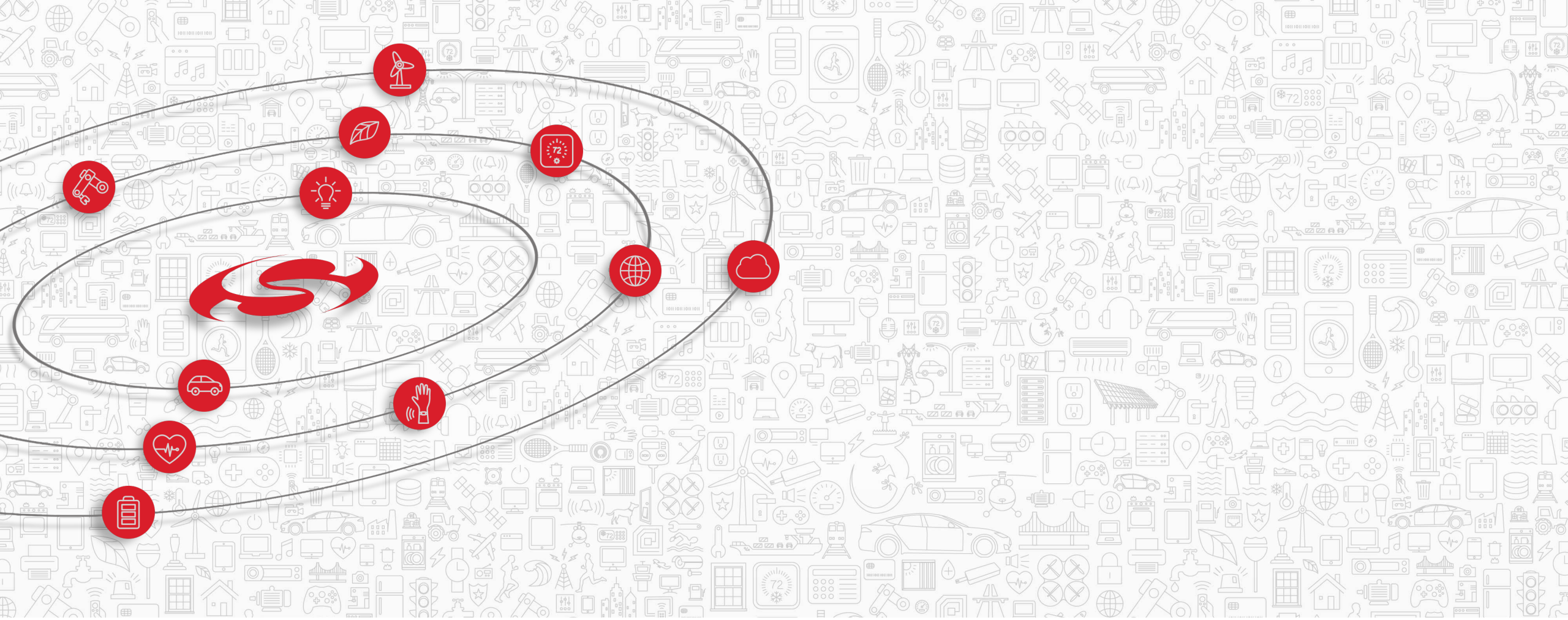


CBPro Tools & Support for In-System Programming

January 14, 2020

For ClockBuilder Pro v2.39.3 and Higher

SILICON LABS

Table of Contents

In-System Programming Scenarios & Tool Support

- **Download full configuration**
  - A ClockBuilder Pro configuration will be written to the device
  - It is not necessary to know the current configuration state
  - Clock generation/etc. will be interrupted during programming
  - Tool support for all CBPro supported I2C/SPI configurable devices
- **Update configuration**
  - Only a subset of configuration registers are written to the device
  - Settings to be written could be determined ahead of time or dynamically in-system (such as dividers)
  - The type of change will dictate whether clock generation is interrupted during programming
  - Must ensure control register writes required to switch to configuration are performed
  - DIY – "Do It Yourself"
- **Frequency-on-the-fly**
  - Special case of the configuration update
  - Change select output frequencies only, leaving all other outputs undisturbed
  - Tool support for Si5340/41/42/42H/44/44H/45/46/47/80/91/92/94/95/96/97
- **Digitally Controlled Oscillator (DCO) mode frequency adjustments**
  - Make small, glitchless frequency changes by adjusting the value of one of the dividers in the device responsible for synthesizing one or more output frequencies
  - Limited range in frequencies can switch to
  - Tool support for all DCO capable devices

Device Family Support

- **Most devices with an I2C or SPI interface support full configuration reprogramming**
  - Si5332
  - Si5338/56
  - Si5351
  - Si534x
  - Si537x
  - Si538x
  - Si539x
- **There are a number of CBPro tools and workflows to support:**
  - In-system via host software
  - In-system via CBPro field programmer
  - EVB
- **Plus various support tools, such as to read/write arbitrary registers on device**

Programming Tools Overview

Edit Clocks in Project

- CBPro Wizard
- CBProProjectEdit CLI

Project + Clock Edits → New Project

Edit Clocks & Create DUT Programming Files

- CBProMultiEditAndExport CLI
- CBProFOTF1 CLI

Base Project + Clock Plans

Most tools are generic except and available on any CBPro supported device with an I2C or SPI interface

Create DUT Programming Files

- Export GUI
- CBProProjectRegistersExport CLI
- CBProProjectSettingsExport CLI
- CBProMultiProjectExport CLI

Design Reports

Setting Files → Write Programming File (Volatile)

Register Scripts → Write Programming File (Volatile)

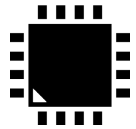
Write Programming File (Volatile)

- EVB GUI
- CBProDeviceWrite CLI
- Embedded System

Project → Burn NVM, Flash Device (Non-Volatile)

Burn NVM, Flash Device (Non-Volatile)

- NVM Program Tool
- CBProSi534x8x FirmwareDownload CLI
- Embedded System

→  → Read/Debug Registers

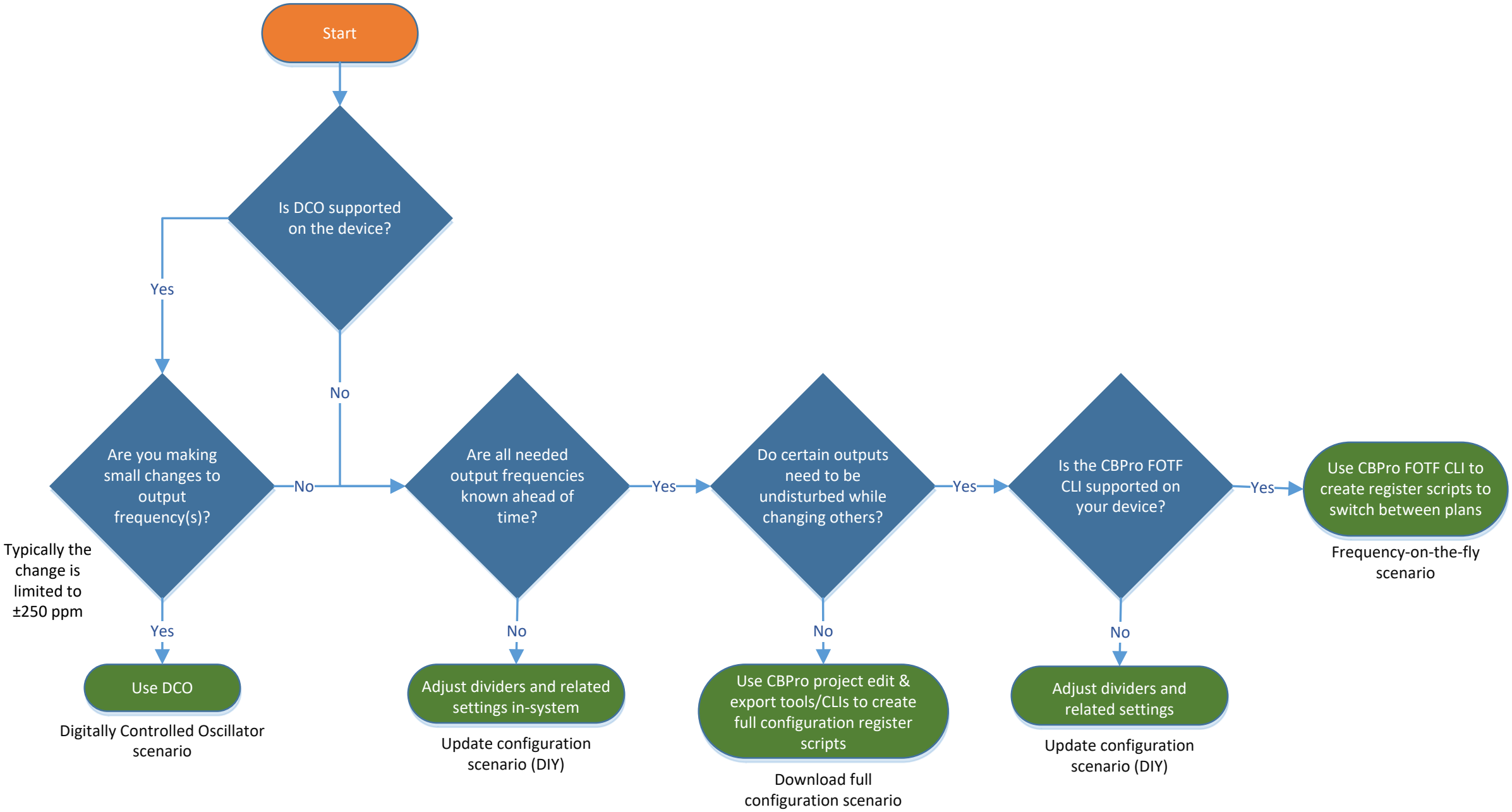
Read/Debug Registers

- EVB GUI
- CBProDeviceRead CLI
- CBProRegistersToSettings CLI
- Embedded System

Programming Tools Overview

Task	Applicable Programming Scenario	GUI Tool	CLI Tool	Support	
				Si534x/ 8x/9x	Other I2C/SPI Capable Devices
Change input frequency, output frequency, and/or bandwidth of a project file & save to a new project file	Full, Update, FOTF	CBPro wizard	CBProProjectEdit	✓	✓
Create config register write script	Full	CBPro Export tool	CBProProjectRegistersExport	✓	✓
Create config settings dump	Full	CBPro Export tool	CBProProjectSettingsExport	✓	✓
Create settings -> register map	Full, Update	CBPro Export tool	CBProRegmapExport	✓	✓
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.	Full, Update	CBPro Export tool	CBProMultiProjectExport	✓	✓
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	Full, Update	None	CBProMultiEditAndExport	✓	✓
Create register write scripts to switch a subset of outputs to a new frequency, leaving other outputs undisturbed	FOTF	None	CBProFOTF1	✓	
Read all or specific settings/registers from a Silicon Labs EVB or your system board via ClockBuilder Pro field programmer	N/A	EVB GUI (partial)	CBProDeviceRead	✓	✓
Write to volatile configuration registers on Silicon Labs EVB or your system board via ClockBuilder Pro field programmer	N/A	CBPro wizard, EVB GUI	CBProDeviceWrite	✓	✓
Create a firmware image, containing both device configuration defaults and latest MCU software	N/A	CBPro Export tool	CBProSi534x8xFirmwareExport	✓	
Download a firmware image to a Silicon Labs EVB or your system board via ClockBuilder Pro field programmer	N/A	None	CBProSi534x8xFirmwareDownload	✓	
Break out device register values by named setting (bitfield)	Debug	None	CBProRegistersToSettings	✓	✓

Si534x/8x/9x In-System Programming Selector



Si534x/8x/9x Programming Control Registers

- **VCO frequency change greater than ±250 ppm can cause loss of lock**
  - Via changes to XAXB_FREQ_OFFSET, PXAXB, M_NUM, M_DEN, MXAXB_NUM, MXAXB_DEN
  - This is often the case in the full configuration reprogramming scenario
  - A special pre- and post-config control register write sequence is required in these cases
- **On multi-PLL devices such as Si5347/97, output frequency changes greater than ±350 ppm may also cause the PLL to become responsive**
  - Via changes to Mx_NUM, Mx_DEN
  - Via DCO
- **Many frequency plan related register changes require special UPDATE control registers be written for change to take effect**
  - Px divider
  - MXAXB divider
  - M divider
  - M_PLLx dividers
  - Nx dividers
  - BWx coefficients
- **CBPro tools handle above for you in full config and FOTF scenarios**
  - Full config: pre- and post-write control sequences are included in config scripts
  - FOTF: divider, etc. update control register writes are included in plan scripts

Programming Scenario Walkthroughs

- **Short walkthroughs for the following tool supported programming scenarios:**
  - Download full configuration (generic)
  - Frequency-on-the-fly (Si534x/8x/9x)
  - Digitally Controlled Oscillator (DCO) mode frequency adjustments (Si534x/8x/9x)
- **"DIY" configuration update scenario**
  - Consult device Family Reference Manual or contact Silicon Labs

Full Configuration Download

- **Clock generation is interrupted briefly during download**
- **Device-specific download workflow usually required**
  - Ready device for programming (write pre-amble)
  - Write configuration registers
  - Enable normal device operation (write post-amble)
- **CBPro register write scripts include the pre-amble and post-amble in form of control register writes customized to the device/configuration**
- **Three full configuration walkthroughs follow:**
  - Single project file
  - Multiple project files (streamline export)
  - Clock variations of a single project file (streamline creation of variations & export)

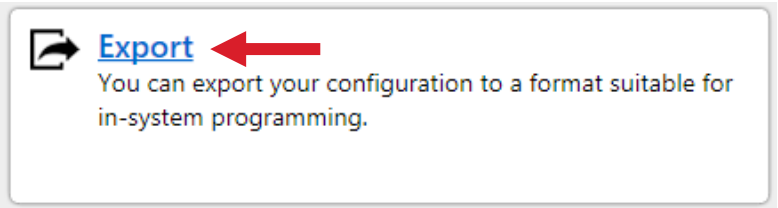
Full Configuration Download – Single Project File

