# A BBC EPROM Programmer

Angus Duggan

1st April 2001

## 1 Introduction

This document describes an EPROM programmer I designed and built circa 1985, for use with a BBC microcomputer.

## 2 Using the programmer

The EPROM programmer should be plugged into the BBC computer's user port. The switch should always be set to the read position before starting the software. The software should be run from disc; this will show a menu allowing ROM images to be loaded, saved, programmed, and verified, and operating system commands to be issued.

## 3 Hardware

I "designed" the EPROM programmer by adapting an existing design, published in the Beebug magazine. My alterations were to add a second address latch and a state machine to select the read/write functions and latches. This allowed the programmer to be driven with just the user port, instead of using the printer port as well (as the Beebug design had done). My knowledge of hardware was just adequate for the task; the board works fine, but more experienced hardware designers will probably find flaws in the design.

The circuit diagram is shown in figure 1. Note that the 27128 EPROM is shown with the pins mirrored left for right, because the socket was mounted on the reverse (track) side of the board I built the programmer on. Also, for cleanliness in the layout, I have shown the +5v and 0v pins at the wrong ends of the edge connector; be careful when laying out a board not to copy this order onto the board.

The circuitry at the bottom selects the programming functions. The CB2 line from the user VIA (6522 interface adapter) is used to control the programmer, with the CB1 line providing feedback. The low and high address latches are loaded first, and then the output enable or program enable are driven low to read or write the EPROM. Table 2 shows the state machine implemented by this hardware. The outputs from the circuit are CB1 (used for feedback to the BBC so it can detect which state the programmer is in), Enable (set low to make the EPROM read or write data from or to the data bus), $CK_{LO}$ (used to set the low order address latch) and $CK_{HI}$ (used to set the high order address latch). The CB2 line controls the state machine; it drives the clock line of the 74LS74 flip-flop, triggering a new state when it is set low and then high again. The 74LS374 latches are also edge-triggered, so the latches will only capture data from the data bus when transitioning from low to high. There are only three states in

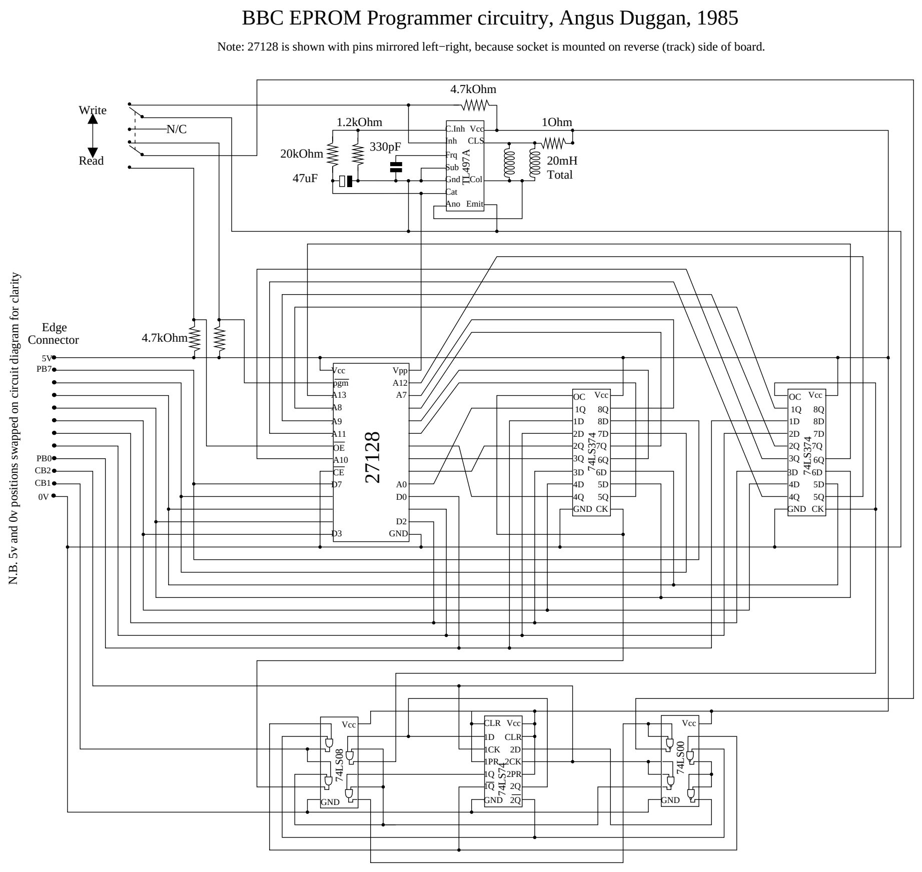
BBC EPROM Programmer circuitry, Angus Duggan, 1985
Note: 27128 is shown with pins mirrored left−right, because socket is mounted on reverse (track) side of board.
N.B. 5v and 0v positions swapped on circuit diagram for clarity
Write
N/C
Read
4.7kOhm
1.2kOhm
330pF
20kOhm
47uF
1Ohm
20mH
Total
L497A
Edge
Connector
5V
PB7
PB0
CB2
CB1
0V
4.7kOhm
27128
74LS374
74LS374
74LS08
74LS74
74LS00

