AN00141: Interfacing FlashRunner with NEC V850 Devices

by Daniele Genero (daniele.genero@softecmicro.com)

FlashRunner is a Universal In-System Programmer which uses the principles of In-Circuit Serial Programming to program NEC V850 microcontrollers. This application note describes how to properly set up and use FlashRunner to program V850 Flash devices.

This Application Note assumes that you are familiar with FlashRunner and with the main features of the V850 family. Full documentation about these topics is available in the FlashRunner user's manual and in device-specific datasheets.

1. Introduction

NEC V850 devices can be divided into two subfamilies, based on the Flash technology:

- Dual Voltage Flash devices: these devices need a dedicated Flash programming voltage (usually called VPP).
- Single Voltage Flash devices: these devices require no dedicated programming voltage.

In-system programming of V850 microcontrollers is performed by entering the device's programming mode, which allows the programming of the MCU memory, through a synchronous or asynchronous serial protocol (depending on the specific device). The following table shows the programming protocols available for each V850 device.

Revision 1.0 – October 2007

DC01465

SofTec Microsystems

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

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

Web: http://www.softecmicro.com

Series	Device	Single/ Dual Voltage	Flash Technology	Write Granularity bytes)	Protocols Available	Baudrate BAUDRATE] bps)	Serial clock [SCLK] (Hz)	Oscillator frequency [FOSC] (Hz)	Security Settings (4)
Ix1	UPD70F3116	D	UC1	-	UART, CSI, CSIHS	(2)	2400 to 2000000	4000000 to 6400000	N/A
	UPD70F3329	S	UC2	16		(1)	2400 to 2500000	2500000 to 4000000	C
Ix2	UPD70F3114	D	UC1	-	UART, CSI, CSIHS	(2)	2400 to 2000000	4000000 to 6400000	N/A
	UPD70F3713 UPD70F3714	S	UC2	256		(1)	2400 to 2500000	2500000 to 10000000	C
Kx1	UPD70F3207H UPD70F3210H UPD70F3211H UPD70F3214H UPD70F3215H UPD70F3217H UPD70F3218H	S	UC2	256	UART, CSI, CSIHS	(1)	2400 to 2500000	2000000 to 5000000	A
	UPD70F32110 UPD70F32114 UPD70F32117	D	UC1	-		(2)			N/A
Kx1+	UPD70F3302 UPD70F3306 UPD70F3308 UPD70F3311 UPD70F3313 UPD70F3316 UPD70F3318	S	CZ6 HSF	4	UART, CSI, CSIHS	(1)	2400 to 2500000	2000000 to 5000000	A
GB1	UPD70F3224 UPD70F3226	D	UC1	-	UART, CSI	(2)	2400 to 1000000	1000000 to 16000000	N/A
Fx2	UPD70F323x	S	UC2	16	UART, CSI, CSIHS	(1)	2400 to 2500000	4000000 to 5000000	B
Sx2	UPD70F326x UPD70F327x UPD70F328x	S	UC2	16	UART, CSI, CSIHS	(1)	2400 to 2500000	2500000 to 10000000	C
	UPD70F326xH UPD70F327xH UPD70F328xH							2500000 to 8000000	
Dx2	UPD70F3319 UPD70F3320 UPD70F3325 UPD70F3326 UPD70F3399	S	UC2	16	UART, CSI, CSIHS	(1)	2400 to 2500000	4000000 to 5000000	B
Sx3	UPD70F333x UPD70F334x UPD70F335x UPD70F336x	S	MF2	8	UART, CSI, CSIHS	(1)	2400 to 2500000	2500000 to 10000000	D
Fx3	UPD70F337x UPD70F338x	S	MF2	(3)	UART, CSI, CSIHS	(1)	2400 to 2500000	4000000 to 16000000	D
Dx3	UPD70F342x	S	MF2	8	UART, CSI, CSIHS	(1)	2400 to 2500000	2000000 to 4000000	D
Hx2	UPD70F370x UPD70F3710 UPD70F3711 UPD70F3712	S	UC2	16	UART, CSI, CSIHS	(1)	2400 to 2500000	4000000 to 5000000	B
Jx2	UPD70F3715 UPD70F3716 UPD70F3717 UPD70F3718 UPD70F3719 UPD70F3720 UPD70F3721 UPD70F3722 UPD70F3723 UPD70F3724	S	UC2	16	UART, CSI, CSIHS	(1)	2400 to 2500000	2500000 to 10000000	C
CG4	UPD70F3433	S	UC2	32	UART, CSI, CSIHS	(1)	2400 to 1000000	2000000 to 5000000	B
RS1	UPD70F3402 UPD70F3403	S	UC2	256	UART, CSI, CSIHS	(1)	2400 to 2500000	4000000 to 8000000	C

