A PROGRAM TO TRANSLATE PUNCHED PAPER TAPES

Paul P. Nicole

A computer program is described that enables a user of the ANL IBM-370/195 computer system to process data punched in standard ASCII code on a paper tape as though they were punched on standard IBM cards.

Introduction

Paper tape is usually read into the ANL computer system through a remote access data station (RADS station). The RADS station reads the hole pattern in the tape and produces a corresponding bit pattern in storage. This is done one character (row of holes on tape) at a time until 79 characters have been read. An 80-character card image made up of hexidecimal (hex) 00 followed by the 79 characters is then assembled. These card images are sent to the computer to be processed.[1]

A translation is necessary before the data read by the tape reader can be processed in a normal way. Even after translation the way in which the characters are arranged on the card images may make them hard to read. Multi-digit numbers, for example, may be split between two card images. There may also be characters that must be removed or handled in a special way, such as carriage returns, line feeds, and other control characters.

A new program translates ASCII (American Standard Code for Information Interchange) coded tape, does some minor editing, reformats the resulting data, and then writes a card image dataset, which may be punched as a deck of cards, read directly by another program, or listed on the printer. The method used here could be easily adapted to translate codes other than ASCII as long as there is a one-to-one correspondence between input characters on the tape and output characters. Also, the output records could easily be changed from 80 characters (card images) to either longer or shorter records by making some minor changes in the program.

Theory

In this discussion, paper tape patterns will be represented by combinations of X's, O's and periods. An X represents an unpunched hole, an O represents a punched hole, and a period represents a sprocket hole. We will deal only with 8-level paper tape, which has room for 8 X's or O's, and 1 period per line (character). In this scheme, XXXXX.XXX would represent a line with no holes punched (blank tape with only a sprocket hole), and OOOOO.OOO would represent a line with all holes punched. Columns will be numbered from right to left. So, XOXOX.XXO has punches in columns 1, 5, and 7. Also, all numbers preceded by a Z will be hex (base 16) numbers.

When the computer reads the tape, it produces a binary bit pattern that is the same as the pattern punched in the tape with all punches (O's) becoming 1's and all unpunched positions (X's) becoming 0's. The result is a binary number in storage with a value equal to the value of the punches on the tape if we count column 1 as 1 ($=2^0$), column 2 as 2 ($=2^1$), up to column 8 as 128 ($= 2^7$). Thus, the letter R, which is XOXOX.XOX on the tape, shows up in storage as 01010010. The computer displays its storage 8 binary bits (1 byte) at a time as 2 hex digits. The first four bits of the R (0101) become Z5, while the remaining bits (0010) become Z2, resulting in Z52 being displayed. In order to print or punch the character R, the computer must have a ZD9 in storage.[2] The translation requires that each Z52 be replaced by a ZD9, and that a corresponding replacement be made for each of the other printable characters.

