```
;
;  PCGET  This CP/M program will obtain a file from a PC sent via a serial
;  port and write it to RAM on the CP/M system. The program utilized the
;  XModem format/protocol. (Use Absolute Telnet).
;
;  The program seems to work up to at least 38,400 Baud, fine. The memory
;  designation can be anywhere in teh S100 bus 16MG addresss space. It is
;  typically used to place an 8086/80386 monitor in high RAM for testing
;  before burnuing EEPROMS etc.  This ability counts on the 4K address window
;  that can be configured to point to above the first 64K of RAM on the
;  S100Computers/N8VEM Z80 CPU board. There is no practical limit to the
;  size of the code that needs to be transfered, but it cannot go across a
;  64K segment boundry, (no fundemental reason just have not added that code).
;  The program can also be used to place data in the lower 64K address
;  space so long as it is located at >= 1000H.
;
;       Author  John Monahan   (monahan@vitasoft.org)

;       V0.1    1/21/2013      Initial version


;DEFINE ASCII CHARACTERS USED
SOH             EQU     1
EOT             EQU     4
ACK             EQU     6
NAK             EQU     15H
LF              EQU     10
CR              EQU     13
ESC             EQU     1BH

CONOUT$PORT     EQU     1H              ;Propeller (or other) console ports
CONIN$PORT      EQU     1H
CONSTAT$PORT    EQU     0H

BASE$PORT       EQU     0A1H            ;>>> SETUP FOR S100Computers Board <<<
SPEECH$CTL$PORT         EQU     0A0H            ;Serial Ctrl port A for speech synthesizer
SPEECH$DATA$PORT EQU    0A2H            ;Serial Data port A for speech synthesizer

MODEM$CTL$PORT EQU      BASE$PORT       ;A1H or 010H
MODEM$SEND$MASK         EQU     4
SEND$READY      EQU     4               ;VALUE WHEN READY
MODEM$RECV$MASK         EQU     1
RECV$READY      EQU     1               ;BIT ON WHEN READY
MODEM$DATA$PORT         EQU     BASE$PORT+2     ;A3H or 012H

ERROR$LIMIT     EQU     5               ;MAX ALLOWABLE ERRORS

CPU$SWITCH1$PORT EQU    0D2H            ;lower Z80 window (0-3FFH, not used)
CPU$SWITCH2$PORT EQU    0D3H            ;a value of 1,2,3...E,F, switches in 4000-7FFFH RAM to RAM above 64K
                                        ;So a value of E0 would switch in E0000H-E3FFFH to RAM at 4000H - 7FFFH



        ORG     100H

START:  LXI     H,0                     ;;INIT PRIVATE STACK,  HL=0
        DAD     SP                      ;HL=STACK FROM CP/M
        SHLD    STACK                   ;..SAVE IT
        LXI     SP,STACK                ;SP=MY STACK

        LXI     D,SIGNON                ;GET SIGNON MESSAGE
        CALL    PRINT$MESSAGE

        CALL    INIT$SCC                ;MASTER RESET THE ACIA
                                        ;GOBBLE UP GARBAGE CHARS FROM THE LINE
        MVI     B,1                     ;TIMEOUT DELAY
        CALL    RECV

START1: LXI     D,ENTER$RAM$LOC
        CALL    PRINT$MESSAGE
        CALL    GET5DIGITS              ;Input Segment (00,10,11,12...20,30,....E0,F0,F1,F2,.. etc.)
