SofTec Microsystems AN00136: Interfacing FlashRunner with Freescale HC08 Devices

by Paolo Dal Ben (paolo.dalben@softecmicro.com)

FlashRunner is a Universal In-System Programmer which uses the principles of In-System Programming (ISP) to program Freescale HC08 microcontrollers. This application note describes how to properly set up and use FlashRunner to program HC08 Flash devices.

This Application Note assumes that you are familiar with FlashRunner and with the main features of the HC08 family. Full documentation about these topics is available in the FlashRunner user's manual and in device-specific datasheets.

1. Introduction

In-system programming of HC08 microcontrollers is performed by entering the device's monitor mode, which allows the programming of the MCU memory through a single-wire interface. In-system communication with HC08 devices is performed through a so-called MON08 interface.

To use FlashRunner to perform in-system programming, you need to implement the MON08 hardware interface on your application board.

2. Hardware Configuration

The microcontroller lines needed to implement the MON08 interface and program an HC08 device are the following:

- **OSC:** Used to provide an input clock signal to use when no other external clock sources are available. FlashRunner can provide a 6.25, 12.5 or 25MHz signal on this pin.

SofTec™ Microsystems

Copyright © 2007 SofTec Microsystems®

DC01289

SofTec Microsystems

E-mail (general information): info@softecmicro.com

E-mail (technical support): support@softecmicro.com

Web: http://www.softecmicro.com

Important

SofTec Microsystems reserves the right to make improvements to its products, their documentation and software routines, without notice. Information in this manual is intended to be accurate and reliable. However, SofTec Microsystems assumes no responsibility for its use; nor for any infringements of rights of third parties which may result from its use.

SOFTEC MICROSYSTEMS WILL NOT BE LIABLE FOR DAMAGES RESULTING FROM LOSS OF DATA, PROFITS, USE OF PRODUCTS, OR INCIDENTAL OR CONSEQUENTIAL DAMAGES, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.

Trademarks

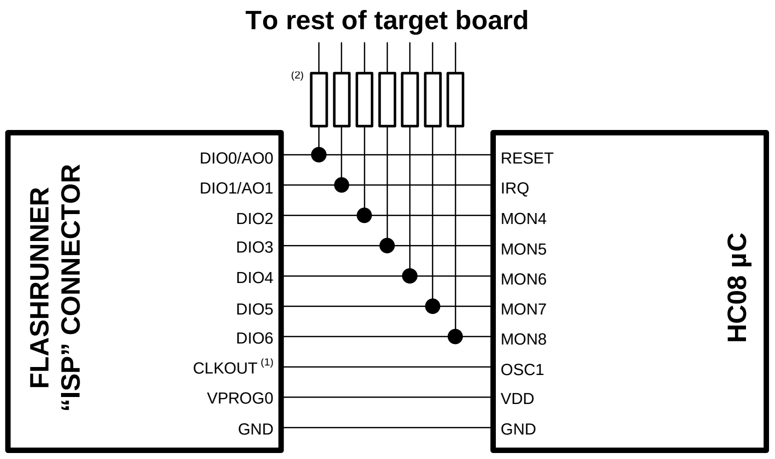
SofTec Microsystems™ and the SofTec Microsystems logo are trademarks of SofTec Microsystems S.p.A.

Other products and company names listed are trademarks or trade names of their respective companies.

- **VDD:** Device power supply voltage. The range of operating voltage is typically 2.7–5.0V, depending on the MCU. This line must always be connected to FlashRunner, because it must be correctly driven in order for the device to enter the monitor mode.
- **RST:** Device reset input/output pin.
- **IRQ:** Monitor mode is entered after a power-on reset (POR) with a high voltage, typically called V_{TST}, on the MCU's IRQ pin. V_{TST} ranges from 7V to 9V, depending on the operating V_{DD} level.
- **GND:** Device power supply ground.
- **MON4:** MON08 interface MON4 line.
- **MON5.** MON08 interface MON5 line.
- **MON6.** MON08 interface MON6 line.
- **MON7.** MON08 interface MON7 line.
- **MON8.** MON08 interface MON8 line.

