



CBPro CLI User's Guide

Installing and Using the ClockBuilder Pro Command Line
Interface (CLI)

1/14/2020

For ClockBuilder Pro v2.39.3 and Higher

Document Contents:

1	Overview	3
2	Installing the CLI	3
3	CLI Files	4
4	Learning About CLI Capabilities and Workflow	5
5	Running a Sample	5
6	CLI Error Reporting	6
7	CLI Tools.....	6
8	Getting Help.....	8
9	Appendix – Command Line Tool Documentation.....	9
	CBProDeviceRead	9
	CBProDeviceWrite	14
	CBProFOTF1 (Previously Named CBProSi534x8xFOTF)	19
	CBProMultiEditAndExport	40
	CBProMultiProjectExport	47
	CBProProjectEdit	52
	CBProProjectRegistersExport	55
	CBProProjectSettingsExport	57
	CBProRegistersToSettings	58
	CBProRegistersToSettings	60
	CBProRegmapExport	62
	CBProSi534x8xFirmwareDownload.....	63
	CBProSi534x8xFirmwareExport.....	66

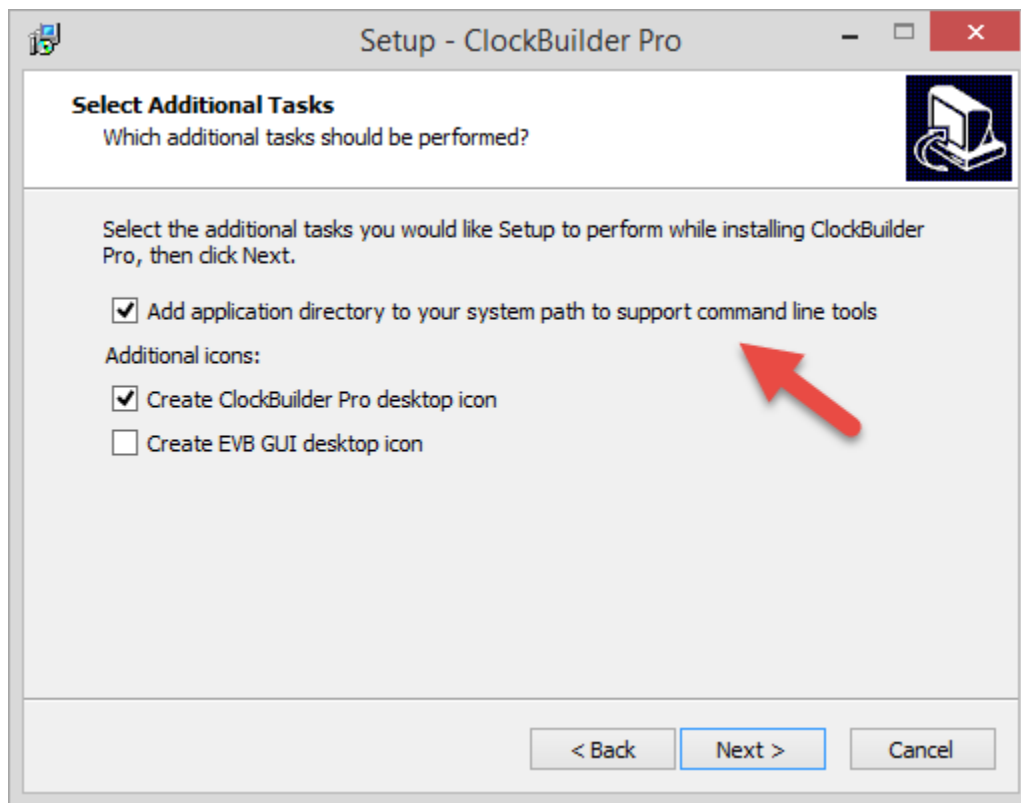
1 Overview

This document describes a Command Line Interface (CLI) that is available to automate common CBPro activities, such as creating an in-system programming file for a CBPro project file. The CLI also provides features not available in the GUI, such as batch creation of alternate frequency plans using simple text description of clock frequencies.

The document target audience is an engineer who wishes to understand how he/she can use the CLI.

2 Installing the CLI

The CLI is bundled with ClockBuilder Pro. When you run the CBPro installer, ensure the following checkbox is clicked:



When checked, the installer will add C:\Program Files (x86)\Silicon Laboratories\ClockBuilder Pro\Bin (or similar, depending on what you selected as the target folder) to your Windows PATH.

3 CLI Files

The following CLI-centric files are installed:

- The CLI executables
- This document
- A training presentation
- User manuals for each CLI tool
- Sample project files, data files, and batch scripts that show how to perform select activities
- Various README files

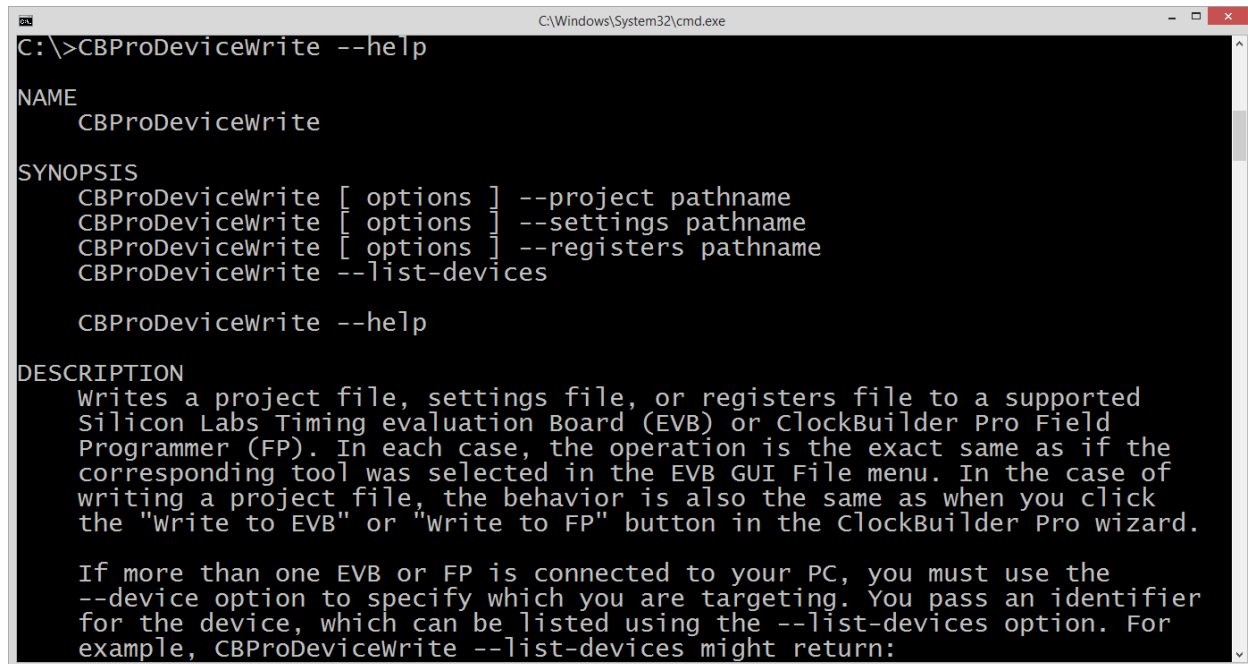
C:\Program Files (x86)\Silicon Laboratories\ClockBuilder Pro

```
├── Bin
│   ├── CBProDeviceRead.exe
│   ├── CBProDeviceWrite.exe
│   ├── CBProFOTF1.exe
│   ├── CBProMultiEditAndExport.exe
│   ├── CBProMultiProjectExport.exe
│   ├── CBProProjectEdit.exe
│   ├── CBProProjectRegistersExport.exe
│   ├── CBProProjectSettingsExport.exe
│   ├── CBProRegmapExport.exe
│   ├── CBProSi534x8xFirmwareDownload.exe
│   └── CBProSi534x8xFirmwareExport.exe
├── CLI
│   ├── Docs
│   │   ├── CBPro CLI User Guide.pdf
│   │   ├── CBPro Tools & Support for In-System Programming.pdf
│   │   └── CBPro*.txt (user manual for each CLI)
│   ├── README.txt
│   └── Samples
│       ├── Edit-And-Export-For-Multi-PLL-Device
│       ├── Edit-And-Export-For-Single-PLL-Device
│       ├── FOTF-For-Multi-PLL-Device
│       ├── FOTF-For-Single-PLL-Device
│       ├── Multi-Edit-And-Export-For-Multi-PLL-Device
│       ├── Multi-Edit-And-Export-For-Single-PLL-Device
│       ├── Multi-Project-Export
│       └── Single-Project-Export
```

4 Learning About CLI Capabilities and Workflow

Please read the [CBPro Tools & Support for In-System Programming](#) training and review applicable [Samples](#).

Every CLI executable has a built-in user manual that can be displayed by typing `cmd --help` from the DOS prompt. For example:



```
C:\Windows\System32\cmd.exe
C:\>CBProDeviceWrite --help

NAME
    CBProDeviceWrite

SYNOPSIS
    CBProDeviceWrite [ options ] --project pathname
    CBProDeviceWrite [ options ] --settings pathname
    CBProDeviceWrite [ options ] --registers pathname
    CBProDeviceWrite --list-devices

    CBProDeviceWrite --help

DESCRIPTION
    Writes a project file, settings file, or registers file to a supported
    Silicon Labs Timing evaluation Board (EVB) or ClockBuilder Pro Field
    Programmer (FP). In each case, the operation is the exact same as if the
    corresponding tool was selected in the EVB GUI File menu. In the case of
    writing a project file, the behavior is also the same as when you click
    the "Write to EVB" or "Write to FP" button in the ClockBuilder Pro wizard.

    If more than one EVB or FP is connected to your PC, you must use the
    --device option to specify which you are targeting. You pass an identifier
    for the device, which can be listed using the --list-devices option. For
    example, CBProDeviceWrite --list-devices might return:
```

All of the user manuals are also included at the end of this user's guide and available in the [C:\Program Files \(x86\)\Silicon Laboratories\ClockBuilder Pro\CLI\Docs](#) folder.

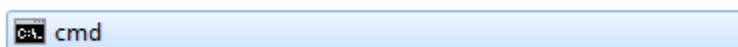
5 Running a Sample

The Samples normally consist of a README.txt file, a DOS batch script to run the sample, input files used by the batch script, and the output of the batch script. This lets you review the files produced by sample without actually running it.

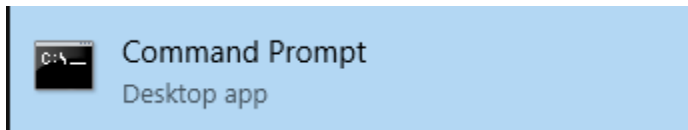
To run the sample, you will need to copy the Samples folder to a user folder. This is required because the installer makes the output files read-only, and the batch script will try to overwrite them and fail.

Once you have made a copy of the samples, launch a DOS prompt/shell. For example, by typing in `cmd` from the Start Menu and clicking:

Windows 7:



Windows 10:



To test CLI execution, change directory to the Multi-Project-Export folder and execute the run.bat script:

```
C:\Windows\System32\cmd.exe
C:\Windows\System32>cd c:/temp/Samples
c:\temp\Samples>cd Multi-Project-Export
c:\temp\Samples\Multi-Project-Export>run.bat
c:\temp\Samples\Multi-Project-Export>CBProMultiProjectExport --out-folder Output
--create-out-folder Si5345-RevD-CFG1-Project.slabtimeproj Si5345-RevD-CFG2-Proj
ect.slabtimeproj Si5345-RevD-CFG3-Project.slabtimeproj
Saving export files to Output
Wrote Settings.csv
Wrote Registers.csv
Wrote P1-Report.txt
Wrote P1-Registers-Script.txt
Wrote P1-Settings.txt
Wrote P2-Report.txt
Wrote P2-Registers-Script.txt
Wrote P2-Settings.txt
Wrote P3-Report.txt
Wrote P3-Registers-Script.txt
Wrote P3-Settings.txt
c:\temp\Samples\Multi-Project-Export>
```

This example used the multi-project export CLI to create in-system programming files for 3 project files. Design reports for each project were also created.

6 CLI Error Reporting

All command line tools return exit code 0 on success, and a non-0 code if there was an error. Status and error messages are printed to standard output.

7 CLI Tools

Many CLI tools have graphical front-ends. The table below summarizes available CLIs and any GUI counterpart.

CLI Tool	GUI Tool	Task
CBProDeviceRead	EVB GUI (partial)	Read all or specific settings/registers on a device. Results can be saved in text, comma separated value, tab separated value, or HTML formats. Device can be on Silicon Labs EVB or your PCB via ClockBuilder Pro field programmer.