This replacement is accomplished by having the original value in storage select the proper element of an array (L) (see program listing in Figure 1) that contains the needed replacement value (see Figure 2). First the original value is read, using FORMAT(1X,79A1), into array NIN, which is a 79-element INTEGER*2 array. This gets rid of the Z00, which is always the first character of card images generated when tape is read, and makes the other 79 characters INTEGER*2 variables. INTEGER*2 variables take up 2 bytes (8 bits/byte), while our original characters took up only 1 byte of storage. The unused bytes are filled by the computer with Z40's. The letter R then becomes Z5240. We

```
C     PRØGRAM TØ READ PAPERTAPE TØ CARD IMAGE FILE
      IMPLICIT INTEGER*2(N,L)
      DIMENSIØN NIN(79),NØUT(80),L(128)
      DATA LF/Z4040/,NP/Z8000/
      DATA L/31*':',' ','!','"','#','$','%','&',Z7D40,'(',')','*',
     1 '+',',','-','.',Z6140,'0','1','2','3','4','5','6','7','8',
     2 '9',':',';','<','=','>','?','@','A','B','C','D','E','F',
     3 'G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U',
     4 'V','W','X','Y','Z','[','\',']','↑','←',33*':'/
      IFLAG=0
      NI=0
      READ(50,102)LIG1,LIG2,LIG3,LIG4,LDL1,LDL2,LDL3,LDL4,
     1 LSP1,LZP1,LSP2,LZP2,LSP3,LZP3,LCØL,LREC
  102 FØRMAT(8(1X,1Z4),8I5)
      L(LSP1)=L(LZP1)
      L(LSP2)=L(LZP2)
      L(LSP3)=L(LZP3)
   10 READ(50,100,END=999) NIN
  100 FØRMAT(1X,79A1)
      DØ 11 N=1,79
      IF(NIN(N) .LT. 0) NIN(N)=NIN(N)-NP
      IF(NIN(N) .EQ. LDL1) GØ TØ 200
      IF(NIN(N) .EQ. LDL2) GØ TØ 200
      IF(NIN(N) .EQ. LDL3) GØ TØ 200
      IF(NIN(N) .EQ. LDL4) GØ TØ 200
      IF(NIN(N) .EQ.   64) GØ TØ 11
      IF(NIN(N) .EQ. LIG1) GØ TØ 11
      IF(NIN(N) .EQ. LIG2) GØ TØ 11
      IF(NIN(N) .EQ. LIG3) GØ TØ 11
      IF(NIN(N) .EQ. LIG4) GØ TØ 11
      IF(IFLAG .EQ. 1) GØ TØ 201
   12 NI=NI+1
      NJ=NIN(N)/256
      NØUT(NI)=L(NJ)
      IF(NØUT(NI) .EQ. L(58)) GØ TØ 206
   13 CØNTINUE
      IF(NI .EQ. LREC) GØ TØ 205
   11 CONTINUE
      GØ TØ 10
  101 FØRMAT(80A1)
  200 IFLAG=1
      GØ TØ 11
  201 IF(NI .EQ. 0) GØ TØ 203
      IF(NI .EQ. 80) GØ TØ 202
      NB=NI+1
      DØ 202 I=NB,80
      NØUT(I)=LF
  202 CØNTINUE
      WRITE(60,101) NØUT
  203 IFLAG=0
      NI=0
      GØ TØ 12
  205 WRITE(60,101) NØUT
      NI=0
      GØ TØ 11
  206 IF(LCØL .EQ. 0) NI=NI-1
      IF(LCØL .EQ. 2) NØUT(NI)=L(32)
      GØ TØ 13
  999 NB=NI+1
      IF(NI .EQ. 0) GØ TØ 99
      IF(NI .EQ. 80) GØ TØ 901
      DØ 902 I=NB,80
      NØUT(I)=LF
  902 CØNTINUE
  901 WRITE(60,101)NØUT
   99 STØP
      END
```

