Using the git repository

Initially, create a local copy of the CMv2 hardware repository with the command:

git clone git@github.com:apollo-lhc/Cornell_CM_Rev2_HW

If you already have a local repository and you want to get all updates:

git pull

If you only want the update for a single file:

git fetch –all

git checkout origin/master name_of_file

Do your work in the sub-directories of the “ClockBuilderPRO” directory. When you are ready to save stuff in the repository, use:

git status

If the status shows that you have files that are not part of the repository, and you want them to be:

git add name_of_file

Update the local repository and push the changes to the repository on github.com:

git commit -a -m “description of changes”

git push origin master

Running the LoadSynth program

Change directory to the “ClockBuilderPRO/programs” directory:

cd programs

Start the LoadSynth program, specifying which synth to configure and which file to use as the source of the configuration data:

python3 LoadSynth.py R0A ../projects/Si5341-RevD-R0Av0001-Registers.h --quiet

The synth is one of [R0A, R0B, R1A, R1B, R1C]. The case does not matter. One must be specified.

The relative path specification ahead of the configuration data file name is required if the register file is not in the same directory as the program (and it should not be). A configuration file must be specified

Running the RampSynth program

Change directory to the “ClockBuilderPRO/programs” directory:

cd programs

Start the RampSynth program, specifying which synth to configure, what type of ramping to do, and which file to use as the source of the configuration data:

```
python3  RampSynth.py R0A  up ../projects/Si5341-RevD-R0Av0003-Registers.h --quiet
```

The synth is one of [R0A, R0B, R1A, R1B, R1C]. The case does not matter.

The type is one of [UP, DOWN, BOTH, CONT]. The case does not matter. “UP” will ramp by increasing the frequency. “DOWN” will ramp by decreasing the frequency. “BOTH” will first ramp down, then ramp up. It will stop at the original frequency. “CONT” will continuously ramp down, then up, then down… It will go until interrupted, at which time it will leave the synthesizer at whatever the frequency currently is. To get back to a known frequency you need to run “LoadSynth” again.

The relative path specification ahead of the configuration data file name is required if the register file is not in the same directory as the program (and it should not be). The only need for the configuration file is to extract the data needed to calculate the number of steps.

NOTE: Maybe add a command line parameter “steps” to allow the user to specify the number of steps, rather than using the number extracted from the configuration data.

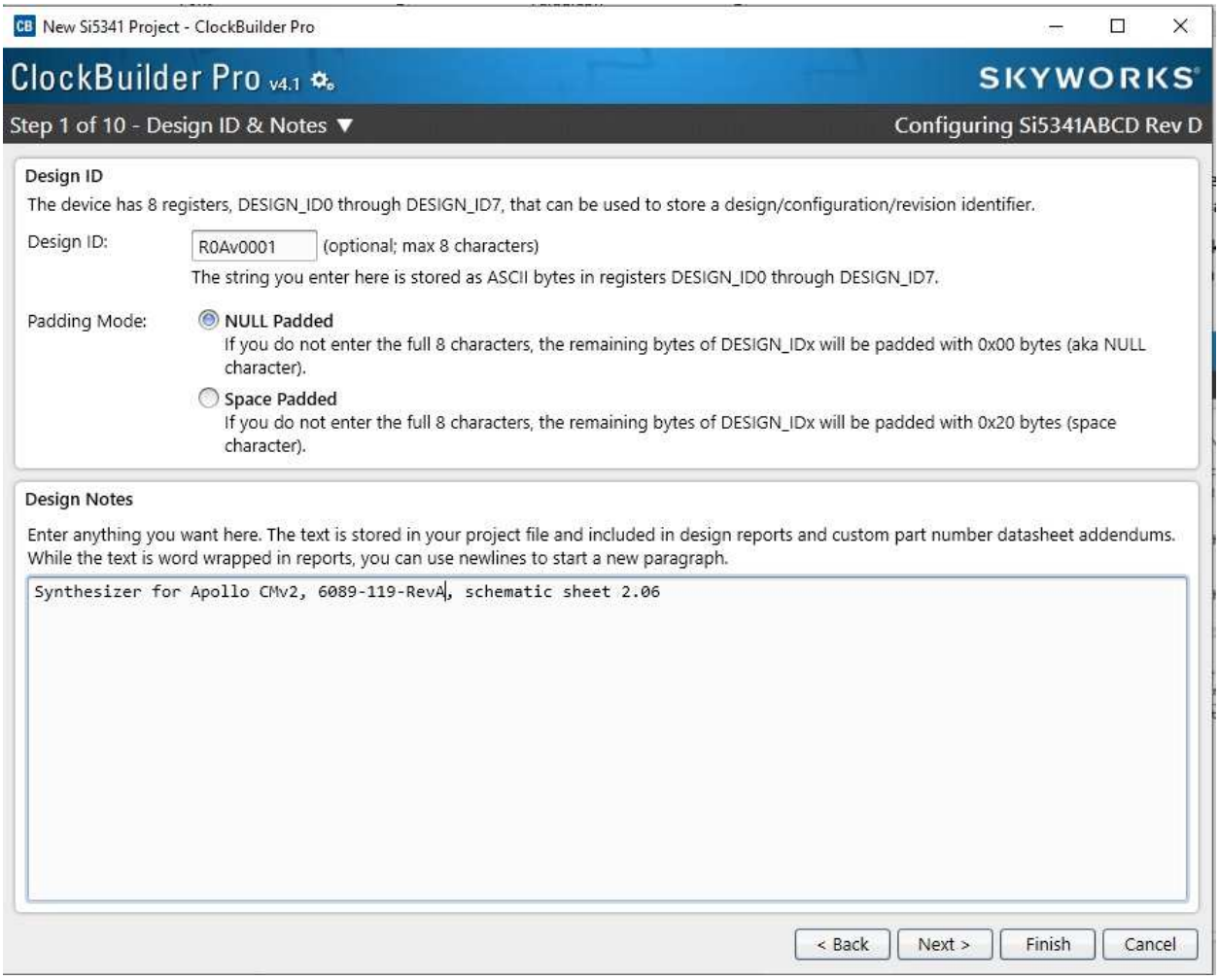
Instructions for using Clock Builder PRO with the Apollo CMv2

This is initially written for an SI5341. Differences for an SI5395 will be accounted for later.

Step 1 – For the device ID register contents, enter the 8-character string consisting of the synthesizer name [one of R0A, R0B, R1A, R1B, R1C], followed by a lowercase “v”, then a 4 digit revision number [in the example “R0Av0001”]. This string will appear later in file names.