```

```
        JNC     START3                  ;NC, if data is valid
        CPI     ESC
        JZ      EXIT3                   ;ESC Abort, back to CPM
        CALL    ZCRLF
        JMP     START1

START3: STA     SEG$POINTER             ;Store requested "segment" here

        LXI     D,START1$RAM$LOC        ;"Will load data starting at RAM location "
        CALL    PRINT$MESSAGE
        LDA     SEG$POINTER             ;Get Seg pointer back
        CALL    A$HEXOUT
        LXI     D,START2$RAM$LOC        ;'000H',CR,LF,LF,'$'
        CALL    PRINT$MESSAGE

        LDA     SEG$POINTER             ;Check if segment destination is 0
        ORA     A
        JNZ     RAM1
        LXI     D,RAM$0$LOC             ;Error segment must be at least 01H
        CALL    PRINT$MESSAGE
        JMP     START1

RAM1:   CPI     10H                     ;Are we in the first 64K of RAM
        JNC     RAM2                    ;NC then > 64K

        RLC                             ;Yes, then shift to high nibble
        RLC
        RLC
        RLC
        ANI     0F0H
        MOV     H,A                     ;Setup HL to point to RAM < 64K
        MVI     L,0
        SHLD    RAM$POINTER
        SHLD    DISPLAY$POINTER                 ;Both are the same if < 64K
        MVI     A,0
        STA     WINDOW$FLAG             ;Store flag for later
        LXI     D,RAM$1$LOC             ;Is it < 64K message
        CALL    PRINT$MESSAGE
        JMP     START2

RAM2:   RLC                             ;Need to add in the offset from 4000H
        RLC
        RLC
        RLC
        ANI     30H                     ;Within the 4K window the code can have an offset of 0,1000H,2000H or 3000H
        MOV     D,A                     ;Setup HL to point to RAM
        MVI     E,0
        LXI     H,4000H                 ;Window location where data will be placed - ALWAYS
        DAD     D                       ;Start point within this 4K window
        SHLD    RAM$POINTER             ;RAM$POINTER is now set to start at the correct location within the 4K window

        LDA     SEG$POINTER             ;Get Segment pointer
        ANI     0FCH                    ;Has to be a 4K boundry
        OUT     CPU$SWITCH2$PORT        ;This will switch out 4000-7FFFH to RAM > 64K value in [A=D3H]
        MVI     A,1
        STA     WINDOW$FLAG             ;Store flag for later

        LDA     SEG$POINTER             ;Next we need to construct a display/progress pointer
        RLC                             ;Shift up 4
        RLC
        RLC
        RLC
        ANI     0F0H
        MOV     H,A
        MVI     L,0
        SHLD    DISPLAY$POINTER                 ;This word store is to keep track of final location display

START2: PUSH    H
        LXI     H,DOWNLOAD$MSG          ;Speak "downloading file"
        CALL    SMSG
        POP     H
```

```
RECV$LOOP:
        XRA     A                       ;GET 0
        STA     ERRCT                   ;INIT ERROR COUNT
RECV$HDR:
        LXI     D,RMSG                  ;Reading sector message
        CALL    PRINT$MESSAGE
        LDA     SECTNO
        INR     A
        CALL    A$HEXOUT

        PUSH    D                       ;Next show next RAM loaction
        PUSH    H
        LXI     D,RAM$MSG               ;"H   Will write to RAM location"
        CALL    PRINT$MESSAGE

        LDA     WINDOW$FLAG             ;Is it > 64K
        ORA     A
        JZ      LESS64K

        CALL    Display5Digits          ;Display distination RAM pointer
        JMP     OVER64K
LESS64K:
        CALL    Display4Digits          ;Display distination RAM pointer
OVER64K:
        LXI     D,HCRLF$MSG             ;"H. CR/LF"
        CALL    PRINT$MESSAGE
        POP     H
        POP     D

        MVI     B,5                     ;5 SEC TIMEOUT
        CALL    RECV
        JNC     RHNTO                   ;NO TIMEOUT
RECV$HDR$TIMEOUT:
        CALL    TOUT                    ;PRINT TIMEOUT

RECV$SECT$ERR:                          ;PURGE THE LINE OF INPUT CHARS
        MVI     B,1                     ;1 SEC W/NO CHARS
        CALL    RECV
        JNC     RECV$SECT$ERR           ;LOOP UNTIL SENDER DONE
        MVI     A,NAK
        CALL    SEND                    ;SEND NAK
        LDA     ERRCT
        INR     A
        STA     ERRCT
        CPI     ERROR$LIMIT
        JC      RECV$HDR
        CALL    CHECK$FOR$QUIT
        JZ      RECV$HDR
        LXI     D,BAD$HEADER
        CALL    PRINT$MESSAGE
        JMP     EXIT                    ;Abort back to CPM


RHNTO:  CPI     SOH                     ;GOT CHAR - MUST BE SOH
        JZ      GOT$SOH
        ORA     A                       ;00 FROM SPEED CHECK?
