U S E R   M A N U A L

f o r

BASIC/Machine Language UUDECODER Version 1.2a

Manual Revision Date: 5/20/87

by John H. Sangster
jhs@mitre-bedford.arpa
(617) 235-8753 (home)
(617) 271-2000 (work)

1. PURPOSE:#

Uudecode is the decoder for uuencode, a program widely used on unix systems to encode binary files into printing ASCII characters for transmission over networks. Because many network components interpret and respond to special "control" characters, attempts to send binary files such as machine-language programs over a network in raw, un-encoded form are usually doomed to failure. Uuencode has become popular for encoding because it is fairly efficient: it encodes each group of three 8-bit bytes into only four bytes of printable characters. Therefore, using uuencode, a file is expanded by a factor of only 1.333 to one as the price of constraining the character set to be limited to printing characters only.

2. USING UUDECODE VER. 1.2a:#

UUDECODE VER. 1.2a is normally sent by e-mail as an Atari BASIC ".LST" file. This means that it consists of only printing characters itself, and can be mailed, printed, etc. without difficulty. On the other hand, the first time you run it, you will have to "ENTER" it into BASIC with a command like 'ENTER "D:UUDECODE.LST"<RETURN>'. Then you should SAVE it with the command 'SAVE "D:UUDECODE.BAS"<RETURN>'. Thereafter, you can run it under BASIC with the RUN command instead of the ENTER command. You will immediately notice that RUN works significantly faster. This is true because SAVED files are in tokenized form. You can delete the .LST file if you wish, because it can always be reconstructed using the LIST command, but you should be sure to keep a backup copy of UUDECODE in some form on a separate disk from the one you normally run it from. Keeping the .LST file around is handy if you decide to e-mail it to someone else someday.

Once you have UUDECODE running, it will ask you for the input file you wish to uudecode and the filename you want the output to be sent to. You should give it the exact input file specification you want it to use, i.e. including device, filename, and extension. Normally, uuencoded files are given the extenion ".UUE", but you will have to tell the program that. On the output file specification, however, the program will accept either a null input (just a <RETURN>) or a device specifier like D1: without a filename, or you can again specify the full filename and extension. In the first two cases, it will attempt to read the filename from the "begin" line of the uuencoded input file. If you don't even specify a device name, it will assume "D1:" and will so inform you. If it cannot successfully OPEN the file it thinks you want, it will give you another chance to enter the output filename. Before it gets to this point, however, it will have printed the "begin" line if one was found, so you may have a clue as to what went wrong, e.g. a filename was given which is not valid for your DOS. In that case, you should think up a valid filename that seems suitable.

After finishing a file, UUDECODE will print "Done!" and will ask if you want to decode another file. If you say "Y" or "yes", it will re-initialize and allow you to enter the new input and output filenames. Otherwise, it will exit to DOS. NOTE: UUDECODE Ver. 1.2a also prints a byte count when it finishes. You should note this count and compare it with the correct byte count as established when the file was being uuencoded.

That's about all there is to it. Comments on this manual, as well as on any difficulties experienced with UUDECODE, should be directed to the author.

3. WHAT UUENCODE AND UUDECODE REALLY DO TO THE DATA:#

The uuencoding process, used to create ASCII files of the type that UUDECODE is designed to decode, is easy to describe. Each group of three input bytes, with any 8-bit pattern whatever in them, is broken up into four sets of six bits. (Note that three times 8 and four times 6 each give exactly 24 bits, so no information is lost.) Each 6-bit pattern is put in the low-order 6 bits of an 8-bit byte, and decimal 32 is added to give a final value in the range 32 (ASCII blank) through 95. All the characters in this range (decimal 32 through 95) are printing characters.

