

1 Introduction

This document explains how to build and run the CMX940 Spur Avoidance Calculator Project, using the CML EC0003 “C” Development Environment and a PE0003 Universal Interface Card.

The Spur Avoidance Calculator implements a search algorithm written in “C” that determines a register configuration for the CMX940 which will produce the desired RF frequency while avoiding any unwanted spurs in the output.

If necessary, the spur avoidance calculator can also apply the register settings directly to an EV9400, CMX940 Evaluation Board if it is connected to the PE0003; thereby enabling rapid analysis of the generated signal.

The same functionality is also built into the ES9400xx GUI application that is available with the EV9400. This Spur Avoidance Calculator demonstration therefore compliments the ES9400xx GUI functionality whilst providing details of the algorithm itself.

2 History

Version	Changes	Date
1.0	First published	April 2022

3 Hardware

PE0003 Universal Interface Card
EV9400 CMX940 Evaluation Board
Power supplies for above
PC with USB connection
Spectrum Analyser (to view output signal)

4 Software

4.1 Project setup using the NXP LPCXpresso IDE

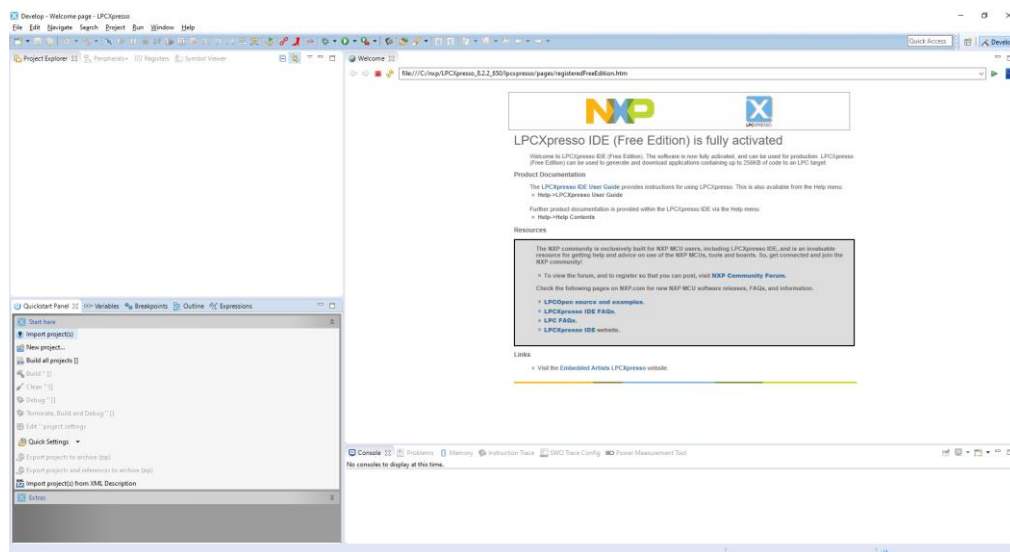
- 1) Install the NXP LPCXpresso Integrated Development Environment. This software IDE may be downloaded from NXP (www.nxp.com). Following registration, a free limited license edition becomes available which is sufficient for building the Spur Avoidance Calculator Project.

This project was built using LPCXpresso version 7.9.2.493, the use of version 8.x or higher is not recommended for use with the current PE0003 software tools.

- 2) Download and run the EC0003 “C” development environment install package from the CML website at <https://www.cmlmicro.com/products/pe0003-universal-interface-card/> under the “Knowledge Base” tab.

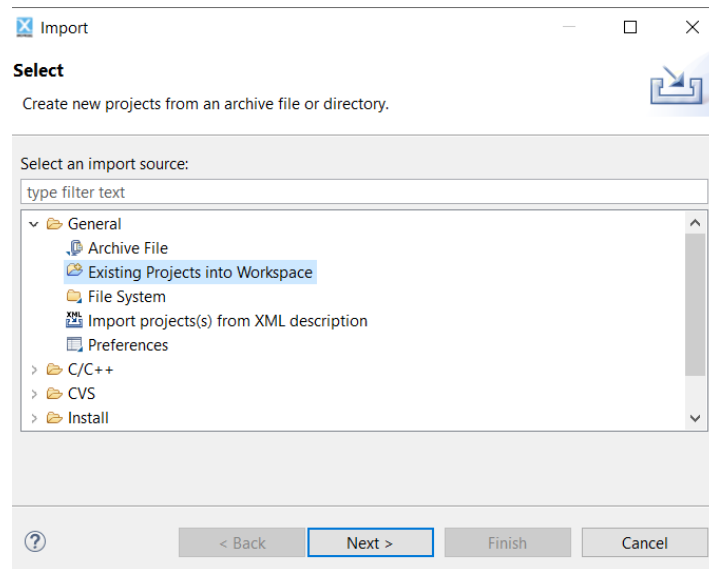
Additional EC0003 setup instructions may also be found with the EC0003 documentation, once downloaded.

