WEMON
The Definitive Monitor

The contents of this booklet, together with the accompanying EProm, are the sole Copyright of Watford Electronics. Their reproduction in whole or part is not permitted without prior consent from the manufacturers.

WEMON MONITOR

This monitor has been designed to compliment the limited facilities offered by the standard monitors of the Ohio Superboard range and the UK101.

The monitor is available for all known commercial variants of these systems ie Standard Superboard (dual mode),Enhanced Video Superboard and UK101

The facilities available are covered in detail in later sections but briefly they are:
1. Full on screen editing using normal cursor including insert and delete.
2. Alpha lock keyboard.
3. Direct keyboard control of tape motor, printer, Video mode (series 2 Superboards).
4. Wide range of user accessible vectors for command extensions etc.
5. Improved tape file handling including named files, fast machine code load and save.
6. Extended machine code support routines.

1.1.0 KEYBOARD

The keyboard operates in two modes:
i) Alpha lock- with shift lock down, capital letters and lower case punctuation are returned. To access upper case, non alpha characters, press either shift key.
ii) Standard typewriter mode- with shift lock up all characters returned are lower case. To access upper case press a shift key in conjunction with a character key.

1.1.1 In addition to the normal characters obtainable, full control characters are available by pressing CTRL in conjunction with a character key. Note however some of these control characters are already allocated to specific machine functions. These are:

CTL E (05) Basic Extension	CTL T (14) Tape On/Off
CTL S (15) Video Swap (Not UK101)	CTL Q (11) Toggle Quotes
CTL P (10) Toggle Print Flag	CTL D (04) Open Screen

Cursor control is accomplished as follows:

SERIES II/SERIES I		UK101
Cursor Left	- Shift Repeat (CTL X (18))	CTL H
Cursor Right	- Repeat (CTL H (08))	CTL I
Cursor Down	- Line Feed (CTL J (0A))	CTL J
Cursor Up	- Shift Line Feed (CTL Z (1A))	CTL Z
Delete	- Rubout (CTL N (0E))	CTL N
Home Cursor	- ESC	CTL A
Clear Screen	- Shift ESC	CTL L
Insert	- Shift Rubout	Shift Rubout

Note that use of the shift key gives the complimentary function, ie Right Shift and Repeat - Cursor Left

1.1.2 Certain combinations of keys give other functions. These are:
CTRL and LEFT SHIFT - followed by a single letter returns a complete Basic command word.
This is effective on all keys except, shift lock, left and right shift, control, escape, cursor left/right and break. See Appendix 1 for key/command allocation.

2.0 SCREEN FACILITIES

2.1 Editing. You can edit program lines by listing them on screen and typing in the corrections directly over the displayed data. Thus to change a character, position the cursor over the character to be changed, type the new character/s, press RETURN and the edited line will be entered into memory. If characters are to be inserted, position the cursor where the insertion is to be made, Press Right Shift and Rubout to open up a space in the line and type in the extra characters. Press RETURN when complete. If you wish to remove some of the line, place the cursor on the first character past the section you wish to delete, then press RUBOUT until the unwanted section is removed.

2.2.0 OTHER SCREEN FUNCTIONS

2.2.1. Open line. Should you wish to place a blank line in between lines already on screen, move the cursor up to the position where you wish the blank line to be, then press CTRL and D, the portion of screen from the cursor line downwards will scroll down leaving a blank line at the cursor position.
2.2.2 Should you require a space at the top or the bottom of the screen simply press cursor up (Right shift + Line Feed) or cursor down (Line Feed) until the cursor reaches the top or bottom of the screen, continuing this action will scroll the screen down or up until the required blank lines are on the screen.
2.2.3 Screen clear/home cursor. Pressing ESC will place the cursor in its home position, top left hand corner of screen. Left shift plus ESC will clear the entire screen and home the cursor.
Note: For UK101 some of these commands use different keys (see 1.1.1).

2.3.0 PROGRAMMED EDITING

Should you wish to obtain the facilities discussed above within a program ie. to have formatted display, then the range of editing commands discussed above may be included singly or in any combination within quotes, just as in an ordinary Print statement. Note that the echo to screen input routine will translate any cursor commands into a representative symbol, this graphic code is stored in program memory. When the program executes, these graphic codes will be translated back into cursor control commands and executed. Should you wish to edit a line containing program cursor commands, and you are still in quotes mode (cursor is a ↖), then toggle the quotes mode (CTRL Q), edit the line as required, toggling CTRL Q as necessary for new programmed cursor commands. This is because, when in quotes mode, cursor commands are translated and will thus not execute until the program runs.

2.3.1 VIDEO SWAP

On series two superboards, video swap toggles the video mode as discussed above. On standard superboards, pressing CONTROL S will cause all subsequent lines listed on screen to have a blank line inserted between them. This is effective until cancelled by another CONTROL S. This greatly improves the readability by altering the format to 25X12. This function is available within a program by POKE 296,32 (25X25) or POKE 296,64 (25X12).

2.4.0 USING CHR$

If CHR$ is used to display ascii characters it is necessary to POKE 550 with 225 to ensure correct ascii operation. This is not necessary with Superboards in the 25X25 mode. The above POKE sets the Quotes Flag and is reset by a carriage return.

3.0 OTHER FACILITIES AVAILABLE FROM THE KEYBOARD

In addition to those already discussed there are more facilities available as direct commands from the keyboard.

3.0.1 Command responses:

M Passes control to MCM
C Jump to Basic Cold Start
W Jump to Basic Warm Start
U JSR to USERVEC
D Indirect jump to DISCVEC +1

DISCVEC Points at $9800 to conform to Watford Electronics'Disc System but may be changed to suit or used for other purposes.