CBPro CLI User's Guide

CBProDeviceWrite	CBPro wizard, EVB GUI	Write to volatile configuration registers on Silicon Labs EVB or your system board via ClockBuilder Pro field programmer.
CBProFOTF1	None	Create register write scripts to switch a subset of outputs to a new frequency, leaving other outputs undisturbed. Select Si538x/4x devices only.
CBProMultiEditAndExport	None	Combines functionality of CBProProjectEdit & CBProMultiProjectExport into single tool: define alternate frequency plan(s) for a project file and export the settings and registers, including full configuration register scripts.
CBProMultiProjectExport	CBPro Export tool	Create setting and register dumps for 2+ project files, including CSV comparison of registers and settings across all project files. Also creates register write script for each project file.
CBProProjectEdit	CBPro wizard	Change input frequency, output frequency, and/or bandwidth of a project file & save to a new project file.
CBProProjectRegistersExport	CBPro Export tool	Create configuration register write script. This script will full reconfigure a device.
CBProProjectSettingsExport	CBPro Export tool	Create configuration settings (bitfield) dump. This is useful to review the device settings associated with a project file.
CBProRegistersToSettings	None	Breaks out device register values by named setting (bitfield).
CBProRegmapExport	CBPro Export tool	Create a custom register map for a given device/revision.
CBProSi534x8xFirmwareDownload	None	Download a firmware image to a Silicon Labs EVB or your system board via ClockBuilder Pro field programmer. Select Si538x/4x devices only.
CBProSi534x8xFirmwareExport	CBPro Export tool	Create a firmware image, containing both device configuration defaults and latest MCU software. Select Si538x/4x devices only.

8 Getting Help

Create a new case with at the Silicon labs support page, <http://www.silabs.com/contactsupport>.

9 Appendix – Command Line Tool Documentation

CBProDeviceRead

SYNOPSIS

```
CBProDeviceRead [ options ] --settings setting-id1 [ setting-id2 ... ]
CBProDeviceRead [ options ] --registers register1 [ register2 ... ]
CBProDeviceRead [ options ] --all
CBProDeviceRead --list-devices
```

```
CBProDeviceRead --help
```

DESCRIPTION

Reads device registers and associated named settings (aka bitfields) from a supported Silicon Labs Timing evaluation Board (EVB) or ClockBuilder Pro Field Programmer (FP). Has three general modes of operation:

- + Read one or more named settings you specify either on the command line (--settings option). Reads are done at the register level, but output is per-setting.

- + Read one or more registers you specify either on the command line (--registers option). In this mode, no mapping to setting names is done: you get back a simple register address, value list.

- + Read all read-only and read-write settings on the device (--all option). Reads are done at the register level, but output is per-setting.

By default a text table is created. Comma separated value (CSV), tab separated value, and HTML are also supported via the --format option.

See EXAMPLES for an example of command line usage and output for each mode.

If more than one EVB or FP is connected to your PC, you must use the --device option to specify which you are targeting. You pass an identifier for the device, which can be listed using the --list-devices option. For example, CBProDeviceRead --list-devices might return:

```
00-00-16-B1-20-A2 (Si5345 EVB)
00-00-15-E2-E2-60 (Si5380 EVB)
```

You can then pass either 00-00-16-B1-20-A2 or 00-00-15-E2-E2-60 to the --device argument. For example:

```
CBProDeviceRead --device 00-00-15-E2-E2-60 --all
```

The program will exit with status code 0 on success, and 1 on error.

FIELD PROGRAMMER

You must configure communication settings when using a ClockBuilder Pro

CBPro CLI User's Guide

Field Programmer. The following options are available:

- family family-id
Device family. Required if field programmer wired to PCB using serial cable. Optional with socket.
- mode i2c|spi3wire|spi4wire
Communication mode. See below for when required.
- io-voltage 1.8|2.5|3.3
I/O supply voltage. See below for when required.
- speed 100k|400k|1M|2M|...|12M
Bus speed. Optional. The fastest speed is selected by default.
- i2c-address
I2C address. Required in I2C mode.

Family ID Reqd?	Description	Mode Req'd?	I/O Voltage
-----	-----	-----	-----
si538x4x	Si534x/7x/8x/9x (not firmware based)	Always	Always
si538x4xfw	Si5383/84 (firmware based)	Never	Never
si538x4xfw2	Si5388/89 (firmware based)	Never	Never
si5332	Si5332/57	Never	Always
si533x	Si5338/56	Never	Always
si55xx	Si55xx	Always	Never

EXAMPLE - READ ALL DEVICE SETTINGS, SAVE TEXT REPORT

```
C:\> CBProDeviceRead --all --outfile settings.txt
```

```
Searching for EVBs/FPs ...
Reading from device ...
Saving to settings.txt ...
```

```
C:\> more settings.txt
```

Location	Type	Setting Name	Decimal Value	Hex Value
-----	----	-----	-----	-----
0x0000[3:0]	R/O	DIE_REV	5	0x5
0x0002[15:0]	R/O	PN_BASE	21317	0x5345
0x0004[7:0]	R/O	GRADE	0	0x00
0x0005[7:0]	R/O	DEVICE_REV	3	0x03
...				
0x0B4A[4:0]	R/W	N_CLK_DIS	0	0x00

0x0B57[11:0] R/W VCO_RESET_CALCODE 270 0x10E

Reads all read-only and read-write settings from the device. Device readings are saved to settings.txt. Uses the default file format, a formatted text table.

EXAMPLE - READ ALL DEVICE SETTINGS, OUTPUT CSV

```
C:\> CBProDeviceRead --all --quiet --format csv
```

```
Location,Type,SettingName,DecimalValue,HexValue
0x0000[3:0],R/O,DIE_REV,5,0x5
0x0002[15:0],R/O,PN_BASE,21317,0x5345
0x0004[7:0],R/O,GRADE,0,0x00
0x0005[7:0],R/O,DEVICE_REV,3,0x03
...
0x0B4A[4:0],R/W,N_CLK_DIS,0,0x00
0x0B57[11:0],R/W,VCO_RESET_CALCODE,270,0x10E
```

Reads all read-only and read-write settings from the device. Device readings are output to the console in CSV format. Enables quiet mode, so that status messages are not displayed on the console.

EXAMPLE - READ SELECT DEVICE SETTINGS, OUTPUT TEXT REPORT

```
C:\> CBProDeviceRead.exe --quiet --settings PN_BASE DEVICE_REV
```

Location	Type	Setting Name	Decimal Value	Hex Value
0x0002[15:0]	R/O	PN_BASE	21317	0x5345
0x0005[7:0]	R/O	DEVICE_REV	3	0x03

Reads two device settings and outputs the results in a text report. Uses quiet mode to skip status messages.

EXAMPLE - READ SELECT DEVICE REGISTERS, OUTPUT TEXT REPORT

```
C:\> CBProDeviceRead.exe --quiet --registers 0x0002 0x0003 0x0005
```

Address	Decimal Value	Hex Value
0x0002	69	0x45
0x0003	83	0x53
0x0005	3	0x03

Reads three registers from the device. The registers read are the underlying registers for PN_BASE and DEVICE_REV in the previous example. Uses quiet mode to skip status messages.

EXAMPLE - READ ALL DEVICE SETTINGS FROM Si5345 DEVICE ON PCB USING FIELD PROGRAMMER

```
C:\> CBProDeviceRead --family si538x4x --mode i2c --i2c-address 0x6c
--io-voltage 3.3 --speed 400k --all --outfile state.txt
```

```
Searching for EVBs/FPs ...
Reading from device ...
Saving to state.txt ...
```

Reads all settings from device connected to the field programmer using an I2C serial connection. The device is at 7-bit address 0x6c. A 3.3V I/O supply voltage is used, and the I2C bus is driven at 400 kbps. The settings are saved to a text file, state.txt.

EXAMPLE - READ ALL DEVICE SETTINGS FROM SI5332 DEVICE ON PCB USING FIELD PROGRAMMER

```
C:\> CBProDeviceRead --family si5332 --i2c-address 0x6a --speed 400k --all
--outfile state.txt
```

```
Searching for EVBs/FPs ...
Reading from device ...
Saving to state.txt ...
```

Reads all settings from Si5332 device connected to the field programmer using an I2C serial connection. The device is at 7-bit address 0x6a. I/O supply voltage is auto set to 3.3V for this device. The I2C bus is driven at 400 kbps. The settings are saved to a text file, state.txt.

OPTIONS

--all

Read all read-only and read-write registers from the device. The output is setting oriented.

--device id

Select the Evaluation Board or Field Programmer to target. Required if there are 2 or more EVBs/FPs attached to the PC.

--family family-id

A device family that is being targeted. See FIELD PROGRAMMER section.

--format text|csv|tab|html

Output format. The default is a text table.

--help

Print detailed usage to the console and exit.

--i2c-address addr

7-bit I2C address when using the ClockBuilder Pro field programmer in I2C mode. There is no default. You can specify the address in 0xNN hex format or NN decimal.

`--io-voltage 1.8|2.5|3.3`

I/O voltage when using the ClockBuilder Pro field programmer. See FIELD PROGRAMMER section.

`--list-devices`

List Silicon Labs Timing devices -- Evaluation Boards and Field Programmers -- connected to the PC and exit.

`--mode i2c|spi3wire|spi4wire`

Communication mode when using the ClockBuilder Pro field programmer. See FIELD PROGRAMMER section.

`--noscan-in-fp-registers-mode`

Communication with the device using the field programmer is normally verified before an operation by reading fixed identification register(s) such as PN_BASE on Si538x/4x. Specifying this option will disable this behavior when --registers mode is used.

`--outfile pathname`

File to save results to. If not specified, device read results are displayed on standard output.

`--quiet`

Enable quiet mode. Normally status messages are printed to standard error. You can disable these using this option.

`--registers`

Read one or more registers from the device. Registers to dump are must be specified on the command line. Use 0x hex format, such as 0x000C.

`--settings`

Read one or more named settings (aka bitfields) from the device. Settings to dump are must be specified on the command line.

`--speed 100k|400k|1M|2M|...|12M`

I2C or SPI bus speed when using the ClockBuilder Pro field programmer. The default is 400 kHz in I2C mode and 12 MHz in SPI mode.

`--version`

Print this program's version number and exit.

CBProDeviceWrite

SYNOPSIS

```
CBProDeviceWrite [ options ] --project pathname
CBProDeviceWrite [ options ] --settings pathname
CBProDeviceWrite [ options ] --registers pathname
CBProDeviceWrite --list-devices
```

```
CBProDeviceWrite --help
```

DESCRIPTION

Writes a project file, settings file, or registers file to a supported Silicon Labs device (DUT). Writes are VOLATILE. The device can be located on a Silicon Labs Timing evaluation Board (EVB) or ClockBuilder Pro Field Programmer (FP), either in a socket or wired to the field programmer via serial cable. In each case, the operation is the exact same as if the corresponding tool was selected in the EVB GUI File menu. In the case of writing a project file, the behavior is also the same as when you click the "Write to EVB" or "Write to FP" button in the ClockBuilder Pro wizard.

If more than one EVB or FP is connected to your PC, you must use the --device option to specify which you are targeting. You pass an identifier for the device, which can be listed using the --list-devices option. For example, CBProDeviceWrite --list-devices might return:

```
00-00-16-B1-20-A2 (Si5345 EVB)
00-00-15-E2-E2-60 (Si5380 EVB)
```

You can then pass either 00-00-16-B1-20-A2 or 00-00-15-E2-E2-60 to the --device argument. For example:

```
CBProDeviceWrite --device 00-00-15-E2-E2-60
                  --project Si5380-RevD-TEST1.slabtimeproj
```

After the write has completed, a status message will be printed to the console indicating success or any error. The program will exit with status code 0 on success, and 1 on error.

VALIDATION

By default, DUT writes in all three modes are validated via DUT read back. With project file writes this occurs at the end of programming. With setting and register file writes, validation is performed after every setting or register write. Any self-clearing settings/registers are automatically not validated. For example, if an Si5380 register script contained a write to SOFT_RST, the read back would be skipped as this is a self-clearing setting that instructs the DSPLL to perform a soft reset, and a read always returns 0.

You can disable validation using the --no-validate command line option.

FIELD PROGRAMMER

CBPro CLI User's Guide

You must configure communication settings when using a ClockBuilder Pro Field Programmer. The following options are available:

- family family-id
Device family. Required if writing a register or settings file via serial cable. Optional in other cases as family can be inferred by the project file or socket.
- mode i2c|spi3wire|spi4wire
Communication mode. See below for when required.
- io-voltage 1.8|2.5|3.3
I/O supply voltage. See below for when required.
- speed 100k|400k|1M|2M|...|12M
Bus speed. Optional. The fastest speed is selected by default.
- i2c-address
I2C address. Required in I2C mode.

Family ID Reqd?	Description	Mode Reqd?	I/O Voltage
-----	-----	-----	
si538x4x	Si534x/7x/8x/9x (not firmware based)	Always	Always
si538x4xfw	Si5383/84 (firmware based)	Never	Never
si538x4xfw2	Si5388/89 (firmware based)	Never	Never
si5332	Si5332/57	Never	Always
si533x	Si5338/56	Never	Always
si55xx	Si55xx	Always	Never

SETTINGS FILE FORMAT

The general syntax is `setting_name<sep>setting_value`, where comma, space, or tab can be used as a separator. The value can be in hex or decimal notation. # and // comments are supported, either at the end or beginning of a line.

For example:

```
# Comma separator
DESIGN_ID0,65 # Decimal value (ASCII 'A')

// Space separator
DESIGN_ID1 0x42 // Hex value (ASCII 'B')
```

REGISTERS FILE FORMAT

The general syntax is `register_address<sep>data_value`, where comma, space, or tab can be used as a separator. Both address and decimal must be in hex notation. # and // comments are supported, either at the end or beginning of a line.