- 3) In LPCXpresso: create or open a clean workspace (File, Switch Workspace)

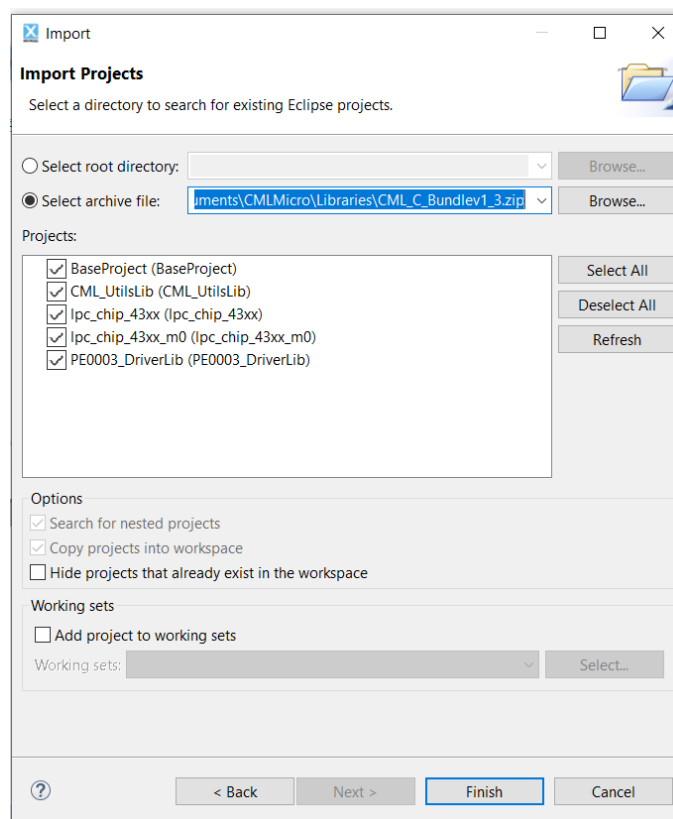


- 4) Import the CML Projects from the EC0003 package.

File, Import, Expand General, Select “Existing Project Into Workspace”, left click Next



- 5) Choose “Select archive file” and browse to the CML C bundle library. This should have been installed as part of the EC0003 package, usually under “My Documents\CMLMicro\libraries”.



- 6) With all Project boxes checked, click Finish – this will now create a project structure in the workspace.
 7) In the project explorer (default left side panel) locate the top level of “BaseProject”.
 8) Right click on this and left click “Build Project”.
 9) If an error occurs on the first attempt to build select Build Project once more, as this usually clears any errors generated from the first attempt.
 10) A successful build will result in the creation of a binary output file which will be located here: “workspace_dir\BaseProject\debug\BaseProject.bin”.

- 11) If not done already, connect a PE0003 via USB to the PC and apply the necessary 5V supply.
- 12) Click on and open the EC0003 GUI. There should be a shortcut available on the desktop that was placed there during installation with the CML Logo and red letter "C"
- 13) On the EC0003 GUI click the Browse button and locate the "BaseProject.bin" file produced above.
- 14) Double left click the binary file or alternatively select it and left click open.
- 15) This will then load the image and should produce the text "Hello from PE0003", confirming that the build environment is working.
- 16) Extract the Spur Avoidance Calculator source files into the folder "workspace_name_here\BaseProject\src". The pre-existing BaseProject.c should be overwritten.

The new source files are:

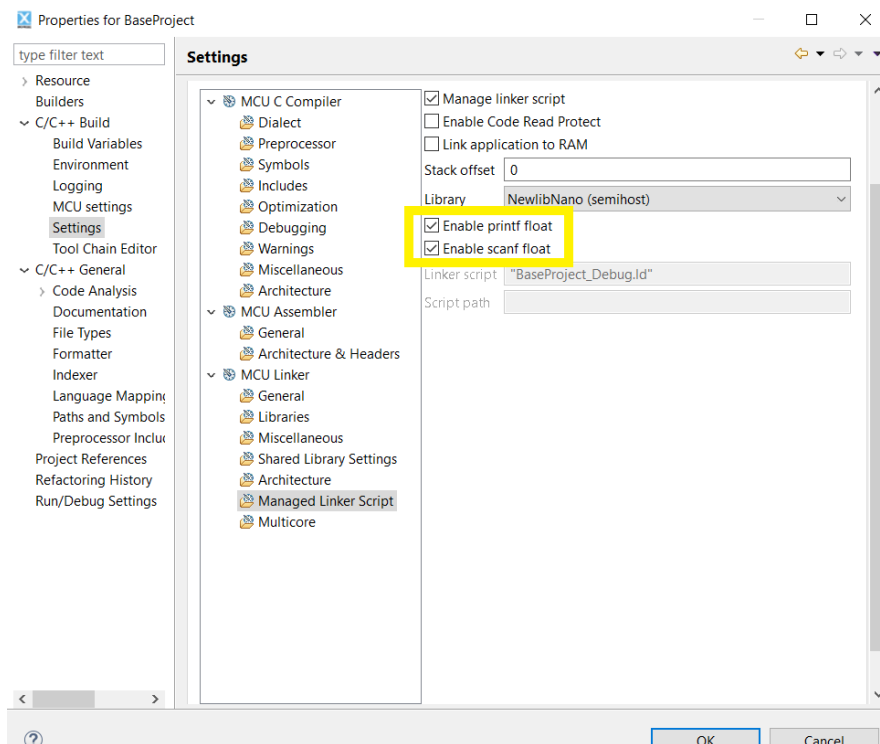
"spurcalc.c"

