```
*****************************************************************************
*
* PLC.ASM      : Package for 16/8 bit PLC system w/LC Display & Keyboard
* written by   : S.J. M.
* copyright owners : S.J. M./Vortex Research Inc/Procyon Research CC (subsidiary)
* date started  : June 12, 1990, finish July 19, 1990.
*
* System Description,  Small PLC with 16 outputs, 8 inputs and a real time
*                      onboard timer generated clock. Unit is user
*                      programmable and has LC Display and Keyboard as the
*                      user programming interface. Has onboard serial
*                      channel for networked operation and 16 relay outputs
*                      or as required up to 16 maximum. Uses 2 12 vdc
*                      batteries for power or AC/DC on board converter for
*                      +/- 12 vdc and + 5vdc. Expected retail price will be
*                      approx. R470 +/- R15.
*
* User Programming,    In format as follows,
*
*                      O1(N) = I1(N)+I2(F)+I7(N)+T(14:22)+D(45.0)
*
*                      The above indicates that OUTPUT 1 will turn on when
*                      INPUT 1 is high and INPUT 2 is low and INPUT 7 is high
*                      and T(time) is 14:22 and will run for a D(duration) of
*                      45 minutes.
*
*                      O2(F) = I2(F)+I3(O)/I5(O)+I6(F)/T(15:44)
*
*                      Output 2 will turn off if Input 2 is off and Input 3
*                      is on or if Input 5 is on and Input 6 is off or if the
*                      time is 15:44.
*
* KEY DESIGNATIONS,
*
*              0 - equals zero
*              1 - equals one          and  INPUT
*              2 - equals two          and  OUTPUT
*              3 - equals three        and  TIME
*              4 - equals four         and  DURATION
*              5 - equals five
*              6 - equals six          and  ON
*              7 - equals seven        and  OFF
*              8 - equals eight        and  AND
*              9 - equals nine         and  OR
*              A - equals ten          and  ADDRESS
*              B - equals eleven       and  SET TIME/DATE
*              C - equals twelve       and  BACKSPACE
*              D - equals thirteen     and  PLC MODE
*              E - equals fourteen     and  PROGRAM MODE
*              F - equals fifteen      and  TEST MODE
*
* DEVICE DESIGNATIONS,
*
*       P4 = DATA     APORT - A0 TO A3, KEYBOARD ROW INPUTS 0 TO 3
*       P5 = DDR      APORT - A4, Keyboard Scan Output 0
*                     APORT - A5, RXD (RECEIVE DATA) FROM SERIAL PORT
*                     APORT - LC Display RS Output, Register Select
*                     APORT - Keyboard Keypressed Beep Output
*       P6 = DATA     BPORT - B0, Keyboard Scan Output 1
*                     BPORT - B1, Keyboard Scan Output 2
*                     BPORT - B2, Keyboard Scan Output 3
*                     BPORT - B3, TXD (TRANSMIT DATA) TO SERIAL PORT
*                     BPORT - B4 TO B7, RESERVED BY FULL EXP. MODE
*                     CPORT - C0 TO C7, DATA/ADDRESS BITS AD0 TO AD7
```

```
*                       DPORT - D0 TO D7, ADDRESS BITS A8 TO A15
*
* MEM LOCATIONS   ;  PIA,          0200 : PIA APORT (PORT1) Output
*                                  0201 : PIA BPORT (PORT2) Output
*                                  0202 : PIA CPORT (PORT3) Input
*                                  0203 : PIA Control Register
*                       OTHER,     0220 : Display Select (on LOW)
*                                  0240 : Keyboard Select (on LOW)
*
********************************************************************************
************************ system initialization ******************************
*
** device definitions and logical locations for TMS77C82 processor
*
IOCNT0     EQU     P0                          ;I/O control register 0
IOCNT2     EQU     P1                          ;I/O control register 2
IOCNT1     EQU     P2                          ;I/O control register 1
RESER0     EQU     P3                          ;TI reserved Port
APORT      EQU     P4                          ;A port data register
ADDR       EQU     P5                          ;A port direction register, 0=input/1=output
BPORT      EQU     P6                          ;B port data register, input only
RESER1     EQU     P7                          ;TI reserved Port
CPORT      EQU     P8                          ;C port data register
CDDR       EQU     P9                          ;C port direction register, 0=input/1=output
DPORT      EQU     P10                         ;D port data register
DDDR       EQU     P11                         ;D port direction register, 0=input/1=output
T1MSDA     EQU     P12                         ;counter T1 MSB data preload
T1LSDA     EQU     P13                         ;counter T1 LSB data preload
T1CTL1     EQU     P14                         ;counter T1 control/MSB capture/cascade control
T1CTL0     EQU     P15                         ;counter T1 control/LSB capture/prescale
preloader
T2MSDA     EQU     P16                         ;counter T2 MSB data preload
T2LSDA     EQU     P17                         ;counter T2 LSB data preload
T2CTL1     EQU     P18                         ;counter T2 control/MSB capture/cascade control
T2CTL0     EQU     P19                         ;counter T2 control/LSB capture/prescale
preloader
SCMODE     EQU     P20                         ;serial port mode control
SCTL0      EQU     P21                         ;serial port control register 0
SSTAT      EQU     P22                         ;serial port status register
T3DATA     EQU     P23                         ;counter T3 data preload, used also to set baud
rate
SCTL1      EQU     P24                         ;serial port control register 1
RXBUF      EQU     P25                         ;serial data receive buffer
TXBUF      EQU     P26                         ;serial data transmit buffer
*
** reserved areas
*
* P27 to P35 are reserved ports / TI
*
** additional porting
*
* P36 to P255 are not available in Single Chip Mode only.
*
** device definitions for external and memory related locations
*
BASEAD     EQU     R0                          ;Base Address of Registers
BREG       EQU     R1
MISC1      EQU     R2
MISC2      EQU     R3
TEMP       EQU     R4
TEMP1      EQU     R5
TEMP2      EQU     R6
TEMP3      EQU     R7
ADDRES     EQU     R8                          ;address of device
```

```
CKBTHH    EQU     R9                  ;MSB hours
CKBTHL    EQU     R10                 ;MSB hours
CKBTMH    EQU     R11                 ;MSB minutes
CKBTML    EQU     R12                 ;MSB minutes
CKBTSH    EQU     R13                 ;MSB seconds
CKBTSL    EQU     R14                 ;MSB seconds
BYTETH    EQU     R15
BYTETM    EQU     R16
YEARH     EQU     R17
YEARL     EQU     R18
MONTH     EQU     R19
DATE      EQU     R20
PBMODE    EQU     R21
COUNTX    EQU     R22
COUNT     EQU     R23
NUMBER    EQU     R24
OUT1ST    EQU     R25
OUT2ST    EQU     R26
INP1ST    EQU     R27
POINTR    EQU     R28
LOCAL     EQU     R29
INDEX     EQU     R30
ALCONT    EQU     R31
LDTBLH    EQU     R32
LDTBLL    EQU     R33
SECOND    EQU     R34
BKFLAG    EQU     R35
*
** PLC programming registers
*
ANDIN0    EQU     R36                 ;AND input byte and turn on/off
ANDOT1    EQU     R37                 ;AND output byte 1 and turn on/off
ORXIN1    EQU     R38                 ;OR input byte and turn on/off
ORXOT1    EQU     R39                 ;OR output 1 and turn on/off
I1STAT    EQU     R40                 ;input on/off set condition
O1STAT    EQU     R41                 ;output on/off set condition
TIMONH    EQU     R42                 ;turn on based on time
TIMONL    EQU     R43                 ;turn off based on time
TIMOFH    EQU     R44                 ;turn on based on time
TIMOFL    EQU     R45                 ;turn off based on time
*
** backflush programmimg registers
*
SDAY1     EQU     R126
SMTH1     EQU     R127
EDAY1     EQU     R128
EMTH1     EQU     R129
SPACH1    EQU     R130
SPACL1    EQU     R131
DURON1    EQU     R132
DUROF1    EQU     R133
DECTH1    EQU     R134
DECTL1    EQU     R135
SDAY2     EQU     R136
SMTH2     EQU     R137
EDAY2     EQU     R138
EMTH2     EQU     R139
SPACH2    EQU     R140
SPACL2    EQU     R141
DURON2    EQU     R142
DUROF2    EQU     R143
DECTH2    EQU     R144
DECTL2    EQU     R145
SDAY3     EQU     R146
```

```
SMTH3    EQU    R147
EDAY3    EQU    R148
EMTH3    EQU    R149
SPACH3   EQU    R150
SPACL3   EQU    R151
DURON3   EQU    R152
DUROF3   EQU    R153
DECTH3   EQU    R154
DECTL3   EQU    R155
SDAY4    EQU    R156
SMTH4    EQU    R157
EDAY4    EQU    R158
EMTH4    EQU    R159
SPACH4   EQU    R160
SPACL4   EQU    R161
DURON4   EQU    R162
DUROF4   EQU    R163
DECTH4   EQU    R164
DECTL4   EQU    R165
*
PORT1    EQU    >0200
PORT2    EQU    >0201
PORT3    EQU    >0202
CNTRL    EQU    >0203
DSPLAY   EQU    >0220                  ;display address
KEYBRD   EQU    >0240                  ;keyboard address
*
*******************************************************************************
************************ program start location ******************************
*
         AORG   >E100                  ;first 6 bytes reserved by Texas Instruments
*                                      ; at E000 to E005
*
** Step 1. : clear processor internal memory and load stack
*
START    DINT                          ;kill the interrupts to be safe
         MOV    %23,B
         LDSP
         CLR    A
STARTX   PUSH   A
         INC    B
         CMP    %>FF,B
         JNE    STARTX
         CLR    R255
*
         MOV    %>C4,B                 ;(C0) set up a stack value at R192
         LDSP                          ; set stack at 44 (upper limit of chip RAM)
