

!A

LLOAD REGS.L,A\$4000

LLOAD INCL.L,A\$4000

LLOAD PAGE0.L,A\$4000

LLOAD PAGE1.L,A\$4000

LLOAD PAGE2.L,A\$4000

LLOAD PAGE3.L,A\$4000

LLOAD PAGE4.L,A\$4000

LLOAD PAGE5.L,A\$4000

LLOAD PAGE6.L,A\$4000

LLOAD PAGE7.L,A\$4000

LLOAD PPB.L,A\$4000

*** End of Pass 1

LLOAD REGS.L,A\$4000

LLOAD INCL.L,A\$4000

LLOAD PAGE0.L,A\$4000

LLOAD PAGE1.L,A\$4000

LLOAD PAGE2.L,A\$4000

LLOAD PAGE3.L,A\$4000

LLOAD PAGE4.L,A\$4000

LLOAD PAGE5.L,A\$4000

LLOAD PAGE6.L,A\$4000

LLOAD PAGE7.L,A\$4000

LLOAD PPB.L,A\$4000

*** End of Pass 2

```
0800      1          ttl "PPB Source Code, PPB.L"
0800      2          src "PPB.L"
0800      3      ;
0800      4      ;
0800      5      ; PPB.L
0800      6      ;
0800      7      ;
0800      8      ; Parallel Printer Buffer Source Code
0800      9      ;
0800     10      ; 1989 May 14 (original Copyright Date)
0800     11      ;
0800     12      ; 2024 February 14
0800     13      ;
0800     14      ;
0800     15      ; DOS 4.5, Build 06
0800     16      ;
0800     17      ; 2024 February 14
0800     18      ;
0800     19      ;
0800     20      ; Start of Source Code:  0x4000
0800     21      ;   End of Source Code:  0x7800
0800     22      ;
0800     23      ;
0800     24      ; Copyright (c) 2024 February 14 by
0800     25      ; Walland Philip Vrbancic Jr
0800     26      ;
0800     27      ; 6223 East Peabody Street
0800     28      ; Long Beach, California  90808
0800     29      ; United States of America
0800     30      ;
0800     31      ; All Rights Reserved
0800     32      ;
0800     33      ; This software is the confidential and
0800     34      ; proprietary intellectual property of
0800     35      ; Walland Philip Vrbancic Jr
0800     36      ;
0800     37      ;
0800     38      ; Original hardware design and function by
0800     39      ; Johnathon Freeman Designs, Inc.
0800     40      ;
0800     41      ; 3067 Dolores Street
0800     42      ; San Francisco, California  94110
0800     43      ;
0800     44      ; 415-822-8451
0800     45      ;
0800     46      ;
0800     47          icl "REGS.L"
```

```
LLOAD REGS.L,A$4000
```

```

0800      1          ttl "PPB Source Code, REGS.L"
0800      2      ;
0800      3      ;
0800      4      ; REGS.L
0800      5      ;
0800      6      ;
0800      7      ; MCS-48 8035 Processor Data Memory Map
0800      8      ;
0800      9      ; -----
0800     10      ; 63 |          User RAM          |
0800     11      ;          |
0800     12      ;          |
0800     13      ;          |
0800     14      ;          | 32 Indexed
0800     15      ;          | Word Locations    |
0800     16      ; 32 |-----
0800     17      ;
0800     18      ; 31 |          Bank 1          |
0800     19      ;          |
0800     20      ;          | Secondary Registers |
0800     21      ; 24 |-----
0800     22      ;
0800     23      ; 23 |          Program Counter  |
0800     24      ;          | Program Status Word |
0800     25      ;          |
0800     26      ;          |
0800     27      ;          | Program Counter
0800     28      ;          | Stack
0800     29      ;          | 8 levels
0800     30      ; 08 |-----
0800     31      ;
0800     32      ; 07 |          Bank 0          |
0800     33      ;          |
0800     34      ;          | Primary Registers   |
0800     35      ; 00 |-----
0800     36      ;
0800     37      ;
0800     38      ;
0800     39      ;
0800     40      ; Primary Register (SELRB0) Utilization (Bank 0):
0800     41      ;
0800     42      ; R7 - Address Bits 09-16 Write Data CAS
0800     43      ; R6 - Address Bits 00-07 Write Data RAS
0800     44      ; R5 - Address Bits 09-16 Read Data CAS
0800     45      ; R4 - Address Bits 00-07 Read Data RAS
0800     46      ; R3 - R/W Block Numbers for Address Bits 08 & 17
0800     47      ; R2 - Temporary Data Byte
0800     48      ; R1 - EXTIRQ reg-A Save
0800     49      ; R0 - Input/Output Working
0800     50      ;
0800     51      ;
0800     52      ; Secondary Register (SELRB1) Utilization (Bank 1):
0800     53      ;
0800     54      ; R7 - Timer Counter, MSB
0800     55      ; R6 - Timer Counter, LSB
0800     56      ; R5 - Refresh Counter, LSB
0800     57      ; R4 - Copy Number
0800     58      ; R3 - System Flags
0800     59      ; R2 - Temporary Data Byte
0800     60      ; R1 - REFRESH reg-A Save

```

```
0800      61 ; R0 - Input/Output Working
0800      62 ;
0800      63 ;
0800      64 ; User RAM Word Utilization:
0800      65 ;
0800      66 ; W32 - Power Byte 1, 0x96
0800      67 ; W33 - Power Byte 2, 0xA5
0800      68 ;
0800      69 ; W34 - Test LED 1 counter
0800      70 ; W35 - Test LED 2 counter
0800      71 ;
0800      72 ;
0800      73 ;
0800      74 ; R/W Block Number Bits (Primary Register R3):
0800      75 ;
0800      76 ; B7 - 0
0800      77 ; B6 - 0
0800      78 ; B5 - Write Block Number MSB
0800      79 ; B4 - Write Block Number LSB
0800      80 ; B3 - 0
0800      81 ; B2 - 0
0800      82 ; B1 - Read Block Number MSB
0800      83 ; B0 - Read Block Number LSB
0800      84 ;
0800      85 ;
0800      86 ; System Flag Bits (Secondary Register R3):
0800      87 ;
0800      88 ; B7 - EXTIRQ State Flag, 0=Off
0800      89 ; B6 - Pause State Flag, 0=Off
0800      90 ; B5 - Copy State Flag, 0=Off
0800      91 ; B4 - Message State Flag, 0=Off
0800      92 ; B3 - Overflow State Flag, 0=Off
0800      93 ; B2 - 0
0800      94 ; B1 - 0
0800      95 ; B0 - Refresh Counter, ADR08
0800      96 ;
0800      97 ;
0800      98 ;
0800      99 ; Port 1 Utilization:
0800     100 ;
0800     101 ; B7 - PPB Memory Enable (PMEMENBL), 1=disable
0800     102 ; B6 - 1
0800     103 ; B5 - 1
0800     104 ; B4 - Outport Data Ready, 1=ready
0800     105 ; B3 - 1
0800     106 ; B2 - Outport Latch Enable (OLATENBL), 1=disable
0800     107 ; B1 - Processor Data Enable (PDATENBL), 1=disable
0800     108 ; B0 - 1
0800     109 ;
0800     110 ;
0800     111 ; Port 2 Utilization:
0800     112 ;
0800     113 ; B7 - Octal Latch Enable (OCTLENBL), 1=disable
0800     114 ; B6 - 1
0800     115 ; B5 - 1
0800     116 ; B4 - Inport Data Strobe (IDATENBL), 1=disable
0800     117 ; B3 - Ready LED (RDLEDON), 1=On
0800     118 ; B2 - Test LED 2 (LED2TGL), 1=On
0800     119 ; B1 - Test LED 1 (LED1TGL), 1=On
0800     120 ; B0 - Address Bit ADR08 (ADR08ON), 1=On
0800     121 ;
```

```
0800      122 ;
0800      123 ;
0800      124 ; Program Status Word (PSW)
0800      125 ;
0800      126 ; The PSW consists of status bits, flags, and a 12-bit
0800      127 ; program counter, and is stored in two stack locations.
0800      128 ;
0800      129 ; The upper four bits of the PSW are stored in the
0800      130 ; Program Counter Stack with every call to a subroutine
0800      131 ; or an interrupt vector, and are optionally restored
0800      132 ; upon return with the RETR instruction. The RET return
0800      133 ; does not update the PSW.
0800      134 ;
0800      135 ; The User Flag 1, F1, is not part of the PSW.
0800      136 ;
0800      137 ; B7 - CY, Carry Flag
0800      138 ; B6 - AC, Auxiliary Carry Flag
0800      139 ; B5 - F0, User Flag 0
0800      140 ; B4 - BS, Register Bank Select
0800      141 ; B3 - 1, not used
0800      142 ; B2 - S2, B2 of Stack Pointer
0800      143 ; B1 - S1, B1 of Stack Pointer
0800      144 ; B0 - S0, B0 of Stack Pointer
0800      145 ;
0800      146 ;
0800      147      icl "INCL.L"
```

```
LLOAD INCL.L,A$4000
```

```

0800      1          ttl "PPB Source Code, INCL.L"
0800      2      ;
0800      3      ;
0800      4      ; INCL.L
0800      5      ;
0800      6      ;
0000      7      ZERO      equ $00
0001      8      ONE       equ $01
0004      9      FOUR      equ $04
0081     10      RTN1       equ $81
0082     11      RTN2       equ $82
0083     12      RTN3       equ $83
00FF     13      NEGONE     equ $FF
0800     14      ;
0000     15      BLOC0      equ $00
0800     16      ;
0020     17      PW1NDX      equ $20
0021     18      PW2NDX      equ $21
0022     19      LED1NDX     equ $22
0023     20      LED2NDX     equ $23
0800     21      ;
0096     22      PW1VAL      equ $96
00A5     23      PW2VAL      equ $A5
0800     24      ;
000A     25      LINEFEED    equ $0A
000D     26      RETURN      equ $0D
0020     27      SPACE       equ ' '
002A     28      STAR        equ '*'
003A     29      COLON       equ ':'
007F     30      ASCIMASK     equ $7F
0800     31      ;
0030     32      CHAR30      equ $30          ; character '0'
0031     33      CHAR31      equ $31          ; character '1'
0032     34      CHAR32      equ $32          ; character '2'
0033     35      CHAR33      equ $33          ; character '3'
0050     36      CHAR50      equ $50          ; character 'P'
0080     37      CHAR80      equ $80          ; character "<Nul>"
0800     38      ;
005A     39      PATRN1      equ $5A
00A5     40      PATRN2      equ $A5
0069     41      PATRN3      equ $69
0096     42      PATRN4      equ $96
0800     43      ;
0800     44      ;
0031     45      TMRCNT      equ 49          ; Interval Timer count
0050     46      TMRPSCAL    equ 80          ; Timer Prescaler (32*Tcy)
0014     47      TMRRSET     equ 20          ; Timer Reset (8*Tcy)
0800     48      ;
0F64     49      TMRTIM      equ TMRRSET+TMRPSCAL*TMRCNT ; Timer time
0800     50      ;
FA00     51      TIME2VAL     equ 64000      ; 64 msec value
0010     52      TIME3VAL     equ 16         ; 1000 ms/64 msec value
0018     53      TIME4VAL     equ 24         ; 1500 ms/64 msec value
0800     54      ;
0010     55      TIME2CNT     equ TIME2VAL/TMRTIM ; 50 msec count
0100     56      TIME3CNT     equ TIME3VAL*TIME2CNT ; 1 second count
0180     57      TIME4CNT     equ TIME4VAL*TIME2CNT ; 2 second count
0800     58      ;
FFCF     59      TIME1       equ ZERO-TMRCNT ; Interval Timer load count
FFF0     60      TIME2       equ ZERO-TIME2CNT ; 64 msec debounce count

```

```

FF00      61  TIME3      equ  ZERO-TIME3CNT      ; 1.0 second interval count
FE80      62  TIME4      equ  ZERO-TIME4CNT      ; 1.5 second interval count
0800      63  ;
0800      64  ;
0001      65  ADDRDBLK   equ  %000000001         ; increment read block number
0010      66  ADDWRBLK   equ  %00010000         ; increment write block number
0011      67  ADDRWBK    equ  ADDRDBLK|ADDWRBLK   ; increment r/w block number
0800      68  ;
0003      69  MSKRDBLK   equ  %000000011         ; mask read block number
0030      70  MSKWRBLK   equ  %00110000         ; mask write block number
0033      71  MSKRWBK    equ  MSKRDBLK|MSKWRBLK   ; mask r/w block number
0030      72  CLRRDBLK   equ  %00110000         ; clear read block number
0800      73  ;
0800      74  ;
0010      75  LEDCNT     equ  %00010000         ; maximum toggle count
0080      76  LEDBIT     equ  %10000000         ; LED toggle bit
007F      77  LEDMSK     equ  LEDBIT^NEGONE      ; LED bit mask
0800      78  ;
0800      79  ;
0800      80  ; Primary Register 2 Bit Usage
0800      81  ;
0080      82  IRQSFON    equ  $80                ; EXTIRQ State Flag on
007F      83  IRQSFOFF   equ  IRQSFON^NEGONE     ; EXTIRQ State Flag off
0800      84  ;
0040      85  PAUSFON    equ  $40                ; Pause State Flag on
00BF      86  PAUSFOFF   equ  PAUSFON^NEGONE     ; Pause State Flag off
0040      87  PAUSFTGL   equ  PAUSFON           ; Pause State Flag toggle
0800      88  ;
0020      89  CPYSFON    equ  $20                ; Copy State Flag on
00DF      90  CPYSFOFF   equ  CPYSFON^NEGONE     ; Copy State Flag off
0800      91  ;
0010      92  MSGSFON    equ  $10                ; Message State Flag on
00EF      93  MSGSFOFF   equ  MSGSFON^NEGONE     ; Message State Flag off
0800      94  ;
0008      95  OVFSFON    equ  $08                ; Overflow State Flag on
00F7      96  OVFSFOFF   equ  OVFSFON^NEGONE     ; Overflow State Flag off
0800      97  ;
0800      98  ;
0800      99  ; Port 1 Bit Usage
0800     100  ;
0080     101  PMEMDSBL   equ  $80                ; PPB Memory disable
007F     102  PMEMENBL   equ  PMEMDSBL^NEGONE     ; PPB Memory enable
0800     103  ;
0004     104  OLATDSBL   equ  $04                ; Outport Latch disable
00FB     105  OLATENBL   equ  OLATDSBL^NEGONE     ; Outport Latch enable
0800     106  ;
0002     107  PDATDSBL   equ  $02                ; Processor Data disable
00FD     108  PDATENBL   equ  PDATDSBL^NEGONE     ; Processor Data enable
0800     109  ;
0800     110  ;
0800     111  ; Port 2 Bit Usage
0800     112  ;
0080     113  OCTLDSBL   equ  $80                ; Octal Latch disable
007F     114  OCTLENBL   equ  OCTLDSBL^NEGONE     ; Octal Latch enable
0800     115  ;
0010     116  IDATDSBL   equ  $10                ; Inport Data Strobe disable
00EF     117  IDATENBL   equ  IDATDSBL^NEGONE     ; Inport Data Strobe enable
0800     118  ;
0008     119  RDLEDON    equ  $08                ; Ready LED on
00F7     120  RDLEDOFF   equ  RDLEDON^NEGONE     ; Ready LED off
0800     121  ;

```

```

0004      122  LED2TGL  equ $04          ; Test LED 2 toggle
0800      123  ;
0002      124  LED1TGL  equ $02          ; Test LED 1 toggle
0800      125  ;
0800      126  ;
0800      127  ; Refresh Counter ADR08
0800      128  ;
0001      129  ADR08MSK equ %000000001    ; ADR08 bit mask
0001      130  ADR08TGL equ %000000001    ; ADR08 bit toggle
0800      131  ;
0001      132  ADR08ON  equ %000000001    ; Address Bit ADR08 on
00FE      133  ADR08OFF equ ADR08ON^NEGONE ; Address Bit ADR08 off
0800      134  ;
0800      135  ;
0800      136  ; MCS-8048 Instruction Set
0800      137  ;
0000      138  NOP      equ $00          ; no operation
0002      139  OUTDBA   equ $02          ; output to bus, BUS=A
0003      140  ADDA     equ $03          ; add immediate, A=A+n
0004      141  JMP0     equ $04          ; unconditional jump, PC=0nn
0005      142  ENI      equ $05          ; enable external interrupts
0007      143  DECA     equ $07          ; decrement reg-A, A=A-1
0008      144  INSABUS  equ $08          ; input BUS with strobe, A=BUS
0009      145  INAP1    equ $09          ; input from I/O port 1, A=P1
000A      146  INAP2    equ $0A          ; input from I/O port 2, A=P2
000C      147  MOVDAP4  equ $0C          ; move from exp port 4, A=EP4
000D      148  MOVDAP5  equ $0D          ; move from exp port 5, A=EP5
000E      149  MOVDAP6  equ $0E          ; move from exp port 6, A=EP6
000F      150  MOVDAP7  equ $0F          ; move from exp port 7, A=EP7
0800      151  ;
0010      152  INC.R0   equ $10          ; increment mem, [R0]=[R0]+1
0011      153  INC.R1   equ $11          ; increment mem, [R1]=[R1]+1
0012      154  JB0      equ $12          ; jump if reg-A bit 0 set
0013      155  ADDCA    equ $13          ; add immediate+carry, A=A+n+C
0014      156  CALL0    equ $14          ; call page 0 subrtn, PC=0nn
0015      157  DISI     equ $15          ; disable external interrupts
0016      158  JTF      equ $16          ; jump if timer flag set
0017      159  INCA     equ $17          ; increment reg-A, A=A+1
0018      160  INCR0    equ $18          ; increment R0, R0=R0+1
0019      161  INCR1    equ $19          ; increment R1, R1=R1+1
001A      162  INCR2    equ $1A          ; increment R2, R2=R2+1
001B      163  INCR3    equ $1B          ; increment R3, R3=R3+1
001C      164  INCR4    equ $1C          ; increment R4, R4=R4+1
001D      165  INCR5    equ $1D          ; increment R5, R5=R5+1
001E      166  INCR6    equ $1E          ; increment R6, R6=R6+1
001F      167  INCR7    equ $1F          ; increment R7, R7=R7+1
0800      168  ;
0020      169  XCHA.R0  equ $20          ; exchange memory, A<->[R0]
0021      170  XCHA.R1  equ $21          ; exchange memory, A<->[R1]
0023      171  MOVA     equ $23          ; move from immediate, A=n
0024      172  JMP1     equ $24          ; unconditional jump, PC=1nn
0025      173  ENTI     equ $25          ; enable timer interrupt
0026      174  JNT0     equ $26          ; jump if T0 clear
0027      175  CLRA     equ $27          ; clear reg-A, A=0
0028      176  XCHAR0   equ $28          ; exchange R0, A<->R0
0029      177  XCHAR1   equ $29          ; exchange R1, A<->R1
002A      178  XCHAR2   equ $2A          ; exchange R2, A<->R2
002B      179  XCHAR3   equ $2B          ; exchange R3, A<->R3
002C      180  XCHAR4   equ $2C          ; exchange R4, A<->R4
002D      181  XCHAR5   equ $2D          ; exchange R5, A<->R5
002E      182  XCHAR6   equ $2E          ; exchange R6, A<->R6

