SMH Technologies
UNIVERSAL PRODUCTION
IN–SYSTEM PROGRAMMING

18/06/2019 Moreno Ortolan

DC11062

AN00183: Performance Optimizations

FlashRunner 2.0 is a Universal In-System Programmer, which feature several options to integrate flashing into your test system. This Application Note describes how to modify a script to make an optimization of execution in order to reduce project execution time.

1. Introduction

There is the possibility of improving the programming cycle time using various methods within the FlashRunner 2.0 project. In this application note we describe what these methods are and when these methods are useful for performing a faster execution.

Systein Italia Srl
VAT (P.I./C.F.): IT01697470936
Via Giovanni Agnelli, 1
33089 Villotta di Chions (PN) Italy
Phone +39 0434.421111
Fax +39 0434.639021
info@smh-tech.com
www.smh-tech.com
Registered office / sede legale:
V.lo del Forno, 9
Zip / Cap 33170 Pordenone (PN) Italy
Share capital / capitale soc.iale € 10.000

2. Conditional Scripting:

Flash Runner 2.0 offers the possibility of making the project flexible and customizable, so it is possible to implement conditional commands able to control the script command flow.

The syntax of conditional commands inside projects is this:

```
#IFERR EXPRESSION

#THEN STATEMENT
```

This conditional script allows us to execute the first statement and if it produces an error, to execute the next statement placed after the *then* (if the latter is actually present).

This type of execution is useful if the second expression is longer in time than the first, because with this conditional command you have the possibility to skip the second if the first work correctly.

One example of using the conditional scripting is to skip the Masserase operation if the device is already blanked. So, with the previous type of command, is possible to run a Blankcheck command before the Masserase and run the latter command if the first return an error.

This is the correct syntax of this conditional script:

```
#IFERR TPCMD BLANKCHECK F

#THEN TPCMD MASSERASE C
```

Obviously, this type of work flow is useful only if Blankcheck is faster that Masserase, because if it wasn't, there isn't a great improvement in skipping the erase command.

With this approach it is often possible to reduce project execution time. This technique applies mostly on conditioning target device memory erasing only if Blankcheck fails.

Systein Italia Srl
VAT (P.I./C.F.): IT01697470936
Via Giovanni Agnelli, 1
33089 Villotta di Chions (PN) Italy
Phone +39 0434.421111
Fax +39 0434.639021
info@smh-tech.com
www.smh-tech.com
Registered office / sede legale:
V.lo del Forno, 9
Zip / Cap 33170 Pordenone (PN) Italy
Share capital / capitale soc.iale € 10.000

Notes:

- Please note that syntax above can be used only inside a script file and it's not recognized on command line.
- Control flows can't be nested.
- Only one expression can be evaluated and only one statement can be executed for each case.
- If expression evaluation returns false, error stack will be traced in the log file. Anyway, if all the subsequent commands will return ">", project will not return with an execution error.

Please refer to your driver specific commands before implementing conditional scripting it in your projects.