         CLR    A
         CLR    B
         CLR    R2
*
*
** Step 2. : Initialize and setup memory modes, etc.
*
WHAT     MOVP   %?10001100,IOCNT0      ;initialize Port 0, 10111011, iocnt0
*                                      ; full expansion mode
*                                      ;  int3 and int1 cleared and enabled
         MOVP   %?00001111,IOCNT1      ;initialize Port 2, 00001010, iocnt1
*                                      ; int5 and int4 cleared and disabled
         MOVP   %?00100010,IOCNT2      ;initialize Port 1, 00100010, iocnt2
*                                      ; int3 edge sensitive, active falling
*                                      ;  int1 edge sensitive, active falling
*
*
```

```
** start timer 1, int 2 for 1 second RTM mode
*
            MOVP    %>B7,T1MSDA             ;
            MOVP    %>1B,T1LSDA             ;
            MOVP    %>00,T1CTL1             ; set preload value to value in A
            MOV     %>00,A                  ;  move prescal value into A
            OR      %?10011111,A            ;  set start timer bit
            MOVP    A,T1CTL0                ;  set prescaler to value in A & start timer
*
** setup port direction registers and clear outputs
*
            MOVP    %>00,APORT              ;clear outputs on Port A, APORT
            MOVP    %?11010000,ADDR         ;initialize Port A, DDR register
            MOVP    %>00,APORT              ;make sure no outputs on APORT
            MOVP    %?11110000,BPORT        ;clear ouputs on BPORT
*
** setup serial port transception modes, speed set on CONFIG0 MSN
*
*           MOVP    %?01000000,SCTL0        ;set serial control
*           MOV     %>00,A                  ;  get the prescale value
*           MOV     %12,A                   ;get the timer value
*           MOVP    A,T3DATA                ; set preload value to value in A
*           MOV     %?01001010,A
*           MOVP    A,SCMODE                ;8N1, asyncronous, motorola mode
*           MOVP    %?00000101,SCTL0        ;enable serial channel
*           CLR     A
*           OR      %?11000000,A
*           MOVP    A,SCTL1                 ;  set prescaler to A & start timer
*
*
** initialise 8255 PIA for 2 (A & B) output ports and 1 (C) input port
*
            MOV     %?10001001,R0
            STA     @CNTRL
            STA     @CNTRL
            CALL    @WAITX
            CLR     A
            STA     @PORT1
            CALL    @WAITX
            STA     @PORT2
            CALL    @WAITX
*
** initialise the LC display
*
            ANDP    %?10111111,APORT
            CALL    @WAIT
            MOV     %?00000001,A            ;clear display and home position
            CALL    @DSCONT
            CALL    @WAITX
            CALL    @WAITX
            CALL    @WAITX
            MOV     %?00111000,A            ;function set, 8 bits, 2 lines, 5x7 dots
            CALL    @DSCONT
            CALL    @WAITX
            MOV     %?00111000,A            ;function set, 8 bits, 2 lines, 5x7 dots
            CALL    @DSCONT
            CALL    @WAITX
            MOV     %?00001000,A            ;display off, cursor off, cursor blink off
            CALL    @DSCONT
            MOV     %?00000110,A            ;entry mode, increment +1, no shift
            CALL    @DSCONT
            MOV     %?00010100,A            ;cursor, no shift, to right
            CALL    @DSCONT
            MOV     %?10000000,A            ;set cg ram address
```

```
            CALL    @DSCONT
            MOV     %?00001100,A            ;display on, cursor on, cursor blink on
            CALL    @DSCONT
            MOV     %?00000010,A            ;return cursor to home position
            CALL    @DSCONT
            CALL    @WAITX
            CALL    @WAITX
            CALL    @WAITX
*
            MOVD    %THE,LDTBLL
            CALL    @SHSTRG
*
            MOV     %?11000000,A            ;set cg ram address
            CALL    @DSCONT
*
            MOVD    %ONBORD,LDTBLL
            CALL    @SHSTRG
*
            CALL    @WAITY
            CALL    @WAITY
            CALL    @WAITY
            CALL    @WAITY
            CALL    @WAITY
            CALL    @DSCLER
*
            MOVD    %FOR,LDTBLL
            CALL    @SHSTRG
*
            MOV     %?11000000,A            ;set cg ram address
            CALL    @DSCONT
*
            MOVD    %CUSTOM,LDTBLL
            CALL    @SHSTRG
*
            CALL    @WAITY
            CALL    @WAITY
            CALL    @WAITY
            CALL    @WAITY
            CALL    @WAITY
            CALL    @DSCLER
            CALL    @STOCK
            MOV     %>FF,NUMBER
            CALL    @EXITQ
*
** get into control loop
*
            BR      @CONTRL
*
** show string routine
*
SHSTRG      PUSH    B
SHSTR1      LDA     *LDTBLL
            CMP     %'\',A
            JEQ     SHSTR0
            CALL    @DSCHAR
            ADD     %>01,LDTBLL
            ADC     %>00,LDTBLH
            JMP     SHSTR1
SHSTR0      POP     B
            RETS
*
** clear the LC display and home cursor
*
DSCLER      MOV     %?00001111,A            ;display on, cursor on, cursor blink on
```

```
            CALL    @DSCONT
            MOV     %?10000000,A            ;set cg ram address
            CALL    @DSCONT
            MOV     %?00000001,A            ;return cursor to home position
            CALL    @DSCONT
            CALL    @WAITX
            CALL    @WAITX
            MOV     %?00000010,A            ;return cursor to home position
            CALL    @DSCONT
            CALL    @WAITX
            RETS
*
** display a character on the LC Display
*
DSCHAR      ORP     %?01000000,APORT
            STA     @DSPLAY
            CALL    @WAIT
            ANDP    %?10111111,APORT
            RETS
*
** send a control to the LC Display
*
DSCONT      ANDP    %?10111111,APORT
            STA     @DSPLAY
            CALL    @WAIT
            RETS
*
*
STOCK       CALL    @DSCLER
            MOV     %?11000000,A            ;set cg ram address
            CALL    @DSCONT
            MOVD    %RUNING,LDTBLL
            CALL    @SHSTRG
*
            MOV     %?10000000,A            ;set cg ram address
            CALL    @DSCONT
            MOVD    %COMAND,LDTBLL
            CALL    @SHSTRG
            ANDP    %?10111111,APORT
            RETS
*
*
*************************************************************************
********************** main control loop for system ************************
*
CONTRL      EINT
CONTRX      MOV     %>FF,NUMBER
            CALL    @SCANB                  ;look for a character at the keyboard
            CMP     %>F0,NUMBER
            JHS     LOOP3
            BR      @DOFUNC
LOOP3       CMP     %00,PBMODE
            JEQ     LOOP5
            CMP     %00,BKFLAG
            JEQ     LOOP5
            CALL    @BFXLSH
            CLR     BKFLAG
LOOP5       CALL    @SCANA                  ;update the outputs
            CALL    @SCAND                  ;get the 8 inputs
LOOP4       BR      @CONTRX
*
** routine called by RTM interrupt to update time on a one minute basis
** routine called 60 times per minute based on timer 1
** routine displays time information on LC Display module
```

```
** EXITQ is used to display time at any time desired in procedures and
** TIME is used by interrupt servicing routine.
*
EXITR   BR      @EXIT
*
TIME    INC     SECOND
        INC     CKBTSL
        CMP     %10,CKBTSL
        JL      EXITR
        CLR     CKBTSL
        INC     CKBTSH
        CMP     %06,CKBTSH
        JL      EXITR
        CLR     CKBTSH
        INC     CKBTML
        CMP     %10,CKBTML
        JL      EXITR
        CLR     CKBTML
        INC     CKBTMH
        CMP     %06,CKBTMH
        JL      EXITR
        CLR     CKBTMH
        INC     CKBTHL
        CMP     %04,CKBTHL
        JL      EXITR
        CMP     %02,CKBTHH
        JEQ     EXITY
        CMP     %10,CKBTHL
        JL      EXIT
        CLR     CKBTHL
        INC     CKBTHH
        CMP     %02,CKBTHH
        JL      EXIT
EXITY   MOVD    %>0000,CKBTHL
        MOVD    %>0000,CKBTML
        MOV     MONTH,B
        LDA     @DAYTBL(B)
        INC     DATE
        CMP     A,DATE
        JL      EXIT
        CLR     DATE
        INC     MONTH
        CMP     %12,MONTH
        JL      EXIT
        CLR     MONTH
        INC     YEARL
        CMP     %100,YEARL
        JL      EXIT
        CLR     YEARL
        INC     YEARH
EXIT    CMP     %>FF,NUMBER
        JNE     EXITT
EXITS   CMP     %>FF,NUMBER
        JNE     EXITT
EXITQ   MOV     %?11001000,A
        CALL    @DSCONT
        MOV     CKBTHH,A
        ADD     %>30,A
        CALL    @DSCHAR
        MOV     CKBTHL,A
        ADD     %>30,A
        CALL    @DSCHAR
        MOV     %':',A
        CALL    @DSCHAR
```

```
            MOV     CKBTMH,A
            ADD     %>30,A
            CALL    @DSCHAR
            MOV     CKBTML,A
            ADD     %>30,A
            CALL    @DSCHAR
            MOV     %':',A
            CALL    @DSCHAR
            MOV     CKBTSH,A
            ADD     %>30,A
            CALL    @DSCHAR
            MOV     CKBTSL,A
            ADD     %>30,A
            CALL    @DSCHAR
            MOV     %?10001010,A
            CALL    @DSCONT
EXITT       RETS
*
** converts 4 bytes of decimal time re. 20:11 to hexidecimal minutes
** routine only used for the conversion of data from the RTM system (Time)
** result is used for TURN ON/TURN OFF TIME test by PLC algorithms
*
TMCNVT      MOVD    %>0000,MISC2
            CLR     B
            MPY     %10,CKBTHH              ;put msb hours into mpy reg
            MOV     B,MISC2                 ;copy it to temp reg
            ADD     CKBTHL,MISC2            ;add the lsb hours to get total
            CLR     B
            MPY     %10,CKBTMH              ;put msb minutes into mpy reg
            MOV     B,MISC1                 ;copy it to temp reg
            ADD     CKBTML,MISC1            ;add the lsb minutes to get total
*
            CLR     B
            CLR     ALCONT                  ;clear a reg
TMCNVX      ADD     %60,B
            ADC     %0,A
            INC     ALCONT
            CMP     MISC2,ALCONT
            JL      TMCNVX
*
            ADD     MISC1,B                 ;add leftover minutes
            ADC     %0,A                    ;add carry in case of overflow on lsb
            MOVD    B,BYTETM                ;copy result to byte time register
            CLR     ALCONT
            RETS
*
** converts 4 bytes of decimal time re. 20:11 to hexidecimal minutes
** routine only used for the conversion of data from the RTM system (Time)
** result is used for TURN ON/TURN OFF TIME test by PLC algorithms
*
ALCNVT      MOVD    %>0000,MISC2
            CLR     B
            MPY     %10,TEMP                ;put msb hours into mpy register
            MOV     B,MISC2                 ;copy it to temp register
            ADD     TEMP1,MISC2             ;add the lsb hours to get total
            CLR     B
            MPY     %10,TEMP2               ;put msb minutes into mpy register
            MOV     B,MISC1                 ;copy it to temp register
            ADD     TEMP3,MISC1             ;add the lsb minutes to get total
*
            PUSH    ADDRES
            CLR     B
            CLR     ADDRES                  ;clear a reg
ALCNVX      ADD     %60,B
```

```
            ADC     %0,A
            INC     ADDRES
            CMP     MISC2,ADDRES
            JL      ALCNVX
            POP     ADDRES
*
            ADD     MISC1,B                 ;add leftover minutes
            ADC     %0,A                    ;add carry in case of overflow on lsb
            MOVD    B,MISC2                 ;copy result to byte time register
            RETS                            ;return to sender
*