3.1.0. TAPE CONTROL

If a reed relay and its associated driver U68 (7406) is fitted correctly as Fig 1 then automatic control of the tape motor is available. Any Save or Load command will, if the circuitry of Fig 1 is incorporated, automatically turn on or off the tape motor as required. There is a 5 second delay built in to allow the tape leader to clear the head and to create inter-file gaps.
In addition you may toggle the tape motor on/off from the keyboard for rewind etc, using CTRL + T, a flag is maintained of the current state of the tape motor in TPFLAG ($23F), Bit 7 of this location signifies tape off, Bit 7 set signifies tape on. The control is exercised using the RTS line of the ACIA. In addition, the tape control line may be set or cleared under program control using the subroutine TSW, to reverse the current state, JSR to TSW. To turn ON the tape, JSR TSW+5, to turn OFF the tape, JSR to OFF. For addresses of these routines see Appendix 3.

3.2.0 QUOTES MODE

As discussed in editing, the quotes mode may be reversed using CTRL Q. The quotes mode is controlled automatically by the output driver, the cursor changes to a [R] to signify when quotes is on. For editing of text whilst in quotes mode, quotes must be turned off temporarily to allow the output driver routine to execute the cursor commands directly rather than enter them as program data. The QUOFLAG ($226) is a copy of the current state of the quotes mode 00 = OFF, FF = ON.

3.3 BASIC COMMAND EXTENSION

To allow the expansion of Basic commands a loop has been provided in the Basic CHRGET routine. This loop commonly known as a wedge is activated by CTRL E. When inactive, Basic will respond with a SN Error to any direct command it does not recognise. If the wedge is activated then further tests are carried out on the input before Basic 'gets' it. These tests are;- Is it the first character in the line? Is it a direct command from the keyboard? Is it alpha numeric? If all these tests succeed then control is passed via a JSR to EXTVEC. This is a 3 byte location which should contain a JUMP to the code that you wish to execute. At entry the stack has +4 RAH +3 RAL +2 FLAGS +1 ACC with the character in the accumulator.
If you wish to have more than one command addition, then your code should begin with comparison and conditional branches. It is advised that before handing control back to Basic the Acc pushed on the stack containing the command letter be exchanged for a ($3F) to prevent the SN Error that would occur.
The wedge can only be deactivated by a Reset. There is no reason why it should be deactivated as long as the contents of EXTVEC ($248,9,A) point to valid code or $248 contains $60 (RTS). Be careful to leave the stack as on entry (apart from the character saved changed to $3F) otherwise Basic may get confused.

3.4. SCROLL SPEED

During list operations, except during SAVE, pressing the space bar will slow the scroll rate to 1/5 the normal rate.

In addition the contents of $206(518dec) will slow the normal scroll rate. Default value is 0.

There is a parallel printer driver routine incorporated, Centronics compatable, which is initialised at power on. The routine expects to find a 6520 addressed at $8800,01,02,03. with line connection as follows: PA0 - 7 D0 - D7. There is an associated print flag ($256) which is toggled by CTRL P. 00 = No Print, FF = Print. Every call to the output routine (through $FF69) tests the print flag, if set a JSR is made to the print driver routine PRINTOUT. If you toggle the print flag with no printer attached and addressed as above, the program will 'hang'. Of course the output point may be used for any other parallel output device that is capable of generating the handshake signals required.

4.0. VECTORS AND COMMAND EXTENSIONS

There are numerous vectors associated with WEMON, all of them intended to ease the problems involved in extending and expanding the system.
4.1.0 The NMI vector no longer points to the middle of the stack but now has a 3 byte location in Page 2 memory at $22E,F,230.Three locations are used to allow a JSR or JUMP command to be placed here.
No addresses are written to these locations at reset. If you wish to make use of them for an NMI related function put the address in NMIVEC + 1 (low byte) and NMIVEC + 2 (high byte), with a $4C in NMIVEC. Don't forget to begin your routine with the register saving code and terminate with an RTI if you wish to resume execution at the point where the interrupt occurred.
4.2. IRQVEC is tested differently. There is an IRQVEC at $231,2,3. However the IRQ Vector in WEMON ($FFFA,B) points at a routine within WEMON which determines whether the IRQ occurred because of an external signal on the IRQ pin or because of a software BRK ($00). In the former case control passes to IRQVEC and similar comments apply as for NMI. In the latter case, however, control passes to the Machine Code Monitor in WEMON (see MCM).
4.3. USERVEC is again a 3 byte location at $229,A,B. Control passes to this location in two ways:-
1) By entering U in response to the command prompt M/C/W/D/U , again, no addresses are loaded at Reset. This is so that you don't have to re-initialise all the vectors after you use Reset unless, of course, you corrupt the vectors. By entering the address of a routine in USERVEC you can execute that routine by responding with U to the command prompt.
2) If, while in MCM command mode, you enter an unrecognised command ie. not M,R,B,L,S,F,G,V,X control passes to USERVEC direct via a JSR. If you have not entered an address in USERVEC the probable result will be entry to MCM on encountering a 00 byte, although this cannot be guaranteed. If you put $60 in USERVEC then an unrecognised command will have no effect since the statement in WEMON after JSR USERVEC is a jump back to the command prompt. Thus USERVEC may be used to add extra Machine Code Commands to WEMON by putting the address of the code to handle the functions in USERVEC, then entering the unique command letter you have assigned to the function, in response to MCM's prompt. Your routine should test the accumulator contents to see if they contain your command letter, if not RTS.

4.4. Cursor flash rate CURFL $22D, is loaded with $7F at reset.Reducing this value will slightly increase the flash rate.If the value is $80 the cursor will stop flashing, disappearing when a key is depressed.
NOTE The cursor is only on the screen when WEMON or BASIC is awaiting input.

5.0. TAPE HANDLING

The tape facilities are greatly expanded over the basic machine. Briefly the facilities available are:-

1) Auto control of tape motor.
2) All files may be named (6 chars).
3) No requirement to enter LIST to start SAVE or LOAD to terminate it.
4) Machine Code and Named Basic Save and Load are approximately 3 times as fast as a standard Load.
5) MC Save Puts the address of data on the tape, LOAD can load to the saved address or to a different location.