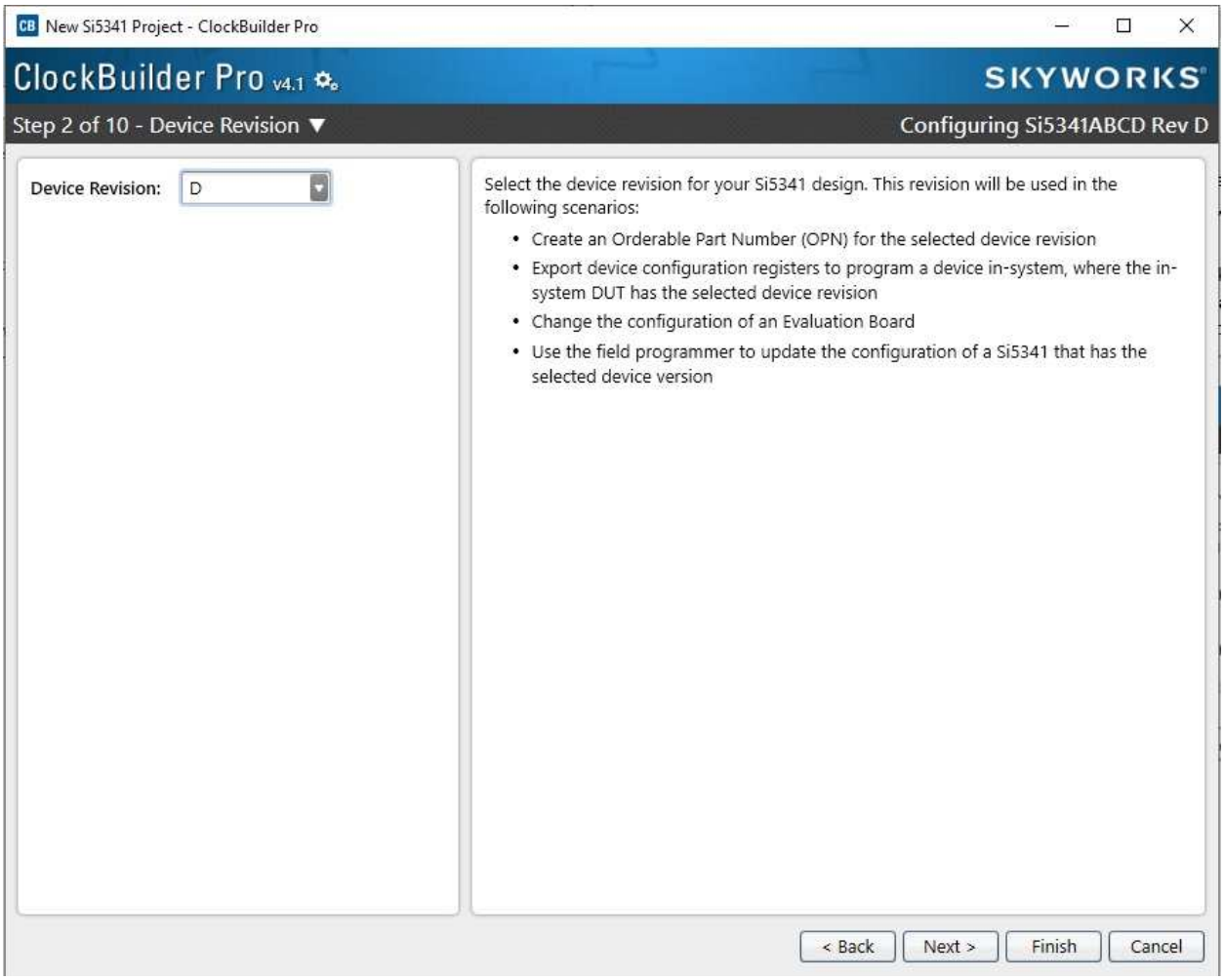
In the design notes block, enter a line like the one shown below. It should contain the schematic drawing number and revision [in the example “6089-119-RevA”] and the schematic sheet number where the associated device can be found [in the example :schematic sheet 2.06”].

Press “Next”



Step 2 – The SI5341 only comes with Device Revision “D”.

Press “Next”.