*******************************************************************************
** test the keyboard for a character present and return a value
*
BAARGH      POP     B
            RETS
*
SCANB       PUSH    B
            ANDP    %?11101111,APORT
            ORP     %?00010000,APORT
            ANDP    %?11111000,BPORT
            ORP     %?00000111,BPORT
            CALL    @WAITX                  ;10 ms total debounce
            CALL    @WAITX
            CALL    @WAITX
            MOVP    APORT,A
            AND     %>0F,A
            CMP     %>00,A
            JEQ     BAARGH
            ANDP    %?11111000,BPORT
            ANDP    %?11101111,APORT
            ORP     %?00010000,APORT        ;set up list base address, list 2
            MOVP    APORT,A
            AND     %>0F,A
            ANDP    %?11101111,APORT
            CMP     %>00,A
            JEQ     SCANB0
            MOV     A,B
            LDA     @CHAR00(B)
            DEC     A
            ADD     %0,A
            MOV     A,B
            LDA     @CHAR01(B)
            MOV     A,NUMBER
            POP     B
            RETS
*
SCANB0      ANDP    %?11111000,BPORT
            ORP     %?00000001,BPORT        ;set up list base address, list 2
            MOVP    APORT,A
            AND     %>0F,A
            ANDP    %?11111000,BPORT
            CMP     %>00,A
            JEQ     SCANB1
            MOV     A,B
            LDA     @CHAR00(B)
            DEC     A
            ADD     %>04,A
            MOV     A,B
            LDA     @CHAR01(B)
            MOV     A,NUMBER
            POP     B
            RETS
*
```

```
SCANB1  ANDP    %?11111000,BPORT
        ORP     %?00000010,BPORT        ;set up list base address, list 2
        MOVP    APORT,A
        AND     %>0F,A
        ANDP    %?11111000,BPORT
        CMP     %>00,A
        JEQ     SCANB2
        MOV     A,B
        LDA     @CHAR00(B)
        DEC     A
        ADD     %>08,A
        MOV     A,B
        LDA     @CHAR01(B)
        MOV     A,NUMBER
        POP     B
        RETS
*
SCANB2  ANDP    %?11111000,BPORT
        ORP     %?00000100,BPORT        ;set up list base address, list 2
        MOVP    APORT,A
        AND     %>0F,A
        ANDP    %?11111000,BPORT
        CMP     %>00,A
        JEQ     SCANB3
        MOV     A,B
        LDA     @CHAR00(B)
        DEC     A
        ADD     %>0C,A
        MOV     A,B
        LDA     @CHAR01(B)
        MOV     A,NUMBER
SCANB3  POP     B
        RETS                            ;return to sender
*
*
*******************************************************************************
*
*** 8 bit wait period, value in R39 (byte), 116 us to perform
*
WAIT    MOV     %>1F,COUNT              ;9 cycles, 3 us
WAIT00  DEC     COUNT                   ;7 cycles, 2.3 us
        CMP     %>00,COUNT              ;9 cycles, 3 us -----> loop=6.9 us
        JNE     WAIT00                  ;5 cycles, 1.6 us
        RETS                            ;7 cycles, 2.3 us
*
*
*** 16 bit wait period, value in R38/R39 (word), 3.29 ms to perform
*
WAITX   MOVD    %>0FFF,COUNT            ;15 cycles, 5 us
WAITX0  DECD    COUNT                   ;11 cycles, 3.6 us
        CMP     %>00,COUNT              ;9 cycles, 3 us -----> loop=8.2 us
        JNE     WAITX0                  ;5 cycles, 1.6 us
        CMP     %>00,COUNTX             ;9 cycles, 3 us -----> loop=12.8 us
        JNE     WAITX0                  ;5 cycles, 1.6 us
        RETS                            ;7 cycles, 2.3 us
*
*** 16 bit wait period, value in R38/R39 (word), 3.29 ms to perform
*
WAITS   MOVD    %>7FFF,COUNT            ;15 cycles, 5 us
        JMP     WAITX0
*
*** 16 bit wait period, value in R38/R39 (word), 536 ms to perform
*
WAITY   MOVD    %>FFFF,COUNT            ;512 uS/FF count * 256 loops = AP. 128 mS
```

```
            JMP     WAITX0
*
*
*******************************************************************************
*** what are we being asked (get command) to do and do it
*
DOFUNC      PUSH    B
            MOV     NUMBER,B
            CMP     %>0A,B
            JL      COMMAD
            CMP     %>10,B
            JHS     COMMAD
            ORP     %?10000000,APORT
            CALL    @WAITX
            CALL    @WAITX
            CALL    @WAITX
            ANDP    %?01111111,APORT
            MOV     NUMBER,B
            MOV     %>03,A
            MPY     A,B
            BR      @DOCOMM(B)
TAKEOF      MOV     %>FF,NUMBER
            ANDP    %?00111111,APORT
            CALL    @DSCLER
            CALL    @STOCK
            CALL    @EXITS
COMMAD      POP     B
            BR      @LOOP3
*
*
DOCOMM      BR      @COM0                   ;
            BR      @COM1                   ;send out a single alarm
            BR      @COM2                   ;
            BR      @COM3                   ;
            BR      @COM4                   ;
            BR      @COM5                   ;
            BR      @COM6                   ;
            BR      @COM7                   ;
            BR      @COM8                   ;
            BR      @COM9                   ;
            BR      @COM10                  ;
            BR      @COM11                  ;
            BR      @COM12                  ;
            BR      @COM13                  ;
            BR      @COM14                  ;
            BR      @COM15                  ;
*
COM0        MOV     %'0',A
            CALL    @DSCHAR
            BR      @TAKEOF
*
COM1        MOV     %'1',A
            CALL    @DSCHAR
            BR      @TAKEOF
*
COM2        MOV     %'2',A
            CALL    @DSCHAR
            BR      @TAKEOF
*
COM3        MOV     %'3',A
            CALL    @DSCHAR
            BR      @TAKEOF
*
COM4        MOV     %'4',A
```

```
            CALL    @DSCHAR
            BR      @TAKEOF
*
COM5        MOV     %'5',A
            CALL    @DSCHAR
            BR      @TAKEOF
*
COM6        MOV     %'6',A
            CALL    @DSCHAR
            BR      @TAKEOF
*
COM7        MOV     %'7',A
            CALL    @DSCHAR
            BR      @TAKEOF
*
COM8        MOV     %'8',A
            CALL    @DSCHAR
            BR      @TAKEOF
*
COM9        MOV     %'9',A
            CALL    @DSCHAR
            BR      @TAKEOF
*
** set device address
*
COM10       CALL    @DSCLER
*
            MOVD    %ADDRSS,LDTBLL
            CALL    @SHSTRG
*
            CLR     B                       ;clear input counter
            CLR     TEMP
            CLR     TEMP1
            CLR     TEMP2
COM101      CALL    @SCANF                  ;get a character (0 to F)
            CMP     %>0C,NUMBER
            JNE     COM106
            CMP     %>00,B
            JEQ     COM106
            CALL    @BKSPAC
            DEC     B
            JMP     COM101
COM106      CMP     %>0A,NUMBER             ;test for F (enter)
            JHS     COM101                  ;if F then exit
            MOV     NUMBER,A                ;copy character to A reg for offsetting
            STA     @TEMP(B)                ;save char at temp+offset
            OR      %>30,A                  ;make character ASCII for display
            CALL    @DSCHAR                 ;send charater to display
            INC     B                       ;inc input counter
            CMP     %3,B                    ;test for max address of three bytes 255
            JHS     COM102                  ;if higher or same then exit
            JMP     COM101                  ;if lower and no 'enter' then get next char
COM102      MOV     TEMP2,ADDRES
            MOV     TEMP1,R1
            MPY     %10,R1
            CLR     A
            ADD     R1,ADDRES
            CLR     A
            MOV     TEMP,R1
            MPY     %100,R1
            ADD     R1,ADDRES
            JNC     COM107
*
            CALL    @DSCLER
```

```
*
            MOVD    %ERROR,LDTBLL
            CALL    @SHSTRG
*
            CALL    @WAITY
            CALL    @WAITY
            ANDP    %?10111111,APORT
            RETS
*
COM107      CALL    @WAITY
            CALL    @WAITY
            BR      @TAKEOF
*
*
** set local clock (base time in hours and minutes)
*
COM11       CALL    @DSCLER
*
            MOVD    %SETDAT,LDTBLL
            CALL    @SHSTRG
*
            MOV     %?11000010,A
            CALL    @DSCONT
*
            CLR     B                       ;clear input counter
COM110      CALL    @SCANF                  ;get a character (0 to F)
            CMP     %>0C,NUMBER
            JNE     COM111
            CMP     %>00,B
            JEQ     COM111
            CALL    @BKSPAC
            DEC     B
            JMP     COM110
COM111      CMP     %>0A,NUMBER             ;test for F (enter)
            JHS     COM110                  ;if F then exit
            MOV     NUMBER,A                ;copy character to A reg for offsetting
            STA     @MISC1(B)               ;save char at temp+offset
            OR      %>30,A                  ;make character ASCII for display
            CALL    @DSCHAR                 ;send charater to display
            INC     R1                      ;inc input counter
            CMP     %>02,R1                 ;test for max address of three bytes 255
            JHS     COM112                  ;if higher or same then exit
            JMP     COM110                  ;if lower and no 'enter' then get next char
COM112      MOV     %'/',A
            CALL    @DSCHAR
            MPY     %10,MISC1
            CLR     A
            DEC     MISC2
            ADD     MISC2,BREG
            CMP     %31,BREG
            JHS     COM11
            MOV     BREG,DATE
*
            CLR     B                       ;clear input counter
COM113      CALL    @SCANF                  ;get a character (0 to F)
            CMP     %>0C,NUMBER
            JNE     COM114
            CMP     %>00,B
            JEQ     COM114
            CALL    @BKSPAC
            DEC     B
            JMP     COM113
COM114      CMP     %>0A,NUMBER             ;test for F (enter)
            JHS     COM113                  ;if F then exit
```

```
        MOV     NUMBER,A        ;copy character to A reg for offsetting
        STA     @MISC1(B)       ;save char at temp+offset
        OR      %>30,A          ;make character ASCII for display
        CALL    @DSCHAR         ;send character to display
        INC     R1              ;inc input counter
        CMP     %>02,R1         ;test for max address of three bytes 255
        JHS     COM115          ;if higher or same then exit
        JMP     COM113          ;if lower and no 'enter' then get next char
COM115  MOV     %'/',A
        CALL    @DSCHAR
        MPY     %10,MISC1
        CLR     A
        DEC     MISC2
        ADD     MISC2,BREG
        CMP     %12,BREG
        JL      COMX16
        BR      @COM11
COMX16  MOV     BREG,MONTH
*
        CLR     R1              ;clear input counter
COM116  CALL    @SCANF          ;get a character (0 to F)
        CMP     %>0C,NUMBER
        JNE     COM117
        CMP     %>00,B
        JEQ     COM117
        CALL    @BKSPAC
        DEC     B
        JMP     COM116
COM117  CMP     %>0A,NUMBER     ;test for F (enter)
        JHS     COM116          ;if F then exit
        MOV     NUMBER,A        ;copy character to A reg for offsetting
        STA     @MISC1(B)       ;save char at temp+offset
        OR      %>30,A          ;make character ASCII for display
        CALL    @DSCHAR         ;send charater to display
        INC     R1              ;inc input counter
        CMP     %>02,R1         ;test for max address of three bytes 255
        JHS     COM118          ;if higher or same then exit
        JMP     COM116          ;if lower and no 'enter' then get next char
COM118  CALL    @SCANF          ;get a character (0 to F)
        CMP     %>0D,NUMBER     ;test for D (enter)
        JNE     COM118          ;if D then exit
        MPY     %10,MISC1
