SOURCE:	DCE 9 OCT 63
TITLE:	Binary-Keyset Code
COPIES TO:	JHW, TLH, DCL, CPB, SDM, LJC, FKT, JJ
SEE ALSO ITEMS:	
FILE FOLDER:	
FORMAL DESCRIPTORS:	

ABSTRACT: Current code for five-key handset, providing beginning of shorthand features, and discussion of future binary-keyset transcription possibilities.

TEXT: The following chart shows our present assignments. The thirty-one useful hand-stroke finger combinations will represent different input designations depending upon the current interpretive case. It is assumed that a computer will interpret these codes, although there may be paper-tape intermediate storage.

At present, I have the logic worked out for the interpretive program, but it is not coded. Judy has been using the full code in practice transcription onto 5-bit punched tape.

I will use the following naming conventions. For Code i, struck in Case n: Cn.i if i is expressed in decimal form, and Cn/i is expressed in octal form. The four printing cases and the Control Case have n of 1,2,3,4, and c respectively. It is often useful, in actions designated from the Control Case, to return interpretation back to the printing case from which Control Case was last entered -- and I designate this as Cx. Besides this, C1, C2, C3, C4, Cc and Cn used alone will designate Cases 1,2,3,4, or Control Case, or any case, respectively.

To shift case one goes through the intermediate Control Case. For instance, to shift from Case 1 to Case 2, hit C1.27 (shifting to Control Case) and Cc.2 (shifting from Control Case to Case 2). Hitting Cn.27 will always ensure being in Control Case.

1. Printing Cases.

Abbreviations of sorts are included. Codes C1.17 and C2.17 X represent the character pair "qu" and "QU". If a "q" is desired without the "u", use Code C3.17 (or C4.17). Also, in Cases 1 and 2, Codes 29 and 30 designate more than "comma" and "period." Code 29, labelled EOF for End Of Phrase, will designate the character pair "comma space". To get a comma without a space following it, use Codes C3.29 or C4.29. Code 30 (in Cases 1 and 2), labelled EOS for End Of Sentence, designates the following: "period", "space", "space", "shift" to Case 2 for one character and then to Case 1." To get a period, use Code 30 in either of X Cases 3 or 4. If a Code 17 is hit after an EOS, the result will be the character pair "Qu".

X upper case on tape

Xs before C on tape

In anticipation of a question following the description of Codes Cc.15 through Cc.19--C4.11, a Backspace code, has the same effect as backspacing a typewriter. The originally written character past which you backspace remain to be overprinted by succeeding printing characters.

2. Control Case.

In Control Case are a number of special features. Codes 1, 2, 3 and 4 designate direct transfer to the corresponding case. Codes 5, 6 and 7 are used for underlining alphabetic characters. Code 5 says, "return to Cx (the case from which you most recently transferred to Control Case) and underline the next character designated." Code 6 will send you back to Cx and cause underlining of all succeeding alphabetic characters up to the next occurrence of a code of 28 or greater--essentially, underlining the next word. Code 7 sends you back to Cx and causes underlining of all alphabetic characters until an occurrence of either a Code 28 or a Code 30--for underlining section titles. For Codes Cc.6 and Cc.7, a hyphen will be assumed as a legitimate "alphabetic" character to be underlined, but a double hyphen (representing a dash) will be a terminater of the underlining action. For Codes Cc.8 through Cc.13 we designate a short one-character or one-word "visit" to a specified case before shifting back automatically to Cx. Special note: sequence Cc.8, C2.17 results in a printing sequence corresponding to C4.26, Cx.21. Similarly, a one-character visit to Code 17 of Case 1 would produce a "q" and a Code 21 from Cx. The "one-word visit" is terminated by any code equal or greater than 27, and that code will be interpreted as though it were in Cx.