```

```

002F      183  XCHAR7      equ  $2F      ; exchange R7, A<->R7
0800      184  ;
0030      185  XCHDA.R0    equ  $30      ; exchange R0 lower nibble
0031      186  XCHDA.R1    equ  $31      ; exchange R1 lower nibble
0032      187  JB1         equ  $32      ; jump if reg-A bit 1 set
0034      188  CALL1       equ  $34      ; call page 1 subrtn, PC=1nn
0035      189  DISTI       equ  $35      ; disable timer interrupt
0036      190  JT0         equ  $36      ; jump if T0 is one
0037      191  CPLA        equ  $37      ; complement reg-A, A=-A
0039      192  OUTLP1A     equ  $39      ; output to P1, P1=A
003A      193  OUTLP2A     equ  $3A      ; output to P2, P2=A
003C      194  MOVDP4A     equ  $3C      ; move to exp port 4, EP4=A
003D      195  MOVDP5A     equ  $3D      ; move to exp port 5, EP5=A
003E      196  MOVDP6A     equ  $3E      ; move to exp port 6, EP6=A
003F      197  MOVDP7A     equ  $3F      ; move to exp port 7, EP7=A
0800      198  ;
0040      199  ORLA.R0     equ  $40      ; OR logical memory, A=A|[R0]
0041      200  ORLA.R1     equ  $41      ; OR logical memory, A=A|[R1]
0042      201  MOVAT       equ  $42      ; move from timer, A=T
0043      202  ORLA        equ  $43      ; OR logical immediate, A=A|n
0044      203  JMP2        equ  $44      ; unconditional jump, PC=2nn
0045      204  STRTC       equ  $45      ; start counter
0046      205  JNT1        equ  $46      ; jump if T1 clear
0047      206  SWAPA       equ  $47      ; swap reg-A nibbles
0048      207  ORLAR0      equ  $48      ; OR logical R0, A=A|R0
0049      208  ORLAR1      equ  $49      ; OR logical R1, A=A|R1
004A      209  ORLAR2      equ  $4A      ; OR logical R2, A=A|R2
004B      210  ORLAR3      equ  $4B      ; OR logical R3, A=A|R3
004C      211  ORLAR4      equ  $4C      ; OR logical R4, A=A|R4
004D      212  ORLAR5      equ  $4D      ; OR logical R5, A=A|R5
004E      213  ORLAR6      equ  $4E      ; OR logical R6, A=A|R6
004F      214  ORLAR7      equ  $4F      ; OR logical R7, A=A|R7
0800      215  ;
0050      216  ANLA.R0     equ  $50      ; AND logical memory, A=A&[R0]
0051      217  ANLA.R1     equ  $51      ; AND logical memory, A=A&[R1]
0052      218  JB2         equ  $52      ; jump if reg-A bit 2 set
0053      219  ANLA        equ  $53      ; AND logical immediate, A=A&n
0054      220  CALL2       equ  $54      ; call page 2 subrtn, PC=2nn
0055      221  STRTT       equ  $55      ; start timer
0056      222  JT1         equ  $56      ; jump if T1 is one
0057      223  DAA         equ  $57      ; BCD adjust reg-A, A=BCD{A}
0058      224  ANLAR0      equ  $58      ; AND logical R0, A=A&R0
0059      225  ANLAR1      equ  $59      ; AND logical R1, A=A&R1
005A      226  ANLAR2      equ  $5A      ; AND logical R2, A=A&R2
005B      227  ANLAR3      equ  $5B      ; AND logical R3, A=A&R3
005C      228  ANLAR4      equ  $5C      ; AND logical R4, A=A&R4
005D      229  ANLAR5      equ  $5D      ; AND logical R5, A=A&R5
005E      230  ANLAR6      equ  $5E      ; AND logical R6, A=A&R6
005F      231  ANLAR7      equ  $5F      ; AND logical R7, A=A&R7
0800      232  ;
0060      233  ADDA.R0     equ  $60      ; add memory, A=A+[R0]
0061      234  ADDA.R1     equ  $61      ; add memory, A=A+[R1]
0062      235  MOVTA       equ  $62      ; move to timer, T=A
0064      236  JMP3        equ  $64      ; unconditional jump, PC=3nn
0065      237  STOPTC      equ  $65      ; stop timer/counter
0067      238  RRCA        equ  $67      ; rotate right+carry, A=->AC
0068      239  ADDAR0      equ  $68      ; add R0, A=A+R0
0069      240  ADDAR1      equ  $69      ; add R1, A=A+R1
006A      241  ADDAR2      equ  $6A      ; add R2, A=A+R2
006B      242  ADDAR3      equ  $6B      ; add R3, A=A+R3
006C      243  ADDAR4      equ  $6C      ; add R4, A=A+R4

```

006D	244	ADDAR5	equ	\$6D	; add R5, A=A+R5
006E	245	ADDAR6	equ	\$6E	; add R6, A=A+R6
006F	246	ADDAR7	equ	\$6F	; add R7, A=A+R7
0800	247	;			
0070	248	ADDCA.R0	equ	\$70	; add memory+carry, A=A+[R0]+C
0071	249	ADDCA.R1	equ	\$71	; add memory+carry, A=A+[R1]+C
0072	250	JB3	equ	\$72	; jump if reg-A bit 3 set
0074	251	CALL3	equ	\$74	; call page 3 subrtn, PC=3nn
0075	252	ENT0	equ	\$75	; enable timer output
0076	253	JF1	equ	\$76	; jump if flag #1 set
0077	254	RRA	equ	\$77	; rotate right, A=->A
0078	255	ADDCAR0	equ	\$78	; add R0 + carry, A=A+R0+C
0079	256	ADDCAR1	equ	\$79	; add R1 + carry, A=A+R1+C
007A	257	ADDCAR2	equ	\$7A	; add R2 + carry, A=A+R2+C
007B	258	ADDCAR3	equ	\$7B	; add R3 + carry, A=A+R3+C
007C	259	ADDCAR4	equ	\$7C	; add R4 + carry, A=A+R4+C
007D	260	ADDCAR5	equ	\$7D	; add R5 + carry, A=A+R5+C
007E	261	ADDCAR6	equ	\$7E	; add R6 + carry, A=A+R6+C
007F	262	ADDCAR7	equ	\$7F	; add R7 + carry, A=A+R7+C
0800	263	;			
0080	264	MOVXA.R0	equ	\$80	; move from ext mem, A=[R0]
0081	265	MOVXA.R1	equ	\$81	; move from ext mem, A=[R1]
0083	266	RET	equ	\$83	; return from subroutine
0084	267	JMP4	equ	\$84	; unconditional jump, PC=4nn
0085	268	CLRF0	equ	\$85	; clear flag #0, F0=0
0086	269	JNI	equ	\$86	; jump if no interrupt
0088	270	ORLBUS	equ	\$88	; OR logical BUS, BUS=BUS n
0089	271	ORLP1	equ	\$89	; OR logical port 1, P1=P1 n
008A	272	ORLP2	equ	\$8A	; OR logical port 2, P2=P2 n
008C	273	ORLDP4A	equ	\$8C	; OR logical EP4, EP4=EP4 A
008D	274	ORLDP5A	equ	\$8D	; OR logical EP5, EP5=EP5 A
008E	275	ORLDP6A	equ	\$8E	; OR logical EP6, EP6=EP6 A
008F	276	ORLDP7A	equ	\$8F	; OR logical EP7, EP7=EP7 A
0800	277	;			
0090	278	MOVX.R0A	equ	\$90	; move to memory, [R0]=A
0091	279	MOVX.R1A	equ	\$91	; move to memory, [R1]=A
0092	280	JB4	equ	\$92	; jump if reg-A bit 4 set
0093	281	RETR	equ	\$93	; return and restore status
0094	282	CALL4	equ	\$94	; call page 4 subrtn, PC=4nn
0095	283	CPLF0	equ	\$95	; complement flag #0, F0=-F0
0096	284	JNZ	equ	\$96	; jump if reg-A not zero
0097	285	CLRC	equ	\$97	; clear carry status, C=0
0098	286	ANLBUS	equ	\$98	; AND logical BUS, BUS=BUS&n
0099	287	ANLP1	equ	\$99	; AND logical port 1, P1=P1&n
009A	288	ANLP2	equ	\$9A	; AND logical port 2, P2=P2&n
009C	289	ANLDP4A	equ	\$9C	; AND logical EP4, EP4=EP4&A
009D	290	ANLDP5A	equ	\$9D	; AND logical EP5, EP5=EP5&A
009E	291	ANLDP6A	equ	\$9E	; AND logical EP6, EP6=EP6&A
009F	292	ANLDP7A	equ	\$9F	; AND logical EP7, EP7=EP7&A
0800	293	;			
00A0	294	MOV.R0A	equ	\$A0	; move to memory, [R0]=A
00A1	295	MOV.R1A	equ	\$A1	; move to memory, [R1]=A
00A3	296	MOVPA.A	equ	\$A3	; move from memory, A={PC+A}
00A4	297	JMP5	equ	\$A4	; unconditional jump, PC=5nn
00A5	298	CLRF1	equ	\$A5	; clear flag #1, F1=0
00A7	299	CPLC	equ	\$A7	; complement carry, C=-C
00A8	300	MOVR0A	equ	\$A8	; move to R0, R0=A
00A9	301	MOVR1A	equ	\$A9	; move to R1, R1=A
00AA	302	MOVR2A	equ	\$AA	; move to R2, R2=A
00AB	303	MOVR3A	equ	\$AB	; move to R3, R3=A
00AC	304	MOVR4A	equ	\$AC	; move to R4, R4=A

```

00AD      305  MOVR5A    equ  $AD      ; move to R5, R5=A
00AE      306  MOVR6A    equ  $AE      ; move to R6, R6=A
00AF      307  MOVR7A    equ  $AF      ; move to R7, R7=A
0800      308  ;
00B0      309  MOV.R0     equ  $B0      ; move immed to memory, [R0]=n
00B1      310  MOV.R1     equ  $B1      ; move immed to memory, [R1]=n
00B2      311  JB5       equ  $B2      ; jump if reg-A bit 5 set
00B3      312  JMP.P.A    equ  $B3      ; jump on reg-A, PC=PC+A
00B4      313  CALL5     equ  $B4      ; call page 5 subrtn, PC=5nn
00B5      314  CPLF1     equ  $B5      ; complement flag #1, F1=-F1
00B6      315  JF0       equ  $B6      ; jump if flag #0 set
00B8      316  MOVR0     equ  $B8      ; move immediate to R0, R0=n
00B9      317  MOVR1     equ  $B9      ; move immediate to R1, R1=n
00BA      318  MOVR2     equ  $BA      ; move immediate to R2, R2=n
00BB      319  MOVR3     equ  $BB      ; move immediate to R3, R3=n
00BC      320  MOVR4     equ  $BC      ; move immediate to R4, R4=n
00BD      321  MOVR5     equ  $BD      ; move immediate to R5, R5=n
00BE      322  MOVR6     equ  $BE      ; move immediate to R6, R6=n
00BF      323  MOVR7     equ  $BF      ; move immediate to R7, R7=n
0800      324  ;
00C4      325  JMP6      equ  $C4      ; unconditional jump, PC=6nn
00C5      326  SELRB0    equ  $C5      ; select register bank 0
00C6      327  JZ        equ  $C6      ; jump if reg-A is zero
00C7      328  MOVAPSW    equ  $C7      ; move from PSW, A=PSW
00C8      329  DECR0     equ  $C8      ; decrement R0, R0=R0-1
00C9      330  DECR1     equ  $C9      ; decrement R1, R1=R1-1
00CA      331  DECR2     equ  $CA      ; decrement R2, R2=R2-1
00CB      332  DECR3     equ  $CB      ; decrement R3, R3=R3-1
00CC      333  DECR4     equ  $CC      ; decrement R4, R4=R4-1
00CD      334  DECR5     equ  $CD      ; decrement R5, R5=R5-1
00CE      335  DECR6     equ  $CE      ; decrement R6, R6=R6-1
00CF      336  DECR7     equ  $CF      ; decrement R7, R7=R7-1
0800      337  ;
00D0      338  XRLA.R0    equ  $D0      ; EOR memory, A=A^[R0]
00D1      339  XRLA.R1    equ  $D1      ; EOR memory, A=A^[R1]
00D2      340  JB6       equ  $D2      ; jump if reg-A bit 6 set
00D3      341  XRLA      equ  $D3      ; EOR immediate, A=A^n
00D4      342  CALL6     equ  $D4      ; call page 6 subrtn, PC=6nn
00D5      343  SELRB1    equ  $D5      ; select register bank 1
00D7      344  MOVPSWA    equ  $D7      ; move to PSW, PSW=A
00D8      345  XRLAR0     equ  $D8      ; EOR R0, A=A^R0
00D9      346  XRLAR1     equ  $D9      ; EOR R1, A=A^R1
00DA      347  XRLAR2     equ  $DA      ; EOR R2, A=A^R2
00DB      348  XRLAR3     equ  $DB      ; EOR R3, A=A^R3
00DC      349  XRLAR4     equ  $DC      ; EOR R4, A=A^R4
00DD      350  XRLAR5     equ  $DD      ; EOR R5, A=A^R5
00DE      351  XRLAR6     equ  $DE      ; EOR R6, A=A^R6
00DF      352  XRLAR7     equ  $DF      ; EOR R7, A=A^R7
0800      353  ;
00E3      354  MOVP3A.A   equ  $E3      ; read Page 3, A=[0x300+A]
00E4      355  JMP7      equ  $E4      ; unconditional jump, PC=7nn
00E5      356  SELMB0    equ  $E5      ; select memory bank 0
00E6      357  JNC       equ  $E6      ; jump if carry clear
00E7      358  RLA       equ  $E7      ; rotate left, A=A<-
00E8      359  DJNZR0     equ  $E8      ; dec R0, jump if not zero
00E9      360  DJNZR1     equ  $E9      ; dec R1, jump if not zero
00EA      361  DJNZR2     equ  $EA      ; dec R2, jump if not zero
00EB      362  DJNZR3     equ  $EB      ; dec R3, jump if not zero
00EC      363  DJNZR4     equ  $EC      ; dec R4, jump if not zero
00ED      364  DJNZR5     equ  $ED      ; dec R5, jump if not zero
00EE      365  DJNZR6     equ  $EE      ; dec R6, jump if not zero

```

```
00EF          366  DJNZR7      equ  $EF          ; dec R7, jump if not zero
0800          367  ;
00F0          368  MOVA.R0     equ  $F0          ; move from memory, A=[R0]
00F1          369  MOVA.R1     equ  $F1          ; move from memory, A=[R1]
00F2          370  JB7        equ  $F2          ; jump if reg-A bit 7 set
00F4          371  CALL7      equ  $F4          ; call page 7 subrtn, PC=7nn
00F5          372  SELMB1     equ  $F5          ; select memory bank 1
00F6          373  JC         equ  $F6          ; jump if carry set
00F7          374  RLCA       equ  $F7          ; rotate left+carry, A=AC<-
00F8          375  MOVAR0     equ  $F8          ; move from R0, A=R0
00F9          376  MOVAR1     equ  $F9          ; move from R1, A=R1
00FA          377  MOVAR2     equ  $FA          ; move from R2, A=R2
00FB          378  MOVAR3     equ  $FB          ; move from R3, A=R3
00FC          379  MOVAR4     equ  $FC          ; move from R4, A=R4
00FD          380  MOVAR5     equ  $FD          ; move from R5, A=R5
00FE          381  MOVAR6     equ  $FE          ; move from R6, A=R6
00FF          382  MOVAR7     equ  $FF          ; move from R7, A=R7
0800          383  ;
0800          384  ;
0800          385              icl  "PAGE0.L"
```

```
LLOAD PAGE0.L,A$4000
```

```

0800          1          ttl "PPB Source Code, PAGE0.L"
0800          2          ;
0800          3          ;
0800          4          ; PAGE0.L
0800          5          ;
0800          6          ;
0000          7          org $0000
0000          8          obj $1000
0000          9          usr
0000         10          ;
0000         11          ;
0000         12          ; The following hardware established vectors require an
0000         13          ; associated routine or handler. Those hardware vector
0000         14          ; addresses are:
0000         15          ;
0000         16          ; 0x000 - RESET Interrupt Vector (needs routine)
0000         17          ;
0000         18          ; 0x003 - External Interrupt Vector (needs handler)
0000         19          ;
0000         20          ; 0x007 - Internal Timer Interrupt Vector (needs handler)
0000         21          ;
0000         22          ;
0000         23          PAGE0:
0000         24          ;
0000         25          RESET:
0000 65         26          byt STOPTC          ; stop timer/counter
0001         27          ;
0001 04 33      28          byt JMP0,MAIN        ; jump to MAIN
0003         29          ;
0003         30          ;
0003         31          EXTIRQ:
0003 15         32          byt DISI          ; disable external interrupts
0004 65         33          byt STOPTC          ; stop timer/counter
0005         34          ;
0005 04 42      35          byt JMP0,SAV2MEM     ; jump to SAV2MEM
0007         36          ;
0007         37          ;
0007         38          TIMRIRQ:
0007 15         39          byt DISI          ; disable external interrupts
0008 65         40          byt STOPTC          ; stop timer/counter
0009         41          ;
0009         42          ;
0009         43          ; Select Register Bank 1 and save reg-A.
0009         44          ;
0009 D5         45          byt SELRB1          ; select register bank 1
000A         46          ;
000A A9        47          byt MOVR1A          ; R1=A
000B         48          ;
000B         49          ;
000B         50          ; Load the timer with the timer count and start the timer.
000B         51          ;
000B 23 CF      52          byt MOVA,TIME1      ; A=#TIME1
000D 62        53          byt MOVTA          ; T=A
000E         54          ;
000E 55        55          byt STRTT          ; start timer
000F         56          ;
000F         57          ;
000F         58          ; Internal Timer Interrupt Vector Handler.
000F         59          ;
000F         60          ; This interrupt is generated when the Internal Timer

```

```

000F      61 ; counts up to 0xFF and overflows to 0x00.
000F      62 ;
000F      63 ; XTAL = 6.0 MHz, Tcy = 15 / XTAL = 2.5 usec.
000F      64 ;
000F      65 ; Timer Prescaler = 32 * Tcy = 80 usec.
000F      66 ;
000F      67 ; If Timer is set to 0xCF, Interrupt = 49 * 80 = 3.920 msec
000F      68 ;
000F      69 ; The instructions to reset the Internal Timer require 8
000F      70 ; cycles.
000F      71 ;
000F      72 ; Timer Reset = 8 * Tcy = 20.0 usec. = 0.020 msec.
000F      73 ;
000F      74 ; Total Timer Duration = 3.920 + 0.020 = 3.940 msec.
000F      75 ;
000F      76 ; The Internal Timer is primarily used to refresh memory.
000F      77 ; It is also used to time events such as how long the Copy
000F      78 ; Button is pressed, the Ready LED flash rate for Copy
000F      79 ; Count, and a debounce delay after the Copy Button has
000F      80 ; been released.
000F      81 ;
000F      82 ; A RAS refresh is performed to insure memory integrity
000F      83 ; every Timer Interrupt. This should appear as a
000F      84 ; background task during normal data Import/Output
000F      85 ; activity.
000F      86 ;
000F      87 ; The Fujitsu data sheet for MB81257-15 NMOS Dynamic
000F      88 ; Random Access Memory stipulates a maximum time between
000F      89 ; refresh to be 4.0 msec.
000F      90 ;
000F      91 ; Enable the PPB external memory.
000F      92 ;
000F B8 00      93          byt MOVR0,ZERO          ; R0=#ZERO
0011 99 7F      94          byt ANLP1,PMEMENBL    ; P1=P1&#PMEMENBL
0013            95 ;
0013            96 ;
0013            97 ; Set ADR08 then check the state of the System Flag Refresh
0013            98 ; Counter ADR08 bit and branch if set.
0013            99 ;
0013 8A 01      100          byt ORLP2,ADR08ON      ; set ADR08
0015            101 ;
0015 FB        102          byt MOVAR3            ; A=R3
0016 12 1A      103          byt JB0,RF01          ; jump if bit 0 set
0018            104 ;
0018 9A FE      105          byt ANLP2,ADR08OFF     ; clear ADR08
001A            106 ;
001A            107 ;
001A            108 ; Write the Refresh Counter LSB address.
001A            109 ;
001A            110 RF01:
001A FD        111          byt MOVAR5            ; A=R5
001B 90        112          byt MOVX.R0A          ; [R0]=A
001C            113 ;
001C            114 ;
001C            115 ; Disable the PPB external memory.
001C            116 ;
001C 89 80      117          byt ORLP1,PMEMDSBL    ; P1=P1|#PMEMDSBL
001E            118 ;
001E            119 ;
001E            120 ; Increment the Refresh Counter LSB and test for zero.
001E            121 ;