UART: 2-wire asynchronous communication
CSI: synchronous communication
CSIHS: synchronous communication with handshaking

(1) 153600, 128000, 115200, 76800, 57600, 38400, 31250, 19200 or 9600
(2) 76800, 38400, 31250, 19200 or 9600
(3) For Flash size > 256K: write granularity = 16; for flash size <= 256K: write granularity = 8
(4) A: `TPCMD PROTECT <security byte> <boot block number>`
B: `TPCMD PROTECT <security byte> <reset vector address>`
C: `TPCMD PROTECT <security byte>`
D: `TPCMD PROTECT <security byte> <boot block number> <reset vector address>`

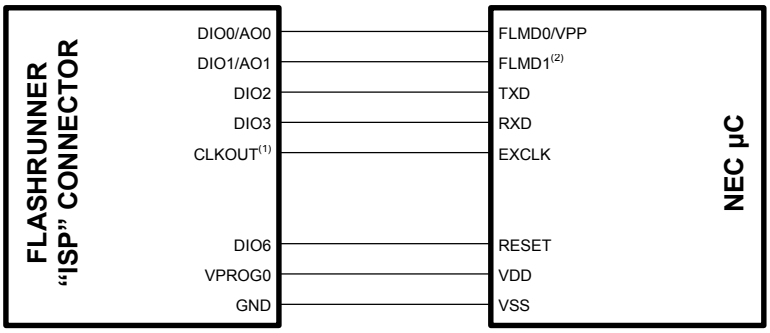
To use FlashRunner to perform in-system programming, you need to implement the appropriate in-circuit programming hardware interface on your application board.

2. Hardware Configuration

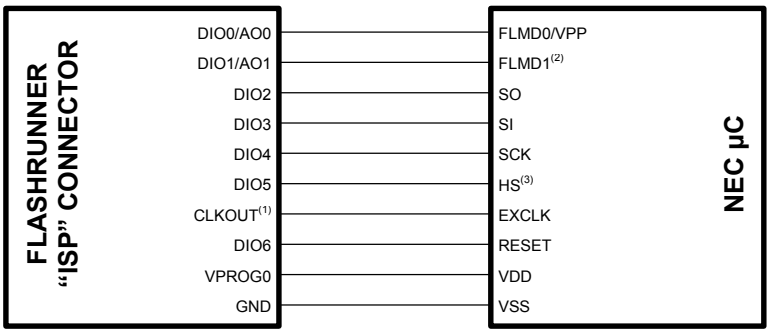
Depending on the device and the communication protocol (see table above), one of the two connection diagrams below need to be implemented.

Note: *if some microcontroller lines are shared with other peripherals/devices, please refer to the microcontroller-specific datasheet for connection guidelines.*

Connections (UART Communication)



Connections (CSI Communication)



Notes

(1) Connect this line only if you want the target device to be clocked by FlashRunner.
(2) Connect this line only if available.
(3) Connect this line only in CSIHS communication mode.

3. Specific TCSETPAR Programming Commands

Overview

`TCSETPAR` 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.

In order to enter the programming mode (which establishes a communication channel between the target device and FlashRunner) and configure it properly, the following parameters must be correctly specified through the relative **`TCSETPAR`** commands (although the order with which these parameters are set is not important):

- Communication mode (UART, CSI or CSIHS);
- Baudrate (for UART mode);
- Serial clock (for CSI and CSIHS modes);
- Oscillator frequency;
- V_{DD};
- Auxiliary V_{DD} (if necessary);
- Power up time;
- Power down time;
- Reset up time;
- Reset down time;
- Reset driving mode (push pull or open drain);
- FlashRunner clock out signal.

