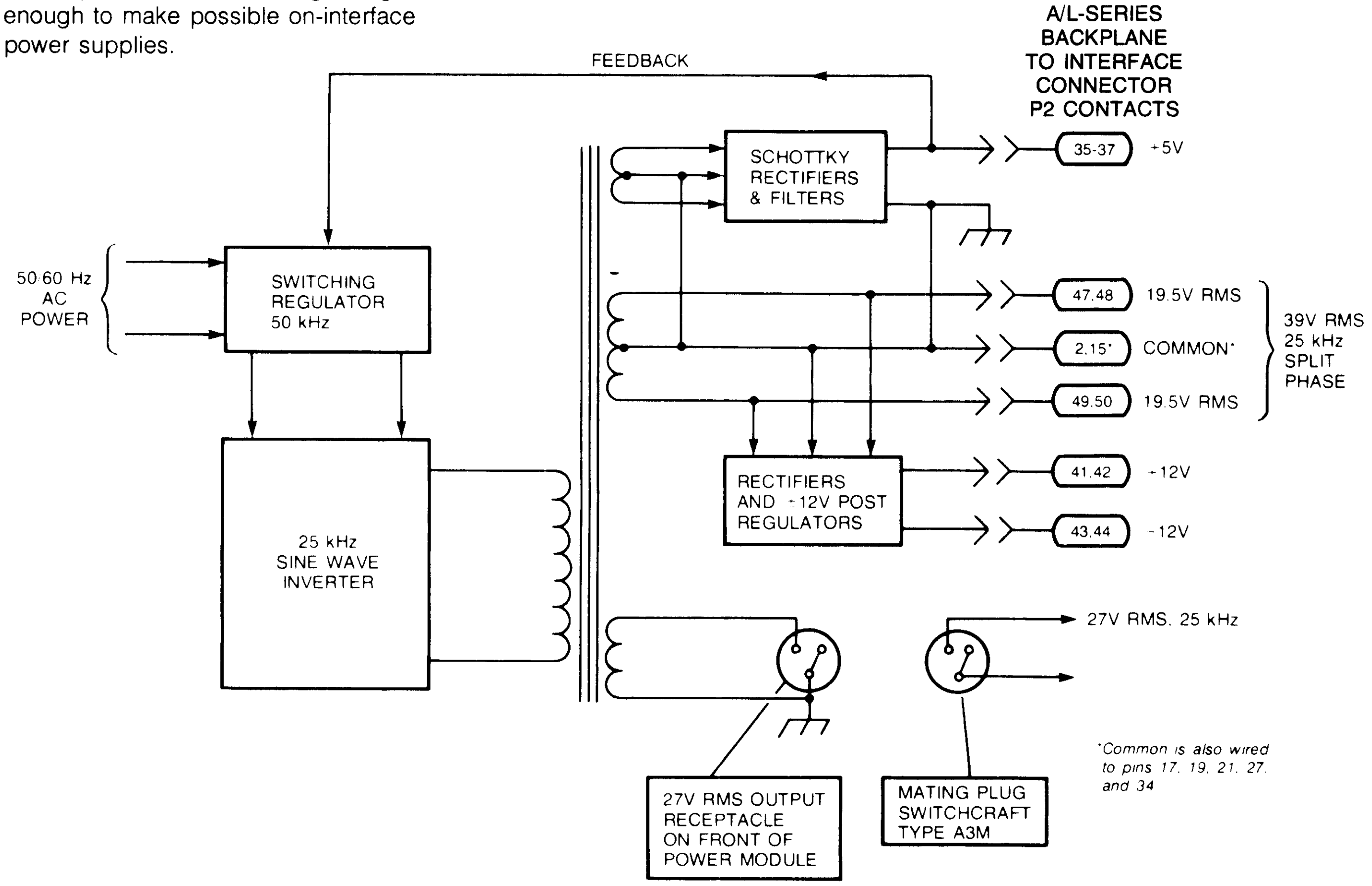

A/L-SERIES BACKPLANE TO INTERFACE CONNECTOR P2 CONTACTS
FEEDBACK
50/60 Hz AC POWER
SWITCHING REGULATOR 50 kHz
25 kHz SINE WAVE INVERTER
SCHOTTKY RECTIFIERS & FILTERS
35-37 +5V
47,48 19.5V RMS
2,15* COMMON*
49,50 19.5V RMS
39V RMS 25 kHz SPLIT PHASE
RECTIFIERS AND ±12V POST REGULATORS
41,42 +12V
43,44 −12V
27V RMS, 25 kHz
27V RMS OUTPUT RECEPTACLE ON FRONT OF POWER MODULE
MATING PLUG SWITCHCRAFT TYPE A3M
*Common is also wired to pins 17, 19, 21, 27, and 34

Figure 1. 12035A Power Module, Simplified Diagram

25 kHz ac power specifications of the 12035A Power Module

39V rms split-phase backplane output: The power module's output to the backplane of A/L-Series card cages, computers, and systems includes 39V rms split phase across pins 47/48 and 49/50 of interface card printed circuit plug P2 with a center tap connected to common (pins 2, 15, 17, 19, 21, 27, 29, and 34 of P2), as shown in Figure 1. With respect to common, the voltages at pins 47/48 and 49/50 of plug P2 are 19.5V rms. This backplane output is conveniently available for powering small on-interface dc power supplies.

27V rms front connector output: A separate transformer winding provides a 27V rms single-phase output to a connector on the front of the power module. This output can be used for signal conditioning power to external sensors or for other small external power supply uses. A Switchcraft type A3M plug is the required mating connector for this output.

Regulation: Within ±8% of nominal.

Available power: 25 kHz power available from either output or total available from both outputs depends on usage of dc current from the 12035A Power Module, as follows:

AC Power	+5Vdc	+12Vdc	−12Vdc
70 Watts	25A	4.0A	2.0A

NOTE: Alternate ac power and dc current output combinations are possible within the 250W to 319W maximum total power output, provided that no more than the highest power or current listed above is drawn from any output. However, because of complex thermal interactions within the power module you cannot rely upon directly trading all of the power not used in one or more dc outputs for additional ac power.

On-interface dc power supplies

Non-isolated, series-regulated dc power supply (Use 1 from page 1)

Purpose and basic design. Where additional +7.5V to +12V dc at up to 1 amp is needed for interface circuits, the 25 kHz backplane power can be used to provide a non-isolated positive regulated power supply as shown in Figure 2. The 19.5V rms potential on either side of common provides at least +14.5V dc after rectification and filtering. An adjustable, off-the-shelf, three-terminal integrated circuit voltage regulator, National Semiconductor Series LM117 or equivalent, can be used to set the regulated output voltage within the range of +7.5V to +12V dc. The regulated voltage output is dependent upon the values of resistors R2 and R3. A negative output voltage supply similar to the positive supply shown in Figure 2 can be made by reversing polarities of the rectifiers and using a negative adjustable regulator, National Semiconductor Series LM137 or equivalent.

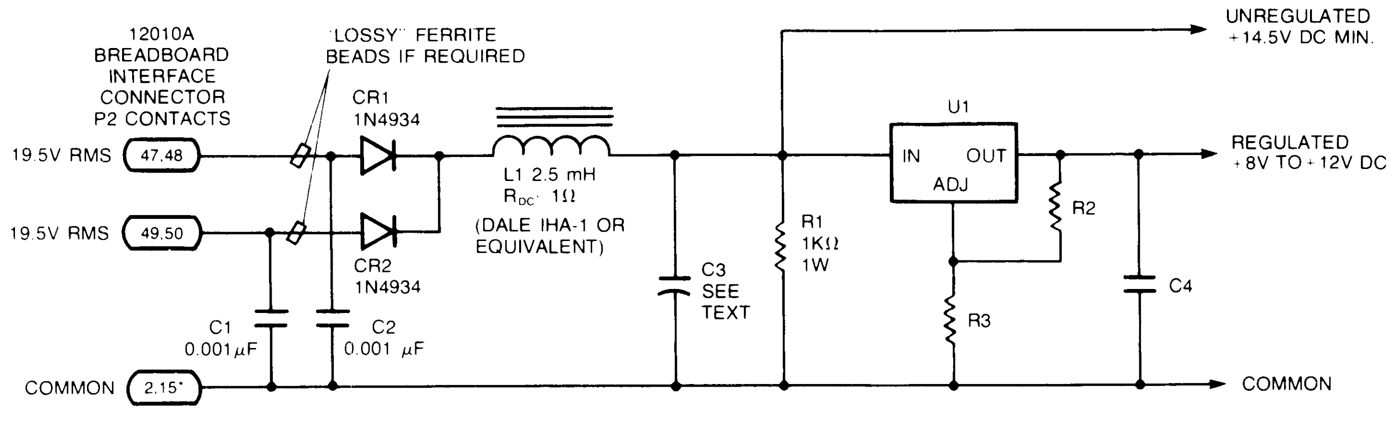
Preserving purity of the 25 kHz ac input sine wave. To maintain the purity of the input 25 kHz sine wave, near 180 degree conduction should be provided in the rectification process, which necessitates the use of a choke input filter. This filter also limits the surge current at turn-on if the requirements for Lmin are met. The equation for Lmin with a 25% safety factor is given by:

L_{min} (in henries) = (K/fs) x R_L

Where: fs = 25 kHz
R1 = Minimum load resistance
K = 0.06 for full wave rectifiers

This implies the need for a minimum load. If the circuits to be powered allow the load current to go to zero, a preloading bleeder resistor is required. The final value of Lmin would then be determined by the allowed power loss (dissipation) of the preloading resistor. When the Lmin requirement is met, the surge current will be acceptable and sine wave distortion will be minimized.

Selection of rectifiers. Rectifiers used with 25 kHz input power must be of the fast recovery type with less than 200 nanosecond recovery time. Allowing for possible transients from leakage inductances, overshoot, and MTBF derating, the rectifiers should also have 100V peak inverse voltage rating.



NOTES: U1 is a National Semiconductor type LM117 Series or equivalent adjustable regulator.
Values of C4, R2, and R3 should be selected in accordance with instructions in U1 manufacturer's data sheet.

Figure 2. On-interface regulated power supply with up to 1A output using 25 kHz ac input from L-Series backplane

Keeping noise off the 25 kHz ac input lines. During rectifier recovery, the removal of stored charge in the rectifiers will appear as spikes on the rectifier inputs. These spikes should be suppressed to keep them from travelling along the 25 kHz ac input lines in the backplane. Small 0.001 to 0.1 microfarad ceramic capacitors (C1 and C2 in Figure 2) will usually damp out these spikes, with the required capacitor value dependent upon the magnitude of stored charge being removed. If underdamped ringing is present because of leakage inductance, small ferrite beads, tubes, or toroids can be threaded onto the rectifier leads to provide a "lossy" inductive reactance at high frequencies to effectively dissipate undesirable recovery currents.

Input filtering. The value of C3 is determined by the amount of ripple voltage that can be tolerated at the input of integrated circuit regulator U1. The Vin-Vout differential of 3 volts must be met for any chosen output voltage as noted in Reference 2. The Ripple factor r for a full-wave rectifier circuit is given by:

$r = (0.83/(L1 \times C1) \times 5.76 \times 10^{-6}$

The case size and construction of capacitor C3 must be capable of conducting the ripple current without excessive dissipation. Ripple current will be at 2 fs and will be sinusoidal when Lmin requirements are met. The rms ripple current in amps is given by:

$i_R = VRMS/(4\pi \times fs \times L_1)$

Where: VRMS is the input voltage phase to common

$fs = 25$ kHz

$L_1 >= L_{min}$

The minimum inductive value of L1 must be present with the dc current flowing through it over the complete load current range. This requires an inductor with gaps in the magnetic circuit, either fixed or distributed, such as in powdered iron cores, or solenoid-wound inductors over ferrite rods (available from Reference 9).

Regulator dissipation. Since the regulator is a linear series pass type, the difference between the voltage developed across C3 at the regulator input and the desired output at the load current must be dissipated in the regulator. This dissipation is given by:

$P_{diss} = (V_{in}\text{-}V_{out}) \times (I_L + V_{in}\, I_q)$

Where: I_q = the quiescent current of the regulator.

Case to junction thermal resistances are given in the regulator manufacturer's data sheet. The dominant thermal resistance will be the case to air stream, which is usually available on heat sink manufacturer's data as a function of air velocity. You can assume a minumum 200 ft/min flow across the board with a maximum air temperature on the exit side of 66°C under worst case conditions. For low power on-card dc supplies, the copper foil on the printed circuit board can be used as a heat sink. However, the suitability of this arrangement should be checked carefully with thermocouples to confirm that the temperature rise of the regulator is not excessive.

Isolated or "floating" dc power supplied (Uses 2 and 3, page 1)

A major advantage of the 25 kHz backplane power is its ease of use for isolated power supplies that can have separate analog grounds, thereby reducing the effects of ground-conducted noise as discussed in References 3 and 4. Isolation is provided by an on-interface transformer, as shown in Figure 3. The use of 25 kHz ac input makes it possible for the isolation transformer to be very small and inexpensive. Toroidal printed circuit mounting types or "P" core (Reference 7) shielded printed circuit mounting types generally offer the best price-performance combination. However, small E-E types can also be used at lower cost with some sacrifice in electromagnetic and electrostatic shielding. High permeability ferrite materials having low losses at 25 kHz are readily available with matching bobbins and mounting hardware from References 6 through 10.

Primary-to-secondary isolation of both dc and high frequency can be somewhat complex. References 3 and 4 describe single and double shielded transformers. It is possible to achieve high isolation with small ferrite cores and proper inter-winding shield design. Simple copper foil inter-winding shields are relatively inexpensive and are effective in decreasing primary-to-secondary electrostatic coupling at frequencies from 100 Hz to about 100 kHz. For higher frequencies, "link" coupling of two cores or other techniques may be required (Reference 3, p 117).

The ground isolation provided by the multi-channel +10V power supply circuits depicted in Figure 3 eliminates errors caused by ground-induced noise. In analog voltage measurement applications, power supply isolation minimizes common mode noise, improving measurement accuracy. With respect to digital data transmission uses, power supply isolation allows data terminals to operate at greater distances from the local system with fewer data errors than would otherwise be possible. When the power supply is not isolated, noise in the 50/60 Hz mains power distribution and grounding system supplying the computer can cause current noise loops that degrade signal integrity.

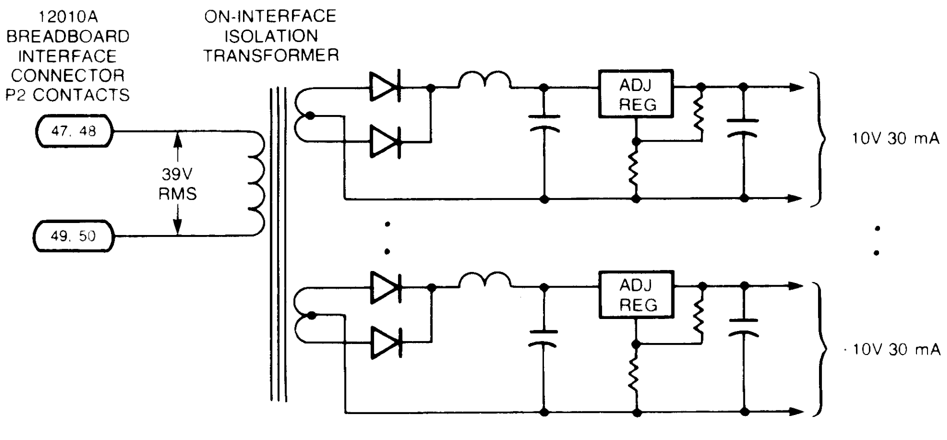

12010A BREADBOARD INTERFACE CONNECTOR P2 CONTACTS
ON-INTERFACE ISOLATION TRANSFORMER
47, 48
49, 50
39V RMS
ADJ REG
ADJ REG
10V 30 mA
10V 30 mA

Figure 3. Multiple, isolated, on-interface +10V/30 mA power supplies

High-efficiency, on-interface low-voltage, high current power supply (Use 4, page 1)

Heat dissipation is often the main factor limiting the current output of on-interface power supplies. This is particularly true for lower voltage, high current supplies, such as required for many digital integrated circuit families. For example, at the +5V used for TTL families of integrated logic circuits, even the dissipation of the rectifiers can be a significant 14% to 20% of total power, because of the inherent 0.7V to 1.0V forward drop across silicon rectifiers, and heat sinking may be required at 3-5 Amp currents. Use of hot carrier or Schottky junction rectifiers, which have a lower forward drop presenting a power loss of only 4%-5% of the total power output, have peak inverse voltage ratings that are suitable for lower voltage power supplies and may not require heat sinks because of their lower power dissipation.

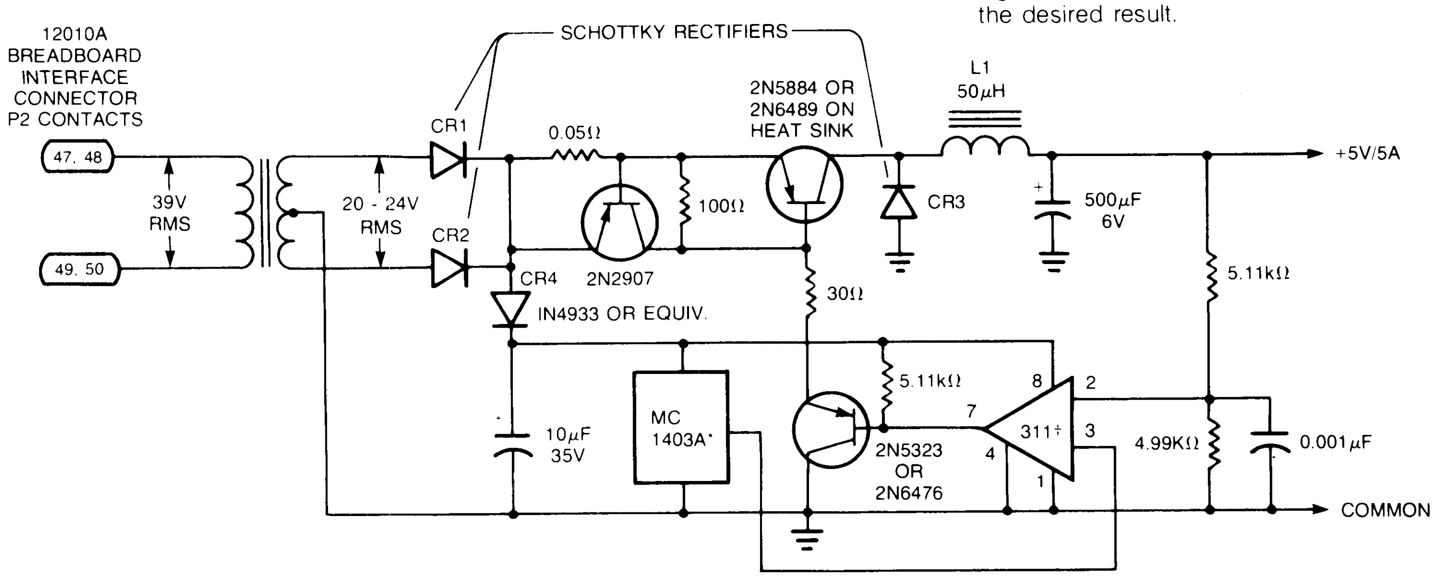
At low output voltages, the 2-3 volt drop required across most three-terminal adjustable integrated circuit series regulators for proper regulation can account for 40%-60% of the total power output, which is lost in the regulator and must be dissipated. Regulator heat sinking becomes difficult for even 1-3 Amp current outputs and impossible for the higher current levels that larger three-terminal regulators are able to pass. Because of these efficiency and dissipation problems, a more efficient circuit approach has evolved, as shown in Figure 4.

The circuit of Figure 4 uses a driven switching regulator for more efficient delivery of low voltage, high current output. This circuit regulates on the basis of the conduction angle of the pulsating rectified, unfiltered dc from the on-interface Schottky rectifiers. The result is efficiencies of 70%-85% with 1 Amp to 5 Amp loads. The duty cycle control is uniform over the half sine wave and the instantaneous energy is low at the switching transitions, which minimizes waveform distortion and RFI emission. Because the regulator operates on the incoming frequency as a driven circuit, it also eliminates the generation of other frequencies that would be a problem if an on-interface switching regulator integrated circuit were used. The circuit of Figure 4 eliminates sum and difference noise frequencies and a host of non-repetitive noise problems, while optimizing efficiency.

External supplies using 25 kHz power from the power module front connector

Power supplies for logic circuits or sensor signal conditioning circuits external to the computer or system card cage can also use 25 kHz power as the primary ac input. In fact, ac input power to the logic circuits power supply for the flexible disc in HP 1000 L-Series Systems is taken from the front 25 kHz connector of the 12035A Power Module. Similar use can be made by the OEM or end user in systems assembled from HP 1000 A/L-Series components. It is important to note, however, that physical clearance for the mating plug is not sufficient in the 2103L (box) Computer to permit use of 25 kHz power from the power module front connector in that configuration.

The design of 25 kHz-driven external power supplies is essentially the same as for on-interface power supplies, as previously discussed. However, less-stringent space constraints can be expected to simplify layout and make heat dissipation easier in the external supplies, so less design effort should be required to achieve the desired result.



CR1, CR2, and CR3 are International Rectifier 80SQ10 5A Schottky rectifiers.
**Motorola MC1403A or equivalent 2.5V low TC reference source.*
†National Semiconductor LM 311 or equivalent Comparator.
L1 is a Dale type IH5 or equivalent solenoid choke coil.

Figure 4. High efficiency on-interface, low voltage, high current 25 kHz driven switching power supply

References:

1. Reference Data for Radio Engineers, Fifth Edition, Howard W. Sams & Co., Inc., 1974; Chapter 13, pp 28-30.
2. National Semiconductor Linear Data Book, 1978, Section I, pp 15-22 and 50-54.
3. Morrison, Ralph, "Grounding and Shielding Techniques in Instrumentation", Second Edition, Wiley Publications, Inc., 1977.
4. Ott, Henry, "Noise Reduction Techniques in Electronic Systems", Wiley Publications, Inc., 1976.
5. Fairchild "Voltage Regulator Handbook" or "Hybrid Data Book", available from Fairchild Semiconductor.
6. Ferroxcube "Linear Ferrite Materials and Components".
7. TDK Data Book, Ferrite Cores — 2 DLE 88-002A.
8. Siemens Data Book, "Soft Magnetic Siferrit", 1975.
9. Fair-Rite Materials Data Book (Rods).
10. Micrometals "Shielded Coil Forms".
11. White, Donald, "EMI Control Methodology and Procedures", Don White Consultants, 1978.

PROCESSOR CARD REFERENCE DATA — APPENDIX B

This appendix contains the logic equations used in implementing the programmable logic devices. Refer to Appendix D for a representative listing of instruction base-set microcode.

13A

B.1 INSTRUCTION DECODE PAL (U1405) 1668

Instruction Decode PAL *** U1405 *** programmed HP P/N : 12101-80015
Use DATA I/O personality card type 1427, socket adapter type 1428-2.

```
 1    2    3    4    5    6    7    8    9
IR08 IR09 IR10 IR11 IR12 IR13 IR14 IR15 /SATEST GND
/LVLO /MRGIFETCH /MRGREAD /UIGOE /EASIOE /MRGOE IR07 IR06 /PCMRG VCC
  11      12        13        14      15      16     17   18    19

/MRGOE = IR14*LVLO +
         /IR14*IR13*LVLO +
         /IR14*/IR13*IR12*LVLO

/EASIOE = /IR15*/IR14*/IR13*/IR12*LVLO +
          IR15*/IR14*/IR13*/IR12*IR10*LVLO +
          IR15*/IR14*/IR13*/IR12*/IR11*/IR10*/IR09*LVLO +
          IR15*/IR14*/IR13*/IR12*IR11*/IR10*/IR09*LVLO +
          IR15*/IR14*/IR13*/IR12*/IR11*/IR10*IR09*/IR08*/IR07*/IR06*LVLO +
          IR15*/IR14*/IR13*/IR12*/IR11*/IR10*IR09*/IR08*/IR07*IR06*LVLO

/UIGOE = IR15*/IR14*/IR13*/IR12*/IR11*/IR10*IR09*IR08*LVLO +
         IR15*/IR14*/IR13*/IR12*IR11*/IR10*IR09*LVLO

/MRGREAD = /SATEST*IR14*LVLO +
           /SATEST*/IR14*/IR13*IR12*/IR11*LVLO +
           /SATEST*/IR14*IR13*/IR12*/IR11*LVLO +
           /SATEST*/IR14*IR13*IR12*LVLO +
           /SATEST*IR15*/IR14*/IR13*IR12*IR11*LVLO +
           /SATEST*IR15*/IR14*IR13*/IR12*IR11*LVLO

/MRGIFETCH = /SATEST*/IR15*/IR14*IR13*/IR12*IR11*LVLO

/PCMRG = /IR15*/IR14*/IR13*IR12*IR11 +
         /IR15*/IR14*IR13*IR11 +
         /IR15*IR14*IR13*IR12
```

B.2 DESTINATION SPECIAL PAL (U507) 1820-2573 14H4

Destination Special PAL *** U507 *** programmed HP P/N : 12101-80016
Use DATA I/O personality card type 1427, socket adapter type 1428-1.

```
PL28 PL26 PL25 SPARE1 YOBUF ABREF IR03 IR11 SPARE2 GND
PL19 PL23 PL29 ALUI3 B1 BO ALUI7 CT PL27 VCC

BO = /PL29*PL25 +
     PL29*/PL28*/PL27*/PL26*/PL25*IR11 +
     PL29*/PL28*/PL27*/PL26*PL25*YOBUF +
     PL29*/PL28*/PL27*PL26*/PL25*IR03

B1 = /PL29*PL26 +
     PL29*/PL28*/PL27*PL26*/PL25

ALUI3 = /PL29*PL19 +
        PL29*/PL28*/PL27*PL19 +
        PL29*/PL28*PL27*/PL26*/PL25*CT

ALUI7 = /PL29*PL23 +
        PL29*/PL28*/PL26*/PL25*PL23 +
        PL29*/PL28*/PL26*PL25*ABREF +
        PL29*/PL28*PL26*/PL25*PL23
```

B.3 A/B ADDRESS SPECIAL PAL (U802) 16H2 1820-2679

A/B Address Special PAL *** U802 *** programmed HP P/N : 12101-80017
Use DATA I/O personality card type 1427, socket adapter type 1428-1.

```
Y11 Y12 Y10 Y13 Y14 Y09 Y08 Y07 Y03 GND
Y00 Y06 Y01 Y02 ONES ZERO Y05 Y15 Y04 VCC

ZERO = /Y15*/Y14*/Y13*/Y12*/Y11*/Y10*/Y09*/Y08*
       /Y07*/Y06*/Y05*/Y04*/Y03*/Y02*/Y01*/Y00

ONES = Y15*Y14*Y13*Y12*Y11*Y10*Y09*Y08*Y07*Y06*Y05*Y04*Y03*Y02*Y01*Y00
```

B.4 INTERRUPT CONTROLLER NO. 1 PAL (U307) 1820-2680 16R6

```
Interrupt Controller #1 *** U307 *** programmed HP P/N : 12101-80018
Use DATA I/O personality card type 1427, socket adapter type 1428-2.
  1    2    3   4   5     6     7    8
/UCLK PL8 PL7 PL6 PL5 /LVLO SMPV SPE /ICRS GND
GND BPON /QPEI /PSFF /QMPI /MPEN /DTST /TDI /ENCN VCC
 11  12    13    14    15    16    17

18 /TDI := ENCN*PL8*/PL7*/PL6*PL5*BPON +
           /ENCN*TDI*/LVLO*BPON +
           ENCN*TDI*/PL8*/LVLO*BPON +
           ENCN*TDI*PL7*/LVLO*BPON +
           ENCN*TDI*PL6*/LVLO*BPON +
           ENCN*TDI*PL5*/LVLO*BPON
17 /DTST := ENCN*PL8*/PL7*PL6*PL5*BPON +
             /ENCN*DTST*BPON +
             ENCN*DTST*/PL8*BPON +
             ENCN*DTST*PL7*BPON +
             ENCN*DTST*/PL6*BPON +
             ENCN*DTST*PL5*BPON
16 /MPEN := ENCN*PL8*PL7*PL6*PL5*BPON +
            /ENCN*MPEN*/SMPV*BPON +
            ENCN*MPEN*/PL8*/SMPV*/ICRS*BPON +
            ENCN*MPEN*/PL7*/SMPV*/ICRS*BPON +
            ENCN*MPEN*/PL6*/SMPV*/ICRS*BPON +
            ENCN*MPEN*PL5*/SMPV*/ICRS*BPON
15 /QMPI := SMPV*BPON +
            /ENCN*QMPI*BPON +
            ENCN*QMPI*PL8*/ICRS*BPON +
            ENCN*QMPI*PL7*/ICRS*BPON +
            ENCN*QMPI*/PL6*/ICRS*BPON +
            ENCN*QMPI*PL5*/ICRS*BPON +
            ENCN*/PL8*/PL7*PL6*PL5*BPON
14 /PSFF := ENCN*PL8*PL7*/PL6*PL5*BPON +
            ENCN*ICRS*BPON +
            /ENCN*PSFF*/SPE*BPON +
            ENCN*PSFF*/PL8*/SPE*BPON +
            ENCN*PSFF*/PL7*/SPE*BPON +
            ENCN*PSFF*PL6*/SPE*BPON +
            ENCN*PSFF*PL5*/SPE*BPON
13 /QPEI := PSFF*SPE*BPON +
            /ENCN*QPEI*BPON +
            ENCN*QPEI*PL8*BPON +
            ENCN*QPEI*PL7*BPON +
            ENCN*QPEI*PL6*BPON +
            ENCN*QPEI*/PL5*BPON
```

B.5 INTERRUPT CONTROLLER NO. 2 PAL (U407) 16R4 1820-2681

```
Interrupt Controller #2  *** U407 *** programmed HP P/N : 12101-80019
Use DATA I/O personality card type 1427, socket adapter type 1428-2.
   1   2   3   4   5  6    7    8     9    10
/BCLK PL7 PL8 PL6 Y1 PL5 PFW1 PFW2 TBGIN GND
GND /LDIM1 BPON /IM1 /STBG /QTBI /SPFW /ICRS /ENCN VCC

18 /ICRS = ENCN*/PL8*PL7*PL6*PL5*BPON

16 /QTBI := ENCN*/PL8*PL7*/PL6*PL5*BPON +
            STBG*BPON +
            /ENCN*QTBI*/ICRS*BPON +
            ENCN*QTBI*PL8*/ICRS*BPON +
            ENCN*QTBI*/PL7*/ICRS*BPON +
            ENCN*QTBI*PL6*/ICRS*BPON +
            ENCN*QTBI*PL5*/ICRS*BPON

   /SPFW := PFW1*/PFW2*BPON +
            /ENCN*SPFW*BPON +
            ENCN*SPFW*PL8*BPON +
            ENCN*SPFW*/PL7*BPON +
            ENCN*SPFW*/PL6*BPON +
            ENCN*SPFW*PL5*BPON

   /IM1 := Y1*LDIM1*/ICRS*BPON +
           IM1*/LDIM1*/ICRS*BPON

   /STBG := TBGIN*BPON
```

B.6 ASG SKIP SPECIAL PAL (U807) 16C1 1820-2752

```
ASG Skip Spec PAL *** U807 *** programmed HP P/N : 12101-80020
Use DATA I/O personality card type 1427, socket adapter 1428-3.
 1   2    3    4    5    6    7    8    9
Y15 YCBUF ZERO IR04 IR03 IR01 IR05 IR10 IR00 GND
SPARE1 SPARE2 SPARE3 IR15 /ASGSKP ASGSKP SPARE4 YN Y0 VCC
                      14     15      16     17   18 19

ASGSKP = /IR15*IR10*/IR00*IR05*/YCBUF +
         /IR15*IR10*/IR00*IR01*ZERO +
         /IR15*IR10*/IR00*IR04*/Y15 +
         /IR15*IR10*/IR00*IR03*/Y0 +
         /IR15*IR10*IR00*/IR05*/IR01*/IR04*/IR03 +
         /IR15*IR10*IR00*IR05*YCBUF +
         /IR15*IR10*IR00*IR01*/ZERO +
         /IR15*IR10*IR00*IR04*/IR03*Y15 +
         /IR15*IR10*IR00*IR03*/IR04*Y0 +
         /IR15*IR10*IR00*IR03*IR04*Y15*Y0 +
         /IR15*/IR10*IR03*/Y0 +
         IR15*YN*/Y15 +
         IR15*/YN*Y15
```

B.7 INTERRUPT JUMP TABLE FPLA (U207)

```
                 82S153 :  Interrupt Jump Table

             programmed :  1820-2787
               checksum :  454E (Hex)

          ==============================================
*POL HHHHHHHHHH
*P 00    *I L-------   *BI ----------   *BO ....A.....
*P 01    *I HH------   *BI ----------   *BO ....A...A.
*P 02    *I HLL-----   *BI ----------   *BO ....A..A..
*P 03    *I HLHLL---   *BI ----------   *BO ....A..AA.
*P 04    *I HLHHL--H   *BI ----------   *BO ....A.A...
*P 05    *I HLH-H--H   *BI ----------   *BO ....A.A...
*P 06    *I HLHHL-HL   *BI L---------   *BO ....A.A.A.
*P 07    *I HLHHLHHL   *BI HHL-------   *BO ....A.AA..
*P 08    *I HLHHLLHL   *BI HH-H------   *BO ....A.AAA.
*P 09    *I HLHHL-HL   *BI HHHH------   *BO ....A.AAA.
*P 10    *I 00000000   *BI 0000000000   *BO AAAAAAAAAA
*P 11    *I 00000000   *BI 0000000000   *BO AAAAAAAAAA
*P 12    *I 00000000   *BI 0000000000   *BO AAAAAAAAAA
*P 13    *I 00000000   *BI 0000000000   *BO AAAAAAAAAA
*P 14    *I 00000000   *BI 0000000000   *BO AAAAAAAAAA
*P 15    *I 00000000   *BI 0000000000   *BO AAAAAAAAAA
*P 16    *I 00000000   *BI 0000000000   *BO AAAAAAAAAA
*P 17    *I 00000000   *BI 0000000000   *BO AAAAAAAAAA
*P 18    *I 00000000   *BI 0000000000   *BO AAAAAAAAAA
*P 19    *I 00000000   *BI 0000000000   *BO AAAAAAAAAA
*P 20    *I 00000000   *BI 0000000000   *BO AAAAAAAAAA
*P 21    *I 00000000   *BI 0000000000   *BO AAAAAAAAAA
*P 22    *I 00000000   *BI 0000000000   *BO AAAAAAAAAA
*P 23    *I 00000000   *BI 0000000000   *BO AAAAAAAAAA
*P 24    *I 00000000   *BI 0000000000   *BO AAAAAAAAAA
*P 25    *I 00000000   *BI 0000000000   *BO AAAAAAAAAA
*P 26    *I 00000000   *BI 0000000000   *BO AAAAAAAAAA
*P 27    *I 00000000   *BI 0000000000   *BO AAAAAAAAAA
*P 28    *I 00000000   *BI 0000000000   *BO AAAAAAAAAA
*P 29    *I 00000000   *BI 0000000000   *BO AAAAAAAAAA
*P 30    *I 00000000   *BI 0000000000   *BO AAAAAAAAAA
*P 31    *I 00000000   *BI 0000000000   *BO AAAAAAAAAA
*P D9    *I 00000000   *BI 0000000000
*P D8    *I 00000000   *BI 0000000000
*P D7    *I 00000000   *BI 0000000000
*P D6    *I 00000000   *BI 0000000000
*P D5    *I --------   *BI ----------
*P D4    *I --------   *BI ---------L
*P D3    *I --------   *BI ---------L
```

```
*P D2    *I --------    *BI ---------L
*P D1    *I --------    *BI ---------L
*P D0    *I 00000000    *BI 0000000000
```

B.8 SOURCE SPECIAL FPLA (U607)

```
                    82S153 :  Source Special FPLA

                programmed :  1820-2788
                  checksum :  8050 (Hex)

            ==========================================
*POL HHHHHHHHHH
*P 00    *I --------    *BI LH--------    *BO ......A...
*P 01    *I H-------    *BI HLL-----L-    *BO ......A...
*P 02    *I -----H--    *BI HHL-----L-    *BO ......A...
*P 03    *I -H------    *BI HLH-----L-    *BO ......A...
*P 04    *I -------H    *BI HHH-----H-    *BO ......A...
*P 05    *I --------    *BI L-H-------    *BO .......A..
*P 06    *I --------    *BI HLH-----L-    *BO .......A..
*P 07    *I -------H    *BI HHH-----H-    *BO .......A..
*P 08    *I --H-----    *BI L---------    *BO .....A....
*P 09    *I --H-----    *BI HLL-----L-    *BO .....A....
*P 10    *I --H---L-    *BI HHL-----L-    *BO .....A....
*P 11    *I --H-----    *BI HLH-----L-    *BO .....A....
*P 12    *I --H-----    *BI HHH-----L-    *BO .....A....
*P 13    *I --H-----    *BI HHH-----H-    *BO .....A....
*P 14    *I ---H----    *BI L---------    *BO ....A.....
*P 15    *I ---H----    *BI HLL-----L-    *BO ....A.....
*P 16    *I ---H--L-    *BI HHL-----L-    *BO ....A.....
*P 17    *I ---H----    *BI HLH-----L-    *BO ....A.....
*P 18    *I ---L----    *BI HHH-----LL    *BO ....A.....
*P 19    *I ---H----    *BI HHH-----H-    *BO ....A.....
*P 20    *I ----H---    *BI ----------    *BO ...A......
*P 21    *I 00000000    *BI 0000000000    *BO AAAAAAAAAA
*P 22    *I 00000000    *BI 0000000000    *BO AAAAAAAAAA
*P 23    *I 00000000    *BI 0000000000    *BO AAAAAAAAAA
*P 24    *I 00000000    *BI 0000000000    *BO AAAAAAAAAA
*P 25    *I 00000000    *BI 0000000000    *BO AAAAAAAAAA
*P 26    *I 00000000    *BI 0000000000    *BO AAAAAAAAAA
*P 27    *I 00000000    *BI 0000000000    *BO AAAAAAAAAA
*P 28    *I 00000000    *BI 0000000000    *BO AAAAAAAAAA
*P 29    *I 00000000    *BI 0000000000    *BO AAAAAAAAAA
*P 30    *I 00000000    *BI 0000000000    *BO AAAAAAAAAA
*P 31    *I 00000000    *BI 0000000000    *BO AAAAAAAAAA
*P D9    *I 00000000    *BI 0000000000
*P D8    *I 00000000    *BI 0000000000
*P D7    *I 00000000    *BI 0000000000
*P D6    *I --------    *BI ----------
*P D5    *I --------    *BI ----------
*P D4    *I --------    *BI ----------
*P D3    *I --------    *BI ----------
```

```
*P D2    *I --------    *BI ----------
*P D1    *I 00000000    *BI 0000000000
*P D0    *I 00000000    *BI 0000000000
```

B.9 I/O HANDSHAKE FPLA (U709)

```
                 82S153 :  I/O Handshake and MEMGO-Killer State Machine

             programmed :  1820-2789
               checksum :  5836 (Hex)

        ==============================================
*POL HLHHHHHHHH
*P 00    *I -LLL-LLH    *BI ----------    *BO ......A...
*P 01    *I -LLLL-LH    *BI ----------    *BO ......A...
*P 02    *I -LHL-LLH    *BI ----------    *BO A....A....
*P 03    *I -LHLL-LH    *BI ----------    *BO .....A....
*P 04    *I -H-L--H-    *BI ----------    *BO A.A..A....
*P 05    *I -H-L--L-    *BI ----------    *BO ..A..A....
*P 06    *I -H-H--HH    *BI ----------    *BO A.A..AA...
*P 07    *I -H-H--LH    *BI ----------    *BO ..A..AA...
*P 08    *I --------    *BI -------L--    *BO ....A.....
*P 09    *I H-------    *BI --------HH    *BO ....A.....
*P 10    *I H-------    *BI -------HHL    *BO .A........
*P 11    *I --------    *BI ---L---HH-    *BO .A........
*P 12    *I 00000000    *BI 0000000000    *BO AAAAAAAAAA
*P 13    *I 00000000    *BI 0000000000    *BO AAAAAAAAAA
*P 14    *I 00000000    *BI 0000000000    *BO AAAAAAAAAA
*P 15    *I 00000000    *BI 0000000000    *BO AAAAAAAAAA
*P 16    *I 00000000    *BI 0000000000    *BO AAAAAAAAAA
*P 17    *I 00000000    *BI 0000000000    *BO AAAAAAAAAA
*P 18    *I 00000000    *BI 0000000000    *BO AAAAAAAAAA
*P 19    *I 00000000    *BI 0000000000    *BO AAAAAAAAAA
*P 20    *I 00000000    *BI 0000000000    *BO AAAAAAAAAA
*P 21    *I 00000000    *BI 0000000000    *BO AAAAAAAAAA
*P 22    *I 00000000    *BI 0000000000    *BO AAAAAAAAAA
*P 23    *I 00000000    *BI 0000000000    *BO AAAAAAAAAA
*P 24    *I 00000000    *BI 0000000000    *BO AAAAAAAAAA
*P 25    *I 00000000    *BI 0000000000    *BO AAAAAAAAAA
*P 26    *I 00000000    *BI 0000000000    *BO AAAAAAAAAA
*P 27    *I 00000000    *BI 0000000000    *BO AAAAAAAAAA
*P 28    *I 00000000    *BI 0000000000    *BO AAAAAAAAAA
*P 29    *I 00000000    *BI 0000000000    *BO AAAAAAAAAA
*P 30    *I 00000000    *BI 0000000000    *BO AAAAAAAAAA
*P 31    *I 00000000    *BI 0000000000    *BO AAAAAAAAAA
*P D9    *I --------    *BI ----------
*P D8    *I --------    *BI ----------
*P D7    *I --------    *BI ----------
*P D6    *I 00000000    *BI 0000000000
*P D5    *I --------    *BI ----------
*P D4    *I --------    *BI ----------
*P D3    *I --------    *BI ----------
```

```
*P D2     *I 00000000     *BI 0000000000
*P D1     *I 00000000     *BI 0000000000
*P D0     *I 00000000     *BI 0000000000
```

VCP, LOADERS, AND SELF-TEST PROGRAMS

APPENDIX C

This appendix contains a listing of the Test 2 portion of the self-test, loaders, and Virtual Control Panel programs contained in ROM located on the memory controller card.

The 4k ROM code is identified by HP part number 5180-0189 and 5180-0190 and is contained in ICs U606 and U706, respectively, on the memory controller card. The sockets for U606 and U706 will also accept most 8k EPROMs (2764s).

A user who intends to change the ROM code for any reason should keep in mind the considerations described below.

In A600 computers, sockets accommodating 8k PROM parts are provided. (The A700 computers provide two additional ROM sockets for user loaders implemented in 4K parts (2732).) A user who wants to create his own loaders should burn the current VCP code into the first half of the 8k parts (2764) and his own loader code into the second half. The start-up switches on the processor card may be set to execute this code on power-up; the code may also be invoked by VCP commands when the VCP program is being run.

The VCP address space is separate from the main memory of the computer, consisting of 1k words of RAM in the base page and 4k or 8k words of ROM space. The VCP program provided with the computer occupies 4k of ROM space (octal addresses 20000 to 27777). Additional space from addresses 30000 to 37777 may be assigned to user loaders, as described above, with user code starting at location 30002. Thus, locations 30000 and 30001 may be used for revision code and checksum. The RAM area of the VCP address space can be accessed only by the VCP or microcode. However, the VCP can access main memory through the use of cross-map instructions. Because the VCP memory area is not mapped, the VCP can execute even when the maps or main memory is not functional.

As the VCP runs from ROM, any instruction that might need to modify ROM cannot be used. Thus, JLB instructions are used for subroutine linkage, rather than JSB instructions.

When the VCP mode is enabled, trap cells for interrupts are in the VCP RAM address space, but DMA self-configuration quadruplets are not, as all DMA transfers still access main memory. In order to test DMA, the VCP reserves the last 64 locations of page 0 in main memory; these locations also are used by the VCP for passing the command string to BOOTEX or diagnostics.

The base page of VCP RAM is divided as follows:

00000 to 00077	Reserved for trap cells.
00100 to 00177	Reserved for microcode use.
00200 to 00777	Reserved for HP-supplied VCP.
01000 to 01377	Available for user ROM code (loaders or power-up).
01400 to 01777	Reserved for error logging in A700 computers.

The VCP program is divided into four pages (page 0 to page 3). Page 0 contains the Pretest (Test 2 portion of the self-test). Page 1 contains the user interface. Page 2 contains the drivers for the ASIC card, the intelligent interface cards, and the DS loader. Page 3 contains the ROM loader, the CTU loader, and the disc loader.

User ROM code can call the existing loader routines through a jump table located at the beginning of page 3.

The following is a summary of points to remember about VCP addressing:

1. The boot ROM code space begins at address 20000 octal and continues to 37777 octal. Addresses above 37777 produce undefined results.

2. Boot RAM space is from 00002 to 17777 octal, but in the A600 and A700 computers, only 1k of boot RAM is installed (addresses 2 to 1777 octal).

3. Portions of boot RAM have been set aside for system functions and may not be used for other purposes.

The following pages contain a sample 4k-ROM listing of the VCP, loaders, and self-test programs. As ROM firmware is subject to change, later versions will differ in minor details from.what is shown in this listing. (Note that there is a Cross Reference Symbol Table at the end of the listing.)