Traditionally, uuencode programs take 45 bytes at a time from the input file as long as bytes are available, and encode them into 60 output bytes which are sent as one "line". Each line is made up as an encoded byte count, the 60 encoded bytes of data, and an end-of-line character or characters. The encoded byte count is the actual number of input bytes encoded on that line, plus decimal 32. For all lines but the last, this gives 45 plus 32, or 77, which translates into ASCII uppercase "M". That is why all lines but the last in a uuencoded file begin with "M". The final line of data begins with a character which is between "blank" (32) and "M" (77) in the ASCII "collating sequence". The exact value depends on how many bytes were left in the input file after the last full line of 45 was used up.

Uuencode programs usually sandwich these lines of data between a "begin" line and an "end" sequence. The "begin" line consists of the word "begin", a single space, a 3-digit "protection code" as used by unix systems, another single space, and finally the filename which should be used for the file into which the data is decoded at the far end. The "end" sequence is supposed to consist of a line containing a blank in the first character position, denoting zero encoded bytes on that line, followed by a line beginning with the word "end". Some unix-based uudecoders seem to require additional blanks following these minimum fields.

Uudecode is supposed to be the exact inverse of uuencode, i.e. after decoding a uuencoded file, you should have the exact binary file that was originally encoded. This is essential, because the whole purpose of uuencode and uudecode is to let you transmit machine language object programs around on networks. If even one bit is changed, all bets are off! The basic idea of uudecode is therefore to take each line in, subtract 32 from each of the 60 or fewer bytes, pack each group of 6 bits in the low-order portion of each byte back into 8-bit packed binary form, and write the re-packed bytes out to a binary output file.

4. PITFALLS:#

Unfortunately, as uuencode has been adopted wholesale for use in transmitting binary files across networks, it has turned out that not all network hosts are as careful what they do with files as are most unix hosts. IBM hosts are among the most notoriously callous about changing byte values to suit their own preconceived notions. Most of the time, the changes consist of things like stripping off trailing blanks on lines that happen to end in a blank. This can be embarrassing if not handled properly by the decoder. Another favorite trick is to change carat into accent grave, or tilde into carat, or what-have-you. The only reliable way to handle this sort of problem seems to be for the uuencode program to send an encoding translation table at the beginning, which lists all the output characters from decimal 32 to 95, and for the uudecode program to capture the values received in their place and decode accordingly. If they have not been mapped one-to-one, of course it can only throw up its hands in dismay and so inform you.

5. CAPABILITIES OF UUDECODE VER. 1.2a:#

UUDECODE Ver. 1.2a as described by this manual includes masking to correct the most common character translation problems, i.e. those which result in encoded values greater than 95. It correctly decodes files in which a "sentinel" character (usually "a" or "x") has been added to the end of the uuencoded lines to prevent stripping of trailing blanks, as well as files in which trailing blanks have actually been stripped! It does NOT process the "translation table" preamble added by some uuencodes; this may be added in a future revision.

This version also is capable of reading the output filename from the uuencoded file. This will be done if you specify only a DEVICE NAME, e.g. "D1:", or respond with just a <RETURN> to the output file prompt.

Finally, Ver. 1.2a is smart enough to ignore extraneous lines in the input file either before the "begin" line or after the "end" line. This means that it can be used to decode uuencoded files which are preceded by explanatory comments in an e-mail message. On the other hand, this version can only decode ONE uuencoded file per input file. If you receive a message containing multiple files, you will have to break up the input file into separate files for uudecoding. A future release may allow more flexibility.