Figure 1: Circuit diagram of EPROM programmer

the cycle; the remaining possible state (when $Q_1$ and $Q_2$ are both 1) will transition to the R/W state, and then the state cycle will repeat. When the software initialises the programmer, it repeatedly clocks the state until the CB1 line changes.

I implemented the design using vero-board, managing to fit it into a 37 by 36 section of 0.1 inch pitch board. Photos may be found on my web site at `http://knackered.org/~angus/beeb/`.

## 4 Software

The software to drive the EPROM programmer is a 6502 assembly program. It performs the functions of loading ROM images, saving ROM images, reading, writing, and verifying EPROMs. The 6502 assembler source is shown below, in the format for my own assembler. Some assembler directives (EQB, EQA, EQD) may be unfamiliar. Their meanings are:

| State | $Q_1$ | $Q_2$ | CB2 | CB1 | Enable | $CK_{LO}$ | $CK_{HI}$ | Next state |
|---|---|---|---|---|---|---|---|---|
| Latch LO | 0 | 0 | 1 | 0 | 1 | 0 | 0 | |
| | 0 | 0 | 0 | 0 | 1 | 1 | 0 | Latch HI |
| Latch HI | 0 | 1 | 1 | 1 | 1 | 0 | 0 | |
| | 0 | 1 | 0 | 1 | 1 | 0 | 1 | R/W |
| R/W | 1 | 0 | 1 | 1 | 1 | 0 | 0 | |
| | 1 | 0 | 0 | 1 | 0 | 0 | 0 | Latch LO |

Figure 2: State machine implemented by 74LS00, 74LS08 and 74LS74

**EQA** Inserts an ASCII string at the current program counter position, with no termination nor preceding length byte.

**EQB** Insert a byte at the current program counter position. A list of bytes may be specified, and constant arithmetic expressions may be used.

**EQW** Insert a two-byte word at the current program counter position, low byte first. A list of words may be specified, and constant arithmetic expressions may be used.

**EQD** Insert a four-byte double word at the current program counter position, least significant byte first. A list of double words may be specified, and constant arithmetic expressions may be used.

**ORG** Sets the default origin (initial program counter) for the code. If an execution address is not also specified, the execution address is set to the origin