3. Verify Checksum – Verify Readout:

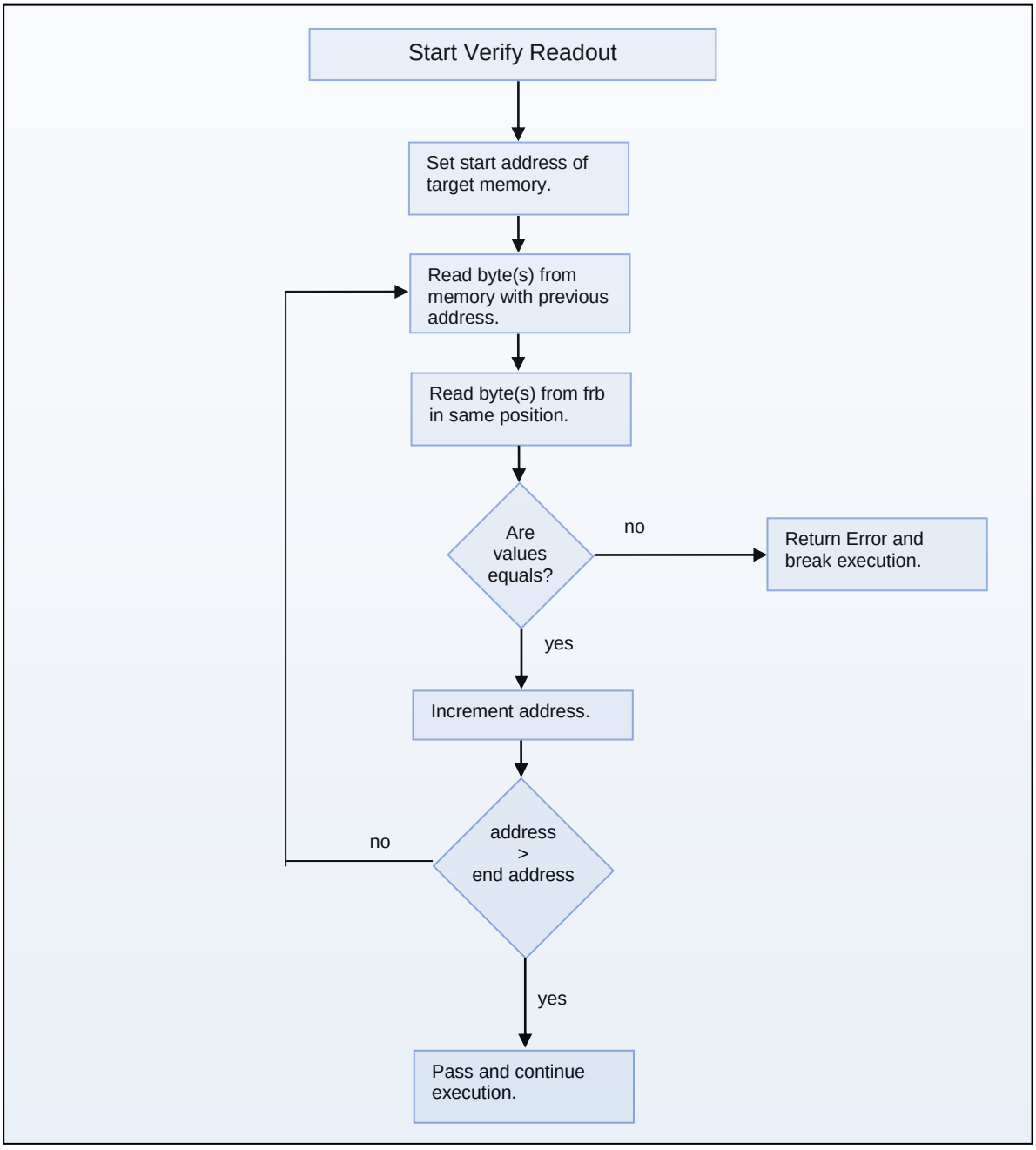
In order to improve the execution time of a project it's useful to use the verify checksum instead the verify readout.

The difference in using one or the other is that often the verify readout is slower than the verify checksum.

This happens because the verify readout checks all the programmed memory, making a comparison between the values that it reads from the memory and the respective values present in the frb.

The verify readout works in a very simple way, reads one or more bytes from the target's memory and compares them with the respective values in the frb. Obviously, this type of verify is very accurate because it compares all the values that have been programmed one by one.

As just mentioned, the verify readout makes a byte by byte comparison, so if it doesn't return an error it means that the programming has been done correctly without the possibility of error.

