This article deals with a very sophisticated issue: programming the FlashROM. You will need thorough understanding of how the chip works. All of the code provided is commented well, but you still have to know what you are doing. You won't physically destroy your ROM chip or the cart, though.

The code originates from my FMEEPROMPP utility. The ordinal numbers and byte codes refer to the FlashROM command sequence steps. PA and PD stand for the address and data to be programmed. All of this is explained in the FlashROM data sheet.

Regarding flashing the chip, read especially the programming algorithm section and also see the waveform tables for timing information. This code appears to work, but especially the programming routine gets so complicated, there may be errors, so take care!

You can find the print routine in string manipulation routines.

```
; Retro Replay FlashROM operating routines by FMan/Tropyx

; these are stripped-down example routines to program
; data of arbitrary length at arbitrary ROM location,
; and perform Block Erase and Chip Erase operations

; WARNING! These routines, as they are presented here,
; have not been tested! To use them, provide a main
; program that does the initialization specified and
; accommodate the given code to suit your needs, or better
; yet: fully understand how it works and write your own!

    BANK0 = 0
    BANK1 = 8
    BANK2 = $10
    BANK3 = $18
    BANK4 = $80
    BANK5 = $88
    BANK6 = $90
    BANK7 = $98

    SECCFG = $42    ; $DE01 base value
    A16BIT = $20    ; value for A16 in $DE01

    tempa = $80    ; temporary variable definitions in
    tempb = $81    ; zero-page area used by BASIC and
    tempc = $82    ; free for us in an MC program

    addrlo = $58    ; beginning of data to flash
    addrhi = $59
    maxlo = $5a    ; end address in C64's RAM
    maxhi = $5b
    secsel = $5e    ; currently selected bank

    srclo = $60    ; used internally by the
    srchi = $61    ; programming routine
    dstlo = $62
```

```
    dsthi = $63
    buflo = $64    ; buffer used by the
    bufhi = $65    ; programming routine

; subroutine that sends the beginning of a Command sequence

RRCmd   sei            ; note that this routines disales
    lda #$13        ; interrupts but does not enable
    sta $de00        ; them (operation is unfinished)
    lda #$aa
    sta $9555        ; 1st: 555 - AA
    ldx #$b
    stx $de00
    lsr
    sta $8aaa        ; 2nd: 2AA - 55
    lda #$13
    sta $de00
    sty $9555        ; 3rd: 555 - input reg Y
    lda #3
    sta $de00
    rts

; this is a table containing $DE01 values for all 16 ROM windows

bank    !byte SECCFG+BANK0,SECCFG+BANK1,SECCFG+BANK2,SECCFG+BANK3
    !byte SECCFG+BANK4,SECCFG+BANK5,SECCFG+BANK6,SECCFG+BANK7
    !byte SECCFG+A16BIT+BANK0,SECCFG+A16BIT+BANK1,SECCFG+A16BIT+BANK2
    !byte SECCFG+A16BIT+BANK3,SECCFG+A16BIT+BANK4,SECCFG+A16BIT+BANK5
    !byte SECCFG+A16BIT+BANK6,SECCFG+A16BIT+BANK7

; this routine will correctly handle wrapping over RR ROM banks
; when address $A000 is reached, to program the FlashROM starting
; from any address and proceeding to do so per custom length

; before calling this routine, set addr and max accordingly,
; set dsthi and dstlo to the starting address in the range of
; $8000-$9FFF inside the currently selected ROM window, and
; to select the correct ROM bank out of the 16 possible 8K
; ones, preset the secsel variable that is used as an index
; to the bank configuration value table above to select them

Program jsr RRReset        ; reset the ROM for safety
    lda addrlo        ; copy starting address of data
    sta srclo        ; to program in the C64's RAM
    lda addrhi
    sta srchi
    lda #11
    sta $d011        ; turn screen off
    ldx secsel
    lda bank,x