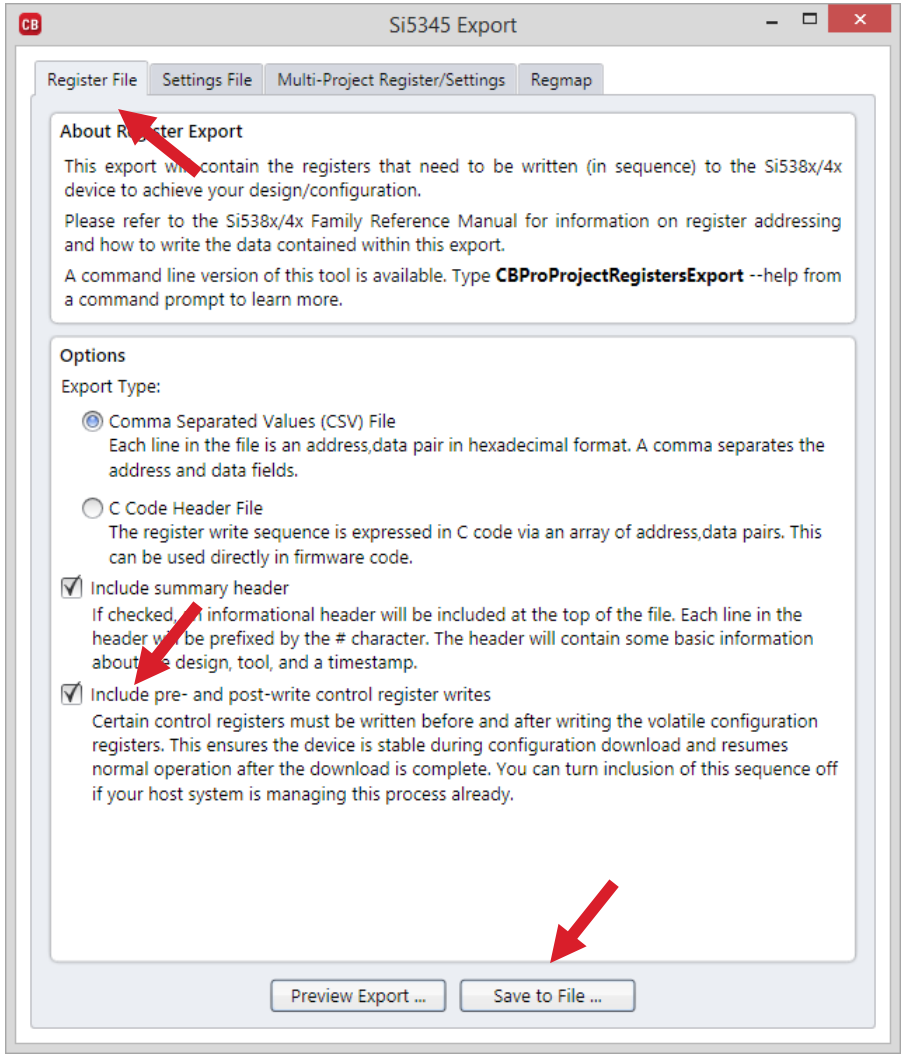
- Create any number of CBPro project files using the wizard
- Create register write scripts for each project
- Script will include any pre- and post-programming control register sequences if appropriate option checked
- Example script:

```
Address,Data
0x0B24,0xC0
0x0B25,0x00
0x0004,0x00
0x0006,0x00
…
```

- Data as C header file can also be exported
- CLI available:

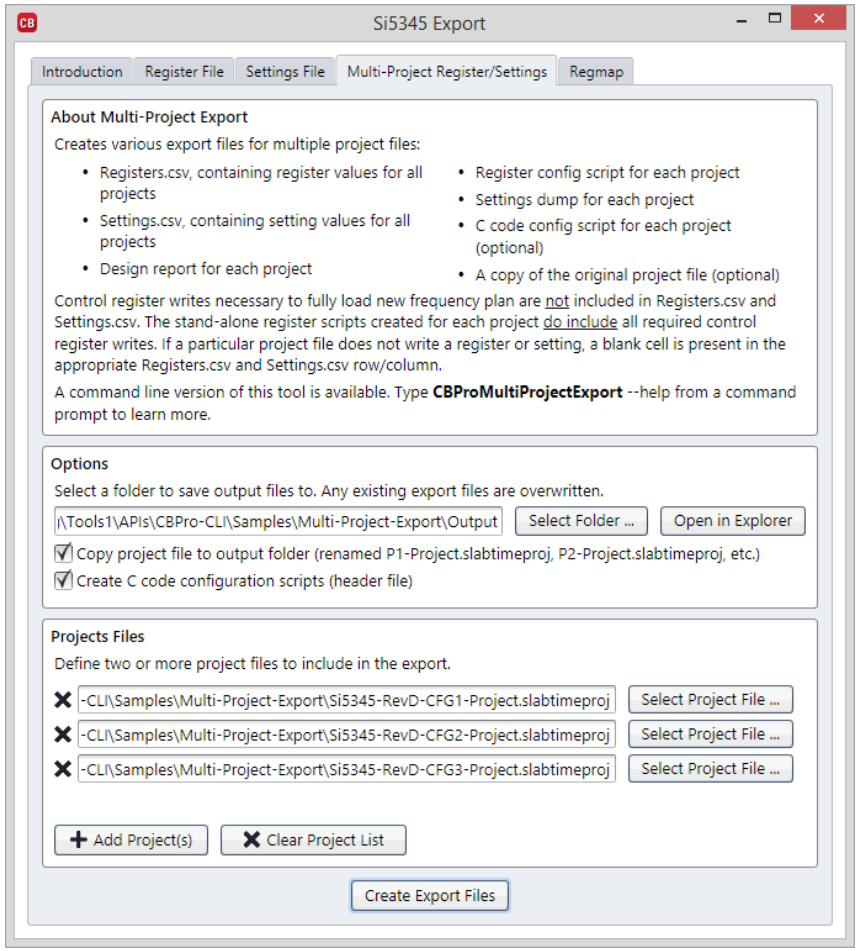
```
CBProProjectRegistersExport.exe
    --project Si5345-Project.slabtimeproj
    --include-load-writes --format csv
    --outfile Si5345-Config-Script.txt
```





Full Configuration Download – Multiple Project Files

- Simplify export creation by using multi-project export
- GUI:



- CLI:

```
CBProMultiProjectExport --out-folder Output --create-out-folder
   --c-register-scripts --copy-projects
   Si5345-RevD-Project1.slabtimeproj Si5345-RevD-Project2.slabtimeproj
   Si5345-RevD-Project3.slabtimeproj
```

Full Configuration Download – Multiple Project Files

- **Files created:**

 P1-Report.txt P2-Report.txt P3-Report.txt	Design reports for each project. Numbering is same as command line project ordering.
 P1-Registers-Script.txt P2-Registers-Script.txt P3-Registers-Script.txt	Full configuration scripts for each project. These contain any required control register pre- and post-download sequence required to ensure the device is ready for programming and fully loads configuration after programming has been completed.
 P1-Registers-Script.h P2-Registers-Script.h P3-Registers-Script.h	Configuration scripts for each project, but in C code format. This is optional.
 P1-Registers-Script-Delta-Only.txt P2-Registers-Script-Delta-Only.txt P3-Registers-Script-Delta-Only.txt	Update reconfigure scripts for each project. These smaller scripts can be used to switch to one of the P1-P3 configurations if one of the P*-Registers-Script.txt full config scripts was previously written. Contains any required control register pre- and post-download sequence.

Full Configuration Download – Multiple Project Files

- **Files created:**

P1-Settings.txt P2-Settings.txt P3-Settings.txt	Setting (register bitfield) dumps for each project.
P1-Project.slabtimeproj P2-Project.slabtimeproj P3-Project.slabtimeproj	A copy of the source project file. This is optional.

Full Configuration Download – Multiple Project Files

Settings.csv

Settings for each project in single file, with “Varies” column to indicate when a setting differs between any one project.

Location	Setting	Varies	P1	P2	P3
0x002E[15:0]	LOS0_TRG_THR	Yes	0x38	0x38	0x37
0x0030[15:0]	LOS1_TRG_THR	Yes	0x38	0x38	0x37
0x0032[15:0]	LOS2_TRG_THR	No	0x0	0x0	0x0
0x0034[15:0]	LOS3_TRG_THR	No	0x0	0x0	0x0
0x0036[15:0]	LOS0_CLR_THR	Yes	0x38	0x38	0x37
0x0038[15:0]	LOS1_CLR_THR	Yes	0x38	0x38	0x37
0x003A[15:0]	LOS2_CLR_THR	No	0x0	0x0	0x0
…	…	…	…	…	…

Registers.csv

Registers for each project in single file, with “Varies” column to indicate when a register differs between any one project. Does not include any required pre-download and post-download control register writes.

Address	Varies	P1	P2	P3
0x002E	Yes	0x38	0x38	0x37
0x002F	No	0x00	0x00	0x00
0x0030	Yes	0x38	0x38	0x37
0x0031	No	0x00	0x00	0x00
0x0032	No	0x00	0x00	0x00
0x0033	No	0x00	0x00	0x00
0x0034	No	0x00	0x00	0x00
0x0035	No	0x00	0x00	0x00
0x0036	Yes	0x38	0x38	0x37
0x0037	No	0x00	0x00	0x00
…	…	…	…	…

Full Configuration Download – Multiple Project Files

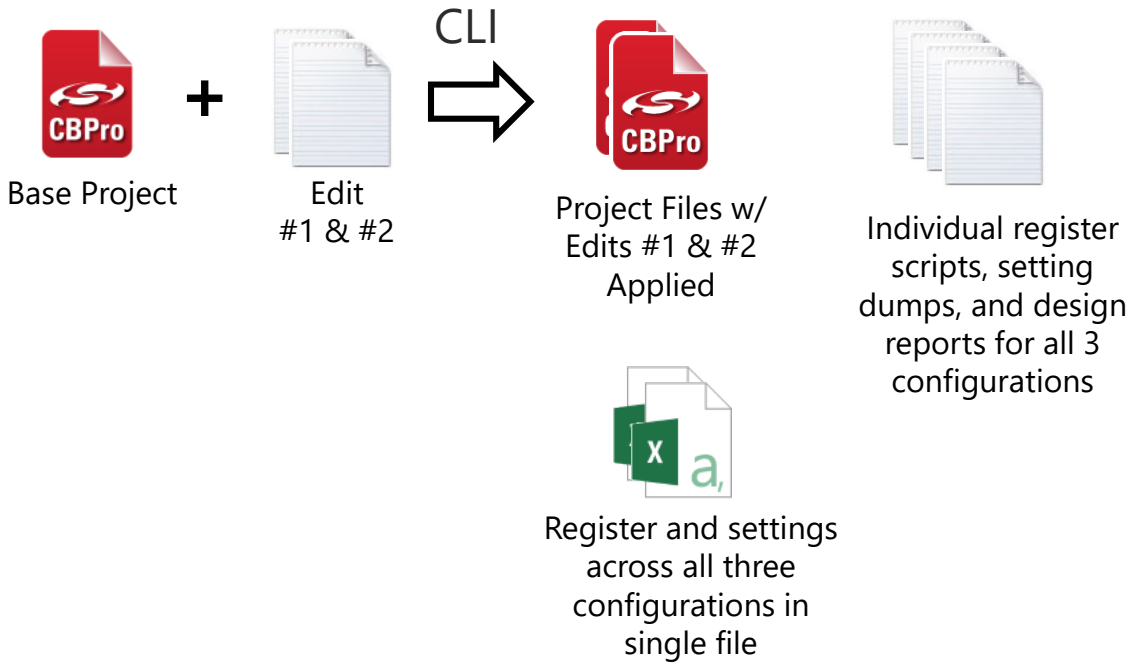
- **Host Workflow**
  - Device power up or reset
  - Write a full configuration write script (P*-Registers-Script.txt)
  - Write any delta update configuration write script (P*-Registers-Script-Delta-Only.txt)
- **Example Workflow**
  - Host writes P2-Registers-Script.txt (P2 config loaded)
  - Host 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)
- **IMPORTANT NOTE ABOUT OPNs**
  - 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.
  - Why? The version of CBPro you are using may generate different registers for the same project file, and therefore different delta would be computed.

Full Configuration Download – Clock Variations

- If configuration changes are limited to input frequency, output frequency, or bandwidth, CBProMultiEditAndExport CLI can be used to create alternate configurations
- Si5345 Example:

Output	Base Project	Edit #1	Edit #2
IN0	19.44 MHz	125 MHz	10 MHz
IN1	19.44 MHz	125 MHz	10 MHz
IN3/IN3	Unused	Unused	Unused
OUT0	156.25 MHz	25 MHz	156.25 MHz
OUT1	156.25 MHz	25 MHz	156.25 MHz
OUT2	156.25 MHz	625 MHz	156.25 MHz
OUT3	156.25 MHz	625 MHz	156.25 MHz
OUT4	156.25 MHz	625 MHz	156.25 MHz
OUT5	Unused	Unused	Unused
OUT6	622.08 MHz	100 MHz	125 MHz
OUT7	622.08 MHz	100 MHz	125 MHz
OUT8	622.08 MHz	168.041015625 MHz	125 MHz
OUT9	622.08 MHz	168.041015625 MHz	125 MHz
Nominal BW	40 Hz	400 Hz	40 Hz
Fastlock BW	1 kHz	2 kHz	1 kHz
Holdover BW	N/A	N/A	N/A

Want 3 configurations which only vary by the items shown (frequencies & bandwidth)



Full Configuration Download – Clock Variations

- **Project file defines base configuration:**

Si5345-Base-Project.slabtimeproj

	Mode	Input Buffer	Gapped Clock	Frequency
IN0	Enabled	Standard	Gapped	19.44 MHz
IN1	Enabled	Standard	Gapped	19.44 MHz

Output	Mode	Frequency
OUT0	Enabled	156.25 MHz
OUT1	Enabled	156.25 MHz
OUT2	Enabled	156.25 MHz
OUT3	Enabled	156.25 MHz
OUT4	Enabled	156.25 MHz
OUT5	Unused	N/A
OUT6	Enabled	622.08 MHz
OUT7	Enabled	622.08 MHz
OUT8	Enabled	622.08 MHz
OUT9	Enabled	622.08 MHz

Bandwidth

Target Loop Bandwidth: 40 Hz (Estimated Actual: 43.27 Hz)

Fastlock

Fastlock Enable: On

Target Fastlock Loop Bandwidth: 1 kHz (Estimated Actual: 692.99 Hz

Full Configuration Download – Clock Variations

- **Text files define the configuration edits**
- **All changes are to base project**
- **A clock can also be set to "Disabled" when supported**
- **Any frequency expression supported by CBPro can be used**
- **Si534x/8x/9x also supports bandwidth changes**
  - OLBW – nominal/open loop
  - FLBW - fastlock
  - HOBW – exit from holdover (when applicable)
- **All clocks and bandwidth do not need to be specified: any missing will inherit configuration from base project**
  - E.g. bandwidth is not specified in edit #2, so base project OLBW=40Hz, FLBW=1kHz used

Si5345-Edits1.txt

```
Clock,State,Frequency

IN0,Enabled,125MHz
IN1,Enabled,125MHz

OUT0,Enabled,25 MHz
OUT1,Enabled,25 MHz
OUT2,Enabled,625M
OUT3,Enabled,625M
OUT4,Enabled,625M
OUT5,Unused,
OUT6,Enabled,100 MHz
OUT7,Enabled,100 MHz
OUT8,Enabled,39813.2*255/236/256M
OUT9,Enabled,39813.2*255/236/256M

OLBW,,400
FLBW,,2k
```

Si5345-Edits2.txt

```
Clock,State,Frequency

IN0,Enabled,10MHz
IN1,Enabled,10MHz

OUT0,Enabled,156.25 MHz
OUT1,Enabled,156.25 MHz
OUT2,Enabled,156.25 MHz
OUT3,Enabled,156.25 MHz
OUT4,Enabled,156.25 MHz
OUT5,Unused,
OUT6,Enabled,125 MHz
OUT7,Enabled,125 MHz
OUT8,Enabled,125 MHz
OUT9,Enabled,125 MHz
```

Full Configuration Download – Clock Variations

- Create a DOS batch script to run the CLI

run.bat

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

Full Configuration Download – Clock Variations

- **Files created in Output folder:**

P1-Project.slabtimeproj P2-Project.slabtimeproj P3-Project.slabtimeproj	Base project file is saved to P1 (with frequency plan calculated using latest CBPro algorithms). P2 has project edits #1. P3 has project edits #2.
P1-Report.txt P2-Report.txt P3-Report.txt	Design reports for P1-P3 configurations.
P1-Registers-Script.txt P2-Registers-Script.txt P3-Registers-Script.txt	Configuration scripts for P1-P3. These contain any required control register pre- and post-download sequence required to ensure the device is ready for programming and fully loads configuration after programming has been completed.
P1-Registers-Script.h P2-Registers-Script.h P3-Registers-Script.h	Configuration scripts for each project, but in C code format. This is optional.
P1-Registers-Script-Delta-Only.txt P2-Registers-Script-Delta-Only.txt P3-Registers-Script-Delta-Only.txt	Update reconfigure scripts for each project. These smaller scripts can be used to switch to one of the P1-P3 configurations if one of the P*-Registers-Script.txt full config scripts was previously written. Contains any required control register pre- and post-download sequence.