```

```
001E 1D          122          byt INCR5          ; R5=R5+1
001F             123          ;
001F FD          124          byt MOVAR5          ; A=R5
0020 96 26       125          byt JNZ,RF02         ; jump if A>0
0022             126          ;
0022             127          ;
0022             128          ; Toggle the System Flag Refresh Counter ADR08 bit.
0022             129          ;
0022 FB          130          byt MOVAR3          ; A=R3
0023 D3 01       131          byt XRLA,ADR08TGL    ; A=A^#ADR08TGL
0025 AB          132          byt MOVR3A          ; R3=A
0026             133          ;
0026             134          ;
0026             135          ; Increment the Timer Count LSB and test for zero.
0026             136          ;
0026             137          RF02:
0026 1E          138          byt INCR6          ; R6=R6+1
0027             139          ;
0027 FE          140          byt MOVAR6          ; A=R6
0028 96 2B       141          byt JNZ,RF03         ; jump if A>0
002A             142          ;
002A             143          ;
002A             144          ; Increment the Timer Count MSB.
002A             145          ;
002A 1F          146          byt INCR7          ; R7=R7+1
002B             147          ;
002B             148          ;
002B             149          ; Enable external interrupts if the EXTIRQ State Flag is
002B             150          ; On.
002B             151          ;
002B             152          RF03:
002B FB          153          byt MOVAR3          ; A=R3
002C F2 30       154          byt JB7,RF04         ; jump if bit 7 set
002E             155          ;
002E             156          ;
002E             157          ; Restore reg-A and return from interrupt.
002E             158          ;
002E F9          159          byt MOVAR1          ; A=R1
002F             160          ;
002F 93          161          byt RETR           ; return and restore status
0030             162          ;
0030             163          ;
0030             164          ; Restore reg-A, enable external interrupts, and return
0030             165          ; from interrupt.
0030             166          ;
0030             167          RF04:
0030 F9          168          byt MOVAR1          ; A=R1
0031             169          ;
0031 05          170          byt ENI           ; enable external interrupts
0032             171          ;
0032 93          172          byt RETR           ; return and restore status
0033             173          ;
0033             174          ;
0033             175          ;
0033             176          ; RESET Interrupt Vector Routine.
0033             177          ;
0033             178          ; This routine is entered when the PPB is first powered up
0033             179          ; and whenever the Clear Button is pressed.
0033             180          ;
0033             181          ; RESET performs the following functions:
0033             182          ;
```

```

0033      183 ; 1. Sets program counter to zero.
0033      184 ; 2. Set stack pointer to zero.
0033      185 ; 3. Selects Register Bank 0.
0033      186 ; 4. Selects Memory Bank 0.
0033      187 ; 5. Sets BUS to high impedance state.
0033      188 ; 6. Sets Port 1 and Port 2 to input mode.
0033      189 ; 7. Disables interrupts (Timer and External).
0033      190 ; 8. Stops Timer.
0033      191 ; 9. Clears Timer Flag.
0033      192 ; 10. Clears F0 and F1 user flags.
0033      193 ; 11. Disables clock output from T0.
0033      194 ;
0033      195 ; Initialize the PPB.
0033      196 ;
0033      197 MAIN:
0033 15      198          byt DISI          ; disable external interrupts
0034 35      199          byt DISTI         ; disable timer interrupts
0035      200 ;
0035 94 00    201          byt CALL4,INIT      ; call INIT
0037      202 ;
0037 05      203          byt ENI           ; enable external interrupts
0038      204 ;
0038      205 ;
0038      206 ; Master program loop.  Manage Copy Button and Outport
0038      207 ; Connection.
0038      208 ;
0038      209 MAIN01:
0038 34 00    210          byt CALL1,CHECKT0      ; call CHECKT0
003A      211 ;
003A      212 ;
003A      213 ; Read Port 1 and check state of Outport Data Ready bit
003A      214 ; (1=Ready).
003A      215 ;
003A      216 ;
003A      217 ;          byt CALL0,TESTLED1    ; call TESTLED1
003A      218 ;
003A      219 ;
003A 09      220          byt INAP1          ; A=P1
003B 37      221          byt CPLA           ; A=-A
003C 92 38    222          byt JB4,MAIN01      ; jump if bit 4 set (clear)
003E      223 ;
003E      224 ;
003E      225 ;          byt CALL0,TESTLED2    ; call TESTLED2
003E      226 ;
003E      227 ;
003E      228 ; Send PPB memory data to the Outport Connection.
003E      229 ;
003E 54 00    230          byt CALL2,MEM2OUT      ; call MEM2OUT
0040      231 ;
0040 04 38    232          byt JMP0,MAIN01      ; jump to MAIN01
0042      233 ;
0042      234 ;
0042      235 ;
0042      236 ; External Interrupt Vector Handler.
0042      237 ;
0042      238 ; This interrupt is generated by the PPB Inport Connection
0042      239 ; strobe signaling that input data is available to be saved
0042      240 ; to the PPB external memory.
0042      241 ;
0042      242 ; Select Register Bank 0 and save reg-A.
0042      243 ;

```

```

0042      244 SAV2MEM:
0042 C5      245      byt SELRB0      ; select register bank 0
0043      246      ;
0043 A9      247      byt MOVR1A      ; R1=A
0044      248      ;
0044      249      ;
0044      250      ; Enable the processor data latch, read the input data,
0044      251      ; and disable the latch.
0044      252      ;
0044 B8 00    253      byt MOVR0,ZERO    ; R0=#ZERO
0046 99 FD    254      byt ANLP1,PDATENBL ; P1=P1&#PDATENBL
0048      255      ;
0048 80      256      byt MOVXA.R0     ; A=[R0]
0049 AA      257      byt MOVR2A     ; R2=A
004A      258      ;
004A 89 02   259      byt ORLP1,PDATDSBL ; P1=P1|#PDATDSBL
004C      260      ;
004C      261      ;
004C      262      ; Generate a 5 usec Inport data strobe.
004C      263      ;
004C 9A EF    264      byt ANLP2,IDATENBL ; P2=P2&#IDATENBL
004E      265      ;
004E 00      266      byt NOP        ; no operation
004F 00      267      byt NOP        ; no operation
0050      268      ;
0050 8A 10    269      byt ORLP2,IDATDSBL ; P2=P2|#IDATDSBL
0052      270      ;
0052      271      ;
0052      272      ; Enable the PPB external memory.
0052      273      ;
0052 99 7F    274      byt ANLP1,PMEMENBL ; P1=P1&#PMEMENBL
0054      275      ;
0054      276      ;
0054      277      ; Set ADR08 then check the state of the write block number
0054      278      ; LSB and branch if set.
0054      279      ;
0054 8A 01    280      byt ORLP2,ADR08ON  ; set ADR08
0056      281      ;
0056 FB      282      byt MOVAR3      ; A=R3
0057 92 5B    283      byt JB4,EI01     ; jump if LSB set
0059      284      ;
0059 9A FE    285      byt ANLP2,ADR08OFF ; clear ADR08
005B      286      ;
005B      287      ;
005B      288      ; Write the write data RAS address.
005B      289      ;
005B      290      EI01:
005B FE      291      byt MOVAR6      ; A=R6
005C 90      292      byt MOVX.R0A     ; [R0]=A
005D      293      ;
005D      294      ;
005D      295      ; Set ADR08 then check the state of the write block number
005D      296      ; MSB and branch if set.
005D      297      ;
005D 8A 01    298      byt ORLP2,ADR08ON  ; set ADR08
005F      299      ;
005F FB      300      byt MOVAR3      ; A=R3
0060 B2 64    301      byt JB5,EI02     ; jump if MSB set
0062      302      ;
0062 9A FE    303      byt ANLP2,ADR08OFF ; clear ADR08
0064      304      ;

```

```

0064      305 ;
0064      306 ; Write the write data CAS address.
0064      307 ;
0064      308 EI02:
0064 FF      309      byt MOVAR7      ; A=R7
0065 90      310      byt MOVX.R0A      ; [R0]=A
0066      311 ;
0066      312 ;
0066      313 ; Write the input data to memory.
0066      314 ;
0066 FA      315      byt MOVAR2      ; A=R2
0067 90      316      byt MOVX.R0A      ; [R0]=A
0068      317 ;
0068      318 ;
0068      319 ; Disable the PPB external memory and start the timer.
0068      320 ;
0068 89 80    321      byt ORLP1,PMEMDSBL ; P1=P1|#PMEMDSBL
006A      322 ;
006A 55      323      byt STRTT      ; start timer
006B      324 ;
006B      325 ;
006B      326 ; Increment the write data RAS address and test for zero.
006B      327 ;
006B 1E      328      byt INCR6      ; R6=R6+1
006C      329 ;
006C FE      330      byt MOVAR6      ; A=R6
006D 96 96   331      byt JNZ,EI04      ; jump if A>0
006F      332 ;
006F      333 ;
006F      334 ; Increment the write data CAS address and test for zero.
006F      335 ;
006F 1F      336      byt INCR7      ; R7=R7+1
0070      337 ;
0070 FF      338      byt MOVAR7      ; A=R7
0071 96 83   339      byt JNZ,EI03      ; jump if A>0
0073      340 ;
0073      341 ;
0073      342 ; Increment the write block number and mask.
0073      343 ;
0073 FB      344      byt MOVAR3      ; A=R3
0074 03 10   345      byt ADDA,ADDWRBLK ; A=A+#ADDWRBLK
0076 53 33   346      byt ANLA,MSKRWBLK ; A=A&#MSKRWBLK
0078 AB      347      byt MOVR3A      ; R3=A
0079      348 ;
0079      349 ;
0079      350 ; If the write block number is zero then data has
0079      351 ; overflowed external memory.
0079      352 ;
0079 53 30     353      byt ANLA,MSKWRBLK ; A=A&#MSKWRBLK
007B 96 83   354      byt JNZ,EI03      ; jump if A>0
007D      355 ;
007D D5      356      byt SELRB1      ; select register bank 1
007E      357 ;
007E FB      358      byt MOVAR3      ; A=R3
007F 43 08   359      byt ORLA,OVFSFON ; A=A|#OVFSFON
0081 AB      360      byt MOVR3A      ; R3=A
0082      361 ;
0082 C5      362      byt SELRB0      ; select register bank 0
0083      363 ;
0083      364 ;
0083      365 ; Check if the write data CAS address has reached the read

```

```

0083      366 ; data CAS address. This will occur if the Outport
0083      367 ; Connection is offline or when the Outport data rate is
0083      368 ; substantially slower than the Inport data rate.
0083      369 ;
0083      370 EI03:
0083 FF      371      byt MOVAR7          ; A=R7
0084 DD      372      byt XRLAR5          ; A=A^R5
0085 96 96    373      byt JNZ,EI04          ; jump if A>0
0087      374 ;
0087      375 ;
0087      376 ; Check if the write block number is the same as the read
0087      377 ; block number.
0087      378 ;
0087 FB      379      byt MOVAR3          ; A=R3
0088 47      380      byt SWAPA            ; swap nibbles
0089 DB      381      byt XRLAR3          ; A=A^R3
008A 96 96    382      byt JNZ,EI04          ; jump if A>0
008C      383 ;
008C      384 ;
008C      385 ; No more data can be saved to PPB memory or previously
008C      386 ; saved data will be overwritten. Turn the EXTIRQ State
008C      387 ; Flag Off and the Ready LED Off. Leave external
008C      388 ; interrupts disabled.
008C      389 ;
008C D5      390      byt SELRB1            ; select register bank 1
008D      391 ;
008D FB      392      byt MOVAR3          ; A=R3
008E 53 7F    393      byt ANLA,IRQSFOFF      ; A=A&#177;IRQSFOFF
0090 AB      394      byt MOVR3A          ; R3=A
0091      395 ;
0091 C5      396      byt SELRB0            ; select register bank 0
0092      397 ;
0092 9A F7    398      byt ANLP2,RDLEDOFF      ; P2=P2&#177;RDLEDOFF
0094      399 ;
0094 F9      400      byt MOVAR1          ; A=R1
0095      401 ;
0095 93      402      byt RETR            ; return and restore status
0096      403 ;
0096      404 ;
0096      405 ; Restore reg-A and return from interrupt.
0096      406 ;
0096      407 EI04:
0096 F9      408      byt MOVAR1          ; A=R1
0097      409 ;
0097 05      410      byt ENI            ; enable external interrupts
0098      411 ;
0098 93      412      byt RETR            ; return and restore status
0099      413 ;
0099      414 ;
0099      415 ; Toggle Port 2 Test LED 1 bit on zero crossing of Test
0099      416 ; LED 1 memory location in User RAM.
0099      417 ;
0099      418 TESTLED1:
0099 B8 22      419      byt MOVR0,LED1NDX      ; R0=#LED1NDX
009B      420 ;
009B F0      421      byt MOVA.R0          ; A=[R0]
009C 96 A4    422      byt JNZ,TESTL01          ; jump if A>0
009E      423 ;
009E      424 ;
009E      425 ; Read Port 2, toggle Test LED 1 bit, save to Port 2, and
009E      426 ; restore reg-A with zero.

```

```

009E      427 ;
009E 0A      428      byt INAP2      ; A=P2
009F D3 02   429      byt XRLA,LED1TGL ; A=A^#LED1TGL
00A1 3A      430      byt OUTLP2A     ; P2=A
00A2      431 ;
00A2 23 00   432      byt MOVA,ZERO      ; A=#ZERO
00A4      433 ;
00A4      434 ;
00A4      435 ;
00A4      436 ; Increment Test LED 1 memory location.
00A4      437 ;
00A4      438 TESTL01:
00A4 17      439      byt INCA      ; A=A+1
00A5 A0      440      byt MOV.R0A     ; [R0]=A
00A6      441 ;
00A6 B8 00   442      byt MOVR0,ZERO     ; R0=#ZERO
00A8      443 ;
00A8      444 ;
00A8      445 ; Return to caller.
00A8      446 ;
00A8 83      447      byt RET      ; return from subroutine
00A9      448 ;
00A9      449 ;
00A9      450 ; Toggle Port 2 Test LED 2 bit on zero crossing of Test
00A9      451 ; LED 2 memory location in User RAM.
00A9      452 ;
00A9      453 TESTLED2:
00A9 B8 23   454      byt MOVR0,LED2NDX     ; R0=#LED2NDX
00AB      455 ;
00AB F0      456      byt MOVA.R0      ; A=[R0]
00AC 96 B4   457      byt JNZ,TESTL02    ; jump if A>0
00AE      458 ;
00AE      459 ;
00AE      460 ; Read Port 2, toggle Test LED 2 bit, save to Port 2, and
00AE      461 ; restore reg-A with zero.
00AE      462 ;
00AE 0A      463      byt INAP2      ; A=P2
00AF D3 04   464      byt XRLA,LED2TGL     ; A=A^#LED2TGL
00B1 3A      465      byt OUTLP2A     ; P2=A
00B2      466 ;
00B2 23 00   467      byt MOVA,ZERO      ; A=#ZERO
00B4      468 ;
00B4      469 ;
00B4      470 ; Increment Test LED 2 memory location.
00B4      471 ;
00B4      472 TESTL02:
00B4 17      473      byt INCA      ; A=A+1
00B5 A0      474      byt MOV.R0A     ; [R0]=A
00B6      475 ;
00B6 B8 00   476      byt MOVR0,ZERO     ; R0=#ZERO
00B8      477 ;
00B8      478 ;
00B8      479 ; Return to caller.
00B8      480 ;
00B8 83      481      byt RET      ; return from subroutine
00B9      482 ;
00B9      483 ;
00B9      484      dfs $100-*-$100**/$100,NEGONE
0100      485 ;
0100      486 ;
0100      487      icl "PAGE1.L"

```



```

0100      1          ttl "PPB Source Code, PAGE1.L"
0100      2      ;
0100      3      ;
0100      4      ; PAGE1.L
0100      5      ;
0100      6      ;
0100      7  PAGE1:
0100      8      ;
0100      9      ;
0100     10      ; This routine checks if the Copy Button has been pressed.
0100     11      ;
0100     12      ; Pause Mode: a momentary press (less than two seconds)
0100     13      ; of the Copy Button will enable Pause Mode. The PPB will
0100     14      ; stop sending data to its Output connection but continue
0100     15      ; to receive data to its Input connection until its memory
0100     16      ; is full. Another momentary press of the Copy Button
0100     17      ; will cancel Pause Mode and data will once again be sent
0100     18      ; to the Output connection.
0100     19      ;
0100     20      ; Copy Mode: press and hold the Copy Button for at least
0100     21      ; 1.5 seconds until the Ready LED begins to flash. Let the
0100     22      ; Ready LED flash for as many times as the desired number
0100     23      ; of copies. Release the Copy Button. If the entire file
0100     24      ; cannot fit in the PPB memory Copy Mode will be disabled.
0100     25      ;
0100     26      ; Exit if the Copy Button is not being pressed.
0100     27      ;
0100     28  CHECKT0:
0100 36 4A     29      byt JT0,CT09          ; jump if T0=1
0102     30      ;
0102     31      ;
0102     32      ; Call the debounce subroutine. Exit if the Copy Button
0102     33      ; is released.
0102     34      ;
0102 34 4B     35      byt CALL1,DEBOUNCE    ; call DEBOUNCE
0104     36      ;
0104 36 4A     37      byt JT0,CT09          ; jump if T0=1
0106     38      ;
0106     39      ;
0106     40      ; Turn Ready LED Off and load Timer Counter with a 1.5
0106     41      ; second interval.
0106     42      ;
0106     43      ; byt SELRB1              ; select register bank 1
0106     44      ;
0106 9A F7     45      byt ANLP2,RDLEDOFF    ; P2=P2&#RDLEDOFF
0108     46      ;
0108 BE 80     47      byt MOVR6,TIME4      ; R6=#TIME4 (LSB)
010A BF FE     48      byt MOVR7,TIME4/256 ; R7=#TIME4 (MSB)
010C     49      ;
010C     50      ;
010C     51      ; If Copy Button is released before Timer Counter is zero,
010C     52      ; branch to handle Pause Mode.
010C     53      ;
010C     54  CT01:
010C 36 3D     55      byt JT0,CT06          ; jump if T0=1
010E     56      ;
010E FF       57      byt MOVR7            ; A=R7
010F 96 0C     58      byt JNZ,CT01         ; jump if A>0
0111     59      ;
0111     60      ;

```

```

0111      61 ; Handle Copy Mode. Exit if Overflow State Flag is set.
0111      62 ;
0111 FB    63      byt MOVAR3      ; A=R3
0112 72 43  64      byt JB3,CT07    ; jump if bit 3 set
0114      65 ;
0114      66 ;
0114      67 ; Disable external interrupts, disable the EXTIRQ State
0114      68 ; Flag, and initialize Copy Count.
0114      69 ;
0114 15     70      byt DISI        ; disable external interrupts
0115      71 ;
0115 FB    72      byt MOVAR3      ; A=R3
0116 53 7F  73      byt ANLA,IRQSF0FF ; A=A&#177;IRQSF0FF
0118 AB    74      byt MOVR3A      ; R3=A
0119      75 ;
0119 BC 00  76      byt MOVR4,ZERO   ; R4=#ZERO
011B      77 ;
011B      78 ;
011B      79 ; Need to count flashes of the Ready LED. Increment Copy
011B      80 ; Count, turn Ready LED On, and load Timer Counter with a
011B      81 ; 1 second interval.
011B      82 ;
011B      83 CT02:
011B 1C    84      byt INCR4        ; R4=R4+1
011C      85 ;
011C 8A 08  86      byt ORLP2,RDLEDON ; P2=P2|&#177;RDLEDON
011E      87 ;
011E BE 00  88      byt MOVR6,TIME3   ; R6=#TIME3 (LSB)
0120 BF FF  89      byt MOVR7,TIME3/256 ; R7=#TIME3 (MSB)
0122      90 ;
0122      91 ;
0122      92 ; If Copy Button is released before Timer Counter is zero,
0122      93 ; branch to Copy Mode exit.
0122      94 ;
0122      95 CT03:
0122 36 34  96      byt JT0,CT05      ; jump if T0=1
0124      97 ;
0124 FF    98      byt MOVAR7      ; A=R7
0125 96 22  99      byt JNZ,CT03     ; jump if A>0
0127     100 ;
0127     101 ;
0127     102 ; Turn Ready LED Off and load Timer Counter with a 1
0127     103 ; second interval.
0127     104 ;
0127 9A F7  105      byt ANLP2,RDLEDOFF ; P2=P2&#177;RDLEDOFF
0129     106 ;
0129 BE 00  107      byt MOVR6,TIME3   ; R6=#TIME3 (LSB)
012B BF FF  108      byt MOVR7,TIME3/256 ; R7=#TIME3 (MSB)
012D     109 ;
012D     110 ;
012D     111 ; If Copy Button is released before Timer Counter is zero,
012D     112 ; branch to Copy Mode exit.
012D     113 ;
012D     114 CT04:
012D 36 34  115      byt JT0,CT05      ; jump if T0=1
012F     116 ;
012F FF    117      byt MOVAR7      ; A=R7
0130 96 2D  118      byt JNZ,CT04     ; jump if A>0
0132     119 ;
0132 C6 1B  120      byt JZ,CT02       ; jump if A=0
0134     121 ;