A600[illegible] CLK [illegible]

```
                  MACRO,A,Q=S,C
*     A   -> ABSOLUTE ASSEMBLY
*     Q=S -> SHORT LISTING
*     C   -> PRINT CROSS REFERENCE TABLE
*
*********************************************************************
*
*  NAME: &VCP
*
*  SOURCE: 24998-18540
*
*  BURN TAPE: 24998-16540 AND 24998-16541
*
*  ROMS: 5180-0189 HIGH BYTE AND 5180-0190 LOW BYTE
*
*  PGMR: D.A.F.
*
*  LAST MODIFIED: 820706.0954
*
*****************************************************************
* (C) COPYRIGHT HEWLETT PACKARD COMPANY 1982. ALL RIGHTS        *
* RESERVED. NO PART OF THIS PROGRAM MAY BE PHOTOCOPIED,         *
* REPRODUCED, OR TRANSLATED TO ANOTHER PROGRAM LANGUAGE WITHOUT *
* THE PRIOR WRITTEN CONSENT OF HEWLETT PACKARD COMPANY          *
*****************************************************************
*
*
*
*
```

```
*
          020000  EPROM EQU 20000B
*
                        MACLIB ^DMS     ;PHOENIX OPCODE MACRO FILE
*
*
*
*  the first 64 locations of boot memory are reserved for trap cells
*
*
*
   00100                ORG 100B
*
*     VIRTUAL REGISTER AREA FOR PROCESSOR (64 LOCATIONS)
*
   00100 000000  WMAP   NOP                 OLD WMAP VALUE ON ENTRY
*
*
*
          000001  CPUST EQU 1
*
*     CPU STATUS IS OBTAINED BY A LIA/B 1
*  SW 1   BIT 8 = BOOT SELECT 0
*     2       9 = BOOT SELECT 1
*     3      10 = BOOT SELECT 2
*     4      11 = BOOT SELECT 3
*     5      12 = SELECT ALTERNATE VCP DRIVER
*     6      13 = RESERVED
*     -      14 = MEMORY LOST  (LOW TRUE) ONLY valid for 5 ms
*     8      15 = INTERRUPT MASK BIT 1 (TBG MASK)
*
*          SWITCH 7 IS RESERVED ON THE PROCESSOR FOR INT/EXT CLOCK
*
*
*     CPU CONTROL OUTPUT BY AN OTA/B 1
*     BIT 0-7 = STATUS LIGHT 0-7
*
                        MIC .JLB,104600B,1
                        MIC .JLA,100600B,1
*
   00200                ORG 200B
```

```
*
*     BASE PAGE STORAGE LOCATIONS
*
         000000  A      EQU 0
         000001  B      EQU 1
         000002  GR     EQU 2
         000030  DATA   EQU 30B
         000032  STATS  EQU 32B
         000031  CMND   EQU 31B
         000030  DATA   EQU 30B
  00200  000000  SAVEI  NOP
  00201  000000  SAVEO  NOP
  00202  000000  SAVEE  NOP
  00203  000000  SAVEP  NOP
  00204  000000  SAVEA  NOP
  00205  000000  SAVEB  NOP
  00206  000000  SAVEG  NOP
  00207  000000  SAVEX  NOP
  00210  000000  SAVEY  NOP
  00211  000000  SAVEQ  NOP
  00212  000000  SAVEZ  NOP
  00213  000000  SAVEM  NOP
  00214  000000  SAVEW  NOP
*
  00215  000000  MLOST      NOP  MEMORY LOST FLAG LOW TRUE IN SIGN BIT
  00216  000000  D1SV       NOP  DATA 1 MAP SAVE FOR %CLEAR MEMORY
  00217  000000  PNTR       NOP
  00220  000000  PNTRS      NOP
  00221  000000  SVCHR      NOP
  00222  000000  SACOMN     NOP
  00223  000000  CTR        NOP
  00224  000000  MCNTR      NOP  COUNT FOR MAP DISPLAY
  00225  000000  PCNTR      NOP  PAGE COUNT FOR MAP DISPLAY
  00226  000000  PUTCT      NOP  CHAR COUNT FOR PUTS
  00227  000000  PETMP      NOP
  00230  000000  PERTN      NOP  RETURN ADDRESS FROM PE ROUTINE
  00231  000232  PEJMPI     JMP PE,I  PUT HERE DURING EXECUTION
  00232  000000  PE         NOP  PLACE FOR DEF TO PEINT ROUTINE
  00233  000000  TBG        NOP  DEF TO TBG ROUTINE
  00234  000000  ILI        NOP
  00235  000000  PFW        NOP  ETC
  00236  000000  MPT        NOP
  00237  000000  UITRTN     NOP  RETURN ADDRESS FROM UIT ROUTINE
  00240  000241  UIJMPI     JMP  UIT,I  PUT HERE DURING EXECUTION
  00241  000000  UIT        NOP
  00242  000000  INTIO      NOP  DEF TO I/O INT ROUTINE
  00243  000000  PEFLAG     NOP  1 IF PARITY ERROR DURING LAST COMMAND
  00244  000000  DISPLAY    NOP  ERROR DISPLAY
```

```
00120  00245 000000  TBGCNT    NOP  COUNT FOR 10 MS FROM TBG
00121  00246 000000  MSIZE     NOP  NUMBER OF 32K BLOCKS OF PHYSICAL MEMORY
00122  00247 000000  ECCCNT    NOP  NUMBER OF 32K ECC BLOCKS
00123  00250 000000  CORCNT    NOP  NUMBER OF SINGLE BIT CORRECTIONS
00124  00251 000000  CNTR      NOP
00125  00252 000000  TRYCT     NOP  RETRY COUNTER FOR AUTO BOOT
00126  00253 000000  DCTO      NOP  TIME OUT FOR DISC LOADER
00127  00254 000000  MPTR      NOP         POINTER TO MAP REG BEING OUTPUT
00128  00255 000000  PPNTR     NOP
00129  00256 000000  BASE      NOP           0 => OCTAL, -1 => HEX
00130  00257 000000  HPIT      NOP
00131  00260 000000  TEMP      NOP
00132  00261 000000  CHAR      NOP
00133  00262 000000  RFTMP     NOP  TEMPORARY FOR RF ROUTINE
00134  00263 000000  IORGN     NOP  I/O REGISTER NUMBER FOR RXX COMMAND
00135  00264 000000  SCETC     NOP  2127 OR WHATEVER TYPED IN AFTER LOAD OR
                                      BOOT
00136  00265 000000  LERR      NOP  LOADER ERROR
00137  00266 000000  PARTIAL   NOP  PARTIAL COUNT FOR DISC LOADER
00138*
00139  00267 000000  UNIT           NOP
00140  00270 000000  SUBCH          NOP
00141  00271 000000  DISC.ID        NOP
00142  00272 000000  UNIT.HEAD      NOP           FLAG
00143  00273 000000  CYLNDR.OFFSET  NOP
00144  00274 000000  FILE           NOP           VECTOR WORD 1
00145  00275 000000  HEAD.CYLINDER  NOP           VECTOR WORD 2
00146  00276 000000  SECTR.TRACK    NOP           VECTOR WORD 3
00147  00277 000000  VW1            NOP           ; WHEN TALKING TO
00148  00300 000000  VW2            NOP           ;  LINUS THESE WILL
00149  00301 000000  VW3            NOP           ;   DIFFER FROM ABOVE
00150*
00151  00302 000000  PEADD     NOP    PARITY ADDRESS
00152  00303 000000  PEMAP     NOP    BLOCK FOR PARITY ADDRESS
00153  00304 000000  VCPTFLG   NOP    FLAG FOR %TEST COMMAND
00154  00305 000000  TRAPFLAG  NOP    FLAG FOR TRAP CELLS CLOBBERED
00155  00306         STRNG     BSS 40   ;BOOT COMMAND STRING (ALLOW 80
                                           CHARACTERS)
00156*
00157  00356 000000  LSTR      NOP            LENGTH OF STRING
00158  00357 000000  GSLR      NOP            LEFT/RIGHT BYTE FLAG
00159  00360 000000  STORE.POINTER     NOP   POINTER TO STRNG
00160  00361 000000  DPNTR     NOP
00161  00362 000000  BFLAG     NOP
00162  00363 000000  DFLAG     NOP  DIGIT FLAG  MSB = 1 => ONE DIGIT
00163  00364 000000  RFLAG     NOP  ROM FLAG USED IN "TREG" ROUTINE
00164  00365 000000  TFLAG     NOP  TRACE FLAG  MSB = 1 => TRACE IN PROGRESS
00165  00366 000000  MAP       NOP          CURRENT MAP
00166  00367 000000  PAGE      NOP          CURRENT PAGE
```

```
00370        MPBUF     BSS 32       COPY OF CURRENT MAP
00430        MZSV      BSS 32       COPY OF MAP ZERO
00470 000000 DIG1      NOP
00471 000000 DIG2      NOP
00472 000000 DIG3      NOP
00473 000000 DIG4      NOP
00474 000000 DIG5      NOP
00475 000000 DIG6      NOP
00476 000000 DIGS      NOP
*
00477 000000 PO.CT     NOP
00500 000000 PO.T3     NOP
00501 000000 TEMP2     NOP
00502 000000 TEMP1     NOP
00503 000000 TEMPO     NOP
00504 000000 TEMP3     NOP
00505 000000 PO.A      NOP
00506 000000 PO.B      NOP
*
00507 000000 FIRST     NOP
00510 000000 NDCLR     NOP
00511 000000 POINTER   NOP
00512 000000 SIDE?     NOP
00513 000000 VCP.FLAG  NOP   IS THERE A VCP??
00514 000000 VCPSC     NOP   SELECT CODE OF VCP
00515 000000 ASFLG     NOP   NONZERO FOR TIC, ZERO FOR DS
*
00516 000000 EXLOAD    NOP   EXTENDED LOAD COUNTER FOR DS LOADER
00517 000000 P3.CT     NOP   RECORD COUNT FOR DS LOADER
00520 000000 .PU       NOP   ASCII P AND UNIT NUMBER FOR CTU LOADER
00521 000000 DSCNT     NOP   WORD COUNT FOR ABS BINARY IN DS LOADER
00522 000000 DSADD     NOP   ADDRESS COUNT FOR DS LOADER
00523 000000 DSCHK     NOP   CHECKSUM FOR DS LOADER
*
00524 000000 XEQT  NOP       THIS IS USED IN THE I/O
00525 000000       NOP        REGISTER ROUTINE
00526 000524      JMP XEQT,I PLANTED HERE DURING EXECUTION
```

```
00204*
00205*     THESE ARE THE SUBROUTINE RETURN REGISTERS
00206*
00207  00527 000000  RPUTS NOP
00208  00530 000000  RENDV NOP
00209  00531 000000  RENQAK NOP
00210  00532 000000  RGETS NOP
00211  00533 000000  ROUT1 NOP
00212  00534 000000  ROUTN NOP
00213  00535 000000  RLCH1 NOP
00214  00536 000000  RLCHR NOP
00215  00537 000000  ROUTD NOP
00216  00540 000000  RECHO NOP
00217  00541 000000  RPUTC NOP
00218  00542 000000  RGETC NOP
00219  00543 000000  RGETREG NOP
00220  00544 000000  RGETN NOP
00221  00545 000000  RSCNSC NOP
00222  00546 000000  RRSTO NOP
00223  00547 000000  RCOMN NOP
00224*
00225  00550 000000  RCTU  NOP
00226  00551 000000  RTI.W NOP
00227  00552 000000  RTI.B NOP
00228  00553 000000  RTO.B NOP
00229  00554 000000  RTO.W NOP
00230  00555 000000  RCTIO NOP
00231*
00232  00556 000000  RRMLD  NOP
00233  00557 000000  RDCLD  NOP
00234  00560 000000  RPHI?  NOP
00235  00561 000000  RPHI   NOP
00236  00562 000000  RPHII  NOP
00237  00563 000000  RHPIB  NOP
00238  00564 000000  RPHIF  NOP
00239  00565 000000  RHPIBX NOP
00240  00566 000000  RDCIN  NOP
00241  00567 000000  RDTPC  NOP
00242  00570 000000  RDCRW  NOP
00243  00571 000000  RDSLD  NOP
00244  00572 000000  RS.SC  NOP
00245  00573 000000  RDS.B  NOP
00246  00574 000000  RDS.GT NOP
00247  00575 000000  RCI.IZ NOP
00248  00576 000000  RCI.ID NOP
00249  00577 000000  RTG.BF NOP
00250  00600 000000  RTG.TB NOP
```

```
 00601 000000  RCL.IZ NOP
 00602 000000  RDS.FT NOP
 00603 000000  RDS.CM NOP
 00604 000000  RCS.FT NOP
 00605 000000  RCS.CM NOP
 00606 000000  ROUT2C NOP
*
 00607 000000  RGT01  NOP
 00610 000000  RI.O   NOP
*
*
*     256 LOCATIONS RESERVED FOR USER ROM CODE
*
 01000                ORG 1000B
 01000                BSS 256
*
*     LAST 256 LOCATIONS RESERVED FOR ERROR LOGGING
 01400                ORG 1400B
 01400                BSS 256
       002000  LAST   EQU *
*
*
 20000               ORG EPROM
```

```
       020000  P0    EQU *          PAGE 0 REFERENCE
*    I.  PRETEST
         THE PRETEST IS USED TO VERIFY EXECUTION OF THE BASIC
         INSTRUCTIONS USED IN THE BOOT LOADERS.  THE ASUMPTION IS
         MADE THAT THE JMP INSTRUCTION IS FUNCTIONAL AND WILL BE
         USED TO STOP EXECUTION.  THE PRETEST IS NOT INTENDED TO
         BE A COMPLETE CHECK OF THE CPU, BUT ONLY THAT THE INSTRUC-
         TIONS USED IN THE BOOT ARE FUNCTIONAL SO THAT A BOOT LOAD
         MAY BE POSSIBLE.
*
20000 000006  RVCODE OCT 6          CONSTANT   (REV CODE GOES HERE)
20001 000000  CHKSUM NOP            CHECKSUM SPOT


*    THE FOLLOWING INSTRUCTIONS CHECK THE CPU ONLY
*    MACROCODE EXECUTION STARTS HERE AFTER POWER UP OR RESET
*
20002 002400  START CLA
20003 000304        STA VCPTFLG     NO TEST, POWER UP
20004 102501        LIA CPUST       GET MLOST BIT
20005 000215        STA MLOST
20006 021711        LDA B3          TRY TO INDICATE IN INSTRUCTION TEST
20007 003000        CMA
20010 102601        OTA CPUST
20011 002701        CLA,CCE,RSS     A=000000  B=XXXXXX  E=1  O=X  +SKP
20012 020012        JMP *           RSS FAILED
20013 006440        CLB,SEZ         A=000000  B=000000  E=1  O=X  -SKP
20014 002102        CLE,SZA         A=000000  B=000000  E=0  O=X  +SKP
20015 020015        JMP *           CCE-SEZ OR CLA-SZA FAILED
20016 003041        CMA,SEZ,RSS     A=177777  B=000000  E=0  O=X  -SKP
20017 006202        CME,SZB         A=177777  B=000000  E=1  O=X  +SKP
20020 020020        JMP *           CCE OR CLB-SZB FAILED
20021 007040        CMB,SEZ         A=177777  B=177777  E=1  O=X  -SKP
20022 006003        SZB,RSS                                       +SKP
20023 020023        JMP *           CME OR CMB FAILED
20024 000001        CPA B                                         -SKP
20025 002414        CLA,SLA,INA     A=000001  B=177777  E=1  O=X  +SKP
20026 020026        JMP *           CMA-CPA B-SLA,INA FAILED
20027 002002        SZA                                           -SKP
20030 002020        SSA                                           +SKP
20031 020031        JMP *           INA OR SSA FAILED
20032 006400        CLB             A=000001  B=000000  E=1  O=X
20033 003420        CCA,SSA         A=177777  B=000000            -SKP
20034 002003        SZA,RSS                                       +SKP
20035 020035        JMP *           CCA-SSA OR SZA,RSS FAILED
20036 002010        OCT 002010      ASG SLA                       -SKP
20037 002131        CLE,SSA,SLA,RSS A=177777  B=000000  E=0  O=X +SKP
20040 020040        JMP *           SLA OR SSA,SLA,RSS FAILED
20041 102101        STO             A=177777  B=000000  E=0  O=1
```

```
00324  20042 102201        SOC                                                -SKP
00325  20043 102301        SOS                                                +SKP
00326  20044 020044        JMP *        STO-SOC-SOS FAILED
00327  20045 103101        CLO          A=177777  B=000000  E=0  O=0
00328  20046 102301        SOS                                                -SKP
00329  20047 102201        SOC                                                +SKP
00330  20050 020050        JMP *        CLO-SOS-SOC FAILED

00332  20051 021761        LDA ALT1     A=125252  B=000000  E=0  O=0
00333  20052 006003        SZB,RSS
00334  20053 000001        CPA B                                              +SKP
00335  20054 020054        JMP *        CPA B OR CLB-SZB,RSS FAILED
00336  20055 021761        CPA ALT1                                           -SKP
00337  20056 000001        STA B        A=125252  B=125252  E=0  O=0
00338  20057 021761        LDA ALT1
00339  20060 000000        CPB A                                              +SKP
00340  20061 003401        CCA,RSS      A=177777  B=125252  E=0  O=0  +SKP
00341  20062 020062        JMP *        CPA-STA-CPB FAILED
00342  20063 021760        AND ALT0     A=052525  B=125252  E=0  O=0
00343  20064 021760        CPA ALT0                                           -SKP
00344  20065 002001        RSS                                                +SKP
00345  20066 020066        JMP *        AND-CPA FAILED
00346  20067 021761        AND ALT1     A=000000  B=125252  E=0  O=0
00347  20070 002002        SZA                                                +SKP
00348  20071 020071        JMP *        AND FAILED
00349  20072 021721        LDA B24      A=000024  B=125252  E=0  O=0
00350  20073 021760        IOR ALT0     A=052525  B=125252  E=0  O=0
00351  20074 021760        CPA ALT0                                           -SKP
00352  20075 003401        CCA,RSS      A=177777  B=125252  E=0  O=0  +SKP
00353  20076 020076        JMP *        XOR FILED
00354  20077 021761        XOR ALT1     A=052525  B=125252  E=0  O=0
00355  20100 021760        CPA ALT0                                           -SKP
00356  20101 002440        CLA,SEZ      A=000000  B=125252  E=0  O=0  +SKP
00357  20102 020102        JMP *        IOR-XOR FAILED
00358  20103 021761        ADA ALT1     A=125252  B=125252  E=0  O=0
00359  20104 021761        CPA ALT1                                           -SKP
00360  20105 002040        SEZ                                                +SKP
00361  20106 020106        JMP *        CLA OR ADA FAILED
00362  20107 021760        ADA ALT0     A=177777  B=125252  E=0  O=0
00363  20110 102301        SOS                                                -SKP
00364  20111 003002        CMA,SZA      A=000000  B=125252  E=0  O=0  +SKP
00365  20112 020112        JMP *        ADA FAILED
00366  20113 003440        CCA,SEZ      A=177777  B=125252  E=0  O=0  +SKP
00367  20114 020114        JMP *        ADA FAILED
00368  20115 021751        ADA M1       A=177776  B=125252  E=1  O=0
00369  20116 021752        CPA .N2                                            -SKP
00370  20117 002041        SEZ,RSS                                            +SKP
00371  20120 020120        JMP *        ADA FAILED
```

```
00372  20121 102301        SOS                                                 -SKP
00373  20122 002101        CLE,RSS       A=177776  B=125252  E=0  O=0  +SKP
00374  20123 020123        JMP *         ADA FAILED
00375  20124 000000        ISZ A         A=177777  B=125252  E=0  O=0  -SKP
00376  20125 000000        ISZ A         A=000000  B=125252  E=0  O=0  +SKP
00377  20126 020126        JMP *         ISZ FAILED

00379  20127 021743        LDA B100K     A=100000  B=125252  E=0  O=0
00380  20130 021751        ADA M1        A=077777  B=125252  E=1  O=1
00381  20131 102201        SOC                                                 -SKP
00382  20132 002141        SEZ,CLE,RSS   A=077777  B=125252  E=0  O=1  +SKP
00383  20133 020133        JMP *         ADA FAILED
00384  20134 103101        CLO           A=077777  B=125252  E=0  O=0
00385  20135 002004        INA           A=100000  B=125252  E=0  O=1
00386  20136 021743        CPA B100K                                           -SKP
00387  20137 002040        SEZ                                                 +SKP
00388  20140 020140        JMP *         ADA FAILED
00389  20141 000001        LDA B         A=125252  B=125252  E=0 O=1
00390  20142 021761        CPA ALT1                                            -SKP
00391  20143 103301        SOS C         A=125252  B=125252  E=0  O=0  +SKP
00392  20144 020144        JMP *         B-REG. WAS MODIFIED

00394*     THE FOLLOWING SEQUENCE IS USED TO CHECK
00395*     JLA, JMP X,I, AND STA X,I
00396*
00397  20145 020207        LDA PTJPR     WILL GET RETURN ADDR IN A
00398  20146 000000        LDB A
00399  20147 100600        .JLA PTRTO    JLA TO PTRTO
       20150 020152
00400  20151 020151        JMP *         JLA FAILED
00401  20152 020206  PTRTO CPA PTDF1     CORRECT RETURN ADDRESS?
00402  20153 002301        CCE,RSS       YES
00403  20154 020154        JMP *         NO
00404  20155 002400        CLA           CLEAR A
00405  20156 006400        CLB             &   B
00406  20157 101741        CAX             &   X
00407  20160 105751        CBY             &   Y
00408  20161 105762        JLY PTRT1     JMP & LOAD Y W/P+2
       20162 020164
00409  20163 020163        JMP *          DID NOT MAKE IT
00410  20164 002002  PTRT1 SZA           A STILL CLEAR ?
00411  20165 020165        JMP *
00412  20166 101754        CYA           COPY Y TO A
00413  20167 020211        CPA PTJYO     P+2 ?
00414  20170 006002        SZB            YEP , B STILL ZERO ?
00415  20171 020171        JMP *         LOOSE
```

```
20172 105744       CXB            CHECK X WHILE WE ARE AT IT
20173 006002       SZB             ?
20174 020174       JMP *           UH UH!
20175 020205       LDA PTDF0      SET PAGE ADDRESS
20176 021707       STA B1,I       PUT IT IN B-REG. INDIRECTLY
20177 000001       CPA B
20200 020205       LDA PTDF0,I
20201 021710       ADB B2         POINT PAST CONSTANTS & SUCH
20202 020210       CPA PTJMP      INDIRECT OK?
20203 000000       JMP 0          YES EXECUTE B-REG.
20204 020204       JMP *
20205 020210 PTDF0 DEF *+3
20206 020151 PTDF1 DEF PTRT0-1
20207 020152 PTJPR JMP PTRT0
20210 000001 PTJMP JMP 1,I
20211 020163 PTJY0 DEF PTRT1-1

20212 021762       LDA SRGP1          B-REG.      E        A-REG.
20213 000000       LDB A          1000100100100111 1
20214 021763       LDA SRGP2                       1 1001100000100000
20215 005025       BLS,ERB        1100100100100111 0
20216 005661       ELB,CLE,BRS    1100100100100111 0
20217 001124       ARS,ALR                         0 0001100000100000
20220 005026       BLS,ELB        0100100100100111 0
20221 005523       ERB,RBR        0100100100100111 0
20222 001720       ALF,ALS                         0 1000010000000010
20223 005124       BRS,BLR        0100100100100110 0
20224 001330       RAR,SLA,ALS                     0 0000010000000010
20225 005221       RBL,BRS        1100100100100110 0
20226 002300       CCE                             1
20227 001726       ALF,ELA                         0 1000000001000001
20230 001522       ERA,RAL                         1 1000000001000000
20231 005427       BLR,BLF        0010010011000001 1
20232 001122       ARS,RAL                         1 1000000001000001
20233 005220       RBL,BLS        0001001100000100 1
20234 001135       ARS,SLA,ERA                     0 1110000000010000
20235 020235       JMP *          SLA FAILED
20236 001623       ELA,RAR                         1 0110000000010000
20237 005327       RBR,BLF        1001100000100000
20240 002040       SEZ            CHECK E-REG.
20241 001460       ALR,CLE,ALS                       0000000001000000
20242 021764       CPA SRGP3
20243 102201       SOC
20244 020244       JMP *          SRG INST A-REG.
20245 000001       LDA B          CHANGE HANDS
20246 021763       CPA SRGP2
20247 006640       CLB,SEZ,CME
20250 020250       JMP *          SRG INST B-REG.
```

```
00465  20251 102101          STO           START WITH O SET
00466  20252 021767          LDA BEAUS     SET B=130272
00467  20253 000000          LDB A           AND
00468  20254 021765          LDA AEAUS         A=076310         E=1
00469  20255 101021          ASR 1         A=037144  B=154135  E=1  O=0
00470  20256 102301          SOS
00471  20257 100117          RRL 15        A=066056  B=117462  E=1  O=0
00472  20260 100022          ASL 2         A=130270  B=176311  E=1  O=1
00473  20261 102201          SOC
00474  20262 101100          RRR 16        A=176311  B=130270
00475  20263 100041          LSL 1         A=174622  B=060561
00476  20264 101025          ASR 5         A=107714  B=001413       O=0
00477  20265 021770          CPA ASR.0     CHECK PRELIMINARY RESULTS
00478  20266 102201          SOC
00479  20267 020267          JMP *         EAU SHIFT FAILED
00480  20270 101040          LSR 16        A=01413   B=0
00481  20271 006002          SZB           INSURE B WAS CLEARED
00482  20272 020272          JMP *         WAS NOT -EAU SHIFT FAILED
00483  20273 102101          STO
00484  20274 100020          ASL 16        A=0       B=001413
00485  20275 102301          SOS
00486  20276 100026          ASL 6         A=0       B=041300
00487  20277 102201          SOC
00488  20300 101100          RRR 16        A=041300  B=0
00489  20301 021771          CPA ASR.1     FINAL OK?
00490  20302 006002          SZB
00491  20303 020303          JMP *         NO -EAU SHIFT FAILED
00492  20304 021742          LDA B76K      A=076000  B=XXXXXX  E=X  O=X
00493  20305 102101          STO                                    O=1
00494  20306 100200          MPY B6412     A=154000  B=003120  E=X  O=0
       20307 021740
00495  20310 102201          SOC
00496  20311 020311          JMP *         O WAS NOT CLEARED BY MPY
00497  20312 100400          DIV ALT1      A=166416  B=020264
       20313 021761
00498  20314 100200          MPY MU2       A=156224  B=002046
       20315 021772
00499  20316 100400          DIV B7777     A=041161  B=007405
       20317 021741
00500  20320 100101          RRL 1         A=102342  B=017012
00501  20321 100200          MPY ALTO      A=024412  B=15336
       20322 021760
00502  20323 100400          DIV B76K      A=125507  B=142412
       20324 021742
00503  20325 100200          MPY ALT1      A=161446  B=016075
       20326 021761
00504  20327 102101          STO                               O=1
00505  20330 100400          DIV DV4       A=126760  B=006606
       20331 021773
```

```
00506  20332 021774        CPA RESUA      RESULT IN A
00507  20333 102201        SOC            O=0
00508  20334 020334        JMP *          MPY OR DIV FAILED
00509  20335 101100        RRR 16         CHANGE HANDS
00510  20336 021775        CPA RESUB      RESULT IN B
00511  20337 020341        JMP *+2
00512  20340 020340        JMP *          MPY OR DIV ERROR
00513  20341 100400        DIV B1         TRY OVER FLOW
       20342 021707
00514  20343 102301        SOS            WAS IT ?
00515  20344 020344        JMP *          NO
00516  20345 100400        DIV .DO        TRY ZERO TO SET OVER FLOW
       20346 021706
00517  20347 102301        SOS            WAS IT ?
00518  20350 020350        JMP *          NO
00519*                    TEST SBT AND LBT
00520  20351 003004        CMA,INA
00521  20352 021766        LDB BLBT       GET DEF TO LBT THING
00522  20353 005200        RBL            MAKE IT A BYTE ADDRESS
00523  20354 105763        LBT            A HAS HIGH BYTE
00524  20355 001727        ALF,ALF
00525  20356 000260        STA TEMP       SAVE ONE BYTE
00526  20357 105763        LBT            GET OTHER BYTE
00527  20360 000260        IOR TEMP       GET OTHER BYTE BACK
00528  20361 021765        CPA AEAUS
00529  20362 002001        RSS
00530  20363 020363        JMP *


00532*
00533*
00534*     AT THIS POINT THE BASIC INSTRUCTION TEST HAS PASSED
00535*
00536*     VERY DESTRUCTIVE TEST OF 1K VCP RAM - CLEARS RAM MEMORY
00537*
00538*
00539  20364 105745        LDX VCPTFLG  MUST SAVE AND RSTORE VCPTFLAG
       20365 000304
00540  20366 105755        LDY MLOST    MUST SAVE AND RESTORE MLOST FLAG BIT
       20367 000215
00541  20370 021715        LDA B7
00542  20371 003000        CMA
00543  20372 102601        OTA CPUST      INDICATE IN BOOT RAM TEST
00544  20373 021727        LDA B100       RAM MEMORY TEST
00545  20374 006400        CLB
00546  20375 101105   .MELO RRR 5
```

```
20376 000000        STA @A        PUT ADDRESS IN LOCATION
20377 000000        LDB A
20400 000000        CPB @A        DID IT STORE?
20401 007001        CMB,RSS       YES
20402 020402        JMP *          BUMMER !
20403 000000        STB @A        SAVE COMPLEMENT
20404 000000        CPB @A
20405 006401        CLB,RSS       NEXT LOCATION
20406 020406        JMP *         DIDNT STORE, BAD BOOT RAM
20407 000000        STB @A        STORE ZERO
20410 000000        CPB @A        DID IT STORE?
20411 002005        INA,RSS
20412 020412        JMP *
20413 100105        RRL 5         TEST BIT 10
20414 002021        SSA,RSS       DONE 1K ?
20415 020375        JMP .MELO     NOT YET
*
20416 105743        STX VCPTFLG   RESTORE VCP TEST FLAG
20417 000304
20420 105753        STY MLOST     RESTORE MLOST BIT
20421 000215

*
*     THE BOOT RAM TEST HAS PASSED
*
*     SET UP TRAP CELLS
*
20422 020000        LDA RVCODE
20423 000205        STA SAVEB     REV CODE IN THE B REGISTER
20424 021505        LDA ILDEF
20425 000234        STA ILI       DEF TO ILINT IN LOCATION ILI
20426 000232        STA PE
20427 000233        STA TBG
20430 000235        STA PFW       FOR NOW ALL INTERRUPTS ARE ILLEGAL
20431 000236        STA MPT
20432 021703        LDA UIT1      DEF TO UIT ROUTINE
20433 000241        STA UIT
20434 021716        LDA B11
20435 021677        LDB ILJMP     JMP ILI,I IN ALL TRAP CELLS
20436 000000  ILLP  STB A,I
20437 002004        INA
20440 021727        CPA B100      STOP AT LOCATION 77B
20441 002001        RSS
20442 020436        JMP ILLP
20443 021676        LDA PFWJMP    SET UP OTHER TRAP CELLS
20444 000004        STA 4
20445 021673        LDA PEJMP
20446 000005        STA 5
```

```
00593  20447 000231         STA PEJMPI    SET UP FOR JSB IN PE TRAP CELL
00594  20450 021700         LDA MPTJMP
00595  20451 000007         STA 7
00596  20452 021672         LDA TBGJMP
00597  20453 000006         STA 6
00598  20454 021702         LDA UITJSB
00599  20455 000010         STA 10B
00600  20456 021701         LDA UITJMP
00601  20457 000240         STA UIJMPI
00602*
00603*     BASIC I/O ON CPU BOARD
00604*
00605  20460 021733         LDA B170360   INDICATE (IF POSSIBLE) IN IO TEST
00606  20461 102601         OTA CPUST
00607*
00608*
00609  20462 102300         SFS 0         CHECK INTERRUPT FF
00610  20463 102200         SFC 0
00611  20464 021503         JMP PROER     INTERRUPT FF ERROR
00612*
00613  20465 102202         SFC 2         CHECK GLOBAL REG.
00614  20466 102302         SFS 2         SHOULD BE OFF (FLAG SET)
00615  20467 021503         JMP PROER     GLOBAL REG. ERROR
00616*
00617  20470 107706         CLC 6,C       INSURE TBG IS OFF
00618  20471 102100         STF 0         TURN ON INTERRUPTS
00619  20472 102200         SFC 0         CHECK IT
00620  20473 102300         SFS 0
00621  20474 021503         JMP PROER     INTERRUPTS NOT ON
00622  20475 002400         CLA
00623  20476 102600         OTA 0         CLEAR INTERRUPT MASK
00624  20477 102604         OTA 4         CLEAR INTERRUPT REGISTER
00625  20500 020507         LDA TBGDEF1
00626  20501 000233         STA TBG       SET JUMP IN TRAP CELL
00627  20502 103706         STC 6,C       TRY TIME BASE TIC
00628  20503 002400         CLA           START COUNT AT ZERO
00629  20504 002306         CCE,INA,SZA       NOW WAIT FOR TIC
00630  20505 020504         JMP *-1
00631  20506 021503         JMP PROER     LONG ENOUGH NOW, ERROR

00633*
00634  20507 020510  TBGDEF1 DEF ITBG     DEF TO TBG INTERRUPT
00635  20510 103100  ITBG   CLF 0         TURN OF INTERRUPTS
00636  20511 003004         CMA,INA       NEGATE COUNT FOR FUTURE USE
00637  20512 001121         ARS,ARS
00638  20513 001100         ARS           DIVIDE BY 8
00639  20514 000245         STA TBGCNT    SAVE COUNT FOR 1.25 MS
```

```
20515 107706        CLC 6,C       TURN OFF TIC
20516 021505        LDA ILDEF
20517 000233        STA TBG       TBG IS ILLEGAL INT NOW
20520 102504        LIA 4         CHECK CENTRAL INTERRUPT
20521 021714        CPA B6        WAS IT THE TBG?
20522 102206        SFC 6         FLAG SHOULD STAY CLEAR
20523 021503        JMP PROER     NOT SO ERROR (OR CIR NOT = 6)
*
*                  DONT TEST TBG MASK BIT ON PROC SINCE NOT
*                  IMPLEMENTED ON A700
*    LIA CPUST
*    SSA           DID IT STAY CLEAR?
*    JMP PROER     NO PROCESSOR ERROR
*    LDA B2        NOW SET MASK BIT
*    OTA 0
*    LIA CPUST     GET MASK BIT
*    SSA,RSS       DID IT SET
*    JMP PROER     NO THEN ERROR
*    CLA           NOW RESTORE MASK BIT
*    OTA 0         IT WAS ORIGINALLY CLEAR
*
*
*                  SEE IF WE HAVE A VCP
*
20524 021720        LDB B20       FIRST SELECT CODE TO TRY
20525 107602  VCPL  OTB GR,C      SET SELECT CODE
20526 021710        LDA B2        DIAGNOSE MODE 2
20527 102602        OTA GR        SET CARD
20530 102502        LIA GR        GET RESULT
20531 001710        ALF,SLA       BREAK BIT SET?
20532 020541        JMP VCPL1     YES, FOUND
*
20533 006004        INB           NEXT SELECT CODE
20534 021727        CPB B100      LAST SELECT CODE DONE??
20535 020552        JMP ION6      YES, NO VCP FOUND
*
20536 002400        CLA           TURN OFF DIAGNOSE MODE
20537 102602        OTA GR
20540 020525        JMP VCPL      GO AROUND AGAIN FOR NEW SELECT CODE
*
20541 003400  VCPL1 CCA
20542 000513        STA VCP.FLAG  GOOD VCP PRESENT !!!
20543 000514        STB VCPSC     VCP SELECT CODE
20544 002404        CLA,INA       DIAGNOSE MODE 1
20545 102602        OTA GR
20546 102502        LIA GR        GET RESPONSE
20547 021726        AND IDM       GET ID ONLY
20550 002003        SZA,RSS       IS TICK CARD??? (ZERO ID)
20551 000515        STB ASFLG     ASIC I/F, NOT INTELLIGENT CARD
```

```
*
*
20552 002400  ION6  CLA           CLEAR DIAGNOSE MODE
20553 102602        OTA GR

*
*     START MEMORY ACCESS FOR FIRST TIME
*     CLEAR MEMORY IF IT WAS LOST DURING POWER DOWN
*     AND CHECK MEMORY, BUT DON'T DESTROY ANY DATA IF NOT LOST
*
20554 000215  MTST  LDA MLOST     GET MLOST BIT
20555 001200        RAL           PUT MLOST BIT IN SIGN
20556 000304        IOR VCPTFLG   SAVE MEMORY IF TESTING
20557 000215        STA MLOST     YES   INDICATE MEMORY NOT LOST
20560 021732        LDA B100340   INDICATE IN MEMORY TEST
20561 102601        OTA CPUST
20562 000302        STA PEADD     NEGATIVE NUMBER FOR NO PARITY ERROR
*
20563 021047        LDB PFWDEF1   GET DEF TO PFW HANDLER FOR MEM TEST
20564 000235        STB PFW       PUT IN CELL, ALL OTHER INTS ARE
                                    ILLEGAL FOR NOW
20565               LWD1          POINT AT MAP ZERO
20566 021706        DEF .DO
20567 102704        STC 4         TURN ON POWER FAIL INTERRUPTS
20570               SMAP          GET MAP ZERO DATA
20571 021706        DEF .DO
20572 000370  ..MBUF DEF MPBUF    TEST MAP ZERO FOR CORRECT INITIAL-
                                    IZATION
20573 002400        CLA
20574 020572        LDB ..MBUF    GET POINTER TO MAP BUFFER
*
20575 000001  MPLP  CPA B,I       IS MAP RIGHT?
20576 002001        RSS
20577 020774        JMP MTSTE     NO, GO REPORT ERROR
20600 002004        INA
20601 006004        INB           NEXT ADDRESS AND VALUE
20602 021723        CPA B40       DONE??
20603 002001        RSS           YES  GO SET UP FOR FIRST 32K CHECK
20604 020575        JMP MPLP      NO, GO AROUND LOOP AGAIN
20605 003400        CCA
20606 000366        STA MAP
```

```
00728*
00729*                    SET UP MAP FOR NEXT 32K AND CHECK PARITY SYSTEM
00730*
00731  20607 000366  MTSTM LDA MAP        CHECK IF END OF MEMORY
00732  20610 002004        INA            MOVE TO NEXT BLOCK
00733  20611 000366        STA MAP
00734  20612 021735        CPA B1000      IS IT END OF ADDRESSABLE MEMORY ?
00735  20613 021072        JMP MTST5      YES
00736  20614 105762        JLY STMAP      SET UP NEXT MAP
       20615 027647
00737*                    CHECK FIRST WORD TO SEE IF MEMORY THERE
00738  20616 106705        CLC 5          DISABLE PARITY INTERRUPTS
00739  20617 006400        CLB
00740  20620 000215        LDA MLOST      POINT AT FIRST WORD
00741  20621 002021        SSA,RSS        WAS MEMORY LOST?
00742  20622 020626        JMP MTM1       YES,SKIP LOAD
00743*
00744  20623 002400        CLA
00745  20624               XLB1 '@A'      READ A WORD
00746  20626 007000  MTM1  CMB
00747  20627 002400        CLA
00748  20630               XSB1 '@A'      COMPLEMENT AND STORE
00749  20632               XSB1 '@A'      STORE AGAIN FOR RAM POWER UP PROBLE
00750  20634               XCB1 '@A'      GET DATA BACK
00751  20636 007001        CMB,RSS        COMPLEMENT DATA BACK
00752  20637 021072        JMP MTST5      DIDNT STORE, END OF MEMORY
00753*
00754  20640 002400        CLA            ADDRESS ZERO AGAIN
00755  20641               XSB1 '@A'      STORE ORIGINAL DATA BACK
00756  20643               XCB1 '@A'      DID IT STORE?
00757  20645 002001        RSS            YES, GO TEST NEXT 32K
00758  20646 021072        JMP MTST5      FOUND END OF MEMORY
00759*
00760*
00761  20647 021071  MTST3 LDA PEDEF2     POINT PE TRAP AT OTHER ENTRY
00762  20650 000232        STA PE
00763  20651 000366        LDB MAP
00764  20652 005100        BRS            MAP OVER 2 FOR WHICH 64K BLOCK
00765  20653 021736        ADB B1400      ADD START OF LOGGING RAM
00766  20654 000503        STB TEMPO      SAVE ADDRESS
00767  20655 000215        LDA MLOST      WAS MEMORY LOST??
00768  20656 002020        SSA
00769  20657               XLB1 '1700B'   GET CURRENT DATA
00770  20661 002400        CLA
00771  20662               XSA1 '1700B'   GET DATA
00772  20664 102105        STF 5          CHANGE PARITY SENSE
00773  20665 021707        XOR B1         MAKE IT A ONE BIT ERROR
00774  20666               XSA1 '1700B'   ESTABLISH BAD PARITY
```

```
20670 103105        CLF 5          REVERSE SENSE
20671 102705        STC 5          ENABLE PARITY
20672               XLA1 '1700B'   READ BAD PARITY
20674 002002        SZA            CORRECTED??
20675 020774        JMP MTSTE      NO, ERROR
20676 000503        LDA TEMPO,I    GET ERROR LOG
20677 002003        SZA,RSS        ERROR LOGGED AND CORRECTED??
20700 020774        JMP MTSTE      IT DIDN'T SO ERROR
20701 000247        ISZ ECCCNT
20702 002400        CLA
20703 000503        STA TEMPO,I    CLEAR ERROR LOGGING RAM
20704               XSA1 '1700B'   RESTORE GOOD PARITY
20706 102105        STF 5
20707 021711        XOR B3         MAKE TWO BIT ERROR
20710               XSA1 '1700B'   STORE BAD PARITY
20712 103105        CLF 5
20713               XLA1 '1700B'   READ BAD PARITY
20715 020774        JMP MTSTE      NO PARITY ERROR, BAD PARITY SYSTEM
*

20716         MTST4 LWD1           PUT DATA 1 MAP BACK LIKE IT WAS
20717 021706        DEF .DO
20720               XSB1 '1700B' RESTORE GOOD PARITY TO LOCATION 1700
20722 021004        LDA PEDEF1
20723 000232        STA PE         PARITY ERRORS TO OTHER HANDLER NOW
20724 102705        STC 5          TURN PARITY INTS BACK ON AGAIN
*
20725 000215  MTSTO LDA MLOST      TEST A 32K BLOCK OF MEMORY
20726 002021        SSA,RSS        IF MEMORY WAS LOST SKIP LOADING DATA

20727 020740        JMP MTSTL      MEMORY CONTENTS LOST
*
20730 002400        CLA            CLEAR A AND B TO COPY DATA TO SELF
20731 006400        CLB
20732 105745        LDX B77777     COUNT FOR 32K
20733 021744
20734               MW11           READ EVERY LOCATION TO CHECK PARITY
20735               XLA1 '@A'      READ LAST LOCATION
20737 020607        JMP MTSTM
*
20740 007400  MTSTL CCB            MAKE ALL ONES
20741 002400        CLA
20742               XSB1 '@A'      STORE IT IN FIRST LOCATION
20744 007004        CMB,INB        MAKE B ONE
20745 105745        LDX B77777     COUNT FOR 32K
20746 021744
20747               MW11           WRITE ONES IN EVERY LOCATION AND
                                      READ BACK
```

```
20750                    XLB1 '@A'    READ THE LAST LOCATION
20752 006006             INB,SZB      IS IT ONES???
20753 020774             JMP MTSTE    NO, MEMORY ERROR
20754 002400             CLA
20755 006400             CLB
20756                    XSB1 '@A'    STORE ZERO IN FIRST LOCATION
20760 006004             INB
20761 105745             LDX B77777   COUNT FOR 32K
20762 021744
20763                    MW11         WRITE ZERO IN ALL LOCATIONS AND READ
20764                    XLB1 '@A'    READ LAST LOCATION
20766 006002             SZB          IS IT ZEROS
20767 020774             JMP MTSTE    NO, MEMORY ERROR
20770 000503             LDA TEMPO,I  GET ERROR LOG
20771 002002             SZA          ZERO STILL??
20772 000250             ISZ CORCNT   ONE MORE CORRECTION
20773 020607             JMP MTSTM


*       MEMORY ERROR ROUTINE
*       EXTENDED MEMORY ERROR DISPLAY
*
20774 002400      MTSTE  CLA
20775 105762             JLY STMAP    PUT MAP ZERO BACK
20776 027647
20777 000366             LDA MAP      GET 32K BLOCK ADDRESS
21000 000246             STA MSIZE    SAVE MEMORY SIZE
21001 001727             ALF,ALF      PUT IT IN UPPER HALF
21002 021732             IOR B100340  ADD EXTENDED MEMORY SECTION
21003 021515             JMP DSPLY    GO DISPLAY IT
*
*       PARITY INTERRUPT ROUTINE
*       A SOFT ERROR WILL NOT CAUSE CPU TO STOP
*
21004 021005      PEDEF1 DEF IPRTY    DEF TO PARITY HANDLER
21005             IPRTY  LWD1         RESTORE DATA 1 MAP (KILLED BY
                                         INTERRUPT)
21006 021706             DEF .DO
21007 000215             LDA MLOST    MEMORY LOST??
21010 002021             SSA,RSS      NO, CHECK FOR SOFT ERROR
21011 020774             JMP MTSTE    YES, NO SOFT ERRORS IF MEMORY LOST
*
21012 002400             CLA
21013 105762             JLY STMAP    SET UP MAP TO FIRST 32K
21014 027647
21015 006400             CLB
21016 000215             STB MLOST    MEMORY LOST NOW
21017                    XSB1 '4'     CLEAR RESTART CONDITION
```

```
21021 000366         LDA MAP
21022 000303         STA PEMAP    SAVE BLOCK WITH PARITY ERROR
21023 105762         JLY STMAP    SET MAP BACK LIKE BEFORE
21024 027647
21025 102505         LIA 5
21026 021744         AND B77777
21027 000302         STA PEADD    SAVE ADDRESS OF THIS PARITY ERROR
21030 007400         CCB
21031                XSB1 '@A'    RESTORE GOOD PARITY TO LOCATION
21033                XCB1 '@A'    READ IT BACK
21035 006005         INB,RSS
21036 020774         JMP MTSTE    NO, A REAL MEMORY PROBLEM
*
21037                XSB1 '@A'    STORE ZEROS
21041                XLB1 '@A'
21043 006002         SZB          WAS A SOFT ERROR
21044 020774         JMP MTSTE    NO, REAL MEMORY PROBLEM
*
21045 102705         STC 5        TURN ON PARITY INTS AGAIN
21046 020725         JMP MTSTO    GO TEST THIS 32 K AGAIN

*                 POWER GOING DOWN
21047 021050 PFWDEF1 DEF PDOWN   POWER DOWN DEF
21050 106704 PDOWN CLC 4        TURN OF POWERFAIL INTERRUPTS
21051                LWD1         RESTORE DATA 1 MAP
21052 021706         DEF .DO
21053 000302         LDA PEADD    YES CHECK IF THERE
21054 002020         SSA            WAS A PARITY ERROR
21055 021065         JMP IPF      NO
*
21056                XLA1 '@A'
21060 021711         XOR B3       MAKE TWO BIT ERROR
21061 102105         STF 5        YES - CHANGE PARITY SENSE
21062                XSB1 '@A'    WRITE AN ERROR
21064 103105         CLF 5        PUT PARITY BACK
21065 102304 IPF    SFS 4        WAIT FOR POWER TO GO DOWN
21066 021065         JMP IPF
21067 107700         CLC 0,C      TURN OFF MACHINE
21070 020002         JMP START    DIDN'T GO ALL THE WAY SO RESTART
*
21071 020716 PEDEF2 DEF MTST4    PARITY TEST ENTRY
*
21072 000366 MTST5 LDA MAP      GET LAST BLOCK NO.
21073 000246         STA MSIZE    SAVE MEMORY SIZE
21074 002003         SZA,RSS
21075 020774         JMP MTSTE    GO SAY NO MEMORY ERROR
21076 002400         CLA
21077 000366         STA MAP      RESET MAP ZERO
```

```
             21100 105762         JLY STMAP
             21101 027647
       *
             21102 103100         CLF 0         RESET THINGS
             21103 102704         STC 4         REENABLE ALSO
             21104 021720         LDA B20
             21105 021735         LDB B1000
             21106 105745         LDX B100      SAVE TRAP CELL AREA OF MAIN MEMORY
             21107 021727
             21110                MW10
             21111 003400         CCA
             21112 000305         STA TRAPFLAG  FLAG THAT TRAP CELLS ARE SWAPPED
             21113 000510         STA NDCLR     NEED TO PRESET IF BREAK DURING
                                                   I/O TEST
             21114 021671         LDA IOLP      SET POINTER FOR I/O TABLE
             21115 000503         STA TEMPO
             21116 002004         INA
             21117 000502         STA TEMP1     SAVE PAGE ADDRESS

       *     START OF I-O INTERFACE CHIP TESTS
       *
       *     USE DIAG. MODE 1 TO BUILD A SELECT CODE TABLE
       *
             21120 021747         LDA B177700 INDICATE IN I/O INTERFACES
             21121 102601         OTA CPUST
             21122 102102         STF 2       INSURE GLOBAL REGISTER IS OFF
             21123 002404         CLA,INA     SET TEST MODE 1 (PRIORITY RESPONSE)
             21124 102602         OTA 2       GIVE MODE TO  CHIPS
No I/O # 1   21125 002400  IOL0   CLA         IN CASE OF NO RESPONSE
             21126 000503         ISZ TEMPO
             21127 000503         STA TEMPO,I GET TABLE POINTER
             21130 102502         LIA 2       GET SELECT CODE
             21131 002003         SZA,RSS     ANY SELECT CODE
             21132 021157  Good ← JMP IONO    NO END-OF-IO CHIPS
             21133 021725         AND SCM     YES - USE SELECT CODE ONLY  Scm 100077
             21134 000503         STA TEMPO,I PUT IT IN TABLE
             21135 001665         ELA,CLE,ERA  MAKE BIT 15 = 0
             21136 021753         ADA .N20    SUBTRACT 20B
             21137 002020  Error  SSA         IS IT A VALID SELECT CODE?
             21140 021155       ← JMP IOE4    NO - INDICATE ERROR 4 ON LEDS
             21141 000502         LDA TEMP1   CHECK FOR DUPLICATE SELECT CODES
             21142 000503  IOL1   CPA TEMPO   END OF TABLE?
             21143 021125 ------- JMP IOL0    YES MOVE TO NEXT IO CHIP
             21144 000000         LDB A,I     GET SC FROM TABLE
             21145 005665         ELB,CLE,ERB  Make bit 15 = 0
             21146 000503         CPB TEMPO,I IS IT THE SAME AS THE NEW SC?
             21147 021152         JMP *+3     YES - DUPLICATE SELECT CODES ERROR 3
             21150 002004         INA
             21151 021142         JMP IOL1    NO DO NEXT ENTRY
```

```
*
21152 000205        STB SAVEB    DUPLICATE SELECT CODE IN B REGISTER
21153 021711        LDA B3
21154 021513        JMP IOER
21155 021712  IOE4  LDA B4
21156 021513        JMP IOER

*     CHECK IF ANY SELECT CODES DID NOT RESPOND TO MODE 1
*     IF THEY DIDN'T PRIORITY CHAIN IS BROKEN
*
21157 102102  IONO  STF 2          INSURE GLOBAL REGISTER IS OFF
21160 002400        CLA            TURN OFF DIAGNOSE MODE
21161 102602        OTA 2
21162 021717        LDA B17        START WITH FIRST SELECT CODE -1
21163 000477        STA PO.CT
21164 000477 ->IOL2 ISZ PO.CT      MOVE TO NEXT SC
21165 000502        LDA TEMP1      CHECK IF IN TABLE
21166 000000  IOL3  LDB A,I        GET SC FROM TABLE
21167 006003        SZB,RSS        END OF TABLE?
21170 021176   OUT<-JMP ION1       YES
21171 005665        ELB,CLE,ERB
21172 000477        CPB PO.CT      NO IS SC IN TABLE?
21173 021164        JMP IOL2       YES
21174 002004        INA
21175 021166        JMP IOL3       NO MOVE TO NEXT ENTRY
21176 000477  ION1  LDA PO.CT      GET SC
21177 021727        CPA B100       END OF SC'S
21200 021210        JMP ION2       YES
21201 102602        OTA 2          NO TRY IT
21202 002400        CLA
21203 102502        LIA 2
21204 002003        SZA,RSS        DID IT COME BACK?
21205 021164        JMP IOL2       NO MOVE TO NEXT ONE
21206 021710        LDA B2         YES - INDICATE ERROR 2
21207 021513        JMP IOER

*     CHECK INDIVIDUAL I/O CHIPS
*
21210 000502  ION2  LDB TEMP1,I    START IO CHECK WITH FIRST ENTRY
21211 002400        CLA
21212 006003        SZB,RSS        WERE THERE ANY ENTRIES?
21213 021513        JMP IOER       NO IO CHIPS PRESENT ERROR 0
21214 021275        LDA IOIDEF     SET UP DEF FOR TRAP CELL JUMP
21215 000242        STA INTIO
21216 000502        LDB TEMP1      GET SC TABLE POINTER
21217 000503        STB TEMPO      SET POINTER
21220 000503  IOL4  LDB TEMPO,I    GET SELECT CODE
21221 006103        CLE,SZB,RSS    END OF TABLE?
21222 021332        JMP ION3       YES CHECK FOR BREAK ENABLE
```

```
01006*
01007  21223 105762        JLY CHKIO     CHECK I/O CHIP ON THIS CARD
       21224 021546
01008  21225 021512        JMP IOESC      * DISPLAY SELECT CODE WITH ERROR
01009*
01010*     CHECK DMA AND INTERRUPTS
01011*
01012  21226 000503        LDA TEMPO,I   GET SELECT CODE
01013  21227 021724        AND B77       MASK TO SELECT CODE
01014  21230 103602        OTA GR,C      SET GLOBAL REGISTER
01015  21231 021675        LDB IOIJMP
01016  21232 000000        STB A,I       PUT I/O INTERRUPT JUMP IN TRAP CELL
01017  21233 021720        LDA B20
01018  21234 021720        LDB B20
01019  21235 105745        LDX B100      UPDATE TRAP CELL AREA FOR THIS
       21236 021727                         INTERRUPT
01020  21237               MW01
01021  21240 021330        LDA DMACF     INCLUDE DMA ADDRESS
01022  21241 102620        OTA 20B       PASS IT TO SELF CONFIGURATION REG
01023  21242               XSA1 'DMA+1'     AND PLACE IN TRIPLET
01024  21244 021327        LDA DMAQD     GET DMA CONTROL WORD
01025  21245               XSA1 'DMA'
01026  21247 021331        LDA DMAQD+2      AND COUNT
01027  21250               XSA1 'DMA+2'
01028  21252 021715        LDA B7        DISABLE SRQ INTERRUPTS
01029  21253 102602        OTA 2         DIAGNOSE MODE 7!!! MUST CLC 0,C TO
                                            GET OUT OF THIS MODE
01030  21254 103720        STC 20B,C     DO SELFCONFIGURATION
01031  21255 102324        SFS 24B       DID IT COMPLETE
01032  21256 021512        JMP IOESC     NO SO ERROR
01033  21257 102521        LIA 21B       CHECK CONTROL WORD
01034  21260 021327        CPA DMAQD
01035  21261 002001        RSS
01036  21262 021512        JMP IOESC     BAD SO ERROR
01037  21263 102523        LIA 23B       CHECK COUNT
01038  21264 021331        CPA DMAQD+2
01039  21265 002001        RSS
01040  21266 021512        JMP IOESC     NO GOOD SO ERROR
01041  21267 021711        LDA B3        NOW USE DIAG. MODE 3
01042  21270 102602        OTA 2
01043  21271 102100        STF 0         TURN ON INTERRUPTS
01044  21272 002006        INA,SZA       WAIT FOR IT
01045  21273 021272        JMP *-1
01046  21274 021512        JMP IOESC     NO GOOD
01047*
```

```
21275 021276  IOIDEF DEF IOINT    DEF FOR TRAP CEEL
21276         IOINT LWD1          PUT DATA 1 MAP BACK
21277 021706        DEF .DO
21300 102504        LIA 4         CHECK CENTRAL INTERRUPT
21301 106502        LIB 2         AGAINST GLOBAL REGISTER
21302 000001        CPA B         WELL?
21303 002001        RSS
21304 021512        JMP IOESC     CARD ERROR
21305 021330        LDB DMACF
21306 021713        ADB B5        MOVE TO CONFIGURATION ADDRESS
21307               XCB1 'DMA+2'  DID IT STORE
21311 102224        SFC 24B       AND DID IT TURN OFF
21312 021512        JMP IOESC     NO SO ERROR
21313 102523        LIA 23B       CHECK COUNT IS ZERO
21314 002002        SZA
21315 021512        JMP IOESC
21316 107720        CLC 20B,C     INSURE DMA IS OFF
21317 107721        CLC 21B,C
21320 000503        LDA TEMPO,I   GET SELECT CODE
21321 021724        AND B77
21322 021674        LDB ILIJMP    PUT TRAP CELL BACK TO ILLEGAL
                                     INTERRUPT
21323 000000        STB A,I
21324 000503        ISZ TEMPO     MOVE TO NEXT ENTRY
21325 000204        ISZ SAVEA     COUNT THIS I/O CARD
21326 021220        JMP IOL4      AND DO IT
*
      001760  DMA   EQU 1760B
21327 000200  DMAQD OCT 200
21330 001760  DMACF DEF DMA
21331 177775        DEC -3


*     CHECK THAT ONLY ONE INTF. HAS A BREAK ENABLE
*     NONE IS OK
*
21332 021710  ION3  LDA B2        USE DIAGNOSE MODE 2
21333 000502        LDB TEMP1     SET POINTER FOR SELECT CODE
21334 000503        STB TEMPO
21335 006400        CLB
21336 000501        STB TEMP2     CLEAR SC FLAG
21337 102102        STF 2         TURN OFF GLOBAL REGISTER
21340 102602        OTA 2
21341 002400  IOL5  CLA           CLEAR IN CASE OF NO RESPONSE
21342 102502        LIA 2         GET PARAMETERS
21343 002002        SZA           DONE WITH I O
21344 021350        JMP ION4      NO
21345 102602        OTA 2         TURN OFF DIAG.MODE 2
21346 006400        CLB           NO ERRORS
```

```
21347 021364        JMP PTSTX     YES NOW CHECK IF VCP OR LOADER
21350 001710  ION4  ALF,SLA       CHECK BREAK ENABLE BIT
21351 021354        JMP *+3
21352 000503        ISZ TEMPO     MOVE TO NEXT ONE
21353 021341        JMP IOL5
21354 002740        CLA,SEZ,CCE     WAS THERE A PREVIOUS ONE
21355 021361        JMP IOEN4     YES SO ERROR 1
21356 000503        LDB TEMPO     NO OK SAVE THIS ONE
21357 000501        STB TEMP2
21360 021341        JMP IOL5      NOW TRY NEXT ONE
*
21361 000513  IOEN4 STA VCP.FLAG  NO VCP IF TWO BREAK ENABLES
21362 002004        INA
21363 021513        JMP IOER      DISPLAY ERROR 1

*
*     PRETEST EXIT TO VFP
*     PRETEST IS FINISHED
*
21364 105762  PTSTX JLY .PSET     CLEAR I/O SYSTEM FROM DIAGNOSE
21365 021633                         MODE 7
21366 021776        LDA ..ENT
21367 102603        OTA 3         INITIALIZE BREAK ENTRY POINT
21370 103603        OTA 3,C
21371 000305        LDA TRAPFLAG  CHECK VCP TEST
21372 002003        SZA,RSS
21373 021401        JMP PTS2
21374 021735        LDA B1000
21375 021720        LDB B20
21376 105745        LDX B100
21377 021727
21400               MW01         RESTORE TRAP CELL AREA OF MAIN MEMORY
21401 000304  PTS2  LDA VCPTFLG   CHECK VCP TEST
21402 002020        SSA
21403 021471        JMP PTS1      IF TEST DONT CHECK SWITCHES
*
21404 102501        LIA CPUST     GET SWITCHES
21405 001727        ALF,ALF
21406 021715        AND B7       SELF TEST LOOP??
21407 002003        SZA,RSS
21410 020002        JMP START     YES, GO AROUND AGAIN
*
21411 000244        LDB DISPLAY   GET SELF TEST STATUS
21412 006002        SZB           DID SELF TEST PASS
21413 021471        JMP PTS1      NO, MUST GO TO VCP
*
21414 021712        CPA B4        LOOP ON SELF TEST
21415 020002        JMP START     YES, GO AROUND AGAIN
```

```
*
21416 021710        CPA B2        JMP TO USER ROM??
21417 021437        JMP ..USER,I  YES, GO
*
*    IF USER WANTS TO CONTINUE TO VCP MUST ENTER AT VCP IN PAGE 1
*    NOTE THAT USER ROM WILL NOT BE ENTERED IF SELF TEST FAILS
*
*    NOW SEE IF CAN AUTO RESTART
*
21420 002011        SLA,RSS
21421 021440        JMP PTS0      AUTO RESTART NOT ENABLED
*
21422 000215        LDB MLOST     CHECK MEMLOST
21423 006021        SSB,RSS       SKIP IF MEMORY SAVED
21424 021440        JMP PTS0      MEMORY LOST, GOTO VCP
*
21425               XLB1 '4'      GET TRAP CELL FOR AUTO RESTART
21427 006003        SZB,RSS       IS INSTRUCTION THERE?
21430 021440        JMP PTS0      NO INSTRUCTION, GOTO VCP
*
21431 102702        STC 2         ENABLE BREAK
21432 002400        CLA
21433 102601        OTA CPUST     INDICATE USER PROGRAM EXECUTING
21434               XJMP '.D0','4'  JUMP TO LOCATION 4 IN SYSTEM MAP
*
21437 030002  ..USER DEF 30002B   START OF USER ROM
*
*    CANT AUTO RESTART, SEE IF MUST AUTOLOAD
*
21440 021715  PTS0  CPA B7     DISC LOADER
21441 021455        JMP .PTDC,I   GO DO DISC LOADR
*
21442 021711        CPA B3
21443 021456        JMP PTDS      DS LOADER
*
21444 021713        CPA B5
21445 021447        JMP PTRM   PROM LOADER
*
21446 021471        JMP PTS1   NO, LOADR, GO TO VCP
*
21447 021705  PTRM  LDA .RMSC
21450 000264        STA SCETC      DEFAULT SELECT CODE
21451 104600        .JLB RMLDR      LOAD FROM ROM CARD
21452 026364
21453 021465        JMP .MRBT,I     GO START IT UP
21454 021466        JMP PTLER       GO REPORT ERROR
*
21455 026503  .PTDC DEF PTDC      DISC LOADER
```

```
*
21456 021704  PTDS  LDA .DSSC      DS AUTOBOOT
21457 000264        STA SCETC      SAVE DEFAULT SELECT CODE ETC
21460 104600        .JLB DSLD      LOAD FROM DS
21461 025040
21462 002001        RSS
21463 021466        JMP PTLER      ERROR, GOTO VCP
21464 021465        JMP .MRBT,I
*
21465 024072  .MRBT DEF MRBT       GO SET UP BOOT PARAMS
*
21466 021470  PTLER LDA DSCER      DISC ERROR CODE
21467 021515        JMP DSPLY
*
21470 100200  DSCER OCT 100200
*
*
*     GO TO VCP IF POSSIBLE
*
21471 000513  PTS1  LDA VCP.FLAG   IS THERE A VCP??
21472 002003        SZA,RSS
21473 021501        JMP NVCP       NO, NOTHING MORE TO DO
*
21474 102702        STC 2          ENABLE BREAK
21475 021715        LDA B7
21476 102601        OTA CPUST      SAY IN FRONT PANEL
21477 021500        JMP *+1,I
21500 022000        DEF VFP
*
*
21501 021713  NVCP  LDA B5         ERROR 5, NO VCP
21502 021513        JMP IOER



*     ERROR REPORTING TO PROCESSOR LEDS
*
21503 021733  PROER LDA B170360    INDICATE PROCESSOR ERROR
21504 021515        JMP DSPLY
*
*
21505 021506  ILDEF DEF ILINT      POINT TO ILLEGAL INT ROUTINE
21506 102504  ILINT LIA 4B         GET CENTRAL INTERRUPT REGISTER
21507 001727        ALF,ALF        PUT IT IN DATA
21510 021731        IOR B100300    INDICATE ILLEGAL INTERRUPT
21511 021515        JMP DSPLY
```

```
*
21512 000503  IOESC LDA TEMPO,I   GET SELECT CODE FOR DISPLAY
21513 001727  IOER  ALF,ALF       PUT DATA IN UPPER HALF
21514 021731        IOR B100300   INDICATE I/O TEST ERROR

*     DISPLAY LOWER BYTE (SECTION)
*         THEN UPPER BYTE (DATA   )
*         THEN BACK TO LOWER BYTE
*
21515 000244  DSPLY STA DISPLAY   SAVE DATA AND SECTION
*
21516 000244  POCOO LDA DISPLAY
21517 002300        CCE           SET TO DO SECOND PART
21520 021734        AND B377
21521 102601        OTA CPUST
21522 021753        LDB .N20
21523 000000        ISZ A
21524 021523        JMP *-1
21525 000001        ISZ B
21526 021523        JMP *-3
21527 002041        SEZ,RSS
21530 021535        JMP PRTLP
21531 000244        LDA DISPLAY
21532 001767        ALF,CLE,ALF
21533 021755        IOR BIT7
21534 021520        JMP POCOO+2   UPPER HALF  DATA
*
21535 102501  PRTLP LIA CPUST     CHECK IF LOOP
21536 001727        ALF,ALF
21537 021715        AND B7
21540 002003        SZA,RSS       ??
21541 021067        JMP IPF+2     YES LOOP ON ERROR
*
21542 000513        LDA VCP.FLAG  IS THERE A VCP??
21543 002002        SZA
21544 021364        JMP PTSTX     GO TO VCP
21545 021516        JMP POCOO
```

```
*
*
*
*     CHECK AN I/O CARD. B HAS SELECT CODE TO CHECK
*
*
21546 005623  CHKIO ELB,RBR       SAVE SELF TEST FLAG
21547 107602        OTB GR,C      SET GLOBAL REGISTER
21550 002400        CLA           CLEAR IN CASE OF NO RESPONSE
21551 102502        LIA GR        GET GLOBAL REGISTER
21552 000001        CPA B         DID IT COME BACK ?
21553 002001        RSS           YES
21554 105772  CHBR  JPY 0         NO ERROR
21555 000000
21556 002041        SEZ,RSS       DOES THIS INTERFACE HAVE SELF TEST
21557 021574        JMP ION.2     NO - THEN DON'T WAIT
21560 021754        LDA .N40      YES THEN WAIT 10 SECS FOR SELF TEST
21561 102230        SFC DATA                    !!!!!!!!!!
21562 021570        JMP ION.1
21563 000001        ISZ B
21564 021561        JMP *-3
21565 000000        ISZ A
21566 021561        JMP *-5
21567 021554        JMP CHBR      TIME OUT SO ERROR
21570 103530  ION.1 LIA DATA,C  GET SELF TEST STATUS & CLEAR THE FLAG
21571 002020        SSA           WAS IT GOOD?
21572 000010        SLA
21573 021554        JMP CHBR      NO SO ERROR

21574 021760  ION.2 LDA ALTO      USE ALTERNATING PATTERN
21575 102623        OTA 23B         TO CHECK I/O CHIP BUS UPPER
21576 001300        RAR           AND OPPOSITE PATTERN
21577 102624        OTA 24B         FOR I/O CHIP BUS LOWER
21600 007400        CCB           CLEAR IN CASE NO RESPONSE
21601 102523        LIA 23B       READ PATTERNS BACK
21602 106524        LIB 24B
21603 005200        RBL
21604 000001        CPA B         DO PATTERNS AGREE
21605 006401        CLB,RSS       YES
21606 021554        JMP CHBR      NO - I/O CHIP BUS ERROR
21607 102624        OTA 24B       REVERSE PATTERN AND
21610 001300        RAR           CHECK BUS AGAIN
21611 102623        OTA 23B
21612 102524        LIA 24B
21613 106523        LIB 23B
21614 005200        RBL
```

```
21615 000001        CPA B          DO PATTERNS AGREE?
21616 102230        SFC DATA       YES CHECK FLAG
21617 021554        JMP CHBR       BUS OR FLAG ERROR
21620 102130        STF DATA       SET THE I/O FLAG
21621 102230        SFC DATA       DID IT GET SET?
21622 102330        SFS DATA
21623 021554        JMP CHBR       NO I/O FLAG ERROR
21624 103130        CLF DATA       NOW CLEAR IT
21625 102330        SFS DATA       DID IT GET CLEARED
21626 102230        SFC DATA
21627 021554        JMP CHBR       NO I/O FLAG ERROR
*
21630 106723        CLC 23B        RESET DMA MACHINE
21631 105772        JPY 1          P+3 (GOOD) RETURN
21632 000001
*
*
21633 104600  .PSET .JLB ENDVCP   EXIT FROM VCP MODE
21634 023533
21635 107700        CLC 0,C        BLOW AWAY I/O SYSTEM
21636 021746        LDA B100000
21637 000211        STA SAVEQ      CS MODE IS OFF!
21640 002400        CLA
21641 000214        STA SAVEW      CLEAR WMAP
21642 000366        STA MAP
21643 000200        STA SAVEI      INTS OFF
21644 000206        STA SAVEG      GLOBAL REG OFF
21645               LWD1           POINT DATA 1 MAP
21646 000000        DEF 0
21647 021650        JMP *+1,I      SET UP MAP ZERO
21650 027647        DEF STMAP
*
* UIT HANDLER TO IGNORE UITS FOR R3 INSTRUCTIONS
*
21651 000227  UITINT STA PETMP
21652 000214        LDA SAVEW
21653 021722        AND .B37
21654               LWD1
21655 000000        DEF A
21656 000227        LDA PETMP
21657 000237        JMP UITRTN,I
*
```

```
*
*
*  PARITY ERROR HANDLER FOR USER INTERFACE.  IT SETS PEFLAG
*  SO THAT PARITY ERROR WILL BE OUTPUT BEFORE NEXT COMMAND ACCEPTED
*
 21660 000227  PEINT STA PETMP     SAVE A REGISTER
 21661 000214        LDA SAVEW     SET DATA ONE MAP BACK LIKE BEFORE
 21662 021722        AND .B37
 21663               LWD1
 21664 000000        DEF A
 21665 000227        LDA PETMP     RESTORE A REGISTER
 21666 102705        STC 5         TURN PARITY INTERRUPTS BACK ON
 21667 000243        ISZ PEFLAG
 21670 000230        JMP PERTN,I   GO GET ANOTHER COMMAND
*

*
*     CONSTANTS
*
*IOLP  DEF P0.CT-77B-P0-1
 21671 000370  IOLP  DEF MPBUF     PLACE FOR I/O SELECT CODE TABLE
 21672 000233  TBGJMP JMP TBG,I    TRAP CELL INSTRUCTION FOR TBG
 21673 000232  PEJMP JMP PE,I      "    "          "       "  PARITY
 21674 000234  ILIJMP JMP ILI,I
 21675 000242  IOIJMP JMP INTIO,I  I/O TRAP CELL CONTENTS
 21676 000235  PFWJMP JMP PFW,I     ETC.
 21677 000234  ILJMP  JMP ILI,I
 21700 000236  MPTJMP JMP MPT,I
 21701 000241  UITJMP JMP UIT,I
 21702 000237  UITJSB JSB UITRTN
 21703 021651  UIT1   DEF UITINT
 21704 000024  .DSSC OCT 0024
 21705 000022  .RMSC OCT 22
*
 21706 000000  .D0   OCT 0
 21707 000001  B1    OCT 1
 21710 000002  B2    OCT 2
 21711 000003  B3    OCT 3
 21712 000004  B4    OCT 4
 21713 000005  B5    OCT 5
 21714 000006  B6    OCT 6
 21715 000007  B7    OCT 7
 21716 000011  B11   OCT 11
 21717 000017  B17   OCT 17
 21720 000020  B20   OCT 20
 21721 000024  B24   OCT 24
 21722 000037  .B37  OCT 37
 21723 000040  B40   OCT 40
```

```
21724 000077 B77     OCT 77
21725 100077 SCM     OCT 100077
21726 077000 IDM     OCT 077000       ID ONLY NO SC OR REV.
21727 000100 B100    OCT 100
21730 000200 B200    OCT 200
21731 100300 B100300 OCT 100300
21732 100340 B100340 OCT 100340
21733 170360 B170360 OCT 170360
21734 000377 B377    OCT 377
21735 001000 B1000   OCT 1000
21736 001400 B1400   OCT 1400
21737 003004 B3004   OCT 3004
21740 006412 B6412   OCT 6412
21741 007777 B7777   OCT 7777
21742 076000 B76K    OCT 76000
21743 100000 B100K   OCT 100000
21744 077777 B77777  OCT 77777
21745 100024 B100024 OCT 100024      DS SELECT CODE & SELF TEST ENABLE
21746 100000 B100000 OCT 100000      CS OFF BIT
21747 177700 B177700 OCT 177700
21750 177777 B177777 OCT 177777
21751 177777 M1      OCT -1
21752 177776 .N2     OCT -2
21753 177760 .N20    OCT -20
21754 177740 .N40    OCT -40
21755 000200 BIT7    OCT 200
21756 102700 NOVCP   OCT 102700       NO VCP ERROR CODE
21757 021760 DALTO   DEF ALTO
21760 052525 ALTO    OCT 052525
21761 125252 ALT1    OCT 125252
21762 104447 SRGP1   OCT 104447       1000100100100111
21763 114040 SRGP2   OCT 114040       1001100000100000
21764 000100 SRGP3   OCT 000100       0000000001000000
21765 076310 AEAUS   OCT 076310
21766 021765 BLBT    DEF AEAUS        DEF FOR LBT TEST
21767 130272 BEAUS   OCT 130272
21770 107714 ASR.0   OCT 107714
21771 041300 ASR.1   OCT 041300
21772 143746 MU2     OCT 143746
21773 123746 DV4     OCT 123746
21774 126760 RESUA   OCT 126760
21775 006606 RESUB   OCT 006606
21776 022114 ..ENT   DEF ENTRY
*
      021777 EOPO    EQU *            END OF PAGE 0
*
```

```
*
*     ENTRY HERE ON POWERUP AFTER MICROCODED SELF TEST & PRETEST
*
*     USER ROM SHOULD ENTER HERE FOR VCP USER INTERFACE
*     DISPLAY HAS SELF TEST ERROR CODE
*
  22000                ORG EPROM+2000B
```

```
01459*
01460*     ENTRY HERE ON POWERUP AFTER MICROCODE SELF TEST & PRETEST
01461*
01462         022000  P1    EQU *
01463  22000 104600  VFP   .JLB CI.IZ     SET GLOBAL REGISTER
       22001 024411
01464  22002 104600        .JLB .ENQAK    DO ENQ ACK OR SEND BUFFER
       22003 024270
01465  22004 000244        LDA DISPLAY   GET SELF TEST ERROR CODE
01466  22005 002003        SZA,RSS
01467  22006 022037        JMP VFP.0     NO ERRORS IN SELF TEST
01468*
01469  22007 023766        LDA SELFERR   OUTPUT ERROR MESSAGE
01470  22010 104600        .JLB PUTS
       22011 024245
01471  22012 000244        LDA DISPLAY
01472  22013 023654        AND ..B377    LOW BYTE
01473  22014 104600        .JLB OUTN     OUTPUT ERROR CODE
       22015 024344
01474  22016 000244        LDA DISPLAY
01475  22017 001727        ALF,ALF
01476  22020 023654        AND ..B377
01477  22021 104600        .JLB OUTN        OUTPUT HIGH BYTE OF ERROR CODE
       22022 024344
01478*
01479*     CHECK FOR LOADER ERRORS
01480*
01481  22023 000265        LDA LERR
01482  22024 002003        SZA,RSS
01483  22025 022037        JMP VFP.0     NO LOADER ERRORS
01484*
01485  22026 023746        LDA CRLF      OUTPUT CRLF
01486  22027 104600        .JLB PUTS
       22030 024245
01487  22031 023614        LDA .LDER     OUTPUT LOADER ERROR MESSAGE
01488  22032 104600        .JLB PUTS
       22033 024245
01489  22034 000265        LDA LERR
01490  22035 104600        .JLB OUTD     OUTPUT ERROR NUMBER
       22036 024312
01491*
01492  22037 000302  VFP.0 LDA PEADD     ANY SOFT ERRORS???
01493  22040 002020        SSA
01494  22041 022057        JMP VFP.1     NO, GO TO FRONT PANEL
```

```
*
22042 023767        LDA SOFTERR   GET SOFT ERROR MESSAGE
22043 104600        .JLB PUTS     OUTPUT IT
22044 024245
22045 000303        LDA PEMAP
22046 000302        LDB PEADD
22047 005200        RBL           GET PAGE NUMBER OF PARITY ERROR
22050 100105        RRL 5
22051 104600        .JLB OUTN     OUTPUT BLOCK NUMBER
22052 024344
22053 000302        LDA PEADD
22054 023655        AND ..B1777   GET OFFSET IN PAGE
22055 104600        .JLB OUTN     OUTPUT ADDRESS IN BLOCK
22056 024344
*
22057 000304  VFP.1 LDA VCPTFLG   IS TEST??
22060 002020        SSA
22061 022176        JMP ENT2      GET NEXT COMMAND
*
22062 023747        LDA VERMG     OUTPUT VERSION
22063 104600        .JLB PUTS     MESSAGE
22064 024245
*
22065 000246        LDA MSIZE
22066 001722        ALF,RAL       MULTIPLY BY 32
22067 001200        RAL           MULTIPLY BY 2
22070 104600        .JLB OUTD     OUTPUT THE MEMORY SIZE
22071 024312
22072 023763        LDA KMES
22073 104600        .JLB PUTS     OUTPUT "K MEMORY IN SYSTEM"
22074 024245
22075 000247        LDA ECCCNT    GET AMOUNT OF ECC
22076 001722        ALF,RAL
22077 001200        RAL           MULTIPLY BY 64
22100 104600        .JLB OUTD
22101 024312
22102 023764        LDA ECMES
22103 104600        .JLB PUTS
22104 024245
22105 104600        .JLB .ENQAK   MAKE SURE TERMINAL READY, OUTPUT
22106 024270                         BUFFER
22107 022176        JMP ENT2      GET FIRST COMMAND
*
22110 021660  PE1   DEF PEINT
22111 000230  PEJSB JSB PERTN
*
```

```
*
*      BREAK COMES HERE !!!!!!
*      SO DO HALT INSTRUCTIONS
*
*
22112 024137  .RENT DEF REENT
22113 022114  .ENT  DEF *+1
22114 103105  ENTRY CLF 5          SET PARITY TO "ODD"
22115 103200        OCT 103200     SFC 0,C
22116 000200        ISZ SAVEI      SET INTERRUPTS ON FLAG
22117 000507        ISZ FIRST
22120 002001        RSS            CHECK NOT FIRST TIME FLAG
22121 022172        JMP AGAIN      BREAK WAS DURING VCP SO DONT CHANGE
                                      REGISTERS
22122 104400        DST SAVEA+2000B  SAVE "A" REGISTER AND B REGISTER
22123 002204
22124               CCQA           GET Q
22125 000211        STA SAVEQ
22126               CZA            GET Z
22127 000212        STA SAVEZ      SAVE IT
22130 002400        CLA
22131 102201        SOC            IS "O" CLEAR ?
22132 002004        INA            NO, INCREMENT "A"
22133 000201        STA SAVEO      SAVE "O" REPLICA
22134 001522        ERA,RAL        "E" INTO LSB OF "A"
22135 000202        STA SAVEE      SAVE IT
22136 102502        LIA GR         GET CURRENT VALUE
22137 102202        SFC GR         IS GLOBAL REGISTER ON ?
22140 023721        IOR BIT15      YES, SET MSB
22141 000206        STA SAVEG      SAVE FOR EXIT
22142 105743        STX SAVEX      SAVE X AND Y REGISTERS
22143 000207
22144 105753        STY SAVEY
22145 000210
22146 000100        LDA WMAP
22147 000214        STA SAVEW      SAVE WMAP VALUE
22150 023720        AND B37
22151               LWD1           SET DATA 1 MAP TO OLD XQT MAP
22152 000000        DEF 0
22153 102503        LIA 3          FETCH "P" VALUE
22154 001665        ELA,CLE,ERA    NO SIGN BIT ON P REGISTER
22155 000203        STA SAVEP      SAVE IT
22156 023724        ADA N1         IF NO, DECREMENT "P"
22157 000213        STA SAVEM      SAVE "M" VALUE
22160               XLB1 '@A'      GET INSTRUCTION WHICH GOT US HERE
22162 023644        CPB  .ENTI     IS BOOTEX CALL BACK?
22163 022112        JMP  .RENT,I   YES, GO DO REQUIRED OPERATION
```

```
*
22164 102501         LIA CPUST     IS BREAK DISABLED?
22165 001727         ALF,ALF
22166 023717         AND .B10      ISOLATE BREAK SWITCH
22167 002002         SZA           IF ITS A ONE, BREAK DISABLED
22170 023530         JMP EXEX2     RESTART IMMEDIATELY IF DISABLED
*
22171 022176         JMP ENT2
*
*
22172 000510   AGAIN ISZ NDCLR     DO WE HAVE TO PRESET??
22173 022176         JMP ENT2      NO WE DONT
22174 105762         JLY .PSET     PRESET
22175 021633
*
22176 003400   ENT2  CCA
22177 000507         STA FIRST
22200 000302         STA PEADD     NO PARITY ERROR
22201 022111         LDA PEJSB     PUT JSB IN TRAP CELL
22202 000005         STA 5
22203 022110         LDA PE1
22204 000232         STA PE        SET PARITY TRAP CELL FOR PE INTS
22205 023745         LDA .RTRN
22206 000526         STA XEQT+2    SET RETURN POINT FOR I/O INSTRUC-
                                     TION SUBROUTINE
*
22207 104600         .JLB CI.IZ    SET GLOBAL REGISTER
22210 024411
*
22211 102702         STC 2         ENABLE BREAK
*
*     OUTPUT THE REGISTERS  ( P, A, B, RW, M, & T )
*
22212 002400         CLA           INITIALIZE NUMBER
22213 000243         STA PEFLAG
22214 000510         STA NDCLR     DONT NEED TO CLEAR IF A BREAK HERE
22215 000363         STA DFLAG     OF DIGITS FLAG
22216 023771         LDA PMESS     OUTPUT A 'P' AND
22217 104600         .JLB PUTS
22220 024245
22221 000203         LDA SAVEP      THE CURRENT P VALUE
22222 104600         .JLB OUTN
22223 024344
```

```
*
22224 023772       LDA AMESS    OUTPUT AN 'A' AND
22225 104600       .JLB PUTS
22226 024245
22227 000204       LDA SAVEA     THE CURRENT A VALUE
22230 104600       .JLB OUTN
22231 024344
*
22232 023773       LDA BMESS    SAME LIKE BEFORE
22233 104600       .JLB PUTS
22234 024245
22235 000205       LDA SAVEB      ONLY THE NAMES HAVE CHANGED
22236 104600       .JLB OUTN
22237 024344
22240 023753       LDA SPC2
22241 104600       .JLB PUTS    OUTPUT TWO SPACES
22242 024245
22243 023702       LDA .R
22244 104600       .JLB PUTCH
22245 024560
22246 023707       LDA .W
22247 104600       .JLB PUTCH
22250 024560
22251 000214       LDA SAVEW    OUTPUT WMAP VALUE
22252 104600       .JLB OUTN
22253 024344
22254 022431       JMP .T02      EARLY EXIT FROM .TREG
*
22255 104600 NEXT  .JLB CI.IZ   ENABLE VCP
22256 024411
*
*
*    HERE IS MAIN COMMAND INTERPRETATION LOOP
*
*
22257 023740 COMND LDA D7          SAY IN FRONT PANEL TO LIGHTS
22260 102601       OTA CPUST
22261 000243       LDA PEFLAG   WAS PARITY ERROR IN LAST COMMAND??
22262 002003       SZA,RSS
22263 022271       JMP COMN1    NO, GO ON
*
22264 023765       LDA PEMES
22265 104600       .JLB PUTS    SAY PARITY ERROR
22266 024245
22267 002400       CLA
22270 000243       STA PEFLAG
22271 023750 COMN1 LDA PRMPT    OUTPUT THE PROMPT
22272 104600       .JLB PUTS       CHARACTER   ">"
22273 024245
```

```
*
22274 002400          CLA            CLEAR COMMAND
22275 000503          STA TEMPO       SAVE
*
22276 104600   COM1  .JLB TG.BF      INITIALIZE TO XMIT & GET BUFFER
22277 024533
22300 104600         .JLB GETCH      FETCH A CHARACTER
22301 024540
22302 023645          CPA .CR        "CR" ?
22303 022257          JMP COMND       JUST TESTING
22304 023653          CPA .?         AH, A PLEA FOR HELP
22305 022416          JMP .HELP       GO DUMP HELP MESSAGE
22306 023704          CPA .T         HOW ABOUT THE "T" REGISTER ?
22307 022422          JMP .TREG       GOOD GUESS
22310 023647          CPA .%         CONTROL SEQUENCE ?
22311 023371          JMP CNTRL       YEP, GO SEE WHICH ONE
22312 023675          CPA .M         IS IT "MEMORY ADDRESS" ?
22313 022512          JMP .MREG
22314 023706          CPA .V         IS IT VIOLATION REGISTER??
22315 023164          JMP .VIO
22316 023702          CPA .R         IS IT A SPECIAL REGISTER ?
22317 022667          JMP .REGS
22320 023674          CPA .L
22321 022551          JMP .LIST      LIST MEMORY
*
22322 023726          LDB N4         INITIALIZE DATA FLAG
22323 023712          CPA .Z
22324 006005          INB,RSS
22325 023701          CPA .Q
22326 006005          INB,RSS
22327 023711          CPA .Y         IS IT "Y REGISTER" ?
22330 006005          INB,RSS        YES, BUMP DATA FLAG
22331 023710          CPA .X
22332 006005          INB,RSS
22333 023672          CPA .G
22334 006005          INB,RSS
22335 023665          CPA .B
22336 006005          INB,RSS
22337 023663          CPA .A
22340 006005          INB,RSS
22341 023700          CPA .P
22342 006005          INB,RSS
22343 023670          CPA .E
22344 006005          INB,RSS
22345 023677          CPA .O
22346 006005          INB,RSS
22347 023673          CPA .I
22350 006005          INB,RSS
```

```
22351 022412        JMP CERR      TRY AGAIN
22352 000367        STA PAGE      SAVE CHAR
*
22353 000363        STB DFLAG     SET TYPE FLAG (< 0 => SINGLE DIGIT)
22354 023742        LDA BUFF      BUILD ADDRESS OF
22355 000363        ADA DFLAG      DESIRED REGISTER
22356 000361        STA DPNTR     SAVE IT FOR LATER
22357 000000        LDA A,I       FETCH CURRENT VALUE
22360 104600        .JLB OUTN        PRINT IT
22361 024344
22362 104600        .JLB TG.BF    OUTPUT BUFFER AND GET INPUT
22363 024533
22364 104600        .JLB GETN       NEW VALUE
22365 024647
22366 023211        JMP COM01     NO NEW DATA TRY AGAIN
22367 006002        SZB           TERMINATION ON "CR" ?
22370 022412        JMP CERR      NO, TELL 'EM ABOUT IT
22371 000363        LDB DFLAG     WAS THIS THE "P"
22372 006003        SZB,RSS        REGISTER  ( IF DFLAG = 0 )
22373 001665        ELA,CLE,ERA     IF YES THEN FORCE MSB TO 0
22374 000361        STA DPNTR,I   YES, UPDATE REGISTER DATA
22375 023746        LDA CRLF
22376 104600        .JLB PUTS
22377 024245
22400 023754        LDA SPC3
22401 104600        .JLB PUTS
22402 024245
22403 000367        LDA PAGE      OUTPUT CHARACTER
22404 104600        .JLB PUTCH
22405 024560
22406 000361        LDA DPNTR,I
22407 023050        JMP .OUTIT     GO SEE WHAT'S NEXT
*
*
22410 104600  CERR2 .JLB CI.IZ     RESTORE INTERFACE
22411 024411
22412 023752  CERR  LDA ERMES       BEEP
22413 104600        .JLB PUTS
22414 024245
22415 022257        JMP COMND     ONE MO' TIME
*
```

```
*
*     OUTPUT THE HELP MESSAGE
*
*
22416 023751  .HELP LDA HELP      OUTPUT THE HELP
22417 104600        .JLB PUTS        MESSAGE
22420 024245
22421 022257        JMP COMND     TRY AGAIN
*
*
*     TOGGLE BASE BETWEEN HEX AND OCTAL
*
*
*
*     ROUTINE TO HANDLE "T" REGISTER ACCESSES
*
*
22422 000363  .TREG STA DFLAG     SET DFLAG FOR MULTIPLE DIGITS
                                     ( DFLAG > 0 )
22423 023746  .T00  LDA CRLF      OUTPUT CR
22424 104600        .JLB PUTS
22425 024245
22426 023753        LDA SPC2      SPACE SPACE
22427 104600        .JLB PUTS
22430 024245
22431 023755  .T02  LDA MMESS     OUTPUT "M"
22432 104600        .JLB PUTS     AND THE  CURRENT
22433 024245
22434 000213        LDA SAVEM        "M" REGISTER
22435 104600        .JLB OUTN         CONTENTS
22436 024344
22437 023756        LDA TMESS     NOW OUTPUT "T" OR "t" DEPENDING
22440 104600        .JLB PUTS
22441 024245
22442               XLA1 '@SAVEM'    GET MAIN MEMORY DATA
22444 104600        .JLB OUTN     OUTPUT THE VALUE
22445 024344
22446 000363        LDB DFLAG     WAS THIS PART OF ( P,A,B,M,& T ) ?
22447 006003        SZB,RSS        IF DFLAG NO. 0 THEN GET INPUT
22450 022257        JMP COMND       ELSE BAIL OUT
22451 104600        .JLB TG.BF
22452 024533
22453 104600        .JLB GETN     GET NEW DATA, MAYBE
22454 024647
22455 022474        JMP .T?       NO NEW DATA, CHECK FOR "N" OR "P"
22456 006003        SZB,RSS       CR?
22457 022465        JMP .T03
22460 023667        CPB .D
```

```
22461 022465        JMP .T03
22462 023676        CPB .N
22463 022465        JMP .T03
22464 022412        JMP CERR      BAD INPUT AFTER VALUE
22465         .T03  XSA1 '@SAVEM'    STORE INTO MAIN MEMORY
*
22467 006002        SZB           WAS IT CR
22470 022474        JMP .T?       NO, SEE WHAT ELSE IT COULD BE
*
22471 002400        CLA
22472 000363        STA DFLAG     INDICATE ECHOING
22473 022423        JMP .T00      ECHO NEW RESULT
*
22474 023645  .T?   CPB .CR        "CR" ?
22475 022257        JMP COMND      YES, EXIT
22476 000213        LDA SAVEM     FETCH "M"
22477 002004        INA           INCREMENT, JUST IN CASE
22500 023676        CPB .N        WAS IT "NEXT" ?
22501 022506        JMP PREV+1    NOT BAD, MUST HAVE BEEN LUCK
22502 023667        CPB .D        WAS IT "DECREMENT" ?
22503 022505        JMP PREV      YES, DECREMENT "M"
22504 022412        JMP CERR      ERROR
*
22505 023725  PREV  ADA N2        DECREMENT "M"
22506 002020        SSA           IS MSB SET ?  NOT VALID FOR "M"
22507 000213        LDA SAVEM     USE OLD VALUE
22510 000213        STA SAVEM     UPDATE "M" SAVE BUFFER
22511 022423        JMP .T00      GO DISPLAY RESULTS
*
*
*
*     ROUTINE TO HANDLE "M" REGISTER STUFF
*
*
22512 000363  .MREG STA DFLAG     MSB = 0 => MULTI DIGIT OUTPUT
22513 000213        LDA SAVEM     AND M
22514 104600        .JLB OUTN        VALUE
22515 024344
*
22516 104600        .JLB TG.BF    TRANSMIT AND GET BUFFER
22517 024533
22520 104600        .JLB GETN   GET NEW VALUE
22521 024647
22522 022546        JMP MT?       NO NEW DATA
22523 006002        SZB           NEW DATA, DID IT END WITH "CR" ?
22524 023704        CPB .T         NO, WAS IT "T" ?
22525 022527        JMP STORM     EITHER WAY GO STORE NEW VALUE
22526 022412        JMP CERR      ERROR, GO BEEP AT THE TURKEY
*
```

```
22527 001665  STORM ELA,CLE,ERA   FORCE MSB TO 0
22530 000213        STA SAVEM     SAVE WHAT'S LEFT
22531 006002        SZB              CR IS OK RESULT
22532 022546        JMP MT?
22533 023746        LDA CRLF
22534 104600        .JLB PUTS
22535 024245
22536 023753        LDA SPC2
22537 104600        .JLB PUTS
22540 024245
22541 023755        LDA MMESS
22542 104600        .JLB PUTS
22543 024245
22544 000213        LDA SAVEM     ECHO THE NEW VALUE
22545 023050        JMP .OUTIT

22546 023704  MT?   CPB .T        WAS IT "T" ?
22547 022423        JMP .T00       YES, GO TO "T REGISTER" DISPLAY
22550 023211        JMP COM01
*
*
*

*
*
*     ROUTINE TO LIST MEMORY CONTENTS
*
22551 000363  .LIST STA DFLAG     MORE THAN ONE DIGIT
22552 104600        .JLB GETN     GET NUMBER TO LIST
22553 024647
22554 002404        CLA,INA       NO NUMBER, ONE LINE
22555 003004        CMA,INA
22556 000224        STA MCNTR     SAVE IN LINE COUNT
22557 006002        SZB
22560 023645        CPB .CR       ENDED WITH CR?
22561 002001        RSS
22562 022412        JMP CERR      ERROR, NO CR AT END
22563 023746        LDA CRLF
22564 104600        .JLB PUTS     OUTPUT CRLF
22565 024245
22566 023755        LDA MMESS
22567 104600        .JLB PUTS     OUTPUT M
22570 024245
22571 000213        LDA SAVEM
22572 000254        STA MPTR
22573 104600        .JLB OUTN     OUTPUT M VALUE
22574 024344
22575 023761        LDA MPMES
```

```
 VIRTUAL CONTROL PANEL PAGE 1

22576 104600       .JLB PUTS     SAY MAP
22577 024245
22600 000214       LDA SAVEW
22601 023720       AND B37
22602 104600       .JLB OUTN     OUTPUT MAP NUMBER
22603 024344
22604 023746       LDA CRLF
22605 104600       .JLB PUTS     OUTPUT CR AND LF
22606 024245
22607 023727  .LLP LDA N8
22610 000225       STA PCNTR
22611 000254       LDA MPTR
22612 000367       STA PAGE   REMEMBER START OF LINE FOR CHAR OUTPUT
22613         .LLP2 XLA1 '@MPTR'  GET DATA
22615 104600       .JLB OUTN     OUTPUT IT
22616 024344
22617 000254       LDA MPTR      NEXT MEMORY LOCATION
22620 002004       INA
22621 001665       ELA,CLE,ERA   CLEAR SIGN BIT
22622 000254       STA MPTR      SAVE IT
22623 000225       ISZ PCNTR
22624 022613       JMP .LLP2
22625 023727       LDA N8
22626 000251       STA CNTR
22627         .LCLP XLA1 '@PAGE'  GET A WORD
22631 001727       ALF,ALF
22632 104600       .JLB .LCH1    OUTPUT A CHAR
22633 022654
22634              XLA1 '@PAGE'
22636 104600       .JLB .LCH1    OUTPUT SECOND CHAR
22637 022654
22640 000367       ISZ PAGE      NEXT WORD
22641 000251       ISZ CNTR
22642 022627       JMP .LCLP
*
22643 023746       LDA CRLF
22644 104600       .JLB PUTS     OUTPUT CRLF AT END OF LINE
22645 024245
22646 000224       ISZ MCNTR     DONE?
22647 002001       RSS
22650 022257       JMP COMND     YES, GET NEXT COMMAND
22651 104600       .JLB .ENQAK   DO ENQACK HANDSHAKE
22652 024270
22653 022607       JMP .LLP
```

```
*
22654 000535  .LCH1 STB RLCH1     SAVE RETURN ADDRESS
22655 023664        AND .DEL        ONLY LOW BYTE
22656 023664        CPA .DEL      IS DELETE??
22657 023645        LDA .CR       YES, MAKE SMALLER ILLEGAL CHAR
22660 023735        ADA N32       SUBTRACT 32
22661 002020        SSA           NEGATIVE??
22662 002400        CLA           YES, MAKE SPACE
22663 023741        ADA D32       PUT CHAR BACK
22664 104600        .JLB PUTCH    OUTPUT CHAR
22665 024560
22666 000535        JMP RLCH1,I   RETURN

*
*
*     ROUTINE TO HANDLE THE SPECIAL REGISTER STUFF
*
*
22667 000363  .REGS STA DFLAG     SET FOR MORE THAN ONE DIGIT
*
22670 104600        .JLB GETCH    GET THE NEXT CHARACTER
22671 024540
22672 000221        STA SVCHR     SAVE CHAR FOR ECHO
22673 023737        LDB D2        SET "B" JUST IN CASE
22674 023666        CPA .C        IS IT "CIR" ?
22675 023043        JMP .CIR
22676 023673        CPA .I        HOW ABOUT "INTERRUPT MASK" ?
22677 023144        JMP .MASK
22700 023707        CPA .W        IS IT WMAP REGISTER
22701 023131        JMP .WMP
22702 023664        CPA .DEL      IS DELETE?
22703 022257        JMP COMND
22704 023675        CPA .M        HOW ABOUT MAP REGISTERS
22705 023214        JMP .MAPS
22706 023700        CPA .P        "PARITY ERROR" MAYBE ?
22707 023152        JMP .PAR
22710 023703        CPA .S        "CPU STATUS SWITCHES" ?
22711 023162        JMP .STAT
22712 023651        CPA .2          I/O REG?
22713 022717        JMP IONXT
22714 023652        CPA .3
22715 006005        INB,RSS
22716 022726        JMP GLCHK
22717 005723  IONXT BLF,RBR         MULT BY 8
22720 000263        STB IORGN
22721 104600        .JLB GETCH      GET NEXT
22722 024540
```

```
22723 023650        XOR ZERO       SAVE LOW BITS
22724 000263        ADA IORGN
22725 000263        STA IORGN      I/O REGISTER NUMBER
22726 000206  GLCHK LDB SAVEG      CHECK GLOBAL REGISTER
22727 107602        OTB 2,C        TURN ON GLOBAL REGISTER
22730 006400        CLB
22731 106502        LIB 2
22732 006003        SZB,RSS
22733 022410        JMP CERR2    ERROR IF NO I/O AT THAT SELECT CODE
22734 000221        LDA SVCHR      GET FIRST CHAR BACK AGAIN
22735 023667        CPA .D
22736 023102        JMP .DIAG
22737 023671        CPA .F
22740 023053        JMP .FLAGS
22741 023651        CPA .2       "I/O"  20 THRU 27 ?
22742 022746        JMP IOREG
22743 023652        CPA .3       "I/O"  30 THRU 32 ?
22744 002001        RSS
22745 022410        JMP CERR2    YOU BLEW IT
*
22746 000263  IOREG LDA IORGN      AND SAVE THE RESULT
22747 023734        ADA N27      WAS IT GREATER
22750 002021        SSA,RSS       THAN 33B ?
22751 022410        JMP CERR2      YES, TOO BEEG
22752 000263        LDA IORGN    CLEAN COPY
22753 023733        ADA N24      WAS IT LESS
22754 002021        SSA,RSS       THAN 30B ?
22755 022766        JMP REGOK      NO => 30, 31, OR 32
22756 000263        LDA IORGN    FRESH COPY
22757 023732        ADA N23      WAS IT GREATER
22760 002021        SSA,RSS       THAN 26B ?
22761 022410        JMP CERR2    YOU DUMMY THERE AIN'T NO 27 !
22762 000263        LDA IORGN    ONCE MORE
22763 023731        ADA N16      IS IT LESS THAN
22764 002020        SSA           20B ?
22765 022410        JMP CERR2    NOW IT'S TOO SMALL
*
22766 104600  REGOK .JLB GETREG  GET REGISTER VALUE
22767 023027
22770 104600        .JLB TG.BF   TRANSMIT AND GET NEW BUFFER
22771 024533
22772 104600        .JLB GETN    GET NEW VALUE
22773 024647
22774 023211        JMP COM01    NO NEW VALUE
22775 006002        SZB
22776 022412        JMP CERR
22777 000525        LDB XEQT+1   NOW MAKE
23000 023722        ADB .B100     IT AN
```

```
23001 000525        STB XEQT+1      "OTA"
23002 000206        LDB SAVEG     SET THE GLOBAL REGISTER
23003 107602        OTB GR,C       TO THE DESIRED VICTIM
23004 000524        JSB XEQT      TRY IT OUT
23005 104600        .JLB CI.IZ      PUT GLOBAL REGISTER BACK
23006 024411
23007 023770        LDA RMESS
23010 104600        .JLB PUTS     ECHO R
23011 024245
23012 000221        LDA SVCHR       GET REGISTER NUMBER BACK
23013 104600        .JLB PUTCH    OUTPUT IT
23014 024560
23015 000525        LDA XEQT+1
23016 023740        AND D7        GET SECOND CHAR
23017 023650        ADA ZERO      MAKE ASCII
23020 104600        .JLB PUTCH
23021 024560
23022 000206        LDA SAVEG
23023 103602        OTA GR,C        ENABLE GLOBAL REGISTER
23024 104600        .JLB GETREG     GET NEW VALUE AND OUTPUT IT
23025 023027
23026 022257        JMP COMND     SEE WHAT'S NEXT
*
23027 000543  GETREG STB RGETREG
23030 023744        LDA .LIA      BUILD THE
23031 000263        IOR IORGN      APPROPRIATE
23032 000525        STA XEQT+1     "LIA" INSTRUCTION
23033 000524        JSB XEQT      GO DO IT !
23034 000503        STA TEMPO     SAVE RESULT
23035 104600        .JLB CI.IZ    PUT THE GLOBAL REGISTER BACK
23036 024411
23037 000503        LDA TEMPO     RESTORE RESULT
23040 104600        .JLB OUTN     OUTPUT THE VALUE
23041 024344
23042 000543        JMP RGETREG,I
*
*
23043 102504  .CIR  LIA 4         GET CURRENT CIR
23044 104600        .JLB COMN     OUTPUT IT AND GET NEW VALUE
23045 023166
23046 102604        OTA 4         UPDATE THE CIR
23047 102504        LIA 4
23050 104600  .OUTIT .JLB OUTN     ECHO NEW VALUE
23051 024344
23052 022257        JMP COMND     SPLIT
*
```

```
*
23053 002400  .FLAGS CLA
23054 102220         SFC 20B
23055 002004         INA
23056 001723         ALF,RAR
23057 102221         SFC 21B
23060 002004         INA
23061 001723         ALF,RAR
23062 102222         SFC 22B
23063 002004         INA
23064 001723         ALF,RAR
23065 102223         SFC 23B
23066 002004         INA
23067 001723         ALF,RAR
23070 102224         SFC 24B
23071 002004         INA
23072 001723         ALF,RAR
23073 102230         SFC 30B
23074 002004         INA
23075 000262         STA RFTMP
23076 104600         .JLB CI.IZ
23077 024411
23100 000262         LDA RFTMP
23101 023050         JMP .OUTIT
*
23102 002404  .DIAG CLA,INA
23103 102602         OTA GR
23104 000206         LDA SAVEG
23105 023654         AND ..B377      GET GLOBAL REGIUSTER
23106 023744         IOR .LIA        MAKE LIA INSTRUCTION
23107 000525         STA XEQT+1
23110 000524         JSB XEQT
23111 000503         STA TEMP0
23112 023737         LDA D2
23113 102602         OTA GR        ESTABLISH DIAGNOSE MODE 2
23114 000206         LDA SAVEG
23115 023654         AND ..B377
23116 023744         IOR .LIA
23117 000525         STA XEQT+1
23120 000524         JSB XEQT
23121 000502         STA TEMP1
23122 104600         .JLB CI.IZ    PUT GLOBAL REGISTER BACK
23123 024411
23124 000503         LDA TEMP0
23125 104600         .JLB OUTN
23126 024344
23127 000502         LDA TEMP1
23130 023050         JMP .OUTIT    GET NEXT COMMAND
```

```
*
*
*
23131 000214  .WMP  LDA SAVEW
23132 104600        .JLB COMN      OUTPUT THE WMAP VALUE
23133 023166
23134 002001        RSS
23135 022257        JMP COMND      NO NEW VALUE
*
23136 000214        STA SAVEW
23137 023720        AND B37
23140               LWD1
23141 000000        DEF 0
23142 000214        LDA SAVEW        ECHO NEW VALUE
23143 023050        JMP .OUTIT
*
*
*
23144 102500  .MASK LIA 0          FETCH INTERRUPT MASK
23145 104600        .JLB COMN        SAME OLE'
23146 023166
23147 102600        OTA 0          NEW INTERRUPT MASK VALUE
23150 102500        LIA 0
23151 023050        JMP .OUTIT
*
*
23152 102505  .PAR  LIA 5          CURRENT PARITY REGISTER
23153 107505        LIB 5,C
23154 101032        ASR 10
23155 104600        .JLB OUTN
23156 024344
23157 102505        LIA 5
23160 023655        AND ..B1777
23161 023050        JMP .OUTIT
*
*
23162 102501  .STAT LIA 1          FETCH THE SWITCHES
23163 023050        JMP .OUTIT
*
*
23164 103507  .VIO  LIA 7,C        GET THE CURRENT VALUE
23165 023050        JMP .OUTIT
*
*
23166 000547  COMN  STB RCOMN      SAVE RETURN ADDRESS
23167 104600        .JLB OUTN     OUTPUT THE CONTENTS OF "A"
23170 024344
23171 104600        .JLB TG.BF
23172 024533
```

```
23173 104600        .JLB GETN    TRY FOR SOME NEW DATA
23174 024647
23175 023211        JMP COM01     NO SUCH LUCK ( NO NEW DATA )
23176 006002        SZB           DATA, BUT WAS THERE A CR ?
23177 022412        JMP CERR       NO, SORRY CHARLIE
23200 000222        STA SACOMN    SAVE A REGISTER
23201 023770        LDA RMESS     START THE ECHO
23202 104600        .JLB PUTS
23203 024245
23204 000221        LDA SVCHR
23205 104600        .JLB PUTCH    OUTPUT THE REGISTER NAME
23206 024560
23207 000222        LDA SACOMN    GET THE VALUE BACK AGAIN
23210 000547        JMP RCOMN,I    YES, WE DONE SOMETHING RIGHT
*
23211 023645  COM01 CPB .CR       NO DATA, BUT WAS IT A CR ?
23212 022257        JMP COMND      YES, GOOD EXIT
23213 022412        JMP CERR       NO, NOT SO GOOD EXIT

*
*
*     PROCESS REGISTER M (MAPS) COMMANDS
*
*
23214 104600  .MAPS .JLB GETN     GET THE MAP NUMBER
23215 024647
23216 022412        JMP CERR        NO NUMBER, ERROR
*
23217 002020        SSA             IS MAP NUMBER NEGATIVE?
23220 022412        JMP CERR        YES, ERROR
*
23221 000260        STB TEMP        SAVE B
23222 023735        LDB N32         MAP NUMBER MUST BE LESS THAN 32
23223 000000        ADB A           SUBTRACT 32 FROM MAP NUMBER
23224 006021        SSB,RSS         RESULT NEGATIVE?
23225 022412        JMP CERR        NO, ERROR SINCE MAP NUMBER > 31
*
23226 000366        STA MAP         SAVE MAP NUMBER (0-31)
23227 000260        LDB TEMP
23230 006003        SZB,RSS         TERMINATED WITH CR?
23231 023235        JMP MAP01       YES, GO OUTPUT A MAP
*
23232 023700        CPB .P          TERMINATED WITH P?
23233 023264        JMP MAPPG     YES, GO FIND OUT WHAT PAGE HE WANTS
23234 022412        JMP CERR        OTHER TERMINATIONS ARE ERRORS
*
```

```
*
23235        MAP01 SMAP            XSM READ MAP (A) INTO MEMORY (B)
23236 000000       DEF 0
23237 000370 MBUF  DEF MPBUF
23240 023726       LDA N4
23241 000225       STA PCNTR       4 LINES OF OUTPUT
23242 023237       LDA MBUF        GET ADDRESS OF MAP
23243 000254       STA MPTR        SAVE POINTER TO IT
23244 023727 MAP15 LDA N8          8 NUMBERS PER LINE
23245 000224       STA MCNTR
23246 023746       LDA CRLF
23247 104600       .JLB PUTS        OUTPUT CR LF
23250 024245
23251 000254 MAP02 LDA MPTR,I      GET A MAP CONTENTS
23252 104600       .JLB OUTN        OUTPUT IT
23253 024344
23254 000254       ISZ MPTR        POINT TO NEXT MAP REGISTER
23255 000224       ISZ MCNTR       DONE WITH LINE?
23256 023251       JMP MAP02       NO, GO DO ANOTHER REGISTER
*
23257 104600       .JLB .ENQAK      TERMINAL READY??
23260 024270
23261 000225       ISZ PCNTR       DONE 4 LINES YET?
23262 023244       JMP MAP15       NO, OUTPUT MORE REGISTERS
*
23263 022257       JMP COMND       DONE, GO GET NEXT COMMAND
*
*
23264 104600 MAPPG .JLB GETN       INPUT PAGE NUMBER
23265 024647
23266 022412       JMP CERR        NO NUMBER, ERROR
23267 023347       JMP MAPP2
*
23270 002020 MAPP1 SSA             IS NUMBER NEGATIVE?
23271 022412       JMP CERR        YES, ERROR
*
23272 023735       ADA N32
23273 002021       SSA,RSS         IS NUMER >= 32?
23274 022412       JMP CERR        YES, ERROR
*
23275 000366       LDA MAP         GET MAP INTO MEMORY
23276              SMAP            XSM READ MAP (A) INTO MEMORY (B)
23277 000000       DEF 0
23300 000370       DEF MPBUF
23301 023237       LDB MBUF        GET BUFFER ADDRESS
23302 000367       ADB PAGE        ADD TO POINT AT PAGE NEEDED
23303 000254       STB MPTR        SAVE PAGE ADDRESS
23304 000001       LDA B,I         GET PAGE
```

```
02192  23305 104600        .JLB OUTN       OUTPUT VALUE AND GET NEW VALUE
       23306 024344
02193  23307 104600        .JLB TG.BF      TRANSMIT AND GET RESULTS
       23310 024533
02194  23311 104600        .JLB GETN       GET NEW VALUE
       23312 024647
02195  23313 023334        JMP NXPG        NO NEW VALUE, SEE IF HE WANTS
                                           ANOTHER PAGE
02196*
02197  23314 000260        STB TEMP        SAVE LETTER INPUT
02198  23315 006003        SZB,RSS         ERROR IF T INPUT SO DONT UPDATE
                                             MAP
02199  23316 023324        JMP NXPG1       CR ENTERED AT END
02200  23317 023676        CPB .N
02201  23320 023324        JMP NXPG1       ONLY CR, N, OR D ARE LEGAL HERE
02202  23321 023667        CPB .D
02203  23322 002001        RSS
02204  23323 022412        JMP CERR        ERROR SINCE BAD CHAR INPUTQ
02205*
02206  23324 000254  NXPG1 STA MPTR,I      PUT NEW PAGE VALUE IN BUFFER
02207  23325 000366        LDA MAP
02208  23326               LMAP            XLM STORE MAP (A) FROM MEMORY (B)
02209  23327 000000        DEF 0
02210  23330 000370        DEF MPBUF
02211  23331 000260        LDB TEMP        GET LETTER BACK
02212  23332 006003        SZB,RSS         WAS CR?
02213  23333 022257        JMP COMND       YES,DONE
02214*
02215  23334 023645  NXPG CPB .CR
02216  23335 022257        JMP COMND
02217  23336 000367        LDA PAGE        GET CURRENT PAGE NUMBER
02218  23337 023676        CPB .N          IS NEXT?
02219  23340 023346        JMP NXPG2       YES, NEXT PAGE
02220  23341 023667        CPB .D          IS PREVIOUS?
02221  23342 002001        RSS
02222  23343 022412        JMP CERR
02223  23344 023724        ADA N1          YES, SUBTRACT 1
02224  23345 002001        RSS
02225  23346 002004  NXPG2 INA
02226  23347 000367  MAPP2 STA PAGE        SAVE NEW PAGE NUMBER
02227  23350 023746        LDA CRLF        ON TO NEXT LINE
02228  23351 104600        .JLB PUTS
       23352 024245
02229  23353 023761        LDA MPMES
02230  23354 104600        .JLB PUTS       OUTPUT MAP
       23355 024245
02231  23356 000366        LDA MAP
02232  23357 104600        .JLB OUTN       OUTPUT MAP NUMBER
       23360 024344
```

```
23361 023762        LDA PGMES
23362 104600        .JLB PUTS     OUTPUT "PAGE"
23363 024245
23364 000367        LDA PAGE
23365 104600        .JLB OUTN     OUTPUT NEW PAGE NUMBER
23366 024344
23367 000367        LDA PAGE      PAGE FOR MAPP1
23370 023270        JMP MAPP1     GO OUTPUT PAGE AND GET NEW VALUE
*

*
*
*    PROCESS "%" COMMANDS
*
*
23371 104600  CNTRL .JLB GETS     GET REST OF STRING
23372 024766
23373 023746        LDA CRLF
23374 104600        .JLB PUTS     OUTPUT CRLF
23375 024245
23376 023643        LDA SPTR,I    GET FIRST CHAR
23377 023654        AND ..B377
23400 000503        STA TEMPO
23401 023705        CPA .U        USER
23402 023434        JMP .USER,I
23403 023674        CPA .L        LOAD SOMETHING ?
23404 023553        JMP .LOAD      GO SEE WHAT IT IS
23405 023707        CPA .W        WRITE SOMETHING
23406 023553        JMP .LOAD      SORT IT OUT LATER
23407 023665        CPA .B        BOOT MAYBE ?
23410 023553        JMP .LOAD      LOAD 'EM AND RUN
23411 023643        LDA SPTR
23412 002004        INA
23413 000000        LDA A,I       GET SECOND WORD
23414 001727        ALF,ALF
23415 023654        AND ..B377    MASK OFF NEXT CHAR
23416 002002        SZA           MUST BE ZERO ( NO NEXT CHAR FOR
                                     FOLLOWING COMMANDS)
23417 022412        JMP CERR      ERROR SINCE CHARS AFTER COMMAND
23420 000503        LDA TEMPO     GET CHAR BACK AGAIN
23421 023670        CPA .E        EXECUTE?
23422 023507        JMP .EX       GO EXECUTE PROGRAM
23423 023702        CPA .R        RUN?
23424 023515        JMP .RUN      GO RUN FROM CURRENT P
23425 023704        CPA .T        TEST?
23426 023545        JMP .TRAC     GO DO PRETEST
23427 023666        CPA .C        MEMORY CLEAR ?
23430 023445        JMP CLRM       GO ZERO MEMORY
```

```
23431 023700       CPA .P        PRESET ??
23432 023435       JMP PRSET      YES, GO DO A "CLC 0,C"
23433 022412       JMP CERR      NUTHIN'

*
23434 030002 .USER DEF 30002B    JUMP TO USER ROM CODE
*
*
*
*
*
*     PRESET THE MACHINE
*
*
*
23435 105762 PRSET JLY .PSET     BLOW EVERYTHING AWAY
23436 021633
23437 104600       .JLB CI.IZ     FIX UP THE INTERFACE CARD
23440 024411
23441 023760       LDA PRMES
23442 104600       .JLB PUTS     ;*** PRESET ***
23443 024245
23444 023502       JMP FXRX      THAT'S ALL GET NEXT COMMAND
*
*
*     ROUTINE TO CLEAR MEMORY  ( ADDRESSES 2 TO 77777 )
*
*
*
23445 023757 CLRM  LDA CLMES
23446 104600       .JLB PUTS     SAY CLEARING MEMORY
23447 024245
23450 002400       CLA
23451 000366       STA MAP       START WITH MAP 0
23452              SMAP          STORE MAP TO MEMORY
23453 000000       DEF 0
23454 000430       DEF MZSV      SAVE MAP ZERO IN MAP ZERO SAVE AREA
23455 002400       CLA
23456              LWD1          USE MAP ZERO FOR CLEAR MEMORY
23457 000000       DEF 0            GET ZERO FROM A REGISTER
*
23460 000366 CLRM1 LDA MAP       GET NEXT MAP TO DO
23461 000246       CPA MSIZE     DONE?
23462 023476       JMP CLDN      YES, NO MORE MAPS
*
23463 105762       JLY STMAP     SET MAP ZERO SEQUENTIALLY
23464 027647
```

```
*
23465 002400        CLA          START ADDRESS ZERO
23466               XSA1 '0'     CLEAR FIRST LOCATION
23470 006404        CLB,INB
23471 105745        LDX ..B77777 COUNT FOR 32K
23472 023656
23473               MW11         CLEAR 32K MEMORY
*
23474 000366        ISZ MAP      ON TO NEXT 32K
23475 023460        JMP CLRM1
*
*
23476 002400 CLDN  CLA
23477               LMAP         RESTORE MAP ZERO AS WAS
23500 000000        DEF 0
23501 000430        DEF MZSV     RESTORE FROM BUFFER
23502 000214 FXRX  LDA SAVEW    RESTORE REGISTER X (WMAP VALUE)
23503 023720        AND B37
23504               LWD1
23505 000000        DEF 0        PUT ALT 1 MAP BACK AS IT WAS
23506 022257        JMP COMND    YES, BACK TO PROMPT
*
*
23507 002400 .EX   CLA
23510 000205        STA SAVEB    FOR %E  B HAS ZERO
23511 003000        CMA
23512 000204        STA SAVEA    A HAS ALL 1S
23513 023737        LDA D2
23514 000203        STA SAVEP    START AT P=2
*
*
23515 003400 .RUN  CCA              SENT ALL 1'S
23516 102624        OTA 24B      TO TELL OS WE'VE BEEN HERE
23517 104600 EXEX  .JLB ENDVCP  TELL CARD TO LEAVE VCP MODE
23520 023533
23521 104600 BEXEX .JLB RSTOR   NOW PUT EVERTHING BACK
23522 024205
23523 102100        STF 0        TURN 'EM BACK ON, IF THEY WERE ON
23524 102702        STC 2        TURN ON BREAK
23525               XJMP 'SAVEW','@SAVEP' ;LAUNCH THE USER WITH HIS
                                                OLD WMAP
*
*                 IF BREAK DISABLED
23530 104600 EXEX2 .JLB CI.IZ   FIX INTERFACE CARD
23531 024411
23532 023515        JMP .RUN     RESTART
*                        SEND END VCP MODE IF INTELLEGENT DRIVER
*
```

```
23533 000530  ENDVCP STB RENDV    SAVE RETURN ADDRESS
23534 104600         .JLB CI.ID   IS INTELLEGENT??
23535 024513
23536 000530         JMP RENDV,I  NO, DO NOTHING
23537 023544         LDA VCPEX    YES, GET END VCP COMMAND
23540 104600         .JLB DS.FT   SEND IT TO CARD AND WAIT FOR FLAG
23541 024463
23542 000000         NOP          TIME OUT, DONT WORRY ABOUT IT
23543 000530         JMP RENDV,I  RETURN
23544 062000  VCPEX OCT 62000     EXIT VCP COMMAND
*
*     SINGLE STEP ROUTINE
*
*
*.STEP .JLB GETCH   GET ONE CHARACTER
*       CPA .CR       IS IT "CR" ?
*       JMP *+2       YES, GO TO IT
*       JMP CERR      NO, YOU BLEW IT
*       STA TFLAG     SET FLAG NON-ZERO => STEP OR TRACE
*STEP1 .JLB RSTOR   RESTORE THE REGISTERS
*       STF 0         TURN INTERRUPTS BACK ON, IF NEEDED
*       CLC 3         START I/O CHIP SEQUENCE
*       STC 2         ENABLE "BREAK"
*       XJMP SAVEW,@SAVEP ; LAUNCH THE USER WITH HIS OLD WMAP
*
*
*     SELF TEST.  GO DO CLC 0,C AND THEN ON TO START
*
*
23545 003400  .TRAC CCA
23546 000304         STA VCPTFLG   FLAG FOR SELFTEST
23547 105762         JLY .PSET     CLEAR OUT MACHINE TO POWER ON STATE
23550 021633
23551 023552         JMP *+1,I     GO TEST
23552 020004         DEF START+2
*
*
23553 105762  .LOAD JLY .PSET     RESET I/O FOR LOADERS
23554 021633
23555 023643         LDA SPTR      GET FIRST CHAR
23556 002004         INA
23557 000000         LDA A,I
23560 023657         CPA .CT       CARTRIDGE TAPE ?
23561 023571         JMP .CTU       YES, THE LEFT ONE
23562 023662         CPA .RM       THAT'S "R" AS IN PROM
23563 023602         JMP .ROM       LOAD FROM PROM
23564 023661         CPA .DC       DISC MAYBE ?
23565 023624         JMP .DISC      A LITTLE HPIB IF YOU PLEASE
```

```
23566 023660        CPA .DS       DS LOADER??
23567 023615        JMP .DISTS    LOAD OVER DS
23570 022410        JMP CERR2     THAT AIN'T ONE OF MINE
*
23571 023716  .CTU  LDA .B20      DEFAULT
23572 104600        .JLB SCNSC    PARSE SCETC
23573 027572
23574 002404        CLA,INA
23575 102601        OTA CPUST     SAY IN LOADER
23576 104600        .JLB CTU      DO THE LOAD
23577 026004
23600 023723        JMP .BOOT?,I     ARE WE BOOTING
23601 023632        JMP BTERR     ERROR RETURN
*
*
*
23602 000503  .ROM  LDA TEMPO     READ OR WRITE ?
23603 023707        CPA .W
23604 022410        JMP CERR2     CANNOT WRITE TO ROM
23605 023715        LDA RMSC      DEFAULT
23606 104600        .JLB SCNSC    GET SELECT CODE AND FILE NUMBER
23607 027572
23610 104600        .JLB RMLDR    GO TO PROM LOADER
23611 026364
23612 023723        JMP .BOOT?,I     GOOD RETURN
23613 023632        JMP BTERR     ERROR RETURN
23614 025772  .LDER DEF MES62
*
*
23615 023714  .DISTS LDA DSSC      DEFAULT
23616 104600         .JLB SCNSC    PARSE SELECT CODE
23617 027572
23620 104600        .JLB DSLD     GO TO DS LOADER
23621 025040
23622 023723        JMP .BOOT?,I     GOOD RETURN
23623 023632        JMP BTERR     ERROR
*
23624 023713  .DISC LDA DCSC      DEFAULT
23625 104600        .JLB SCNSC    GET SELECT CODE ETC FROM STRING
23626 027572
23627 104600        .JLB DCLDR  GO TO HPIB LOADER
23630 026531
23631 023723        JMP .BOOT?,I    GOOD RETURN
*
23632 104600  BTERR .JLB CI.IZ     ENABLE VCP
23633 024411
23634 023614        LDA .LDER   OUTPUT ERROR MESSAGE
```

```
02443  23635 104600        .JLB PUTS
       23636 024245
02444  23637 000265        LDA LERR     GET ERROR NUMBER
02445  23640 104600        .JLB OUTD    OUTPUT ERROR NUMBER
       23641 024312
02446  23642 022257        JMP COMND    ERROR RETURN
02447*
02448*
02449*
02450  23643 000306  SPTR  DEF STRNG        POINTER TO STRING
02451*

02453*     CONSTANTS AND EQUATES
02454*
02455*
02456  23644 103003  .ENTI OCT 103003   HALT 03,C FOR REENTERING FRONT PANEL
02457  23645 000015  .CR   OCT 15              "CARRIAGE RETURN"
02458  23646 000024  .CTLT OCT 24        CONTROL T
02459  23647 000045  .%    OCT 45              "%"
02460  23650 000060  ZERO  OCT 60              "0"
02461  23651 000062  .2    OCT 62
02462  23652 000063  .3    OCT 63
02463  23653 000077  .?    OCT 77              "?"
02464  23654 000377  ..B377  OCT 377
02465  23655 001777  ..B1777 OCT 1777
02466  23656 077777  ..B77777 OCT 77777
02467  23657 041524  .CT   OCT 041524
02468  23660 042123  .DS   OCT 042123
02469  23661 042103  .DC   OCT 042103
02470  23662 051115  .RM   OCT 051115
02471  23663 000101  .A    OCT 101             "A"
02472  23664 000177  .DEL  OCT 177
02473  23665 000102  .B    OCT 102         ETC
02474  23666 000103  .C    OCT 103          ETC
02475  23667 000104  .D    OCT 104           ETC
02476  23670 000105  .E    OCT 105
02477  23671 000106  .F    OCT 106
02478  23672 000107  .G    OCT 107
02479  23673 000111  .I    OCT 111
02480  23674 000114  .L    OCT 114
02481  23675 000115  .M    OCT 115
02482  23676 000116  .N    OCT 116
02483  23677 000117  .O    OCT 117
02484  23700 000120  .P    OCT 120
02485  23701 000121  .Q    OCT 121
02486  23702 000122  .R    OCT 122
02487  23703 000123  .S    OCT 123
02488  23704 000124  .T    OCT 124
```

```
02489  23705 000125  .U     OCT 125
02490  23706 000126  .V     OCT 126
02491  23707 000127  .W     OCT 127
02492  23710 000130  .X     OCT 130
02493  23711 000131  .Y     OCT 131
02494  23712 000132  .Z     OCT 132
02495  23713 002027  DCSC   OCT 002027    SELECT CODE OF DISC
02496  23714 000024  DSSC   OCT 000024  SELECT CODE FOR DS LOADER
02497  23715 000022  RMSC   OCT 000022    SELECT CODE OF ROM CARD
02498  23716 000020  .B20   OCT 000020
02499  23717 000010  .B10   OCT 000010
02500  23720 000037  B37    OCT 000037
02501  23721 100000  BIT15 OCT 100000
02502  23722 000100  .B100 OCT 000100
02503  23723 024077  .BOOT? DEF BOOT?
02504  23724 177777  N1     DEC -1
02505  23725 177776  N2     DEC -2
02506  23726 177774  N4     DEC -4
02507  23727 177770  N8     DEC -8
02508  23730 177766  N10    DEC -10
02509  23731 177760  N16    DEC -16
02510  23732 177751  N23    DEC -23
02511  23733 177750  N24    DEC -24
02512  23734 177745  N27    DEC -27
02513  23735 177740  N32    DEC -32
02514  23736 177720  N48    DEC -48
02515  23737 000002  D2     DEC +2
02516  23740 000007  D7     DEC +7
02517  23741 000040  D32    DEC +32
02518  23742 000203  BUFF   DEF SAVEP
02519  23743 000470  .DIG1 DEF DIG1
02520  23744 102500  .LIA   OCT 102500
02521  23745 000524  .RTRN JMP XEQT,I

02523*
02524*
02525*     MESSAGE "DEFS"
02526*
02527  23746 025340  CRLF   DEF MES00
02528  23747 025313  VERMG DEF MES01
02529  23750 025342  PRMPT DEF MES02
02530  23751 025350  HELP   DEF MES09
02531  23752 025762  ERMES DEF MES46
02532  23753 025667  SPC2   DEF MES11
02533  23754 025703  SPC3   DEF MES22
02534  23755 025677  MMESS DEF MES15
02535  23756 025701  TMESS DEF MES16
02536  23757 025705  CLMES DEF MES32
```

```
02537  23760 025714  PRMES DEF MES33
02538  23761 025720  MPMES DEF MES35
02539  23762 025723  PGMES DEF MES36
02540  23763 025726  KMES  DEF MES37
02541  23764 025751  ECMES DEF MES43
02542  23765 025734  PEMES DEF MES38
02543  23766 025742  SELFERR DEF MES41
02544  23767 025734  SOFTERR DEF MES38
02545  23770 025756  RMESS  DEF MES44
02546  23771 025671  PMESS  DEF MES12
02547  23772 025675  AMESS  DEF MES13
02548  23773 025676  BMESS  DEF MES14
02549        023774  EOP1  EQU *
02550*
02551  24000               ORG EPROM+4000B
```

```
      024000  P2    EQU *
*
*    CONSTANTS AND SUCH FOR THIS PAGE
*
24000 000010  ..BKS  OCT 10        ASCII "BACKSPACE"
24001 000102  ..B    OCT 102
24002 023632  .BTERR DEF BTERR    DISC ERROR
24003 000177  ..DEL  OCT 177      ASCII DELETE
24004 000005  .ENQ   OCT 5        ASCII ENQ
24005 023435  .PRSET DEF PRSET      ENTRY FOR PRESET
24006 022172  .AGAIN DEF AGAIN
24007 022257  .COMND DEF COMND
24010 025704  .SPC1  DEF MES22+1
24011 025667  .SPC2  DEF MES11
24012 025347  ..BEL  DEF MES07
24013 001700  STRTR  DEF 1700B     COMMUNICATION AREA
24014 023517  EXEX.P1 DEF EXEX     CROSS TO PAGE 1
24015 022412  CERR.P1 DEF CERR           "
24016 023521  ..RUN   DEF BEXEX
24017 022255  ..NEXT  DEF NEXT
24020 000015  ...CR  OCT 15              "CARRIAGE RETURN"
24021 000127  ...W   OCT 127
24022 000077  ..?    OCT 77              "?"
24023 000060  .ZERO  OCT 60              "0"
24024 000001  .B1    OCT 000001
24025 000007  .B7    OCT 7
24026 000017  .B17   OCT 000017
24027 000024  .B24   OCT 000024
24030 000101  .B101  OCT 000101
      024003  .B177  EQU ..DEL
24031 000377  .B377  OCT 000377
24032 060000  .B60K  OCT 060000
24033 177777  .N1    DEC -1
24034 177400  .BLR   OCT  177400
24035 177773  .N5    DEC -5
24036 177772  .N6    DEC -6
24037 177771  .N7    DEC -7
24040 177770  .N8    DEC -8
24041 177766  .N10   DEC -10
24042 177730  ..N40   DEC -40
24043 177720  .N48   DEC -48       THIS IS NEGATIVE "ASCII ZERO"
24044 177745  .N27   DEC -27
24045 177746  .N26   DEC -26
24046 101400  .N32000 DEC -32000
24047 177677  .N65   DEC -65       THIS IS NEGATIVE "ASCII A"
24050 177637  .N97   DEC -97
24051 177645  .N91   DEC -91
24052 000002  .D2    DEC +2
```

```
02601  24053 000012  .D10   DEC +10
02602  24054 000020  .D16   DEC +16
02603  24055 000040  .D32   DEC +32
02604  24056 000050  .D40   DEC +40
02605  24057 000100  .D64   DEC +64
02606  24060 000141  .D97   DEC +97
02607  24061 000133  .D91   DEC +91
02608  24062 000466  .D310  DEC 310
02609  24063 000470  .D312  DEC 312
02610  24064 000467  .D311  DEC 311
02611  24065 000472  .D314  DEC 314
02612  24066 000500  .D320  DEC 320
02613  24067 000473  .D315  DEC 315
02614  24070 000475  .D317  DEC 317
02615  24071 000470  ..DG1  DEF DIG1

02617*
02618  24072 024001  MRBT  LDA ..B           MAKE IT A BOOT
02619  24073 000503        STA TEMPO
02620  24074 024765  MRBT2 LDA .SPTR
02621  24075 001200        RAL
02622  24076 000360        STA STORE.POINTER
02623*
02624  24077 024052  BOOT? LDA .D2
02625  24100 000203        STA SAVEP          SET P FOR STARTING ADDRESS
02626  24101 002400        CLA
02627  24102 000366        STA MAP
02628  24103               LWD1               STORE STRING THROUGH MAP 0
02629  24104 000000        DEF 0              POINT AT ZERO
02630  24105 105762        JLY STMAP          SET UP MAP ZERO AGAIN
       24106 027647
02631  24107 024013        LDA STRTR          POINT AT COMMUNICATION AREA
02632  24110 000205        STA SAVEB   B SHOULD POINT AT COMMUNICATION AREA
02633  24111 000246        LDB MSIZE          GET MEMORY SIZE
02634  24112               XSB1 '@A'         CROSS STORE
02635  24114 002004        INA
02636  24115 000360        LDB STORE.POINTER
02637  24116 007004        CMB,INB            SUBTRACT STORE.POINTER
02638  24117 024765        ADB .SPTR
02639  24120 024765        ADB .SPTR          ADD START OF CHARS * 2
02640  24121 000356        ADB LSTR           LAST CHAR B HAS NUMBER OF CHARS
                                                 IN STRING
02641  24122               XSB1 '@A'          SAVE NUMBER OF CHARS
02642  24124 002004        INA
02643  24125 001200        RAL                MAKE IT A BYTE ADDRESS
02644  24126 006004        INB                COPY ONE EXTRA CHARACTER
02645  24127 105741        CBX                SAVE COUNT IN X
02646  24130 000000        LDB A              STORE LOCATION
```

```
24131 000360        LDA STORE.POINTER GET FROM LOCATION
24132               MBO1               MOVE STRING TO USER MAP
24133 000503        LDA TEMPO
24134 024001        CPA ..B          IS BOOT?
24135 024016        JMP ..RUN,I         YES, GO DO IT
24136 024017        JMP ..NEXT,I         NO, GO GET COMMAND
*
*
*     REENT IS WHEN BOOTEX OR A DIAGNOSTIC CALLS BACK THE FRONT PANEL
*
*
24137 002004  REENT INA      POINT AT HPIB ADDRESS
24140               XLB1 '@A'         GET SUBCHANNEL
24142 000270        STB SUBCH
24143 002004        INA      POINT AT UNIT NO.
24144               XLB1 '@A'         GET UNIT
24146 000267        STB UNIT
24147 002004        INA      POINT AT SECTOR NUMBER
24150               XLB1 '@A'         GET SECTOR NUMBER
24152 000274        STB FILE          SAVE IT           VW=1
24153 002004        INA      POINT AT CYLINDER OFFSET
24154               XLB1 '@A'         GET CYLINDER OFFSET
24156 000273        STB CYLNDR.OFFSET
24157 000275        STB HEAD.CYLINDER  ; SAVE IT      VW=2
24160 002004        INA
24161               XLB1 '@A'         GET VECTOR WORD THREE
24163 000276        STB SECTR.TRACK                   VW=3
24164 105762        JLY .PSET
24165 021633
24166 002400        CLA
24167 000265        STA LERR      NO LOADER ERROR
24170 104600        .JLB DCRLD        GO LOAD FROM DISK
24171 026524
24172 024175        JMP RENT2     GOOD RETURN
24173 102702        STC 2         ENABLE BREAK
24174 024002        JMP .BTERR,I  ERROR RETURN
24175 024001  RENT2 LDA ..B       B FOR BOOT
24176 000202        LDB SAVEE     GET E REG VALUE
24177 004010        SLB           IS SET?
24200 002400        CLA           ZERO FOR BOOT FLAG
24201 000503        STA TEMPO     SAVE BOOT FLAG
24202 003400        CCA
24203 000204        STA SAVEA     A GETS -1 FOR CALL BACK
24204 024074        JMP MRBT2
*
```

```
*
*     ROUTINE TO RESTORE "A", "B",  ETC BEFORE RUNNING
*
*     CALLING SEQUENCE:
*
*               JLB*  RSTOR
*               P+1  INTERRUPTS WERE ON
*               P+2  INTERRUPTS WERE OFF
*
*
*
24205 000546  RSTOR STB RRSTO     SAVE RETURN ADDRESS
24206 002400        CLA
24207 102601        OTA CPUST     INDICATE IN USER PROGRAM
24210 105745        LDX SAVEX     RESTORE X AND Y
24211 000207
24212 105755        LDY SAVEY
24213 000210
24214 000200        LDA SAVEI     GET INTERRUPT STATUS
24215 006400        CLB           CLEAR IT FOR
24216 000200        STB SAVEI      NEXT TIME
24217 000507        STB FIRST     RESET NOT FIRST TIME FLAG
24220 002011        SLA,RSS       WERE INTERRUPTS ON ?
24221 000546        ISZ RRSTO      NO, BUMP RETURN ADDRESS
24222 000206        LDA SAVEG     FETCH OLD GLOBAL REGISTER
24223 001621        ELA,ARS       IF IT WAS ON => E <= 1
24224 002002        SZA           WAS THE GR ZERO, IF SO NO OTA
24225 102602        OTA GR        RESTORE GLOBAL REGISTER VALUE
24226 002040        SEZ           WAS IT ON ?
24227 103102        CLF GR         YES, TURN IT BACK ON
24230 000201        LDA SAVEO     FETCH "O" REPLICA
24231 103101        CLO           WAS IT
24232 000010        SLA            OFF ?
24233 102101        STO             NO, BUT YOU WERE CLOSE
24234 000202        LDA SAVEE     PUT "E" BACK
24235 001500        ERA            THE WAY YOU FOUND IT
24236 000212        LDA SAVEZ     RESTORE Z
24237               CAZ
24240 000211        LDA SAVEQ     RESTORE Q, POSSIBLY TURN ON R3
24241               CACQ
24242 104200        DLD SAVEA+2000B NOW "A" AND  "B"
24243 002204
24244 000546        JMP RRSTO,I
*
*
*     OUTPUT A MESSAGE, TERMINATE ON NULL BYTE
*     ENTER WITH "A" = DEF MESSAGE
*
```

```
                    *
24245 000527  PUTS   STB RPUTS      SAVE RETURN ADDRESS
24246 001200         RAL            MAKE IT A BYTE ADDRESS
24247 000255         STA PPNTR      SAVE MESSAGE DEF
24250 024042         LDA ..N40
24251 000226         STA PUTCT      COUNTER FOR ENQ ACK
                    *
24252 000255  P.1    LDB PPNTR      FETCH A WORD
24253 105763         LBT
24254 002003         SZA,RSS        NULL ?
24255 000527         JMP RPUTS,I     YES, BAIL OUT
24256 104600         .JLB PUTCH     NO, PRINT IT
24257 024560
24260 000255         ISZ PPNTR       YES, BUMP POINTER
24261 000226         ISZ PUTCT      CHECK CHAR COUNT
24262 024252         JMP P.1        DO IT ALL AGAIN
24263 024042         LDA ..N40
24264 000226         STA PUTCT      COUNT FOR NEXT 40 CHARS
24265 104600         .JLB .ENQAK      DO ENQ ACK HANDSHAKE
24266 024270
24267 024252         JMP P.1
                    *

24270 000531  .ENQAK STB RENQAK
24271 104600         .JLB CI.ID     IDENTIFY
24272 024513
24273 024277         JMP .ENQAS     ASCII
24274 104600         .JLB TG.TB     TRANSMIT BUFFER TO DS
24275 024520
24276 000531         JMP RENQAK,I   RETURN
                    *
24277 102501  .ENQAS LIA CPUST     GET SWITCHES
24300 001727         ALF,ALF
24301 024055         AND .D32      MASK ENQ SWITCH
24302 002002         SZA           MUST BE ZERO FOR ENQ TO WORK
24303 000531         JMP RENQAK,I   RETURN DOING NOTHING
                    *
24304 024004         LDA .ENQ       GET ENQ CHAR
24305 104600         .JLB PUTCH
24306 024560
24307 104600         .JLB GETCH
24310 024540
24311 000531         JMP RENQAK,I RETURN
                    *
```

```
*
*
*     ROUTINE TO OUTPUT WHAT IS IN A AS A DECIMAL INTEGER.
*     POSITIVE NUMBERS ONLY
*
24312 000537  OUTD  STB ROUTD
24313 024071        LDB ..DG1     POINT TO DIGIT BUFFER
24314 000217        STB PNTR
24315 006400        CLB
24316 000251        STB CNTR      DIGIT COUNTER
24317 006400  OTDL  CLB           MAKE TWO WORD VALUE
24320 100400        DIV .D10      DIVIDE BY 10
24321 024053
24322 000217        STB PNTR,I    SAVE REMAINDER AS DIGIT
24323 000217        ISZ PNTR      POINT AT NEXT DIGIT
24324 000251        ISZ CNTR      ADD 1 TO COUNT
24325 002002        SZA           QUOTIENT ZERO YET??
24326 024317        JMP OTDL      NO, GET NEXT DIGIT
*
24327 000251        LDA CNTR
24330 003004        CMA,INA       MAKE COUNT NEGATIVE
24331 000251        STA CNTR
24332 000217  OTDL2 LDA PNTR
24333 024033        ADA ..N1      SUBTRACT ONE
24334 000217        STA PNTR      POINT AT THE PREVIOUS CHAR
24335 000217        LDA PNTR,I    GET DIGIT
24336 024023        ADA .ZERO     MAKE ASCII
24337 104600        .JLB PUTCH    OUTPUT THE CHAR
24340 024560
24341 000251        ISZ CNTR      MORE LEFT??
24342 024332        JMP OTDL2     YES
*
24343 000537        JMP ROUTD,I   RETURN

*
*
*     ROUTINE TO OUTPUT HEX OR OCTAL DIGITS     *
*     ENTER WITH NUMBER IN "A" REGISTER
*     IF "DFLAG" < 0 THEN OUTPUT ONLY ONE DIGIT
*
*
24344 000534  OUTN  STB ROUTN     RETURN ADDRESS
24345 000260        STA TEMP      SAVE NUMBER
24346 024010        LDA .SPC1      GO OUTPUT ONE SPACE
24347 104600        .JLB PUTS       SPACE
24350 024245
24351 000260        LDA TEMP      RESTORE NUMBER
24352 000363        LDB DFLAG     FETCH DATA TYPE FLAG
```

```
24353 006021        SSB,RSS       ONE DIGIT ?  ( < 0 => ONE DIGIT )
24354 024362        JMP OT1        NOPE, MORE THAN THAT
24355 024024        AND .B1
24356 024023        IOR .ZERO
24357 104600        .JLB PUTCH    YEP, JUST ONE OUTPUT IT
24360 024560
24361 024405        JMP OT2       NOW LEAVE
24362 024035  OT1   LDB .N5        SET DIGIT     ( DEC -5 )
24363 000251        STB CNTR       COUNTER
*
*    NONSENSE TO HANDLE SIGN BIT IN OCTAL MODE
*
24364 000066        CLE,ELA
24365 000260        STA TEMP      SAVE PARTIAL
24366 024023        LDA .ZERO      IS IT
24367 002040        SEZ            ZERO ?
24370 002004        INA             NO, MAKE IT A ONE
24371 104600        .JLB PUTCH      PRINT IT
24372 024560
24373 000260        LDA TEMP      FETCH PARTIAL
24374 001723  L1    ALF,RAR       NEXT DIGIT
24375 000260        STA TEMP      SAVE NEW PARTIAL
24376 024025        AND .B7       SAVE ONLY LOW NIBBLE    ( DEC +15 )
24377 024023        ADA .ZERO      IS IT GREATER          ( DEC -10 )
24400 104600        .JLB PUTCH     PRINT IT
24401 024560
24402 000260        LDA TEMP      FETCH PARTIAL
24403 000251        ISZ CNTR      DONE ?
24404 024374        JMP L1         NOPE
24405 024010  OT2   LDA .SPC1       YEP, NOW
24406 104600        .JLB PUTS    OUTPUT 2 SPACES
24407 024245
24410 000534        JMP ROUTN,I   BYE BYE
*
*
*
*
*    ROUTINE TO "BEEP" AT ERRORS
*
*
*
*
*    OUTPUT WHAT EVER IS IN "A"
*
```

```
 VIRTUAL CONTROL PANEL PAGE 2

*
*UTCH STB RPUTC
*     OTA DATA      SEND CHARACTER TO TERMINAL
*     STC DATA,C    START OPERATION
*     SFS DATA      DONE YET ?
*     JMP *-1        NOPE
*     JMP RPUTC,I    YES, EXIT
*
*
*     GET ONE CHARACTER, RETURNED IN THE LOW END OF "A"
*
*
*ETCH STB RGETC
*     LDA ..ICW       PUT ASIC INTO
*     OTA CMND       INPUT MODE
*     STC DATA,C    START INPUT OPERATION
*T.00 SFC DATA      IS IT SOUP YET?
*     JMP GT.01      YES !
*     LIA STATS     NO - CHECK FOR BREAK
*     RAL
*     SSA
*     JMP .PRSET,I  YES - BREAK
*     JMP GT.00     NO KEEP WAITING
*
*T.01 LIA DATA      OK, LET'S SEE WHAT YOU'VE DONE
*     AND .B377      WELL, HALF OF IT ANYWAY
*     STA CHAR      SAVE IT FOR ECHO
*     JMP RGETC,I

*
*
24411 000575  CI.IZ STB RCI.IZ
24412 102102        STF GR        CLEAR ALL INTERFACES
24413 002400        CLA
24414 102602        OTA GR
24415 000514        LDA VCPSC    (P+1) ASCII
24416 103602        OTA GR,C      TURN ON CARD
24417 107723        CLC 23B,C     TURN OFF DMA
24420 104600        .JLB CI.ID    INITIALIZE INTERFACE
24421 024513
24422 024616        JMP AS.IZ
*
24423 002400  DS.IZ CLA           INITIALIZE DS INTERFACE
24424 000000        ISZ A         DELAY TIME  THIS MUST BE > 1 MS
24425 024424        JMP *-1       FOR SET UP
24426 024434        LDA VCPDS     VCP COMMAND
24427 104600        .JLB DS.FT    TRY IT
24430 024463
```

```
  VIRTUAL CONTROL PANEL PAGE 2

02906  24431 024005        JMP .PRSET,I   NO GOOD
02907  24432 104600        .JLB CS.CM    TELL CARD AGAIN TO GO INTO VCP MODE
       24433 024577
02908  24434 067400  VCPDS OCT 67400
02909  24435 000575        JMP RCI.IZ,I  RETURN
02910*
02911  24436 024031  DS.TG LDA .B377     ADD RUB OUT <REQUEST INPUT>
02912  24437 104600        .JLB OUT2C    OUTPUT TWO CHARACTERS        <<<<<
       24440 024566
02913  24441 104600        .JLB CS.CM    TELL CARD TO TRANSMIT
       24442 024577
02914  24443 060400        OCT 60400
02915  24444 104600        .JLB CS.CM    NOW ASK FOR BUFFER
       24445 024577
02916  24446 061400        OCT 61400
02917  24447 000577        JMP RTG.BF,I  RETURN
02918*
02919  24450 104600  DS.IN .JLB CS.CM    ASK FOR INPUT
       24451 024577
02920  24452 061000        OCT 61000
02921  24453 024031        AND .B377     MASK
02922  24454 024545        JMP GETCR     RETURN VIA GETCH
02923*
02924  24455 024032  DS.OT IOR .B60K     DS PUT BYTE REQUEST
02925  24456 104600        .JLB I.O
       24457 024607
02926  24460 000541        JMP RPUTC,I
02927*

02929*
02930  24461 000602  DS.WF STB RDS.FT    SAVE RETURN ADDRESS
02931  24462 024466        JMP DS.FT+3  SKIP OUTPUT JUST FLAG
02932*
02933  24463 000602  DS.FT STB RDS.FT    RETURN ADDRESS
02934  24464 102630        OTA DATA
02935  24465 103730        STC DATA,C
02936  24466 024046        LDB .N32000  40 SEC TIME OUT, MACHINE INDEPENDENT