TCSETPAR CMODE

Command syntax:

```
TCSETPAR CMODE CSI|CSIHS|UART
```

Command options:

`CSI`: Uses the synchronous communication mode.

`CSIHS`: Uses the synchronous communication mode with handshaking.

`UART`: Uses the 2-wire asynchronous communication mode.

Description:

Sets the communication protocol. Please refer to the table on page 2 for the communication protocols supported by your target device.

TCSETPAR BAUDRATE

Command syntax:

`TCSETPAR BAUDRATE <baudrate>`

Parameters:

`<baudrate>`: UART communication speed, in bits per second.

Description:

This command is used to set the communication speed for the UART communication protocol. Please refer to table on page 2 for a list of allowed baudrates for your specific target devices.

TCSETPAR SCLK

Command syntax:

`TCSETPAR SCLK <frequency Hz>`

Parameters:

`frequency Hz`: Serial communication clock frequency for synchronous communication (CSI and CSIHS protocols), expressed in Hertz.

Description:

This commands sets the serial communication clock frequency when using the CSI and CSIHS protocols.

When using the UART protocol, this command is not necessary.

TCSETPAR FOSC

Command syntax:

`TCSETPAR FOSC <frequency Hz>`

Parameters:

`frequency Hz`: Target device's oscillator frequency.

Description:

Sets the target device's internal or external oscillator frequency.

TCSETPAR 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.

TCSETPAR VDD_AUX

Command syntax:

`TCSETPAR VDD_AUX <voltage mV>`

Parameters:

`voltage mV`: Auxiliary supply voltage, expressed in millivolts, in the range 3000-14500mV.

Description:

This command is used to generate an optional, auxiliary voltage level for user purposes. The specified voltage is routed to the VPROG1 line of the FlashRunner “ISP” connector.

A value of 0 drives the VPROG1 line to GND. If the **`TCSETPAR VDD_AUX`** is not sent, the VPROG1 line is driven to HiZ.

TCSETPAR PWUP

Command syntax:

`TCSETPAR PWUP <time ms>`

Parameters:

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

Description:

This command is necessary because, to enter the programming 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 programming mode.

TCSETPAR PWDOWN

Command syntax:

`TCSETPAR PWDOWN <time ms>`

Parameters:

`time ms`: Power falling 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.

TCSETPAR RSTUP

Command syntax:

`TCSETPAR RSTUP <time µs>`

Parameters:

`time µs`: Reset rising time, expressed in microseconds.

Description:

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

TCSETPAR RSTDOWN

Command syntax:

`TCSETPAR RSTDOWN <time µs>`

Parameters:

`time µs`: Reset falling time, expressed in microseconds.

Description:

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

TCSETPAR RSTDRV

Command syntax:

`TCSETPAR RSTDRV PUSHPULL|OPENDRAIN`

Parameters:

`PUSHPULL`: The Reset line is driven in push-pull mode.

`OPENDRAIN`: The Reset line is driven in open drain mode.

Description:

Drives the Reset line in push-pull or open drain mode. If the **`TCSETPAR RSTDRV`** command is not sent, FlashRunner drives the Reset line in open drain mode by default.

TCSETPAR CLKOUT

Command syntax:

`CSETPAR CLKOUT <frequency Hz>`

Command options:

`frequency Hz`: Frequency of the clock signal to be generated at the CLKOUT pin of the FlashRunner “ISP” connector, expressed in Hertz.

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 that is not otherwise available on the target board.

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

Note: *due to the FlashRunner internal circuitry and other factors, the frequency generated by FlashRunner is usually not the exact frequency specified by the* ***`TCSETPAR CLKOUT`*** *command. However, as a response to the* ***`TCSETPAR CLKOUT`*** *command, FlashRunner answers with the actual frequency that will be used.*

4. 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.

NEC V850-specific target programming commands are the following:

- **TPCMD BLANKCHECK**;
- **TPCMD MASSERASE**;
- **TPCMD BLOCKERASE**;
- **TPCMD PROGRAM**;
- **TPCMD VERIFY**;
- **TPCMD PROTECT**;
- **TPCMD RUN**.

