```
BEGIN "SMITCH-0 downloader"

REQUIRE "<> <>" DELIMITERS;
DEFINE ! = <COMMENT>;

! ***********************************************************************
*                                                                      *
*                          SMITCH-0 8080 DOWNLOADER                    *
*                                                                      *
*                                 Reid G. Smith                        *
*                                                                      *
*    This program accepts a file assembled by the MICRO-SYMBOL 8080    *
* assembler and transfers it in records to a SMITCH-0 terminal via an  *
* asynchronous communications link. The format used by the downloader is *
* detailed in the procedure called "download". Records are acknowledged  *
* individually, and up to four attempts will be made to download records *
* that have been incorrectly received.                                 *
*                                                                      *
************************************************************************;

! The following macros are used by the downloader;

DEFINE crlf = <('15 & '12)>,
       colon = <'72>,
       ack = <'6>,
       nak = <'25>,
       eot = <'4>,
       !yes(s) = <s = "y" OR s = "Y">,
       yes!or!no(s) = <s = "y" OR s = "Y" OR s = "n" OR s = "N">;
^L
```

```
! *****************************************************************;

! The following are used when the downloader is called from the
  Micro-Symbol assembler;

EXTERNAL BOOLEAN  RPGSW;

EXTERNAL INTEGER INIACS;  ! used to store the accumulator contents;

! *****************************************************************;

! The following is a list of global variables used by the downloader;

INTEGER chan,    ! channel number for 8080 file;
        ch!ctrl, ! channel number for control file when called from
                   assembler;
        b!ctrl,  ! break table for control file;
        break!table, ! The break table used to read the .rel file;
        count,   ! The maximum number of characters per "input" on the
                    .rel file;
        brchar,  ! The break character for reading the .rel file;
        eof,     ! The end of file character for reading the .rel file;
        r#,      ! record number;
        rl,      ! Record length;
        lah,     ! high order load address byte;
        lal,     ! low order load address byte;
        lah0,    ! high order start address byte;
        lal0,    ! low order start address byte;
        ctrl,    ! control byte;
        ctrl0,   ! control byte for last record;
        cksm;    ! checksum for data bytes in a record;

STRING  s,       ! string used tor reading .rel file;
        ss;      ! Temporary string;

BOOLEAN gtjfn!flags, ! used to get channel for .rel file when downloader
                       is called from assembler;
        openf!flags, ! as above;
        success;     ! TRUE if file successfully closed;

INTEGER ARRAY data!byte[1:256];   ! data byte array - dimensioned to
                                    maximum record length;
^L
```

```
! ********************************************************************;

! Procedure "binary!mode" sets the terminal mode to binary, immediate or
  deferred echoing, lowercase input and output, and full wakeup control;

PROCEDURE binary!mode;

    BEGIN "binary!mode"
    DEFINE terminal = <'777777>; ! Designator for controlling terminal;
    DEFINE mode = <'046120774001>;  ! JFN mode word for data media;
    START!CODE
    MOVE 1,[terminal];
    MOVE 2,[mode];
    SFMOD;
    END;
    END "binary!mode";

! ********************************************************************;

! Procedure "hex!convert" converts a HEX digit represented as an ASCII
  string in the range '0-9', or 'A-F' to a four bit decimal representation.

Argument:
  s => the ASCII string;

INTEGER PROCEDURE hex!convert
    (STRING s);

    BEGIN "hex!convert"
    INTEGER i;
    IF s GEQ "0" AND s LEQ "9" THEN
        BEGIN "convert digit"
        i _ CVD(s);
        END "convert digit"
    ELSE IF s GEQ "a" AND s LEQ "F" THEN
        BEGIN "convert letter"
        i _ CVD(s - 17) + 10;
        END "convert letter"
    ELSE
        BEGIN
        OUTSTR("
***** ERROR IN 'HEX!CONVERT' - TRIED TO CONVERT ILLEGAL HEX DIGIT
***** OCTAL CODE = " & CVOS(s) & "
");
        i _ 0;
        END;
    RETURN(i);
    END "hex!convert";
^L
```

```
! *******************************************************************;

! Procedure "download" sets up a record and downloads it to the terminal.
  The download format is as follows:
            byte 1 - ^U
            byte 2 - ^D  These two characters specify the start of a
                         download
            byte 3 - low order byte of load address
            byte 4 - high order byte of load address
            byte 5 - low order byte of record length
            byte 6 - high order byte of record length
            byte 7 - control byte | 0 = load and continue
                                  | 255 => load and go
            byte 8 - checksum - this is the negative of the sum of the
                     data byte values mod 256. Add this value to the sum
                     of the received data byte values. If the result is
                     zero, then the data has been correctly received;

PROCEDURE download;

    BEGIN "download"
    INTEGER i;   ! index variable;
    PBOUT(nak);
    PBOUT(eot);
    PBOUT(lal);
    PBOUT(lah);
    PBOUT(rl);
    PBOUT(0);    ! current max record length is 255; PBOUT(ctrl);
    PBOUT(cksum);
    FOR i _ 1 STEP 1 UNTIL rl DO PBOUT(data!byte[i]);
    END "download";
^L
```

```
! ******************************************************************;

! Procedure "read!record" reads one record from the .rel file and stores
  the appropriate information in various global variables;

PROCEDURE read!record;

    BEGIN "read!record"
    INTEGER i;        ! index variable;
    s _ INPUT(chan,break!table);
    ! tne first two bytes constitute the record length (in bytes of data);
    rl _ (hex!convert(LOP(s)) ash 4) + hex!convert(LOP(s));
    ! the next four bytes constitute the load address;
    lah _ (hex!convert(LOP(s)) ASH 4) + hex!convert(LOP(s));
    lal _ (hex!convert(LOP(s)) ASH 4) + hex!convert(LOP(s));
    IF r# = 0 THEN
        BEGIN    ! Save start address of program for load and go;
        lah0 _ lah;
        lal0 _ lal;
        END;
    IF rl = 0 THEN
        BEGIN
        ctrl _ ctrl0; ! set control byte for last record;
        lah _ lah0;
        lal _ lal0; ! set up start address;
        END;
    ! The next two bytes constitute the record type (always zero).