**<, >** Use the low or high (second) byte of an expression evaluating to an address.

```
\...................*...................*...................*............<
brkmsg=&FD
escape=&FF          \ escape flag
brkvec=&202         \ BRK vector
irqvec=&206         \ interrupt vector 2
orb=&FE60           \ User VIA registers
irb=&FE60
ddrb=&FE62
t2l=&FE68
t2h=&FE69
acr=&FE6B
pcr=&FE6C
ifr=&FE6D
ier=&FE6E
osfile=&FFDD        \ OS addresses
osnewl=&FFE7
osrdch=&FFE0
oswrch=&FFEE
osword=&FFF1
osbyte=&FFF4
oscli=&FFF7

zpwork=&70          \ work space for PRINT routine
length=&74          \ length of EPROM (high byte)
eaddr=&75           \ address in EPROM (high byte)
baddr=&76           \ address in buffer (page address)
middle=&78          \ address of loop routine

buffer=&3C00        \ buffer for ROMS &3C00-&7C00

delay=50000         \ 50ms delay in 1MHz clock cycles

menux=4             \ menu indentation
menuy=5             \ menu position
menu=7              \ number of menu options
filey=14            \ menuy+menu+2 = filename window position
selecty=13          \ filey-1 = cursor select position
```

```
commy=16            \ filey+2 = command window position
commx=0             \ command window indent
width=40            \ command window width

                    org &3000

>EPROG              \ eprom programmer
                    jsr PRINT
                    eqw P2-P1
P1                  eqb 22, 7, 134, 157, 129, 141, 31, 10, 0
                    eqa "AJCD Eprom Programmer"
                    eqb 31, 0, 1, 134, 157, 129, 141, 31, 10, 1
                    eqa "AJCD Eprom Programmer"
                    eqb 31, menux, menuy
                    eqa "1  -  Load buffer from file"
                    eqb 13, 129, 32, 134, 31, menux, menuy+1
                    eqa "2  -  Save buffer to file"
                    eqb 13, 129, 32, 134, 31, menux, menuy+2
                    eqa "3  -  Copy EPROM to buffer"
                    eqb 13, 129, 32, 134, 31, menux, menuy+3
                    eqa "4  -  Program EPROM from buffer"
                    eqb 13, 129, 32, 134, 31, menux, menuy+4
                    eqa "5  -  Compare EPROM with buffer"
                    eqb 13, 129, 32, 134, 31, menux, menuy+5
                    eqa "6  -  Check EPROM is blank"
                    eqb 13, 129, 32, 134, 31, menux, menuy+6
                    eqa "*  -  Issue MOS command"
                    eqb 13, 129, 32, 134, 31, menux-1, menuy+7
                    eqa "ESC -  Exit from program"
                    eqb 31, 3, filey, 131, 157, 132
                    eqa "Filename"
                    eqb 31, 35, filey, 156
P2                  lda #234
                    ldx #0
                    ldy #255
                    jsr osbyte
                    cpx #0
                    beq NOTUBE
                    jsr PRINT
                    eqw P4-P3
P3                  eqb 31, commx, commy
                    eqa "Please turn your TUBE off and re-run"
                    eqb 13, 10
P4                  jmp ABORT

NOTUBE              sei                 \ continue with setting up
                    lda irqvec
                    sta OLDIRQ
                    lda #<IRQ
                    sta irqvec
                    lda irqvec+1
                    sta OLDIRQ+1
                    lda #>IRQ
                    sta irqvec+1
                    lda #&B0
                    sta ier             \ enable CB1 & t2 interrupts
                    lda #0
                    sta acr             \ disable input latching etc
                    lda pcr
                    ora #&F0
                    sta pcr             \ set CB2 to high output
                    cli
                    lda brkvec
                    sta OLDBRK
                    lda #<BRKERR
                    sta brkvec
                    lda brkvec+1
                    sta OLDBRK+1
                    lda #>BRKERR
                    sta brkvec+1
                    tsx
                    stx STACK
MAIN                jsr ESCAPE          \ ignore escape
                    jsr PRINT           \ restore cursor position
                    eqw P8-P7
P7                  eqb 26, 31, 1, menuy
                    eqb 32, 10, 8, 32, 10, 8, 32, 10, 8, 32, 10, 8
                    eqb 32, 10, 8, 32, 10, 8, 32, 10, 8
                    eqb 31, menux, selecty
P8                  lda #21
                    ldx #0
                    jsr osbyte          \ flush keyboard buffer
                    jsr osrdch          \ main menu loop
                    bcc NOABORT         \ error condition?
                    cmp #27             \ escape?
                    bne ABORT
                    lda #126
                    jsr osbyte
ABORT               lda OLDIRQ+1
                    beq R1
                    sei
                    sta irqvec+1
                    lda OLDIRQ
                    sta irqvec
                    lda OLDBRK
                    sta brkvec
                    lda OLDBRK
                    sta brkvec+1
                    cli
R1                  rts