        JZ      RECV$HDR
        CPI     EOT
        JZ      GOT$EOT
                                        ;DIDN'T GET SOH -
        CALL    A$HEXOUT
        LXI     D,ERRSOH
        CALL    PRINT$MESSAGE
        JMP     RECV$SECT$ERR

GOT$SOH:
        MVI     B,1
        CALL    RECV
        JC      RECV$HDR$TIMEOUT
        MOV     D,A                     ;D=BLK #
```

```
        MVI     B,1
        CALL    RECV                    ;GET CMA'D SECT #
        JC      RECV$HDR$TIMEOUT
        CMA
        CMP     D                       ;GOOD SECTOR #?
        JZ      RECV$SECTOR
                                        ;GOT BAD SECTOR #
        LXI     D,ERR2
        CALL    PRINT$MESSAGE
        JMP     RECV$SECT$ERR


RECV$SECTOR:                            ;Get 128 Bytes
        MOV     A,D                     ;GET SECTOR #
        STA     RECVD$SECT$NO
        MVI     C,0                     ;INIT CKSUM
        LXI     H,80H                   ;POINT TO BUFFER
RECV$CHAR:
        MVI     B,1                     ;1 SEC TIMEOUT
        CALL    RECV                    ;GET CHAR
        JC      RECV$HDR$TIMEOUT
        MOV     M,A                     ;STORE CHAR
        INR     L                       ;DONE?
        JNZ     RECV$CHAR
                                        ;VERIFY CHECKSUM
        MOV     D,C                     ;SAVE CHECKSUM
        MVI     B,1                     ;TIMEOUT
        CALL    RECV                    ;GET CHECKSUM
        JC      RECV$HDR$TIMEOUT
        CMP     D                       ;CHECK
        JNZ     RECV$CKSUM$ERR

        LDA     RECVD$SECT$NO           ;GOT A SECTOR, WRITE IF = 1+PREV SECTOR
        MOV     B,A                     ;SAVE IT
        LDA     SECTNO                  ;GET PREV
        INR     A                       ;CALC NEXT SECTOR #
        CMP     B                       ;MATCH?
        JNZ     DO$ACK

                                        ;GOT NEW SECTOR - NOW WRITE IT TO MEMORY (NOT DISK)
        LXI     D,80H                   ;Where data is stored (128 bytes)
        LHLD    RAM$POINTER             ;Will deposit all data starting here (Default 4000H)
MORE:   LDAX    D                       ;Get data
        MOV     M,A                     ;Store data in RAM area
        INX     H
        INR     E                       ;Always 80H -> FFH
        JNZ     MORE                    ;Do 128 Bytes at a time

        LDA     WINDOW$FLAG
        ORA     A
        JZ      MORE1                   ;If < 64K then just continue

        MOV     A,H                     ;Allow 4000,7FFFH
        CPI     80H
        JNZ     MORE1                   ;4K not done yet
        LDA     SEG$POINTER
        ANI     0FCH                    ;Has to be a 4K boundry
        ADI     4                       ;Point to next 4K segment
        STA     SEG$POINTER             ;Store it
        OUT     CPU$SWITCH2$PORT        ;This will switch in the next 4K segment
        LXI     H,4000H                 ;Always begin here

MORE1:  SHLD    RAM$POINTER             ;Store for next sector

        PUSH    H
        LHLD    DISPLAY$POINTER                 ;Next update Destination display pointer
        LXI     D,0080H
        DAD     D
        SHLD    DISPLAY$POINTER
        POP     H
```

```
        LDA     RECVD$SECT$NO           ;Indicate we transferred a sector
        STA     SECTNO                  ;UPDATE SECTOR #
DO$ACK  MVI     A,ACK
        CALL    SEND
        JMP     RECV$LOOP


RECV$CKSUM$ERR:
        LXI     D,ERR3
        CALL    PRINT$MESSAGE
        JMP     RECV$SECT$ERR

GOT$EOT:                                ;DONE - CLOSE UP SHOP
        MVI     A,ACK                   ;ACK THE EOT
        CALL    SEND
        CALL    ZCRLF
        LXI     H,FINISH$MSG            ;Speak downloading finished
        CALL    SMSG
        LXI     D,TRANS$DONE
EXIT2:  CALL    PRINT$MESSAGE