      They are ignored;
    ss _ LOP(s);
    ss _ LOP(s);
    ! Translate the data part of the record;
    FOR i _ 1 STEP 1 UNTIL rl DO
        data!byte[i] _ (hex!convert(LOP(s)) ASH 4) + hex!convert(LOP(s));
    ! read the checksum for the complete record;
    cksm _ (hex!convert(LOP(s)) ASH 4) + hex!convert(LOP(s));
    ! ignore any remaining characters in record;
    END "read!record";
^L
```

```
! *********************************************************************;

! Procedure "download!record" downloads one record of the .rel file. It
  handles acknowledgement characters and retries on incorrectly
  acknowledged records.

  Value;
    The procedure returns TRUE if the download can be continued, or FALSE
    if the download must be terminated;

BOOLEAN PROCEDURE download!record;

    BEGIN "download!record"
    INTEGER retry,    ! retry counter;
            ack!char; ! acknowledgement character for download;
    retry _ 0;    ! set up the retry counter;
    DO  BEGIN "download one record"
        download; ! download the record;
        ! If the data is successfully downloaded, the terminal
          will respond with ^F (ACK). Otherwise it will respond
          with ^U (NAK). In this case an error message is
          displayed and another download of the record is
          attempted;
        ack!char _ PBIN;
        retry _ retry + 1;
        IF ack!char NEQ ack THEN
            BEGIN "download failed"
            IF ack!char NEQ ack THEN PRINT("
***** Failed to download record ",r#)
            ELSE PRINT("
***** Received incorrect acknowledgement on record ",r#);
            IF retry < 4 then PRINT(" - Trying again
")
            ELSE PRINT(" - Download Terminated
");
            END "download failed";
        END "download one record"
    UNTIL ack!char = ack OR retry GEQ 4;
    RETURN( IF ack!char NEQ ack AND retry GEQ 4 THEN FALSE ELSE TRUE );
    END "download!record";
^L
```

```
! *********************************************************************;

! Get name of .rel file (which contains the 8080 program to be
  downloaded);

! open the .rel file;
IF RPGSW THEN ! The loader has been called from the assembler;
    BEGIN
    ! open a channel to the control file, get the information, then
      delete the control file;
    ! open a channel to the .rel file;
    ch!ctrl _ OPENFILE("dload.file","R*");
    BREAKSET(b!ctrl _ GETBREAK,","  & '15,"IS");
    SETINPUT(ch!ctrl,200,brchar,eof);
    chan _ OPENFILE(INPUT(ch!ctrl,b!ctrl),"R*");
    ctrl0 _ CVD(INPUT(ch!ctrl,b!ctrl));
    CLOSF(ch!ctrl);
    DELF(ch!ctrl);
    OUTSTR("
Load and " & (IF ctrl0 = 0 THEN "continue" ELSE "go") & " from " &
    JFNS(chan,0) & crlf);
    CLRBUF;
    END
ELSE ! The loader has been called from the exec;
    BEGIN PRINT("
    File Name* ");
    chan _ OPENFILE(NULL,"RC*");
    END;

! Get ready to process the .rel file;

BREAKSET(break!table _ GETBREAK,colon,"IS"); ! set up to break on ":";
count - 600;   ! maximum number of characters per input;

SETINPUT(chan,count,brchar,eof);  ! set break characters and so on;

! Ask for load/go or load/continue specification;

LOAD/GO means that the SMITCH-0 will store the program in the
  appropriate locations and jump to the start address specified in
  the record that contains a load/go control byte.
LOAD/CONTINUE means that the SMITCH-0 will simply store the program
  in the appropriate locations and continue normal operation as a
  terminal.

A load/continue control byte will be downloaded with all records
except the zero length record which the assembler appends to the .rel
file. Then the appropriate control byte (as established here) will be
downloaded with this last record;
^L
```

```
! *********************************************************************;

IF NOT RPGSW THEN
    BEGIN
    DO  BEGIN
        PRINT("
Load and go, or load and continue (y or n): ");
        s _ INTTY;
        END
    UNTIL yes!or!no(<s>);
    ctrl0 _ ( IF !yes(<s>) THEN 255 ELSE 0 );
    END;

binary!mode;  ! Set the controlling terminal to binary mode (so that
                full 8 bit bytes can be transferred);

! read to the first break character - and ignore all before it;
s _ INPUT(cnan,break!table);

! Download the .rel file;

ctrl _ 0;     ! set control byte for load and continue;
r# _ 0;       ! initialize the record index;

DO  BEGIN "process one record"
    read!record;  ! read one record;
    IF NOT download!record THEN DONE
    ELSE r# _ r# + 1;
    END "process one record"
UNTIL rl = 0;  ! Terminate on a record with zero length;

success _ CFILE(chan); ! close the .rel file;

END "SMITCH-0 downloader";
```