5.1.1. Wemon Save and Load format.

Save for a Basic file takes the form of a Hex dump of the program memory, rather than the normal Ascii file. This gives a significant economy in file length and thus Save and Load time(approx.3 times faster). This is because the program is stored in tokenised form.There will be no listing of the program, however, during save and load. If you wish to to load a program saved in Ohio/UK101 form type LOAD(RETURN). If you wish to save a program in this form type POKE517,255 start the tape, then type LIST(RETURN).
SAVE"AAAAAA(RETURN) will save in the new WEMON format with no further action required. AAAAAA is the file name which can beup to six alpha-numeric characters and must be preceded by quotes. Closing quotes are not required. If automatic tape control is not fitted start the tape unit before pressing RETURN. There will be a 5 second delay to allow the leader to clear the tape head followed by the message SAVING AAAAAA.
When the cursor and OK reappears the program is saved.
SAVE(RETURN) will save with a null string file name.Reload this tape with LOAD"*(RETURN).

5.2. MACHINE CODE TAPE FILES.

The method is very similar to 5.1.1. but no quotes are required and only the first letter of the command is needed, ie, S or L. Machine code files may only be handled from the MC Monitor.

5.2.1.MC Save. The syntax for a MC Save is S AAAAAA(RETURN) XXXX-YYYY where AAAAAA is the optional file name, when no file name is requested, the RETURN should directly follow the S. The cursor will step right one space after the first RETURN and continue flashing to signify the address parameters should be entered. XXXX is the Start Address of the Code and YYYY is the End Address+1. The address parameters are mandatory.

5.2.2. MC Load. There are two forms of MC Load, one where the code is loaded into the same locations as it was saved from,the other loads the code to a different location. Both forms may have file names if required.
The syntax for the first form is L AAAAAA(RETURN) (RETURN). AAAAAA is the optional file name.
The syntax for the second form is L AAAAAA(RETURN) XXXX(RETURN) where XXXX is the Start Address where you wish the code to be loaded to.
Note that leading zeroes are not required in parameter values, parameters must be delimited by hyphens.

6.0. MACHINE CODE MONITOR

The machine code support routines are considerably improved over the basic machine and include:-

1) MEMORY EXAMINE/MODIFY
2) MEMORY SEARCH
3) MEMORY VERIFY (TABULATE)
4) BLOCK MOVE
5) REGISTER MODIFY
6) SAVE
7) LOAD
8) GO TO ADDRESS
9) BLOCK FILL

6.1. ENTRY TO MCM

Entry to the MCM is by two methods:
1) By answering the Command Prompt MCWDU with M.
2) Upon encountering a BRK ($00) during execution of a program.
In either case the result is the same. The return address is saved along with all the register and flags in a register map at $247 - 24D.(see Appendix 3 for allocation). The screen is cleared and the contents of the register map are displayed on the top (non scrolling) line of the screen. The MCM is now in command mode. The register/status information displayed is as at the moment the call to MCM occurred except the PC value has been adjusted to point at either, a return address if the call was by M, or the address of the byte following the $00 if by a break.
The information is:

PC Program Counter
FR Flags Register (status)
SP Stack Pointer
ACC Accumulator
XR X Register

6.2. USE OF MCM COMMANDS

6.2.1. R. Modify Register Contents.

This is a single command ie. no parameters. When entered it will cause the cursor to home (top LH corner)and clear the cursor line. By use of either the space bar or cursor left/right commands, position the cursor beneath the value in the status line you wish to change. Type in the new value. Move the cursor to any other values you wish and type those in. If you make an error backspace over it and retype the correct value. When complete press return and the status line is updated. Note that only the register map and status line are changed. Registers will be reloaded from the register map on exit from MCM using the G command.

6.2.2. MEMORY EXAMINE/MODIFY

This command has two options:
i) M (return) will pass control to the normal memory/examine routines using . or / as address or data commands. Any address changes will take place on the current line, as will data changes. Pressing RETURN in data mode will put the reset address/data information on a new line, scrolling up as required. Thus the consecutive address/data appear as a list. Whilst in address mode, X will return to MCM command mode, the normal G,L commands of the SY600 monitor have been retained for compatability.

ii) If M is followed by XX-YYYY-ZZZZ then a memory search will be carried out for the byte XX, from address YYYY to ZZZZ, each occurrence of XX will be listed on screen with its address. At the end of the operation the number of occurrences will be given.

6.2.3. V Verify Memory.

This is a 2 parameter command V XXXX - YYYY. The effect is to list memory contents, 8 bytes to a line,with the address of the first byte on each line, preceding the data, between the addresses XXXX to YYYY - 1. Note that pressing SPACE will slow the scroll rate.

6.2.4. B. Block Move.

This is a 3 Parameter command. The syntax is B XXXX-YYYY-ZZZZ where XXXX is the new start address and YYYY-ZZZZ are the Present Start and End addresses of the data block. There are no restrictions on the size of block or the direction of move, similarly there are no checks on either the Possibility or advisability of the move!

6.2.5. S Save machine code.

A two Parameter command, S XXXX - YYYY will save to tape the data block XXXX to YYYY, where XXXX is the start address and YYYY is the end address -1. The file may have a name as discussed in the Tape Facilities section. The format of the file is:

32 Sync characters ($16)
Sync Terminator.
File name or Null Padding to 6 chars.
End Addr Hi
End Addr Lo
Start Addr Hi
Start Addr Lo
Data Block

6.2.6 L Load a machine code tape.

L has two options:

L (Return) loads to address saved with tape
L XXXX saves block to address starting at XXXX.

6.2.7 G Go to address and execute.

G is a single letter command ie. no Parameters. If it is desired to commence execution at a different value to that shown in PC, modify the PC using R.

6.2.8 BLOCK FILL

This command F XX-YYYY-ZZZZ will fill memory from YYYY to ZZZZ with the byte XX.