Please note that each microcontroller implements the MON08 interface on different ports and pins. MON4-MON8 lines must be therefore tied to the appropriate pins of the specific target microcontroller. To know exactly which pins must be connected to FlashRunner, please refer to the device datasheet or to the script examples included in the CD-ROM you received along with FlashRunner. At the beginning of each script example you will find the specific signal connections required for each microcontroller. Note that, although not specified in the scripts, VDD and GND lines must always be connected.

The lines mentioned above must be connected to the FlashRunner's "ISP" connector according to the following diagram:



Notes

[1] Connect this line if you want the target device to be clocked by FlashRunner.

[2] If the lines needed to enter monitor mode are used for other purposes in the application, series resistors should be implemented to avoid a conflict in case the rest of the target board forces the signal level. If these lines are used as outputs, these resistors are not necessary. Special care should be taken in isolating the IRQ line from the rest of your target board because this signal could have a voltage level (V_{TST}) higher than V_{DD} which could damage the target board if the necessary measures are not taken.

Note that the voltage level of FlashRunner’s CLKOUT signal depends on the specified V_{DD} parameter (see below). This could be an issue for certain devices (e.g. MC68HC908AP) for which the clock signal has a voltage level different than the supply voltage. In this case you should take the necessary measures in order to provide the target device with a clock source of the proper voltage level.

3. HC08-Specific TPSETPAR Programming Commands

Overview

`TPSETPAR` commands set device-specific and programming algorithm-specific parameters. These commands must be sent after the **`TCSETDEV`** command and before a **`TPSTART`** / **`TPEND`** command block.

All of the FlashRunner programming capabilities rely on the monitor mode of the target device. In order to enter this special mode (which establishes a communication channel between the target device and FlashRunner) and configure it properly, all of the following parameters must be correctly specified through the relative **`TPSETPAR`** commands (although the order with which these parameters are set is not important):

- Oscillator frequency;
- External oscillator frequency divisor;
- $V_{DD;}$
- Power up time;
- Power down time;
- CLKOUT frequency.

TPSETPAR FOSC

Command syntax:

`TCSETPAR FOSC <frequency Hz>`

Parameters:

`frequency Hz:` External oscillator frequency, expressed in Hertz.

Description:

Specifies the external oscillator frequency. If you use FlashRunner’s CLKOUT line as the target device’s clock source, the **`frequency Hz`** parameter must be the same as the frequency you specify with the **`TCSETPAR CLKOUT`** command.

TPSETPAR FDIV

Command syntax:

`TCSETPAR FDIV <divisor>`

Parameters:

`divisor:` External oscillator frequency divisor. Typical values are 2 or 4.

Description:

Specifies the target device’s external oscillator frequency divisor. Specify 2 if your device’s bus frequency is half the external clock frequency. Specify 4 if your device’s bus frequency is one fourth of the external clock frequency. Note that, for most devices, only 4 is a valid divisor. See your device datasheet for more information.

TPSETPAR VDD

Command syntax:

```
TCSETPAR VDD <voltage mV>
```

Parameters:

`voltage mV`: Target device supply voltage, expressed in millivolts.

Description:

This command is used to properly generate the voltage level of the ISP signals. Additionally, the specified voltage is routed to the VPROG0 line of the FlashRunner “ISP” connector, which can be used as a supply voltage for the target board.

TPSETPAR PWUP

Command syntax:

```
TCSETPAR PWUP <time ms>
```

Parameters:

`time ms`: Power up time, expressed in milliseconds.

Description:

This command is necessary because, to enter the monitor mode, FlashRunner must properly drive the V_{DD} line during the power-on reset.

The V_{DD} rising time (PWUP) is expressed in milliseconds and depends on the features of your target board. Make sure to choose a value large enough to ensure that the V_{DD} signal reaches the high logic level within the specified time. Note that, if the V_{DD} line has a high load, a longer time is required for the V_{DD} signal to reach the high logic level. If PWUP is not long enough, FlashRunner could not be able to enter the monitor mode.