02937  24467 102230  FTLP  SFC DATA      WAIT FOR FLAG
02938  24470 024477        JMP FTGF      GOT IT
02939  24471 000245        LDA TBGCNT
02940  24472 002306        CCE,INA,SZA   WAIT 1.25 MS
```

```
24473 024472       JMP *-1
24474 000001       ISZ B
24475 024467       JMP FTLP      NOT DONE, CHECK FLAG AGAIN
24476 000602       JMP RDS.FT,I  TIMED OUT
24477 000602 FTGF  ISZ RDS.FT     GOT THE FLAG
24500 000602       JMP RDS.FT,I  GOOD RETURN
             *
24501 067400 DSVCP OCT 67400
             *
24502 000603 DS.CM STB RDS.CM    SAVE RETURN ADDRESS
24503 000001       LDA B,I       GET COMMAND
24504 104600       .JLB DS.FT    WAIT FOR FLAG
24505 024463
24506 025140       JMP DSLER     DS LOADER ERROR
24507 102530       LIA DATA      GET DATA
24510 000603       ISZ RDS.CM    ADJUST RETURN ADDRESS
24511 000000       LDB A         BOTH !!!!
24512 000603       JMP RDS.CM,I  RETURN
             *
             *
             *
             *
             *
             *
             *

24513 000576 CI.ID STB RCI.ID
24514 000515       LDB ASFLG     CHECK WHICH INTERFACE
24515 006003       SZB,RSS
24516 000576       ISZ RCI.ID    INTELLIGENT TYPE
24517 000576       JMP RCI.ID,I     RETURN
             *
24520 000600 TG.TB STB RTG.TB
24521 104600       .JLB CI.ID    IDENTIFY INTERFACE
24522 024513
24523 000600       JMP RTG.TB,I  ASCII, DO NOTHING
24524 003400       CCA           TRANSMIT TWO -1 BYTES TO SAY
                                    TRANSMIT BUFFER
24525 104600       .JLB OUT2C
24526 024566
24527 104600       .JLB CS.CM    TELL CARD TO TRANSMIT
24530 024577
24531 060400       OCT 60400
24532 000600       JMP RTG.TB,I  RETURN
```

```
24533 000577  TG.BF STB RTG.BF   TRANSMIT BUFFER & REQUEST NEW BUFFER
24534 104600        .JLB CI.ID    IDENTIFY INTERFACE
24535 024513
24536 000577        JMP RTG.BF,I  RETURN <NO FUNCTION>
24537 024436        JMP DS.TG     DS 1000
*
      024540  GETCH EQU *
24540 000542  IN1C  STB RGETC
24541 104600        ..JLB CI.ID   IDENTIFY THE INTERFACE
24542 024513
24543 024632        JMP AS.IN     ASCII
24544 024450        JMP DS.IN     DS 1000
*
24545 024050  GETCR ADA .N97     FOLD 6BIT ASCII BY
24546 002020        SSA           SUBTRACTING TO TEST
24547 024556        JMP GETCR2    IS < a SO DO NOTHING
24550 024045        ADA .N26      CHECK FOR Z
24551 002021        SSA,RSS       GREATER THAN Z???
24552 024055        ADA .D32      DONT FOLD THESE
24553 024061        ADA .D91      DO FOLDING
24554 000261  GETCR3 STA CHAR    SAVE CHARACTER
24555 000542        JMP RGETC,I   RETURN
*
24556 024060  GETCR2 ADA .D97    UNDO SUBTRACT
24557 024554         JMP GETCR3

      024560  PUTCH EQU *
24560 000541  OUT1C STB RPUTC
24561 024031        AND .B377       MASK OFF LOWER CHARACTER
24562 104600        .JLB CI.ID    IDENTIFY
24563 024513
24564 024640        JMP AS.OT
24565 024455        JMP DS.OT     DS 1000
*
24566 000606  OUT2C STB ROUT2C
24567 000260        STA TEMP      SAVE
24570 001727        ALF,ALF       POSITION
24571 104600        .JLB PUTCH    OUTPUT UPPER HALF
24572 024560
24573 000260        LDA TEMP      GET WORD AGAIN
24574 104600        .JLB PUTCH    OUTPUT LOWER HALF
24575 024560
24576 000606        JMP ROUT2C,I
*
```

```
03021*
03022  24577 000605  CS.CM STB RCS.CM    SAVE RETURN ADDRESS
03023  24600 000001        LDA B,I       GET COMMAND
03024  24601 104600        .JLB I.O      DO I/O
       24602 024607
03025  24603 000605        ISZ RCS.CM    ADJUST RETURN ADDRESS
03026  24604 000605        JMP RCS.CM,I  RETURN
03027*
03028  24605 000602  CS.WF STB RDS.FT    RETURN ADDRESS
03029  24606 024466        JMP DS.FT+3   WAIT FOR FLAG ONLY
03030*
03031*
03032  24607 000610  I.O   STB RI.O      RETURN ADDRESS
03033  24610 102630        OTA DATA      OTB ???
03034  24611 103730        STC DATA,C    START TRANSFER
03035  24612 102330  I.00  SFS DATA
03036  24613 024612        JMP I.00      KEEP TRYING
03037  24614 102530  I.01  LIA DATA      GET DATA
03038  24615 000610        JMP RI.O,I    RETURN
03039*

03041  24616 102532  AS.IZ LIA STATS   CLEAR BREAK BIT
03042  24617 024645        LDA TCCWO
03043  24620 102632        OTA STATS   SET TRANSMITT
03044  24621 102631        OTA CMND     (REMOVE DIAGNOSTIC)
03045  24622 002400        CLA
03046  24623 102630        OTA DATA      TRANSMIT A NUL
03047  24624 103730        STC DATA,C
03048  24625 102230        SFC DATA      WAIT FOR FLAG
03049  24626 000575        JMP RCI.IZ,I  RETURN
03050  24627 000000        ISZ A         TEST FOR TIME OUT
03051  24630 024625        JMP *-3
03052  24631 024005        JMP .PRSET,I      GO RESET COMPUTER
03053*
03054  24632 024646  AS.IN LDA TCCWI     INPUT CONTROL WORD
03055  24633 102631        OTA CMND
03056  24634 104600        .JLB I.O      DO I/O
       24635 024607
03057  24636 024031        AND .B377     MASK UPPER BYTE
03058  24637 024545        JMP GETCR     RETURN VIA GETCH
03059*
03060  24640 024645  AS.OT LDB TCCWO     OUTPUT CONTROL WORD
03061  24641 106631        OTB CMND
03062  24642 104600        .JLB I.O      DO I/O
       24643 024607
03063  24644 000541        JMP RPUTC,I  RETURN
03064*
03065  24645 001010  TCCWO OCT 001010
03066  24646 006412  TCCWI OCT 006412
```

```
*
*
*     ROUTINE TO INPUT HEX OR OCTAL DIGITS
*     CALLING SEQUENCE:
*             JLB*  GETN
*             P+1  NO DATA ENTERED; JUST CHARACTER NOT BS OR ?
*             P+2  NEW DATA ENTERED
*
*     IF P+1:  A = XXXX  B = LAST CHAR
*     IF P+2:  A = DATA  B = 0  IF LAST CHAR WAS CR
*     IF P+2:  A = DATA  B = LAST CHAR
*
24647 000544  GETN  STB RGETN
24650 002400        CLA           INITIALIZE
24651 000470        STA DIG1       DIGIT
24652 000471        STA DIG2        STORAGE
24653 000472        STA DIG3         LOCATIONS
24654 000473        STA DIG4
24655 000474        STA DIG5
24656 000475        STA DIG6
*
24657 024036        LDA .N6        SETUP FOR OCTAL
24660 000251        STA CNTR      INITIALIZE COUNTER
24661 000476        STA DIGS       AND SPARE
24662 024071        LDA ..DG1     DEF FOR OCTAL   ( DIG1 )
24663 000217        STA PNTR      SET POINTER
24664 000220        STA PNTRS      AND SPARE
*
24665 104600  GET1  .JLB GETCH   GET ONE CHARACTER
24666 024540
24667 105762        JLY ISDIG     IS DIGIT IN RANGE??
24670 027632
24671 024703        JMP EXIT?
24672 024043        ADA .N48       MAKE "0" THRU "9" OUT OF IT
24673 000251        LDB CNTR      IS THE DIGIT
24674 006003        SZB,RSS        ZERO ?
24675 024711        JMP IERR      YES, THE BUFFER IS FULL
24676 000217        STA PNTR,I    SAVE IT IN A DIGIT BUFFER
24677 000217        ISZ PNTR      UPDATE THE DIGIT POINTER
24700 006004        INB           UPDATE THE
24701 000251        STB CNTR       DIGIT COUNTER
24702 024665        JMP GET1       WE NEED MORE
*
24703 000261  EXIT? LDB CHAR      A FRESH COPY OF "IT"
24704 024000        CPB ..BKS     IS IT BACKSPACE ?
24705 024721        JMP BKUP       YES, GO PROCESS IT
24706 024003        CPB ..DEL
24707 024007        JMP .COMND,I
24710 024735        JMP EX1
```

```
*
24711 024012  IERR  LDA ..BEL         INDICATE ERROR AND
24712 104600        .JLB PUTS
24713 024245
24714 024720        LDA BKKMS
24715 104600        .JLB PUTS       WIPE OUT THE OFFENDING CHAR
24716 024245
24717 024665        JMP GET1          GIVE 'EM ANOTHER TRY
24720 025770  BKKMS DEF MES48
*
24721 000251  BKUP  LDA CNTR        IS THERE ANYTHING
24722 000476        CPA DIGS         LEFT ?
24723 024015        JMP CERR.P1,I     NO, END OF BUFFER
24724 024033        ADA .N1          MOVE DIGIT COUNTER
24725 000251        STA CNTR         BACK ONE
24726 000217        LDA PNTR        NOW BACKUP
24727 024033        ADA .N1           THE BUFFER
24730 000217        STA PNTR          POINTER
24731 025037        LDA BKSMES
24732 104600        .JLB PUTS       MAKE THE CHAR GO AWAY
24733 024245
24734 024665        JMP GET1        TRY AGAIN
*
24735 000251  EX1   LDA CNTR        ANY NEW
24736 000476        CPA DIGS         DATA ?
24737 000544        JMP RGETN,I      NO, COUNTER NOT INC'ED
*
24740 000544        ISZ RGETN       SET FOR "GOOD" RETURN
24741 000251        LDA CNTR        HOW MANY DIGITS
24742 003004        CMA,INA          HAVE BEEN
24743 000476        ADA DIGS          INPUT ?
24744 000251        STA CNTR
24745 000220        LDA PNTRS       INITIALIZE DIGIT
24746 000217        STA PNTR         POINTER
24747 002400        CLA             START WITH A CLEAN SLATE
24750 000217  EX2   ADA PNTR,I      ADD A DIGIT TO PARTIAL SUM
24751 000217        ISZ PNTR        POINT TO NEXT DIGIT
24752 000251        ISZ CNTR        DONE ?
24753 024755        JMP *+2          NO, MORE TO GO
24754 024761        JMP EX3          YEP
24755 000066        CLE,ELA
24756 000066        CLE,ELA         MULTIPLY BY 8
24757 000066        CLE,ELA
24760 024750        JMP EX2         PROCESS NEXT DIGIT
*
24761 000261  EX3   LDB CHAR        RETRIEVE LAST CHARACTER
24762 024020        CPB ...CR
24763 006400        CLB
24764 000544        JMP RGETN,I
```

```
*
*     GETS   INPUTS A STRING FROM THE TERMINAL.  IT PUTS IT IN
*     STRNG WITH THE LENGTH IN LSTR. THE FIRST CHAR SHOULD BE IN CHAR
*     40 CHARS IS MAXIMUM INPUT.
*     IT ALLOWS BACKSPACING.
*
  24765 000306  .SPTR DEF STRNG
*
  24766 000532  GETS  STB RGETS       SAVE RETURN
  24767 002400        CLA
  24770 000356        STA LSTR        LENGTH OF STRING
  24771 002004        INA
  24772 000357        STA GSLR      TEMP IS LEFT/RIGHT BYTE FLAG 1 LEFT
                                        0 RIGHT
  24773 024765        LDA .SPTR
  24774 001200        RAL              MAKE IT A BYTE ADDRESS
  24775 000360        STA STORE.POINTER       STRING POINTER
  24776 000261  GETSL LDA CHAR        GET CHARACTER
  24777 024000        CPA ..BKS       WAS BACKSPACE
  25000 025021        JMP GSBS        BACK OUT A CHARACTER
  25001 024020        CPA ...CR        CARRIAGE RETURN?
  25002 000532        JMP RGETS,I      YES, RETURN. DONE WITH STRING
  25003 024003        CPA ..DEL
  25004 024007        JMP .COMND,I
*
  25005 000360        LDB STORE.POINTER GET FLAG
  25006 105764        SBT
  25007 002400        CLA
  25010 105764        SBT              CLEAR NEXT WORD
  25011 000356        LDA LSTR
  25012 024057        CPA .D64
  25013 024015        JMP CERR.P1,I
  25014 000356        ISZ LSTR        ADD TO CHARACTER COUNTER
  25015 000360        ISZ STORE.POINTER  NEXT CHAR
  25016 104600  GET.  .JLB GETCH       GET ANOTHER CHARACTER
  25017 024540
  25020 024776        JMP GETSL       GO AROUND AGAIN
*
*
  25021 000356  GSBS  LDA LSTR
  25022 024024        CPA .B1          CANT BACKSPACE OVER % CHAR
  25023 024015        JMP CERR.P1,I   CANT BACKSPACE IF NO CHARACTERS
*
```

```
25024 024033         ADA .N1        DECREMENT NUMBER OF CHARS
25025 000356         STA LSTR
25026 000360         LDB STORE.POINTER
25027 024033         ADB .N1
25030 000360         STB STORE.POINTER  BACK UP A CHAR
25031 002400         CLA
25032 105764         SBT            CLEAR OUT THE LAST CHAR
25033 025037         LDA BKSMES
25034 104600         .JLB PUTS      OUTPUT SPACE BACKSPACE
25035 024245
25036 025016         JMP GET.        GET NEXT CHARACTER
25037 025766  BKSMES DEF MES47

*  DISTIBUTED SYSTEMS LOADER
*
*
*
25040 000571  DSLD   STB RDSLD
25041 024024         LDA .B1
25042 102601         OTA CPUST        SAY IN LOADER
25043 102702         STC 2           ALLOWED TO BREAK FROM DS LOADER
25044 104600         .JLB S.SC       SET SELECT CODE
25045 026330
25046 025140         JMP DSLER     ERROR IN SELECT CODE SPECIFIED
25047 024062         LDA .D310
25050 000265         STA LERR     ERROR 310 = TIME OUT AFTER SELF TEST
25051 104600         .JLB DS.WF      WAIT FOR DS SELF TEST
25052 024461
25053 025140         JMP DSLER     TIMED OUT
25054 000503         LDA TEMPO     CHECK IF THIS IS A DUMP
25055 024021         CPA ...W       READ OR WRITE?
25056 025225         JMP DSWR      IT'S A WRITE!!
25057 006400         CLB
25060 000516         STB EXLOAD    CLEAR EXTENDED LOAD FLAG
25061 024063  DSLD0  LDA .D312
25062 000265         STA LERR   ERROR 312 = TO AFTER DOWN LOAD REQUEST
25063 025312         LDA DSDNL     ASK FOR A DOWN LOAD
25064 104600         .JLB DS.FT     WAIT FOR COMPLETION OF REQUEST
25065 024463
25066 025140         JMP DSLER     TIMED OUT
25067 000274         LDB FILE      GET FILE NUMBER
25070 000265  DSLD1  ISZ LERR       ERROR 313 TO AFTER FILE NUMBER
25071 107630         OTB DATA,C      PASS IT TO THE CARD
25072 104600         .JLB DS.WF     WAIT FOR IT TO COMPLETE
25073 024461
25074 025140         JMP DSLER     TIMED OUT SO ERROR
25075 007400         CCB           SET TO READ A FRAME
25076 000517         STB P3.CT     (FRAME COUNT TO -1)
```

```
*                             READ IN ONE RECORD
25077 104600  DSRD   .JLB DS.GT      GET WORD COUNT
25100 025175
25101 101050         LSR 8          POSITION COUNT IN B
25102 007007         CMB,INB,SZB,RSS MAKE COUNT NEG. (DONE?)
25103 025131         JMP DSDUN      YES
25104 000521         STB DSCNT      SAVE COUNT
25105 104600         .JLB DS.GT      GET LOAD ADDRESS
25106 025175
25107 000522         STB DSADD      SAVE LOAD ADDRESS
25110 000523         STB DSCHK      AND START CHECKSUM
25111 104600  DSRDL  .JLB DS.GT      GET WORD REQUEST
25112 025175
25113                XSB1 '@DSADD'   STORE IT
25115 000522         ISZ DSADD
25116 000523         ADB DSCHK      ADD TO CHECKSUM
25117 000523         STB DSCHK
25120 000521         ISZ DSCNT      DONE WITH RECORD
25121 025111         JMP DSRDL      NO
25122 104600         .JLB DS.GT      GET CHECKSUM
25123 025175
25124 000523         CPB DSCHK      DOES CHECKSUM AGREE?
25125 025077         JMP DSRD       YES DO NEXT RECORD
25126 024064         LDA .D311
25127 000265         STA LERR       ERROR 311 = CHECKSUM ERROR
25130 025140         JMP DSLER      NO RETURN WITH ERROR

25131 104600  DSDUN  .JLB DS.GT     GET ADDRESS AS FLAG
25132 025175
25133 006003         SZB,RSS        GOOD OR BAD
25134 025142         JMP DSCONT     GOOD COMPLETED
25135 024065         LDA .D314
25136 000265         STA LERR       ERROR 314 = BAD TRANSFER
25137 000205         STB SAVEB      SAVE STATUS IN B REG
25140 000571  DSLER  ISZ RDSLD      INDICATE ERROR
25141 000571  DSEX   JMP RDSLD,I    RETURN
*
25142 000516  DSCONT ISZ EXLOAD      DONE?
25143 002001         RSS            NO
25144 025141         JMP DSEX       YES, ALL BLOCKS LOADED
*
25145 000516         LDA EXLOAD     FIRST TIME THROUGH??
25146 002020         SSA
25147 025165         JMP DSNXT      NO, GO GET NEXT FILE
```

```
03286*
03287  25150               XLA1 '0'
03288  25152               XLB1 '1'
03289  25154 002003        SZA,RSS
03290  25155 025141        JMP DSEX   IF LESS THAN ONE 32K CHUNK WE ARE DONE
03291  25156 006002        SZB          IF PARTIAL ADD ONE
03292  25157 002004        INA
03293  25160 003004        CMA,INA      MAKE IT NEGATIVE
03294  25161 000516        STA EXLOAD   SAVE COUNT
03295  25162 000516        ISZ EXLOAD   DONE???
03296  25163 025165        JMP DSNXT    NO, GO GET NEXT FILE
03297  25164 025141        JMP DSEX     ALL BLOCKS LOADED
03298*
03299  25165 000366  DSNXT ISZ MAP      NEXT 32K BLOCK
03300  25166 000366        LDA MAP
03301  25167 105762        JLY STMAP   SET MAP REGISTERS
       25170 027647
03302  25171 003400        CCA
03303  25172 001665        ELA,CLE,ERA  ELIMINATE BIT 15
03304  25173 000274        STA FILE     INDICATE CONTINUE LOAD
03305  25174 025061        JMP DSLDO    DO NEXT LOAD
03306*
03307  25175 000574  DS.GT STB RDS.GT
03308  25176 000517        ISZ P3.CT    TIME FOR NEW FRAME?
03309  25177 025216        JMP DS%GO    NO JUST READ A WORD
03310  25200 024067        LDA .D315
03311  25201 000265        STA LERR     ERROR 315 = TO AFTER BUFFER REQUEST
03312  25202 025250        LDA DSINR    GET BUFFER REQUEST
03313  25203 104600        .JLB DS.FT    GIVE IT TO CARD
       25204 024463
03314  25205 025140        JMP DSLER    TIMED OUT
03315  25206 102530        LIA DATA     NO GET BUFFER COUNT
03316  25207 000001        STA B
03317  25210 007004        CMB,INB      MAKE FRAME COUNT NEGATIVE
03318  25211 000517        STB P3.CT    SAVE IT
03319  25212 000265        ISZ LERR     ERROR 316 = TO AFTER COUNT ECHO
03320  25213 104600        .JLB DS.FT    TELL CARD HOW MUCH TO TRANSFER
       25214 024463
03321  25215 025140        JMP DSLER    TIMED OUT
03322  25216 024070  DS%GO LDA .D317    ERROR 317 = TO WAITING FOR DATA
03323  25217 000265        STA LERR
03324  25220 104600        .JLB DS.WF    WAIT FOR FLAG
       25221 024461
03325  25222 025140        JMP DSLER    IT DID SO ERROR
03326  25223 107530        LIB DATA,C   OK GET DATA
03327  25224 000574        JMP RDS.GT,I  RETURN
```

```
*     THIS ROUTINE DUMPS A MEMORY IMMAGE TO A REMOTE COMPUTER
*
25225 024066  DSWR  LDA .D320
25226 000265        STA LERR      ERROR 320 = VCP MODE TIME OUT
25227 024501        LDA DSVCP     TELL INTF TO GO INTO VCP MODE
25230 104600        .JLB DS.FT
25231 024463
25232 025140        JMP DSLER     TIMED OUT
25233 006400        CLB           SET STARTING ADDRESS
25234 000522        STB DSADD     SAVE IT
25235 002404  DSWR0 CLA,INA       1 PLUS RUBOUT
25236 104600        .JLB DS.B      OUTPUT 1 BYTE
25237 025302
25240 024031        LDA .B377      NOW RUBOUT
25241 104600        .JLB DS.B
25242 025302
25243 104600        .JLB DS.CM     TRANSMIT BUFFER
25244 024502
25245 060400        OCT 60400
25246 104600        .JLB DS.CM     ASK FOR BUFFER
25247 024502
25250 061400  DSINR OCT 61400
25251 104600        .JLB DS.CM     ASK FOR BYTE
25252 024502
25253 061000        OCT 61000
25254 024031        CPA .B377      CAN IT BE ACCEPTED?
25255 025277        JMP DSWEX     NO SO ERROR
25256 002003        SZA,RSS       DONE?
25257 025141        JMP DSEX      YES
25260 003004        CMA,INA       MAKE IT NEGATIVE
25261 000517        STA P3.CT     SAVE AS COUNTER
25262         DSWR1 XLA1 '@DSADD'  GET DATA
25264 000504        STA TEMP3     SAVE DATA
25265 001727        ALF,ALF
25266 104600        .JLB DS.B      TRANSFER CHARACTER
25267 025302
25270 000504        LDA TEMP3
25271 104600        .JLB DS.B
25272 025302
25273 000522        ISZ DSADD     MOVE ADDRESS UP ONE
25274 000517        ISZ P3.CT     DONE WITH THIS ONE?
25275 025262        JMP DSWR1     NO
25276 025235        JMP DSWR0     YES THEN MOVE TO NEXT TRANSFER
*
25277 000205  DSWEX STA SAVEB
25300 000265        ISZ LERR    ERROR 321 = CENTRAL WONT ACCEPT DATA
25301 025140        JMP DSLER     ERROR RETURN
```

```
25302 000573  DS.B  STB RDS.B
25303 024031        AND .B377
25304 024032        IOR .B60K       DS PUT BYTE REQUEST
25305 104600        .JLB DS.FT      WAIT FOR FLAG
25306 024463
25307 025140        JMP DSLER      TIMED OUT
25310 102530        LIA DATA       GET DATA
25311 000573        JMP RDS.B,I     RETURN
*
25312 161001  DSDNL OCT 161001     DOWN LOAD COMMAND
*

*
      MESSAGES AND WORDS OF WISDOM
*
      SUP
*
25313 006412  MES01 OCT 6412
25314 044120        ASC 8,HP A600/700 VCP
25315 020101
25316 033060
25317 030057
25320 033460
25321 030040
25322 053103
25323 050040
25324 006412        OCT 6412       THESE ARE "CARRIAGE RETURN"
25325 006412        OCT 6412        AND "LINE FEED" IN DISGUISE
25326 020040        ASC 9,  Type ? for help
25327 052171
25330 070145
25331 020077
25332 020146
25333 067562
25334 020150
25335 062554
25336 070040
25337 006412        OCT 6412
25340 006412  MES00 OCT 6412
25341 000000        OCT 0          NULL CHARACTER TO TERMINATE
*
25342 006412  MES02 OCT 6412
25343 053103  MES03 ASC 2,VCP>
25344 050076
25345 020021        OCT 020021     SPC AND DC1
25346 000000        OCT 000000
*
25347 003400  MES07 OCT 003400     BELL AND NULL
*
```

```
03401*
03402  25350 006412  MES09 OCT 6412
03403  25351 006412        OCT 6412
03404  25352 015463        OCT 015463    ESC 3    CLEAR ALL TABS
03405  25353 015446        OCT 015446    ESC &    MOVE THE
03406  25354 060464        OCT 060464    a   4     CURSOR TO
03407  25355 030103        OCT 030103    0   C      COLUMN 40
03408  25356 015461        OCT 015461    ESC 1    SET A TAB HERE
03409  25357 006412        OCT 6412
03410  25360 040454        ASC 9,A,B,X,Y,Q,Z,P,G,V
       25361 041054
       25362 054054
       25363 054454
       25364 050454
       25365 055054
       25366 050054
       25367 043454


       25370 053040
03411  25371 015511        OCT 015511    ESC I    TAB
03412  25372 051062        ASC 6,R20-R32 I/O
       25373 030055
       25374 051063
       25375 031040
       25376 044457
       25377 047440
03413  25400 006412        OCT 6412
03414  25401 042454        ASC 3,E,O,I
       25402 047454
       25403 044440
03415  25404 015511        OCT 015511
03416  25405 051103        ASC 3,RC CIR
       25406 020103
       25407 044522
03417  25410 006412        OCT 6412
03418  25411 046440        ASC 9,M Address  T data
       25412 040544
       25413 062162
       25414 062563
       25415 071440
       25416 020124
       25417 020144
       25420 060564
       25421 060440
03419  25422 015511        OCT 015511
03420  25423 051111        ASC 6,RI Int Mask
       25424 020111
```

```
      25425 067164
      25426 020115
      25427 060563
      25430 065440
03421 25431 006412          OCT 6412
03422 25432 046156          ASC 8,Lnn List memory
      25433 067040
      25434 046151
      25435 071564
      25436 020155
      25437 062555
      25440 067562
      25441 074440
03423 25442 015511          OCT 015511
03424 25443 051120          ASC 5,RP Parity
      25444 020120
      25445 060562
      25446 064564
      25447 074440
03425 25450 006412          OCT 6412
03426 25451 015511          OCT 015511
03427 25452 051123          ASC 6,RS Switches
      25453 020123
      25454 073551
      25455 072143

      25456 064145
      25457 071440
03428 25460 006412          OCT 6412
03429 25461 041517          ASC 4,COMMANDS
      25462 046515
      25463 040516
      25464 042123
03430 25465 015511          OCT 015511
03431 25466 051115          ASC 13,RMnn Map nn [Pnn Page nn]
      25467 067156
      25470 020115
      25471 060560
      25472 020156
      25473 067040
      25474 055520
      25475 067156
      25476 020120
      25477 060547
      25500 062440
      25501 067156
      25502 056440
```

```
03432  25503 006412          OCT 6412
03433  25504 022522          ASC 3,%R Run
       25505 020122
       25506 072556
03434  25507 015511          OCT 015511
03435  25510 051127          ASC 4,RW WMAP
       25511 020127
       25512 046501
       25513 050040
03436  25514 006412          OCT 6412
03437  25515 022505          ASC 5,%E Run P=2
       25516 020122
       25517 072556
       25520 020120
       25521 036462
03438  25522 015511          OCT 015511
03439  25523 051104          ASC 8,RD Diagnose Mode
       25524 020104
       25525 064541
       25526 063556
       25527 067563
       25530 062440
       25531 046557
       25532 062145
03440  25533 006412          OCT 6412
03441  25534 022524          ASC 4,%T Test
       25535 020124
       25536 062563
       25537 072040
03442  25540 015511          OCT 015511
03443  25541 051106          ASC 6,RF I/O flags
       25542 020111
       25543 027517

       25544 020146
       25545 066141
       25546 063563
03444  25547 006412          OCT 6412
03445  25550 022503          ASC 8,%C Clear memory
       25551 020103
       25552 066145
       25553 060562
       25554 020155
       25555 062555
       25556 067562
       25557 074440
```

```
03446  25560 006412         OCT 6412
03447  25561 022520         ASC 5,%P Preset
       25562 020120
       25563 071145
       25564 071545
       25565 072040
03448  25566 006412         OCT 6412
03449  25567 022530         ASC 9,%XDVFFBUSC[string]
       25570 042126
       25571 043106
       25572 041125
       25573 051503
       25574 055563
       25575 072162
       25576 064556
       25577 063535
03450  25600 006412         OCT 6412
03451  25601 054072         ASC 13,X: Boot, Load, Write, User
       25602 020102
       25603 067557
       25604 072054
       25605 020114
       25606 067541
       25607 062054
       25610 020127
       25611 071151
       25612 072145
       25613 026040
       25614 052563
       25615 062562
03452  25616 006412         OCT 6412
03453  25617 042126         ASC 15,DV: Cart. Tape, RoM, DisC, DS
       25620 035040
       25621 041541
       25622 071164
       25623 027040
       25624 052141
       25625 070145
       25626 026040
       25627 051157
       25630 046454
       25631 020104

       25632 064563
       25633 041454
       25634 020104
       25635 051440
```

```
03454  25636 006412         OCT 6412
03455  25637 043106         ASC 22,FF File, B Bus add., U Unit, SC Select code
       25640 020106
       25641 064554
       25642 062454
       25643 020102
       25644 020102
       25645 072563
       25646 020141
       25647 062144
       25650 027054
       25651 020125
       25652 020125
       25653 067151
       25654 072054
       25655 020123
       25656 041440
       25657 051545
       25660 066145
       25661 061564
       25662 020143
       25663 067544
       25664 062440
03456  25665 006412         OCT 6412
03457  25666 000000         OCT 0
03458*
03459  25667 020040   MES11 OCT 020040    2 SPACES
03460  25670 000000         OCT 0
03461*
03462  25671 006412   MES12 OCT 6412
03463  25672 020040         ASC 2,   P    3 SPACES AND P
       25673 020120
03464  25674 000000         OCT 0
03465*
03466  25675 040400   MES13 OCT 040400    A AND NULL
03467*
03468  25676 041000   MES14 OCT 041000    B AND NULL
03469*
03470  25677 020115   MES15 ASC 1, M      M AND NULL
03471  25700 000000         OCT 0
03472*
03473  25701 052000   MES16 OCT 052000    T AND NULL
03474  25702 072000         OCT 072000    t and null
03475*
03476  25703 020040   MES22 OCT 020040    SPC AND NULL
03477  25704 020000         OCT 020000
03478*
```

```
03479*
03480  25705 041514  MES32 ASC 6,CLEAR MEMORY
       25706 042501

       25707 051040
       25710 046505
       25711 046517
       25712 051131
03481  25713 020000        OCT 020000
03482*
03483  25714 050122  MES33 ASC 3,PRESET
       25715 042523
       25716 042524
03484  25717 020000        OCT 020000
03485*

03487  25720 020115  MES35 ASC 2, MAP
       25721 040520
03488  25722 000000        OCT 0
03489*
03490  25723 050101  MES36 ASC 2,PAGE
       25724 043505
03491  25725 000000        OCT 000
03492*
03493  25726 045502  MES37 ASC 5,KB MEMORY
       25727 020115
       25730 042515
       25731 047522
       25732 054440
03494  25733 020000        OCT 20000   SPACE NULL
03495*
03496  25734 006412  MES38 OCT 6412      CRLF
03497  25735 050101        ASC 4,PAR ERR
       25736 051040
       25737 042522
       25740 051040
03498  25741 000000        OCT 0
03499*
03500  25742 006412  MES41 OCT 6412   CRLF
03501  25743 050124        ASC 5,PTEST ERR
       25744 042523
       25745 052040
       25746 042522
       25747 051040
03502  25750 000000        OCT 000000
```

```
03503*
03504  25751 045502  MES43 ASC 3,KB ECA
       25752 020105
       25753 041501
03505  25754 006412        OCT 6412
03506  25755 000000        OCT 0
03507*
03508  25756 006412  MES44 OCT 6412   CRLF
03509  25757 020040        ASC 2,   R    SPACE SPACE
       25760 020122
03510  25761 000000        OCT 000000    R AND NULL
03511*
03512  25762 006412  MES46 OCT 6412   CRLF
03513  25763 020477        ASC 2,!?
       25764 020040
03514  25765 000000        OCT 000
03515*
03516  25766 020010  MES47 OCT 020010    SPACE BACKSPACE
03517  25767 000000        OCT 0
03518*
03519  25770 004040  MES48 OCT 004040    BACKSPACE  SPACE
03520  25771 004000        OCT 004000    BACKSPACE NULL
03521*
03522  25772 046104  MES62 ASC 4,LDER ERR
       25773 042522
       25774 020105
       25775 051122
03523  25776 020000        OCT 020000     SPACE AND NULL
03524*
03525        025777  EOP2 EQU *
03526  26000               ORG EPROM+6000B
```

```
             026000  P3    EQU *
*******************************************************************
*                                                                 *
*                                                                 *
*     LOADER ROUTINES                                             *
*                                                                 *
*                                                                 *
*******************************************************************
*
 26000 026004     JMP CTU
 26001 026364     JMP RMLDR
 26002 026531     JMP DCLDR
 26003 026524     JMP DCRLD
*
*  THESE JUMPS ARE FOR USER ROM CODE ENTRY TO THE LOADERS.
*  CALL THE LOADERS WITH A JLB INSTRUCTION. (JUMP AND LOAD B).
*
*  THE CALLING SEQUENCE IS
*
*        .JLB LOADER
*        JMP ERROR      ERROR RETURN ERROR NUMBER IN LERR
*  GOOD  ...            GOOD RETURN
*        ...
*
*
*  BEFORE CALLING THE LOADERS CERTAIN PARAMETERS MUST BE SET UP
*  SCETC  CONTAINS THE SELECT CODE, BUS ADDRESS, AND UNIT IN OCTAL
*  FILE   CONTAINS THE FILE NUMBER.
*  TEMPO  CONTAINS ASCII W IN THE LOW 8 BITS IF A WRITE IS TO BE DONE
*
*  THE LOADERS ASSUME THAT MAP ZERO IS SET UP PROPERLY FOR THE FIRST
*  32K OF THE LOAD, THAT THE DATA 1 MAP IS SET TO ZERO, AND THAT A
*  CLC 0,C INSTRUCTION HAS JUST BEEN EXECUTED. (THE I/O SYSTEM IS
*  QUIESCENT)
*
*  FOR THE CTU LOADER FILE 0 MEANS CURRENT LOCATION ON THE TAPE
*  FILE 1-N MEANS FIND THAT FILE ON TAPE FIRST, THEN LOAD.
*  UNIT IS 0 FOR LEFT TAPE AND 1 FOR RIGHT TAPE.
*  THE FORMAT OF THE TAPE IS ABSOLUTE BINARY. A ZERO LENGTH RECORD
*  WHICH IS NOT THE FIRST RECORD OF THE FILE INDICATES A SWITCH TO
*  THE NEXT 32K OF PHYSICAL MEMORY.
*  A WRITE TO TAPE WILL WRITE 4K WORDS WHERE THE BUS ADDRESS INDICATES
*  WHICH 4K TO WRITE.  IT WRITES IN ABSOLUTE BINARY, AND DOES
*  NOT WRITE A FILE MARK WHEN IT IS DONE.
*
*  FOR THE ROM LOADER TEMPO IS IGNORED. IT ALWAYS READS THE ROM.
*  THE FIRST TWO WORDS OF A ROM FILE ARE THE NUMBER OF 32K WORD CHUNKS
*  (BLOCK) AND THE REMAINDER AFTER THE LAST CHUNK (PARTIAL).
*  A FILE CAN HAVE MORE THAN ONE BLOCK IF IT STARTS ON A CARD BOUNDARY.
*  THE LOADER WILL GO TO THE NEXT CONSECUTIVE SELECT CODE WHEN IT RUNS
*  OUT OF THE CURRENT CARD.
*  THE ROM LOADER IGNORES THE BUS ADDRESS AND UNIT FIELDS OF SCETC.
*
*  THE DISC LOADER HAS TWO ENTRY POINTS, DCLDR AND DCRLD.
*  DCLDR LOADS A DISC FILE USING THE FILE NUMBER.
*  DCRLD LOADS A DISC FILE USING A STARTING TRACK AND SECTOR.
*
*  A DISC FILE HAS THE SAME FORMAT AS A ROM FILE.  THE FIRST TWO WORDS
*  ARE BLOCK AND PARTIAL. FOR A LOAD THESE INDICATE HOW MUCH MEMORY TO
*  LOAD & THE BLOCK GOES IN PHYSICAL MEMORY LOCATION 0 (NOT THE A REG)
*  AND THE PARTIAL GOES INTO MEMORY LOCATION 1.  FOR A WRITE
*  (TEMPO IS W)  THESE TWO WORDS OF MEMORY INDICATE HOW MUCH TO WRITE.
*
*  NOTE THAT THE LOADER ALWAYS TRANSFERS AT LEAST 32K WORDS.
*  BUS ADDRESS IS USED AS THE HPIB ADDRESS AND UNIT IS THE HEAD NUMBER
*  FOR 7906 DISC OR UNIT FOR FLOPPIES OR MINIFLOPPIES.
*
*  DCLDR MULTIPLIES THE FILE NUMBER BY 256 TO GET THE STARTING SECTOR
*  OF THE FILE.  FILE ZERO IS AT TRACK ZERO, SECTOR ZERO.
*
*  DCRLD EXPECTS THE GLOBAL REGISTER TO
*  BE SET TO THE SELECT CODE, THE HPIB ADDRESS IN SUBCH, THE UNIT
*  IN UNIT, THE SECTOR NUMBER OR VECTOR WORD ONE IN FILE,
*  THE CYLINDER OFFSET OR VECTOR WORD 2 IN CYLNDR.OFFSET AND
*  HEAD.CYLINDER, AND VECTOR WORD 3 IN SECTR.TRACK.
*  DCRLD DOES NOT LOOK AT SCETC.  THE VECTOR WORDS ARE FOR COMMAND SET
*  80 DISCS.


*
*     CARTRIDGE TAPE LOADER
*
*
 26004 000550  CTU   STB RCTU      RETURN ADDRESS
 26005 002400        CLA           CLEAR RECORD
 26006 000501        STA TEMP2      FLAG
 26007 104600        .JLB S.SC      SET SELECT CODE
 26010 026330
 26011 026157        JMP CTER        ERROR RETURN
*
 26012 027756        LDA ..D110    ERRORS IN 100 RANGE FOR CTU
 26013 000265        STA LERR
 26014 000267        ISZ UNIT       LEFT CTU IS UNIT 1
 26015 027677        LDA .PO
 26016 000267        IOR UNIT      MAKE P UNIT
 26017 000520        STA .PU
```

```
03623  26020 000274        LDB FILE       FILE ZERO ?
03624  26021 006003        SZB,RSS
03625  26022 026061        JMP CTLD        YES, SKIP FILE FIND
03626  26023 027676        LDA ECSAND    FIND THE FILE
03627  26024 104600        .JLB CTO.W    OCT 15446
       26025 026317
03628  26026 000520        LDA .PU       OCT 70060
03629  26027 104600        .JLB CTO.W
       26030 026317
03630  26031 000274        LDA FILE
03631  26032 006400        CLB           TO GET 1ST NUMBER
03632  26033 100400        DIV 012       DIVIDE BY 10
       26034 027720
03633  26035 000502        STB TEMP1
03634  26036 027705        IOR .U0       OCT 72460
03635  26037 104600        .JLB CTO.W
       26040 026317
03636  26041 000502        LDA TEMP1     DO SECOND NUMBER
03637  26042 027723        IOR 060       MAKE IT A NUMBER
03638  26043 104600        .JLB CTO.B
       26044 026310
03639  26045 027703        LDA .P2
03640  26046 104600        .JLB CTO.W
       26047 026317
03641  26050 027674        LDA CDC1      OCT 41421
03642  26051 104600        .JLB CTO.W
       26052 026317
03643  26053 104600        .JLB CTI.B    GET STATUS
       26054 026301
03644  26055 027710        CPA S         OK ?
03645  26056 026061        JMP CTLD      YES
03646  26057 000205        STA SAVEB     SAVE STATUS CODE IN B
03647  26060 026157        JMP CTER      ERROR
03648*
03649  26061 000503  CTLD  LDA TEMP0     CHECK IF READ OR WRITE
03650  26062 027711        CPA W
03651  26063 026167        JMP CT.DP      WANTS TO WRITE
03652*
03653  26064 027676  GTREC LDA ECSAND       ESC    &
03654  26065 104600        .JLB CTO.W
       26066 026317
03655  26067 000520        LDA .PU
03656  26070 104600        .JLB CTO.W
       26071 026317
03657  26072 027704        LDA .S2
03658  26073 104600        .JLB CTO.W
       26074 026317
03659  26075 027675        LDA RDC1          R     (DC1)
```

```
03660  26076 104600        .JLB CTO.W
       26077 026317
03661*
03662  26100 104600        .JLB CTI.W   GET FIRST WORD
       26101 026270
03663  26102 027673        CPA CTRS      DONE ?
03664  26103 026161        JMP DONE       YES, DONE WITH LOAD
03665*
03666  26104 104600        .JLB CTI.W   SKIP THE
       26105 026270
03667  26106 104600        .JLB CTI.B    COUNT WORDS
       26107 026301
03668*
03669  26110 027706        LDA .DC1      ASCII "DC1"
03670  26111 000501        STA TEMP2     INITIALIZE RECORD FLAG
03671  26112 104600        .JLB CTO.B      TELL TERMINAL TO TRANSMIT
       26113 026310
03672*
03673  26114 104600        .JLB CTI.B   GET FIRST BYTE ( RECORD LENGTH )
       26115 026301
03674  26116 003004        CMA,INA       MAKE IT NEGATIVE
03675  26117 000251        STA CNTR       INITIALIZE COUNTER
03676  26120 104600        .JLB CTI.B   SKIP UNUSED BYTE
       26121 026301
03677*
03678  26122 104600        .JLB CTI.W   GET LOAD ADDRESS
       26123 026270
03679  26124 000511        STA POINTER     INITIALIZE POINTER
03680  26125 000502        STA TEMP1      AND CHECKSUM
03681  26126 000251        LDA CNTR      CHECK FOR ZERO COUNT
03682  26127 002002        SZA
03683  26130 026141        JMP CTLDL     NONZERO, GO LOAD A RECORD
03684*
03685  26131 104600        .JLB CTI.W    SKIP THE CHECKSUM
       26132 026270
03686  26133 000366        LDA MAP
03687  26134 002004        INA
03688  26135 000366        STA MAP       NEXT MAP
03689  26136 105762        JLY STMAP    SET UP THE MAP
       26137 027647
03690  26140 026064        JMP GTREC    GET RECORDS FOR NEW MAP
03691*
03692  26141 104600  CTLDL .JLB CTI.W   GET A WORD OF DATA
       26142 026270
03693*
```

```
26143                 XSA1 '@POINTER'   STORE IT IN MAIN MEMORY
*
26145 000502          ADA TEMP1      ADD IT TO THE
26146 000502          STA TEMP1       CHECKSUM
26147 000511          ISZ POINTER      BUMP POINTER
26150 000251          ISZ CNTR       DONE WITH RECORD ?
26151 026141          JMP CTLDL       NO, GET ANOTHER WORD
*
26152 104600          .JLB CTI.W   GET CHECK SUM FROM THE TAPE
26153 026270
26154 000502          CPA TEMP1      DOES IT MATCH ?
26155 026064          JMP GTREC       YES, GET ANOTHER RECORD
*
26156 000265   .CKSM  ISZ LERR        ERROR 111 = CHECKSUM ERROR
26157 000550   CTER   ISZ RCTU        BUMP RETURN ADDRESS TO
26160 000550          JMP RCTU,I       ERROR RETURN
*
26161 000501   DONE   LDB TEMP2      WAS A RECORD
26162 006002          SZB             READ ? ( FLAG NO. 0 )
26163 000550   CTEX   JMP RCTU,I       YES, SPLIT
26164 000265          ISZ LERR
26165 000265          ISZ LERR       ERROR 112 = EOF ONLY
26166 026157          JMP CTER         OUTPUT ERROR MESSAGE
*
26167 000270   CT.DP  LDA SUBCH      WRITE TO CTU
26170 001727          ALF,ALF
26171 001700          ALF            SET ADDRESS
26172 000511          STA POINTER    ADDRESS POINTER
26173 027751          LDA M64
26174 000501          STA TEMP2      SET NUMBER OF BLOCKS
*
26175 000501   CTDPO  LDB TEMP2      END OF WRITE ?
26176 006003          SZB,RSS
26177 026163          JMP CTEX
26200 027676          LDA ECSAND     ESC &
26201 104600          .JLB CTO.W
26202 026317
26203 000520          LDA .PU
26204 104600          .JLB CTO.W
26205 026317
26206 027700          LDA .$D1
26207 104600          .JLB CTO.W
26210 026317
26211 027701          LDA ASC34      3  4
26212 104600          .JLB CTO.W
26213 026317
26214 027702          LDA WENQ       W  (ENQ)
```

```
  VIRTUAL CONTROL PANEL PAGE 3

03736  26215 104600        .JLB CTO.W
       26216 026317
03737  26217 104600        .JLB CTI.B
       26220 026301
03738  26221 027715        CPA 06        WAIT FOR ACKNOWLEDGEMENT
03739  26222 002001        RSS
03740  26223 026175        JMP CTDPO     TRY AGAIN
03741  26224 027751        LDA M64
03742  26225 000251        STA CNTR      SET FOR ONE BLOCK
03743  26226 003004        CMA,INA
03744  26227 001727        ALF,ALF       PUT POSITIVE COUNT IN UPPER HALF
03745  26230 104600        .JLB CTO.W
       26231 026317
03746  26232 000511        LDA POINTER   GET ADDRESS
03747  26233 000502        STA TEMP1     START CHECKSM
03748  26234 104600        .JLB CTO.W    SEND ADDRESS
       26235 026317
03749  26236         CTDPL XLA1 '@POINTER' GET WORD
03750  26240 000000        LDB A
03751  26241 000502        ADB TEMP1     ADD TO CHETEMP1
03752  26242 000502        STB TEMP1
03753  26243 104600        .JLB CTO.W
       26244 026317
03754  26245 000511        ISZ POINTER
03755  26246 000251        ISZ CNTR      DONE ?
03756  26247 026236        JMP CTDPL     NO
03757  26250 000502        LDA TEMP1
03758  26251 104600        .JLB CTO.W    OUTPUT CHECKSUM
       26252 026317
03759  26253 000501        ISZ TEMP2     MORE?
03760  26254 000000        NOP           NO
03761  26255 027706        LDA 021       DC1
03762  26256 104600        .JLB CTO.B
       26257 026310
03763  26260 104600        .JLB CTI.B    CHECK RESULTS
       26261 026301
03764  26262 027710        CPA S         OK ?
03765  26263 026175        JMP CTDPO    YES
03766  26264 000205        STA SAVEB     SAVE STATUS CODE RETURNED
03767  26265 027757        LDA ..D120
03768  26266 000265        STA LERR      LOADER ERROR 120 = CTU WRITE ERROR
03769  26267 026157        JMP CTER      ERROR
03770*
03771*********************************************************************
```

```
03772*
03773  26270 000551  CTI.W STB RTI.W     RETURN ADDRESS
03774  26271 104600        .JLB CTI.B   GET THE FIRST BYTE
       26272 026301
03775  26273 001727        ALF,ALF      PUT IN UPPER BYTE
03776  26274 000260        STA TEMP     SAVE IT
03777  26275 104600        .JLB CTI.B   NOW THE SECOND BYTE
       26276 026301
03778  26277 000260        ADA TEMP     BUILD A WORD
03779  26300 000551        JMP RTI.W,I
03780*
03781* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
03782*
03783  26301 000552  CTI.B STB RTI.B
03784  26302 027671        LDA .ICW     PUT ASIC INTO
03785  26303 102631        OTA CMND      INPUT MODE
03786  26304 104600        .JLB I.O      READ A BYTE
       26305 024607
03787  26306 027731        AND 0377     SAVE LOW BYTE ONLY
03788  26307 000552        JMP RTI.B,I
03789*
03790* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
03791*
03792  26310 000553  CTO.B STB RTO.B
03793  26311 027731        AND 0377     MASK OFF UPPER BYTE
03794  26312 027672        LDB .OCW     PUT ASIC INTO
03795  26313 106631        OTB CMND      OUTPUT MODE
03796  26314 104600        .JLB I.O     OUTPUT A BYTE
       26315 024607
03797  26316 000553        JMP RTO.B,I
03798*
03799* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
03800*
03801  26317 000554  CTO.W STB RTO.W
03802  26320 000260        STA TEMP     SAVE A COPY
03803  26321 001727        ALF,ALF      POSITION FIRST BYTE
03804  26322 104600        .JLB CTO.B   GO OUTPUT ONE BYTE
       26323 026310
03805  26324 000260        LDA TEMP     GET A FRESH COPY
03806  26325 104600        .JLB CTO.B   OUTPUT THE OTHER BYTE
       26326 026310
03807  26327 000554        JMP RTO.W,I
03808*
```

```
*     SET SELECT CODE AND OTHER USEFUL VALUES
*
 26330 000572  S.SC  STB RS.SC     SAVE RETURN ADDRESS
 26331 027713        LDA 02
 26332 000265        STA LERR      ERROR 2 = SELECT CODE < 20
 26333 000264        LDA SCETC     GET DEFAULT SELECT CODE
 26334 000204        STA SAVEA     GOING TO START WITH THIS VALUE
 26335 027723        AND 060       MUST BE OVER 20 OCT
 26336 002003        SZA,RSS
 26337 026363        JMP SCER      INTERNAL.ERROR
 26340 000265        ISZ LERR      ERROR 3 = I/O CARD NO RESPONSE
 26341 000264        LDA SCETC
 26342 027725        AND 077
 26343 103602        OTA GR,C      SET AND ENABLE GLOBAL REGISTER
 26344 002400        CLA
 26345 103502        LIA GR,C      CHECK FOR RESPONCE
 26346 002003        SZA,RSS
 26347 026363        JMP SCER      NO RESPONSE
*
 26350 000264        LDB SCETC
 26351 005700        BLF
 26352 005723        BLF,RBR       MOVE TO BUSS ADDRESS
 26353 000001        LDA B
 26354 027716        AND 07        MASK
 26355 000270        STA SUBCH
 26356 005723        BLF,RBR
 26357 000001        LDA B
 26360 027716        AND 07
 26361 000267        STA UNIT
*
*
 26362 000572        ISZ RS.SC     SKIP OVER ERROR RETURN
 26363 000572  SCER  JMP RS.SC,I    AND RETURN

*
*
*     ROM LOADER
*
*
 26364 000556  RMLDR STB RRMLD     SAVE RETURN ADDRESS
*
 26365 002404        CLA,INA
 26366 102601        OTA CPUST     SAY IN LOADER
 26367 104600        .JLB S.SC     SET SELECT CODE
 26370 026330
 26371 026475        JMP RMERR       ERROR RETURN
*
```

```
26372 000274         LDA FILE      GET "FILE"
26373 003000         CMA           MAKE IT NEGATIVE
26374 000274         STA FILE       AND SET FILE COUNTER
26375 027762         LDA ..D211
26376 000265         STA LERR      ERROR 211 = END OF PROGRAMS
26377 002400         CLA           START AT ADDRESS 0
26400 006400         CLB
26401 000260   ROM2  STB TEMP      SAVE CURRENT ADDRESS
26402 106631         OTB CMND      OUTPUT IT TO THE PROM CARD
26403 102730         STC DATA      READ ONE LOCATION
26404 007400         CCB           SETUP FOR END-OF-PROGRAM TEST
26405 106530         LIB DATA      FETCH ONE WORD
26406 006007         INB,SZB,RSS   CHECK FOR ALL ONES
26407 026475         JMP RMERR     ALL ONES FOUND, END OF PROGRAMS
26410 006020         SSB           SHOULD BE POSITIVE FOR NEW FORMAT
26411 026474         JMP RMERR2    ANOTHER ROM ERROR
26412 102530         LIA DATA      GET BLOCK COUNT AGAIN
26413                XSA1 '0'        STORE NUMBER OF BLOCKS
26415 102730         STC DATA      ON TO PARTIAL
26416 106530         LIB DATA      GET PARTIAL
26417                XSB1 '1'        STORE PARTIAL
26421 000266         STB PARTIAL   SAVE PARTIAL
26422 006002         SZB           PARTIAL?
26423 002004         INA           YES, ANOTHER BLOCK
26424 003004         CMA,INA       COMPLEMENT BLOCK
26425 000216         STA D1SV      SAVE BLOCK NUMBER
26426 002007         INA,SZA,RSS   ONLY ONE BLOCK?
26427 026433         JMP ROM4      YES, DONT CHECK FOR CARD BOUNDARY
26430 000260         LDA TEMP
26431 002002         SZA           MORE THAN ONE BLOCK MUST START ON
                                      CARD BOUNDARY
26432 026473         JMP RMERR3    ANOTHER ERROR
26433 000260   ROM4  ADB TEMP      STILL IN THE RUNNING,  BUILD  NEXT
                                      ADDRESS
26434 000274         ISZ FILE      IS THIS THE GOOD ONE ?
26435 026401         JMP ROM2       NO, KEEP TRYING
*
26436 027713         LDA O2        ALREADY READ FIRST 2 LOCATION
26437 027742   ROM5  LDB O100000   32K COUNT
26440 000216         ISZ D1SV      LAST BLOCK?
26441 002001         RSS
26442 000266         LDB PARTIAL   YES,USE PARTIAL AS COUNT
26443 007004         CMB,INB
26444 000000         ADB A         ADD A TO COUNT SINCE IS TWO FIRST
                                      TIME THROUGH
26445 000251         STB CNTR      STORE COUNT IN CNTR
26446 102730   ROM3  STC DATA      NEXT ADDRESS
26447 106530         LIB DATA      READ CONTENTS
```

```
26450                     XSB1 '@A'        STORE IN MAIN MEMORY
26452 002004              INA           NEXT ADDRESS
26453 000251              ISZ CNTR      COUNT THE WORD, DONE ?
26454 026446              JMP ROM3       NO, JUST TRY ONE MORE
26455 000216              LDA D1SV
26456 002003              SZA,RSS       DONE?
26457 000556              JMP RRMLD,I   GO SEE WHAT KIND OF LOAD THIS WAS
*
26460 000366              LDA MAP
26461 002004              INA
26462 000366              STA MAP
26463 105762              JLY STMAP    SET UP MAP FOR NEXT 32K
26464 027647
26465 102502              LIA 2
26466 002004              INA           ON TO NEXT CARD
26467 102602              OTA 2
26470 002400              CLA           START AT ADDRESS ZERO
26471 102631              OTA CMND      SET ADDRESS ON CARD
26472 026437              JMP ROM5      GO LOAD NEXT BUNCH
*
26473 000265  RMERR3 ISZ LERR      LOADER ERROR 113 = BIGGER THAN 32K
                                      MUST START ON CARD
*                       BOUNDARY
26474 000265  RMERR2 ISZ LERR      LOADER ERROR 112 = BAD FORMAT
26475 000556  RMERR ISZ RRMLD      BUMP RETURN ADDRESS
26476 000556         JMP RRMLD,I   ERROR RETURN

*
*
*      HPIB DISC LOADER
*
*      AUTO BOOT FROM DISC
*
26477 024072  ..MRBT DEF MRBT
26500 021466  .PTLER DEF PTLER
26501 177704  TRYNM  DEC -60       TRY 60 TIMES EVERY 2 SECONDS FOR
                                      2 MINUTES
26502 002027  .DCSC  OCT 2027      DEFAULT SELECT CODE FOR DISC
*
26503 026501  PTDC  LDA TRYNM         NUMBER OF RETRYS ON POWERUP
26504 000252        STA TRYCT
26505 102702        STC 2          BREAK ALLOWED DURING DISC LOAD
26506 026502  PTLP  LDA .DCSC   GET DEFAULT SELECT CODE FOR AUTO BOOT
26507 000264        STA SCETC      SAVE IT
26510 104600        .JLB DCLDR
26511 026531
26512 026477        JMP ..MRBT,I  GOOD BOOT, GO FINISH IT
26513 027754        LDB M1600      WAIT 2 SECONDS BEFORE RETRY
```

```
03944  26514 000245  PTWLP LDA TBGCNT    GET COUNT FOR 1.25 MS
03945  26515 002306        CCE,INA,SZA
03946  26516 026515        JMP *-1       WAIT  1.25 MS
03947  26517 000001        ISZ B
03948  26520 026514        JMP PTWLP
03949  26521 000252        ISZ TRYCT     ANOTHER RETRY?
03950  26522 026506        JMP PTLP
03951*
03952  26523 026500        JMP .PTLER,I   NO, DISC ERROR
03953*
03954*
03955  26524 000557  DCRLD  STB RDCLD    REENTER FOR DISK CALL BACK ENTRY
                                            TO LOADER
03956  26525 103102         CLF 2
03957  26526 002404         CLA,INA
03958  26527 102601         OTA CPUST
03959  26530 026546         JMP DISCO
03960*
03961*
03962*      NORMAL ENTRY AFTER %BOOT DISC OR %LOAD DISK
03963*
03964*
03965  26531 000557  DCLDR STB RDCLD     SAVE RETURN ADDRESS
03966*
03967  26532 002404        CLA,INA
03968  26533 102601        OTA CPUST
03969*
03970  26534 006400        CLB           ZERO SELECT CODE, UNIT, FILE, ETC
03971  26535 000273        STB CYLNDR.OFFSET
03972  26536 000275        STB HEAD.CYLINDER
03973  26537 000276        STB SECTR.TRACK
03974*
03975  26540 104600        .JLB S.SC     SET SELECT CODE
       26541 026330
03976  26542 026632        JMP DCER      ERROR RETURN
03977  26543 000274        LDA FILE
03978  26544 001727        ALF,ALF        MULT BY 256 TO GET SECTOR NUMBER
03979  26545 000274        STA FILE
03980*
03981*
03982  26546 002400  DISCO  CLA
03983  26547 000271         STA DISC.ID
03984  26550 104600        .JLB DC.IN    INITIALIZE
       26551 026634
03985*
03986  26552 002400        CLA              DO 64KB TRANSFER
03987  26553 104600        .JLB DC.RW          NOW READ/WRITE IT
       26554 027137
03988  26555 026632        JMP DCER            ERROR,CAN WE RETRY ?
```

```
*
26556                XLA1 '0'
26560                XLB1 '1'
26562 000266         STB PARTIAL   SAVE PARTIAL
26563 002003         SZA,RSS       ZERO BLOCKS?
26564 026633         JMP DCEX      YES, WE ARE DONE THEN
*
26565 006002         SZB           NONZERO PARTIAL??
26566 002004         INA           IF SO GET NEXT BLOCK
26567 027743         ADA  .M1    SUBTRACT ONE SO COUNT STARTS AT ZERO
26570 002003         SZA,RSS    ONE BLOCK NO PARTIAL OR PARTIAL ONLY?
26571 026633         JMP DCEX      YES, WE ARE DONE
*
26572 000216         STA D1SV      SAVE BLOCK NO.
*
26573 000366         LDA MAP       CURRENT MAP (ZERO)
26574 002104   DCLP  CLE,INA           BUMP TO NEXT BLOCK
26575 000366         STA MAP
26576 105762         JLY STMAP    SET IT UP
26577 027647
*
*     BUMP TO NEXT DISC ADDRESS (NEXT FILE?)
*
26600 000274         LDA FILE      ADD VALUE OF 32K TO FILE
26601 027732         ADA 0400      IE. 256 BLOCKS
26602 000274         STA FILE      SAVE AS SECTOR ADDRESS
26603 000275         LDB HEAD.CYLINDER
26604 002040         SEZ           RIPPLE THROUGH VECTOR
26605 006104         CLE,INB
26606 000275         STB HEAD.CYLINDER
26607 000276         LDB SECTR.TRACK
26610 002040         SEZ
26611 006004         INB
26612 000276         STB SECTR.TRACK
*
26613 104600         .JLB DC.IN    SET UP
26614 026634
*
26615 002400         CLA           DO 64KB TRANSFER
26616 000366         LDB MAP
26617 000216         CPB D1SV      LAST TRANSFER??
26620 000266         LDA PARTIAL   YES, ONLY LOAD PARTIAL
26621 001200         RAL
26622 003004         CMA,INA       COMPLEMENT FOR NEGATIVE COUNT
26623 104600         .JLB DC.RW    DO THE XFER
26624 027137
26625 026632         JMP DCER      BAD NEWS
*
```

```
04034  26626 000366        LDA MAP       GET MAP JUST USED
04035  26627 000216        CPA D1SV      DONE?
04036  26630 026633         JMP DCEX     YES
04037  26631 026574        JMP DCLP      KEEP ON TRUCKING
04038*
04039*
04040  26632 000557  DCER  ISZ RDCLD     SET FOR ERROR
04041  26633 000557  DCEX  JMP RDCLD,I    RETURN
04042*

04044*
04045*     INITIALIZE BUSS
04046*
04047  26634 000566  DC.IN STB RDCIN    SAVE RETURN ADDRESS
04048  26635 027760        LDA ..D411
04049  26636 000265        STA LERR     ERROR 411 = TO READING DISC TYPE
04050  26637 027751        LDA M64
04051  26640 000253        STA DCTO      SET TIME OUT TO 30 SECONDS
04052  26641 000271        LDA DISC.ID
04053  26642 027714        CPA 03        7906 ?
04054  26643 026646        JMP *+3       MUST INITIALIZE !!
04055*
04056  26644 002002        SZA           FIRST TIME ?
04057  26645 026710        JMP DC.IO     NO SKIP INITIALIZE
04058  26646 027744        LDA M2
04059  26647 104600        .JLB PHIN
       26650 027476
04060  26651 070200        OCT 070200    PHI ON-LINE
04061  26652 060063        OCT 060063    REN,IFC,WRITE,FLUSH FIFO
04062  26653 000245        LDA TBGCNT    SET TIME OUT. GET 1.25 MS TIME
04063  26654 002306        CCE,INA,SZA   THIS IS WORTH MACHINE INDEPENDENT
                                            IFC TIME
04064  26655 026654        JMP *-1         1.25 mSEC
04065  26656 000272        STA UNIT.HEAD   ; HEAD NUMBER ZERO
04066  26657 104600        .JLB PHIFL        FLUSH THE FIFO
       26660 027535
04067*
04068*     READ AND SET DISC TYPE
04069*      AND FILE POSITION
04070*
04071  26661 104600        .JLB PHI.L        TELL PHI TO LISTEN
       26662 027436
04072  26663 000537        OCT 000537          WITH A SECONDARY OF UNTALK
04073  26664 000270        LDA SUBCH       BUILD SECONDARY WITH HPIB ADDRESS
04074  26665 027667        IOR TLK
04075  26666 027670        IOR LSN
04076  26667 104600        .JLB HPIB         SEND IT TO THE CARD
       26670 027521
```

```
04077  26671 104600        .JLB PHI
       26672 027475
04078  26673 001002        OCT 001002
04079*
04080  26674 104600        .JLB PHI.I         GET DISC TYPE
       26675 027507
04081  26676 001727        ALF,ALF
04082  26677 000271        STA DISC.ID        SAVE UPPER BYTE
04083  26700 104600        .JLB PHI.I          GET SECOND BYTE
       26701 027507
04084  26702 000271        ADA DISC.ID        MERGE
04085  26703 000271        STA DISC.ID         DISC TYPE
04086*
04087*  DO A UNIVERSAL CLEAR AND READ STATUS
04088*
04089  26704 104600        .JLB PHI.TALK     PHI TALK
       26705 027435
04090  26706 000424        OCT 00424         UNIVERSAL DEVICE CLEAR
04091*
04092*    SEE IF DISC IS CS80 TYPE ??
04093*
04094  26707 000271        LDA DISC.ID       DISC TYPE
04095  26710 027737  DC.IO AND 01101          MASK
04096  26711 027163        CPA 01001         DISC PLUS LINUS ?
04097  26712 027107        JMP DC80          YEP
04098  26713 027672        CPA 01000         DISC ONLY?
04099  26714 027107        JMP DC80          YEP
04100  26715 027736        CPA 01100         LINUS ONLY ?
04101  26716 027121        JMP DC80.         YEP
04102*
04103*     NOPE CHECK FUTHER
04104*
04105  26717 000265        ISZ LERR      ERROR 412 = TO UDC OR READ STATUS
04106  26720 104600        .JLB PHI.TALK
       26721 027435
04107  26722 000550        OCT 00550         PHI TALK
04108*
04109  26723 104600        .JLB PHI
       26724 027475
04110  26725 000003        OCT 3             READ STATUS
04111*
04112  26726 000267        LDA UNIT
04113  26727 000271        LDB DISC.ID        CHECK FOR IDC
04114  26730 027714        CPB 03            IF IT IS THEN
04115  26731 002400         CLA             MAKE UNIT ZERO
04116  26732 027672        IOR BIT9        ADD BIT 9
04117  26733 104600        .JLB HPIB       PASS IT TO CARD
       26734 027521
```

```
04118  26735 104600        .JLB PHI.L    PHI LISTEN
       26736 027436
04119  26737 000550        OCT 00550
04120*
04121  26740 104600        .JLB PHI
       26741 027475
04122  26742 001003        OCT 1003      TRANSFER 3 BYTES
04123*
04124  26743 104600        .JLB PHI.I    GET BYTE
       26744 027507
04125  26745 000265        ISZ LERR   ERROR 413 = STATUS ERROR, STATUS IN B
04126  26746 000205        STA SAVEB     SAVE STATUS IN B
04127  26747 002002        SZA           CHECK FOR ERROR
04128  26750 026632        JMP DCER        ;ERROR 13  POSSIBLE RETRY !!
04129  26751 000265        ISZ LERR      ERROR 414 = TO DURING FILE MASK
                                             COMMAND
04130  26752 104600        .JLB PHI.I    SKIP NEXT BYTE
       26753 027507
04131  26754 104600        .JLB PHI.I    READ DISC TYPE
       26755 027507
04132  26756 001300        RAR           ELIMINATE BIT ZERO
04133  26757 027721        AND 017       USE 4 BITS FOR ID
04134  26760 000260        STA TEMP      SAVE.DISC.TYPE

04136*
04137*      USE DISC TYPE TO CONVERT DISC PARAMETERS
04138*
04139  26761 027766        LDB DCTYP     SET "DISC TYPE"
04140  26762 000217        STB PNTR       POINTER
04141  26763 000271        LDA DISC.ID   RETREIVE DISC TYPE
04142  26764 000626        CPA 0406      MSC 9800L?
04143  26765 027042        JMP DTYPE     YES
04144  26766 000217        ISZ PNTR
04145  26767 027730        CPA 0204      MINI-FLOPPY ?
04146  26770 027042        JMP DTYPE
04147  26771 027733        CPA 0404      SPARROW
04148  26772 027042        JMP DTYPE
04149  26773 000217        ISZ PNTR
04150  26774 027727        CPA 0201      88020 FLOPPY ?
04151  26775 027042        JMP DTYPE
04152  26776 000217        ISZ PNTR
04153  26777 027712        CPA 01        7910 FIXED DISC ?
04154  27000 027042        JMP DTYPE
04155  27001 000217        ISZ PNTR
04156  27002 027714        CPA 03        INTEGRATED DISC CONTROLLER?
04157  27003 027013        JMP DC.ID     YES
04158  27004 000205  DTYER STA SAVEB
04159  27005 027763        LDA ..D460    DISC NOT IDENTIFIED
```

```
04160  27006 000265        STA LERR      ERROR 460
04161  27007 000274        LDB FILE      CHECK IF FILE NO. IS ZERO
04162  27010 006002        SZB           IF SO THEN GO AHEAD
04163  27011 026632        JMP DCER        ; ELSE, ERROR
04164  27012 027026        JMP DCFM      USE CYLINDER MODE
04165*
04166  27013 000260  DC.ID LDA TEMP      SAVE.DISC.TYPE
04167  27014 027712        CPA 01        7920?
04168  27015 027026        JMP DCFM      YES, DO FILE MASK FIRST
04169  27016 000217        ISZ PNTR
04170  27017 027714        CPA 03        7925?
04171  27020 027026        JMP DCFM      YES, DO A FILE MASK FIRST
04172  27021 000217        ISZ PNTR
04173  27022 002002        SZA           7906?
04174  27023 027004        JMP DTYER      ;NOT IDENTIFIED
04175  27024 007400        CCB
04176  27025 000272        STB UNIT.HEAD   ;FLAG TO INDICATE (UNIT = HEAD)
04177*
04178  27026 104600  DCFM  .JLB PHI.TALK    PHI TALK
       27027 027435
04179  27030 000550        OCT 00550     SEND MASK TO 7906
04180*
04181  27031 104600        .JLB PHI
       27032 027475
04182  27033 000017        OCT 17        SET FILE MASK
04183  27034 027740        LDA 01005     ENABLE AUTO TRACK INCREMENT AND
                                            SPARING
04184  27035 000272        LDB UNIT.HEAD
04185  27036 006003        SZB,RSS       IS THIS A 7906?
04186  27037 027713        IOR 02        NO THEN CYLINDER MODE
04187  27040 104600        .JLB HPIB
       27041 027521

04189*
04190*    CONVERT FILE NO. TO CYLINDER.HEAD.SECTOR
04191*
04192  27042 104600  DTYPE .JLB DTPC     GO CALCULATE SEEK INFO FROM
       27043 027045                          FILE NUMBER
04193  27044 000566        JMP RDCIN,I
04194*
```

```
04195*
04196*
04197  27045 000567  DTPC  STB RDTPC
04198  27046 000217        LDA PNTR,I    GET NUMBER OF SECTORS PER TRACK
04199  27047 027731        AND 0377
04200  27050 003004        CMA,INA       MAKE IT NEGATIVE
04201  27051 000276        STA SECTR.TRACK ; SAVE IT
04202  27052 000217        LDA PNTR,I
04203  27053 001727        ALF,ALF       SET NUMBER OF HEADS PER CYLINDER
04204  27054 027721        AND 017
04205  27055 003004        CMA,INA
04206  27056 000275        STA HEAD.CYLINDER
04207  27057 002400        CLA
04208  27060 000274        LDB FILE      NOW GET NO SECTRS
04209  27061 000260        STB TEMP
04210  27062 000276        ADB SECTR.TRACK
04211  27063 006020        SSB
04212  27064 027067        JMP *+3
04213  27065 002004        INA
04214  27066 027061        JMP *-5
04215  27067 000260        LDB TEMP      REMAINDER IS THE SECTOR OFFSET
04216  27070 000276        STB SECTR.TRACK    ;SAVE IT
04217  27071 000000        LDB A         NOW GET NUMBER OF CYLINDERS
04218  27072 002400        CLA
04219  27073 000260        STB TEMP
04220  27074 000275        ADB HEAD.CYLINDER
04221  27075 006020        SSB
04222  27076 027101        JMP *+3
04223  27077 002004        INA
04224  27100 027073        JMP *-5
04225  27101 000275        STA HEAD.CYLINDER ;SAVE CYLINDER
04226  27102 000260        LDB TEMP      NOW ADD HEAD TO SECTOR WORD
04227  27103 005727        BLF,BLF
04228  27104 000276        ADB SECTR.TRACK
04229  27105 000276        STB SECTR.TRACK ;SAVE
04230  27106 000567        JMP RDTPC,I   ; NOW RETURN

04232*
04233*    SET SINGLE VECTOR
04234*
04235  27107 000274  DC80  LDB FILE
04236  27110 000277        STB VW1
04237  27111 000275        LDB HEAD.CYLINDER
04238  27112 000300        STB VW2
04239  27113 000276        LDB SECTR.TRACK
04240  27114 000301        STB VW3
04241  27115 000267        LDB UNIT
```

```
27116 000010        SLA                 IF ODD UNIT
27117 006011        SLB,RSS             AND LINUS TYPE
27120 027134        JMP DC80A           THEN USE LINUS NUMBERS
27121 000274  DC80. LDA FILE            LINUS TYPE
27122 000275        LDB HEAD.CYLINDER ; THEN DEVIDE BY
27123 101042        LSR 2             ; FOUR
27124 000277        STA VW1
27125 027753        LDA M350       160 SECOND TIME OUT FIRST TIME FOR
                                      LONG LINUS TAPES
27126 000253        STA DCTO
27127 000275        LDA HEAD.CYLINDER
27130 000276        LDB SECTR.TRACK
27131 101042        LSR 2
27132 000300        STA VW2
27133 000301        STB VW3
27134 002404  DC80A CLA,INA           ; INDICATE CS80 TYPE
27135 000272        STA UNIT.HEAD
27136 000566        JMP RDCIN,I       ; RETURN

*
*  SEEK READ/WRITE DSJ
*
27137 000570  DC.RW STB RDCRW
27140 102623        OTA 23B       OUTPUT COUNT
27141 027764        LDA ..D415
27142 000265        STA LERR      ERROR 415 = TO DURING SEEK COMMAND
*
*
*    CHECK IF READ OR WRITE
*
27143 027665        LDA DMACW     GET DMA CONTROL WORD
27144 000503        LDB TEMPO     GET LBW CHAR
27145 027711        CPB W         WRITE?
27146 027726        XOR 0200      YES, CLEAR BIT 7
27147 102621        OTA 21B       OUTPUT TO DMA
27150 002400        CLA           SET ADDRESS TO ZERO
27151 102622        OTA 22B
27152 000272        LDB UNIT.HEAD    ;CHECK FOR CS 80
27153 004010        SLB
27154 006020        SSB
27155 027252        JMP DSEEK     NO, NOT CS 80
```

```
04283*
04284*     FOR CS 80  DO THE SPECIAL DANCE
04285*
04286  27156 104600        .JLB PHI.L    PHI LISTEN
       27157 027436
04287  27160 000560        OCT 000560    SECONDARY (DSJ)
04288  27161 104600        .JLB PHI
       27162 027475
04289  27163 001001  01001 OCT 001001    COUNTED TRANSFER OF ONE
04290  27164 104600        .JLB PHI.I    GET IT BUT IGNORE IT
       27165 027507
04291  27166 104600        .JLB PHI.TALK
       27167 027435
04292  27170 000545        OCT 000545    COMMAND MESSAGE
04293  27171 000267        LDA UNIT
04294  27172 027735        IOR BIT5
04295  27173 104600        .JLB HPIB     SEND SET UNIT
       27174 027521
04296  27175 027722        LDA 020
04297  27176 104600        .JLB HPIB     SET ADDRESS
       27177 027521
04298  27200 000301        LDA VW3
04299  27201 104600        .JLB HPIBX
       27202 027557
04300  27203 000300        LDA VW2
04301  27204 104600        .JLB HPIBX
       27205 027557
04302  27206 000277        LDA VW1
04303  27207 104600        .JLB HPIBX
       27210 027557
04304  27211 027724        LDA 076       SET STATUS MASK
04305  27212 104600        .JLB HPIB
       27213 027521
04306  27214 027717        LDA 010       MESSAGE LENGTH
04307  27215 104600        .JLB HPIBX
       27216 027557
04308  27217 002400        CLA
04309  27220 104600        .JLB HPIBX
       27221 027557
04310  27222 002400        CLA
04311  27223 104600        .JLB HPIBX
       27224 027557
04312  27225 003400        CCA           OVERRUN
04313  27226 104600        .JLB HPIBX
       27227 027557
04314*
```

```
04315  27230 000265          ISZ LERR      ERROR 416 = TO DURING READ/WRITE
                                                COMMAND
04316  27231 000503          LDB TEMPO     GET RW CHAR
04317  27232 027711          CPB W         WRITE?
04318  27233 027243          JMP DC.01     GO WRITE
04319  27234 104600          .JLB PHI
       27235 027475
04320  27236 001000          OCT 001000    LOCATE > READ + EOI
04321*
04322  27237 104600          .JLB PHI.L
       27240 027436
04323  27241 000556          OCT 000556    <EXECUTE>
04324  27242 027331          JMP DS.01
04325*
04326  27243 104600  DC.01 .JLB PHI
       27244 027475
04327  27245 001002          OCT 001002    LOCATE > WRITE + EOI
04328*
04329  27246 104600          .JLB PHI.TALK
       27247 027435
04330  27250 000556          OCT 000556    <EXECUTE>
04331  27251 027351          JMP DCOMN

04333*      SEEK    FOR NON CS80
04334*
04335  27252 104600  DSEEK .JLB PHI.TALK       PHI TALK
       27253 027435
04336  27254 000550          OCT 000550
04337  27255 104600          .JLB PHI1
       27256 027475
04338  27257 000002          OCT 000002          SEEK
04339  27260 000272          LDB UNIT.HEAD
04340  27261 000267          LDA UNIT            GET UNIT
04341  27262 006020          SSB                 CHECK FOR UNIT HEAD SWAP
04342  27263 002400          CLA                 YEP SWAP
04343  27264 104600          .JLB HPIB           SEND TO THE CARD
       27265 027521
04344  27266 000275          LDA HEAD.CYLINDER ; SET UPPER CYLINDER
04345  27267 000273          ADA CYLNDR.OFFSET     CYLINDER OFFSET
04346  27270 104600          .JLB HPIBX
       27271 027557
04347  27272 000276          LDA SECTR.TRACK   ; SET HEAD
04348  27273 001727          ALF,ALF
04349  27274 027731          AND 0377
04350  27275 000272          LDB UNIT.HEAD     ; CHECK FOR UNIT HEAD SWAP
04351  27276 006020          SSB
04352  27277 000267          LDB UNIT
04353  27300 000001          IOR B
```

```
04354  27301 104600          .JLB HPIB
       27302 027521
04355  27303 000276          LDA SECTR.TRACK   ; SET SECTOR
04356  27304 027731          AND 0377
04357  27305 027672          IOR BIT9
04358  27306 104600          .JLB HPIB            SECTOR + EOI
       27307 027521

04360*
04361*     READ OR WRITE
04362*
04363  27310 000265          ISZ LERR       ERROR 416 = TO DURING READ/WRITE
                                                    COMMAND
04364  27311 104600          .JLB PHI.TALK      PHI TALK
       27312 027435
04365  27313 000550          OCT 000550
04366  27314 000503          LDB TEMPO      CHECK READ OR WRITE
04367  27315 027711          CPB W
04368  27316 027337          JMP DWRT        NOPE
04369  27317 104600          .JLB PHI1
       27320 027475
04370  27321 000005          OCT 000005     READ
04371  27322 000267          LDA UNIT       GET UNIT
04372  27323 027672          IOR BIT9        ADD EOI
04373  27324 104600          .JLB HPIB       SEND IT TO THE CARD
       27325 027521
04374  27326 104600          .JLB PHI.L      PHI LISTEN
       27327 027436
04375  27330 000540          OCT 000540     SECONDARY
04376  27331 027744    DS.01 LDA M2
04377  27332 104600          .JLB PHIN
       27333 027476
04378  27334 001400          OCT 001400     UNCOUNTED TRANSFER
04379  27335 060040          OCT 060040     TELL PHI TO INPUT
04380  27336 027351          JMP DCOMN
04381*
04382  27337 104600    DWRT  .JLB PHI1
       27340 027475
04383  27341 000010          OCT 000010     WRITE
04384  27342 000267          LDA UNIT       GET UNIT
04385  27343 027672          IOR BIT9        ADD EOI
04386  27344 104600          .JLB HPIB       OUTPUT TO THE CARD
       27345 027521
04387  27346 104600          .JLB PHI.TALK
       27347 027435
04388  27350 000540          OCT 000540     WRITE
```

```
*
*     COMMON DMA ROUTINE
*
27351 027666  DCOMN LDA CMDF      SET PHI FOR BYTE PACKED DMA
27352 000265        ISZ LERR      ERROR 417 = TO DURING DATA READ
27353 000503        LDB TEMPO     CHECK READ OR WRITE
27354 027711        CPB W         WRITE??
27355 001665        ELA,CLE,ERA    YES, CLEAR THE MSB
27356 102631        OTA CMND      SEND TO THE PHI
27357 103721        STC 21B,C    START DMA
27360 002400        CLA
27361 000253        LDB DCTO      LONG TIME OUT
*                              START
27362 101117  DC.NO RRR 15        DELAY WITHOUT MEMORY ACCESS
27363 100117        RRL 15
27364 102223        SFC 23B       DONE ?
27365 027400        JMP DC.N1       YEP
27366 101117        RRR 15
27367 100117        RRL 15
27370 000000        ISZ A        WAIT
27371 027362        JMP DC.NO    GO WAIT SOME MORE
27372 101117        RRR 15
27373 100117        RRL 15
27374 000001        ISZ B        TIMED OUT?
27375 027362        JMP DC.NO    NO. GO WAIT
27376 107721        CLC 21B,C    STOP DMA
27377 027434        JMP DCRWE     TIMED OUT
*
27400 107721  DC.N1 CLC 21B,C     KILL ANY ADDITIONAL DMA
27401 027765        LDA ..D420
27402 000265        STA LERR      ERROR 420 = PARITY ERROR
27403 102222        SFC 22B       CHECK FOR PARITY ERROR
27404 027434        JMP DCRWE      YEP, BAIL OUT
27405 000265        ISZ LERR      ERROR 421 = TO DURING PHI FLUSH
27406 027445        LDA UNL       GET UNLISTEN
27407 000503        LDB TEMPO     READ OR WRITE??
27410 027711        CPB W
27411 027415        JMP DC.N2     YES, FLUSH FIFO
27412 104600        .JLB PHIFL    FLUSH FIFO FOR READ
27413 027535
27414 027417        JMP .DSJ
*
27415 104600  DC.N2 .JLB HPIB      WRITE SO OUTPUT UNL
27416 027521
*
```

```
*    DSJ REQUEST
*
27417 000265  .DSJ  ISZ LERR      ERROR 422 = TIME OUT DURING DSJ
27420 104600        .JLB PHI.L     PHI LISTEN
27421 027436
27422 000560        OCT 000560    SECONDARY DSJ
27423 104600        .JLB PHI1
27424 027475
27425 001001        OCT 001001    COUNTED TRANSFER OF 1
27426 104600        .JLB PHI.I
27427 027507
27430 000205        STA SAVEB     SAVE DSJ ERROR CODE
27431 000265        ISZ LERR      ERROR 423 = BAD DSJ STATUS
27432 002003        SZA,RSS       WAS THERE AN ERROR ?
27433 000570        ISZ RDCRW     NO, TAKE GOOD EXIT
27434 000570  DCRWE JMP RDCRW,I   YES, ERROR RETURN

*
*
*    PHI SERVICE ROUTINES
*
*
*  PHI TALK AND PHI LISTEN SEND OUT TALK AND LISTEN COMMANDS
*  RESPECTIVELY TO THE SUBCHANNEL ADDRESED IN SUBCH.  THE WORD AFTER
*  THE JLB PHI... IS THE SECONDARY.
*
27435 002301  PHI.TALK CCE,RSS
27436 000040  PHI.L    CLE
27437 000560             STB RPHI?
27440 027745             LDA M3        SET COUNT
27441 104600        .JLB PHIN         "PHIN"
27442 027476
27443 031002        OCT 031002    PHI OUTPUT COMMAND
27444 000537        OCT 000537    UNT
27445 000477  UNL   OCT 000477    UNL
27446 002041        SEZ,RSS
27447 027461         JMP LISEN
27450 104600        .JLB PHI1
27451 027475
27452 000536        OCT 000536    CTLR LSN
27453 027752        LDA M100
27454 000000        ISZ A         KLUDGE TO MAKE MINIFLOPPY WORK
27455 027454        JMP *-1
27456 000270        LDA SUBCH     GET DISC ADDRESS
27457 027670        IOR LSN        MERGE LISTEN BIT
27460 027466        JMP PCOMN
*
```

```
27461 104600  LISEN .JLB PHI1
27462 027475
27463 000476        OCT 000476    CTLR LSN
27464 000270        LDA SUBCH     GET DISC ADDRESS
27465 027667        IOR TLK        MERGE TALK BIT
27466 104600  PCOMN .JLB HPIB      SEND TO CARD
27467 027521
27470 000560        LDA RPHI?,I   GET DATA
27471 104600        .JLB HPIB      SEND TO THE CARD
27472 027521
27473 000560        ISZ RPHI?     BUMP RETURN ADDRESS
27474 000560        JMP RPHI?,I   SPLIT
*
*  THIS ROUTINE UNDER ALL ITS MANY NAMES OUTPUTS ONE OR MORE WORDS TO
*  THE PHI CHIP.
*
      027475  PHI1  EQU *
27475 003400  PHI   CCA           SET FOR ONE CONTROL WORD
27476 000561  PHIN  STB RPHI      SAVE RETURN ADDRESS
27477 000223        STA CTR       SET CONTROL WORD COUNTER
27500 000561  PH    LDA RPHI,I    FETCH A WORD
27501 102630        OTA DATA      SEND IT TO THE CARD
27502 103730        STC DATA,C    PASS IT TO THE PHI
27503 000561        ISZ RPHI      MOVE POINTER
27504 000223        ISZ CTR       DONE ?
27505 027500        JMP PH         NO, TRY AGAIN
27506 000561        JMP RPHI,I     YES, BYE BYE
*
*  THIS ROUTINE INPUTS A WORD FROM THE PHI CHIP.
*
27507 000562  PHI.I STB RPHII     SAVE RETURN ADDRESS
27510 027546        LDA PIN       GET INPUT COMMAND
27511 104600        .JLB HPIB      SEND IT TO THE CARD
27512 027521
27513 104600        .JLB PHI1
27514 027475
27515 100000        OCT 100000    TELL CARD TO INPUT
27516 102530        LIA DATA      FETCH DATA
27517 027731        AND 0377      MASK OFF UPPER BYTE
27520 000562        JMP RPHII,I   RETURN
```

```
*
*  THIS ROUTINE OUTPUTS A WORD TO THE PHI CHIP AND WAITS FOR IT TO BE
*  SENT OUT TO THE BUS.  IF IT TAKES TOO LONG A TIMEOUT OCCURS
*  AND THE DISC LOAD IS TERMINATED WITHOUT FURTHER RETRIES
*
27521 000563  HPIB  STB RHPIB     SAVE RETURN ADDRESS
27522 102630        OTA DATA      OUTPUT DATA
27523 103730        STC DATA,C    START THE OUTPUT
27524 027755        LDB M5600     PROCESSOR INDEPENDENT TIMEOUT 7
                                     SECONDS FOR CS80
27525 000245  HPIBLP LDA TBGCNT
27526 002306        CCE,INA,SZA      WAIT 1.25 MS
27527 027526        JMP *-1
27530 102230        SFC DATA      FLAG 30 INDICATES FIFO EMPTY
27531 000563        JMP RHPIB,I   RETURN WHEN FLAG SET
27532 000001        ISZ B         DONE WITH TIME OUT?
27533 027525        JMP HPIBLP    NO, GO AROUND AGAIN
27534 026632        JMP DCER      ERROR, TIME OUT
*
*
27535 000564  PHIFL STB RPHIF     SAVE RETURN ADDRESS
27536 027666        LDA CMDF      ENABLE
27537 102631        OTA CMND       FLAG
27540 027746        LDA M4        SET CONTROL WORD COUNT
27541 104600        .JLB PHIN
27542 027476
27543 060043        OCT 060043    FLUSH OUTBOUND FIFO
27544 031002        OCT 031002    PHI OUTPUT COMMAND
27545 000537        OCT 000537    TELL DISC TO SHUT UP
27546 031004  PIN   OCT 031004    SET FLAG WHEN FIFO HAS DATA
27547 027666        LDA CMDF      ENABLE
27550 102631        OTA CMND       FLAG
27551 027752        LDA M100      SET MAXIMUM LOOP
27552 002006        INA,SZA
27553 102330        SFS DATA      ANY DATA ?
27554 000564        JMP RPHIF,I    NO, EXIT
27555 103730        STC DATA,C     YES, EMPTY IT
27556 027552        JMP *-4         TRY AGAIN
*
```

```
*     OUTPUT 2 BYTES TO THE HPIB CARD
*
 27557 000565  HPIBX STB RHPIBX    SAVE RETURN ADDRESS
 27560 000257        STA HPIT      SAVE DATA
 27561 001727        ALF,ALF
 27562 027731        AND 0377
 27563 104600        .JLB HPIB
 27564 027521
 27565 000257        LDA HPIT      GET LOW BYTE
 27566 027731        AND 0377
 27567 104600        .JLB HPIB
 27570 027521
 27571 000565        JMP RHPIBX,I
*
*
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*
*   SCNSC   SCANS THE SELECT CODE ETC FROM THE STRING INTO
*          THE A REGISTER. IT SKIPS IF THERE IS A NUMBER IN THE
*          STRING.
*          IT LEAVES STORE.POINTER WITH THE BYTE ADDRESS OF THE
*          FIRST CHAR AFTER THE NUMBER
*
 27572 000545  SCNSC STB RSCNSC       SAVE RETURN ADDRESS
 27573 000264        STA SCETC        SAVE DEFAULT VALUE
 27574 002400        CLA
 27575 000274        STA FILE      FILE ZERO IF NO NUMBER
 27576 000260        STA TEMP
 27577 027776        LDB ...SPTR         POINT AT START OF STRING
 27600 027713        ADB 02           POINT AT FIRST WORD OF SCETC
 27601 005200        RBL              MULT BY 2 TO MAKE BYTE ADDRESS
 27602 000360        STB STORE.POINTER  SAVE POINTER TO REST OF STRING
 27603 000360  SCNLP LDB STORE.POINTER  GET BYTE ADDRESS OF NEXT DIGIT
 27604 105763        LBT
 27605 105762        JLY ISDIG        IS IT A DIGIT??
 27606 027632
 27607 027620        JMP SCNDN        NO, DONE
 27610 027723        XOR 060          MAKE DIGIT BINARY
 27611 000260        ADA TEMP
 27612 000274        LDB FILE      SECOND WORD OF TWO WORD NUMBER
 27613 100043        LSL 3            MULT BY 8
 27614 000260        STA TEMP
 27615 000274        STB FILE      SAVE 2ND WORD
 27616 000360        ISZ STORE.POINTER  NEXT BYTE
 27617 027603        JMP SCNLP        GO DO NEXT BYTE
*
```

```
04590  27620 000260  SCNDN LDA TEMP
04591  27621 000274        LDB FILE      GET TWO WORD QUANT.
04592  27622 100041        LSL 1         SHIFT FILE NUMBER TO SECOND WORD
04593  27623 000274        STB FILE      SAVE IT
04594  27624 000207        STB SAVEX     PASS FILE NUMBER IN X REGISTER
04595  27625 006400        CLB
04596  27626 101044        LSR 4         PUT REST OF STUFF IN RIGHT PLACE
04597  27627 002002        SZA
04598  27630 000264        STA SCETC     SAVE IT
04599  27631 000545        JMP RSCNSC,I     RETURN
04600*     ISDIG CHECKS THE CHAR IN A AGAINST RANGE ZERO TO 7 AND
04601*     SKIPS IF IT IS IN RANGE.  DOES NOT CHANGE A OR B
04602*
04603  27632 000261  ISDIG STA CHAR          SAVE CHARACTER
04604  27633 027750        ADA M48           CHECK AGAINST ZERO
04605  27634 002020        SSA
04606  27635 027644        JMP ISDIGDN       NOT A DIGIT
04607*
04608  27636 027747        ADA M8            CHECK AGAINST 8
04609  27637 002021        SSA,RSS
04610  27640 027644        JMP ISDIGDN       NO GOOD
04611*
04612  27641 000261        LDA CHAR
04613  27642 105772        JPY 1             RETURN AND SKIP
       27643 000001
04614*
04615  27644 000261  ISDIGDN LDA CHAR        RESTORE A REG
04616  27645 105772        JPY 0             RETURN AND DONT SKIP
       27646 000000
04617*

04619*
04620  27647 001722  STMAP ALF,RAL      *32
04621  27650 105745        LDX .N40      32 ENTRIES
       27651 021754
04622  27652 027662        LDB .MBUF     BUFFER ADDRESS
04623  27653 000001        STA B,I       PUT ENTRY IN MAP BUFFER
04624  27654 002004        INA
04625  27655 006004        INB
04626  27656 105760        ISX           ISZ X REG.
04627  27657 027653        JMP *-4
04628  27660               LMAP
04629  27661 021706        DEF .DO       TO VALUES CONTAINED
04630  27662 000370  .MBUF DEF MPBUF
04631  27663 105772        JPY 0          RETURN
       27664 000000
04632*
```

```
*
*     CONSTANTS AND STUFF LIKE THAT
*
27665 060200  DMACW OCT 060200
27666 103004  CMDF  OCT 103004
27667 000500  TLK   OCT 000500
27670 000440  LSN   OCT 000440
27671 002400  .ICW  OCT 002400
27672 001000  .OCW  OCT 001000
27673 017015  CTRS  OCT 017015
27674 041421  CDC1  OCT 41421
27675 051021  RDC1  OCT 51021
27676 015446  ECSAND  OCT 15446
27677 070060  .PO   OCT 70060
27700 062061  .$D1  OCT 62061
27701 031464  ASC34 ASC 1,34
27702 053405  WENQ  OCT 53405
27703 070062  .P2   OCT 70062
27704 071462  .S2   OCT 71462
27705 072460  .U0   OCT 72460
27706 000021  .DC1  OCT 000021
27707 000122  R     OCT 122
27710 000123  S     OCT 123
27711 000127  W     OCT 127
27712 000001  01    OCT 000001
27713 000002  02    OCT 000002
27714 000003  03    OCT 000003
27715 000006  06    OCT 000006
27716 000007  07    OCT 000007
27717 000010  010   OCT 000010
27720 000012  012   OCT 000012
27721 000017  017   OCT 000017
27722 000020  020   OCT 000020
      027706  021   EQU .DC1
27723 000060  060   OCT 000060
27724 000076  076   OCT 000076
27725 000077  077   OCT 000077
27726 000200  0200  OCT 000200
27727 000201  0201  OCT 000201
27730 000204  0204  OCT 000204
27731 000377  0377  OCT 000377
27732 000400  0400  OCT 000400
27733 000404  0404  OCT 000404
27734 000406  0406  OCT 000406
27735 000040  BIT5  OCT 000040
      027672  BIT9  EQU .OCW
      027672  01000 EQU BIT9
27736 001100  01100 OCT 001100
```

```
 VIRTUAL CONTROL PANEL PAGE 3

04682  27737 001101  01101 OCT 001101
04683  27740 001005  01005 OCT 001005
04684  27741 007777  07777 OCT 007777
04685  27742 100000  0100000 OCT 100000
04686  27743 177777  .M1    DEC -1
04687  27744 177776  M2    DEC -2
04688  27745 177775  M3    DEC -3
04689  27746 177774  M4    DEC -4
04690  27747 177770  M8    DEC -8
04691  27750 177720  M48   DEC -48
04692  27751 177700  M64   DEC -64
04693  27752 177634  M100  DEC -100
04694  27753 177242  M350  DEC -350
04695  27754 174700  M1600 DEC -1600
04696  27755 165040  M5600 DEC -5600
04697  27756 000156  ..D110 DEC 110
04698  27757 000170  ..D120 DEC 120
04699  27760 000633  ..D411 DEC 411
04700  27761 000634  ..D412 DEC 412
04701  27762 000323  ..D211 DEC 211
04702  27763 000714  ..D460 DEC 460
04703  27764 000637  ..D415 DEC 415
04704  27765 000644  ..D420 DEC 420
04705  27766 027767  DCTYP DEF *+1     HEADS-CYL/SECTORS-TRACK
04706  27767 002037        OCT 002037       4/31  MSC9800L
04707  27770 001020        OCT 001020       2/16  MINI-FLOPPY
04708  27771 001036        OCT 001036       2/30  88010-20
04709  27772 001040        OCT 001040       2/32  7910
04710  27773 002460        OCT 002460       5/48  7920
04711  27774 004500        OCT 004500       9/64  7925
04712  27775 000460        OCT 000460       1/48  7906
04713*
04714  27776 000306  ...SPTR DEF STRNG
04715*
04716        027777  EOP3 EQU *
04717                      END


Lines where ORG command appeared:
42
71
264
268
273
1457
2551
3526
```

```
* - Volatile reference (store, jump, call...)

 .$D1 . . . . . . . . 4648:  3731
 .% . . . . . . . . . 2459:  1663
 ...CR  . . . . . . . 2573:  3158   3181
 ...SPTR  . . . . . . 4714:  4573
 ...W . . . . . . . . 2574:  3230
 ..?  . . . . . . . . 2575:  Symbol not referenced
 ..B  . . . . . . . . 2558:  2618   2650   2681
 ..B1777  . . . . . . 2465:  1504   2091
 ..B377 . . . . . . . 2464:  1472   1476   2044   2052   2250   2264
 ..B77777 . . . . . . 2466:  2322
 ..BEL  . . . . . . . 2567:  3116
 ..BKS  . . . . . . . 2557:  3110   3179
 ..D110 . . . . . . . 4697:  3617
 ..D120 . . . . . . . 4698:  3767
 ..D211 . . . . . . . 4701:  3858
 ..D411 . . . . . . . 4699:  4048
 ..D412 . . . . . . . 4700:  Symbol not referenced
 ..D415 . . . . . . . 4703:  4265
 ..D420 . . . . . . . 4704:  4419
 ..D460 . . . . . . . 4702:  4159
 ..DEL  . . . . . . . 2560:  2582*  3112   3183
 ..DG1  . . . . . . . 2615:  2781   3092
 ..ENT  . . . . . . . 1446:  1117
 ..MBUF . . . . . . . .714:   716
 ..MRBT . . . . . . . 3931:  3942*
 ..N40  . . . . . . . 2592:  2740   2751
 ..NEXT . . . . . . . 2572:  2652*
 ..RUN  . . . . . . . 2571:  2651*
 ..USER . . . . . . . 1168:  1145*
 .2 . . . . . . . . . 2461:  1927   1949
 .3 . . . . . . . . . 2462:  1929   1951
 .? . . . . . . . . . 2463:  1659
 .A . . . . . . . . . 2471:  1687
 .AGAIN . . . . . . . 2563:  Symbol not referenced
 .B . . . . . . . . . 2473:  1685   2258
 .B1  . . . . . . . . 2577:  2822   3200   3220
 .B10 . . . . . . . . 2499:  1578
 .B100  . . . . . . . 2502:  1979
 .B101  . . . . . . . 2581:  Symbol not referenced
 .B17 . . . . . . . . 2579:  Symbol not referenced
 .B177  . . . . . . . 2582:  Symbol not referenced
 .B20 . . . . . . . . 2498:  2409
 .B24 . . . . . . . . 2580:  Symbol not referenced
 .B37 . . . . . . . . 1402:  1350   1363
 .B377  . . . . . . . 2583:  2911   2921   3007   3057   3340   3348   3370
 .B60K  . . . . . . . 2584:  2924   3371
```

```
.B7  . . . . . . . . 2578:  2840
.BLR . . . . . . . . 2586:  Symbol not referenced
.BOOT? . . . . . . . 2503:  2414*  2425*  2433*  2439*
.BTERR . . . . . . . 2559:  2680*
.C . . . . . . . . . 2474:  1913   2274
.CIR . . . . . . . . 2010:  1914*
.CKSM  . . . . . . . 3706:  Symbol not referenced
.COMND . . . . . . . 2564:  3113*  3184*
.CR  . . . . . . . . 2457:  1657   1781   1844   1895   2118   2215
.CT  . . . . . . . . 2467:  2399
.CTLT  . . . . . . . 2458:  Symbol not referenced
.CTU . . . . . . . . 2409:  2400*
.D . . . . . . . . . 2475:  1767   1787   1945   2202   2220
.DO  . . . . . . . . 1390:   516    710    713    796    854    888   1051
                     1166   4629
.D10 . . . . . . . . 2601:  2786
.D16 . . . . . . . . 2602:  Symbol not referenced
.D2  . . . . . . . . 2600:  2624
.D310  . . . . . . . 2608:  3225
.D311  . . . . . . . 2610:  3266
.D312  . . . . . . . 2609:  3234
.D314  . . . . . . . 2611:  3273
.D315  . . . . . . . 2613:  3310
.D317  . . . . . . . 2614:  3322
.D32 . . . . . . . . 2603:  2765   2997
.D320  . . . . . . . 2612:  3331
.D40 . . . . . . . . 2604:  Symbol not referenced
.D64 . . . . . . . . 2605:  3191
.D91 . . . . . . . . 2607:  2998
.D97 . . . . . . . . 2606:  3002
.DC  . . . . . . . . 2469:  2403
.DC1 . . . . . . . . 4654:  3669   4667*
.DCSC  . . . . . . . 3934:  3939
.DEL . . . . . . . . 2472:  1893   1894   1919
.DIAG  . . . . . . . 2041:  1946*
.DIG1  . . . . . . . 2519:  Symbol not referenced
.DISC  . . . . . . . 2436:  2404*
.DISTS . . . . . . . 2430:  2406*
.DS  . . . . . . . . 2468:  2405
.DSJ . . . . . . . . 4435:  4429*
.DSSC  . . . . . . . 1387:  1191
.E . . . . . . . . . 2476:  1691   2268
.ENQ . . . . . . . . 2561:  2769
.ENQAK . . . . . . . 2757:  1464*  1526   1889*  2166*  2753*
.ENQAS . . . . . . . 2763:  2759*
.ENT . . . . . . . . 1538:  Symbol not referenced
.ENTI  . . . . . . . 2456:  1573
.EX  . . . . . . . . 2340:  2269*
```

```
.F . . . . . . . . . 2477:  1947
.FLAGS . . . . . . . 2018:  1948*
.G . . . . . . . . . 2478:  1683
.HELP  . . . . . . . 1734:  1660*
.I . . . . . . . . . 2479:  1695   1915
.ICW . . . . . . . . 4641:  3784
.L . . . . . . . . . 2480:  1671   2254
.LCH1  . . . . . . . 1892:  1877*  1879
.LCLP  . . . . . . . 1875:  1882*
.LDER  . . . . . . . 2427:  1487   2442
.LIA . . . . . . . . 2520:  1999   2045   2053
.LIST  . . . . . . . 1838:  1672*
.LLP . . . . . . . . 1861:  1890*
.LLP2  . . . . . . . 1865:  1872*
.LOAD  . . . . . . . 2395:  2255*  2257*  2259*
.M . . . . . . . . . 2481:  1665   1921
.M1  . . . . . . . . 4686:  3998
.MAPS  . . . . . . . 2127:  1922*
.MASK  . . . . . . . 2079:  1916*
.MBUF  . . . . . . . 4630:  4622
.MELO  . . . . . . . .546:   562*
.MRBT  . . . . . . . 1198:  1186*  1196*
.MREG  . . . . . . . 1802:  1666*
.N . . . . . . . . . 2482:  1769   1785   2200   2218
.N1  . . . . . . . . 2585:  2797   3126   3129   3203   3206
.N10 . . . . . . . . 2591:  Symbol not referenced
.N2  . . . . . . . . 1426:   369
.N20 . . . . . . . . 1427:   945   1248
.N26 . . . . . . . . 2595:  2995
.N27 . . . . . . . . 2594:  Symbol not referenced
.N32000  . . . . . . 2596:  2936
.N40 . . . . . . . . 1428:  1286   4621
.N48 . . . . . . . . 2593:  3099
.N5  . . . . . . . . 2587:  2826
.N6  . . . . . . . . 2588:  3089
.N65 . . . . . . . . 2597:  Symbol not referenced
.N7  . . . . . . . . 2589:  Symbol not referenced
.N8  . . . . . . . . 2590:  Symbol not referenced
.N91 . . . . . . . . 2599:  Symbol not referenced
.N97 . . . . . . . . 2598:  2992
.O . . . . . . . . . 2483:  1693
.OCW . . . . . . . . 4642:  3794   4679*
.OUTIT . . . . . . . 2014:  1722*  1825*  2039*  2061*  2075*  2083*  2092*
                     2096*  2100*
.P . . . . . . . . . 2484:  1689   1923   2144   2276
.P0  . . . . . . . . 4647:  3620
.P2  . . . . . . . . 4651:  3639
.PAR . . . . . . . . 2086:  1924*
```

```
.PRSET . . . . . . 2562:  2906*  3052*
.PSET  . . . . . . 1332:  1116*  1587*  2291*  2390*  2395*  2674*
.PTDC  . . . . . . 1189:  1173*
.PTLER . . . . . . 3932:  3952*
.PU  . . . . . . . .196:  3622*  3628   3655   3729
.Q . . . . . . . . 2485:  1677
.R . . . . . . . . 2486:  1625   1669   2270
.REGS  . . . . . . 1908:  1670*
.RENT  . . . . . . 1537:  1574*
.RM  . . . . . . . 2470:  2401
.RMSC  . . . . . . 1388:  1183
.ROM . . . . . . . 2419:  2402*
.RTRN  . . . . . . 2521:  1596
.RUN . . . . . . . 2348:  2271*  2358*
.S . . . . . . . . 2487:  1925
.S2  . . . . . . . 4652:  3657
.SPC1  . . . . . . 2565:  2816   2846
.SPC2  . . . . . . 2566:  Symbol not referenced
.SPTR  . . . . . . 3168:  2620   2638   2639   3175
.STAT  . . . . . . 2095:  1926*
.T . . . . . . . . 2488:  1661   1810   1827   2272
.T00 . . . . . . . 1747:  1779*  1795*  1828*
.T02 . . . . . . . 1751:  1631*
.T03 . . . . . . . 1772:  1766*  1768*  1770*
.T?  . . . . . . . 1781:  1764*  1775*
.TRAC  . . . . . . 2388:  2273*
.TREG  . . . . . . 1746:  1662*
.U . . . . . . . . 2489:  2252
.U0  . . . . . . . 4653:  3634
.USER  . . . . . . 2281:  2253*
.V . . . . . . . . 2490:  1667
.VIO . . . . . . . 2099:  1668*
.W . . . . . . . . 2491:  1627   1917   2256   2420
.WMP . . . . . . . 2065:  1918*
.X . . . . . . . . 2492:  1681
.Y . . . . . . . . 2493:  1679
.Z . . . . . . . . 2494:  1675
.ZERO  . . . . . . 2576:  2800   2823   2833   2841
A  . . . . . . . . . .0:   339    375*   376*   398    434    467    547*
                   548     549    552*   553    556*   557    584*   745
                   748     749    750    755    756    811    816    820
                   825     829    871    872    876    877    893    896
                   951     974   1016*  1070*  1249*  1291*  1352   1365
                  1572    1704   2135   2262   2398   2634   2641   2646
                  2659    2662   2665   2668   2672   2902*  2956   3050*
                  3750    3896   3900   4217   4409*  4469*
AEAUS  . . . . . . 1437:   468    528   1438
AGAIN  . . . . . . 1585:  1544*  2563
```

```
ALT0 . . . . . . . . 1432:  342   343   350   351   355   362   501
                     1299  1431
ALT1 . . . . . . . . 1433:  332   336   338   346   354   358   359
                      390   497   503
AMESS  . . . . . . . 2547: 1614
AS.IN  . . . . . . . 3054: 2989*
AS.IZ  . . . . . . . 3041: 2899*
AS.OT  . . . . . . . 3060: 3009*
ASC34  . . . . . . . 4649: 3733
ASFLG  . . . . . . . .192:  688*  2967
ASR.0  . . . . . . . 1440:  477
ASR.1  . . . . . . . 1441:  489
B  . . . . . . . . . . .0:  310   334   337*  389   421   460   718
                     1054  1251* 1281  1289* 1307  1316  2191  2942*
                     2951  3023  3316* 3831  3835  3947* 4353  4413*
                     4524* 4623*
B1 . . . . . . . . . 1391:  420*  513   773
B100 . . . . . . . . 1407:  544   586   673   917   983  1019  1125
B1000  . . . . . . . 1413:  734   916  1123
B100000  . . . . . . 1422: 1334
B100024  . . . . . . 1421: Symbol not referenced
B100300  . . . . . . 1409: 1231  1236
B100340  . . . . . . 1410:  703   846
B100K  . . . . . . . 1419:  379   386
B11  . . . . . . . . 1398:  582
B1400  . . . . . . . 1414:  765
B17  . . . . . . . . 1399:  970
B170360  . . . . . . 1411:  605  1224
B177700  . . . . . . 1423:  931
B177777  . . . . . . 1424: Symbol not referenced
B2 . . . . . . . . . 1392:  423   666   990  1084  1144
B20  . . . . . . . . 1400:  664   915  1017  1018  1124
B200 . . . . . . . . 1408: Symbol not referenced
B24  . . . . . . . . 1401:  349
B3 . . . . . . . . . 1393:  296   788   894   959  1041  1175
B3004  . . . . . . . 1415: Symbol not referenced
B37  . . . . . . . . 2500: 1564  1857  2071  2334
B377 . . . . . . . . 1412: 1246
B4 . . . . . . . . . 1394:  961  1141
B40  . . . . . . . . 1403:  723
B5 . . . . . . . . . 1395: 1058  1178  1219
B6 . . . . . . . . . 1396:  644
B6412  . . . . . . . 1416:  494
B7 . . . . . . . . . 1397:  541  1028  1133  1172  1213  1262
B76K . . . . . . . . 1418:  492   502
B77  . . . . . . . . 1404: 1013  1068
B7777  . . . . . . . 1417:  499
B77777 . . . . . . . 1420:  809   818   827   868
```

```
BASE . . . . . . . . .129:  Symbol not referenced
BEAUS  . . . . . . . 1439:   466
BEXEX  . . . . . . . 2351:  2571
BFLAG  . . . . . . . .161:  Symbol not referenced
BIT15  . . . . . . . 2501:  1558
BIT5 . . . . . . . . 4678:  4294
BIT7 . . . . . . . . 1429:  1257
BIT9 . . . . . . . . 4679:  4116   4357   4372   4385   4680*
BKKMS  . . . . . . . 3121:  3118
BKSMES . . . . . . . 3213:  3131   3210
BKUP . . . . . . . . 3123:  3111*
BLBT . . . . . . . . 1438:   521
BMESS  . . . . . . . 2548:  1619
BOOT?  . . . . . . . 2624:  2503
BTERR  . . . . . . . 2441:  2415*  2426*  2434*  2559
BUFF . . . . . . . . 2518:  1701
CDC1 . . . . . . . . 4644:  3641
CERR . . . . . . . . 1726:  1697*  1710*  1771*  1789*  1812*  1846*  1977*
                     2109*  2120*  2128*  2131*  2137*  2146*  2174*  2178*
                     2182*  2204*  2222*  2266*  2278*  2570
CERR.P1  . . . . . . 2570:  3125*  3192*  3201*
CERR2  . . . . . . . 1725:  1943*  1953*  1958*  1966*  1970*  2407*  2421*
CHAR . . . . . . . . .132:  2999*  3109   3157   3178   4603*  4612   4615
CHBR . . . . . . . . 1283:  1293*  1297*  1309*  1318*  1322*  1326*
CHKIO  . . . . . . . 1277:  1007*
CHKSUM . . . . . . . .286:  Symbol not referenced
CI.ID  . . . . . . . 2966:  2362*  2758*  2898*  2973*  2982*  2988*  3008
CI.IZ  . . . . . . . 2891:  1463*  1599*  1633*  1725*  1984*  2004*  2037*
                     2057*  2292*  2357   2441*
CLDN . . . . . . . . 2329:  2315*
CLMES  . . . . . . . 2536:  2302
CLRM . . . . . . . . 2302:  2275*
CLRM1  . . . . . . . 2313:  2326*
CMDF . . . . . . . . 4638:  4393   4530   4538
CMND . . . . . . . . . 80:  3044*  3055*  3061*  3785*  3795*  3863*  3916*
                     4398*  4531*  4539*
CNTR . . . . . . . . .124:  1874*  1881*  2784*  2789*  2793   2795*  2802*
                     2827*  2844*  3090*  3100   3106*  3123   3127*  3135
                     3140   3143*  3149*  3675*  3681   3699*  3742*  3755*
                     3897*  3902*
CNTRL  . . . . . . . 2246:  1664*
COM01  . . . . . . . 2118:  1708*  1829*  1975*  2107*
COM1 . . . . . . . . 1655:  Symbol not referenced
COMN . . . . . . . . 2103:  2011*  2066   2080*
COMN1  . . . . . . . 1649:  1643*
COMND  . . . . . . . 1639:  1658*  1728*  1736*  1761*  1782*  1888*  1920*
                     1996*  2015*  2068*  2119*  2170*  2213*  2216*  2337*
                     2446*  2564
```

```
CORCNT . . . . . . . .123:   834*
CPUST  . . . . . . . . 50:   294*   298*   543*   606*   704*   932*  1131*
                      1165*  1214*  1247*  1260*  1576*  1640*  2412*  2704*
                      2763*  3221*  3851*  3958*  3968*
CRLF . . . . . . . . 2527:  1485   1715   1747   1818   1847   1859   1884
                      2158   2227   2247
CS.CM  . . . . . . . 3022:  2907*  2913   2915*  2919*  2977*
CS.WF  . . . . . . . 3028:  Symbol not referenced
CT.DP  . . . . . . . 3717:  3651*
CTDPO  . . . . . . . 3724:  3740*  3765*
CTDPL  . . . . . . . 3749:  3756*
CTER . . . . . . . . 3707:  3615*  3647*  3715*  3769*
CTEX . . . . . . . . 3712:  3726*
CTI.B  . . . . . . . 3783:  3643   3667*  3673*  3676*  3737   3763   3774*
                      3777*
CTI.W  . . . . . . . 3773:  3662   3666*  3678*  3685*  3692*  3702*
CTLD . . . . . . . . 3649:  3625*  3645*
CTLDL  . . . . . . . 3692:  3683*  3700*
CTO.B  . . . . . . . 3792:  3638   3671*  3762   3804*  3806
CTO.W  . . . . . . . 3801:  3627   3629   3635   3640   3642   3654   3656
                      3658   3660   3728   3730   3732   3734   3736   3745*
                      3748*  3753*  3758
CTR  . . . . . . . . .102:  4491*  4496*
CTRS . . . . . . . . 4643:  3663
CTU  . . . . . . . . 3611:  2413*  3537*
CYLNDR.OFFSET  . . . .143:  2669*  3971*  4345
D1SV . . . . . . . . . 97:  3880*  3892*  3904   4002*  4027   4035
D2 . . . . . . . . . 2515:  1912   2049   2344
D32  . . . . . . . . 2517:  1899
D7 . . . . . . . . . 2516:  1639   1990
DALTO  . . . . . . . 1431:  Symbol not referenced
DATA . . . . . . . . . 78:  1287*  1294*  1317*  1319*  1320*  1321*  1323*
                      1324*  1325*  2934*  2935*  2937*  2954*  3033*  3034*
                      3035*  3037*  3046*  3047*  3048*  3241*  3315*  3326*
                      3374*  3864*  3866*  3871*  3873*  3874*  3898*  3899*
                      4493*  4494*  4507*  4516*  4517*  4522*  4542*  4544*
DC.01  . . . . . . . 4326:  4318*
DC.ID  . . . . . . . 4166:  4157*
DC.IN  . . . . . . . 4047:  3984*  4023*
DC.IO  . . . . . . . 4095:  4057*
DC.NO  . . . . . . . 4403:  4410*  4414*
DC.N1  . . . . . . . 4418:  4406*
DC.N2  . . . . . . . 4431:  4427*
DC.RW  . . . . . . . 4263:  3987*  4031*
DC80 . . . . . . . . 4235:  4097*  4099*
DC80.  . . . . . . . 4245:  4101*
DC80A  . . . . . . . 4256:  4244*
DCER . . . . . . . . 4040:  3976*  3988*  4032*  4128*  4163*  4526*
```

```
DCEX . . . . . . . . 4041:  3994*  4000*  4036*
DCFM . . . . . . . . 4178:  4164*  4168*  4171*
DCLDR  . . . . . . . 3965:  2438   3539*  3941*
DCLP . . . . . . . . 4005:  4037*
DCOMN  . . . . . . . 4393:  4331*  4380*
DCRLD  . . . . . . . 3955:  2677*  3540*
DCRWE  . . . . . . . 4445:  4416*  4422*
DCSC . . . . . . . . 2495:  2436
DCTO . . . . . . . . .126:  4051*  4250*  4401
DCTYP  . . . . . . . 4705:  4139
DFLAG  . . . . . . . .162:  1608*  1700*  1702   1711   1746*  1759   1778*
                     1802*  1838*  1908*  2819
DIG1 . . . . . . . . .169:  2519   2615   3082*
DIG2 . . . . . . . . .170:  3083*
DIG3 . . . . . . . . .171:  3084*
DIG4 . . . . . . . . .172:  3085*
DIG5 . . . . . . . . .173:  3086*
DIG6 . . . . . . . . .174:  3087*
DIGS . . . . . . . . .175:  3091*  3124   3136   3142
DISC.ID  . . . . . . .141:  3983*  4052   4082*  4084   4085*  4094   4113
                     4141
DISCO  . . . . . . . 3982:  3959*
DISPLAY  . . . . . . .119:  1137   1242*  1244   1255   1465   1471   1474
DMA  . . . . . . . . 1075:  1023   1025   1027   1059   1077
DMACF  . . . . . . . 1077:  1021   1057
DMACW  . . . . . . . 4637:  4271
DMAQD  . . . . . . . 1076:  1024   1026   1034   1038
DONE . . . . . . . . 3710:  3664*
DPNTR  . . . . . . . .160:  1703*  1714*  1721
DS%GO  . . . . . . . 3322:  3309*
DS.01  . . . . . . . 4376:  4324*
DS.B . . . . . . . . 3369:  3339*  3341   3357*  3359
DS.CM  . . . . . . . 2950:  3342   3344*  3346*
DS.FT  . . . . . . . 2933:  2365   2905   2931*  2952   3029*  3237   3313
                     3320*  3334   3372
DS.GT  . . . . . . . 3307:  3248*  3253*  3256*  3263*  3270*
DS.IN  . . . . . . . 2919:  2990*
DS.IZ  . . . . . . . 2901:  Symbol not referenced
DS.OT  . . . . . . . 2924:  3010*
DS.TG  . . . . . . . 2911:  2984*
DS.WF  . . . . . . . 2930:  3227*  3242*  3324*
DSADD  . . . . . . . .198:  3254*  3257   3258*  3337*  3354   3360*
DSCER  . . . . . . . 1203:  1200
DSCHK  . . . . . . . .199:  3255*  3259   3260*  3264
DSCNT  . . . . . . . .197:  3252*  3261*
DSCONT . . . . . . . 3279:  3272*
DSDNL  . . . . . . . 3377:  3236
DSDUN  . . . . . . . 3270:  3251*
```

```
DSEEK  . . . . . . 4335:  4281*
DSEX . . . . . . . 3277:  3281*  3290*  3297*  3351*
DSINR  . . . . . . 3345:  3312
DSLD . . . . . . . 3219:  1193*  2432
DSLD0  . . . . . . 3234:  3305*
DSLD1  . . . . . . 3240:  Symbol not referenced
DSLER  . . . . . . 3276:  2953*  3224*  3228*  3238*  3243*  3268*  3314*
                   3321*  3325*  3335*  3367*  3373*
DSNXT  . . . . . . 3299:  3285*  3296*
DSPLY  . . . . . . 1242:   847*  1201*  1225*  1232*
DSRD . . . . . . . 3248:  3265*
DSRDL  . . . . . . 3256:  3262*
DSSC . . . . . . . 2496:  2430
DSVCP  . . . . . . 2948:  3333
DSWEX  . . . . . . 3365:  3349*
DSWR . . . . . . . 3331:  3231*
DSWR0  . . . . . . 3338:  3363*
DSWR1  . . . . . . 3354:  3362*
DTPC . . . . . . . 4197:  4192*
DTYER  . . . . . . 4158:  4174*
DTYPE  . . . . . . 4192:  4143*  4146*  4148*  4151*  4154*
DV4  . . . . . . . 1443:   505
DWRT . . . . . . . 4382:  4368*
ECCCNT . . . . . . .122:   783*  1520
ECMES  . . . . . . 2541:  1524
ECSAND . . . . . . 4646:  3626   3653   3727
ENDVCP . . . . . . 2361:  1332*  2350*
ENT2 . . . . . . . 1589:  1509*  1527*  1582*  1586*
ENTRY  . . . . . . 1539:  1446
EOP0 . . . . . . . 1448:  Symbol not referenced
EOP1 . . . . . . . 2549:  Symbol not referenced
EOP2 . . . . . . . 3525:  Symbol not referenced
EOP3 . . . . . . . 4716:  Symbol not referenced
EPROM  . . . . . . . 32:   273*  1457*  2551*  3526*
ERMES  . . . . . . 2531:  1726
EX1  . . . . . . . 3135:  3114*
EX2  . . . . . . . 3147:  3155*
EX3  . . . . . . . 3157:  3151*
EXEX . . . . . . . 2350:  2569
EXEX.P1  . . . . . 2569:  Symbol not referenced
EXEX2  . . . . . . 2357:  1580*
EXIT?  . . . . . . 3109:  3098*
EXLOAD . . . . . . .194:  3233*  3279*  3283   3294*  3295*
FILE . . . . . . . .144:  2666*  3239   3304*  3623   3630   3855   3857*
                   3887*  3977   3979*  4011   4013*  4161   4208   4235
                   4245   4571*  4583   4586*  4591   4593*
FIRST  . . . . . . .186:  1542*  1590*  2710*
FTGF . . . . . . . 2945:  2938*
```

```
FTLP . . . . . . . . 2937:  2943*
FXRX . . . . . . . . 2333:  2295*
GET. . . . . . . . . 3195:  3212*
GET1 . . . . . . . . 3096:  3107*  3120*  3133*
GETCH  . . . . . . . 2986:  1656*  1910*  1934*  2771   3096*  3195*
GETCR  . . . . . . . 2992:  2922*  3058*
GETCR2 . . . . . . . 3002:  2994*
GETCR3 . . . . . . . 2999:  3003*
GETN . . . . . . . . 3080:  1707   1763*  1807   1839*  1974*  2106*  2127*
                     2173*  2194
GETREG . . . . . . . 1998:  1972*  1995*
GETS . . . . . . . . 3170:  2246*
GETSL  . . . . . . . 3178:  3196*
GLCHK  . . . . . . . 1938:  1931*
GR . . . . . . . . . . 77:   665*   667*   668*   677*   684*   685*   692*
                     1014*  1278*  1280*  1556*  1557*  1982*  1994*  2042*
                     2050*  2716*  2718*  2892*  2894*  2896*  3822*  3824*
GSBS . . . . . . . . 3199:  3180*
GSLR . . . . . . . . .158:  3174*
GTREC  . . . . . . . 3653:  3690*  3704*
HEAD.CYLINDER  . . . .145:  2670*  3972*  4014   4017*  4206*  4220   4225*
                     4237   4246   4251   4344
HELP . . . . . . . . 2530:  1734
HPIB . . . . . . . . 4515:  4076   4117   4187   4295   4297   4305   4343*
                     4354   4358   4373   4386   4431*  4479   4481   4504
                     4553   4556
HPIBLP . . . . . . . 4519:  4525*
HPIBX  . . . . . . . 4549:  4299   4301   4303   4307   4309*  4311*  4313*
                     4346
HPIT . . . . . . . . .130:  4550*  4554
I.0  . . . . . . . . 3032:  2925   3024   3056*  3062*  3786*  3796*
I.00 . . . . . . . . 3035:  3036*
I.01 . . . . . . . . 3037:  Symbol not referenced
IDM  . . . . . . . . 1406:   686
IERR . . . . . . . . 3116:  3102*
ILDEF  . . . . . . . 1228:   574    641
ILI  . . . . . . . . .111:   575*  1379*  1382*
ILIJMP . . . . . . . 1379:  1069
ILINT  . . . . . . . 1229:  1228
ILJMP  . . . . . . . 1382:   583
ILLP . . . . . . . . .584:   588*
IN1C . . . . . . . . 2987:  Symbol not referenced
INTIO  . . . . . . . .117:  1000*  1380*
IOE4 . . . . . . . . .961:   947*
IOEN4  . . . . . . . 1108:  1103*
IOER . . . . . . . . 1235:   960*   962*   991*   998*  1110*  1220*
IOESC  . . . . . . . 1234:  1008*  1032*  1036*  1040*  1046*  1056*  1061*
                     1064*
```

```
IOIDEF . . . . . . . 1049:   999
IOIJMP . . . . . . . 1380:  1015
IOINT  . . . . . . . 1050:  1049
IOL0 . . . . . . . . .936:   950*
IOL1 . . . . . . . . .949:   956*
IOL2 . . . . . . . . .972:   979*   989*
IOL3 . . . . . . . . .974:   981*
IOL4 . . . . . . . . 1003:  1073*
IOL5 . . . . . . . . 1091:  1101*  1106*
IOLP . . . . . . . . 1376:   922
ION.1  . . . . . . . 1294:  1288*
ION.2  . . . . . . . 1299:  1285*
ION0 . . . . . . . . .967:   941*
ION1 . . . . . . . . .982:   976*
ION2 . . . . . . . . .995:   984*
ION3 . . . . . . . . 1084:  1005*
ION4 . . . . . . . . 1098:  1094*
ION6 . . . . . . . . .691:   674*
IONXT  . . . . . . . 1932:  1928*
IOREG  . . . . . . . 1955:  1950*
IORGN  . . . . . . . .134:  1933*  1936   1937*  1955   1959   1963   1967
                     2000
IPF  . . . . . . . . .898:   891*   899*  1264*
IPRTY  . . . . . . . .853:   852
ISDIG  . . . . . . . 4603:  3097*  4579*
ISDIGDN  . . . . . . 4615:  4606*  4610*
ITBG . . . . . . . . .635:   634
KMES . . . . . . . . 2540:  1518
L1 . . . . . . . . . 2838:  2845*
LAST . . . . . . . . .270:  Symbol not referenced
LERR . . . . . . . . .136:  1481   1489   2444   2676*  3226*  3235*  3240*
                     3267*  3274*  3311*  3319*  3323*  3332*  3366*  3618*
                     3706*  3713*  3714*  3768*  3813*  3819*  3859*  3919*
                     3921*  4049*  4105*  4125*  4129*  4160*  4266*  4315*
                     4363*  4394*  4420*  4423*  4435*  4442*
LISEN  . . . . . . . 4475:  4465*
LSN  . . . . . . . . 4640:  4075   4472
LSTR . . . . . . . . .157:  2640   3172*  3190   3193*  3199   3204*
M1 . . . . . . . . . 1425:   368    380
M100 . . . . . . . . 4693:  4468   4540
M1600  . . . . . . . 4695:  3943
M2 . . . . . . . . . 4687:  4058   4376
M3 . . . . . . . . . 4688:  4459
M350 . . . . . . . . 4694:  4249
M4 . . . . . . . . . 4689:  4532
M48  . . . . . . . . 4691:  4604
M5600  . . . . . . . 4696:  4518
M64  . . . . . . . . 4692:  3721   3741   4050
M8 . . . . . . . . . 4690:  4608
```

```
MAP . . . . . . . . .165:   727*   731    733*   763    843    864    905
                    910*   1338*  2139*  2184   2207   2231   2305*  2313
                   2325*   2627*  3299*  3300   3686   3688*  3908   3910*
                   4004    4006*  4026   4034
MAP01  . . . . . . . 2149:  2142*
MAP02  . . . . . . . 2160:  2164*
MAP15  . . . . . . . 2156:  2168*
MAPP1  . . . . . . . 2177:  2238*
MAPP2  . . . . . . . 2226:  2175*
MAPPG  . . . . . . . 2173:  2145*
MBUF . . . . . . . . 2151:  2154   2188
MCNTR  . . . . . . . .103:  1842*  1886*  2157*  2163*
MES00  . . . . . . . 3391:  2527
MES01  . . . . . . . 3385:  2528
MES02  . . . . . . . 3394:  2529
MES03  . . . . . . . 3395:  Symbol not referenced
MES07  . . . . . . . 3399:  2567
MES09  . . . . . . . 3402:  2530
MES11  . . . . . . . 3459:  2532   2566
MES12  . . . . . . . 3462:  2546
MES13  . . . . . . . 3466:  2547
MES14  . . . . . . . 3468:  2548
MES15  . . . . . . . 3470:  2534
MES16  . . . . . . . 3473:  2535
MES22  . . . . . . . 3476:  2533   2565
MES32  . . . . . . . 3480:  2536
MES33  . . . . . . . 3483:  2537
MES35  . . . . . . . 3487:  2538
MES36  . . . . . . . 3490:  2539
MES37  . . . . . . . 3493:  2540
MES38  . . . . . . . 3496:  2542   2544
MES41  . . . . . . . 3500:  2543
MES43  . . . . . . . 3504:  2541
MES44  . . . . . . . 3508:  2545
MES46  . . . . . . . 3512:  2531
MES47  . . . . . . . 3516:  3213
MES48  . . . . . . . 3519:  3121
MES62  . . . . . . . 3522:  2427
MLOST  . . . . . . . . 96:   295*   540    565*   699    702*   740    767
                    802     855    862*  1155
MMESS  . . . . . . . 2534:  1751   1822   1849
MPBUF  . . . . . . . .167:   714   1376   2151   2187   2210   4630
MPLP . . . . . . . . .718:   725*
MPMES  . . . . . . . 2538:  1854   2229
MPT  . . . . . . . . .113:   579*  1383*
MPTJMP . . . . . . . 1383:   594
MPTR . . . . . . . . .127:  1852*  1863   1865   1867   1870*  2155*  2160
                   2162*   2190*  2206*
```

```
MRBT . . . . . . . . 2618:  1198   3931
MRBT2  . . . . . . . 2620:  2688*
MSIZE  . . . . . . . .121:   844*   906*  1514   2314   2633
MT?  . . . . . . . . 1827:  1808*  1817*
MTM1 . . . . . . . . .746:   742*
MTST . . . . . . . . .699:  Symbol not referenced
MTST0  . . . . . . . .802:   882*
MTST3  . . . . . . . .761:  Symbol not referenced
MTST4  . . . . . . . .795:   903
MTST5  . . . . . . . .905:   735*   752*   758*
MTSTE  . . . . . . . .841:   720*   779*   782*   792*   822*   831*   857*
                     874*    879*   908*
MTSTL  . . . . . . . .814:   805*
MTSTM  . . . . . . . .731:   812*   835*
MU2  . . . . . . . . 1442:   498
MZSV . . . . . . . . .168:  2308   2332
N1 . . . . . . . . . 2504:  1570   2223
N10  . . . . . . . . 2508:  Symbol not referenced
N16  . . . . . . . . 2509:  1968
N2 . . . . . . . . . 2505:  1791
N23  . . . . . . . . 2510:  1964
N24  . . . . . . . . 2511:  1960
N27  . . . . . . . . 2512:  1956
N32  . . . . . . . . 2513:  1896   2134   2180
N4 . . . . . . . . . 2506:  1674   2152
N48  . . . . . . . . 2514:  Symbol not referenced
N8 . . . . . . . . . 2507:  1861   1873   2156
NDCLR  . . . . . . . .187:   921*  1585*  1607*
NEXT . . . . . . . . 1633:  2572
NOVCP  . . . . . . . 1430:  Symbol not referenced
NVCP . . . . . . . . 1219:  1210*
NXPG . . . . . . . . 2215:  2195*
NXPG1  . . . . . . . 2206:  2199*  2201*
NXPG2  . . . . . . . 2225:  2219*
O1 . . . . . . . . . 4658:  4153   4167
O10  . . . . . . . . 4663:  4306
O1000  . . . . . . . 4680:  4098
O100000  . . . . . . 4685:  3891
O1001  . . . . . . . 4289:  4096
O1005  . . . . . . . 4683:  4183
O1100  . . . . . . . 4681:  4100
O1101  . . . . . . . 4682:  4095
O12  . . . . . . . . 4664:  3632
O17  . . . . . . . . 4665:  4133   4204
O2 . . . . . . . . . 4659:  3812   3890   4186   4574
O20  . . . . . . . . 4666:  4296
O200 . . . . . . . . 4671:  4274
O201 . . . . . . . . 4672:  4150
```

```
0204 . . . . . . . . 4673:  4145
021  . . . . . . . . 4667:  3761
03 . . . . . . . . . 4660:  4053   4114   4156   4170
0377 . . . . . . . . 4674:  3787   3793   4199   4349   4356   4508   4552
                     4555
0400 . . . . . . . . 4675:  4012
0404 . . . . . . . . 4676:  4147
0406 . . . . . . . . 4677:  Symbol not referenced
06 . . . . . . . . . 4661:  3738
060  . . . . . . . . 4668:  3637   3816   4581
07 . . . . . . . . . 4662:  3832   3836
076  . . . . . . . . 4669:  4304
077  . . . . . . . . 4670:  3821
07777  . . . . . . . 4684:  Symbol not referenced
OT1  . . . . . . . . 2826:  2821*
OT2  . . . . . . . . 2846:  2825*
OTDL . . . . . . . . 2785:  2791*
OTDL2  . . . . . . . 2796:  2803*
OUT1C  . . . . . . . 3006:  Symbol not referenced
OUT2C  . . . . . . . 3012:  2912   2976*
OUTD . . . . . . . . 2780:  1490   1517*  1523*  2445
OUTN . . . . . . . . 2814:  1473   1477   1502*  1505   1612   1617   1622
                     1630   1705   1754   1758   1804   1853*  1858   1866
                     2006   2014*  2059   2089*  2104*  2161   2192   2232
                     2236
P.1  . . . . . . . . 2743:  2750*  2754*
P0 . . . . . . . . . .275:  Symbol not referenced
P0.A . . . . . . . . .183:  Symbol not referenced
P0.B . . . . . . . . .184:  Symbol not referenced
P0.CT  . . . . . . . .177:   971*   972*   978    982
P0.T3  . . . . . . . .178:  Symbol not referenced
P0C00  . . . . . . . 1244:  1258*  1269*
P1 . . . . . . . . . 1462:  Symbol not referenced
P2 . . . . . . . . . 2553:  Symbol not referenced
P3 . . . . . . . . . 3528:  Symbol not referenced
P3.CT  . . . . . . . .195:  3245*  3308*  3318*  3353*  3361*
PAGE . . . . . . . . .166:  1698*  1719   1864*  1875   1878   1880*  2189
                     2217   2226*  2235   2237
PARTIAL  . . . . . . .137:  3876*  3894   3992*  4028
PCNTR  . . . . . . . .104:  1862*  1871*  2153*  2167*
PCOMN  . . . . . . . 4479:  4473*
PDOWN  . . . . . . . .886:   885
PE . . . . . . . . . .109:   108*   576*   762*   799*  1378*  1595*
PE1  . . . . . . . . 1529:  1594
PEADD  . . . . . . . .151:   705*   869*   889   1492   1499   1503   1591*
PEDEF1 . . . . . . . .852:   798
PEDEF2 . . . . . . . .903:   761
PEFLAG . . . . . . . .118:  1368*  1606*  1641   1648*
PEINT  . . . . . . . 1361:  1529
```

```
PEJMP  . . . . . . . 1378:   591
PEJMPI . . . . . . . .108:   593*
PEJSB  . . . . . . . 1530:  1592
PEMAP  . . . . . . . .152:   865*  1498
PEMES  . . . . . . . 2542:  1645
PERTN  . . . . . . . .107:  1369*  1530*
PETMP  . . . . . . . .106:  1348*  1353   1361*  1366
PFW  . . . . . . . . .112:   578*   708*  1381*
PFWDEF1  . . . . . . .885:   707
PFWJMP . . . . . . . 1381:   589
PGMES  . . . . . . . 2539:  2233
PH . . . . . . . . . 4492:  4497*
PHI  . . . . . . . . 4489:  4077   4109*  4121*  4181*  4288*  4319*  4326*
PHI.I  . . . . . . . 4502:  4080*  4083*  4124*  4130*  4131*  4290*  4440*
PHI.L  . . . . . . . 4457:  4071*  4118   4286*  4322*  4374   4436*
PHI.TALK . . . . . . 4456:  4089*  4106*  4178*  4291*  4329*  4335*  4364*
                     4387
PHI1 . . . . . . . . 4488:  4337*  4369*  4382*  4438*  4466*  4475*  4505
PHIFL  . . . . . . . 4529:  4066*  4428*
PHIN . . . . . . . . 4490:  4059   4377   4460   4533
PIN  . . . . . . . . 4537:  4503
PMESS  . . . . . . . 2546:  1609
PNTR . . . . . . . . . 98:  2782*  2787*  2788*  2796   2798*  2799   3093*
                     3103*  3104*  3128   3130*  3145*  3147   3148*  4140*
                     4144*  4149*  4152*  4155*  4169*  4172*  4198   4202
PNTRS  . . . . . . . . 99:  3094*  3144
POINTER  . . . . . . .188:  3679*  3694   3698*  3720*  3746   3749   3754*
PPNTR  . . . . . . . .128:  2739*  2743   2748*
PREV . . . . . . . . 1791:  1786*  1788*
PRMES  . . . . . . . 2537:  2293
PRMPT  . . . . . . . 2529:  1649
PROER  . . . . . . . 1224:   611*   615*   621*   631*   646*
PRSET  . . . . . . . 2291:  2277*  2562
PRTLP  . . . . . . . 1260:  1254*
PTDC . . . . . . . . 3936:  1189
PTDFO  . . . . . . . .427:   419    422
PTDF1  . . . . . . . .428:   401
PTDS . . . . . . . . 1191:  1176*
PTJMP  . . . . . . . .430:   424
PTJPR  . . . . . . . .429:   397
PTJYO  . . . . . . . .431:   413
PTLER  . . . . . . . 1200:  1187*  1195*  3932
PTLP . . . . . . . . 3939:  3950*
PTRM . . . . . . . . 1183:  1179*
PTRTO  . . . . . . . .401:   399    428    429*
PTRT1  . . . . . . . .410:   408*   431
PTSO . . . . . . . . 1172:  1153*  1157*  1161*
PTS1 . . . . . . . . 1208:  1129*  1139*  1181*
PTS2 . . . . . . . . 1127:  1122*
```

```
PTSTX  . . . . . . . 1116:  1097*  1268*
PTWLP  . . . . . . . 3944:  3948*
PUTCH  . . . . . . . 3005:  1626   1628   1720   1900   1988   1992   2114
                     2747*  2770   2801   2824   2836*  2842   3015*  3017
PUTCT  . . . . . . . .105:  2741*  2749*  2752*
PUTS . . . . . . . . 2737:  1470   1486   1488   1497   1512   1519   1525
                     1610   1615   1620   1624   1646   1650   1716   1718
                     1727   1735   1748   1750   1752   1756   1819   1821
                     1823   1848   1850   1855   1860   1885   1986   2112
                     2159   2228   2230   2234   2248   2294   2303   2443
                     2817   2847   3117   3119   3132   3211
R  . . . . . . . . . 4655:  Symbol not referenced
RCI.ID . . . . . . . .248:  2966*  2969*  2970*
RCI.IZ . . . . . . . .247:  2891*  2909*  3049*
RCL.IZ . . . . . . . .251:  Symbol not referenced
RCOMN  . . . . . . . .223:  2103*  2116*
RCS.CM . . . . . . . .255:  3022*  3025*  3026*
RCS.FT . . . . . . . .254:  Symbol not referenced
RCTIO  . . . . . . . .230:  Symbol not referenced
RCTU . . . . . . . . .225:  3611*  3707*  3708*  3712*
RDC1 . . . . . . . . 4645:  3659
RDCIN  . . . . . . . .240:  4047*  4193*  4258*
RDCLD  . . . . . . . .233:  3955*  3965*  4040*  4041*
RDCRW  . . . . . . . .242:  4263*  4444*  4445*
RDS.B  . . . . . . . .245:  3369*  3375*
RDS.CM . . . . . . . .253:  2950*  2955*  2957*
RDS.FT . . . . . . . .252:  2930*  2933*  2944*  2945*  2946*  3028*
RDS.GT . . . . . . . .246:  3307*  3327*
RDSLD  . . . . . . . .243:  3219*  3276*  3277*
RDTPC  . . . . . . . .241:  4197*  4230*
RECHO  . . . . . . . .216:  Symbol not referenced
REENT  . . . . . . . 2658:  1537
REGOK  . . . . . . . 1972:  1962*
RENDV  . . . . . . . .208:  2361*  2363*  2367*
RENQAK . . . . . . . .209:  2757*  2761*  2767*  2772*
RENT2  . . . . . . . 2681:  2678*
RESUA  . . . . . . . 1444:   506
RESUB  . . . . . . . 1445:   510
RFLAG  . . . . . . . .163:  Symbol not referenced
RFTMP  . . . . . . . .133:  2036*  2038
RGETC  . . . . . . . .218:  2987*  3000*
RGETN  . . . . . . . .220:  3080*  3137*  3139*  3160*
RGETREG  . . . . . . .219:  1998*  2007*
RGETS  . . . . . . . .210:  3170*  3182*
RGTO1  . . . . . . . .258:  Symbol not referenced
RHPIB  . . . . . . . .237:  4515*  4523*
RHPIBX . . . . . . . .239:  4549*  4557*
RI.O . . . . . . . . .259:  3032*  3038*
RLCH1  . . . . . . . .213:  1892*  1901*
```

```
RLCHR  . . . . . . .214:  Symbol not referenced
RMERR  . . . . . . 3922:  3853*  3868*
RMERR2 . . . . . . 3921:  3870*
RMERR3 . . . . . . 3919:  3885*
RMESS  . . . . . . 2545:  1985   2111
RMLDR  . . . . . . 3848:  1185*  2424   3538*
RMSC . . . . . . . 2497:  2422
ROM2 . . . . . . . 3862:  3888*
ROM3 . . . . . . . 3898:  3903*
ROM4 . . . . . . . 3886:  3882*
ROM5 . . . . . . . 3891:  3917*
ROUT1  . . . . . . .211:  Symbol not referenced
ROUT2C . . . . . . .256:  3012*  3018*
ROUTD  . . . . . . .215:  2780*  2805*
ROUTN  . . . . . . .212:  2814*  2848*
RPHI . . . . . . . .235:  4490*  4492   4495*  4498*
RPHI?  . . . . . . .234:  4458*  4480   4482*  4483*
RPHIF  . . . . . . .238:  4529*  4543*
RPHII  . . . . . . .236:  4502*  4509*
RPUTC  . . . . . . .217:  2926*  3006*  3063*
RPUTS  . . . . . . .207:  2737*  2746*
RRMLD  . . . . . . .232:  3848*  3906*  3922*  3923*
RRSTO  . . . . . . .222:  2702*  2712*  2730*
RS.SC  . . . . . . .244:  3811*  3840*  3841*
RSCNSC . . . . . . .221:  4568*  4599*
RSTOR  . . . . . . 2702:  2351*
RTG.BF . . . . . . .249:  2917*  2981*  2983*
RTG.TB . . . . . . .250:  2972*  2974*  2979*
RTI.B  . . . . . . .227:  3783*  3788*
RTI.W  . . . . . . .226:  3773*  3779*
RTO.B  . . . . . . .228:  3792*  3797*
RTO.W  . . . . . . .229:  3801*  3807*
RVCODE . . . . . . .285:   572
S  . . . . . . . . 4656:  3644   3764
S.SC . . . . . . . 3811:  3223*  3614*  3852*  3975*
SACOMN . . . . . . .101:  2110*  2115
SAVEA  . . . . . . . 86:  1072*  1545*  1616   2343*  2687*  2729   3815*
SAVEB  . . . . . . . 87:   573*   958*  1621   2341*  2632*  3275*  3365*
                   3646*  3766*  4126*  4158*  4441*
SAVEE  . . . . . . . 84:  1555*  2682   2723
SAVEG  . . . . . . . 88:  1340*  1559*  1938   1981   1993   2043   2051
                   2713
SAVEI  . . . . . . . 82:  1339*  1541*  2707   2709*
SAVEM  . . . . . . . 93:  1571*  1753   1757   1772   1783   1793   1794*
                   1803   1815*  1824   1851
SAVEO  . . . . . . . 83:  1553*  2719
SAVEP  . . . . . . . 85:  1569*  1611   2345*  2354   2518   2625*
SAVEQ  . . . . . . . 91:  1335*  1547*  2727
```

```
SAVEW  . . . . . . . . 94:  1337*  1349   1362   1563*  1629   1856   2065
                      2070*  2074   2333   2354
SAVEX  . . . . . . . . 89:  1560*  2705   4594*
SAVEY  . . . . . . . . 90:  1561*  2706
SAVEZ  . . . . . . . . 92:  1549*  2725
SCER . . . . . . . . 3841:  3818*  3826*
SCETC  . . . . . . . .135:  1184*  1192*  3814   3820   3828   3940*  4569*
                      4598*
SCM  . . . . . . . . 1405:   942
SCNDN  . . . . . . . 4590:  4580*
SCNLP  . . . . . . . 4577:  4588*
SCNSC  . . . . . . . 4568:  2410   2423   2431   2437
SECTR.TRACK  . . . . .146:  2673*  3973*  4018   4021*  4201*  4210   4216*
                      4228   4229*  4239   4252   4347   4355
SELFERR  . . . . . . 2543:  1469
SIDE?  . . . . . . . .189:  Symbol not referenced
SOFTERR  . . . . . . 2544:  1496
SPC2 . . . . . . . . 2532:  1623   1749   1820
SPC3 . . . . . . . . 2533:  1717
SPTR . . . . . . . . 2450:  2249   2260   2396
SRGP1  . . . . . . . 1434:   433
SRGP2  . . . . . . . 1435:   435    461
SRGP3  . . . . . . . 1436:   457
START  . . . . . . . .292:   901*  1135*  1142*  2392
STATS  . . . . . . . . 79:  3041*  3043*
STMAP  . . . . . . . 4620:   736*   842*   860*   866*   911*  1344   2317*
                      2630*  3301*  3689*  3911*  4007*
STORE.POINTER  . . . .159:  2622*  2636   2647   3177*  3186   3194*  3205
                      3207*  4576*  4577   4587*
STORM  . . . . . . . 1814:  1811*
STRNG  . . . . . . . .155:  2450   3168   4714
STRTR  . . . . . . . 2568:  2631
SUBCH  . . . . . . . .140:  2660*  3717   3833*  4073   4471   4477
SVCHR  . . . . . . . .100:  1911*  1944   1987   2113
TBG  . . . . . . . . .110:   577*   626*   642*  1377*
TBGCNT . . . . . . . .120:   639*  2939   3944   4062   4519
TBGDEF1  . . . . . . .634:   625
TBGJMP . . . . . . . 1377:   596
TCCWI  . . . . . . . 3066:  3054
TCCWO  . . . . . . . 3065:  3042   3060
TEMP . . . . . . . . .131:   525*   527   2133*  2140   2197*  2211   2815*
                      2818   2832*  2837   2839*  2843   3013*  3016   3776*
                      3778   3802*  3805   3862*  3883   3886   4134*  4166
                      4209*  4215   4219*  4226   4572*  4582   4585*  4590
TEMPO  . . . . . . . .181:   766*   780    785*   832    923*   937*   938*
                       943*   949    953   1002*  1003   1012   1067   1071*
                      1086*  1100*  1104   1234   1653*  2003*  2005   2048*
                      2058   2251*  2267   2419   2619*  2649   2685*  3229
                      3649   4272   4316   4366   4395   4425
```

```
TEMP1 . . . . . . . .180:  925*  948   973   995   1001  1085  2056*
                    2060   3633* 3636  3680* 3696  3697* 3703  3747*
                    3751   3752* 3757
TEMP2 . . . . . . . .179:  1088* 1105* 3613* 3670* 3710  3722* 3724
                    3759*
TEMP3 . . . . . . . .182:  3355* 3358
TFLAG . . . . . . . .164:  Symbol not referenced
TG.BF . . . . . . . 2981:  1655* 1706  1762* 1806  1973* 2105* 2193
TG.TB . . . . . . . 2972:  2760*
TLK . . . . . . . . 4639:  4074  4478
TMESS . . . . . . . 2535:  1755
TRAPFLAG . . . . . .154:   920* 1120
TRYCT . . . . . . . .125:  3937* 3949*
TRYNM . . . . . . . 3933:  3936
UIJMPI . . . . . . . .115:  601*
UIT . . . . . . . . .116:   115*  581* 1384*
UIT1 . . . . . . . . 1386:  580
UITINT . . . . . . . 1348: 1386
UITJMP . . . . . . . 1384:  600
UITJSB . . . . . . . 1385:  598
UITRTN . . . . . . . .114: 1354* 1385*
UNIT . . . . . . . . .139: 2663* 3619* 3621  3837* 4112  4241  4293
                    4340   4352  4371  4384
UNIT.HEAD . . . . . .142:  4065* 4176* 4184  4257* 4278  4339  4350
UNL . . . . . . . . 4463:  4424
VCP.FLAG . . . . . . .190:  681* 1108* 1208  1266
VCPDS . . . . . . . 2908:  2904
VCPEX . . . . . . . 2368:  2364
VCPL . . . . . . . . .665:  678*
VCPL1 . . . . . . . .680:   670*
VCPSC . . . . . . . .191:   682* 2895
VCPTFLG . . . . . . .153:   293*  539   564*  701  1127  1507  2389*
VERMG . . . . . . . 2528:  1511
VFP . . . . . . . . 1463:  1216
VFP.0 . . . . . . . 1492:  1467* 1483*
VFP.1 . . . . . . . 1507:  1494*
VW1 . . . . . . . . .147:  4236* 4248* 4302
VW2 . . . . . . . . .148:  4238* 4254* 4300
VW3 . . . . . . . . .149:  4240* 4255* 4298
W . . . . . . . . . 4657:  3650  4273  4317  4367  4396  4426
WENQ . . . . . . . . 4650: 3735
WMAP . . . . . . . . . 46: 1562
XEQT . . . . . . . . .201:  203* 1597* 1978  1980* 1983* 1989  2001*
                    2002*  2046* 2047* 2054* 2055* 2521*
ZERO . . . . . . . . 2460: 1935  1991
Macro:  No  errors total
```