        CLR     A
        ADD     MISC2,BREG
        MOV     BREG,YEARL
*
        CALL    @DSCLER
*
        MOVD    %SETTIM,LDTBLL
        CALL    @SHSTRG
*
        DINT
        MOV     %?11000100,A
        CALL    @DSCONT
*
        CLR     B               ;clear input counter
COM119  CALL    @SCANF          ;get a character (0 to F)
        CMP     %>0C,NUMBER
        JNE     COM120
        CMP     %>00,B
        JEQ     COM120
        CALL    @BKSPAC
        DEC     B
        JMP     COM119
```

```
COM120    CMP     %>0A,NUMBER             ;test for F (enter)
          JHS     COM119                  ;if F then exit
          MOV     NUMBER,A                ;copy character to A reg for offsetting
          STA     @CKBTHH(B)              ;save char at temp+offset
          OR      %>30,A                  ;make character ASCII for display
          CALL    @DSCHAR                 ;send charater to display
          INC     R1                      ;inc input counter
          CMP     %>02,R1                 ;test for max address of three bytes 255
          JHS     COM121                  ;if higher or same then exit
          JMP     COM119                  ;if lower and no 'enter' then get next char
COM121    MOV     %':',A
          CALL    @DSCHAR
*
          CLR     R1                      ;clear input counter
COM122    CALL    @SCANF                  ;get a character (0 to F)
          CMP     %>0C,NUMBER
          JNE     COM123
          CMP     %>00,B
          JEQ     COM123
          CALL    @BKSPAC
          DEC     B
          JMP     COM122
COM123    CMP     %>0A,NUMBER             ;test for F (enter)
          JHS     COM122                  ;if F then exit
          MOV     NUMBER,A                ;copy character to A reg for offsetting
          STA     @CKBTMH(B)              ;save char at temp+offset
          OR      %>30,A                  ;make character ASCII for display
          CALL    @DSCHAR                 ;send charater to display
          INC     R1                      ;inc input counter
          CMP     %>02,R1                 ;test for max address of three bytes 255
          JHS     COM124                  ;if higher or same then exit
          JMP     COM122                  ;if lower and no 'enter' then get next char
COM124    CALL    @SCANF                  ;get a character (0 to F)
          CMP     %>0D,NUMBER             ;test for D (enter)
          JNE     COM124                  ;if D then exit
          CLR     CKBTSH                  ;clear seconds
          CLR     CKBTSL                  ;clear seconds
          EINT
          BR      @TAKEOF
*
** backspace key
*
COM12     CALL    @DSCLER
*
          MOVD    %TESTNG,LDTBLL
          CALL    @SHSTRG
*
          ANDP    %?10111111,APORT
          LDA     @PORT3
          LDA     @PORT3
          CALL    @WAITX
          CALL    @SPORT1
          INV     A
          CALL    @SPORT2
          CLR     A
          CALL    @SPORT1
          CALL    @SPORT2
*
          MOV     %>01,A
          CALL    @SPORT1
          CLR     A
          CALL    @SPORT1
          MOV     %>02,A
          CALL    @SPORT1
```

```
        CLR     A
        CALL    @SPORT1
        MOV     %>04,A
        CALL    @SPORT1
        CLR     A
        CALL    @SPORT1
        MOV     %>08,A
        CALL    @SPORT1
        CLR     A
        CALL    @SPORT1
        MOV     %>10,A
        CALL    @SPORT1
        CLR     A
        CALL    @SPORT1
        MOV     %>20,A
        CALL    @SPORT1
        CLR     A
        CALL    @SPORT1
        MOV     %>40,A
        CALL    @SPORT1
        CLR     A
        CALL    @SPORT1
        MOV     %>80,A
        CALL    @SPORT1
        CLR     A
        CALL    @SPORT1
*
        MOV     %>01,A
        CALL    @SPORT2
        CLR     A
        CALL    @SPORT2
        MOV     %>02,A
        CALL    @SPORT2
        CLR     A
        CALL    @SPORT2
        MOV     %>04,A
        CALL    @SPORT2
        CLR     A
        CALL    @SPORT2
        MOV     %>08,A
        CALL    @SPORT2
        CLR     A
        CALL    @SPORT2
        MOV     %>10,A
        CALL    @SPORT2
        CLR     A
        CALL    @SPORT2
        MOV     %>20,A
        CALL    @SPORT2
        CLR     A
        CALL    @SPORT2
        MOV     %>40,A
        CALL    @SPORT2
        CLR     A
        CALL    @SPORT2
        MOV     %>80,A
        CALL    @SPORT2
*
        CLR     A
        CALL    @SPORT1
        CALL    @SPORT2
        CLR     OUT1ST
        CLR     OUT2ST
        BR      @TAKEOF
```

```
*
SPORT1  STA     @PORT1
        CALL    @WAITS
        RETS
*
SPORT2  STA     @PORT2
        CALL    @WAITS
        RETS
*
BKSPAC  MOV     %?00000100,A
        CALL    @DSCONT
        MOV     %' ',A
        CALL    @DSCHAR
        MOV     %?00000110,A
        CALL    @DSCONT
        RETS
*
** enter key & plc mode
*
COM13   CALL    @DSCLER
*
        MOVD    %OPRTYP,LDTBLL
        CALL    @SHSTRG
*
        MOV     %?11000000,A
        CALL    @DSCONT
*
        MOVD    %OPRTY1,LDTBLL
        CALL    @SHSTRG
*
COMTS6  CALL    @SCANF                  ;get a character (0 to F)
        CMP     %>03,NUMBER
        JHS     COMTS6
        MOV     NUMBER,A
        MOV     NUMBER,PBMODE
        OR      %>30,A
        CALL    @DSCHAR
        CALL    @WAITY
        CALL    @WAITY
        CALL    @WAITY
*
        BR      @TAKEOF
*
* program mode key
*
COM14   CALL    @DSCLER
*
        MOVD    %DEVTYP,LDTBLL
        CALL    @SHSTRG
*
COMTS3  CALL    @SCANF                  ;get a character (0 to F)
        CMP     %>01,NUMBER
        JEQ     COMTS4
        CMP     %>02,NUMBER
        JEQ     COMTS5
        JMP     COMTS3
COMTS4  CMP     %00,PBMODE
        JEQ     CMOXT
        CMP     %02,PBMODE
        JEQ     CMOXT
        JMP     OUTNOW
CMOXT   CALL    @PROGMP
        JMP     OUTNOW
COMTS5  CMP     %01,PBMODE
```

```
            JL      OUTNOW
            CALL    @PROGMB
OUTNOW      BR      @TAKEOF
*
PROGMP      CALL    @DSCLER
*
            MOV     %'(',A
            CALL    @DSCHAR
            MOV     %'P',A
            CALL    @DSCHAR
            MOV     %')',A
            CALL    @DSCHAR
            MOV     %' ',A
            CALL    @DSCHAR
            CLR     B
            MOV     %'O',A              ;move the 'O' into display
            CALL    @DSCHAR
*
            MOV     %?00000111,A        ;return cursor to home position
            CALL    @DSCONT
            MOV     %?00011000,A        ;return cursor to home position
            CALL    @DSCONT
*
COM141      CALL    @SCANF              ;get a character (0 to F)
            MOV     NUMBER,A            ;copy character to A reg for offsetting
            CMP     %>00,PBMODE
            JEQ     COMTS2
            CMP     %>08,NUMBER
            JL      COM14E
            JHS     COM141
COMTS2      CMP     %10,A
            JL      COM14E
            MOV     A,MISC1
            SUB     %10,MISC1
            MOV     %'1',A
            CALL    @DSCHAR
            MOV     MISC1,A
COM14E      OR      %>30,A              ;make character ASCII for display
            CALL    @DSCHAR             ;send charater to display
*
COM142      MOV     NUMBER,B            ;setup base address of output port
            MPY     %10,B
            CLR     A
            MOV     B,POINTR
            STA     @ANDIN0(B)
            INC     B
            STA     @ANDIN0(B)
            INC     B
            STA     @ANDIN0(B)
            INC     B
            STA     @ANDIN0(B)
            INC     B
            STA     @ANDIN0(B)
            INC     B
            STA     @ANDIN0(B)
            INC     B
            STA     @ANDIN0(B)
            INC     B
            STA     @ANDIN0(B)
            INC     B
            STA     @ANDIN0(B)
            INC     B
            STA     @ANDIN0(B)
            INC     B
            STA     @ANDIN0(B)
*
```

```
          MOV     %'=',A
          CALL    @DSCHAR                 ;'O'+port number+'('N or F+')=' in display
*
COM14X    CALL    @SCANF                  ;get a character (0 to F)
          CMP     %>0D,NUMBER
          JEQ     COM14Z
          CMP     %>08,NUMBER
          JEQ     COM14W
          CMP     %>09,NUMBER
          JEQ     COM14V
          JMP     COM14X
COM14W    MOV     %'+',A
          CALL    @DSCHAR
          JMP     COM144
COM14V    MOV     %'/',A
          CALL    @DSCHAR
*
COM14Y    MOV     POINTR,B
          CALL    @TYPEA
          JMP     COM14X
*
COM14Z    MOV     %?00000110,A            ;return cursor to home position
          CALL    @DSCONT
          MOV     %?00010100,A            ;return cursor to home position
          CALL    @DSCONT
          RETS
*
COM144    MOV     POINTR,B
          CALL    @TYPEB
          JMP     COM14X
*
PROGMB    CALL    @DSCLER
*
          CMP     %>01,PBMODE
          JHS     COMTS7
          BR      @TAKEOF
*
COMTS7    MOVD    %POSIT,LDTBLL
          CALL    @SHSTRG
*
          MOV     %?11000000,A
          CALL    @DSCONT
*
          MOVD    %POSIT1,LDTBLL
          CALL    @SHSTRG
*
COMTS8    CALL    @SCANF                  ;get a character (0 to F)
          CMP     %>04,NUMBER
          JHS     COMTS8
          MOV     NUMBER,A
          OR      %>30,A
          CALL    @DSCHAR
          MPY     %10,NUMBER
          MOV     B,POINTR
          CALL    @WAITS
          CLR     A
          STA     @SDAY1(B)
          INC     B
          STA     @SDAY1(B)
          INC     B
          STA     @SDAY1(B)
          INC     B
          STA     @SDAY1(B)
          INC     B
```

```
            STA     @SDAY1(B)
            INC     B
            STA     @SDAY1(B)
            INC     B
            STA     @SDAY1(B)
            INC     B
            STA     @SDAY1(B)
            INC     B
            STA     @SDAY1(B)
            INC     B
            STA     @SDAY1(B)
*
            BR      @BACK
COMTS9      BR      @TAKEOF
*
** enter/escape key
*
COM15       CALL    @DSCLER
*
            MOVD    %MANUAL,LDTBLL
            CALL    @SHSTRG
*
            CALL    @SCANF
            MOV     NUMBER,A
            OR      %>30,A
            CALL    @DSCHAR
            MOV     NUMBER,B
            LDA     @ALTBLN(B)
            CMP     %8,NUMBER
            JHS     COM151
            MOV     A,OUT1ST
            CALL    @WAITY
            CALL    @WAITY
            CLR     OUT2ST
            JMP     COM152
COM151      MOV     A,OUT2ST
            CALL    @WAITY
            CALL    @WAITY
            CLR     OUT1ST
COM152      BR      @TAKEOF
*
BACKOF      BR      @COMTS9
*
BACK        CALL    @DSCLER                 ;get start date
            MOVD    %SDAY,LDTBLL
            CALL    @SHSTRG
BACK0       CALL    @SCANF
            CMP     %>0D,NUMBER
            JEQ     BACKOF
            CMP     %>00,NUMBER
            JEQ     BACK0
            CMP     %>04,NUMBER
            JHS     BACK0
            MOV     NUMBER,MISC1
            MOV     NUMBER,A
            OR      %>30,A
            CALL    @DSCHAR
BACK1       CALL    @SCANF
            CMP     %>00,NUMBER
            JEQ     BACK1
            CMP     %>0A,NUMBER
            JHS     BACK1
            MOV     NUMBER,MISC2
            DEC     MISC2
```

```
        MOV     NUMBER,A
        OR      %>30,A
        CALL    @DSCHAR
        MPY     %10,MISC1
        ADD     MISC2,B
        MOV     B,A
        MOV     POINTR,B
        ADD     %0,B
        STA     @SDAY1(B)
        CALL    @WAITY
*
        MOV     %?11000000,A            ;get start month
        CALL    @DSCONT
        MOVD    %SMNTH,LDTBLL
        CALL    @SHSTRG
BACK2   CALL    @SCANF
        CMP     %>02,NUMBER
        JHS     BACK2
        CMP     %>00,NUMBER
        JEQ     BACK2
        MOV     NUMBER,MISC1
        MOV     NUMBER,A
        OR      %>30,A