EXIT:   XRA     A
        OUT     CPU$SWITCH1$PORT        ;This will set Z80 window back to 0000-3FFFH in first 64K RAM space
        OUT     CPU$SWITCH2$PORT        ;This will set Z80 window back to 4000-7FFFH in first 64K RAM space
        LHLD    STACK                   ;GET ORIGINAL STACK
        SPHL                            ;RESTORE IT
        JMP     0F000H                  ;EXIT -- Z80 Monitor (Not to CPM because Banked CPM may have been using some
                                        ;of this extended memory RAM)
EXIT1:  LXI     D,ABORT$MSG
        JMP     EXIT2

EXIT3:  LXI     D,ABORT$MSG
        JMP     0                       ;Can still get back to CPM. We have not yet altered window


Display5Digits:                                 ;Display the 5 digit RAM pointer Hex address
        LDA     SEG$POINTER
        CALL    A$HIGH$NIBBLE           ;Assume 64K segments (i.e. not accross boundries)
        LHLD    DISPLAY$POINTER
        MOV     A,H
        CALL    A$HEXOUT
        MOV     A,L
        CALL    A$HEXOUT
        RET


Display4Digits:                                 ;Display the 4 digit RAM pointer Hex address
        LHLD    RAM$POINTER
        MOV     A,H
        CALL    A$HEXOUT
        MOV     A,L
        CALL    A$HEXOUT
        RET



;-----------------------   S U B R O U T I N E S -----------------------------

                                        ;INITITIALIZE THE Zilog SCC SERIAL PORT
INIT$SCC:
        LXI     D,MSG$INIT              ;Say Initilizing ACIA
        CALL    PRINT$MESSAGE

        MVI     A,MODEM$CTL$PORT        ;Program the SCC Channel B (A1,A3 or 10,12H) for 19K Baud
        MOV     C,A
        MVI     B,14                    ;Byte count (14), for OTIR below
        LXI     H,SCCINIT
        DB      0EDH, 0B3H              ;Z80 opcode for OTIR
        LXI     H,SPEED$MSG             ;Speak baud rate set
        CALL    SMSG
        RET
```

```
PRINT$MESSAGE:                      ;Print a string (Pointer = DE, ending in '$')
       PUSH    PSW                  ;No registers changed
       PUSH    B
PRINT1: LDAX   D
       CPI     '$'
       JZ      PRINT$DONE
       MOV     C,A
       CALL    ZCO
       INX     D
       JMP     PRINT1
PRINT$DONE:
       POP     B
       POP     PSW
       RET

;-----------------------------------------------------
;   SERIAL PORT GET CHARACTER ROUTINE
;-----------------------------------------------------
;
RECV   PUSH    D                    ;SAVE
       MVI     A,5H                 ;Lower RTS line
       OUT     MODEM$CTL$PORT       ;Sel Reg 5
       MVI     A,11101010B          ;EAH
       OUT     MODEM$CTL$PORT
       NOP
       NOP
MSEC:  LXI     D,0BBBBH             ;1 SEC DCR COUNT
MWTI:  IN      MODEM$CTL$PORT
       ANI     MODEM$RECV$MASK
       CPI     RECV$READY
       JZ      MCHAR                ;GOT CHAR
       DCR     E                    ;COUNT DOWN
       JNZ     MWTI                 ;FOR TIMEOUT
       DCR     D
       JNZ     MWTI
       DCR     B                    ;DCR # OF SECONDS
       JNZ     MSEC                 ;MODEM TIMED OUT RECEIVING
       POP     D                    ;RESTORE D,E
       STC                          ;CARRY SHOWS TIMEOUT
       RET
                                    ;GOT MODEM CHAR
MCHAR  IN      MODEM$DATA$PORT
       POP     D                    ;RESTORE DE
       PUSH    PSW                  ;CALC CHECKSUM
       ADD     C
       MOV     C,A
       POP     PSW
       ORA     A                    ;TURN OFF CARRY TO SHOW NO TIMEOUT
       RET

;-----------------------------------------------------
;   SERIAL PORT SEND CHARACTER ROUTINE
;-----------------------------------------------------

SEND   PUSH    PSW                  ;CHECK IF MONITORING OUTPUT
       ADD     C                    ;CALC CKSUM
       MOV     C,A
SENDW  IN      MODEM$CTL$PORT       ;Don't worry PC is always fast enough!