A600 BASESET MICROCODE — APPENDIX D

A representative listing of the baseset microcode used in the A600 computer is presented in the following pages. This listing may not accurately reflect the as-installed configuration of microcode actually contained in ROM. The listing is solely intended to serve as an aid to those users attempting to develop microprograms.

```
LINE          STATEMENT

TITLE   A600 BASESET DEFINITIONS 01/19/82
LIST  E
;
; &LBDEF 12101-18024 REV 2210 820524
;
WORD 56
;
; REGISTER DEFINITIONS
;
R0:         EQU   B 00000
R1:         EQU   B 00001
R2:         EQU   B 00010
R3:         EQU   B 00011
R4:         EQU   B 00100
R5:         EQU   B 00101
R6:         EQU   B 00110
R7:         EQU   B 00111
R8:         EQU   B 01000
R9:         EQU   B 01001
R10:        EQU   B 01010
R11:        EQU   B 01011
R12:        EQU   B 01100
R13:        EQU   B 01101
R14:        EQU   B 01110
R15:        EQU   B 01111
;
A:          EQU   B 00000              ; MACRO A REGISTER
B:          EQU   B 00001              ; MACRO B REGISTER
X:          EQU   B 00010              ; MACRO X REGISTER
Y:          EQU   B 00011              ; MACRO Y REGISTER
MAPD:       EQU   B 01101              ; DATA1 AND DATA2 MAP REGISTER
MAPX:       EQU   B 01110              ; EXECUTE MAP NUMBER REGISTER
PC:         EQU   B 01111              ; MACRO P REGISTER
;
CAB:        EQU   B 10000              ; IR11 SELECTS A OR B
CXY:        EQU   B 10010              ; IR03 SELECTS X OR Y
TAB:        EQU   B 10001              ; MEMORY ADDR SELECTS A OR B
MPY:        EQU   B 10011              ; Q0 SELECTS AB OR ZB IN REG 0
MPY4:       EQU   B 10111              ; Q0 SELECTS AB OR ZB IN REG 4
DIV:        EQU   B 10100              ; DIVIDEND SIGN XOR DIVISOR SIGN
                                               SELECTS ADD/SUB
PORM:       EQU   B 11111              ; PCMRG FROM IR DECODE SELECTS PC
                                               OR R12
;
; AM2901 SOURCE OPERANDS (R S)
;
AQ:         EQU Q 0
AB:         EQU Q 1
ZQ:         EQU Q 2
```

```
LINE           STATEMENT

  48  ZB:       EQU Q 3
  49  ZA:       EQU Q 4
  50  DA:       EQU Q 5
  51  DQ:       EQU Q 6
  52  DZ:       EQU Q 7
  53  ;
  54  ; AM2901 ALU FUNCTIONS (R FUNCTION S)
  55  ;
  56  ADD:      EQU Q 0
  57  SUBR:     EQU Q 1
  58  SUBS:     EQU Q 2
  59  OR:       EQU Q 3
  60  AND:      EQU Q 4
  61  NOTRS:    EQU Q 5
  62  EXOR:     EQU Q 6
  63  EXNOR:    EQU Q 7
  64  PASS:     EQU Q 3
  65  ;
  66  ; AM2901 DESTINATION CONTROL
  67  ;
  68  QREG:     EQU Q 0
  69  NOP:      EQU Q 1
  70  RAMA:     EQU Q 2
  71  RAMF:     EQU Q 3
  72  SRAMQR:   EQU Q 4
  73  SRAMR:    EQU Q 5
  74  SRAMQL:   EQU Q 6
  75  SRAML:    EQU Q 7
  76  ;
  77  ;      JTAB OPERAND VALUES
  78  ;
  79  LVLO:     EQU B 00000000          ; INITIAL DECODE VALUE
  80  LOWSC:    EQU B 00000100          ; LOW SELECT CODE I/O INSTR
  81  IROT3:    EQU B 00001000          ; IR BITS 0 TO 3 FOR COUNTER
  82  SRG1:     EQU B 00001001          ; MAP CLE,SL* SRG OPERATION
  83  SRG2:     EQU B 00001010          ; MAP SECOND SRG SHIFT OPERATION
  84  WORDCNT:  EQU B 00001011          ; I/O CNTRL WORD MAPS
  85  ;
  86  ;      CONDITION DEFINITIONS
  87  ;
  88  ;   -EXTERNAL STATUS REGISTER
  89  ;
  90  IRSKIP:   EQU B 0000            ; SKIP BASED ON COND SELECTED BY IR
  91  IR11:     EQU B 0010            ; INSTRUCTION REG BIT 11
  92  QPEI:     EQU B 0011            ; PENDING PARITY ERROR
  93  SINTRQ:   EQU B 0100            ; PENDING I/O INTERRUPT
  94  MPEN:     EQU B 0101            ; MEM PROTECT ON
  95  IORQ:     EQU B 0110            ; I/O REQUEST
  96  INTRPT:   EQU B 0111            ; INTERRUPT
```

```
LINE          STATEMENT

;
;          2904 STATUS REGISTERS
;
; NOTE:    THE 2910 CC INPUT IS ACTIVE LOW, SO THE
;          2904 CODE MUST BE USED TO PRODUCE CT=L
;          FOR THE DESIRED TEST.
;
SGNXOVR: EQU B 0010                            ; SIGN .XOR. OVR
Z:       EQU B 0101                            ; ZERO
NZ:      EQU B 0100                            ; NOT ZERO
OVR:     EQU B 0111                            ; OVERFLOW
NOVR:    EQU B 0110                            ; NOT OVERFLOW
C:       EQU B 1011                            ; CARRY
NC:      EQU B 1010                            ; NOT CARRY
SIGN:    EQU B 1111                            ; SIGN SET (NEGATIVE)
NSIGN:   EQU B 1110                            ; SIGN CLR (POSITIVE)
LT:      EQU B 0011                            ; 2'S COMPLEMENT LESS THAN
ULE:     EQU B 1101                            ; UNSIGNED LESS EQUAL
;
;         INSTRUCTION DEFINITIONS
;
FILLER: DEF H 00,H 00,H 00,H 00,H 00,H 00,H 00   ; NO PROG 27S35 WORD
;
AM2901: DEF 21X,5VB 00000,5VB 00000,3VX,3VX,3VX,16X
;
;         2910 OPERATIONS
;
JZ:     DEF B 0000,52X                          ; JUMP ZERO, INITIALIZE
CALL:   DEF B 0001,10X,B 1,29X,12V:%X           ; UNCOND CALL
CCALL:  DEF B 0001,10X,B 0,29X,12V:%            ; CONDITIONAL CALL
JMAP:   DEF B 0010,52X                          ; JUMP THRU MAP
JP:     DEF B 0011,10X,B 1,29X,12V:%            ; UNCOND JUMP
CJP:    DEF B 0011,10X,B 0,29X,12V:%            ; CONDITIONAL JUMP
PUSH:   DEF B 0100,10X,B 1,29X,12V:%X           ; PUSH/UNCOND LOAD COUNTER
CPUSH:  DEF B 0100,10X,B 0,29X,12V:%X           ; PUSH/COND LOAD COUNTER
CVECT:  DEF B 0110,10X,B 0,29X,12V:%            ; CONDITIONAL VECTOR
JRP:    DEF B 0111,10X,B 0,29X,12V:%   ; CONDITIONAL JUMP THRU REG OR PL
RFCT:   DEF B 1000,52X                          ; REPEAT LOOP IN FILE
RPCT:   DEF B 1001,40X,12V:%                    ; REPEAT LOOP IN PIPELINE
RET:    DEF B 1010,10X,B 1,41X                  ; UNCONDITIONAL RETURN
CRET:   DEF B 1010,10X,B 0,41X                  ; CONDITIONAL RETURN
CJPP:   DEF B 1011,10X,B 0,29X,12V:%            ; COND JUMP PIPELINE AND POP
JPP:    DEF B 1011,10X,B 1,29X,12V:%           ; UNCOND JUMP PIPELINE AND POP
LDCT:   DEF B 1100,40X,12V:%X                   ; LOAD COUNTER AND CONTINUE
LOOP:   DEF B 1101,10X,B 0,41X                  ; END LOOP TEST
CONT:   DEF B 1110,52X                          ; CONTINUE
TWB:    DEF B 1111,10X,B 0,29X,12V:%            ; LOOP AND COND BRANCH
;
```

```
LINE          STATEMENT

 145  IMM:        DEF 4X,B 0110,32X,16V            ; IMMEDIATE DATA (16 BITS)
 146  IMMB:       DEF 4X,B 0110,40X,8V:%
 147  ;
 148  ;       2904 OPERATIONS
 149  ;          PL38,39 / 53,54
 150  CARRYH:     DEF 16X,B 01,38X               ; CARRY HIGH INTO 2901 CONTROL
 151  CARRYL:     DEF 16X,B 00,38X               ; CARRY LOW INTO 2901 CONTROL
 152  CARRYEXT:   DEF 16X,B 10,38X        1016   ; EXTERNAL CARRY INTO 2901
 153  CARRYREG:   DEF 16X,B 11,38X               ; CARRY STATUS REG
 154  CARRYUC:    DEF 16X,B 11,1X,B 01,19X,B 0110,12X ; MICRO STATUS REG CARRY
                                                          BIT
 155  CARRYNUC:   DEF 16X,B 11,1X,B 01,19X,B 1000,12X ; USR NOT CARRY BIT
 156  ;
 157  SETMSR:     DEF 18X,B 0,B 00,19X,B 0001,12X  ; MACHINE STATUS REG OP
 158  RSTMSR:     DEF 18X,B 0,B 00,19X,B 0011,12X
 159  INVMSR:     DEF 18X,B 0,B 00,19X,B 0101,12X
 160  LODMSR:     DEF 18X,B 0,B 10,19X,B 1111,12X ; LOAD MACHINE STATUS REG
 161  LODMSRCI:   DEF 18X,B 0,B 10,19X,B 1000,12X ; LOAD MSR WITH CARRY INVERT
 162  LODUSR:     DEF 18X,B 1,B 01,19X,16X        ; MICRO STATUS REG OP
 163  LODUSRCI:   DEF 18X,B 1,B 01,19X,B 1000,12X ; LOAD USR WITH CARRY INVERT
 164  LODUSROR:   DEF 18X,B 1,B 00,19X,B 0110,12X ; MICRO REG, OVERFLOW RETAIN
 165  SWAPEO:     DEF 8X,B 00,8X,B 0,B 00,19X,B 0100,12X ; SWAP MC & MO IN MSR
 166  ENBLC:      DEF 9X,B 0,46X                        ; ENABLE MC BIT
 167  ENBLO:      DEF 8X,B 0,47X                        ; ENABLE MO BIT
 168  ;
 169  ;       DOUBLE INTEGER SPECIALS
 170  ;
 171  ; * NUCLDMSR - SELECT NOT UC AS CARRY-IN AND
 172  ;              LOAD MSR & USR WITH CARRY INVERT
 173  ;
 174  ; * UCLDMSR  - SELECT UC AS CARRY-IN AND
 175  ;              LOAD MSR & USR WITH CARRY
 176  ;
 177  NUCLDMSR: DEF  18X,B 001,19X,B 1000,12X        ; CARRY INVERTED
 178  UCLDMSR:  DEF  18X,B 001,19X,B 0110,12X        ; CARRY NORMAL
 179  ;
 180  ;       SELECT CONDITION
 181  ;
 182  CONDMSR:  DEF  15X,B 0,3X,B 10,19X,4VX,12X ; MACHINE STATUS REG
 183  CONDUSR:  DEF  15X,B 0,3X,B 01,19X,4VX,12X ; MICRO STATUS REG
 184  CONDEXT:  DEF  15X,B 1,3X,B 10,19X,4VX,12X ; EXTERNAL STATUS REG
 185  CONDLMSR: DEF  15X,B 0,2X,B 0,B 10,19X,4VX,12X ; MSR TEST COND AND LOAD
 186  DIVCOND:  DEF  15X,B 0,3X,B 00,19X,B 1111,12X ;DIVIDE SPECIAL-In EQV Mn
 187  DIVUCOND: DEF  15X,B 0,3X,B 11,19X,B 1111,12X ; DIV SPECIAL- (NOT) In
 188  ;
 189  SHIFT:    DEF  52X,4V@                      ; SHIFT LINKAGE OPCODE
 190  ROTATEC:  EQU  B 1001                       ; ROTATE WITH CARRY
 191  ROTATE:   EQU  B 1010                       ; VANILLA ROTATE
```

```
LINE           STATEMENT

     ;
     JTAB:     DEF 13X,B 0,30X,8V,4X           ;ENABLE INST DECODE
     ;
     ;       FULL CYCLE SPECIAL
     ;
     SPFNOP:   DEF 4X,B 0000,48X             ; NOP
     MREAD:    DEF 4X,B 0001,48X             ; MEMORY READ
     MWRITE:   DEF 4X,B 0010,48X             ; MEMORY WRITE
     IRMRG:    DEF 4X,B 0011,48X             ; MRG READ SPECIAL
     LDMAPD:   DEF 4X,B 0100,48X
     L4D:      DEF 4X,B 0101,48X             ; Y BUS ROTATED LEFT 4 TO D BUS
     MIAK:     DEF 4X,B 0111,48X             ; I/O INTERRUPT ACKNOWLEDGE
     IFETCH:   DEF 4X,B 1000,48X             ; FETCH NEXT INSTRUCTION
     RFETCH:   DEF 4X,B 1001,48X             ; REFETCH I/O INSTRUCTION
     IORD:     DEF 4X,B 1010,48X             ; I/O READ
     IOWR:     DEF 4X,B 1011,48X             ; I/O WRITE
     MKLRON:   DEF 4X,B 1101,48X             ; MEMGO INTERRUPT KILLER ON
     MKLROFF:  DEF 4X,B 1100,48X             ; MEMGO INTERRUPT KILLER OFF
     STRD:     DEF 4X,B 1110,48X             ; STATUS READ
     SPRD:     DEF 4X,B 1111,45X,3V          ; SPECIAL READ
     ;
     ;       SPRD OPERANDS
     ;
     MAPRD:    EQU B 001                     ; READ A MAP REGISTER
     PELENL:   EQU B 010                     ; PARITY ERR LATCH ENBL LOW
     PELENH:   EQU B 011                     ; PARITY ERR LATCH ENBL HIGH
     PRLEN:    EQU B 100                     ; MEMORY PROTECT LATCH
     ECIRRD:   EQU B 101                     ; EXT CENTRAL INTRPT REG
     SWRD:     EQU B 110                     ; SWITCH REG READ
     SLACK:    EQU B 111                     ; SLAVE ACKNOWLEDGE
     ;
     ;       HALF CYCLE SPECIAL
     ;                      45 43
     SPHNOP:   DEF 10X,B 000,43X             ; NOP
     LDMDOR:   DEF 10X,B 011,43X             ; LOAD MDOR
     LDMAR:    DEF 10X,B 001,43X             ; LOAD MAR
     ENVE:     DEF 10X,B 010,43X             ; LOAD/RETAIN E AND O IN MSR
     SPWR:     DEF 10X,B 111,38X,2V,3X       ; SPECIAL WRITE
     LDST:     DEF 10X,B 100,43X             ; LOAD STATUS REG
     ENCN:     DEF 10X,B 101,34X,4V,5X       ; ENABLE CONTROL DECODER
     LDAER:    DEF 10X,B 110,43X             ; LOAD ADDR EXTENTION REG
     ;
     ;       SPWR OPERANDS
     ;
     LDIM1:    EQU B 01                 ; LOAD INTERRUPT MASK REG
     MAPWR:    EQU B 10                 ; LOAD MAP DATA OUT REG / MAP WRITE
     LEDWR:    EQU B 11                 ; LOAD LED REGISTER
     ;
```

```
LINE            STATEMENT

 240  ;        ENCN OPERANDS
 241  ;
 242  CNNOP:     EQU B 0000       ;NO OPERATION
 243  CLRPEI:    EQU B 0001       ;CLEAR PENDING PARITY ERROR INTERRUPT
 244  CLRMPI:    EQU B 0010       ;CLEAR PENDING MEMORY PROTECT INTERRUPT
 245  SETMPI:    EQU B 0011       ;GENERATE A PENDING MEMORY PROTECT INTERRUPT
 246  CLRTBT:    EQU B 0100       ;CLEAR PENDING TIME BASE TICK INTERRUPT
 247  SETTBT:    EQU B 0101       ;SET A PENDING TIME BASE TICK INTERRUPT
 248  CLRPFWI:   EQU B 0110       ;CLEAR PENDING POWER FAIL WARNING INTERRUPT
 249  ICRS:      EQU B 0111       ;GENERATE A CRS
 250  CLRTDI:    EQU B 1000       ;TURN OFF TEMPORARY INTERRUPT DISABLE
 251  SETTDI:    EQU B 1001       ;TURN ON TEMPORARY INTERRUPT DISABLE
 252  CLRDTST:   EQU B 1010       :TURN OFF DATA BUS TEST LOOP BACK
 253  SETDTST:   EQU B 1011       ;TURN ON DATA BUS TEST LOOP BACK
 254  CLRPSFF:   EQU B 1100       ;TURN OFF PARITY SYSTEM FLIP-FLOP
 255  SETPSFF:   EQU B 1101       ;TURN ON PARITY SYSTEM FLIP-FLOP
 256  CLRMPEN:   EQU B 1110       ;TURN OFF MEMORY PROTECT SYSTEM
 257  SETMPEN:   EQU B 1111       ;TURN ON MEMORY PROTECT SYSTEM
 258  ;
 259  END

  TOTAL DEFINITION ERRORS =    0
```