6.3.0. COMMAND SYNTAX As noted above all Parameters should be delimited by - (hyphens) leading zeroes are not required.Each address should be +1, the required end Point.0

Parameters are rotated through as entered so if you make an error simply continue typing until the four characters entered are correct, these will be the Parameter data.

6.4.0 DEBUGGING

The MCM has been designed to facilitate debugging of Programs in a similar fashion to breakPoint insertion. If you insert a breakPoint in your code, on encountering it a software BRK will occur and the Processor will suspend execution, fetch the address contained in the IRQ VECTOR within WEMON ($FFFA,B,) which Points at MONBRK, and transfer control to that address. The BRK flag will be tested, if set, indicating a BRK, control then Passes to MCM. The return address and register information is saved in the register map. The full range of MCM facilities are now available. To resume execution use G. If the BRK flag was not set then control would Pass to UIRQVEC.

WARM RESTART

A warm restart facility may be implemented using the NMI. A Push to make switch should be connected between the NMI Pin on the 6502 and ground, and the NMI vector $22E,F,230 loaded with $4C 50 02. The following short routine should then be located in Page two RAM, starting at $250.

```
250   PLA   $68
251   PLA   $68
252   PLA   $68
253   CLI   $58
254   JMP   $4C    ;RECALL
```

This will cause a non conditional Jump to the command Prompt Point. No vectors will be changed, the stack Pointer is not initialised or screen cleared. If the above functions are required alter the Jump Point to VECLD.

7.0. RS 232C PRINTER USE

In order to use a Printer connected via the RS232C output it is necessary to initialise the relevent vectors to Prevent the delay and list routines outputing to the Printer. Whenever a Printout is needed POKE517,255 will initialise and set the save flag, thereafter all screen Prints will also appear on the RS232C line. To exit from this mode enter POKE517,0.

FUNCTION LOCATIONS

200 CURPSL Cursor Position low byte
201 CURCHR Character under cursor
202 TEMCHR Temporary character store
203 LDFLAG Load flag
204 ERRORFL FF = Run error
205 SCRSPD Scroll delay
212 CCFLAG Control C Flag
218 INVEC Input
21A OUTVEC Output vector
21C CCVEC Control C Vector
21E LDVEC Load vector
220 SAVVEC Save vector
222 MOVFLAG Move flag for insert routine
223 INSFLAG Insert/No Insert Flag
224 SKBFLAG FF next character returns basic command
225 CURPSH Cursor Position High byte
226 QUOFLAG Quotes flag
227 EDFLAG Cursor Position does not relate to input buffer so get line from screen
228 CREGFLAG Copy of Control Register contents (series 2 only) (mode flag in standard superboard)
229-22B USERVEC
22C CURSYM Cursor symbol
22D CURFL Cursor Flash Rate/Suppress
22E,F,30 NMIVEC
231,2,3 UIRQVEC User IRQ Vector
234,5,6,7,8,9 NAM1 File name to search for or write
23A-23F NAM2 File name found
241-247 Register map
248,9,A EXTVEC Basic command external vector
24B,C,D DISC Disc bootstrap vector
2HE PRFLAG Print/No Print flag
24F TPFLAG Tape on/off flag

Wemon Routine Entry Points and functions.

RESET. Resets all vectors and flags to default values, turns off tape, initialises the print driver, clears screen, waits for command.

VECLD. Enters above routine after the flag clearing etc, at point where vectors (218-221) are reloaded with default values, MC load flag ($E0) is cleared, tape turned off, screen cleared, cursor symbol restored, passes to command mode.

RECALL Command Mode Entry point, does not clear screen, prints command prompt, waits for command.

ACIAINIT. Initialises ACIA, tape off, clears screen, and homes cursor to top LH corner, or to bottom LH corner if in 25x25 mode (series 2 Superboard)

HOCUR. Puts cursor at top LH corner of screen.

SETCUR. Loads $F6 with 00$F7 from CURPSH, YREG from CURPSL.

WRITESCRN. General routine for putting characters to screen (not used for 25x25 mode in Series 2 Superboards), terminates with a jump to OUTPUT + 3. Characters handled include non printable (ie. cursor control etc.) calls scroll etc. as necessary, exits with next printing position in CURPSH and CURPSL. No registers need be saved before entry. Sub Routines available within WRITESCRN are:-

PUTCHR. Puts character under cursor onto screen.

WRITE. Does most that WRITESCRN does except it ends with an RTS. If WRITE is used no registers are saved (A,X,Y will be corrupted).
The characters should be in TEMCHR on entry, a JSR to SETCUR should be made before calling WRITE and a JSR to PUTCUR made after return from WRITE, to update CURPSH and CURPSL

CR. Executes a C return, note this does not include a Line Feed, (use CRLF for combined CR and LF). CR clears EDFLAG, MOVFLAG, QUOFLAG, INSFLAG, and restores CURSYM to $A1.
LNFD. Executes a line feed, scrolling as required.

VTAB. Ditto for cursor up 1 line

BELL Series 2 Superboards, with suitable external circuitry (a Pulse stretcher monostable controlling an astable driving a small speaker) will cause a signal to appear on the speaker. Executed by a read of $D800, this produces a negative strobe on Pin 14 of U20, to which the input of the M/stable should be connected. Note the CREG is a write-only register so no change of data will occur.

PUT1 Puts the character in the Accumulator to screen memory, the address being the current contents of $F6, $F7 and the Y Register.

EOP Tests F6, F7 for end of page, returns with carry set if F6, F7, Y are equal to last printing position + 1 on screen.

EOL Tests Y Register for equality to the low byte of the end of any screen line.

NEWLN Puts Y equal to first printing position -1 on present line, adjusts F7 if required.

PUTCUR Saves screen character at cursor position in CURCHR. Updates CURPSL and CURPSH.

SOP Tests F6, F7, Y for start of page. Returns carry clear if F6, F7 and Y point to first printing position on screen -1. Otherwise carry set.

SOL. Tests Y for start of line. Carry set if equal .

OLDLN. Puts Y equal to last printing position on previous line. Adjusts F7 if required.

