COMPUKIT UK101 NEW MONITOR

BY ROGER CUTHBERT March 1980

PLEASE NOTE:—

- The New Monitor is supplied with all COMPUKIT UK101 purchases.
- References to the old monitor in the COMPUKIT UK101 manual must be ignored.
- The New Monitor technical overview is printed below.

Insert ROM into socket observing correct polarity. Apply power and press reset.

OPERATION

Some movement of subroutines has been inevitable in designing the new monitor but if the vectors have been used to enter the five main subroutines listed below then no problems will arise.

	INDIRECT JUMP IN.	VECTOR REF.
INPUT ROUTINE	$FFEB	JMP ($0218)
OUTPUT ROUTINE	$FFEE	JMP ($021A)
CTRL C ROUTINE	$FFF1	JMP ($021C)
LOAD ROUTINE	$FFF4	JMP ($021E)
SAVE ROUTINE	$FFF7	JMP ($0220)

Small changes - Input ASCII character from tape now in $FE6D and DISC bootstrap has been removed.

CURSOR - In all modes except edit there is a choice of steady or flashing cursor. In edit mode for BASIC only the cursor will always flash and at a rate that is faster than when flashing in non edit modes. This allows identification of being in editor.

The default mode is flashing cursor, this is always set up when reset is used but if you wish for a steady cursor then place a non zero value in $020F (decimal 527) eg. POKE 527 with 1

NOTE! - When using flashing cursor then a key value is entered as the key is lifted and so auto-repeat is not available. However when using steady cursor the key is entered on pressing the key and if held will auto-repeat.

STORING OF DATA ON TAPE - Sending of DATA

PRINT CHR$(2);P$ - for strings
PRINT CHR$(2);X - for variables

On execution of either of the above lines the data is sent out to the tape as well as the screen. The CHR$(2) is a signal to the output routine to send all following until the next RETURN out to the tape. But the CHR$(2) will only work if it is the first print character of a line ie. any PRINT statement that preceded this one must NOT have a comma or a semi-colon. In addition the string or variable must NOT be terminated with a comma or semi-colon but the CHR$(2) MUST always have a semi-colon. These rules also apply to the retrieval of data.

Retrieval of DATA

PRINT CHR$(1): INPUT P$ - To retrieve a string
PRINT CHR$(1): INPUT X - To retrieve a variable

The comments above about comma's and semi-colons are the same for this retrieval but note the colon before the INPUT
It is possible of course to use two lines eg.
PRINT CHR$(1) }
INPUT P$ - } the above is neater

Remember it is not possible to retrieve data that does not exist and the routine would stay in a continuous search. If you try to input a string into a variable the BASIC will print error. However a variable can always be input as a string.

To avoid this, start any data storage with a string that provides information about the stored data including its length if known. If not then use an end marker.
eg. PRINT CHR$(2); "END"
Then to retrieve we seek the end.
PRINT CHR$(1): INPUT P$
IF P$ = "END" THEN .....

Therefore always retrieve in the same order as sent and use some method of data identification with something to tell when all data is in. You will need something to signal tape on or off.

eg. INPUT "Type 'GO' when tape running";Z$
PRINT CHR$(2);P$

No action is needed on Z$ unless you wish to add an exit in case data is not to be sent after all.

EDITOR

This is only available when using BASIC and is for amending lines of program and only one at a time. To enter the editor type CTRL 'E' and EDIT will print on the screen. It is now waiting for a line number. If however you press only RETURN then the cursor moves to the next line and editor is not entered. If you type a non-decimal character it will exit immediatly but if you type a line number then that line will be listed and the cursor will be seen to flash faster.

N.B. if you type a line number that is not in the program then editor will be entered but only blank spaces appear. To exit press RETURN.
When in EDITOR -
You may move the cursor at will to edit;
UP .....CTRL 'K'
DOWN .....CTRL 'J'
RIGHT .....CTRL 'I'
LEFT .....CTRL 'H'

To ERASE place the cursor over the character and press RUBOUT.

To INSERT between two characters place the cursor over the right hand of the two between which the insertion is to be made and type.

TO ENTER the amended line the cursor must sit somewhere in the line and press RETURN.

This line will now replace the old one of the same line number so note if you alter the line number it will replace the line of that number or become a new line if no such line was present. The extracted one will then be unchanged.

CLEAR SCREEN

This may be done directly from the keyboard with CTRL 'L' and is blind to any routine seeking input. From program PRINT CHR$(12); - the semi colon is to stop the automatic C/R L/F.