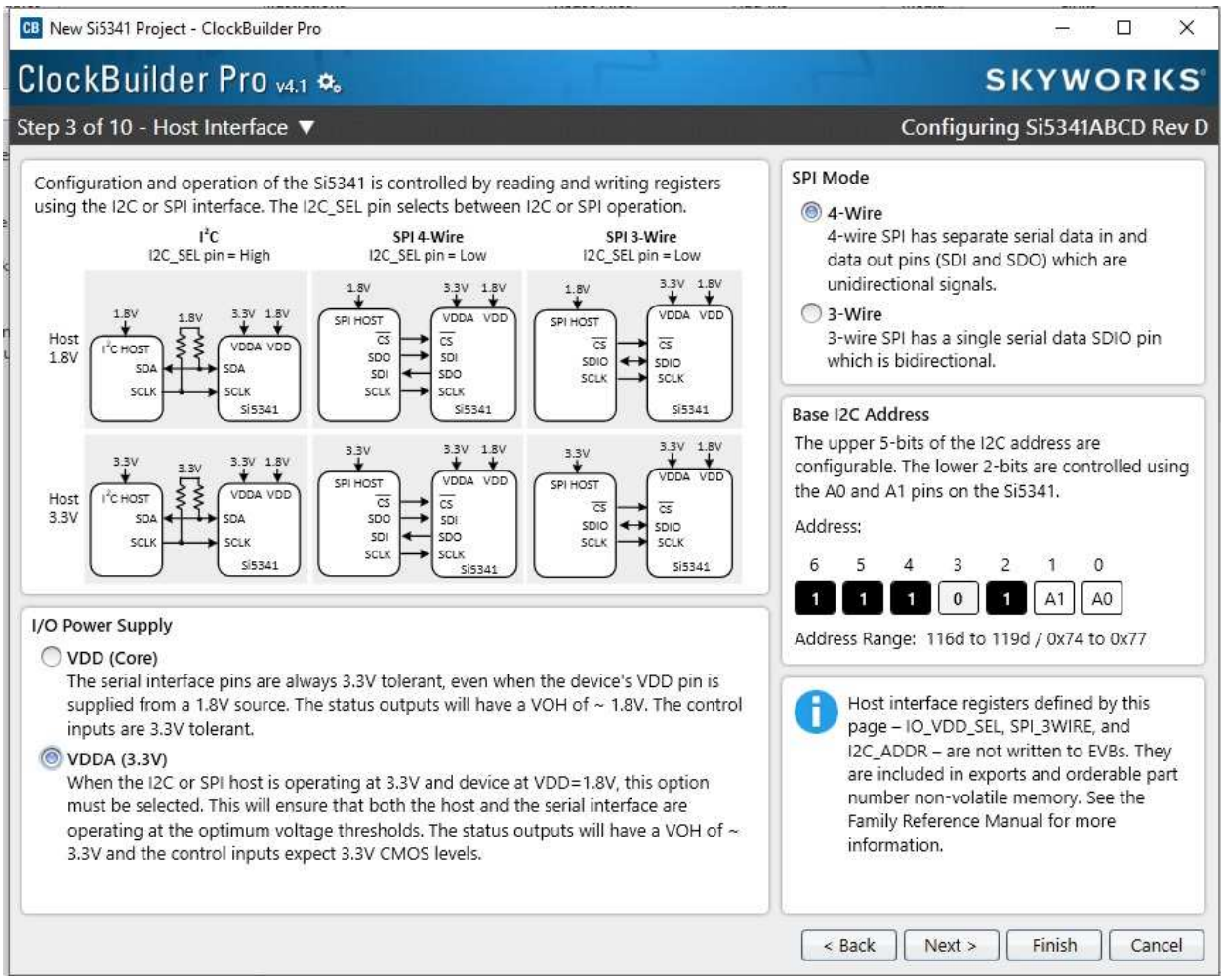


Step 3 – In the “I/O Power Supply” box, select “VDDA (3.3V)”. That is the only selection that needs to be made on this page.

The device will be run in “I2C” mode, not “SPI”. This is selected by having the “I2C_SEL” pin on the device be not connected. The internal pullup selects I2C.

The base I2C address is 0x77. It is set by configuration resistors on the board. Each of the five synthesizers is on its own single-device I2C bus, so they are all configured for address 0x77. Leave the Base I2C Address setting unchanged [1 1 1 0 1 A1 A0] for a range of 0x74 to 0x77

Press “Next”.

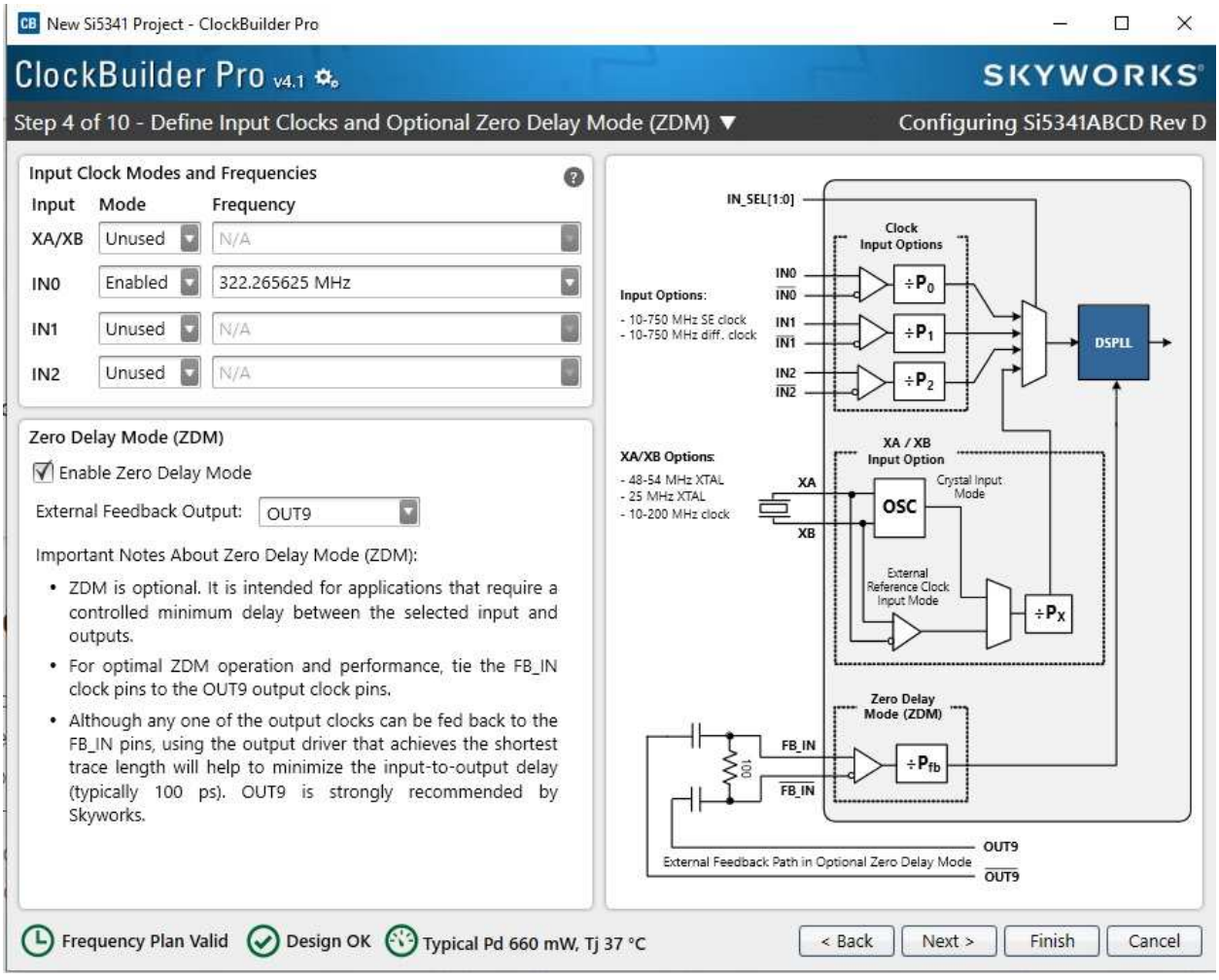


Step 4 – Generally chose one input for a design and leave all other inputs unused, even if there is a signal that is potentially available. Live switching is not usually needed. For instance, all five synthesizers have a 48 MHz crystal connected to the XA/XB pins. The R0A synthesizer also has a 322.265625 oscillator connected to the IN0 pins, and an output from the R0B synthesizer connected to the IN1 pins. This design will just use the 322.265625 MHz oscillator. In the “Mode” column for IN0 select “Enabled” from the popup window. Set the frequency to 322.265625 MHz. Be sure to indicate “MHz” in the value.

