Apple Manuals Apple][Pascal Macintosh SwyftCard Canon Cat

Jef Raskin Information

SWYFT CARD VERSION 972
TECHNICAL NOTES

AUG 1985 / PAGES: 7

SOURCE: JOHN BUMGARNER / APR 1997

DOCUMENT NO. 53

COMPILED BY
DAVID T. CRAIG
736 EDGEWATER, WICHITA, KS 67230 USA
E-MAIL: 71533.606@compuserve.com

Apple Manuals Apple][Pascal Macintosh SwyftCard Canon Cat

VERSION 972 -- features added or changed since Swyftcard version 820.
Jonathan Sand and Dave Lavond
30 AUGUST 1985

New key locations

There are now only five commands. They are assigned to new keys:

Control key	Routine
A	Insert
D	Send
G	Calculate
N	Print
L	Disk

Creeper cursor

If there is no selection, pressing (and letting go of) the left leap key will move one character to the left; pressing (and letting go of) the right leap key will move one character to the right. This is fondly called creeping. If there is a selection, pressing the left leap key unhighlights the text and leaves the cursor at the left end of the selection; pressing the right leap key unhighlights the text and leaves the cursor at the right end of the selection. Creeping is useful for adjusting the size of the selection. Creeping always leaves the cursor collapsed. Creep is as fast as possible.

Leaping

If you already have a selection and you start Leaping left, the cursor will now land on a pattern within the selection first, as opposed to after exhausting the locations of that pattern outside the selection.

Leap will not miss pattern characters even if you type fast in a large text. **Care must be taken now to make sure that you hold the Leap key down until the pattern is found or else the last character or two will be inserted into the text.**

Leaping is very fast. Try it.

Word Wrap

A word at the right end of the screen will wrap to the next line if it is followed by two spaces, when there is only room for one or none. A third space, however, will wrap to the following line.

1

Disk

The Disk command will always save the text to disk no matter what change or lack of change has been made to the text.

You can now visually tell that you have saved a text. The rate at which the cursor blinks is doubled after a save. It slows back to the normal rate after you add or delete characters.

The Get command is no longer.

Drag forward has been added. To use Drag forward, the current text is saved using the Disk Command. The passage of text to be copied to another disk is selected, the destination disk then inserted in the drive, and the Disk command used. The appearance of the Swyftcard message indicates that the selection is being Dragged. The selected text will be inserted into the destination text where the cursor had been when the disk was saved. This final text can then be saved to the disk with another Disk command.

The Disk command is now faster because it removes the gap prior to saving the text and restores the gap when loading text. Only the text is placed on the disk. This means that text made with previous versions cannot be accessed by the Disk command. Instructions for updating old text will be available.

The ProDos conversion disk is now available.

If the text is empty and an Apple disk is in the drive, then the Disk command will boot it.

After using an Apple disk, Swyftcard can be rebooted by placing a text disk in the drive and pressing the control open-apple and reset keys.

Reassignable keys

All control keys (alphabetical, ^, -,], and \) can be assigned to execute code at any address. Multiple keys can be assigned to a single routine. The code **must be in the high 64K bank when using the Disk based version, and in the low 64K in the Rom based version. A Magic number in key translation table, when modified, reassigns all keys to their original routines. There are extensive** instructions later in this handout. The address of the bottom of the text area can be adjusted up or down within the decimal range, 5632 to -19757, by using BASIC to store this value in BT%. The algorithm which moves the text will neither destroy text nor set the bottom too low. See the instructions which follow.

Insert

When text is nearly full and the user Inserts enough text to fill all available space then the system inserts the text, empties the cut buffer, and beeps. Further attempts to Insert will cause a beep because the cut buffer is empty. Insert is now faster.

Send

A user can tell if the text he is sending over a modem is being received by a Swyftcard by sending control-Z control-C: the receiving Swyftcard responds with the message "SWYFT", preceeded and followed by carriage returns.

Control-G received over the modem will cause the receiving user to hear one bell; no character is entered into the text.