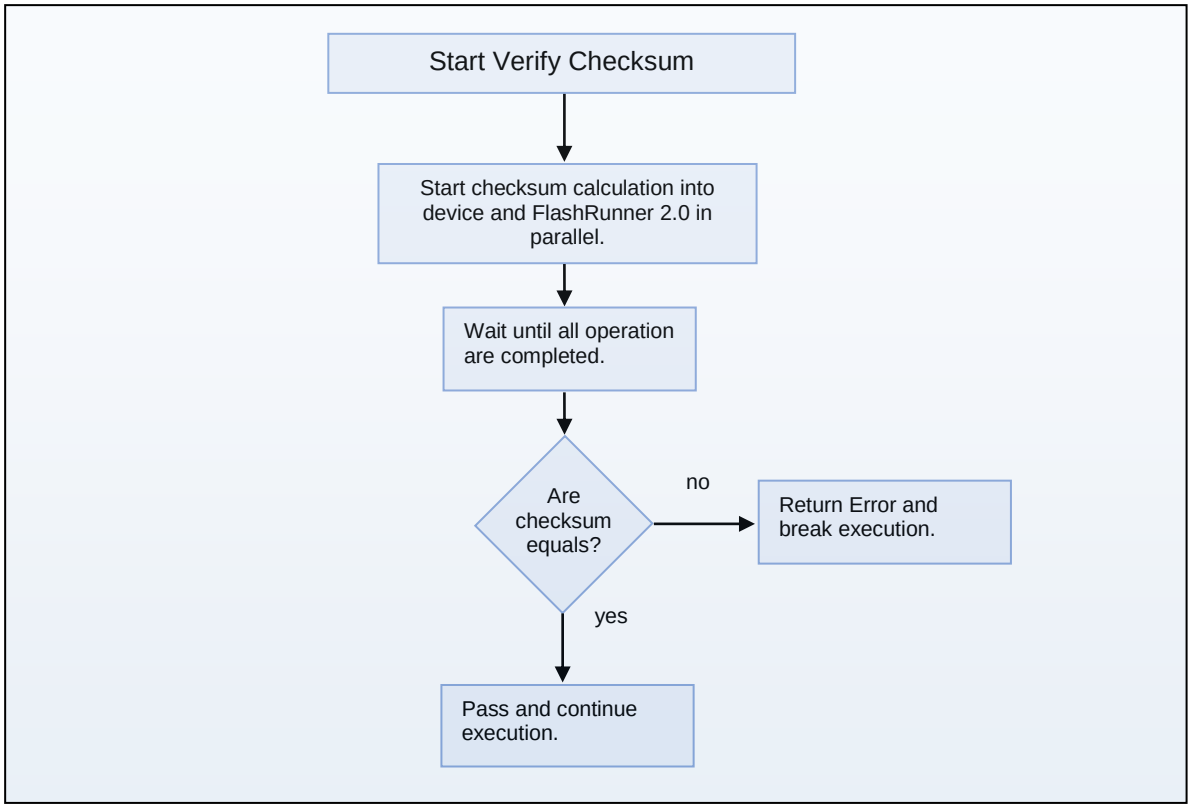


Systein Italia Srl
VAT (P.I./C.F.): IT01697470936
Via Giovanni Agnelli, 1
33089 Villotta di Chions (PN) Italy
Phone +39 0434.421111
Fax +39 0434.639021
info@smh-tech.com
www.smh-tech.com
Registered office / sede legale:
V.lo del Forno, 9
Zip / Cap 33170 Pordenone (PN) Italy
Share capital / capitale soc.iale € 10.000

The verify checksum instead sends a command to the target device, which calculates a checksum of a specific section of memory. Simultaneously with this execution, FlashRunner 2.0 calculates the checksum of the expected values in that memory location, using the values present in the frb. At this point, when both have completed the calculation, a comparison is made between the two checksums. If they correspond, then FlashRunner 2.0 proceeds to the next memory section until it is completed.

Please note that the verify checksum is not safe as Verify readout command as it is based on a memory area calculation result.

So, if there are incorrect values in the memory, the checksum calculated by FlashRunner 2.0 and the device are different, which leads verify checksum returns an error and interrupts the execution of the script.



Verify Checksum is broadly implemented in our drivers but some cases could be unuseful. Checksum require a CPU which actually does the calculation, so memories are automatically excluded. Doing target calculation from bare FlashRunner 2.0 read would lead to the same execution time as Verify Readout method.

Microcontrollers must also include in bootloader specific functions the possibility to launch this calculation. This would lead to speedup benefit by doing frb calculation in parallel with target device memory calculation.

4. TPSETSRC with Ignore Blank Pages

In order to improve the execution time of a project it possible to use one additional parameter of TPSETSRC. This additional parameter is IGNORE_BLANK_PAGE**.**

Command syntax:
TPSETSRC <filename> IGNORE_BLANK_PAGE

Scriptable:
Yes.

Available on:
Site engines only.

Parameters:

filename: name of the file in the binaries folder inside FlashRunner 2.0.

IGNORE_BLANK_PAGE: optional parameter, avoid to program FRB pages which are filled with blank value.

Answer data:
Success: none.
Error: the error code.

Description:
Sets the source of data to be programmed and verified in subsequent **TPCMD** commands.

With this additional parameter the TPSETSRC command has this syntax:

```
#1*TPSETSRC <name_file.frb> IGNORE_BLANK_PAGE
```

IGNORE_BLANK_PAGE parameter will skip all pages which are fullfilled with values composed by “blank value”. Therefore with this feature is possible to reduce the total programming time.