Four of the five synthesizers have wiring in place to support “Zero Delay Mode”. If it is available, and the application warrants using it, check the “Enable Zero Delay Mode” box and select the output that is connected for ZDM. ZDM is not available for the crystal input. For synthesizer R0A, it is not necessary when using the oscillator on IN_0, but may be desirable when using the R0B output on IN_1.

If you wanted to use the crystal instead, then select “Crystal Mode” for the XA/XB input and disable the other inputs. Disable ZDM as well, otherwise an error will be indicated.

Press “Next”.

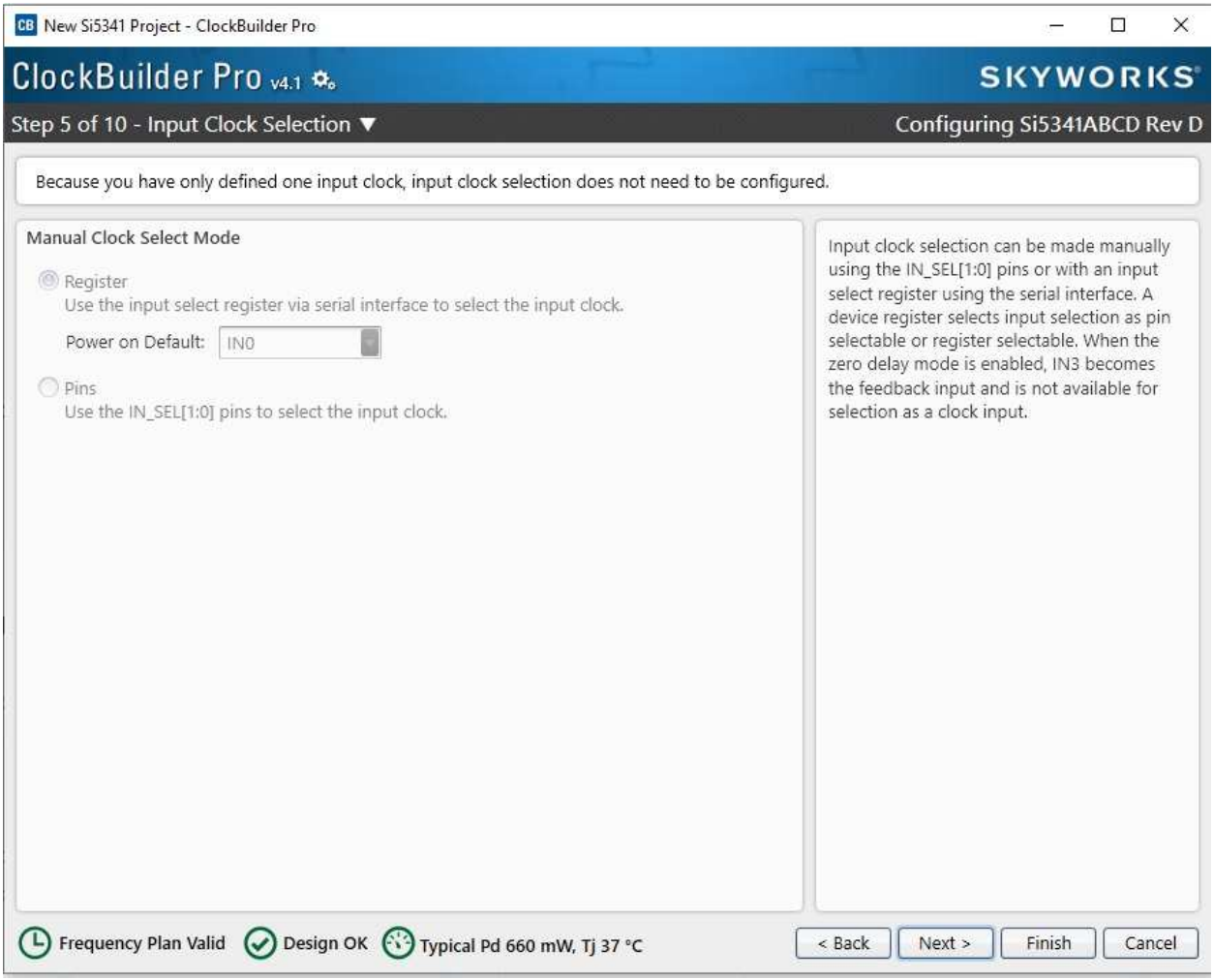


Step 5 – For this design, only one input clock is defined and input clock selection is not used.

If input clock selection is needed, the board is designed to support using the “IN_SEL[1:0]” pins. These pins are driven from I2C registers that appear on schematic sheet 4.03. These registers will power-up with all zeroes on the outputs, so the power-up selection will be the 48 MHz crystal (even if it not enabled).

The input clock selection can also be done from a register.

Press “Next”.



Step 6 – Enable the outputs that are both connected and will be used by selecting “Enabled (Powered-up with Output Enabled)” in the “Mode” column.

For boards with only FPGA#1 populated, outputs to FPGA#2 should be left disabled. This will both save power and minimize electrical noise. The 100 ohm terminators are internal to the FPGA and are not present if the FPGA is not installed, so the signal will reflect and ring on the PCB traces.

For boards with only FPGA#1 populated and no jumper board in the FPGA#2 site, outputs to the right side of FPGA#1 should be left disabled. This will save power. [Verify that this is true for TCDS signals.]

Some outputs connect to other synthesizers and will only be used under special circumstances. Leave them disabled until needed. This will save power.