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 Code Flash memory and **E** stands for Data Flash memory.
tgt start address:	Device memory location from where the blankcheck operation will start.
len:	Number of locations to be blankchecked.

Description:

Blankchecks Code or Data Flash memory. Blankchecks **len** locations starting from the address specified by **tgt start address**.

tgt start address must be the first location of a block. The location (**tgt start address** + **len** – 1) must be the last location of a block.

TPCMD MASSERASE

Command syntax:

```
TPCMD MASSERASE F|E|C
```

Command options and parameters:

F|E|C:	Memory type to be mass erased. **F** stands for Code Flash memory, **E** stands for Data Flash memory and **C** stands for both.

Description:

Mass erases Code or Data Flash memory, or both.

TPCMD BLOCKERASE

Command syntax:

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

Command options and parameters:

F|E:	Block memory type. **F** stands for Code Flash memory and **E** stands for Data Flash memory.
tgt start address:	Device memory location from where the block erase operation will start.
len:	Number of locations to be erased.

Description:

Erases one or more Code or Data Flash memory blocks.

tgt start address must be the first location of a block. The location (**tgt start address** + **len** – 1) must be the last location of a block.

TPCMD PROGRAM

Command syntax:

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

Command options and parameters:

F|E:	Memory type to be programmed. **F** stands for Code Flash memory and **E** stands for Data Flash memory.
src offset:	Offset from the beginning of the source memory.
tgt start addr:	Device memory location from where the program operation will start.
len:	Number of locations to be programmed.
BLOCK|BYTE:	Programs by block (filling unused locations with $FF) or by byte (max 256).

Description:

Programs **len** locations in the Code or Data Flash memory starting from the **tgt start addr** address.

For Block Programming

tgt start address must be the first location of a block.

On single-voltage devices, if **len** is not a multiple of block length, remaining locations in the last affected block are filled with $FF.

On dual-voltage devices, if **len** is not a multiple of 256, additional locations are filled with $FF until a number of locations which is a multiple of 256 is reached. In other words, if **len** is not a multiple of 256, 256 - (**len** mod 256) additional locations are filled with $FF.

For Byte Programming

tgt start address must be aligned according to the write granularity (see table on page 2).

On single-voltage devices, **len** must be less than or equal to 256 and, if **len** is not a multiple of the write granularity, additional locations are filled with $FF until a number of locations which is a multiple of the write granularity is reached.

Dual-voltage devices do not support byte programming.

Note: *the **TPCMD PROGRAM** command returns an error if the device is not blank. Therefore, the **TPCMD BLANKCHECK** command is not necessary.*

TPCMD VERIFY

Command syntax:

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

Command options and parameters:

F|E:	Memory type to be verified. **F** stands for Code Flash memory and **E** stands for Data Flash memory.
R|S|I:	Verify method. Reads back all written data (**R**, slow but secure), compares a checksum (**S**, fast but not secure), or performs an internal check (**I**, fast but not secure). The **R** option is available for all devices, the **S** option is available for single-voltage devices only, and the **I** options is available for dual-voltage devices only.
src offset:	Offset from the beginning of the source memory.
tgt start addr:	Device memory location from where the verify operation will start.
len:	Number of locations to be verified.

Description:

Verifies **len** locations in the Flash memory starting from the **tgt start addr** address.

tgt start address must be the first location of a block.

For R Verify Method

On single-voltage devices, if **len** is not a multiple of the write granularity, additional locations are verified against $FF until a number of locations which is a multiple of the write granularity is reached.

On dual-voltage devices, if **len** is not a multiple of 256, additional locations are verified against $FF until a number of locations which is a multiple of 256 is reached. In other words, if **len** is not a multiple of 256, 256 - (**len** mod 256) additional locations are verified against $FF.

For S Verify Method

On single-voltage devices, the location
(**tgt start address** + **len** – 1) must be the last location of a block.

Dual-voltage devices do not support the **S** verify method.

For I Verify Method

Single-voltage devices do not support the **I** verify method.

On dual-voltage devices, the location
(**tgt start address** + **len** – 1) must be the last location of a block.