For example:

```
# Comma separator
0x026B,0x41 # DESIGN_ID0 = 'A'
```

```
// Space separator
0x026C 0x42 // DESIGN_ID1 = 'B'
```

```
// Hex is assumed for registers; this is same as above
026C 42
```

You can also specify a delay in microseconds in a register script. Some examples:

```
Pause 500
# Pause 500 msec
// Pause 500msec
Delay 500 msec
```

EXAMPLE - WRITE PROJECT FILE TO EVB

```
C:\> CBProDeviceWrite --project Si5345-RevD-TEST1-Project.slabtimeproj
```

```
Searching for EVBs/FPs ...
Loading project file ...
Writing project to Si5345 Rev D EVB ...
Validating write ...
Success
```

Writes a CBPro project file to an Evaluation Board.

EXAMPLE - WRITE REGISTER SCRIPT TO SI5383 DEVICE ON PCB USING FIELD PROGRAMMER

```
C:\> CBProDeviceWrite --registers Si5383-RevD-TEST2-Registers.txt --family
si538x4xfw --i2c-address 0x6c
```

```
Searching for EVBs/FPs ...
Writing registers on Field Programmer (with validation) ...
Success
```

Writes a register export script created by CBPro using the field programmer. Because the target family is si538x4xfw, only I2C address needs to be specified (the device family only supports I2C at 3.3V I/O voltage).

EXAMPLE - WRITE SETTINGS ON SI5346 DEVICE ON PCB USING FIELD PROGRAMMER


```
C:\> CBProDeviceWrite --settings settings.txt --mode i2c --i2c-address
0x7c --io-voltage 3.3 --family si538x4x
```

```
Searching for EVBs/FPs ...
Writing settings on Field Programmer (with validation) ...
Success
```

Writes a number of device settings (register bitfields) using the field programmer. Because the target family is si538x4x and it supports multiple serial protocols, --mode must be used to specify I2C mode. Likewise, I/O voltage must be configured using the --io-voltage option.

EXAMPLE - WRITE SETTINGS ON SI5332 DEVICE ON PCB USING FIELD PROGRAMMER

```
C:\> CBProDeviceWrite --settings settings.txt --i2c-address 0x6a --family
si5332
```

```
Searching for EVBs/FPs ...
Writing settings on Field Programmer (with validation) ...
Success
```

Writes a number of device settings (register bitfields) using the field programmer. Because the target family is si5332 and it only supports I2C, --mode is not required. Also, fixed 3.3V I/O voltage is used so --io-voltage is also not specified.

OPTIONS

--device id

Select the Evaluation Board or Field Programmer to target. Required if there are 2 or more EVBs/FPs attached to the PC.

--family family-id

A device family that is being targeted. See FIELD PROGRAMMER section.

--help

Print detailed usage to the console and exit.

--i2c-address addr

7-bit I2C address when using the ClockBuilder Pro field programmer in I2C mode. There is no default. You can specify the address in 0xNN hex format or NN decimal.

--io-voltage 1.8|2.5|3.3

I/O voltage when using the ClockBuilder Pro field programmer. See FIELD PROGRAMMER section.

--list-devices

List Silicon Labs Timing devices -- Evaluation Boards and Field Programmers -- connected to the PC and exit.

--mode i2c|spi3wire|spi4wire

Communication mode when using the ClockBuilder Pro field programmer. See FIELD PROGRAMMER section.

--no-validate

Normally, writes are validated via read back. Specify this option to disable. See VALIDATION section for details related to setting and register file modes.

--noscan-in-fp-registers-mode

Communication with the device using the field programmer is normally verified before an operation by reading fixed identification register(s) such as PN_BASE on Si538x/4x. Specifying this option will disable this behavior when --registers mode is used.

--project pathname

A project file to write to the device.

--registers pathname

A registers file to write to the device.

--settings pathname

A settings file to write to the device.

--speed 100k|400k|1M|2M|...|12M

I2C or SPI bus speed when using the ClockBuilder Pro field programmer. The default is 400 kHz in I2C mode and 12 MHz in SPI mode.

--version

Print this program's version number and exit.

CBProFOTF1 (Previously Named CBProSi534x8xFOTF)

SYNOPSIS

```
CBProFOTF1 [ opts ] --project pathname
           [ --plans-dspllX pathname ]
           [ --plans-nX pathname ]
```

```
CBProFOTF1 --help
```

DESCRIPTION

This tool is used to define alternate output frequency plans for a supported device. This is referred to as frequency-on-the-fly (FOTF). Currently Si5346/47/48/83/84/96/97 multi-DSPLL devices and Si5340/41/42/42H/44/44H/45/80/91/92/94/95 single-DSPLL / multi-N divider (MultiSynth) devices are supported.

For multi-DSPLL devices, an alternate frequency plan can mean changing one or all of the following:

- * Clock output frequency
- * Clock input frequency
- * DSPLL bandwidth
- * LOL thresholds
- * OOF thresholds

For single-DSPLL devices, only clock output frequency can be changed.

For multi-PLL devices Si5346/47/48/83/84/88/89/96/97, alternate plans are defined on a per-DSPLL basis, with changes made to that DSPLL guaranteed not to disturb outputs on another DSPLL. The key output of the tool is a number of register "scripts" that, when written to the device, will switch a DSPLL from one plan to another, with other DSPLLs unaffected.

Similarly, for single PLL-devices Si5340/41/42/42H/44/44H/45/71/72/80/91/92/94/95, alternate plans are defined on a per-N divider basis, with changes made to that N divider guaranteed not to disturb outputs on another N divider. Register scripts for each N divider are created.

Note: the CBProFOTF1 tool does not support Precision grades of the Si5391/2/4/5 such as P-Grade. These grades do not support manual N divider assignment for an output. Instead, CBPro selects the N divider that will achieve lowest possible jitter. Due to this, there is little flexibility to do FOTF. The CBProMultiEditAndExport or CBProProjectRegistersExport CLIs can be used to completely reconfigure a Precision grade device. See "CBPro Tools & Support for In-System Programming" available from the CBPro Welcome page to learn more.

The tool has two primary inputs:

- * A project file for the "base" design. All alternate plans will be derived from this base config. The --project command line option is

used specify the base project file.

- * One or more plan files. There is a file per-DSPLL or per-N divider. You don't have to define alternate plans for every DSPLL/N, however. --plans-dspla, --plans-dspllb, etc. and --plans-n0, --plans-n1, etc. command line options are used specify each plan file.

Usage is explained by working through an example in the following sections:

MULTI-DSPLL USAGE AND EXAMPLE

Si5346/47/48/83/84/96/97

SINGLE-DSPLL USAGE AND EXAMPLE

Si5340/41/42/42H/44/44H/45/80/91/92/94/95

Examples are bundled with CBPro in C:\Program Files (x86)\Silicon Laboratories\ClockBuilder Pro\CLI\Samples (or equivalent). The multi-PLL Si5347 example is in the FOTF-For-Multi-PLL-Device folder. The single PLL Si5345 example is in the FOTF-For-Single-PLL-Device folder. The output of the samples is included. If you want to experiment with the samples by re-running or editing the plan files, make a copy to your user area to avoid file system permission issues. Note any future update of CBPro will overwrite any changes you make under the CBPro installation folder.

```
*****
* IMPORTANT: the frequency plan and register settings for your "base" *
* design will usually be different when using the FOTF tool. The FOTF *
* tool is optimizing VCO frequency and divider values for ALL required *
* plan combinations, not simply the base design/project. Also, certain *
* device settings will be adjusted to support FOTF and will be        *
* reflected in the base configuration script.                            *
*                                                                       *
* Any settings on the "Configure Frequency Planner" page of CBPro     *
* (Si5340/41/42/44/45/91/92/94/95 and similar) will be ignored when   *
* creating doing FOTF. The tool will optimize for lowest jitter across *
* all plans.                                                           *
*                                                                       *
* Your embedded system must therefore write the base configuration    *
* script (Base-Plan-Register-Script.txt) generated by this tool to the *
* device when it is powered up or reset. This is true even if you     *
* generated an Orderable Part Number (OPN) using on the base project  *
* file.                                                                *
*****
```

EMBEDDED HOST WORKFLOW

On Silicon Labs device power up or reset, your host ****must**** write Base-Plan-Register-Script.txt. This will fully configure the part to synthesize the clocks defined in the base project.

After writing this base configuration, your host will only need to write one of the small plan scripts when it needs to switch one of the PLLs or

Nx dividers to another configuration. Plan scripts are created for the base config and each numbered alternate plan.

For example, for an Si5346/47/48/83/84/88/89/96/97:

- * Write Base-Plan-Register-Script.txt to fully configure the device for base plan and ready for further plan changes
- * Write DSPLLA-Register-Script-Plan1.csv to change DSPLL A outputs to plan #1
- * Write DSPLLB-Register-Script-Plan3.csv to change DSPLL B outputs to plan #3
- * Write DSPLLA-Register-Script-Base-Plan.csv to change DSPLL A outputs back to base plan

And for an Si5340/41/42/42H/44/44H/45/71/72/80/91/92/94/95:

- * Write Base-Plan-Register-Script.txt to fully configure the device for base plan and ready for further plan changes
- * Write N0-Register-Script-Plan1.csv to change N0 outputs to plan #1
- * Write N2-Register-Script-Plan4.csv to change N2 outputs to plan #4
- * Write N0-Register-Script-Base-Plan.csv to change N0 outputs back to base plan

CUSTOM PART NUMBER WARNING

The base frequency plan calculated by this tool may be different from what CBPro calculates for only the stand-alone project. The FOTF tool is optimizing for ALL required plan combinations, not simply the base design/project. If your host is doing FOTF via the DSPLL or N divider plan script files generated by this tool, your host must write the base plan script whenever the Si538x/4x is reset. The work flow is explained in the sections below.

TESTING WITH CBPRO

A key output of this tool are register write script files that your host can write to the Silicon Labs device via I2C or SPI. The EVB GUI can be used to test scripts on an EVB or when connected to your system board using the ClockBuilder Pro Field Programmer.

When the EVB GUI is launched, select "File->Write Register File to Device" to write a script file. Per the workflow described below, you would:

- (1) Write Base-Plan-Register-Script.txt
- (2) Write any of *-Register-Script-Plan*.csv or
-Register-Script-Base-Plan.csv (is wildcard)

Step 2 can be repeated to switch between different plans or back to the "base" plan.

Alternatively, the CBProDeviceWrite command line tool can be used to write a register script. For example:

Write to EVB:

CBPro CLI User's Guide

```
CBProDeviceWrite --registers Base-Plan-Register-Script.txt
```

Write to System Board via Field Programmer:

```
CBProDeviceWrite --registers --mode i2c --io-voltage 3.3
                  --i2c-address 0x6c --speed 400k
                  Base-Plan-Register-Script.txt
```

Both the EVB GUI's Write Register File feature and the CBProDeviceWrite tool will use the optional read-modify-write mask to perform the write. The mask is not populated for single PLL devices, but is for multi-PLL devices like the Si5346/7.

The format that these two tools accept is:

```
Address,Data[,Mask]
```

Spaces and tabs can also be used to separate fields. Comments are supported. For example:

```
// Comment
# Comment
0x009B,0x56,0xF0
0x009B 0x56 0xF0
0x00AA 0xAC # Comment; no read-modify-write mask in this
```

MULTI-DSPLL USAGE AND EXAMPLE

On multi-DSPLL devices, frequency-on-the-fly can only be performed on a PLL that has exclusive clock inputs. That is, an input to the FOTF PLL cannot also be MUX'd to another DSPLL. For example, given the following configuration:

```
IN0 -> DSPLL A
IN1 -> DSPLL A
IN2 -> DSPLL B
IN3 -> DSPLL C/D
```

FOTF can be performed on DSPLL A and B. FOTF is not supported on DSPLL C/D because IN3 is shared between these PLLs. The tool enforces this restriction, and will raise an error if you attempt FOTF on DSPLL C or D.

FOTF can technically mean not changing input or output frequencies and instead only reconfiguring one of OOF, LOS, or bandwidth. A plan file only has to reconfigure *at least one* of the following:

- * Clock output frequency
- * Clock input frequency
- * DSPLL bandwidth
- * LOL thresholds
- * OOF thresholds

There are some restrictions on Si5348/83/84/88/89 network synchronizers:

- * DSPLL B inputs, outputs, bandwidth, etc. cannot be adjusted
- * A 1 Hz output frequency cannot be set in a plan file; it can only be present in the base project (but can be switched to non-1 Hz in a plan)
- * On Si5383/84/88/89, if a 1 Hz input is present, DSPLL D cannot be adjusted
- * On Si5388/89, DSPLL D cannot be adjusted unless in network synchronizer mode

The plan file format is best explained through some examples. Continuing with the scenario above, the following DSPLL to output MUX is defined:

```
DSPLL A -> OUT0/OUT1
DSPLL B -> OUT2/OUT3
DSPLL C -> OUT4/OUT5
DSPLL D -> OUT6/OUT7
```

Here is a sample DSPLL A plan file for this configuration. Three plans are defined:

```
# DSPLL A Plans

Plan,Item,Value

1,IN0,155.52M
1,IN1,155.52M
1,OUT0,148.5M
1,OUT1,27M
1,OLBW,20
1,FLBW,100
1,LOL_SET_THR,30
1,LOL_CLR_THR,3
1,OOF_SET_THR0,10
1,OOF_CLR_THR0,1
1,FAST_OOF_SET_THR0,2000
1,FAST_OOF_CLR_THR0,1000

2,IN0,19.44M
2,IN1,19.44M
2,OUT0,155.52M
2,OUT1,622.08M
2,LOL_SET_THR,300
2,LOL_CLR_THR,30

3,IN0,155.52*255/236/16M
3,IN1,155.52*255/236/16M
3,OUT0,168+21/512M
3,OUT1,672+21/128M
```

Note how what is reconfigured in each numbered plan does not have to be the same as another plan. For example, plan #1 changes input frequency, output frequency, bandwidth, LOL, and OOF settings while plan #2 only changes input/output frequencies and LOL settings. Those items that are not reconfigured will inherit settings from the base project (*not* the

lower numbered plan).