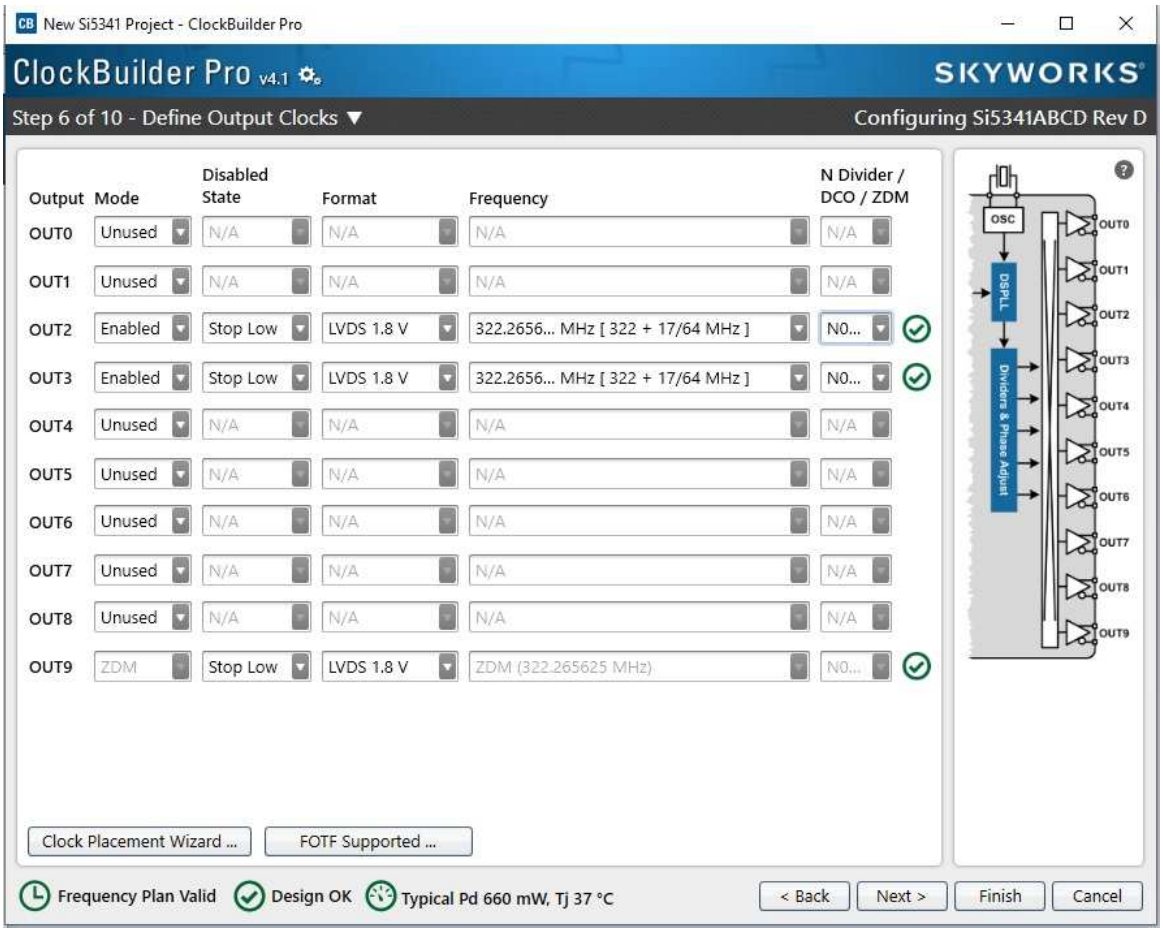
The “Disabled State” for each output can be left at “Stop Low”.

The “Format” for each output should be set to “LVDS 1.8 V”.

Enter the desired frequency for each enabled output. Be sure to indicate “MHz” in the value.

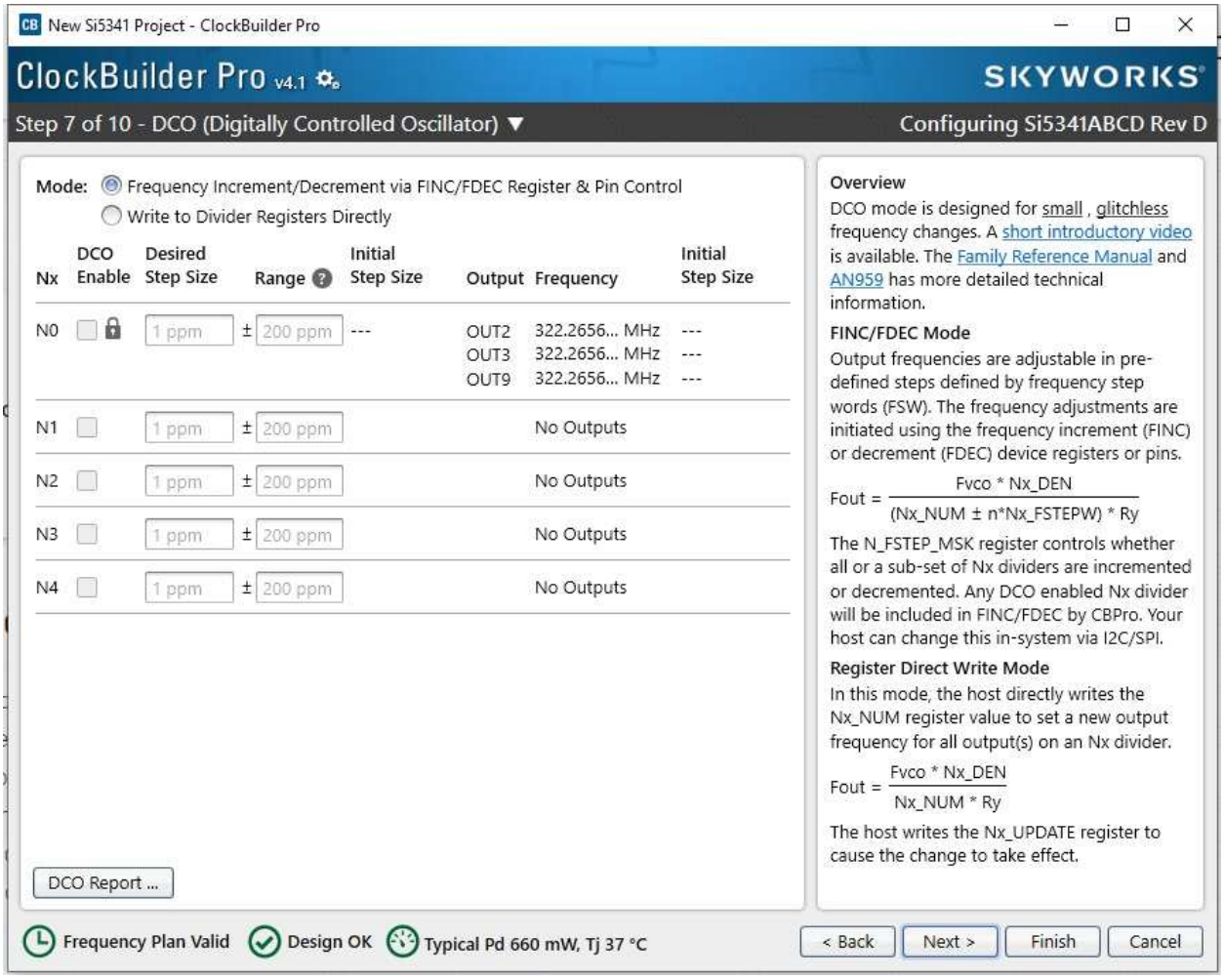
If ZDM is being used, pick the “N0” manual assignment in the “N Divider/DCO/ZDM” column.