SYMBOL TABLE

A	A	00000	AB	A	00001	ADD	A	00000	AM2901	D	
AND	A	00004	AQ	A	00000	B	A	00001	C	A	0000B
CAB	A	00010	CALL	D		CARRYEXT	D		CARRYH	D	
CARRYL	D		CARRYNUC	D		CARRYREG	D		CARRYUC	D	
CCALL	D		CJP	D		CJPP	D		CLRDTST	A	0000A
CLRMPEN	A	0000E	CLRMPI	A	00002	CLRPEI	A	00001	CLRPFWI	A	00006
CLRPSFF	A	0000C	CLRTBT	A	00004	CLRTDI	A	00008	CNNOP	A	00000
CONDEXT	D		CONDLMSR	D		CONDMSR	D		CONDUSR	D	
CONT	D		CPUSH	D		CRET	D		CVECT	D	
CXY	A	00012	DA	A	00005	DIV	A	00014	DIVCOND	D	
DIVUCOND	D		DQ	A	00006	DZ	A	00007	ECIRRD	A	00005
ENBLC	D		ENBLO	D		ENCN	D		ENVE	D	
EXNOR	A	00007	EXOR	A	00006	FILLER	D		ICRS	A	00007
IFETCH	D		IMM	D		IMMB	D		INTRPT	A	00007
INVMSR	D		IORD	D		IORQ	A	00006	IOWR	D	
IROT3	A	00008	IR11	A	00002	IRMRG	D		IRSKIP	A	00000
JMAP	D		JP	D		JPP	D		JRP	D	
JTAB	D		JZ	D		L4D	D		LDAER	D	
LDCT	D		LDIM1	A	00001	LDMAPD	D		LDMAR	D	
LDMDOR	D		LDST	D		LEDWR	A	00003	LODMSR	D	
LODMSRCI	D		LODUSR	D		LODUSRCI	D		LODUSROR	D	
LOOP	D		LOWSC	A	00004	LT	A	00003	LVL0	A	00000
MAPD	A	0000D	MAPRD	A	00001	MAPWR	A	00002	MAPX	A	0000E
MIAK	D		MKLROFF	D		MKLRON	D		MPEN	A	00005
MPY	A	00013	MPY4	A	00017	MREAD	D		MWRITE	D	
NARG	A	00000	NC	A	0000A	NOP	A	00001	NOTRS	A	00005
NOVR	A	00006	NSIGN	A	0000E	NUCLDMSR	D		NZ	A	00004
OR	A	00003	OVR	A	00007	PASS	A	00003	PC	A	0000F
PELENH	A	00003	PELENL	A	00002	PORM	A	0001F	PRLEN	A	00004
PUSH	D		QPEI	A	00003	QREG	A	00000	R0	A	00000
R1	A	00001	R10	A	0000A	R11	A	0000B	R12	A	0000C
R13	A	0000D	R14	A	0000E	R15	A	0000F	R2	A	00002
R3	A	00003	R4	A	00004	R5	A	00005	R6	A	00006
R7	A	00007	R8	A	00008	R9	A	00009	RAMA	A	00002
RAMF	A	00003	RET	D		RFCT	D		RFETCH	D	
ROTATE	A	0000A	ROTATEC	A	00009	RPCT	D		RSTMSR	D	
SETDTST	A	0000B	SETMPEN	A	0000F	SETMPI	A	00003	SETMSR	D	
SETPSFF	A	0000D	SETTBT	A	00005	SETTDI	A	00009	SGNXOVR	A	00002
SHIFT	D		SIGN	A	0000F	SINTRQ	A	00004	SLACK	A	00007
SPFNOP	D		SPHNOP	D		SPRD	D		SPWR	D	
SRAML	A	00007	SRAMQL	A	00006	SRAMQR	A	00004	SRAMR	A	00005
SRG1	A	00009	SRG2	A	0000A	STRD	D		SUBR	A	00001
SUBS	A	00002	SWAPEO	D		SWRD	A	00006	TAB	A	00011
TWB	D		UCLDMSR	D		ULE	A	0000D	WORDCNT	A	0000B
X	A	00002	Y	A	00003	Z	A	00005	ZA	A	00004
ZB	A	00003	ZQ	A	00002						

```
LINE    ADDR    STATEMENT

                TITLE       A600 BASESET MICROCODE 06/14/82 *A [&LBPRM]
                NOLIST  L
                LIST    B,E
                ;
                ; &LBPRM 12101-18002 REV 2226 820524
                ;
                ; .FDIV FIXED
                ;
                NEWPC:    EQU   1          ; NON-ZERO FOR NEW PC BOARDS
                FILL:     EQU   1          ; NON-ZERO FOR FILL EMPTY AREAS
                ;
                ;*************************************************************
                ;**                                                         **
                ;**        A600 Baseset Microcode                           **
                ;**                                                         **
                ;** Assumes:                                                **
                ;**- 2 cycle memory                                         **
                ;**- Target of memory write can be read first               **
                ;**- Target of inst fetch can be read first                 **
                ;**                                                         **
                ;** TDI:                                                    **
                ;**- Temporary Disable Interrupt is set to inhibit          **
                ;**  interrupts at the conclusion of JMP.I, JSB.I or I/O   **
                ;**  instructions so the next instruction is executed.      **
                ;**- The ENCN SETTDI order must be given at least one       **
                ;**  cycle before IFETCH.  TDI is automatically cleared     **
                ;**  by JTAB LVLO.                                          **
                ;**                                                         **
                ;** CARRY:                                                  **
                ;**- The Cn input to the ALU is XOR'D with I3.  Thus        **
                ;**  the sense of carry is inverted for SUBR, OR, NOTRS     **
                ;**  and EXNOR.                                             **
                ;**                                                         **
                ;** TAB Special:                                            **
                ;**- In A reg field, modifies source operand of DZ to ZA    **
                ;**  if macro A or B register was addressed on last MREAD   **
                ;**  or IFETCH (MRGREAD or MRGFETCH).  Note that when TAB  **
                ;**  is used, only ONE operand can be sent to the ALU.      **
                ;**  I.E.  AB can not be coded as a source operand select   **
                ;**                                                         **
                ;**- In B reg field, modifies destination operand of NOP    **
                ;**  to RAMF if 0 or 1 is in MAR.  Normally, MAR is loaded **
                ;**  before MWRITE is issued.                               **
                ;**                                                         **
                ;*************************************************************
                ;
```

```
LINE    ADDR    STATEMENT

                ;************************************************************
                ;**                                                        **
                ;**       EQUATES                                          **
                ;**                                                        **
                ;************************************************************
                ;
                IMAPLOC:  EQU   H 0003    ; BOOT MEMORY LOCATION OF IMAP REG
                VMALOC:   EQU   H 0004    ; VMA FAULT ROUTINE ADDRESS
                VMAPTE:   EQU   H 0002    ; VMA PAGE TABLE POINTER LOCATION
                ;
                CPUID:    EQU   H 0002    ; A600 PROCESSOR ID NUMBER
                MICREVID: EQU   H 0800    ; MICRO CODE REVISION NUMBER
                ;
                ;         UN-COMMENT THE FOLLOWING "NOLIST" TO LIST ONLY VMA
                ;
                ;NOLIST
                ;
```

```
ADDR    STATEMENT

        ;
        ;****************************************************************
        ;*                                                              *
        ;*          Basic Instruction Decode                            *
        ;*          ------------------------                            *
        ;*                                                              *
        ;* - At the FETCH line, PC points to Next instruction.          *
        ;*                                                              *
        ;* - After instruction mapping, MAR points to Next              *
        ;*   instruction and PC points to Next instruction + 1.         *
        ;*                                                              *
        ;* - First microinstruction:                                    *
        ;*       - Wait for previous instruction read to finish         *
        ;*       - Load MAR and scratch reg 12 with MRG C/Z address     *
        ;*       - Start read if MRG read inst or MRG Indirect          *
        ;*       - Save MRG address in scratch register 12.             *
        ;*       - Load 2910 counter reg with microaddress of           *
        ;*         instruction microcode routine.                       *
        ;*                                                              *
        ;* - Second microinstruction:                                   *
        ;*     - Increments PC                                          *
        ;*     - IF ISZ, ST* or JSB THEN load MAR with MRG address      *
        ;*       ELSE load MAR with next instruction address or         *
        ;*       instruction address word address.                      *
        ;*     - IF interrupt pending THEN jump to Interrupt handler    *
        ;*       ELSE jump to instruction microcode routine.            *
        ;*                                                              *
        ;****************************************************************
        ;
        FETCH:   LDCT                & AM2901 ,R12,RAMF,PASS,DZ
        /                            & JTAB LVLO
        /                            & IRMRG  - MRG READ SPECIAL
00000   /                            & LDMAR ;
                 JRP    INTERRPT     & AM2901 PORM,PC,RAMA,ADD,ZB
        /                            & CONDEXT INTRPT
        /                            & CARRYH
        /                            & SPFNOP
00001   /                            & LDMAR;
```

```
ADDR     STATEMENT

         ;*************************************************************
         ;*                                                           *
         ;*         Memory Reference Group                            *
         ;*         ----------------------                            *
         ;*                                                           *
         ;*                                                           *
         ;*    Assumes: - Result of memory read ends up in Q reg      *
         ;*                                                           *
         ;*************************************************************
         ;
         ;         ADD
         ;
         AD.I::    CALL  MRGIND     & AM2901 TAB,,QREG,PASS,DZ
         /                          & LDMAR
00002    /                          & MREAD;
         AD.::     CONT             & AM2901 TAB,,QREG,PASS,DZ
         /                          & SPHNOP
00003    /                          & IFETCH;
                   JZ               & AM2901 CAB,CAB,RAMF,ADD,AQ
         /                          & CARRYL
         /                          & LODMSR & ENBLC & ENBLO
         /                          & SPFNOP
00004    /                          & ENVE ;
         ;
         ;         LOGICAL AND
         ;
         IANDI::   CALL  MRGIND     & AM2901 TAB,,QREG,PASS,DZ
         /                          & LDMAR
00005    /                          & MREAD;
         IAND::    CONT             & AM2901 TAB,,QREG,PASS,DZ
         /                          & SPHNOP
00006    /                          & IFETCH;
                   JZ               & AM2901 A,A,RAMF,AND,AQ
         /                          & SPFNOP
00007    /                          & SPHNOP ;
         ;
         ;         LOGICAL COMPARE
         ;
         ;     If the two operands are not identical, skip the
         ; next instruction.  Note the PC points to the next
         ; instruction already.
         ;
         CP.I:     CALL  MRGIND     & AM2901 TAB,,QREG,PASS,DZ
         /                          & LDMAR
00008    /                          & MREAD;
         CP.:      CONT             & AM2901 TAB,R4,RAMF,PASS,DZ
         /                          & SPHNOP
00009    /                          & IFETCH;
         CP.1:     CONT             & AM2901 CAB,R4,NOP,EXOR,AB
```

```
LINE    ADDR    STATEMENT

 154            /                            & LODUSR
 155            /                            & SPFNOP
 156    0000A   /                            & SPHNOP ;
 157                      CJP   FETCH        & AM2901 ,,NOP,PASS,ZQ
 158            /                            & CONDUSR Z
 159            /                            & SPFNOP
 160    0000B   /                            & SPHNOP ;
 161                      JZ                 & AM2901 PC,PC,RAMA,ADD,ZB
 162            /                            & CARRYH
 163            /                            & LDMAR
 164    0000C   /                            & IFETCH;
 165            ;
 166            ;         LOGICAL OR
 167            ;
 168            IORI::    CALL  MRGIND       & AM2901 TAB,,QREG,PASS,DZ
 169            /                            & LDMAR
 170    0000D   /                            & MREAD;
 171            IOR::     CONT               & AM2901 TAB,,QREG,PASS,DZ
 172            /                            & SPHNOP
 173    0000E   /                            & IFETCH;
 174                      JZ                 & AM2901 A,A,RAMF,OR,AQ
 175            /                            & SPFNOP
 176    0000F   /                            & SPHNOP ;
 177            ;
 178            ;         INCREMENT AND SKIP IF ZERO
 179            ;
 180            ISZI::    CALL  MWRTIND      & AM2901 TAB,,QREG,PASS,DZ
 181            /                            & LDMAR
 182    00010   /                            & MREAD;
 183            ISZ::     CONT               & AM2901 TAB,TAB,NOP,ADD,DZ
 184            /                            & CARRYH
 185            /                            & LODUSR
 186            /                            & LDMDOR
 187    00011   /                            & MWRITE ;
 188            ;
 189            ;   Increment of OFFFFH to 0 will produce a carry out.
 190            ; Thus, if carry out, then SKIP.  Since PC points to Next
 191            ; instruction + 1, PC points to target of SKIP.
 192            ; Algorithm is subtract NOT carry out from PC and fetch.
 193            ;
 194                      CONT               & AM2901 ,PC,RAMF,SUBR,ZB
 195            /                            & CARRYNUC
 196            /                            & SPFNOP
 197    00012   /                            & SPHNOP ;
 198                      JZ                 & AM2901 PC,PC,RAMA,ADD,ZB
 199            /                            & CARRYH
 200            /                            & LDMAR
 201    00013   /                            & IFETCH;
 202            ;
```

```
LINE   ADDR    STATEMENT

               ;          JMP Indirect
               ;
               ;      -It is assumed that the target of the JMP indirect
               ; can be read before it is fetched.  This speeds up the
               ; indirect resolution.
               ;
               JMPI::     CALL  MWRTIND     & AM2901 TAB,,QREG,PASS,DZ
               /                            & LDMAR
       00014   /                            & MREAD ;
                          CONT              & AM2901 ,PC,RAMF,PASS,ZQ
               /                            & ENCN SETTDI
       00015   /                            & SPFNOP ;    MAR POINTS TO TARGET
                          JZ                & AM2901 ,PC,RAMF,ADD,ZB
               /                            & CARRYH
               /                            & SPHNOP
       00016   /                            & IFETCH  ;       FETCH NEXT INST
               ;
               ;          JMP Direct
               ;
               JMP::      JZ                & AM2901 R12,PC,RAMF,ADD,ZA
               /                            & SPFNOP
               /                            & SPHNOP
       00017   /                            & CARRYH ;                 INC PC
               ;
               ;          JSB   Direct/Indirect
               ;
               ;  -Since PC points to Next instruction + 1, PC-1 must
               ;   be stored at JSB target.
               ;
               JSBI:      CALL  MWRTIND     & AM2901 TAB,,QREG,PASS,DZ
               /                            & LDMAR
       00018   /                            & MREAD ;
               JSBIVMA:   CONT              & AM2901 PC,TAB,NOP,SUBR,ZA
               /                            & CARRYH
               /                            & LDMDOR
       00019   /                            & MWRITE ;  WRITE PC TO JSB TARGET
                          CONT              & AM2901 ,PC,RAMF,ADD,ZQ
               /                            & CARRYH
               /                            & ENCN SETTDI
       0001A   /                            & SPFNOP ;  SET PC TO JSB TARGET +
                          JZ                & AM2901 PC,PC,RAMA,ADD,ZB
               /                            & CARRYH
               /                            & LDMAR
       0001B   /                            & IFETCH ;  FETCH NEXT INSTRUCTION
               ;
               JSB:       CONT              & AM2901 PC,TAB,NOP,SUBR,ZA
               /                            & CARRYH
               /                            & LDMDOR
       0001C   /                            & MWRITE;
```

```
LINE   ADDR    STATEMENT

252                      CONT               & AM2901 R12,PC,RAMF,ADD,ZA
253            /                            & CARRYH
254            /                            & SPFNOP
255    0001D   /                            & SPHNOP ;
256                      JZ                 & AM2901 PC,PC,RAMA,ADD,ZB
257            /                            & CARRYH
258            /                            & IFETCH
259    0001E   /                            & LDMAR;
260            ;
261            ;         LOAD
262            ;
263            LD.I::    CALL  MRGIND       & AM2901 TAB,,QREG,PASS,DZ
264            /                            & LDMAR
265    0001F   /                            & MREAD;
266            LD.::     JZ                 & AM2901 TAB,CAB,RAMF,PASS,DZ
267            /                            & SPHNOP
268    00020   /                            & IFETCH;
269            ;
270            ;         STORE
271            ;
272            ST.I::    CALL  MWRTIND      & AM2901 TAB,,QREG,PASS,DZ
273            /                            & LODUSR
274            /                            & LDMAR
275    00021   /                            & MREAD ;
276            ST.::     CONT               & AM2901 CAB,TAB,NOP,PASS,ZA
277            /                            & LDMDOR
278    00022   /                            & MWRITE ;
279                      CONT               & AM2901 ,PC,NOP,SUBR,ZB
280            /                            & CARRYH
281            /                            & SPFNOP
282    00023   /                            & LDMAR ;
283                      JZ                 & AM2901 ,,NOP,PASS,ZQ
284            /                            & SPHNOP
285    00024   /                            & IFETCH ;
286            ;
287            ;         LOGICAL EXCLUSIVE OR
288            ;
289            XORI::    CALL  MRGIND       & AM2901 TAB,,QREG,PASS,DZ
290            /                            & LDMAR
291    00025   /                            & MREAD;
292            XOR::     CONT               & AM2901 TAB,,QREG,PASS,DZ
293            /                            & SPHNOP
294    00026   /                            & IFETCH;
295                      JZ                 & AM2901 A,A,RAMF,EXOR,AQ
296            /                            & SPFNOP
297    00027   /                            & SPHNOP ;
```

```
ADDR    STATEMENT

        ;***************************************************************
        ;*                                                             *
        ;*         Alter-Skip Group                                    *
        ;*         ----------------                                    *
        ;*                                                             *
        ;*  -ASG is mapped 16 ways based on CL*/CC*/CM*/NOP            *
        ;*   and CCE/CLE/CME/NOP.                                      *
        ;*                                                             *
        ;*  - Accumulator ops: CL* - Clears A/B to 0000H               *
        ;*                     CC* - Sets A/B to 0FFFFH                *
        ;*                     CM* - Inverts A/B (One's Complement)    *
        ;*                                                             *
        ;*  - E-reg ops:       CLE - Clears E reg                      *
        ;*                     CCE - Sets E reg                        *
        ;*                     CME - Inverts E reg                     *
        ;*                                                             *
        ;*  - Algorithm:                                               *
        ;*                                                             *
        ;*      CL* - And register with zero with carry-in             *
        ;*            LOW.   Cn+4 will be 0.                           *
        ;*      CC* - Subtract register from itself with carry-in      *
        ;*            LOW.   Cn+4 will be 0.                           *
        ;*      CM* - Subtract register from zero with carry-in        *
        ;*            LOW.   Cn+4 will be 0.                           *
        ;*                                                             *
        ;***************************************************************
        ;
        ASCLANOP: JP    ASCONT      & AM2901 ,CAB,RAMF,AND,ZB
        /                           & SPFNOP
        /                           & SPHNOP
        /                           & CARRYL
00028   /                           & LODMSR ;          LOAD YC BUFFER
        ;
        ASCMANOP: JP    ASCONT      & AM2901 ,CAB,RAMF,SUBS,ZB
        /                           & SPFNOP
        /                           & SPHNOP
        /                           & CARRYL
00029   /                           & LODMSR ;          LOAD YC BUFFER
        ;
        ASCCANOP: CONT              & AM2901 CAB,CAB,RAMF,SUBS,AB
        /                           & SPFNOP
        /                           & SPHNOP
        /                           & CARRYL
0002A   /                           & LODMSR ;          LOAD YC BUFFER
        ;
        ASCONT:   LDCT  FETCH       & AM2901 CAB,CAB,RAMA,ADD,ZB
        /                           & LODMSR & ENVE & ENBLC & ENBLO
        /                           & CARRYEXT
```

```
LINE    ADDR    STATEMENT

        0002B   /                               & IFETCH ;
                          JRP   SKIP            & AM2901 ,,NOP,PASS,ZQ
                /                               & CONDEXT IRSKIP
                /                               & SPFNOP
        0002C   /                               & SPHNOP ;
                ;
                ;         Decode CLE - Reset Mc of 2904
                ;
                ;     Method is to Load Machine Status Register bit Mc
                ; with state of Ic.  Since Cn+4 is zero, a load with Cn+4
                ; will clear Mc bit.
                ;
                ASCLACLE: JP    ASCONT          & AM2901 ,CAB,RAMF,AND,ZB
                /                               & SPFNOP
                /                               & SPHNOP
                /                               & CARRYL
        0002D   /                               & LODMSR & ENBLC ;    CN+4 IS ZERO
                ;
                ASCMACLE: JP    ASCONT          & AM2901 ,CAB,RAMF,SUBS,ZB
                /                               & SPFNOP
                /                               & SPHNOP
                /                               & CARRYL
        0002E   /                               & LODMSR & ENBLC ;    CN+4 IS ZERO
                ;
                ASCCACLE: JP    ASCONT          & AM2901 CAB,CAB,RAMF,SUBS,AB
                /                               & SPFNOP
                /                               & SPHNOP
                /                               & CARRYL
        0002F   /                               & LODMSR & ENBLC ;    CN+4 IS ZERO
                ;
                ASNOPCLE: JP    ASCONT          & AM2901 ,CAB,NOP,ADD,ZB
                /                               & SPFNOP
                /                               & SPHNOP
                /                               & CARRYL
        00030   /                               & LODMSR & ENBLC ;    CN+4 IS ZERO
                ;
                ;         Decode CCE - Set 2904 Mc bit
                ;
                ;     Method is to use Load Machine Status Register with
                ; Carry Invert.  Since the state of Cn+4 is known to be zero,
                ; a load with carry invert will set the 2904 Mc bit.
                ;
                ASCLACCE: JP    ASCONT          & AM2901 ,CAB,RAMF,AND,ZB
                /                               & SPFNOP
                /                               & SPHNOP
                /                               & CARRYL
        00031   /                               & LODMSRCI & ENBLC ;  CN+4 IS ZERO
                ;
```

```
ADDR    STATEMENT

        ASCMACCE: JP    ASCONT      & AM2901 ,CAB,RAMF,SUBS,ZB
        /                           & SPFNOP
        /                           & SPHNOP
        /                           & CARRYL
00032   /                           & LODMSRCI & ENBLC ;  CN+4 IS ZERO
        ;
        ASCCACCE: JP    ASCONT      & AM2901 CAB,CAB,RAMF,SUBS,AB
        /                           & SPFNOP
        /                           & SPHNOP
        /                           & CARRYL
00033   /                           & LODMSRCI & ENBLC ;  CN+4 IS ZERO
        ;
        ASNOPCCE: JP    ASCONT      & AM2901 ,CAB,NOP,ADD,ZB
        /                           & SPFNOP
        /                           & SPHNOP
        /                           & CARRYL
00034   /                           & LODMSRCI & ENBLC ;  CN+4 IS ZERO
        ;
        ;         Decode CME - Invert 2904 Mc bit
        ;
        ;     Method is to use Load Machine Status Register with
        ; Carry Invert, while asserting ENVE to gate current state
        ; of Mc into Ic.  If Cn+4 is zero, then Mc is complemented.
        ;
        ASCLACME: JP    ASCONT      & AM2901 ,CAB,RAMF,AND,ZB
        /                           & SPFNOP
        /                           & CARRYL
00035   /                           & LODMSRCI & ENBLC & ENVE ;  CN+4
                                                                IS ZERO
        ;
        ASCMACME: JP    ASCONT      & AM2901 ,CAB,RAMF,SUBS,ZB
        /                           & SPFNOP
        /                           & CARRYL
00036   /                           & LODMSRCI & ENBLC & ENVE ;  CN+4
                                                                IS ZERO
        ;
        ASCCACME: JP    ASCONT      & AM2901 CAB,CAB,RAMF,SUBS,AB
        /                           & SPFNOP
        /                           & CARRYL
00037   /                           & LODMSRCI & ENBLC & ENVE ;  CN+4
                                                                IS ZERO
        ;
        ASNOPCME: JP    ASCONT      & AM2901 ,CAB,NOP,ADD,ZB
        /                           & SPFNOP
        /                           & CARRYL
00038   /                           & LODMSRCI & ENBLC & ENVE ;  CN+4
                                                                IS ZERO
        ;
```

```
ADDR    STATEMENT

        ;***********************************************************
        ;*                                                         *
        ;*       Shift Rotate Group                                *
        ;*       ------------------                                *
        ;*                                                         *
        ;*  The initial instruction decode jumps to the SRGO entry *
        ;*  points.                                                *
        ;*                                                         *
        ;*  Specials:                                              *
        ;*                                                         *
        ;*  SRG1 - Maps bottom 6 bits of IR to control store address*
        ;*       - Decodes CLE, SL*, and NOP                       *
        ;*       - If no CLE or SL*, then SRG1 same as SRG2        *
        ;*                                                         *
        ;*  SRG2 - Maps bottom 6 bits of IR to control store address*
        ;*       - Decodes second Shift/Rotate operation:          *
        ;*         ALS,ARS,RAL,RAR,ALR,ERA,ELA,ALF,ELAD,ERAD       *
        ;*       - NOP goes directly to fetch line                 *
        ;*                                                         *
        ;***********************************************************
        ;
        ;       SRG NOP
        ;
        ;    - NO FIRST SHIFT/ROTATE OPERATION
        ;
        SRGONOP:  JMAP              & AM2901 ,,NOP,PASS,ZQ
        /                           & SPFNOP
        /                           & SPHNOP
00039   /                           & JTAB SRG1;
        ;
        ;       ARITH LEFT SHIFT
        ;
        ;    - SET SIGN AND LEFT SHIFT
        ;    - LEFT SHIFT AGAIN
        ;    - RIGHT SHIFT WITH SIGN
        ;
        SRGOALS:  CONT              & AM2901 ,CAB,SRAML,PASS,ZB
        /                           & LODMSR
        /                           & SPFNOP
        /                           & SPHNOP
0003A   /                           & SHIFT B 0010 ;
                  CONT              & AM2901 ,CAB,SRAML,PASS,ZB
        /                           & SPFNOP
        /                           & SPHNOP
0003B   /                           & SHIFT B 0010 ;
                  JMAP              & AM2901 ,CAB,SRAMR,PASS,ZB
        /                           & SPFNOP
        /                           & SPHNOP
```

```
ADDR    STATEMENT

        /                               & JTAB SRG1
0003C   /                               & SHIFT B 0101 ;
        ;
        ;          ARITH RIGHT SHIFT
        ;
        ;    - SET SIGN
        ;    - RIGHT SHIFT WITH SIGN
        ;
        SRGOARS:   CONT                 & AM2901 ,CAB,NOP,PASS,ZB
        /                               & LODMSR
        /                               & SPFNOP
0003D   /                               & SPHNOP ;
                   JMAP                 & AM2901 ,CAB,SRAMR,PASS,ZB
        /                               & SPFNOP
        /                               & SPHNOP
        /                               & JTAB SRG1
0003E   /                               & SHIFT  B 0101 ;
        ;
        ;          ROTATE LEFT
        ;
        SRGORAL:   JMAP                 & AM2901 ,CAB,SRAML,PASS,ZB
        /                               & SPFNOP
        /                               & SPHNOP
        /                               & JTAB SRG1
0003F   /                               & SHIFT ROTATE ;
        ;
        ;          ROTATE RIGHT
        ;
        SRGORAR:   JMAP                 & AM2901 ,CAB,SRAMR,PASS,ZB
        /                               & SPFNOP
        /                               & SPHNOP
        /                               & JTAB SRG1
00040   /                               & SHIFT ROTATE ;
        ;
        ;          LEFT SHIFT ONE, CLEAR SIGN
        ;
        ;   - MASK OFF BIT 14 AND 15
        ;   - SHIFT LEFT NORMAL
        ;
        SRGOALR:   CONT                 & AM2901 ,,QREG,PASS,DZ
        /                               & SPHNOP
00041   /                               & IMM H 3FFF ;
                   JMAP                 & AM2901 CAB,CAB,SRAML,AND,AQ
        /                               & SPFNOP
        /                               & SPHNOP
        /                               & JTAB SRG1
00042   /                               & SHIFT B 0010 ;
        ;
```

```
ADDR     STATEMENT

         ;          ROTATE RIGHT WITH E
         ;
         SRGOERA:  JMAP                & AM2901 ,CAB,SRAMR,PASS,ZB
         /                             & SPFNOP
         /                             & SPHNOP
         /                             & JTAB SRG1
00043    /                             & SHIFT ROTATEC ;
         ;
         ;          ROTATE LEFT WITH E
         ;
         SRGOELA:  JMAP                & AM2901 ,CAB,SRAML,PASS,ZB
         /                             & SPFNOP
         /                             & SPHNOP
         /                             & JTAB SRG1
00044    /                             & SHIFT ROTATEC ;
         ;
         ;          ROTATE LEFT 4
         ;
         ;     - LOAD EXTERNAL ROTATE BY 4 REG AND THEN READ IT BACK
         ;
         SRGOALF:  JMAP                & AM2901 CAB,CAB,RAMA,PASS,DZ
         /                             & SPHNOP
         /                             & JTAB SRG1
00045    /                             & L4D ;
         ;
         ;          COPY SIGN INTO E
         ;
         ;     - SAME AS ROTATE LEFT WITH E EXCEPT SCRATCH DESTINATION
         ;
         SRGOELAD: JMAP                & AM2901 CAB,R4,SRAML,PASS,ZA
         /                             & SPFNOP
         /                             & SPHNOP
         /                             & JTAB SRG1
00046    /                             & SHIFT ROTATEC ;
         ;
         ;          COPY LSB INTO E
         ;
         ;     - SAME AS ROTATE RIGHT WITH E EXCEPT SCRATCH DESTINATION
         ;
         SRGOERAD: JMAP                & AM2901 CAB,R4,SRAMR,PASS,ZA
         /                             & SPFNOP
         /                             & SPHNOP
         /                             & JTAB SRG1
00047    /                             & SHIFT ROTATEC ;
         ;
```

```
ADDR     STATEMENT

         ;         PERFORM SRG CLE/SL* OPERATION
         ;
         SR1CLE:   JMAP             & AM2901 ,,NOP,PASS,ZQ
         /                          & SPFNOP
         /                          & SPHNOP
         /                          & RSTMSR & ENBLC
00048    /                          & JTAB SRG2;
         ;
         SR1SL:    LDCT             & AM2901 ,CAB,NOP,PASS,ZB
         /                          & SPFNOP
         /                          & SPHNOP
00049    /                          & JTAB SRG2 ;
                   JRP   SRGSKIP    & AM2901 ,,NOP,PASS,ZQ
         /                          & SPFNOP
         /                          & SPHNOP
0004A    /                          & CONDEXT IRSKIP ;
         ;
         SR1CLESL: LDCT             & AM2901 ,CAB,NOP,PASS,ZB
         /                          & SPFNOP
         /                          & SPHNOP
         /                          & RSTMSR & ENBLC
0004B    /                          & JTAB SRG2 ;
                   JRP   SRGSKIP    & AM2901 ,,NOP,PASS,ZQ
         /                          & SPFNOP
         /                          & SPHNOP
0004C    /                          & CONDEXT IRSKIP ;
         ;
         SRGSKIP:  JMAP             & AM2901 PC,PC,RAMA,ADD,ZB
         /                          & CARRYH
         /                          & JTAB SRG2
         /                          & SPFNOP
0004D    /                          & LDMAR;
         ;
         ;         PERFORM SECOND SRG OPERATION
         ;
         SRG2NOP:  JZ               & AM2901 ,,NOP,PASS,ZQ
         /                          & SPHNOP
0004E    /                          & IFETCH ;
         SRG2ALS:  CONT             & AM2901 ,CAB,SRAML,PASS,ZB
         /                          & LODMSR
         /                          & SPFNOP
         /                          & SPHNOP
0004F    /                          & SHIFT B 0010 ;
                   CONT             & AM2901 ,CAB,SRAML,PASS,ZB
         /                          & SPHNOP
         /                          & IFETCH
00050    /                          & SHIFT B 0010 ;
                   JZ               & AM2901 ,CAB,SRAMR,PASS,ZB
         /                          & SPFNOP
```

```
ADDR    STATEMENT

        /                        & SPHNOP
00051   /                        & SHIFT B 0101 ;
        SRG2ARS:  CONT           & AM2901 ,CAB,NOP,PASS,ZB
        /                        & LODMSR
        /                        & SPHNOP
00052   /                        & IFETCH ;
                  JZ             & AM2901 ,CAB,SRAMR,PASS,ZB
        /                        & SPFNOP
        /                        & SPHNOP
00053   /                        & SHIFT  B 0101 ;
        SRG2RAL:  JZ             & AM2901 ,CAB,SRAML,PASS,ZB
        /                        & IFETCH
        /                        & SPHNOP
00054   /                        & SHIFT ROTATE ;
        SRG2RAR:  JZ             & AM2901 ,CAB,SRAMR,PASS,ZB
        /                        & SPHNOP
        /                        & IFETCH
00055   /                        & SHIFT ROTATE ;
        SRG2ALR:  CONT           & AM2901 ,,QREG,PASS,DZ
        /                        & SPHNOP
00056   /                        & IMM H 3FFF ;
                  JZ             & AM2901 CAB,CAB,SRAML,AND,AQ
        /                        & SPHNOP
        /                        & IFETCH
00057   /                        & SHIFT B 0010 ;
        SRG2ERA:  JZ             & AM2901 ,CAB,SRAMR,PASS,ZB
        /                        & SPHNOP
        /                        & IFETCH
00058   /                        & SHIFT ROTATEC ;
        SRG2ELA:  JZ             & AM2901 ,CAB,SRAML,PASS,ZB
        /                        & SPHNOP
        /                        & IFETCH
00059   /                        & SHIFT ROTATEC ;
        SRG2ALF:  CONT           & AM2901,,NOP,PASS,ZQ
        /                        & SPHNOP
0005A   /                        & IFETCH ;
                  JZ             & AM2901 CAB,CAB,RAMA,PASS,DZ
        /                        & SPHNOP
0005B   /                        & L4D ;
        SRG2ELAD: JZ             & AM2901 CAB,R4,SRAML,PASS,ZA
        /                        & SPHNOP
        /                        & IFETCH
0005C   /                        & SHIFT ROTATEC ;
        SRG2ERAD: JZ             & AM2901 CAB,R4,SRAMR,PASS,ZA
        /                        & SPHNOP
        /                        & IFETCH
0005D   /                        & SHIFT ROTATEC ;
```

```
ADDR    STATEMENT

        ;************************************************************
        ;*                                                          *
        ;*        Extended Arithmetic Register Group                *
        ;*        ----------------------------------                *
        ;*                                                          *
        ;************************************************************
        ;
        ;         ARITH SHIFT LEFT
        ;
        ASL:      JP   ASLCONT     & AM2901 ,,NOP,PASS,ZQ
        /                          & SPFNOP
0005E   /                          & SPHNOP ;     JUMP TO ACTUAL CODE
        ;
        ;         JUMP AND LOAD A/B
        ;
        JL.:      CALL  WRTIND     & AM2901 PC,CAB,RAMF,PASS,ZA
        /                          & MREAD
0005F   /                          & SPHNOP ;           A/B := PC
                  JP    SKIP       & AM2901 ,PC,RAMF,PASS,ZQ
        /                          & SPFNOP
00060   /                          & SPHNOP ;
        ;
        ;         ARITH SHIFT RIGHT
        ;
        ASR:      CONT             & AM2901 ,B,NOP,PASS,ZB
        /                          & SPFNOP
        /                          & SPHNOP
00061   /                          & LODMSR ;                SET SIGN
                  PUSH             & AM2901 ,A,QREG,PASS,ZB
        /                          & SPHNOP
        /                          & IFETCH
00062   /                          & JTAB IROT3  ;
                  RFCT             & AM2901 ,B,SRAMQR,PASS,ZB
        /                          & SPFNOP
        /                          & SPHNOP
00063   /                          & SHIFT B 0101 ;
                  JZ               & AM2901 ,A,RAMF,PASS,ZQ
        /                          & SPFNOP
        /                          & SPHNOP
00064   /                          & RSTMSR & ENBLO ;  CLEAR OVERFLOW
        ;
        ;         LOGICAL SHIFT LEFT
        ;
        LSL:      PUSH             & AM2901 ,A,QREG,PASS,ZB
        /                          & SPHNOP
        /                          & IFETCH
00065   /                          & JTAB IROT3 ;
                  RFCT             & AM2901 ,B,SRAMQL,PASS,ZB
```

```
ADDR    STATEMENT

        /                        & SPFNOP
        /                        & SPHNOP
00066   /                        & SHIFT B 0110 ;
                JZ               & AM2901 ,A,RAMF,PASS,ZQ
        /                        & SPFNOP
00067   /                        & SPHNOP ;
        ;
        ;       LOGICAL SHIFT RIGHT
        ;
        LSR:    PUSH             & AM2901 ,A,QREG,PASS,ZB
        /                        & SPHNOP
        /                        & IFETCH
00068   /                        & JTAB IROT3 ;
                RFCT             & AM2901 ,B,SRAMQR,PASS,ZB
        /                        & SPFNOP
        /                        & SPHNOP
00069   /                        & SHIFT B 0110 ;
                JZ               & AM2901 ,A,RAMF,PASS,ZQ
        /                        & SPFNOP
0006A   /                        & SPHNOP ;
        ;
        ;       ROTATE LEFT
        ;
        RRL:    PUSH             & AM2901 ,A,QREG,PASS,ZB
        /                        & SPHNOP
        /                        & IFETCH
0006B   /                        & JTAB IROT3;
                RFCT             & AM2901 ,B,SRAMQL,PASS,ZB
        /                        & SPFNOP
        /                        & SPHNOP
0006C   /                        & SHIFT B 1111 ;
                JZ               & AM2901 ,A,RAMF,PASS,ZQ
        /                        & SPFNOP
0006D   /                        & SPHNOP ;
        ;
        ;       ROTATE RIGHT
        ;
        RRR:    PUSH             & AM2901 ,A,QREG,PASS,ZB
        /                        & SPHNOP
        /                        & IFETCH
0006E   /                        & JTAB IROT3;
                RFCT             & AM2901 ,B,SRAMQR,PASS,ZB
        /                        & SPFNOP
        /                        & SPHNOP
0006F   /                        & SHIFT B 1111 ;
                JZ               & AM2901 ,A,RAMF,PASS,ZQ
        /                        & SPFNOP
00070   /                        & SPHNOP ;
```

```
ADDR     STATEMENT

         ;************************************************************

         ;*                                                           *
         ;*        Extended Arithmetic Group                          *
         ;*        -------------------------                          *
         ;*                                                           *
         ;*        opcode word                                        *
         ;*        D/I ¦ addr word                                    *
         ;*                                                           *
         ;************************************************************
         ;
         DLD:      CALL  DLOAD      & AM2901 ,,NOP,PASS,ZQ
         /                          & SPHNOP
00071    /                          & MREAD;              LOAD A AND B
                   JZ               & AM2901 ,,NOP,PASS,ZQ
         /                          & SPHNOP
00072    /                          & IFETCH ;        NEXT INSTRUCTION
00073              FILLER
         ;
00074              FILLER
         ;
00075              FILLER
         ;
         DST:      CALL  WRTIND     & AM2901 ,,NOP,PASS,ZQ
         /                          & SPHNOP
00076    /                          & MREAD ;            GET ADDR WORD
                   CONT             & AM2901 A,TAB,NOP,PASS,ZA
         /                          & LDMDOR
00077    /                          & MWRITE ;           STORE A
                   CONT             & AM2901 ,,QREG,ADD,ZQ
         /                          & CARRYH
         /                          & SPFNOP
00078    /                          & LDMAR ;            GEN B ADDRESS
                   JP    SKIP       & AM2901 B,TAB,NOP,PASS,ZA
         /                          & LDMDOR
00079    /                          & MWRITE ;           STORE B


         ;
         ;         DIVIDE ENTRY POINT
         ;
         DIVENT:   JP    DIVD       & AM2901 ,,NOP,PASS,ZQ
         /                          & SPHNOP
0007A    /                          & SPFNOP ;     JUMP TO ACTUAL CODE
         ;
```

```
ADDR    STATEMENT

        ;***************************************************************
        ;*                                                             *
        ;*         MULTIPLY - 2'S COMPLEMENT SIGNED                    *
        ;*                                                             *
        ;*       - MPY SPECIAL IN SOURCE FIELD CHANGES SOURCE SELECT   *
        ;*         FROM AB TO ZB BASED ON Q0.                          *
        ;*                                                             *
        ;***************************************************************
        ;
        MULT:     CALL  WRTIND      & AM2901 ,,NOP,PASS,ZQ
        /                           & SPHNOP
0007B   /                           & MREAD ;          FETCH ADDR WORD
                  CONT              & AM2901 TAB,,QREG,PASS,DZ
        /                           & SPFNOP
0007C   /                           & SPHNOP        ; MULTIPLIER TO QR
        ;
        ;         Zero B Reg and shift Q right one bit.  This
        ;    puts Q0 into Q0BUF FLIP-FLOP for Multiply step.
        ;    Note:  In next line, CARRYL is used to force a zero
        ;           into sign of B Reg during shift.  Also, shift
        ;           opcode is same as value loaded into counter!!!!!
        ;
                  PUSH  H 00E       & AM2901 ,B,SRAMQR,AND,ZB
        /                           & CARRYL
        /                           & SHIFT B 1110
        /                           & SPFNOP
0007D   /                           & SPHNOP ; B:=0; Q0BUF:=Q0; COUNTE
                  RFCT              & AM2901 MPY,B,SRAMQR,ADD,AB
        /                           & CARRYL
        /                           & SPFNOP
        /                           & SPHNOP
0007E   /                           & SHIFT B 1110 ;     MULTIPLY STEP
                  CONT              & AM2901 MPY,B,SRAMQR,SUBR,AB
        /                           & CARRYL
        /                           & SHIFT B 1110
        /                           & SPHNOP
0007F   /                           & SPFNOP ;
                  JP    SKIP        & AM2901 ,A,RAMF,PASS,ZQ
        /                           & SPFNOP
        /                           & SPHNOP
00080   /                           & RSTMSR & ENBLO;
```

```
ADDR   STATEMENT

       ;**************************************************************
       ;*                                                            *
       ;*        Input - Output Group                                *
       ;*        --------------------                                *
       ;*                                                            *
       ;*   - In the FETCH line, I/O instructions are mapped into   *
       ;*     two groups:  select code >= 200 and < 200.             *
       ;*                                                            *
       ;*   - Instructions with select codes less than 20 are        *
       ;*     mapped a second time.                                  *
       ;*                                                            *
       ;*   - All HLT instructions are mapped to same entry point   *
       ;*                                                            *
       ;**************************************************************
       ;
       ;         HLT - All select codes
       ;
       HLT:      JP    HLT0        & AM2901 ,,QREG,PASS,ZQ
       /                161        & SPHNOP
00081  /                           & SPFNOP ;     JUMP TO ACTUAL CODE
       ;
       ;         Select Code >= 20 I/O Instructions
       ;
       IOGGE20:  CJP   GENMPV      & AM2901 ,,NOP,PASS,ZQ
       /                           & CONDEXT MPEN
       /                           & SPHNOP
00082  /                           & SPFNOP ;     IF MEM PROT ENABLED
                 CALL  REFETCH     & AM2901 ,,NOP,PASS,ZQ
       /                           & SPHNOP
00083  /                           & SPFNOP ;     REFETCH INSTRUCTION
                 CALL  IOHSHAKE    & AM2901 ,,NOP,PASS,ZQ
       /                           & SPHNOP
00084  /                           & SPFNOP ;    GET I/O CONTROL WORD
                 CALL  IODECODE    & AM2901 ,,NOP,PASS,ZQ
       /                           & SPHNOP
00085  /                           & SPFNOP ; DECODE I/O CONTROL WORD
                 JP    IONOP       & AM2901 PC,PC,RAMA,ADD,ZB
       /                           & CARRYH
       /                           & LDMAR
00086  /                           & SPFNOP ;
```

```
ADDR     STATEMENT

         ;
         ;************************************************************
         ;*                                                          *
         ;*        Select Code < 20 I/O Instructions                 *
         ;*                                                          *
         ;*  IOGLT20 - I/O GROUP, SC < 20 ENTRY POINT                *
         ;*                                                          *
         ;* -Basic Flow is to remap instruction a second time and    *
         ;*  call routine to perform function.  A call is used so    *
         ;*  TDI can be asserted at end of instruction, before next  *
         ;*  instruction is fetched.                                 *
         ;*                                                          *
         ;************************************************************
         ;
         IOGLT20:  CONT                 & AM2901 R12,R12,RAMF,AND,DA
         /                              & IMM H 003F
00087    /                              & SPHNOP ;    MASK OFF SELECT CODE
                   LDCT  IONOP          & AM2901 ,R12,NOP,SUBR,ZB
         /                              & CARRYH
         /                              & LODUSR
         /                              & SPHNOP
00088    /                              & SPFNOP ;      ZERO IMPLIES SC=01
                   CJP   IOGLT20V       & AM2901 ,,NOP,PASS,ZQ
         /                              & CONDUSR Z
         /                              & SPHNOP
00089    /                              & SPFNOP ;            IF SC == 1
                   CJP   GENMPV         & AM2901 ,,NOP,PASS,ZQ
         /                              & CONDEXT MPEN
         /                              & SPHNOP
0008A    /                              & SPFNOP ;  IF MEM PROTECT ENABLED
         IOGLT20V: CALL                 & AM2901 ,,NOP,PASS,ZQ
         /                              & SPFNOP
         /                              & SPHNOP
0008B    /                              & JTAB LOWSC ;  DECODE LOW SC BITS
         IONOP:    CONT                 & AM2901 ,,NOP,PASS,ZQ
         /                              & ENCN SETTDI
0008C    /                              & SPFNOP ;   SET TEMP INT DISABLE
                   JZ                   & AM2901 ,,NOP,PASS,ZQ
         /                              & SPHNOP
0008D    /                              & IFETCH ;  FETCH NEXT INSTRUCTION
         ;
         ;         Select Code 00 I/O Instructions
         ;
         ;         CLC 00 - System Reset (CRS-)
         ;
         CLC00:    CONT                 & AM2901 ,,QREG,PASS,DZ
         /                              & STRD
0008E    /                              & ENCN ICRS ;       GENERATE CRS-
         ;
```

```
LINE   ADDR    STATEMENT

               ;         Set STATUS reg as follows:
               ;
               ;            - TBG off
               ;            - Int Inhibit off
               ;            - Glogal Reg Flag disabled
               ;            - Int Sys off
               ;
                         CONT                & AM2901 ,,QREG,AND,DQ
               /                             & IMM H FFE4
       0008F   /                             & LDST   ; ZERO BITS IN STATUS REG
                         JP    CLC06         & AM2901 ,,NOP,PASS,ZQ
               /                             & SPHNOP
       00090   /                             & SPFNOP ;RESET TBG (HARDWARE BUG)
               ;
               ;         STF 00 - Enable Interrrupt System
               ;
               STF00:    CONT                & AM2901 ,,QREG,PASS,DZ
               /                             & SPHNOP
       00091   /                             & STRD ;         ENABLE INTERRUPTS
                         RET                 & AM2901 ,,NOP,OR,DQ
               /                             & IMM H 0001
       00092   /                             & LDST ;
               ;
               ;         CLF 00 - Disable Interrupt System
               ;
               ;   NOTE:  This routine used by SFS 00,C and SFC 00,C
               ;
               CLF00:    CONT                & AM2901 ,,QREG,PASS,DZ
               /                             & SPHNOP
       00093   /                             & STRD ;        DISABLE INTERRUPTS
                         RET                 & AM2901 ,,NOP,NOTRS,DQ
               /                             & IMM H 0001
       00094   /                             & LDST ;
               ;
               ;         SFS 00,C - Skip if interrupt system enabled and
               ;                    disable interrupt system
               ;
               SFS00.C:  CALL  CLF00         & AM2901 ,,NOP,PASS,ZQ
               /                             & SPHNOP
       00095   /                             & SPFNOP ;    RECORD AND CLEAR BIT
                         JP    SFS00.1       & AM2901 ,,NOP,PASS,ZQ
               /                             & SPHNOP
       00096   /                             & SPFNOP ;
               ;
               ;         SFS 00 - Skip if interrupt system enabled
               ;
               SFS00:    CONT      ,         & AM2901 ,,QREG,PASS,DZ
               /                             & STRD
       00097   /                             & SPHNOP ;        SKIP IF ENABLED
```

```
ADDR    STATEMENT

                SFS00.1:  CONT                  & AM2901 ,,NOP,AND,DQ
                /                               & LODUSR
                /                               & SPHNOP
00098           /                               & IMM H 0001;
                SFSXX:    JRP   IOSKIP          & AM2901 ,,NOP,PASS,ZQ
                /                               & CONDUSR NZ
                /                               & SPHNOP
00099           /                               & SPFNOP ;
                ;
                ;         SFC 00,C - Skip if interrupt system disabled and
                ;                    disable interrupt system
                ;
                SFC00.C:  CALL  CLF00           & AM2901 ,,NOP,PASS,ZQ
                /                               & SPHNOP
0009A           /                               & SPFNOP ;RECORD STATE OF BIT AND
                                                                 CLEAR I
                          JP    SFC00.1         & AM2901 ,,NOP,PASS,ZQ
                /                               & SPHNOP
0009B           /                               & SPFNOP ;
                ;
                ;         SFC 00 - Skip if interrupt system disabled
                ;
                SFC00:    CONT                  & AM2901 ,,QREG,PASS,DZ
                /                               & STRD
0009C           /                               & SPHNOP ;        SKIP IF DISABLED
                SFC00.1:  CONT                  & AM2901 ,,NOP,AND,DQ
                /                               & LODUSR
                /                               & SPHNOP
0009D           /                               & IMM H 0001
                SFCXX:    JRP   IOSKIP          & AM2901 ,,NOP,PASS,ZQ
                /                               & CONDUSR Z
                /                               & SPHNOP
0009E           /                               & SPFNOP ;
                ;
                ;         OT. 00 - Output to interrupt mask register
                ;
                OT.00:    CONT                  & AM2901 ,CAB,NOP,PASS,ZB
                /                               & SPWR LDIM1
0009F           /                               & SPFNOP ; LOAD TBG INTRPT MASK BI
                          JP    OT.02H          & AM2901 ,,NOP,PASS,ZQ
                /                               & SPHNOP
000A0           /                               & SPFNOP ;     JUST LIKE OTA/B 02
                ;
                ;         Select code 01 I/O instructions
                ;
                STO:      JZ                    & AM2901 ,,NOP,PASS,ZQ
                /                               & IFETCH
                /                               & SPHNOP
000A1           /                               & SETMSR & ENBLO ;    SET OVERFLOW
```

```
ADDR    STATEMENT

        CLO:     JZ                  & AM2901 ,,NOP,PASS,ZQ
        /                            & IFETCH
        /                            & SPHNOP
000A2   /                            & RSTMSR & ENBLO ;  CLEAR OVERFLOW
        SOC.H:   JRP   SKIP          & AM2901 ,,NOP,PASS,ZQ
        /                            & SPFNOP
        /                            & SPHNOP
000A3   /                            & CONDMSR NOVR ; SKIP IF OVERFLOW
                                                        CLEAR
        SOS.H:   JRP   SKIP          & AM2901 ,,NOP,PASS,ZQ
        /                            & SPFNOP
        /                            & SPHNOP
000A4   /                            & CONDMSR OVR ;SKIP IF OVERFLOW
                                                       SET
        SOC.C:   JRP   SKIP          & AM2901 ,,NOP,ADD,ZQ
        /                            & SPFNOP
        /                            & SPHNOP
        /                            & CARRYL
000A5   /                            & CONDLMSR NOVR & ENBLO ;
        SOS.C:   JRP   SKIP          & AM2901 ,,NOP,ADD,ZQ
        /                            & SPFNOP
        /                            & SPHNOP
        /                            & CARRYL
000A6   /                            & CONDLMSR OVR & ENBLO ;
        ;
        ;        OUTPUT TO LED REG
        ;
        ;  NOTE: THE LED'S ON A600 ARE HIGH TRUE:
        ;
        ;        LED ON => 1 WRITTEN TO BIT IN REG
        ;
        OT.01:   JZ                  & AM2901 ,CAB,NOP,EXNOR,ZB
        /                            & SPWR LEDWR
000A7   /                            & IFETCH ;       OUTPUT TO LED REG
        LI.01:   RET                 & AM2901 ,CAB,RAMF,PASS,DZ
        /                            & SPRD SWRD
000A8   /                            & SPHNOP ;   INPUT FROM SWITCH REG
        MI.01:   RET                 & AM2901 CAB,CAB,RAMF,OR,DA
        /                            & SPRD SWRD
000A9   /                            & SPHNOP ;
        ;
        ;        Select Code 02 I/O Instructions
        ;
        ;
        ;        STC 02 - ENABLE BREAK FEATURE  (NOT LIKE L-SERIES)
        ;
        STC02:   CALL  REFETCH        & AM2901 ,,NOP,PASS,ZQ
        /                             & SPHNOP
```

```
ADDR    STATEMENT

000AA   /                                & SPFNOP ;   REFETCH INSTRUCTION
                   JP    IONOP           & AM2901 PC,PC,RAMA,ADD,ZB
        /                                & CARRYH
        /                                & LDMAR
000AB   /                                & SPFNOP ;
        ;
        ;          STF 02 - DISABLE GLOBAL REG
        ;
        STF02:     CALL  REFETCH         & AM2901 ,,QREG,PASS,DZ
        /                                & SPHNOP
000AC   /                                & STRD    ;  REFETCH STF INSTRUTION
                   CONT                  & AM2901 ,,NOP,NOTRS,DQ
        /                                & IMM H 0002
000AD   /                                & LDST    ;    CLEAR GLOBAL REG BIT
        IOSKIP:    JP    IONOP           & AM2901 PC,PC,RAMA,ADD,ZB
        /                                & CARRYH
        /                                & LDMAR
000AE   /                                & SPFNOP ;  SET MAR FOR FETCH INST
        ;
        ;          CLF 02 - ENABLE GLOBAL REG
        ;
        CLF02:     CALL  REFETCH         & AM2901 ,,QREG,PASS,DZ
        /                                & SPHNOP
000AF   /                                & STRD ;  REFETCH CLF INSTRUCTION
                   CONT                  & AM2901 ,,NOP,OR,DQ
        /                                & IMM H 0002
000B0   /                                & LDST ;       SET GLOBAL REG BIT
                   RET                   & AM2901 PC,PC,RAMA,ADD,ZB
        /                                & CARRYH
        /                                & LDMAR
000B1   /                                & SPFNOP ;       NEXT INSTRUCTION
        ;
        ;          SFS 02 - SKIP IF GLOBAL REG DISABLED
        ;
        SFS02:     CONT                  & AM2901 ,,QREG,PASS,DZ
        /                                & IMM H 0002
000B2   /                                & SPHNOP ;    TEST GLOBAL REG BIT
                   JP    SFCXX           & AM2901 ,,NOP,AND,DQ
        /                                & LODUSR
        /                                & SPHNOP
000B3   /                                & STRD ;
        ;
        ;          SFC 02 - SKIP IF GLOBAL REG ENABLED
        ;
        SFC02:     CONT                  & AM2901 ,,QREG,PASS,DZ
        /                                & IMM H 0002
000B4   /                                & SPHNOP ;    TEST GLOBAL REG BIT
                   JP    SFSXX           & AM2901 ,,NOP,AND,DQ
        /                                & LODUSR
```

```
 ADDR    STATEMENT

         /                            & STRD
000B5    /                            & SPHNOP ;
         ;
         ;         OT* 02 - OUTPUT TO GLOBAL REGISTER
         ;
         ;   OT.02H - SIMPLY OUTPUT TO GR
         ;   OT.02C - OUTPUT TO GR AND CLEAR FLAG (ENABLE)
         ;
         OT.02C:   CONT               & AM2901 ,,QREG,PASS,DZ
         /                            & SPHNOP
000B6    /                            & STRD ;         READ STATUS REG
                   CONT               & AM2901 ,,NOP,OR,DQ
         /                            & IMM H 0002
000B7    /                            & LDST ;
         OT.02H:   CALL  REFETCH      & AM2901 ,,NOP,PASS,ZQ
         /                            & SPHNOP
000B8    /                            & SPFNOP ;           REFETCH OT*
                   CALL  IOHSHAKE     & AM2901 ,,NOP,PASS,ZQ
         /                            & SPFNOP
000B9    /                            & SPHNOP ;    HANDSHAKE I/O CHIP
                   CALL  IOHSHAK2     & AM2901 ,CAB,NOP,PASS,ZB
         /                            & LDMDOR
000BA    /                            & IOWR ;
                   JP    IONOP        & AM2901 PC,PC,RAMA,ADD,ZB
         /                            & CARRYH
         /                            & LDMAR
000BB    /                            & SPFNOP ; FETCH NEXT INSTRUCTION
         ;
         ;         Select Code 04 I/O Instructions
         ;
         STC04:    CONT              & AM2901 ,,QREG,PASS,DZ
         /                           & SPHNOP
000BC    /                           & STRD ;      INTERRUPT UNINHIBIT
                   RET               & AM2901 ,,NOP,NOTRS,DQ
         /                           & IMM H 0008
000BD    /                           & LDST ;
         ;
         CLC04:    CONT              & AM2901 ,,QREG,PASS,DZ
         /                           & SPHNOP
000BE    /                           & STRD ;        INTERRUPT INHIBIT
                   RET               & AM2901 ,,NOP,OR,DQ
         /                           & IMM H 0008
000BF    /                           & LDST ;
         ;
         ;         SFS 04 - Skip if power not going down
         ;                - Status bit is reverse sense!
         ;
         SFS04:    CONT              & AM2901 ,,QREG,PASS,DZ
         /                           & IMM H 0080
```

```
LINE   ADDR    STATEMENT

1191   000C0   /                          & SPHNOP ;          PONI BIT
1192                     JP    SFCXX      & AM2901 ,,NOP,AND,DQ
1193           /                          & LODUSR
1194           /                          & SPHNOP
1195   000C1   /                          & STRD   ;    MASK WITH STATUS REG
1196           ;
1197           ;         SFC 04 - Skip if power going down
1198           ;                - Status bit is reverse sense!
1199           ;
1200           SFC04:    CONT             & AM2901 ,,QREG,PASS,DZ
1201           /                          & IMM H 0080
1202   000C2   /                          & SPHNOP ;          PFW BIT
1203                     JP    SFSXX      & AM2901 ,,NOP,AND,DQ
1204           /                          & LODUSR
1205           /                          & SPHNOP
1206   000C3   /                          & STRD ;     MASK WITH STATUS REG
1207           ;
1208           ;         SC 04 - Central Interrupt Reg I/O Instructions
1209           ;
1210           OT.04:    CALL  SETCIR     & AM2901 CAB,R6,RAMF,PASS,ZA
1211           /                          & SPHNOP
1212   000C4   /                          & SPFNOP;         PUT SC INTO R6
1213                     RET              & AM2901 ,,NOP,PASS,ZQ
1214           /                          & SPHNOP
1215   000C5   /                          & SPFNOP ;
1216           ;
1217           LI.04:    CALL  READCIR    & AM2901 ,,NOP,PASS,ZQ
1218           /                          & SPHNOP
1219   000C6   /                          & SPFNOP ;
1220                     RET              & AM2901 R6,CAB,RAMF,PASS,ZA
1221           /                          & SPHNOP
1222   000C7   /                          & SPFNOP ;
1223           ;
1224           ;         Select Code 05 I/O Instructions
1225           ;
1226           STC05:    RET              & AM2901 ,,NOP,PASS,DZ
1227           /                          & SPFNOP
1228   000C8   /                          & ENCN SETPSFF ; ENABLE PARITY
                                                             SYSTEM
1229           ;
1230           CLC05:    RET              & AM2901 ,,NOP,PASS,DZ
1231           /                          & SPFNOP
1232   000C9   /                          & ENCN CLRPSFF ; DISABLE PARITY
                                                            SYSTEM
1233           ;
1234           STF05:    CONT             & AM2901 ,,QREG,PASS,DZ
1235           /                          & SPHNOP
1236   000CA   /                          & STRD ; SET EVEN PARITY SENSE (1)
1237                     RET              & AM2901 ,,NOP,OR,DQ
```

```
ADDR    STATEMENT

        /                               & IMM H 0004
000CB   /                               & LDST ;
        ;
        CLF05:   CONT                   & AM2901 ,,QREG,PASS,DZ
        /                               & SPHNOP
000CC   /                               & STRD ;  SET ODD PARITY SENSE (0)
                 RET                    & AM2901 ,,NOP,NOTRS,DQ
        /                               & IMM H 0004
000CD   /                               & LDST ;
        ;
        SFS05:   CONT                   & AM2901 ,,QREG,PASS,DZ
        /                               & IMM H 0004
000CE   /                               & SPHNOP ; SKIP IF PARITY SENSE
                                                     EVEN
                 JP     SFSXX           & AM2901 ,,NOP,AND,DQ
        /                               & LODUSR
        /                               & SPHNOP
000CF   /                               & STRD ;
        ;
        SFC05:   CONT                   & AM2901 ,,QREG,PASS,DZ
        /                               & IMM H 0004
000D0   /                               & SPHNOP ;SKIP IF PARITY SENSE ODD
                 JP     SFCXX           & AM2901 ,,NOP,AND,DQ
        /                               & LODUSR
        /                               & STRD
000D1   /                               & SPHNOP ;
        ;
        LIA05H:  CONT                   & AM2901 ,CAB,RAMF,PASS,DZ
        /                               & SPRD PELENL
000D2   /                               & SPHNOP ; LOAD PARITY LOW ORDER
                                                      REG
                 RET                    & AM2901 CAB,CAB,RAMF,EXOR,DA
        /                               & IMM H FC00
000D3   /                               & SPHNOP ;     INVERT HIGH BYTE
        ;
        LIA05C:  CONT                   & AM2901 ,CAB,RAMF,EXNOR,DZ
        /                               & SPRD PELENH
000D4   /                               & SPHNOP ; LOAD PARITY HIGH ORDER
                                                       REG
                 RET                    & AM2901 CAB,CAB,RAMF,AND,DA
        /                               & IMM H 00FF
000D5   /                               & SPHNOP ; MASK OFF BITS 16-24 OF
                                                       ADDR
        ;
```

```
ADDR    STATEMENT

        ;         Select Code 06 I/O Instructions
        ;
        ;         STC 06 - Turn on TBG
        ;
        STC06:    CONT              & AM2901 ,,QREG,PASS,DZ
        /                           & SPHNOP
000D6   /                           & STRD ;          READ STATUS REG
                  RET               & AM2901 ,,NOP,OR,DQ
        /                           & IMM H 0010
000D7   /                           & LDST ;              SET BIT
        ;
        ;         CLC 06   - Turn off TBG (same as CLC 06,C)
        ;         CLC 06,C - Turn off TBG and clear flag
        ;
        CLC06:    PUSH  H 003       & AM2901 ,,QREG,PASS,DZ
        /                           & SPHNOP
000D8   /                           & STRD ;          READ STATUS REG
        ;
        ;         THE FOLLOWING INSTRUCTION WAITS FOR TBG INTERRUPT
        ;   FLIP-FLOP TO CLEAR.  THIS IS A HARDWARE BUG.  WE MUST
        ;   WAIT FOUR CLOCKS AFTER TURNING OFF THE TBG TO BE SURE
        ;   WE ACTUALLY CLEAR THE TBG INTERRUPT FF WHEN WE EXECUTE
        ;   THE CLF06 INSTRUCTION.
        ;
                  RFCT              & AM2901 ,,NOP,NOTRS,DQ
        /                           & IMM H 0010
000D9   /                           & LDST ;      CLEAR TBG ENABLE BIT
        ;
        ;         CLF 06 - Clear flag on TBG
        ;
        CLF06:    RET               & AM2901 ,,NOP,PASS,ZQ
        /                           & SPFNOP
000DA   /                           & ENCN CLRTBT ; CLEAR TBG TIC FLAG
        ;
        ;         STF 06 - Set flag on TBG
        ;
        STF06:    RET               & AM2901 ,,NOP,PASS,ZQ
        /                           & SPFNOP
000DB   /                           & ENCN SETTBT ;  SET TBG TIC FLAG
        ;
        ;         SFS 06 - Skip if TBG flag set
        ;
        SFS06:    CONT              & AM2901 ,,QREG,PASS,DZ
        /                           & IMM H 0040
000DC   /                           & SPHNOP ;            TBG BIT
                  JP    SFSXX       & AM2901 ,,NOP,AND,DQ
        /                           & LODUSR
        /                           & STRD
000DD   /                           & SPHNOP ;            TEST BIT
```

```
ADDR     STATEMENT

         ;
         ;         SFC 06 - Skip if TBG flag clear
         ;
         SFC06:    CONT              & AM2901 ,,QREG,PASS,DZ
         /                           & IMM H 0040
000DE    /                           & SPHNOP ;          TBG BIT
                   JP    SFCXX       & AM2901 ,,NOP,AND,DQ
         /                           & LODUSR
         /                           & STRD
000DF    /                           & SPHNOP ;          TEST BIT
         ;
         ;         Select Code 07 I/O Instructions
         ;
         ;         STC 07 - Turn on Memory Protect
         ;
         STC07:    RET               & AM2901 ,,NOP,PASS,DZ
         /                           & ENCN SETMPEN
000E0    /                           & SPFNOP ;  TURN ON MEMORY PROTECT
         ;
         ;         LI. 07 - Input from Mem Prot Violation Reg
         ;
         LI.07:    RET               & AM2901 ,CAB,RAMF,PASS,DZ
         /                           & SPRD PRLEN
000E1    /                           & SPHNOP ; LOAD FROM VIOLATION REG


         ;
         ;         DLOAD - DOUBLE REG LOAD SUBROUTINE
         ;
         DLOAD:    CALL  WRTIND      & AM2901 ,,NOP,PASS,ZQ
         /                           & SPHNOP
000E2    /                           & SPFNOP ;        RESOLVE ADDRESS
                   CONT              & AM2901 ,,QREG,ADD,ZQ
         /                           & CARRYH
         /                           & LDMAR
000E3    /                           & SPFNOP ; INC ADDR TO SECOND WORD
                   CONT              & AM2901 TAB,A,RAMF,PASS,DZ
         /                           & SPHNOP
000E4    /                           & MREAD ;         LOAD A REG VALUE
                   CONT              & AM2901 PC,PC,RAMA,ADD,ZB
         /                           & CARRYH
         /                           & LDMAR
000E5    /                           & SPFNOP ;        NEXT INSTR ADDR
                   RET               & AM2901 TAB,B,RAMF,PASS,DZ
         /                           & SPHNOP
000E6    /                           & SPFNOP ;        LOAD B REG VALUE
```

```
ADDR    STATEMENT

        ;
        ;***********************************************************
        ;*                                                         *
        ;*         OPERATING SYSTEM SET                            *
        ;*                                                         *
        ;*    105300 - .CPUID - Processor Identification           *
        ;*    105301 - .FWID  - Microcode Identification           *
        ;*    105302 - .WFI   - Wait for Interrupt                 *
        ;*    105303 - .SIP   - Skip if Interrupt Pending          *
        ;*                                                         *
        ;***********************************************************
        ;
        ;          CPUID - LOAD A REG WITH CPU ID
        ;
        .CPUID:    CONT                & AM2901 ,A,RAMF,PASS,DZ
        /                              & IMM CPUID
000E7   /                              & SPHNOP ;
                   JZ                  & AM2901 ,,NOP,PASS,ZQ
        /                              & SPHNOP
000E8   /                              & IFETCH ;
        ;
        ;          FWID - LOAD A REG WITH MICROCODE ID
        ;
        .FWID:     CONT                & AM2901 ,A,RAMF,PASS,DZ
        /                              & IMM MICREVID
000E9   /                              & SPHNOP ;
                   JZ                  & AM2901 ,,NOP,PASS,ZQ
        /                              & SPHNOP
000EA   /                              & IFETCH ;
        ;
        ;          WFI - WAIT FOR INTERRUPT
        ;
        .WFI:      CJP   INTERRPT      & AM2901 ,,NOP,PASS,ZQ
        /                              & CONDEXT INTRPT
        /                              & SPHNOP
000EB   /                              & SPFNOP ;
                   JP    .WFI          & AM2901 ,,NOP,PASS,ZQ
        /                              & SPHNOP
000EC   /                              & SPFNOP ;
        ;
        ;          SIP - SKIP IF INTERRUPT PENDING
        ;
        .SIP:      CJP   SKIP          & AM2901 ,,NOP,PASS,ZQ
        /                              & CONDEXT SINTRQ
        /                              & SPHNOP
000ED   /                              & SPFNOP ;    IF INTERRUPT PENDING
                   JZ                  & AM2901 ,,NOP,PASS,ZQ
        /                              & SPHNOP
000EE   /                              & IFETCH ;
```

```
LINE    ADDR    STATEMENT

                ;***********************************************************
                ;                                                          *
                ;      .SETP    DEPOSIT (A:=A+1) TO (B:=B+1) (COUNT) TIMES, *
                ;               WHERE (COUNT) IS 2ND WORD OF INSTRUCTION.   *
                ;                                                          *
                ;***********************************************************
                ;
                ;          R5 := A; SET CNTR TO 'SKIP'.
                ;          TEST (B);  START READ OF COUNT.
                ;          CLEAR B<15>; IF IT WAS SET,
                ;            START READ FROM (B-1)
                ;            A := DATA READ + 1
                ;          ELSE
                ;            R5 := COUNT JUST READ
                ;
                .SETP:     CALL  WRTIND     & AM2901 A,R5,RAMF,PASS,ZA
                /                           & LODMSR
                /                           & MREAD
        000EF   /                           & SPHNOP ;
                           CALL  SETP1      & AM2901 B,B,RAMF,ADD,AB
                /                           & CARRYL
                /                           & LODUSR
                /                           & SPFNOP
        000F0   /                           & SPHNOP ;
                           CONT             & AM2901 ,B,NOP,SUBR,ZB
                /                           & CARRYH
                /                           & MREAD
        000F1   /                           & LDMAR ;
                           JP    SETP3      & AM2901 TAB,A,RAMF,ADD,DZ
                /                           & CARRYH
                /                           & MREAD
        000F2   /                           & SPHNOP ;
                SETP1:     CRET             & AM2901 ,B,SRAMR,PASS,ZB
                /                           & SHIFT B 0000
                /                           & CONDUSR C
                /                           & SPFNOP
        000F3   /                           & SPHNOP ;
                           JP    SETP3      & AM2901 TAB,R5,RAMF,PASS,DZ
                /                           & LODMSR
                /                           & SPFNOP
        000F4   /                           & SPHNOP ;

                ;          (ENTER LOOP AT BOTTOM TO HANDLE ZERO-TRIP CASE)
                ;
                ;          MAR := B;   B := B+1
                ;          MOR := A;   A := A+1; START WRITE; CHECK FOR INT.
                ;          R5 := R5-1; IF PREV VALUE WAS NONZERO, LOOP.
                ;                         ELSE GOTO SKIP.
                ;
```

```
LINE    ADDR    STATEMENT

                           CONT                & AM2901 B,B,RAMA,ADD,ZB
                /                              & CARRYH
                /                              & LDMAR
        000F5   /                              & SPFNOP ;
                           CJP    SETP4        & AM2901 A,A,RAMA,ADD,ZB
                /                              & CARRYH
                /                              & CONDEXT INTRPT
                /                              & LDMDOR
        000F6   /                              & MWRITE ;
                SETP3:     CJP    SETP2        & AM2901 ,R5,RAMF,SUBR,ZB
                /                              & CARRYH
                /                              & CONDLMSR NZ
                /                              & SPFNOP
        000F7   /                              & SPHNOP ;
                           JZ                  & AM2901 PC,PC,RAMA,ADD,ZB
                /                              & CARRYH
                /                              & LDMAR
        000F8   /                              & IFETCH ;       NEXT INSTRUCTION
                ;
                ;          INTERRUPT.
                ;            B<15> := 1
                ;            A  := R5
                ;            GOTO INTERRPT
                ;
                SETP4:     CONT                & AM2901 B,B,RAMF,OR,DA
                /                              & IMM H 8000
        000F9   /                              & SPHNOP ;
                           JP     INTERRPT     & AM2901 R5,A,RAMF,PASS,ZA
                /                              & SPFNOP
        000FA   /                              & SPHNOP ;
                ;
                ;          SETP PATCH FOR MDOR HOLD TIME HARDWARE BUG
                ;
                SETP2PAT:  CJP    SETP4        & AM2901 ,A,RAMF,ADD,ZB
                /                              & CARRYH
                /                              & CONDEXT INTRPT
                /                              & SPHNOP
        000FB   /                              & SPFNOP ;
                           JP     SETP3        & AM2901 ,,NOP,PASS,ZQ
                /                              & SPHNOP
        000FC   /                              & SPFNOP ;
                SETP2:     CONT                & AM2901 B,B,RAMA,ADD,ZB
                /                              & CARRYH
                /                              & LDMAR
        000FD   /                              & SPFNOP ;
                           JP     SETP2PAT     & AM2901 ,A,NOP,PASS,ZB
                /                              & LDMDOR
        000FE   /                              & MWRITE ;
```

```
ADDR      STATEMENT

          ;
          ;************************************************************
          ;*                                                           *
          ;*      Unimplemented Instruction Entry                      *
          ;*                                                           *
          ;* Note:  Placing entry point at H FF allows unimplemented   *
          ;*        instructions to map here since blank PROM (7649)   *
          ;*        is all ones.                                       *
          ;*                                                           *
          ;************************************************************
          ;*
000FF               ORG   H FF
000FF     ;
          EAGUIT:   JP    INTUIT     & AM2901 ,,NOP,PASS,ZQ
          /                          & SPHNOP
000FF     /                          & SPFNOP ;

00100               ORG  H 100
00100     ;
00100     ;************************************************************
00100     ;*                                                           *
00100     ;*        SKIP  -  All skips come here                       *
00100     ;*                                                           *
00100     ;*     Since PC points to next instruction + 1 during        *
00100     ;* instruction execution, PC already points to target        *
00100     ;* of skip.                                                  *
00100     ;*                                                           *
00100     ;************************************************************
00100     ;
          SKIP:     JZ               & AM2901 PC,PC,RAMA,ADD,ZB
          /                          & CARRYH
          /                          & LDMAR
00100     /                          & IFETCH ;
```

```
ADDR    STATEMENT

        ;
        ;*************************************************************
        ;*                                                           *
        ;*         INDIRECT  Resolver                                *
        ;*                                                           *
        ;*   Assumes: - Address word read cycle started              *
        ;*            - Allows 3 levels of indirect                  *
        ;*            - Last address read returned in Q reg          *
        ;*            - MAR loaded with PC-1 at exit                 *
        ;*                                                           *
        ;*************************************************************
        ;
        MRGIND:   CRET              & AM2901 ,PC,NOP,SUBR,ZB
        /                           & CONDUSR NSIGN
        /                           & CARRYH
        /                           & SPFNOP
00101   /                           & LDMAR;
        RESOLVE:  LDCT  H 003       & AM2901 TAB,,QREG,PASS,DZ
        /                           & LODUSR
        /                           & LDMAR
00102   /                           & MREAD;
        RESOLVE1: CRET              & AM2901 ,PC,NOP,SUBR,ZB
        /                           & CONDUSR NSIGN
        /                           & CARRYH
        /                           & SPFNOP
00103   /                           & LDMAR;
                  RPCT  RESOLVE1    & AM2901 TAB,,QREG,PASS,DZ
        /                           & LODUSR
        /                           & LDMAR
00104   /                           & MREAD;
                  CONT              & AM2901 ,,NOP,PASS,ZQ
        /                           & ENCN CLRTDI
00105   /                           & SPFNOP ;          CLEAR TDI
                  CJPP  INTERRPT    & AM2901 ,,NOP,PASS,ZQ
        /                           & CONDEXT INTRPT
        /                           & SPHNOP
00106   /                           & SPFNOP ;    IF INTERRUPT PENDING
                  JP    MRGIND      & AM2901 ,,NOP,PASS,ZQ
        /                           & LODUSR
        /                           & SPHNOP
00107   /                           & SPFNOP ;          TEST Q REG
        ;
        ;         Write Indirect
        ;
        MWRTIND:  CRET              & AM2901 ,,NOP,PASS,ZQ
        /                           & SPFNOP
        /                           & SPHNOP
00108   /                           & CONDUSR NSIGN ;     MRG TYPE WRI
```

```
ADDR     STATEMENT

         WRTIND:   LDCT  H 003      & AM2901 TAB,,QREG,PASS,DZ
         /                          & LODUSR
         /                          & LDMAR
00109    /                          & MREAD ;
         WRTIND1:  CRET             & AM2901 ,,NOP,PASS,ZQ
         /                          & SPFNOP
         /                          & SPHNOP
0010A    /                          & CONDUSR NSIGN ;
                   RPCT  WRTIND1    & AM2901 TAB,,QREG,PASS,DZ
         /                          & LODUSR
         /                          & LDMAR
0010B    /                          & MREAD;
                   CRET             & AM2901 ,,NOP,PASS,ZQ
         /                          & CONDUSR NSIGN ;
         /                          & ENCN CLRTDI
0010C    /                          & SPFNOP ;           CLEAR TDI
                   CJPP  INTERRPT   & AM2901 ,,NOP,PASS,ZQ
         /                          & CONDEXT INTRPT
         /                          & SPHNOP
0010D    /                          & SPFNOP ;   IF INTERRUPT PENDING
                   JP    WRTIND     & AM2901 ,,NOP,PASS,ZQ
         /                          & SPHNOP
0010E    /                          & SPFNOP ;     CONTINUE RESOLVING
```

```
LINE    ADDR    STATEMENT
                ;*************************************************************
                ;*                                                           *
                ;*          DOUBLE INTEGER OPERATIONS                        *
                ;*                                                           *
                ;* - Calling sequence:                                       *
                ;*                                                           *
                ;*   .DIN, .DDE & .DNG         All others                    *
                ;*                                                           *
                ;*     JSB  .OPCODE            JSB  .OPCODE                  *
                ;*                             DEF  OPERAND                  *
                ;*                                                           *
                ;* - Operands are stored in memory with most significant     *
                ;*   word in lower address, least significant word in upper  *
                ;*   address:                                                *
                ;*                                                           *
                ;*        JSB  .DAD     +---->OCT   MSB    address           *
                ;*        DEF  *-------+      OCT   LSB    address + 1       *
                ;*                                                           *
                ;* - The operations performed are:                           *
                ;*                                                           *
                ;*       .DAD  (A,B) := (A,B) + (OPERAND)                    *
                ;*       .DSB  (A,B) := (A,B) - (OPERAND)                    *
                ;*       .DSBR (A,B) := (OPERAND) - (A,B)                    *
                ;*       .DNG  (A,B) := - (A,B)                              *
                ;*       .DIN  (A,B) := (A,B) + 1                            *
                ;*       .DDE  (A,B) := (A,B) - 1                            *
                ;*       .DIS  (OPERAND) := (OPERAND) + 1, SKIP IF ZERO      *
                ;*       .DDS  (OPERAND) := (OPERAND) - 1, SKIP IF ZERO      *
                ;*       .DCO  IF (A,B) = (OPERAND) THEN PC := PC + 1        *
                ;*             IF (A,B) < (OPERAND) THEN PC := PC + 2        *
                ;*             IF (A,B) > (OPERAND) THEN PC := PC + 3        *
                ;*                                                           *
                ;* - E Reg can be set but NEVER cleared as follows:          *
                ;*                                                           *
                ;*       .DAD  E set if an unsigned carryout occurs          *
                ;*       .DSB  E set if an unsigned borrow occurs            *
                ;*       .DSBR E set if an unsigned borrow occurs            *
                ;*       .DNG  E set if the (A,B) = 0                        *
                ;*       .DIN  E set if the (A,B) = -1                       *
                ;*       .DDE  E set if the (A,B) = 0                        *
                ;*                                                           *
                ;* - O Reg is cleared and can be set as follows:             *
                ;*                                                           *
                ;*     .DAD  O set if carry into sign XOR carry out of sign  *
                ;*     .DSB  O set if carry into sign XOR carry out of sign  *
                ;*     .DSBR O set if carry into sign XOR carry out of sign  *
                ;*     .DNG  O set if the (A,B) = 2**31                      *
                ;*     .DIN  O set if the (A,B) = 2**31 - 1                  *
                ;*     .DDE  O set if the (A,B) = 2**31                      *
```

```
ADDR    STATEMENT

        ;*                                                            *
        ;**************************************************************
        ;
        ;
        ;
        ;
        ;
        ;
        ;         NOTE:     The double integer scratch register
        ;                   assignments below are also used
        ;                   floating point.
        ;
        ;
        ;
        ;
        ;
        DBLIMSB:  EQU   B 00101     ; DOUBLE INTEGER SCRATCH REGS
        DBLILSB:  EQU   B 00110
        ;

        ;
        ;         .DAD - DOUBLE ADD
        ;
        .DAD:     CALL  DIARG       & AM2901 ,,NOP,PASS,ZQ
        /                           & SPHNOP
0010F   /                           & SPFNOP ; GET OPERAND INTO SCRATC
                  CONT              & AM2901 DBLILSB,B,RAMF,ADD,AB
        /                           & CARRYL
        /                           & LODUSR
        /                           & SPHNOP
00110   /                           & SPFNOP ;         ADD LOWER WORDS
                  CONT              & AM2901 DBLIMSB,A,RAMF,ADD,AB
        /                           & CARRYREG
        /                           & UCLDMSR & ENBLO
        /                           & SPHNOP
00111   /                           & SPFNOP ; ADD UPPER WORDS + CARRY
        .DADE:    CJP   FETCH       & AM2901 PC,PC,RAMA,ADD,ZB
        /                           & CARRYH
        /                           & CONDUSR NC
        /                           & LDMAR
00112   /                           & IFETCH ;         IF NO CARRY OUT
                  JZ                & AM2901 ,,NOP,PASS,ZQ
        /                           & SETMSR & ENBLC
        /                           & SPHNOP
00113   /                           & SPFNOP ;             SET E REG
        ;
        .DSB:     CALL  DIARG       & AM2901 ,,NOP,PASS,ZQ
        /                           & SPHNOP
00114   /                           & SPFNOP ;             GET OPERANDS
```

```
LINE    ADDR    STATEMENT

                          CONT                 & AM2901 DBLILSB,B,RAMF,SUBR,AB
                /                              & CARRYL
                /                              & LODUSR
                /                              & SPHNOP
        00115   /                              & SPFNOP ;           B := B - LSB
                          JP     .DADE         & AM2901 DBLIMSB,A,RAMF,SUBR,AB
                /                              & CARRYREG
                /                              & NUCLDMSR & ENBLO
                /                              & SPHNOP
        00116   /                              & SPFNOP ;A := A - MSB - CARRY,
                                                              LOAD OVERF
                ;
                ;         .DSBR - DOUBLE SUBTRACT REVERSE
                ;
                ;  - NOTE: Sense of carry is adjusted for DADE test
                ;
                .DSBR:    CALL   DIARG         & AM2901 ,,NOP,PASS,ZQ
                /                              & SPHNOP
        00117   /                              & SPFNOP ;           GET OPERAND
                          CONT                 & AM2901 DBLILSB,B,RAMF,SUBS,AB
                /                              & CARRYH
                /                              & LODUSRCI
                /                              & SPHNOP
        00118   /                              & SPFNOP ;           B := B - LSB
                          JP     .DADE         & AM2901 DBLIMSB,A,RAMF,SUBS,AB
                /                              & CARRYREG
                /                              & NUCLDMSR & ENBLO
                /                              & SPHNOP
        00119   /                              & SPFNOP ;    A := A - MSB - CARRY

                ;
                ;         .DNG - DOUBLE NEGATE
                ;         .DIN - DOUBLE INCREMENT
                ;
                .DNG:     CONT                 & AM2901 ,A,RAMF,EXNOR,ZB
                /                              & SPHNOP
        0011A   /                              & SPFNOP ;   ONES COMPLEMENT A & B
                          CONT                 & AM2901 ,B,RAMF,EXNOR,ZB
                /                              & SPHNOP
        0011B   /                              & SPFNOP ;
                ;
                .DIN:     CONT                 & AM2901 ,B,RAMF,ADD,ZB
                /                              & CARRYH
                /                              & LODUSR
                /                              & SPHNOP
        0011C   /                              & SPFNOP ;           B := B + 1
                          CONT                 & AM2901 ,A,RAMF,ADD,ZB
                /                              & CARRYREG
```

```
LINE    ADDR    STATEMENT

                /                                 & UCLDMSR & ENBLO
                /                                 & SPHNOP
        0011D   /                                 & IFETCH ;      A := A + CARRY OUT
                .DINE:      CJP    FETCH          & AM2901 ,,NOP,PASS,ZQ
                /                                 & CONDUSR NC
                /                                 & SPHNOP
        0011E   /                                 & SPFNOP ;        DONE IF NO CARRY
                            JZ                    & AM2901 ,,NOP,PASS,ZQ
                /                                 & SETMSR & ENBLC
                /                                 & SPHNOP
        0011F   /                                 & SPFNOP ;            SET E REG
                ;
                ;           .DDE - DOUBLE DECREMENT
                ;
                .DDE:       CONT                  & AM2901 ,B,RAMF,SUBR,ZB
                /                                 & CARRYH
                /                                 & LODUSR
                /                                 & SPHNOP
        00120   /                                 & SPFNOP ;            B := B - 1
                            JP     .DINE          & AM2901 ,A,RAMF,SUBR,ZB
                /                                 & CARRYREG
                /                                 & NUCLDMSR & ENBLO
                /                                 & SPHNOP
        00121   /                                 & IFETCH ; A := A - CARRY, UC := B


                ;
                ;           DIS - DOUBLE INCREMENT AND SKIP IF ZERO
                ;
                .DIS:       CALL   DIARG          & AM2901 ,,NOP,PASS,ZQ
                /                                 & SPHNOP
        00122   /                                 & SPFNOP ; GET OPERAND INTO SCRATC
                            CONT                  & AM2901 ,DBLILSB,RAMF,ADD,ZB
                /                                 & CARRYH
                /                                 & LODUSR
                /                                 & SPHNOP
        00123   /                                 & SPFNOP ;            INC 2ND WORD
                            JP     .DSKIPZ        & AM2901 ,DBLIMSB,RAMF,ADD,ZB
                /                                 & CARRYUC
                /                                 & SPHNOP
        00124   /                                 & SPFNOP ;        PROPOGATE CARRY
                ;
                ;           DDS - DOUBLE DECREMENT AND SKIP IF ZERO
                ;
                .DDS:       CALL   DIARG          & AM2901 ,,NOP,PASS,ZQ
                /                                 & SPHNOP
        00125   /                                 & SPFNOP ; GET OPERAND INTO SCRATC
                            CONT                  & AM2901 ,DBLILSB,RAMF,SUBR,ZB
                /                                 & CARRYH
```

```
ADDR   STATEMENT

       /                                  & LODUSRCI
       /                                  & SPHNOP
00126  /                                  & SPFNOP ;             DEC 2ND WORD
                 CONT                     & AM2901 ,DBLIMSB,RAMF,SUBR,ZB
       /                                  & CARRYUC
       /                                  & SPHNOP
00127  /                                  & SPFNOP ;         PROPOGATE BORROW
       .DSKIPZ:  CONT                     & AM2901 DBLILSB,TAB,NOP,PASS,ZA
       /                                  & LDMDOR
00128  /                                  & MWRITE
                 CONT                     & AM2901 ,,QREG,SUBR,ZQ
       /                                  & CARRYH
       /                                  & LDMAR
00129  /                                  & SPFNOP ;         ADDR OF 1ST WORD
                 CONT                     & AM2901 DBLIMSB,TAB,NOP,PASS,ZA
       /                                  & LDMDOR
0012A  /                                  & MWRITE ;              WRITE 1 ST
                 CONT                     & AM2901 DBLIMSB,DBLILSB,NOP,OR,AB
       /                                  & LODUSR
       /                                  & SPHNOP
0012B  /                                  & SPFNOP ;            TEST FOR ZERO
                 CJP    SKIP              & AM2901 ,,NOP,PASS,ZQ
       /                                  & CONDUSR NZ
       /                                  & SPHNOP
0012C  /                                  & SPFNOP ;              IF NOT ZERO
                 JP     SKIP              & AM2901 ,PC,RAMF,ADD,ZB
       /                                  & CARRYH
       /                                  & SPHNOP
0012D  /                                  & SPFNOP ;              PC := PC + 1
```

```
LINE    ADDR    STATEMENT

                ;
                ;          .DCO - DOUBLE INTEGER ARITH COMPARE
                ;
                ;    - Calling sequence:
                ;
                ;          JSB   .DCO
                ;          DEF   OPERAND
                ;          JMP   EQUAL              IF (A,B) == (OPERAND)
                ;          JMP   LESS_THAN          IF (A,B) <  (OPERAND)
                ;          JMP   GREATER_THAN       IF (A,B) >  (OPERAND)
                ;
                ;- Both operands are considered 32 bit 2's complement numbers
                ;
                ;- Algorithm is do a signed compare of upper words.  If equal
                ;  then do an unsigned compare of lower words.
                ;
                .DCO:      CALL  DIARG       & AM2901 ,,NOP,PASS,ZQ
                /                            & SPHNOP
        0012E   /                            & SPFNOP ;
                           CONT              & AM2901 A,DBLIMSB,NOP,SUBS,AB
                /                            & CARRYH
                /                            & LODUSR
                /                            & SPHNOP
        0012F   /                            & SPFNOP ;           A :: 1ST WORD
                           CJP   .DCOEQ      & AM2901 A,DBLIMSB,NOP,SUBS,AB
                /                            & CARRYH
                /                            & CONDUSR Z
                /                            & SPHNOP
        00130   /                            & SPFNOP ; IF EQUAL, TEST LOWER BI
                .DCOLT:    CJP   SKIP        & AM2901 ,PC,RAMF,ADD,ZB
                /                            & CARRYH
                /                            & CONDUSR LT
                /                            & SPHNOP
        00131   /                            & SPFNOP ;            IF LESS THAN
                .DCOX:     JP    SKIP        & AM2901 ,PC,RAMF,ADD,ZB
                /                            & CARRYH
                /                            & SPHNOP
        00132   /                            & SPFNOP ;        IF GREATER THAN
                ;
                ;          DO UNSIGNED COMPARE OF LOWER WORDS
                ;
                .DCOEQ:    LDCT  .DCOX       & AM2901 B,DBLILSB,NOP,SUBS,AB
                /                            & CARRYH
                /                            & LODUSR
                /                            & SPHNOP
        00133   /                            & SPFNOP ;           B :: 2ND WORD
                           CJP   SKIP        & AM2901 B,DBLILSB,NOP,SUBS,AB
                /                            & CARRYH
                /                            & CONDUSR Z
```

```
ADDR    STATEMENT

        /                            & SPHNOP
00134   /                            & SPFNOP ;      IF EQUAL THEN EXIT
                   JRP   SKIP        & AM2901 ,PC,RAMF,ADD,ZB
        /                            & CARRYH
        /                            & CONDUSR NC
        /                            & SPHNOP
00135   /                            & SPFNOP ; A BORROW MEANS LESS THA
        ;
        ;          .CPM - SINGLE INTEGER ARITH COMPARE
        ;
        ;  - Calling sequence:
        ;
        ;          JSB   .CPM
        ;  MAR -> DEF   OP1[,I]           - DEF's may reference A/B
        ;   PC -> DEF   OP2[,I]
        ;          JMP   EQUAL            - IF OP1 == OP2
        ;          JMP   LESS_THAN        - IF OP1 <  OP2
        ;          JMP   GREATER_THAN     - IF OP1 >  OP2
        ;
        ;  - Both operands are considered signed 16 bit numbers
        ;
        .CPM:      CALL  BITSB       & AM2901 ,,NOP,PASS,ZQ
        /                            & SPHNOP
00136   /                            & MREAD ;       RESOLVE OPERANDS
        ;
        ;          ON RETURN, R4 := (OP1) AND R5 := (OP2)
        ;
                   LDCT  .DCOLT      & AM2901 R4,R5,NOP,SUBS,AB
        /                            & CARRYH
        /                            & LODUSR
        /                            & SPHNOP
00137   /                            & SPFNOP ;     COMPARE OP1 :: OP2
        ;
        ;          IF Z SET, THEN OP1 == OP2 AND FETCH NEXT INST
        ;                    ELSE GOTO DOUBLE COMPARE LESS THAN TEST
        ;
                   JRP   SKIP        & AM2901 R4,R5,NOP,SUBS,AB
        /                            & CARRYH
        /                            & CONDUSR Z
        /                            & SPHNOP
00138   /                            & SPFNOP ;      IF EQUAL THEN SKIP
```

```
ADDR    STATEMENT

        ;
        ;          DOUBLE INTEGER UTILITIES
        ;
        ;
        ;          DIARG - DOUBLE INTEGER ARGUMENT LOAD
        ;
        ; - Operand is loaded into DBLIMSB and DBLILSB reg pair
        ;
        DIARG:     CALL  WRTIND      & AM2901 ,,NOP,PASS,ZQ
        /                            & SPHNOP
00139   /                            & MREAD  ;        RESOLVE OPERAND
                   CONT              & AM2901 ,,QREG,ADD,ZQ
        /                            & CARRYH
        /                            & LDMAR
0013A   /                            & SPFNOP ;        Q := LSB ADDRESS
                   CONT              & AM2901 TAB,DBLIMSB,RAMF,PASS,DZ
        /                            & SPHNOP
0013B   /                            & MREAD ;         DBLIMSB := MSB
                   RET               & AM2901 TAB,DBLILSB,RAMF,PASS,DZ
        /                            & SPHNOP
0013C   /                            & MREAD  ; DBLILSB := LSB, FREEZE


        ;
        ;***********************************************************
        ;*
        ;*         ASL - ARITH SHIFT LEFT (!)
        ;*
        ;***********************************************************
        ;
        ASLCONT:   CONT              & AM2901 ,A,QREG,PASS,ZB
        /                            & RSTMSR & ENBLO
        /                            & SPHNOP
0013D   /                            & SPFNOP ;          RESET OVERFLOW
                   CONT              & AM2901 ,B,NOP,PASS,ZB
        /                            & SWAPEO
        /                            & SPHNOP
0013E   /                            & SPFNOP ;
                   PUSH              & AM2901 ,B,SRAMQL,PASS,ZB
        /                            & SHIFT B 0110
        /                            & JTAB IROT3
        /                            & SPHNOP
0013F   /                            & SPFNOP ;      LOAD 2910 COUNTER
                   CCALL ASLOVFL     & AM2901 ,,NOP,PASS,ZQ
        /                            & CONDEXT IRSKIP
        /                            & SPHNOP
00140   /                            & SPFNOP ; SET OVERFLOW IF SIGN CH
                   RFCT              & AM2901 ,B,SRAMQL,PASS,ZB
        /                            & CONDMSR SIGN
```

```
LINE    ADDR    STATEMENT

1990            /                         & SHIFT B 0110
1991            /                         & SPHNOP
1992   00141    /                         & SPFNOP ;        2904 OUTPUT MSR
1993                     CCALL ASLOVFL    & AM2901 ,,NOP,PASS,ZQ
1994            /                         & CONDEXT IRSKIP
1995            /                         & SPHNOP
1996   00142    /                         & SPFNOP ; SET OVERFLOW IF SIGN CH
1997                     CONT             & AM2901 ,B,SRAMQR,PASS,ZB
1998            /                         & SHIFT B 0101
1999            /                         & SPHNOP
2000   00143    /                         & IFETCH ;             RESTORE SIGN
2001                     JZ               & AM2901 ,A,RAMF,PASS,ZQ
2002            /                         & SWAPEO
2003            /                         & SPHNOP
2004   00144    /                         & SPFNOP ;           PUT A REG BACK
2005            ;
2006            ASLOVFL:  RET             & AM2901 ,R4,SRAML,PASS,DZ
2007            /                         & IMM H 8000
2008   00145    /                         & SPHNOP ; SET CARRY (SHIFT
                                                     B 0000 IMPLIE
```

```
LINE   ADDR     STATEMENT

                ;
                ;*************************************************************
                ;*                                                           *
                ;*    DIVIDE - 1ST AND 2ND QUADRANT SIGNED DIVISOR           *
                ;*                                                           *
                ;* - DIVIDEND IS IN MACRO B AND A REG                        *
                ;* - DIVIDEND IS MADE POSITIVE                               *
                ;* - DIVISOR IS TWO'S COMPLEMENT SIGNED                      *
                ;*                                                           *
                ;* - AT EACH ITERATION STEP, THE PARTIAL REMAINDER IS        *
                ;*   REDUCED TOWARD ZERO BY ADD OR SUBTRACT:                 *
                ;*                                                           *
                ;* IF PARTIAL REMAINDER SIGN XOR DIVISOR SIGN == 0 THEN SUBR*
                ;*                                                ELSE ADD   *
                ;*                                                           *
                ;* - REMAINDER SIGN IS ALWAYS SAME AS DIVIDEND SIGN          *
                ;* - QOUTIENT SIGN IS DIVIDEND SIGN XOR DIVISOR SIGN         *
                ;*                                                           *
                ;* - OVERFLOW IF DIVIDE BY ZERO LEAVES ABS(DIVIDEND) IN B&A *
                ;* - OVERFLOW IF DIVISOR TOO SMALL LEAVES B & A UNDEFINED    *
                ;* - QUOTIENT BITS ARE GENERATED IN ONE'S COMPLEMENT FORM    *
                ;*                                                           *
                ;*************************************************************
                ;
                DIVD:     CALL  WRTIND      & AM2901 ,,NOP,PASS,ZQ
                /                           & SPHNOP
       00146    /                           & MREAD ;        FETCH ADDR WORD
                ;
                ;         PUT DIVISOR IN R5
                ;
                ;  - SET SIGN AND ZERO IN MACHINE STATUS REG ACCORDINGLY
                ;  - CLEAR OVERFLOW IN MACHINE STATUS REG
                ;
                          CONT              & AM2901 TAB,R5,RAMF,ADD,DZ
                /                           & CARRYL
                /                           & SPHNOP
                /                           & SPFNOP
       00147    /                           & LODMSR & ENBLO ;
                ;
                ;         SAVE SIGN OF DIVIDEND IN R6
                ;
                          CONT              & AM2901 B,R6,RAMF,PASS,ZA
                /                           & LODUSR
                /                           & SPFNOP
       00148    /                           & SPHNOP ;
                ;
                ;         TEST FOR NEGATIVE DIVIDEND
                ;
                ;  - PUT LOWER WORD OF DIVIDEND IN Q REG
```

```
LINE   ADDR    STATEMENT

               ;
                        CJP   DIVD05      & AM2901 ,A,QREG,PASS,ZB
               /                          & SPFNOP
               /                          & SPHNOP
       00149   /                          & CONDUSR NSIGN ; IF POSITIVE
                                                              DIVIDEND
               ;
               ;        TWO'S COMPLEMENT DIVIDEND
               ;
                        CONT              & AM2901 ,,QREG,SUBS,ZQ
               /                          & LODUSR
               /                          & SPFNOP
               /                          & SPHNOP
       0014A   /                          & CARRYH ;
                        CONT              & AM2901 ,B,RAMF,SUBS,ZB
               /                          & SPFNOP
               /                          & SPHNOP
       0014B   /                          & CARRYUC ;
               ;
               ;        TEST FOR ZERO DIVISOR
               ;
               ;    - PUT MOST SIGNIFICANT WORD OF DIVIDEND IN R4
               ;
               DIVD05:  CJP   DIVDOVFL    & AM2901 B,R4,RAMF,PASS,ZA
               /                          & CONDMSR Z
               /                          & SPHNOP
       0014C   /                          & SPFNOP ; IF DIVISOR IS ZERO,
                                                         OVERFLOW
               ;
               ;        TEST FOR DIVISOR TOO SMALL (QUOTIENT >= 2**16)
               ;
                        CONT              & AM2901 ,R4,NOP,PASS,ZB
               /                          & DIVCOND
               /                          & SPHNOP
       0014D   /                          & SPFNOP ; SET SIGN DIFF FLIP-FLOP
                        CONT              & AM2901 R5,DIV,RAMF,SUBR,AB
               /                          & LODUSR
               /                          & CARRYL
               /                          & SPFNOP
       0014E   /                          & SPHNOP ;
                        CJP   DIVDOVFL    & AM2901 ,,NOP,PASS,ZQ
               /                          & SPFNOP
               /                          & SPHNOP
       0014F   /                          & CONDUSR NSIGN ; IF POSITIVE THEN
                                                                OVERFL
               ;
               ;        DIVIDE ITERATION STEP
```

```
ADDR    STATEMENT

        ;
        ; Note:  Shift opcode is same as value loaded into counter in
        ;        in next line.  This shift produces a bit in the
        ;        remainder (B reg) that is discarded (see right shift
        ;        remainder below), so the bit shifted into the Q reg
                 in the next line is a dont care.
        ;
                  PUSH  H 00F      & AM2901 ,R4,SRAMQL,PASS,ZB
        /                          & DIVCOND
        /                          & SPFNOP
        /                          & SPHNOP
00150   /                          & SHIFT B 1111 ;  SET SIGN DIFF FF
                  RFCT             & AM2901 R5,DIV,SRAMQL,SUBR,AB
        /                          & DIVCOND
        /                          & CARRYL
        /                          & SPFNOP
        /                          & SPHNOP
00151   /                          & SHIFT B 1111 ;
        ;
        ;         SHIFT REMAINDER RIGHT ONE
        ;
                  CONT             & AM2901 ,R4,SRAMR,PASS,ZB
        /                          & SPFNOP
        /                          & SPHNOP
00152   /                          & SHIFT B 1111  ;    SHIFT IN SIGN
        ;
        ;         TEST FOR REMAINDER NEGATIVE
        ;
                  CONT             & AM2901 ,R4,NOP,PASS,ZB
        /                          & LODUSR
        /                          & SPFNOP
00153   /                          & SPHNOP ;
                  CJP   DIVD20     & AM2901 ,R4,NOP,PASS,ZB
        /                          & SPFNOP
        /                          & SPHNOP
00154   /                          & CONDUSR NSIGN ; IF POSITIVE THEN
                                                       OKAY
        ;
        ;         IF REMAINDER IS MINUS, RESTORE REMAINDER BY
        ;  ADDING/SUBTRACTING DIVISOR.
        ;
                  CONT             & AM2901 ,R4,NOP,PASS,ZB
        /                          & DIVCOND
        /                          & SPFNOP
00155   /                          & SPHNOP ;      SET SIGN DIFF FF
                  CONT             & AM2901 R5,DIV,RAMF,SUBR,AB
        /                          & CARRYL
        /                          & SPFNOP
00156   /                          & SPHNOP ;            RESTORE
```

```
ADDR    STATEMENT

        ;
        ;          IF DIVIDEND WAS NEGATIVE, COMPLEMENT REMAINDER
        ;
        ;   - COMPUTE EXPECTED SIGN:  DIVIDEND SIGN XOR DIVISOR SIGN
        ;
        DIVD20:    CONT              & AM2901 ,R6,NOP,PASS,ZB
        /                            & LODUSR
        /                            & SPHNOP
00157   /                            & SPFNOP ;
                   CJP   DIVD25      & AM2901 R6,R5,RAMF,EXOR,AB
        /                            & CONDUSR NSIGN
        /                            & SPHNOP
00158   /                            & SPFNOP ;    IF POSITIVE DIVIDEND
                   CONT              & AM2901 ,R4,RAMF,SUBS,ZB
        /                            & CARRYH
        /                            & SPHNOP
00159   /                            & SPFNOP ;   COMPLEMENT REMAINDER
        ;
        ;          IF DIVIDEND AND DIVISOR HAD DIFFERENT SIGN,
        ;  COMPLEMENT QUOTIENT
        ;
                   CONT              & AM2901 ,R5,NOP,PASS,ZB
        /                            & LODUSR
        /                            & SPFNOP
0015A   /                            & SPHNOP ;     TEST EXPECTED SIGN
        DIVD25:    CJP   DIVD30      & AM2901 R4,B,RAMF,PASS,ZA
        /                            & SPFNOP
        /                            & SPHNOP
0015B   /                            & CONDUSR SIGN ; IF DIFFERENT SIGN
        ;
        ;          ONE'S COMPLEMENT QUOTIENT TO FORM TRUE QUOTIENT
        ;
                   JP    DIVD40      & AM2901 ,A,RAMF,SUBS,ZQ
        /                            & LODUSR
        /                            & SPFNOP
        /                            & SPHNOP
0015C   /                            & CARRYL ;
        ;
        ;         CHANGE ONE'S COMPLEMENT TO TWO'S COMPLEMENT QUOTIENT
        ;
        DIVD30:    JP    DIVD40      & AM2901 ,A,RAMF,ADD,ZQ
        /                            & LODUSR
        /                            & SPFNOP
        /                            & SPHNOP
0015D   /                            & CARRYH ;
        ;
        ;          DONE IF QUOTIENT IS ZERO
        ;
```

```
LINE    ADDR     STATEMENT

                 DIVD40:   CJP   SKIP        & AM2901 A,R5,NOP,EXOR,AB
                 /                           & SPFNOP
                 /                           & SPHNOP
        0015E    /                           & CONDUSR Z ;    SET USR WITH SIGN
                 ;
                 ;         CALCULATE OVERFLOW BY:
                 ;
                 ;  IF DIVISOR SIGN XOR DIVIDEND SIGN XOR QUOTIENT SIGN != 0
                     OVERFLOW
                 ;
                           CJP   SKIP        & AM2901 ,A,QREG,PASS,ZB
                 /                           & SPFNOP
                 /                           & SPHNOP
        0015F    /                           & CONDUSR NSIGN ;
                 ;
                 ;         DIVIDE OVERFLOW
                 ;
                 ;   SET OVERFLOW AND PUT Q REG BACK IN A REG
                 ;
                 DIVDOVFL: JP    SKIP        & AM2901 ,A,RAMF,PASS,ZQ
                 /                           & SPFNOP
                 /                           & SPHNOP
        00160    /                           & SETMSR & ENBLO ;    SET OVERFLOW


                 ;
                 ;***********************************************************
                 ;*                                                         *
                 ;*         I/O Instruction Routines (cont)                 *
                 ;*                                                         *
                 ;***********************************************************
                 ;
                 ;         HLT INSTRUCTION
                 ;
                 ;   - Refetch HLT instruction.  I/O master that is
                 ;     front panel will recognize HLT and start slave
                 ;     request process.
                 ;
                 ;   - Wait for slave request interrupt
                 ;
                 HLTO:     CJP   GENMPV      & AM2901 ,,NOP,PASS,ZQ
                 /                           & CONDEXT MPEN
                 /                           & SPHNOP
        00161    /                           & SPFNOP ;  IF MEM PROTECT ENABLED
                           CALL  REFETCH     & AM2901 ,,NOP,PASS,ZQ
                 /                 165       & SPHNOP
        00162    /                           & SPFNOP ;    REFETCH INSTRUCTION
                 HLT1:     CJP   INTHLDPC    & AM2901 ,,NOP,PASS,ZQ
                 /                           & CONDEXT INTRPT
```

```
LINE   ADDR   STATEMENT

2245          /                          & SPHNOP
2246   00163  /                          & SPFNOP ; IF INTRPT (SLAVE REQUES
2247                    JP    HLT1       & AM2901 ,,NOP,PASS,ZQ
2248          /               163        & SPHNOP
2249   00164  /                          & SPFNOP ;      WAIT FOR INTERRUPT

2251          ;
2252          ;************************************************************
2253          ;*                                                          *
2254          ;*        I/O Instruction Subroutines                       *
2255          ;*                                                          *
2256          ;************************************************************
2257          ;
2258          ;
2259          ;         REFETCH - REFETCH I/O INSTRUCTION
2260          ;
2261          ;      ALL INSTRUCTIONS ARE FETCHED WITHOUT ASSERTING RNI
2262          ;  SO I/O CHIPS DO NOT HAVE TO LOOK AT INSTRUCTIONS GOING BY.
2263          ;  HOWEVER, THE I/O CHIPS MUST BE FORCED TO LOOK AT I/O
2264          ;  INSTRUCTIONS.  THEREFORE, ALL I/O INSTRUCTIONS ARE FETCHED
2265          ;  A SECOND TIME, ASSERTING RNI.
2266          ;
2267          REFETCH:  CONT             & AM2901 PC,PC,RAMF,SUBR,DA
2268          /                          & CARRYL
2269          /                          & LDMAR
2270   00165  /                          & IMM H 0002 ;      BACKUP PC
2271          ;
2272          ;         TEST FOR FETCH FROM LOCATION 0 OR 1
2273          ;
2274                    CONT             & AM2901 ,PC,NOP,SUBR,ZB
2275          /                          & CARRYH
2276          /                          & LODUSR
2277          /                          & SPHNOP
2278   00166  /                          & SPFNOP ;             SUBTRACT ONE
2279                    CCALL REFETCH2   & AM2901 ,,NOP,PASS,ZQ
2280          /                          & CONDUSR ULE
2281          /                          & SPHNOP
2282   00167  /                          & SPFNOP ;             IF PC <= 1
2283          ;
2284          ;         NORMAL INSTRUCTION REFETCH
2285          ;
2286                    PUSH  H 002      & AM2901 ,PC,RAMF,ADD,ZB
2287          /                          & CARRYH
2288          /                          & SPHNOP
2289   00168  /                          & RFETCH ;     REFETCH INSTRUCTION
2290          ;
2291          ;         THIS IS A WAIT FOR RFETCH DURING REFRESH
2292          ;
```

```
LINE   ADDR    STATEMENT

                         RFCT              & AM2901 ,,NOP,PASS,ZQ
               /                           & SPHNOP
       00169   /                           & SPFNOP ;
                         RET               & AM2901 ,MAPX,NOP,PASS,ZB
               /                           & LDAER
       0016A   /                           & SPFNOP ;
               ;
               ;         FETCH WAS FROM A OR B
               ;
               ;  - ASSUME INSTRUCTION ALREADY IN LOC 2 OF BOOT MEMORY
               ;
               REFETCH2: CONT              & AM2901 ,,NOP,PASS,DZ
               /                           & LDMAR
       0016B   /                           & IMM H 0002 ; REFETCH FROM
                                                               LOCATION 2
                         RET               & AM2901 ,,NOP,PASS,DZ
               /                           & LDAER
       0016C   /                           & IMM H 0020 ;      IN BOOT MEMORY
               ;
               ;         IOHSHAKE - I/O CHIP HANDSHAKE
               ;
               ;      THIS ROUTINE HANDSHAKES A WORD FROM A I/O CHIP.
               ;  IT WILL WAIT 6 MICROCYCLES BEFORE ABORTING THE HANDSHAKE
               ;  AND FETCHING THE NEXT INSTRUCTION.
               ;
               IOHSHAKE: PUSH  H 002       & AM2901 ,,NOP,PASS,ZQ
               /                           & SPHNOP
       0016D   /                           & SPFNOP ;
                         CJPP  IOHSHAK1    & AM2901 ,,NOP,PASS,ZQ
               /                 171       & CONDEXT IORQ
               /                           & SPHNOP
       0016E   /                           & SPFNOP ;
                         RFCT              & AM2901 ,,NOP,PASS,ZQ
               /                           & SPHNOP
       0016F   /                           & SPFNOP ;
               ;
               ;         NO RESPONSE FROM THE I/O CHIP
               ;
               ;  - TREAT THIS AS A NOP
               ;
                         JP    IONOP       & AM2901 PC,PC,RAMA,ADD,ZB
               /                           & CARRYH
               /                           & LDMAR
       00170   /                           & SPFNOP ;
               ;
               ;         I/O CHIP ASSERTED IORQ
               ;
               ;  - READ CONTROL WORD
               ;
```

```
LINE   ADDR    STATEMENT

               IOHSHAK1: CONT                 & AM2901 ,,NOP,PASS,ZQ
               /                              & IORD
       00171   /                              & SPHNOP ;
               IOHSHAK2: CJP   IOHSHAK2       & AM2901 ,,NOP,PASS,ZQ
               /                              & CONDEXT IORQ
               /                              & SPHNOP
       00172   /                              & SPFNOP ;   WAIT FOR IORQ TO FALL
               ;
               ;          CONTROL WORD IS NOW IN MDIR
               ;
                          RET                 & AM2901 ,,QREG,PASS,DZ
               /                              & SPHNOP
       00173   /                              & SPFNOP ;RETURN CONTROL WORD IN Q
               ;
               ;          IODECODE - I/O CONTROL WORD DECODE
               ;
               ;   - ALGORITHM IS LINEAR SEARCH
               ;
               ;   - CONTROL WORD MEANINGS:
               ;
               ;     0000 - NOP
               ;   * 0001 - LOAD P FROM DATA BUS
               ;   * 0010 - LOAD A FROM DATA BUS
               ;   * 0011 - LOAD B FROM DATA BUS
               ;     0100 - SET THE OVERFLOW BIT
               ;     0101 - CLEAR THE OVERFLOW BIT
               ;   * 0110 - MERGE INTO A REG FROM DATA BUS
               ;   * 0111 - INCREMENT P
               ;     1000 - PUT STATUS REG ON DATA BUS
               ;   * 1001 - ENABLE BOOT ROM
               ;   * 1010 - PUT A ON DATA BUS
               ;   * 1011 - PUT B ON DATA BUS
               ;     1100 - CLEAR E REG
               ;     1101 - SET E REG
               ;   * 1110 - PUT P ON DATA BUS
               ;     1111 - PUT P ON DATA BUS, INCREMENT P
               ;
               ;   - NOTE: ONLY STARRED (*) OPCODES ARE GENERATED BY
               ;           I/O MASTER
               ;
               IODECODE: PUSH   H 003         & AM2901 ,R4,RAMF,PASS,DZ
               /                              & SPFNOP
       00174   /                              & SPHNOP ;  PUT CONTROL WORD IN R4
                         RFCT                 & AM2901 ,R4,SRAMR,PASS,ZB
               /                              & SHIFT B 0000
               /                              & SPHNOP
       00175   /                              & SPFNOP ; POSITION IN BITS 0 TO 3
                         CONT                 & AM2901 R4,R4,RAMF,AND,DA
               /                              & LODUSR
```

```
ADDR    STATEMENT

         /                           & IMM H 000F
00176    /                           & SPHNOP ; MASK OFF CNTL HEX &
                                                SET Z ST
                    CJP    IODCXXXX  & AM2901 ,R4,RAMF,SUBR,ZB
         /                           & CONDUSR Z
         /                           & CARRYH
         /                           & SPHNOP
00177    /                           & SPFNOP ;            IF 0000
                    CJP    IODC0001  & AM2901 ,R4,RAMF,SUBR,ZB
         /                           & CONDUSR Z
         /                           & CARRYH
         /                           & SPHNOP
00178    /                           & SPFNOP ;            IF 0001
                    CJP    IODC0010  & AM2901 ,R4,RAMF,SUBR,ZB
         /                           & CONDUSR Z
         /                           & CARRYH
         /                           & SPHNOP
00179    /                           & SPFNOP ;            IF 0010
                    CJP    IODC0011  & AM2901 ,R4,RAMF,SUBR,ZB
         /                           & CONDUSR Z
         /                           & CARRYH
         /                           & SPHNOP
0017A    /                           & SPFNOP ;            IF 0011
                    CJP    IODCXXXX  & AM2901 ,R4,RAMF,SUBR,ZB
         /                           & CONDUSR Z
         /                           & CARRYH
         /                           & SPHNOP
0017B    /                           & SPFNOP ;            IF 0100
                    CJP    IODCXXXX  & AM2901 ,R4,RAMF,SUBR,ZB
         /                           & CONDUSR Z
         /                           & CARRYH
         /                           & SPHNOP
0017C    /                           & SPFNOP ;            IF 0101
                    CJP    IODC0110  & AM2901 ,R4,RAMF,SUBR,ZB
         /                           & CONDUSR Z
         /                           & CARRYH
         /                           & SPHNOP
0017D    /                           & SPFNOP ;            IF 0110
                    CJP    IODC0111  & AM2901 ,R4,RAMF,SUBR,ZB
         /                           & CONDUSR Z
         /                           & CARRYH
         /                           & SPHNOP
0017E    /                           & SPFNOP ;            IF 0111
                    CJP    IODCXXXX  & AM2901 ,R4,RAMF,SUBR,ZB
         /                           & CONDUSR Z
         /                           & CARRYH
         /                           & SPHNOP
0017F    /                           & SPFNOP ;            IF 1000
                    CJP    IODC1001  & AM2901 ,R4,RAMF,SUBR,ZB
```

```
LINE    ADDR    STATEMENT

                /                               & CONDUSR Z
                /                               & CARRYH
                /                               & SPHNOP
        00180   /                               & SPFNOP ;              IF 1001
                          CJP     IODC1010      & AM2901 ,R4,RAMF,SUBR,ZB
                /                               & CONDUSR Z
                /                               & CARRYH
                /                               & SPHNOP
        00181   /                               & SPFNOP ;              IF 1010
                          CJP     IODC1011      & AM2901 ,R4,RAMF,SUBR,ZB
                /                               & CONDUSR Z
                /                               & CARRYH
                /                               & SPHNOP
        00182   /                               & SPFNOP ;              IF 1011
                          CJP     IODCXXXX      & AM2901 ,R4,RAMF,SUBR,ZB
                /                               & CONDUSR Z
                /                               & CARRYH
                /                               & SPHNOP
        00183   /                               & SPFNOP ;              IF 1100
                          CJP     IODCXXXX      & AM2901 ,R4,RAMF,SUBR,ZB
                /                               & CONDUSR Z
                /                               & CARRYH
                /                               & SPHNOP
        00184   /                               & SPFNOP ;              IF 1101
                          CJP     IODC1110      & AM2901 ,R4,RAMF,SUBR,ZB
                /                 19A           & CONDUSR Z
                /                               & CARRYH
                /                               & SPHNOP
        00185   /                               & SPFNOP ;              IF 1110
                ;
                ;         UNDEFINED CONTROL WORD IS NOP
                ;
                IODCXXXX: RET                   & AM2901 ,R4,NOP,PASS,ZB
                /                               & SPHNOP
        00186   /                               & SPFNOP ;       IF 1111 OR GREATER
                ;
                ;         LOAD P FROM DATA BUS
                ;
                IODC0001: CALL  IOHSHAKE        & AM2901 ,,NOP,PASS,ZQ
                /                               & SPHNOP
        00187   /                               & SPFNOP ;    LOAD P FROM DATA BUS
                          RET                   & AM2901 ,PC,RAMF,PASS,DZ
                /                               & SPHNOP
        00188   /                               & SPFNOP ;
                ;
                ;         LOAD A FROM DATA BUS
                ;
```

```
ADDR    STATEMENT

        IODC0010: CALL  IOHSHAKE   & AM2901 ,,NOP,PASS,ZQ
        /                          & SPHNOP
00189   /                          & SPFNOP ;    LOAD A FROM DATA BUS
                  RET              & AM2901 ,A,RAMF,PASS,DZ
        /                          & SPHNOP
0018A   /                          & SPFNOP ;
        ;
        ;         LOAD B FROM DATA BUS
        ;
        IODC0011: CALL  IOHSHAKE   & AM2901 ,,NOP,PASS,ZQ
        /                          & SPHNOP
0018B   /                          & SPFNOP ;    LOAD B FROM DATA BUS
                  RET              & AM2901 ,B,RAMF,PASS,DZ
        /                          & SPHNOP
0018C   /                          & SPFNOP ;
        ;
        ;         MERGE INTO A/B
        ;
        ;  NOTE:  Register is selected by IR bit 10
        ;
        IODC0110: CALL  IOHSHAKE   & AM2901 ,,NOP,PASS,ZQ
        /                          & SPHNOP
0018D   /                          & SPFNOP ;   MERGE INTO A/B (IR10)
                  RET              & AM2901 CAB,CAB,RAMF,OR,DA
        /                          & SPHNOP
0018E   /                          & SPFNOP ;
        ;
        ;         INCREMENT PC
        ;
        IODC0111: RET              & AM2901 ,PC,RAMF,ADD,ZB
        /                          & CARRYH
        /                          & SPHNOP
0018F   /                          & SPFNOP ;            INCREMENT PC
        ;
        ;         ENABLE BOOT ROM
        ;
        ; - In addition to setting the BOOT enable bit, the following
        ;   operations must be done:
        ;
        ;       * Format WMAP
        ;       * Turn off memory protect
        ;       * Enable boot memory
        ;       * Store WMAP into location 100Q of boot memory
        ;       * Assert TDI so no interrupt before VCP start up
        ;
```

```
LINE   ADDR   STATEMENT

              IODC1001: CALL  STWMAP      & AM2901 ,,NOP,PASS,ZQ
              /                           & SPHNOP
       00190  /                           & SPFNOP ;            FORMAT WMAP
                        CONT              & AM2901 ,,NOP,PASS,ZQ
              /                           & ENCN CLRMPEN
       00191  /                           & SPFNOP ;    TURN OFF MEMORY PROT
                        CONT              & AM2901 MAPX,MAPX,RAMF,OR,DA
              /                           & IMM H 0020
       00192  /                           & LDAER  ;         ENABLE BOOT ROM
                        CONT              & AM2901 ,,NOP,PASS,DZ
              /                           & LDMAR
       00193  /                           & IMM H 0040 ;       LOCATION 100Q
                        CONT              & AM2901 ,R4,NOP,PASS,ZB
              /                           & LDMDOR
       00194  /                           & MWRITE ;
                        RET               & AM2901 ,,NOP,PASS,ZQ
              /                           & ENCN SETTDI
       00195  /                           & SPFNOP ;            ASSERT TDI
              ;
              ;         PLACE A REG ON DATA BUS
              ;
              IODC1010: CALL  IOHSHAK2    & AM2901 A,A,RAMA,PASS,ZB
              /                           & LDMDOR
       00196  /                           & IOWR   ;     PLACE A ON DATA BUS
                        RET               & AM2901 ,,NOP,PASS,ZQ
              /                           & SPHNOP
       00197  /                           & SPFNOP ;
              ;
              ;         PLACE B ON DATA BUS
              ;
              IODC1011: CALL  IOHSHAK2    & AM2901 B,B,RAMA,PASS,ZB
              /                           & LDMDOR
       00198  /                           & IOWR   ;     PLACE B ON DATA BUS
                        RET               & AM2901 ,,NOP,PASS,ZQ
              /                           & SPHNOP
       00199  /                           & SPFNOP ;
              ;
              ;         PLACE PC ON DATA BUS
              ;
              IODC1110: CALL  IOHSHAK2    & AM2901 PC,PC,RAMA,PASS,ZB
              /                   172     & LDMDOR
       0019A  /                           & IOWR   ;    PLACE PC ON DATA BUS
                        RET               & AM2901 ,,NOP,PASS,ZQ
              /                           & SPHNOP
       0019B  /                           & SPFNOP ;
```

```
LINE   ADDR    STATEMENT

               ;
               ;**************************************************************
               ;*                                                            *
               ;*          Interrupt Handler                                 *
               ;*                                                            *
               ;*   - Since PC points to next instruction + 1, back up PC   *
               ;*                                                            *
               ;**************************************************************
               ;
               INTERRPT: CONT               & AM2901 PC,PC,RAMF,SUBR,DA
               /                            & CARRYL
               /                            & SPHNOP
       0019C   /                            & IMM H 0002 ;       BACKUP PC
               INTHLDPC: CVECT  INTTBL      & AM2901 ,,NOP,PASS,ZQ
               /                            & SPHNOP
       0019D   /                            & MKLROFF ;
               ;
               ;          POWER ON INTERRUPT
               ;
               ;          SELF-TEST MICROCODE
               ;
               ;    - A PATTERN OF ALL ONES WITH A SINGLE ZERO BIT IS
               ;      ROTATED THRU ALL DATA BUS BITS.
               ;
               INTPON:   CONT               & AM2901 B,B,RAMF,EXNOR,AB
               /                            & SPHNOP
       0019E   /                            & SPFNOP ;            B := H FFFF
                         PUSH               & AM2901 B,A,RAMF,SUBR,ZA
               /                            & CARRYH
               /                            & ENCN SETDTST
       0019F   /                            & SPFNOP ;            A := H FFFE
                         CONT               & AM2901 ,A,SRAML,PASS,ZB
               /                            & SHIFT ROTATE
               /                            & LODMSR
               /                            & LDMDOR
       001A0   /                            & SPFNOP ;      WRITE PATTERN IN A
                         LOOP               & AM2901 B,B,RAMA,EXNOR,DA
               /                            & CONDLMSR NSIGN
               /                            & LDMAR
       001A1   /                            & SPFNOP ;BUILD CHECK PATTERN IN B
                         CJP   LASTWD       & AM2901 ,A,RAMF,AND,ZB
               /                            & CARRYH
               /                            & CONDLMSR Z  & ENBLC
               /                            & SPHNOP
       001A2   /                            & SPFNOP ;      IF B = 0 THEN PASS
               ;
               ;          SELF-TEST FAILURE
               ;
```

```
ADDR     STATEMENT

         INTDEAD:  CONT                & AM2901 ,A,SRAML,PASS,ZB
         /                             & SHIFT ROTATEC
         /                             & LDMDOR
001A3    /                             & SPFNOP ;        GENERATE PATTERN
                   JP    INTDEAD       & AM2901 A,A,RAMA,PASS,ZA
         /                             & SPHNOP
001A4    /                             & SPFNOP ;         LOOP ON PATTERN
         ;
         ;         INIT REGISTERS
         ;
         ;   MAPX - POINTS TO BOOT MEMORY
         ;   MAPD - POINTS TO MAP ZERO AND MAP ZERO
         ;   PC   - POINTS TO LOCATION 20002Q IN BOOT MEMORY
         ;   LEDS - RESULT OF SELF-TEST
         ;
         INTPONOK: CONT                & AM2901 ,,NOP,PASS,ZQ
         /                             & ENCN CLRDTST
001A5    /                             & SPFNOP ; TURN OFF DATA BUS LOOPB
                   CONT                & AM2901 ,,NOP,PASS,DZ
         /                             & IMM H 0020
001A6    /                             & LDST ;   INIT PROCESSOR STATUS R
                   CONT                & AM2901 B,MAPD,RAMA,AND,ZB
         /                             & SPWR LEDWR
001A7    /                             & MKLRON ;     INIT DATA 1 TO ZERO
                   CONT                & AM2901 ,PC,RAMF,PASS,DZ
         /                             & SPHNOP
001A8    /                             & IMM H 2002 ;   INIT PC TO 20002Q
         ;
         ;         INIT MAPS
         ;
         ;   - ALL MAP REGS ARE INITIALIZED TO THEIR OWN NUMBER
         ;     EX:  MAP0000 := 0;
         ;          MAP0001 := 1;
         ;          ...
         ;          MAP1023 := 1023;
         ;
                   CONT                & AM2901 ,R5,RAMF,AND,ZB
         /                             & SPHNOP
001A9    /                             & SPFNOP ;   INIT MAP REG CONTENTS
                   PUSH  H 01F         & AM2901 ,R4,RAMF,AND,ZB
         /                             & LODMSR
         /                             & SPHNOP
001AA    /                             & SPFNOP ;2910 := NUMBER OF MAP SE
         ;
         ;         THE FOLLOWING CPUSH NEVER LOADS THE COUNTER
         ;
```

```
LINE   ADDR    STATEMENT

                        CPUSH              & AM2901 R4,R4,RAMA,ADD,ZA
               /                           & CARRYH
               /                           & CONDMSR NZ
               /                           & SPFNOP
       001AB   /                           & LDAER ;  LOAD AER WITH MAP NUMBE
                        CONT               & AM2901 R4,R4,RAMA,ADD,DA
               /                           & CARRYL
               /                           & LODUSR
               /                           & LDMAR
       001AC   /                           & IMM H 0400 ; LOAD MAR WITH MAP
                                                          REG NUMBER
                        LOOP               & AM2901 R5,R5,RAMA,ADD,ZB
               /                           & CARRYH
               /                           & CONDUSR SIGN
               /                           & SPWR MAPWR
       001AD   /                           & SPFNOP ;         WRITE TO MAP REG
                        RFCT               & AM2901 R4,R4,RAMF,AND,DA
               /                           & SPHNOP
       001AE   /                           & IMM H 001F ; ZERO MAP REG NUMBER
                        CONT               & AM2901 ,MAPX,RAMF,PASS,DZ
               /                           & LDAER
       001AF   /                           & IMM H 0020 ; SET MAPX TO BOOT
                                                          MEMORY
               ;
               ;        FETCH FIRST INSTRUCTION
               ;
                        JZ                 & AM2901 PC,PC,RAMA,ADD,ZB
               /                           & CARRYH
               /                           & LDMAR
       001B0   /                           & IFETCH ;
               ;
               ;        A/B Instruction Fetch Interrupt
               ;
               ;  Method:  Interrupt is generated when a MAR contains
               ;           zero or one and IFETCH is asserted.  This
               ;           condition is latched for the source special.
               ;           The assumption is made that no read or fetch
               ;           is done before the interrupt is detected.
               ;           After interrupt vectoring, the appropriate
               ;           register is written to BOOT memory and refetched.
               ;
               INTFTCH: CONT               & AM2901 ,,NOP,PASS,DZ
               /                           & IMM H 0020
       001B1   /                           & LDAER ;       ENABLE BOOT MEMORY
                        CONT               & AM2901 ,,NOP,PASS,DZ
               /                           & IMM H 0002
       001B2   /                           & LDMAR  ;    LOC 2 IN BOOT MEMORY
                        CONT               & AM2901 TAB,,NOP,PASS,DZ
               /                           & LDMDOR
```

```
ADDR    STATEMENT

001B3   /                                 & MWRITE ;     WRITE INST TO LOC 2
                    CONT                  & AM2901 ,PC,RAMF,ADD,ZB
        /                                 & CARRYH
        /                                 & SPHNOP
001B4   /                                 & IFETCH ;             REFETCH INST
                    JZ                    & AM2901 ,MAPX,NOP,PASS,ZB
        /                                 & LDAER
001B5   /                                 & MKLRON ; PUT EXEC MAP BACK IN
                                                          ADR EXT
        ;
        ;           PARITY ERROR INTERRUPT
        ;
        INTPARTY:   CONT                  & AM2901 ,,NOP,PASS,ZQ
        /                                 & ENCN CLRPEI
001B6   /                                 & SPFNOP ;              CLEAR ERROR
                    CALL  SETMAPS         & AM2901 ,,NOP,PASS,ZQ
        /                                 & SPHNOP
001B7   /                                 & SPFNOP ; SET DMS MAPS FOR INTERR
                    CONT                  & AM2901 ,R6,RAMF,PASS,DZ
        /                                 & IMM H 0005
001B8   /                                 & LDMAR  ;   FETCH FROM LOCATION 5
                    CONT                  & AM2901 ,,NOP,PASS,ZQ
        /                                 & ENCN CLRMPI
001B9   /                                 & IFETCH ; CLEAR POSSIBLE MEM PROT
        INTEXIT:    CALL  SETCIR          & AM2901 ,,NOP,PASS,ZQ
        /                                 & MKLRON
001BA   /                                 & SPHNOP ;     PUT SC IN HIGH BYTE
                    CONT                  & AM2901 ,,NOP,PASS,ZQ
        /                                 & ENCN SETTDI
001BB   /                                 & SPFNOP ;    TEMP INTERPT DISABLE
        ;
        ;           DECODE INSTRUCTION
        ;
        ; NOTE: TRAP CELL MUST CONTAIN A 1 WORD INSTRUCTION (JSB OR
              JMP)
        ;
                    LDCT                  & AM2901 ,R12,RAMF,PASS,DZ
        /                                 & SETMSR
        /                                 & JTAB LVL0
        /                                 & IRMRG
001BC   /                                 & LDMAR ;   FORCE A 2910 FAIL COND
                    JRP   $               & AM2901 R12,PC,RAMA,ADD,ZB
        /                                 & CONDMSR NSIGN
        /                                 & CARRYH
        /                                 & LDMAR
001BD   /                                 & SPFNOP ; MAP INST, MAR := MRG AD
        ;
        ;           TBG INTERRUPT
        ;
```

```
ADDR    STATEMENT

        INTTBG:   CALL   SETMAPS     & AM2901 ,,NOP,PASS,ZQ
        /                            & SPHNOP
001BE   /                            & SPFNOP ;
                  CONT               & AM2901 ,,NOP,PASS,ZQ
        /                            & ENCN CLRTBT
001BF   /                            & SPFNOP  ;        CLEAR TBG INT
                  CONT               & AM2901 ,R6,RAMF,PASS,DZ
        /                            & IMM H 0006
001C0   /                            & LDMAR ;        FETCH FROM LOC 6
                  JP     INTEXIT     & AM2901 ,,NOP,PASS,ZQ
        /                            & SPHNOP
001C1   /                            & IFETCH ;
        ;
        ;         I/O INTERRUPT
        ;
        INTIO:    CALL   SETMAPS     & AM2901 ,,NOP,PASS,ZQ
        /                            & SPHNOP
001C2   /                            & SPFNOP ;
                  CONT               & AM2901 ,,NOP,PASS,ZQ
        /                            & SPHNOP
001C3   /                            & MIAK   ;  ACKNOWLEDGE I/O INTRPT
                  CONT               & AM2901 ,R6,RAMF,PASS,DZ
        /                            & SPRD ECIRRD
001C4   /                            & SPHNOP ;   READ I/O SELECT CODE
                  JP     INTEXIT     & AM2901 ,,NOP,PASS,ZQ
        /                            & SPHNOP
001C5   /                            & SPFNOP ;
        ;
        ;         SLAVE REQUEST PSEUDO-INTERRUPT
        ;
        ;  - Simply handshake I/O control words.
        ;
        INTSLRQ:  CONT               & AM2901 ,,NOP,PASS,ZQ
        /                            & SPRD SLACK
001C6   /                            & SPHNOP ;          SLAVE ACK
        INTSLRQL: CALL   IOHSHAKE    & AM2901 ,,NOP,PASS,ZQ
        /                  16D       & SPHNOP
001C7   /                            & SPFNOP ;
                  CALL   IODECODE    & AM2901 ,R6,RAMF,PASS,DZ
        /                  174       & SPHNOP
001C8   /                            & SPFNOP ; SAVE CONTROL WORD IN R6
                  CONT               & AM2901 R6,R6,RAMF,AND,DA
        /                            & LODUSR
        /                            & IMM H 0100
001C9   /                            & SPHNOP ;          TEST LOOP BIT
                  CJP    INTSLRQL    & AM2901 ,,NOP,PASS,ZQ
        /                  1C7       & CONDUSR NZ
        /                            & SPHNOP
001CA   /                            & SPFNOP ;     IF LOOP BIT WAS SET
```

```
LINE    ADDR    STATEMENT

2813                        JP     IONOP        & AM2901 PC,PC,RAMA,ADD,ZB
2814            /                               & CARRYH
2815            /                               & LDMAR
2816   001CB    /                               & SPFNOP ;    GET NEXT INSTRUCTION
2817            ;
2818            ;           UNIMPLEMENTED INSTRUCTION TRAP
2819            ;
2820            INTUIT:     CJP    INTERRPT     & AM2901 ,,NOP,PASS,ZQ
2821            /                               & CONDEXT QPEI
2822            /                               & SPHNOP
2823   001CC    /                               & SPFNOP ; IF PENDING PARITY ERROR
2824                        CALL   SETMAPS      & AM2901 ,,NOP,PASS,ZQ
2825            /                               & SPHNOP
2826   001CD    /                               & MKLROFF ;
2827                        CONT                & AM2901 ,R6,RAMF,PASS,DZ
2828            /                               & IMM H 0008
2829   001CE    /                               & LDMAR ;    FETCH FROM LOCATION 8
2830                        JP     INTEXIT      & AM2901 ,PC,RAMF,SUBR,ZB
2831            /                               & CARRYH
2832            /                               & SPHNOP
2833   001CF    /                               & IFETCH ;  MAKE PC -> UIT LOC + 1
2834            ;
2835            ;           MEMORY PROTECT INTERRUPT
2836            ;
2837            INTPROT:    CONT                & AM2901 ,,NOP,PASS,ZQ
2838            /                               & ENCN CLRMPI
2839   001D0    /                               & SPFNOP ; CLEAR PROTECT VIOLATION
2840                        CALL   SETMAPS      & AM2901 ,,NOP,PASS,ZQ
2841            /                               & SPHNOP
2842   001D1    /                               & SPFNOP ;   SETUP MAPS FOR INTRPT
2843                        CONT                & AM2901 ,R6,RAMF,PASS,DZ
2844            /                               & IMM H 0007
2845   001D2    /                               & LDMAR ;         FETCH FROM LOC 7
2846                        JP     INTEXIT      & AM2901 ,,NOP,PASS,ZQ
2847            /                               & SPHNOP
2848   001D3    /                               & IFETCH ;        WRAPUP INTERRUPT
2849            ;
2850            ;           POWER FAIL WARNING INTERRUPT
2851            ;
2852            INTPFW:     CALL   SETMAPS      & AM2901 ,,NOP,PASS,ZQ
2853            /                               & SPHNOP
2854   001D4    /                               & SPFNOP ;
2855                        CONT                & AM2901 ,,NOP,PASS,ZQ
2856            /                               & ENCN CLRPFWI
2857   001D5    /                               & SPFNOP ;     CLEAR PFW INDICATOR
2858                        CALL   CLC04        & AM2901 ,,NOP,PASS,ZQ
2859            /                               & SPHNOP
2860   001D6    /                               & SPFNOP ;      INHIBIT INTERRUPTS
2861                        CONT                & AM2901 ,R6,RAMF,PASS,DZ
```

```
ADDR     STATEMENT

         /                         & IMM H 0004
001D7    /                         & LDMAR ;        FETCH FROM LOC 4
                   JP    INTEXIT   & AM2901 ,,NOP,PASS,ZQ
         /                         & SPHNOP
001D8    /                         & IFETCH ;       WRAPUP INTERRUPT
001D9              FILLER

         ;
         ;         SET MAPS FOR INTERRUPT HANDLING
         ;
         ;    - Save current working map set in IMAP
         ;
         ;    - Note: DATA1 set to EXEC before interrupt
         ;
         SETMAPS:  CALL  STWMAP    & AM2901 ,,NOP,PASS,ZQ
         /                         & SPHNOP
001DA    /                         & SPFNOP ;           R4 := WMAP
                   CONT            & AM2901 ,,NOP,PASS,DZ
         /                         & LDAER
001DB    /                         & IMM H 0020 ;  ENABLE BOOT MEMORY
                   CONT            & AM2901 ,,NOP,PASS,DZ
         /                         & LDMAR
001DC    /                         & IMM IMAPLOC ;    ADDRESS OF IMAP
                   CONT            & AM2901 MAPX,MAPD,RAMF,PASS,ZA
         /                         & ENCN CLRMPEN
001DD    /                         & SPFNOP ;       CLEAR MEM PROTECT
                   CONT            & AM2901 ,R4,NOP,PASS,ZB
         /                         & LDMDOR
001DE    /                         & MWRITE ; STORE WMAP INTO IMAP LO
                   RET             & AM2901 MAPX,MAPX,RAMF,AND,DA
         /                         & LDAER
001DF    /                         & IMM H FF20 ;       MAPX := MAP 0
         ;
         ;         SET CENTRAL INTERRUPT REGISTER
         ;
         ;    - VALUE FOR CIR IS PASSED IN BITS 0-7 OF R6
         ;
         SETCIR:   CONT            & AM2901 R6,R6,RAMF,AND,DA
         /                         & IMM H 003F
001E0    /                         & SPHNOP ;    MASK OFF SC (6 BITS)
                   CALL  BYTESWAP  & AM2901 ,R6,NOP,PASS,ZB
         /                         & SPHNOP
001E1    /                         & SPFNOP ;    PUT SC IN UPPER BYTE
                   CONT            & AM2901 MAPX,MAPX,RAMF,AND,DA
         /                         & IMM H 00FF
001E2    /                         & SPHNOP ;  MASK OUT SC FROM MAPX
                   RET             & AM2901 R6,MAPX,RAMF,OR,AB
         /                         & SPHNOP
001E3    /                         & SPFNOP ;         PUT SC INTO MAPX
```

```
ADDR    STATEMENT

        ;*************************************************************
        ;                                                            *
        ;          DFER/CFER/ZFER/XFER.                              *
        ;                                                            *
        ;*************************************************************
        ;
        ;          READ 1ST DEF & RESOLVE;  Q:=ADDR
        ;          R5 := Q; MAR:=PC
        ;          READ 2ND DEF & RESOLVE; Q:=ADDR.
        ;          B := R5
        ;          A := Q
        ;          PC := PC+2
        ;
        DCZFER:    CALL  WRTIND      & AM2901 ,,NOP,PASS,ZQ
        /                            & MREAD
001E4   /                            & SPHNOP ;
                   CALL  WRTIND      & AM2901 PC,R5,RAMA,PASS,ZQ
        /                            & MREAD
001E5   /                            & LDMAR ;
                   CONT              & AM2901 R5,B,RAMF,PASS,ZA
        /                            & SPFNOP
001E6   /                            & SPHNOP ;
                   CONT              & AM2901 ,A,RAMF,PASS,ZQ
        /                            & SPFNOP
001E7   /                            & SPHNOP ;
                   CONT              & AM2901 PC,PC,RAMF,ADD,DA
        /                            & CARRYL
        /                            & IMM H 0002
001E8   /                            & SPHNOP ;
        ;
        ;          CNTR := COUNT-1, DECODED FROM INSTRUCTION.
        ;            READ FROM (A); A:=A+1
        ;            MAR:=B; B:=B+1
        ;            WRITE DATA JUST READ.
        ;            LOOP FOR (CNTR-1) TIMES.
        ;
        .XFER:     PUSH              & AM2901 ,,NOP,PASS,ZQ
        /                            & JTAB WORDCNT
        /                            & SPFNOP
001E9   /                            & SPHNOP ;
                   CONT              & AM2901 A,A,RAMA,ADD,ZB
        /                            & CARRYH
        /                            & LDMAR
001EA   /                            & MREAD ;
                   CONT              & AM2901 B,B,RAMA,ADD,ZB
        /                            & CARRYH
        /                            & RSTMSR
        /                            & LDMAR
001EB   /                            & SPFNOP ;
```

```
ADDR    STATEMENT

                  TWB   NOSKIP      & AM2901 TAB,TAB,NOP,PASS,DZ
        /                           & CONDMSR Z
        /                           & LDMDOR
001EC   /                           & MWRITE ; TWB NEVER FALLS THRU,
                                                 MSR RESET
        ;
        ;         READ CENTRAL INTERRUPT REG INTO R6
        ;
        READCIR:  CALL  BYTESWAP    & AM2901 MAPX,R6,RAMF,PASS,ZA
        /                           & SPHNOP
001ED   /                           & SPFNOP ;
                  RET               & AM2901 R6,R6,RAMF,AND,DA
        /                           & IMM H 00FF
001EE   /                           & SPHNOP ;    MASK OFF SELECT CODE
        ;
        ;         FILL AREA
        ;
        IF        FILL
        LIST
        ENDIF


        ;
        ;***********************************************************
        ;*                                                         *
        ;*        Interrupt Vector Table                           *
        ;*                                                         *
        ;***********************************************************
        ;
001F0             ORG    H 1F0
        INTTBL:   EQU    $
        ;
        IF        NEWPC
                  JP     INTPON     & AM2901 ,,NOP,PASS,ZQ
        /                           & SPFNOP
001F0   /                           & SPHNOP ;     0 - POWER ON
                  JP     INTFTCH    & AM2901 ,PC,NOP,PASS,ZB
        /                           & LDMAR
001F1   /                           & MREAD  ;     1 - EXECUTE A OR B
                  JP     INTPARTY   & AM2901 ,,NOP,PASS,ZQ
        /                           & SPFNOP
001F2   /                           & SPHNOP ;      2 - PARITY ERROR
                  JP     INTPROT    & AM2901 ,,NOP,PASS,ZQ
        /                           & SPFNOP
001F3   /                           & SPHNOP ;      3 - MEMORY PROTECT
                  JP     INTSLRQ    & AM2901 ,,NOP,PASS,ZQ
        /                           & SPFNOP
001F4   /                           & SPHNOP ;      4 - SLAVE REQUEST
                  JP     INTPFW     & AM2901 ,,NOP,PASS,ZQ
```

```
LINE    ADDR    STATEMENT

3011            /                          & SPFNOP
3012    001F5   /                          & SPHNOP ;      5 - POWER FAIL
3013                    JP      INTTBG     & AM2901 ,,NOP,PASS,ZQ
3014            /                          & SPFNOP
3015    001F6   /                          & SPHNOP ;      6 - TBG
3016                    JP      INTIO      & AM2901 ,,NOP,PASS,ZQ
3017            /                          & SPFNOP
3018    001F7   /                          & SPHNOP ;      7 - I/O INTERRUPT
3019            ;
3020    001F8           FILLER
3021            ;
3022    001F9           FILLER
3023            ;
3024    001FA           FILLER
3025            ;
3026    001FB           FILLER
3027            ;
3028    001FC           FILLER
3029            ;
3030    001FD           FILLER
3031            ;
3032    001FE           FILLER
3033            ;
3034    001FF           FILLER
3035            ELSE
3080            ENDIF
```

```
ADDR     STATEMENT

         ;
         ;**************************************************************
         ;*                                                            *
         ;*          Index Register Group                              *
         ;*          --------------------                              *
         ;*                                                            *
         ;**************************************************************
         ;
00200              ORG   512
         LDX.:     CALL  WRTIND       & AM2901 ,,NOP,PASS,ZQ
         /                            & SPHNOP
00200    /                            & MREAD ;          FETCH ADDR WO
                   JP    SKIP         & AM2901 TAB,CXY,RAMF,PASS,DZ
         /                            & SPHNOP
00201    /                            & SPFNOP ;
         ADX.:     CALL  WRTIND       & AM2901 ,,NOP,PASS,ZQ
         /                            & SPHNOP
00202    /                            & MREAD ;
                   CONT               & AM2901 TAB,,QREG,PASS,DZ
         /                            & SPHNOP
```

```
ADDR     STATEMENT

00203    /                                & SPFNOP ;
                   JP     SKIP            & AM2901 CXY,CXY,RAMF,ADD,AQ
         /                                & CARRYL
         /                                & LODMSR
         /                                & ENVE  & ENBLC & ENBLO
00204    /                                & SPFNOP ;
         ;
         ;         COPY A/B TO/FROM X/Y
         ;
         COPYABXY: JZ                     & AM2901 CAB,CXY,RAMF,PASS,ZA
         /                                & SPHNOP
00205    /                                & IFETCH;
         COPYXYAB: JZ                     & AM2901 CXY,CAB,RAMF,PASS,ZA
         /                                & SPHNOP
00206    /                                & IFETCH;
         ;
         ;         DEC/INC X/Y AND SKIP IF ZERO
         ;
         DSXY:     LDCT   FETCH           & AM2901 CXY,CXY,RAMF,SUBR,ZA
         /                                & CARRYH
         /                                & LODUSR
         /                                & SPHNOP
00207    /                                & IFETCH;
                   JRP    SKIP            & AM2901 ,,NOP,PASS,ZQ
         /                                & CONDUSR Z
         /                                & SPHNOP
00208    /                                & SPFNOP ;
         ISXY:     LDCT   FETCH           & AM2901 CXY,CXY,RAMF,ADD,ZA
         /                                & CARRYH
         /                                & LODUSR
         /                                & SPHNOP
00209    /                                & IFETCH ;
                   JRP    SKIP            & AM2901 ,,NOP,PASS,ZQ
         /                                & CONDUSR Z
         /                                & SPHNOP
0020A    /                                & SPFNOP ;
         ;
         ;         LOAD A/B INDEXED BY X/Y
         ;
         LABXY:    CALL   WRTIND          & AM2901 ,,NOP,PASS,ZQ
         /                                & SPHNOP
0020B    /                                & MREAD ;  GET EFFECTIVE ADDR IN Q
                   CONT                   & AM2901 CXY,,NOP,ADD,AQ
         /                                & CARRYL
         /                                & LDMAR
0020C    /                                & MREAD ;      GEN INDEXED ADDRESS
                   CONT                   & AM2901 PC,PC,RAMA,ADD,ZB
         /                                & CARRYH
         /                                & LDMAR
```

```
ADDR     STATEMENT

0020D    /                            & SPFNOP ;         NEXT INST ADDR
                   JZ                 & AM2901 TAB,CAB,RAMF,PASS,DZ
         /                            & SPHNOP
0020E    /                            & IFETCH ;
         ;
         ;         STORE A/B INDEXED BY X/Y
         ;
         SABXY:    CALL  WRTIND       & AM2901 ,,NOP,PASS,ZQ
         /                            & SPHNOP
0020F    /                            & MREAD ;  GET EFFECTIVE ADDR IN Q
                   CONT               & AM2901 CXY,,NOP,ADD,AQ
         /                            & CARRYL
         /                            & LDMAR
00210    /                            & SPFNOP ;       GEN INDEXED ADDR
                   JP    SKIP         & AM2901 CAB,TAB,NOP,PASS,ZA
         /                            & LDMDOR
00211    /                            & MWRITE ; WRITE AND FETCH NEXT IN
         ;
         ;         EXCHANGE A/B WITH X/Y
         ;
         XABXY:    CONT               & AM2901 ,CAB,QREG,PASS,ZB
         /                            & SPHNOP
00212    /                            & SPFNOP ;
                   CONT               & AM2901 CXY,CAB,RAMF,PASS,ZA
         /                            & SPHNOP
00213    /                            & IFETCH ;
                   JZ                 & AM2901 ,CXY,RAMF,PASS,ZQ
         /                            & SPHNOP
00214    /                            & SPFNOP ;
         ;
         ;         STX - STORE X/Y TO MEMORY
         ;
         STX.:     CALL  WRTIND       & AM2901 ,,NOP,PASS,ZQ
         /                            & SPHNOP
00215    /                            & MREAD ;        READ ADDRESS WORD
                   JP    SKIP         & AM2901 CXY,TAB,NOP,PASS,ZA
         /                            & LDMDOR
00216    /                            & MWRITE ;    WRITE X/Y TO MEMORY
         ;
         ;         JLY - JUMP AND LOAD Y
         ;
         JLY:      CALL  WRTIND       & AM2901 PC,Y,RAMF,PASS,ZA
         /                            & SPHNOP
00217    /                            & MREAD ;        READ ADDRESS WORD
                   JP    SKIP         & AM2901 ,PC,RAMF,PASS,ZQ
         /                            & SPFNOP
00218    /                            & SPHNOP ;  PC LOADED WITH ADDRESS
         ;
         ;         JPY - JUMP INDEXED BY Y
```

```
LINE   ADDR     STATEMENT

                ;
                ;   - PC := Operand Addr + Y
                ;   - Indirection IS allowed for the operand
                ;
                JPY:      CALL  WRTIND      & AM2901 ,,NOP,PASS,ZQ
                /                           & SPHNOP
       00219    /                           & MREAD ;   READ ADDRESS WORD INTO
                          JP    SKIP        & AM2901 Y,PC,RAMF,ADD,AQ
                /                           & CARRYL
                /                           & SPHNOP
       0021A    /                           & SPFNOP ;  COMPUTE ADDR AND FETCH

                ;
                ;*************************************************************
                ;*                                                           *
                ;*        Bit Manipulation Instructions                      *
                ;*        -----------------------------                      *
                ;*                                                           *
                ;* Format is:   BIT-OPCODE                                   *
                ;*              DEF MASK                                     *
                ;*              DEF TARGET                                   *
                ;*                                                           *
                ;* Bit opcodes are:                                          *
                ;*        CBS - clear bits                                   *
                ;*        SBS - set bits                                     *
                ;*        TBS - test bits                                    *
                ;*                                                           *
                ;*************************************************************
                ;
                CBS:      CALL  BITSB       & AM2901 ,,NOP,PASS,ZQ
                /                           & SPHNOP
       0021B    /                           & MREAD  ;           GET OPERANDS
                          JP    BITWRT      & AM2901 R4,R5,RAMF,NOTRS,AB
                /                           & SPHNOP
       0021C    /                           & SPFNOP ;         CLEAR THE BITS
                SBS:      CALL  BITSB       & AM2901 ,,NOP,PASS,ZQ
                /                           & SPHNOP
       0021D    /                           & MREAD  ;           GET OPERANDS
                          CONT              & AM2901 R4,R5,RAMF,OR,AB
                /                           & SPHNOP
       0021E    /                           & SPFNOP ;           SET THE BITS
                BITWRT:   JP    SKIP        & AM2901 R5,TAB,NOP,PASS,ZA
                /                           & LDMDOR
       0021F    /                           & MWRITE ; WRITE RESULT BACK TO ME
                TBS:      CALL  BITSB       & AM2901 ,,NOP,PASS,ZQ
                /                           & SPHNOP
       00220    /                           & MREAD  ;           GET OPERANDS
                          CONT              & AM2901 R4,R5,RAMF,AND,AB
                /                           & SPHNOP
```

```
LINE   ADDR    STATEMENT

       00221   /                             & SPFNOP ;           SELECT BITS
                         CONT                & AM2901 R4,R5,RAMF,EXOR,AB
               /                             & LODUSR
               /                             & SPHNOP
       00222   /                             & SPFNOP ;       TEST FOR ALL ONES
                         CJP   SKIP          & AM2901 ,,NOP,PASS,ZQ
               /                             & CONDUSR Z
               /                             & SPHNOP
       00223   /                             & SPFNOP ;           IF ALL ONES
                         JP    SKIP          & AM2901 ,PC,RAMF,ADD,ZB
               /                             & CARRYH
               /                             & SPHNOP
       00224   /                             & SPFNOP ;   SKIP NEXT INSTRUCTION
               ;
               ;         BITSB - TWO OPERAND DEF RESOLVER FOR BIT INST, ETC.
               ;
               ;    - R4: LOADED WITH RESOLVED FIRST DEF
               ;      R5: LOADED WITH RESOLVED SECOND DEF
               ;      PC: POINTS TO NEXT INSTRUCTION (FOR EXIT THRU SKIP)
               ;
               BITSB:    CALL  WRTIND        & AM2901 ,,NOP,PASS,ZQ
               /                 109         & SPHNOP
       00225   /                             & SPFNOP ;           RESOLVE MASK
                         CONT                & AM2901 TAB,R4,RAMF,PASS,DZ
               /                             & SPHNOP
       00226   /                             & SPFNOP ;     R4 := FIRST OPERAND
                         CALL  WRTIND        & AM2901 PC,PC,RAMA,PASS,ZB
               /                             & LDMAR
       00227   /                             & MREAD ;          RESOLVE TARGET
                         CONT                & AM2901 TAB,R5,RAMF,PASS,DZ
               /                             & SPHNOP
       00228   /                             & SPFNOP ;    R5 := SECOND OPERAND
                         RET                 & AM2901 ,PC,RAMF,ADD,ZB
               /                             & CARRYH
               /                             & SPHNOP
       00229   /                             & SPFNOP ;           PC := PC + 1
```

```
LINE   ADDR    STATEMENT

               ;
               ;***********************************************************
               ;*                                                         *
               ;*       Byte Manipulation Instructions                    *
               ;*       ------------------------------                    *
               ;*                                                         *
               ;*    All of these instructions use a byte address. A byte *
               ;* address is two times the normal address plus 0 or 1 if  *
               ;* the high (bits 8-15) or low (bits 0-7) byte is desired. *
               ;*                                                         *
               ;* Notes: R4 contains byte address                         *
               ;*        R5 contains a H 00FF to mask off a byte          *
               ;*        R6 contains byte to load or to store             *
               ;*                                                         *
               ;***********************************************************
               ;
               LBT:      CONT                & AM2901 B,R4,SRAMR,PASS,ZA
               /                             & SHIFT ROTATE
               /                             & SPHNOP
       0022A   /                             & SPFNOP ; SHIFT BYTE ADDR CIRCULA
                         CALL  LDBYTE        & AM2901 R4,R4,RAMA,PASS,ZB
               /                             & LODMSR
               /                             & LDMAR
       0022B   /                             & MREAD ;      READ WORD WITH BYTE
                         CONT                & AM2901 ,PC,NOP,SUBR,ZB
               /                             & CARRYH
               /                             & LDMAR
       0022C   /                             & IFETCH ;  FETCH NEXT INSTRUCTION
                         CONT                & AM2901 ,B,RAMF,ADD,ZB
               /                             & CARRYH
               /                             & SPHNOP
       0022D   /                             & SPFNOP ;        INC BYTE ADDRESS
                         JZ                  & AM2901 R6,A,RAMF,PASS,ZA
               /                             & SPHNOP
       0022E   /                             & SPFNOP ;
               ;
               SBT:      CONT                & AM2901 ,R6,RAMF,PASS,DZ
               /                             & IMM H 00FF
       0022F   /                             & SPHNOP ;  MASK FOR BYTE TO STORE
                         CALL  STBYTE        & AM2901 A,R6,RAMF,AND,AB
               /                             & SPHNOP
       00230   /                             & SPFNOP ;     R6 := BYTE TO STORE
               NOSKIP:   JZ                  & AM2901 ,PC,NOP,SUBR,ZB
               /                             & CARRYH
               /                             & LDMAR
       00231   /                             & IFETCH ;
               ;
               ;         Byte Common Subroutines
```

```
ADDR     STATEMENT

         ;
         ;  LDBYTE   - Load byte into R6, right justified
         ;  STBYTE   - Store byte
         ;  BYTESWAP - Swap bytes in R6
         ;
         LDBYTE:   CONT                & AM2901 ,R5,RAMF,PASS,DZ
         /                             & IMM H 00FF
00232    /                             & SPHNOP ;     R5 := MASK FOR BYTE
                   CCALL BYTESWAP      & AM2901 TAB,R6,RAMF,PASS,DZ
         /                             & CONDMSR NSIGN
         /                             & SPHNOP
00233    /                             & SPFNOP ;       IF EVEN BYTE ADDR
                   RET                 & AM2901 R5,R6,RAMF,AND,AB
         /                             & SPHNOP
00234    /                             & SPFNOP ;              R6 := BYTE
         ;
         ;         STBYTE - READ, MODIFY AND WRITE BYTE
         ;
         STBYTE:   CONT                & AM2901 B,R4,SRAMR,PASS,ZA
         /                             & SHIFT ROTATE
         /                             & SPFNOP
00235    /                             & SPHNOP ; SHIFT BYTE ADDR CIRCULA
                   CONT                & AM2901 R4,R4,RAMA,PASS,ZB
         /                             & LODMSR
         /                             & LDMAR
00236    /                             & MREAD ;      READ WORD FOR STORE
                   CONT                & AM2901 ,R5,RAMF,PASS,DZ
         /                             & IMM H 00FF
00237    /                             & SPHNOP ; MASK FOR (MEM) TO UPDAT
                   CJP   STBYTEOD      & AM2901 TAB,,QREG,PASS,DZ
         /                             & CONDMSR SIGN
         /                             & SPHNOP
00238    /                             & SPFNOP ; Q := (MEM); IF ODD BYTE
         ;
         ;         STORE IS TO EVEN BYTE
         ;
         ;    - SWAP BYTES IN R6, TO POSITION NEW BYTE
         ;    - UPDATE BITS 8-15
         ;
                   CALL  BYTESWAP      & AM2901 R5,,QREG,AND,AQ
         /                             & SPHNOP
00239    /                             & SPFNOP ;       MASK BYTE TO SAVE
         STBYTEWR: CONT                & AM2901 R6,TAB,NOP,OR,AQ
         /                             & LDMDOR
0023A    /                             & MWRITE ; STORE WORD BACK IN MEMO
                   RET                 & AM2901 ,B,RAMF,ADD,ZB
         /                             & CARRYH
         /                             & SPHNOP
0023B    /                             & SPFNOP ;       BUMP BYTE ADDRESS
```

```
LINE   ADDR    STATEMENT

               ;
               ;          STORE IS TO ODD BYTE
               ;
               ;   - UPDATE BITS 0-7
               ;   - MASK TO PRESERVE BYTE IS COMPLEMENT OF EVEN BYTE MASK
               ;
               STBYTEOD: JP    STBYTEWR   & AM2901 R5,,QREG,NOTRS,AQ
               /                          & SPHNOP
       0023C   /                          & SPFNOP ;      MASK BYTE TO SAVE
               ;
               ;          BYTE SWAP R6
               ;
               BYTESWAP: CONT             & AM2901 R6,R6,RAMA,PASS,DZ
               /                          & SPHNOP
       0023D   /                          & L4D ;               BYTE SWAP R6
                         RET              & AM2901 R6,R6,RAMA,PASS,DZ
               /                          & SPHNOP
       0023E   /                          & L4D ;

               ;
               ;***********************************************************
               ;*                                                         *
               ;*         Word Manipulation Instructions                  *
               ;*         ------------------------------                  *
               ;*                                                         *
               ;*  These are the old MEF Series interruptible instructions.*
               ;* A zero word must follow the instruction which is used to *
               ;* store a residual count. The from address is in the A reg,*
               ;* the to address is in the B-reg and the count is pointed  *
               ;* to by the second word of the instruction.               *
               ;*                                                         *
               ;***********************************************************
               ;
               MVW:      CALL  INITIAL    & AM2901 ,,NOP,PASS,ZQ
               /                          & SPHNOP
       0023F   /                          & MREAD  ;
               LMVW:     LDCT  $          & AM2901 A,A,RAMA,ADD,ZB
               /                          & CARRYH
               /                          & LDMAR
       00240   /                          & MREAD ;        READ SOURCE WORD
                         CONT             & AM2901 B,B,RAMA,ADD,ZB
               /                          & CARRYH
               /                          & LDMAR
       00241   /                          & SPFNOP ; MAR := DESTINATION ADDR
                         CONT             & AM2901 TAB,TAB,NOP,PASS,DZ
               /                          & LDMDOR
       00242   /                          & MWRITE ;  WRITE DESTINATION WORD
                         CJP   INTPEND    & AM2901 ,R12,RAMF,SUBR,ZB
               /                          & CARRYH
```

```
LINE   ADDR    STATEMENT

               /                            & CONDEXT INTRPT
               /                            & SPHNOP
       00243   /                            & SPFNOP ;    IF INTERRUPT PENDING
                          JRP   SKIP        & AM2901 ,,NOP,PASS,ZQ
               /                            & CONDUSR Z
               /                            & SPHNOP
       00244   /                            & SPFNOP ;         IF COUNT IS ZERO
               ;
               ;          CMW - COMPARE WORDS
               ;
               CMW:       CALL  INITIAL     & AM2901 ,,NOP,PASS,ZQ
               /                            & SPHNOP
       00245   /                            & MREAD  ;
               LCMW:      LDCT  $           & AM2901 A,A,RAMA,ADD,ZB
               /                            & CARRYH
               /                            & LDMAR
       00246   /                            & MREAD ;         READ ARRAY 1 WORD
                          CONT              & AM2901 B,B,RAMA,ADD,ZB
               /                            & CARRYH
               /                            & LDMAR
       00247   /                            & SPFNOP ;     MAR := ARRAY 2 ADDR
                          CONT              & AM2901 TAB,R5,RAMF,PASS,DZ
               /                            & SPHNOP
       00248   /                            & MREAD ; R5 := WORD1, READ ARRAY
                                                          2 WORD
                          CONT              & AM2901 TAB,R6,RAMF,PASS,DZ
               /                            & SPHNOP
       00249   /                            & MREAD  ; R6 := WORD 2 (FORCE
                                                          FREEZE)
                          CONT              & AM2901 R6,R5,NOP,SUBR,AB
               /                            & CARRYL
               /                            & LODUSR
               /                            & SPHNOP
       0024A   /                            & SPFNOP ; COND REG := WORD1-WORD2
                          CJP   CMWNEQ      & AM2901 R6,R5,NOP,SUBR,AB
               /                            & CARRYL
               /                            & CONDUSR NZ
               /                            & SPHNOP
       0024B   /                            & SPFNOP ;            IF NOT EQUAL
                          CJP   INTPEND     & AM2901 ,R12,RAMF,SUBR,ZB
               /                            & CARRYH
               /                            & CONDEXT INTRPT
               /                            & SPHNOP
       0024C   /                            & SPFNOP ;    IF INTERRUPT PENDING
                          JRP   SKIP        & AM2901 ,,NOP,PASS,ZQ
               /                            & CONDUSR Z
               /                            & SPHNOP
       0024D   /                            & SPFNOP ;        IF COUNT IS ZERO
               ;
```

```
LINE    ADDR    STATEMENT

3479            CMWNEQ:   CJP   CMWLT       & AM2901 ,PC,RAMF,ADD,ZB
3480            /                           & CARRYH
3481            /                           & CONDUSR LT
3482            /                           & SPHNOP
3483    0024E   /                           & SPFNOP ;         ' INC PC ONCE
3484                      CONT              & AM2901 ,PC,RAMF,ADD,ZB
3485            /                           & CARRYH
3486            /                           & SPHNOP
3487    0024F   /                           & SPFNOP ;           INC PC TWICE
3488            CMWLT:    CONT              & AM2901 R12,B,RAMF,ADD,AB
3489            /                           & CARRYL
3490            /                           & SPHNOP
3491    00250   /                           & SPFNOP ; B POINTS PAST END OF AR
3492                      CONT              & AM2901 ,B,RAMF,SUBR,ZB
3493            /                           & CARRYH
3494            /                           & SPHNOP
3495    00251   /                           & SPFNOP ;
3496                      JP    SKIP        & AM2901 ,A,RAMF,SUBR,ZB
3497            /                           & CARRYH
3498            /                           & SPHNOP
3499    00252   /                           & SPFNOP;    A POINTS TO LAST WORD
3500            ;
3501            ;         MBT - MOVE BYTES
3502            ;
3503            MBT:      CALL  INITIAL     & AM2901 ,,NOP,PASS,ZQ
3504            /                           & SPHNOP
3505    00253   /                           & MREAD  ;
3506                      LDCT  LMBT        & AM2901 ,,NOP,PASS,ZQ
3507            /                           & SPHNOP
3508    00254   /                           & SPFNOP ;      SETUP ADDR FOR JRP
3509            LMBT:     CONT              & AM2901 A,R4,SRAMR,PASS,ZA
3510            /                           & SHIFT ROTATE
3511            /                           & SPHNOP
3512    00255   /                           & SPFNOP ;        MAKE R4 WORD ADDR
3513                      CALL  LDBYTE      & AM2901 R4,R4,RAMA,PASS,ZB
3514            /                           & LODMSR
3515            /                           & LDMAR
3516    00256   /                           & MREAD ;  READ WORD CONTAINING BY
3517                      CALL  STBYTE      & AM2901 ,A,RAMF,ADD,ZB
3518            /                           & CARRYH
3519            /                           & SPHNOP
3520    00257   /                           & SPFNOP ;  INC A (STBYTE INC'S B)
3521                      CJP   INTPEND     & AM2901 ,R12,RAMF,SUBR,ZB
3522            /                           & CARRYH
3523            /                           & CONDEXT INTRPT
3524            /                           & SPHNOP
3525    00258   /                           & SPFNOP ;    IF INTERRUPT PENDING
3526                      JRP   SKIP        & AM2901 ,,NOP,PASS,ZQ
3527            /                           & CONDUSR Z
```

```
ADDR    STATEMENT

        /                          & SPHNOP
00259   /                          & SPFNOP ;        IF COUNT IS ZERO
        ;
        ;         CBT - COMPARE BYTES
        ;
        CBT:      CALL  INITIAL    & AM2901 ,,NOP,PASS,ZQ
        /                          & SPHNOP
0025A   /                          & MREAD  ;
        LCBT:     CONT             & AM2901 A,R4,SRAMR,PASS,ZA
        /                          & SHIFT ROTATE
        /                          & SPHNOP
0025B   /                          & SPFNOP ;
                  CALL  LDBYTE     & AM2901 ,R4,NOP,PASS,ZB
        /                          & LODMSR
        /                          & LDMAR
0025C   /                          & MREAD ;         READ BYTE ARRAY 1
                  CONT             & AM2901 ,A,RAMF,ADD,ZB
        /                          & CARRYH
        /                          & SPHNOP
0025D   /                          & SPFNOP ;             INC A
                  LDCT  LCBT       & AM2901 R6,R7,RAMF,PASS,ZA
        /                          & SPHNOP
0025E   /                          & SPFNOP ;             R7 := BYTE 1
                  CONT             & AM2901 B,R4,SRAMR,PASS,ZA
        /                          & SHIFT ROTATE
        /                          & SPHNOP
0025F   /                          & SPFNOP ;
                  CALL  LDBYTE     & AM2901 ,R4,NOP,PASS,ZB
        /                          & LODMSR
        /                          & LDMAR
00260   /                          & MREAD ;         READ BYTE ARRAY 2
                  CONT             & AM2901 ,B,RAMF,ADD,ZB
        /                          & CARRYH
        /                          & SPHNOP
00261   /                          & SPFNOP ;             INC B
                  CONT             & AM2901 R6,R7,NOP,SUBR,AB
        /                          & CARRYL
        /                          & LODUSR
        /                          & SPHNOP
00262   /                          & SPFNOP ; COND REG := BYTE1 - BYT
                  CJP   CMWNEQ     & AM2901 R6,R7,NOP,SUBR,AB
        /                          & CARRYL
        /                          & CONDUSR NZ
        /                          & SPHNOP
00263   /                          & SPFNOP ;    IF STRINGS NOT EQUAL
                  CJP   INTPEND    & AM2901 ,R12,RAMF,SUBR,ZB
        /                          & CARRYH
        /                          & CONDEXT INTRPT
        /                          & SPHNOP
```

```
ADDR    STATEMENT

00264   /                             & SPFNOP ;    IF INTERRUPT PENDING
                  JRP   SKIP          & AM2901 ,,NOP,PASS,ZQ
        /                             & CONDUSR Z
        /                             & SPHNOP
00265   /                             & SPFNOP ;        IF COUNT IS ZERO
        ;
        ;         SFB - SCAN FOR BYTE
        ;
        SFB:      CONT                & AM2901 A,R7,RAMF,AND,DA
        /                             & IMM H 00FF
00266   /                             & SPHNOP ;          R7 := TEST BYTE
                  CALL  BYTESWAP      & AM2901 A,R6,RAMF,PASS,ZA
        /                             & SPHNOP
00267   /                             & SPFNOP ;
                  CONT                & AM2901 R6,R8,RAMF,AND,DA
        /                             & IMM H 00FF
00268   /                             & SPHNOP ;  R8 := TERMINATION BYTE
        LSFB:     CONT                & AM2901 B,R4,SRAMR,PASS,ZA
        /                             & SHIFT ROTATE
        /                             & SPHNOP
00269   /                             & SPFNOP ;       MAKE WORD ADDRESS
                  CALL  LDBYTE        & AM2901 R4,R4,RAMA,PASS,ZB
        /                             & LODMSR
        /                             & LDMAR
0026A   /                             & MREAD ;  READ WORD CONTAINING BY
                  LDCT  LSFB          & AM2901 R6,R7,NOP,EXOR,AB
        /                             & LODMSR
        /                             & SPHNOP
0026B   /                             & SPFNOP ;    COMPARE TO TEST BYTE
                  CJP   NOSKIP        & AM2901 R6,R8,NOP,EXOR,AB
        /                             & CONDLMSR Z
        /                             & SPHNOP
0026C   /                             & SPFNOP ;    IF TEST BYTE MATCHES
                  CJP   SKIP          & AM2901 ,B,RAMF,ADD,ZB
        /                             & CARRYH
        /                             & CONDMSR Z
        /                             & SPHNOP
0026D   /                             & SPFNOP ; IF TERMINATION BYTE MAT
                  JRP   INTERRPT      & AM2901 ,,NOP,PASS,ZQ
        /                             & CONDEXT INTRPT
        /                             & SPHNOP
0026E   /                             & SPFNOP ;    IF INTERRUPT PENDING
```

```
LINE   ADDR    STATEMENT

                ;
                ;          Word Manipulation Subroutines
                ;
                ;          INITIAL - Initialize R12 with the proper word count
                ;
                INITIAL:   CALL  WRTIND      & AM2901 ,,NOP,PASS,ZQ
                /                            & SPHNOP
       0026F    /                            & SPFNOP ;   RESOLVE COUNT
                           CONT              & AM2901 TAB,R12,RAMF,PASS,DZ
                /                            & LODMSR
                /                            & SPHNOP
       00270    /                            & SPFNOP ;           R12 := COUNT
                           CJP   SKIP        & AM2901 PC,PC,RAMA,ADD,ZB
                /                            & CARRYH
                /                            & CONDMSR Z
                /                            & LDMAR
       00271    /                            & MREAD  ; READ RESIDUAL, EXIT IF
                                                          COUNT == 0
                           CONT              & AM2901 TAB,,QREG,PASS,DZ
                /                            & LODMSR
                /                            & SPHNOP
       00272    /                            & MREAD  ; FREEZE FOR RESIDUAL WOR
                           CRET              & AM2901 ,,NOP,AND,ZQ
                /                            & CONDMSR Z
                /                            & LDMDOR
       00273    /                            & MWRITE ; IF RESIDUAL WAS ZERO &
                                                            ZERO IT
                           RET               & AM2901 ,R12,RAMF,PASS,ZQ
                /                            & SPHNOP
       00274    /                            & SPFNOP ;
                ;
                ;   INTPEND - Interrupt pending during block type instruction
                ;
                ;  NOTE: It is possible for interrupt to happen on last word.
                ;        Therefore, check for zero before doing interrupt.
                ;
                INTPEND:   CJP   SKIP        & AM2901 ,,NOP,PASS,ZQ
                /                            & CONDUSR Z
                /                            & SPHNOP
       00275    /                            & SPFNOP ; IF BLOCK INST JUST FINI
                           CONT              & AM2901 ,PC,RAMF,SUBR,ZB
                /                            & CARRYH
                /                            & LDMAR
       00276    /                            & SPFNOP ;           BACK UP PC
                           JP    INTERRPT    & AM2901 R12,TAB,NOP,PASS,ZA
                /                            & LDMDOR
       00277    /                            & MWRITE ; WRITE RESIDUAL COUNT TO
```

```
LINE    ADDR    STATEMENT

                ;**************************************************************
                ;*                                                            *
                ;*        DMS Map Feature                                     *
                ;*        ---------------                                     *
                ;*                                                            *
                ;*  - There are 32 sets of 32 map registers. The map set is  *
                ;*    selected by the Address Extention Reg.  The actual     *
                ;*    map reg is selected by bits 10 to 14 of the MAR.       *
                ;*    The AER is loaded from bits 0-5 of the Y bus.  Ex:     *
                ;*                                                            *
                ;*    +-----+-----+-----+-----+-----+-----+                  *
                ;*    | ROM |       Map Set Number        |  AER             *
                ;*    +-----+-----+-----+-----+-----+-----+                  *
                ;*                                                            *
                ;*    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+      *
                ;*    |  | Map Reg Num  |          Offset             |  MAR*
                ;*    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+      *
                ;*                                                            *
                ;*  - The Macro machine can access two map sets directly.    *
                ;*    They are called Execute map and Data 1 map. These map  *
                ;*    numbers are kept in two 2901 registers:                *
                ;*                                                            *
                ;*    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+      *
                ;*    |                       |      Data 1 Map       | MAPD*
                ;*    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+      *
                ;*                                                            *
                ;*    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+      *
                ;*    |                       |       Exec Map        | MAPX*
                ;*    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+      *
                ;*                                                            *
                ;*                                                            *
                ;*  - The MAPD and MAPX registers are loaded from WMAP or    *
                ;*    the Working MAP set.  WMAP has the format:             *
                ;*                                                            *
                ;*    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+      *
                ;*    |MP|              |   DATA1      |     EXEC     | WMAP*
                ;*    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+      *
                ;*                                                            *
                ;*    Where: MP     - Memory Protect enable                   *
                ;*           DATA1  - Data 1 map number                       *
                ;*           EXEC   - Execute map number                      *
                ;*                                                            *
                ;**************************************************************
                ;
```

```
LINE    ADDR    STATEMENT

                ;
                ;**********************************************************
                ;*                                                        *
                ;*       Priviledged DMS Instructions                     *
                ;*       ----------------------------                     *
                ;*                                                        *
                ;* - The following DMS instructions are priviledged:      *
                ;*                                                        *
                ;*       LPMR - Load Page Mapping Register                *
                ;*       SPMR - Store Page Mapping Register               *
                ;*       LMAP - Load Map                                  *
                ;*       SMAP - Store Map                                 *
                ;*       LWD1 - Load Data 1 map number                    *
                ;*       SWMP - Store Working Map set                     *
                ;*       SIMP - Store Interrupt Map set                   *
                ;*       XJMP - Cross map JMP                             *
                ;*                                                        *
                ;* - If any of these instructions are executed while memory *
                ;*   protect is enabled, a memory protect violation will be *
                ;*   generated by the firmware.  After the instruction is  *
                ;*   decoded, a test is made for Memory Protect enable.  If *
                ;*   true, generate protect violation.                    *
                ;*                                                        *
                ;**********************************************************
                ;
```

```
LINE   ADDR    STATEMENT

                ;
                ;*****************************************************
                ;*                                                   *
                ;*        XL.1                                       *
                ;*        DEF  ADDRESS                               *
                ;*                                                   *
                ;*        CROSS LOAD A/B FROM ALTERNATE MAP 1        *
                ;*        INDIRECTS ARE RESOLVED IN THE EXECUTE MAP  *
                ;*        THE FINAL REFERENCE TAKES PLACE WITH BOOT  *
                ;*        MEMORY AND A/B ADDRESSABILITY TURNED OFF   *
                ;*                                                   *
                ;*****************************************************
                ;
                XL.10:    CALL  WRTIND     & AM2901 ,,NOP,PASS,ZQ
                /                          & SPHNOP
       00278    /                          & MREAD ;        READ ADDRESS WORD
                          CONT             & AM2901 ,MAPD,NOP,PASS,ZB
                /                          & LDAER
       00279    /                          & MREAD ;  START A READ IN DATA1 M
                          CONT             & AM2901 ,MAPX,NOP,PASS,ZB
                /                          & LDAER
       0027A    /                          & SPFNOP;  PUT EXECUTE MAP INTO AE
                          JP    SKIP       & AM2901 ,CAB,RAMF,PASS,DZ
                /                          & SPHNOP
       0027B    /                          & SPFNOP ;      LOAD A/B WITH DATA
                ;
                ;*****************************************************
                ;*                                                   *
                ;*        XC.1                                       *
                ;*        DEF  ADDRESS                               *
                ;*                                                   *
                ;*        CROSS COMPARE A/B WITH LOCATION IN         *
                ;*        ALTERNATE MAP 1.  ADDRESS POINTS TO        *
                ;*        A LOCATION IN DATA1 MAP.                   *
                ;*                                                   *
                ;*****************************************************
                ;
                XC.10:    CALL  WRTIND     & AM2901 ,,NOP,PASS,ZQ
                /                          & SPHNOP
       0027C    /                          & MREAD ;          RESOLVE ADDRESS
                          CONT             & AM2901 ,MAPD,NOP,PASS,ZB
                /                          & LDAER
       0027D    /                          & MREAD  ;     READ FROM DATA1 MAP
                          CONT             & AM2901 ,MAPX,NOP,PASS,ZB
                /                          & LDAER
       0027E    /                          & SPFNOP ;     PUT EXEC MAP IN AER
                          CONT             & AM2901 PC,PC,RAMA,ADD,ZB
                /                          & CARRYH
                /                          & LDMAR
```

```
ADDR    STATEMENT

0027F   /                          & SPFNOP ;
                  JP    CP.1       & AM2901 ,R4,RAMF,PASS,DZ
        /                          & SPHNOP
00280   /                          & IFETCH ; R4 := DATA1 WORD, FETCH
        ;
        ;********************************************************
        ;*                                                      *
        ;*        XS.1                                          *
        ;*        DEF  ADDRESS                                  *
        ;*                                                      *
        ;*        CROSS STORE A/B THROUGH MAP 1                 *
        ;*        INDIRECTS ARE RESOLVE IN THE EXECUTE MAP      *
        ;*        A/B ADDRESSABILITY AND BOOT MODE ARE TURNED   *
        ;*        OFF FOR THE FINAL REFERENCE (STORE)           *
        ;*                                                      *
        ;********************************************************
        ;
        XS.10:    CALL  WRTIND     & AM2901 ,,NOP,PASS,ZB
        /                          & SPHNOP
00281   /                          & MREAD ;        READ ADDRESS WORD
                  CONT             & AM2901 ,MAPD,NOP,PASS,ZB
        /                          & LDAER
00282   /                          & SPFNOP;  LOAD AER WITH DATA1 MAP
                  CONT             & AM2901 ,CAB,NOP,PASS,ZB
        /                          & LDMDOR
00283   /                          & MWRITE ;   WRITE DATA WITH DATA1
                  JP    SKIP       & AM2901 ,MAPX,NOP,PASS,ZB
        /                          & LDAER
00284   /                          & SPFNOP;      PUT EXEC MAP IN AER

        ;
        ;*****************************************************
        ;*                                                   *
        ;*        LWD1                                       *
        ;*        DEF NEWDATA1                               *
        ;*                                                   *
        ;*        LOADS THE DATA1 MAP PORTION OF MAPD REG    *
        ;*        FROM NEWDATA1.  THE LOWER 8 BITS OF        *
        ;*        THE DATA WORD ARE LOADED INTO MAPD.        *
        ;*                                                   *
        ;*****************************************************
        ;
        LWD10:    CJP   GENMPV     & AM2901 ,,NOP,PASS,ZQ
        /                          & CONDEXT MPEN
        /                          & SPHNOP
00285   /                          & SPFNOP ;     IF MEM PROT ENABLED
                  CALL  WRTIND     & AM2901 ,,NOP,PASS,ZQ
        /                          & SPHNOP
00286   /                          & MREAD ;         RESOLVE NEWDATA1
```

```
ADDR    STATEMENT

                       CONT                  & AM2901 PC,PC,RAMA,ADD,ZB
        /                                    & CARRYH
        /                                    & LDMAR
00287   /                                    & SPFNOP ;       ADDR OF NEXT INST
                       CONT                  & AM2901 TAB,R4,RAMF,PASS,DZ
        /                                    & SPHNOP
00288   /                                    & IFETCH ;   R4 := NEW DATA1 VALUE
                       JZ                    & AM2901 R4,MAPD,RAMF,AND,DA
        /                                    & SPHNOP
00289   /                                    & IMM H 00FF ; SET NEW DATA1 IN
                                                                 MAPD
        ;
        ;*****************************************************
        ;*                                                   *
        ;*             LPMR                                  *
        ;*                                                   *
        ;*             LOAD PAGE MAPPING REG ADDRESSED BY A REG   *
        ;*             FROM B REG.  A REG IS INCREMENTED     *
        ;*                                                   *
        ;*****************************************************
        ;
        LPMR0:     CJP    GENMPV             & AM2901 ,,NOP,PASS,ZQ
        /                                    & CONDEXT MPEN
        /                                    & SPHNOP
0028A   /                                    & SPFNOP ;  IF MEM PROTECT ENABLED
                   CALL   POSIMAPR           & AM2901 ,A,RAMF,ADD,ZB
        /                                    & CARRYH
        /                                    & SPHNOP
0028B   /                                    & SPFNOP ; POSITION MAP REG NUMBER
                   CONT                      & AM2901 ,R4,RAMF,PASS,DZ
        /                                    & IMM H 7FFF
0028C   /                                    & SPHNOP ;      MASK FOR READ PROT
                   CONT                      & AM2901 R4,B,NOP,AND,AB
        /                                    & SPWR MAPWR
0028D   /                                    & SPFNOP ; MASK OFF READ PROT AND
                                                             LOAD MA
        DMSEXIT:   CONT                      & AM2901 ,MAPX,NOP,PASS,ZB
        /                                    & LDAER
0028E   /                                    & SPFNOP ;  LOAD AER WITH EXEC MAP
                   JZ                        & AM2901 ,PC,NOP,SUBR,ZB
        /                                    & CARRYH
        /                                    & LDMAR
0028F   /                                    & IFETCH ;        FETCH NEXT INSTR
        ;
```

```
ADDR    STATEMENT

        ;****************************************************
        ;*                                                  *
        ;*         SPMR                                     *
        ;*                                                  *
        ;*         STORE PAGE MAPPING REG ADDRESSED BY A REG *
        ;*         INTO B REG.  A REG IS INCREMENTED.       *
        ;*                                                  *
        ;****************************************************
        ;
        SPMRO:    CJP   GENMPV      & AM2901 ,,NOP,PASS,ZQ
        /                           & CONDEXT MPEN
        /                           & SPHNOP
00290   /                           & SPFNOP ;
                  CALL  POSIMAPR    & AM2901 ,A,RAMF,ADD,ZB
        /                           & CARRYH
        /                           & SPHNOP
00291   /                           & SPFNOP ; POSITION MAP REG NUMBER
                  CONT              & AM2901 ,B,RAMF,PASS,DZ
        /                           & SPRD MAPRD
00292   /                           & SPHNOP ;     B := MAP REG NUMBER
                  JP    DMSEXIT     & AM2901 ,,NOP,PASS,ZB
        /                           & SPHNOP
00293   /                           & SPFNOP ;   WRAP UP DMS EXECUTION
        ;
        ;         POSIMAPR - POSITION MAP REG NUMBER
        ;
        ;     THIS ROUTINE POSITIONS THE MAP REGISTER NUMBER IN A REG
        ; SO THE ACTUAL MAP REG CAN BE READ OR WRITTEN.
        ;
        POSIMAPR: PUSH  H 004       & AM2901 A,R4,RAMF,SUBR,ZA
        /                           & CARRYH
        /                           & SPHNOP
00294   /                           & SPFNOP ;           R4 := A - 1
                  RFCT              & AM2901 ,R4,SRAMR,PASS,ZB
        /                           & SHIFT ROTATE
        /                           & SPHNOP
00295   /                           & SPFNOP ;  ROTATE R4 RIGHT 5 BITS
                  CONT              & AM2901 ,R4,SRAMR,PASS,ZB
        /                           & SHIFT ROTATE
        /                           & LDAER
00296   /                           & SPFNOP ;    PUT MAPSET # IN AER
                  RET               & AM2901 ,R4,NOP,PASS,ZB
        /                           & LDMAR
00297   /                           & LDMAPD ; MAR := PAGE #, READ
```

```
ADDR     STATEMENT

         ;
         ;*****************************************************
         ;*                                                   *
         ;*        LMAP                                       *
         ;*        DEF MAP_NUMBER                             *
         ;*        DEF MAP_IMAGE                              *
         ;*                                                   *
         ;*        LOADS THE 32 PAGE REGISTERS IN MAP         *
         ;*        MAP_NUMBER FROM CONSECUTIVE MEMORY         *
         ;*        LOCATIONS STARTING AT MAP_IMAGE            *
         ;*                                                   *
         ;*                                                   *
         ;* Note:  Map reg value is masked with H 7FFF        *
         ;*        to turn off read protect.                  *
         ;*                                                   *
         ;*        All reads and writes are done in the       *
         ;*        EXEC map and are A/B addressable           *
         ;*                                                   *
         ;*****************************************************
         ;
         LMAP0:    CJP   GENMPV     & AM2901 ,,NOP,PASS,ZQ
         /                          & CONDEXT MPEN
         /                          & SPHNOP
00298    /                          & SPFNOP ;  IF MEM PROTECT ENABLED
                   CALL  BITSB      & AM2901 ,,NOP,PASS,ZQ
         /                          & SPHNOP
00299    /                          & MREAD ;         RESOLVE OPERANDS
                   CONT             & AM2901 ,R5,RAMF,ADD,ZQ
         /                          & CARRYH
         /                          & SPHNOP
0029A    /                          & SPFNOP ;R5 := MAP_IMAGE ADDR + 1
                   CONT             & AM2901 ,R6,RAMF,PASS,DZ
         /                          & IMM H 7FFF
0029B    /                          & SPHNOP ;      MASK FOR READ PROT
         ;
         ;         LOOP TO TRANSFER MAP
         ;
         ;  - R8 IS PAGE NUMBER
         ;
         ;   NOTE:  WRTIND STARTED A READ OF MAP_IMAGE
         ;
                   PUSH  H 01F      & AM2901 ,R8,RAMF,AND,ZB
         /                          & SPHNOP
0029C    /                          & SPFNOP ;        ZERO PAGE NUMBER
                   CONT             & AM2901 R4,R4,RAMA,PASS,ZB
         /                          & LDAER
0029D    /                          & SPFNOP ; LOAD AER WITH MAP_NUMBE
                   CONT             & AM2901 R8,R8,RAMA,ADD,DA
         /                          & CARRYL
```

```
ADDR     STATEMENT

         /                           & IMM H 0400
0029E    /                           & LDMAR ;  LOAD MAR WITH PAGE_NUMB
                   CONT              & AM2901 R6,,NOP,AND,DA
         /                           & SPWR MAPWR
0029F    /                           & SPFNOP ; MASK READ PROT DURING
                                                    MAP WRIT
                   CONT              & AM2901 MAPX,MAPX,RAMA,PASS,ZB
         /                           & LDAER
002A0    /                           & SPFNOP ; LOAD AER WITH EXECUTE M
                   RFCT              & AM2901 R5,R5,RAMA,ADD,ZB
         /                           & CARRYH
         /                           & LDMAR
002A1    /                           & MREAD ;  READ NEXT MAP_IMAGE LOC
                   JZ                & AM2901 PC,PC,RAMA,ADD,ZB
         /                           & CARRYH
         /                           & LDMAR
002A2    /                           & IFETCH ;  FETCH NEXT INSTRUCTION
         ;*****************************************************
         ;*                                                   *
         ;*        SMAP                                       *
         ;*        DEF MAP_NUMBER                             *
         ;*        DEF MAP_IMAGE                              *
         ;*                                                   *
         ;*        STORES THE 32 PAGE REGISTERS IN MAP        *
         ;*        MAP_NUMBER INTO CONSECUTIVE MEMORY         *
         ;*        LOCATIONS STARTING AT MAP_IMAGE            *
         ;*                                                   *
         ;*        ALL READS AND WRITES ARE DONE IN THE       *
         ;*        EXECUTE MAP, AND ARE A/B ADDRESSABLE       *
         ;*****************************************************
         ;
         SMAP0:    CJP   GENMPV      & AM2901 ,,NOP,PASS,ZQ
         /                           & CONDEXT MPEN
         /                           & SPHNOP
002A3    /                           & SPFNOP ;  IF MEM PROTECT ENABLED
                   CALL  BITSB       & AM2901 ,,NOP,PASS,ZQ
         /                           & SPHNOP
002A4    /                           & MREAD ;         RESOLVE OPERANDS
         ;
         ;         PLACE MAP_IMAGE ADDRESS IN R5
         ;
                   CONT              & AM2901 ,R5,RAMF,PASS,ZQ
         /                           & SPHNOP
002A5    /                           & SPFNOP ;  ADDR RETURNED IN Q REG
         ;
         ;         LOOP TO TRANSFER MAP
         ;
         ;   - R8 IS PAGE NUMBER REG
         ;
```

```
LINE   ADDR    STATEMENT

                         PUSH  H 01F      & AM2901 ,R8,RAMF,AND,ZB
               /                          & SPHNOP
       002A6   /                          & SPFNOP ;       ZERO PAGE NUMBER
                         CONT             & AM2901 R8,R8,RAMA,ADD,DA
               /                          & CARRYL
               /                          & IMM H 0400
       002A7   /                          & LDMAR ;  LOAD MAR WITH PAGE NUMB
                         CONT             & AM2901 R4,R4,RAMA,PASS,ZB
               /                          & LDAER
       002A8   /                          & LDMAPD ; LOAD AER AND DO MAP REA
                         CONT             & AM2901 ,,NOP,PASS,DZ
               /                          & SPRD MAPRD
       002A9   /                          & LDMDOR ; PASS MAP-DATA IN REG
                                                          TO MDO
                         CONT             & AM2901 R5,R5,RAMA,ADD,ZA
               /                          & CARRYH
               /                          & LDMAR
       002AA   /                          & SPFNOP ; LOAD MAR AND INC MAP_IM
                         RFCT             & AM2901 MAPX,MAPX,RAMA,PASS,ZB
               /                          & LDAER
       002AB   /                          & MWRITE ;      LOAD AER AND WRITE
               ;
               ;         FETCH NEXT INSTRUCTION
               ;
                         JZ               & AM2901 PC,PC,RAMA,ADD,ZB
               /                          & CARRYH
               /                          & LDMAR
       002AC   /                          & IFETCH;

               ;
               ;*****************************************************
               ;*                                                   *
               ;*        SWMP                                       *
               ;*        DEF   SAVEWMAP                             *
               ;*                                                   *
               ;*        STORES THE CURRENT WMAP INTO MEMORY.       *
               ;*                                                   *
               ;*****************************************************
               ;
               SWMP0:    CJP   GENMPV     & AM2901 ,,NOP,PASS,ZQ
               /                          & CONDEXT MPEN
               /                          & SPHNOP
       002AD   /                          & SPFNOP ;  IF MEM PROTECT ENABLED
                         CALL  WRTIND     & AM2901 ,,NOP,PASS,ZQ
               /                          & SPHNOP
       002AE   /                          & MREAD  ;   RESOLVE SAVEWMAP ADDR
                         CALL  STWMAP     & AM2901 ,,NOP,PASS,ZQ
               /                          & SPHNOP
```

```
ADDR    STATEMENT

002AF   /                               & SPFNOP ;      STORE WMAP INTO R4
                  JP     SKIP           & AM2901 R4,TAB,NOP,PASS,ZA
        /                               & LDMDOR
002B0   /                               & MWRITE ;      STORE R4 INTO MEM
        ;
        ;*****************************************************
        ;*                                                   *
        ;*        SIMP                                       *
        ;*        DEF   SAVEIMAP                             *
        ;*                                                   *
        ;*        STORES THE CONTENTS OF IMAP REG INTO       *
        ;*        MEMORY.  IMAP REG IS ACTUALLY IN BOOT      *
        ;*        MEMORY AT LOCATION "IMAPLOC".              *
        ;*                                                   *
        ;*****************************************************
        SIMPO:    CJP    GENMPV         & AM2901 ,,NOP,PASS,ZQ
        /                               & CONDEXT MPEN
        /                               & SPHNOP
002B1   /                               & SPFNOP ;  IF MEM PROTECT ENABLED
                  CALL   WRTIND         & AM2901 ,,NOP,PASS,ZQ
        /                               & SPHNOP
002B2   /                               & MREAD ;    RESOLVE SAVEIMAP ADDR
                  CONT                  & AM2901 ,,NOP,PASS,DZ
        /                               & IMM H 0020
002B3   /                               & LDAER ;       ENABLE BOOT MEMORY
                  CONT                  & AM2901 ,,NOP,PASS,DZ
        /                               & IMM IMAPLOC
002B4   /                               & LDMAR ;  MAR := ADDR OF IMAP REG
                  CONT                  & AM2901 ,,NOP,PASS,ZQ
        /                               & SPHNOP
002B5   /                               & MREAD ;               READ IMAP
                  CONT                  & AM2901 ,MAPX,NOP,PASS,ZB
        /                               & LDAER
002B6   /                               & SPFNOP ;  PUT MAPX BACK INTO AER
                  CONT                  & AM2901 ,,NOP,PASS,ZQ
        /                               & LDMAR
002B7   /                               & SPFNOP ;  PUT SAVEIMAP ADDR INTO
                  JP     SKIP           & AM2901 ,TAB,NOP,PASS,DZ
        /                               & LDMDOR
002B8   /                               & MWRITE ;     STORE IMAP INTO MEM
```

```
ADDR   STATEMENT

       ;
       ;*****************************************************
       ;*                                                   *
       ;*         XJMP                                      *
       ;*         DEF   NEWWMAP                             *
       ;*         DEF   NEXTINST                            *
       ;*                                                   *
       ;*         LOADS NEW WORKING MAP SET FROM NEWWMAP.   *
       ;*         SETS PC TO NEXTINST AND CONTINUES.        *
       ;*         SETS MEMORY PROTECT ON IF SIGN SET        *
       ;*                                                   *
       ;*****************************************************
       ;
       XJMPO:    CJP   GENMPV      & AM2901 ,,NOP,PASS,ZQ
       /                           & CONDEXT MPEN
       /                           & SPHNOP
002B9  /                           & SPFNOP ;     IF MEM PROT ENABLED
                 CALL  BITSB       & AM2901 ,,NOP,PASS,ZQ
       /                           & SPHNOP
002BA  /                           & MREAD  ;        RESOLVE OPERANDS
       ;
       ;         ON RETURN, R4 IS NEWWMAP AND R5 IS NEXTINST
       ;
       ;     - LOAD MAPD AND MAPX FROM THE NEW WMAP VALUE
       ;
                 CONT              & AM2901 R4,MAPX,RAMF,AND,DA
       /                           & IMM H 001F
002BB  /                           & LDAER ;           SET EXECUTE MAP
                 PUSH  H 004       & AM2901 ,R4,NOP,PASS,ZB
       /                           & LODMSR
       /                           & SPHNOP
002BC  /                           & SPFNOP ;   SET SIGN WITH MP ENBL
                 RFCT              & AM2901 ,R4,SRAMR,PASS,ZB
       /                           & SHIFT B 0000
       /                           & SPHNOP
002BD  /                           & SPFNOP ;           POSITION DATA1
                 CONT              & AM2901 R4,MAPD,RAMF,AND,DA
       /                           & IMM H 001F
002BE  /                           & SPHNOP ;    LOAD DATA1 INTO MAPD
       ;
       ;         TURN ON MEM PROTECT IF SIGN OF WMAP WAS SET
       ;
                 CCALL STC07       & AM2901 ,,NOP,PASS,ZQ
       /                           & CONDMSR SIGN
       /                           & SPHNOP
002BF  /                           & SPFNOP ;
                 JP    IOSKIP      & AM2901 ,PC,RAMF,PASS,ZQ
       /                  AE       & SPHNOP
002C0  /                           & SPFNOP ;        PC := NEXTINST
```

```
ADDR   STATEMENT

       ;************************************************************
       ;*
       ;*         DMS INSTRUCTION SUBROUTINES
       ;*
       ;************************************************************
       ;
       ;          GENMPV - Generate memory protect violation
       ;
       ; - This routine will generate a protect violation for a
       ;   priviledged instruction that was executed with Memory
       ;   Protect enabled.
       ; - The procedure is to simply assert the set Mem Prot signal
       ;   (and turn off memory protect due to bug in PAL).
       ;   No new fetch is executed.  The idea is to leave the PC
       ;   pointing to the word after the offending instruction.
       ;
       GENMPV:  CONT                & AM2901 ,,NOP,PASS,ZQ
       /                            & ENCN SETMPI
002C1  /                            & SPFNOP ;    GEN PENDING MEM PROT
                JZ                  & AM2901 ,,NOP,PASS,ZQ
       /                            & ENCN CLRMPEN
002C2  /                            & SPFNOP ;   TURN OFF MP (PAL BUG)
       ;
       ;        STWMAP - Format and store WMAP into R4
       ;
       STWMAP:  CONT                & AM2901 MAPX,R4,RAMF,AND,DA
       /                            & SPHNOP
002C3  /                            & IMM H 001F ; MASK EXEC MAP INTO
                                                        R4
                CONT                & AM2901 MAPD,R5,RAMA,PASS,DZ
       /                            & SPHNOP
002C4  /                            & L4D ;             POSITION DATA1
                CONT                & AM2901 ,R5,SRAML,PASS,ZB
       /                            & SHIFT B 0010
       /                            & SPHNOP
002C5  /                            & SPFNOP ;
                CONT                & AM2901 R5,R5,RAMF,AND,DA
       /                            & IMM H 03E0
002C6  /                            & SPHNOP ;          MASK DATA1 OFF
                CJP   STWMAP5       & AM2901 R5,R4,RAMF,OR,AB
       /                            & CONDEXT MPEN
       /                            & SPHNOP
002C7  /                            & SPFNOP ;          IF MEM PROT ON
                RET                 & AM2901 ,R4,NOP,PASS,ZB
       /                            & SPHNOP
002C8  /                            & SPFNOP ;    EXIT (DISPLAY WMAP)
       STWMAP5: RET                 & AM2901 R4,R4,RAMF,OR,DA
       /                            & IMM H 8000
002C9  /                            & SPHNOP ;        SET MEM PROT BIT
```

```
ADDR    STATEMENT

        ;
        ;***********************************************************
        ;*
        ;*      DMS Move Instruciton Group
        ;*
        ;*  Move Words:
        ;*
        ;*      MW00 - move words from EXECUTE to EXECUTE
        ;*      MW01 - move words from EXECUTE to DATA1
        ;*      MW10 - move words from DATA1 to EXECUTE
        ;*      MW11 - move words from DATA1 to DATA1
        ;*
        ;*  Move Bytes:
        ;*
        ;*      MB00 - move bytes from EXECUTE to EXECUTE
        ;*      MB01 - move bytes from EXECUTE to DATA1
        ;*      MB10 - move bytes from DATA1 to EXECUTE
        ;*      MB11 - move bytes from DATA1 to DATA1
        ;*
        ;*  Operation:
        ;*
        ;*      The A register contains the source address.  The
        ;*      B register contains the destination address. The
        ;*      X register contains the word count which can be
        ;*      zero.  When the instruction completes, A and B
        ;*      are incremented by the number of words moved and
        ;*      X is zero.  These instructions are interruptable.
        ;*
        ;***********************************************************
        ;

        ;
        ;       MW00/MB00 - EXEC TO EXEC
        ;
        MOV000: LDCT  MOVBYTE    & AM2901 MAPX,R7,RAMF,PASS,ZA
        /                        & SPHNOP
002CA   /                        & SPFNOP ;
                JRP   MOVWORD    & AM2901 MAPX,R8,RAMF,PASS,ZA
        /                        & CONDEXT IR11
        /                        & SPHNOP
002CB   /                        & SPFNOP ;
        ;
        ;       MW01/MB01 - EXEC TO DATA1
        ;
        MOV010: LDCT  MOVBYTE    & AM2901 MAPX,R7,RAMF,PASS,ZA
        /                        & SPHNOP
002CC   /                        & SPFNOP ;
                JRP   MOVWORD    & AM2901 MAPD,R8,RAMF,PASS,ZA
        /                        & CONDEXT IR11
```

```
ADDR    STATEMENT

        /                             & SPHNOP
002CD   /                             & SPFNOP ;
        ;
        ;          MW10/MB10 - DATA1 TO EXEC
        ;
        MOV100:    LDCT  MOVBYTE      & AM2901 MAPD,R7,RAMF,PASS,ZA
        /                             & SPHNOP
002CE   /                             & SPFNOP ;
                   JRP   MOVWORD      & AM2901 MAPX,R8,RAMF,PASS,ZA
        /                             & CONDEXT IR11
        /                             & SPHNOP
002CF   /                             & SPFNOP ;
        ;
        ;          MW11/MB11 - DATA1 TO DATA1
        ;
        MOV110:    LDCT  MOVBYTE      & AM2901 MAPD,R7,RAMF,PASS,ZA
        /                             & SPHNOP
002D0   /                             & SPFNOP ;
                   JRP   MOVWORD      & AM2901 MAPD,R8,RAMF,PASS,ZA
        /                             & CONDEXT IR11
        /                             & SPHNOP
002D1   /                             & SPFNOP ;
        ;

        ;
        ;          MOVWORD
        ;
        ;  This routine performs all the cross map move word instruc-
        ;  tions. At entry R7 is the from map and R8 is the to map.
        ;
        MOVWORD:   LDCT  LMOVWORD     & AM2901 ,X,NOP,PASS,ZB
        /                             & LODUSR
        /                             & SPHNOP
002D2   /                             & SPFNOP ;    ALLOW FOR ZERO COUNT
        LMOVWORD:  CJP   DMSEXIT      & AM2901 ,R7,NOP,PASS,ZB
        /                             & CONDUSR Z
        /                             & LDAER
002D3   /                             & SPFNOP ;         IF X == 0, EXIT
                   CONT               & AM2901 A,A,RAMA,ADD,ZB
        /                             & CARRYH
        /                             & LDMAR
002D4   /                             & MREAD ;              READ WORD
                   CONT               & AM2901 ,R8,NOP,PASS,ZB
        /                             & LDAER
002D5   /                             & SPFNOP ;
                   CONT               & AM2901 B,B,RAMA,ADD,ZB
        /                             & CARRYH
        /                             & LDMAR
002D6   /                             & SPFNOP ;           MAR := TO ADDR
```

```
ADDR    STATEMENT

                   CONT                & AM2901 ,,NOP,PASS,DZ
        /                              & LDMDOR
002D7   /                              & MWRITE ;
                   JRP   INTERRPT      & AM2901 ,X,RAMF,SUBR,ZB
        /                              & CARRYH
        /                              & CONDEXT INTRPT
        /                              & SPHNOP
002D8   /                              & SPFNOP ; REPEAT IF NOT INTERRUPT

        ;
        ;          MOVBYTE
        ;
        ;  This routine performs all the cross map move byte instruc-
        ;  tions. At entry, R7 is the from map and R8 is the to map.
        ;
        MOVBYTE:   LDCT  LMOVBYTE      & AM2901 ,X,NOP,PASS,ZB
        /                              & LODUSR
        /                              & SPHNOP
002D9   /                              & SPFNOP ;    ALLOW FOR ZERO COUNT
        LMOVBYTE:  CJP   DMSEXIT       & AM2901 ,R7,NOP,PASS,ZB
        /                              & CONDUSR Z
        /                              & LDAER
002DA   /                              & SPFNOP ;             SET FROM MAP
                   CONT                & AM2901 A,R4,SRAMR,PASS,ZA
        /                              & SHIFT ROTATE
        /                              & SPHNOP
002DB   /                              & SPFNOP ;  MAKE R4 A WORD ADDRESS
                   CALL  LDBYTE        & AM2901 R4,R4,RAMA,PASS,ZB
        /                              & LODMSR
        /                              & LDMAR
002DC   /                              & MREAD ;  READ WORD CONTAINING BY
                   CONT                & AM2901 ,R8,NOP,PASS,ZB
        /                              & LDAER
002DD   /                              & SPFNOP ;             SET TO MAP
                   CALL  STBYTE        & AM2901 ,A,RAMF,ADD,ZB
        /                              & CARRYH
        /                              & SPHNOP
002DE   /                              & SPFNOP ;  INC A (STBYTE INC'S B)
                   JRP   INTERRPT      & AM2901 ,X,RAMF,SUBR,ZB
        /                              & CARRYH
        /                              & CONDEXT INTRPT
        /                              & SPHNOP
002DF   /                              & SPFNOP ; REPEAT IF NOT INTERRUPT
```

```
LINE   ADDR    STATEMENT

               ;
               ;*************************************************************
               ;*                                                           *
               ;*          .ENTR - RTE Parameter Passing Routine            *
               ;*                                                           *
               ;*  - Calling Sequence:                                      *
               ;*                                 PARMS  EQU   *    <----+ *
               ;*      JSB   SUB                         BSS   N          | *
               ;*  +--> DEF   *+M+1  ------+        SUB   NOP   ------+    | *
               ;*  |    DEF   P[1]         |               JSB   .ENTR |    | *
               ;*  |    ...                |    MAR --->   DEF   PARMS | --+ *
               ;*  |    DEF   P[M]         |    PC ---->               |      *
               ;*  |    <RETURN ADDR>  <---+                           |      *
               ;*  +------------------------------------------------+       *
               ;*                                                           *
               ;*  - Operation:                                             *
               ;*                                                           *
               ;*    - Each DEF is resolved to a true address and moved to *
               ;*      the parameter block before the subroutine.           *
               ;*                                                           *
               ;*    - The actual return address is stored in SUB entry    *
               ;*                                                           *
               ;*    - If M >= N then N parameters are passed.             *
               ;*                                                           *
               ;*    - If M < N then M parameters are passed.              *
               ;*                                                           *
               ;*    - M or N can be zero.                                  *
               ;*                                                           *
               ;*************************************************************
               ;
               .ENTR:    CONT                & AM2901 ,R4,RAMF,PASS,DZ
               /                             & SPHNOP
       002E0   /                             & IMM H FFFD ; CONST TO LOCATE
                                                       NOP ENT
                         CALL  .ENTRSUB      & AM2901 ,,NOP,PASS,ZQ
               /                             & SPHNOP
       002E1   /                             & MREAD ;    RESOLVE PARMS ADDRESS
               .ENTR05:  JP    .ENTRL        & AM2901 R7,R8,RAMF,SUBR,AB
               /                             & CARRYH
               /                             & LODMSR
               /                             & SPHNOP
       002E2   /                             & SPFNOP ; R8 == M := RET ADDR -
                                                      JSB RET AD
               ;
               ;         .ENTP
               ;
               ;  - SAME AS .ENTR EXCEPT TWO WORDS BETWEEN NOP AND JSB .ENTP
               ;
```

```
ADDR    STATEMENT

        .ENTP:    CONT                  & AM2901 ,R4,RAMF,PASS,DZ
        /                               & SPHNOP
002E3   /                               & IMM H FFFB ; CONST TO LOCATE
                                                           NOP ENT
                  CALL   .ENTRSUB       & AM2901 ,,NOP,PASS,ZQ
        /                               & SPHNOP
002E4   /                               & MREAD ;    RESOLVE PARMS ADDRESS
                  JP     .ENTR05        & AM2901 R8,A,RAMF,PASS,ZA
        /                               & SPHNOP
002E5   /                               & SPFNOP ;     A := RETURN ADDRESS
        ;
        ;         .ENTN
        ;
        ;  - SAME AS .ENTR EXCEPT NO DEF TO RETURN ADDR AFTER
        ;    CALLERS JSB
        ;
        .ENTN:    CONT                  & AM2901 ,R4,RAMF,PASS,DZ
        /                               & SPHNOP
002E6   /                               & IMM H FFFD ; CONSTANT TO LOCATE
                                                          ENTR
                  CALL   .ENTRSUB       & AM2901 ,,NOP,PASS,ZQ
        /                               & SPHNOP
002E7   /                               & MREAD ;    RESOLVE PARMS ADDRESS
        .ENTN05:  CONT                  & AM2901 R6,R8,RAMF,PASS,ZA
        /                               & LDMSR
        /                               & SPHNOP
002E8   /                               & SPFNOP ;             SET M := N
                  JP     .ENTRL1        & AM2901 R7,R6,NOP,ADD,AB
        /                               & CARRYL
        /                               & LDMDOR
002E9   /                               & MWRITE ; RET ADDR := WORD AFTER
                                                      JSB + N

        ;
        ;         .ENTC
        ;
        ;  - SAME AS .ENTP EXCEPT NO DEF TO RETURN ADDR AFTER
        ;    CALLERS JSB
        ;
        .ENTC:    CONT                  & AM2901 ,R4,RAMF,PASS,DZ
        /                               & SPHNOP
002EA   /                               & IMM H FFFB ; CONSTANT TO LOCATE
                                                           ENTR
                  CALL   .ENTRSUB       & AM2901 ,,NOP,PASS,ZQ
        /                               & SPHNOP
002EB   /                               & MREAD ;    RESOLVE PARMS ADDRESS
                  CONT                  & AM2901 R7,A,RAMF,PASS,ZA
        /                               & SPHNOP
002EC   /                               & SPFNOP ;
                  JP     .ENTN05        & AM2901 R6,A,RAMF,ADD,AB
```

```
ADDR    STATEMENT

        /                          & CARRYL
        /                          & SPHNOP
002ED   /                          & SPFNOP ;     A := RETURN ADDRESS
        ;
        ;         .ENTX PARM MOVE LOOP
        ;
        .ENTRL:   CONT             & AM2901 ,R7,RAMF,ADD,ZB
        /                          & CARRYH
        /                          & SPHNOP
002EE   /                          & SPFNOP ;    SKIP DEF TO RET ADDR
        .ENTRL1:  CJP   SKIP       & AM2901 R7,R7,RAMA,ADD,ZB
        /                          & CARRYH
        /                          & CONDMSR Z
        /                          & LDMAR
002EF   /                          & MREAD ;      IF END OF P[N] LIST
                  CALL  WRTIND     & AM2901 ,R6,RAMF,SUBR,ZB
        /                          & CARRYH
        /                          & LODMSR
        /                          & SPHNOP
002F0   /                          & SPFNOP ;            N := N - 1;
                  CJP   SKIP       & AM2901 R5,R5,RAMA,ADD,ZB
        /                          & CARRYH
        /                          & CONDMSR SIGN
        /                          & LDMAR
002F1   /                          & SPFNOP ;    IF END OF PARMS LIST
                  CONT             & AM2901 ,,NOP,PASS,ZQ
        /                          & LDMDOR
002F2   /                          & MWRITE ; PARMS[Y] := RESOLVED AD
                  JP    .ENTRL1    & AM2901 ,R8,RAMF,SUBR,ZB
        /                          & CARRYH
        /                          & LODMSR
        /                          & SPHNOP
002F3   /                          & SPFNOP ;            M := M - 1
        ;
        ;         .ENTR/.ENTP SETUP SUBROUTINE
        ;
        .ENTRSUB: CONT             & AM2901 PC,R4,RAMF,ADD,AB
        /                          & CARRYL
        /                          & LDMAR
002F4   /                          & SPFNOP ;   R4 := SUB ENTRY POINT
                  CONT             & AM2901 TAB,R5,RAMF,PASS,DZ
        /                          & SPHNOP
002F5   /                          & MREAD ;  R5 := ADDR OF PARMS BLO
                  CONT             & AM2901 R5,R6,RAMF,PASS,ZA
        /                          & SPHNOP
002F6   /                          & SPFNOP ;            R6 WILL BE N
                  CONT             & AM2901 TAB,R7,RAMF,PASS,DZ
        /                          & LDMAR
```

```
ADDR    STATEMENT

002F7   /                                & MREAD ; R7 := ADDR OF WRD AFTER
                                                        JSB
                  CONT                   & AM2901 R4,R6,RAMA,SUBS,AB
        /                                & CARRYH
        /                                & LDMAR
002F8   /                                & SPFNOP ; R6 == N := ADDR(SUB) -
                                                        ADDR(PARM
                  RET                    & AM2901 TAB,R8,RAMF,PASS,DZ
        /                                & LDMDOR
002F9   /                                & MWRITE ; STORE ACTUAL RETURN ADD
        ;
        ;         SAVEAB - COPIES A AND B TO R6 AND R7
        ;
        SAVEAB:   CONT                   & AM2901 A,R6,RAMF,PASS,ZA
        /                                & SPHNOP
002FA   /                                & SPFNOP ;
                  RET                    & AM2901 B,R7,RAMF,PASS,ZA
        /                                & SPHNOP
002FB   /                                & SPFNOP ;
        ;
        ;         FILL AREA
        ;
        IF        FILL
        LIST
        ENDIF

        ;
        ;         PATCH AREA
        ;
00300             ORG    768
00300   ;
        LIST
        ;
```