Full Configuration Download – Clock Variations

- **Files created in Output folder:**

 P1-Settings.txt P2-Settings.txt P3-Settings.txt	Setting (register bitfield) dumps for P1-P3.

Full Configuration Download – Clock Variations

Settings.csv

Settings for P1-P3 in single file, with “Varies” column to indicate when a setting differs between any one project.

Location	Setting	Varies	P1	P2	P3
0x002E[15:0]	LOS0_TRG_THR	Yes	0x38	0x38	0x37
0x0030[15:0]	LOS1_TRG_THR	Yes	0x38	0x38	0x37
0x0032[15:0]	LOS2_TRG_THR	No	0x0	0x0	0x0
0x0034[15:0]	LOS3_TRG_THR	No	0x0	0x0	0x0
0x0036[15:0]	LOS0_CLR_THR	Yes	0x38	0x38	0x37
0x0038[15:0]	LOS1_CLR_THR	Yes	0x38	0x38	0x37
0x003A[15:0]	LOS2_CLR_THR	No	0x0	0x0	0x0
…	…	…	…	…	…

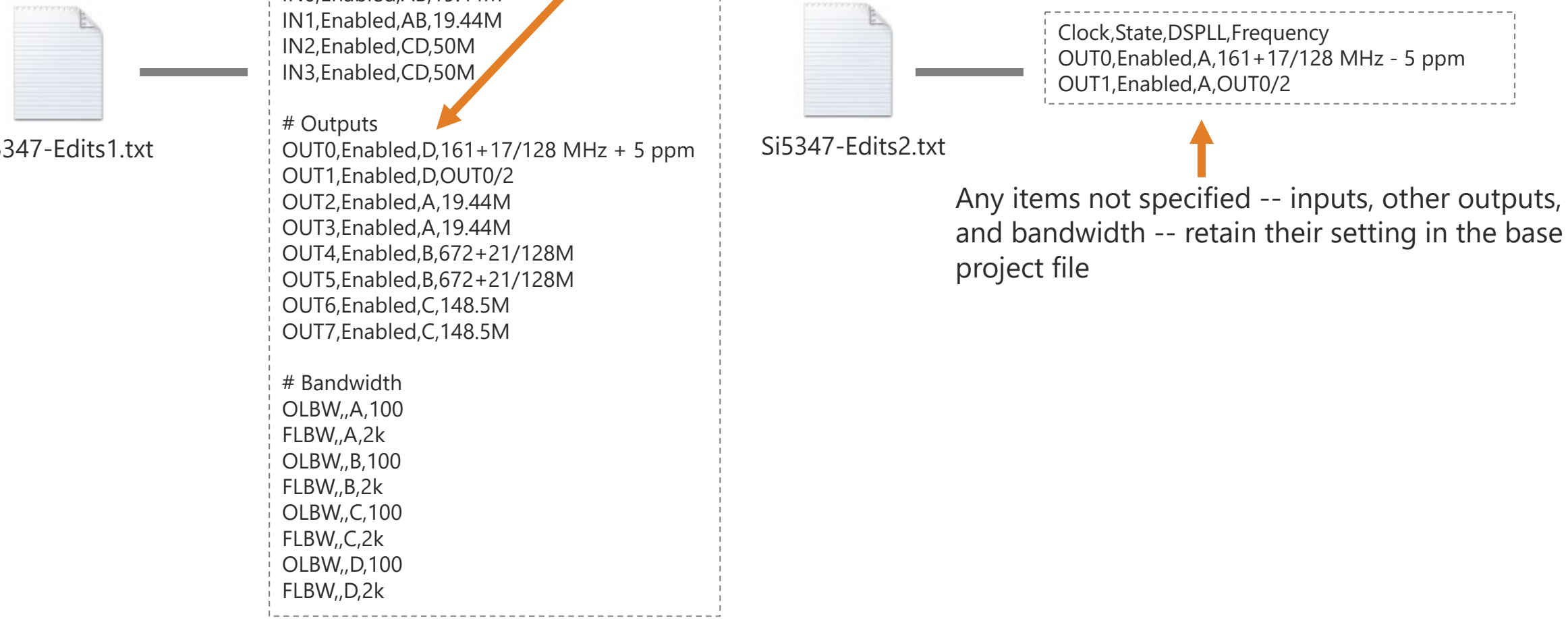
Registers.csv

Registers for P1-P3 in single file, with “Varies” column to indicate when a register differs between any one project. Does not include any required pre-download and post-download control register writes.

Address	Varies	P1	P2	P3
0x002E	Yes	0x38	0x38	0x37
0x002F	No	0x00	0x00	0x00
0x0030	Yes	0x38	0x38	0x37
0x0031	No	0x00	0x00	0x00
0x0032	No	0x00	0x00	0x00
0x0033	No	0x00	0x00	0x00
0x0034	No	0x00	0x00	0x00
0x0035	No	0x00	0x00	0x00
0x0036	Yes	0x38	0x38	0x37
0x0037	No	0x00	0x00	0x00
…	…	…	…	…

Full Configuration Download – Clock Variations

- PLL assignment required on multi-PLL devices
- Edit file can change MUX from base project
- Example:



Full Configuration Download – Clock Variations

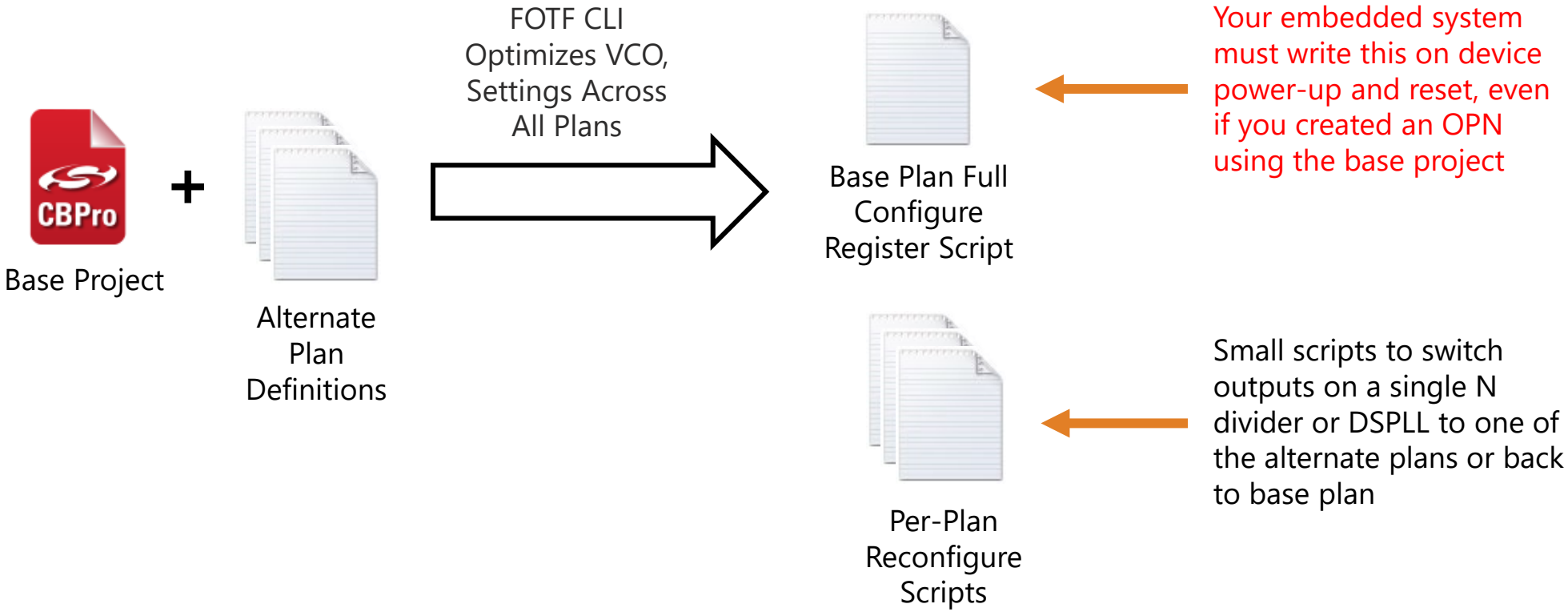
- Host workflow: see multi-project export

Si534x/8x/9x Frequency-On-The-Fly

- **Change a subset of output frequencies, leaving others undisturbed**
- **On Si5346/47/96/97 can also change input frequencies, PLL bandwidth, LOL thresholds, and OOF thresholds**
- **CBProFOTF1 CLI (no GUI)**
  - Note: was previously named CBProSi534x8xFOTF; this CLI name will continue to work and automatically use newer version
- **Supported on Si5340/41/42/42H/44/44H/45/46/47/48/71/72/80/83/84/88/89/91/92/94/95/96/97**
- **For single-PLL devices**
  - You must manually assign outputs to an N divider
  - You then define frequency plan(s) for each N divider: output frequencies
  - CLI generates register write script(s) that can be played by host MCU to switch a single N divider from one plan to another
  - Because all N dividers share same PLL, input frequencies and bandwidth cannot be adjusted
- **For multi-PLL devices**
  - Define frequency plan(s) for each PLL: change any of:
    - Clock output frequency (always)
    - Clock input frequency (select scenarios)
    - DSPLL bandwidth (always)
    - LOL thresholds (always)
    - OOF thresholds (select scenarios)
  - CLI generates register write script(s) that can be played by host MCU to switch a single PLL from one plan to another

Si534x/8x/9x Frequency-On-The-Fly

- The FOTF tool is optimizing VCO frequency and divider values for ALL required plan combinations, not only the base design/project
- The frequency plan and register settings for your "base" design will therefore usually be different when using the FOTF tool
- Also, certain device settings will be adjusted to support FOTF and will be reflected in the base configuration script



Si534x/8x/9x Frequency-On-The-Fly Example Walkthroughs

- **Si5345: 10-Output Single PLL Low Phase Noise Jitter Attenuating Clock Generator**
  - Also applicable to Si5340/41/42/42H/44/44H/71/72/80/91/92/94/95
- **Si5347: 8-Output Quad PLL Any-Frequency Jitter Attenuating Clock Multiplier**
  - Also applicable to Si5346/48/83/84/88/89/96/97

Si5345 Frequency-On-The-Fly Example

N0 Divider:

Output	Base Project	Plan #1
OUT0	161.1328125 MHz	155.52 MHz
OUT1	322.265625 MHz	311.04 MHz

N1 Divider:

Output	Base Project	Plan #1
OUT2	155.52 MHz	161.1328125 MHz
OUT3	311.04 MHz	322.265625 MHz

N2 Divider:

Output	Base Project	Plan #1	Plan #2	Plan #3	Plan #4	Plan #5
OUT4	148.5 MHz	155.52*255/237M	156.25M	132.8125M	159.375M	156.25*66/64*255/237M
OUT5	148.5 MHz	155.52*255/237M	156.25M	132.8125M	159.375M	156.25*66/64*255/237M
OUT6	148.5 MHz	155.52*255/237M	156.25M	132.8125M	159.375M	156.25*66/64*255/237M
OUT7	148.5 MHz	155.52*255/237M	156.25M	132.8125M	159.375M	156.25*66/64*255/237M
OUT8	148.5 MHz	155.52*255/237M	156.25M	132.8125M	159.375M	156.25*66/64*255/237M
OUT9	148.5 MHz	155.52*255/237M	156.25M	132.8125M	159.375M	156.25*66/64*255/237M

Base = Startup

- The base project essentially is Plan #0: it is called the "Base" plan in documentation and the files created
- Note how there does not need to be an equal number of plans between N dividers
- 10 register write scripts are generated to load a plan:
  - 2 for N0 (base, plan #1)
  - 2 for N1 (base, plan #1
  - 6 for N2 (base, plan #1-5)
- More details in next slide

This example is bundled in CBPro at C:\Program Files (x86)\Silicon Laboratories\ClockBuilder Pro\CLI\Samples\Single-PLL-FOTF

User manual goes into more detail (CBProFOTF1 --help)