UUDECODE Ver. 1.2a is fairly fast. An assembly-language subroutine is used to do the "bit-picking" dirty work. This routine is quite efficient, despite the fact that it includes such conveniences as checking the DIMensioning of the string used as the output buffer and setting its LENgth parameter correctly. These features cost a small amount in assembly language but they save the BASIC calling program from having to worry about such details, which would be far more costly to implement in BASIC. The main program of UUDECODE.LST is also optimized for speed, mainly by putting the inner loop "up front", where BASIC doesn't have to search very far for statement numbers, and by keeping the loop short, especially on the most frequently used path. To give you an idea of just HOW fast this program is, a fairly knowledgeable programmer coded uudecode up in C in an effort to get better performance and then noted afterward that Ver. 1.2a in BASIC and assembly language ran approximately FIVE TIMES as fast as his C version! I think you will find Ver. 1.2a highly satisfactory with regard to execution speed.

```
1 GOTO 100:REM jump around time-critical stuff to start up program.
2 M=USR(UUDADR)
10 IF M=0 THEN 2090
20 IF M>OBUFDIM THEN ? "Error at line 20, M=";M:GOTO 3000
30 PRINT #2;OBUF$;:BYTES=BYTES+M
40 INPUT #1,IBUF$:L=LEN(IBUF$)+1:IF L<62 THEN IBUF$(L)=" ":IBUF$(L+1)=IBUF$(L)
50 GOTO 2
100 ? "Uudecode Ver. 1.2a":? "Report errors to John Sangster at"
101 ? "(617) 235-8753/jhs@mitre-bedford.arpa":?
102 POKE 6,1:REM Turn BASIC flag ON.
105 DIM OBUF$(80),IBUF$(62),OFILE$(16),IFILE$(16),A$(1)
110 DIM UUDECODE$(400)
120 UUDADR=ADR(UUDECODE$):IBUF=ADR(IBUF$):OBUF=ADR(OBUF$)
130 OBUFDIM=80:UUDDIM=400:BEEP=150:RETRY=500
140 GOTO 200
149 REM BEEP Subroutine:
150 SOUND 0,85,10,15:FOR I=1 TO 80:NEXT I:SOUND 0,0,0,0:RETURN
199 END
200 ? "Loading uudecode subroutine..."
201 RESTORE 4000:POKADR=UUDADR:MAXADR=POKADR+UUDDIM-1:PRGTOP=UUDADR-1
202 READ X:IF X=255 THEN READ X:IF X=255 THEN 204
203 ? "BAD LOAD FILE FOR UUD":END
204 READ LO1,HI1,LO2,HI2:BYTES=HI2*256+LO2-(HI1*256+LO1)+1:PRGTOP=PRGTOP+BYTES
```

```
205 IF BYTES<0 OR BYTES>UUDDIM THEN ? "BYTE COUNT ERROR FOR UUD":END
206 FOR I=1 TO BYTES:READ X:POKE POKADR,X:POKADR=POKADR+1:IF POKADR>MAXADR THEN ? "UUD
207 NEXT I
208 TRAP 209:READ LO1,HI1,LO2,HI2:BYTES=256*HI2+LO2-(256*HI1+LO1)+1
209 PRGTOP=PRGTOP+BYTES:IF LO1<>224 OR HI1<>2 THEN 206
500 REM COMMAND DISPATCHER
530 ? :? "INPUT FILE";:INPUT IFILE$:IF LEN(IFILE$)=0 THEN 530
540 ? :? "OUTPUT FILE SPEC OR":? "DEFAULT DEVICE ID";:INPUT OFILE$
550 L=LEN(OFILE$):IF L=0 THEN OFILE$="D1:":? :? "Output to D1: assumed.":?
2000 ? "Beginning uudecode processing..."
2011 IBUF$(1,1)=" ":M=USR(UUDADR,IBUF,OBUF):IF M<>0 THEN ? "Error in initializing UUDE
2019 TRAP 2020:CLOSE #1:OPEN #1,4,0,IFILE$:GOTO 2030