```

```
0134      122 ;
0134      123 ; Copy button is released. Call the debounce subroutine.
0134      124 ;
0134      125 CT05:
0134 34 4B      126      byt CALL1,DEBOUNCE ; call DEBOUNCE
0136      127 ;
0136      128 ;
0136      129 ; Enable the Copy State Flag and turn the Ready LED Off.
0136      130 ;
0136 FB      131      byt MOVAR3 ; A=R3
0137 43 20      132      byt ORLA,CPYSFON ; A=A|#CPYSFON
0139 AB      133      byt MOVR3A ; R3=A
013A      134 ;
013A 9A F7      135      byt ANLP2,RDLEDOFF ; P2=P2&#RDLEDOFF
013C      136 ;
013C      137 ;
013C      138 ; Return to caller.
013C      139 ;
013C 83      140      byt RET ; return from subroutine
013D      141 ;
013D      142 ;
013D      143 ; Handle Pause Mode. Copy button is released. Call the
013D      144 ; debounce subroutine and toggle the Pause State Flag.
013D      145 ;
013D      146 CT06:
013D 34 4B      147      byt CALL1,DEBOUNCE ; call DEBOUNCE
013F      148 ;
013F FB      149      byt MOVAR3 ; A=R3
0140 D3 40      150      byt XRLA,PAUSFTGL ; A=A^#PAUSFTGL
0142 AB      151      byt MOVR3A ; R3=A
0143      152 ;
0143      153 ;
0143      154 ; If the Pause State Flag is Off turn the Ready LED On or
0143      155 ; if the Pause State Flag is On turn the Ready LED Off.
0143      156 ;
0143      157 CT07:
0143 D2 48      158      byt JB6,CT08 ; jump if bit 6 set
0145      159 ;
0145 8A 08      160      byt ORLP2,RDLEDON ; P2=P2|#RDLEDON
0147      161 ;
0147      162 ;
0147      163 ; Return to caller.
0147      164 ;
0147 83      165      byt RET ; return from subroutine
0148      166 ;
0148      167 ;
0148      168 CT08:
0148 9A F7      169      byt ANLP2,RDLEDOFF ; P2=P2&#RDLEDOFF
014A      170 ;
014A      171 ;
014A      172 ; Return to caller.
014A      173 ;
014A      174 CT09:
014A 83      175      byt RET ; return from subroutine
014B      176 ;
014B      177 ;
014B      178 ; Load Timer Counter with a 64 msec debounce interval.
014B      179 ;
014B      180 DEBOUNCE:
014B D5      181      byt SELRB1 ; select register bank 1
014C      182 ;
```

```
014C BE F0      183      byt MOVR6,TIME2      ; R6=#TIME2 (LSB)
014E BF FF      184      byt MOVR7,TIME2/256 ; R7=#TIME2 (MSB)
0150            185      ;
0150            186      ;
0150            187      ; Wait for debounce interval to complete.
0150            188      ;
0150            189      DB01:
0150 00          190      byt NOP                ; no operation
0151            191      ;
0151 FF          192      byt MOVAR7             ; A=R7
0152 96 50       193      byt JNZ,DB01           ; jump if A>0
0154            194      ;
0154            195      ;
0154            196      ; Return to caller.
0154            197      ;
0154 83          198      byt RET                ; return from subroutine
0155            199      ;
0155            200      ;
0155            201      ; Load Timer Counter with a one second time interval.
0155            202      ;
0155            203      WAIT1SEC:
0155 D5          204      byt SELRB1             ; select register bank 1
0156            205      ;
0156 BE 00        206      byt MOVR6,TIME3       ; R6=#TIME3 (LSB)
0158 BF FF        207      byt MOVR7,TIME3/256 ; R7=#TIME3 (MSB)
015A            208      ;
015A            209      ;
015A            210      ; Wait for time interval to complete.
015A            211      ;
015A            212      W1S01:
015A 00          213      byt NOP                ; no operation
015B            214      ;
015B FF          215      byt MOVAR7             ; A=R7
015C 96 5A       216      byt JNZ,W1S01         ; jump if A>0
015E            217      ;
015E            218      ;
015E            219      ; Return to caller.
015E            220      ;
015E 83          221      byt RET                ; return from subroutine
015F            222      ;
015F            223      ;
015F            224      dfs $100-*-$100**/$100,NEGONE
0200            225      ;
0200            226      ;
0200            227      icl "PAGE2.L"
```

LLOAD PAGE2.L,A\$4000

```

0200          1          ttl "PPB Source Code, PAGE2.L"
0200          2          ;
0200          3          ;
0200          4          ; PAGE2.L
0200          5          ;
0200          6          ;
0200          7          PAGE2:
0200          8          ;
0200          9          ;
0200         10          ; This routine sends stored data to the PPB Output
0200         11          ; connection and manages Copy Mode if it has been enabled.
0200         12          ;
0200         13          ; Exit if Pause Mode is enabled.
0200         14          ;
0200         15          MEM2OUT:
0200 D5         16          byt SELRB1          ; select register bank 1
0201         17          ;
0201 FB         18          byt MOVAR3          ; A=R3
0202 D2 53      19          byt JB6,MO05        ; jump if bit 6 set
0204         20          ;
0204         21          ;
0204         22          ; If the read data RAS address does not equal the write
0204         23          ; data RAS address, branch to output data.
0204         24          ;
0204 C5         25          byt SELRB0          ; select register bank 0
0205         26          ;
0205 FC         27          byt MOVAR4          ; A=R4
0206 DE         28          byt XRLAR6          ; A=A^R6
0207 96 12      29          byt JNZ,MO01        ; jump if A>0
0209         30          ;
0209         31          ;
0209         32          ; If the read data CAS address does not equal the write
0209         33          ; data CAS address, branch to output data.
0209         34          ;
0209 FD         35          byt MOVAR5          ; A=R5
020A DF         36          byt XRLAR7          ; A=A^R7
020B 96 12      37          byt JNZ,MO01        ; jump if A>0
020D         38          ;
020D         39          ;
020D         40          ; If the read block number is the same as the write block
020D         41          ; number, branch to handle Copy Mode if enabled.
020D         42          ;
020D FB         43          byt MOVAR3          ; A=R3
020E 47         44          byt SWAPA          ; swap nibbles
020F DB         45          byt XRLAR3          ; A=A^R3
0210 C6 5D      46          byt JZ,MO09         ; jump if A=0
0212         47          ;
0212         48          ;
0212         49          ; Output data. Disable further external interrupts and
0212         50          ; stop the timer.
0212         51          ;
0212         52          MO01:
0212 15         53          byt DISI          ; disable external interrupts
0213 65         54          byt STOPTC         ; stop timer/counter
0214         55          ;
0214         56          ;
0214         57          ; Enable the PPB external memory.
0214         58          ;
0214 B8 00        59          byt MOVR0,ZERO      ; R0=#ZERO
0216 99 7F      60          byt ANLP1,PMEMENBL  ; P1=P1&#PMEMENBL

```

```

0218      61 ;
0218      62 ;
0218      63 ; Set ADR08 then check the state of the read block number
0218      64 ; LSB and branch if set.
0218      65 ;
0218 8A 01      66      byt ORLP2,ADR08ON      ; set ADR08
021A      67 ;
021A FB      68      byt MOVAR3      ; A=R3
021B 12 1F      69      byt JB0,MO02      ; jump if LSB set
021D      70 ;
021D 9A FE      71      byt ANLP2,ADR08OFF      ; clear ADR08
021F      72 ;
021F      73 ;
021F      74 ; Write the read data RAS address.
021F      75 ;
021F      76 MO02:
021F FC      77      byt MOVAR4      ; A=R4
0220 90      78      byt MOVX.R0A      ; [R0]=A
0221      79 ;
0221      80 ;
0221      81 ; Set ADR08 then check the state of the read block number
0221      82 ; MSB and branch if set.
0221      83 ;
0221 8A 01      84      byt ORLP2,ADR08ON      ; set ADR08
0223      85 ;
0223 FB      86      byt MOVAR3      ; A=R3
0224 32 28      87      byt JB1,MO03      ; jump if MSB set
0226      88 ;
0226 9A FE      89      byt ANLP2,ADR08OFF      ; clear ADR08
0228      90 ;
0228      91 ;
0228      92 ; Write the read data CAS address.
0228      93 ;
0228      94 MO03:
0228 FD      95      byt MOVAR5      ; A=R5
0229 90      96      byt MOVX.R0A      ; [R0]=A
022A      97 ;
022A      98 ;
022A      99 ; Read the data from memory.
022A      100 ;
022A 80      101      byt MOVXA.R0      ; A=[R0]
022B      102 ;      byt MOVR2A      ; R2=A
022B      103 ;
022B      104 ;
022B      105 ; Disable the PPB external memory.
022B      106 ;
022B 89 80      107      byt ORLP1,PMEMDSBL      ; P1=P1|#PMEMDSBL
022D      108 ;
022D      109 ;
022D      110 ; Enable the processor data latch, write the output data,
022D      111 ; and disable the latch.
022D      112 ;
022D 99 FD      113      byt ANLP1,PDATENBL      ; P1=P1&#PDATENBL
022F      114 ;
022F      115 ;      byt MOVAR2      ; A=R2
022F 90      116      byt MOVX.R0A      ; [R0]=A
0230      117 ;
0230 89 02      118      byt ORLP1,PDATDSBL      ; P1=P1|#PDATDSBL
0232      119 ;
0232      120 ;
0232      121 ; Generate a 5 usec Outport Latch Enable.

```

```

0232      122 ;
0232 99 FB   123      byt ANLP1,OLATENBL ; P1=P1&#OLATENBL
0234      124 ;
0234 00      125      byt NOP ; no operation
0235 00      126      byt NOP ; no operation
0236      127 ;
0236 89 04   128      byt ORLP1,OLATDSBL ; P1=P1|#OLATDSBL
0238      129 ;
0238      130 ;
0238      131 ; Start the timer.
0238      132 ;
0238 55      133      byt STRTT ; start timer
0239      134 ;
0239      135 ;
0239      136 ; Increment the read data RAS address and test for zero.
0239      137 ;
0239 1C      138      byt INCR4 ; R4=R4+1
023A      139 ;
023A FC      140      byt MOVAR4 ; A=R4
023B 96 47   141      byt JNZ,MO04 ; jump if A>0
023D      142 ;
023D      143 ;
023D      144 ; Increment the read data CAS address and test for zero.
023D      145 ;
023D 1D      146      byt INCR5 ; R5=R5+1
023E      147 ;
023E FD      148      byt MOVAR5 ; A=R5
023F 96 47   149      byt JNZ,MO04 ; jump if A>0
0241      150 ;
0241      151 ;
0241      152 ; Increment the read block number and mask.
0241      153 ;
0241 FB      154      byt MOVAR3 ; A=R3
0242 03 01   155      byt ADDA,ADDRDBLK ; A=A+#ADDRDBLK
0244 53 33   156      byt ANLA,MSKRWBLK ; A=A&#MSKRWBLK
0246 AB      157      byt MOVR3A ; R3=A
0247      158 ;
0247      159 ;
0247      160 ; Exit if Copy Mode is enabled.
0247      161 ;
0247      162 MO04:
0247 D5      163      byt SELRB1 ; select register bank 1
0248      164 ;
0248 FB      165      byt MOVAR3 ; A=R3
0249 B2 53   166      byt JB5,MO05 ; jump if bit 5 set
024B      167 ;
024B      168 ;
024B      169 ; Copy Mode is not enabled so check if the read data RAS is
024B      170 ; zero. If true then memory page is finished, so branch.
024B      171 ;
024B C5      172      byt SELRB0 ; select register bank 0
024C      173 ;
024C FC      174      byt MOVAR4 ; A=R4
024D C6 54   175      byt JZ,MO06 ; jump if A=0
024F      176 ;
024F      177 ;
024F      178 ; Memory page is not finished so branch if the EXTIRQ
024F      179 ; State Flag is On.
024F      180 ;
024F D5      181      byt SELRB1 ; select register bank 1
0250      182 ;

```

```

0250 FB          183          byt MOVAR3          ; A=R3
0251 F2 5B       184          byt JB7,MO08         ; jump if bit 7 set
0253            185          ;
0253            186          ;
0253            187          ; Return to caller.
0253            188          ;
0253            189          MO05:
0253 83           190          byt RET              ; return from subroutine
0254            191          ;
0254            192          ;
0254            193          ; A memory page of data has been sent to the Output
0254            194          ; connection so there is now room in external memory for
0254            195          ; more data from the Input connection. Turn the EXTIRQ
0254            196          ; State Flag On and the Ready LED On even if they are
0254            197          ; already On.
0254            198          ;
0254            199          MO06:
0254 D5           200          byt SELRB1            ; select register bank 1
0255            201          ;
0255            202          ;
0255            203          MO07:
0255 FB          204          byt MOVAR3          ; A=R3
0256 43 80       205          byt ORLA,IRQSFON      ; A=A|#IRQSFON
0258 AB          206          byt MOVR3A          ; R3=A
0259            207          ;
0259 8A 08       208          byt ORLP2,RDLEDON      ; P2=P2|#RDLEDON
025B            209          ;
025B            210          ;
025B            211          ; Enable external interrupts and return to caller.
025B            212          ;
025B            213          MO08:
025B 05         214          byt ENI              ; enable external interrupts
025C            215          ;
025C 83         216          byt RET              ; return from subroutine
025D            217          ;
025D            218          ;
025D            219          ; The read and write addresses are now the same. Check if
025D            220          ; Copy Mode is enabled and branch. Otherwise return to
025D            221          ; caller.
025D            222          ;
025D            223          MO09:
025D D5         224          byt SELRB1            ; select register bank 1
025E            225          ;
025E FB         226          byt MOVAR3          ; A=R3
025F 37         227          byt CPLA            ; A=-A
0260 B2 55      228          byt JB5,MO07         ; jump if bit 5 set (clear)
0262            229          ;
0262            230          ;
0262            231          ; Branch if copy count is not zero.
0262            232          ;
0262 FC         233          byt MOVAR4          ; A=R4
0263 96 6B      234          byt JNZ,MO10         ; jump if A>0
0265            235          ;
0265            236          ;
0265            237          ; Disable Copy State Flag and return to caller.
0265            238          ;
0265 FB         239          byt MOVAR3          ; A=R3
0266 53 DF      240          byt ANLA,CPYSFOFF      ; A=A&#CPYSFOFF
0268 AB         241          byt MOVR3A          ; R3=A
0269            242          ;
0269 44 55      243          byt JMP2,MO07         ; jump to MO07

```

```
026B      244 ;
026B      245 ;
026B      246 ; Begin another copy.  Decrement the copy count.
026B      247 ;
026B      248 MO10:
026B CC    249      byt DECR4          ; R4=R4-1
026C      250 ;
026C      251 ;
026C      252 ; Clear the read block number.
026C      253 ;
026C C5    254      byt SELRB0         ; select register bank 0
026D      255 ;
026D FB    256      byt MOVAR3         ; A=R3
026E 53 30 257      byt ANLA,CLRRDBLK  ; A=A&#CLRRDBLK
0270 AB    258      byt MOVR3A         ; R3=A
0271      259 ;
0271      260 ;
0271      261 ; Set the read data RAS and the read data CAS to zero.
0271      262 ;
0271 BC 00 263      byt MOVR4,ZERO      ; R4=#ZERO
0273 BD 00 264      byt MOVR5,ZERO      ; R5=#ZERO
0275      265 ;
0275 83    266      byt RET            ; return from subroutine
0276      267 ;
0276      268 ;
0276      269      dfs $100-*-$100**/$100,NEGONE
0300      270 ;
0300      271 ;
0300      272      icl "PAGE3.L"
```

LLOAD PAGE3.L,A\$4000

```
0300          1          ttl "PPB Source Code, PAGE3.L"
0300          2          ;
0300          3          ;
0300          4          ; PAGE3.L
0300          5          ;
0300          6          ;
0300          7          PAGE3:
0300          8          ;
0300          9          ;
0300         10          ; Memory Check Messages.
0300         11          ;
0300         12          ; Diag Messages.
0300         13          ;
0300         14          DMESG1:
0300 82         15          byt RTN2          ; 1 CR and 2 LF
0301         16          ;
0301 50 61 72 17          asc 'Parallel Printer Buffer Diagnostics'
0304 61 6C 6C
0307 65 6C 20
030A 50 72 69
030D 6E 74 65
0310 72 20 42
0313 75 66 66
0316 65 72 20
0319 44 69 61
031C 67 6E 6F
031F 73 74 69
0322 63 73
0324         18          ;
0324 82         19          byt RTN2          ; 1 CR and 2 LF
0325         20          ;
0325 43 68 65 21          asc 'Checking Buffer Capacity'
0328 63 6B 69
032B 6E 67 20
032E 42 75 66
0331 66 65 72
0334 20 43 61
0337 70 61 63
033A 69 74 79
033D         22          ;
033D 82         23          byt RTN2          ; 1 CR and 2 LF
033E         24          ;
033E 20 20 57 25          asc '  Write Block:  '
0341 72 69 74
0344 65 20 42
0347 6C 6F 63
034A 6B 3A 20
034D 20
034E         26          ;
034E 00         27          byt ZERO
034F         28          ;
034F         29          ;
034F         30          DMESG2:
034F 81         31          byt RTN1          ; 1 CR and 1 LF
0350         32          ;
0350 20 20 20 33          asc '  Read Block:  '
0353 52 65 61
0356 64 20 42
0359 6C 6F 63
035C 6B 3A 20
```

```
035F 20
0360          34 ;
0360 00        35      byt ZERO
0361          36 ;
0361          37 ;
0361          38 DMESG3:
0361 82        39      byt RTN2          ; 1 CR and 2 LF
0362          40 ;
0362 56 65 72  41      asc `Verified 256 KBytes`
0365 69 66 69
0368 65 64 20
036B 32 35 36
036E 20 4B 42
0371 79 74 65
0374 73
0375          42 ;
0375 83        43      byt RTN3          ; 1 CR and 3 LF
0376          44 ;
0376 52 75 6E  45      asc `Running Memory Test`
0379 6E 69 6E
037C 67 20 4D
037F 65 6D 6F
0382 72 79 20
0385 54 65 73
0388 74
0389          46 ;
0389 81        47      byt RTN1          ; 1 CR and 1 LF
038A          48 ;
038A 00        49      byt ZERO
038B          50 ;
038B          51 ;
038B          52 DMESG4:
038B 82        53      byt RTN2          ; 1 CR and 2 LF
038C          54 ;
038C 4D 65 6D  55      asc `Memory Test Completed`
038F 6F 72 79
0392 20 54 65
0395 73 74 20
0398 43 6F 6D
039B 70 6C 65
039E 74 65 64
03A1          56 ;
03A1 83        57      byt RTN3          ; 1 CR and 3 LF
03A2          58 ;
03A2 41 53 43  59      asc `ASCII Character Set`
03A5 49 49 20
03A8 43 68 61
03AB 72 61 63
03AE 74 65 72
03B1 20 53 65
03B4 74
03B5          60 ;
03B5 82        61      byt RTN2          ; 1 CR and 2 LF
03B6 00        62      byt ZERO
03B7          63 ;
03B7          64 ;
03B7          65 DMESG5:
03B7 83        66      byt RTN3          ; 1 CR and 3 LF
03B8          67 ;
03B8 45 6E 64  68      asc `End of Tests`
03BB 20 6F 66
```

```

03BE 20 54 65
03C1 73 74 73
03C4          69 ;
03C4 82          70      byt RTN2          ; 1 CR and 2 LF
03C5 00          71      byt ZERO
03C6          72 ;
03C6          73 ;
03C6          74 ; Test Messages.
03C6          75 ;
03C6          76 TMESG1:
03C6 81          77      byt RTN1          ; 1 CR and 1 LF
03C7          78 ;
03C7 20 20 57    79      asc ' Writing Block '
03CA 72 69 74
03CD 69 6E 67
03D0 20 42 6C
03D3 6F 63 6B
03D6 20
03D7          80 ;
03D7 00          81      byt ZERO
03D8          82 ;
03D8          83 TMESG2:
03D8 81          84      byt RTN1          ; 1 CR and 1 LF
03D9          85 ;
03D9 20 20 52    86      asc ' Reading Block '
03DC 65 61 64
03DF 69 6E 67
03E2 20 42 6C
03E5 6F 63 6B
03E8 20
03E9          87 ;
03E9 00          88      byt ZERO
03EA          89 ;
03EA          90 ;
03EA          91 ; Error Messages.
03EA          92 ;
03EA          93 EMESG1:
03EA 82          94      byt RTN2          ; 1 CR and 2 LF
03EB          95 ;
03EB 46 6F 75    96      asc 'Found Memory Error'
03EE 6E 64 20
03F1 4D 65 6D
03F4 6F 72 79
03F7 20 45 72
03FA 72 6F 72
03FD          97 ;
03FD 81          98      byt RTN1          ; 1 CR and 1 LF
03FE 00          99      byt ZERO
03FF          100 ;
03FF          101 ;
03FF          102      dfs $100-*-$100**/$100,NEGONE
0400          103 ;
0400          104 ;
0400          105      icl "PAGE4.L"