A line feed or carriage return or both can be Sent by using the BASIC variable LE%. Add the first character to 256 times the second character. A carriage return (13) and the line feed (10), yields LE%=2573. The default is -1.

When a character is received and the text is full, the bell rings until the delete key is pressed.

You can now Send at 153,600 baud! Unfortunately, text is received and display at only 4,800 baud, because the receiving computer must update its screen. The baud rates and the asscociated value of SE% are tabulated here:

baud	SE% value
300	...
1200	...
2400	...
4800	...
9600	...
19,200	...
153,600	...

Print

The page length can be changed with the BASIC variable PL%. Swytfcard makes sure that PL% is big enough for a top (AB%) and bottom (BE%) margin and at least one line of text.

The maximum number of pages was 125; it is now 200.

Underlining is now available. It is toggled on and off in the text by the underline character. The underline character is not printed; a space is substituted for it in the printout. Underlining is also turned off by a pair of carriage returns or by a pagebreak (actual or implied). A string of underlines is printed as a solid underline, except that the first and last underline are, for consistency, printed as blanks.

The BASIC strings US$ and UE$ are passed to the printer to turn underlining on and off, respectively. The user customizes them for their own printer. They can be up to 5 characters each. The first element in UE$ is a group flag. If each character is individually underlined, by backspacing, then the group flag should be set to CHR$(0). If underlining is done automatically by the printer, set the group flag to CHR$(1), as with the EPSON.

```
US$=CHR$(27)+CHR$(45)+CHR$(49)               default (for the EPSON printer)
UE$=CHR$(1)+CHR$(27)+CHR$(45)+CHR$(48)

US$=CHR$(95)+CHR$(8)                                     (for other printers)
UE$=CHR$(0)
```

BASIC

The BASIC commands RUN and NEW now delete all user strings except printer strings for initialization, underline on and underline off. If you have configured for your printer, RUN and NEW will no longer return you to the default printer configuration! You no longer have to use GOTO to avoid this.

A BASIC program which reads or writes Swyftcard variables or strings must be executed with a GOTO statement. RUN clears all of the variables and strings.

A handy BASIC command to observe the contents of strings is the following, note the ",I," in MID$. For example, substitute PR$ for string-name$ below to see the string used for printer initialization:

```
FOR I = 1 TO LEN(string-name$) : ? ASC(MID$(string-name$,I,1)) : NEXT I
```

The default BASIC strings, for the EPSON FX-80 printer, are:

```
PR$=CHR$(0)+CHR$(0)+CHR$(5)+CHR$(27)+CHR$(77)+CHR$(27)+CHR$(108)+CHR$(8)
US$=CHR$(27)+CHR$(45)+CHR$(49)
UE$=CHR$(1)+CHR$(27)+CHR$(45)+CHR$(48)
```

For other printers, the underline strings that will always work are:

```
US$=CHR$(95)+CHR$(8)
UE$=CHR$(0)
```

Reassigning the Key Commands

All control keys execute a SwyftCard command. They can be reassigned with simple BASIC statements. The commands are executed through two tables, the translation and execution tables, at decimal 5536 and 5568, respectively. As only some of the control keys are assigned, both tables are only sparsely populated:

Control Char	ASCII	Translation Table address	Execution Index	Execution Table Address	Name of Routine	Routine Address
Delete	0	5536	1	5570	DELETE	58263
A	1	5537	7	5582	PASTE	58622
D	4	5540	5	5578	SEND	59859
G	7	5543	8	5584	BASIC	63510
Tab,I	9	5545	27	5622	TAB	58483
L	12	5548	4	5576	DISK	64854
N	14	5546	2	5572	PRINT	60745
Z	26	5562	12	5592	CONTROL	59817
Escape,[	27	5563	28	5624	ENTER	58464
			0	5568	NOOP	54607
			29	5626	magic #	12855
1	2	3	4	5	6	7

The control key (column 1) is received, by the SwyftCard, as its ASCII value (column 2). This value is first converted to an index (column 4) by using the value as an offset into the translation table (column 3). Since each element in the execution table is 2 bytes, the index is doubled to use as an offset into the execution table to yield the table address (column 5) containing the execution address of the routine (column 6 and 7). All values in column 7 change with each version. Control keys not included in the above table have been assigned an index of 0 which executes NOOP. Unincluded indices also execute NOOP.