       ANI     MODEM$SEND$MASK
       CPI     SEND$READY
       JNZ     SENDW
       POP     PSW                  ;GET CHAR
       OUT     MODEM$DATA$PORT
;
                                    ;Raise RTS line to prevent the next character arriving
       MVI     A,5H                 ;while the Z80 is busy processing info
       OUT     MODEM$CTL$PORT       ;Sel Reg 5
       MVI     A,11101000B          ;E8H
       OUT     MODEM$CTL$PORT
       RET
```

```
;-------------------------------------------------
;   SPEECH PORT, SEND CHARACTER ROUTINE
;-------------------------------------------------
SPEAK   PUSH    PSW                     ;CHECK IF MONITORING OUTPUT
SPEAKW  IN      SPEECH$CTL$PORT
        ANI     MODEM$SEND$MASK
        CPI     SEND$READY
        JNZ     SPEAKW
        POP     PSW                     ;GET CHAR
        OUT     SPEECH$DATA$PORT
        RET

SMSG:   MOV     A,M                     ;Speak string at [HL] up to '$'
        INX     H
        CPI     CR                      ;Note CR ends string AND initilizes V-Stamp chip to speak
        JZ      DONE$SP
        CALL    SPEAK
        JMP     SMSG
DONESP: CALL    SPEAK
        RET



TOUT:   LXI     D,TOUTM                 ;PRINT TIMEOUT MESSAGE
        CALL    PRINT$MESSAGE
PRINT$ERRCT:
        LDA     ERRCT
        CALL    A$HEXOUT                ;FALL INTO CR/LF
        CALL    ZCRLF
        RET


A$HEXOUT:                               ;HEX OUTPUT in [A]
        PUSH    PSW
        RAR
        RAR
        RAR
        RAR
        CALL    NIBBL
        POP     PSW
NIBBL:  ANI     0FH
        CPI     10
        JC      ISNUM
        ADI     7
ISNUM:  ADI     '0'
        PUSH    PSW
        PUSH    B
        MOV     C,A
        CALL    ZCO
        POP     B
        POP     PSW
        RET

A$HIGH$NIBBLE:                          ;HEX OUTPUT of high nibble in [A]
        PUSH    PSW
        RAR
        RAR
        RAR
        RAR
        CALL    NIBBL
        POP     PSW
        RET

                                        ;MULTIPLE ERRORS, ASK IF TIME TO QUIT
CHECK$FOR$QUIT:
        XRA     A                       ;GET 0
        STA     ERRCT                   ;RESET ERROR COUNT
        LXI     D,QUITM
        CALL    PRINT$MESSAGE
        CALL    ZCI
```

```
        PUSH    PSW                     ;SAVE CHAR
        CALL    ZCRLF
        POP     PSW
        CPI     'R'
        RZ                              ;RETURN IF RETRY
        CPI     'r'
        RZ
        CPI     'Q'                     ;QUIT?