NOABORT             ldx #menu
CHKOPT              dex
                    bmi MAIN
                    cmp OPTIONS, X
                    bne CHKOPT
                    lda #31
                    jsr oswrch          \ indicate which option selected
                    lda #1
                    jsr oswrch
                    txa
                    clc
                    adc #menuy
                    jsr oswrch
                    lda #157
                    jsr oswrch
                    dex
                    bpl SAVE

\ Load file to buffer
                    jsr FILEWIND        \ filename window
                    lda #0              \ filename input line
                    ldx #<FILINE
                    ldy #>FILINE
                    jsr osword          \ read a line from input
                    bcs KILLOAD         \ input error
                    jsr COMMWIND
                    jsr PRINT
                    eqw P12-P11
P11                 eqb 31, width/2-7, 0
                    eqa "Loading..."
P12                 ldx #15
LI                  lda LOADINFO, X
                    sta BLOCK+2, X      \ clear out block
                    dex
                    bpl LI
                    ldx #<BLOCK
                    ldy #>BLOCK
                    lda #&FF
                    jsr osfile
                    jmp DONE2
KILLOAD             jmp MAIN

SAVE                dex
                    bpl COPY
\ Save file to buffer
                    jsr FILEWIND        \ filename window
                    lda #0              \ filename input line
                    ldx #<FILINE
                    ldy #>FILINE
                    jsr osword          \ read a line from input
                    bcs KILSAVE         \ input error
                    jsr COMMWIND
                    jsr PRINT
                    eqw P14-P13
P13                 eqb 31, width/2-7, 0
                    eqa "Saving..."
P14                 ldx #15
SI                  lda SAVEINFO, X
                    sta BLOCK+2, X      \ clear out block
                    dex
                    bpl SI
                    ldx #<BLOCK
                    ldy #>BLOCK
                    lda #0
                    jsr osfile
                    jmp DONE2
KILSAVE             jmp MAIN

COPY                dex
                    bpl PROGRAM
\ Copy Eprom to buffer
                    jsr FILEWIND        \ Delete filename
                    jsr READY
                    bcs KILCOPY         \ Error
                    jsr PRINT
                    eqw P16-P15
P15                 eqb 31, width/2-8, 4
                    eqa "Copying"
P16                 ldx #<LOOPCOPY
                    ldy #>LOOPCOPY
                    jsr LOOP
                    jmp DONE
KILCOPY             jmp MAIN

LOOPCOPY            jsr READ
                    sta (baddr), Y
                    clc
                    rts

PROGRAM             dex
                    bpl VERIFY
\ Program Eprom from buffer
                    jsr WRITEY
                    bcs KILPROG
                    jsr PRINT
                    eqw P26-P25
P25                 eqb 31, width/2-10, 4
                    eqa "Programming"
P26                 ldx #<LOOPPROG
                    ldy #>LOOPPROG
                    jsr LOOP
                    jmp DONE
KILPROG             jmp MAIN

LOOPPROG            lda #&FF
                    sta ddrb
                    sta TIMER
                    lda (baddr), Y
                    sta orb
                    lda #&DF            \ CB2 = 0
                    and pcr
                    sta pcr
                    lda #<delay         \ 50 ms delay
                    sta t2l
                    lda #>delay
                    sta t2h
WAIT                lda TIMER
                    bne WAIT
                    lda #&F0            \ CB2 = 1
                    ora pcr
                    sta pcr
                    clc
                    rts

VERIFY              dex
                    bpl CHECK
\ Verify eprom against buffer
                    jsr READY
                    bcs KILPROG
                    jsr PRINT
                    eqw P28-P27
P27                 eqb 31, width/2-9, 4
                    eqa "Verifying"
P28                 ldx #<LOOPVFY
                    ldy #>LOOPVFY
                    jsr LOOP
                    jmp DONE
KILVFY              jmp MAIN

LOOPVFY             jsr READ
                    cmp (baddr), Y
                    clc
                    beq R4
                    sec
R4                  rts

CHECK               dex
                    bpl MOSCALL
\ Check blank eprom
                    jsr READY
                    bcs KILPROG
                    jsr PRINT
                    eqw P30-P29
P29                 eqb 31, width/2-8, 4
                    eqa "Checking"
P30                 ldx #<LOOPCHK
                    ldy #>LOOPCHK
                    jsr LOOP
                    jmp DONE
KILCHK              jmp MAIN

LOOPCHK             jsr READ
                    cmp #&FF
                    clc