START. Puts Y equal to start -1 of current line

SCROLL. Scrolls whole screen up 1 line. Delays if Space Bar Pressed. Returns with cursor position up dated.

SCROLL 1. As above but no test for Space Bar and cursor position not updated.

SCR1 Will scroll up portion of screen from the line whose absolute start address (low) is in Y Register and whose absolute start address high byte is in the X Register.

SCRDN. Scrolls screen down 1 line from cursor Position. Cursor Position remains as before. Start line absolute address should be in $F7 (high) and Y Register (low), $F6 should be $00.

KBRD. Main entry point of keyboard routine. Any keyboard controlled function command while in the wait loop will be executed. Character will not be echoed to screen. No registers need be saved.

Sub routines available within the KB routine are:-

1. EXTEND. Modifies the CHARGET routine.

PRERRL. Prints the Basic line whose number (in binary fixed point) is in $87, $88) Note a warm start should be executed following this routine.

TQUOTE. Toggles quotes flag and adjusts cursor symbol.

OPEN. Opens a blank line or cursor one Position. Cursor Position (or Position of line to be opened) should be in CURPSL and CURPSH.

GETST. Returns with absolute address -1 of start of a data line, regardless of whether it is less than or more than one line. Value is in F8 (low) F9 (high).

INSERT. Opens a 1 character space in a line of text at cursor Position. All text at and to the right of the cursor is moved right 1 space, will not open a line to a length greater than 72 characters. Scrolls up if on the bottom line and if the line goes beyond one full line. Scrolls down the remainder of screen if it is in the middle of screen and a new line is required.

DELETE. Does opposite of insert. Moves character under cursor and all characters (up to 2 lines, left 1 space) overwriting character Previously to left of cursor. Will not delete past the start of the line.

DELAY. Delay routine that gives approximately 1 m S delay for every unit value in Y Reg on entry. Y should not be 00 on entry or delay will be 255 mS.

DELAY5S is approximately a 5 second delay. In both these routines A,X,Y will be corrupted.

INVWR. Inverts contents of Accumulator and stores in keyboard latch. Returns with Accumulator unchanged.

RDCOL. Reads Kbd Port and returns with inverted value read in X.Acc unchanged.

RDCOLA. Reads Kbd and Port, inverts value and returns.

TSW. Inverts state of tape control leaves TPFL set if tape now on, clear if tape off.

SWAP. Series 2 Superboards only. Toggles video modes and adjusts INVEC accordingly. Note: sreen is cleared and Basic warm start executed. Screen editing not available in 25x25 though true delete, screen clear and flashing cursor are operative as are keyboard control functions (see 2.3.1).

FINTAPE. Turns off tape, clears all tape related flags LOAD, SAVE, TPFL, MCLOAD.

TAPOUT checks if space bar down, if so calls, FINTAPE then returns, if not, passes to....

TAP1. Puts a character in the Acc to the ACIA TX buffer when it is empty, then returns.

MSGOUT. Puts a message to screen, start of message relative to MSGS in X, message terminated with $00

```
MSGS + $00 = WEMON (c) 1981
     + $1C = $0D $0A Loading
     + $25 = $0D $0A Saving
     + $47 = List
```

HEXCHECK. Converts HEX character in accumulator to Hex digit. Returns with Hex character in accumulator or $80 if non Hex

MONPRINT. Prints out in ASCII the contents of FD, FC, FA as FD, FC, SPACE, SPACE, FA

NXTDIG. Prints out Hex, a byte pointed to by X relative to FA, ie. if X = 3 then a call to NXTDIG prints out the contents of FD.

CRLF. Executes a carriage return followed by a line feed.

SPC2. Prints 2 spaces

SPC1. Prints 1 space

HEXIT. Converts an ASCII character in the range 30-39, 41-46 to a single hex character and prints it.

ROTCHR. Rotates the lower nibble (4LSB's) of the Acc and the two adjacent bytes pointed to by X, relative to FA, left by 4 positions, ie. a call to ROTCHR with X=0, ACC=07 and FA, FB=0000 would leave on exit, FB=00, FA=07, ACC=07, X and Y are corrupted.

MINPUT. Gets Monitor input either from keyboard if MC load flag ($ED) is clear or ACIA if $E0 is FF.

FCHAR. Tests the keyboard for Space Bar down, returns with X=0 if not down, Z=0 if it is down.

MONBRK MCM entry point puts A to Acc. Pulls status, tests B4 (BRK flag) indirect jumps to UIRQVEC +1 if B4=0, If B4=1 X,Y,SP are stored in the register map, flags are pulled from stack and put in map, return address high and low are pulled from stack, adjusted, stored in register map.

PUTREG. Writes the contents of the register map across the top of screen

ZERPAR, zeroes $FA,$FB,$FC,$FD,$FE,$FF

GETPAR. Gets up to 3 16 BIT parameters into:

FA, FB, LSB, MSB, P1
FC, FD, LSB, MSB, P2
FE, FF, LSB, MSB, P3

On return $E1 contains the parameter count (number of parameters entered).

INCPTR. Increments $FC, and $FD if $FC = 00. $FC is then compared to $FE and $FD compared to $FF returns with Carry Flag set if $FC,FD equals $FE, EF, clear if not.

DELAY5S. Delays approximately 5 secs, A,Y,X corrupted.

COMPNAME. Compares two six character strings in NAM1, NAM2. Returns carry clear if equal, set if not.
Note: If one of the characters in NAM1 is a * then the test terminates and carry is clear to that point.

GETNAM1. Reads 6 characters from tape and stores in NAM2.

PUTNAM. Puts contents of NAM2 out to tape.

GETNAM. Gets 6 characters from keyboard and puts in NAM1. Note: On return, $E2 = number of characters up to but not including RETURN. Zeroes NAM1 and NAM2 first. Only accept characters $30 to $5A. Automatically returns after 6 characters or RETURN.

ADJPTR. Subtracts FC, FD from FE, FF and returns with the difference in F8, F9.

