MITSCNVT is a program which operates under CP/M and enables the user to list the directory and convert MITS ALTAIR (TM Pertec Computer Corp) disk files to CP/M files. The reason conversion is required is that the disk formats are different and that the ALTAIR DISK is a hard sectored disk which contains 32 sectors of 137 bytes each per track. A CP/M compatable disk is soft sectored and is composed of 28 sectors of 128 or 256 bytes each sector.

The ALTAIR disk is capable of containing a 255 entry directory on each disk and each directory entry "points" to a threaded list of sectors. In order to ensure the integrity of the threaded list, each sector in the list contains a single byte which is the "file number" - an integer calculated from the locus of the directory entry. If, during operations on the chain, the file number changes a 'file link error' is given and the program aborts. The calculation for the file number is given erroneously in the MITS documentation and is corrected here:

```
      8*SECTOR+(SLOT+1)
```

Where SECTOR is the directory sector number (0-31) in which the file name is found and SLOT is the group number of a 16 byte wide directory entry of the format:

Bytes	Description
00-07	ASCII File Name - if the first character is a zero (binary zero) the entry is ignored. If it is 0ff (all one bits) this is the last entry in the directory. An ignored entry is the result of deleting a file in the MITS directory.
08	Track address of the first sector in the file. Track 0-5 are never used as they are 'system' tracks. Track 70 is the directory track.
09	Sector number of the first sector of the file. Sectors are always 0,8,16 or 24. Sectors are allocated in groups of eight. There is no pattern for it, but, because of latency consid- erations odd numbered sectors are 16+sector number AND 1Fh away from the even sector in a logical sense. My calculations indicate this is a poor choice and if you have ever heard ALTAIR Disk Extended Basic or DOS run, you'll know its not too good.
10	Flag to indicate file organization - 4=random 2=sequential
11-15	Wasted bytes NOTE: at one time or another the program MITSCNVT uses these bytes to store some information about each file -not on the disk but in memory.

As an example let me use the file &OCTBIN for illustration purposes. It - quite arbitrarily - will be assumed to start in track 72 sector 0. This would be its directory entry if it were a sequential file:

```
__________________________________________________________
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |....
----------------------------------------------------------
  &   O   C   T   B   I   N   b  72

----------------------------------------------------------
| 10 | 11 | 12 | 13 | 14 | 15 | 16 | byte number
----------------------------------------------------------
  0   2   unused bytes------------>
```

If this were the 5th entry in directory sector zero it would file number 5.

Moving along in the example each data file begins with a format in which the data are embedded.

The format is as follows:

Byte	Description (For Sequential files)
00	Track number +80h. This high order bit must be set for every track number. The physical track is identified in every sector on that track in this manner.
01	Physical sector number of each sector. Every sector is identified this way. Sectors are numbered 0-31.
02	File number. If =0 then this sector is NOT in use at this time. (When a file is deleted you have to wait for each sector in the chain to be rewritten so that the file number can be set to zero).
03	Number of data bytes written in this sector
04	Checksum of all data in the sector except bytes 0,1,4 and 136. Byte 135 is also not used in the checksum calculation. The checksum is the sum of all of the data without regard to overlow as done in an eight bit register.
05	Track number of the NEXT data sector in the chain.
06	Sector number of the next data sector in the chain. If track and sector number are both zero there is no more file (EOF).
07-134	128 words of data. If the first byte is ffh then the

file is binary. If the first byte is other than ffh then the file is ASCII. The first byte is the first byte of the first sector of the file - not the first byte of any sector.

135 Check byte - always ffH. This is used to ensure that everything else in the sector was in the right place.

Each of the bytes in these sectors are linked to the other sectors in a "forward" direction until bytes 5+6 are zero. This signals that the current data block is the last block in the file. The directory sectors are located in track 70. Each directory sector begins in the 8th byte of the physical sector (the first 7 bytes are garbage).

USE OF THE PROGRAM:

The program accepts two types of input. The first type of input is a disk number prefixed by an equals sign. This will result in recovery of the MITS directory on that disk and the listing of that directory. As an example:

```
=5
```

will result in selection of disk 5 and the recovery of the directories on that disk which will be listed.

The second type of input is a CP/M file name followed by a file type, an equal sign and a MITS file name and disk number. This will search the directory on the specified MITS disk, locate the file and write it on the CP/M disk specified. As examples:

```
 (1) CPMFILE.MIT=ZCTABLE,0
 (2) DIET.BAS=DIET,0
 (3) OCTOCT.ASM=&OCTOCT,2
```

In example (1) the file 'ZCTABLE' located on MITS disk zero is copied onto the CP/M disk as file CPMFILE.MIT. In the next example, (2), file DIET on disk zero is copied to CP/M file DIET.BAS. In the last example, MITS file &OCTOCT on MITS disk two is copied to CP/M file OCTOCT.ASM. Thus the MITS files are always on the right side of the equals sign and the CP/M files are on the left hand side. Note that an ambiguous file name is not permitted because the CP/M files are open for writing.

Because the files are accepted in list form, the program may be used as a submit job. Problems with this program may be submitted in writing to the author.