Codes Cc.15 through Cc.19 are used for deleting preceding characters. They all have the effect upon the eventual output text of backspacing a magic typewriter that erases any character in a position to which it backspaces. Cc.15 can be hit n times successively to cause n such effective backspace-deletes, each of which eats up one line space. Cc.16 is used for the last of such a string of backspace-deletes (or when you just want one) if you want to transfer automatically thereafter back to Cx. Cc.17 will automatically do backspace-deletes back to <u>but not including</u> the previous "space", and then will transfer you back to Cx. Cc.18 will automatically do backspace-deletes back to <u>and including</u> (i.e., through) the previous "period," and then transfer you to Cx. Cc.19 will automatically do backspace-deletes back to an including (i.e., through) the previous "tab", and then transfer you to Cx. IMPORTANT NOTE: A delete command is not actually executed until the next printing-case code other than Code 27.

Code Cc.21, "Hand Spec," is used as part of a procedure to tell the interpretive device which hand you are going to use on the keyset. Following recognition of this code, the computer will assume that the very next input code will use either or both the thumb and index finger, but no others -- i. e. you must jump to C1, C2 or C3 on the next stroke -- and it deduces the specified hand from this stroke. Notice that getting to this code from any state can be done entirely with symmetrical codes (codes that are the same for both hands), so that it doesn't matter what hand the interpretation <u>was</u> assuming, or what interpretive case <u>was</u> being used, when you choose to use a give hand to specify the hand-interpretation made. (A Hand Spec code automatically actuates a Cc.27 code as part of its response.)

	FINGER CODE					CODE	CASE			
CONTROL CASE	4	3	2	1	T	NO.	1	2	2	4
						0				
CASE 1					X	1	a	plus	'	
CASE 2				X		2	b	B	-	"
CASE 3				X	X	3	c	C	*	
CASE 4			X			4	d	D	div.	;
Cx, underline next alphabetic character			X		X	5	e	E	=	:
Cx, underline alphabetics until Code greater than 28			X	X		6	f	F	/	?
Cx, underline alphabetics until Code 28 or 30			X	X	X	7	g	G	(	
1-ch C2, Cx		X				8	h	H	)	$\overline{\&}$
1-wd C2, Cx		X			X	9	i	I	TAB	degrees
1-ch C3, Cx		X		X		10	j	J	$	1/2
1-wd C3, Cx		X		X	X	11	k	K	¢	B.S.
1-ch C4, Cx		X	X			12	l	L	#	
1-wd C4, Cx		X	X		X	13	m	M	@	
		X	X	X		14	n	N	%	
Backspace-Delete		X	X	X	X	15	o	O		
Backspace-Delete, Cx	X					16	p	P	0	
Delete to SP, Cx	X				X	17	qu	QU	1	
Delete thru . Cx	X			X		18	r	R	2	
Delete thru TAB, Cx	X			X	X	19	s	S	3	
	X		X			20	t	T	4	
Hand Spec.	X		X		X	21	u	U	5	
	X		X	X		22	v	V	6	
	X		X	X	X	23	w	W	7	
	X	X				24	x	X	8	
	X	X			X	25	y	Y	9	
	X	X		X		26	z	Z	q	Q
Reset Control	X	X		X	X	27	CC	CC	CC	CC
EOP, Cx	X	X	X			28	CR	CR	CR	CR
	X	X	X		X	29	EOF	EOF	,	,
	X	X	X	X		30	EOS	EOS	.	.
	X	X	X	X	X	31	SP	SP	SP	SP

X in wrong place

Note: Cn refers to case N, where N = 1,2,3, or 4.

Cc refers to Control Case.

Cx refers to Case from which last jumped to Cc.

Code Cc.27 (which in other cases causes transfer to control case) leaves you in control case, but clears any pending control action which the code interpreter may be writing to complete -- e.g. any of the underline, case-visit or delete actions.

Code Cc.28, labelled EOP for End Of Paragraph, will automatically set up the output for a new paragraph. For typewriter output, present convention, this would provide the sequence equivalent to C3.30, C3.31, C3.28, C3.28, C3.9, Cc.1, Cc.8.