PRINTINIT. Initialises a 6520 at $8800.1,2,3 to operate as a parallel printer port with handshake.

PRINTOUT. Writes character in accumulator to PIA at $8800 configured as output port.

```
Note: PA0 - PA7 = D0 - D7
      CA2    =  DS   Active low
      RBQ    -  off line/empty active high
      PB1    -  ACK
      PB6    -  Busy   Active high
```

WEMON subroutine addresses

	S2	STAND	UK101
RESET	F000	F000	F000
VECLD	F049	F041	F041
RECALL	F063	F05D	F05D
ACIAINIT	F09A	F094	F094
HOCUR	F0C5	F0BA	F0B8
SETCUR	F0D9	F0CE	F0C6
WRITESCRN	F0E6	F0DB	F0D3
WRITE	F100	F0F5	F0ED
CR	F107	F0FC	F0F4
LNFD	F18A	F17F	F171
VTAB	F1AF	F1A4	F19A
BELL	F1EA	F1DD	F1D5
PUT1	F207	F1F7	F1EF
EOP	F224	F214	F20C
EOL	F234	F224	F21A
NEWLN	F23C	F22C	F222
PUTCUR	F243	F233	F229
SOP	F259	F249	F23F
SOL	F264	F254	F24A
OLDLN	F26A	F25A	F250
START	F276	F266	F25C
SCROLL	F27F	F26F	F265

	S2	STAND	UK101
SCR1	F2A7	F292	F28D
SCRDN	F2C2	F2AD	F2B0
KBRD	F3B2	F360	F369
EXTEND	F51E	F4C5	F4A7
PRERRL	F549	F4F0	F4D2
TQUOTE	F578	F51F	F501
OPEN	F5C7	F56E	F550
GETST	F5E3	F589	F56B
INSERT	F65F	F610	F5E7
DELETE	F723	F6CD	F6AB
DELAY	F785	F72F	F70D
INVWR	F78E	F738	F716
RDCOL	F796	F740	F71E
RDCOLA	F79F	F749	F727
TSW	F7A5	F74F	F72D
SWAP	F7BE	----	----
TAPOUT	F816	F781	F75F
TAP1	F81D	F788	F766
MSGOUT	F839	F7A4	F782
HEXCHECK	F883	F7EE	F7CC

	S2	STAND	UK101
MONPRINT	F89C	F807	F7E5
NXTDIG	F8AD	F818	F7F6
CRLF	F8BB	F826	F804
SPC2	F8C5	F830	F80E
SPC1	F8CA	F835	F813
HEXIT	F8CF	F83A	F818
ROTCHR	F8DC	F847	F825
MINPUT	F8EB	F856	F834
FCHAR	F8F5	F860	F83E
MONBRK	F900	F86B	F849
PUTREG	FA20	F98B	F969
GETPAR	FA8C	F9E5	F9D5
DELAY55	FC32	FB8B	FB7B
COMPNAME	FCDF	FC38	FC28
GETNAM1	FCF6	FC4F	FC3F
PUTNAM	FD2A	FC83	FC73
FINTAPE	F7FD	F768	F746
GETNAM	FD4D	FCA6	FC96
ADJPTR	FDEC	FD45	FD42
PRINTINIT	FEEF	FEEF	FEEF
PRINTOUT	FF04	FF04	FF04

WEMON 2 KEY BASIC COMMANDS

One of WEMON's many commands is the one key Basic command entry.
Below is a table of the words accessed by pressing 'SHIFT' and 'CTRL' at the same time, releasing, and then pressing the required key.

A	ASC	L	LEN	W	WAIT	8	RUN	LINEFEED ^
B	GOSUB	M	MID$(	X	ABS	9	TAN	
C	CHR$(	N	NEXT	Y	POS	0	RND	
D	DATA	O	ON	Z	STEP	,	ATN	
E	END	P	POKE	1	LOAD	.	INT	
F	FOR	Q	SQR	2	SAVE	/	DIM	
G	GOTO	R	RIGHT$(	3	REM	;	SIN	
H	READ	S	STR$(	4	RESTORE	-	LOG	
I	INPUT	T	THEN	5	NOT	:	EXP	
J	USR	U	NULL	6	TAB	rub	CLEAR	
K	PEEK	V	VAL	7	LIST	ret	RETURN	

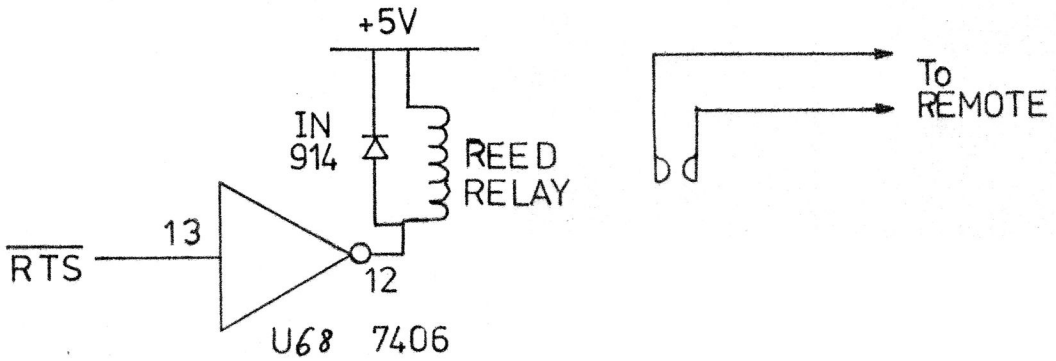


Fig 1 Automatic tape control.