                    beq R6
                    sec
R6                  rts

MOSCALL             \ Operating system call
                    jsr COMMWIND        \ set up command window
                    lda #'*'
                    jsr oswrch          \ indicate input required
                    lda #0              \ OS input line
                    ldx #<OSLINE
                    ldy #>OSLINE
                    jsr osword          \ read a line from input
                    bcs KILLOSC         \ input error
                    lda #14
                    jsr oswrch          \ page mode on
                    ldx #<INPUT
                    ldy #>INPUT
                    jsr oscli
                    lda #15
                    jsr oswrch
KILLOSC             jmp MAIN

LOOP                stx middle          \ Common loop for prog etc.
                    sty middle+1
                    jsr PRINT
                    eqw P32-P31
P31                 eqa "...&  "
P32                 ldy #0
RECHECK             jsr ESCAPE          \ check escape key
                    bcs QUITLOOP
                    lda #8              \ get to right place
                    jsr oswrch
                    jsr oswrch
                    lda eaddr
                    jsr HEX             \ print high address
REPEAT              tya                 \ print low address
                    jsr HEX
                    lda #8              \ move back into position
                    jsr oswrch
                    jsr oswrch
                    lda #&FF
                    sta ddrb
                    sty orb             \ load low address
                    jsr TOGGLE
                    lda eaddr           \ load high address
                    sta orb
```

```
                    jsr TOGGLE
                    ldx CB1
                    jsr MIDDLE          \ do centre routine
                    bcs NOTEQUAL
                    cpx CB1             \ check CB1 interrupt
                    bne PROGOK
                    jmp PROGERROR
PROGOK              iny                 \ increment address
                    bne REPEAT
                    inc eaddr
                    inc baddr+1
                    dec length
                    bne RECHECK
                    clc
QUITLOOP            rts
NOTEQUAL            jsr PRINT
                    eqw P34-P33
P33                 eqb 7, 31, width/2-8, 5
                    eqa "Comparison error"
P34                 sec
                    rts

MIDDLE              jmp (middle)        \ indirect

DONE                bcs P37
                    jsr PRINT           \ Print Message
                    eqw DONE2-P35
P35                 eqb 31, width/2-2, 5
DONE2               jsr PRINT
                    eqw P37-P36
P36                 eqa "Done"
P37                 jmp MAIN

HEX                 pha                 \ print two hex digits
                    lsr A               \ get left digit
                    lsr A
                    lsr A
                    lsr A
                    jsr DIGIT
                    pla
                    and #&F             \ get right digit
DIGIT               cmp #10
                    bcc NUMBER
                    adc #6
NUMBER              adc #48             \ add digit base
                    jmp oswrch

READY               jsr COMMWIND        \ get ready to READ
                    jsr PRINT
                    eqw P18-P17
P17                 eqa " Set the programmer switch to"
                    eqb 130
                    eqa "READ,"
P18                 jmp PREPARE

READ                lda #0              \ read byte from programmer
                    sta ddrb
                    lda #&DF            \ CB2 = 0
                    and pcr
                    sta pcr
                    lda irb
                    pha
                    lda #&F0            \ CB2 = 1
                    ora pcr
                    sta pcr
                    pla
                    rts

WRITEY              jsr COMMWIND        \ get ready to WRITE
                    jsr PRINT
                    eqw P20-P19
P19                 eqa " Set the programmer switch to"
                    eqb 129
                    eqa "WRITE,"
P20                 \ jmp PREPARE

PREPARE             jsr PRINT
                    eqw P22-P21
P21                 eqb 31, 1, 1
                    eqa "then select the EPROM type -"
                    eqb 13, 10, 132, 31, menux-1, 2, 135
                    eqa "1  -  2764"
                    eqb 13, 10, 132, 31, menux-1, 3, 135
                    eqa "2  -  27128"
                    eqb 13, 10
P22                 lda #&80
                    sta eaddr           \ EPROM start at &8000
                    lda #&40
                    sta length          \ default length = &4000
                    lda #&3C
                    sta baddr+1         \ Buffer start at &3C00
                    lda #0
                    sta baddr
GETLEN              jsr osrdch
                    bcs R5
                    cmp #'2'
                    beq LENOK