Si5345 Frequency-On-The-Fly Example

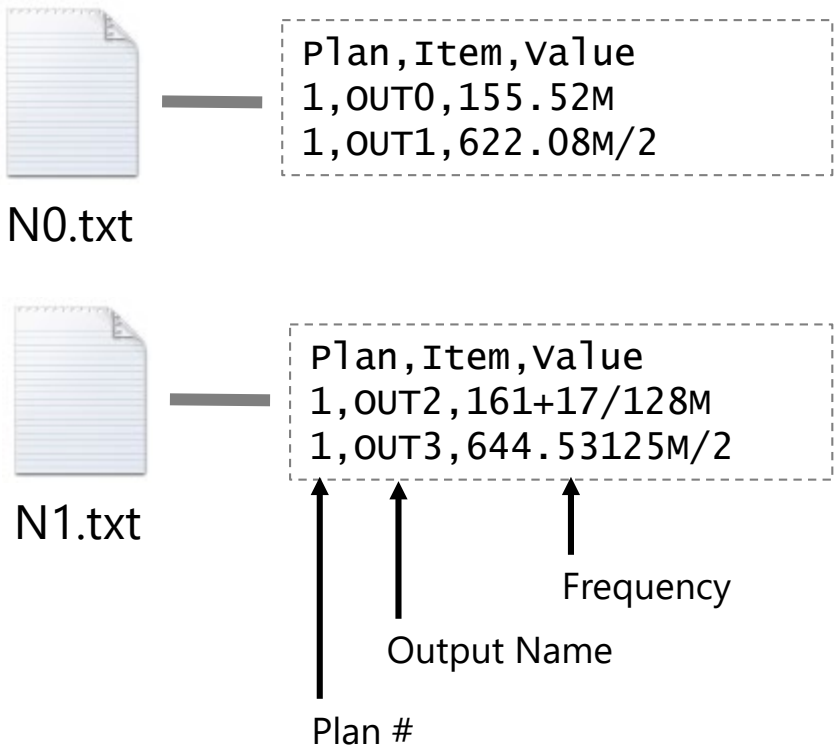
- **Project file defines base output frequencies:**

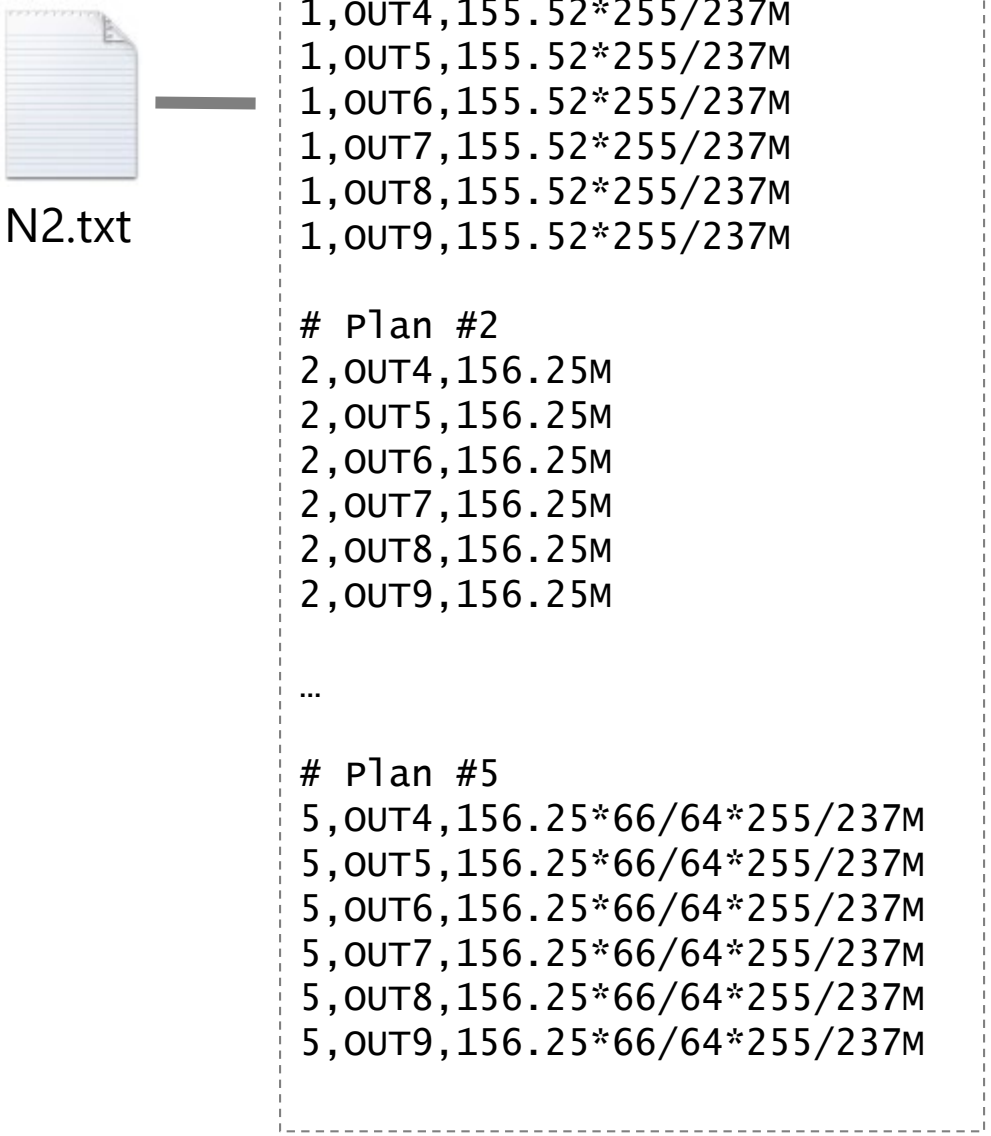
Si5345-RevD-Project.slabtimeproj

Output	Mode	Frequency	N Divider / DCO / ZDM
OUT0	Enabled	161.1328125 MHz	N0
OUT1	Enabled	322.265625 MHz	N0
OUT2	Enabled	155.52 MHz	N1
OUT3	Enabled	311.04 MHz	N1
OUT4	Enabled	148.5 MHz	N2
OUT5	Enabled	148.5 MHz	N2
OUT6	Enabled	148.5 MHz	N2
OUT7	Enabled	148.5 MHz	N2
OUT8	Enabled	148.5 MHz	N2
OUT9	Enabled	148.5 MHz	N2

Si5345 Frequency-On-The-Fly Example

- **Plan Definition Files**





Si5345 Frequency-On-The-Fly Example

- Create a DOS batch script to run the CLI

run.bat

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

- When run, the tool selects the VCO frequency (aka Fvco) that is optimal for all plans

Si5345 Frequency-On-The-Fly Example

- **Files 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, as the VCO frequency may be different to handle all the alternate plans.
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. This must be written at device start-up and reset, even for Orderable Part Numbers.
N*-Register-Script-All-Plans.csv	Register writes to perform to switch to a plan on an N divider. There is one file per divider. This includes all necessary control register writes required to fully load the new configuration (such as Nx_UPDATE writes).
N*-Register-Script-Base-Plan.csv N*-Register-Script-Plan*.csv	This contains the same data as N*-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.

Si5345 Frequency-On-The-Fly Example

- **Host MCU Workflow Example**
  - Host writes Base-Plan-Register-Script.txt
    - Even if you have created an custom part number (aka OPN) for your base project, this step is needed because the CLI may have selected a different VCO frequency to optimize across all plans, not just the base project
  - Host writes plan "script" such as:
    - N0-Register-Script-Plan1.csv – change N0 outputs to plan #1
    - N2-Register-Script-Plan4.csv – change N2 outputs to plan #4
    - N0-Register-Script-Base-Plan.csv – change N0 outputs back to base plan
    - Etc.
  - N*-Register-Script-All-Plans.csv could be used instead, which has a column for each plan

Si5347 Frequency-On-The-Fly Example

DSPLL A:

Output	Base Project	Plan #1	Plan #2	Plan #3
OUT0	161.1328125 MHz	148.5 MHz	155.52 MHz	168.041015625 MHz
OUT1	644.53125 MHz	27 MHz	622.08 MHz	672.1640625 MHz

DSPLL B:

Output	Base Project	Plan #1	Plan #2	Plan #3	Plan #4	Plan #5	Plan #6
OUT2	100M	155.52*255/237M	155.52M	156.25M	132.8125M	159.375M	156.25*66/64*255/237M
OUT3	100M	155.52*255/237M	155.52M	156.25M	132.8125M	159.375M	156.25*66/64*255/237M

DSPLL C:

Output	Base Project
OUT4	155.52 MHz
OUT5	622.08 MHz

DSPLL D:

Output	Base Project
OUT6	148.5 MHz
OUT7	27 MHz

Base = Startup

- The base project essentially is Plan #0: it is called the "Base" plan in documentation and the files created
- Note how there does not need to be an equal number of plans between DSPLLs
- 11 register write scripts are generated to load a plan:
  - 4 for DSPLL A (base, plan #1-3)
  - 7 for DSPLL B (base, plan #1-6
- More details in next slide
- Alternate plans are optional. You'll have at least one or the tool doesn't make sense. In this example FOTF is only done on DSPLL A & B outputs. Outputs on DSPLL C & D are fixed.

This example is bundled in CBPro at C:\Program Files (x86)\Silicon Laboratories\ClockBuilder Pro\CLI\Samples\Multi-PLL-FOTF

User manual goes into more detail (CBProFOTF1 --help)

Si5347 Frequency-On-The-Fly Example

- **Project file defines base output frequencies:**

Si5347-RevD-Project.slabtimeproj

Output	Mode	DSPLL	Frequency
OUT0	Enabled	DSPLL A	161.1328125 MHz
OUT1	Enabled	DSPLL A	644.53125 MHz
OUT2	Enabled	DSPLL B	100 MHz
OUT3	Enabled	DSPLL B	100 MHz
OUT4	Enabled	DSPLL C	155.52 MHz
OUT5	Enabled	DSPLL C	622.08 MHz
OUT6	Enabled	DSPLL D	148.5 MHz
OUT7	Enabled	DSPLL D	27 MHz

Si5347 Frequency-On-The-Fly Example

- **Plan Definition Files**

DSPLLA.txt

```
# DSPLL A  Plans

Plan,Item,Value

# Plan #1
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

# Plan #2
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

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

- Any combination of input frequency, output frequency, bandwidth, and LOL, OOF changes supported
- If a configuration item is not listed, the base project value is used

DSPLLB.txt

```
# 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

# Plan #6
6,OLBW,800
6,FLBW,1000
```

Si5347 Frequency-On-The-Fly Example

- Create a DOS batch script to run the CLI

run.bat

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

- When run, the tool selects the VCO frequency (aka Fvco) that is optimal for all plans

Si5347 Frequency-On-The-Fly Example

- **Files 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, as the VCO frequency may be different to handle all the alternate plans.
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. This must be written at device start-up and reset, even for Orderable Part Numbers.
DSPLL*-Register-Script-Base-Plan.csv DSPLL*-Register-Script-Plan*.csv	This contains the same data as N*-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.

Si5347 Frequency-On-The-Fly Example

- **Host MCU Workflow Example**
  - Host writes Base-Plan-Register-Script.txt
    - Even if you have created an custom part number (aka OPN) for your base project, this step is needed because the CLI may have selected a different VCO frequency to optimize across all plans, not just the base project
  - Host writes plan "script" such as:
    - DSPLLA-Register-Script-Plan1.csv – change DSPLL A outputs to plan #1
    - DSPLLB-Register-Script-Plan3.csv – change DSPLL B outputs to plan #3
    - DSPLLA-Register-Script-Base-Plan.csv – change DSPLL A outputs back to base plan
    - Etc.
  - Example:

```
Address,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
…
0x005A,0xAA,0xFF
0x005B,0xAA,0xFF
0x005C,0x0A,0xFF
0x005D,0x01,0xFF
0x0096,0x88,0x0F
0x0098,0x66,0x0F
0x009B,0x66,0x0F
…
```

If mask is 0xFF, no read-modify-write is required.
Otherwise, read register value and write to device
Read&~Mask + Value&Mask

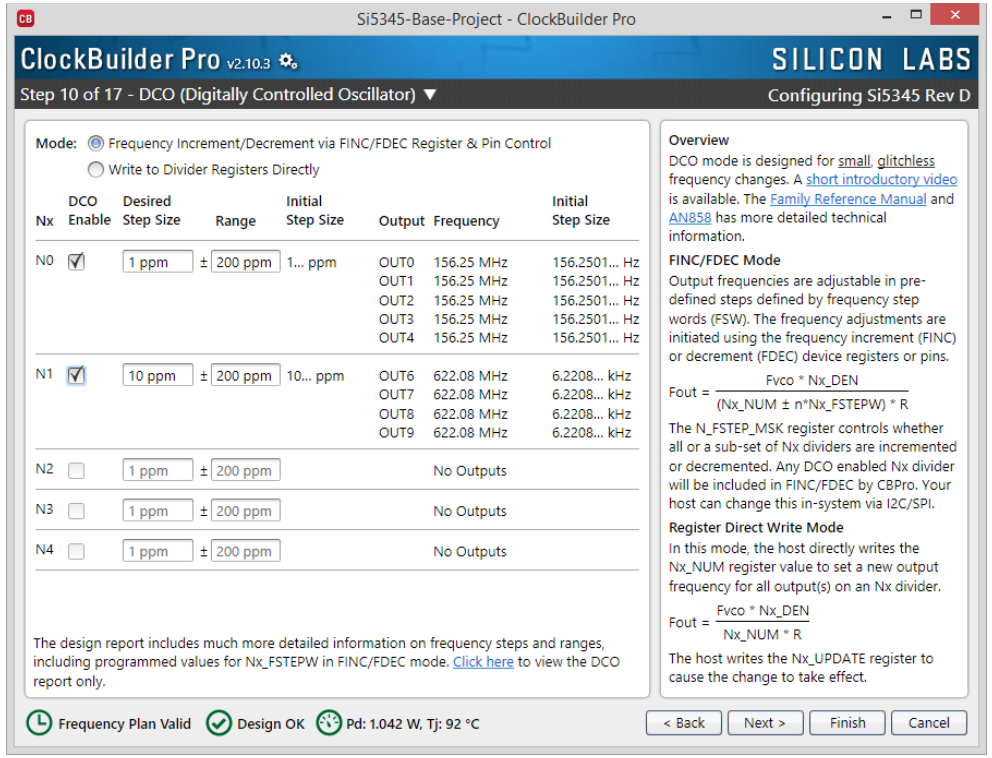
```
# Configuration Postamble
0x0420,0x01,0xFF
0x030C,0x01,0xFF
0x0414,0x01,0xFF
0x0230,0x01,0xFF
0x0092,0x0F,0xFF
0x009A,0x0F,0xFF
0x002C,0x0F,0xFF
0x003F,0xFF,0xFF
0x0436,0x04,0xFF
0x042C,0x87,0xFF
0x010C,0x01,0xFF
0x0116,0x01,0xFF
0x001C,0x02,0xFF
0x0B3C,0xFF,0xFF
0x0B3D,0x00,0xFF
```

Multi-PLL FOTF Restrictions

- **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 MUXd to another DSPLL.**
  - Given
    - IN0 -> DSPLL A
    - IN1 -> DSPLL A
    - IN2 -> DSPLL B
    - IN3 -> DSPLL C/D
  - Can do FOTF on DSPLL A/B
  - Cannot do FOTF on DSPLL C/D
- **Restrictions for 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

Digitally Controlled Oscillator

- When supported, CBPro is used to configure DCO step size and range
- Si5345 example:

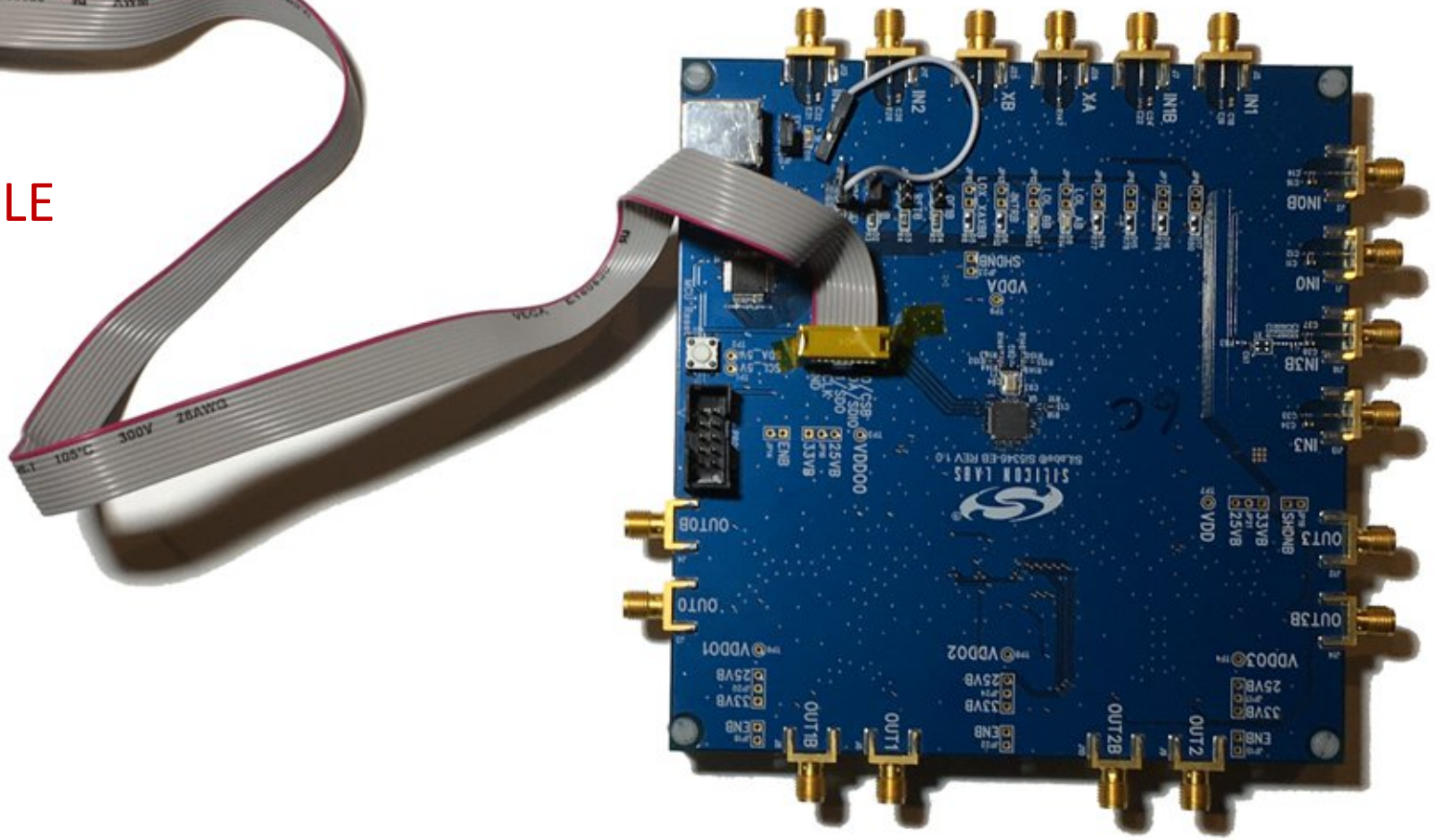


- Frequency step registers set up for you, VCO optimized for DCO range
- See
  - AN858: DCO Applications with the Si5345/44/42
  - AN909: DCO Application with the Si5347/46

Testing Programming with the Field Programmer

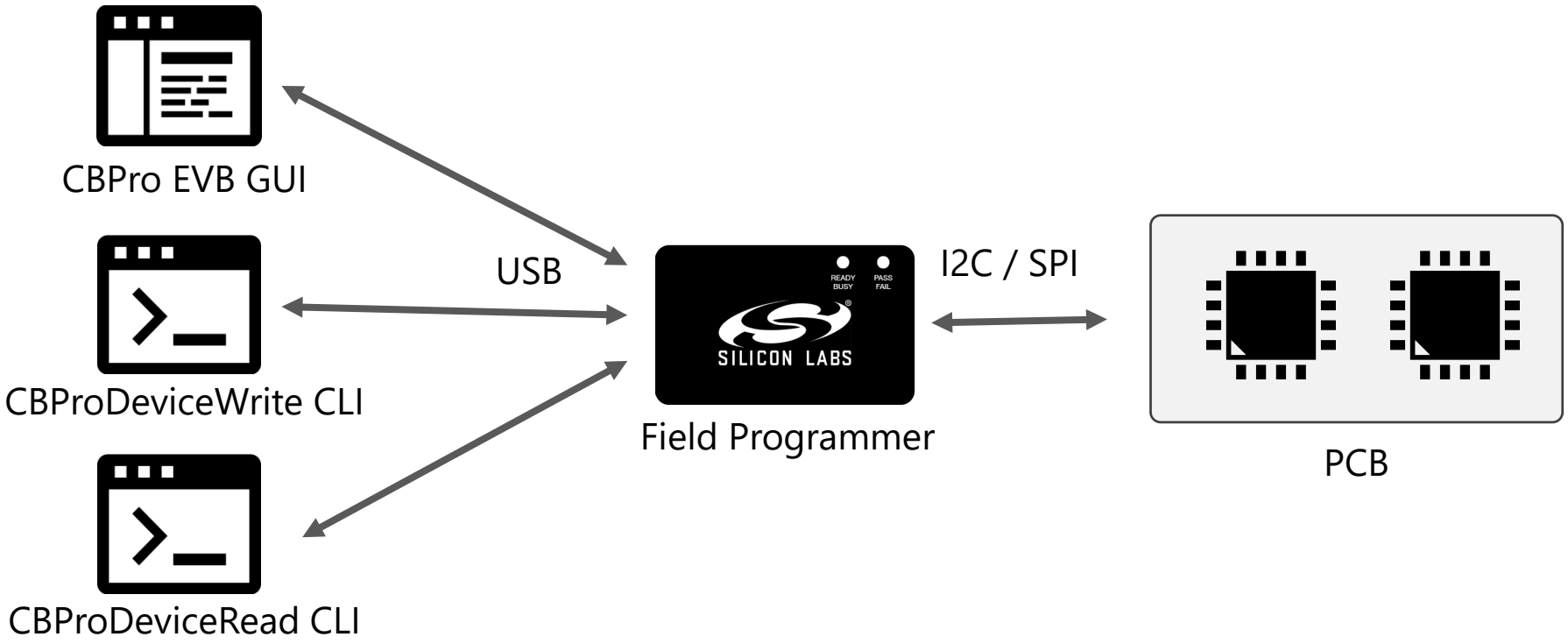
- Officially the "ClockBuilder Pro Field Programmer"
- Can be used to connect to a PCB using I2C or SPI

P/N: CBPROG-DONGLE



Testing Programming with the Field Programmer

- **The following can be written by the EVB GUI or CBProDeviceWrite command line tool:**
  - Project files
  - Register files
  - Setting files
- **Can also read settings and registers via CBProDeviceRead CLI (and EVB GUI in limited way)**
- **Volatile read/write**



Testing Programming

- **Use these tools to experiment with and validate export files created by CBPro**
- **Or test DIY scenarios by creating setting or register files by hand or programmatically**
- **Setting file:**

```
# Write Configuration
OUT0_RDIV_FORCE2,0x0
OUT1_RDIV_FORCE2,0x0
R0_REG,0x1
R1_REG,0x1
N0_NUM,0x2848800000
N0_DEN,0xAD800000

# Load Configuration
N0_UPDATE,0x1
```

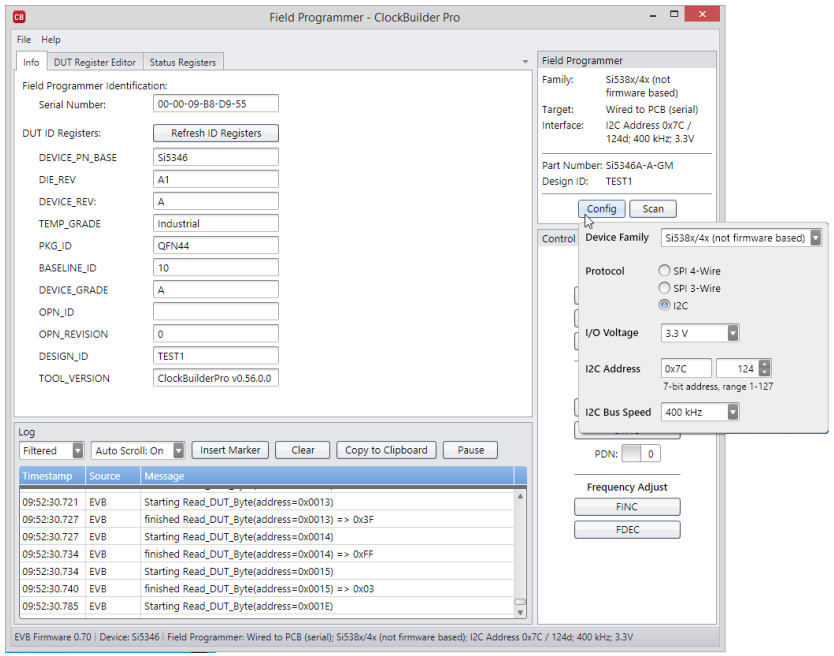
- **Register file:**

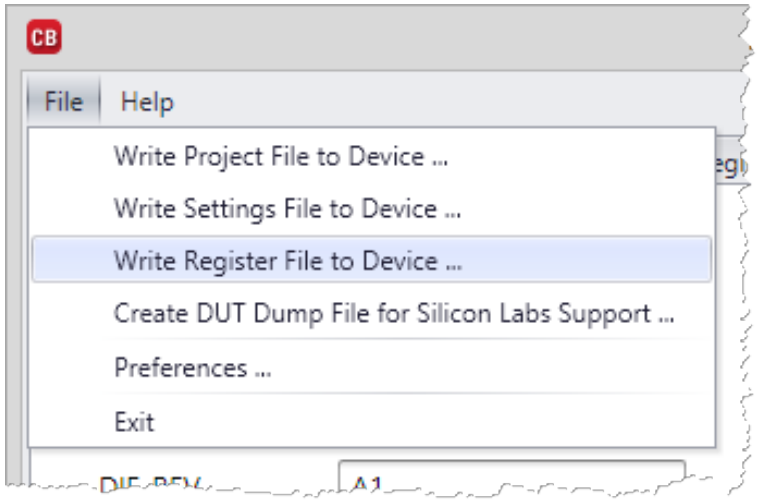
```
# Write Configuration
0x0108,0x02
0x010D,0x02
0x024A,0x01
0x024D,0x01
0x0304,0x80
0x0305,0x48
0x0306,0x28
0x030A,0x80
0x030B,0xAD