```

```

LLOAD PAGE4.L,A$4000

```

```

0400      1          ttl "PPB Source Code, PAGE4.L"
0400      2      ;
0400      3      ;
0400      4      ; PAGE4.L
0400      5      ;
0400      6      ;
0400      7  PAGE4:
0400      8      ;
0400      9      ;
0400     10      ; This routine performs RESET initialization.
0400     11      ;
0400     12      ; Soft RESET - bypass diagnostics
0400     13      ;
0400     14      ; Hard RESET - diagnostics done, messaging disabled
0400     15      ;
0400     16      ; Clear Button - sets Hard RESET, messaging enabled
0400     17      ;
0400     18      ;
0400     19      ; Initialize F0 to zero for Hard RESET.
0400     20      ;
0400     21  INIT:
0400 85      22          byt CLRF0          ; F0=0
0401 A5      23          byt CLRF1          ; F1=0
0402      24      ;
0402      25      ;
0402      26      ; Test User RAM for power up values in order to select
0402      27      ; Soft or Hard RESET.
0402      28      ;
0402 C5      29          byt SELRB0          ; select register bank 0
0403      30      ;
0403 B8 20    31          byt MOVR0,PW1NDX    ; R0=#PW1NDX
0405 B9 21    32          byt MOVR1,PW2NDX    ; R1=#PW2NDX
0407      33      ;
0407 F0      34          byt MOVA.R0         ; A=[R0]
0408 D3 96    35          byt XRLA,PW1VAL    ; A=A^#PW1VAL
040A 96 12    36          byt JNZ,INIT01     ; jump for Hard RESET
040C      37      ;
040C F1      38          byt MOVA.R1         ; A=[R1]
040D D3 A5    39          byt XRLA,PW2VAL    ; A=A^#PW2VAL
040F 96 12    40          byt JNZ,INIT01     ; jump for Hard RESET
0411      41      ;
0411      42      ;
0411      43      ; Select Soft RESET.
0411      44      ;
0411 95      45          byt CPLF0          ; F0=-F0
0412      46      ;
0412      47      ;
0412      48      ; Initialize all registers in Register Bank 0.
0412      49      ;
0412      50  INIT01:
0412 B8 00    51          byt MOVR0,ZERO      ; R0=#ZERO
0414 B9 00    52          byt MOVR1,ZERO      ; R1=#ZERO
0416 BA 00    53          byt MOVR2,ZERO      ; R2=#ZERO
0418 BB 00    54          byt MOVR3,ZERO      ; R3=#ZERO
041A BC 00    55          byt MOVR4,ZERO      ; R4=#ZERO
041C BD 00    56          byt MOVR5,ZERO      ; R5=#ZERO
041E BE 00    57          byt MOVR6,ZERO      ; R6=#ZERO
0420 BF 00    58          byt MOVR7,ZERO      ; R7=#ZERO
0422      59      ;
0422      60      ;

```

```

0422      61 ; Initialize all registers in Register Bank 1.
0422      62 ;
0422 D5    63      byt SELRB1          ; select register bank 1
0423      64 ;
0423 B8 00  65      byt MOVR0,ZERO      ; R0=#ZERO
0425 B9 00  66      byt MOVR1,ZERO      ; R1=#ZERO
0427 BA 00  67      byt MOVR2,ZERO      ; R2=#ZERO
0429 BB 00  68      byt MOVR3,ZERO      ; R3=#ZERO
042B BC 00  69      byt MOVR4,ZERO      ; R4=#ZERO
042D BD 00  70      byt MOVR5,ZERO      ; R5=#ZERO
042F BE 00  71      byt MOVR6,ZERO      ; R6=#ZERO
0431 BF 00  72      byt MOVR7,ZERO      ; R7=#ZERO
0433      73 ;
0433      74 ;
0433      75 ; Initialize Ports P1 and P2 to -1, and turn Ready LED Off.
0433      76 ;
0433 89 FF  77      byt ORLP1,NEGONE     ; P1=P1|#NEGONE
0435 8A FF  78      byt ORLP2,NEGONE     ; P2=P2|#NEGONE
0437      79 ;
0437 9A F7  80      byt ANLP2,RDLEDOFF   ; P2=P2&#RDLEDOFF
0439      81 ;
0439      82 ;
0439      83 ; Initialize the octal latches of U26 to zero, the value in
0439      84 ; reg-A. These latches are enabled when P1, bit 7 (PPB
0439      85 ; Memory) is enabled with the second write to External
0439      86 ; Memory (RAS, then CAS). Why???
0439      87 ;
0439 9A 7F  88      byt ANLP2,OCTLENBL    ; P2=P2&#OCTLENBL
043B      89 ;
043B 23 00  90      byt MOVA,ZERO        ; A=#ZERO
043D 90    91      byt MOVX.R0A         ; [R0]=A
043E      92 ;
043E 8A 80  93      byt ORLP2,OCTLDSBL   ; P2=P2|#OCTLDSBL
0440      94 ;
0440      95 ;
0440      96 ; Generate a 5 usec Inport Data Strobe.
0440      97 ;
0440 9A EF  98      byt ANLP2,IDATENBL    ; P2=P2&#IDATENBL
0442      99 ;
0442 00    100     byt NOP                ; no operation
0443 00    101     byt NOP                ; no operation
0444     102 ;
0444 8A 10  103     byt ORLP2,IDATDSBL    ; P2=P2|#IDATDSBL
0446     104 ;
0446     105 ;
0446     106 ; Load the Timer Register, enable Timer interrupts, and
0446     107 ; start the Timer.
0446     108 ;
0446 23 CF  109     byt MOVA,TIME1        ; A=#TIME1
0448 62    110     byt MOVTA            ; T=A
0449     111 ;
0449 25    112     byt ENTI              ; enable timer interrupts
044A 55    113     byt STRTT            ; start timer
044B     114 ;
044B     115 ;
044B     116 ; Check Copy Button and branch if it is not being pressed.
044B     117 ;
044B 36 5C  118     byt JT0,INIT03        ; jump if T0=1
044D     119 ;
044D     120 ;
044D     121 ; Turn Ready LED on, messaging on, and enable diagnostics.

```

```

044D      122 ;
044D 8A 08 123      byt ORLP2,RDLEDON      ; P2=P2|#RDLEDON
044F      124 ;
044F FB    125      byt MOVAR3            ; A=R3
0450 43 10 126      byt ORLA,MSGSFON      ; A=A|#MSGSFON
0452 AB    127      byt MOVR3A            ; R3=A
0453      128 ;
0453 85    129      byt CLRF0              ; F0=0
0454      130 ;
0454      131 ;
0454      132 ; Loop as long as Copy Button is pressed, then turn
0454      133 ; Ready LED off and pause for one second.
0454      134 ;
0454      135 INIT02:
0454 26 54 136      byt JNT0,INIT02          ; jump if T0=0
0456      137 ;
0456 34 4B 138      byt CALL1,DEBOUNCE      ; call DEBOUNCE
0458      139 ;
0458 9A F7 140      byt ANLP2,RDLEDOFF     ; P2=P2&#RDLEDOFF
045A      141 ;
045A 34 55 142      byt CALL1,WAIT1SEC     ; call WAIT1SEC
045C      143 ;
045C      144 ;
045C      145 ; Perform memory diagnostics if F0 flag is zero.
045C      146 ;
045C      147 INIT03:
045C B6 62 148      byt JF0,INIT04          ; jump if F0=1
045E      149 ;
045E 65    150      byt STOPTC              ; stop timer/counter
045F      151 ;
045F B4 00 152      byt CALL5,DIAGS        ; call DIAGS
0461      153 ;
0461 55    154      byt STRTT              ; start timer
0462      155 ;
0462      156 ;
0462      157 ; Initialize User RAM with Power Byte 1 and Power Byte 2.
0462      158 ;
0462      159 INIT04:
0462 C5    160      byt SELRB0              ; select register bank 0
0463      161 ;
0463 B8 20 162      byt MOVR0,PW1NDX        ; R0=#PW1NDX
0465 B9 21 163      byt MOVR1,PW2NDX        ; R1=#PW2NDX
0467      164 ;
0467 B0 96 165      byt MOV.R0,PW1VAL        ; [R0]=#PW1VAL
0469 B1 A5 166      byt MOV.R1,PW2VAL        ; [R1]=#PW2VAL
046B      167 ;
046B      168 ;
046B      169 ; Clear all Bank 0 registers again.
046B      170 ;
046B B8 00 171      byt MOVR0,ZERO          ; R0=#ZERO
046D B9 00 172      byt MOVR1,ZERO          ; R1=#ZERO
046F BA 00 173      byt MOVR2,ZERO          ; R2=#ZERO
0471 BB 00 174      byt MOVR3,ZERO          ; R3=#ZERO
0473 BC 00 175      byt MOVR4,ZERO          ; R4=#ZERO
0475 BD 00 176      byt MOVR5,ZERO          ; R5=#ZERO
0477 BE 00 177      byt MOVR6,ZERO          ; R6=#ZERO
0479 BF 00 178      byt MOVR7,ZERO          ; R7=#ZERO
047B      179 ;
047B      180 ;
047B      181 ; Clear all Bank 1 registers again.
047B      182 ;

```

```

047B D5          183          byt SELRB1          ; select register bank 1
047C             184          ;
047C B8 00       185          byt MOVR0,ZERO        ; R0=#ZERO
047E B9 00       186          byt MOVR1,ZERO        ; R1=#ZERO
0480 BA 00       187          byt MOVR2,ZERO        ; R2=#ZERO
0482 BB 00       188          byt MOVR3,ZERO        ; R3=#ZERO
0484 BC 00       189          byt MOVR4,ZERO        ; R4=#ZERO
0486 BD 00       190          byt MOVR5,ZERO        ; R5=#ZERO
0488 BE 00       191          byt MOVR6,ZERO        ; R6=#ZERO
048A BF 00       192          byt MOVR7,ZERO        ; R7=#ZERO
048C             193          ;
048C             194          ;
048C             195          ; Turn the EXTIRQ State Flag On, turn the Ready LED On,
048C             196          ; enable external interrupts, and return to caller.
048C             197          ;
048C FB          198          byt MOVAR3          ; A=R3
048D 43 80       199          byt ORLA,IRQSFON      ; A=A|#IRQSFON
048F AB          200          byt MOVR3A          ; R3=A
0490             201          ;
0490 8A 08       202          byt ORLP2,RDLEDON      ; P2=P2|#RDLEDON
0492             203          ;
0492             204          ;
0492             205          ; Return to caller.
0492             206          ;
0492 83          207          byt RET              ; return from subroutine
0493             208          ;
0493             209          ;
0493             210          ; Save message offset in R1.
0493             211          ;
0493             212          PRNTEMESG:
0493 D5          213          byt SELRB1          ; select register bank 1
0494             214          ;
0494 A9          215          byt MOVR1A          ; R1=A
0495             216          ;
0495             217          ;
0495             218          ; Recall message index in reg-A and read the byte in Page 3
0495             219          ; from that index into reg-A. Branch if that byte is
0495             220          ; negative or exit if that byte is zero.
0495             221          ;
0495             222          PM01:
0495 F9          223          byt MOVAR1          ; A=R1
0496 E3          224          byt MOVP3A.A          ; A=[0x300+A]
0497 F2 A1       225          byt JB7,PM04          ; jump if A<0
0499             226          ;
0499 96 9C       227          byt JNZ,PM02          ; jump if A>0
049B             228          ;
049B             229          ;
049B             230          ; Return to caller.
049B             231          ;
049B 83          232          byt RET              ; return from subroutine
049C             233          ;
049C             234          ;
049C             235          ; Send the data in reg-A to the Outport Connection.
049C             236          ;
049C             237          PM02:
049C 94 CD       238          byt CALL4,PRINT        ; call PRINT
049E             239          ;
049E             240          ;
049E             241          ; Increment the message index and branch back to PM01.
049E             242          ;
049E             243          PM03:

```

```

049E 19          244          byt INCR1          ; R1=R1+1
049F            245          ;
049F 84 95       246          byt JMP4,PM01        ; jump to PM01
04A1            247          ;
04A1            248          ;
04A1            249          ; Mask reg-A and save as a linefeed counter.
04A1            250          ;
04A1            251          PM04:
04A1 53 7F       252          byt ANLA,ASCIMASK    ; A=A&#ASCIMASK
04A3 AC          253          byt MOVR4A          ; R4=A
04A4            254          ;
04A4            255          ;
04A4            256          ; Send a carriage return to the Outport Connection.
04A4            257          ;
04A4 23 0D       258          byt MOVA,RETURN      ; A=#RETURN
04A6 94 CD       259          byt CALL4,PRINT      ; call PRINT
04A8            260          ;
04A8            261          ;
04A8            262          ; Check if linefeed counter is zero.
04A8            263          ;
04A8 FC          264          byt MOVAR4          ; A=R4
04A9 C6 9E       265          byt JZ,PM03          ; jump if A=0
04AB            266          ;
04AB            267          ;
04AB            268          ; Send a linefeed to the Outport Connection as long as the
04AB            269          ; R4 counter is not zero.
04AB            270          ;
04AB            271          PM05:
04AB 23 0A       272          byt MOVA,LINEFEED    ; A=#LINEFEED
04AD 94 CD       273          byt CALL4,PRINT      ; call PRINT
04AF            274          ;
04AF EC AB       275          byt DJNZR4,PM05       ; R4=R4-1, jump if R4>0
04B1            276          ;
04B1 84 9E       277          byt JMP4,PM03        ; jump to PM03
04B3            278          ;
04B3            279          ;
04B3            280          ; This routine sends the ASCII character set from 0x20 to
04B3            281          ; 0x7F to the Outport Connection.
04B3            282          ;
04B3            283          ; Initialize R1 with 0x20, the Space character.
04B3            284          ;
04B3            285          PRNTASCI:
04B3 23 20       286          byt MOVA,SPACE      ; A=#SPACE
04B5 A9          287          byt MOVR1A          ; R1=A
04B6            288          ;
04B6            289          ;
04B6            290          ; Send the data in reg-A to the Outport Connection.
04B6            291          ;
04B6            292          PA01:
04B6 F9          293          byt MOVAR1          ; A=R1
04B7 94 CD       294          byt CALL4,PRINT      ; call PRINT
04B9            295          ;
04B9            296          ;
04B9            297          ; Increment R1, the next ASCII character.
04B9            298          ;
04B9 19          299          byt INCR1          ; R1=R1+1
04BA            300          ;
04BA            301          ;
04BA            302          ; Branch if the next ASCII character is not 0x50, the "P"
04BA            303          ; character.
04BA            304          ;

```

```

04BA F9          305          byt MOVAR1          ; A=R1
04BB D3 50       306          byt XRLA,CHAR50       ; A=A^#CHAR50
04BD 96 C7       307          byt JNZ,PA02          ; jump if A>0
04BF            308          ;
04BF            309          ;
04BF            310          ; Send a RETURN and LINEFEED to the Outport Connection.
04BF            311          ;
04BF 23 0D       312          byt MOVA,RETURN        ; A=#RETURN
04C1 94 CD       313          byt CALL4,PRINT        ; call PRINT
04C3            314          ;
04C3 23 0A       315          byt MOVA,LINEFEED       ; A=#LINEFEED
04C5 94 CD       316          byt CALL4,PRINT        ; call PRINT
04C7            317          ;
04C7            318          ;
04C7            319          ; Branch if the next ASCII character is not 0x80, otherwise
04C7            320          ; return to caller.
04C7            321          ;
04C7            322          PA02:
04C7 F9          323          byt MOVAR1          ; A=R1
04C8 D3 80       324          byt XRLA,CHAR80       ; A=A^#CHAR80
04CA 96 B6       325          byt JNZ,PA01          ; jump if A>0
04CC            326          ;
04CC 83          327          byt RET              ; return from subroutine
04CD            328          ;
04CD            329          ;
04CD            330          ; This routine sends the data byte in reg-A to the Outport
04CD            331          ; Connection.
04CD            332          ;
04CD            333          ; Save the data byte in reg-A to R2 and check if the
04CD            334          ; Message State Flag is On.
04CD            335          ;
04CD            336          PRINT:
04CD D5          337          byt SELRB1            ; select register bank 1
04CE            338          ;
04CE AA          339          byt MOVR2A            ; R2=A
04CF            340          ;
04CF FB          341          byt MOVAR3            ; A=R3
04D0 37          342          byt CPLA              ; A=-A
04D1 92 E5       343          byt JB4,PR02          ; jump if bit 4 set (clear)
04D3            344          ;
04D3            345          ;
04D3            346          ; Wait until the Outport Data Ready bit is enabled on
04D3            347          ; Port 1 (1=Ready).
04D3            348          ;
04D3            349          PR01:
04D3 09          350          byt INAP1              ; A=P1
04D4 37          351          byt CPLA              ; A=-A
04D5 92 D3       352          byt JB4,PR01          ; jump if bit 4 set (clear)
04D7            353          ;
04D7            354          ;
04D7            355          ; Enable the processor data latch, write the output data,
04D7            356          ; and disable the latch.
04D7            357          ;
04D7 99 FD       358          byt ANLP1,PDATENBL      ; P1=P1^#PDATENBL
04D9            359          ;
04D9 B8 00       360          byt MOVR0,ZERO        ; R0=#ZERO
04DB            361          ;
04DB FA          362          byt MOVAR2            ; A=R2
04DC 90          363          byt MOVX.R0A          ; [R0]=A
04DD            364          ;
04DD 89 02       365          byt ORLP1,PDATDSBL     ; P1=P1|^#PDATDSBL

```

```
04DF      366 ;
04DF      367 ;
04DF      368 ; Generate a 5 usec Outport Latch Enable.
04DF      369 ;
04DF 99 FB 370      byt ANLP1,OLATENBL ; P1=P1&#OLATENBL
04E1      371 ;
04E1 00    372      byt NOP           ; no operation
04E2 00    373      byt NOP           ; no operation
04E3      374 ;
04E3 89 04 375      byt ORLP1,OLATDSBL ; P1=P1|#OLATDSBL
04E5      376 ;
04E5      377 ;
04E5      378 ; Return to caller.
04E5      379 ;
04E5      380 PR02:
04E5 83    381      byt RET           ; return from subroutine
04E6      382 ;
04E6      383 ;
04E6      384      dfs $100--$100**/$100,NEGONE
0500      385 ;
0500      386 ;
0500      387      icl "PAGE5.L"
```