        CALL    @DSCHAR
BACK3   CALL    @SCANF
        CMP     %>00,NUMBER
        JEQ     BACK3
        CMP     %>0A,NUMBER
        JHS     BACK3
        MOV     NUMBER,MISC2
        DEC     MISC2
        MOV     NUMBER,A
        OR      %>30,A
        CALL    @DSCHAR
        MPY     %10,MISC1
        ADD     MISC2,B
        MOV     B,A
        MOV     POINTR,B
        ADD     %1,B
        STA     @SDAY1(B)
        CALL    @WAITY
*
*
        CALL    @DSCLER                 ;get end date
        MOVD    %EDAY,LDTBLL
        CALL    @SHSTRG
BACK4   CALL    @SCANF
        CMP     %>00,NUMBER
        JEQ     BACK4
        CMP     %>04,NUMBER
        JHS     BACK4
        MOV     NUMBER,MISC1
        MOV     NUMBER,A
        OR      %>30,A
        CALL    @DSCHAR
BACK5   CALL    @SCANF
        CMP     %>00,NUMBER
        JEQ     BACK5
        CMP     %>0A,NUMBER
        JHS     BACK5
        MOV     NUMBER,MISC2
        DEC     MISC2
        MOV     NUMBER,A
        OR      %>30,A
```

```
        CALL    @DSCHAR
        MPY     %10,MISC1
        ADD     MISC2,B
        MOV     B,A
        MOV     POINTR,B
        ADD     %2,B
        STA     @SDAY1(B)
        CALL    @WAITY
*
*
        MOV     %?11000000,A            ;get end month
        CALL    @DSCONT
        MOVD    %EMNTH,LDTBLL
        CALL    @SHSTRG
BACK6   CALL    @SCANF
        CMP     %>00,NUMBER
        JEQ     BACK6
        CMP     %>02,NUMBER
        JHS     BACK6
        MOV     NUMBER,MISC1
        MOV     NUMBER,A
        OR      %>30,A
        CALL    @DSCHAR
BACK7   CALL    @SCANF
        CMP     %>00,NUMBER
        JEQ     BACK7
        CMP     %>0A,NUMBER
        JHS     BACK7
        MOV     NUMBER,MISC2
        DEC     MISC2
        MOV     NUMBER,A
        OR      %>30,A
        CALL    @DSCHAR
        MPY     %10,MISC1
        ADD     MISC2,B
        MOV     B,A
        MOV     POINTR,B
        ADD     %3,B
        STA     @SDAY1(B)
        CALL    @WAITY
*
        CALL    @DSCLER                 ;get duration between cycles
        MOVD    %SPACE,LDTBLL
        CALL    @SHSTRG
        MOV     %?11000000,A
        CALL    @DSCONT
        MOVD    %CYCLES,LDTBLL
        CALL    @SHSTRG
*
        CLR     B                       ;clear input counter
BACK8   CALL    @SCANF                  ;get a character (0 to F)
        CMP     %>0C,NUMBER
        JNE     BACK9
        CMP     %>00,B
        JEQ     BACK9
        CALL    @BKSPAC
        DEC     B
        JMP     BACK8
BACK9   CMP     %>0A,NUMBER             ;test for F (enter)
        JHS     BACK8                   ;if F then exit
        MOV     NUMBER,A                ;copy character to A reg for offsetting
        STA     @TEMP(B)                ;save char at temp+offset
        OR      %>30,A                  ;make character ASCII for display
        CALL    @DSCHAR                 ;send charater to display
```

```
        INC     B               ;inc input counter
        CMP     %>02,R1         ;test for max address of three bytes 255
        JHS     BACK10          ;if higher or same then exit
        JMP     BACK8           ;if lower and no 'enter' then get next char
BACK10  MOV     %':',A
        CALL    @DSCHAR
*
        CLR     B               ;clear input counter
BACK11  CALL    @SCANF          ;get a character (0 to F)
        CMP     %>0C,NUMBER
        JNE     BACK12
        CMP     %>00,B
        JEQ     BACK12
        CALL    @BKSPAC
        DEC     B
        JMP     BACK11
BACK12  CMP     %>0A,NUMBER     ;test for F (enter)
        JHS     BACK11          ;if F then exit
        MOV     NUMBER,A        ;copy character to A reg for offsetting
        STA     @TEMP2(B)       ;save char at temp+offset
        OR      %>30,A          ;make character ASCII for display
        CALL    @DSCHAR         ;send character to display
        INC     R1              ;inc input counter
        CMP     %>02,R1         ;test for max address of three bytes 255
        JHS     BACK14          ;if higher or same then exit
        JMP     BACK11          ;if lower and no 'enter' then get next char
BACK14  CALL    @SCANF          ;get a character (0 to F)
        CMP     %>0D,NUMBER     ;test for D (enter)
        JNE     BACK14          ;if D then exit
        CALL    @ALCNVT
        MOV     POINTR,B
        ADD     %4,B
        MOV     MISC1,A
        STA     @SDAY1(B)
        MOV     MISC2,A
        INC     B
        STA     @SDAY1(B)
        MOV     POINTR,B
        ADD     %8,B
        MOV     MISC1,A
        STA     @SDAY1(B)
        MOV     MISC2,A
        INC     B
        STA     @SDAY1(B)
        MOV     %1,BKFLAG
*
BACK19  CALL    @DSCLER         ;get duration on in seconds
        MOVD    %DURON,LDTBLL
        CALL    @SHSTRG
        CLR     B               ;clear input counter
        CLR     TEMP
        CLR     TEMP1
        CLR     TEMP2
        CLR     TEMP3
BACK15  CALL    @SCANF          ;get a character (0 to F)
        CMP     %>0C,NUMBER
        JNE     BACK16
        CMP     %>00,B
        JEQ     BACK16
        CALL    @BKSPAC
        DEC     B
        JMP     BACK15
BACK16  CMP     %>0A,NUMBER     ;test for F (enter)
        JHS     BACK15          ;if F then exit
```

```
            MOV     NUMBER,A                ;copy character to A reg for offsetting
            STA     @TEMP(B)                ;save char at temp+offset
            OR      %>30,A                  ;make character ASCII for display
            CALL    @DSCHAR                 ;send charater to display
            INC     B                       ;inc input counter
            CMP     %3,B                    ;test for max address of three bytes 255
            JHS     BACK17                  ;if higher or same then exit
            JMP     BACK15                  ;if lower and no 'enter' then get next char
BACK17      MPY     %100,TEMP
            MOV     B,TEMP3
            MPY     %10,TEMP1
            ADD     B,TEMP3
            ADD     TEMP2,TEMP3
            JC      BACK19
            MOV     POINTR,B
            ADD     %6,B
            MOV     TEMP3,A
            STA     @SDAY1(B)
*
*
BACK18      MOV     %?11000000,A            ;get duration off in seconds
            CALL    @DSCONT
            MOVD    %DUROF,LDTBLL
            CALL    @SHSTRG
            CLR     B                       ;clear input counter
            CLR     TEMP
            CLR     TEMP1
            CLR     TEMP2
            CLR     TEMP3
BACK20      CALL    @SCANF                  ;get a character (0 to F)
            CMP     %>0C,NUMBER
            JNE     BACK21
            CMP     %>00,B
            JEQ     BACK21
            CALL    @BKSPAC
            DEC     B
            JMP     BACK20
BACK21      CMP     %>0A,NUMBER             ;test for F (enter)
            JHS     BACK20                  ;if F then exit
            MOV     NUMBER,A                ;copy character to A reg for offsetting
            STA     @TEMP(B)                ;save char at temp+offset
            OR      %>30,A                  ;make character ASCII for display
            CALL    @DSCHAR                 ;send charater to display
            INC     B                       ;inc input counter
            CMP     %3,B                    ;test for max address of three bytes 255
            JHS     BACK22                  ;if higher or same then exit
            JMP     BACK20                  ;if lower and no 'enter' then get next char
BACK22      MPY     %100,TEMP
            MOV     B,TEMP3
            MPY     %10,TEMP1
            ADD     B,TEMP3
            ADD     TEMP2,TEMP3
            JC      BACK18
            MOV     POINTR,B
            ADD     %7,B
            MOV     TEMP3,A
            STA     @SDAY1(B)
            CALL    @WAITY
*
            BR      @COMTS9

*
*******************************************************************************
** type of bit set/reset when OR'ed
```

```
*
TYPEA   NOP
TYPA1   CALL    @SCANF                  ;get a character (0 to F)
        CMP     %5,NUMBER
        JHS     TYPA1
        CMP     %0,NUMBER
        JEQ     TYPA1
        MPY     %3,NUMBER
        CLR     A
        BR      @TYPTBL(B)
TYPA2   RETS
*
TYPTBL  BR      @TYPAI                  ;input for comparison (out of range)
        BR      @TYPAI                  ;input for comparison
        BR      @TYPAO                  ;output for comparison
        BR      @TYPAT                  ;time for comparison
        BR      @TYPAD                  ;duration for comparison
        BR      @TYPAI                  ;input for comparison (out of range)
*
** use an OR'd input bit
*
TYPAI   MOV     %'I',A
        CALL    @DSCHAR
TYPAI1  CALL    @SCANF                  ;get a character (0 to F)
        CMP     %8,NUMBER
        JHS     TYPAI1
        MOV     NUMBER,A                ;copy character to A reg for offsetting
        OR      %>30,A                  ;make character ASCII for display
        CALL    @DSCHAR                 ;send charater to display
*
        MOV     NUMBER,B
        MOV     NUMBER,LOCAL
        LDA     @ALTBLN(B)
        PUSH    A
        MOV     POINTR,B
        ADD     %2,B
        LDA     @ANDIN0(B)
        MOV     A,B
        POP     A
        OR      B,A
        MOV     POINTR,B
        ADD     %2,B
        STA     @ANDIN0(B)
TYPAI2  BR      @ONFINP
*
** use an OR'd output bit
*
TYPAO   MOV     %'O',A
        CALL    @DSCHAR
TYPAO1  CALL    @SCANF                  ;get a character (0 to F)
        CMP     %8,NUMBER
        JHS     TYPAO1
        MOV     NUMBER,A                ;copy character to A reg for offsetting
        OR      %>30,A                  ;make character ASCII for display
        CALL    @DSCHAR                 ;send charater to display
*
        MOV     NUMBER,B
        MOV     NUMBER,LOCAL
        LDA     @ALTBLN(B)
        PUSH    A
        MOV     POINTR,B
        ADD     %3,B
        LDA     @ANDIN0(B)
        MOV     A,B
```

```
            POP     A
            OR      B,A
            MOV     POINTR,B
            ADD     %3,B
            STA     @ANDIN0(B)
*
TYPA02      BR      @ONFOUT
*
** use the time as a start point
*
TYPAT       MOV     %'T',A
            CALL    @DSCHAR
            MOV     %'(',A                  ;'O'+port number+'(' in display
            CALL    @DSCHAR
            CLR     B
TYPAT1      CALL    @SCANF                  ;get a character (0 to F)
            MOV     NUMBER,A                ;copy character to A reg for offsetting
            CMP     %>0C,NUMBER
            JNE     TYPAT5
            CMP     %>00,B
            JEQ     TYPAT5
            CALL    @BKSPAC
            DEC     B
            JMP     TYPAT1
TYPAT5      CMP     %>0A,NUMBER
            JHS     TYPAT1
            STA     @TEMP(B)                ;save char at temp+offset
            OR      %>30,A                  ;make character ASCII for display
            CALL    @DSCHAR                 ;send character to display
            INC     B                       ;inc input counter
            CMP     %>02,B                  ;test for max address of three bytes 255
            JHS     TYPAT2                  ;if higher or same then exit
            JMP     TYPAT1                  ; 'O' plus port in display
TYPAT2      MOV     %':',A
            CALL    @DSCHAR
            CLR     B
TYPAT3      CALL    @SCANF                  ;get a character (0 to F)
            MOV     NUMBER,A                ;copy character to A reg for offsetting
            CMP     %>0C,NUMBER
            JNE     TYPAT6
            CMP     %>00,B
            JEQ     TYPAT6
            CALL    @BKSPAC
            DEC     B
            JMP     TYPAT3
TYPAT6      CMP     %>0A,NUMBER
            JHS     TYPAT3
            STA     @TEMP2(B)               ;save char at temp+offset