2020 ? "INPUT FILE NOT FOUND":GOSUB BEEP:GOTO RETRY
2030 INPUT #1,IBUF$:BYTES=0
2035 L=LEN(IBUF$):IF L>5 THEN L=5
2036 IF L=0 THEN L=1:IBUF$=" "
2040 TRAP 2095:IF IBUF$(1,L)<>"begin" THEN 2030:REM skip header
2042 ? IBUF$:REM Print "begin" line to screen & get OFILE$ if default case.
2043 L=LEN(OFILE$):IF L<=0 OR OFILE$(L)=":" THEN 2046
2044 TRAP 2050:CLOSE #2:OPEN #2,8,0,OFILE$:TRAP 2095:GOTO 40
2046 L=LEN(IBUF$):FOR I=L TO 1 STEP -1
2047 IF IBUF$(I,I)<>" " THEN 2049
2048 OFILE$(4)=IBUF$(I+1,L):GOTO 2044
2049 NEXT I
2050 ? "OUTPUT FILESPEC":INPUT OFILE$:IF LEN(OFILE$)=0 THEN 2050
2052 GOTO 2044
2090 INPUT #1,IBUF$:IF IBUF$(1,3)="end" THEN PRINT IBUF$:GOTO 3000
2095 NERR=PEEK(195):IF NERR=136 THEN PRINT "EOF unexpected!":GOTO 3000
2097 ? "Error Code=";NERR
3000 ? "Done!":CLOSE #1:CLOSE #2:? "Output byte count = ";BYTES:? :?
3010 ? "More files to decode (Y/N)";:INPUT A$:IF A$="Y" OR A$="y" THEN 500
3020 DOS
4000 DATA 255,255,8,6,3,7
4010 DATA 104,240,67,170,56,233
4020 DATA 2,240,14,104,104,202
4030 DATA 208,251,169,254,133,212
4040 DATA 169,255,133,213,96,104
4050 DATA 141,1,6,104,141,0
4060 DATA 6,104,141,3,6,104
4070 DATA 141,2,6,165,6,240
4080 DATA 100,165,134,133,203,165
4090 DATA 135,133,204,173,2,6
4100 DATA 56,229,140,133,208,173
4110 DATA 3,6,229,141,133,209
4120 DATA 162,128,208,2,240,72
4130 DATA 160,0,177,203,201,129
4140 DATA 208,42,160,2,177,203
4150 DATA 56,229,208,208,33,200
4160 DATA 177,203,229,209,208,26
4170 DATA 165,203,141,6,6,165
4180 DATA 204,141,7,6,160,6
4190 DATA 177,203,141,4,6,200
4200 DATA 177,203,141,5,6,24
4210 DATA 144,22,24,165,203,105
4220 DATA 8,133,203,144,2,230
4230 DATA 204,202,208,192,169,255
4240 DATA 133,212,133,213,96,234
4250 DATA 173,0,6,133,204,173
4260 DATA 1,6,133,205,160,0
```

```
4270 DATA 132,213,177,204,56,233
4280 DATA 32,41,63,133,212,208
4290 DATA 24,165,6,240,19,173
4300 DATA 6,6,133,208,173,7
4310 DATA 6,133,209,160,4,169
4320 DATA 0,145,208,200,145,208
4330 DATA 96,165,6,240,43,173
4340 DATA 4,6,56,229,212,173
4350 DATA 5,6,229,213,16,9
4360 DATA 169,255,133,212,133,213
4370 DATA 32,54,185,173,6,6
4380 DATA 133,208,173,7,6,133
4390 DATA 209,169,0,160,5,145
4400 DATA 208,136,165,212,145,208
4410 DATA 230,204,208,2,230,205
4420 DATA 173,2,6,133,206,173
4430 DATA 4,7,108,7,3,6
4440 DATA 133,207,166,212,160,1
4450 DATA 177,204,56,233,32,133
4460 DATA 203,6,203,6,203,136
4470 DATA 177,204,56,233,32,6
4480 DATA 203,42,6,203,42,145
4490 DATA 206,202,240,68,169,0
4500 DATA 133,208,160,2,177,204
4510 DATA 56,233,32,41,63,74
4520 DATA 102,208,74,102,208,5
4530 DATA 203,136,145,206,202,240
4540 DATA 41,160,3,177,204,56
4550 DATA 233,32,41,63,5,208
4560 DATA 136,145,206,202,240,24
4570 DATA 24,165,204,105,4,133
4580 DATA 204,144,2,230,205,24
4590 DATA 165,206,105,3,133,206
4600 DATA 144,162,230,207,176,158
4610 DATA 96,224,2,225,2,0
```