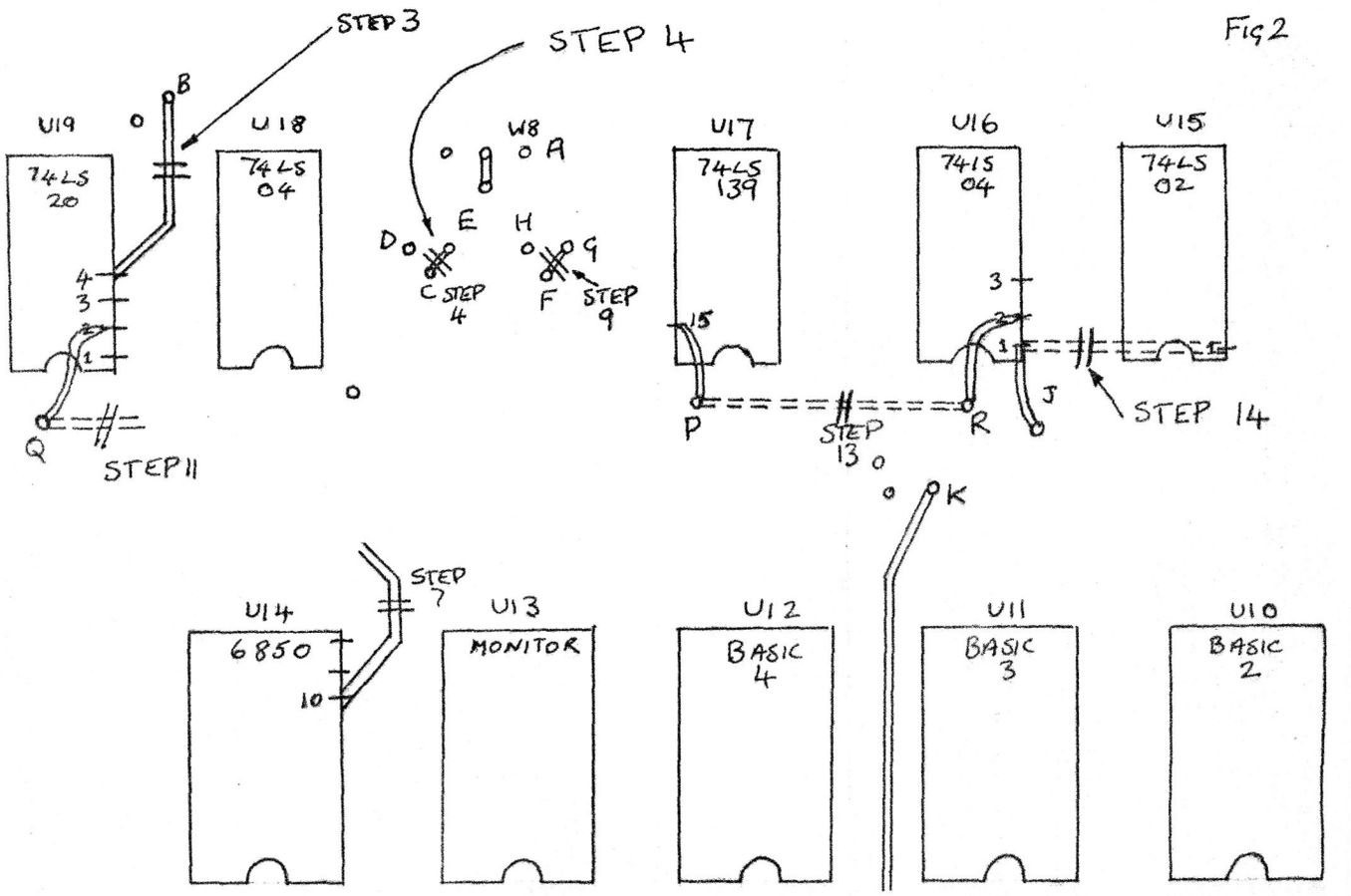
WEMON is supplied on an "as is" basis. Watford Electronics guarantee the functioning of the WEMON as per our specification and are unable to accept any liability for any loss or damage arising from its use, directly or indirectly. WEMON cannot be guaranteed to be bug free, although obviously to the best of our knowledge it is.

WEMON INSTALLATION PROCEDURE
UK101, Standard Superboard and Enhanced Series 1.

Carry out the following modifications with reference to Fig.2. It is recommended that for the interconnection steps, wire-wrap wire be used, passing the wire down through adjacent holes to where the connection is to be made. Where a suitable through plated hole does not exist to make the actual connection, the wire should be soldered to the connecting track, NOT to the IC pin.

NOTE:- If Point D is linked to C and H is linked to F on your board break the links and break the link between Pins 20 and 21 on U13 if it exists.

Step 1. Locate and remove the ACIA, U14.
Step 2. Locate and remove the UK101 monitor, U13.
Step 3. Cut topside track between Pin 4,U19, and Point B.
Step 4. Cut topside track between Points C and E.
Step 5. Link Point B to Point C.
Step 6. Link Pin 4, U19 to Point A.
Step 7. Locate and cut topside track from Pin 10, U14.
Step 8. Link Pin 10, U14 to Point R.
Step 9. Cut topside track between Points G and F.
Step 10. Link Point F to Point D.
Step 11. Cut underside track between Points Q and J.
Step 12. Link Point Q to Pin 1, U15.
Step 13. Cut underside track between Points P and R.
Step 14. Cut underside track between Pin 1, U15 and Pin 1, U16.
Step 15. Link Point J to Point K.
Step 16. Link Pin 3, U15 to Pin 15, U17.



Check carefully all modifications then replace the ACIA,U14 in it's socket and carefully insert the WEMON Eprom into the monitor socket observing the correct orientation in both cases.

For the technically minded the effect of the above modifications is to move the ACIA from its former address of $F000/F001 to $E000/E001.Additionally the CS for the monitor is expanded to enable the monitor socket from $F000 to $FFFF,ie. 4K. Finally the additional address line required,A11 is brought to the socket.

SERIES 2 SUPERBOARDS.

WEMON INSTALLATION.

Carry out the following steps with reference to Fig.2.It is reccomended that for the interconnection steps wire wrap wire be used,passing the wire down through adjacent holes to where the connection is to be made.Where a suitable through plated hole does not exist for making the actual soldered connection,make the connection to the adjoining track not to the IC pin.