LINE ADDR STATEMENT

```
;
;**********************************************************************
;*                                                                    *
;*        VMA Microcode Routines                                      *
;*                                                                    *
;* - VMA pointer:                                                     *
;*                                                                    *
;*                                                                    *
;* |       A  Register            |       B  Register            |   *
;* +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+   *
;* |L|0 0 0 0 0| VMA Seg # |   PTE  Index      | Log Page Offset  |   *
;* +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+   *
;*                                                                    *
;* - L is local reference bit.  If L is set, then B register is       *
;*   normal address.  Simply resolve B for indirects and return       *
;*   address in B.                                                    *
;*                                                                    *
;* - The PTE index is an index into a table of VMA pages.  The page   *
;*   number of this table is contained in location 5.  Access to the  *
;*   PTE table is by reading loc 5, loading contents into reg 31 of   *
;*   MAPX, then using 011111B as the upper bits of an address and     *
;*   the PTE Index as the lower bits.                                 *
;*                                                                    *
;* - The PTE table entries can have three forms:                      *
;*                                                                    *
;*              15        10 9                0     Circumstance      *
;*              +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                     *
;*    Case A    | VMA suit  |   Page offset     |   Normal            *
;*              +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                     *
;*                                                                    *
;*              +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                     *
;*    Case B    | VMA suit  |0 0 0 0 0 0 0 0 0 0|   Last+1 VMA page   *
;*              +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                     *
;*                                                                    *
;*              +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                     *
;*    Case C    |1 1 1 1 1 1|0 0 0 0 0 0 0 0 0 0|   Fault             *
;*              +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                     *
;*                                                                    *
;*    When a PTE entry is case A, normal VMA operation proceeds.      *
;*    The PTE case B entry is indicated by a Page offset              *
;*    field of all zeros and a VMA suit of not all ones.              *
;*    In this case, the page is the last+1 VMA page which can         *
;*    be mapped, but not accessed.  The PTE case C entry is           *
;*    indicated by a Page offset of all zeros and a VMA suit          *
;*    of all ones.  In this case, the VMA page is not in memory       *
;*    and a page fault error is generated.                            *
;*                                                                    *
;**********************************************************************
```

```
ADDR    STATEMENT

        ;
        ;*************************************************************
        ;*                                                           *
        ;*       .LBPR - Load and map VMA pointer                    *
        ;*                                                           *
        ;* - Calling Sequence:                                       *
        ;*                                                           *
        ;*       JSB   .LBPR                                         *
        ;*       DEF   POINTER                                       *
        ;*                                                           *
        ;* - Resolves pointer, double loads A & B and performs .LBP *
        ;*                                                           *
        ;*************************************************************
        ;
        .LBPR:   CALL  SAVEAB      & AM2901 ,,NOP,PASS,ZQ
        /                          & SPHNOP
00300   /                          & SPFNOP ;      COPY A-B TO R6-R7
                 CALL  DLOAD       & AM2901 PC,R10,RAMF,PASS,ZA
        /                          & SPHNOP
00301   /                          & MREAD  ;      SAVE PC FOR FAULT
                 JP    VMAMAP      & AM2901 ,,NOP,PASS,ZQ
        /                          & SPHNOP
00302   /                          & SPFNOP ;
        ;
        ;*************************************************************
        ;*                                                           *
        ;*       .LPX - Add offset to VMA pointer and map            *
        ;*                                                           *
        ;* - Calling Sequence:                                       *
        ;*                                                           *
        ;*       DLD   VMA_POINTER                                   *
        ;*       JSB   .LPX                                          *
        ;*       DEF   OFFSET                                        *
        ;*                                                           *
        ;* - Resolve offset, double integer add to A&B, perform .LBP*
        ;*                                                           *
        ;*************************************************************
        ;
        .LPX0:   CALL  SAVEAB      & AM2901 PC,R10,RAMF,PASS,ZA
        /                          & SPHNOP
00303   /                          & SPFNOP ;      COPY A/B TO R6/R7
        .LPX01:  CALL  WRTIND      & AM2901 ,,NOP,PASS,ZQ
        /                          & SPHNOP
00304   /                          & MREAD ;           RESOLVE OFFSET
                 CALL  DADD        & AM2901 ,,NOP,PASS,ZQ
        /                          & SPHNOP
00305   /                          & SPFNOP ;    READ AND ADD OFFSET
                 JP    VMAMAP      & AM2901 ,PC,RAMF,ADD,ZB
        /                          & CARRYH
```

```
ADDR   STATEMENT

       /                              & SPHNOP
00306  /                              & SPFNOP ;        INC PC PAST DEF
       ;
       ;*************************************************************
       ;*                                                           *
       ;*        .LPXR - Load VMA pointer, add offset and map       *
       ;*                                                           *
       ;* - Calling Sequence:                                       *
       ;*                                                           *
       ;*        JSB   .LPXR                                        *
       ;*        DEF   POINTER                                      *
       ;*        DEF   OFFSET                                       *
       ;*                                                           *
       ;* - Resolve pointer, double load A & B, resolve offset,     *
       ;*   and double add offset.                                  *
       ;*                                                           *
       ;* - Note:  OFFSET may NOT address A or B                    *
       ;*                                                           *
       ;*************************************************************
       ;
       .LPXR:   CALL  SAVEAB          & AM2901 ,,NOP,PASS,ZQ
       /                              & SPHNOP
00307  /                              & SPFNOP ;       COPY A-B TO R6-R7
                CALL  DLOAD           & AM2901 PC,R10,RAMF,PASS,ZA
       /                              & SPHNOP
00308  /                              & MREAD  ;       SAVE PC FOR FAULT
                JP    .LPX01          & AM2901 ,,NOP,PASS,ZQ
       /                              & SPHNOP
00309  /                              & SPFNOP ;     JUST LIKE .LPX NOW
       ;
       ;*************************************************************
       ;*                                                           *
       ;*        .LBP - Map VMA pointer in A/B to logical address   *
       ;*                                                           *
       ;* -At entry, A and B contain the VMA pointer.               *
       ;*                                                           *
       ;*************************************************************
       ;
       .LBP0:   CALL  SAVEAB          & AM2901 PC,R10,RAMF,PASS,ZA
       /                              & SPHNOP
0030A  /                              & SPFNOP ;       COPY A/B TO R6/R7
```

```
LINE    ADDR    STATEMENT

                ;
                ;************************************************************
                ;*                                                          *
                ;*         VMAMAP - Perform VMA mapping function            *
                ;*                                                          *
                ;* - If A(15) is set, then this is a local reference. The   *
                ;*   B reg is resolved and microcode exits.                 *
                ;*                                                          *
                ;************************************************************
                ;
                VMAMAP:   CONT                & AM2901 ,A,NOP,PASS,ZB
                /                             & LODUSR
                /                             & SPHNOP
        0030B   /                             & SPFNOP ;      TEST FOR LOCAL REF
                          CJP   VMAMAP01      & AM2901 ,,NOP,PASS,ZQ
                /                             & CONDUSR NSIGN
                /                             & SPHNOP
        0030C   /                             & SPFNOP ;  IF NOT LOCAL REFERENCE
                          CALL  MRGIND        & AM2901 ,B,QREG,PASS,ZB
                /                             & LODUSR
                /                             & LDMAR
        0030D   /                             & MREAD ;
                          JZ                  & AM2901 ,B,RAMF,PASS,ZQ
                /                             & SPHNOP
        0030E   /                             & IFETCH ;
                VMAMAP01: CONT                & AM2901 ,Y,RAMF,PASS,DZ
                /                             & IMM H 7800
        0030F   /                             & SPHNOP ; Y := PAGE 30 & LOGICAL
                                                                ADDR OF
                          CALL  GETPTE        & AM2901 ,B,QREG,PASS,ZB
                /                             & RSTMSR & ENBLO
                /                             & SPHNOP
        00310   /                             & SPFNOP ;         GET PAGE OF PTE
                ;
                ;         BUILD PAGEID IN X
                ;
                          PUSH  H 005         & AM2901 A,X,RAMF,PASS,ZA
                /                             & SPHNOP
        00311   /                             & SPFNOP ;  SHIFT VMA POINTER LEFT
                          RFCT                & AM2901 ,X,SRAMQL,PASS,ZB
                /                             & SHIFT B 0110
                /                             & SPHNOP
        00312   /                             & SPFNOP ;            X := PAGE ID
                          CALL  PTELKUP       & AM2901 ,,NOP,PASS,ZQ
                /                             & SPHNOP
        00313   /                             & SPFNOP ;     LOOKUP PAGE IN PTE
                          CONT                & AM2901 ,Y,RAMF,PASS,DZ
                /                             & IMM H 7C00
        00314   /                             & LDMAR  ;            Y := PAGE 31
```

```
LINE   ADDR    STATEMENT

                         CALL  PTELKUP      & AM2901 ,X,RAMF,ADD,ZB
                /                           & CARRYH
                /                           & SPHNOP
       00315    /                           & SPFNOP ;           INC PAGID
                         CONT               & AM2901 B,B,RAMF,NOTRS,DA
                /                           & IMM H FC00
       00316    /                           & SPHNOP ; MASK OFF LOGICAL OFFSET
                                                       IN PAGE
                         CONT               & AM2901 B,B,RAMF,OR,DA
                /                           & SPHNOP
       00317    /                           & IMM H 7800 ;RETURN VMA ADDR IN B
                         JZ                 & AM2901 ,PC,NOP,SUBR,ZB
                /                           & CARRYH
                /                           & LDMAR
       00318    /                           & IFETCH ;

                ;
                ;*************************************************************
                ;*                                                           *
                ;*       .PMAP - PAGE MAP                                    *
                ;*                                                           *
                ;*  - Calling Sequence:                                      *
                ;*                                                           *
                ;*       LDA   REG# IN MAPX                                  *
                ;*       LDB   PAGE_ID                                       *
                ;*       JSB   .PMAP                                         *
                ;*       <ERROR RETURN>                                      *
                ;*       <NORMAL RETURN>                                     *
                ;*                                                           *
                ;*  - Used to load arbitrary map reg in MAPX for VMA         *
                ;*                                                           *
                ;* - A reg contains page reg in MAPX to load. B reg contains*
                ;*   same VMA page_id as bits 25-10 of VMA pointer.          *
                ;*                                                           *
                ;*  - Perform VMA lookup as in .LBP.  If fault and A15 == 1 *
                ;*    then take <error return>.  If successful lookup, then *
                ;*    load page reg in MAPX, inc A & B & take <normal ret>. *
                ;*                                                           *
                ;*  - O reg is set to 1                                      *
                ;*                                                           *
                ;*  - E reg set if to 1 if last+1 page mapped                *
                ;*                                                           *
                ;*************************************************************
                ;
```

```
LINE  ADDR   STATEMENT

             .PMAP0:   CONT               & AM2901 ,Y,RAMF,PASS,DZ
             /                            & IMM H 7C00
      00319  /                            & SPHNOP ;    USE PAGE 31 FOR PTE
                       CALL  GETPTE       & AM2901 PC,R10,RAMF,PASS,ZA
             /                            & SETMSR & ENBLO
             /                            & SPHNOP
      0031A  /                            & SPFNOP ;           SET O REG
                       CALL  PTELKUP      & AM2901 B,X,RAMF,PASS,ZA
             /                            & SPHNOP
      0031B  /                            & SPFNOP ;
             ;
             ;         POSITION MAP REGISTER NUMBER
             ;
                       PUSH  H 005        & AM2901 A,R7,RAMF,PASS,ZA
             /                            & SPHNOP
      0031C  /                            & SPFNOP ; ROTATE PAGE NUM RIGHT 6
                       RFCT               & AM2901 ,R7,SRAMR,PASS,ZB
             /                            & SHIFT ROTATE
             /                            & SPHNOP
      0031D  /                            & SPFNOP ;
                       CALL  PTELKUPX     & AM2901 ,R7,NOP,PASS,ZB
             /                            & LDMAR
      0031E  /                            & SPFNOP ;PAG NUM TO MAR, MAP WRIT
                       CONT               & AM2901 ,B,RAMF,ADD,ZB
             /                            & CARRYH
             /                            & SPHNOP
      0031F  /                            & SPFNOP ;             INC B
                       JP    SKIP         & AM2901 ,A,RAMF,ADD,ZB
             /                            & CARRYH
             /                            & SPHNOP
      00320  /                            & SPFNOP ;             INC A

             ;
             ;*************************************************************
             ;*                                                           *
             ;*         .IRES - Perform single int subscript calculation  *
             ;*                                                           *
             ;*************************************************************
             ;
             .IRES:    CALL  IRES01       & AM2901 PC,R10,RAMF,PASS,ZA
             /                            & SPHNOP
      00321  /                            & SPFNOP ;             PERFORM CALC
                       JZ                 & AM2901 R8,PC,RAMA,ADD,ZA
             /                            & CARRYH
             /                            & LDMAR
      00322  /                            & IFETCH ;
```

```
LINE   ADDR    STATEMENT

         ;
         ;*************************************************************
         ;*                                                           *
         ;*     .IMAP - Perform subscript calculation and map result *
         ;*                                                           *
         ;*************************************************************
         ;
         .IMAP:    CALL  SAVEAB     & AM2901 ,,NOP,PASS,ZQ
         /                          & SPHNOP
00323    /                          & SPFNOP ;       SAVE ORIGINAL A/B
                   CALL  IRES01     & AM2901 PC,R10,RAMF,PASS,ZA
         /                          & SPHNOP
00324    /                          & SPFNOP ;             PERFORM CALC
                   JP    VMAMAP     & AM2901 R8,PC,RAMF,ADD,ZA
         /                          & CARRYH
         /                          & SPHNOP
00325    /                          & SPFNOP ; INC PC, MAP VMA POINTER

         ;
         ;*************************************************************
         ;*                                                           *
         ;*        VMAFAULT                                           *
         ;*                                                           *
         ;* - VMA page fault handler.  This routine is common         *
         ;*   to .LBP, .IMAP and .PMAP.  Note that .PMAP must         *
         ;*   take a error return if A(15) is set.                    *
         ;*                                                           *
         ;* - X reg is set to the PAGID of the page not in the PTE.   *
         ;*                                                           *
         ;* - Y reg is set to the logical address of the PTE.         *
         ;*                                                           *
         ;* - Perform a JSB indirect thru location $VMA$ (VMALOC).    *
         ;*                                                           *
         ;*************************************************************
         ;
         ;         RESTORE PC TO VALUE AT ENTRY TO VMA
         ;
         ;  - TEST FOR PMAP SPECIAL HANDLING
         ;
         VMAFAULT: CJP   VMAFPMAP   & AM2901 R10,PC,RAMF,PASS,ZA
         /                          & CONDMSR OVR
         /                          & SPHNOP
00326    /                          & SPFNOP ; RESTORE PC TO ENTRY VAL
                   CONT             & AM2901 R6,A,RAMF,PASS,ZA
         /                          & SPHNOP
00327    /                          & SPFNOP ;          RESTORE A AND B
                   CONT             & AM2901 R7,B,RAMF,PASS,ZA
         /                          & SPHNOP
00328    /                          & SPFNOP ;
```

```
ADDR    STATEMENT

        VMAFAUL1: CONT              & AM2901 ,,NOP,PASS,DZ
        /                           & IMM VMALOC
00329   /                           & LDMAR ;
                  CONT              & AM2901 ,PC,RAMF,SUBR,ZB
        /                           & CARRYH
        /                           & SPHNOP
0032A   /                           & MREAD  ;  READ CONTENTS OF $VMA$
                  JP    JSBIVMA     & AM2901 ,,QREG,PASS,DZ
        /                           & LDMAR
0032B   /                           & MREAD  ;          MAR := ($VMA$)
0032C             FILLER
0032D             FILLER
0032E             FILLER

        ;
        ;*************************************************************
        ;*                                                           *
        ;*       GETPTE - GET PTE ADDRESS                            *
        ;*                                                           *
        ;*  - At entry:   Y - Logical address of PTE                 *
        ;*                                                           *
        ;*  -Get PAGE TABLE page number from VMAPTE in map 0         *
        ;*   and load it into map reg number indicated by Y          *
        ;*                                                           *
        ;*  -If the sign bit is set on the word read from            *
        ;*   memory, then VMA has not been initialized               *
        ;*   and a page fault is generated.                          *
        ;*                                                           *
        ;*************************************************************
        ;
        GETPTE:   CONT              & AM2901 ,,NOP,PASS,DZ
        /                           & IMM VMAPTE
0032F   /                           & LDMAR ;READ PTE PG FROM MAP ZERO
                  CONT              & AM2901 ,,NOP,AND,ZQ
        /                           & LDAER
00330   /                           & MREAD ;
                  CONT              & AM2901 MAPX,R5,RAMA,PASS,DZ
        /                           & LDAER
00331   /                           & IMM H 8000 ;       PUT MAPX BACK
                  CONT              & AM2901 Y,R12,RAMA,PASS,DZ
        /                           & LODUSR
        /                           & LDMAR
00332   /                           & SPFNOP ; R12 := PTE PAGE, MAR :=
                                                   PTE AD
                  CRET              & AM2901 R5,R12,RAMF,NOTRS,AB
        /                           & CONDUSR NSIGN
        /                           & SPWR MAPWR
```

```
ADDR    STATEMENT

00333   /                            & SPFNOP ; IF SIGN SET, VMA NOT
                                                 INITIALI
                  JP    VMAFAULT     & AM2901 R5,Y,RAMF,OR,AB
        /                            & SPHNOP
00334   /                            & SPFNOP ;          SET SIGN IN Y
        ;
        ;*************************************************************
        ;*                                                           *
        ;*        PTELKUP - Lookup page in PTE                       *
        ;*                                                           *
        ;* - At entry:                                               *
        ;*                                                           *
        ;*      X   - Page ID                                        *
        ;*      R12 - PTE page number                                *
        ;*  Y & MAR - Logical address of PTE (& map reg to update)   *
        ;*                                                           *
        ;* - At exit:                                                *
        ;*                                                           *
        ;*        R8 - VMA Page loaded into map register             *
        ;*        E  - Set if last+1 page, Clear otherwise           *
        ;*        Map register updated                               *
        ;*                                                           *
        ;*************************************************************
        ;
        PTELKUP:  CONT               & AM2901 ,R11,RAMF,PASS,DZ
        /                            & SPHNOP
00335   /                            & IMM H FC00 ;        CONSTANT
                  CONT               & AM2901 ,R12,NOP,PASS,ZB
        /                            & RSTMSR & ENBLC
        /                            & SPWR MAPWR
00336   /                    & SPFNOP ; CLEAR E, SET PAGE TO ACCESS PT
                  CONT               & AM2901 X,R5,RAMF,NOTRS,DA
        /                            & SPHNOP
00337   /                            & IMM H FC00 ;  ISOLATE PTE OFFSET
                  CONT               & AM2901 Y,R5,NOP,OR,AB
        /                            & LDMAR
00338   /                            & MREAD  ;          INDEX INTO PTE
                  CONT               & AM2901 X,R5,RAMF,AND,DA
        /                            & SPHNOP
00339   /                            & IMM H FC00 ;ISOLATE VMA SEG NUMB
        ;
        ;         DETERMINE WHICH PTE ENTRY CASE IS THIS?
        ;
        ;   Algorithm is:
        ;        1. Test for 1111110000000000B.  If same, then fault.
        ;        2. Test for VMA suit match. If not same then fault.
        ;        3. Test for page offset from PTE of zero.
        ;           If zero, then last+1 page case and map 077777Q
        ;                    else perform normal mapping
```

```
ADDR      STATEMENT

          ;
                      CONT                 & AM2901 ,,QREG,PASS,DZ
          /                                & SPHNOP
0033A     /                                & SPFNOP ;       Q := (PTE ENTRY)
                      CONT                 & AM2901 R11,,NOP,EXOR,AQ
          /                                & LODUSR
          /                                & SPHNOP
0033B     /                                & SPFNOP ; TEST FOR
                                                      1111110000000000 ENTRY
          ;
          ;           MASK OFF VMA SUIT NUMBER
          ;
                      CJP   VMAFAULT       & AM2901 R11,R9,RAMF,AND,AQ
          /                                & CONDUSR Z
          /                                & SPHNOP
0033C     /                                & SPFNOP ;IF NOT THEIR ENTRY,FAULT
                      CONT                 & AM2901 R5,R9,NOP,EXOR,AB
          /                                & LODUSR
          /                                & SPHNOP
0033D     /                                & SPFNOP ;       COMPARE VMA SUIT
          ;
          ;           MASK OFF PTE PAGE OFFSET
          ;
                      CJP   VMAFAULT       & AM2901 R11,R8,RAMF,NOTRS,AQ
          /                                & CONDUSR NZ
          /                                & SPHNOP
0033E     /                                & SPFNOP ;  IF NOT SAME THEN FAULT
                      CJP   PTELKUP5       & AM2901 R12,R8,RAMF,ADD,AB
          /                                & CARRYL
          /                                & CONDUSR Z
          /                                & SPHNOP
0033F     /                                & SPFNOP ;       IF OFFSET IS ZERO
          PTELKUPX:   RET                  & AM2901 ,R8,NOP,PASS,ZB
          /                                & SPWR MAPWR
00340     /                                & SPFNOP ;  PUT PAGE INTO MAPX REG
          ;
          ;           MAP LAST+1 PAGE TO 077777B
          ;
          PTELKUP5:   CONT                 & AM2901 ,R8,RAMF,PASS,DZ
          /                                & SPHNOP
00341     /                                & IMM H 7FFF ;
                      RET                  & AM2901 ,R8,NOP,PASS,ZB
          /                                & SETMSR & ENBLC
          /                                & SPWR MAPWR
00342     /                                & SPFNOP ;
```

```
LINE    ADDR    STATEMENT

                ;
                ;**************************************************************
                ;*                                                            *
                ;*          .IRES01 - Calculate subscripted array address     *
                ;*                                                            *
                ;* -Calling Sequence:                                         *
                ;*                                                            *
                ;* JSB   .IRES/.IMAP                                          *
                ;* DEF   DOPE VECTOR ---------> DEC  N    # DIMENSIONS        *
                ;* DEF   An   SUBSCRIPT N       DEC  Dn-1 DIMENSION N-1       *
                ;* DEF   An-1 SUBSCRIPT N-1     DEC  Dn-2 DIMENSION N-2       *
                ;*  .                            .                            *
                ;*  .                            .                            *
                ;*  .                            .                            *
                ;* DEF   A2   SUBSCRIPT 2       DEC  D1   DIMENSION 1         *
                ;* DEF   A1   SUBSCRIPT 1       DEC  E   # WORDS PER ELEMENT*
                ;*                              OCT  UPPER HALF OF OFFSET     *
                ;*                              OCT  LOWER HALF OF OFFSET     *
                ;*                                                            *
                ;*  -Calculation for B(A1,A2,A3,A4) is:                       *
                ;*                                                            *
                ;*   offset(B) + E * {A1 + D1*[A2 + D2*[A3 + D3*[A4 + 0]]]} *
                ;*                                                            *
                ;*                                                            *
                ;*  -Notes:                                                   *
                ;*      - Subscripts are sign extended to 32 bits             *
                ;*      - If a dimension is zero, it is really 2**16          *
                ;*      - Calculation is accumulated in A and B               *
                ;*      - It is possible for # dimensions (N) to be zero      *
                ;*      - DEF'S can NOT be A/B addressable                    *
                ;*                                                            *
                ;**************************************************************
                ;
                IRES01:   CALL  WRTIND      & AM2901 PC,R8,RAMF,PASS,ZA
                /                           & SPHNOP
        00343   /                           & MREAD ;      RESOLVE DOPE VECTOR
                          CONT              & AM2901 ,R12,RAMF,ADD,ZQ
                /                           & CARRYH
                /                           & SPHNOP
        00344   /                           & SPFNOP ; R12 -> 1ST DIM IN DOPE
                                                           VECTOR
                          CONT              & AM2901 ,R11,RAMF,PASS,DZ
                /                           & LODMSR
                /                           & SPHNOP
        00345   /                           & SPFNOP ;      R11 := # DIMENSIONS
                          CONT              & AM2901 ,B,RAMF,AND,ZB
                /                           & SPHNOP
        00346   /                           & SPFNOP ;         ZERO ACCUMULATOR
                IRES04:   CONT              & AM2901 ,A,RAMF,AND,ZB
```

```
LINE    ADDR    STATEMENT

                /                           & SPHNOP
        00347   /                           & SPFNOP ;
                ;
                ;          ITERATION LOOP
                ;
                ;    - SIGN EXTEND SUBSCRIPT AND ADD TO ACCUMULATOR
                ;    - MULTIPLY BY DIMENSION
                ;
                IRES05:    CJP   IRES80     & AM2901 ,,NOP,PASS,ZQ
                /                           & CONDMSR Z
                /                           & SPHNOP
        00348   /                           & SPFNOP ;       IF LAST DIMENSION
                           CALL  WRTIND     & AM2901 R8,R8,RAMA,ADD,ZB
                /                           & CARRYH
                /                           & LDMAR
        00349   /                           & MREAD ;            READ SUBSCRIPT
                           CONT             & AM2901 ,,QREG,PASS,DZ
                /                           & LODMSR
                /                           & SPHNOP
        0034A   /                           & SPFNOP;            Q := SUBSCRIPT
                           CJP   IRES10     & AM2901 R12,R12,RAMA,ADD,ZB
                /                           & CARRYH
                /                           & CONDMSR NSIGN
                /                           & LDMAR
        0034B   /                           & MREAD ; READ DIMENSION, IF
                                                                 SUBSCRIPT >=
                ;
                ;          A NEGATIVE SUBSCRIPT MUST BE SIGN EXTENDED
                ;
                ;    - This implies a word of all 1'S (H FFFF) MUST be added
                ;      to the most significant word of the accumulator to
                ;      extend the sign of the subscript to 32 bits.
                ;
                           CONT             & AM2901 ,A,RAMF,SUBR,ZB
                /                           & CARRYH
                /                           & SPHNOP
        0034C   /                           & SPFNOP ;     MSB := MSB + H FFFF
                ;
                ;          ADD SUBSCRIPT TO ACCUMULATOR
                ;
                IRES10:    CONT             & AM2901 B,B,RAMF,ADD,AQ
                /                           & CARRYL
                /                           & LODUSR
                /                           & SPHNOP
        0034D   /                           & SPFNOP ;  LSB := LSB + SUBSCRIPT
                           CONT             & AM2901 ,A,RAMF,ADD,ZB
                /                           & CARRYUC
                /                           & SPHNOP
        0034E   /                           & SPFNOP ;    ADD CARRY OUT TO MSB
```

```
LINE   ADDR     STATEMENT

                          LDCT  H 00F       & AM2901 ,,QREG,PASS,DZ
                /                           & LODMSR
                /                           & SPHNOP
       0034F    /                           & SPFNOP ;          Q := DIMENSION
                ;
                ;         MULTIPLY ACCUMULATOR BY DIMENSION
                ;
                ;    - Unsigned multiply.
                ;
                ;    - Algorithm is:
                ;                 R5 & Q := B * DIMENSION
                ;                      B := Q
                ;                 R5 & Q := A * DIMENSION + R5
                ;
                          CJP   IRES50      & AM2901 B,R4,RAMF,PASS,ZA
                /                           & CONDMSR Z
                /                           & SPHNOP
       00350    /                           & SPFNOP ;       IF DIMENSION ZERO
                ;
                ;         THE FOLLOWING PUSH MUST NOT LOAD COUNTER
                ;
                          CPUSH             & AM2901 ,R5,SRAMQR,AND,ZB
                /                           & CONDMSR Z
                /                           & SHIFT B 0110
                /                           & SPHNOP
       00351    /                           & SPFNOP ;   R5 := 0, Q0 BUF := Q0
                          RFCT              & AM2901 MPY4,R5,SRAMQR,ADD,AB
                /                           & CARRYL
                /                           & SHIFT B 1011
                /                           & SPHNOP
       00352    /                           & SPFNOP ;       UNSIGNED MULTIPLY
                          CONT              & AM2901 ,B,RAMF,PASS,ZQ
                /                           & SPHNOP
       00353    /                           & SPFNOP ;           LOW WORD DONE
                          LDCT  H 00F       & AM2901 ,,QREG,PASS,DZ
                /                           & SPHNOP
       00354    /                           & SPFNOP ;          Q := DIMENSION
                ;
                ;         THE FOLLOWING PUSH MUST NOT LOAD THE COUNTER
                ;
                ;-NOTE:R5 is a partial product that must be added, UNSHIFTED!
                ;
                          CPUSH             & AM2901 ,R9,SRAMQR,PASS,ZB
                /                           & CONDMSR Z
                /                           & SHIFT B 0110
                /                           & SPHNOP
       00355    /                           & SPFNOP ;  SET Q0 BUF, NEVER PUSH
                          RFCT              & AM2901 MPY,R5,SRAMQR,ADD,AB
                /                           & CARRYL
```

```
ADDR    STATEMENT

        /                              & SHIFT B 1011
        /                              & SPHNOP
00356   /                              & SPFNOP ;
                   CONT                & AM2901 ,A,RAMF,PASS,ZQ
        /                              & SPHNOP
00357   /                              & SPFNOP ;  PUT HIGH WORD IN ACCUM
        ;
        ;          DECREMENT NUMBER OF DIMENSIONS
        ;
        IRES40:    JP    IRES05        & AM2901 ,R11,RAMF,SUBR,ZB
        /                              & CARRYH
        /                              & LODMSR
        /                              & SPHNOP
00358   /                              & SPFNOP ;
        ;
        ;          DIMENSION IS ZERO
        ;
        ;  - If actual dimension then A&B := A&B * 2**16
        ;
        ;  - If number of words/element then A&B := A&B * 0
        ;
        IRES50:    CONT                & AM2901 ,R11,RAMF,SUBR,ZB
        /                              & CARRYH
        /                              & LODMSR
        /                              & SPHNOP
00359   /                              & SPFNOP ;            DEC
                   LDCT  IRES05        & AM2901 B,A,RAMF,PASS,ZA
        /                              & SPHNOP
0035A   /                              & SPFNOP ;            A := B
        ;
        ;          IF NUMBER OF WORDS/ELEMENT, ZERO A TOO
        ;
                   JRP   IRES04        & AM2901 ,B,RAMF,AND,ZB
        /                              & CONDMSR Z
        /                              & SPHNOP
0035B   /                              & SPFNOP ;            B := 0
        ;
        ;          ADD OFFSET TO ACCUMULATOR AND RETURN
        ;
        IRES80:    JP    DADD          & AM2901 ,R12,QREG,PASS,ZB
        /                              & LDMAR
0035C   /                              & MREAD ;  READ UPPER WORD OF OFFS
```

```
ADDR    STATEMENT

        ;
        ;          VMA UTILITIES
        ;
        ;
        ;          DADD - DOUBLE INTEGER ADD
        ;
        ;  COMPUTES:  A&B := A&B + <OPERAND>
        ;
        ;  Q - Points to operand (MUST be in memory)
        ;
        DADD:      CONT              & AM2901 ,,QREG,ADD,ZQ
        /                            & CARRYH
        /                            & LDMAR
0035D   /                            & SPFNOP ;
                   CONT              & AM2901 A,A,RAMF,ADD,DA
        /                            & CARRYL
        /                            & SPHNOP
0035E   /                            & MREAD;    A := A + MOST SIG WORD
                   CONT              & AM2901 B,B,RAMF,ADD,DA
        /                            & CARRYL
        /                            & LODUSR
        /                            & SPHNOP
0035F   /                            & MREAD ; FREEZE, B := B + LEAST
                                                   SIG WORD
                   RET               & AM2901 ,A,RAMF,ADD,ZB
        /                            & CARRYUC
        /                            & SPHNOP
00360   /                            & SPFNOP ;

        ;**************************************************************
        ;     Lightning Single Precision Floating Point Microcode   *
        ;                                                            *
        ; The operands are manipulated in unpacked format: two words*
        ; of mantissa and one word of (two's complement) exponent.  *
        ; The names are:                                             *
        ;                                                            *
        ;     1st argument:  (XU,XL,XEXP) => result                  *
        ;     2nd argument:  (YU,YL,YEXP)         scratch: ZU,ZL    *
        ;**************************************************************
        ;
        XU:        EQU R0  (A)       ; REGISTER ASSIGNMENTS.
        XL:        EQU R1  (B)
        XEXP:      EQU R7
        ;
        YU:        EQU R5
        YL:        EQU R8
        YEXP:      EQU R6
        ;
```

```
ADDR    STATEMENT

        ZU:        EQU R9
        ZL:        EQU R10
        ;
        ;
        ;          VERIFY CORRECT REGISTERS: YU=DBLIMSB, YEXP=DBLILSB.
        ;
        YUERR:     SET YU_NE_DBLIMSB
        YXERR:     SET YEXP_NE_DBLILSB
                   IF  YUERR_OR_YXERR
                   ENDIF

        ;*************************************************************
        ;                                                            *
        ;     'FIX' FLOATING TO SINGLE INTEGER CONVERSION.           *
        ;                                                            *
        ;     CONVERT (A,B) FROM FLOATING TO INTEGER; A = RESULT.    *
        ;                                                            *
        ;     IF THE EXPONENT EXCEEDS +15, THE RESULT IS 77777B      *
        ;     AND OVERFLOW IS SET.  IF THE EXPONENT IS LESS THAN 0,  *
        ;     THE RESULT IS ZERO.                                    *
        ;                                                            *
        ;*************************************************************
        ;
        ;          (Q=0 ALREADY)
        ;          UNPACK.  (SEE CALLING SEQ. FOR UNPACK)
        ;          IF XEXP < 0, RESULT = 0.
        ;          XEXP := 15-XEXP
        ;          Q := 0
        ;          IF XEXP < 0, OVERFLOW.
        ;
        FIX:       CALL  UNPACK1      & AM2901 B,XEXP,RAMF,PASS,ZA
        /                             & SWAPEO
        /                             & IFETCH
00361   /                             & SPHNOP ;
                   CONT               & AM2901 ,YEXP,RAMF,PASS,DZ
        /                             & IMM H 000F
00362   /                             & SPHNOP ;
                   CJP   FIXZERO      & AM2901 YEXP,XEXP,RAMF,SUBS,AB
        /                             & CARRYH
        /                             & CONDLMSR SIGN
        /                             & SPFNOP
00363   /                             & SPHNOP ;
                   CJP   FIX2         & AM2901 ,,QREG,AND,ZQ
        /                             & CONDMSR NSIGN
        /                             & SPFNOP
00364   /                             & SPHNOP ;
        ;
        ;          XEXP < 0: OVERFLOW.  A=77777, O=1.
        ;
```

```
ADDR      STATEMENT

                      JZ                 & AM2901 A,A,SRAMR,EXNOR,AB
          /                              & SHIFT B 0000
          /                              & SETMSR & ENBLO
          /                              & SPFNOP
00365     /                              & SPHNOP ;

          ;           FIX1  (A,Q) := (A,Q) ARITH RIGHT SHIFT 1
          ;           FIX2  XEXP := XEXP-1
          ;                 MZ := (XEXP.EQ.0)
          ;                 IF NOT(OLD MZ) GOTO FIX1
          ;
          FIX1:       CONT               & AM2901 ,A,SRAMQR,ADD,ZB
          /                              & CARRYL
          /                              & SHIFT B 1110
          /                              & SPFNOP
00366     /                              & SPHNOP ;
          FIX2:       CJP    FIX1        & AM2901 ,XEXP,RAMF,SUBR,ZB
          /                              & CARRYH
          /                              & CONDLMSR NZ
          /                              & SPFNOP
00367     /                              & SPHNOP ;
          ;
          ;         DONE SHIFTING. IF VALUE IS NEGATIVE, AND B 0 OR ANY
          ;         BITS WERE SHIFTED INTO Q, ADD ONE TO THE FIXED VALUE.
          ;
                      CONT               & AM2901 ,XU,NOP,PASS,ZB
          /                              & CONDUSR
          /                              & SPFNOP
00368     /                              & SPHNOP ;
                      CJP   FETCH        & AM2901 B,,NOP,OR,AQ
          /                              & CONDUSR NSIGN
          /                              & SPFNOP
00369     /                              & SPHNOP ;
                      CJP   FETCH        & AM2901 ,,NOP,PASS,ZQ
          /                              & CONDUSR Z
          /                              & SPFNOP
0036A     /                              & SPHNOP ;
                      JZ                 & AM2901 ,XU,RAMF,ADD,ZB
          /                              & CARRYH
          /                              & SPFNOP
0036B     /                              & SPHNOP ;
          ;
          ;           NEGATIVE EXPONENT, RESULT=0.
          ;
          FIXZERO:    JZ                 & AM2901 XU,XU,RAMF,EXOR,AB
          /                              & SPFNOP
0036C     /                              & SPHNOP ;
```

```
ADDR    STATEMENT

        ;**********************************************************
        ;                                                         *
        ;    'FLT' SINGLE INTEGER TO FLOATING-POINT CONVERSION.   *
        ;                                                         *
        ;    CONVERT (A) FROM INTEGER TO FLOATING; (A,B) = RESULT. *
        ;                                                         *
        ;    NO UNDERFLOW, OVERFLOW OR ERROR CONDITIONS CAN OCCUR. *
        ;                                                         *
        ;**********************************************************
        ;
        ;         XU := A  (BY DEFINITION)
        ;         XEXP := 15
        ;         XL := 0
        ;         GO NORMALIZE & PACK.
        ;
        FLT:      CONT               & AM2901 ,XEXP,RAMF,PASS,DZ
        /                            & IMM H 000F
0036D   /                            & SPHNOP ;
                  CONT               & AM2901 XL,XL,RAMF,EXOR,AB
        /                            & SPFNOP
0036E   /                            & SPHNOP ;
                  JP    NORM         & AM2901 ,PC,RAMF,SUBR,ZB
        /                            & CARRYH
        /                            & SPFNOP
0036F   /                            & SPHNOP ;

        ;**********************************************************
        ;                                                         *
        ;         '..FCM'  NEGATE FLOATING-POINT VALUE IN (A,B).  *
        ;                                                         *
        ;         UNDERFLOW AND OVERFLOW CAN OCCUR.               *
        ;                                                         *
        ;**********************************************************
        ;
        ;         TEST (A)
        ;         PC := PC - 1
        ;         IF A=0 GOTO SKIP
        ;
        ..FCM:    CONT               & AM2901 ,A,NOP,PASS,ZB
        /                            & SPFNOP
00370   /                            & SPHNOP ;
                  CJP   SKIP         & AM2901 ,PC,RAMF,SUBR,ZB
        /                            & CARRYH
        /                            & CONDUSR Z
        /                            & SPFNOP
00371   /                            & SPHNOP ;
        ;
```

```
LINE    ADDR    STATEMENT

                ;          UNPACK (A,B) TO (XU,XL,XEXP)     (XU=A,XL=B)
                ;          B := - B
                ;          A := - A - BORROW
                ;          GOTO ROUNDO
                ;
                           CALL   UNPACK1      & AM2901 B,XEXP,RAMF,PASS,ZA
                /                              & SWAPEO
                /                              & SPFNOP
        00372   /                              & SPHNOP ;
                           CONT                & AM2901 ,B,RAMF,SUBS,ZB
                /                              & CARRYH
                /                              & SPFNOP
        00373   /                              & SPHNOP ;
                           JP     ROUNDO       & AM2901 ,A,RAMF,SUBS,ZB
                /                              & CARRYUC
                /                              & SPFNOP
        00374   /                              & SPHNOP ;
                ;

                ;*************************************************************
                ;                                                           *
                ;          '.FSB' FLOATING-POINT SUBTRACT.                  *
                ;                                                           *
                ;          SUBTRACT THE UNPACKED NUMBERS                    *
                ;                                                           *
                ;              (XU,XL,XEXP) - (YU,YL,YEXP) --> (XU,XL,XEXP) *
                ;                                                           *
                ;          OVERFLOW AND UNDERFLOW ARE POSSIBLE.  THE RESULT *
                ;          MAY BE UNNORMALIZED.                             *
                ;                                                           *
                ;*************************************************************
                ;
                ;        READ SECOND OPERAND FROM MEMRY; UNPACK BOTH OPERANDS.
                ;          YL := -YL
                ;          YU := -YU - BORROW
                ;          IF OVERFLOW,
                ;            YU := YU RS 1
                ;            YEXP := YEXP + 1
                ;          MERGE INTO .FAD CODE.
                ;
                .FSB:      CALL   UNPACK       & AM2901 ,,NOP,PASS,ZQ
                /                              & SPFNOP
        00375   /                              & SPHNOP ;
                           CONT                & AM2901 ,YL,RAMF,SUBS,ZB
                /                              & CARRYH
                /                              & LODUSR
                /                              & SPFNOP
        00376   /                              & SPHNOP ;
                           CONT                & AM2901 ,YU,RAMF,SUBS,ZB
```

```
LINE    ADDR    STATEMENT

                /                        & CARRYUC
                /                        & SPFNOP
        00377   /                        & SPHNOP ;
                          CJP   FADD1    & AM2901 ,,NOP,PASS,ZQ
                /                        & CONDUSR NOVR
                /                        & SPFNOP
        00378   /                        & SPHNOP ;
                          CONT           & AM2901 ,YU,SRAMR,PASS,ZB
                /                        & SHIFT B 0000
                /                        & SPFNOP
        00379   /                        & SPHNOP ;
                          JP    FADD1    & AM2901 ,YEXP,RAMF,ADD,ZB
                /                        & CARRYH
                /                        & SPFNOP
        0037A   /                        & SPHNOP ;

                ;*************************************************************
                ;                                                            *
                ;         '.FAD' FLOATING-POINT ADD.                         *
                ;                                                            *
                ;         ADD THE UNPACKED NUMBERS                           *
                ;                                                            *
                ;             (XU,XL,XEXP) + (YU,YL,YEXP) --> (XU,XL,XEXP)   *
                ;                                                            *
                ;         OVERFLOW AND UNDERFLOW ARE POSSIBLE.  THE RESULT   *
                ;         MAY BE UNNORMALIZED.                               *
                ;                                                            *
                ;*************************************************************
                ;
                ;       READ SECOND OPERAND FROM MEMRY; UNPACK BOTH OPERANDS.
                ;
                .FAD:     CALL  UNPACK   & AM2901 ,,NOP,PASS,ZQ
                /                        & SPFNOP
        0037B   /                        & SPHNOP ;
                ;
                ;         ZU := XU
                ;         IF XU=0, SWAP (XU,XL)<=>(YU,YL) & SET XEXP := YEXP
                ;         IF YU=0 THEN DONE.
                ;         Q := (XEXP-YEXP); TEST IT.
                ;
                FADD1:    CONT           & AM2901 XU,ZU,RAMF,PASS,ZA
                /                        & LODMSR
                /                        & SPFNOP
        0037C   /                        & SPHNOP ;
                          CCALL SWAPFARG & AM2901 ,YU,NOP,PASS,ZB
                /                        & CONDLMSR Z
                /                        & SPFNOP
```

```
ADDR    STATEMENT

0037D   /                             & SPHNOP ;
                CJP   NORM            & AM2901 YEXP,XEXP,QREG,SUBR,AB
        /                             & CONDLMSR Z
        /                             & CARRYL
        /                             & SPFNOP
0037E   /                             & SPHNOP ;
        ;
        ;       ZL := XL
        ;       IF Q < 0 THEN
        ;         SWAP (XU,XL)<=>(YU,YL)   (USING ZU,ZL)
        ;         XEXP := YEXP
        ;         Q := -Q
        ;
                CCALL SWAPFARG        & AM2901 XL,ZL,RAMF,PASS,ZA
        /                             & CONDMSR SIGN
        /                             & SPFNOP
0037F   /                             & SPHNOP ;

        ;       YEXP := Q   (SHIFT COUNT); TEST IT.
        ;       TEST (YEXP-25)
        ;       Q := YL
        ;       IF (YEXP-25) >= 0, DONE (SWAMP).
        ;         ELSE GOTO FADD3 (IN COUNTER)
        ;
                LDCT  FADD3           & AM2901 ,YEXP,RAMF,PASS,ZQ
        /                             & LODMSR
        /                             & SPFNOP
00380   /                             & SPHNOP ;
                CONT                  & AM2901 YEXP,,NOP,ADD,DA
        /                             & CARRYL
        /                             & LODUSR
        /                             & IMM H FFE7
00381   /                             & SPHNOP ;
                JRP   NORM            & AM2901 ,YL,QREG,PASS,ZB
        /                             & CONDUSR NSIGN
        /                             & SPFNOP
00382   /                             & SPHNOP ;
        ;
        ;       SHIFT (YU,Q) RIGHT BY (YEXP) BITS:
        ;
        ;       FADD2   (YU,Q) := (YU,Q) RS 1  (ARITHMETIC)
        ;               YEXP := YEXP - 1
        ;       FADD3   IF PREVIOUS YEXP # 0, GOTO FADD2
        ;
        FADD2:  CONT                  & AM2901 ,YU,SRAMQR,ADD,ZB
        /                             & CARRYL
        /                             & SHIFT B 1110
        /                             & SPFNOP
```

```
LINE   ADDR    STATEMENT

       00383   /                                & SPHNOP ;
               FADD3:    CJP   FADD2            & AM2901 ,YEXP,RAMF,SUBR,ZB
               /                                & CARRYH
               /                                & CONDLMSR NZ
               /                                & SPFNOP
       00384   /                                & SPHNOP ;

               ;         (XU,Q) := (XU,XL) + (YU,Q)
               ;         IF OVERFLOW,
               ;           (XU,Q) := (XU,Q) RS 1
               ;           XEXP := XEXP + 1
               ;         XL := Q
               ;         GO NORMALIZE & PACK.
               ;
                         CONT                   & AM2901 XL,,QREG,ADD,AQ
               /                                & CARRYL
               /                                & LODUSR
               /                                & SPFNOP
       00385   /                                & SPHNOP ;
                         CONT                   & AM2901 YU,XU,RAMF,ADD,AB
               /                                & CARRYUC
               /                                & SPFNOP
       00386   /                                & SPHNOP ;
                         CJP   FADD4            & AM2901 ,,NOP,PASS,ZQ
               /                                & CONDUSR NOVR
               /                                & SPFNOP
       00387   /                                & SPHNOP ;
                         CONT                   & AM2901 ,XU,SRAMQR,ADD,ZB
               /                                & CARRYL
               /                                & SHIFT B 1110
               /                                & SPFNOP
       00388   /                                & SPHNOP ;
                         CONT                   & AM2901 XU,XU,RAMF,EXOR,DA
               /                                & IMM H 8000
       00389   /                                & SPHNOP ;
                         CONT                   & AM2901 ,XEXP,RAMF,ADD,ZB
               /                                & CARRYH
               /                                & SPFNOP
       0038A   /                                & SPHNOP ;
               FADD4:    JP    NORM             & AM2901 ,XL,RAMF,PASS,ZQ
               /                                & SPFNOP
       0038B   /                                & SPHNOP ;
```

```
ADDR    STATEMENT

        ;    ROUTINE TO SWAP (XU,XL) AND (YU,YL) (GIVEN ZU,ZL SET UP)
        ;         ALSO:   XEXP := YEXP     Q := -Q
        ;         MZ := ZERO CONDITION FOR NEW YU.
        ;
        SWAPFARG: CONT                & AM2901 YU,XU,RAMF,PASS,ZA
        /                             & SPFNOP
0038C   /                             & SPHNOP ;
                  CONT                & AM2901 YL,XL,RAMF,PASS,ZA
        /                             & SPFNOP
0038D   /                             & SPHNOP ;
                  CONT                & AM2901 YEXP,XEXP,RAMF,PASS,ZA
        /                             & SPFNOP
0038E   /                             & SPHNOP ;
                  CONT                & AM2901 ZU,YU,RAMF,PASS,ZA
        /                             & LODMSR
        /                             & SPFNOP
0038F   /                             & SPHNOP ;
                  CONT                & AM2901 ZL,YL,RAMF,PASS,ZA
        /                             & SPFNOP
00390   /                             & SPHNOP ;
                  RET                 & AM2901 ,,QREG,SUBS,ZQ
        /                             & CARRYH
        /                             & SPFNOP
00391   /                             & SPHNOP ;

        ;************************************************************
        ;                                                           *
        ;         '.FMP' FLOATING-POINT MULTIPLY.                   *
        ;                                                           *
        ;         MULTIPLY THE UNPACKED NUMBERS                     *
        ;                                                           *
        ;             (XU,XL,XEXP) * (YU,YL,YEXP) --> (XU,XL,XEXP)  *
        ;                                                           *
        ;         OVERFLOW AND UNDERFLOW ARE POSSIBLE.  THE RESULT  *
        ;         MAY BE UNNORMALIZED BY ONE BIT.                   *
        ;                                                           *
        ;************************************************************
        ;
        ;       READ SECOND OPERAND FROM MEMRY; UNPACK BOTH OPERANDS.
        ;         XEXP := XEXP + YEXP + 1
        ;
        .FMP:     CALL  UNPACK        & AM2901 XL,ZL,RAMF,PASS,ZA
        /                             & SPFNOP
00392   /                             & SPHNOP ;
                  CONT                & AM2901 YEXP,XEXP,RAMF,ADD,AB
        /                             & CARRYH
        /                             & LODUSR
        /                             & SPFNOP
00393   /                             & SPHNOP ;
```

```
LINE    ADDR    STATEMENT

                ;
                ;       (RO=XU INITIALLY)
                ;       Q   := YU
                ;       ZU := 0     R.S. Q INTO QOBUF  CNTR = SHIFT OP = 14
                ;       MULTIPLY; (ZU,Q) := XU * YU  (SEE CODE FOR 'MPY')
                ;
                        CONT                & AM2901 ,YU,QREG,PASS,ZB
                /                           & SPFNOP
        00394   /                           & SPHNOP ;
                        PUSH  H 00E         & AM2901 ,ZU,SRAMQR,AND,ZB
                /                           & CARRYL
                /                           & SHIFT B 1110
                /                           & SPFNOP
        00395   /                           & SPHNOP ;
                        RFCT                & AM2901 MPY,ZU,SRAMQR,ADD,AB
                /                           & CARRYL
                /                           & SHIFT B 1110
                /                           & SPFNOP
        00396   /                           & SPHNOP ;
                        CONT                & AM2901 MPY,ZU,SRAMQR,SUBR,AB
                /                           & CARRYL
                /                           & SHIFT B 1110
                /                           & SPFNOP
        00397   /                           & SPHNOP ;
                        CONT                & AM2901 ,XL,RAMF,PASS,ZQ
                /                           & SPFNOP
        00398   /                           & SPHNOP ;

                ;       (RO=XU STILL)
                ;       YL := YL LS 4      (FOR FMPYSUB)
                ;       XL := XL + RO*YL; PROPOGATE CARRY.
                ;       YL := ZL LS 4      (DONE IN FMPYSUB)
                ;       RO  := YU
                ;       XL := XL + RO*YL; PROPOGATE CARRY.
                ;       XU := ZU
                ;
                ;
                        CALL  FMPYSUB       & AM2901 YL,YL,RAMA,PASS,DZ
                /                           & L4D
        00399   /                           & SPHNOP ;
                        CALL  FMPYSUB       & AM2901 YU,RO,RAMF,PASS,ZA
                /                           & SPFNOP
        0039A   /                           & SPHNOP ;
                        JP    NORM          & AM2901 ZU,XU,RAMF,PASS,ZA
                /                           & SPFNOP
        0039B   /                           & SPHNOP ;
```

```
ADDR    STATEMENT

        ;          MULTIPLY SUBROUTINE; MULTIPLIES RO*YL, AND ADDS THE
        ;          MSW TO (ZU,XL).  RO IS SIGNED, AND THE LOWER 8 BITS
        ;          OF YL ARE ZERO.
        ;          THE PRODUCT IS DEVELOPED IN (YL,Q); IF IT IS
        ;          NEGATIVE, WE MUST SUBTRACT ONE FROM ZU. IF THE ADD
        ;          TO XL HAS A CARRY OUT, WE MUST ADD ONE TO ZU.
        ;
        ;          YL := YL LS 4  (COMPLETE SWAPPING BYTES)
        ;          Q   := YL
        ;          (DO 8 MULTIPLY STEPS; RESULT -> (YL,Q)
        ;
        FMPYSUB:   CONT                & AM2901 YL,YL,RAMA,PASS,DZ
        /                              & L4D
0039C   /                              & SPHNOP ;
                   LDCT  H 007         & AM2901 ,YL,QREG,PASS,ZB
        /                              & RSTMSR
        /                              & SPFNOP
0039D   /                              & SPHNOP ;      (RSTMSR: MZ:=0)
                   CPUSH               & AM2901 ,YL,SRAMQR,AND,ZB
        /                              & CARRYL
        /                              & CONDMSR Z
        /                              & SHIFT B 1110
        /                              & SPFNOP
0039E   /                              & SPHNOP ;(CONDMSR Z ALWAYS FALSE)
                   RFCT                & AM2901 MPY,YL,SRAMQR,ADD,AB
        /                              & CARRYL
        /                              & SHIFT B 1110
        /                              & LODMSR &ENBLO
        /                              & SPFNOP
0039F   /                              & SPHNOP ;
        ;
        ;          (BIT16 INDICATES IF PRODUCT < 0)
        ;          XL := XL + YL;  PROPOGATE CARRY TO ZU.
        ;          YL := ZL LS 4  (HAVE TO USE THE CYCLE ANYWAY)
        ;          IF PRODUCT<0, DECREMENT ZU.
        ;
                   CONT                & AM2901 YL,XL,RAMF,ADD,AB
        /                              & CARRYL
        /                              & LODUSR
        /                              & SPFNOP
003A0   /                              & SPHNOP ;
                   CONT                & AM2901 ,ZU,RAMF,ADD,ZB
        /                              & CARRYUC
        /                              & SPFNOP
003A1   /                              & SPHNOP ;
                   CRET                & AM2901 ZL,YL,RAMA,PASS,DZ
        /                              & CONDMSR SGNXOVR
        /                              & L4D
```

```
LINE    ADDR    STATEMENT

        003A2   /                           & SPHNOP ;
                          RET               & AM2901 ,ZU,RAMF,SUBR,ZB
                /                           & CARRYH
                /                           & SPFNOP
        003A3   /                           & SPHNOP ;

                ;************************************************************
                ;                                                           *
                ;         '.FDV' FLOATING-POINT DIVIDE.                     *
                ;                                                           *
                ;         DIVIDE THE UNPACKED NUMBERS                       *
                ;                                                           *
                ;             (XU,XL,XEXP) / (YU,YL,YEXP) --> (XU,XL,ZEXP)  *
                ;                                                           *
                ;         THE OPERANDS MUST BE NORMALIZED OR ZERO.          *
                ;                                                           *
                ;         OVERFLOW AND UNDERFLOW ARE POSSIBLE;  DIVIDE BY   *
                ;         ZERO IS TREATED AS OVERFLOW.  THE RESULT MAY BE   *
                ;         UNNORMALIZED BY ONE BIT.                          *
                ;                                                           *
                ;************************************************************
                ;
                ;       READ SECOND OPERAND FROM MEMRY; UNPACK BOTH OPERANDS.
                ;
                .FDV:     CALL  UNPACK      & AM2901 ,,NOP,PASS,ZQ
                /                           & SPFNOP
        003A4   /                           & SPHNOP ;
                ;
                ;         (R4,Q) = (XU,XL) RS 2
                ;         XEXP := XEXP - YEXP
                ;         IF R4<0 THEN
                ;           R4 := -R4
                ;           Q  := -Q  - BORROW
                ;         XU := XU .XOR. YU
                ;         XL := 0
                ;         IF YU<0 THEN
                ;           YL := -YL
                ;           YU := -YU - BORROW
                ;
                          CONT              & AM2901 ,XL,QREG,PASS,ZB
                /                           & SPFNOP
        003A5   /                           & SPHNOP ;
                          CONT              & AM2901 XU,R4,SRAMQR,ADD,ZA
                /                           & CARRYL
                /                           & SHIFT B 1110
                /                           & SPFNOP
        003A6   /                           & SPHNOP ;
                          CONT              & AM2901 ,R4,SRAMQR,ADD,ZB
                /                           & CARRYL
```

```
LINE    ADDR    STATEMENT

                /                                   & SHIFT B 1110
                /                                   & SPFNOP
        003A7   /                                   & SPHNOP ;
                          CJP   FDIV1               & AM2901 YEXP,XEXP,RAMF,SUBR,AB
                /                                   & CARRYL
                /                                   & CONDUSR NSIGN
                /                                   & SPFNOP
        003A8   /                                   & SPHNOP ;
                          CONT                      & AM2901 ,,QREG,SUBS,ZQ
                /                                   & CARRYH
                /                                   & LODUSR
                /                                   & SPFNOP
        003A9   /                                   & SPHNOP ;
                          CONT                      & AM2901 ,R4,RAMF,SUBS,ZB
                /                                   & CARRYUC
                /                                   & SPFNOP
        003AA   /                                   & SPHNOP ;
                FDIV1:    CONT                      & AM2901 YU,YEXP,RAMF,PASS,ZA
                /                                   & LODMSR
                /                                   & SPFNOP
        003AB   /                                   & SPHNOP ;
                          CJP   FOFL                & AM2901 XU,YU,RAMF,EXOR,AB
                /                                   & CONDMSR Z
                /                                   & SPFNOP
        003AC   /                                   & SPHNOP ;
                          CJP   FDIV2               & AM2901 ,XL,RAMF,AND,ZB
                /                                   & CONDMSR NSIGN
                /                                   & SPFNOP
        003AD   /                                   & SPHNOP ;
                          CONT                      & AM2901 ,YL,RAMF,SUBS,ZB
                /                                   & CARRYH
                /                                   & LODUSR
                /                                   & SPFNOP
        003AE   /                                   & SPHNOP ;
                          CONT                      & AM2901 ,YEXP,RAMF,SUBS,ZB
                /                                   & CARRYUC
                /                                   & SPFNOP
        003AF   /                                   & SPHNOP ;
                ;
                ;         YEXP := YEXP + 1
                ;         CALL FDIVSUB TO DIVIDE:
                ;           R4 := REM (W/O LAST CORRECTION)
                ;           Q  := .NOT. QUOTIENT
                ;
                FDIV2:    CALL  FDIVSUB             & AM2901 ,YEXP,RAMF,ADD,ZB
                /                                   & CARRYH
                /                                   & SPFNOP
        003B0   /                                   & SPHNOP ;
```

```
LINE   ADDR    STATEMENT

               ;          IF R4<0 THEN
               ;            R4 := R4 + YEXP
               ;          ZU := .NOT. Q
               ;          IF ZU<0 THEN RESULT := 0
               ;          R4 := R4 + ZU
               ;
                          CONT                & AM2901 ,R4,NOP,PASS,ZB
               /                              & LODMSR
               /                              & SPFNOP
       003B1   /                              & SPHNOP ;
                          CJP   FDIV3         & AM2901 ,ZU,RAMF,EXNOR,ZQ
               /                              & CONDLMSR NSIGN
               /                              & SPFNOP
       003B2   /                              & SPHNOP ;
                          CONT                & AM2901 YEXP,R4,RAMF,ADD,AB
               /                              & CARRYL
               /                              & SPFNOP
       003B3   /                              & SPHNOP ;
               FDIV3:     CJP   FDIV5         & AM2901 ZU,R4,RAMF,ADD,AB
               /                              & CARRYL
               /                              & CONDMSR SIGN
               /                              & SPFNOP
       003B4   /                              & SPHNOP ;
               ;
               ;          R0 := ZU
               ;          YL := YL LS 4
               ;          CALL FMPYSUB TO:  XL := YL*R0 (UPPER)
               ;
                          CONT                & AM2901 ZU,R0,RAMF,PASS,ZA
               /                              & SPFNOP
       003B5   /                              & SPHNOP ;
                          CALL  FMPYSUB       & AM2901 YL,YL,RAMA,PASS,DZ
               /                              & L4D
       003B6   /                              & SPHNOP ;
               ;
               ;          R4 := R4-XL
               ;          (R4,Q) := (R4,Q) RS 2
               ;          XEXP := XEXP + 1
               ;          CALL FDIVSUB DIVIDE:  Q := .NOT. QUOTIENT
               ;
                          CONT                & AM2901 XL,R4,SRAMQR,SUBR,AB
               /                              & CARRYL
               /                              & SHIFT B 0110
               /                              & SPFNOP
       003B7   /                              & SPHNOP ;
                          CONT                & AM2901 ,R4,SRAMQR,PASS,ZB
               /                              & SHIFT B 0110
               /                              & SPFNOP
```

```
LINE    ADDR    STATEMENT

        003B8   /                            & SPHNOP ;
                          CALL  FDIVSUB      & AM2901 ,XEXP,RAMF,ADD,ZB
                /                            & CARRYH
                /                            & SPFNOP
        003B9   /                            & SPHNOP ;

                ;         XL := .NOT. Q
                ;         Q := 0
                ;         (XL,Q) := (XL,Q) LS 2 CIRCULAR
                ;
                          CONT               & AM2901 ,XL,RAMF,EXNOR,ZQ
                /                            & SPFNOP
        003BA   /                            & SPHNOP ;
                          PUSH  H 001        & AM2901 ,,QREG,AND,ZQ
                /                            & SPFNOP
        003BB   /                            & SPHNOP ;
                          RFCT               & AM2901 ,XL,SRAMQL,PASS,ZB
                /                            & SHIFT B 1111
                /                            & SPFNOP
        003BC   /                            & SPHNOP ;
                ;
                ;         XU := ZU + Q
                ;         IF OPERAND SIGNS DIFFER, NEGATE RESULT.
                ;
                          CONT               & AM2901 ,YU,NOP,PASS,ZB
                /                            & LODUSR
                /                            & SPFNOP
        003BD   /                            & SPHNOP ;
                          CJP   NORM         & AM2901 ZU,XU,RAMF,ADD,AQ
                /                            & CARRYL
                /                            & CONDUSR NSIGN
                /                            & SPFNOP
        003BE   /                            & SPHNOP ;
                          CONT               & AM2901 ,XL,RAMF,SUBS,ZB
                /                            & CARRYH
                /                            & LODUSR
                /                            & SPFNOP
        003BF   /                            & SPHNOP ;
                          JP    NORM         & AM2901 ,XU,RAMF,SUBS,ZB
                /                            & CARRYUC
                /                            & SPFNOP
        003C0   /                            & SPHNOP ;
                ;
                ;         ZERO DIVIDEND (FOUND OUT THE HARD WAY).
                ;         XU := 0    (XL=0 ALREADY)
                ;
                FDIV5:    JP    NORM         & AM2901 ,XU,RAMF,AND,ZB
                /                            & SPFNOP
        003C1   /                            & SPHNOP ;
```

```
ADDR    STATEMENT

        ;         DIVIDE SUBROUTINE; DIVIDES (R4,Q) BY YEXP.
        ;         THE FINAL REMAINDER RESTORE IS NOT DONE, SINCE
        ;         ON THE SECOND CALL THE REMAINDER IS DISCARDED.
        ;
        FDIVSUB:  PUSH  H 010      & AM2901 ,R4,NOP,PASS,ZB
        /                          & DIVUCOND
        /                          & SPFNOP
003C2   /                          & SPHNOP ;
                  RFCT             & AM2901 YEXP,DIV,SRAMQL,SUBR,AB
        /                          & DIVUCOND
        /                          & CARRYL
        /                          & SHIFT B 1111
        /                          & SPFNOP
003C3   /                          & SPHNOP ;
                  RET              & AM2901 ,R4,SRAMR,PASS,ZB
        /                          & SHIFT B 1111
        /                          & SPFNOP
003C4   /                          & SPHNOP ;

        ;******************************************************
        ;                                                     *
        ;         UNPACK.  FETCH 2ND OPERAND & UNPACK BOTH:   *
        ;                                                     *
        ;            (A,B) => (XU,XL,XEXP)                    *
        ;                                                     *
        ;            (MEM,MEM+1) => (YU,YL,YEXP)              *
        ;                                                     *
        ;         NOTE THAT A=XU AND B=XL.                    *
        ;         TO UNPACK (A,B) ONLY (FOR FIX), ENTER AT    *
        ;         'UNPACK1' WITH E & O SWAPPED AND XEXP=B.    *
        ;                                                     *
        ;******************************************************
        ;
        ;         XEXP := XL
        ;         SWAP E & O
        ;         (YU,YEXP) := (MEM,MEM+1)
        ;
        UNPACK:   CALL  DIARG      & AM2901 XL,XEXP,RAMF,PASS,ZA
        /                          & SWAPEO
        /                          & SPFNOP
003C5   /                          & SPHNOP ;
        ;
        ;         YL   := (YEXP .AND. FF00)
        ;         YEXP := (YEXP .XOR. YL) SHIFT RIGHT INTO MC
        ;         IF YEXP<0 THEN
        ;           YEXP := YEXP .OR. FF80
        ;         EXIT
        ;
```

```
LINE    ADDR     STATEMENT

                           CONT                 & AM2901 YEXP,YL,RAMF,AND,DA
                 /                              & IMM H FF00
        003C6    /                              & SPHNOP ;
                           CONT                 & AM2901 YL,YEXP,SRAMR,EXOR,AB
                 /                              & SHIFT B 0010
                 /                              & SPFNOP
        003C7    /                              & SPHNOP ;
                           CJP   UNPACK1        & AM2901 ,,NOP,PASS,ZQ
                 /                              & CONDMSR NC
                 /                              & SPFNOP
        003C8    /                              & SPHNOP ;
                           CONT                 & AM2901 YEXP,YEXP,RAMF,OR,DA
                 /                              & IMM H FF80
        003C9    /                              & SPHNOP ;

                 ;         XL   := (XEXP .AND. FF00)
                 ;         XEXP := (XEXP .XOR. XL) RIGHT SHIFT INTO MC
                 ;         IF MC=1 THEN
                 ;           XEXP := XEXP .OR. FF80
                 ;         MC := 0
                 ;         SWAP E,O BACK:  E RESTORED, O=0.
                 ;
                 UNPACK1:  CONT                 & AM2901 XL,XL,RAMF,AND,DA
                 /                              & IMM H FF00
        003CA    /                              & SPHNOP ;
                           CONT                 & AM2901 XL,XEXP,SRAMR,EXOR,AB
                 /                              & SHIFT B 0010
                 /                              & SPFNOP
        003CB    /                              & SPHNOP ;
                           CJP   UNPACK2        & AM2901 ,XEXP,NOP,ADD,ZB
                 /                              & CARRYL
                 /                              & CONDLMSR NC & ENBLC
                 /                              & SPFNOP
        003CC    /                              & SPHNOP ;
                           CONT                 & AM2901 XEXP,XEXP,RAMF,OR,DA
                 /                              & IMM H FF80
        003CD    /                              & SPHNOP ;
                 UNPACK2:  RET                  & AM2901 ,XEXP,NOP,PASS,ZB
                 /                              & SWAPEO
                 /                              & SPFNOP
        003CE    /                              & SPHNOP ;
```

```
ADDR    STATEMENT

        ;          NORMALIZE. ENTER WITH (XU,XL)=VALUE; XEXP=EXPONENT.
        ;
        ;          EXIT TO NEXT INSTRC WITH (A,B)=RESULT, O SET/RESET.
        ;
        ;          Mn  := A<15>
        ;          Q   := XL
        ;          YEXP := XEXP + 1
        ;          IF XU=0 THEN
        ;            B := 0
        ;            IF XL=0 THEN EXIT
        ;          XEXP := YEXP + 1
        ;
        NORM:      CONT                & AM2901 ,XL,QREG,ADD,ZB
        /                              & CARRYL
        /                              & LODMSR &ENBLO
        /                              & SPFNOP
003CF   /                              & SPHNOP ;
                   CJP   NORM1         & AM2901 ,XU,NOP,PASS,ZB
        /                              & CONDLMSR NZ
        /                              & SPFNOP
003D0   /                              & SPHNOP ;
                   CJP   SKIP          & AM2901 B,B,RAMF,EXOR,AB
        /                              & CONDMSR Z
        /                              & SPFNOP
003D1   /                              & SPHNOP ;
        NORM1:     PUSH                & AM2901 XEXP,XEXP,RAMF,ADD,DA
        /                              & CARRYL
        /                              & IMM H 0002
003D2   /                              & SPHNOP ;
        ;
        ;          REPEAT
        ;            CNTR := 1                        (FOR USE LATER)
        ;            IRSKIP := Mn .XOR. A<15>
        ;            XEXP := XEXP-1
        ;            (A,Q) := (A,Q) LS 1
        ;          UNTIL IRSKIP
        ;
                   LDCT  H 001         & AM2901 A,XEXP,RAMA,SUBR,ZB
        /                              & CARRYH
        /                              & CONDMSR SIGN
        /                              & SPFNOP
003D3   /                              & SPHNOP ;
                   LOOP                & AM2901 ,A,SRAMQL,PASS,ZB
        /                              & SHIFT B 0110
        /                              & CONDEXT IRSKIP
        /                              & SPFNOP
003D4   /                              & SPHNOP ;
```

```
LINE    ADDR    STATEMENT

6037            ;       ON THE NEXT-TO-LAST SHIFT, THE LAST COPY OF THE SIGN
6038            ;       WAS SHIFTED OUT.  ON THE LAST SHIFT, THE FIRST BIT
6039            ;       NOT EQUAL TO THE SIGN WAS SHIFTED OUT.  THESE BITS
6040            ;       ARE NOW RESTORED. THE EXPONENT WAS DECREMENT TWO EXTRA
6041            ;       TIMES, BUT WAS INCREMENTED TWICE BEFORE THE LOOP.
6042            ;
6043            ;          (A,Q) := (Mn,Mn,A,Q)
6044            ;          A := A .XOR. 40000B
6045            ;
6046                       CONT                 & AM2901 ,A,SRAMQR,PASS,ZB
6047            /                               & SHIFT B 0101
6048            /                               & SPFNOP
6049    003D5   /                               & SPHNOP ;
6050                       CONT                 & AM2901 ,A,SRAMQR,PASS,ZB
6051            /                               & SHIFT B 0101
6052            /                               & SPFNOP
6053    003D6   /                               & SPHNOP ;
6054                       CONT                 & AM2901 A,A,RAMF,EXOR,DA
6055            /                               & IMM H 4000
6056    003D7   /                               & SPHNOP ;
6057            ;
6058            ;          ROUND:
6059            ;
6060            ;          YL := 177B
6061            ;          B := Q + YL  (+1 IF A>=0)
6062            ;          A := A + CARRY
6063            ;
6064                       CONT                 & AM2901 ,YL,RAMF,PASS,DZ
6065            /                               & IMM H 007F
6066    003D8   /                               & SPHNOP ;
6067                       CONT                 & AM2901 A,A,NOP,ADD,AB
6068            /                               & CARRYL
6069            /                               & LODUSR
6070            /                               & SPFNOP
6071    003D9   /                               & SPHNOP ;  ADD 200B IF A NEGATIVE
6072                       CONT                 & AM2901 YL,B,RAMF,ADD,AQ
6073            /                               & CARRYNUC
6074            /                               & SPFNOP ;
6075    003DA   /                               & SPHNOP ;
6076                       CONT                 & AM2901 ,A,RAMF,ADD,ZB
6077            /                               & CARRYNUC
6078            /                               & SPFNOP
6079    003DB   /                               & SPHNOP ;
```

```
ADDR    STATEMENT

        ;          SPECIAL CASES:
        ;
        ;          IF OVERFLOW ON CARRY PROPOGATE,
        ;            A := A RS 1
        ;            XEXP := XEXP + 1
        ;          ELSE IF A=140000,
        ;            A := A LS 1
        ;            XEXP := XEXP - 1
        ;
        ROUND0:    CJP   ROUND1      & AM2901 A,A,NOP,ADD,AB
        /                            & CARRYL
        /                            & CONDUSR NOVR
        /                            & SPFNOP
003DC   /                            & SPHNOP ;
                   CONT              & AM2901 ,A,SRAMR,PASS,ZB
        /                            & SHIFT B 0000
        /                            & SPFNOP
003DD   /                            & SPHNOP ;
                   JP    ROUND2      & AM2901 ,XEXP,RAMF,ADD,ZB
        /                            & CARRYH
        /                            & SPFNOP
003DE   /                            & SPHNOP ;
        ROUND1:    CJP   ROUND2      & AM2901 ,,NOP,PASS,ZQ
        /                            & CONDUSR OVR
        /                            & SPFNOP
003DF   /                            & SPHNOP ;
                   CONT              & AM2901 ,A,SRAML,PASS,ZB
        /                            & SHIFT B 0000
        /                            & SPFNOP
003E0   /                            & SPHNOP ;
                   CONT              & AM2901 ,XEXP,RAMF,SUBR,ZB
        /                            & CARRYH
        /                            & SPFNOP
003E1   /                            & SPHNOP ;

        ;          CHECK FOR EXPONENT UNDERFLOW OR OVERFLOW:
        ;          UPPER 9 BITS MUST BE THE SAME.  FORMAT EXPONENT.
        ;
        ;          YEXP := XEXP .AND. 200B
        ;          YEXP := XEXP + YEXP
        ;          YEXP := YEXP + YEXP + CARRY
        ;          IF (YEXP.AND.177400B) .NE. 0, OVERFLOW/UNDERFLOW
        ;
        ROUND2:    CONT              & AM2901 XEXP,YEXP,RAMF,AND,DA
        /                            & IMM H 0080
003E2   /                            & SPHNOP ;
                   CONT              & AM2901 XEXP,YEXP,RAMF,ADD,AB
        /                            & CARRYL
        /                            & LODUSR
```

```
LINE   ADDR    STATEMENT

                /                               & SPFNOP
       003E3    /                               & SPHNOP ;
                          CONT                  & AM2901 YEXP,YEXP,RAMF,ADD,AB
                /                               & CARRYUC
                /                               & SPFNOP
       003E4    /                               & SPHNOP ;
                          CONT                  & AM2901 YEXP,,NOP,AND,DA
                /                               & LODUSR
                /                               & IMM H FF00
       003E5    /                               & SPHNOP ;
                          CJP    FOFLUFL        & AM2901 ,XEXP,NOP,PASS,ZB
                /                               & CONDUSR NZ
                /                               & SPFNOP
       003E6    /                               & SPHNOP ;
                ;
                ; PACK EXPONENT WITH 2ND MANTISSA WORD, CLEAR OVRFLW, & EXIT.
                ;
                          CONT                  & AM2901 B,B,RAMF,AND,DA
                /                               & IMM H FF00
       003E7    /                               & SPHNOP ;
                          JP     SKIP           & AM2901 YEXP,B,RAMF,OR,AB
                /                               & RSTMSR & ENBLO
                /                               & SPFNOP
       003E8    /                               & SPHNOP ;

                ;         FLOATING UNDERFLOW & OVERFLOW.
                ;         TEST SIGN OF EXPONENT.
                ;         NEGATIVE: UNDERFLOW, RESULT = 0
                ;         POSITIVE: OVERFLOW, RESULT = 77777B,177776B
                ;       *** NOTE: THE SHIFT BELOW TAKES ADVANTAGE OF THE
                ;       *** FACT THAT THE FOUR LSB OF 'SKIP' ARE ALL ZERO.
                ;
                FOFLUFL:  CJP    FOFL           & AM2901 ,A,RAMF,AND,ZB
                /                               & CONDUSR NSIGN
                /                               & SPFNOP
       003E9    /                               & SPHNOP ;
                          JP     SKIP           & AM2901 ,B,RAMF,AND,ZB
                /                               & SETMSR & ENBLO
                /                               & SPFNOP
       003EA    /                               & SPHNOP ;
                FOFL:     CONT                  & AM2901 A,A,SRAMR,EXNOR,AB
                /                               & SHIFT B 0000
                /                               & SPFNOP
       003EB    /                               & SPHNOP ;
                          JP     SKIP           & AM2901 B,B,SRAML,EXNOR,AB
                /                               & SETMSR & ENBLO
                /                               & SPFNOP
       003EC    /                               & SPHNOP ;
```

```
LINE    ADDR    STATEMENT

                ;
                ;          VMA PMAP FAULT FIX PATCH
                ;
                ;    - This code can be moved into the fault handler routine
                ;      when new 105XXX decode proms are made.
                ;
                VMAFPMAP: CONT                & AM2901 ,A,NOP,PASS,ZB
                /                             & LODUSR
                /                             & SPHNOP
        003ED   /                             & SPFNOP ;          TEST SIGN OF A
                          CJP    VMAFAUL1     & AM2901 ,,NOP,PASS,ZQ
                /                             & CONDUSR NSIGN
                /                             & SPHNOP
        003EE   /                             & SPFNOP ;      IF NO ERROR RETURN
                          CONT                & AM2901 ,PC,NOP,SUBR,ZB
                /                             & CARRYH
                /                             & LDMAR
        003EF   /                             & IFETCH ;       TAKE ERROR RETURN
                          JZ                  & AM2901 ,A,RAMF,PASS,DZ
                /                             & IMM H 0050
        003F0   /                             & SPHNOP ;   ERROR CODE 80 DECIMAL
                ;
                ;         FILL AREA
                ;
                IF        FILL
                LIST
                ENDIF

                ;
                ;         LAST WORD IN MICROSTORE
                ;
                ;    - THE PON SELF-TEST JUMPS HERE TO TEST THE MICROSEQUENCER
                ;      Y-BUS
                ;
        003FF             ORG    H 3FF
                LASTWD:   JP     INTPONOK     & AM2901 ,B,RAMF,ADD,ZB
                /                             & CARRYH
                /                             & SPHNOP
        003FF   /                             & SPFNOP ;        B := 1 FOR LED'S
                END

   TOTAL ASSEMBLY ERRORS =    0
```