                    cmp #'1'
                    bne GETLEN
                    lsr length
                    lda #11
                    jsr oswrch
LENOK               lda #11
                    jsr oswrch
                    lda #9
                    jsr oswrch
                    lda #157
                    jsr oswrch
                    lda #&FF            \ make outputs safe
                    sta ddrb
                    sta orb
                    ldx CB1             \ take old count
                    jsr TOGGLE
                    cpx CB1             \ once
                    bne QUITPREP
                    jsr TOGGLE
                    cpx CB1             \ twice
                    bne QUITPREP
                    jsr TOGGLE
                    cpx CB1             \ three times...
                    beq PROGERROR
QUITPREP            clc
R5                  rts
PROGERROR           jsr PRINT           \ No response from programmer
                    eqw P24-P23
P23                 eqb 7, 31, width/2-12, 5, 136
                    eqa "EPROM Programmer Error"
                    eqb 13, 10
P24                 sec
                    rts

TOGGLE              lda #&DF            \ make CB2 go low then high
                    and pcr
                    sta pcr             \ low
                    ora #&F0
                    sta pcr             \ high
                    rts

ESCAPE              lda escape          \ test & reset escape condition
                    clc
                    bpl NOESC
                    lda #126
                    jsr osbyte
                    sec                 \ carry set if escape detected
NOESC               rts

PRINT               pla                 \ print in-line codes
                    sta zpwork
                    pla                 \ two byte size
                    sta zpwork+1
                    jsr GETIND
                    sta zpwork+2
                    jsr GETIND
                    sta zpwork+3
PRLOOP              jsr GETIND
                    jsr oswrch
                    lda zpwork+2
                    bne DECLOW
                    dec zpwork+3
DECLOW              dec zpwork+2
                    bne PRLOOP
                    lda zpwork+3
                    bne PRLOOP
                    jsr INCADR
                    jmp (zpwork)

GETIND              ldy #1              \ get data from indirect address
                    lda (zpwork), Y
INCADR              inc zpwork
```

```
                    bne R2
                    inc zpwork+1
R2                  rts

BRKERR              ldx STACK           \ action taken on BRK
                    txs
                    jsr PRINT
                    eqw P39-P38
P38                 eqb 15, 13, 10, 10, 7
                    eqa "          OS Error : "
P39                 ldy #1
BRKMSG              lda (brkmsg), Y
                    beq BRKQUIT
                    jsr oswrch
                    iny
                    bne BRKMSG
BRKQUIT             jsr osnewl
                    jmp MAIN

IRQ                 pha                 \ interrupt routine
                    lda ifr             \ test interrupt condition
                    and #32             \ timeout ?
                    beq AGAIN
                    lda t2l             \ clear interrupt condition
                    inc TIMER
AGAIN               lda ifr
                    and #16             \ CB1 ?
                    beq CHAIN
                    lda orb             \ clear interrupt condition
                    inc CB1
CHAIN               pla
                    jmp (OLDIRQ)        \ goto next interrupt handler

COMMWIND            jsr PRINT           \ setup command window
                    eqw P6-P5
P5                  eqb 28, commx, 24, commx+width-1, commy, 12
P6                  rts

FILEWIND            jsr PRINT           \ setup filename window
                    eqw P10-P9
P9                  eqb 28, 15, filey, 34, filey, 12
P10                 rts

\ Data area follows...

OPTIONS             eqb '1', '2', '3', '4', '5', '6', '*'

OSLINE              eqw INPUT           \ OSWORD 0 block for commands
                    eqb 255, 32, 127

FILINE              eqw INPUT           \ OSWORD 0 block for filenames
                    eqb 19, 32, 127

CB1                 eqb 0               \ counter for CB1 interrupts
TIMER               eqb 0               \ counter for t2 timeouts
OLDIRQ              eqw 0               \ Old IRQV2

STACK               eqb 0               \ stack pointer
OLDBRK              eqw 0               \ Old brkvec

BLOCK               eqw INPUT           \ Load file parameter block
                    eqd 0
                    eqd 0
                    eqd 0
                    eqd 0

LOADINFO            eqd buffer
                    eqd 0
                    eqd 0
                    eqd 0

SAVEINFO            eqd &FFFF8000
                    eqd &FFFF8000
                    eqd buffer
                    eqd buffer+&4000

INPUT               \ input buffer
```