            OR      %>30,A                  ;make character ASCII for display
            CALL    @DSCHAR                 ;send charater to display
            INC     B                       ;inc input counter
            CMP     %>02,B                  ;test for max address of three bytes 255
            JHS     TYPAT4                  ;if higher or same then exit
            JMP     TYPAT3                  ; 'O' plus port in display
TYPAT4      MOV     %')',A
            CALL    @DSCHAR
            DINT
            CALL    @ALCNVT
            MOV     POINTR,B
            ADD     %6,B
            AND     %>7F,MISC1
            MOV     MISC1,A
            STA     @ANDIN0(B)
            INC     B
```

```
            MOV     MISC2,A
            STA     @ANDIN0(B)
*
            MOV     POINTR,B
            ADD     %8,B
            MOV     %>00,A
            STA     @ANDIN0(B)
            INC     B
            MOV     %>01,A
            STA     @ANDIN0(B)
            EINT
            RETS
*
** set a duration on
*
TYPAD       MOV     %'D',A
            CALL    @DSCHAR
            MOV     %'(',A                  ;'O'+port number+'(' in display
            CALL    @DSCHAR
            CLR     B
TYPAD1      CALL    @SCANF                  ;get a character (0 to F)
            MOV     NUMBER,A                ;copy character to A reg for offsetting
            CMP     %>0C,NUMBER
            JNE     TYPAD5
            CMP     %>00,B
            JEQ     TYPAD5
            CALL    @BKSPAC
            DEC     B
            JMP     TYPAD1
TYPAD5      CMP     %>0A,NUMBER
            JHS     TYPAD1
            STA     @TEMP(B)                ;save char at temp+offset
            OR      %>30,A                  ;make character ASCII for display
            CALL    @DSCHAR                 ;send character to display
            INC     B                       ;inc input counter
            CMP     %>02,B                  ;test for max address of three bytes 255
            JHS     TYPAD2                  ;if higher or same then exit
            JMP     TYPAD1                  ; 'O' plus port in display
TYPAD2      MOV     %':',A
            CALL    @DSCHAR
            CLR     B
TYPAD3      CALL    @SCANF                  ;get a character (0 to F)
            MOV     NUMBER,A                ;copy character to A reg for offsetting
            CMP     %>0C,NUMBER
            JNE     TYPAD6
            CMP     %>00,B
            JEQ     TYPAD6
            CALL    @BKSPAC
            DEC     B
            JMP     TYPAD3
TYPAD6      CMP     %>0A,NUMBER
            JHS     TYPAD3
            STA     @TEMP2(B)               ;save char at temp+offset
            OR      %>30,A                  ;make character ASCII for display
            CALL    @DSCHAR                 ;send charater to display
            INC     B                       ;inc input counter
            CMP     %>02,B                  ;test for max address of three bytes 255
            JHS     TYPAD4                  ;if higher or same then exit
            JMP     TYPAD3                  ; 'O' plus port in display
TYPAD4      MOV     %')',A
            CALL    @DSCHAR
            DINT
            CALL    @ALCNVT
            MOV     POINTR,B
```

```
            ADD     %8,B
            MOV     MISC1,A
            STA     @ANDIN0(B)
            INC     B
            MOV     MISC2,A
            STA     @ANDIN0(B)
            EINT
            RETS
*
*
*******************************************************************************
** type of bit set/reset when AND'ed
*
TYPEB       NOP
TYPB1       CALL    @SCANF                  ;get a character (0 to F)
            CMP     %5,NUMBER
            JHS     TYPB1
            CMP     %0,NUMBER
            JEQ     TYPB1
            MPY     %3,NUMBER
            CLR     A
            BR      @TYPTAL(B)
TYPB2       RETS
*
TYPTAL      BR      @TYPBI                  ;input for comparison (out of range)
            BR      @TYPBI                  ;input for comparison
            BR      @TYPBO                  ;output for comparison
            BR      @TYPBT                  ;time for comparison
            BR      @TYPBD                  ;duration for comparison
            BR      @TYPBI                  ;input for comparison (out of range)
*
** use an input bit
*
TYPBI       MOV     %'I',A
            CALL    @DSCHAR
TYPBI1      CALL    @SCANF                  ;get a character (0 to F)
            CMP     %8,NUMBER
            JHS     TYPBI1
            MOV     NUMBER,A                ;copy character to A reg for offsetting
            OR      %>30,A                  ;make character ASCII for display
            CALL    @DSCHAR                 ;send charater to display
*
            MOV     NUMBER,B
            MOV     NUMBER,LOCAL
            LDA     @ALTBLN(B)
            PUSH    A
            MOV     POINTR,B
            ADD     %0,B
            LDA     @ANDIN0(B)
            MOV     A,B
            POP     A
            OR      B,A
            MOV     POINTR,B
            ADD     %0,B
            STA     @ANDIN0(B)
TYPBI2      BR      @ONFINP
*
** use an output bit
*
TYPBO       MOV     %'O',A
            CALL    @DSCHAR
TYPBO1      CALL    @SCANF                  ;get a character (0 to F)
            MOV     NUMBER,A                ;copy character to A reg for offsetting
            CMP     %8,A
```

```
            JHS     TYPBO1
            OR      %>30,A                  ;make character ASCII for display
            CALL    @DSCHAR                 ;send charater to display
*
            MOV     NUMBER,B
            MOV     NUMBER,LOCAL
            LDA     @ALTBLN(B)
            PUSH    A
            MOV     POINTR,B
            ADD     %1,B
            LDA     @ANDIN0(B)
            MOV     A,B
            POP     A
            OR      B,A
            MOV     POINTR,B
            ADD     %1,B
            STA     @ANDIN0(B)
*
TYPB02      BR      @ONFOUT
*
** use the time as a start point
*
TYPBT       MOV     %'T',A
            CALL    @DSCHAR
            MOV     %'(',A                  ;'O'+port number+'(' in display
            CALL    @DSCHAR
            CLR     B
TYPBT1      CALL    @SCANF                  ;get a character (0 to F)
            MOV     NUMBER,A                ;copy character to A reg for offsetting
            CMP     %>0C,NUMBER
            JNE     TYPBT5
            CMP     %>00,B
            JEQ     TYPBT5
            CALL    @BKSPAC
            DEC     B
            JMP     TYPBT1
TYPBT5      CMP     %>0A,NUMBER
            JHS     TYPBT1
            STA     @TEMP(B)                ;save char at temp+offset
            OR      %>30,A                  ;make character ASCII for display
            CALL    @DSCHAR                 ;send character to display
            INC     B                       ;inc input counter
            CMP     %>02,B                  ;test for max address of three bytes 255
            JHS     TYPBT2                  ;if higher or same then exit
            JMP     TYPBT1                  ; 'O' plus port in display
TYPBT2      MOV     %':',A
            CALL    @DSCHAR
            CLR     B
TYPBT3      CALL    @SCANF                  ;get a character (0 to F)
            MOV     NUMBER,A                ;copy character to A reg for offsetting
            CMP     %>0C,NUMBER
            JNE     TYPBT6
            CMP     %>00,B
            JEQ     TYPBT6
            CALL    @BKSPAC
            DEC     B
            JMP     TYPBT3
TYPBT6      CMP     %>0A,NUMBER
            JHS     TYPBT3
            STA     @TEMP2(B)               ;save char at temp+offset
            OR      %>30,A                  ;make character ASCII for display
            CALL    @DSCHAR                 ;send charater to display
            INC     B                       ;inc input counter
            CMP     %>02,B                  ;test for max address of three bytes 255
```

```
            JHS     TYPBT4                  ;if higher or same then exit
            JMP     TYPBT3                  ; 'O' plus port in display
TYPBT4      MOV     %')',A
            CALL    @DSCHAR
            DINT
            CALL    @ALCNVT
            MOV     POINTR,B
            ADD     %6,B
            MOV     MISC1,A
            OR      %?10000000,A
            STA     @ANDIN0(B)
            INC     B
            MOV     MISC2,A
            STA     @ANDIN0(B)
*
            MOV     POINTR,B
            ADD     %8,B
            MOV     %>00,A
            STA     @ANDIN0(B)
            INC     B
            MOV     %01,A
            STA     @ANDIN0(B)
            EINT
            RETS
*
** set a duration on
*
TYPBD       MOV     %'D',A
            CALL    @DSCHAR
            MOV     %'(',A                  ;'O'+port number+'(' in display
            CALL    @DSCHAR
            CLR     B
TYPBD1      CALL    @SCANF                  ;get a character (0 to F)
            MOV     NUMBER,A                ;copy character to A reg for offsetting
            CMP     %>0C,NUMBER
            JNE     TYPBD5
            CMP     %>00,B
            JEQ     TYPBD5
            CALL    @BKSPAC
            DEC     B
            JMP     TYPBD1
TYPBD5      CMP     %>0A,NUMBER
            JHS     TYPBD1
            STA     @TEMP(B)                ;save char at temp+offset
            OR      %>30,A                  ;make character ASCII for display
            CALL    @DSCHAR                 ;send character to display
            INC     B                       ;inc input counter
            CMP     %>02,B                  ;test for max address of three bytes 255
            JHS     TYPBD2                  ;if higher or same then exit
            JMP     TYPBD1                  ; 'O' plus port in display
TYPBD2      MOV     %':',A
            CALL    @DSCHAR
            CLR     B
TYPBD3      CALL    @SCANF                  ;get a character (0 to F)
            MOV     NUMBER,A                ;copy character to A reg for offsetting
            CMP     %>0C,NUMBER
            JNE     TYPBD6
            CMP     %>00,B
            JEQ     TYPBD6
            CALL    @BKSPAC
            DEC     B
            JMP     TYPBD3
TYPBD6      CMP     %>0A,NUMBER
            JHS     TYPBD3
```

```
            STA     @TEMP2(B)               ;save char at temp+offset
            OR      %>30,A                  ;make character ASCII for display
            CALL    @DSCHAR                 ;send charater to display
            INC     B                       ;inc input counter
            CMP     %>02,B                  ;test for max address of three bytes 255
            JHS     TYPBD4                  ;if higher or same then exit
            JMP     TYPBD3                  ; 'O' plus port in display
TYPBD4      MOV     %')',A
            CALL    @DSCHAR
            DINT
            CALL    @ALCNVT
            MOV     POINTR,B
            ADD     %8,B
            MOV     MISC1,A
            STA     @ANDIN0(B)
            INC     B
            MOV     MISC2,A
            STA     @ANDIN0(B)
            EINT
            RETS
*
*
*******************************************************************************
** set bit on or off
*
ONFINP      MOV     %'(',A                  ;'O'+port number+'(' in display
            CALL    @DSCHAR
ONFIN4      CALL    @SCANF                  ;get a character (0 to F)
            CMP     %>06,NUMBER
            JEQ     ONFIN1
            CMP     %>07,NUMBER
            JEQ     ONFIN3
            JMP     ONFIN4
ONFIN3      MOV     %'F',A
            CALL    @DSCHAR
            MOV     LOCAL,B
            LDA     @ALTBLN(B)
            INV     A
            PUSH    A
            MOV     POINTR,B
            ADD     %4,B
            LDA     @ANDIN0(B)
            MOV     A,B
            POP     A
            AND     B,A