        JNZ     LCQ
        ORA     A                       ;TURN OFF ZERO FLAG
        RET
LCQ:    CPI     'q'
        JNZ     CHECK$FOR$QUIT
        ORA     A                       ;TURN OFF ZERO FLAG
        RET


GET5DIGITS:
        CALL    CICO                    ;Get 1st character, echo it
        CPI     ESC
        JZ      HEXABORT                ;ESC to abort
        CPI     '/'                     ;check 0-9, A-F
        JC      HEXABORT1
        CPI     'F'+1
        JNC     HEXABORT1
        CALL    ASBIN                   ;Convert to binary
        STA     Digit5                  ;Store it for now

        CALL    CICO                    ;Get 2nd character, echo it
        CPI     ESC
        JZ      HEXABORT                ;ESC to abort
        CPI     '/'                     ;check 0-9, A-F
        JC      HEXABORT1
        CPI     'F'+1
        JNC     HEXABORT1
        CALL    ASBIN                   ;Convert to binary
        STA     Digit4                  ;Store it for now

        CALL    CICO                    ;Get 3nd character, echo it
        CPI     ESC
        JZ      HEXABORT                ;ESC to abort
        CPI     '0'
        JNZ     HEXABORT1

        CALL    CICO                    ;Get 4nd character, echo it
        CPI     ESC
        JZ      HEXABORT                ;ESC to abort
        CPI     '0'
        JNZ     HEXABORT1

        CALL    CICO                    ;Get 5nd character, echo it
        CPI     ESC
        JZ      HEXABORT                ;ESC to abort
        CPI     CR
        LDA     Digit5                  ;Get back first digit entered above
        JZ      FOUR$DIGITS             ;For special case of < 46K

        RLC                             ;Shift to high nibble
        RLC
        RLC
        RLC
        MOV     B,A                     ;Store it
        LDA     Digit4
        ORA     B                       ;add in the first digit
FOUR$DIGITS:
        ORA     A                       ;To return NC
        RET


HEXABORT:                               ;Character was ESC
        MVI     A,ESC
```

```
        STC                     ;Set Carry flag
        RET

HEXABORT1:
        LXI     D,INVALID$MSG
        CALL    PRINT$MESSAGE
        MVI     A,0
        STC                     ;Set Carry flag
        RET


CICO:   CALL    ZCI             ;GET A CHARACTER, convert to UC, ECHO it
        CALL    UPPER
        CPI     ESC
        RZ                      ;Don't echo an ESC
        PUSH    PSW             ;Save it
        PUSH    B
        MOV     C,A
        CALL    ZCO             ;Echo it
        POP     B
        POP     PSW             ;get it back
        RET


UPPER:  CPI     'a'             ;must be >= lowercase a
        RC                      ; else go back...
        CPI     'z'+1           ;must be <= lowercase z
        RNC                     ; else go back...
        SUI     'a'-'A'         ;subtract lowercase bias
        RET

ASBIN:  SUI     30H             ;ASCII TO BINARY CONVERSION ROUTINE
        CPI     0AH
        RM
        SUI     07H
        RET

;-------------------------- CONSOLE I/O Routines ---------------------------------

ZCO:    PUSH    PSW             ;Write character that is in [C]
ZCO1:   IN      0H              ;Show Character
        ANI     04H
        JZ      ZCO1
        MOV     A,C
        CPI     ESC             ;Don't send ESC character confuses console
        JZ      ZCO2
        OUT     CONOUT$PORT
ZCO2:   POP     PSW
        RET

ZCSTS:  IN      0H              ;Get Keyboard Status
        ANI     02H
        RET

ZCI:    IN      CONSTAT$PORT    ;Return keyboard character in [A]
        ANI     02H
        JZ      ZCI
        IN      CONIN$PORT
        RET

ZCRLF:
        PUSH    PSW
        MVI     C,CR
        CALL    ZCO
        MVI     C,LF
        CALL    ZCO
        POP     PSW
        RET
```

```
;--------------------- DATA AREA ----------------------------------------------------

;Initilization table for SCC registers
SCCINIT:        DB      04H     ;1,  Point to WR4
                DB      44H     ;2,  X16 clock,1 Stop,NP
;
                DB      03H     ;3,  Point to WR3
                DB      0C1H    ;4,  Enable reciever, No Auto Enable (Hardware CTS), Recieve 8 bits
;               DB      0E1H    ;4,  Enable reciever, Auto Enable, Recieve 8 bits (for CTS bit)
;
                DB      05H     ;5,  Point to WR5
                DB      0EAH    ;6,  Enable, Transmit 8 bits
;                       ;    Set RTS,DTR, Enable
;
                DB      0BH     ;7,  Point to WR11
                DB      56H     ;8,  Recieve/transmit clock = BRG
;
                DB      0CH     ;9,  Point to WR12
                DB      02H     ;10, Low byte 38,400 Baud
;               DB      06H     ;10, Low byte 19,200 Baud <<<<<<<<<<<
;               DB      0EH     ;10, Low byte 9600 Baud
;               DB      1EH     ;10, Low byte 4800 Baud
;               DB      7EH     ;10, Low byte 1200 Baud for debugging.