In the DSPLL B plan file, six plans defined that simply reconfigure bandwidth between plans:

```
# DSPLL B Plans
```

```
Plan,Item,Value
```

```
# Plan #1
```

```
1,OLBW,20
```

```
1,FLBW,100
```

```
# Plan #2
```

```
2,OLBW,40
```

```
2,FLBW,1000
```

```
# Plan #3
```

```
3,OLBW,100
```

```
3,FLBW,1000
```

```
# Plan #4
```

```
4,OLBW,200
```

```
4,FLBW,1000
```

```
# Plan #5
```

```
5,OLBW,400
```

```
5,FLBW,1000
```

Rules for plan files:

- * Comments are supported via # syntax.
- * Blank lines are skipped.
- * The header "Plan,Item,Value" must be present.
- * Plans are numbered starting at 1, and numbering must be consecutive.
- * There is no limit on number of plans, other than that imposed by computation time and virtual memory used by the tool to solve for your plans.
- * The number of plans does not have to be equal between DSPLLs: there is no relationship between plan #1 in DSPLL A and plan #1 in DSPLL B, and therefore you can vary the number of plans between DSPLLs.
- * You DO NOT have to specify an inputs or outputs: the base design's clocks will be used if not defined.
- * You DO NOT have to specify bandwidth: the base design's values will be used by default.
- * If at least one input is specified on a DSPLL plan, all inputs on that DSPLL must be defined.
- * If at least one output is specified on a DSPLL plan, all outputs on that DSPLL must be defined.
- * You can change open loop bandwidth using the item OLBW; you can change Fastlock bandwidth using the item FLBW; if you specify one, you must specify the other.
- * Exit from Holdover bandwidth may be specified if used by the device revision/configuration. The HOBW keyword is used. The CLI

- will generate an error if not specified when required, such as OLBW was specified and ramping has been disabled.
- * LOL thresholds for a DSPLL can optionally be changed using the LOL_SET_THR and LOL_CLR_THR keywords. Decimal values are accepted. Can only be used if the base project file has "Auto Set" unchecked.
 - * OOF thresholds for an input can be set using the OOF_SET_THRx, OOF_CLR_THRx (precision OOF) and FAST_OOF_SET_THRx, FAST_OOF_CLR_THRx (fast OOF) keywords, where x is the INx input number starting at 0. Thresholds can only be specified if the Enabled checkbox is set on the corresponding input. Decimal values are supported for precision OOF and integer values for fast OOF.
 - * Changes made to a previous plan have no effect on a successive plan. For example, in DSPLL A plan #1, the base design's bandwidth is changed. In DSPLL B plan #2, there is no entry for bandwidth. The bandwidth will be therefore be defined based on the base design, NOT plan #1.
 - * The same frequency expressions used in CBPro can be used in the plan files, with the exception that you cannot reference another clock in the system (i.e. an expression such as OUT0*2 is not supported).
 - * Item keywords such as IN1, OUT0, and OLBW are case insensitive.

Given above, the general requirement for a plan is that it must specify at least one change to input frequency, output frequency, bandwidth, LOL, or OOF for a DSPLL. The following are therefore allowed plans for our ongoing DSPLL A example:

```
Plan,Item,Value

1,IN0,155.52*255/236/16M

2,OUT0,155.52M
2,OUT1,622.08M

3,OLBW,20
3,FLBW,100

4,LOL_SET_THR,30
4,LOL_CLR_THR,3

5,OOF_SET_THR0,10
5,OOF_CLR_THR0,1
5,FAST_OOF_SET_THR0,2000
5,FAST_OOF_CLR_THR0,1000
```

The tool does NOT support doing FOTF on a DSPLL if ****ALL**** of the following conditions are true:

- * The device is Revision B
- * An input to the DSPLL is shared with another DSPLL
- * The "optimize plan for zero ppm frequency offset" setting in the CBPro "Define Output Frequencies" wizard page is checked

The tool will check for these conditions and will abort with a fatal error if true. You can uncheck the "optimize plan for zero ppm frequency offset"

checkbox and re-save your project file to use the FOTF tool. Please review <http://www.silabs.com/Support%20Documents/TechnicalDocs/Si534x-Applications%20Notice-Part-Per-Trillion%20Frequency-Offset.pdf> before making this change.

To create the alternate frequency plans for DSPLL A & B, the CBProFOTF1.exe command line tool is run:

```
CBProFOTF1 --project Si5347-RevD.slabtimeproj
           --out-folder Output --create-out-folder
           --plans-dspla DSPLLA.txt --plans-dspllb DSPLLB.txt
```

This will calculate the best base frequency plan and alternate frequency plans. Status messages will be logged to the console:

```
Loading project Si5347-RevD-Project.slabtimeproj ...
Output files will be saved in "Output"
Solving for shared PFD frequency(s) based on shared input(s) on DSPLL
...
Success
DSPLL A Fpfd will allowed to vary between plans
DSPLL B Fpfd will allowed to vary between plans
DSPLL C Fpfd will be fixed at 1.944 MHz [ 1 + 118/125 MHz ]
across plans
DSPLL D Fpfd will be fixed at 1.944 MHz [ 1 + 118/125 MHz ]
across plans
Calculating base project frequency plan ...
Saving Base design files ...
Calculating DSPLLA Plan #1
Calculating DSPLLA Plan #2
Calculating DSPLLA Plan #3
Saving DSPLLA files ...
Calculating DSPLLB Plan #1
Calculating DSPLLB Plan #2
Calculating DSPLLB Plan #3
Calculating DSPLLB Plan #4
Calculating DSPLLB Plan #5
Calculating DSPLLB Plan #6
Saving DSPLLB files ...
Done; 5 seconds elapsed
```

It is important to note that the base frequency plan calculated by this tool may be different from what CBPro calculates for only the stand-alone project. Why? The FOTF tool is optimizing Fpfd for ALL required plan combinations, not simply the base design/project.

The output files created by the tool will be described below. The general work flow of a host doing in-system update to these alternate frequency plans is:

- * Write the complete register script for the base configuration. This needs to be done even if your device has an OPN (factory) configuration, since the base design frequency plan calculated by the FOTF tool may differ from that calculated by CBPro in

stand-alone mode.

- * Write a DSPLL plan script to switch a particular DSPLL to an alternate plan (or back to the base plan).

When the example command sequence is run, the following files are created:

Base-Plan-Design-Report.txt

Design report for the base design. This report may differ from what CBPro would generate for the same base project file.

Base-Plan-Register-Script.txt

A sequence of register writes to perform to load the base configuration. This may be different from what CBPro would export for the same base project file. ** YOUR HOST MUST WRITE THIS SCRIPT WHEN THE DEVICE IS POWERED ON OR RESET. **

DSPLLA-Register-Script-Base-Plan.csv

DSPLLA-Register-Script-Plan1.csv

DSPLLA-Register-Script-Plan2.csv

DSPLLA-Register-Script-Plan3.csv

Register writes to perform to switch to a plan on DSPLL A. This includes all necessary control register writes required to fully load the new configuration (such as divider UPDATE writes and DSPLL soft reset).

The format is:

Address,Value,Mask

With all values 0x hexadecimal syntax.

Mask is new as of CBPro 2.20.3. If mask is 0xFF, no read-modify-write is required. Otherwise, read register value and write to device $\text{Read} \& \sim \text{Mask} + \text{Value} \& \text{Mask}$.

Read-modify-write is only required for a small number of registers where settings for multiple DSPLLs are stored at a single address (register).

Pseudo code implementation of handling mask:

```
UInt16 address = row[0]
UInt8 value = row[1]
UInt8 mask = row[2]

If (mask != 0xFF)
{
    read = Read_Register(address)
    value = read&~mask + value&mask
}
```

Write_Register(address, value)

A comment header at the start of the script summarizes the configuration. Example:

```
# DSPLLA Register Script - Base-Plan
#
# Base Project: Si5347-RevD-Project.slabtimeproj
# Part: Si5347
# Design ID: <none>
# Inputs Vary Between Plans: Yes
# Created By: CBProFOTF1 v2.20.2.101 [2018-01-09]
# Timestamp: 2018-01-09 16:18:46 GMT-06:00
#
# IN0: 25 MHz
#      on DSPLL A
# OUT0: 161.1328125 MHz
# OUT1: 644.53125 MHz
#
# Fpfd = 1.9230769230769230... MHz [ 1 + 12/13 MHz ]
# Fdco = 1.2890625 GHz [ 1 + 37/128 GHz ]
#
# P0 = 13/1
# MA = 670.3125 [ 670 + 5/16 ]
# N0 = 11.0219636363636363... [ 11 + 151/6875 ]
# R0 = 8
# R1 = 2
#
# Nominal Bandwidth:
#   Desired: 80.000 Hz
#   Actual: 77.716 Hz
#   Coefficients:
#     BW0: 15
#     BW1: 30
#     BW2: 13
#     BW3: 12
#     BW4: 3
#     BW5: 63
# Fastlock Bandwidth:
#   Desired: 1.000 kHz
#   Actual: 1.246 kHz
#   Coefficients:
#     BW0: 19
#     BW1: 40
#     BW2: 9
#     BW3: 8
#     BW4: 3
#     BW5: 63
# Holdover Bandwidth:
#   N/A (Ramped Exit from Holdover)
#
# LOL:
#   Set Threshold: 10 ppm
```

```
# Clear Threshold: 1 ppm
#
# Precision OOF:
#
# Input   Enable   Assert Threshold   De-Assert Threshold
# -----
# IN0     Enabled  100.0000 ppm      98.0000 ppm
#
# Fast OOF:
#
# Input   Enable   Assert Threshold   De-Assert Threshold
# -----
# IN0     Enabled  4000 ppm          3000 ppm
#
Address,Value,Mask
# If mask is 0xFF, no read-modify-write is required. Otherwise,
# read register value and write to device Read&~Mask + Value&Mask
# Configuration Preamble
0x010C,0x00,0xFF
0x0116,0x00,0xFF
0x0B3C,0x00,0xFF
0x0B3D,0x00,0xFF
0x042C,0x86,0xFF
0x0436,0x00,0xFF
0x002C,0x0E,0xFF
0x003F,0xEE,0xFF
0x0092,0x0E,0xFF
0x009A,0x0E,0xFF
# Write Configuration
0x002E,0x3C,0xFF
0x0036,0x3C,0xFF
0x0041,0x0B,0xFF
0x0046,0x32,0xFF
0x004A,0x31,0xFF
0x0051,0x03,0xFF
0x0055,0x02,0xFF
0x005A,0xAA,0xFF
0x005B,0xAA,0xFF
0x005C,0x0A,0xFF
0x005D,0x01,0xFF
0x0096,0x88,0x0F
0x0098,0x66,0x0F
0x009B,0x66,0x0F
...
0x04A0,0x0C,0xFF
0x04A1,0x03,0xFF
# Configuration Postamble
0x0420,0x01,0xFF
0x030C,0x01,0xFF
0x0414,0x01,0xFF
0x0230,0x01,0xFF
0x0092,0x0F,0xFF
0x009A,0x0F,0xFF
0x002C,0x0F,0xFF
```

```
0x003F,0xFF,0xFF
0x0436,0x04,0xFF
0x042C,0x87,0xFF
0x010C,0x01,0xFF
0x0116,0x01,0xFF
0x001C,0x02,0xFF
0x0B3C,0xFF,0xFF
0x0B3D,0x00,0xFF
```

DSPLLA-Report.txt

A custom frequency-on-the-fly report for DSPLL A. This summarizes the key design goals and frequency plan elements for each plan, including the base plan. For example:

DSPLLA Frequency-On-The-Fly Report

```
Base Project: Si5347-RevD-Project.slabtimeproj
Part: Si5347
Design ID: <none>
Inputs Vary Between Plans: Yes
Created By: CBProFOTF1 v2.20.2.101 [2018-01-09]
Timestamp: 2018-01-09 16:18:46 GMT-06:00
```

Base Plan

=====

```
IN0: 25 MHz
      on DSPLL A
OUT0: 161.1328125 MHz
OUT1: 644.53125 MHz
```

```
Fpfd = 1.9230769230769230... MHz [ 1 + 12/13 MHz ]
Fdco = 1.2890625 GHz [ 1 + 37/128 GHz ]
```

```
P0 = 13/1
MA = 670.3125 [ 670 + 5/16 ]
N0 = 11.0219636363636363... [ 11 + 151/6875 ]
R0 = 8
R1 = 2
```

Nominal Bandwidth:

[excluded in this example]

Fastlock Bandwidth:

[excluded in this example]

Holdover Bandwidth:

N/A (Ramped Exit from Holdover)

LOL:

```
Set Threshold: 10 ppm
Clear Threshold: 1 ppm
```

Precision OOF:

```
Input  Enable  Assert Threshold  De-Assert Threshold
```

```
-----
IN0      Enabled  100.0000 ppm      98.0000 ppm
```

Fast OOF:

```
Input  Enable  Assert Threshold  De-Assert Threshold
-----
IN0     Enabled  4000 ppm          3000 ppm
```

Plan #1

=====

IN0: 155.52 MHz

on DSPLL A

OUT0: 148.5 MHz

OUT1: 27 MHz

Fpfd = 1.9938461538461538... MHz [1 + 323/325 MHz]

Fdco = 1.188 GHz [1 + 47/250 GHz]

P0 = 78/1

MA = 595.8333333333333... [595 + 5/6]

N0 = 11.95959595959595... [11 + 95/99]

R0 = 8

R1 = 44

...