3. Future Possibilities.

a. Extension of the above

Below are a few salient possibilities among the many that will undoubtedly be uncovered with future study. It will generally require coordinated analysis of editorial conventions and occurrence frequencies to establish efficient transcription codes for an interpretive program of reasonable size and operating cost.

(1) Automatic Underlining. A number of odd conditions seem to arise in trying to specify how far ahead in the text you want printed characters to be underlined. Cc.5 through Cc.7 as designated above don't easily handle all of these. More study is required of occurrence frequency for such conditions before one could say how best to designate codes to handle them.

One new-code possibility, would say, "return to Cx and underline all legitimate characters until a Cc.27 code is struck." Perhaps several such would be useful, depending upon what are to be specified as legitimate characters (e.g. sometimes a space is underlined, sometimes not).

Another new-code possibility, call it Cc.j, would be used in a two-stroke sequence Cc.j, Cx.k, which is then to be followed by printing designations in Cx. Underlining will automatically occur, for all legitimate characters, up to and including the next occurrence of Code k in Case x.

(2) Automatic Delete (and Place Finding). In some situations, e.g. involving, prior case-shift, between-sentence and between-paragraph that are to be corrected, it is not clear that delete commands such as Cc.15 through Cc.19 provide the best facility for deleting and relocating. Especially for an off-line operator who can't see the effect of such commands as interpreted by the computer. Also, an off-line operator may become lost by losing track of recent code strokes and/or their consequences, and may need a straightforward way to relocate correctly.

A Control-Case code could be used to tell the interpreter: "Go to Cc, observe the next word, W, that I enter and search backward until the first occurrence of W. Delete all old entry back through the old W,and, beginning with my new W entry, accept subsequent entry as normal continuation." If you got lost, you could thus go back and restart at a previous unique word where you knew you had been doing all right. This command also allows flexible specification of deletion.

A variant of this code might be preferable, where you follow the Cc command code with a digit (or, two digits) that specify a number, n. The interpreter is to observe the symbol string S composed of the next n input character, and use S (exactly as it would use W in the original variation) to locate the renewed starting point back in the prior input.

For on-line use with CRT feedback, one need not specify n in the latter variant of the command. He could (after giving the command) merely start entering characters and watching a "place marker" until the computer had enough input to have set the marker at the desired spot, then hit two Code 27 strokes to clear control and stop the search.

(3) First Person Singular. We can rather easily incorporate the convention that a C1.9 code (lower-case i) immediately preceded and followed by space codes, is to be outputted as if it were a C2.9 code.

(4) New Sections. This regards test-block categories larger than the paragraph. More study is needed, with establishment of consistent conventions for headings to sections of different levels. Automatic format help for these headings -- e.g. capitalization and underlining -- could be obtained, as well as coordinated subsequent-paragraph indentation. A Cc Code (or set of them) could be used to designate a new section and the section type or level. This can include listings and quoted inserts.

(5) Specifying Abbreviations. The "qu", EOF, EOS, underline, visit, delete, and EOP codes all are abbreviations, and further study of standard transcription message types will undoubtedly suggest other of the types. This section, however, concerns abbreviations the user may wish to establish temporarily or permanently for himself.

One Control-Case code could be used to declare that you were going to establish an abbreviation form. You then enter the full form (both control and data) of the input string you wish to abbreviate, and install a unique separate input code. You follow with the unique characters (abbreviation form, F) by which you wish to enter (designate) the abbreviation in the future. Henceforth,

entry of the abbreviation form will automatically cause substitution of the full form, to produce words and/or control sequences.

Another Cc code could be used to effect removal of a given abbreviation form -- and still another Cc code could be used to designate which group or groups of abbreviations you might wish to make use of -- if such would be useful. There could well be a standard set you would wish to use basically, to which in some given writing place you could usefully add some of your own. Unless you remembered these local-use additions, and could control their accumulation and discarding, it could be very handy to call for a fresh start with a standard group of abbreviations.