FIG. 1.--Program listing

IN	NJ	L(NJ)	OUT	IN	NJ	L(NJ)	OUT	IN	NJ	L(NJ)	OUT
0140	01	7A40	:	2C40	44	6B40	,	5740	87	E640	W
0240	02	7A40	:	2D40	45	6040	-	5840	88	E740	X
0340	03	7A40	:	2E40	46	4B40	.	5940	89	E840	Y
0440	04	7A40	:	2F40	47	6140	/	5A40	90	E940	Z
0540	05	7A40	:	3040	48	F040	0	5B40	91	7940	[
0640	06	7A40	:	3140	49	F140	1	5C40	92	CF40	\
0740	07	7A40	:	3240	50	F240	2	5D40	93	4940	]
0840	08	7A40	:	3340	51	F340	3	5E40	94	B040	↑
0940	09	7A40	:	3440	52	F440	4	5F40	95	6D40	←
0A40	10	7A40	: (a)	3540	53	F540	5	6040	96	7A40	:
0B40	11	7A40	:	3640	54	F640	6	6140	97	7A40	:
0C40	12	7A40	:	3740	55	F740	7	6240	98	7A40	:
0D40	13	7A40	:(b)	3840	56	F840	8	6340	99	7A40	:
0E40	14	7A40	:	3940	57	F940	9	6440	100	7A40	:
0F40	15	7A40	:	3A40	58	7A40	:	6540	101	7A40	:
1040	16	7A40	:	3B40	59	5E40	;	6640	102	7A40	:
1140	17	7A40	:	3C40	60	4C40	<	6740	103	7A40	:
1240	18	7A40	:	3D40	61	7E40	=	6840	104	7A40	:
1340	19	7A40	:	3E40	62	6E40	>	6940	105	7A40	:
1440	20	7A40	:	3F40	63	6F40	?	6A40	106	7A40	:
1540	21	7A40	:	4040	64	7C40	@	6B40	107	7A40	:
1640	22	7A40	:	4140	65	C140	A	6C40	108	7A40	:
1740	23	7A40	:	4240	66	C240	B	6D40	109	7A40	:
1840	24	7A40	:	4340	67	C340	C	6E40	110	7A40	:
1940	25	7A40	:	4440	68	C440	D	6F40	111	7A40	:
1A40	26	7A40	:	4540	69	C540	E	7040	112	7A40	:
1B40	27	7A40	:	4640	70	C640	F	7140	113	7A40	:
1C40	28	7A40	:	4740	71	C740	G	7240	114	7A40	:
1D40	29	7A40	:	4840	72	C840	H	7340	115	7A40	:
1E40	30	7A40	:	4940	73	C940	I	7440	116	7A40	:
1F40	31	7A40	:	4A40	74	D140	J	7540	117	7A40	:
2040	32	4040	SP	4B40	75	D240	K	7640	118	7A40	:
2140	33	5A40	!	4C40	76	D340	L	7740	119	7A40	:
2240	34	7F40	"	4D40	77	D440	M	7840	120	7A40	:
2340	35	7B40	#	4E40	78	D540	N	7940	121	7A40	:
2440	36	5B40	$	4F40	79	D640	Ø	7A40	122	7A40	:
2540	37	6C40	%	5040	80	D740	P	7B40	123	7A40	:
2640	38	5040	&	5140	81	D840	Q	7C40	124	7A40	:
2740	39	7D40	'	5240	82	D940	R	7D40	125	7A40	:
2840	40	4D40	(	5340	83	E240	S	7E40	126	7A40	:
2940	41	5D40	)	5440	84	E340	T	7F40	127	7A40	:
2A40	42	5C40	*	5540	85	E440	U				
2B40	43	4E40	+	5640	86	E540	V				

(a) LINE FEED
(b) CARRIAGE RETURN

FIG. 2.--Translation table

also need Z40 fillers in our output character list. The output R is then ZD940.

Next, the input is checked to see if the parity hole (column 8) was punched. Depending on the tape punching system, this hole may or may not be punched for any given character. Our R with the parity hole punched is ZD240 instead of Z5240. The bit generated by the parity punch makes the value of our input negative when looked at by the computer as an integer. To ignore the parity punch we check our input to see if it is a negative integer, and, if it is, subtract Z8000. If positive, the number is left as is. The value is then divided by 256 (Z100) to get rid of the Z40 filler. Our R is now Z0052, or, as a deciman integer, 82 (= 5 × 16 + 2). We make element 82 of array L contain ZD940 (R) so that each R input on tape will give an R output character. This same procedure is followed in selecting the other elements of array L. The values in array L are initialized by a data statement at the beginning of the program. Note that any value could be put into a particular element of L to cause a different output character to result from a given input character. The program allows for three such nonstandard translations.

After ignoring the parity punch, but before translation, the input characters are searched for special characters which have been designated either as delimiters or as characters to be ignored. If a delimiter is encountered, a flag is set. When the next translatable character or an end of file is read, the remainder of the 80-character record being assembled is filled with blanks, and the record is written to the output device. If a character to be ignored is encountered, the next character is read. Unpunched tape (Z0040) is ignored.

Use

The program reads one card which specifies all the changeable parameters in the program. This card is read according to FORMAT (8(1X,1Z4),8I5)— that is, 16 5-column fields. The first 8 fields are read as hex variables, while the last 8 are read as integer variables. The card is read into variables: L1G1, L1G2, L1G3, L1G4, LDL1, LDL2, LDL3, LDL4, LSP1, LZP1, LSP2, LZP2, LSP3, LZP3, LCOL, LREC. Variables L1G1 to 4 are hex representations of input characters to be ignored. Blank tape (Z0040) is automatically ignored.

If no characters are to be ignored, put b0040 into L1G1 to 4. For example, if all Q's are to be ignored, put b5140 into L1G1 (see Figure 2), where b is a blank.