Press “Next”.



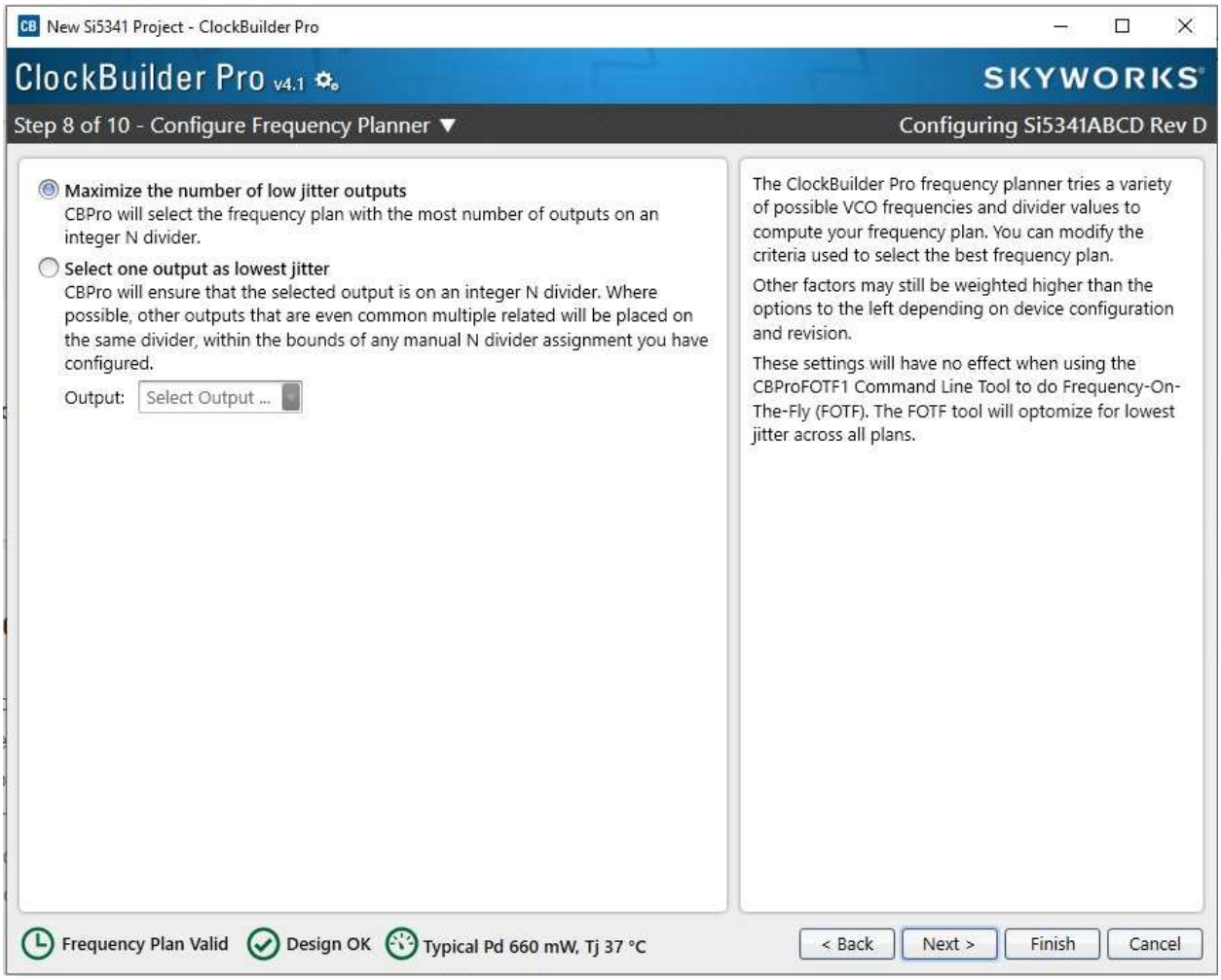
Step 7 – This project does not use “DCO Mode”

Press “Next”.