CURSOR MOVEMENT

These can only be used from program.

UP.....	PRINT CHR$(11);
DOWN.....	PRINT CHR$(10); - same as line feed
BACK SPACE.....	PRINT CHR$(8);
RIGHT.....	PRINT CHR$(9);
START LINE.....	PRINT CHR$(13); - same as carriage return

NOTE a semi colon must always be used to stop the carriage return line feed that BASIC will send if not there.
The above can be put into strings.
eg. CL$ = CHR$(12)
Then to clear screen PRINT CL$;
or to place the cursor top left with out clearing the screen some times called home cursor;
HM$ = CHR(13): FOR J = 1 TO 15:HM$ = HM$ + CHR$(11):NEXT
Now to home cursor PRINT HM$;
N.B. The characters are counted as printed characters by Basic and can upset the correct position if used when TAB is involved. On these occasions it will be better to calculate spaces and use SPC.

STACK

All stack initialisation has been set to $FF to use the full stack.

INTERRUPT

The vectors have been changed to take them out of the stack area but compatability is maintained as RESET places jumps in the new locations back to the old settings. This makes old routines compatible but allows the chance to write new programs that do not conflict with the stack.

NMI $0222
IRQ $0225

More notes on data saving
The format is as follows;
/02/....string or variable../03/CR/..10nulls../LF/

The marks 02 and 03 are used by the routine to identify start and finish of a line.

The CR nulls LF serve two purposes;
1. They provide a break between data and allow time for some processing but take care on the amount.
2. As the tape is read then the CR and LF are already there as it goes to the screen.

N.B. When forming strings for saving remember that on retrieval BASIC will ignore any ASCII value less than eleven.

PAGE 2 STORE ALLOCATION

ADDRESS HEX	DECIMAL	CONTENTS
$0200	512	Temporary holding
$0201	513	BYTE from under the cursor
$0202	514	Temporary hold for A during screen print
$0203	515	LOAD flag
$0204	516	Unused
$0205	517	SAVE flag
$0206	518	CRT baud rate
$0207	519	CURSOR position on a line 0-47
$0208	520	CURSOR row number 0-15
$0209	521	Temporary hold of $0207 in EDITOR
$020A	522	Temporary hold of $0208 in EDITOR
$020B	523	Count of number of characters per line for EDITOR
$020C	524	DATA SAVE flag
$020D	525	DATA INPUT flag
$020E	526	DATA INPUT flag
$020F	527	FLASHING CURSOR flag - 0 for flash <>0 for steady
$0210	528	FLASH rate
$0211	529	Unused
$0212	530	CONTROL C flag
$0213	531	}
$0214	532	}
$0215	533	} used by keyboard routine
$0216	534	}
$0217	535	Unused
$0218	536	} INPUT VECTOR
$0219	537	}
$021A	540	} OUTPUT VECTOR
$021B	541	}
$021C	542	} CONTROL C VECTOR
$021D	543	}
$021E	544	} LOAD VECTOR
$021F	545	}
$0220	546	} SAVE VECTOR
$0221	547	}
$0222	548	}
$0223	549	} NMI but reset puts in a JMP $0130
$0224	550	}
$0225	551	}
$0226	552	} IRQ but reset puts in a JMP $01C0
$0227	553	}

Subroutine Entries

$F80B — Editor
$F9E7 — Test for key down. A = 0 for no key A<>0 for a key pressed
$F9F2 — Increase cursor position record by one.
$FA05 — Decrease cursor position record by one.
$FA13 — Input then check for edit, rubout and clear screen.
$FA57 — Screen print routine.
$FB22 — Clear screen.
$FB60 — Move display up one line.
$FB8D — Form cursor address in $E3/$E4.
$FBAC — Input routine.
$FBD4 — Output routine.
$FCB1 — Send A to cassette port.
$FD00 — Keyboard routine.
$FE00 — Monitor.
$FE05 — Entry to monitor by-passing stack initialisation.
$FE6D — Input ASCII from port, bit 7 clear; was in $FE80.
$FE93 — Convert ASCII hex to binary result in A = 80 if bad.
$FF00 — Reset.
$FF8B — Load flag routine.
$FF96 — Save flag routine.
$FF9B — Control C routine.
$FFEB — Indirect input — JSR here to enter via vector.
$FFEE — Indirect output — JSR here to enter via vector.
$FFF1 — Indirect control C — JSR here to enter via vector.
$FFF4 — Indirect load flag — JSR here to enter via vector.
$FFF7 — Indirect save flag — JSR here to enter via vector.