# Load Configuration
0x030C,0x01

# Pause 500 ← milliseconds assumed
# Pause 500 msec
```

- Setting values can be hex or decimal
- Register values must be hex
- Newlines OK
- # or // comments
- Comma, space, or tab separator

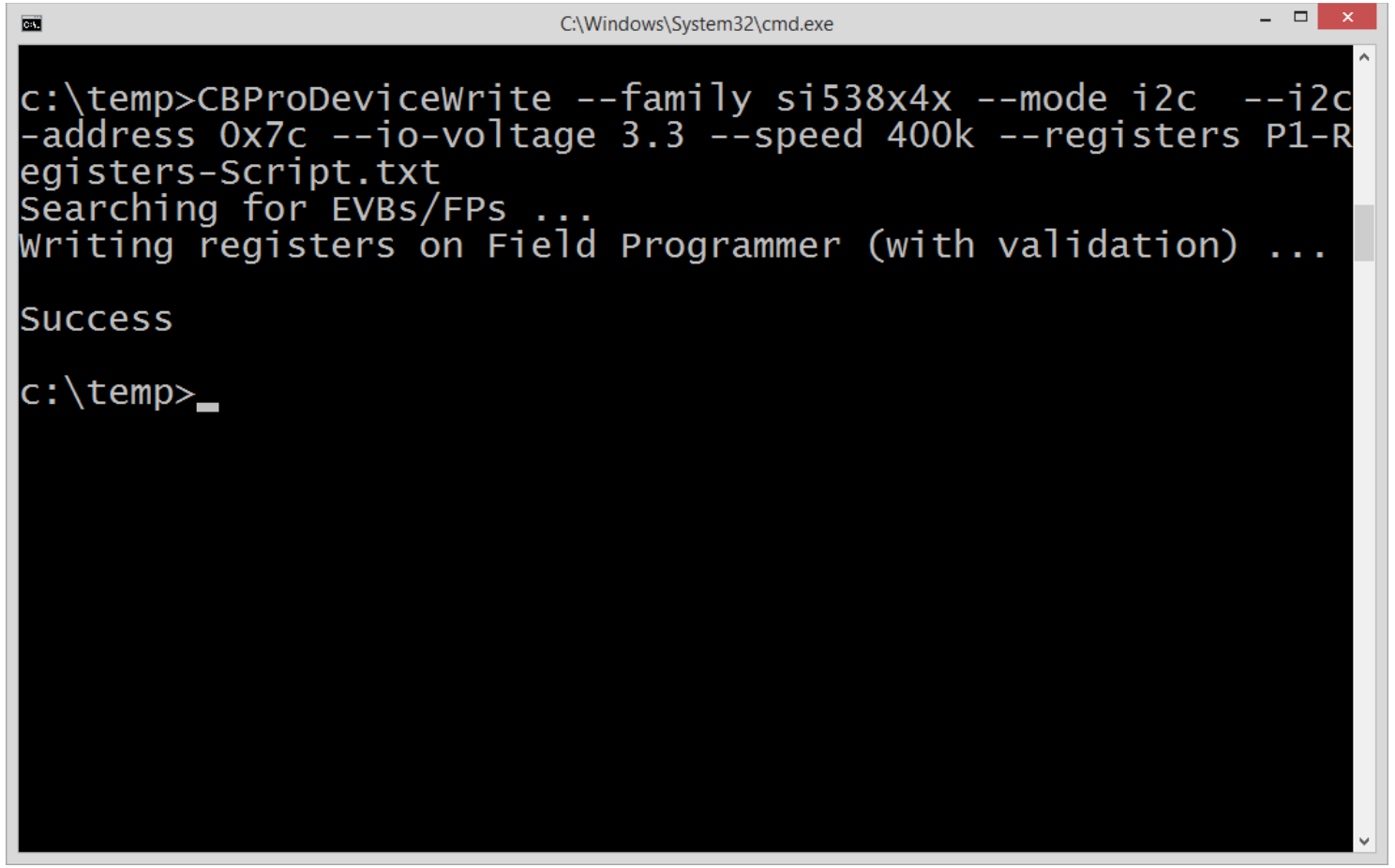
Testing Programming with the Field Programmer

- EVB GUI



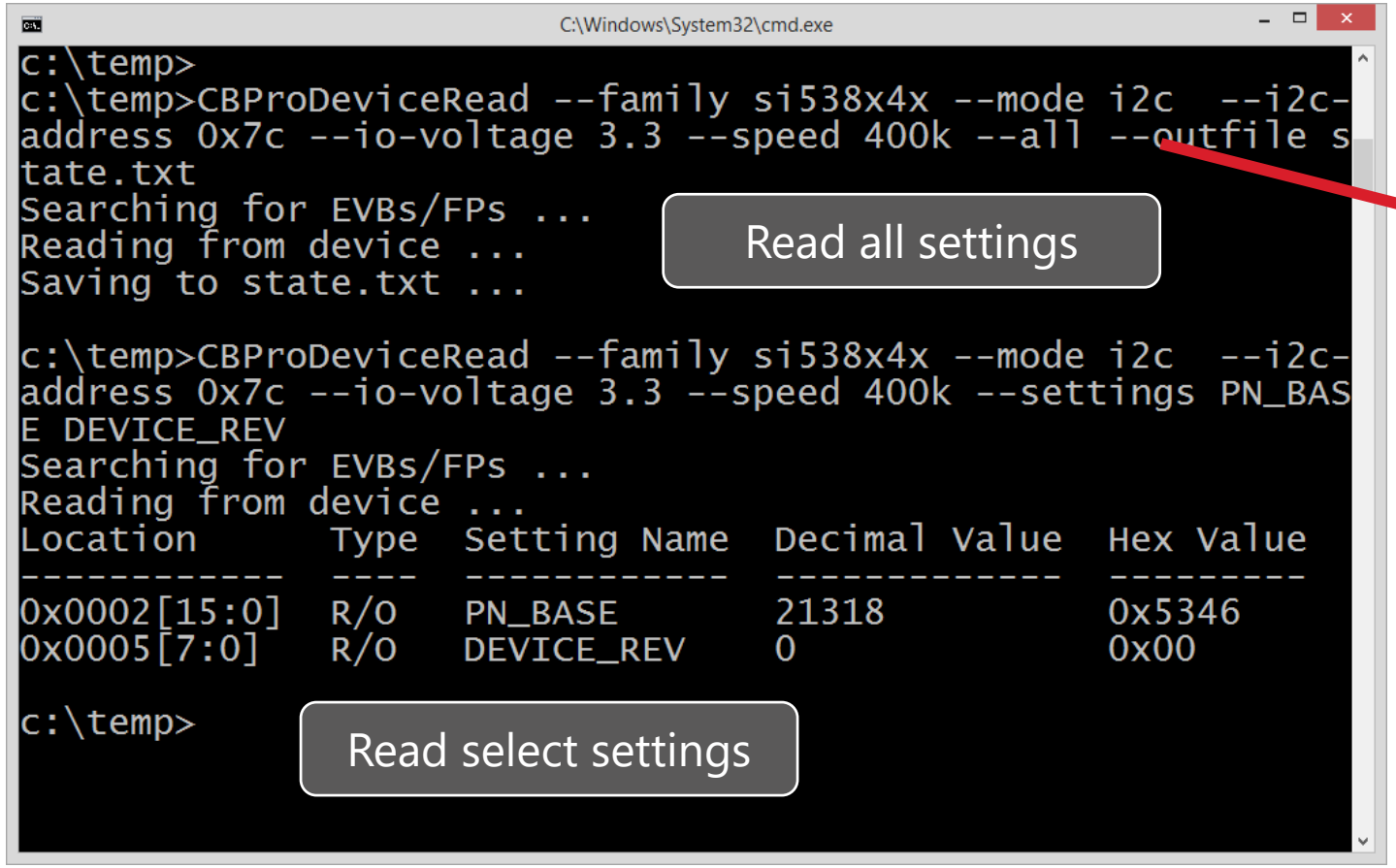


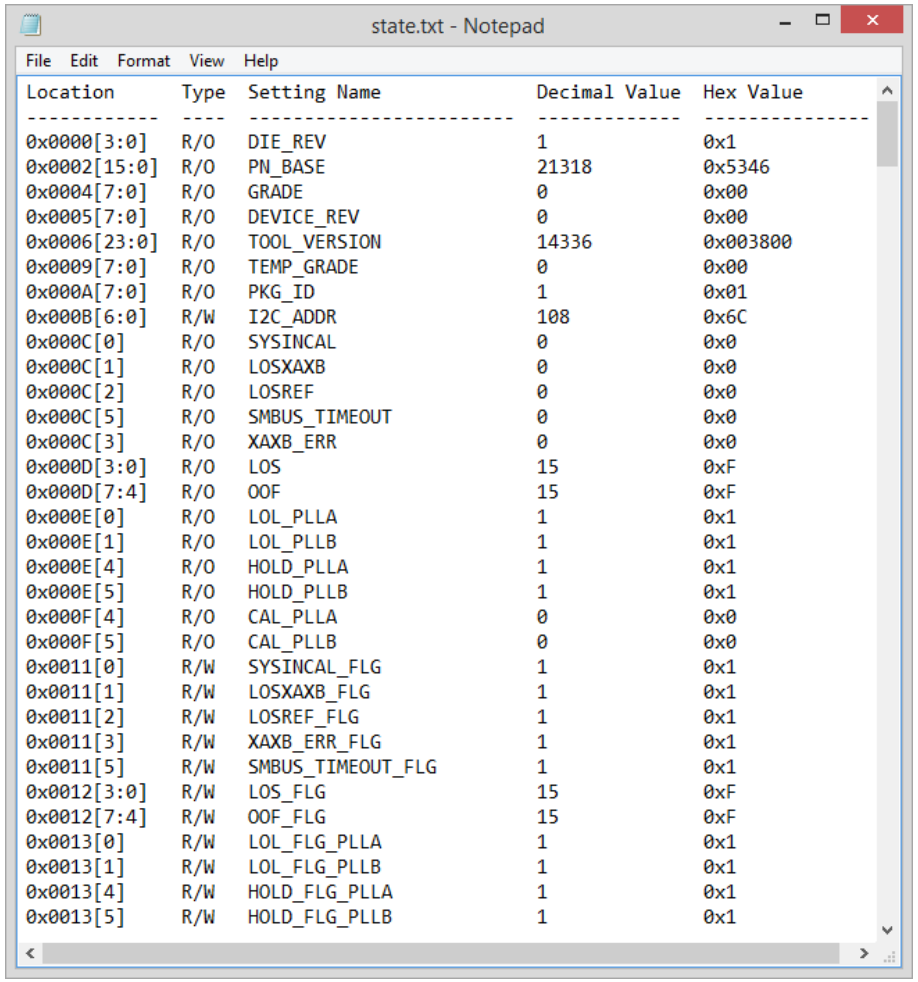
- CBProDeviceWrite CLI



Reading DUT Settings with the Field Programmer

- CBProDeviceRead CLI





Location	Type	Setting Name	Decimal Value	Hex Value
0x0000[3:0]	R/O	DIE_REV	1	0x1
0x0002[15:0]	R/O	PN_BASE	21318	0x5346
0x0004[7:0]	R/O	GRADE	0	0x00
0x0005[7:0]	R/O	DEVICE_REV	0	0x00
0x0006[23:0]	R/O	TOOL_VERSION	14336	0x003800
0x0009[7:0]	R/O	TEMP_GRADE	0	0x00
0x000A[7:0]	R/O	PKG_ID	1	0x01
0x000B[6:0]	R/W	I2C_ADDR	108	0x6C
0x000C[0]	R/O	SYSINCAL	0	0x0
0x000C[1]	R/O	LOSXAXB	0	0x0
0x000C[2]	R/O	LOSREF	0	0x0
0x000C[5]	R/O	SMBUS_TIMEOUT	0	0x0
0x000C[3]	R/O	XAXB_ERR	0	0x0
0x000D[3:0]	R/O	LOS	15	0xF
0x000D[7:4]	R/O	OOF	15	0xF
0x000E[0]	R/O	LOL_PLLA	1	0x1
0x000E[1]	R/O	LOL_PLLB	1	0x1
0x000E[4]	R/O	HOLD_PLLA	1	0x1
0x000E[5]	R/O	HOLD_PLLB	1	0x1
0x000F[4]	R/O	CAL_PLLA	0	0x0
0x000F[5]	R/O	CAL_PLLB	0	0x0
0x0011[0]	R/W	SYSINCAL_FLG	1	0x1
0x0011[1]	R/W	LOSXAXB_FLG	1	0x1
0x0011[2]	R/W	LOSREF_FLG	1	0x1
0x0011[3]	R/W	XAXB_ERR_FLG	1	0x1
0x0011[5]	R/W	SMBUS_TIMEOUT_FLG	1	0x1
0x0012[3:0]	R/W	LOS_FLG	15	0xF
0x0012[7:4]	R/W	OOF_FLG	15	0xF
0x0013[0]	R/W	LOL_FLG_PLLA	1	0x1
0x0013[1]	R/W	LOL_FLG_PLLB	1	0x1
0x0013[4]	R/W	HOLD_FLG_PLLA	1	0x1
0x0013[5]	R/W	HOLD_FLG_PLLB	1	0x1

Programming Debugging

- Have a register file (address,data) but want to know what it means?
- New tool as of CBPro 2.21.6 can map register data to named settings (aka bitfields)
- CBProRegistersToSettings CLI
- Register file:

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

- CLI:

```
CBProRegistersToSettings --part si5345 --rev d --infile regs.txt
```

- Output:

```
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
```

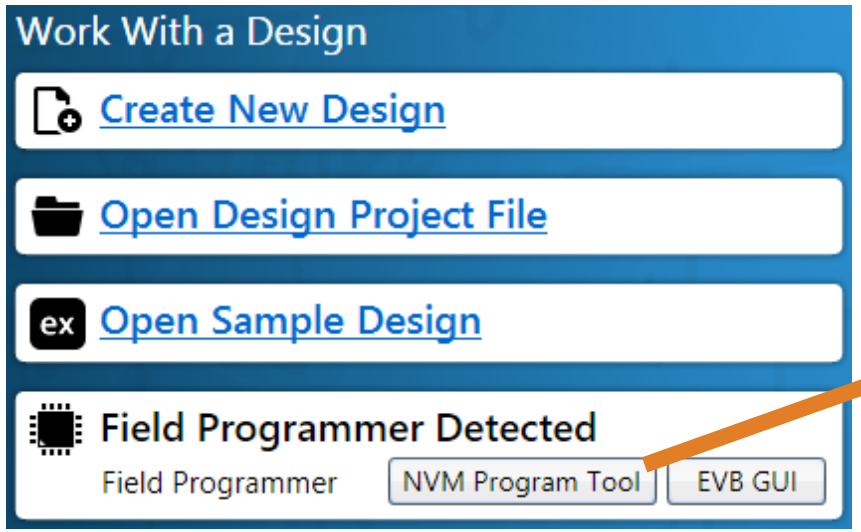
- CSV, HTML output options available

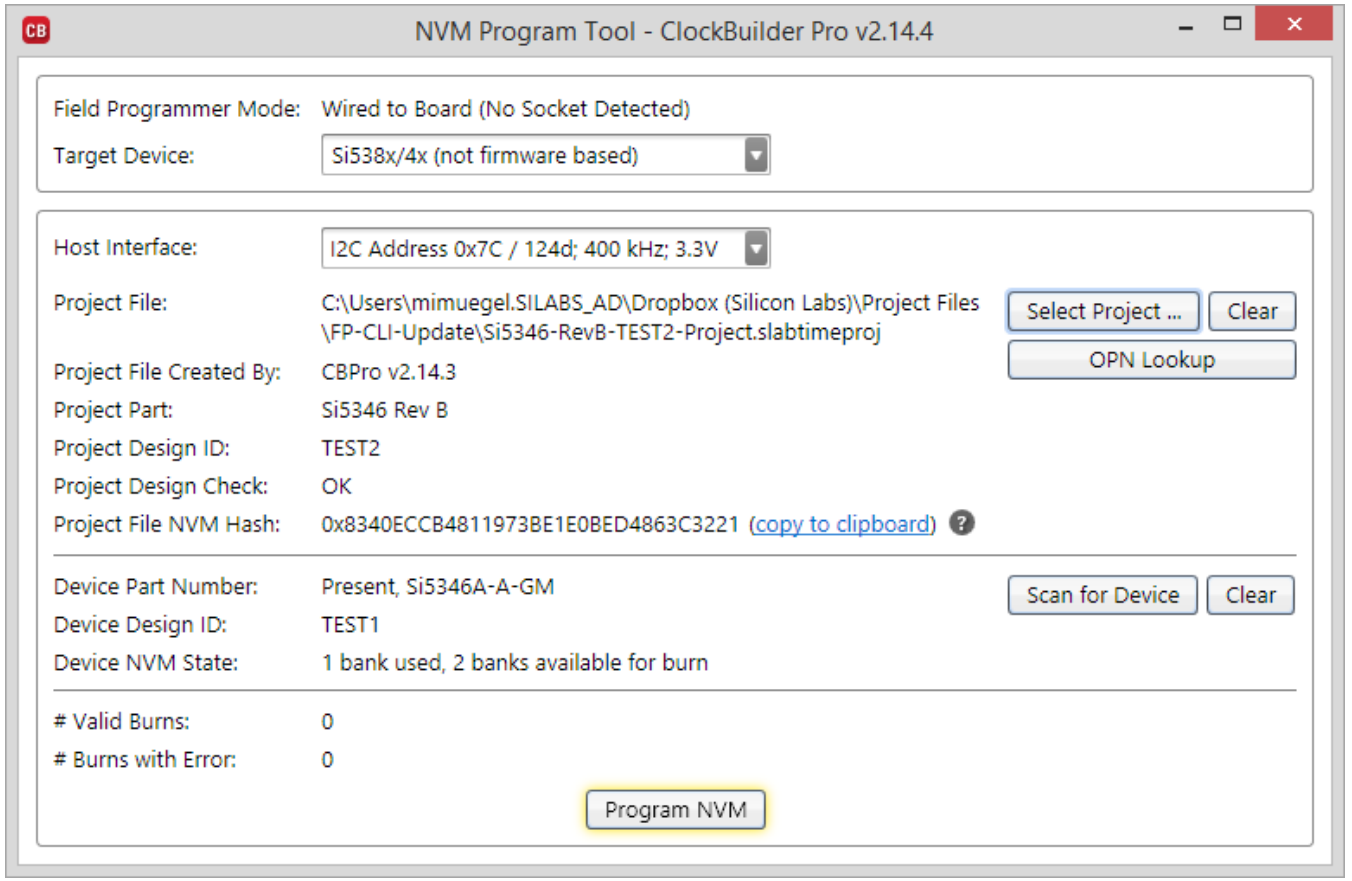
Burning NVM with the Field Programmer

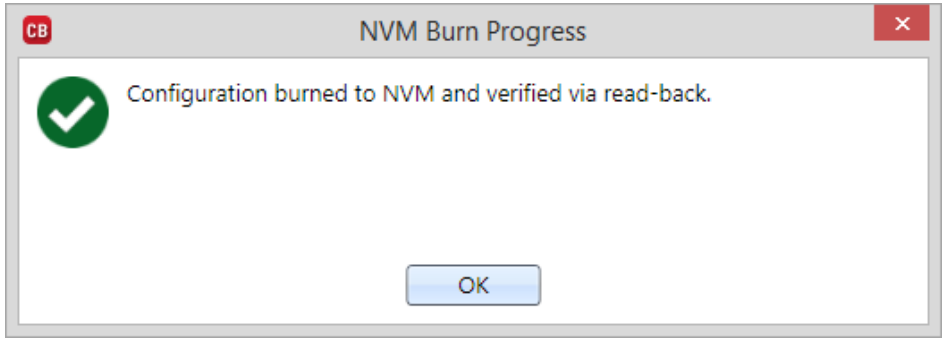
- On supported devices, can also use FP to burn non-volatile memory
- Modes:
  - IC in socket
  - IC on PCB connected via I2C or SPI
- Sockets for Si5332, Si534x, Si538x, and Si539x

Burning NVM with the Field Programmer

- **Burn Tool:**

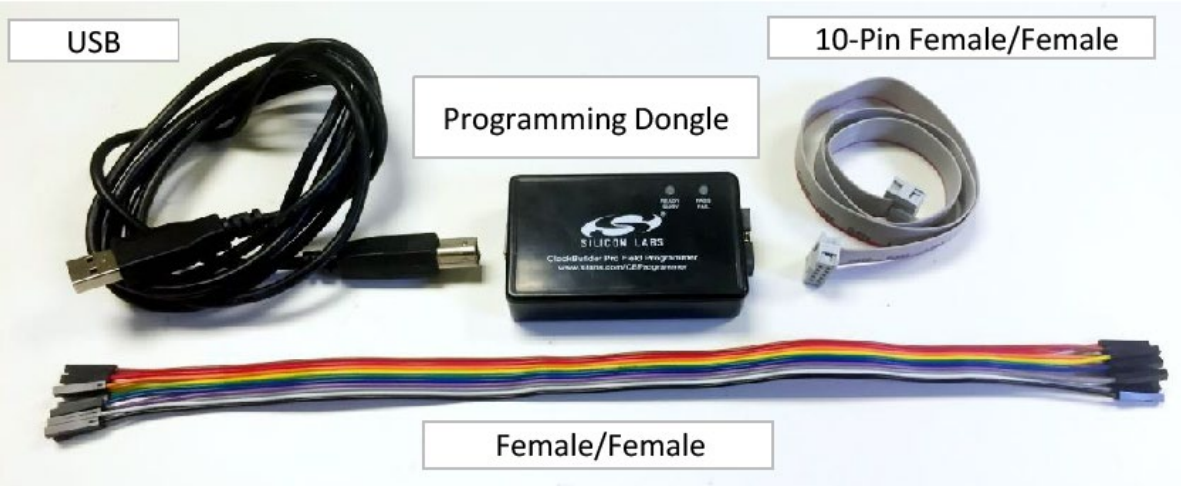






Field Programmer Kit

- **Included:**



P/N CBPROG-DONGLE

- **Sockets available separately:**
  - Si538X4X-64SKT-DK
  - Si538x4x-56SKT-DK
  - Si538X4X-44SKT-DK
  - Si5332-32SKT-DK
  - Si5332-40SKT-DK
  - Si5332-48SKT-DK

- [User guide]

Learning More

- **Read the CBPro Tools & Support for In-System Programming training deck (some overlap with this training)**
- **Read the CLI User Guide**
  - Installation overview
  - User manual for each CLI tool
- **CLI user manuals also available via:**
  - Type "*cmd* `--help`" from DOS prompt
  - C:\Program Files (x86)\Silicon Laboratories\ClockBuilder Pro\CLI\Docs
- **Review CLI samples in C:\Program Files (x86)\Silicon Laboratories\ClockBuilder Pro\CLI\Samples**
  - They are the samples referenced in this training materials

Tool Briefs

- Edit Project File CLI
- Register Export GUI & CLI
- Settings Export GUI & CLI
- Multi-Project Export GUI & CLI
- Multiple Edits & Export CLI
- Register Map Export GUI & CLI
- Device Read GUIs & CLI
- Device Write GUIs & CLI
- Registers -> Settings CLI
- Export Firmware GUI & CLI
- Download Firmware CLI

Edit Project File CLI

- Change input frequency, output frequency, clock state, DSPLL assignment, and/or bandwidth of a project file & save to a new project file (support for some items varies based on device)
- CLI is CBProProjectEdit; GUI editing done via standard CBPro wizard
- Edit file defines changes; Si5345 example:

```
Clock,State,Frequency

# Comments are supported
# IN1,14.4MHz

OUT0,Unused,
OUT5,Enabled,OUT4*2

OLBW,,1k
FLBW,,2k
```

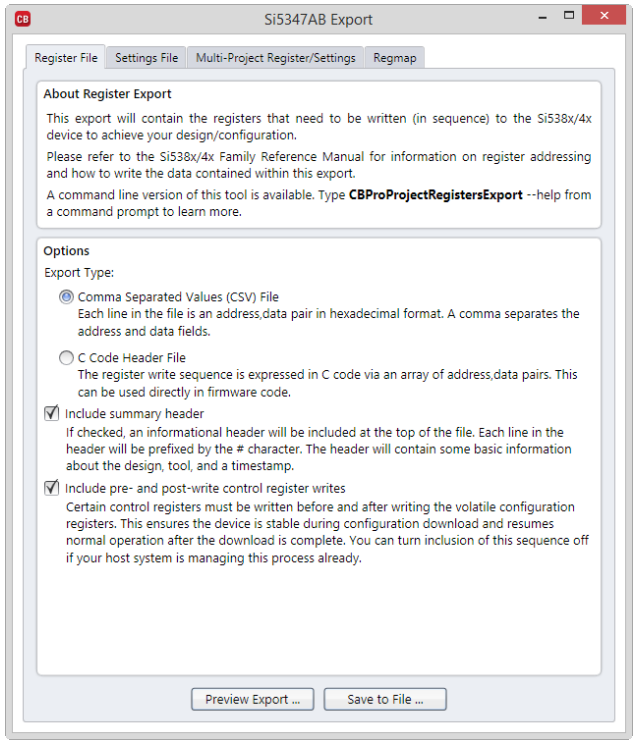
- Run:

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

Register Export GUI & CLI

- Creates register write script that includes all necessary control register writes
- So turn key
- Can generate script as CSV or "C" code file
- GUI front end via Export tool
- CLI is CBProProjectRegistersExport
- CLI Example:

```
CBProProjectRegistersExport.exe --project Si5345-Project.slabtimeproj
                                --include-load-writes --format csv
                                --outfile Si5345-Config-Script.txt
```



- Example CSV script:

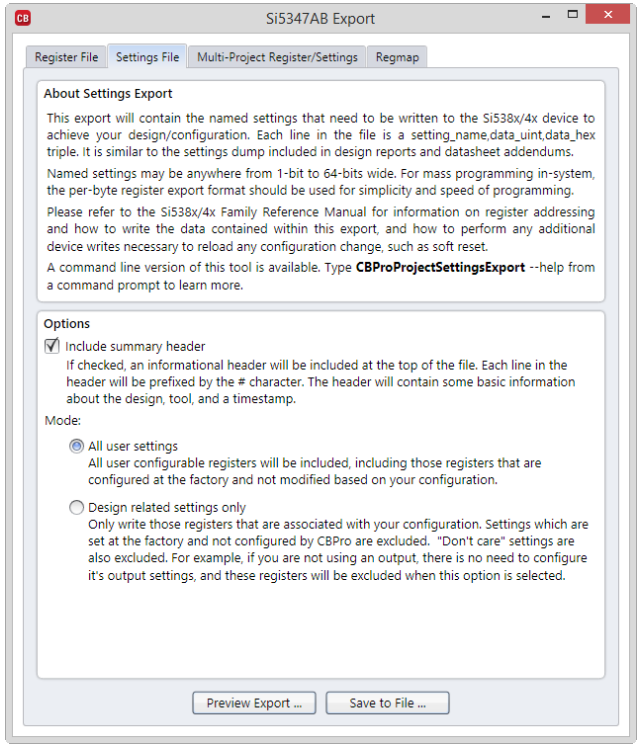
```
# Part: Si5345
# Project File: Si5345-Original-Project.slabtimeproj
# Design ID: <none>
# Includes Pre/Post Download Control Register Writes: Yes
# Die Revision: B1
# Creator: CBProProjectRegistersExport v2.10.1 [2016-09-08]
# Created On: 2016-09-08 15:50:16 GMT-05:00
Address,Data
0x0B24,0xC0
0x0B25,0x00
0x0004,0x00
…
```

- Example C header file:

```
#define SI5345_REVD_REG_CONFIG_NUM_REGS 512

typedef struct
{
        unsigned int address; /* 16-bit register address */
        unsigned char value; /* 8-bit register data */

} si5345_revd_register_t;

si5345_revd_register_t const
si5345_revd_registers[SI5345_REVD_REG_CONFIG_NUM_REGS] =
{
        { 0x0B24, 0xC0 },
        { 0x0B25, 0x00 },
        { 0x0004, 0x00 },
        …
}
```

Settings Export GUI & CLI

- Exports configuration named settings (register bitfields)
- Same information that is in design report, but CSV formatted
- GUI front end via Export tool



- CLI is CBProProjectSettingsExport
- CLI Example:

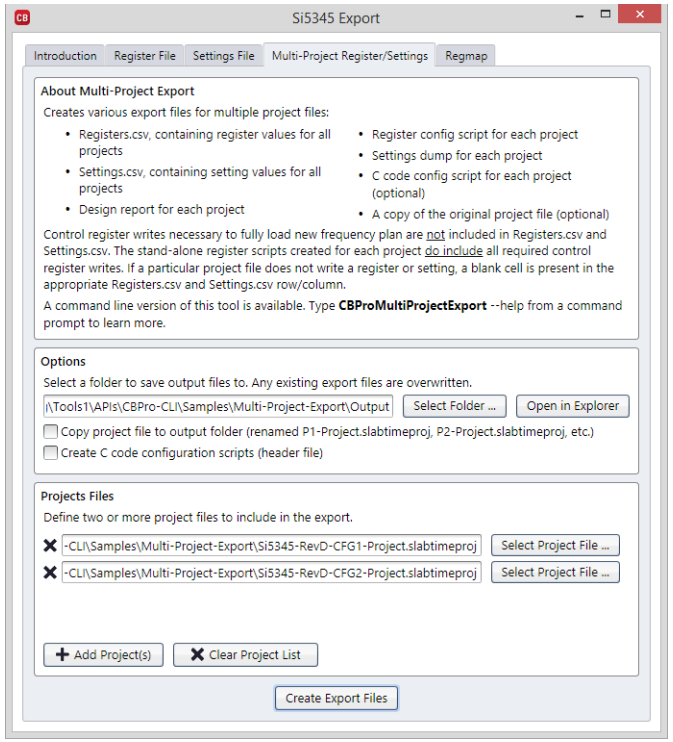
```
CBProProjectSettingsExport.exe --project Si5345-Project.slabtimeproj
                               --outfile Si5345-Settings.txt
```

- Output