****NOTE**:** if any outputs in the plan may have worse case jitter approaching 400 fs, it will be flagged in the plan mini report. For example:

Warning: worse case jitter may approach 400 fs on OUT0,OUT1

DSPLLA-Settings-Diff-All-Plans.csv

A breakdown of what register settings are different between each plan on DSPLL A. This is for informational/debug use. It does NOT include plan update control register settings, such as MA_FUPDATE and DSPLLA_SOFTRESET.

A similar set of files is created for DSLL B. In our example, there are more plans for this DSPLL. There are therefore additional columns in the "All" files and additional broken out register script files:

DSPLLB-Register-Script-Base-Plan.csv

DSPLLB-Register-Script-Plan1.csv

DSPLLB-Register-Script-Plan2.csv

DSPLLB-Register-Script-Plan3.csv

DSPLLB-Register-Script-Plan4.csv

DSPLLB-Register-Script-Plan5.csv

DSPLLB-Register-Script-Plan6.csv

DSPLLB-Report.txt

DSPLLB-Settings-Diff-All-Plans.csv

An example sequence of activities by a host could be:

1. Write Base-Plan-Register-Script.txt at system startup, reset, etc.: any time NVM would be reloaded by the Si538x/4x device. The base input/output/bandwidth configuration will be operational after this.
2. Write DSPLLA-Register-Script-Plan2.csv to switch DSPLL A to plan #2. DSPLL B will be unaffected.
3. Write DSPLLB-Register-Script-Plan6.csv to switch DSPLL B to plan #6. DSPLL A will be unaffected.

If the desired base + alternate plans can be achieved, the tool exits with status code 0. If it cannot, a message will be printed to standard error and the tool exits with status code 1.

The example described above is available in C:\Program Files (x86)\Silicon Laboratories\ClockBuilder Pro\CLI\Samples\Multi-PLL-F0TF (or equivalent).

SINGLE-DSPLL USAGE AND EXAMPLE

In order to use this tool, all outputs in a design must be manually assigned to an N divider. The tool checks your design and will exit with an error if this is not the case.

The plan file format and tool work flow will be explained through an Si5345 example. The Si5345 is a 10-channel, any-frequency, any-output jitter attenuator/clock multiplier. It has 5 N dividers (aka MultiSynths). The base project has the following divider assignment:

```
N0:
  OUT0: 161.1328125 MHz [ 161 + 17/128 MHz ]
  OUT1: 322.265625 MHz [ 322 + 17/64 MHz ]
N1:
  OUT2: 155.52 MHz [ 155 + 13/25 MHz ]
  OUT3: 311.04 MHz [ 311 + 1/25 MHz ]
N2:
  OUT4: 148.5 MHz [ 148 + 1/2 MHz ]
  OUT5: 148.5 MHz [ 148 + 1/2 MHz ]
  OUT6: 148.5 MHz [ 148 + 1/2 MHz ]
  OUT7: 148.5 MHz [ 148 + 1/2 MHz ]
  OUT8: 148.5 MHz [ 148 + 1/2 MHz ]
  OUT9: 148.5 MHz [ 148 + 1/2 MHz ]
N3:
  Unused
N4:
  Unused
```

Here is a sample N0 plan file for this configuration. Only one plan is defined:

```
# N0 Plans

Plan,Item,Value
```



```
# Plan #1
1,OUT0,155.52M
1,OUT1,622.08M/2
```

And N1 plan file, with one plan defined:

```
# N1 Plans

Plan,Item,Value

# Plan #1
1,OUT2,161+17/128M
1,OUT3,644.53125M/2
```

Finally, the N2 plan file defines 5 different plans:

```
# N2 Plans

Plan,Item,Value

# Plan #1
1,OUT4,155.52*255/237M
1,OUT5,155.52*255/237M
1,OUT6,155.52*255/237M
1,OUT7,155.52*255/237M
1,OUT8,155.52*255/237M
1,OUT9,155.52*255/237M

# Plan #2
2,OUT4,156.25M
2,OUT5,156.25M
2,OUT6,156.25M
2,OUT7,156.25M
2,OUT8,156.25M
2,OUT9,156.25M

# Plan #3
3,OUT4,132.8125M
3,OUT5,132.8125M
3,OUT6,132.8125M
3,OUT7,132.8125M
3,OUT8,132.8125M
3,OUT9,132.8125M

# Plan #4
4,OUT4,159.375M
4,OUT5,159.375M
4,OUT6,159.375M
4,OUT7,159.375M
4,OUT8,159.375M
4,OUT9,159.375M

# Plan #5
5,OUT4,156.25*66/64*255/237M
```

```
5,OUT5,156.25*66/64*255/237M
5,OUT6,156.25*66/64*255/237M
5,OUT7,156.25*66/64*255/237M
5,OUT8,156.25*66/64*255/237M
5,OUT9,156.25*66/64*255/237M
```

In these examples, N0 and N1 support switching between two plans: base and plan #1. N2 supports switching between 6 plans: base and plan 1-5.

Rules for plan files:

- * Comments are supported via # syntax.
- * Blank lines are skipped.
- * The header "Plan,Item,Value" must be present.
- * Plans are numbered starting at 1, and numbering must be consecutive.
- * There is no limit on number of plans, other than that imposed by computation time and virtual memory used by the tool to solve for your plans.
- * The number of plans does not have to be equal between N dividers: there is no relationship between plan #1 in N0 and plan #1 in N1, and therefore you can vary the number of plans between N dividers.
- * Bandwidth and input frequency cannot be modified because these are global to the device configuration.
- * All outputs on the N divider must be defined for each plan.
- * The same frequency expressions used in CBPro can be used in the plan files, with the exception that you cannot reference another clock in the system (i.e. an expression such as OUT0*2 is not supported).
- * Clock names (OUT0, OUT1, etc.) are case insensitive.

To create the alternate frequency plans for N0, N1, and N2 the CBProFOTF1.exe command line tool is run:

```
CBProFOTF1 --project Si5345-RevB-Project.slabtimeproj
           --out-folder Output --create-out-folder
           --plans-n0 N0.txt --plans-n1 N1.txt
           --plans-n2 N2.txt
```

This will calculate the best base frequency plan and alternate frequency plans. Status messages will be logged to the console:

```
Output files will be saved in "Output"
Solving for shared VCO frequency ...
Success
Calculating base project frequency plan ...
Saving Base design files ...
Calculating N0 Plan #1
Saving N0 files ...
Calculating N1 Plan #1
Saving N1 files ...
Calculating N2 Plan #1
Calculating N2 Plan #2
Calculating N2 Plan #3
Calculating N2 Plan #4
Calculating N2 Plan #5
```

```
Saving N2 files ...  
Done; 5 seconds elapsed
```

The output files created by the tool will be described below. The general work flow of a host doing in-system update to these alternate frequency plans is:

- * Write the complete register script for the base configuration. This needs to be done even if your device has an OPN (factory) configuration, since the base design frequency plan calculated by the FOTF tool may differ from that calculated by CBPro in stand-alone mode.
- * Write a N divider plan script to switch a particular N divider's outputs to an alternate plan (or back to the base plan).

When the example command sequence is run, the following files are created:

Base-Plan-Design-Report.txt

Design report for the base design. This report may differ from what CBPro would generate for the same base project file.

Base-Plan-Register-Script.txt

A sequence of register writes to perform to load the base configuration. This may be different from what CBPro would export for the same base project file. ** YOUR HOST MUST WRITE THIS SCRIPT WHEN THE DEVICE IS POWERED ON OR RESET. **

N0-Register-Script-All-Plans.csv

Register writes to perform to switch to a plan on N0. This includes all necessary control register writes required to fully load the new configuration (such as Nx_UPDATE writes).

```
Address,Base,Plan1  
# Write Configuration  
0x0303,0x00,0x80  
0x0304,0xC0,0xFD  
0x0305,0x75,0x1E  
0x030A,0xE0,0x00  
0x030B,0xAB,0xA2  
# Load Configuration  
0x030C,0x01,0x01
```

All files contain comments starting with the # character.

N0-Register-Script-Base-Plan.csv

N0-Register-Script-Plan1.csv

This contains the same data as N0-Register-Script-All-Plans.csv, but broken out by unique file for each plan. The file also contains a synopsis of the design goals and frequency plan as a

comment. For example:

```
# N0 Register Script - Base-Plan
#
# Base Project: Si5345-RevB-Project.slabtimeproj
# Part: Si5345
# Design ID: <none>
# Inputs Vary Between Plans: No
# Created By: CBProFOTF1 v2.8.7.1 [2016-06-17]
# Timestamp: 2016-06-20 17:54:43 GMT-05:00
#
# OUT0: 161.1328125 MHz
# OUT1: 322.265625 MHz
#
# Fms0 = 644.53125 MHz [ 644 + 17/32 MHz ]
# N0 = 21.537454545454545... [ 21 + 739/1375 ]
#
# R0 = 4
# R1 = 2
#
# Write Configuration
Address,Value
0x0303,0x00
0x0304,0xC0
0x0305,0x75
0x030A,0xE0
0x030B,0xAB
# Load Configuration
0x030C,0x01
```

N0-Report.txt

A custom frequency-on-the-fly report for N0. This summarizes the key design goals and frequency plan elements for each plan, including the base plan. For example:

N0 Frequency-On-The-Fly Report

```
Base Project: Si5345-RevB-Project.slabtimeproj
Part: Si5345
Design ID: <none>
Inputs Vary Between Plans: No
Created By: CBProFOTF1 v2.8.7.1 [2016-06-17]
Timestamp: 2016-06-20 17:54:43 GMT-05:00
```

Base Plan

=====

```
OUT0: 161.1328125 MHz
OUT1: 322.265625 MHz
```

```
Fms0 = 644.53125 MHz [ 644 + 17/32 MHz ]
N0 = 21.537454545454545... [ 21 + 739/1375 ]
```

```
R0 = 4
```

R1 = 2

Plan #1

=====

OUT0: 155.52 MHz

OUT1: 311.04 MHz

Fms0 = 622.08 MHz [622 + 2/25 MHz]

N0 = 22.3147545331790123... [22 + 26107/82944]

R0 = 4

R1 = 2

****NOTE**:** If any outputs in the plan may have worse case jitter approaching 400 fs, it will be flagged in the plan mini report. For example:

Warning: worse case jitter may approach 400 fs on OUT0,OUT1

N0-Settings-Diff-All-Plans.csv

A breakdown of what register settings are different between each plan on N0. This is for informational/debug use. It does NOT include plan update control register settings, such as N0_FUPDATE.

A similar set of files is created for N1 and N2. In our example, there are more plans for N2. There are therefore additional columns in the "All" files and additional broken out register script files:

N1-Register-Script-All-Plans.csv
N1-Register-Script-Base-Plan.csv
N1-Register-Script-Plan1.csv
N1-Report.txt
N1-Settings-Diff-All-Plans.csv

N2-Register-Script-All-Plans.csv
N2-Register-Script-Base-Plan.csv
N2-Register-Script-Plan1.csv
N2-Register-Script-Plan2.csv
N2-Register-Script-Plan3.csv
N2-Register-Script-Plan4.csv
N2-Register-Script-Plan5.csv
N2-Report.txt
N2-Settings-Diff-All-Plans.csv

An example sequence of activities by a host could be:

1. Write Base-Plan-Register-Script.txt at system startup, reset, etc.: any time NVM would be reloaded by the Si538x/4x device. The base input/output/bandwidth configuration will be operational after this.
2. Write N0-Register-Script-Plan1.csv to switch N0 to plan

- #1. N1 and N2 will be unaffected.
3. Write N2-Register-Script-Plan5.csv to switch N2 to plan
- #5. N0 and N1 will be unaffected.

If the desired base + alternate plans can be achieved, the tool exits with status code 0. If it cannot, a message will be printed to standard error and the tool exits with status code 1.

The example described above is available in C:\Program Files (x86)\Silicon Laboratories\ClockBuilder Pro\CLI\Samples\Single-PLL-FOTF (or equivalent).

OPTIONS

--create-out-folder

Create the output folder does not exist already.

--help

Print detailed usage to the console and exit.

--out-folder folder

The output folder. If this is not specified, the folder where the tool is run is used.

--outfile-prefix string

A prefix to append to every file created by the tool. By default, there is no prefix and files like DSPLL-Report.txt and Registers-Base.txt are created. If you specified --outfile-prefix "Design1", these files would be named Design1-DSPLL-Report.txt and Design1-Registers-Base.txt.