TPSETPAR PWDOWN

Command syntax:

```
TCSETPAR PWDOWN <time ms>
```

Parameters:

`time ms`: Power down time, expressed in milliseconds.

Description:

The V_{DD} falling time (PWDOWN) is expressed in milliseconds and depends on the features of your target board. Make sure to choose a value large enough to ensure that the V_{DD} signal reaches the low logic level within the specified time. Note that, if the V_{DD} line has a high load, a longer time is required for the V_{DD} signal to reach the low logic level.

TPSETPAR CLKOUT

Command syntax:

```
CSETPAR CLKOUT 25000000 | 12500000 | 6250000 | 0
```

Command options:

Frequency of the clock signal to be generated at the CLKOUT pin of the FlashRunner “ISP” connector, expressed in Hertz. The available clock frequency values are 25MHz, 12.5MHz and 6.25MHz.

Description:

Generates an auxiliary clock signal at the CLKOUT pin of the FlashRunner “ISP” connector. This signal can be used as an auxiliary clock source, and is particularly useful when the target microcontroller requires an external clock which is not otherwise available on the target board.

Furthermore, this signal can be used to speed up programming (when you want to use a clock faster than that provided by your target board).

Make sure that the clock frequency you select is not greater than the maximum allowed frequency for your device. Note that, since all devices feature an internal frequency divisor, the actual bus frequency will be a fraction (typically one fourth or one half) of the CLKOUT frequency.

If you specify **`0`** as the CLKOUT frequency, no clock signal is generated.

Please note that all of the **`TPSETPAR`** commands are mandatory except **`TPSETPAR CLKOUT`**. This command is mandatory only if you use FlashRunner’s CLKOUT pin as clock source.

4. HC08-Specific TPCMD Programming Commands

Overview

`TPCMD` commands perform a programming operation (i.e. mass erase, program, verify, etc.) These command must be sent within a **`TPSTART`** / **`TPEND`** command block.

The HC08-specific target programming commands are the following:

- **`TPCMD SETPWD`**;
- **`TPCMD BLANKCHECK`**;
- **`TPCMD MASSERASE`**;
- **`TPCMD TRIM`**;
- **`TPCMD PROGRAM`**;
- **`TPCMD VERIFY`**;
- **`TPCMD RUN`**.

TPCMD SETPWD

Command syntax:

```
TPCMD SETPWD FILE <filename>
TPCMD SETPWD CONST <B0> <B1> ... <B7>
```

Parameters:

`filename:` Name of the file containing the security bytes.

`B0, B1, ..., B7:` Value of the security bytes.

Description:

All HC08 devices have a security feature that prevents unauthorized reading of FLASH locations. You can enter monitor mode only by sending eight security bytes that match the bytes at locations $FFF6–$FFFD. When the device is blank, the security bytes are all set to $FF.

FlashRunner allows you to specify the security bytes in two ways. You can read them from a .FRB file stored in the \BINARIES directory of the SD card (e.g. **`TPCMD SETPWD FILE filename.frb`**) or you can send them directly as a sequence of numbers (e.g. **`TPCMD SETPWD CONST $FF $FF $FF $FF $FF $FF $FF $FF`**).

TPCMD BLANKCHECK

Command syntax:

```
TPCMD BLANKCHECK F|E <tgt start addr> <len>
```

Command options and parameters:

`F|E`: Memory type to be blankchecked. **`F`** stands for Flash memory and **`E`** stands for EEPROM memory.

`tgt start address`: Device memory location from where the blankcheck operation will start.

`len`: Number of bytes to be blankchecked.

Description:

Blankchecks Flash or EEPROM memory. Blankchecks **`len`** bytes starting from the address specified by **`tgt start address`**.

TPCMD MASSERASE

Command syntax:

```
TPCMD MASSERASE F|E
```

Command options:

`F|E`: Memory type to be mass erased. **`F`** stands for Flash memory and **`E`** stands for EEPROM memory.

Description:

Mass erases Flash or EEPROM memory.