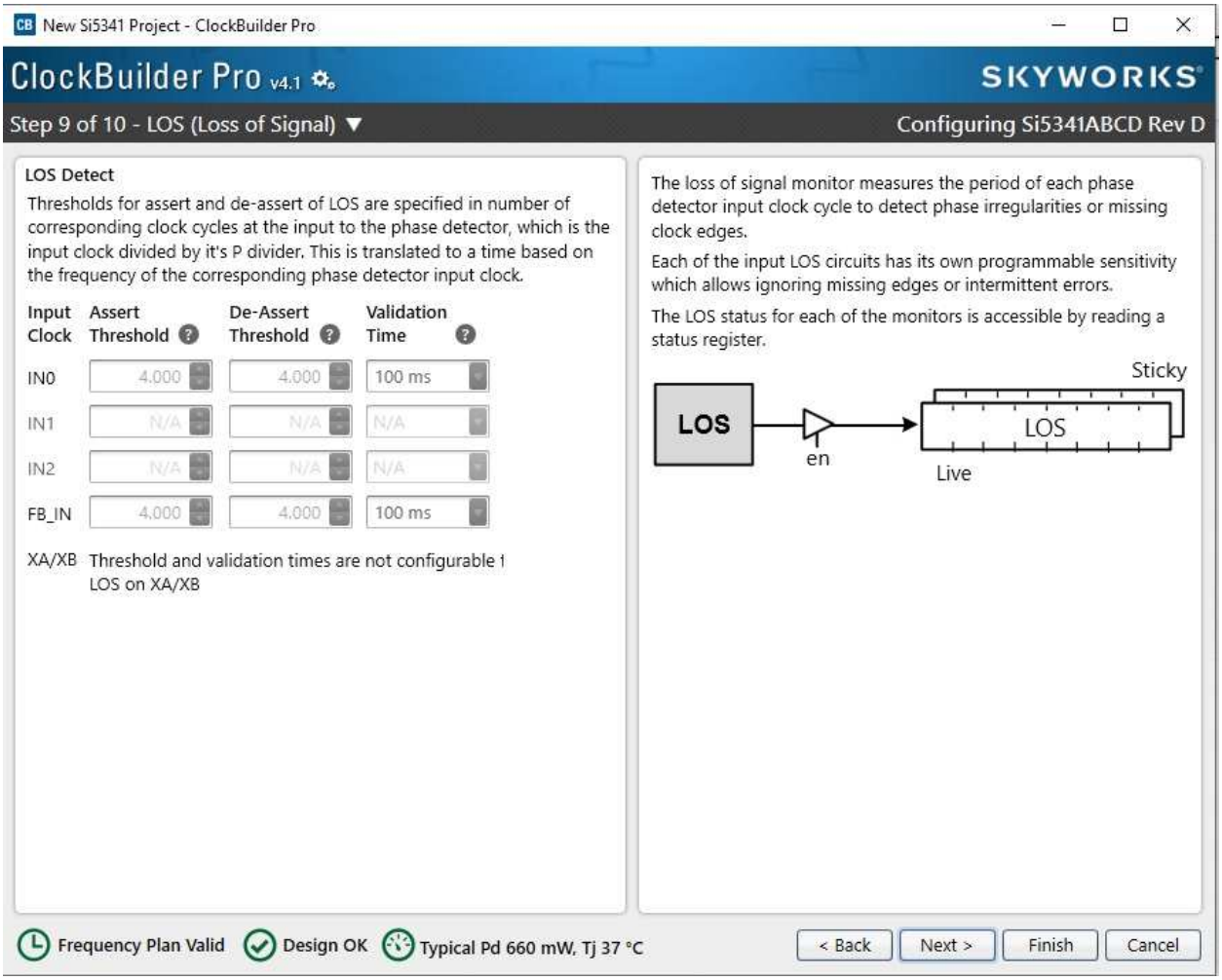
Step 8 – Unless otherwise known, the frequency planner should be instructed to maximize the number of low jitter outputs.

Press “Next”.



Step 9 – Unless otherwise known, the LOS (Loss Of Signal) values should be left as is.

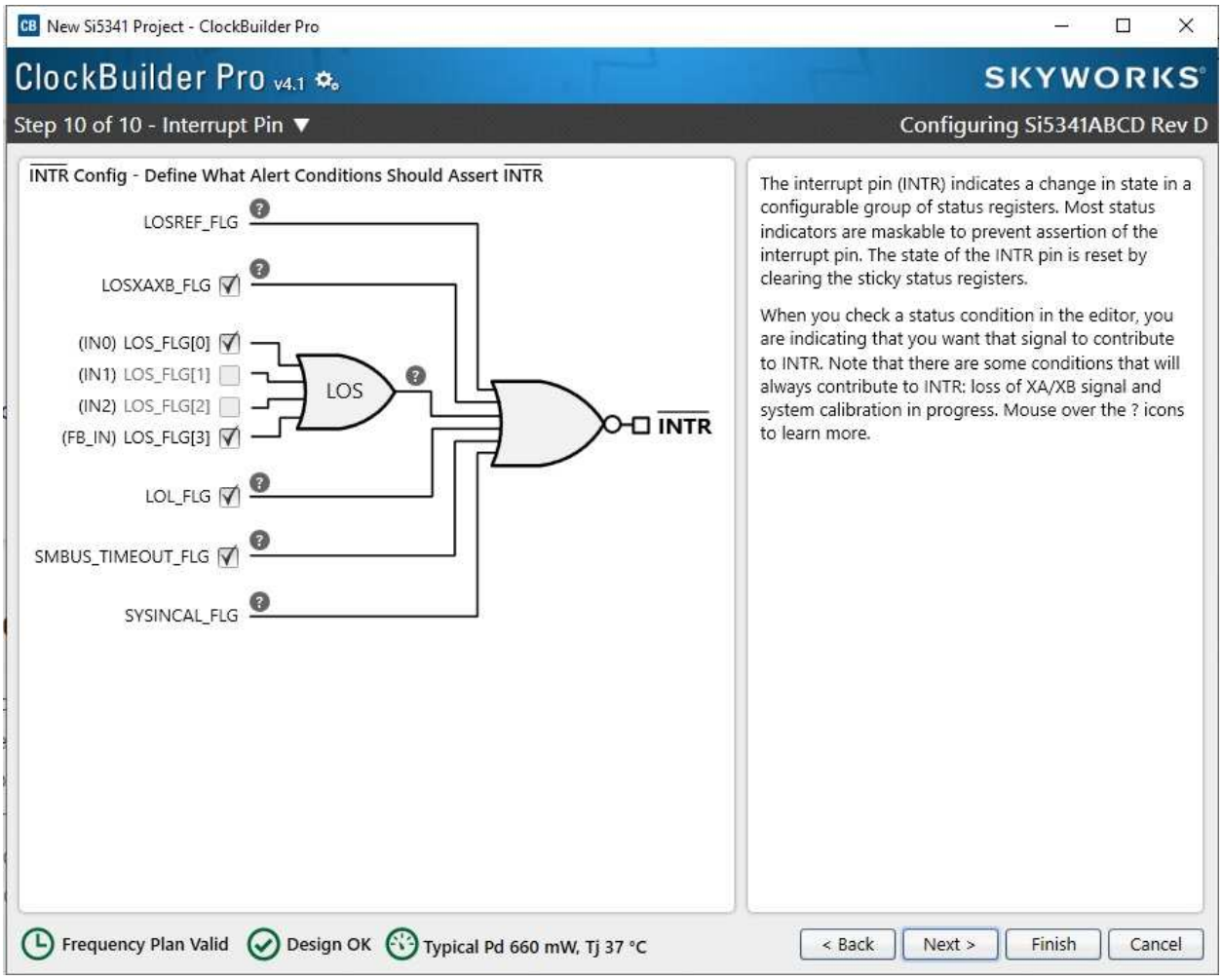
Press “Next”.



Step 10 – The interrupt output pin is connected to an I2C registers that appear on schematic sheet 4.03. It will not actually generate an interrupt. It will be available for polling by the MCU.

Use the default settings unless something different is desired.