SYMBOL TABLE

A	A	00000	AB	A	00001	ADD	A	00000	ADX.	A	00202
AD.	E	00003	AD.I	E	00002	AM2901	D		AND	A	00004
AQ	A	00000	ASCCACCE	A	00033	ASCCACLE	A	0002F	ASCCACME	A	00037
ASCCANOP	A	0002A	ASCLACCE	A	00031	ASCLACLE	A	0002D	ASCLACME	A	00035
ASCLANOP	A	00028	ASCMACCE	A	00032	ASCMACLE	A	0002E	ASCMACME	A	00036
ASCMANOP	A	00029	ASCONT	A	0002B	ASL	A	0005E	ASLCONT	A	0013D
ASLOVFL	A	00145	ASNOPCCE	A	00034	ASNOPCLE	A	00030	ASNOPCME	A	00038
ASR	A	00061	B	A	00001	BITSB	A	00225	BITWRT	A	0021F
BYTESWAP	A	0023D	C	A	0000B	CAB	A	00010	CALL	D	
CARRYEXT	D		CARRYH	D		CARRYL	D		CARRYNUC	D	
CARRYREG	D		CARRYUC	D		CBS	A	0021B	CBT	A	0025A
CCALL	D		CJP	D		CJPP	D		CLC00	A	0008E
CLC04	A	000BE	CLC05	A	000C9	CLC06	A	000D8	CLF00	A	00093
CLF02	A	000AF	CLF05	A	000CC	CLF06	A	000DA	CLO	A	000A2
CLRDTST	A	0000A	CLRMPEN	A	0000E	CLRMPI	A	00002	CLRPEI	A	00001
CLRPFWI	A	00006	CLRPSFF	A	0000C	CLRTBT	A	00004	CLRTDI	A	00008
CMW	A	00245	CMWLT	A	00250	CMWNEQ	A	0024E	CNNOP	A	00000
CONDEXT	D		CONDLMSR	D		CONDMSR	D		CONDUSR	D	
CONT	D		COPYABXY	A	00205	COPYXYAB	A	00206	CPUID	A	00002
CPUSH	D		CP.	A	00009	CP.1	A	0000A	CP.I	A	00008
CRET	D		CVECT	D		CXY	A	00012	DA	A	00005
DADD	A	0035D	DBLILSB	A	00006	DBLIMSB	A	00005	DCZFER	A	001E4
DIARG	A	00139	DIV	A	00014	DIVCOND	D		DIVD	A	00146
DIVD05	A	0014C	DIVD20	A	00157	DIVD25	A	0015B	DIVD30	A	0015D
DIVD40	A	0015E	DIVDOVFL	A	00160	DIVENT	A	0007A	DIVUCOND	D	
DLD	A	00071	DLOAD	A	000E2	DMSEXIT	A	0028E	DQ	A	00006
DST	A	00076	DSXY	A	00207	DZ	A	00007	EAGUIT	A	000FF
ECIRRD	A	00005	ENBLC	D		ENBLO	D		ENCN	D	
ENVE	D		EXNOR	A	00007	EXOR	A	00006	FADD1	A	0037C
FADD2	A	00383	FADD3	A	00384	FADD4	A	0038B	FDIV1	A	003AB
FDIV2	A	003B0	FDIV3	A	003B4	FDIV5	A	003C1	FDIVSUB	A	003C2
FETCH	A	00000	FILL	A	00001	FILLER	D		FIX	A	00361
FIX1	A	00366	FIX2	A	00367	FIXZERO	A	0036C	FLT	A	0036D
FMPYSUB	A	0039C	FOFL	A	003EB	FOFLUFL	A	003E9	GENMPV	A	002C1
GETPTE	A	0032F	HLT	A	00081	HLT0	A	00161	HLT1	A	00163
IAND	E	00006	IANDI	E	00005	ICRS	A	00007	IFETCH	D	
IMAPLOC	A	00003	IMM	D		IMMB	D		INITIAL	A	0026F
INTDEAD	A	001A3	INTERRPT	A	0019C	INTEXIT	A	001BA	INTFTCH	A	001B1
INTHLDPC	A	0019D	INTIO	A	001C2	INTPARTY	A	001B6	INTPEND	A	00275
INTPFW	A	001D4	INTPON	A	0019E	INTPONOK	A	001A5	INTPROT	A	001D0
INTRPT	A	00007	INTSLRQ	A	001C6	INTSLRQL	A	001C7	INTTBG	A	001BE
INTTBL	A	001F0	INTUIT	A	001CC	INVMSR	D		IODC0001	A	00187
IODC0010	A	00189	IODC0011	A	0018B	IODC0110	A	0018D	IODC0111	A	0018F
IODC1001	A	00190	IODC1010	A	00196	IODC1011	A	00198	IODC1110	A	0019A
IODCXXXX	A	00186	IODECODE	A	00174	IOGGE20	A	00082	IOGLT20	A	00087
IOGLT20V	A	0008B	IOHSHAK1	A	00171	IOHSHAK2	A	00172	IOHSHAKE	A	0016D
IONOP	A	0008C	IOR	E	0000E	IORD	D		IORI	E	0000D
IORQ	A	00006	IOSKIP	A	000AE	IOWR	D		IROT3	A	00008

```
                              SYMBOL TABLE

IR11      A  00002  IRES01    A  00343  IRES04    A  00347  IRES05    A  00348
IRES10    A  0034D  IRES40    A  00358  IRES50    A  00359  IRES80    A  0035C
IRMRG     D         IRSKIP    A  00000  ISXY      A  00209  ISZ       E  00011
ISZI      E  00010  JLY       A  00217  JL.       A  0005F  JMAP      D
JMP       E  00017  JMPI      E  00014  JP        D         JPP       D
JPY       A  00219  JRP       D         JSB       A  0001C  JSBI      A  00018
JSBIVMA   A  00019  JTAB      D         JZ        D         L4D       D
LABXY     A  0020B  LASTWD    A  003FF  LBT       A  0022A  LCBT      A  0025B
LCMW      A  00246  LDAER     D         LDBYTE    A  00232  LDCT      D
LDIM1     A  00001  LDMAPD    D         LDMAR     D         LDMDOR    D
LDST      D         LDX.      A  00200  LD.       E  00020  LD.I      E  0001F
LEDWR     A  00003  LIA05C    A  000D4  LIA05H    A  000D2  LI.01     A  000A8
LI.04     A  000C6  LI.07     A  000E1  LMAP0     A  00298  LMBT      A  00255
LMOVBYTE  A  002DA  LMOVWORD  A  002D3  LMVW      A  00240  LODMSR    D
LODMSRCI  D         LODUSR    D         LODUSRCI  D         LODUSROR  D
LOOP      D         LOWSC     A  00004  LPMR0     A  0028A  LSFB      A  00269
LSL       A  00065  LSR       A  00068  LT        A  00003  LVL0      A  00000
LWD10     A  00285  MAPD      A  0000D  MAPRD     A  00001  MAPWR     A  00002
MAPX      A  0000E  MBT       A  00253  MIAK      D         MICREVID  A  00800
MI.01     A  000A9  MKLROFF   D         MKLRON    D         MOV000    A  002CA
MOV010    A  002CC  MOV100    A  002CE  MOV110    A  002D0  MOVBYTE   A  002D9
MOVWORD   A  002D2  MPEN      A  00005  MPY       A  00013  MPY4      A  00017
MREAD     D         MRGIND    A  00101  MULT      A  0007B  MVW       A  0023F
MWRITE    D         MWRTIND   A  00108  NARG      A  00000  NC        A  0000A
NEWPC     A  00001  NOP       A  00001  NORM      A  003CF  NORM1     A  003D2
NOSKIP    A  00231  NOTRS     A  00005  NOVR      A  00006  NSIGN     A  0000E
NUCLDMSR  D         NZ        A  00004  OR        A  00003  OT.00     A  0009F
OT.01     A  000A7  OT.02C    A  000B6  OT.02H    A  000B8  OT.04     A  000C4
OVR       A  00007  PASS      A  00003  PC        A  0000F  PELENH    A  00003
PELENL    A  00002  PORM      A  0001F  POSIMAPR  A  00294  PRLEN     A  00004
PTELKUP   A  00335  PTELKUP5  A  00341  PTELKUPX  A  00340  PUSH      D
QPEI      A  00003  QREG      A  00000  R0        A  00000  R1        A  00001
R10       A  0000A  R11       A  0000B  R12       A  0000C  R13       A  0000D
R14       A  0000E  R15       A  0000F  R2        A  00002  R3        A  00003
R4        A  00004  R5        A  00005  R6        A  00006  R7        A  00007
R8        A  00008  R9        A  00009  RAMA      A  00002  RAMF      A  00003
READCIR   A  001ED  REFETCH   A  00165  REFETCH2  A  0016B  RESOLVE   A  00102
RESOLVE1  A  00103  RET       D         RFCT      D         RFETCH    D
ROTATE    A  0000A  ROTATEC   A  00009  ROUND0    A  003DC  ROUND1    A  003DF
ROUND2    A  003E2  RPCT      D         RRL       A  0006B  RRR       A  0006E
RSTMSR    D         SABXY     A  0020F  SAVEAB    A  002FA  SBS       A  0021D
SBT       A  0022F  SETCIR    A  001E0  SETDTST   A  0000B  SETMAPS   A  001DA
SETMPEN   A  0000F  SETMPI    A  00003  SETMSR    D         SETP1     A  000F3
SETP2     A  000FD  SETP2PAT  A  000FB  SETP3     A  000F7  SETP4     A  000F9
SETPSFF   A  0000D  SETTBT    A  00005  SETTDI    A  00009  SFB       A  00266
SFC00     A  0009C  SFC00.1   A  0009D  SFC00.C   A  0009A  SFC02     A  000B4
SFC04     A  000C2  SFC05     A  000D0  SFC06     A  000DE  SFCXX     A  0009E
```