Please note that, after a device is mass erased, the trimming value is lost. Thus, if your application uses the device's internal clock, it is suggested that, after a mass erase command, you trim the device's internal clock.

TPCMD PROGRAM

Command syntax:

```
TPCMD PROGRAM F|E <src offset> <tgt start addr> <len>
```

Command options and parameters:

`F|E`: Memory type to be programmed. **`F`** stands for Flash memory and **`E`** stands for EEPROM memory.

`src offset`: Offset from the beginning of the .FRB source file.

`tgt start addr`: Device memory location from where the program operation will start.

`len`: Number of bytes to be programmed.

Description:

Programs **`len`** bytes in the Flash or EEPROM memory starting from the **`tgt start addr`** address. If the memory range includes the trimming location specified by the **`TPCMD TRIM`** command, this location will be programmed with the calculated trimming value (see the description of the **`TPCMD TRIM`** command for more info about this topic).

TPCMD TRIM

Command syntax:

```
TPCMD TRIM <bus frequency Hz> <addr>
```

Command parameters:

`bus frequency Hz`: Value of the desired bus frequency, expressed in Hertz.

`addr:` Address of the memory location where the trimming value is to be programmed.

Description:

This command trims (calibrates) the device’s internal oscillator, if present. It calculates the trimming value for the bus frequency specified in the **`bus frequency Hz`** parameter, and prepares to program it at the **`addr`** address. Please note that this command does not program the calculated value in the Flash memory: the actual programming will occur during the next **`TPCMD PROGRAM`** command involving the trimming location.

TPCMD VERIFY

Command syntax:

```
TPCMD VERIFY F|E R|S <src offset> <tgt start addr>
<len>
```

Command options and parameters:

`F|E`: Memory type to be verified. **`F`** stands for Flash memory and **`E`** stands for EEPROM memory.

`R|S`: Specifies the verifying method. Choose **`R`** for a slower, but safer method that reads back all the written data. Choose **`S`** for a faster, but less safe method that only compares the checksum.

`src offset`: Offset from the beginning of the .FRB source file.

`tgt start addr`: Device memory location from which the verify operation will start.

`len`: Number of bytes to be verified.

Description:

Verifies that the contents of the target device memory correspond to that of the source image file.

TPCMD RUN

Command syntax:

`TPCMD RUN`

Command parameters:

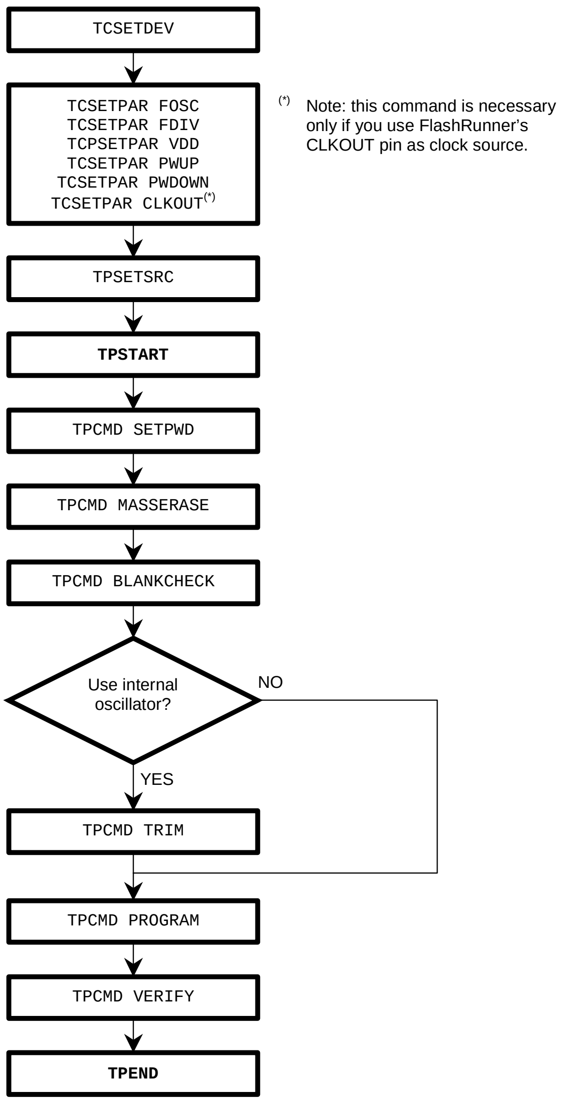
None.