LLOAD PAGE5.L,A\$4000

```

0500          1          ttl "PPB Source Code, PAGE5.L"
0500          2          ;
0500          3          ;
0500          4          ; PAGE5.L
0500          5          ;
0500          6          ;
0500          7          PAGE5:
0500          8          ;
0500          9          ;
0500         10          ; This routine performs memory check diagnostics.
0500         11          ;
0500         12          ; A total of 256 KB of memory is verified as four 64 KB
0500         13          ; blocks. Each 64 KB block is checked and verified.
0500         14          ;
0500         15          ; Print Message 1.
0500         16          ;
0500         17          DIAGS:
0500 23 00         18          byt MOVA,DMESG1      ; A=#DMESG1
0502 94 93         19          byt CALL4,PRNTMESG   ; call PRNTMESG
0504          20          ;
0504          21          ;
0504          22          ; Write a test pattern to the first page of each block.
0504          23          ;
0504 B4 26         24          byt CALL5,WRITBLKS   ; call WRITBLKS
0506          25          ;
0506          26          ;
0506          27          ; Print Message 2.
0506          28          ;
0506 23 4F         29          byt MOVA,DMESG2      ; A=#DMESG2
0508 94 93         30          byt CALL4,PRNTMESG   ; call PRNTMESG
050A          31          ;
050A          32          ;
050A          33          ; Read the first page of each block and verify its test
050A          34          ; pattern. Write an error message if Carry Flag is set
050A          35          ; and return to caller.
050A          36          ;
050A B4 69         37          byt CALL5,READBLKS   ; call READBLKS
050C E6 13         38          byt JNC,DIAGS2       ; jump if carry clear
050E          39          ;
050E          40          ;
050E          41          DIAGS1:
050E 23 EA         42          byt MOVA,EMESG1      ; A=#EMESG1
0510 94 93         43          byt CALL4,PRNTMESG   ; call PRNTMESG
0512          44          ;
0512          45          ;
0512          46          ; Return to caller.
0512          47          ;
0512 83           48          byt RET              ; return from subroutine
0513          49          ;
0513          50          ;
0513          51          ; Print Message 3.
0513          52          ;
0513          53          DIAGS2:
0513          54          ;
0513 23 61         55          byt MOVA,DMESG3      ; A=#DMESG3
0515 94 93         56          byt CALL4,PRNTMESG   ; call PRNTMESG
0517          57          ;
0517          58          ;
0517          59          ; Test each of the 64 KB blocks of memory found.
0517          60          ;

```

```

0517 D4 00      61          byt CALL6,TESTMEM      ; call TESTMEM
0519 F6 0E      62          byt JC,DIAGS1          ; jump if carry set
051B           63          ;
051B           64          ;
051B           65          ; Print Message 4.
051B           66          ;
051B 23 8B      67          byt MOVA,DMESG4          ; A=#DMESG4
051D 94 93      68          byt CALL4,PRNTMESG        ; call PRNTMESG
051F           69          ;
051F           70          ;
051F           71          ; Print ASCII character set.
051F           72          ;
051F 94 B3      73          byt CALL4,PRNTASCI        ; call PRNTASCI
0521           74          ;
0521           75          ;
0521           76          ; Print Message 5.
0521           77          ;
0521 23 B7      78          byt MOVA,DMESG5          ; A=#DMESG5
0523 94 93      79          byt CALL4,PRNTMESG        ; call PRNTMESG
0525           80          ;
0525           81          ;
0525           82          ; Return to caller.
0525           83          ;
0525 83         84          byt RET                  ; return from subroutine
0526           85          ;
0526           86          ;
0526           87          ; This routine verifies there exists four 64 KB blocks of
0526           88          ; memory.
0526           89          ;
0526           90          ; Initialize the CAS and RAS registers to zero.
0526           91          ;
0526           92          WRITBLKS:
0526 C5         93          byt SELRB0                ; select register bank 0
0527           94          ;
0527 BE 00      95          byt MOVR6,ZERO              ; R6=#ZERO
0529 BF 00      96          byt MOVR7,ZERO              ; R7=#ZERO
052B           97          ;
052B           98          ;
052B           99          ; Write the 64 KB Block 0 with PATRN1.
052B          100         ;
052B BB 00     101         byt MOVR3,BLOC0              ; R3=#BLOC0
052D          102         ;
052D 23 5A     103         byt MOVA,PATRN1              ; A=#PATRN1
052F F4 00     104         byt CALL7,WRITPAGE          ; call WRITPAGE
0531          105         ;
0531 23 30     106         byt MOVA,CHAR30             ; A=#CHAR30
0533 94 CD     107         byt CALL4,PRINT              ; call PRINT
0535          108         ;
0535 23 20     109         byt MOVA,SPACE              ; A=#SPACE
0537 94 CD     110         byt CALL4,PRINT              ; call PRINT
0539          111         ;
0539          112         ;
0539          113         ; Write the 64 KB Block 1 with PATRN2.
0539          114         ;
0539 C5        115         byt SELRB0                ; select register bank 0
053A          116         ;
053A FB        117         byt MOVR3                  ; A=R3
053B 03 10     118         byt ADDA,ADDWRBLK            ; A=A+#ADDWRBLK
053D AB        119         byt MOVR3A                  ; R3=A
053E          120         ;
053E 23 A5     121         byt MOVA,PATRN2              ; A=#PATRN2

```

```

0540 F4 00      122      byt CALL7,WRITPAGE      ; call WRITPAGE
0542           123      ;
0542 23 31      124      byt MOVA,CHAR31          ; A=#CHAR31
0544 94 CD      125      byt CALL4,PRINT          ; call PRINT
0546           126      ;
0546 23 20      127      byt MOVA,SPACE           ; A=#SPACE
0548 94 CD      128      byt CALL4,PRINT          ; call PRINT
054A           129      ;
054A           130      ;
054A           131      ; Write the 64 KB Block 2 with PATRN3.
054A           132      ;
054A C5         133      byt SELRB0                ; select register bank 0
054B           134      ;
054B FB         135      byt MOVAR3                ; A=R3
054C 03 10      136      byt ADDA,ADDWRBLK         ; A=A+#ADDWRBLK
054E AB         137      byt MOVR3A                ; R3=A
054F           138      ;
054F 23 69      139      byt MOVA,PATRN3           ; A=#PATRN3
0551 F4 00      140      byt CALL7,WRITPAGE         ; call WRITPAGE
0553           141      ;
0553 23 32      142      byt MOVA,CHAR32           ; A=#CHAR32
0555 94 CD      143      byt CALL4,PRINT          ; call PRINT
0557           144      ;
0557 23 20      145      byt MOVA,SPACE           ; A=#SPACE
0559 94 CD      146      byt CALL4,PRINT          ; call PRINT
055B           147      ;
055B           148      ;
055B           149      ; Write the 64 KB Block 3 with PATRN4.
055B           150      ;
055B C5         151      byt SELRB0                ; select register bank 0
055C           152      ;
055C FB         153      byt MOVAR3                ; A=R3
055D 03 10      154      byt ADDA,ADDWRBLK         ; A=A+#ADDWRBLK
055F AB         155      byt MOVR3A                ; R3=A
0560           156      ;
0560 23 96      157      byt MOVA,PATRN4           ; A=#PATRN4
0562 F4 00      158      byt CALL7,WRITPAGE         ; call WRITPAGE
0564           159      ;
0564 23 33      160      byt MOVA,CHAR33           ; A=#CHAR33
0566 94 CD      161      byt CALL4,PRINT          ; call PRINT
0568           162      ;
0568           163      ;
0568           164      ; Return to caller.
0568           165      ;
0568 83         166      byt RET                    ; return from subroutine
0569           167      ;
0569           168      ;
0569           169      READBLKS:
0569 C5         170      byt SELRB0                ; select register bank 0
056A           171      ;
056A BC 00      172      byt MOVR4,ZERO             ; R4=#ZERO
056C BD 00      173      byt MOVR5,ZERO             ; R5=#ZERO
056E           174      ;
056E           175      ;
056E           176      ; Read the 64 KB Block 0 and verify with PATRN1.
056E           177      ;
056E BB 00      178      byt MOVR3,BLOC0           ; R3=#BLOC0
0570           179      ;
0570 23 5A      180      byt MOVA,PATRN1           ; A=#PATRN1
0572 F4 1E      181      byt CALL7,READPAGE        ; call READPAGE
0574 F6 B4      182      byt JC,RBS01              ; jump if carry set

```

```

0576      183 ;
0576 23 30      184      byt MOVA,CHAR30      ; A=#CHAR30
0578 94 CD      185      byt CALL4,PRINT      ; call PRINT
057A      186 ;
057A 23 20      187      byt MOVA,SPACE      ; A=#SPACE
057C 94 CD      188      byt CALL4,PRINT      ; call PRINT
057E      189 ;
057E      190 ;
057E      191 ; Read the 64 KB Block 1 and verify with PATRN2.
057E      192 ;
057E C5      193      byt SELRB0      ; select register bank 0
057F      194 ;
057F FB      195      byt MOVAR3      ; A=R3
0580 03 01      196      byt ADDA,ADDRDBLK      ; A=A+#ADDRDBLK
0582 AB      197      byt MOVR3A      ; R3=A
0583      198 ;
0583 23 A5      199      byt MOVA,PATRN2      ; A=#PATRN2
0585 F4 1E      200      byt CALL7,READPAGE      ; call READPAGE
0587 F6 B4      201      byt JC,RBS01      ; jump if carry set
0589      202 ;
0589 23 31      203      byt MOVA,CHAR31      ; A=#CHAR31
058B 94 CD      204      byt CALL4,PRINT      ; call PRINT
058D      205 ;
058D 23 20      206      byt MOVA,SPACE      ; A=#SPACE
058F 94 CD      207      byt CALL4,PRINT      ; call PRINT
0591      208 ;
0591      209 ;
0591      210 ; Read the 64 KB Block 2 and verify with PATRN3.
0591      211 ;
0591 C5      212      byt SELRB0      ; select register bank 0
0592      213 ;
0592 FB      214      byt MOVAR3      ; A=R3
0593 03 01      215      byt ADDA,ADDRDBLK      ; A=A+#ADDRDBLK
0595 AB      216      byt MOVR3A      ; R3=A
0596      217 ;
0596 23 69      218      byt MOVA,PATRN3      ; A=#PATRN3
0598 F4 1E      219      byt CALL7,READPAGE      ; call READPAGE
059A F6 B4      220      byt JC,RBS01      ; jump if carry set
059C      221 ;
059C 23 32      222      byt MOVA,CHAR32      ; A=#CHAR32
059E 94 CD      223      byt CALL4,PRINT      ; call PRINT
05A0      224 ;
05A0 23 20      225      byt MOVA,SPACE      ; A=#SPACE
05A2 94 CD      226      byt CALL4,PRINT      ; call PRINT
05A4      227 ;
05A4      228 ;
05A4      229 ; Read the 64 KB Block 3 and verify with PATRN4.
05A4      230 ;
05A4 C5      231      byt SELRB0      ; select register bank 0
05A5      232 ;
05A5 FB      233      byt MOVAR3      ; A=R3
05A6 03 01      234      byt ADDA,ADDRDBLK      ; A=A+#ADDRDBLK
05A8 AB      235      byt MOVR3A      ; R3=A
05A9      236 ;
05A9 23 96      237      byt MOVA,PATRN4      ; A=#PATRN4
05AB F4 1E      238      byt CALL7,READPAGE      ; call READPAGE
05AD F6 B4      239      byt JC,RBS01      ; jump if carry set
05AF      240 ;
05AF 23 33      241      byt MOVA,CHAR33      ; A=#CHAR33
05B1 94 CD      242      byt CALL4,PRINT      ; call PRINT
05B3      243 ;

```

```
05B3      244 ;
05B3      245 ; Clear the Carry Flag.
05B3      246 ;
05B3 97    247      byt CLRC      ; C=0
05B4      248 ;
05B4      249 ;
05B4      250 ; Return to caller.
05B4      251 ;
05B4      252 RBS01:
05B4 83    253      byt RET      ; return from subroutine
05B5      254 ;
05B5      255 ;
05B5      256      dfs $100-*-$100**/$100,NEGONE
0600      257 ;
0600      258 ;
0600      259      icl "PAGE6.L"
```

```
LLOAD PAGE6.L,A$4000
```

```
0600          1          ttl "PPB Source Code, PAGE6.L"
0600          2          ;
0600          3          ;
0600          4          ; PAGE6.L
0600          5          ;
0600          6          ;
0600          7          PAGE6:
0600          8          ;
0600          9          ;
0600         10          ; This routine verifies each 64 KB block of memory.
0600         11          ;
0600         12          ; Initialize LED counter, Block number, and read and write
0600         13          ; CAS and RAS to zero.
0600         14          ;
0600         15          TESTMEM:
0600 C5         16          byt SELRB0          ; select register bank 0
0601         17          ;
0601 BA 00      18          byt MOVR2,ZERO      ; R3=#ZERO
0603 BB 00      19          byt MOVR3,BLOC0     ; R3=#BLOC0
0605 BC 00      20          byt MOVR4,ZERO      ; R4=#ZERO
0607 BD 00      21          byt MOVR5,ZERO      ; R5=#ZERO
0609 BE 00      22          byt MOVR6,ZERO      ; R6=#ZERO
060B BF 00      23          byt MOVR7,ZERO      ; R7=#ZERO
060D         24          ;
060D         25          ;
060D         26          ; Print Writing Block Number 0.
060D         27          ;
060D 23 C6      28          byt MOVA,TMSG1       ; A=#TMSG1
060F D4 60      29          byt CALL6,TESTMSG    ; call TESTMSG
0611         30          ;
0611         31          ;
0611         32          ; Write the 64 KB Block 0 with PATRN1.
0611         33          ;
0611 23 5A      34          byt MOVA,PATRN1       ; A=#PATRN1
0613 F4 44      35          byt CALL7,WRITBLK    ; call WRITBLK
0615         36          ;
0615         37          ;
0615         38          ; Print Reading Block Number 0.
0615         39          ;
0615 23 D8      40          byt MOVA,TMSG2       ; A=#TMSG2
0617 D4 60      41          byt CALL6,TESTMSG    ; call TESTMSG
0619         42          ;
0619         43          ;
0619         44          ; Read the 64 KB Block 0 with PATRN1.
0619         45          ;
0619 23 5A      46          byt MOVA,PATRN1       ; A=#PATRN1
061B F4 51      47          byt CALL7,READBLK    ; call READBLK
061D F6 5F      48          byt JC,TM01         ; jump if carry set
061F         49          ;
061F         50          ;
061F         51          ; Increment block number.
061F         52          ;
061F FB        53          byt MOVR3           ; A=R3
0620 03 11      54          byt ADDA,ADDRWBLK    ; A=A+#ADDRWBLK
0622 AB        55          byt MOVR3A          ; R3=A
0623         56          ;
0623         57          ;
0623         58          ; Print Writing Block Number 1.
0623         59          ;
0623 23 C6      60          byt MOVA,TMSG1       ; A=#TMSG1
```

```
0625 D4 60      61          byt CALL6,TESTMSG ; call TESTMSG
0627           62      ;
0627           63      ;
0627           64      ; Write the 64 KB Block 1 with PATRN2.
0627           65      ;
0627 23 A5      66          byt MOVA,PATRN2      ; A=#PATRN2
0629 F4 44      67          byt CALL7,WRITBLK    ; call WRITBLK
062B           68      ;
062B           69      ;
062B           70      ; Print Reading Block Number 1.
062B           71      ;
062B 23 D8      72          byt MOVA,TMSG2      ; A=#TMSG2
062D D4 60      73          byt CALL6,TESTMSG    ; call TESTMSG
062F           74      ;
062F           75      ;
062F           76      ; Read the 64 KB Block 1 with PATRN2.
062F           77      ;
062F 23 A5      78          byt MOVA,PATRN2      ; A=#PATRN2
0631 F4 51      79          byt CALL7,READBLK    ; call READBLK
0633 F6 5F      80          byt JC,TM01          ; jump if carry set
0635           81      ;
0635           82      ;
0635           83      ; Increment block number.
0635           84      ;
0635 FB        85          byt MOVAR3           ; A=R3
0636 03 11      86          byt ADDA,ADDRWBLK     ; A=A+#ADDRWBLK
0638 AB        87          byt MOVR3A           ; R3=A
0639           88      ;
0639           89      ;
0639           90      ; Print Writing Block Number 2.
0639           91      ;
0639 23 C6      92          byt MOVA,TMSG1      ; A=#TMSG1
063B D4 60      93          byt CALL6,TESTMSG    ; call TESTMSG
063D           94      ;
063D           95      ;
063D           96      ; Write the 64 KB Block 2 with PATRN3.
063D           97      ;
063D 23 69      98          byt MOVA,PATRN3      ; A=#PATRN3
063F F4 44      99          byt CALL7,WRITBLK    ; call WRITBLK
0641           100     ;
0641           101     ;
0641           102     ; Print Reading Block Number 2.
0641           103     ;
0641 23 D8      104         byt MOVA,TMSG2      ; A=#TMSG2
0643 D4 60      105         byt CALL6,TESTMSG    ; call TESTMSG
0645           106     ;
0645           107     ;
0645           108     ; Read the 64 KB Block 2 with PATRN3.
0645           109     ;
0645 23 69      110         byt MOVA,PATRN3      ; A=#PATRN3
0647 F4 51      111         byt CALL7,READBLK    ; call READBLK
0649 F6 5F      112         byt JC,TM01          ; jump if carry set
064B           113     ;
064B           114     ;
064B           115     ; Increment block number.
064B           116     ;
064B FB        117         byt MOVAR3           ; A=R3
064C 03 11      118         byt ADDA,ADDRWBLK     ; A=A+#ADDRWBLK
064E AB        119         byt MOVR3A           ; R3=A
064F           120     ;
064F           121     ;
```

```

064F      122 ; Print Writing Block Number 3.
064F      123 ;
064F 23 C6 124      byt MOVA,TMMSG1      ; A=#TMMSG1
0651 D4 60 125      byt CALL6,TESTMSG    ; call TESTMSG
0653      126 ;
0653      127 ;
0653      128 ; Write the 64 KB Block 3 with PATRN4.
0653      129 ;
0653 23 96 130      byt MOVA,PATRN4      ; A=#PATRN4
0655 F4 44 131      byt CALL7,WRITBLK    ; call WRITBLK
0657      132 ;
0657      133 ;
0657      134 ; Print Reading Block Number 3.
0657      135 ;
0657 23 D8 136      byt MOVA,TMMSG2      ; A=#TMMSG2
0659 D4 60 137      byt CALL6,TESTMSG    ; call TESTMSG
065B      138 ;
065B      139 ;
065B      140 ; Read the 64 KB Block 3 with PATRN4.
065B      141 ;
065B 23 96 142      byt MOVA,PATRN4      ; A=#PATRN4
065D F4 51 143      byt CALL7,READBLK    ; call READBLK
065F      144 ;
065F      145 ;
065F      146 ; Return to caller.
065F      147 ;
065F      148 TM01:
065F 83    149      byt RET              ; return from subroutine
0660      150 ;
0660      151 ;
0660      152 ; Print test message in reg-A.
0660      153 ;
0660      154 TESTMSG:
0660 94 93 155      byt CALL4,PRNTMSG      ; call PRNTMSG
0662      156 ;
0662 C5    157      byt SELRB0           ; select register bank 0
0663      158 ;
0663 FB    159      byt MOVAR3            ; A=R3
0664 53 03 160      byt ANLA,MSKRDBLK      ; A=A&#MSKRDBLK
0666 43 30 161      byt ORLA,CHAR30       ; A=A|#CHAR30
0668 94 CD 162      byt CALL4,PRINT      ; call PRINT
066A      163 ;
066A 23 3A 164      byt MOVA,COLON        ; A=#COLON
066C 94 CD 165      byt CALL4,PRINT      ; call PRINT
066E      166 ;
066E 23 20 167      byt MOVA,SPACE        ; A=#SPACE
0670 94 CD 168      byt CALL4,PRINT      ; call PRINT
0672      169 ;
0672 23 20 170      byt MOVA,SPACE        ; A=#SPACE
0674 94 CD 171      byt CALL4,PRINT      ; call PRINT
0676      172 ;
0676      173 ;
0676      174 ; Return to caller.
0676      175 ;
0676 83    176      byt RET              ; return from subroutine
0677      177 ;
0677      178 ;
0677      179      dfs $100-*-$100**/$100,NEGONE
0700      180 ;
0700      181 ;
0700      182      icl "PAGE7.L"