            MOV     POINTR,B
            ADD     %4,B
            STA     @ANDIN0(B)
            JMP     ONFIN2
*
ONFIN1      MOV     %'N',A
            CALL    @DSCHAR
            MOV     LOCAL,B
            LDA     @ALTBLN(B)
            PUSH    A
            MOV     POINTR,B
            ADD     %4,B
            LDA     @ANDIN0(B)
            MOV     A,B
            POP     A
            OR      B,A
            MOV     POINTR,B
            ADD     %4,B
            STA     @ANDIN0(B)
```

```
ONFIN2    MOV     %')',A
          CALL    @DSCHAR
          RETS
*
ONFOUT    MOV     %'(',A                  ;'O'+port number+'(' in display
          CALL    @DSCHAR
ONFOU4    CALL    @SCANF                  ;get a character (0 to F)
          CMP     %>06,NUMBER
          JEQ     ONFOU1
          CMP     %>07,NUMBER
          JEQ     ONFOU3
          JMP     ONFOU4
ONFOU3    MOV     %'F',A
          CALL    @DSCHAR
*
          MOV     LOCAL,B
          LDA     @ALTBLN(B)
          INV     A
          PUSH    A
          MOV     POINTR,B
          ADD     %5,B
          LDA     @ANDIN0(B)
          MOV     A,B
          POP     A
          AND     B,A
          MOV     POINTR,B
          ADD     %5,B
          STA     @ANDIN0(B)
          JMP     ONFOU2
*
ONFOU1    MOV     %'N',A
          CALL    @DSCHAR
*
          MOV     LOCAL,B
          LDA     @ALTBLN(B)
          PUSH    A
          MOV     POINTR,B
          ADD     %5,B
          LDA     @ANDIN0(B)
          MOV     A,B
          POP     A
          OR      B,A
          MOV     POINTR,B
          ADD     %5,B
          STA     @ANDIN0(B)
*
ONFOU2    MOV     %')',A
          CALL    @DSCHAR
          RETS
*
** get a key
*
SCANF     CALL    @ISKEY                  ;get on/off delay
SCANF1    MOV     %>F0,NUMBER
          CALL    @SCANB                  ;get key pressed
          CMP     %>F0,NUMBER
          JEQ     SCANF1                  ;loop if not valid
          RETS                            ;return to sender
*
** wait for key released
*
ISKEY     ORP     %?00010000,APORT        ;set scan high bit 0
          ORP     %?00000111,BPORT        ;set scan high bit 1 to 3
ISKEY1    CALL    @WAIT                   ;wait a bit
```

```
            CALL    @WAIT                       ;wait a bit
            BTJOP   %?00001111,APORT,ISKEY1     ;loop on any column one
            ANDP    %?11101111,APORT            ;set scan low bit 0
            ANDP    %?11111000,BPORT            ;set scan low bit 1 to 3
            CALL    @WAITX                      ;wait a bit
            CALL    @WAITX                      ;wait a bit
            RETS                                ;return to sender
*
*******************************************************************************
** update the outputs
*
SCANA       MOV     OUT1ST,A
            STA     @PORT1
            CALL    @WAIT
            MOV     OUT2ST,A
            STA     @PORT2
            CALL    @WAIT
            RETS
*
*
*******************************************************************************
** get the inputs
*
SCAND       LDA     @PORT3
            MOV     A,INP1ST
            CALL    @WAIT
            RETS
*
*
*******************************************************************************
** control routine to test status for PLC routines
*
ALCNTL      CLR     B
            PUSH    ADDRES
ALCT00      MOV     B,ALCONT
            MPY     %10,B
            CLR     A
            MOV     B,INDEX
            CALL    @ANDON
            MOV     ALCONT,B
            INC     B
            CMP     %2,PBMODE
            JEQ     ALCT02
            CMP     %16,B                       ;WAS 16
            JHS     ALCT01
            JMP     ALCT00
ALCT02      CMP     %8,B
            JHS     ALCT01
            JMP     ALCT00
ALCT01      POP     ADDRES
            RETS
*
*******************************************************************************
** OR and AND bit ON plus time turn on/off x of 16 bits
*
*
ANDON       CLR     ADDRES
            MOV     INDEX,B
            LDA     @ANDIN0(B)
            MOV     A,MISC1
            INC     B
            LDA     @ANDIN0(B)
            OR      A,MISC1
            INC     B
```

```
            LDA     @ANDIN0(B)
            OR       A,MISC1
            INC     B
            LDA     @ANDIN0(B)
            OR      A,MISC1
            INC     B
            LDA     @ANDIN0(B)
            OR      A,MISC1
            INC     B
            LDA     @ANDIN0(B)
            OR      A,MISC1
            INC     B
            LDA     @ANDIN0(B)
            MOV     A,MISC2
            INC     B
            LDA     @ANDIN0(B)
            OR      A,MISC2
            INC     B
            LDA     @ANDIN0(B)
            OR      A,MISC2
            INC     B
            LDA     @ANDIN0(B)
            OR      A,MISC2
            CMP     %>00,MISC1
            JNE     ANDONG
            CMP     %>00,MISC2
            JNE     ANDONA
            BR      @ANDONX
*
ANDONA      BR      @ANDFX1
*
ANDONG      MOV     INDEX,B                 ;AND input on
            LDA     @ANDIN0(B)
            MOV     A,MISC2
            ADD     %>04,B
            LDA     @ANDIN0(B)
            AND     A,MISC2
            MOV     MISC2,MISC1
            CMP     %00,MISC2
            JEQ     ANDONH
            AND     INP1ST,MISC1
            CMP     MISC2,MISC1
            JNE     ANDONL
            MOV     %1,ADDRES
*
ANDONH      MOV     INDEX,B                 ;AND output on
            ADD     %>01,B
            LDA     @ANDIN0(B)
            MOV     A,MISC2
            ADD     %>05,B
            LDA     @ANDIN0(B)
            AND     A,MISC2
            MOV     MISC2,MISC1
            CMP     %00,MISC2
            JEQ     ANDONI
            AND     OUT1ST,MISC1
            CMP     MISC2,MISC1
            JNE     ANDONL
            MOV     %1,ADDRES
            JMP     ANDONI
*
ANDONL      BR      @ANDONX
*
ANDONI      MOV     INDEX,B                 ;AND input off
```

```
            ADD     %>00,B
            LDA     @ANDIN0(B)
            MOV     A,MISC2
            ADD     %>04,B
            LDA     @ANDIN0(B)
            INV     A
            AND     A,MISC2
            CMP     %00,MISC2
            JEQ     ANDONJ
            MOV     INP1ST,MISC1
            INV     MISC1
            AND     MISC2,MISC1
            CMP     MISC2,MISC1
            JNE     ANDONL
            MOV     %1,ADDRES
*
ANDONJ      MOV     INDEX,B                 ;AND output off
            ADD     %>01,B
            LDA     @ANDIN0(B)
            MOV     A,MISC2
            ADD     %>05,B
            LDA     @ANDIN0(B)
            INV     A
            AND     A,MISC2
            CMP     %00,MISC2
            JEQ     ANDONP
            MOV     OUT1ST,MISC1
            INV     MISC1
            AND     MISC2,MISC1
            CMP     MISC2,MISC1
            JNE     ANDONL
            MOV     %1,ADDRES
*
ANDONP      MOV     INDEX,B
            ADD     %6,B
            LDA     @ANDIN0(B)
            AND     %>80,A
            CMP     %>80,A
            JNE     ANDONB
*
            MOV     INDEX,B                 ;AND TIME ON
            ADD     %>06,B
            LDA     @ANDIN0(B)
            AND     %>7F,A
            MOV     A,MISC1
            INC     B
            LDA     @ANDIN0(B)
            MOV     A,MISC2
            MOVD    BYTETM,B
            AND     %>7F,A
            CMP     MISC1,A
            JHS     ANDONQ
            JMP     ANDONL
*
ANDONQ      CMP     MISC2,B
            JHS     ANDONR
            JMP     ANDONL
*
ANDONR      MOV     INDEX,B
            ADD     %>08,B
            LDA     @ANDIN0(B)
            MOV     A,MISC1
            INC     B
            LDA     @ANDIN0(B)
```

```
            MOV     A,MISC2
            MOVD    BYTETM,B
            AND     %>7F,A
            CMP     MISC1,A
            JGT     CUMQAT
            JEQ     ANDONS
            MOV     %1,ADDRES
            BR      @ANDONM
*
CUMQAT      BR      @ANDONX
*
ANDONS      CMP     MISC2,B
            JHS     CUMQAT
            MOV     %1,ADDRES
            BR      @ANDONM
*
ANDONB      MOV     INDEX,B             ;OR input on
            ADD     %>02,B
            LDA     @ANDIN0(B)
            MOV     A,MISC2
            MOV     INDEX,B
            ADD     %>04,B
            LDA     @ANDIN0(B)
            AND     MISC2,A
            AND     INP1ST,A
            CMP     %>00,A
            JNE     ANDONF
*
ANDONC      MOV     INDEX,B             ;OR output on
            ADD     %>03,B
            LDA     @ANDIN0(B)
            MOV     A,MISC2
            MOV     INDEX,B
            ADD     %>05,B
            LDA     @ANDIN0(B)
            AND     MISC2,A
            AND     OUT1ST,A
            CMP     %>00,A
            JNE     ANDONF
*
ANDOND      MOV     INDEX,B             ;OR input off
            ADD     %>02,B
            LDA     @ANDIN0(B)
            MOV     A,MISC2
            MOV     INDEX,B
            ADD     %>04,B
            LDA     @ANDIN0(B)
            INV     A
            AND     MISC2,A
            MOV     INP1ST,MISC1
            INV     MISC1
            AND     MISC1,A
            CMP     %>00,A
            JNE     ANDONF
*
ANDONE      MOV     INDEX,B             ;OR output off
            ADD     %>03,B
            LDA     @ANDIN0(B)
            MOV     A,MISC2
            MOV     INDEX,B
            ADD     %>05,B
            LDA     @ANDIN0(B)
            INV     A
            AND     MISC2,A
```

```
            MOV     OUT1ST,MISC1
            INV     MISC1
            AND     MISC1,A
            CMP     %>00,A
            JNE     ANDONF
            JMP     ANDON3
*
ANDONF      BR      @ANDONT
*
ANDON3      MOV     INDEX,B
            ADD     %6,B
            LDA     @ANDIN0(B)
            AND     %>80,A
            CMP     %>80,A
            JEQ     ANDONM
*
ANDFX1      MOV     INDEX,B                 ;OR TIME ON
            ADD     %>06,B
            LDA     @ANDIN0(B)
            AND     %>7F,A
            MOV     A,MISC1
            INC     B
            LDA     @ANDIN0(B)
            MOV     A,MISC2
            MOVD    BYTETM,B
            AND     %>7F,A
            CMP     MISC1,A
            JHS     ANDONY
            BR      @ANDONM
*
ANDONY      CMP     MISC2,B
            JHS     ANDONO
            BR      @ANDONM
*
ANDONO      MOV     INDEX,B                 ;OR TIME OFF
            ADD     %>08,B
            LDA     @ANDIN0(B)
            AND     %>7F,A
            MOV     A,MISC1
            INC     B
            LDA     @ANDIN0(B)
            MOV     A,MISC2
            MOVD    BYTETM,B
            AND     %>7F,A
            CMP     MISC1,A
            JGT     ANDONM
            JEQ     ANDONW
            BR      @ANDONT
*
ANDONW      CMP     MISC2,B
            JHS     ANDONM
            BR      @ANDONT
*
ANDONM      CMP     %00,ADDRES
            JEQ     ANDONX
*
ANDONT      MOV     ALCONT,B
            LDA     @ALTBLN(B)
            CMP     %>08,B
            JHS     ANDONU
            OR      A,OUT1ST
            JMP     ANDONV
ANDONU      OR      A,OUT2ST
ANDONV      RETS
```

```
*
ANDONX  MOV     ALCONT,B
        LDA     @ALTBLN(B)
        INV     A
        CMP     %>08,B
        JHS     ANDON0
        AND     A,OUT1ST
        JMP     ANDON1
ANDON0  AND     A,OUT2ST
ANDON1  RETS
*
*
*******************************************************************************
** control routine to test status for Backflush routines
*
BKFLSH  CLR     B
BKFL00  MOV     B,ALCONT
        MPY     %10,ALCONT
        CLR     A
        MOV     B,INDEX
        CALL    @BKFON
        MOV     ALCONT,B
        INC     B
        CMP     %4,B                    ;WAS 16
        JHS     BKFL01
        JMP     BKFL00
BKFL01  RETS
*
*