Step 1. Locate and remove U14,6850.
Step 2. Locate and remove U13,Monitor ROM SYN 600.
Step 3. Cut track between Pin 4,U19 and Point B.
Step 4. Locate points C and D and cut underside track between them.
Step 5. Link point B to point C.
Step 6. Link Pin 4 of U19 to point A.
Step 7. Locate and cut topside track from pin 10 U14.
Step 8. Link pin 10 of U14 to pin 2 of U16.
Step 9. Locate and cut underside track between points G and F.
Step 10. Link point F to point D.
Step 11. Locate and cut topside track between points N and M
Step 12. Link point M to point P.
Step 13. Locate and cut underside track between point Q and point J.
Step 14. Link point Q to pin 1 U15.
Step 15. Locate and cut underside track between pin 2 of U16 and pin 15 of U17.
Step 16. Locate and cut underside track between pin 1 U15 and pin 1 U16.
Step 17. Link point J to point K.
Step 18. Link point H to pin 15 of U17.

Check carefully all modifications then replace the ACIA,U14 in it's socket and carefully insert the WEMON Eprom into the monitor socket observing the correct orientation in both cases.

For the technically minded the effect of the above modifications is to move the ACIA from its former address of $F000/F001 to $E000/E001.Additionally the CS for the monitor is expanded to enable the monitor socket from $F000 to $FFFF,ie. 4K. Finally the additional address line required,A11 is brought to the socket.

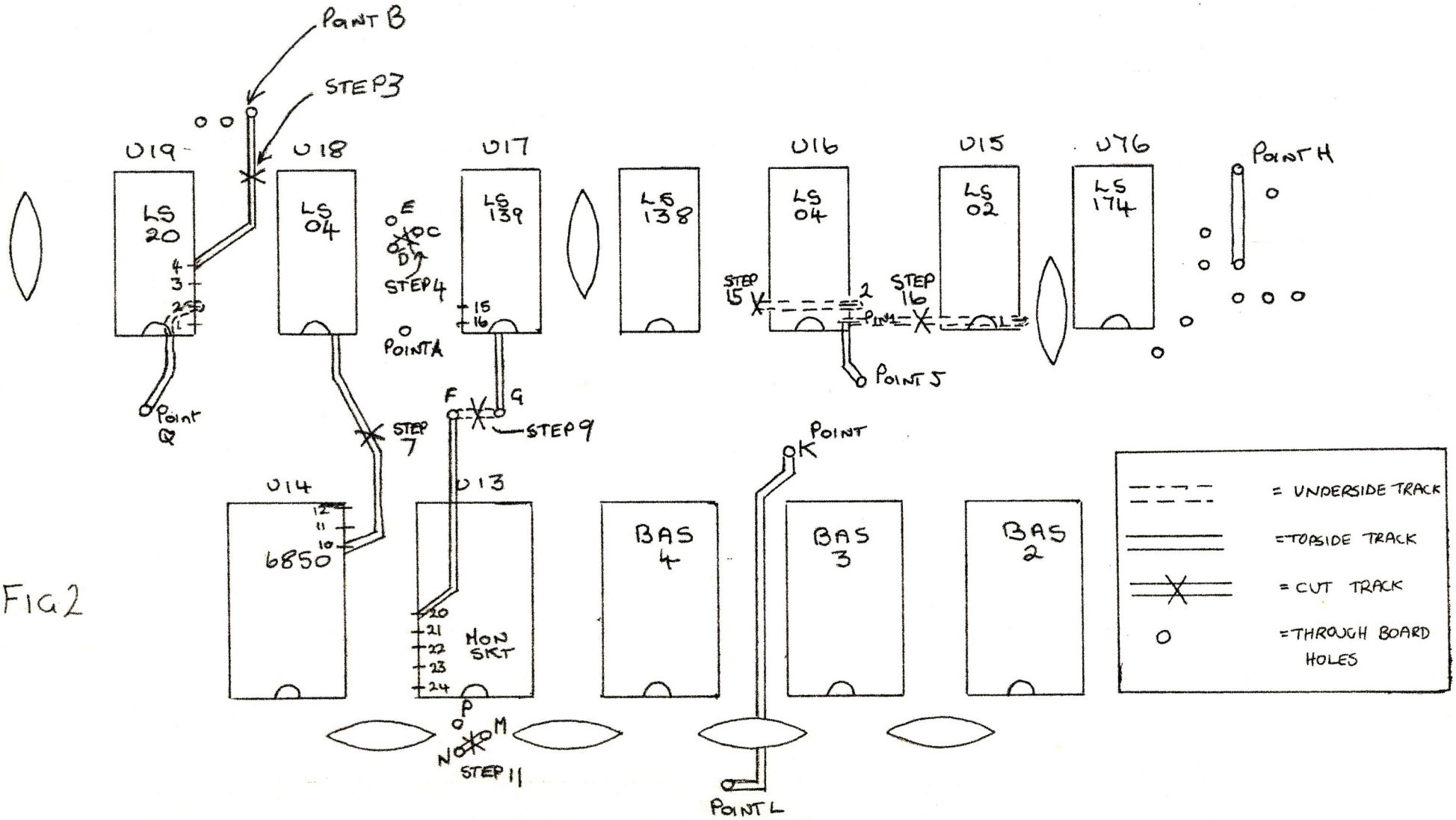

POINT B
STEP 3
U19
U18
U17
U16
U15
U14
U13
LS 20
LS 04
LS 139
LS 138
LS 04
LS 02
LS 174
POINT H
E
C
D
STEP 4
POINT A
STEP 15
STEP 16
POINT J
POINT Q
F
G
STEP 7
STEP 9
POINT K
6850
MON SKT
BAS 4
BAS 3
BAS 2
P
M
N
STEP 11
POINT L
= UNDERSIDE TRACK
= TOPSIDE TRACK
= CUT TRACK
= THROUGH BOARD HOLES

FIG 2