;               DB      0FEH    ;10, Low byte 300 Baud for debugging.

                DB      0DH     ;11, Point to WR13
                DB      00H     ;12, High byte for Baud
;               DB      01H     ;12, High byte for Baud
;
                DB      0EH     ;13, Point to WR14
                DB      01H     ;14, Use 4.9152 MHz Clock. Note SD Systems uses a 2.4576 MHz clock, enable BRG
;
                DB      0FH     ;15, Point to WR15
                DB      00H     ;16, Generate Int with CTS going high

SIGNON:         DB      'Load a File from a PC into RAM using the S100Computers IO Board',CR,LF
                DB      'Zilog SCC Ports A1H & A3H. Requires RTS & CTS, 38,400 Baud.',CR,LF,'$'
MSG$INIT        DB      'SCC Port A initilized to 38,400 Baud.',CR,LF,LF,'$'

DOWNLOAD$MSG    DB      'Downloading file Started.',CR
SPEED$MSG       DB      '38,400 Baaud.',CR
RMSG            DB      'WAITING FOR SECTOR #$'
ERRSOH          DB      'H RECEIVED, NOT SOH',CR,LF,'$'
ERR1            DB      'H RECEIVED, NOT ACK',CR,LF,'$'
ERR2            DB      '++BAD SECTOR # IN HDR',CR,LF,'$'
ERR3            DB      '++BAD CKSUM ON SECTOR',CR,LF,'$'
TOUTM           DB      'TIMEOUT $'
QUITM           DB      CR,LF,'+++ MULTIPLE ERRORS ENCOUNTERED +++'
                DB      CR,LF,'TYPE Q TO QUIT, R TO RETRY:$'
RAM$MSG         DB      'H   Will write to RAM location $'
HCRLF$MSG       DB      'H',CR,LF,'$'
FINISH$MSG      DB      'Down loading of file complete.  No Errors',CR
TRANS$DONE      DB      CR,LF,LF,'TRANSFER COMPLETE$'
BAD$HEADER      DB      '++UNABLE TO GET VALID HEADER',0DH,0AH,'$'
ENTER$RAM$LOC   DB      CR,LF,'Enter start RAM destination (1000H - FF000H) (xx000H): $'
START1$RAM$LOC  DB      CR,LF,LF,'Will load data starting at RAM location $'
START2$RAM$LOC  DB      '000H',CR,LF,LF,'$'
ABORT$MSG       DB      CR,LF,LF,'Invalid Character or Program Aborted',CR,LF,'$'
RAM$0$LOC       DB      CR,LF,'Sorry the RAM location cannot start at less than 1000H',CR,LF,'$'
RAM$1$LOC       DB      CR,LF,'Data will be placed within the first 64K of RAM',CR,LF,'$'
INVALID$MSG     DB      CR,LF,'Invalid data. RAM lcoacton must be 1000H-FF000H',CR,LF,'$'



                DS      40      ;STACK AREA
STACK           DS      2       ;STACK POINTER
RECVD$SECT$NO   DB      0
SECTNO          DB      0       ;CURRENT SECTOR NUMBER
ERRCT           DB      0       ;ERROR COUNT
RAM$POINTER     DW      0       ;Store data starting here.
DISPLAY$POINTER DW      0       ;Pointer for ststus display (only)
```

```
SEG$POINTER     DB      0       ;Segment above 64K
WINDOW$FLAG     DB      0       ;Flag for > 64K destination
Digit5          DB      0
Digit4          DB      0


; END
```