    sta tempa        ; current bank byte
```

```
    stx tempc        ; bank index
    sei
Progk   lda #$36        ; switch BASIC off, to be able
    sta 1            ; to load data from RAM under it
    ldy #0
Progh   lda (srclo),y        ; copy a pageful of data to the
    sta (buflo),y        ; buffer (initialize it to $600)
    iny
    bne Progh
    lda #$37
    sta 1
Proga   ldy #$a0        ; 3rd: 555 - A0
    jsr RRCmd
    lda tempa
    sta $de01
    ldy #0
    lda (buflo),y
    sta (dstlo),y        ; 4th: PA - PD
    and #$80
    sta tempb        ; store bit 7 for comparison
Progb   lda (dstlo),y
    tax
    and #$80        ; see if memory reads bit 7
    cmp tempb
    beq Progc        ; Program operation finished
    txa
    and #$20        ; check error bit
    beq Progb
    lda #0
    sta $de00
    cli
    lda #<errstr
    sta dst
    lda #>errstr
    bne Progd        ; unconditional - error msg
Progc   lda srchi
    cmp maxhi
    bne Progj
    lda buflo
    cmp maxlo
    beq Progf
Progj   inc dstlo        ; increment destination address
    bne Proge
    inc dsthi
    lda dsthi
    cmp #$a0
    bne Proge
    lda #$80        ; next bank
    sta dsthi
    ldx tempc
    inx
```

```
    lda bank,x
    sta tempa
    stx tempc
Proge   inc buflo        ; increment source address
    bne Proga
    inc srchi
    bne Progk        ; unconditional
Progf   lda #32
    ldx #0
Progg   sta $600,x        ; clean up screen (buffer page)
    inx
    bne Progg
Done    lda #0
    sta $de00
    cli
    lda #<donestr        ; print "Done"
    sta dst
    lda #>donestr
Progd   sta dst+1
    jsr print        ; bring in this and dst definition
    lda #27
    sta $d011        ; screen back on

; this routine sends a Read/Reset command to the FlashROM

RRReset ldy #$f0        ; 3rd: F0 - and this is enough
    jsr RRCmd
    lda #2            ; set the cart to off state
    sta $de00
    cli
    rts

; this routine performs a Block Erase operation on one of the eight
; 16K internal blocks of which the M29F010B chip's 128K is made

; select a 8K ROM window that falls within the desired 16K region
; to be erased - for example, if you have ROM bank 10 or 11 selected,
; both will result in the ROM region $14000-17FFF to be erased

; if you wish to change the operation of the routine so that
; the secsel variable is interpreted to directly stand for the
; 16K FlashROM block, uncomment the included asl instruction

BErase  ldy #$80        ; 3rd: 555 - 80
    jsr RRCmd
    ldx #SECCFG+BANK2
    stx $de01
    lda #$aa
    sta $9555        ; 4th: 555 - AA
    ldy #SECCFG+BANK1
    sty $de01
```

```
    lsr
    sta $8aaa        ; 5th: 2AA - 55
    lda secsel
;   asl            ; see note above
    tax
    lda bank,x        ; set bank corresponding with
    sta $de01        ; currently selected ROM window
    lda #$30
    sta $8000        ; 6th: BA - 30
BErasea ldx #80
BErased dex            ; wait for a little while
    bpl BErased
BEraseb lda $8000        ; wait for Erase to finish
    and #$a0
    beq BEraseb
    bmi Done
BErasec inc $d020        ; chip failure
    jmp BErasec

; this routine performs Chip Erase - something discouraged by Jens,
; but useful if you know you are going to be flashing the entire ROM

CErase  ldy #$80        ; 3rd: 555 - 80
    jsr RRCmd
    ldx #SECCFG+BANK2
    stx $de01
    lda #$aa
    sta $9555        ; 4th: 555 - AA
    ldy #SECCFG+BANK1
    sty $de01
    lsr
    sta $8aaa        ; 5th: 2AA - 55
    stx $de01
    lda #$10
    sta $9555        ; 6th: 555 - 10
    bne BErasea

errstr  !pet 13,"programming error! aborted. ",0
donestr !pet 13,"done! ",0
```

From:
https://codebase64.org/ - **Codebase 64 wiki**

Permanent link:
https://codebase64.org/doku.php?id=base:rr_flashing

Last update: **2015-04-17 04:33**