LDL1 to 4 are hex representations of input characters which are to be used as delimiters. The delimiters trigger the start of a new record when the next translatable character is read. If more than one delimiter is read at one time, only one new record will be started. A delimiter can be any character that can be punched on the tape (parity holes are ignored). Common examples are carriage return, b0D40, line feed, b0A40, and blank tape b0040. If fewer than 4 delimiters are needed, unused positions can be filled with repeats of used delimiters.

LSP1 to 3 are array L element numbers which are to be changed to be the same as a corresponding LZP1 to 3 element number. For example, if all A's (element 65) are to be translated as C's (element 67), LSP1 would be bbb65, and LZP1 would be bbb67. This can be done with any character that can be punched on the tape. For example, line feeds could be translated as L's by entering LSP2 as bbb10 and LZP2 as bbb76. Line feeds, in this case, cannot be also used as a delimiter because delimiters are not translated. Unused pairs of LSP's and LZP's can be filled with any equal numbers.

LCOL determines how a colon (:) is to be handled. In the translation array (L) all unprintable characters are filled with colons. If LCOL is 0, all colons are ignored. If LCOL is 1, all colons are written as colons. If LCOL is 2, all colons are replaced by spaces.

LREC sets the maximum number of characters that will be written on a record before a new record is started. If, for example, data are in groups of 25 characters followed by carriage returns and line feeds, setting LREC to 25 would start a new record every 25 characters, even though the carriage returns and line feeds were sometimes missing.

A standard card for translating a tape from a teletype machine would ignore rubouts (7F40), use carriage returns and line feeds (0D40, 0A40) as delimiters, have no nonstandard translations, print colons as colons, and set a maximum line length of 80. This card would be b7F40b000 0b0000b0000

b0D40b0A40b0A40b0A40bbbb1bbbb1bbbb1bbbb1bbbb1bbbb1bbbb1bbb80.

The job control (JCL) cards needed to run this program to make a card image dataset from a standard teletype punched tape using the above described parameter card would look like this:

```
//JOBNAMEbJOBb(FXXXXX,2,0,1),REGION=64K
ACCOUNT CARD
//STEP1bEXECbFGILG,PRELIB='LOADLIBRARYNAME',EP='MAIN'
//GO.SYSLINbDDbDISP=SHR,DSN=LOADLIBRARYNAME(MEMBER)
//GO.FT60F001bDDbDISP=(NEW,CATLG),DCB=(RECFM=FB,LRECL=80,
      BLKSIZE=1680),
//bbbbUNIT=LONGSHRD,SPACE=(TRK,(10,2),RLSE),
//bbbbDSN=DATASETNAME
//GO.FT50F001bDDb*
b7F40b0000b0000b0000b0D40b0A40b0A40b0A40bbbb1bbbb1bbbb1bbbb1
      bbbb1bbbb1bbbb1bbb80
/*SYSINbDDbPT
SPECIAL CARD TO SPECIFY DELETE AND EOT CHARACTERS
/*ENDbOFbFILE
```

Two things in the JCL should be explained. First, the "special card" to designate a teletype RUBOUT CHARACTER as the delete character and CNTL D as the end of the tape mark will have, in column 1, punches 2,3,4,5,6,7,8, and 9, and in column 2, punches 2 and 7. More information on feeding tape into RADS stations can be found in Ref. 1. Second, the //GO.FT60F001 card and the following two cards, which are continuations, designate where the output is going. If these three cards are replaced with the card //GO.FT60F001bDDb SYSOUT=A the result will be a listing on the printer instead of a dataset. If the three cards are replaced with the card //GO.FT60F001bDDbSYSOUT=B the result will be a card deck instead of a dataset.

I would like to thank Dr. J. C. Person and Mr. P. H. Froehle for many helpful suggestions, both while writing the program and while writing this report.

References

1. Amiot, L., R. Barr, C. Harrison, T. Murphy, F. Salter, R. Schwanke, and V. Tantillo, Argonne National Laboratory Applied Mathematics Division Technical Memorandum No. 207, p. 218 (Sept 1, 1971).
2. IBM System/370 Reference Summary, IBM Publication No. BX20-1950-2 (March 1974).