--plans-dsplla pathname

The plan scenarios for DSPLL A are defined in this file. See PLAN FILES for details. You only need to specify this for a DSPLL you want to make in-system changes to.

--plans-dspllb pathname

The plan scenarios for DSPLL B are defined in this file. See PLAN FILES for details. You only need to specify this for a DSPLL you want to make in-system changes to.

--plans-dspllc pathname

The plan scenarios for DSPLL C are defined in this file. See PLAN FILES for details. You only need to specify this for a DSPLL you want to make in-system changes to.

--plans-dsplld pathname

The plan scenarios for DSPLL D are defined in this file. See PLAN FILES for details. You only need to specify this for a DSPLL you want to make in-system changes to.

--plans-n0 pathname

The plan scenarios for N0 are defined in this file. See PLAN FILES for details. You only need to specify this for an N divider you want to make in-system changes to.

--plans-n1 pathname

The plan scenarios for N1 are defined in this file. See PLAN FILES for details. You only need to specify this for an N divider you want to make in-system changes to.

--plans-n2 pathname

The plan scenarios for N2 are defined in this file. See PLAN FILES for details. You only need to specify this for an N divider you want to make in-system changes to.

--plans-n3 pathname

The plan scenarios for N3 are defined in this file. See PLAN FILES for details. You only need to specify this for an N divider you want to make in-system changes to.

--plans-n4 pathname

The plan scenarios for N4 are defined in this file. See PLAN FILES for details. You only need to specify this for an N divider you want to make in-system changes to.

--project pathname

The base configuration project file. You are modifying this design. The underlying file will never be changed. You must specify this.

--unlock password

Provides Silicon Labs employees access to extended reporting in tool output.

--version

Print this program's version number and exit.

CBProMultiEditAndExport

SYNOPSIS

```
CBProMultiEditAndExport [ opts ] --project base.slabtimeproj  
                        edits1.txt [ edits2.txt ... ]
```

```
CBProMultiEditAndExport --help
```

DESCRIPTION

Combines the features of CBProProjectEdit and CBProMultiProjectExport into a single tool:

- You define a base project file
- You define alternate clock configurations in the form of simple text file(s)
- You run this tool to create new (edited) project files and export files for both the base project file and created project files

Clock frequencies and state can be defined in the edit files. On supported devices, DSPLL bandwidth and assignment can also be defined. Editing is done via a simple text configuration file you specify on the command line.

See IN-SYSTEM PROGRAMMING WORKFLOW section for recommended ways for a host microprocessor to use these exports to switch between configurations in-system.

EXAMPLE

```
CBProMultiEditAndExport --out-folder Output --create-out-folder  
                        --project Si5345-Base-Project.slabtimeproj  
                        Si5345-Edits1.txt Si5345-Edits2.txt
```

Output files are placed in the folder named Output, creating it if necessary. Two edit files are applied to the base project file, resulting in three sets of projects/exports labeled P1 (Si5345-Base-Project.slabtimeproj), P2 (base+Si5345-Edits1.txt), and P3 (base+Si5345-Edits2.txt).

The base project gets resaved as P1-Project.slabtimeproj, with frequency plan recalculated with latest algorithms. P2-Project.slabtimeproj and P3-Project.slabtimeproj with the edits are created.

Design reports (P*-Report.txt), register write scripts (P*-Registers-Script.txt), and setting/bitfield dumps (P*-Settings.txt) are created for each project.

Multi-project export files are created for P1-P3. Registers.csv contains register values for all projects and Settings.csv contains all setting/bitfield values.

This example is available in Multi-Edit-And-Export-For-Single-PLL-Device within the C:\Program Files (x86)\Silicon Laboratories\ClockBuilder Pro\CLI\Samples folder. A second example for a multi-PLL Si5347 is in the Multi-Edit-And-Export-For-Multi-PLL-Device Samples folder.

EDIT FILE

You must specify at least one edit file on the command line. This file will define what changes you want to make to one or more clock inputs and/outputs, as well as bandwidth on supported devices. The format of the file varies depending on whether the device in your project file has a single or multiple DSPLLs.

If a single DSPLL device:

```
Clock,State,Frequency
IN0,Enabled,19.44 MHz
IN1,Unused
OUT0,Enabled,25*(825/128)M
OUT1,Disabled,625M
OUT5,Unused
```

The header (Clock,State,Frequency) is required. Comments via lines starting with # are allowed, as are blank lines. Inputs can have one of two states: Enabled or Unused. Outputs have three possible states: Enabled, Disabled, or Unused. You ****do not**** have to list every input or output clock: any not listed will retain the setting in the project file.

If a multiple DSPLL device:

```
Clock,State,DSPLL,Frequency
IN0,Enabled,ABCD,19.44 MHz
IN1,Unused
OUT0,Enabled,A,25*(825/128)M
OUT1,Enabled,B,625M
OUT5,Unused
```

This adds a column for DSPLL assignment to the format. Note that in the case of an input, you list all DSPLLs that the input will feed.

On select devices, DSPLL bandwidth can also be changed by specifying OLBW (nominal), FLBW (fastlock), or HOBW (holdover exit) as a clock name.

Example for single DSPLL device:

```
OLBW,,100 Hz
FLBW,,1k
```

Example for multi-DSPLL device:

```
OLBW,,A,100
FLBW,,A,1k
OLBW,,B,200
FLBW,,B,2k
```

You can also specify bandwidth for multiple DSPLLs at once:

```
OLBW,,AB,100 Hz
FLBW,,AB,1 kHz
```

EXPORT FILES

```
Registers.csv
-----
```

Configuration register values for each project file. Example:

```
Address,Varies,P1,P2,P3
...
0x043D,Yes,0x12,0x12,0x0A
0x043E,No,0x06,0x06,0x06
0x0442,No,0x00,0x00,
0x0443,No,0x00,0x00,
0x0444,No,0x00,0x00,
...
```

The first column, labeled Address, is the two byte hex encoded register address. The second column, labeled Varies, indicates whether the current register has the same value for all projects included in the export (No) or at least two projects differ in value (Yes). Columns three onward contain the single byte register value for each project. Projects get labeled P1 onward using the order that projects were specified in the command line.

If a register is not written (not defined and a don't care) by one project but is defined by another, the value cell will be empty for the project(s) that do not define the register. These empty cells will not count towards the Varies computation. For example, in the 0x0442-0x0444 example above project 3 does not define the register so the value is empty. Project 1 and 2 have the same value. So Varies is No and this register need not be written when switching from one configuration to another.

```
Settings.csv
-----
```

This breaks out registers by setting (aka bitfield). Example:

```
Location,Setting,Varies,P1,P2,P3
...
0x043D[4:0],HSW_COARSE_PM_LEN_PLLA,Yes,0x12,0x12,0xA
0x043E[4:0],HSW_COARSE_PM_DLY_PLLA,No,0x6,0x6,0x6
0x0442[17:0],SLAB_OLA_FINE_ADJ_OVR,No,0x0,0x0,
...
```

The first column, labeled Location, is the register(s) location of the setting. The next column, Setting, identifies the setting by name. Similar to Registers.csv, there is a Varies column that indicates whether the current setting has the same value for all projects included in the export

(No) or at least two projects differ in value (Yes). Successive columns contain the hexadecimal setting values for each project. Settings may be 1-bit to 64-bits wide. The named settings are converted to individual single byte register values by this tool, to greatly simplify and speed up programming.

If a setting is not written (not defined and a don't care) by one project but is defined by another, the value cell will be empty for the project(s) that do not define the setting. These empty cells will not count towards the Varies computation. For example, in the example above project 3 does not define a value for SLAB_OLA_FINE_ADJ_OVR above, so the value is empty. Project 1 and 2 have the same value. So Varies is No and this setting need not be written when switching from one configuration to another.

Note any control register writes necessary to fully load new frequency plan are ****NOT**** included in the Registers.csv and Settings.csv exports.

P*-Report.txt

A design report is also created for each project file, and is named Pn-Report.txt where n is the project number. For example, P1-Report.txt, P2-Report.txt, and P3-Report.txt.

P*-Registers-Script.txt

A full config register write script is created for each project file. This is the same output that CBProProjectRegistersExport when passed --format csv --include-load-writes. This script contains any required control register pre- and post-write sequence required to ready the device for programming and fully load the configuration when complete.

For example:

```
# Part: Si5345
# Project File: Si5345-Edited-Project.slabtimeproj
# Design ID: <none>
# Includes Pre/Post Download Control Register Writes: Yes
# Die Revision: B1
# Creator: CBProMultiProjectExport v2.10.3 [2016-09-13]
# Created On: 2016-09-13 09:43:18 GMT-05:00
Address,Data
0x0B24,0xC0
0x0B25,0x00
...
```

P*-Registers-Script.h

You can optionally create the same programming script as a C header file. Add --c-register-scripts to the command line to generate these.

P*-Registers-Script-Delta-Only.txt

Smaller, delta reconfigure register write script are also created for each project file. These scripts contain only the register updates required to switch to one of the configurations. USE OF THESE UPDATE SCRIPTS REQUIRES THAT ONE OF THE FULL CONFIGURATION WRITE SCRIPTS HAVE BEEN PREVIOUSLY WRITTEN. For example:

- Device reset
- Host writes P2-Registers-Script.txt (P2 config loaded)
- Host writes P3-Registers-Script-Delta-Only.txt (P3 config loaded)
- Host writes P1-Registers-Script-Delta-Only.txt (P1 config loaded)
- Host writes P2-Registers-Script-Delta-Only.txt (P2 config loaded)

The delta update scripts contain any required programming preamble and postamble.

IMPORTANT: if your device contains a configuration pre-programmed by Silicon Labs via an Orderable Part Number (OPN) such as Si5345B-B07024-GM and you have the project file that was used to create this OPN, you MUST STILL WRITE ONE OF THE FULL CONFIGURATION SCRIPTS THIS TOOL GENERATES BEFORE USING ONE OF THE UPDATE DELTA SCRIPTS. The version of CBPro you are using may generate different registers for the same project file, and therefore different delta would be computed.

P*-Settings.txt

Per-project named setting (bitfield) export files are also created. This is the same information included in Settings.csv, but broken out on a per-project basis. This is the same output that CBProProjectSettingsExport generates.

For example:

```
# Si538x/4x Settings Export
#
# Part: Si5345
# Project File: Si5345-Edited-Project.slabtimeproj
# Design ID: <none>
# Export Mode: AllNonSpecialUserRegisters
# Setting Names: Customer
# Die Revision: B1
# Creator: CBProMultiProjectExport v2.10.3 [2016-09-13]
# Created On: 2016-09-13 09:43:18 GMT-05:00
Location,SettingName,DecimalValue,HexValue
0x0004[7:0],GRADE,0,0x00
0x0006[23:0],TOOL_VERSION,0,0x000000
0x000B[6:0],I2C_ADDR,104,0x68
0x0016[1],LOL_ON_HOLD,1,0x1
0x0017[0],SYSINCAL_INTR_MSK,0,0x0
...
```

IN-SYSTEM PROGRAMMING WORKFLOW

There are many possible workflows given the different types of exports produced by this tool. Some of the schemes are described below.

Use Register Scripts

Your host software would have a copy of a single full configuration script for a base configuration. For example, P2-Registers-Script.txt. It would also have a copy of the update register script for each project the host may want to switch to, such as P1-Registers-Script-Delta-Only.txt through P3-Registers-Script-Delta-Only.txt. On device reset, the host will write the P2 full configuration script, P2-Registers-Script.txt. It can then write one of the smaller update delta scripts at any time to reconfigure the device to P1-P3.

For example:

- Device reset
- Host writes P2-Registers-Script.txt (P2 config loaded)
- Host writes P3-Registers-Script-Delta-Only.txt (P3 config loaded)
- Host writes P1-Registers-Script-Delta-Only.txt (P1 config loaded)
- Host writes P2-Registers-Script-Delta-Only.txt (P2 config loaded)

Both full and update scripts contain any required control register preamble needed to ready the device for programming and postamble required to complete programming and load new configuration.

NOTE: your host must write a full configuration script before using one of the update scripts: the update scripts assume that one of P1-Registers-Script.txt - P3-Registers-Script.txt was programmed. The host does not need to select P1 as the full config base config to program: any of the full register script files can be written on device reset.

Use Registers.csv Only

Your host software would have a copy of the Registers.csv data. On device boot and reset, the host would:

- Write any required pre-configuration control register sequence to the device
- Write all registers from one of the configuration columns
- Write any required post-configuration control register sequence to the device

To switch to a new configuration, the host only need write those registers that have Varies=Yes in the change column. The host would be responsible for determining whether any additional pre- and post-writes are required when switching to a new configuration.

OPTIONS

--c-register-scripts

Create a register script in the form of a C header file for the base project and each edit. Text/CSV style register write scripts for each project are always created.

--create-out-folder

Create the output folder does not exist already.

--help

Print detailed usage to the console and exit.

--out-folder folder

The output folder. If this is not specified, the folder where the tool is run is used.

--project pathname

The base configuration project file. You are modifying this design. The underlying file will never be changed. You must specify this.

--version

Print this program's version number and exit.