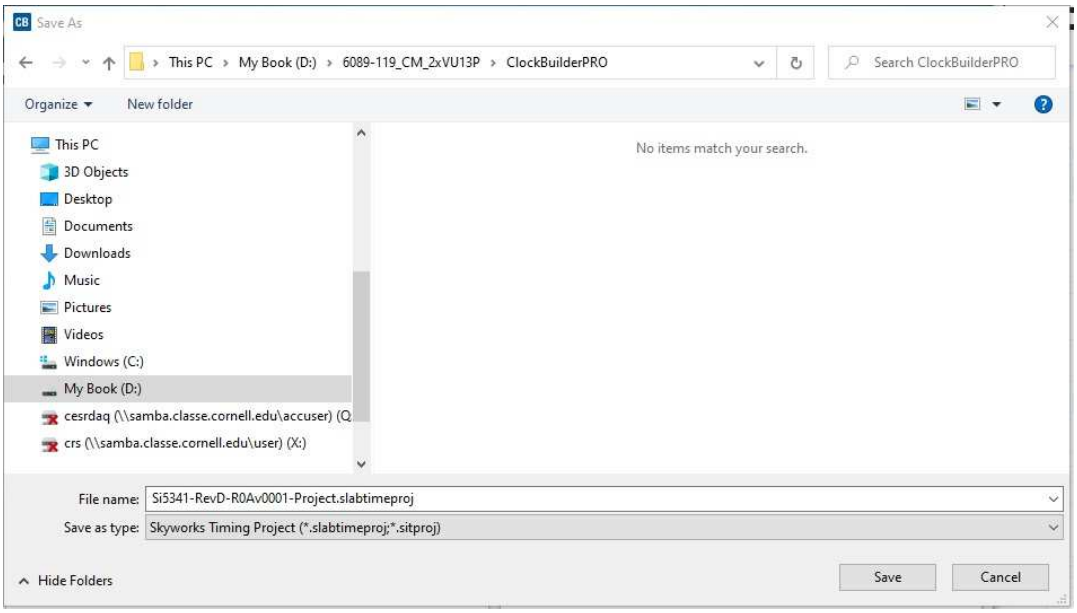
Press “Next”.

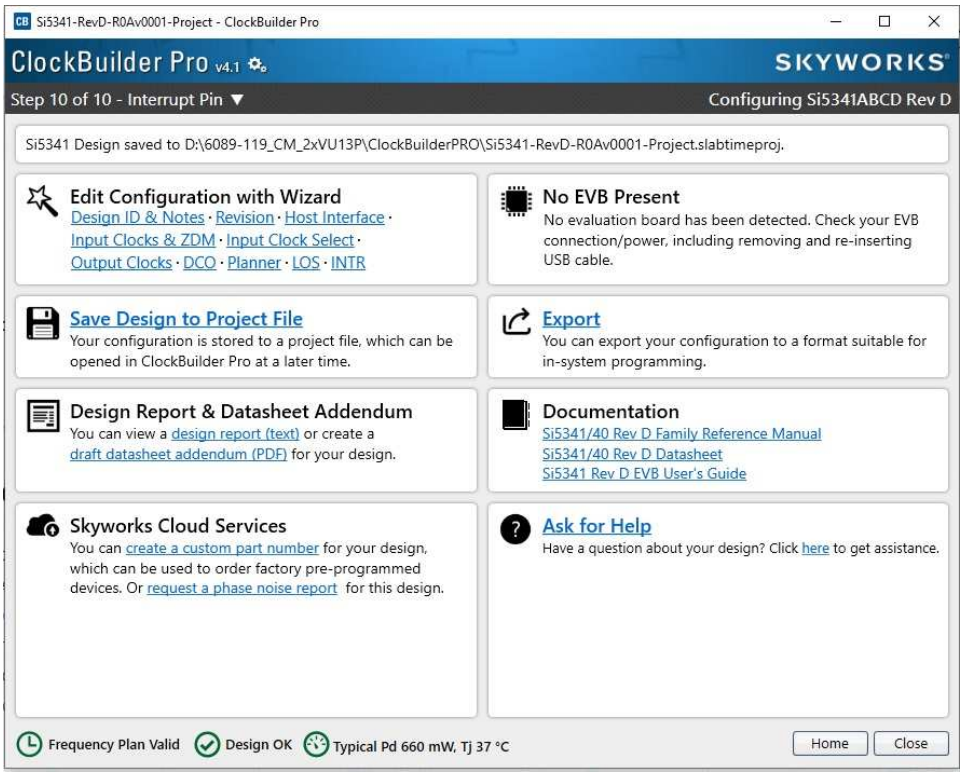


Step 11 - The design dashboard will appear after step 10.

After making any changes, save the design to a project file. The name of the project file should reflect the “Design ID” entered in step 1.

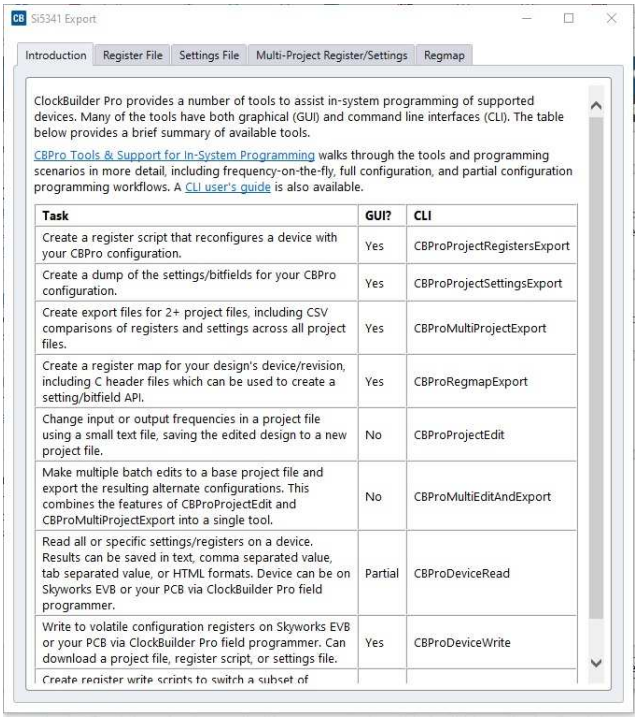
Press “Save”. The design dashboard will reappear with a confirmation message.





Step 12 – Export a configuration file that can be downloaded to the hardware. Select “Export” on the dashboard.

In the popup “export” window, select the “Register File” tab.



Under the register file tab, select “C Code Header File”. Check the box for “Include pre- and post-…”. Press “Save to File”. It will be saved as a “.h” file.