TPCMD PROTECT

Command syntax:

```
TPCMD PROTECT <security byte>
TPCMD PROTECT <security byte> <reset vector address>
TPCMD PROTECT <security byte> <boot block number>
TPCMD PROTECT <security byte> <boot block number>
              <reset vector address>
```

Command options and parameters:

security byte:	Security byte.
reset vector address:	Reset vector address.
boot block number:	Boot block number.

Description:

Programs the security byte. This operation is not supported by dual voltage devices.

The syntax of this command depends on the specific target device. Please refer to the table on page 2 for more information.

TPCMD RUN

Command syntax:

```
TPCMD RUN
```

Command parameters:

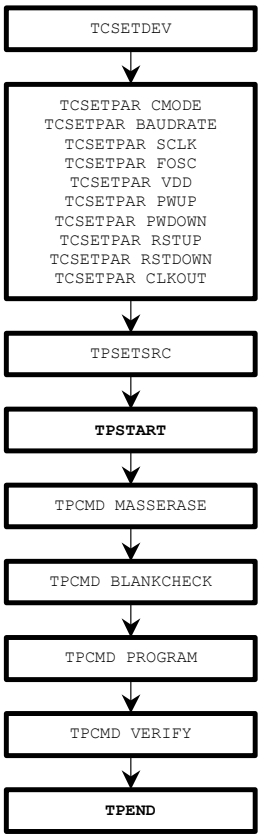
None.

Description:

Runs the target application.

5. Typical Programming Flow

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



TCSETPAR commands depend on the specific target device and communication protocol.

6. Script Examples

The example provided below will help you understand how the commands discussed above should be used for a typical V850 device, in CSIHS communication mode. In this case, the device is a UPD70F3370.

```
;
; FLASHRUNNER SCRIPT EXAMPLE FOR NEC UPD70F3370
;
; Use this example as a starting point for your specific programming needs
;
; ----------
;
; Hardware connections
;
; DIO0 (FLMD0)
; DIO1 (FLMD1)
; DIO2 (TxD) Device Output
; DIO3 (RxD) Device Input
; DIO4 (SCK)
; DIO5 (HS)
; DIO6 (RESET)
; CLKOUT (Not used)
;

;
; MF2 Flash technology
; 8 bytes Flash granularity
;

; Turns off logging
#LOG_ON OFF
; Halt on errors
#HALT_ON FAIL

; Sets device
TCSETDEV NEC UPD70F3370 V850

;-------------------------
;FLASHRUNNER I/O Settings
;-------------------------

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

; Clock oscillator frequency driven by FlashRunner, Hz
; The possible frequencies are: 25000000, 12500000, 6250000, 3125000, 1562500, 0 (DISABLED)
TCSETPAR CLKOUT 0

; VDD rise-time, ms (from 0 ms to 65535 ms)
TCSETPAR PWUP 10

; VDD fall-time, ms (from 0 ms to 65535 ms)
TCSETPAR PWDOWN 10

; RESET rise-time, us (from 0 us to 65535 us)
TCSETPAR RSTUP 100

; RESET fall-time, us (from 0 us to 65535 us)
TCSETPAR RSTDOWN 100

; RESET drive mode: OPENDRAIN or PUSHPULL
TCSETPAR RSTDRV OPENDRAIN

;--------------------
;V850 ALGO Settings
;--------------------

; Communication mode settings (UART, CSI or CSIHS supported)
TCSETPAR CMODE CSIHS

; External clock source frequency, Hz (change as needed)
; For this device the maximum FOSC is 16000000 Hz
TCSETPAR FOSC 16000000

; Serial clock settings, Hz
; For this device the maximum SCLK is 2500000 Hz.
TCSETPAR SCLK 2500000
```

```
;---------------------------
;Start Programming operation
;---------------------------

; Starts programming block
TPSTART

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

;--------------
;FLASH commands
;--------------

; Mass erases Flash memory
TPCMD MASSERASE F

; Blank checks Flash memory (change address and length as needed)
TPCMD BLANKCHECK F $0 $20000

; Programs Flash memory (change addresses and length as needed)
TPCMD PROGRAM F $0 $0 $20000 BLOCK

; Verifies Flash memory (change addresses and length as needed)
TPCMD VERIFY F R $0 $0 $20000

; Ends programming block
TPEND
```