*******************************************************************************
*** individual program number tests for date and month
*
BKFON   MOV     INDEX,B
        LDA     @SDAY1(B)
        MOV     A,MISC1
        INC     B
        LDA     @SDAY1(B)
        MOV     A,MISC2
        CMP     MISC2,MONTH
        JHS     BKFS03
        JMP     BKFOUT
*
BKFS03  CMP     MISC1,DATE
        JHS     BKFS00
        JMP     BKFOUT
*
BKFS00  MOV     INDEX,B
        ADD     %2,B
        LDA     @SDAY1(B)
        MOV     A,MISC1
        INC     B
        LDA     @SDAY1(B)
        MOV     A,MISC2
        CMP     MISC2,MONTH
        JGT     BKFOUT
        JMP     BKFS01
*
BKFS01  CMP     MISC1,DATE
        JGT     BKFOUT
*
BKFS02  BR      @DOBKFS
BKFOUT  RETS
*
** test for decrementer present and decrement until zero then reload
```

```
*
DOBKFS  MOV     CKBTSH,B            ;decrement counter if <> 0, only on minutes
        OR      CKBTSL,B
        CMP     %00,B
        JNE     DOBKF0              ;if seconds <> 0 then exit
*
        MOV     INDEX,B
        ADD     %4,B                ;find out if there is a decrementer value
        LDA     @SDAY1(B)
        MOV     A,MISC1
        INC     B
        LDA     @SDAY1(B)
        OR      A,MISC1
        CMP     %00,MISC1           ;if no program then exit
        JEQ     DOBKF0

        MOV     INDEX,B             ;test decrementer registers
        ADD     %8,B
        LDA     @SDAY1(B)
        MOV     A,MISC1
        INC     B
        LDA     @SDAY1(B)
        OR      A,MISC1
        CMP     %00,MISC1
        JNE     DOBKF1              ;if <> 0 then decrement
        BR      @PURGE              ;set flag type
*
DOBKF2  MOV     INDEX,B             ;reload decrementer
        ADD     %4,B
        LDA     @SDAY1(B)
        MOV     A,MISC1
        INC     B
        LDA     @SDAY1(B)
        MOV     A,MISC2
        MOV     INDEX,B
        ADD     %8,B
        MOV     MISC1,A
        STA     @SDAY1(B)
        INC     B
        MOV     MISC2,A
        STA     @SDAY1(B)
        JMP     DOBKF0              ;don't decrement and then exit
*
DOBKF1  CALL    @WAITY
        MOV     INDEX,B             ;get decrementer value
        ADD     %8,B
        LDA     @SDAY1(B)
        MOV     A,MISC1
        INC     B
        LDA     @SDAY1(B)
        MOV     A,MISC2
        SUB     %1,MISC2
        SBB     %0,MISC1            ;decrement word
        MOV     INDEX,B
        ADD     %8,B
        MOV     MISC1,A
        STA     @SDAY1(B)           ;save it back
        INC     B
        MOV     MISC2,A
        STA     @SDAY1(B)
DOBKF0  BR      @BKFOUT             ;take off fasssssst
*
PURGE   CMP     %02,PBMODE
        JEQ     PURGX
```

```
            CMP     %01,PBMODE
            JEQ     PURGY
            BR      @DOBKF0
*
PURGX       MOV     %01,BKFLAG
            BR      @DOBKF2
*
PURGY       MOV     %02,BKFLAG
            BR      @DOBKF2
*
*** backflush control routines
*
BFXLSH      CLR     B
BFXL00      PUSH    B
            MOV     B,A
            MPY     %10,A
            CLR     A
            MOV     B,POINTR
            CALL    @BFXON
            POP     B
            INC     B
            CMP     %4,B                    ;WAS 16
            JHS     BFXL01
            JMP     BFXL00
BFXL01      RETS
*
BFXON       MOV     POINTR,B
            LDA     @SDAY1(B)
            MOV     A,MISC1
            INC     B
            LDA     @SDAY1(B)
            MOV     A,MISC2
            CMP     MISC2,MONTH
            JHS     BFXS03
            JMP     BFXOUT
*
BFXS03      CMP     MISC1,DATE
            JHS     BFXS00
            JMP     BFXOUT
*
BFXS00      MOV     POINTR,B
            ADD     %2,B
            LDA     @SDAY1(B)
            MOV     A,MISC1
            INC     B
            LDA     @SDAY1(B)
            MOV     A,MISC2
            CMP     MISC2,MONTH
            JGT     BFXOUT
            JMP     BFXS01
*
BFXS01      CMP     MISC1,DATE
            JGT     BFXOUT
*
BFXS02      CMP     %1,BKFLAG
            JNE     BFXS05
            BR      @PURGA
BFXS05      CMP     %2,BKFLAG
            JNE     BFXOUT
            BR      @PURGB
BFXOUT      RETS
*
PURGA       CLR     B
            MOV     %>80,A
```

```
            STA     @PORT2
PURGA0      PUSH    B
            MOV     POINTR,B
            ADD     %6,B
            LDA     @SDAY1(B)
            MOV     A,MISC2
            POP     B
            PUSH    B
            LDA     @ALTBLN(B)
            OR      %>80,A
            STA     @PORT2
            CLR     SECOND
PURGA1      CMP     MISC2,SECOND
            JL      PURGA1
*
            MOV     POINTR,B
            ADD     %7,B
            LDA     @SDAY1(B)
            MOV     A,MISC2
            POP     B
            PUSH    B
            LDA     @ALTBLN(B)
            INV     A
            AND     %>80,A
            STA     @PORT2
            CLR     SECOND
PURGA2      CMP     MISC2,SECOND
            JL      PURGA2
            POP     B
            INC     B
            CMP     %7,B
            JL      PURGA0
            MOV     %00,A
            STA     @PORT2
            BR      @BFXOUT
*
PURGB       CLR     B
            MOV     %>80,A
            STA     @PORT2
PURGB0      PUSH    B
            MOV     POINTR,B
            ADD     %6,B
            LDA     @SDAY1(B)
            MOV     A,MISC2
            POP     B
            PUSH    B
            LDA     @ALTBLN(B)
            OR      %>80,A
            STA     @PORT1
            CLR     SECOND
PURGB1      CMP     MISC2,SECOND
            JL      PURGB1
*
            MOV     POINTR,B
            ADD     %7,B
            LDA     @SDAY1(B)
            MOV     A,MISC2
            POP     B
            PUSH    B
            LDA     @ALTBLN(B)
            INV     A
            AND     %>80,A
            STA     @PORT1
            CLR     SECOND
```

```
PURGB2  CMP     MISC2,SECOND
        JL      PURGB2
        POP     B
        INC     B
        CMP     %8,B
        JL      PURGB0
*
PURGC   CLR     B
PURGC0  PUSH    B
        MOV     POINTR,B
        ADD     %6,B
        LDA     @SDAY1(B)
        MOV     A,MISC2
        POP     B
        PUSH    B
        LDA     @ALTBLN(B)
        OR      %>80,A
        STA     @PORT2
        CLR     SECOND
PURGC1  CMP     MISC2,SECOND
        JL      PURGC1
*
        MOV     POINTR,B
        ADD     %7,B
        LDA     @SDAY1(B)
        MOV     A,MISC2
        POP     B
        PUSH    B
        LDA     @ALTBLN(B)
        INV     A
        AND     %>80,A
        STA     @PORT2
        CLR     SECOND
PURGC2  CMP     MISC2,SECOND
        JL      PURGC2
        POP     B
        INC     B
        CMP     %7,B
        JL      PURGC0
        MOV     %>00,A
        STA     @PORT2
        BR      @BFXOUT
*
*
********************************************************************************
********************** system interrupt handlers ***************************
*
** TRAP 01, performs the task of inserting a new number for timer 1
*
INT01   RETI                                    ;counter input #1 routine
*
** TRAP 02, performs the task of inserting a new number for timer 2
*
RTMUPD  PUSH    A                               ;protect a register
        PUSH    B
        PUSH    MISC1
        PUSH    MISC2
        CALL    @TIME                           ;call one second update routine
        CALL    @TMCNVT
*
        CMP     %>00,PBMODE
        JEQ     RTMUP1
        CMP     %>01,PBMODE
        JEQ     RTMUP2
```

```
            CMP     %>02,PBMODE
            JEQ     RTMUP3

RTMUP1      CALL    @ALCNTL                 ;call the bit set/reset algorithm
            POP     MISC2
            POP     MISC1
            POP     B
            POP     A                       ;restore a register
            RETI                            ;return from interrupt
*
RTMUP2      CALL    @BKFLSH                 ;call the bit set/reset algorithm
            POP     MISC2
            POP     MISC1
            POP     B
            POP     A                       ;restore a register
            RETI                            ;return from interrupt
*
RTMUP3      CALL    @BKFLSH                 ;call the bit set/reset algorithm
            CALL    @ALCNTL                 ;call the bit set/reset algorithm
            POP     MISC2
            POP     MISC1
            POP     B
            POP     A                       ;restore a register
            RETI                            ;return from interrupt
*
** baud rate interrupt handler
*
INT03       RETI                            ;counter input #2 routine
*
** timer 4 interrupt handler
*
INT04       RETI                            ;return from interrupt
*
** timer 5 interrupt handler
*
INT05       RETI                            ;return from interrupt
*
** trap 6 routine handler
*
TRAP06      RETS                            ;return to sender
*
** trap 7 routine handler
*
TRAP07      RETS                            ;return to sender
*
*
*******************************************************************************
****************************** system tables *********************************
*
** text for for show string routine
*
*     SIZE    0123456789ABCDEF **
*
THE         TEXT    '     THE\'
ONBORD      TEXT    'ONBOARD SOLUTION\'
FOR         TEXT    '     For\'
CUSTOM      TEXT    ' Put Name Here \'
COMAND      TEXT    'Command ? \'
RUNING      TEXT    'Running\'
SETTIM      TEXT    'Set Time = \'
SETDAT      TEXT    'Set Date = \'
ADDRSS      TEXT    'Address = \'
MANUAL      TEXT    'Manual Testing\'
MODE        TEXT    'Mode = \'
```

```
TESTNG    TEXT     'AutoTesting I/O\'
ERROR     TEXT     '  ** ERROR **\'
DEVTYP    TEXT     'PLC/BF (1/2) \'
OPRTYP    TEXT     'PLC/BF/BOTH\'
OPRTY1    TEXT     '(0/1/2) \'
SDAY      TEXT     ' Start Date \'
SMNTH     TEXT     'Start Month \'
EDAY      TEXT     ' End Date \'
EMNTH     TEXT     'End Month \'
SPACE     TEXT     'Time Between\'
CYCLES    TEXT     'Cycles \'
DURON     TEXT     ' On Time \'
DUROF     TEXT     'Off Time \'
POSIT     TEXT     'Which Prog. \'
POSIT1    TEXT     '(0/1/2/3) \'
BACKF     TEXT     'Backflush Mode \'
*
*     SIZE    0123456789ABCDEF **
*
DAYTBL    BYTE     32,29,32,31,32,31,32,32,31,32,31,32
*
ALTBLN    BYTE     >01,>02,>04,>08,>10,>20,>40,>80
ALTBLX    BYTE     >01,>02,>04,>08,>10,>20,>40,>80
*
CHAR00    BYTE     >00,>01,>02,>02,>03,>03,>03,>03
          BYTE     >04,>04,>04,>04,>04,>04,>04,>04
*
CHAR01    BYTE     >00,>01,>02,>03,>04,>05,>06,>07
          BYTE     >08,>09,>0A,>0B,>0C,>0D,>0E,>0F
*
*
COPYRT    TEXT     'This program is the sole property of Vortex '
          TEXT     'Research Inc, Canada, Copyright (C) 1990'
          TEXT     'Unauthorized duplication is prohibited by'
          TEXT     'International Copyright Laws. Program Author:'
          TEXT     'S.J.M./Vortex Research Inc. : July 19, 1990.'
*
*
** interrupt vectors and trap initialization
*
VECTOR    AORG     >FFF0                    ;trap vector start location
*
          DATA     TRAP07                   ;trap 7, dummy routine, not used
          DATA     TRAP06                   ;trap 6, dummy routine, not used
          DATA     INT05                    ;trap 5, dummy interrupt, not used
          DATA     INT04                    ;trap 4, dummy interrupt, not used
          DATA     INT03                    ;trap 3, timer 3 controller, serial baud rate
          DATA     RTMUPD                   ;trap 2, RTM MODE timer 1 controller
          DATA     INT01                    ;trap 1, timer 1 controller, Real Time Clock
          DATA     START                    ;trap 0, reset vector, program start-up
*
*
*******************************************************************************
**************************** program end ***********************************
*
          END
```