CBProMultiProjectExport

SYNOPSIS

```
CBProMultiProjectExport [ opts ] project-filename1 project-filename2
                        [ project-filename3 ... ]
```

```
CBProMultiProjectExport --help
```

DESCRIPTION

Creates various export files for multiple project files:

- Registers.csv, containing register values for all projects
- Settings.csv, containing setting values for all projects
- Design report for each project
- Full register config script for each project
- Delta reconfigure script for each project
- Settings dump for each project
- C code full config script for each project
(if --c-register-scripts is specified)
- A copy of each source project file
(if --copy-projects is specified)

See IN-SYSTEM PROGRAMMING WORKFLOW section for recommended ways for a host microprocessor to use these exports to switch between configurations in-system.

EXAMPLE

```
CBProMultiProjectExport --out-folder Output --create-out-folder
                        Si5345-RevD-CFG1-Project.slabtimeproj
                        Si5345-RevD-CFG2-Project.slabtimeproj
                        Si5345-RevD-CFG3-Project.slabtimeproj
```

Output files are placed in the folder named Output, creating it if necessary. Design reports (P*-Report.txt), full register write scripts (P*-Registers-Script.txt), delta reconfigure scripts (P*-Registers-Script-Delta-Only.txt), and setting/bitfield dumps (P*-Settings.txt) are created for each project.

Multi-project export files are created for P1-P3. Registers.csv contains register values for all projects and Settings.csv contains all setting/bitfield values.

This example is available in C:\Program Files (x86)\Silicon Laboratories\ClockBuilder Pro\CLI\Samples\Multi-Project-Export.

EXPORT FILES

```
Registers.csv
-----
```

Configuration register values for each project file. Example:

```
Address,Varies,P1,P2,P3
...
0x043D,Yes,0x12,0x12,0x0A
0x043E,No,0x06,0x06,0x06
0x0442,No,0x00,0x00,
0x0443,No,0x00,0x00,
0x0444,No,0x00,0x00,
...
```

The first column, labeled Address, is the two byte hex encoded register address. The second column, labeled Varies, indicates whether the current register has the same value for all projects included in the export (No) or at least two projects differ in value (Yes). Columns three onward contain the single byte register value for each project. Projects get labeled P1 onward using the order that projects were specified in the command line.

If a register is not written (not defined and a don't care) by one project but is defined by another, the value cell will be empty for the project(s) that do not define the register. These empty cells will not count towards the Varies computation. For example, in the 0x0442-0x0444 example above project 3 does not define the register so the value is empty. Project 1 and 2 have the same value. So Varies is No and this register need not be written when switching from one configuration to another.

Settings.csv

This breaks out registers by setting (aka bitfield). Example:

```
Location,Setting,Varies,P1,P2,P3
...
0x043D[4:0],HSW_COARSE_PM_LEN_PLLA,Yes,0x12,0x12,0xA
0x043E[4:0],HSW_COARSE_PM_DLY_PLLA,No,0x6,0x6,0x6
0x0442[17:0],SLAB_OLA_FINE_ADJ_OVR,No,0x0,0x0,
...
```

The first column, labeled Location, is the register(s) location of the setting. The next column, Setting, identifies the setting by name. Similar to Registers.csv, there is a Varies column that indicates whether the current setting has the same value for all projects included in the export (No) or at least two projects differ in value (Yes). Successive columns contain the hexadecimal setting values for each project. Settings may be 1-bit to 64-bits wide. The named settings are converted to individual single byte register values by this tool, to greatly simplify and speed up programming.

If a setting is not written (not defined and a don't care) by one project but is defined by another, the value cell will be empty for the project(s) that do not define the setting. These empty cells will not count towards the Varies computation. For example, in the example above project 3 does not define a value for SLAB_OLA_FINE_ADJ_OVR above, so the value is empty.

Project 1 and 2 have the same value. So Varies is No and this setting need not be written when switching from one configuration to another.

Note any control register writes necessary to fully load new frequency plan are ****NOT**** included in the Registers.csv and Settings.csv exports.

P*-Report.txt

A design report is also created for each project file, and is named Pn-Report.txt where n is the project number. For example, P1-Report.txt, P2-Report.txt, and P3-Report.txt.

P*-Registers-Script.txt

A full config register write script is created for each project file. This is the same output that CBProProjectRegistersExport when passed --format csv --include-load-writes. This script contains any required control register pre- and post-write sequence required to ready the device for programming and fully load the configuration when complete.

For example:

```
# Part: Si5345
# Project File: Si5345-Edited-Project.slabtimeproj
# Design ID: <none>
# Includes Pre/Post Download Control Register Writes: Yes
# Die Revision: B1
# Creator: CBProMultiProjectExport v2.10.3 [2016-09-13]
# Created On: 2016-09-13 09:43:18 GMT-05:00
Address,Data
0x0B24,0xC0
0x0B25,0x00
...
```

P*-Registers-Script.h

You can optionally create the same programming script as a C header file. Add --c-register-scripts to the command line to generate these.

P*-Registers-Script-Delta-Only.txt

Smaller, delta reconfigure register write script are also created for each project file. These scripts contain only the register updates required to switch to one of the configurations. USE OF THESE UPDATE SCRIPTS REQUIRES THAT ONE OF THE FULL CONFIGURATION WRITE SCRIPTS HAVE BEEN PREVIOUSLY WRITTEN. For example:

- Device reset
- Host writes P2-Registers-Script.txt (P2 config loaded)
- Host writes P3-Registers-Script-Delta-Only.txt (P3 config loaded)

- Host writes P1-Registers-Script-Delta-Only.txt (P1 config loaded)
- Host writes P2-Registers-Script-Delta-Only.txt (P2 config loaded)

The delta update scripts contain any required programming preamble and postamble.

IMPORTANT: if your device contains a configuration pre-programmed by Silicon Labs via an Orderable Part Number (OPN) such as Si5345B-B07024-GM and you have the project file that was used to create this OPN, you MUST STILL WRITE ONE OF THE FULL CONFIGURATION SCRIPTS THIS TOOL GENERATES BEFORE USING ONE OF THE UPDATE DELTA SCRIPTS. The version of CBPro you are using may generate different registers for the same project file, and therefore different delta would be computed.

P*-Settings.txt

Per-project named setting (bitfield) export files are also created. This is the same information included in Settings.csv, but broken out on a per-project basis. This is the same output that CBProProjectSettingsExport generates.

For example:

```
# Si538x/4x Settings Export
#
# Part: Si5345
# Project File: Si5345-Edited-Project.slabtimeproj
# Design ID: <none>
# Export Mode: AllNonSpecialUserRegisters
# Setting Names: Customer
# Die Revision: B1
# Creator: CBProMultiProjectExport v2.10.3 [2016-09-13]
# Created On: 2016-09-13 09:43:18 GMT-05:00
Location,SettingName,DecimalValue,HexValue
0x0004[7:0],GRADE,0,0x00
0x0006[23:0],TOOL_VERSION,0,0x000000
0x000B[6:0],I2C_ADDR,104,0x68
0x0016[1],LOL_ON_HOLD,1,0x1
0x0017[0],SYSINCAL_INTR_MSK,0,0x0
...
```

IN-SYSTEM PROGRAMMING WORKFLOW

There are many possible workflows given the different types of exports produced by this tool. Some of the schemes are described below.

Use Register Scripts

Your host software would have a copy of a single full configuration script for a base configuration. For example, P2-Registers-Script.txt. It would also have a copy of the update register script for each project the host may want to switch to, such as P1-Registers-Script-Delta-Only.txt through

P3-Registers-Script-Delta-Only.txt. On device reset, the host will write the P2 full configuration script, P2-Registers-Script.txt. It can then write one of the smaller update delta scripts at any time to reconfigure the device to P1-P3.

For example:

- Device reset
- Host writes P2-Registers-Script.txt (P2 config loaded)
- Host writes P3-Registers-Script-Delta-Only.txt (P3 config loaded)
- Host writes P1-Registers-Script-Delta-Only.txt (P1 config loaded)
- Host writes P2-Registers-Script-Delta-Only.txt (P2 config loaded)

Both full and update scripts contain any required control register preamble needed to ready the device for programming and postamble required to complete programming and load new configuration.

NOTE: your host must write a full configuration script before using one of the update scripts: the update scripts assume that one of P1-Registers-Script.txt - P3-Registers-Script.txt was programmed. The host does not need to select P1 as the full config base config to program: any of the full register script files can be written on device reset.

Use Registers.csv Only

Your host software would have a copy of the Registers.csv data. On device boot and reset, the host would:

- Write any required pre-configuration control register sequence to the device
- Write all registers from one of the configuration columns
- Write any required post-configuration control register sequence to the device

To switch to a new configuration, the host only need write those registers that have Varies=Yes in the change column. The host would be responsible for determining whether any additional pre- and post-writes are required when switching to a new configuration.

OPTIONS

--c-register-scripts

Create a register script in the form of a C header file for each project file. Text/CSV style register write scripts for each project are always created.

--copy-projects

Copy project file to output folder (renamed P1-Project.slabtimeproj, P2-Project.slabtimeproj, etc.)

--create-out-folder

Create the output folder does not exist already.

--design-id-labels

Use Design ID embedded in project file to define labels in Settings.csv and Registers.csv headers as well as filenames. For example, something like EVB53451 and EVB53452 or EVB53451-P1 and EVB53451-P2. Using this option enables CBPro 2.10 and earlier behavior. Without this option, the labels are simply P1, P2, etc. based on the order projects are specified on the command line.

--help

Print detailed usage to the console and exit.

--out-folder folder

The output folder. If this is not specified, the folder where the tool is run is used.

--version

Print this program's version number and exit.

CBProProjectEdit

SYNOPSIS

```
CBProProjectEdit [ --design-id ascii-text ]
                  --edit-file edits.txt
                  --in-project oldproj.slabtimeproj
                  --out-project newproj.slabtimeproj
```

```
CBProProjectEdit --help
```

DESCRIPTION

Allows you to make some basic edits to an Si538x/4x configuration, and save the resulting new design to a project file. Clock frequencies and state can be edited. On supported devices, DSPLL bandwidth and assignment can also be defined. Editing is done via a simple text configuration file you specify on the command line.

EDIT FILE

You must specify an edit file on the command line using the --edit-file option. This file will define what changes you want to make to one or more clock inputs and/outputs. The format of the file varies depending on whether the device in your project file has a single or multiple DSPLLs.

If a single DSPLL device:

Clock,State,Frequency

```
IN0,Enabled,19.44 MHz
IN1,Unused
OUT0,Enabled,25*(825/128)M
OUT1,Disabled,625M
OUT5,Unused
```

The header (Clock,State,Frequency) is required. Comments via lines starting with # are allowed, as are blank lines. Inputs can have one of two states: Enabled or Unused. Outputs have three possible states: Enabled, Disabled, or Unused. You ****do not**** have to list every input or output clock: any not listed will retain the setting in the project file.

If a multiple DSPLL device:

```
Clock,State,DSPLL,Frequency
IN0,Enabled,ABCD,19.44 MHz
IN1,Unused
OUT0,Enabled,A,25*(825/128)M
OUT1,Enabled,B,625M
OUT5,Unused
```

This adds a column for DSPLL assignment to the format. Note that in the case of an input, you list all DSPLLs that the input will feed.

On select devices, DSPLL bandwidth can also be changed by specifying OLBW (nominal), FLBW (fastlock), or HOBW (holdover exit) as a clock name.

Example for single DSPLL device:

```
OLBW,,100 Hz
FLBW,,1k
```

Example for multi-DSPLL device:

```
OLBW,,A,100
FLBW,,A,1k
OLBW,,B,200
FLBW,,B,2k
```

You can also specify bandwidth for multiple DSPLLs at once:

```
OLBW,,AB,100 Hz
FLBW,,AB,1 kHz
```

EXAMPLE

```
CBProProjectEdit --in-project Si5345-Original-Project.slabtimeproj
                  --edit-file Si5345-Edits.txt
                  --out-project Si5345-Edited-Project.slabtimeproj
```

Applies configuration edits in Si5345-Edits.txt to project file Si5345-Original-Project.slabtimeproj, creating Si5345-Edited-Project.slabtimeproj.

This example is in Edit-And-Export-For-Single-PLL-Device under the

C:\Program Files (x86)\Silicon Laboratories\ClockBuilder
Pro\CLI\Samples folder. An example for a multi-PLL device is in the
Edit-And-Export-For-Multi-PLL-Device folder.

OPTIONS

--design-id ascii-text

Some ASCII text to store in the DESIGN_ID registers. This is optional.
If you do not specify, then the value in the input project file will
be left as-is.

--edit-file pathname

The edits to make are defined in this file. See EDIT FILE for details.
You must specify this.

--help

Print detailed usage to the console and exit.

--in-project pathname

The input project. You are modifying this design. The underlying file
will not be changed unless you specify --out_project to be the same
file. You must specify this.

--out-project pathname

The project file to save the edited design to. This can be the same
file as --in_project, but normally you will be saving to another file.
You must specify this.

--version

Print this program's version number and exit.

CBProProjectRegistersExport

SYNOPSIS

```
CBProProjectRegistersExport [ opts ] --format csv|cheader
                             --project myproj.slabtimeproj
                             --outfile export.txt
```

```
CBProProjectRegistersExport --help
```

DESCRIPTION

Exports the sequence of registers that must be written to achieve the design/configuration present in the specified project file. Two formats are supported:

- CSV: each line in the file is an address,data pair in hexadecimal format. A comma separates the address and data fields. By default, a #-style header is included at the top of the file.
- "C" Code Header: the register write sequence is expressed in C code via an array of address,data pairs. This can be used directly in firmware code.