The example provided below will help you understand how the commands discussed above should be used for a typical V850 device, in UART communication mode. In this case, the device is a UPD70F3233.

```
;
; FLASHRUNNER SCRIPT EXAMPLE FOR NEC UPD70F3233
;
; Use this example as a starting point for your specific programming needs
;
; ----------
;
; Hardware connections
;
; DIO0 (FLMD0)
; DIO1 (FLMD1)
; DIO2 (TxD) Device Output
; DIO3 (RxD) Device Input
; DIO4 (Not used)
; DIO5 (Not used)
; DIO6 (RESET)
; CLKOUT (Not used)
;

;
; UC2 Flash technology
; 16 bytes Flash granularity
;

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

; Sets device
TCSETDEV NEC UPD70F3233 V850

;------------------------
;FLASHRUNNER I/O Settings
;------------------------

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

; Clock oscillator frequency driven by FlashRunner, Hz
; Possible frequencies are: 25000000 divided by a 16-bit prescaler, 0 (DISABLED)
TCSETPAR CLKOUT 0

; VDD rise-time, ms (from 0 ms to 65535 ms)
TCSETPAR PWUP 10
```

```
; VDD fall-time, ms (from 0 ms to 65535 ms)
TCSETPAR PWDOWN 10

; RESET rise-time, us (from 0 us to 65535 us)
TCSETPAR RSTUP 100

; RESET fall-time, us (from 0 us to 65535 us)
TCSETPAR RSTDOWN 100

; RESET drive mode: OPENDRAIN or PUSHPULL
TCSETPAR RSTDRV OPENDRAIN

;--------------------
;V850 ALGO Settings
;--------------------

; Communication mode settings (UART, CSI or CSIHS supported)
TCSETPAR CMODE UART

; External clock source frequency, Hz (change as needed)
; For this device the maximum FOSC is 20000000 Hz
TCSETPAR FOSC 4000000

; Baudrate settings, bps (change as needed)
; Only the UART communication mode need to set the baudrate.
; For this device the possible values are 153600, 76800, 31250, 19200 and 9600 bps.
TCSETPAR BAUDRATE 153600

;---------------------------
;Start Programming operation
;---------------------------

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

; Starts programming block
TPSTART

;--------------
;FLASH commands
;--------------

; Mass erases Flash memory
TPCMD MASSERASE F

; Blank checks Flash memory (change address and length as needed)
TPCMD BLANKCHECK F $0 $40000

; Programs Flash memory (change addresses and length as needed)
TPCMD PROGRAM F $0 $0 $40000 BLOCK

; Verifies Flash memory (change addresses and length as needed)
TPCMD VERIFY F R $0 $0 $40000

; Ends programming block
TPEND
```

The FlashRunner's system software setup will install script examples specific for each device of the V850 family in your PC.

7. Programming Times

The following table shows programming times for selected NEC V850 devices.

Device	Memory Size	Comm. Mode	Osc. Freq.	Operations	Time
UPD70F3233	128KB Flash	CSIHS (2.5 MHz)	4 MHz	Program	6.99 s
UPD70F3233	128KB Flash	CSIHS (2.5 MHz)	4 MHz	Erase + Program + Verify	22.24 s
UPD70F3308	256KB Flash	CSIHS (2.5 MHz)	6.25 MHz	Program	36.25 s
UPD70F3308	256KB Flash	CSIHS (2.5 MHz)	6.25 MHz	Erase + Program + Verify	47.12 s
UPD70F3366	640KB Flash	CSI (2.5 MHz)	5 MHz	Program	16.53 s
UPD70F3366	640KB Flash	CSI (2.5 MHz)	5 MHz	Erase + Program + Verify	30.98 s
UPD70F3370	128KB Flash	CSIHS (2.5 MHz)	16 MHz	Program	3.63 s
UPD70F3370	128KB Flash	CSIHS (2.5 MHz)	16 MHz	Erase + Program + Verify	6.19 s

Programming times depend on Programming Algorithm version, target board connections, communication mode, target microcontroller mask, and other conditions. Programming times for your actual system may therefore be different than the ones listed here. SofTec Microsystems reserves the right to modify Programming Algorithms at any time.

8. References

FlashRunner user's manual

Microcontroller-specific datasheets