SYMBOL TABLE

SFS00	A	00097	SFS00.1	A	00098	SFS00.C	A	00095	SFS02	A	000B2
SFS04	A	000C0	SFS05	A	000CE	SFS06	A	000DC	SFSXX	A	00099
SGNXOVR	A	00002	SHIFT	D		SIGN	A	0000F	SIMP0	A	002B1
SINTRQ	A	00004	SKIP	A	00100	SLACK	A	00007	SMAP0	A	002A3
SOC.C	A	000A5	SOC.H	A	000A3	SOS.C	A	000A6	SOS.H	A	000A4
SPFNOP	D		SPHNOP	D		SPMR0	A	00290	SPRD	D	
SPWR	D		SR1CLE	A	00048	SR1CLESL	A	0004B	SR1SL	A	00049
SRAML	A	00007	SRAMQL	A	00006	SRAMQR	A	00004	SRAMR	A	00005
SRG0ALF	A	00045	SRG0ALR	A	00041	SRG0ALS	A	0003A	SRG0ARS	A	0003D
SRG0ELA	A	00044	SRG0ELAD	A	00046	SRG0ERA	A	00043	SRG0ERAD	A	00047
SRG0NOP	A	00039	SRG0RAL	A	0003F	SRG0RAR	A	00040	SRG1	A	
SRG2	A	0000A	SRG2ALF	A	0005A	SRG2ALR	A	00056	SRG2ALS	A	0004F
SRG2ARS	A	00052	SRG2ELA	A	00059	SRG2ELAD	A	0005C	SRG2ERA	A	00058
SRG2ERAD	A	0005D	SRG2NOP	A	0004E	SRG2RAL	A	00054	SRG2RAR	A	00055
SRGSKIP	A	0004D	STBYTE	A	00235	STBYTEOD	A	0023C	STBYTEWR	A	0023A
STC02	A	000AA	STC04	A	000BC	STC05	A	000C8	STC06	A	000D6
STC07	A	000E0	STF00	A	00091	STF02	A	000AC	STF05	A	000CA
STF06	A	000DB	STO	A	000A1	STRD	D		STWMAP	A	002C3
STWMAP5	A	002C9	STX.	A	00215	ST.	E	00022	ST.I	E	00021
SUBR	A	00001	SUBS	A	00002	SWAPE0	D		SWAPFARG	A	0038C
SWMP0	A	002AD	SWRD	A	00006	TAB	A	00011	TBS	A	00220
TWB	D		UCLDMSR	D		ULE	A	0000D	UNPACK	A	003C5
UNPACK1	A	003CA	UNPACK2	A	003CE	VMAFAUL1	A	00329	VMAFAULT	A	00326
VMAFPMAP	A	003ED	VMALOC	A	00004	VMAMAP	A	0030B	VMAMAP01	A	0030F
VMAPTE	A	00002	WORDCNT	A	0000B	WRTIND	A	00109	WRTIND1	A	0010A
X	A	00002	XABXY	A	00212	XC.10	A	0027C	XEXP	A	00007
XJMP0	A	002B9	XL	A	00001	XL.10	A	00278	XOR	E	00026
XORI	E	00025	XS.10	A	00281	XU	A	00000	Y	A	00003
YEXP	A	00006	YL	A	00008	YU	A	00005	YUERR	X	00000
YXERR	X	00000	Z	A	00005	ZA	A	00004	ZB	A	00003
ZL	A	0000A	ZQ	A	00002	ZU	A	00009	.CPM	A	00136
.CPUID	A	000E7	.DAD	A	0010F	.DADE	A	00112	.DCO	A	0012E
.DCOEQ	A	00133	.DCOLT	A	00131	.DCOX	A	00132	.DDE	A	00120
.DDS	A	00125	.DIN	A	0011C	.DINE	A	0011E	.DIS	A	00122
.DNG	A	0011A	.DSB	A	00114	.DSBR	A	00117	.DSKIPZ	A	00128
.ENTC	A	002EA	.ENTN	A	002E6	.ENTN05	A	002E8	.ENTP	A	002E3
.ENTR	A	002E0	.ENTR05	A	002E2	.ENTRL	A	002EE	.ENTRL1	A	002EF
.ENTRSUB	A	002F4	.FAD	A	0037B	.FDV	A	003A4	.FMP	A	00392
.FSB	A	00375	.FWID	A	000E9	.IMAP	A	00323	.IRES	A	00321
.LBP0	A	0030A	.LBPR	A	00300	.LPX0	A	00303	.LPX01	A	00304
.LPXR	A	00307	.PMAP0	A	00319	.SETP	A	000EF	.SIP	A	000ED
.WFI	A	000EB	.XFER	A	001E9	..FCM	A	00370			

OBJECT MODULE - A600 BASESET MICROCODE (06/14/82 *A)

```
00000   11000011XX0010XX XXXXX00000011000 11011111XXXX0000 0000XXXX
00001   01110000XX001X01 01X1011111011110 1000001101110001 10011100
00002   00010001XX001X1X XXXXX10001000000 00011111XXXX0001 00000001
00003   11101000XX000XXX XXXXX10001000000 00011111XXXXXXXX XXXXXXXX
00004   000000000010XXX 0001010000100000 110000001111XXXX XXXXXXXX
00005   00010001XX001X1X XXXXX10001000000 00011111XXXX0001 00000001
00006   11101000XX000XXX XXXXX10001000000 00011111XXXXXXXX XXXXXXXX
00007   00000000XX000XXX XXXXX00000000000 1110000XXXXXXXX XXXXXXXX
00008   00010001XX001X1X XXXXX10001000000 00011111XXXX0001 00000001
00009   11101000XX000XXX XXXXX10001001000 1101111XXXXXXXXX XXXXXXXX
0000A   11100000XX000XXX XX10110000001000 0111000 1XXXXXXXX XXXXXXXX
0000B   00110000XX000X00 XXX0100000000000 0101101001010000 00000000
0000C   00001000XX001XXX 01XXX01111011110 1000001 1XXXXXXXX XXXXXXXX
0000D   00010001XX001X1X XXXXX10001000000 00011111XXXX0001 00000001
0000E   11101000XX000XXX XXXXX10001000000 00011111XXXXXXXX XXXXXXXX
0000F   00000000XX000XXX XXXXX00000000000 11011000XXXXXXXX XXXXXXXX
00010   00010001XX001X1X XXXXX10001000000 00011111XXXX0001 00001000
00011   11100010XX011XXX 0110110001100010 0100011 1XXXXXXXX XXXXXXXX
00012   11100000XX000XXX 11X0100000011110 110010111000XXXX XXXXXXXX
00013   00001000XX001XXX 01XXX01111011110 1000001 1XXXXXXXX XXXXXXXX
00014   00010001XX001X1X XXXXX10001000000 00011111XXXX0001 00001000
00015   11100000XX101XXX XXXXX00000011110 11011010XXXXXXX1 001XXXXX
00016   00001000XX000XXX 01XXX00000011110 1100001 1XXXXXXXX XXXXXXXX
00017   00000000XX000XXX 01XXX01100011110 11000100XXXXXXXX XXXXXXXX
00018   00010001XX001X1X XXXXX10001000000 00011111XXXX0001 00001000
00019   11100010XX011XXX 01XXX01111100010 01001100XXXXXXXX XXXXXXXX
0001A   11100000XX101XXX 01XXX00000011110 11000010XXXXXXX1 001XXXXX
0001B   00001000XX001XXX 01XXX01111011110 1000001 1XXXXXXXX XXXXXXXX
0001C   11100010XX011XXX 01XXX01111100010 01001100XXXXXXXX XXXXXXXX
0001D   11100000XX000XXX 01XXX01100011110 11000100XXXXXXXX XXXXXXXX
0001E   00001000XX001XXX 01XXX01111011110 1000001 1XXXXXXXX XXXXXXXX
0001F   00010001XX001X1X XXXXX10001000000 00011111XXXX0001 00000001
00020   00001000XX000XXX XXXXX10001100000 1101111 1XXXXXXXX XXXXXXXX
00021   00010001XX001X1X XX10110001000000 00011111XXXX0001 00001000
00022   11100010XX011XXX XXXXX10000100010 01011100XXXXXXXX XXXXXXXX
00023   11100000XX001XXX 01XXX00000011110 0100101 1XXXXXXXX XXXXXXXX
00024   00001000XX000XXX XXXXX00000000000 01011010XXXXXXXX XXXXXXXX
00025   00010001XX001X1X XXXXX10001000000 00011111XXXX0001 00000001
00026   11101000XX000XXX XXXXX10001000000 00011111XXXXXXXX XXXXXXXX
00027   00000000XX000XXX XXXXX00000000000 11110000XXXXXXXX XXXXXXXX
00028   00110000XX000X1X 0001000000100000 1110001111110000 00101011
00029   00110000XX000X1X 0001000000100000 1101001111110000 00101011
0002A   11100000XX000XXX 0001010000100000 110100011111XXXX XXXXXXXX
0002B   1100100000010XXX 1001010000100000 1000001111110000 00000000
0002C   01110000XX000X01 XXX1000000000000 0101101000000001 00000000
0002D   00110000X0000X1X 0001000000100000 1110001111110000 00101011
0002E   00110000X0000X1X 0001000000100000 1101001111110000 00101011
0002F   00110000X0000X1X 0001010000100000 1101000111110000 00101011
00030   00110000X0000X1X 0001000000100000 0100001111110000 00101011
```

A600 BASESET MICROCODE (06/14/82 *A)

```
00031  00110000X0000X1X 0001000000100000 1110001110000000 00101011
00032  00110000X0000X1X 0001000000100000 1101001110000000 00101011
00033  00110000X0000X1X 0001010000100000 1101000110000000 00101011
00034  00110000X0000X1X 0001000000100000 0100001110000000 00101011
00035  00110000X0010X1X 0001000000100000 1110001110000000 00101011
00036  00110000X0010X1X 0001000000100000 1101001110000000 00101011
00037  00110000X0010X1X 0001010000100000 1101000110000000 00101011
00038  00110000X0010X1X 0001000000100000 0100001110000000 00101011
00039  00100000XX0000XX XXXXX00000000000 01011010XXXX0000 1001XXXX
0003A  11100000XX000XXX XX01000000100001 110110111111XXXX XXXX0010
0003B  11100000XX000XXX XXXXX00000100001 11011011XXXXXXXX XXXX0010
0003C  00100000XX0000XX XXXXX00000100001 01011011XXXX0000 10010101
0003D  11100000XX000XXX XX01000000100000 010110111111XXXX XXXXXXXX
0003E  00100000XX0000XX XXXXX00000100001 01011011XXXX0000 10010101
0003F  00100000XX0000XX XXXXX00000100001 11011011XXXX0000 10011010
00040  00100000XX0000XX XXXXX00000100001 01011011XXXX0000 10011010
00041  11100110XX000XXX XXXXX00000000000 0001111100111111 11111111
00042  00100000XX0000XX XXXXX10000100001 11100000XXXX0000 10010010
00043  00100000XX0000XX XXXXX00000100001 01011011XXXX0000 10011001
00044  00100000XX0000XX XXXXX00000100001 11011011XXXX0000 10011001
00045  00100101XX0000XX XXXXX10000100000 10011111XXXX0000 1001XXXX
00046  00100000XX0000XX XXXXX10000001001 11011100XXXX0000 10011001
00047  00100000XX0000XX XXXXX10000001001 01011100XXXX0000 10011001
00048  00100000X00000XX XX00000000000000 0101101000110000 1010XXXX
00049  11000000XX0000XX XXXXX00000100000 01011011XXXX0000 1010XXXX
0004A  01110000XX000X01 XXX1000000000000 0101101000000000 01001101
0004B  11000000X00000XX XX00000000100000 0101101100110000 1010XXXX
0004C  01110000XX000X01 XXX1000000000000 0101101000000000 01001101
0004D  00100000XX0010XX 01XXX01111011110 10000011XXXX0000 1010XXXX
0004E  00001000XX000XXX XXXXX00000000000 01011010XXXXXXXX XXXXXXXX
0004F  11100000XX000XXX XX01000000100001 110110111111XXXX XXXX0010
00050  11101000XX000XXX XXXXX00000100001 11011011XXXXXXXX XXXX0010
00051  00000000XX000XXX XXXXX00000100001 01011011XXXXXXXX XXXX0101
00052  11101000XX000XXX XX01000000100000 010110111111XXXX XXXXXXXX
00053  00000000XX000XXX XXXXX00000100001 01011011XXXXXXXX XXXX0101
00054  00001000XX000XXX XXXXX00000100001 11011011XXXXXXXX XXXX1010
00055  00001000XX000XXX XXXXX00000100001 01011011XXXXXXXX XXXX1010
00056  11100110XX000XXX XXXXX00000000000 0001111100111111 11111111
00057  00001000XX000XXX XXXXX10000100001 11100000XXXXXXXX XXXX0010
00058  00001000XX000XXX XXXXX00000100001 01011011XXXXXXXX XXXX1001
00059  00001000XX000XXX XXXXX00000100001 11011011XXXXXXXX XXXX1001
0005A  11101000XX000XXX XXXXX00000000000 01011010XXXXXXXX XXXXXXXX
0005B  00000101XX000XXX XXXXX10000100000 10011111XXXXXXXX XXXXXXXX
0005C  00001000XX000XXX XXXXX10000001001 11011100XXXXXXXX XXXX1001
0005D  00001000XX000XXX XXXXX10000001001 01011100XXXXXXXX XXXX1001
0005E  00110000XX000X1X XXXXX00000000000 01011010XXXX0001 00111101
0005F  00010001XX000X1X XXXXX01111100000 11011100XXXX0001 00001001
00060  00110000XX000X1X XXXXX00000011110 11011010XXXX0001 00000000
00061  11100000XX000XXX XX01000000000010 010110111111XXXX XXXXXXXX
00062  01001000XX00001X XXXXX00000000000 00011011XXXX0000 1000XXXX
00063  10000000XX000XXX XXXXX00000000011 00011011XXXXXXXX XXXX0101
```

```
00064  000000000X000XXX XX00000000000000 110110100011XXXX XXXXXXXX
00065  01001000XX00001X XXXXX00000000000 00011011XXXX0000 1000XXXX
00066  10000000XX000XXX XXXXX00000000011 10011011XXXXXXXX XXXX0110
00067  00000000XX000XXX XXXXX00000000000 1101101OXXXXXXXX XXXXXXXX
00068  01001000XX00001X XXXXX00000000000 00011011XXXX0000 1000XXXX
00069  10000000XX000XXX XXXXX00000000011 00011011XXXXXXXX XXXX0110
0006A  00000000XX000XXX XXXXX00000000000 1101101OXXXXXXXX XXXXXXXX
0006B  01001000XX00001X XXXXX00000000000 00011011XXXX0000 1000XXXX
0006C  10000000XX000XXX XXXXX00000000011 10011011XXXXXXXX XXXX1111
0006D  00000000XX000XXX XXXXX00000000000 1101101OXXXXXXXX XXXXXXXX
0006E  01001000XX00001X XXXXX00000000000 00011011XXXX0000 1000XXXX
0006F  10000000XX000XXX XXXXX00000000011 00011011XXXXXXXX XXXX1111
00070  00000000XX000XXX XXXXX00000000000 1101101OXXXXXXXX XXXXXXXX
00071  00010001XX000X1X XXXXX00000000000 0101101OXXXX0000 11100010
00072  00001000XX000XXX XXXXX00000000000 0101101OXXXXXXXX XXXXXXXX
00073  0000000000000000 0000000000000000 0000000000000000 00000000
00074  0000000000000000 0000000000000000 0000000000000000 00000000
00075  0000000000000000 0000000000000000 0000000000000000 00000000
00076  00010001XX000X1X XXXXX00000000000 0101101OXXXX0001 00001001
00077  11100010XX011XXX XXXXX00000100010 01011100XXXXXXXX XXXXXXXX
00078  11100000XX001XXX 01XXX00000000000 0000001OXXXXXXXX XXXXXXXX
00079  00110010XX011X1X XXXXX00001100010 01011100XXXX0001 00000000
0007A  00110000XX000X1X XXXXX00000000000 0101101OXXXX0001 01000110
0007B  00010001XX000X1X XXXXX00000000000 0101101OXXXX0001 00001001
0007C  11100000XX000XXX XXXXX10001000000 00011111XXXXXXXX XXXXXXXX
0007D  01000000XX000X1X 00XXX00000000011 00100011XXXX0000 00001110
0007E  10000000XX000XXX 00XXX10011000011 00000001XXXXXXXX XXXX1110
0007F  11100000XX000XXX 00XXX10011000011 00001001XXXXXXXX XXXX1110
00080  001100000X000X1X XX00000000000000 1101101000110001 00000000
00081  00110000XX000X1X XXXXX00000000000 0001101OXXXX0001 01100001
00082  00110000XX000X01 XXX1000000000000 0101101001010010 11000001
00083  00010000XX000X1X XXXXX00000000000 0101101OXXXX0001 01100101
00084  00010000XX000X1X XXXXX00000000000 0101101OXXXX0001 01101101
00085  00010000XX000X1X XXXXX00000000000 0101101OXXXX0001 01110100
00086  00110000XX001X1X 01XXX01111011110 10000011XXXX0000 10001100
00087  11100110XX000XXX XXXXX01100011000 1110010100000000 00111111
00088  11000000XX000XXX 0110100000011000 01001011XXXX0000 10001100
00089  00110000XX000X00 XXX0100000000000 0101101001010000 10001011
0008A  00110000XX000X01 XXX1000000000000 0101101001010010 11000001
0008B  00010000XX00001X XXXXX00000000000 0101101OXXXX0000 0100XXXX
0008C  11100000XX101XXX XXXXX00000000000 0101101OXXXXXXX1 001XXXXX
0008D  00001000XX000XXX XXXXX00000000000 0101101OXXXXXXXX XXXXXXXX
0008E  11101110XX101XXX XXXXX00000000000 00011111XXXXXXX0 111XXXXX
0008F  11100110XX100XXX XXXXX00000000000 0010011011111111 11100100
00090  00110000XX000X1X XXXXX00000000000 0101101OXXXX0000 11011000
00091  11101110XX000XXX XXXXX00000000000 00011111XXXXXXXX XXXXXXXX
00092  10100110XX100X1X XXXXX00000000000 0101111000000000 00000001
00093  11101110XX000XXX XXXXX00000000000 00011111XXXXXXXX XXXXXXXX
00094  10100110XX100X1X XXXXX00000000000 0110111000000000 00000001
00095  00010000XX000X1X XXXXX00000000000 0101101OXXXX0000 10010011
00096  00110000XX000X1X XXXXX00000000000 0101101OXXXX0000 10011000
```

```
00097  11101110XX000XXX XXXXX00000000000 00011111XXXXXXXX XXXXXXXX
00098  11100110XX000XXX XX10100000000000 0110011000000000 00000001
00099  01110000XX000X00 XXX0100000000000 0101101001000000 10101110
0009A  00010000XX000X1X XXXXX00000000000 01011010XXXX0000 10010011
0009B  00110000XX000X1X XXXXX00000000000 01011010XXXX0000 10011101
0009C  11101110XX000XXX XXXXX00000000000 00011111XXXXXXXX XXXXXXXX
0009D  11100110XX000XXX XX10100000000000 0110011000000000 00000001
0009E  01110000XX000X00 XXX0100000000000 0101101001010000 10101110
0009F  11100000XX111XXX XXXXX00000100000 01011011XXXXXXXX XXX01XXX
000A0  00110000XX000X1X XXXXX00000000000 01011010XXXX0000 10111000
000A1  000010000X000XXX XX00000000000000 010110100001XXXX XXXXXXXX
000A2  000010000X000XXX XX00000000000000 010110100011XXXX XXXXXXXX
000A3  01110000XX000X00 XXX1000000000000 0101101001100001 00000000
000A4  01110000XX000X00 XXX1000000000000 0101101001110001 00000000
000A5  011100000X000X00 0001000000000000 0100001001100001 00000000
000A6  011100000X000X00 0001000000000000 0100001001110001 00000000
000A7  00001000XX111XXX XXXXX00000100000 01111011XXXXXXXX XXX11XXX
000A8  10101111XX000X1X XXXXX00000100000 11011111XXXXXXXX XXXXX110
000A9  10101111XX000X1X XXXXX10000100000 11011101XXXXXXXX XXXXX110
000AA  00010000XX000X1X XXXXX00000000000 01011010XXXX0001 01100101
000AB  00110000XX001X1X 01XXX01111011110 10000011XXXX0000 10001100
000AC  00011110XX000X1X XXXXX00000000000 00011111XXXX0001 01100101
000AD  11100110XX100XXX XXXXX00000000000 0110111000000000 00000010
000AE  00110000XX001X1X 01XXX01111011110 10000011XXXX0000 10001100
000AF  00011110XX000X1X XXXXX00000000000 00011111XXXX0001 01100101
000B0  11100110XX100XXX XXXXX00000000000 0101111000000000 00000010
000B1  10100000XX001X1X 01XXX01111011110 10000011XXXXXXXX XXXXXXXX
000B2  11100110XX000XXX XXXXX00000000000 0001111100000000 00000010
000B3  00111110XX000X1X XX10100000000000 01100110XXXX0000 10011110
000B4  11100110XX000XXX XXXXX00000000000 0001111100000000 00000010
000B5  00111110XX000X1X XX10100000000000 01100110XXXX0000 10011001
000B6  11101110XX000XXX XXXXX00000000000 00011111XXXXXXXX XXXXXXXX
000B7  11100110XX100XXX XXXXX00000000000 0101111000000000 00000010
000B8  00010000XX000X1X XXXXX00000000000 01011010XXXX0001 01100101
000B9  00010000XX000X1X XXXXX00000000000 01011010XXXX0001 01101101
000BA  00011011XX011X1X XXXXX00000100000 01011011XXXX0001 01110010
000BB  00110000XX001X1X 01XXX01111011110 10000011XXXX0000 10001100
000BC  11101110XX000XXX XXXXX00000000000 00011111XXXXXXXX XXXXXXXX
000BD  10100110XX100X1X XXXXX00000000000 0110111000000000 00001000
000BE  11101110XX000XXX XXXXX00000000000 00011111XXXXXXXX XXXXXXXX
000BF  10100110XX100X1X XXXXX00000000000 0101111000000000 00001000
000C0  11100110XX000XXX XXXXX00000000000 0001111100000000 10000000
000C1  00111110XX000X1X XX10100000000000 01100110XXXX0000 10011110
000C2  11100110XX000XXX XXXXX00000000000 0001111100000000 10000000
000C3  00111110XX000X1X XX10100000000000 01100110XXXX0000 10011001
000C4  00010000XX000X1X XXXXX10000001100 11011100XXXX0001 11100000
000C5  10100000XX000X1X XXXXX00000000000 01011010XXXXXXXX XXXXXXXX
000C6  00010000XX000X1X XXXXX00000000000 01011010XXXX0001 11101101
000C7  10100000XX000X1X XXXXX00110100000 11011100XXXXXXXX XXXXXXXX
000C8  10100000XX101X1X XXXXX00000000000 01011111XXXXXXX1 101XXXXX
000C9  10100000XX101X1X XXXXX00000000000 01011111XXXXXXX1 100XXXXX
```

```
000CA  11101110XX000XXX XXXXX00000000000 00011111XXXXXXXX XXXXXXXX
000CB  10100110XX100X1X XXXXX00000000000 0101111000000000 00000100
000CC  11101110XX000XXX XXXXX00000000000 00011111XXXXXXXX XXXXXXXX
000CD  10100110XX100X1X XXXXX00000000000 0110111000000000 00000100
000CE  11100110XX000XXX XXXXX00000000000 0001111100000000 00000100
000CF  00111110XX000X1X XX10100000000000 01100110XXXX0000 10011001
000D0  11100110XX000XXX XXXXX00000000000 0001111100000000 00000100
000D1  00111110XX000X1X XX10100000000000 01100110XXXX0000 10011110
000D2  11101111XX000XXX XXXXX00000100000 11011111XXXXXXXX XXXXX010
000D3  10100110XX000X1X XXXXX10000100000 1111010111111100 00000000
000D4  11101111XX000XXX XXXXX00000100000 11111111XXXXXXXX XXXXX011
000D5  10100110XX000X1X XXXXX10000100000 1110010100000000 11111111
000D6  11101110XX000XXX XXXXX00000000000 00011111XXXXXXXX XXXXXXXX
000D7  10100110XX100X1X XXXXX00000000000 0101111000000000 00010000
000D8  01001110XX000X1X XXXXX00000000000 00011111XXXX0000 00000011
000D9  10000110XX100XXX XXXXX00000000000 0110111000000000 00010000
000DA  10100000XX101X1X XXXXX00000000000 01011010XXXXXXX0 100XXXXX
000DB  10100000XX101X1X XXXXX00000000000 01011010XXXXXXX0 101XXXXX
000DC  11100110XX000XXX XXXXX00000000000 0001111100000000 01000000
000DD  00111110XX000X1X XX10100000000000 01100110XXXX0000 10011001
000DE  11100110XX000XXX XXXXX00000000000 0001111100000000 01000000
000DF  00111110XX000X1X XX10100000000000 01100110XXXX0000 10011110
000E0  10100000XX101X1X XXXXX00000000000 01011111XXXXXXX1 111XXXXX
000E1  10101111XX000X1X XXXXX00000100000 11011111XXXXXXXX XXXXX100
000E2  00010000XX000X1X XXXXX00000000000 01011010XXXX0001 00001001
000E3  11100000XX001XXX 01XXX00000000000 00000010XXXXXXXX XXXXXXXX
000E4  11100001XX000XXX XXXXX10001000000 11011111XXXXXXXX XXXXXXXX
000E5  11100000XX001XXX 01XXX01111011110 10000011XXXXXXXX XXXXXXXX
000E6  10100000XX000X1X XXXXX10001000010 11011111XXXXXXXX XXXXXXXX
000E7  11100110XX000XXX XXXXX00000000000 1101111100000000 00000010
000E8  00001000XX000XXX XXXXX00000000000 01011010XXXXXXXX XXXXXXXX
000E9  11100110XX000XXX XXXXX00000000000 1101111100001000 00000000
000EA  00001000XX000XXX XXXXX00000000000 01011010XXXXXXXX XXXXXXXX
000EB  00110000XX000X01 XXX1000000000000 0101101001110001 10011100
000EC  00110000XX000X1X XXXXX00000000000 01011010XXXX0000 11101011
000ED  00110000XX000X01 XXX1000000000000 0101101001000001 00000000
000EE  00001000XX000XXX XXXXX00000000000 01011010XXXXXXXX XXXXXXXX
000EF  00010001XX000X1X XX01000000001010 1101110011110001 00001001
000F0  00010000XX000X1X 0010100001000010 11000001XXXX0000 11110011
000F1  11100001XX001XXX 01XXX00000000010 01001011XXXXXXXX XXXXXXXX
000F2  00110001XX000X1X 01XXX10001000000 11000111XXXX0000 11110111
000F3  10100000XX000X00 XXX0100000000011 010110111011XXXX XXXX0000
000F4  00110000XX000X1X XX01010001001010 1101111111110000 11110111
000F5  11100000XX001XXX 01XXX00001000010 10000011XXXXXXXX XXXXXXXX
000F6  00110010XX011X01 01X1000000000000 1000001101110000 11111001
000F7  00110000XX000X00 0101000000001010 1100101101000000 11111101
000F8  00001000XX001XXX 01XXX01111011110 10000011XXXXXXXX XXXXXXXX
000F9  11100110XX000XXX XXXXX00001000010 1101110110000000 00000000
000FA  00110000XX000X1X XXXXX00101000000 11011100XXXX0001 10011100
000FB  00110000XX000X01 01X1000000000000 1100001101110000 11111001
000FC  00110000XX000X1X XXXXX00000000000 01011010XXXX0000 11110111
```

```
000FD   11100000XX001XXX 01XXX00001000010 10000011XXXXXXXX XXXXXXXX
000FE   00110010XX011X1X XXXXX00000000000 01011011XXXX0000 11111011
000FF   00110000XX000X1X XXXXX00000000000 01011010XXXX0001 11001100
00100   00001000XX001XXX 01XXX01111011110 10000011XXXXXXXX XXXXXXXX
00101   10100000XX001X00 01X0100000011110 010010111110XXXX XXXXXXXX
00102   11000001XX001XXX XX10110001000000 00011111XXXX0000 00000011
00103   10100000XX001X00 01X0100000011110 010010111110XXXX XXXXXXXX
00104   10010001XX001XXX XX10110001000000 00011111XXXX0001 00000011
00105   11100000XX101XXX XXXXX00000000000 01011010XXXXXXX1 000XXXXX
00106   10110000XX000X01 XXX1000000000000 0101101001110001 10011100
00107   00110000XX000X1X XX10100000000000 01011010XXXX0001 00000001
00108   10100000XX000X00 XXX0100000000000 010110101110XXXX XXXXXXXX
00109   11000001XX001XXX XX10110001000000 00011111XXXX0000 00000011
0010A   10100000XX000X00 XXX0100000000000 010110101110XXXX XXXXXXXX
0010B   10010001XX001XXX XX10110001000000 00011111XXXX0001 00001010
0010C   10100000XX101X00 XXX0100000000000 010110101110XXX1 000XXXXX
0010D   10110000XX000X01 XXX1000000000000 0101101001110001 10011100
0010E   00110000XX000X1X XXXXX00000000000 01011010XXXX0001 00001001
0010F   00010000XX000X1X XXXXX00000000000 01011010XXXX0001 00111001
00110   11100000XX000XXX 0010100110000010 11000001XXXXXXXX XXXXXXXX
00111   111000000X000XXX 1100100101000000 110000010110XXXX XXXXXXXX
00112   00111000XX001X00 01X0101111011110 1000001110100000 00000000
00113   00000000X0000XXX XX00000000000000 010110100001XXXX XXXXXXXX
00114   00010000XX000X1X XXXXX00000000000 01011010XXXX0001 00111001
00115   11100000XX000XXX 0010100110000010 11001001XXXXXXXX XXXXXXXX
00116   001100000X000X1X 1100100101000000 1100100110000001 00010010
00117   00010000XX000X1X XXXXX00000000000 01011010XXXX0001 00111001
00118   11100000XX000XXX 0110100110000010 110100011000XXXX XXXXXXXX
00119   001100000X000X1X 1100100101000000 1101000110000001 00010010
0011A   11100000XX000XXX XXXXX00000000000 11111011XXXXXXXX XXXXXXXX
0011B   11100000XX000XXX XXXXX00000000010 11111011XXXXXXXX XXXXXXXX
0011C   11100000XX000XXX 0110100000000010 11000011XXXXXXXX XXXXXXXX
0011D   111010000X000XXX 1100100000000000 110000110110XXXX XXXXXXXX
0011E   00110000XX000X00 XXX0100000000000 0101101010100000 00000000
0011F   00000000X0000XXX XX00000000000000 010110100001XXXX XXXXXXXX
00120   11100000XX000XXX 0110100000000010 11001011XXXXXXXX XXXXXXXX
00121   001110000X000X1X 1100100000000000 1100101110000001 00011110
00122   00010000XX000X1X XXXXX00000000000 01011010XXXX0001 00111001
00123   11100000XX000XXX 0110100000001100 11000011XXXXXXXX XXXXXXXX
00124   00110000XX000X1X 11X0100000001010 1100001101100001 00101000
00125   00010000XX000X1X XXXXX00000000000 01011010XXXX0001 00111001
00126   11100000XX000XXX 0110100000001100 110010111000XXXX XXXXXXXX
00127   11100000XX000XXX 11X0100000001010 110010110110XXXX XXXXXXXX
00128   11100010XX011XXX XXXXX00110100010 01011100XXXXXXXX XXXXXXXX
00129   11100000XX001XXX 01XXX00000000000 00001010XXXXXXXX XXXXXXXX
0012A   11100010XX011XXX XXXXX00101100010 01011100XXXXXXXX XXXXXXXX
0012B   11100000XX000XXX XX10100101001100 01011001XXXXXXXX XXXXXXXX
0012C   00110000XX000X00 XXX0100000000000 0101101001000001 00000000
0012D   00110000XX000X1X 01XXX00000011110 11000011XXXX0001 00000000
0012E   00010000XX000X1X XXXXX00000000000 01011010XXXX0001 00111001
0012F   11100000XX000XXX 0110100000001010 01010001XXXXXXXX XXXXXXXX
```

```
00130  00110000XX000X00 01X0100000001010 0101000101010001 00110011
00131  00110000XX000X00 01X0100000011110 1100001100110001 00000000
00132  00110000XX000X1X 01XXX00000011110 11000011XXXX0001 00000000
00133  11000000XX000XXX 0110100001001100 01010001XXXX0001 00110010
00134  00110000XX000X00 01X0100001001100 0101000101010001 00000000
00135  01110000XX000X00 01X0100000011110 1100001110100001 00000000
00136  00010001XX000X1X XXXXX00000000000 01011010XXXX0010 00100101
00137  11000000XX000XXX 0110100100001010 01010001XXXX0001 00110001
00138  01110000XX000X00 01X0100100001010 0101000101010001 00000000
00139  00010001XX000X1X XXXXX00000000000 01011010XXXX0001 00001001
0013A  11100000XX001XXX 01XXX00000000000 0000010XXXXXXXXX XXXXXXXX
0013B  11100001XX000XXX XXXXX10001001010 1101111XXXXXXXXX XXXXXXXX
0013C  10100001XX000X1X XXXXX10001001100 1101111XXXXXXXXX XXXXXXXX
0013D  111000000X000XXX XX00000000000000 000110110011XXXX XXXXXXXX
0013E  1110000000000XXX XX00000000000010 010110110100XXXX XXXXXXXX
0013F  01000000XX00001X XXXXX00000000011 10011011XXXX0000 10000110
00140  00010000XX000X01 XXX1000000000000 0101101000000001 01000101
00141  10000000XX000XX0 XXX1000000000011 100110111111XXXX XXXX0110
00142  00010000XX000X01 XXX1000000000000 0101101000000001 01000101
00143  11101000XX000XXX XXXXX00000000011 00011011XXXXXXXX XXXX0101
00144  0000000000000XXX XX00000000000000 110110100100XXXX XXXXXXXX
00145  10100110XX000X1X XXXXX00000001001 1101111110000000 00000000
00146  00010001XX000X1X XXXXX00000000000 01011010XXXX0001 00001001
00147  111000000X000XXX 0001010001001010 110001111111XXXX XXXXXXXX
00148  11100000XX000XXX XX10100001001100 11011100XXXXXXXX XXXXXXXX
00149  00110000XX000X00 XXX0100000000000 0001101111100001 01001100
0014A  11100000XX000XXX 0110100000000000 00010010XXXXXXXX XXXXXXXX
0014B  11100000XX000XXX 11X0100000000010 110100110110XXXX XXXXXXXX
0014C  00110000XX000X00 XXX1000001001000 1101110001010001 01100000
0014D  11100000XX000XX0 XXX0000000001000 010110111111XXXX XXXXXXXX
0014E  11100000XX000XXX 0010100101101000 11001001XXXXXXXX XXXXXXXX
0014F  00110000XX000X00 XXX0100000000000 0101101011100001 01100000
00150  01000000XX000X10 XXX0000000001001 1001101111110000 00001111
00151  10000000XX000XX0 00X0000101101001 100010011111XXXX XXXX1111
00152  11100000XX000XXX XXXXX00000001001 01011011XXXXXXXX XXXX1111
00153  11100000XX000XXX XX10100000001000 01011011XXXXXXXX XXXXXXXX
00154  00110000XX000X00 XXX0100000001000 0101101111100001 01010111
00155  11100000XX000XX0 XXX0000000001000 010110111111XXXX XXXXXXXX
00156  11100000XX000XXX 00XXX00101101000 11001001XXXXXXXX XXXXXXXX
00157  11100000XX000XXX XX10100000001100 01011011XXXXXXXX XXXXXXXX
00158  00110000XX000X00 XXX0100110001010 1111000111100001 01011011
00159  11100000XX000XXX 01XXX00000001000 11010011XXXXXXXX XXXXXXXX
0015A  11100000XX000XXX XX10100000001010 01011011XXXXXXXX XXXXXXXX
0015B  00110000XX000X00 XXX0100100000010 1101110011110001 01011101
0015C  00110000XX000X1X 0010100000000000 11010010XXXX0001 01011110
0015D  00110000XX000X1X 0110100000000000 11000010XXXX0001 01011110
0015E  00110000XX000X00 XXX0100000001010 0111000101010001 00000000
0015F  00110000XX000X00 XXX0100000000000 0001101111100001 00000000
00160  001100000X000X1X XX00000000000000 1101101000010001 00000000
00161  00110000XX000X01 XXX1000000000000 0101101001010010 11000001
00162  00010000XX000X1X XXXXX00000000000 01011010XXXX0001 01100101
```

```
00163  00110000XX000X01 XXX1000000000000 0101101001110001 10011101
00164  00110000XX000X1X XXXXX00000000000 01011010XXXX0001 01100011
00165  11100110XX001XXX 00XXX01111011110 1100110100000000 00000010
00166  11100000XX000XXX 0110100000011110 01001011XXXXXXXX XXXXXXXX
00167  00010000XX000X00 XXX0100000000000 0101101011010001 01101011
00168  01001001XX000X1X 01XXX00000011110 11000011XXXX0000 00000010
00169  10000000XX000XXX XXXXX00000000000 01011010XXXXXXXX XXXXXXXX
0016A  10100000XX110X1X XXXXX00000011100 01011011XXXXXXXX XXXXXXXX
0016B  11100110XX001XXX XXXXX00000000000 0101111100000000 00000010
0016C  10100110XX110X1X XXXXX00000000000 0101111100000000 00100000
0016D  01000000XX000X1X XXXXX00000000000 01011010XXXX0000 00000010
0016E  10110000XX000X01 XXX1000000000000 0101101001100001 01110001
0016F  10000000XX000XXX XXXXX00000000000 01011010XXXXXXXX XXXXXXXX
00170  00110000XX001X1X 01XXX01111011110 10000011XXXX0000 10001100
00171  11101010XX000XXX XXXXX00000000000 01011010XXXXXXXX XXXXXXXX
00172  00110000XX000X01 XXX1000000000000 0101101001100001 01110010
00173  10100000XX000X1X XXXXX00000000000 00011111XXXXXXXX XXXXXXXX
00174  01000000XX000X1X XXXXX00000001000 11011111XXXX0000 00000011
00175  10000000XX000XXX XXXXX00000001001 01011011XXXXXXXX XXXX0000
00176  11100110XX000XXX XX10100100001000 1110010100000000 00001111
00177  00110000XX000X00 01X0100000001000 1100101101010001 10000110
00178  00110000XX000X00 01X0100000001000 1100101101010001 10000111
00179  00110000XX000X00 01X0100000001000 1100101101010001 10001001
0017A  00110000XX000X00 01X0100000001000 1100101101010001 10001011
0017B  00110000XX000X00 01X0100000001000 1100101101010001 10000110
0017C  00110000XX000X00 01X0100000001000 1100101101010001 10000110
0017D  00110000XX000X00 01X0100000001000 1100101101010001 10001101
0017E  00110000XX000X00 01X0100000001000 1100101101010001 10001111
0017F  00110000XX000X00 01X0100000001000 1100101101010001 10000110
00180  00110000XX000X00 01X0100000001000 1100101101010001 10010000
00181  00110000XX000X00 01X0100000001000 1100101101010001 10010110
00182  00110000XX000X00 01X0100000001000 1100101101010001 10011000
00183  00110000XX000X00 01X0100000001000 1100101101010001 10000110
00184  00110000XX000X00 01X0100000001000 1100101101010001 10000110
00185  00110000XX000X00 01X0100000001000 1100101101010001 10011010
00186  10100000XX000X1X XXXXX00000001000 01011011XXXXXXXX XXXXXXXX
00187  00010000XX000X1X XXXXX00000000000 01011010XXXX0001 01101101
00188  10100000XX000X1X XXXXX00000011110 11011111XXXXXXXX XXXXXXXX
00189  00010000XX000X1X XXXXX00000000000 01011010XXXX0001 01101101
0018A  10100000XX000X1X XXXXX00000000000 11011111XXXXXXXX XXXXXXXX
0018B  00010000XX000X1X XXXXX00000000000 01011010XXXX0001 01101101
0018C  10100000XX000X1X XXXXX00000000010 11011111XXXXXXXX XXXXXXXX
0018D  00010000XX000X1X XXXXX00000000000 01011010XXXX0001 01101101
0018E  10100000XX000X1X XXXXX10000100000 11011101XXXXXXXX XXXXXXXX
0018F  10100000XX000X1X 01XXX00000011110 11000011XXXXXXXX XXXXXXXX
00190  00010000XX000X1X XXXXX00000000000 01011010XXXX0010 11000011
00191  11100000XX101XXX XXXXX00000000000 01011010XXXXXXX1 110XXXXX
00192  11100110XX110XXX XXXXX01110011100 1101110100000000 00100000
00193  11100110XX001XXX XXXXX00000000000 0101111100000000 01000000
00194  11100010XX011XXX XXXXX00000001000 01011011XXXXXXXX XXXXXXXX
00195  10100000XX101X1X XXXXX00000000000 01011010XXXXXXX1 001XXXXX
```

```
00196  00011011XX011X1X XXXXX00000000000 10011011XXXX0001 01110010
00197  10100000XX000X1X XXXXX00000000000 01011010XXXXXXXX XXXXXXXX
00198  00011011XX011X1X XXXXX00001000010 10011011XXXX0001 01110010
00199  10100000XX000X1X XXXXX00000000000 01011010XXXXXXXX XXXXXXXX
0019A  00011011XX011X1X XXXXX01111011110 10011011XXXX0001 01110010
0019B  10100000XX000X1X XXXXX00000000000 01011010XXXXXXXX XXXXXXXX
0019C  11100110XX000XXX 00XXX01111011110 1100110100000000 00000010
0019D  01101100XX000X0X XXXXX00000000000 01011010XXXX0001 11110000
0019E  11100000XX000XXX XXXXX00001000010 11111001XXXXXXXX XXXXXXXX
0019F  01000000XX101X1X 01XXX00001000000 11001100XXXXXXX1 011XXXXX
001A0  11100000XX011XXX XX01000000000001 110110111111XXXX XXXX1010
001A1  11010000XX001X00 XX01000001000010 101111011110XXXX XXXXXXXX
001A2  00110000X0000X00 0101000000000000 1110001101010011 11111111
001A3  11100000XX011XXX XXXXX00000000001 11011011XXXXXXXX XXXX1001
001A4  00110000XX000X1X XXXXX00000000000 10011100XXXX0001 10100011
001A5  11100000XX101XXX XXXXX00000000000 01011010XXXXXXX1 010XXXXX
001A6  11100110XX100XXX XXXXX00000000000 0101111100000000 00100000
001A7  11101101XX111XXX XXXXX00001011010 10100011XXXXXXXX XXX11XXX
001A8  11100110XX000XXX XXXXX00000011110 1101111100100000 00000010
001A9  11100000XX000XXX XXXXX00000001010 11100011XXXXXXXX XXXXXXXX
001AA  01000000XX000X1X XX01000000001000 1110001111110000 00011111
001AB  01000000XX110X00 01X1000100001000 100001000100XXXX XXXXXXXX
001AC  11100110XX001XXX 0010100100001000 1000010100000100 00000000
001AD  11010000XX111X00 01X0100101001010 100000111111XXXX XXX10XXX
001AE  10000110XX000XXX XXXXX00100001000 1110010100000000 00011111
001AF  11100110XX110XXX XXXXX00000011100 1101111100000000 00100000
001B0  00001000XX001XXX 01XXX01111011110 10000011XXXXXXXX XXXXXXXX
001B1  11100110XX110XXX XXXXX00000000000 0101111100000000 00100000
001B2  11100110XX001XXX XXXXX00000000000 0101111100000000 00000010
001B3  11100010XX011XXX XXXXX10001000000 01011111XXXXXXXX XXXXXXXX
001B4  11101000XX000XXX 01XXX00000011110 11000011XXXXXXXX XXXXXXXX
001B5  00001101XX110XXX XXXXX00000011100 01011011XXXXXXXX XXXXXXXX
001B6  11100000XX101XXX XXXXX00000000000 01011010XXXXXXX0 001XXXXX
001B7  00010000XX000X1X XXXXX00000000000 01011010XXXX0001 11011010
001B8  11100110XX001XXX XXXXX00000001100 1101111100000000 00000101
001B9  11101000XX101XXX XXXXX00000000000 01011010XXXXXXX0 010XXXXX
001BA  00011101XX000X1X XXXXX00000000000 01011010XXXX0001 11100000
001BB  11100000XX101XXX XXXXX00000000000 01011010XXXXXXX1 001XXXXX
001BC  11000011XX0010XX XX00000000011000 1101111100010000 0000XXXX
001BD  01110000XX001X00 01X1001100011110 1000001111100001 10111101
001BE  00010000XX000X1X XXXXX00000000000 01011010XXXX0001 11011010
001BF  11100000XX101XXX XXXXX00000000000 01011010XXXXXXX0 100XXXXX
001C0  11100110XX001XXX XXXXX00000001100 1101111100000000 00000110
001C1  00111000XX000X1X XXXXX00000000000 01011010XXXX0001 10111010
001C2  00010000XX000X1X XXXXX00000000000 01011010XXXX0001 11011010
001C3  11100111XX000XXX XXXXX00000000000 01011010XXXXXXXX XXXXXXXX
001C4  11101111XX000XXX XXXXX00000001100 11011111XXXXXXXX XXXXX101
001C5  00110000XX000X1X XXXXX00000000000 01011010XXXX0001 10111010
001C6  11101111XX000XXX XXXXX00000000000 01011010XXXXXXXX XXXXX111
001C7  00010000XX000X1X XXXXX00000000000 01011010XXXX0001 01101101
001C8  00010000XX000X1X XXXXX00000001100 11011111XXXX0001 01110100
```

```
001C9  11100110XX000XXX XX10100110001100 1110010100000001 00000000
001CA  00110000XX000X00 XXX0100000000000 0101101001000001 11000111
001CB  00110000XX001X1X 01XXX01111011110 10000011XXXX0000 10001100
001CC  00110000XX000X01 XXX1000000000000 0101101000110001 10011100
001CD  00011100XX000X1X XXXXX00000000000 01011010XXXX0001 11011010
001CE  11100110XX001XXX XXXXX00000001100 1101111100000000 00001000
001CF  00111000XX000X1X 01XXX00000011110 11001011XXXX0001 10111010
001D0  11100000XX101XXX XXXXX00000000000 01011010XXXXXXX0 010XXXXX
001D1  00010000XX000X1X XXXXX00000000000 01011010XXXX0001 11011010
001D2  11100110XX001XXX XXXXX00000001100 1101111100000000 00000111
001D3  00111000XX000X1X XXXXX00000000000 01011010XXXX0001 10111010
001D4  00010000XX000X1X XXXXX00000000000 01011010XXXX0001 11011010
001D5  11100000XX101XXX XXXXX00000000000 01011010XXXXXXX0 110XXXXX
001D6  00010000XX000X1X XXXXX00000000000 01011010XXXX0000 10111110
001D7  11100110XX001XXX XXXXX00000001100 1101111100000000 00000100
001D8  00111000XX000X1X XXXXX00000000000 01011010XXXX0001 10111010
001D9  0000000000000000 0000000000000000 0000000000000000 00000000
001DA  00010000XX000X1X XXXXX00000000000 01011010XXXX0010 11000011
001DB  11100110XX110XXX XXXXX00000000000 0101111100000000 00100000
001DC  11100110XX001XXX XXXXX00000000000 0101111100000000 00000011
001DD  11100000XX101XXX XXXXX01110011010 11011100XXXXXXX1 110XXXXX
001DE  11100010XX011XXX XXXXX00000001000 01011011XXXXXXXX XXXXXXXX
001DF  10100110XX110X1X XXXXX01110011100 1110010111111111 00100000
001E0  11100110XX000XXX XXXXX00110001100 1110010100000000 00111111
001E1  00010000XX000X1X XXXXX00000001100 01011011XXXX0010 00111101
001E2  11100110XX000XXX XXXXX01110011100 1110010100000000 11111111
001E3  10100000XX000X1X XXXXX00110011100 11011001XXXXXXXX XXXXXXXX
001E4  00010001XX000X1X XXXXX00000000000 01011010XXXX0001 00001001
001E5  00010001XX001X1X XXXXX01111001010 10011010XXXX0001 00001001
001E6  11100000XX000XXX XXXXX00101000010 11011100XXXXXXXX XXXXXXXX
001E7  11100000XX000XXX XXXXX00000000000 11011010XXXXXXXX XXXXXXXX
001E8  11100110XX000XXX 00XXX01111011110 1100010100000000 00000010
001E9  01000000XX00001X XXXXX00000000000 01011010XXXX0000 1011XXXX
001EA  11100001XX001XXX 01XXX00000000000 10000011XXXXXXXX XXXXXXXX
001EB  11100000XX001XXX 0100000001000010 100000110011XXXX XXXXXXXX
001EC  11110010XX011X00 XXX1010001100010 0101111101010010 00110001
001ED  00010000XX000X1X XXXXX01110001100 11011100XXXX0010 00111101
001EE  10100110XX000X1X XXXXX00110001100 1110010100000000 11111111
001EF  0000000000000000 0000000000000000 0000000000000000 00000000
001F0  00110000XX000X1X XXXXX00000000000 01011010XXXX0001 10011110
001F1  00110001XX001X1X XXXXX00000011110 01011011XXXX0001 10110001
001F2  00110000XX000X1X XXXXX00000000000 01011010XXXX0001 10110110
001F3  00110000XX000X1X XXXXX00000000000 01011010XXXX0001 11010000
001F4  00110000XX000X1X XXXXX00000000000 01011010XXXX0001 11000110
001F5  00110000XX000X1X XXXXX00000000000 01011010XXXX0001 11010100
001F6  00110000XX000X1X XXXXX00000000000 01011010XXXX0001 10111110
001F7  00110000XX000X1X XXXXX00000000000 01011010XXXX0001 11000010
001F8  0000000000000000 0000000000000000 0000000000000000 00000000
001F9  0000000000000000 0000000000000000 0000000000000000 00000000
001FA  0000000000000000 0000000000000000 0000000000000000 00000000
001FB  0000000000000000 0000000000000000 0000000000000000 00000000
```

```
001FC  0000000000000000 0000000000000000 0000000000000000 00000000
001FD  0000000000000000 0000000000000000 0000000000000000 00000000
001FE  0000000000000000 0000000000000000 0000000000000000 00000000
001FF  0000000000000000 0000000000000000 0000000000000000 00000000
00200  00010001XX000X1X XXXXX00000000000 01011010XXXX0001 00001001
00201  00110000XX000X1X XXXXX10001100100 11011111XXXX0001 00000000
00202  00010001XX000X1X XXXXX00000000000 01011010XXXX0001 00001001
00203  11100000XX000XXX XXXXX10001000000 00011111XXXXXXXX XXXXXXXX
00204  0011000000010X1X 0001010010100100 1100000011110001 00000000
00205  00001000XX000XXX XXXXX10000100100 11011100XXXXXXXX XXXXXXXX
00206  00001000XX000XXX XXXXX10010100000 11011100XXXXXXXX XXXXXXXX
00207  11001000XX000XXX 0110110010100100 11001100XXXX0000 00000000
00208  01110000XX000X00 XXX0100000000000 0101101001010001 00000000
00209  11001000XX000XXX 0110110010100100 11000100XXXX0000 00000000
0020A  01110000XX000X00 XXX0100000000000 0101101001010001 00000000
0020B  00010001XX000X1X XXXXX00000000000 01011010XXXX0001 00001001
0020C  11100001XX001XXX 00XXX10010000000 01000000XXXXXXXX XXXXXXXX
0020D  11100000XX001XXX 01XXX01111011110 10000011XXXXXXXX XXXXXXXX
0020E  00001000XX000XXX XXXXX10001100000 11011111XXXXXXXX XXXXXXXX
0020F  00010001XX000X1X XXXXX00000000000 01011010XXXX0001 00001001
00210  11100000XX001XXX 00XXX10010000000 01000000XXXXXXXX XXXXXXXX
00211  00110010XX011X1X XXXXX10000100010 01011100XXXX0001 00000000
00212  11100000XX000XXX XXXXX00000100000 00011011XXXXXXXX XXXXXXXX
00213  11101000XX000XXX XXXXX10010100000 11011100XXXXXXXX XXXXXXXX
00214  00000000XX000XXX XXXXX00000100100 11011010XXXXXXXX XXXXXXXX
00215  00010001XX000X1X XXXXX00000000000 01011010XXXX0001 00001001
00216  00110010XX011X1X XXXXX10010100010 01011100XXXX0001 00000000
00217  00010001XX000X1X XXXXX01111000110 11011100XXXX0001 00001001
00218  00110000XX000X1X XXXXX00000011110 11011010XXXX0001 00000000
00219  00010001XX000X1X XXXXX00000000000 01011010XXXX0001 00001001
0021A  00110000XX000X1X 00XXX00011011110 11000000XXXX0001 00000000
0021B  00010001XX000X1X XXXXX00000000000 01011010XXXX0010 00100101
0021C  00110000XX000X1X XXXXX00100001010 11101001XXXX0010 00011111
0021D  00010001XX000X1X XXXXX00000000000 01011010XXXX0010 00100101
0021E  11100000XX000XXX XXXXX00100001010 11011001XXXXXXXX XXXXXXXX
0021F  00110010XX011X1X XXXXX00101100010 01011100XXXX0001 00000000
00220  00010001XX000X1X XXXXX00000000000 01011010XXXX0010 00100101
00221  11100000XX000XXX XXXXX00100001010 11100001XXXXXXXX XXXXXXXX
00222  11100000XX000XXX XX10100100001010 11110001XXXXXXXX XXXXXXXX
00223  00110000XX000X00 XXX0100000000000 0101101001010001 00000000
00224  00110000XX000X1X 01XXX00000011110 11000011XXXX0001 00000000
00225  00010000XX000X1X XXXXX00000000000 01011010XXXX0001 00001001
00226  11100000XX000XXX XXXXX10001001000 11011111XXXXXXXX XXXXXXXX
00227  00010001XX001X1X XXXXX01111011110 10011011XXXX0001 00001001
00228  11100000XX000XXX XXXXX10001001010 11011111XXXXXXXX XXXXXXXX
00229  10100000XX000X1X 01XXX00000011110 11000011XXXXXXXX XXXXXXXX
0022A  11100000XX000XXX XXXXX00001001001 01011100XXXXXXXX XXXX1010
0022B  00010001XX001X1X XX01000100001000 1001101111110010 00110010
0022C  11101000XX001XXX 01XXX00000011110 01001011XXXXXXXX XXXXXXXX
0022D  11100000XX000XXX 01XXX00000000010 11000011XXXXXXXX XXXXXXXX
0022E  00000000XX000XXX XXXXX00110000000 11011100XXXXXXXX XXXXXXXX
```

```
0022F  11100110XX000XXX XXXXX00000001100 1101111100000000 11111111
00230  00010000XX000X1X XXXXX00000001100 11100001XXXX0010 00110101
00231  00001000XX001XXX 01XXX00000011110 01001011XXXXXXXX XXXXXXXX
00232  11100110XX000XXX XXXXX00000001010 1101111100000000 11111111
00233  00010000XX000X00 XXX1010001001100 1101111111100010 00111101
00234  10100000XX000X1X XXXXX00101001100 11100001XXXXXXXX XXXXXXXX
00235  11100000XX000XXX XXXXX00001001001 01011100XXXXXXXX XXXX1010
00236  11100001XX001XXX XX01000100001000 100110111111XXXX XXXXXXXX
00237  11100110XX000XXX XXXXX00000001010 1101111100000000 11111111
00238  00110000XX000X00 XXX1010001000000 0001111111110010 00111100
00239  00010000XX000X1X XXXXX00101000000 00100000XXXX0010 00111101
0023A  11100010XX011XXX XXXXX00110100010 01011000XXXXXXXX XXXXXXXX
0023B  10100000XX000X1X 01XXX00000000010 11000011XXXXXXXX XXXXXXXX
0023C  00110000XX000X1X XXXXX00101000000 00101000XXXX0010 00111010
0023D  11100101XX000XXX XXXXX00110001100 10011111XXXXXXXX XXXXXXXX
0023E  10100101XX000X1X XXXXX00110001100 10011111XXXXXXXX XXXXXXXX
0023F  00010001XX000X1X XXXXX00000000000 01011010XXXX0010 01101111
00240  11000001XX001XXX 01XXX00000000000 10000011XXXX0010 01000000
00241  11100000XX001XXX 01XXX00001000010 10000011XXXXXXXX XXXXXXXX
00242  11100010XX011XXX XXXXX10001100010 01011111XXXXXXXX XXXXXXXX
00243  00110000XX000X01 01X1000000011000 1100101101110010 01110101
00244  01110000XX000X00 XXX0100000000000 0101101001010001 00000000
00245  00010001XX000X1X XXXXX00000000000 01011010XXXX0010 01101111
00246  11000001XX001XXX 01XXX00000000000 10000011XXXX0010 01000110
00247  11100000XX001XXX 01XXX00001000010 10000011XXXXXXXX XXXXXXXX
00248  11100001XX000XXX XXXXX10001001010 11011111XXXXXXXX XXXXXXXX
00249  11100001XX000XXX XXXXX10001001100 11011111XXXXXXXX XXXXXXXX
0024A  11100000XX000XXX 0010100110001010 01001001XXXXXXXX XXXXXXXX
0024B  00110000XX000X00 00X0100110001010 0100100101000010 01001110
0024C  00110000XX000X01 01X1000000011000 1100101101110010 01110101
0024D  01110000XX000X00 XXX0100000000000 0101101001010001 00000000
0024E  00110000XX000X00 01X0100000011110 1100001100110010 01010000
0024F  11100000XX000XXX 01XXX00000011110 11000011XXXXXXXX XXXXXXXX
00250  11100000XX000XXX 00XXX01100000010 11000001XXXXXXXX XXXXXXXX
00251  11100000XX000XXX 01XXX00000000010 11001011XXXXXXXX XXXXXXXX
00252  00110000XX000X1X 01XXX00000000000 11001011XXXX0001 00000000
00253  00010001XX000X1X XXXXX00000000000 01011010XXXX0010 01101111
00254  11000000XX000XXX XXXXX00000000000 01011010XXXX0010 01010101
00255  11100000XX000XXX XXXXX00000001001 01011100XXXXXXXX XXXX1010
00256  00010001XX001X1X XX01000100001000 1001101111110010 00110010
00257  00010000XX000X1X 01XXX00000000000 11000011XXXX0010 00110101
00258  00110000XX000X01 01X1000000011000 1100101101110010 01110101
00259  01110000XX000X00 XXX0100000000000 0101101001010001 00000000
0025A  00010001XX000X1X XXXXX00000000000 01011010XXXX0010 01101111
0025B  11100000XX000XXX XXXXX00000001001 01011100XXXXXXXX XXXX1010
0025C  00010001XX001X1X XX01000000001000 0101101111110010 00110010
0025D  11100000XX000XXX 01XXX00000000000 11000011XXXXXXXX XXXXXXXX
0025E  11000000XX000XXX XXXXX00110001110 11011100XXXX0010 01011011
0025F  11100000XX000XXX XXXXX00001001001 01011100XXXXXXXX XXXX1010
00260  00010001XX001X1X XX01000000001000 0101101111110010 00110010
00261  11100000XX000XXX 01XXX00000000010 11000011XXXXXXXX XXXXXXXX
```

```
00262  11100000XX000XXX 0010100110001110 01001001XXXXXXXX XXXXXXXX
00263  00110000XX000X00 00X0100110001110 0100100101000010 01001110
00264  00110000XX000X01 01X1000000011000 1100101101110010 01110101
00265  01110000XX000X00 XXX0100000000000 0101101001010001 00000000
00266  1110011OXX000XXX XXXXX00000001110 1110010100000000 11111111
00267  00010000XX000X1X XXXXX00000001100 11011100XXXX0010 00111101
00268  11100110XX000XXX XXXXX00110010000 1110010100000000 11111111
00269  11100000XX000XXX XXXXX00001001001 01011100XXXXXXXX XXXX1010
0026A  00010001XX001X1X XX01000100001000 1001101111110010 00110010
0026B  11000000XX000XXX XX01000110001110 0111000111110010 01101001
0026C  00110000XX000X00 XX01000110010000 0111000101010010 00110001
0026D  00110000XX000X00 01X1000000000010 1100001101010001 00000000
0026E  01110000XX000X01 XXX1000000000000 0101101001110001 10011100
0026F  00010000XX000X1X XXXXX00000000000 01011010XXXX0001 00001001
00270  11100000XX000XXX XX01010001011000 110111111111XXXX XXXXXXXX
00271  00110001XX001X00 01X1001111011110 1000001101010001 00000000
00272  11100001XX000XXX XX01010001000000 000111111111XXXX XXXXXXXX
00273  10100010XX011X00 XXX1000000000000 011000100101XXXX XXXXXXXX
00274  10100000XX000X1X XXXXX00000011000 11011010XXXXXXXX XXXXXXXX
00275  00110000XX000X00 XXX0100000000000 0101101001010001 00000000
00276  11100000XX001XXX 01XXX00000011110 11001011XXXXXXXX XXXXXXXX
00277  00110010XX011X1X XXXXX01100100010 01011100XXXX0001 10011100
00278  00010001XX000X1X XXXXX00000000000 01011010XXXX0001 00001001
00279  11100001XX110XXX XXXXX00000011010 01011011XXXXXXXX XXXXXXXX
0027A  11100000XX110XXX XXXXX00000011100 01011011XXXXXXXX XXXXXXXX
0027B  00110000XX000X1X XXXXX00000100000 11011111XXXX0001 00000000
0027C  00010001XX000X1X XXXXX00000000000 01011010XXXX0001 00001001
0027D  11100001XX110XXX XXXXX00000011010 01011011XXXXXXXX XXXXXXXX
0027E  11100000XX110XXX XXXXX00000011100 01011011XXXXXXXX XXXXXXXX
0027F  11100000XX001XXX 01XXX01111011110 10000011XXXXXXXX XXXXXXXX
00280  00111000XX000X1X XXXXX00000001000 11011111XXXX0000 00001010
00281  00010001XX000X1X XXXXX00000000000 01011011XXXX0001 00001001
00282  11100000XX110XXX XXXXX00000011010 01011011XXXXXXXX XXXXXXXX
00283  11100010XX011XXX XXXXX00000100000 01011011XXXXXXXX XXXXXXXX
00284  00110000XX110X1X XXXXX00000011100 01011011XXXX0001 00000000
00285  00110000XX000X01 XXX1000000000000 0101101001010010 11000001
00286  00010001XX000X1X XXXXX00000000000 01011010XXXX0001 00001001
00287  11100000XX001XXX 01XXX01111011110 10000011XXXXXXXX XXXXXXXX
00288  11101000XX000XXX XXXXX10001001000 11011111XXXXXXXX XXXXXXXX
00289  00000110XX000XXX XXXXX00100011010 1110010100000000 11111111
0028A  00110000XX000X01 XXX1000000000000 0101101001010010 11000001
0028B  00010000XX000X1X 01XXX00000000000 11000011XXXX0010 10010100
0028C  11100110XX000XXX XXXXX00000001000 1101111101111111 11111111
0028D  11100000XX111XXX XXXXX00100000010 01100001XXXXXXXX XXX10XXX
0028E  11100000XX110XXX XXXXX00000011100 01011011XXXXXXXX XXXXXXXX
0028F  00001000XX001XXX 01XXX00000011110 01001011XXXXXXXX XXXXXXXX
00290  00110000XX000X01 XXX1000000000000 0101101001010010 11000001
00291  00010000XX000X1X 01XXX00000000000 11000011XXXX0010 10010100
00292  11101111XX000XXX XXXXX00000000010 11011111XXXXXXXX XXXXX001
00293  00110000XX000X1X XXXXX00000000000 01011011XXXX0010 10001110
00294  01000000XX000X1X 01XXX00000001000 11001100XXXX0000 00000100
```

```
00295  10000000XX000XXX XXXXX00000001001 01011011XXXXXXXX XXXX1010
00296  11100000XX110XXX XXXXX00000001001 01011011XXXXXXXX XXXX1010
00297  10100100XX001X1X XXXXX00000001000 01011011XXXXXXXX XXXXXXXX
00298  00110000XX000X01 XXX1000000000000 0101101001010010 11000001
00299  00010001XX000X1X XXXXX00000000000 01011010XXXX0010 00100101
0029A  11100000XX000XXX 01XXX00000001010 11000010XXXXXXXX XXXXXXXX
0029B  11100110XX000XXX XXXXX00000001100 1101111101111111 11111111
0029C  01000000XX000X1X XXXXX00000010000 11100011XXXX0000 00011111
0029D  11100000XX110XXX XXXXX00100001000 10011011XXXXXXXX XXXXXXXX
0029E  11100110XX001XXX 00XXX01000010000 1000010100000100 00000000
0029F  11100000XX111XXX XXXXX00110000000 01100101XXXXXXXX XXX10XXX
002A0  11100000XX110XXX XXXXX01110011100 10011011XXXXXXXX XXXXXXXX
002A1  10000001XX001XXX 01XXX00101001010 10000011XXXXXXXX XXXXXXXX
002A2  00001000XX001XXX 01XXX01111011110 10000011XXXXXXXX XXXXXXXX
002A3  00110000XX000X01 XXX1000000000000 0101101001010010 11000001
002A4  00010001XX000X1X XXXXX00000000000 01011010XXXX0010 00100101
002A5  11100000XX000XXX XXXXX00000001010 11011010XXXXXXXX XXXXXXXX
002A6  01000000XX000X1X XXXXX00000010000 11100011XXXX0000 00011111
002A7  11100110XX001XXX 00XXX01000010000 1000010100000100 00000000
002A8  11100100XX110XXX XXXXX00100001000 10011011XXXXXXXX XXXXXXXX
002A9  11101111XX011XXX XXXXX00000000000 01011111XXXXXXXX XXXXX001
002AA  11100000XX001XXX 01XXX00101001010 10000100XXXXXXXX XXXXXXXX
002AB  10000010XX110XXX XXXXX01110011100 10011011XXXXXXXX XXXXXXXX
002AC  00001000XX001XXX 01XXX01111011110 10000011XXXXXXXX XXXXXXXX
002AD  00110000XX000X01 XXX1000000000000 0101101001010010 11000001
002AE  00010001XX000X1X XXXXX00000000000 01011010XXXX0001 00001001
002AF  00010000XX000X1X XXXXX00000000000 01011010XXXX0010 11000011
002B0  00110010XX011X1X XXXXX00100100010 01011100XXXX0001 00000000
002B1  00110000XX000X01 XXX1000000000000 0101101001010010 11000001
002B2  00010001XX000X1X XXXXX00000000000 01011010XXXX0001 00001001
002B3  11100110XX110XXX XXXXX00000000000 0101111100000000 00100000
002B4  11100110XX001XXX XXXXX00000000000 0101111100000000 00000011
002B5  11100001XX000XXX XXXXX00000000000 01011010XXXXXXXX XXXXXXXX
002B6  11100000XX110XXX XXXXX00000011100 01011011XXXXXXXX XXXXXXXX
002B7  11100000XX001XXX XXXXX00000000000 01011010XXXXXXXX XXXXXXXX
002B8  00110010XX011X1X XXXXX00000100010 01011111XXXX0001 00000000
002B9  00110000XX000X01 XXX1000000000000 0101101001010010 11000001
002BA  00010001XX000X1X XXXXX00000000000 01011010XXXX0010 00100101
002BB  11100110XX110XXX XXXXX00100011100 1110010100000000 00011111
002BC  01000000XX000X1X XX01000000001000 0101101111110000 00000100
002BD  10000000XX000XXX XXXXX00000001001 01011011XXXXXXXX XXXX0000
002BE  11100110XX000XXX XXXXX00100011010 1110010100000000 00011111
002BF  00010000XX000X00 XXX1000000000000 0101101011110000 11100000
002C0  00110000XX000X1X XXXXX00000011110 11011010XXXX0000 10101110
002C1  11100000XX101XXX XXXXX00000000000 01011010XXXXXXX0 011XXXXX
002C2  00000000XX101XXX XXXXX00000000000 01011010XXXXXXX1 110XXXXX
002C3  11100110XX000XXX XXXXX01110001000 1110010100000000 00011111
002C4  11100101XX000XXX XXXXX01101001010 10011111XXXXXXXX XXXXXXXX
002C5  11100000XX000XXX XXXXX00000001011 11011011XXXXXXXX XXXX0010
002C6  11100110XX000XXX XXXXX00101001010 1110010100000011 11100000
002C7  00110000XX000X01 XXX1000101001000 1101100101010010 11001001
```

```
002C8  10100000XX000X1X XXXXX00000001000 01011011XXXXXXXX XXXXXXXX
002C9  10100110XX000X1X XXXXX00100001000 1101110110000000 00000000
002CA  11000000XX000XXX XXXXX01110001110 11011100XXXX0010 11011001
002CB  01110000XX000X01 XXX1001110010000 1101110000100010 11010010
002CC  11000000XX000XXX XXXXX01110001110 11011100XXXX0010 11011001
002CD  01110000XX000X01 XXX1001101010000 1101110000100010 11010010
002CE  11000000XX000XXX XXXXX01101001110 11011100XXXX0010 11011001
002CF  01110000XX000X01 XXX1001110010000 1101110000100010 11010010
002D0  11000000XX000XXX XXXXX01101001110 11011100XXXX0010 11011001
002D1  01110000XX000X01 XXX1001101010000 1101110000100010 11010010
002D2  11000000XX000XXX XX10100000000100 01011011XXXX0010 11010011
002D3  00110000XX110X00 XXX0100000001110 0101101101010010 10001110
002D4  11100001XX001XXX 01XXX00000000000 10000011XXXXXXXX XXXXXXXX
002D5  11100000XX110XXX XXXXX00000010000 01011011XXXXXXXX XXXXXXXX
002D6  11100000XX001XXX 01XXX00001000010 10000011XXXXXXXX XXXXXXXX
002D7  11100010XX011XXX XXXXX00000000000 01011111XXXXXXXX XXXXXXXX
002D8  01110000XX000X01 01X1000000000100 1100101101110001 10011100
002D9  11000000XX000XXX XX10100000000100 01011011XXXX0010 11011010
002DA  00110000XX110X00 XXX0100000001110 0101101101010010 10001110
002DB  11100000XX000XXX XXXXX00000001001 01011100XXXXXXXX XXXX1010
002DC  00010001XX001X1X XX01000100001000 1001101111110010 00110010
002DD  11100000XX110XXX XXXXX00000010000 01011011XXXXXXXX XXXXXXXX
002DE  00010000XX000X1X 01XXX00000000000 11000011XXXX0010 00110101
002DF  01110000XX000X01 01X1000000000100 1100101101110001 10011100
002E0  11100110XX000XXX XXXXX00000001000 1101111111111111 11111101
002E1  00010001XX000X1X XXXXX00000000000 01011010XXXX0010 11110100
002E2  00110000XX000X1X 0101000111010000 1100100111110010 11101110
002E3  11100110XX000XXX XXXXX00000001000 1101111111111111 11111011
002E4  00010001XX000X1X XXXXX00000000000 01011010XXXX0010 11110100
002E5  00110000XX000X1X XXXXX01000000000 11011100XXXX0010 11100010
002E6  11100110XX000XXX XXXXX00000001000 1101111111111111 11111101
002E7  00010001XX000X1X XXXXX00000000000 01011010XXXX0010 11110100
002E8  11100000XX000XXX XX01000110010000 110111001111XXXX XXXXXXXX
002E9  00110010XX011X1X 00XXX00111001100 01000001XXXX0010 11101111
002EA  11100110XX000XXX XXXXX00000001000 1101111111111111 11111011
002EB  00010001XX000X1X XXXXX00000000000 01011010XXXX0010 11110100
002EC  11100000XX000XXX XXXXX00111000000 11011100XXXXXXXX XXXXXXXX
002ED  00110000XX000X1X 00XXX00110000000 11000001XXXX0010 11101000
002EE  11100000XX000XXX 01XXX00000001110 11000011XXXXXXXX XXXXXXXX
002EF  00110001XX001X00 01X1000111001110 1000001101010001 00000000
002F0  00010000XX000X1X 0101000000001100 1100101111110001 00001001
002F1  00110000XX001X00 01X1000101001010 1000001111110001 00000000
002F2  11100010XX011XXX XXXXX00000000000 01011010XXXXXXXX XXXXXXXX
002F3  00110000XX000X1X 0101000000010000 1100101111110010 11101111
002F4  11100000XX001XXX 00XXX01111001000 11000001XXXXXXXX XXXXXXXX
002F5  11100001XX000XXX XXXXX10001001010 11011111XXXXXXXX XXXXXXXX
002F6  11100000XX000XXX XXXXX00101001100 11011100XXXXXXXX XXXXXXXX
002F7  11100001XX001XXX XXXXX10001001110 11011111XXXXXXXX XXXXXXXX
002F8  11100000XX001XXX 01XXX00100001100 10010001XXXXXXXX XXXXXXXX
002F9  10100010XX011X1X XXXXX10001010000 11011111XXXXXXXX XXXXXXXX
002FA  11100000XX000XXX XXXXX00000001100 11011100XXXXXXXX XXXXXXXX
```

```
002FB   10100000XX000X1X XXXXX00001001110 11011100XXXXXXXX XXXXXXXX
002FC   0000000000000000 0000000000000000 0000000000000000 00000000
002FD   0000000000000000 0000000000000000 0000000000000000 00000000
002FE   0000000000000000 0000000000000000 0000000000000000 00000000
002FF   0000000000000000 0000000000000000 0000000000000000 00000000
00300   00010000XX000X1X XXXXX00000000000 01011010XXXX0010 11111010
00301   00010001XX000X1X XXXXX01111010100 11011100XXXX0000 11100010
00302   00110000XX000X1X XXXXX00000000000 01011010XXXX0011 00001011
00303   00010000XX000X1X XXXXX01111010100 11011100XXXX0010 11111010
00304   00010001XX000X1X XXXXX00000000000 01011010XXXX0001 00001001
00305   00010000XX000X1X XXXXX00000000000 01011010XXXX0011 01011101
00306   00110000XX000X1X 01XXX00000011110 11000011XXXX0011 00001011
00307   00010000XX000X1X XXXXX00000000000 01011010XXXX0010 11111010
00308   00010001XX000X1X XXXXX01111010100 11011100XXXX0000 11100010
00309   00110000XX000X1X XXXXX00000000000 01011010XXXX0011 00000100
0030A   00010000XX000X1X XXXXX01111010100 11011100XXXX0010 11111010
0030B   11100000XX000XXX XX10100000000000 01011011XXXXXXXX XXXXXXXX
0030C   00110000XX000X00 XXX0100000000000 0101101011100011 00001111
0030D   00010001XX001X1X XX10100000000010 00011011XXXX0001 00000001
0030E   00001000XX000XXX XXXXX00000000010 11011010XXXXXXXX XXXXXXXX
0030F   11100110XX000XXX XXXXX00000000110 1101111101111000 00000000
00310   000100000X000X1X XX00000000000010 0001101100110011 00101111
00311   01000000XX000X1X XXXXX00000000100 11011100XXXX0000 00000101
00312   10000000XX000XXX XXXXX00000000101 10011011XXXXXXXX XXXX0110
00313   00010000XX000X1X XXXXX00000000000 01011010XXXX0011 00110101
00314   11100110XX001XXX XXXXX00000000110 1101111101111100 00000000
00315   00010000XX000X1X 01XXX00000000100 11000011XXXX0011 00110101
00316   11100110XX000XXX XXXXX00001000010 1110110111111100 00000000
00317   11100110XX000XXX XXXXX00001000010 1101110101111000 00000000
00318   00001000XX001XXX 01XXX00000011110 01001011XXXXXXXX XXXXXXXX
00319   11100110XX000XXX XXXXX00000000110 1101111101111100 00000000
0031A   000100000X000X1X XX00001111010100 1101110000010011 00101111
0031B   00010000XX000X1X XXXXX00001000100 11011100XXXX0011 00110101
0031C   01000000XX000X1X XXXXX00000001110 11011100XXXX0000 00000101
0031D   10000000XX000XXX XXXXX00000001111 01011011XXXXXXXX XXXX1010
0031E   00010000XX001X1X XXXXX00000001110 01011011XXXX0011 01000000
0031F   11100000XX000XXX 01XXX00000000010 11000011XXXXXXXX XXXXXXXX
00320   00110000XX000X1X 01XXX00000000000 11000011XXXX0001 00000000
00321   00010000XX000X1X XXXXX01111010100 11011100XXXX0011 01000011
00322   00001000XX001XXX 01XXX01000011110 10000100XXXXXXXX XXXXXXXX
00323   00010000XX000X1X XXXXX00000000000 01011010XXXX0010 11111010
00324   00010000XX000X1X XXXXX01111010100 11011100XXXX0011 01000011
00325   00110000XX000X1X 01XXX01000011110 11000100XXXX0011 00001011
00326   00110000XX000X00 XXX1001010011110 1101110001110011 11101101
00327   11100000XX000XXX XXXXX00110000000 11011100XXXXXXXX XXXXXXXX
00328   11100000XX000XXX XXXXX00111000010 11011100XXXXXXXX XXXXXXXX
00329   11100110XX001XXX XXXXX00000000000 0101111100000000 00000100
0032A   11100001XX000XXX 01XXX00000011110 11001011XXXXXXXX XXXXXXXX
0032B   00110001XX001X1X XXXXX00000000000 00011111XXXX0000 00011001
0032C   0000000000000000 0000000000000000 0000000000000000 00000000
0032D   0000000000000000 0000000000000000 0000000000000000 00000000
```

```
0032E  0000000000000000 0000000000000000 0000000000000000 00000000
0032F  11100110XX001XXX XXXXX00000000000 0101111100000000 00000010
00330  11100001XX110XXX XXXXX00000000000 01100010XXXXXXXX XXXXXXXX
00331  11100110XX110XXX XXXXX01110001010 1001111110000000 00000000
00332  11100000XX001XXX XX10100011011000 10011111XXXXXXXX XXXXXXXX
00333  10100000XX111X00 XXX0100101011000 111010011110XXXX XXX10XXX
00334  00110000XX000X1X XXXXX00101000110 11011001XXXX0011 00100110
00335  11100110XX000XXX XXXXX00000010110 1101111111111100 00000000
00336  11100000X0111XXX XX00000000011000 010110110011XXXX XXX10XXX
00337  11100110XX000XXX XXXXX00010001010 1110110111111100 00000000
00338  11100001XX001XXX XXXXX00011001010 01011001XXXXXXXX XXXXXXXX
00339  11100110XX000XXX XXXXX00010001010 1110010111111100 00000000
0033A  11100000XX000XXX XXXXX00000000000 00011111XXXXXXXX XXXXXXXX
0033B  11100000XX000XXX XX10101011000000 01110000XXXXXXXX XXXXXXXX
0033C  00110000XX000X00 XXX0101011010010 1110000001010011 00100110
0033D  11100000XX000XXX XX10100101010010 01110001XXXXXXXX XXXXXXXX
0033E  00110000XX000X00 XXX0101011010000 1110100001000011 00100110
0033F  00110000XX000X00 00X0101100010000 1100000101010011 01000001
00340  10100000XX111X1X XXXXX00000010000 01011011XXXXXXXX XXX10XXX
00341  11100110XX000XXX XXXXX00000010000 1101111101111111 11111111
00342  10100000X0111X1X XX00000000010000 010110110001XXXX XXX10XXX
00343  00010001XX000X1X XXXXX01111010000 11011100XXXX0001 00001001
00344  11100000XX000XXX 01XXX00000011000 1100010XXXXXXXXX XXXXXXXX
00345  11100000XX000XXX XX01000000010110 110111111111XXXX XXXXXXXX
00346  11100000XX000XXX XXXXX00000000010 1110001XXXXXXXXX XXXXXXXX
00347  11100000XX000XXX XXXXX00000000000 1110001XXXXXXXXX XXXXXXXX
00348  00110000XX000X00 XXX1000000000000 0101101001010011 01011100
00349  00010001XX001X1X 01XXX01000010000 10000011XXXX0001 00001001
0034A  11100000XX000XXX XX01000000000000 000111111111XXXX XXXXXXXX
0034B  00110001XX001X00 01X1001100011000 1000001111100011 01001101
0034C  11100000XX000XXX 01XXX00000000000 11001011XXXXXXXX XXXXXXXX
0034D  11100000XX000XXX 0010100001000010 11000000XXXXXXXX XXXXXXXX
0034E  11100000XX000XXX 11X0100000000000 110000110110XXXX XXXXXXXX
0034F  11000000XX000XXX XX01000000000000 0001111111110000 00001111
00350  00110000XX000X00 XXX1000001001000 1101110001010011 01011001
00351  01000000XX000X00 XXX1000000001011 001000110101XXXX XXXX0110
00352  10000000XX000XXX 00XXX10111001011 00000001XXXXXXXX XXXX1011
00353  11100000XX000XXX XXXXX00000000010 11011010XXXXXXXX XXXXXXXX
00354  11000000XX000XXX XXXXX00000000000 00011111XXXX0000 00001111
00355  01000000XX000X00 XXX1000000010011 000110110101XXXX XXXX0110
00356  10000000XX000XXX 00XXX10011001011 00000001XXXXXXXX XXXX1011
00357  11100000XX000XXX XXXXX00000000000 11011010XXXXXXXX XXXXXXXX
00358  00110000XX000X1X 0101000000010110 1100101111110011 01001000
00359  11100000XX000XXX 0101000000010110 110010111111XXXX XXXXXXXX
0035A  11000000XX000XXX XXXXX00001000000 11011100XXXX0011 01001000
0035B  01110000XX000X00 XXX1000000000010 1110001101010011 01000111
0035C  00110001XX001X1X XXXXX00000011000 00011011XXXX0011 01011101
0035D  11100000XX001XXX 01XXX00000000000 00000010XXXXXXXX XXXXXXXX
0035E  11100001XX000XXX 00XXX00000000000 11000101XXXXXXXX XXXXXXXX
0035F  11100001XX000XXX 0010100001000010 11000101XXXXXXXX XXXXXXXX
00360  10100000XX000X1X 11X0100000000000 110000110110XXXX XXXXXXXX
```

```
00361  0001100000000X1X XX00000001001110 1101110001000011 11001010
00362  11100110XX000XXX XXXXX00000001100 1101111100000000 00001111
00363  00110000XX000X00 0101000110001110 1101000111110011 01101100
00364  00110000XX000X00 XXX1000000000000 0010001011100011 01100111
00365  00000000X000XXX XX00000000000001 01111001000 1XXXX XXXX0000
00366  11100000XX000XXX 00XXX00000000001 0000011XXXXXXXXX XXXX1110
00367  00110000XX000X00 0101000000001110 1100101101000011 01100110
00368  11100000XX000XX0 XXX0100000000000 0101101 1XXXXXXXX XXXXXXXX
00369  00110000XX000X00 XXX0100001000000 0101100011100000 00000000
0036A  00110000XX000X00 XXX0100000000000 0101101001010000 00000000
0036B  00000000XX000XXX 01XXX00000000000 1100001 1XXXXXXXX XXXXXXXX
0036C  00000000XX000XXX XXXXX00000000000 1111000 1XXXXXXXX XXXXXXXX
0036D  11100110XX000XXX XXXXX00000001110 1101111100000000 00001111
0036E  11100000XX000XXX XXXXX00001000010 1111000 1XXXXXXXX XXXXXXXX
0036F  00110000XX000X1X 01XXX00000011110 1100101 1XXXX0011 11001111
00370  11100000XX000XXX XXXXX00000000000 0101101 1XXXXXXXX XXXXXXXX
00371  00110000XX000X00 01X0100000011110 1100101101010001 00000000
00372  0001000000000X1X XX00000001001110 1101110001000011 11001010
00373  11100000XX000XXX 01XXX00000000010 1101001 1XXXXXXXX XXXXXXXX
00374  00110000XX000X1X 11X0100000000000 1101001101100011 11011100
00375  00010000XX000X1X XXXXX00000000000 0101101 0XXXX0011 11000101
00376  11100000XX000XXX 0110100000010000 1101001 1XXXXXXXX XXXXXXXX
00377  11100000XX000XXX 11X0100000001010 110100110110XXXX XXXXXXXX
00378  00110000XX000X00 XXX0100000000000 0101101001100011 01111100
00379  11100000XX000XXX XXXXX00000001011 0101101 1XXXXXXXX XXXX0000
0037A  00110000XX000X1X 01XXX00000001100 1100001 1XXXX0011 01111100
0037B  00010000XX000X1X XXXXX00000000000 0101101 0XXXX0011 11000101
0037C  11100000XX000XXX XX01000000010010 110111001111XXXX XXXXXXXX
0037D  00010000XX000X00 XX01000000001010 0101101101010011 10001100
0037E  00110000XX000X00 0001000110001110 0000100101010011 11001111
0037F  00010000XX000X00 XXX1000001010100 1101110011110011 10001100
00380  11000000XX000XXX XX01000000001100 1101101011110011 10000100
00381  11100110XX000XXX 0010100110000000 0100010111111111 11100111
00382  01110000XX000X00 XXX0100000010000 0001101111100011 11001111
00383  11100000XX000XXX 00XXX00000001011 0000011XXXXXXXXX XXXX1110
00384  00110000XX000X00 0101000000001100 1100101101000011 10000011
00385  11100000XX000XXX 0010100001000000 0000000XXXXXXXXX XXXXXXXX
00386  11100000XX000XXX 11X0100101000000 110000010110XXXX XXXXXXXX
00387  00110000XX000X00 XXX0100000000000 0101101001100011 10001011
00388  11100000XX000XXX 00XXX00000000001 0000011XXXXXXXXX XXXX1110
00389  11100110XX000XXX XXXXX00000000000 1111010110000000 00000000
0038A  11100000XX000XXX 01XXX00000001110 1100001 1XXXXXXXX XXXXXXXX
0038B  00110000XX000X1X XXXXX00000000010 1101101 0XXXX0011 11001111
0038C  11100000XX000XXX XXXXX00101000000 11011100XXXXXXXX XXXXXXXX
0038D  11100000XX000XXX XXXXX01000000010 11011100XXXXXXXX XXXXXXXX
0038E  11100000XX000XXX XXXXX00110001110 11011100XXXXXXXX XXXXXXXX
0038F  11100000XX000XXX XX01001001001010 110111001111XXXX XXXXXXXX
00390  11100000XX000XXX XXXXX01010010000 11011100XXXXXXXX XXXXXXXX
00391  10100000XX000X1X 01XXX00000000000 00010010XXXXXXXX XXXXXXXX
00392  00010000XX000X1X XXXXX00001010100 11011100XXXX0011 11000101
00393  11100000XX000XXX 0110100110001110 1100000 1XXXXXXXX XXXXXXXX
```

```
00394  11100000XX000XXX XXXXX00000001010 00011011XXXXXXXX XXXXXXXX
00395  01000000XX000X1X 00XXX00000010011 00100011XXXX0000 00001110
00396  10000000XX000XXX 00XXX10011010011 00000001XXXXXXXX XXXX1110
00397  11100000XX000XXX 00XXX10011010011 00001001XXXXXXXX XXXX1110
00398  11100000XX000XXX XXXXX00000000010 11011010XXXXXXXX XXXXXXXX
00399  00010101XX000X1X XXXXX01000010000 10011111XXXX0011 10011100
0039A  00010000XX000X1X XXXXX00101000000 11011100XXXX0011 10011100
0039B  00110000XX000X1X XXXXX01001000000 11011100XXXX0011 11001111
0039C  11100101XX000XXX XXXXX01000010000 10011111XXXXXXXX XXXXXXXX
0039D  11000000XX000XXX XX00000000010000 0001101100110000 00000111
0039E  01000000XX000X00 00X1000000010001 001000110101XXXX XXXX1110
0039F  100000000X000XXX 0001010011010001 000000011111XXXX XXXX1110
003A0  11100000XX000XXX 0010101000000010 11000001XXXXXXXX XXXXXXXX
003A1  11100000XX000XXX 11X0100000010010 110000110110XXXX XXXXXXXX
003A2  10100101XX000X00 XXX1001010010000 100111110010XXXX XXXXXXXX
003A3  10100000XX000X1X 01XXX00000010010 11001011XXXXXXXX XXXXXXXX
003A4  00010000XX000X1X XXXXX00000000000 01011010XXXX0011 11000101
003A5  11100000XX000XXX XXXXX00000000010 00011011XXXXXXXX XXXXXXXX
003A6  11100000XX000XXX 00XXX00000001001 00000100XXXXXXXX XXXX1110
003A7  11100000XX000XXX 00XXX00000001001 00000011XXXXXXXX XXXX1110
003A8  00110000XX000X00 00X0100110001110 1100100111100011 10101011
003A9  11100000XX000XXX 0110100000000000 00010010XXXXXXXX XXXXXXXX
003AA  11100000XX000XXX 11X0100000001000 110100110110XXXX XXXXXXXX
003AB  11100000XX000XXX XX01000101001100 110111001111XXXX XXXXXXXX
003AC  00110000XX000X00 XXX1000000001010 1111000101010011 11101011
003AD  00110000XX000X00 XXX1000000000010 1110001111100011 10110000
003AE  11100000XX000XXX 0110100000010000 11010011XXXXXXXX XXXXXXXX
003AF  11100000XX000XXX 11X0100000001100 110100110110XXXX XXXXXXXX
003B0  00010000XX000X1X 01XXX00000001100 11000011XXXX0011 11000010
003B1  11100000XX000XXX XX01000000001000 010110111111XXXX XXXXXXXX
003B2  00110000XX000X00 XX01000000010010 1111101011100011 10110100
003B3  11100000XX000XXX 00XXX00110001000 11000001XXXXXXXX XXXXXXXX
003B4  00110000XX000X00 00X1001001001000 1100000111110011 11000001
003B5  11100000XX000XXX XXXXX01001000000 11011100XXXXXXXX XXXXXXXX
003B6  00010101XX000X1X XXXXX01000010000 10011111XXXX0011 10011100
003B7  11100000XX000XXX 00XXX00001001001 00001001XXXXXXXX XXXX0110
003B8  11100000XX000XXX XXXXX00000001001 00011011XXXXXXXX XXXX0110
003B9  00010000XX000X1X 01XXX00000001110 11000011XXXX0011 11000010
003BA  11100000XX000XXX XXXXX00000000010 11111010XXXXXXXX XXXXXXXX
003BB  01000000XX000X1X XXXXX00000000000 00100010XXXX0000 00000001
003BC  10000000XX000XXX XXXXX00000000011 10011011XXXXXXXX XXXX1111
003BD  11100000XX000XXX XX10100000001010 01011011XXXXXXXX XXXXXXXX
003BE  00110000XX000X00 00X0101001000000 1100000011100011 11001111
003BF  11100000XX000XXX 0110100000000010 11010011XXXXXXXX XXXXXXXX
003C0  00110000XX000X1X 11X0100000000000 1101001101100011 11001111
003C1  00110000XX000X1X XXXXX00000000000 11100011XXXX0011 11001111
003C2  01000000XX000X10 XXX1100000001000 0101101111110000 00010000
003C3  10000000XX000XX0 00X1100110101001 100010011111XXXX XXXX1111
003C4  10100000XX000X1X XXXXX00000001001 01011011XXXXXXXX XXXX1111
003C5  0001000000000X1X XX00000001001110 1101110001000001 00111001
003C6  11100110XX000XXX XXXXX00110010000 1110010111111111 00000000
```

```
003C7  11100000XX000XXX XXXXX01000001101 01110001XXXXXXXX XXXX0010
003C8  00110000XX000X00 XXX1000000000000 0101101010100011 11001010
003C9  11100110XX000XXX XXXXX00110001100 1101110111111111 10000000
003CA  11100110XX000XXX XXXXX00001000010 1110010111111111 00000000
003CB  11100000XX000XXX XXXXX00001001111 01110001XXXXXXXX XXXX0010
003CC  00110000X0000X00 0001000000001110 0100001110100011 11001110
003CD  11100110XX000XXX XXXXX00111001110 1101110111111111 10000000
003CE  1010000000000X1X XX00000000001110 010110110100XXXX XXXXXXXX
003CF  111000000X000XXX 0001000000000010 000000111111XXXX XXXXXXXX
003D0  00110000XX000X00 XX01000000000000 0101101101000011 11010010
003D1  00110000XX000X00 XXX1000001000010 1111000101010001 00000000
003D2  01000110XX000X1X 00XXX00111001110 1100010100000000 00000010
003D3  11000000XX000XX0 01X1000000001110 1000101111110000 00000001
003D4  11010000XX000X01 XXX1000000000001 100110110000XXXX XXXX0110
003D5  11100000XX000XXX XXXXX00000000001 00011011XXXXXXXX XXXX0101
003D6  11100000XX000XXX XXXXX00000000001 00011011XXXXXXXX XXXX0101
003D7  11100110XX000XXX XXXXX00000000000 1111010101000000 00000000
003D8  11100110XX000XXX XXXXX00000010000 1101111100000000 01111111
003D9  11100000XX000XXX 0010100000000000 0100001XXXXXXXXX XXXXXXXX
003DA  11100000XX000XXX 11X0101000000010 110000001000XXXX XXXXXXXX
003DB  11100000XX000XXX 11X0100000000000 110000111000XXXX XXXXXXXX
003DC  00110000XX000X00 00X0100000000000 0100000101100011 11011111
003DD  11100000XX000XXX XXXXX00000000001 01011011XXXXXXXX XXXX0000
003DE  00110000XX000X1X 01XXX00000001110 1100011XXXX0011 11100010
003DF  00110000XX000X00 XXX0100000000000 0101101001110011 11100010
003E0  11100000XX000XXX XXXXX00000000001 11011011XXXXXXXX XXXX0000
003E1  11100000XX000XXX 01XXX00000001110 11001011XXXXXXXX XXXXXXXX
003E2  11100110XX000XXX XXXXX00111001100 1110010100000000 10000000
003E3  11100000XX000XXX 0010100111001100 1100001XXXXXXXXX XXXXXXXX
003E4  11100000XX000XXX 11X0100110001100 110000010110XXXX XXXXXXXX
003E5  11100110XX000XXX XX10100110000000 0110010111111111 00000000
003E6  00110000XX000X00 XXX0100000001110 0101101101000011 11101001
003E7  11100110XX000XXX XXXXX00001000010 1110010111111111 00000000
003E8  001100000X000X1X XX00000110000010 1101100100110001 00000000
003E9  00110000XX000X00 XXX0100000000000 1110001111100011 11101011
003EA  001100000X000X1X XX00000000000010 1110001100010001 00000000
003EB  11100000XX000XXX XXXXX00000000001 01111001XXXXXXXX XXXX0000
003EC  001100000X000X1X XX00000001000011 1111100100010001 00000000
003ED  11100000XX000XXX XX10100000000000 01011011XXXXXXXX XXXXXXXX
003EE  00110000XX000X00 XXX0100000000000 0101101011100011 00101001
003EF  11101000XX001XXX 01XXX00000011110 01001011XXXXXXXX XXXXXXXX
003F0  00000110XX000XXX XXXXX00000000000 1101111100000000 01010000
003F1  0000000000000000 0000000000000000 0000000000000000 00000000
003F2  0000000000000000 0000000000000000 0000000000000000 00000000
003F3  0000000000000000 0000000000000000 0000000000000000 00000000
003F4  0000000000000000 0000000000000000 0000000000000000 00000000
003F5  0000000000000000 0000000000000000 0000000000000000 00000000
003F6  0000000000000000 0000000000000000 0000000000000000 00000000
003F7  0000000000000000 0000000000000000 0000000000000000 00000000
003F8  0000000000000000 0000000000000000 0000000000000000 00000000
003F9  0000000000000000 0000000000000000 0000000000000000 00000000
```

```
003FA  0000000000000000 0000000000000000 0000000000000000 00000000
003FB  0000000000000000 0000000000000000 0000000000000000 00000000
003FC  0000000000000000 0000000000000000 0000000000000000 00000000
003FD  0000000000000000 0000000000000000 0000000000000000 00000000
003FE  0000000000000000 0000000000000000 0000000000000000 00000000
003FF  00110000XX000X1X 01XXX00000000010 11000011XXXX0001 10100101
```

HEWLETT
PACKARD

Vol. 2
MANUAL PART NO. 02156-90003
Printed in U.S.A. March 1983
E0383

HEWLETT-PACKARD COMPANY
Data Systems Division
11000 Wolfe Road
Cupertino, California 95014