So, when you press the control N key, the Swyftcard converts it to an index of 2, which it got from location 5546. The index is used to locate, at 5572, the address of the PRINT command, which is executed.

Since these tables are located in RAM, their values can be modified so as to reassign keys to new commands. For instance, control P can be assigned to also execute the PRINT command, in addition to control N. Alternatively, control P can be reassigned to a custom program you write that, say, replaces one word with another (actually not impossible...), or that does a Sort.

At the end of the execution table there is a magic number, related to the version number. If this number is changed, the table will be reinitialized, either after a BASIC command, or after pressing control Reset.

So, if you want to assign that control P key to the print command, you place, using a BASIC statement, the index of the print command, 2, into the 16th position (control P) past the beginning of the translation table:

```
?5536+16 = 5552
```

Try changing it and using the control P key:

```
POKE5552,2
```

On the other hand, if you want to assign the control P key to a completely new routine, you must find an unused index, say 10, and store that at 5552:

```
POKE5552,10
```

At this point, since the index 10 is assigned to a NOOP, the control P key will once again be inert. To assign index 10 a routine, you must store the address in the 10th 2 byte position past the beginning of the execution table:

```
?2*10+5568 = 5588
```

Let's store the warm start routine (53760). Since the POKE command only stores a byte at a time, we have to figure out the value of the high byte and the low byte. The high byte is highest integer lower than 53760 over 256:

```
?53760/256 = 210
```

The low byte is the remainder:

```
?53760-(256*210) = 0

POKE5588,0      (low byte)
POKE5589,210    (high byte)
```

The Get command can be reassigned to the control R key. The Get routine is at 63598; the high byte is 248, and the low byte is 110. The control key is already assigned an index of 6. The execution table location for this index is 5580. The BASIC statement is thus:

```
POKE5580,110:POKE5581,248
```

If you make a mistake, you can force the Swyftcard to reinitialize the key command tables by storing a zero at location 5626:

```
POKE5626,0
```

Remember not to reassign the key that performs BASIC without first assigning another key to it...

This BASIC program will print the current key assignments. Below it is its output. May all of your reassignments be good ones.

```
10 ?:?"ASC trans  index  exec  routine":?
20 FOR I = 0 TO 31
30 A=5536+I
40 B=PEEK(A)
50 C=5568+(2*B)
60 D=PEEK(C)+(256*PEEK(C+1))
70 ?I;:?"    ";
80 ?A;:?"    ";
90 ?B;:?"    ";
100 ?C;:?"    ";
110 ?D
120 NEXT
130 END
```

```
ASC trans  index  exec  routine

0    5536    1    5570    58263
1    5537    7    5582    58622
2    5538    0    5568    54607
3    5539    0    5568    54607
4    5540    5    5578    59859
5    5541    0    5568    54607
6    5542    0    5568    54607
7    5543    8    5584    63510
8    5544    0    5568    54607
9    5545    27    5622    58483
10    5546    0    5568    54607
11    5547    0    5568    54607
12    5548    4    5576    64854
13    5549    29    5626    12855
14    5550    2    5572    60745
15    5551    0    5568    54607
16    5552    0    5568    54607
17    5553    0    5568    54607
18    5554    0    5568    54607
19    5555    0    5568    54607
20    5556    0    5568    54607
21    5557    0    5568    54607
22    5558    0    5568    54607
23    5559    0    5568    54607
24    5560    0    5568    54607
25    5561    0    5568    54607
26    5562    12    5592    59817
27    5563    28    5624    58464
28    5564    0    5568    54607
29    5565    9    5586    55976
30    5566    3    5574    54607
31    5567    11    5590    54607
```

END