```
Location,SettingName,DecimalValue,HexValue
0x0004[7:0],GRADE,0,0x00
0x0006[23:0],TOOL_VERSION,0,0x000000
0x0016[1],LOL_ON_HOLD,1,0x1
0x0017[0],SYSINCAL_INTR_MSK,0,0x0
0x0017[1],LOSXAXB_INTR_MSK,0,0x0
0x0017[5],SMB_TMOUT_INTR_MSK,0,0x0
…
```

Multi-Project Export GUI & CLI

- **Given 2+ project files, creates unified register and setting dumps in CSV format**
  - Has "Varies" column that can be used to quickly determine whether a register/setting differs between projects
  - Note this does not include any programming pre- and post-amble
  - Useful to analyze differences in project files
- **Also creates per project:**
  - Design report, register write script (full config scenario), settings dump
  - Optionally create C code config script, copy original project file
- **GUI front end via Export tool**
- **CLI is CBProMultiProjectExport**
- **Example Output:**



Address	Varies	P1	P2
0x009A	No	0x02	0x02
0x009B	No	0x60	0x60
0x009D	Yes	0x08	0x04
0x009E	No	0x40	0x40
0x00A0	No	0x20	0x20
0x00A2	No	0x00	0x00
0x00A9	Yes	0x33	0x37
0x00AA	Yes	0x63	0x30
0x00AB	No	0x00	0x00
0x00AC	No	0x00	0x00
0x00E5	No	0x21	0x21
…	…	…	…

Registers.csv

Location	Setting	Varies	P1	P2
0x009A[1]	LOL_SLOW_EN_PLL	No	0x1	0x1
0x009B[7:4]	LOL_SLW_DETWIN_SEL	No	0x6	0x6
0x009D[3:2]	LOL_SLW_VALWIN_SEL	Yes	0x2	0x1
0x009E[7:4]	LOL_SLW_SET_THR	No	0x4	0x4
0x00A0[7:4]	LOL_SLW_CLR_THR	No	0x2	0x2
0x00A2[1]	LOL_TIMER_EN	No	0x0	0x0
0x00A9[28:0]	LOL_CLR_DELAY_DIV256	Yes	0x6333	0x3037
0x00E5[5]	FASTLOCK_EXTEND_EN	No	0x1	0x1
…	…	…	…	…

Settings.csv

P1-Report.txt
P2-Report.txt

P1-Registers-Script.txt
P2-Registers-Script.txt

P1-Registers-Script-Delta-Only.txt
P2-Registers-Script-Delta-Only.txt

Multiple Edits & Export CLI

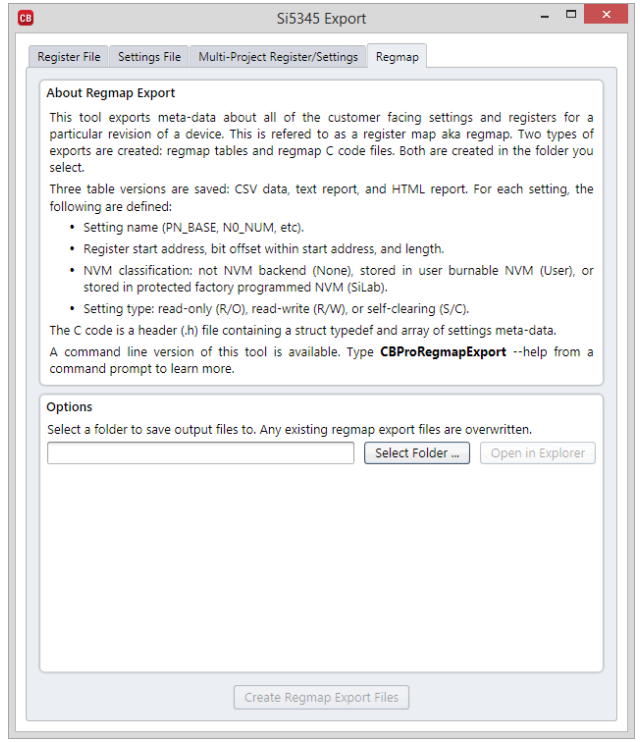
- CBProMultiEditAndExport CLI
- Combines the features of CBProProjectEdit and CBProMultiProjectExport CLIs

- Run:

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

Register Map Export GUI & CLI

- Exports meta-data about all of the customer facing settings and registers for a particular revision of a device: the regmap
- Creates regmap CSV,HTML,Text tables and C code header file
- Both could be used to create a bitfield-level (named settings) API to the DUT registers
- See device family reference manual for DUT R/W details
- GUI front end via Export tool
- CLI is CBProRegmapExport
- Example Output:



Si5345-RevD-Regmap.csv

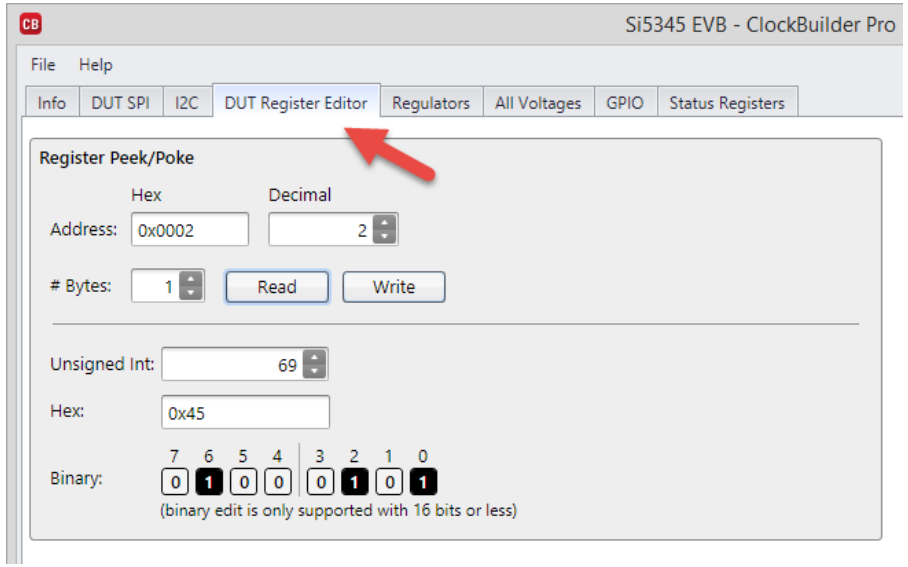
Setting Name	Location	Start Address	Start Bit	Num Bits	NVM	Type
DIE_REV	0x0000[0:3]	0x0000	0	4	None	R/O
PAGE	0x0001[0:7]	0x0001	0	8	None	R/W
PN_BASE	0x0002[0:15]	0x0002	0	16	SiLab	R/W
GRADE	0x0004[0:7]	0x0004	0	8	SiLab	R/W
…	…	…	…	…	…	…

Si5345-RevD-Regmap.h

```
typedef struct
{
     CHAR*  name;           /* Setting/bitfield name                                          */
     UINT8  read_only;      /* 1 for read only setting/regs or 0 for read/write               */
     UINT8  self_clearing;  /* 1 for self clearing setting/registers or 0 otherwise           */
     UINT8  nvm_type;       /* 0 for not NVM backed; 1 for "silabs" bank; 2 for "user" bank   */
     UINT8  bit_length;     /* Number of bits in setting                                      */
     UINT8  start_bit;      /* Least significant bit of the setting                           */
     UINT8  reg_length;     /* Number of registers that the setting is stored in              */
     UINT16 addr[SI5345_REVD_MAX_NUM_REGS]; /* Addresses the setting is contained in          */
     UINT8  mask[SI5345_REVD_MAX_NUM_REGS]; /* Bitmask for each register containing the setting */
} si5345_revd_regmap_t;

si5345_revd_regmap_t const si5345_revd_settings[SI5345_REVD_NUM_SETTINGS] =
{

     /* DIE_REV */
     {
          "DIE_REV",
          1, /* 1 = IS Read Only */
          0, /* 0 = NOT Self Clearing */
          SLAB_NVMT_NONE, /* Not stored in NVM */
          4, /* 4 bits in this setting */
          0, /* setting starts at b0 in first register */
          1, /* contained in 1 registers(s) */
          { 0x0000, /* Register address 0 b7:0 */ },
          { 0x0F, /* Register mask 0 */ }
     },
     …
}
```

Device Read GUIs & CLI

- **EVB GUI has simple register read/write**



- **CBProDeviceRead CLI**
  - Read all read-only and read-write settings
  - Read select settings specified on command line
  - Read select registers specified on command line
  - Supports EVB or field programmer; examples to right for EVB; field programmer requires more command line options to configure comms

- **CLI Examples**

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

Searching for devices ...
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
```

```
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
```

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

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

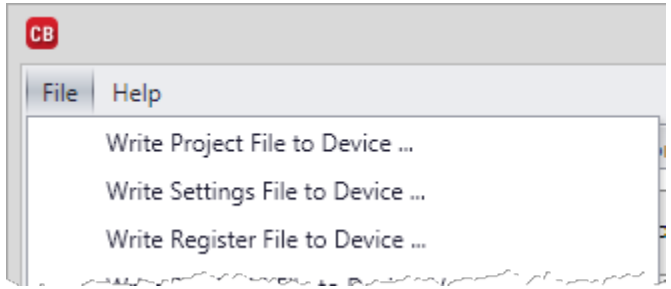
Device Write GUIs & CLI

- **Project files, setting files, and register files can be written to:**
  - Silicon Labs EVB
  - Your system board using the ClockBuilder Pro field programmer via serial connection
- **Writing register files – such as scripts output from config export and FOTF CLI tool – provide an easy way to prototype in-system programming using Silicon Labs EVB or by using field programmer wired to your system board over I2C or SPI**
- **Register file syntax**
  - Address data pairs: comma or space seperated
  - Hex assumed
  - # and // comments supported
  - Delay between writes supported
- **Register file 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

Pause 500 ← msec assumed
# Pause 500 msec
// Pause 500msec
Delay 500 msec
```

Device Write GUIs & CLI

- **Writing from EVB GUI**



- **Writing from CLI**
  - EVB Example – Write a Project File:

    ```
    CBProDeviceWrite --project Si5345.slabtimeproj
    ```

  - Field Programmer Example – Write a Register Script:

    ```
    CBProDeviceWrite --family si538x4x --io-voltage 3.3 --mode spi4wire --speed 6M
                     --registers Si5345-Config1.txt
    ```

Register -> Settings CLI

- CBProRegistersToSettings CLI
- Map registers to the device named settings (aka bitfields)
- Input is a comma or space separated list of register address value pairs
- If a setting's registers are only partially specified, 0x00 is assumed for missing addresses
- Register file:

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

- CLI:

```
CBProRegistersToSettings --part si5345 --rev d --infile regs.txt
```

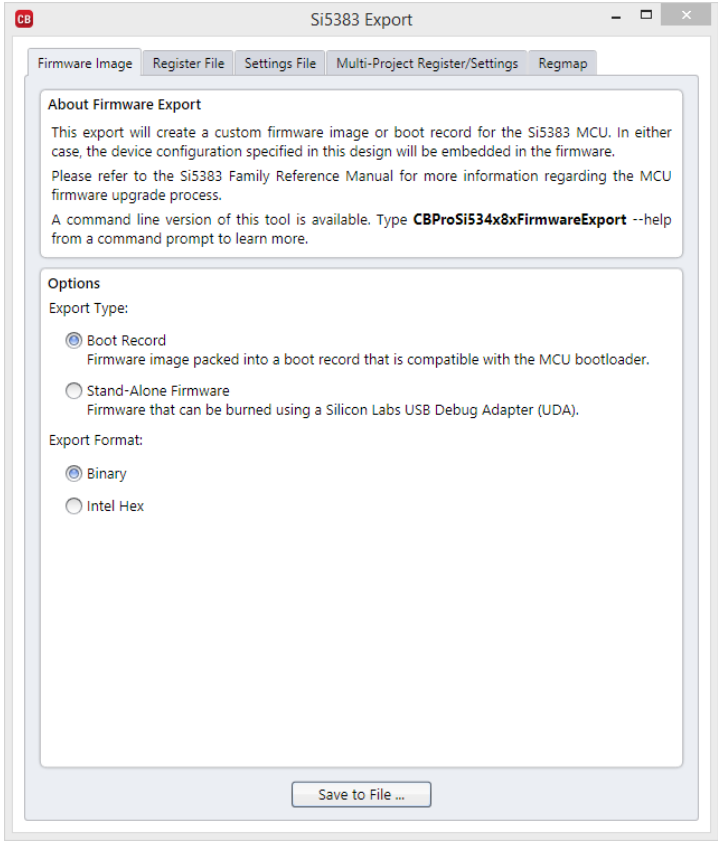
- Output:

```
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
```

- CSV, HTML output options available

Export Firmware GUI & CLI

- Create a custom firmware image for MCU embedded in select devices
- The device configuration in the specified project file will be embedded in the firmware, ensuring program and configuration are both flashed
- Boot record or stand-alone file files can be created
- Binary or intel hex formats supported
- GUI front end via Export tool



- CLI is CBProSi534x8xFirmwareExport
- Example:

```
CBProSi534x8xFirmwareExport --type bootrecord --format hex
                            --project sixxxx.slabtimeproj
                            --outfile sixxxx.hex
```

```
:020000040000FA
:20006000123DE78F2F123DE7EF547FF52E301A0343E6807B007A00792E22D29922FF022A05
:2000800067D208D20CC203C20978F02483330080E2B40E004003020CDF900090F828287340
:2000A0000200BA0200CA0200F702019202038B0203F30205260205B1020724020868020913
:2000C000B4020ACE020B58020C4178F07401F2755AFF755B1B755C428027123A2378F04085
:2000E000117402F2123D17755AFF755B1B755C4B800F7401F2123D17755AFF755B1B755C66
:2001000042E4F55DF55E2278E0E230E11978F07424833301000DF2D2037F0312229F7F032C
:20012000122F0AD21A53917F020CCC7823E220E06A7820E230E52278F0740BF2C208D21727
:20020000AA06A905A80490029CE0FCA3E0FDA3E0FEA3E0FFC31213E45005248332020090BB
:20022000029C800390023EE0FCA3E0FDA3E0FEA3E0FF90023E121499900228E0FCA3E0FDC9
…
```

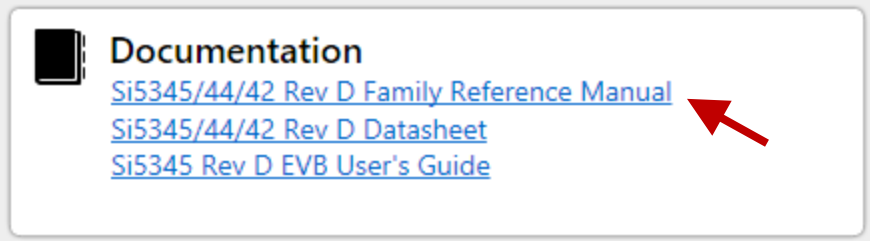
Download Firmware CLI

- Flashes firmware on device with MCU
- Supports any of the file formats that the firmware export can create (boot record or stand-alone; intel hex or binary)
- Can target Silicon Labs EVB or your system board using via ClockBuilder Pro field programmer and serial cable
- Currently no GUI: CLI only
- Example of flashing device on system board via I2C:

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

Parsing firmware file ...
Trying to send device to bootloader mode ...
Success
Flashing firmware ...
0% complete
… skipped …
100% complete
Verifying flash ...
Firmware flash complete
Detected Si5383 in program mode
FIRMWARE_TYPE is 8
…
```

Learning More

- **Read the CLI User Guide**
- **Review documentation for each command line tool**
  - Embedded in PDF above
  - Or type "*cmd* `–help`" from DOS prompt
  - Also in C:\Program Files (x86)\Silicon Laboratories\ClockBuilder Pro\CLI\Command-Help
- **Review samples in C:\Program Files (x86)\Silicon Laboratories\ClockBuilder Pro\CLI\Samples**
  - They are the samples referenced in this guide
- **Device documentation**

Family Reference Manuals (FRMs)	Documents all user registers on a device. Device reset and initialization is also covered, which may be needed under certain use cases. FRMs are available from the CBPro dashboard: 
AN926: Reading and Writing Registers with SPI and I2C for Si534x/8x Devices	While each Family Reference Manual describes available customer registers and general addressing overview, this document describes Si534x/8x/9x I/O in detail, including providing C code snippets.
AN858: DCO Applications with the Si5345/44/42 AN909: DCO Application with the Si5347/46	Defines Si534x/8x/9x DCO registers, the mathematical equations to define their values, and documents general DCO workflow. An example is worked through.