```



```
0700          1          ttl "PPB Source Code, PAGE7.L"
0700          2          ;
0700          3          ;
0700          4          ; PAGE7.L
0700          5          ;
0700          6          ;
0700          7          PAGE7:
0700          8          ;
0700          9          ;
0700         10          ; This routine writes a 256 byte page of memory with a
0700         11          ; data pattern.
0700         12          ;
0700         13          ; Copy the data pattern to R1 and enable the PPB external
0700         14          ; memory.
0700         15          ;
0700         16          WRITPAGE:
0700 A9         17          byt MOVR1A          ; R1=A
0701         18          ;
0701 99 7F      19          byt ANLP1,PMEMENBL  ; P1=P1&#PMEMENBL
0703         20          ;
0703         21          ;
0703         22          ; Set ADR08 then check the state of the write block number
0703         23          ; LSB and branch if set.
0703         24          ;
0703         25          WP01:
0703 8A 01      26          byt ORLP2,ADR08ON    ; set ADR08
0705         27          ;
0705 FB        28          byt MOVAR3          ; A=R3
0706 92 0A      29          byt JB4,WP02        ; jump if LSB set
0708         30          ;
0708 9A FE      31          byt ANLP2,ADR08OFF    ; clear ADR08
070A         32          ;
070A         33          ;
070A         34          ; Write the write data RAS address.
070A         35          ;
070A         36          WP02:
070A FE        37          byt MOVAR6          ; A=R6
070B 90        38          byt MOVX.R0A        ; [R0]=A
070C         39          ;
070C         40          ;
070C         41          ; Set ADR08 then check the state of the write block number
070C         42          ; MSB and branch if set.
070C         43          ;
070C 8A 01      44          byt ORLP2,ADR08ON    ; set ADR08
070E         45          ;
070E FB        46          byt MOVAR3          ; A=R3
070F B2 13     47          byt JB5,WP03        ; jump if MSB set
0711         48          ;
0711 9A FE      49          byt ANLP2,ADR08OFF    ; clear ADR08
0713         50          ;
0713         51          ;
0713         52          ; Write the write data CAS address.
0713         53          ;
0713         54          WP03:
0713 FF        55          byt MOVAR7          ; A=R7
0714 90        56          byt MOVX.R0A        ; [R0]=A
0715         57          ;
0715         58          ;
0715         59          ; Write the data pattern to memory.
0715         60          ;
```

```

0715 F9          61          byt MOVAR1          ; A=R1
0716 90          62          byt MOVX.R0A         ; [R0]=A
0717            63          ;
0717            64          ;
0717            65          ; Increment the write data RAS address and test for zero.
0717            66          ;
0717 1E          67          byt INCR6             ; R6=R6+1
0718            68          ;
0718 FE          69          byt MOVAR6           ; A=R6
0719 96 03       70          byt JNZ,WP01         ; jump if A>0
071B            71          ;
071B            72          ;
071B            73          ; Disable the PPB external memory and return to caller.
071B            74          ;
071B 89 80       75          byt ORLP1,PMEMDSBL    ; P1=P1|#PMEMDSBL
071D            76          ;
071D 83          77          byt RET              ; return from subroutine
071E            78          ;
071E            79          ;
071E            80          ; This routine reads and verifies a 256 byte page of memory
071E            81          ; against a known data pattern.
071E            82          ;
071E            83          ; Copy the data pattern to R1 and enable the PPB external
071E            84          ; memory.
071E            85          ;
071E            86          READPAGE:
071E A9          87          byt MOVR1A           ; R1=A
071F            88          ;
071F 99 7F       89          byt ANLP1,PMEMENBL    ; P1=P1&#PMEMENBL
0721            90          ;
0721            91          ;
0721            92          ; Set ADR08 then check the state of the read block number
0721            93          ; LSB and branch if set.
0721            94          ;
0721            95          RP01:
0721 8A 01       96          byt ORLP2,ADR08ON      ; set ADR08
0723            97          ;
0723 FB          98          byt MOVAR3           ; A=R3
0724 12 28       99          byt JB0,RP02         ; jump if LSB set
0726           100          ;
0726 9A FE      101          byt ANLP2,ADR08OFF    ; clear ADR08
0728           102          ;
0728           103          ;
0728           104          ; Write the read data RAS address.
0728           105          ;
0728           106          RP02:
0728           107          ;
0728 FC          108          byt MOVAR4           ; A=R4
0729 90          109          byt MOVX.R0A         ; [R0]=A
072A           110          ;
072A           111          ;
072A           112          ; Set ADR08 then check the state of the read block number
072A           113          ; MSB and branch if set.
072A           114          ;
072A 8A 01      115          byt ORLP2,ADR08ON      ; set AR08
072C           116          ;
072C FB          117          byt MOVAR3           ; A=R3
072D 32 31      118          byt JB1,RP03         ; jump if MSB set
072F           119          ;
072F 9A FE      120          byt ANLP2,ADR08OFF    ; clear ADR08
0731           121          ;

```

```

0731      122 ;
0731      123 ; Write the read data CAS address.
0731      124 ;
0731      125 RP03:
0731 FD      126      byt MOVAR5      ; A=R5
0732 90      127      byt MOVX.R0A      ; [R0]=A
0733      128 ;
0733      129 ;
0733      130 ; Read the data in memory and compare to the saved data
0733      131 ; pattern. Return to caller if they differ.
0733      132 ;
0733 80      133      byt MOVXA.R0      ; A=[R0]
0734      134 ;
0734 D9      135      byt XRLAR1      ; A=A^R1
0735 96 3F    136      byt JNZ,RP04      ; jump if A>0
0737      137 ;
0737      138 ;
0737      139 ; Increment the read data RAS address and test for zero.
0737      140 ;
0737 1C      141      byt INCR4      ; R4=R4+1
0738      142 ;
0738 FC      143      byt MOVAR4      ; A=R4
0739 96 21    144      byt JNZ,RP01      ; jump if A>0
073B      145 ;
073B      146 ;
073B      147 ; Disable the PPB external memory, clear the Carry Flag,
073B      148 ; and return to caller.
073B      149 ;
073B 89 80    150      byt ORLP1,PMEMDSBL ; P1=P1|#PMEMDSBL
073D      151 ;
073D 97      152      byt CLRC      ; C=0
073E      153 ;
073E 83      154      byt RET      ; return from subroutine
073F      155 ;
073F      156 ;
073F      157 ; Disable the PPB external memory, set the Carry Flag,
073F      158 ; and return to caller.
073F      159 ;
073F      160 RP04:
073F 89 80    161      byt ORLP1,PMEMDSBL ; P1=P1|#PMEMDSBL
0741      162 ;
0741 97      163      byt CLRC      ; C=0
0742 A7      164      byt CPLC      ; C=-C
0743      165 ;
0743 83      166      byt RET      ; return from subroutine
0744      167 ;
0744      168 ;
0744      169 ; Copy the data pattern to R1.
0744      170 ;
0744      171 WRITBLK:
0744 C5      172      byt SELRB0      ; select register bank 0
0745      173 ;
0745 A9      174      byt MOVR1A      ; R1=A
0746      175 ;
0746      176 ;
0746      177 ; Call the LED counter and write a page of memory.
0746      178 ;
0746      179 WB01:
0746 F4 61    180      byt CALL7,TOGGLED      ; call TOGGLED
0748      181 ;
0748 C5      182      byt SELRB0      ; select register bank 0

```

```
0749          183 ;
0749 F9        184          byt MOVAR1          ; A=R1
074A F4 00     185          byt CALL7,WRITPAGE      ; call WRITPAGE
074C          186 ;
074C          187 ;
074C          188 ; Increment the write data CAS address and test for zero.
074C          189 ;
074C 1F        190          byt INCR7          ; R7=R7+1
074D          191 ;
074D FF        192          byt MOVAR7          ; A=R7
074E 96 46     193          byt JNZ,WB01          ; jump if A>0
0750          194 ;
0750          195 ;
0750          196 ; Return to caller.
0750          197 ;
0750 83        198          byt RET          ; return from subroutine
0751          199 ;
0751          200 ;
0751          201 ; Copy the data pattern to R1.
0751          202 ;
0751          203 READBLK:
0751 C5        204          byt SELRB0          ; select register bank 0
0752          205 ;
0752 A9        206          byt MOVR1A          ; R1=A
0753          207 ;
0753          208 ;
0753          209 ; Call the LED counter and read a page of memory.
0753          210 ;
0753          211 RB01:
0753 F4 61     212          byt CALL7,TOGGLED      ; call TOGGLED
0755          213 ;
0755 C5        214          byt SELRB0          ; select register bank 0
0756          215 ;
0756 F9        216          byt MOVAR1          ; A=R1
0757 F4 1E     217          byt CALL7,READPAGE     ; call READPAGE
0759 F6 60     218          byt JC,RB02          ; jump if carry set
075B          219 ;
075B          220 ;
075B          221 ; Increment the read data CAS address and test for zero.
075B          222 ;
075B 1D        223          byt INCR5          ; R5=R5+1
075C          224 ;
075C FD        225          byt MOVAR5          ; A=R5
075D 96 53     226          byt JNZ,RB01          ; jump if A>0
075F          227 ;
075F          228 ;
075F          229 ; Clear the Carry Flag.
075F          230 ;
075F 97        231          byt CLRC          ; C=0
0760          232 ;
0760          233 ;
0760          234 ; Return to caller.
0760          235 ;
0760          236 RB02:
0760 83        237          byt RET          ; return from subroutine
0761          238 ;
0761          239 ;
0761          240 ; This routine toggles the Ready LED On and Off every 8
0761          241 ; calls.
0761          242 ;
0761          243 ; Increment the LED toggle counter and check count.
```

```

0761          244 ;
0761          245 TOGGLED:
0761 1A        246          byt INCR2          ; R2=R2+1
0762          247 ;
0762 FA       248          byt MOVAR2          ; A=R2
0763 53 7F    249          byt ANLA,LEDMSK      ; A=A&#LEDMSK
0765 D3 10    250          byt XRLA,LEDCNT      ; A=A^#LEDCNT
0767 96 7E    251          byt JNZ,TL02        ; jump if A>0
0769          252 ;
0769          253 ;
0769          254 ; Reset counter and toggle LED bit.
0769          255 ;
0769 FA       256          byt MOVAR2          ; A=R2
076A 53 80    257          byt ANLA,LEDBIT      ; A=A&#LEDBIT
076C D3 80    258          byt XRLA,LEDBIT      ; A=A^#LEDBIT
076E AA       259          byt MOVR2A          ; R2=A
076F          260 ;
076F          261 ;
076F          262 ; Check LED bit and turn LED On if bit is On or turn LED
076F          263 ; Off if bit is Off.
076F          264 ;
076F F2 78    265          byt JB7,TL01          ; jump if bit 7 set
0771          266 ;
0771          267 ;
0771          268 ; Turn Ready LED Off and return to caller.
0771          269 ;
0771 9A F7     270          byt ANLP2,RDLEDOFF    ; P2=P2&#RDLEDOFF
0773          271 ;
0773 23 2A     272          byt MOVA,STAR            ; A=#STAR
0775 94 CD     273          byt CALL4,PRINT      ; call PRINT
0777          274 ;
0777 83        275          byt RET            ; return from subroutine
0778          276 ;
0778          277 ;
0778          278 ; Turn Ready LED On and return to caller.
0778          279 ;
0778          280 TL01:
0778 8A 08     281          byt ORLP2,RDLEDON            ; P2=P2|#RDLEDON
077A          282 ;
077A 23 2A     283          byt MOVA,STAR            ; A=#STAR
077C 94 CD     284          byt CALL4,PRINT      ; call PRINT
077E          285 ;
077E          286 ;
077E          287 ; Return to caller.
077E          288 ;
077E          289 TL02:
077E 83        290          byt RET            ; return from subroutine
077F          291 ;
077F          292 ;
077F          293          dfs $100-*-$100**/$100,NEGONE
0800          294 ;
0800          295 ;

```

BSAVE PPB,A\$1000,B,L\$0800

```

0800          296          usr PPB
0800          297 ;
0800          298 ;
0800          299          stt "PPB Symbol Table"
0800          300 ;
0800          301 ;
0800          302          end 111

```

*** End of Assembly

Symbol List starts at 0x7800, ends at 0x884A, used 0x104A, remaining 0x2F66

Symbols unsorted:

ZERO	0000	ONE	0001	FOUR	0004	RTN1	0081	RTN2	0082
RTN3	0083	NEGONE	00FF	BLOC0	0000	PW1NDX	0020	PW2NDX	0021
LED1NDX	0022	LED2NDX	0023	PW1VAL	0096	PW2VAL	00A5	LINEFEED	000A
RETURN	000D	SPACE	0020	STAR	002A	COLON	003A	ASCIMASK	007F
CHAR30	0030	CHAR31	0031	CHAR32	0032	CHAR33	0033	CHAR50	0050
CHAR80	0080	PATRN1	005A	PATRN2	00A5	PATRN3	0069	PATRN4	0096
TMRCNT	0031	TMRPSCAL	0050	TMRRSET	0014	TMRTIM	0F64	TIME2VAL	FA00
TIME3VAL	0010	TIME4VAL	0018	TIME2CNT	0010	TIME3CNT	0100	TIME4CNT	0180
TIME1	FFCF	TIME2	FFF0	TIME3	FF00	TIME4	FE80	ADDRDBLK	0001
ADDWRBLK	0010	ADDRWBLK	0011	MSKRDBLK	0003	MSKWRBLK	0030	MSKRWBLK	0033
CLRRDBLK	0030	LEDCNT	0010	LEDBIT	0080	LEDMSK	007F	IRQSFON	0080
IRQSFOFF	007F	PAUSFON	0040	PAUSFOFF	00BF	PAUSFTGL	0040	CPYSFON	0020
CPYSFOFF	00DF	MSGSFON	0010	MSGSF0FF	00EF	OVFSFON	0008	OVFSFOFF	00F7
PMEMDSBL	0080	PMEMENBL	007F	OLATDSBL	0004	OLATENBL	00FB	PDATDSBL	0002
PDATENBL	00FD	OCTLDSBL	0080	OCTLENBL	007F	IDATDSBL	0010	IDATENBL	00EF
RDLEDON	0008	RDLEDOFF	00F7	LED2TGL	0004	LED1TGL	0002	ADR08MSK	0001
ADR08TGL	0001	ADR08ON	0001	ADR08OFF	00FE	NOP	0000	OUTDBA	0002
ADDA	0003	JMP0	0004	ENI	0005	DECA	0007	INSABUS	0008
INAP1	0009	INAP2	000A	MOVDAP4	000C	MOVDAP5	000D	MOVDAP6	000E
MOVDAP7	000F	INC.R0	0010	INC.R1	0011	JB0	0012	ADDCA	0013
CALL0	0014	DISI	0015	JTF	0016	INCA	0017	INCR0	0018
INCR1	0019	INCR2	001A	INCR3	001B	INCR4	001C	INCR5	001D
INCR6	001E	INCR7	001F	XCHA.R0	0020	XCHA.R1	0021	MOVA	0023
JMP1	0024	ENTI	0025	JNT0	0026	CLRA	0027	XCHAR0	0028
XCHAR1	0029	XCHAR2	002A	XCHAR3	002B	XCHAR4	002C	XCHAR5	002D
XCHAR6	002E	XCHAR7	002F	XCHDA.R0	0030	XCHDA.R1	0031	JB1	0032
CALL1	0034	DISTI	0035	JT0	0036	CPLA	0037	OUTLP1A	0039
OUTLP2A	003A	MOVDP4A	003C	MOVDP5A	003D	MOVDP6A	003E	MOVDP7A	003F
ORLA.R0	0040	ORLA.R1	0041	MOVAT	0042	ORLA	0043	JMP2	0044
STRTC	0045	JNT1	0046	SWAPA	0047	ORLAR0	0048	ORLAR1	0049
ORLAR2	004A	ORLAR3	004B	ORLAR4	004C	ORLAR5	004D	ORLAR6	004E
ORLAR7	004F	ANLA.R0	0050	ANLA.R1	0051	JB2	0052	ANLA	0053
CALL2	0054	STRTT	0055	JT1	0056	DAA	0057	ANLAR0	0058
ANLAR1	0059	ANLAR2	005A	ANLAR3	005B	ANLAR4	005C	ANLAR5	005D
ANLAR6	005E	ANLAR7	005F	ADDA.R0	0060	ADDA.R1	0061	MOVTA	0062
JMP3	0064	STOPTC	0065	RRCA	0067	ADDAR0	0068	ADDAR1	0069
ADDAR2	006A	ADDAR3	006B	ADDAR4	006C	ADDAR5	006D	ADDAR6	006E
ADDAR7	006F	ADDCA.R0	0070	ADDCA.R1	0071	JB3	0072	CALL3	0074
ENT0	0075	JF1	0076	RRA	0077	ADDCAR0	0078	ADDCAR1	0079
ADDCAR2	007A	ADDCAR3	007B	ADDCAR4	007C	ADDCAR5	007D	ADDCAR6	007E
ADDCAR7	007F	MOVXA.R0	0080	MOVXA.R1	0081	RET	0083	JMP4	0084
CLRF0	0085	JNI	0086	ORLBUS	0088	ORLP1	0089	ORLP2	008A
ORLDP4A	008C	ORLDP5A	008D	ORLDP6A	008E	ORLDP7A	008F	MOVX.R0A	0090
MOVX.R1A	0091	JB4	0092	RETR	0093	CALL4	0094	CPLF0	0095
JNZ	0096	CLRC	0097	ANLBUS	0098	ANLP1	0099	ANLP2	009A
ANLDP4A	009C	ANLDP5A	009D	ANLDP6A	009E	ANLDP7A	009F	MOV.R0A	00A0
MOV.R1A	00A1	MOVPA.A	00A3	JMP5	00A4	CLRF1	00A5	CPLC	00A7
MOVR0A	00A8	MOVR1A	00A9	MOVR2A	00AA	MOVR3A	00AB	MOVR4A	00AC
MOVR5A	00AD	MOVR6A	00AE	MOVR7A	00AF	MOV.R0	00B0	MOV.R1	00B1
JB5	00B2	JMPP.A	00B3	CALL5	00B4	CPLF1	00B5	JF0	00B6
MOVR0	00B8	MOVR1	00B9	MOVR2	00BA	MOVR3	00BB	MOVR4	00BC
MOVR5	00BD	MOVR6	00BE	MOVR7	00BF	JMP6	00C4	SELRB0	00C5
JZ	00C6	MOVAPSW	00C7	DECR0	00C8	DECR1	00C9	DECR2	00CA
DECR3	00CB	DECR4	00CC	DECR5	00CD	DECR6	00CE	DECR7	00CF
XRLA.R0	00D0	XRLA.R1	00D1	JB6	00D2	XRLA	00D3	CALL6	00D4
SELRB1	00D5	MOVPSWA	00D7	XRLAR0	00D8	XRLAR1	00D9	XRLAR2	00DA

XRLAR3	00DB	XRLAR4	00DC	XRLAR5	00DD	XRLAR6	00DE	XRLAR7	00DF
MOVPA.A	00E3	JMP7	00E4	SELMB0	00E5	JNC	00E6	RLA	00E7
DJNZR0	00E8	DJNZR1	00E9	DJNZR2	00EA	DJNZR3	00EB	DJNZR4	00EC
DJNZR5	00ED	DJNZR6	00EE	DJNZR7	00EF	MOVA.R0	00F0	MOVA.R1	00F1
JB7	00F2	CALL7	00F4	SELMB1	00F5	JC	00F6	RLCA	00F7
MOVAR0	00F8	MOVAR1	00F9	MOVAR2	00FA	MOVAR3	00FB	MOVAR4	00FC
MOVAR5	00FD	MOVAR6	00FE	MOVAR7	00FF	PAGE0	0000	RESET	0000
EXTIRQ	0003	TIMRIRQ	0007	RF01	001A	RF02	0026	RF03	002B
RF04	0030	MAIN	0033	MAIN01	0038	SAV2MEM	0042	EI01	005B
EI02	0064	EI03	0083	EI04	0096	TESTLED1	0099	TESTL01	00A4
TESTLED2	00A9	TESTL02	00B4	PAGE1	0100	CHECKT0	0100	CT01	010C
CT02	011B	CT03	0122	CT04	012D	CT05	0134	CT06	013D
CT07	0143	CT08	0148	CT09	014A	DEBOUNCE	014B	DB01	0150
WAIT1SEC	0155	W1S01	015A	PAGE2	0200	MEM2OUT	0200	MO01	0212
MO02	021F	MO03	0228	MO04	0247	MO05	0253	MO06	0254
MO07	0255	MO08	025B	MO09	025D	MO10	026B	PAGE3	0300
DMESG1	0300	DMESG2	034F	DMESG3	0361	DMESG4	038B	DMESG5	03B7
TMESG1	03C6	TMESG2	03D8	EMESG1	03EA	PAGE4	0400	INIT	0400
INIT01	0412	INIT02	0454	INIT03	045C	INIT04	0462	PRNTMESG	0493
PM01	0495	PM02	049C	PM03	049E	PM04	04A1	PM05	04AB
PRNTASCI	04B3	PA01	04B6	PA02	04C7	PRINT	04CD	PR01	04D3
PR02	04E5	PAGE5	0500	DIAGS	0500	DIAGS1	050E	DIAGS2	0513
WRITBLKS	0526	READBLKS	0569	RBS01	05B4	PAGE6	0600	TESTMEM	0600
TM01	065F	TESTMESG	0660	PAGE7	0700	WRITPAGE	0700	WP01	0703
WP02	070A	WP03	0713	READPAGE	071E	RP01	0721	RP02	0728
RP03	0731	RP04	073F	WRITBLK	0744	WB01	0746	READBLK	0751
RB01	0753	RB02	0760	TOGGLED	0761	TL01	0778	TL02	077E