Description:

Runs the target application.

5. Typical Programming Flow

The following flow chart illustrates typical steps to help you write your own script file.



6. Script Example

The example provided below will help you understand how the commands discussed above should be used for a typical HC08 device, in this case the MC68HC908EY16.

```
;
; FLASHRUNNER SCRIPT EXAMPLE FOR FREESCALE MC68HC908EY16
;
; Use this example as a starting point for your specific programming needs
;
; ----------
;
; Hardware connections
;
; DIO0 (RST): #RST
; DIO1 (IRQ): #IRQ
; DIO2 (MON4): PTA0
; DIO3 (MON5): PTA1
; DIO4 (MON6): PTB3
; DIO5 (MON7): PTB4
; DIO6 (MON8): NC
;

; Turns off logging
#LOG_OFF
; Halt on errors
#HALT_ON FAIL

; Sets device
TCSETDEV FREESCALE MC68HC908EY16 HC08

; Oscillator frequency, Hz (change as needed)
TCSETPAR FOSC 16000000

; Oscillator divisor (change as needed)
TCSETPAR FDIV 4

; Target voltage, mV (change as needed)
TCSETPAR VDD 5000

; Power-down time, ms (change as needed)
TCSETPAR PWDOWN 10

; Power-up time, ms (change as needed)
TCSETPAR PWUP 10

; Image file to be programmed (must be placed in the \BINARIES directory)
TPSETSRC FILE TEST.FRB

; Starts programming block
TPSTART

; Security bytes needed to perform subsequent operations (change as needed)
TPCMD SETPWD CONST $FF $FF $FF $FF $FF $FF $FF $FF

; Mass erases Flash memory
TPCMD MASSERASE F

; Blank checks Flash memory (change address and length as needed)
TPCMD BLANKCHECK F $C000 16384

; Trims internal oscillator (change frequency and trimming location as needed)
TPCMD TRIM 4000000 $FF80

; Programs Flash memory (change addresses and length as needed)
TPCMD PROGRAM F $C000 $C000 16384

; Verifies Flash memory, read-out method (change addresses and length as needed)
TPCMD VERIFY F R $C000 $C000 16384

; Ends programming block
TPEND
```

The FlashRunner’s system software setup will install script examples specific for each device of the HC08 family in your PC.

7. Programming Times

The following table shows programming times for several devices representative of the HC08 family.

Device	Memory Size	Bus Freq.	Operations	Time
MC68HC908QY4A	4KB	8MHz	Program	0.55s
MC68HC908QY4A	4KB	8MHz	Verify	0.40s
MC68HC908QY4A	4KB	8MHz	Erase + Program + Verify	0.90s
MC68HC908KX8	8KB	8MHz	Program	1.27s
MC68HC908KX8	8KB	8MHz	Verify	0.83s
MC68HC908KX8	8KB	8MHz	Erase + Program + Verify	1.75s
MC68HC908EY16	16KB	8MHz	Program	1.20s
MC68HC908EY16	16KB	8MHz	Verify	1.10s
MC68HC908EY16	16KB	8MHz	Erase + Program + Verify	1.95s
MC68HC908GP32	32KB	8MHz	Program	1.65s
MC68HC908GP32	32KB	8MHz	Verify	1.55s
MC68HC908GP32	32KB	8MHz	Erase + Program + Verify	2.95s
MC68HC908AS60A	60KB	8MHz	Program	2.75s
MC68HC908AS60A	60KB	8MHz	Verify	2.60s
MC68HC908AS60A	60KB	8MHz	Erase + Program + Verify	5.12s

8. References

FlashRunner user’s manual

Microcontroller-specific datasheets