We could ask the interpreter to check every input word against a table of abbreviation forms -- or we could constrain all such forms to begin with one particular character (or one of a special set of characters) so that lookup was done for only those input words beginning with a special character -- or we could say that a special Cc code must accompany every input word which is to be interpreted as an abbreviation form.

X wrong on tape

X Personally, I would think there are few enough normal words beginning with z or x to let one or both of them, as an initial character of a word, tell the interpreter to look at the abbreviation table. Using a special Cc code to designate this, where this code has no other use, would seem to be of advantage mainly if you wanted feedback if the interpreter found no such abbreviation form in its table -- since with such as the z or x designater we would have to let the interpreter point the input word if it didn't find it on the table.

(6) Control Trees. We could forsee possibility in the foregoing section of a number of different control commands associated with setting up or deleting abbreviations. These wouldn't be too frequently used, and Cc codes will become precious as we learn more use for them. For less-frequent control commands, it could pay to bundle them in one or more second-level control cases to which entry is made from the first-level Control Case (the one we now have). For still-less-frequent commands, third-level control cases might be fitting, etc.

X wrong on tape.

b. Special Two-Handed Possibilities. Alternating two hands on two keysets definitely seems a useful possibility for faster transcription. The X interpreter has to keep track of which keyset provides a given entry, and separate Hand Spec procedure is needed for each keyset. Otherwise the string of sequential input codes is interpreted without regard for which keyset enters which code.

We can stipulate that the user, for this type of interpretation, never overlaps his key strokes -- i.e. only one set of keys is depressed at a time. We can then stipulate that if overlap exists, the two overlapped codes are to be interpreted differently.

For instance, we may stipulate that a code stroke that is held so as to be overlapped by the succeeding other-hand stroke is to be interpreted as a Cc code -- and with the succeeding overlap code being interpreted normally. But this only saves one stroke, and at the expense of a break in stroke rythmn. I think there is better use to be made of overlap.

A better use would seem to stem from saying that the two overlapped five-bit codes are to be interpreted as an independent ten-bit code. This gives us 961 unique codes for which to designate special abbreviations, control tricks, etc. and opens the door to a coordinated "shorthand" code. Phrases, words, n-grams (including spaces punctuation and control characters), whose payoff in frequency of use times 5-bit code strokes saved is large enough, would thus be encoded as single two-handed overlap strokes.

It seems reasonable to say that it is the order of keyset actuation, rather than which hand actuates which code, that should determine the way the interpreter groups the two five-bit codes into a ten-bit code. This would seem less distrubing to the character-at-a-time alternate-hand rythmn in which the shorthand codes would be embedded.

However, it would be possible for the order of overlapping -- which hand was first -- to be of significance. This could provide two independent 961-code interpretive repertoires. It is also possible X to say that no pair, the interpretation gives the same designater role to the same hand every time. X wrong on Taped

The shorthand forms can be remembered and stroked as a pair of C1 characters rather than as a ten-bit code. Also, this shorthand springs compatibly from the five-bit character-by-character code, and a person can gradually add new shorthand forms to his repertoire as his familiarity and skill grow.

c. More Code Forms Per Hand Stroke. I think that some or all of the fingers could probably control more than one bit per hand stroke. The pressure-sensitive pads (see CBC 4 OCT 63 will allow us rather easily to experiment. Two pads per finger gives five-level radix-3 codes (3 exp 5 = 243, yielding a 242-code-form hand stroke) and three pads per finger gives five-level radix-4 codes (4 exp 5 = 1024, a full ten-bits, yielding a 1023-code-form hand stroke). It can be arranged that one finger can activate either or both of two pads, but this is equivalent to a three-pad, one-out-of-three arrangement.

It remains to be learned how best there different code-forms-per-stroke and encoding possibilities can be used. I plan to experiment (Judy and me as subjects) on basic feasibility of these different approaches, but it would require a full-blown project to explore and evaluate meaningfully.