Embedded System Design
Intel Hex Records and Motorola S-Records

Since the early days of computer engineering, there has been a need for transferring binary data between devices, with some assurance that data corruption did not occur during the transfer. Two widely used formats for accomplishing this transfer of data are the Intel Hex Record and the Motorola S-Record. Each of these formats represents all of the data as ASCII characters, so that these records can be printed, modified or created with a simple text editor, viewed on an ASCII terminal, sent across phone lines or serial ports without binary protocols, etc. For maximum portability, hex records should contain a maximum of 16 or 32 data bytes per line. Note: The compiler and assembler tools used in this class support a maximum of two address bytes, corresponding to a maximum of 64KB of data per file.

Intel Hex Record Format

Data Record: `':nnaaaattdddddddddddddddddddddddddddddddd...cc<cr><lf>'`

Field	Contents
:	ASCII colon indicates start of data record.
nn	Count of number of data bytes in record. (in ASCII/HEX)
aaaa	Load address of data record. (in ASCII/HEX)
tt	Record type. (00=Data, 01=End of file)
dd	Actual data bytes in record (two ASCII characters per data byte). (in ASCII/HEX)
cc	Checksum of count, address, and data. (in ASCII/HEX)
<cr>	ASCII carriage return character. (0D hex, 13 decimal)
<lf>	ASCII line feed character. (0A hex, 10 decimal)

Note 1: The checksum is computed as the two's complement of the eight bit sum of all values from 'nn' to the end of data (last 'dd' byte).

Note 2: The end of file record contains a count of 00 and address of 0000h.

Motorola S-Record Format

Data Record: `'Stnnaaaadddddddddddddddddddddddddddddddd...cc<cr><lf>'`

Field	Contents
S	ASCII 'S' indicates start of data record.
t	Record type, '1','2','3'=data, '9'=end of file. (DDS software only supports types '1' and '9')
nn	Count of number of bytes in record (including address and checksum). (in ASCII/HEX)
aaaa	Load address of data record. (in ASCII/HEX)
dd	Actual data bytes in record (two ASCII characters per data byte). (in ASCII/HEX)
cc	Checksum of count, address, and data. (in ASCII/HEX)
<cr>	ASCII carriage return character. (0D hex, 13 decimal)
<lf>	ASCII line feed character. (0A hex, 10 decimal)

Note 1: The checksum is computed as the one's complement of the eight bit sum of all values from 'nn' to the end of data (last 'dd' byte).

Note 2: Count 'nn' is three greater then the number of data bytes in the record, since two bytes of address and one byte of checksum are included in the count.

Note 3: The end of file record contains a count of 3 and address of 0000h.

(continued)

Example

Contents of the source file test.asm.

```
; File: test.asm

        ORG $0000       ; initialize the reset vector at address 0000h
        AJMP *+$0032

        ORG $0032       ; start the main code at address 0032h
start   NOP
        CPL P1.0        ; toggle bit 0 on Port 1
        LJMP start
```

Contents of the listing file test.lst after executing 'asm51 test -F'.

```
DUNFIELD 8051 ASSEMBLER: test                                         PAGE: 1


0000                         1  ; File: test.asm
0000                         2
0000                         3          ORG $0000       ; initialize the reset vector at address 0000h
0000  01 32                  4          AJMP *+$0032
0002                         5
0032                         6          ORG $0032       ; start the main code at address 0032h
0032  00                     7  start   NOP
0033  B2 90                  8          CPL P1.0        ; toggle bit 0 on Port 1
0034  02 00 32               9          LJMP start
```

Contents of the Motorola S-Record file test.hex after executing 'asm51 test -F'.

```
S10500000132C7
S109003200B2900200324E
S9030000FC
```

Contents of the Intel Hex Record file test.hex after executing 'asm51 test -F -I'.

```
:020000000132CB
:0600320000B29002003252
:00000001FF
```

Examine the contents of the Motorola S-Record and Intel Hex files above and be able to identify each field in each record. Verify the checksums for the lines in each file. Note that each byte of machine code translates into two bytes of ASCII characters in the hex record file (i.e., each hex nybble is represented as one ASCII byte). Note that the carriage return and line feed data for each hex record are not displayed.