You specify which format to export using the --format options. There is no default: you must select a format.

Please refer to the Si538x/4x Family Reference Manual for information on register addressing and how to write the data contained within this export.

OPTIONS

```
--format csv|cheader
```

The export format. You must specify this.

```
--help
```

Print detailed usage to the console and exit.

```
--include-load-writes
```

Certain control registers must be written before and after writing the volatile configuration registers. This ensures the device is stable during configuration download and resumes normal operation after the download is complete. If you want to include these special pre- and post-write register writes in the export, specify this option.

```
--no-header
```

Normally, a summary header will be included in the export. Each line in the header will be prefixed by the # character. The header will contain some basic information about the design, tool, and a timestamp. Specify this argument to exclude the header. Not applicable

in CBProProjectRegistersExport "C" code mode.

--outfile pathname

The file to save the export data to. If this file already exists, it will be overwritten. You must specify this.

--project pathname

The project file to export. You must specify this.

--version

Print this program's version number and exit.

CBProProjectSettingsExport

SYNOPSIS

```
CBProProjectSettingsExport [ opts ] --project myproj.slabtimeproj
                             --outfile export.txt
```

```
CBProProjectSettingsExport --help
```

DESCRIPTION

Creates an export file containing the named settings that need to be written to the Si538x/4x device to achieve your design/configuration. Each line in the file is a `setting_name,data_uint,data_hex` triple. It is similar to the settings dump included in design reports and datasheet addendums.

Named settings may be anywhere from 1-bit to 64-bits wide. For mass programming in-system, the per-byte register export format should be used for simplicity and speed of programming.

Please refer to the Si538x/4x Family Reference Manual for information on register addressing and how to write the data contained within this export, and how to perform any additional device writes necessary to reload any configuration change, such as soft reset.

OPTIONS

--help

Print detailed usage to the console and exit.

--no-header

Normally, a summary header will be included in the export. Each line in the header will be prefixed by the # character. The header will contain some basic information about the design, tool, and a timestamp. Specify this argument to exclude the header. Not applicable in CBProProjectRegistersExport "C" code mode.

--outfile pathname

The file to save the export data to. If this file already exists, it will be overwritten. You must specify this.

--project pathname

The project file to export. You must specify this.

--version

Print this program's version number and exit.

CBProRegistersToSettings

SYNOPSIS

```
CBProRegistersToSettings --part name --revision rev
                        --infile pathname [ --outfile pathname ]
                        [ --format csv|text|html ]

CBProRegistersToSettings --list-parts

CBProRegistersToSettings --help
```

DESCRIPTION

Breaks out device register values by named setting (bitfield). For example, if the device is Si5345 Revision D and regs.txt contains:

```
0x0109,0x09
0x0302,0x00
0x0303,0x00
0x0304,0x00
0x0305,0x80
0x0306,0x05
0x0307,0x00
0x0308,0x00
0x0309,0x00
0x030A,0x00
0x030B,0x80
```

CBProRegistersToSettings --part si5345 --rev d --infile regs.txt will translate these registers to the settings:

Location	NVM	Flag	SettingName	DecValue	HexValue
-----	----	----	-----	-----	-----
0x0109[2:0]	User	R/W	OUT0_FORMAT	1	0x01
0x0109[3]	User	R/W	OUT0_SYNC_EN	1	0x01
0x0109[5:4]	User	R/W	OUT0_DIS_STATE	0	0x00
0x0109[7:6]	User	R/W	OUT0_CMOS_DRV	0	0x00
0x0302[43:0]	User	R/W	N0_NUM	23622320128	0x580000000
0x0308[31:0]	User	R/W	N0_DEN	2147483648	0x80000000

Text, CSV, and HTML output format is supported using the --format option.

If a setting's registers are only partially specified, 0x00 is assumed for missing addresses.

OPTIONS

```
--format csv|text|html
```

The output format. The default is a text table.

```
--help
```

Print detailed usage to the console and exit.

`--infile pathname`

The register file to parse. Required.

`--list-parts`

List part numbers supported by this tool.

`--order file|address|name`

The order to list the settings by. The default is file order, which lists the setting names in the order they are set in the original file. Address order sorts by device address low to high and within an register bitfields get sorted by bit position low to high. Name order uses simple alphanumeric sort of setting name.

`--outfile pathname`

The output file where the settings breakout will be saved. Optional. If not specified, output will be to the console (standard output).

`--part name`

The part the register file is associated with. For example, si5345. Argument is case insensitive. Required. Use `--list-parts` to see supported parts.

`--rev rev`

The part device revision the register file is associated with. For example, D. Argument is case insensitive. Required.

`--unlock password`

Provides Silicon Labs employees access to extended reporting.

`--version`

Print this program's version number and exit.

CBProRegistersToSettings

SYNOPSIS

```
CBProRegistersToSettings --part name --revision rev
                        --infile pathname [ --outfile pathname ]
                        [ --format csv|text|html ]

CBProRegistersToSettings --list-parts

CBProRegistersToSettings --help
```

DESCRIPTION

Breaks out device register values by named setting (bitfield). For example, if the device is Si5345 Revision D and regs.txt contains:

```
0x0109,0x09
0x0302,0x00
0x0303,0x00
0x0304,0x00
0x0305,0x80
0x0306,0x05
0x0307,0x00
0x0308,0x00
0x0309,0x00
0x030A,0x00
0x030B,0x80
```

CBProRegistersToSettings --part si5345 --rev d --infile regs.txt will translate these registers to the settings:

Location	NVM	Flag	SettingName	DecValue	HexValue
-----	----	----	-----	-----	-----
0x0109[2:0]	User	R/W	OUT0_FORMAT	1	0x01
0x0109[3]	User	R/W	OUT0_SYNC_EN	1	0x01
0x0109[5:4]	User	R/W	OUT0_DIS_STATE	0	0x00
0x0109[7:6]	User	R/W	OUT0_CMOS_DRV	0	0x00
0x0302[43:0]	User	R/W	N0_NUM	23622320128	0x580000000
0x0308[31:0]	User	R/W	N0_DEN	2147483648	0x80000000

Text, CSV, and HTML output format is supported using the --format option.

If a setting's registers are only partially specified, 0x00 is assumed for missing addresses.

OPTIONS

```
--format csv|text|html
```

The output format. The default is a text table.

```
--help
```

Print detailed usage to the console and exit.

`--infile pathname`

The register file to parse. Required.

`--list-parts`

List part numbers supported by this tool. Unreleased or customer specific parts are not listed.

`--order file|address|name`

The order to list the settings by. The default is file order, which lists the setting names in the order they are set in the original file. Address order sorts by device address low to high and within an register bitfields get sorted by bit position low to high. Name order uses simple alphanumeric sort of setting name.

`--outfile pathname`

The output file where the settings breakout will be saved. Optional. If not specified, output will be to the console (standard output).

`--part name`

The part the register file is associated with. For example, si5345. Argument is case insensitive. Required. Use `--list-parts` to see supported parts.

`--rev rev`

The part device revision the register file is associated with. For example, D. Argument is case insensitive. Required.

`--unlock password`

Provides Silicon Labs employees access to extended reporting.

`--version`

Print this program's version number and exit.

CBProRegmapExport

SYNOPSIS

```
CBProRegmapExport [ opts ] --project project-filename
```

```
CBProRegmapExport --help
```

DESCRIPTION

This tool exports meta-data about all of the customer facing settings and registers for a particular revision of a device. This is referred to as a register map aka regmap. Two types of exports are created: regmap tables and regmap C code files. Both are created in the folder you select.

Three table versions are saved: CSV data, text report, and HTML report. For each setting, the following are defined:

- Setting name (PN_BASE, N0_NUM, etc).
- Register start address, bit offset within start address, and length.
- NVM classification: not NVM backend (None), stored in user burnable NVM (User), or stored in protected factory programmed NVM (SiLab).
- Setting type: read-only (R/O), read-write (R/W), or self-clearing (S/C).

The C code is a header (.h) file containing a struct typedef and array of settings meta-data.

OPTIONS

```
--create-out-folder
```

Create the output folder does not exist already.

```
--help
```

Print detailed usage to the console and exit.

```
--out-folder folder
```

The output folder. If this is not specified, the folder where the tool is run is used.

```
--project pathname
```

Select the regmap to export based on this project file. You must specify this.

```
--version
```

Print this program's version number and exit.

CBProSi534x8xFirmwareDownload

SYNOPSIS

```
CBProSi534x8xFirmwareDownload [ options ] --bootrecord-file pathname
CBProSi534x8xFirmwareDownload [ options ] --standalone-file pathname
CBProSi534x8xFirmwareDownload --list-devices
```

```
CBProSi534x8xFirmwareDownload --help
```

DESCRIPTION

Writes a firmware file to a supported Silicon Labs device. Supports device being in program mode or already in bootloader mode. Firmware write is verified.

The following firmware formats are supported:

- + Boot Record: a firmware image packed into a boot record that is compatible with the device's bootloader. Use the `--bootrecord-file` option to specify this type of firmware image.
- + Stand-Alone: a firmware image normally used with Silicon Labs MCU development tools and debug adapter. CBPro will convert this type of file to the boot record file automatically. Use the `--standalone-file` option to specify this type of firmware image

Supports the following types of targets:

- + The device is on a Silicon Labs Evaluation Board (EVB)
- + The device is on a 3rd party board and connected to a Silicon Labs ClockBuilder Field Programmer (FP) via wired serial connection
- + The device is in a supported socket connected to the Silicon Labs ClockBuilder Field Programmer (FP)

If more than one EVB or FP is connected to your PC, you must use the `--device` option to specify which you are targeting. You pass an identifier for the device, which can be listed using the `--list-devices` option. For example, `CBProSi534x8xFirmwareDownload --list-devices` might return:

```
00-00-16-B1-20-A2 (Si5383 EVB)
00-00-15-E2-E2-60 (Si5380 EVB)
```

You can then pass either `00-00-16-B1-20-A2` or `00-00-15-E2-E2-60` to the `--device` argument. For example:

```
CBProSi534x8xFirmwareDownload --device 00-00-15-E2-E2-60
                                --bootrecord-file si5383.bin
```

Status messages are displayed to the console as the download progresses. The program will exit with status code 0 on success, and 1 on error.

FIELD PROGRAMMER

CBPro CLI User's Guide

You must configure communication settings when using a ClockBuilder Pro Field Programmer. The following options are available:

- `--i2c-address`
I2C address. Required.
- `--speed 100k|400k`
Bus speed. Optional. 400 kHz is selected by default.

EXAMPLES

```
CBProSi534x8xFirmwareDownload.exe --i2c-address 0x6c --bootrecord-file  
sibrecord.bin
```

Downloads a bootrecord-based firmware image using the ClockBuilder Pro Field Programmer. Sets I2C bus speed to 400 kHz (the default). The device must be configured for address 0x6C. The device can be in program mode or already in bootloader mode.

```
CBProSi534x8xFirmwareDownload.exe --standalone-file sifw.hex
```

Downloads a stand-alone firmware image to a Silicon Labs Evaluation Board. The file swfw.hex is converted internally to bootrecord format. The device on the EVB can be in program mode or already in bootloader mode.

OPTIONS

`--bootrecord-file pathname`

A bootrecord-based firmware file to program on the device. Can be a .hex or .bin file.

`--device id`

Select the Evaluation Board or Field Programmer to target. Required if there are 2 or more EVBs/FPs attached to the PC.

`--help`

Print detailed usage to the console and exit.

`--i2c-address addr`

7-bit I2C address when using the ClockBuilder Pro field programmer. There is no default. You can specify the address in 0xNN hex format or NN decimal.

`--list-devices`

List Silicon Labs Timing devices -- Evaluation Boards and Field Programmers -- connected to the PC and exit.

`--speed 100k|400k`

I2C bus speed when using the ClockBuilder Pro field programmer. The default is 400 kHz.

--standalone-file pathname

A standard firmware image that is normally downloaded using a Silicon Labs USB Debug Adapter. This tool will convert it to a the boot record format and flash using the device's bootloader. Can be a .hex or .bin file.

--version

Print this program's version number and exit.

CBProSi534x8xFirmwareExport

SYNOPSIS

```
CBProSi534x8xFirmwareExport [ --config-only ]  
                             --type bootrecord|standalone --format bin|hex  
                             --project pathname --outfile pathname  
  
CBProSi534x8xFirmwareExport --help
```

DESCRIPTION

This tool creates a firmware image for the Si5383/84/88/89. The device configuration specified in this design will be embedded in the firmware. Please refer to the Si5383/84 or Si5388/89 Family Reference Manual for more information regarding the MCU firmware upgrade process.

OPTIONS

--config-only

Do not include program flash in the firmware image. Only include the device configuration data. Use this option with extreme caution: the device you are programming must have a compatible version of firmware loaded.

--format bin|hex

The file format: binary or Intel Hex.

--help

Print detailed usage to the console and exit.

--outfile pathname

The file to save the firmware to. If this file already exists, it will be overwritten. You must specify this.

--project pathname

The CBPro project file. The configuration present in this design will be embedded in the firmware.

--type bootrecord|standalone

The type of firmware image to create: boot record that can be used with bootloader or stand-alone firmware image.

--version

Print this program's version number and exit.