Symbols alphabetically sorted:

ADDA	0003	ADDA.R0	0060	ADDA.R1	0061	ADDAR0	0068	ADDAR1	0069
ADDAR2	006A	ADDAR3	006B	ADDAR4	006C	ADDAR5	006D	ADDAR6	006E
ADDAR7	006F	ADDCA	0013	ADDCA.R0	0070	ADDCA.R1	0071	ADDCAR0	0078
ADDCAR1	0079	ADDCAR2	007A	ADDCAR3	007B	ADDCAR4	007C	ADDCAR5	007D
ADDCAR6	007E	ADDCAR7	007F	ADDRDBLK	0001	ADDRWBLK	0011	ADDRWBLK	0010
ADR08MSK	0001	ADR08OFF	00FE	ADR08ON	0001	ADR08TGL	0001	ANLA	0053
ANLA.R0	0050	ANLA.R1	0051	ANLAR0	0058	ANLAR1	0059	ANLAR2	005A
ANLAR3	005B	ANLAR4	005C	ANLAR5	005D	ANLAR6	005E	ANLAR7	005F
ANLBUS	0098	ANLDP4A	009C	ANLDP5A	009D	ANLDP6A	009E	ANLDP7A	009F
ANLP1	0099	ANLP2	009A	ASCIMASK	007F	BLOC0	0000	CALL0	0014
CALL1	0034	CALL2	0054	CALL3	0074	CALL4	0094	CALL5	00B4
CALL6	00D4	CALL7	00F4	CHAR30	0030	CHAR31	0031	CHAR32	0032
CHAR33	0033	CHAR50	0050	CHAR80	0080	CHECKT0	0100	CLRA	0027
CLRC	0097	CLRF0	0085	CLRF1	00A5	CLRRDBLK	0030	COLON	003A
CPLA	0037	CPLC	00A7	CPLF0	0095	CPLF1	00B5	CPYSFOFF	00DF
CPYSFON	0020	CT01	010C	CT02	011B	CT03	0122	CT04	012D
CT05	0134	CT06	013D	CT07	0143	CT08	0148	CT09	014A
DAA	0057	DB01	0150	DEBOUNCE	014B	DECA	0007	DECR0	00C8
DECR1	00C9	DECR2	00CA	DECR3	00CB	DECR4	00CC	DECR5	00CD
DECR6	00CE	DECR7	00CF	DIAGS	0500	DIAGS1	050E	DIAGS2	0513
DISI	0015	DISTI	0035	DJNZR0	00E8	DJNZR1	00E9	DJNZR2	00EA
DJNZR3	00EB	DJNZR4	00EC	DJNZR5	00ED	DJNZR6	00EE	DJNZR7	00EF
DMESG1	0300	DMESG2	034F	DMESG3	0361	DMESG4	038B	DMESG5	03B7
EI01	005B	EI02	0064	EI03	0083	EI04	0096	EMESG1	03EA
ENI	0005	ENT0	0075	ENTI	0025	EXTIRQ	0003	FOUR	0004
IDATDSBL	0010	IDATENBL	00EF	INAP1	0009	INAP2	000A	INC.R0	0010
INC.R1	0011	INCA	0017	INCR0	0018	INCR1	0019	INCR2	001A
INCR3	001B	INCR4	001C	INCR5	001D	INCR6	001E	INCR7	001F
INIT	0400	INIT01	0412	INIT02	0454	INIT03	045C	INIT04	0462

INSABUS	0008	IRQSFOFF	007F	IRQSFON	0080	JB0	0012	JB1	0032
JB2	0052	JB3	0072	JB4	0092	JB5	00B2	JB6	00D2
JB7	00F2	JC	00F6	JF0	00B6	JF1	0076	JMP0	0004
JMP1	0024	JMP2	0044	JMP3	0064	JMP4	0084	JMP5	00A4
JMP6	00C4	JMP7	00E4	JMPP.A	00B3	JNC	00E6	JNI	0086
JNT0	0026	JNT1	0046	JNZ	0096	JT0	0036	JT1	0056
JTF	0016	JZ	00C6	LED1NDX	0022	LED1TGL	0002	LED2NDX	0023
LED2TGL	0004	LEDBIT	0080	LEDCNT	0010	LEDMSK	007F	LINEFEED	000A
MAIN	0033	MAIN01	0038	MEM2OUT	0200	MO01	0212	MO02	021F
MO03	0228	MO04	0247	MO05	0253	MO06	0254	MO07	0255
MO08	025B	MO09	025D	MO10	026B	MOV.R0	00B0	MOV.R0A	00A0
MOV.R1	00B1	MOV.R1A	00A1	MOVA	0023	MOVA.R0	00F0	MOVA.R1	00F1
MOVAPSW	00C7	MOVAR0	00F8	MOVAR1	00F9	MOVAR2	00FA	MOVAR3	00FB
MOVAR4	00FC	MOVAR5	00FD	MOVAR6	00FE	MOVAR7	00FF	MOVAT	0042
MOVDAP4	000C	MOVDAP5	000D	MOVDAP6	000E	MOVDAP7	000F	MOVDP4A	003C
MOVDP5A	003D	MOVDP6A	003E	MOVDP7A	003F	MOVVP3A.A	00E3	MOVPA.A	00A3
MOVPSWA	00D7	MOVR0	00B8	MOVR0A	00A8	MOVR1	00B9	MOVR1A	00A9
MOVR2	00BA	MOVR2A	00AA	MOVR3	00BB	MOVR3A	00AB	MOVR4	00BC
MOVR4A	00AC	MOVR5	00BD	MOVR5A	00AD	MOVR6	00BE	MOVR6A	00AE
MOVR7	00BF	MOVR7A	00AF	MOVTA	0062	MOVX.R0A	0090	MOVX.R1A	0091
MOVXA.R0	0080	MOVXA.R1	0081	MSGSF0FF	00EF	MSGSFON	0010	MSKRDBLK	0003
MSKRWBLK	0033	MSKWRBLK	0030	NEGONE	00FF	NOP	0000	OCTLDSBL	0080
OCTLENBL	007F	OLATDSBL	0004	OLATENBL	00FB	ONE	0001	ORLA	0043
ORLA.R0	0040	ORLA.R1	0041	ORLAR0	0048	ORLAR1	0049	ORLAR2	004A
ORLAR3	004B	ORLAR4	004C	ORLAR5	004D	ORLAR6	004E	ORLAR7	004F
ORLBUS	0088	ORLDP4A	008C	ORLDP5A	008D	ORLDP6A	008E	ORLDP7A	008F
ORLP1	0089	ORLP2	008A	OUTDBA	0002	OUTLP1A	0039	OUTLP2A	003A
OVFSFOFF	00F7	OVFSFON	0008	PA01	04B6	PA02	04C7	PAGE0	0000
PAGE1	0100	PAGE2	0200	PAGE3	0300	PAGE4	0400	PAGE5	0500
PAGE6	0600	PAGE7	0700	PATRN1	005A	PATRN2	00A5	PATRN3	0069
PATRN4	0096	PAUSFOFF	00BF	PAUSFON	0040	PAUSFTGL	0040	PDATDSBL	0002
PDATENBL	00FD	PM01	0495	PM02	049C	PM03	049E	PM04	04A1
PM05	04AB	PMEMDSBL	0080	PMEMENBL	007F	PR01	04D3	PR02	04E5
PRINT	04CD	PRNTASCI	04B3	PRNTMESG	0493	PW1NDX	0020	PW1VAL	0096
PW2NDX	0021	PW2VAL	00A5	RB01	0753	RB02	0760	RBS01	05B4
RDLEDOFF	00F7	RDLEDON	0008	READBLK	0751	READBLKS	0569	READPAGE	071E
RESET	0000	RET	0083	RETR	0093	RETURN	000D	RF01	001A
RF02	0026	RF03	002B	RF04	0030	RLA	00E7	RLCA	00F7
RP01	0721	RP02	0728	RP03	0731	RP04	073F	RRA	0077
RRCA	0067	RTN1	0081	RTN2	0082	RTN3	0083	SAV2MEM	0042
SELMB0	00E5	SELMB1	00F5	SELRB0	00C5	SELRB1	00D5	SPACE	0020
STAR	002A	STOPTC	0065	STRTC	0045	STRTT	0055	SWAPA	0047
TESTL01	00A4	TESTL02	00B4	TESTLED1	0099	TESTLED2	00A9	TESTMEM	0600
TESTMESG	0660	TIME1	FFCF	TIME2	FFF0	TIME2CNT	0010	TIME2VAL	FA00
TIME3	FF00	TIME3CNT	0100	TIME3VAL	0010	TIME4	FE80	TIME4CNT	0180
TIME4VAL	0018	TIMRIRQ	0007	TL01	0778	TL02	077E	TM01	065F
TMESG1	03C6	TMESG2	03D8	TMRCNT	0031	TMRPSCAL	0050	TMRRSET	0014
TMRTIM	0F64	TOGGLED	0761	W1S01	015A	WAIT1SEC	0155	WB01	0746
WP01	0703	WP02	070A	WP03	0713	WRITBLK	0744	WRITBLKS	0526
WRITPAGE	0700	XCHA.R0	0020	XCHA.R1	0021	XCHAR0	0028	XCHAR1	0029
XCHAR2	002A	XCHAR3	002B	XCHAR4	002C	XCHAR5	002D	XCHAR6	002E
XCHAR7	002F	XCHDA.R0	0030	XCHDA.R1	0031	XRLA	00D3	XRLA.R0	00D0
XRLA.R1	00D1	XRLAR0	00D8	XRLAR1	00D9	XRLAR2	00DA	XRLAR3	00DB
XRLAR4	00DC	XRLAR5	00DD	XRLAR6	00DE	XRLAR7	00DF	ZERO	0000

Symbols numerically sorted:

ZERO	0000	RESET	0000	PAGE0	0000	NOP	0000	BLOC0	0000
ONE	0001	ADR08TGL	0001	ADR08ON	0001	ADR08MSK	0001	ADDRDBLK	0001

PDATDSBL	0002	OUTDBA	0002	LED1TGL	0002	MSKRDBLK	0003	EXTIRQ	0003
ADDA	0003	OLATDSBL	0004	LED2TGL	0004	JMP0	0004	FOUR	0004
ENI	0005	TIMRIRQ	0007	DECA	0007	RDLEDON	0008	OVFSFON	0008
INSABUS	0008	INAP1	0009	LINEFEED	000A	INAP2	000A	MOVDAP4	000C
RETURN	000D	MOVDAP5	000D	MOVDAP6	000E	MOVDAP7	000F	TIME3VAL	0010
TIME2CNT	0010	MSGSFON	0010	LEDCNT	0010	INC.R0	0010	IDATDSBL	0010
ADDWRBLK	0010	INC.R1	0011	ADDRWBLK	0011	JB0	0012	ADDCA	0013
TMRRSET	0014	CALL0	0014	DISI	0015	JTF	0016	INCA	0017
TIME4VAL	0018	INCR0	0018	INCR1	0019	RF01	001A	INCR2	001A
INCR3	001B	INCR4	001C	INCR5	001D	INCR6	001E	INCR7	001F
XCHA.R0	0020	SPACE	0020	PW1NDX	0020	CPYSFON	0020	XCHA.R1	0021
PW2NDX	0021	LED1NDX	0022	MOVA	0023	LED2NDX	0023	JMP1	0024
ENTI	0025	RF02	0026	JNT0	0026	CLRA	0027	XCHAR0	0028
XCHAR1	0029	XCHAR2	002A	STAR	002A	XCHAR3	002B	RF03	002B
XCHAR4	002C	XCHAR5	002D	XCHAR6	002E	XCHAR7	002F	XCHDA.R0	0030
RF04	0030	MSKWRBLK	0030	CLRRDBLK	0030	CHAR30	0030	XCHDA.R1	0031
TMRCNT	0031	CHAR31	0031	JB1	0032	CHAR32	0032	MSKRWBLK	0033
MAIN	0033	CHAR33	0033	CALL1	0034	DISTI	0035	JT0	0036
CPLA	0037	MAIN01	0038	OUTLP1A	0039	OUTLP2A	003A	COLON	003A
MOVDP4A	003C	MOVDP5A	003D	MOVDP6A	003E	MOVDP7A	003F	PAUSFTGL	0040
PAUSFON	0040	ORLA.R0	0040	ORLA.R1	0041	SAV2MEM	0042	MOVAT	0042
ORLA	0043	JMP2	0044	STRTC	0045	JNT1	0046	SWAPA	0047
ORLAR0	0048	ORLAR1	0049	ORLAR2	004A	ORLAR3	004B	ORLAR4	004C
ORLAR5	004D	ORLAR6	004E	ORLAR7	004F	TMRPSCAL	0050	CHAR50	0050
ANLA.R0	0050	ANLA.R1	0051	JB2	0052	ANLA	0053	CALL2	0054
STRTT	0055	JT1	0056	DAA	0057	ANLAR0	0058	ANLAR1	0059
PATRN1	005A	ANLAR2	005A	EI01	005B	ANLAR3	005B	ANLAR4	005C
ANLAR5	005D	ANLAR6	005E	ANLAR7	005F	ADDA.R0	0060	ADDA.R1	0061
MOVTA	0062	JMP3	0064	EI02	0064	STOPTC	0065	RRCA	0067
ADDAR0	0068	PATRN3	0069	ADDAR1	0069	ADDAR2	006A	ADDAR3	006B
ADDAR4	006C	ADDAR5	006D	ADDAR6	006E	ADDAR7	006F	ADDCA.R0	0070
ADDCA.R1	0071	JB3	0072	CALL3	0074	ENT0	0075	JF1	0076
RRA	0077	ADDCAR0	0078	ADDCAR1	0079	ADDCAR2	007A	ADDCAR3	007B
ADDCAR4	007C	ADDCAR5	007D	ADDCAR6	007E	PMEMENBL	007F	OCTLENBL	007F
LEDMSK	007F	IRQSFOFF	007F	ASCIMASK	007F	ADDCAR7	007F	PMEMDSBL	0080
OCTLDSBL	0080	MOVXA.R0	0080	LEDBIT	0080	IRQSFON	0080	CHAR80	0080
RTN1	0081	MOVXA.R1	0081	RTN2	0082	RTN3	0083	RET	0083
EI03	0083	JMP4	0084	CLRF0	0085	JNI	0086	ORLBUS	0088
ORLP1	0089	ORLP2	008A	ORLDP4A	008C	ORLDP5A	008D	ORLDP6A	008E
ORLDP7A	008F	MOVX.R0A	0090	MOVX.R1A	0091	JB4	0092	RETR	0093
CALL4	0094	CPLF0	0095	PW1VAL	0096	PATRN4	0096	JNZ	0096
EI04	0096	CLRC	0097	ANLBUS	0098	TESTLED1	0099	ANLP1	0099
ANLP2	009A	ANLDP4A	009C	ANLDP5A	009D	ANLDP6A	009E	ANLDP7A	009F
MOV.R0A	00A0	MOV.R1A	00A1	MOVPA.A	00A3	TESTL01	00A4	JMP5	00A4
PW2VAL	00A5	PATRN2	00A5	CLRF1	00A5	CPLC	00A7	MOVR0A	00A8
TESTLED2	00A9	MOVR1A	00A9	MOVR2A	00AA	MOVR3A	00AB	MOVR4A	00AC
MOVR5A	00AD	MOVR6A	00AE	MOVR7A	00AF	MOV.R0	00B0	MOV.R1	00B1
JB5	00B2	JMPP.A	00B3	TESTL02	00B4	CALL5	00B4	CPLF1	00B5
JF0	00B6	MOVR0	00B8	MOVR1	00B9	MOVR2	00BA	MOVR3	00BB
MOVR4	00BC	MOVR5	00BD	MOVR6	00BE	PAUSFOFF	00BF	MOVR7	00BF
JMP6	00C4	SELRB0	00C5	JZ	00C6	MOVAPSW	00C7	DECR0	00C8
DECR1	00C9	DECR2	00CA	DECR3	00CB	DECR4	00CC	DECR5	00CD
DECR6	00CE	DECR7	00CF	XRLA.R0	00D0	XRLA.R1	00D1	JB6	00D2
XRLA	00D3	CALL6	00D4	SELRB1	00D5	MOVPSWA	00D7	XRLAR0	00D8
XRLAR1	00D9	XRLAR2	00DA	XRLAR3	00DB	XRLAR4	00DC	XRLAR5	00DD
XRLAR6	00DE	XRLAR7	00DF	CPYSFOFF	00DF	MOVP3A.A	00E3	JMP7	00E4
SELMB0	00E5	JNC	00E6	RLA	00E7	DJNZR0	00E8	DJNZR1	00E9
DJNZR2	00EA	DJNZR3	00EB	DJNZR4	00EC	DJNZR5	00ED	DJNZR6	00EE
MSGSF0FF	00EF	IDATENBL	00EF	DJNZR7	00EF	MOVA.R0	00F0	MOVA.R1	00F1
JB7	00F2	CALL7	00F4	SELMB1	00F5	JC	00F6	RLCA	00F7
RDLEDOFF	00F7	OVFSFOFF	00F7	MOVAR0	00F8	MOVAR1	00F9	MOVAR2	00FA

OLATENBL	00FB	MOVAR3	00FB	MOVAR4	00FC	PDATENBL	00FD	MOVAR5	00FD
MOVAR6	00FE	ADR08OFF	00FE	NEGONE	00FF	MOVAR7	00FF	TIME3CNT	0100
PAGE1	0100	CHECKT0	0100	CT01	010C	CT02	011B	CT03	0122
CT04	012D	CT05	0134	CT06	013D	CT07	0143	CT08	0148
CT09	014A	DEBOUNCE	014B	DB01	0150	WAIT1SEC	0155	W1S01	015A
TIME4CNT	0180	PAGE2	0200	MEM2OUT	0200	MO01	0212	MO02	021F
MO03	0228	MO04	0247	MO05	0253	MO06	0254	MO07	0255
MO08	025B	MO09	025D	MO10	026B	PAGE3	0300	DMESG1	0300
DMESG2	034F	DMESG3	0361	DMESG4	038B	DMESG5	03B7	TMESG1	03C6
TMESG2	03D8	EMESG1	03EA	PAGE4	0400	INIT	0400	INIT01	0412
INIT02	0454	INIT03	045C	INIT04	0462	PRNTMSG	0493	PM01	0495
PM02	049C	PM03	049E	PM04	04A1	PM05	04AB	PRNTASCI	04B3
PA01	04B6	PA02	04C7	PRINT	04CD	PR01	04D3	PR02	04E5
PAGE5	0500	DIAGS	0500	DIAGS1	050E	DIAGS2	0513	WRITBLKS	0526
READBLKS	0569	RBS01	05B4	TESTMEM	0600	PAGE6	0600	TM01	065F
TESTMSG	0660	WRITPAGE	0700	PAGE7	0700	WP01	0703	WP02	070A
WP03	0713	READPAGE	071E	RP01	0721	RP02	0728	RP03	0731
RP04	073F	WRITBLK	0744	WB01	0746	READBLK	0751	RB01	0753
RB02	0760	TOGGLED	0761	TL01	0778	TL02	077E	TMRTIM	0F64
TIME2VAL	FA00	TIME4	FE80	TIME3	FF00	TIME1	FFCF	TIME2	FFF0