"spurcalc.h"

"registers_cmx940.h"

"BaseProject.c" – The new updated version.

- 17) Return to LPCXpresso and right click on the BaseProject again, and then left click Refresh if the files are not visible under the "src" folder in the project.
- 18) Some project properties need to be modified for correct functioning of the code.
 - a. Right click on "BaseProject" and left click "Properties" (at the bottom of the menu).
 - b. Expand "C/C++ Build" in the menu to the left, and left click "Settings"
 - c. In the new page that opens expand "MCU Linker" and select "Managed Linker Script".
 - d. In the right-hand panel check the boxes for "Enable printf float" and "Enable scanf float", left click OK to apply the new options.

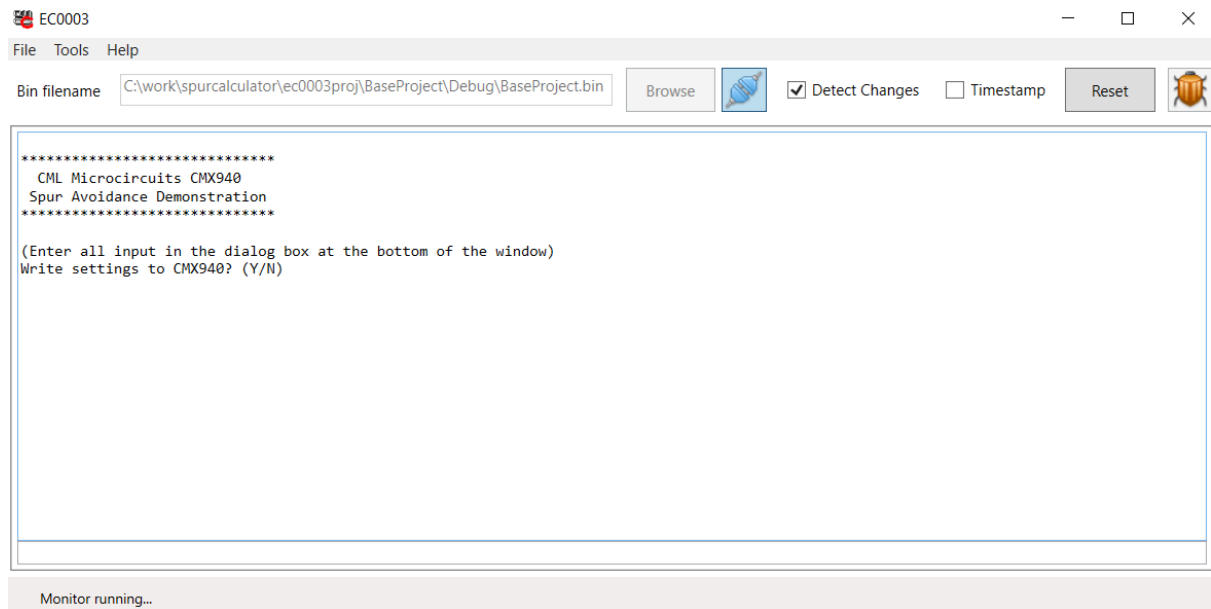


- 19) Right click on BaseProject again and left click "Build Project".
- 20) This will replace the "Hello from PE0003" binary, with the newly built spur avoidance calculator binary.

- 21) Returning to the EC0003 GUI the new binary may have been loaded automatically, otherwise left click Browse and left click “BaseProject.bin” to reload it.

4.2 Operating the image

Upon loading the newly built CMX940 Spur Avoidance Calculator binary image in the EC0003 GUI the user is presented with the following screen



The calculator may be used to produce a set of register values that may be applied later, or to write to and configure the attached EV9400 Evaluation Board.

All user input is entered via the text box near the bottom of the GUI window.

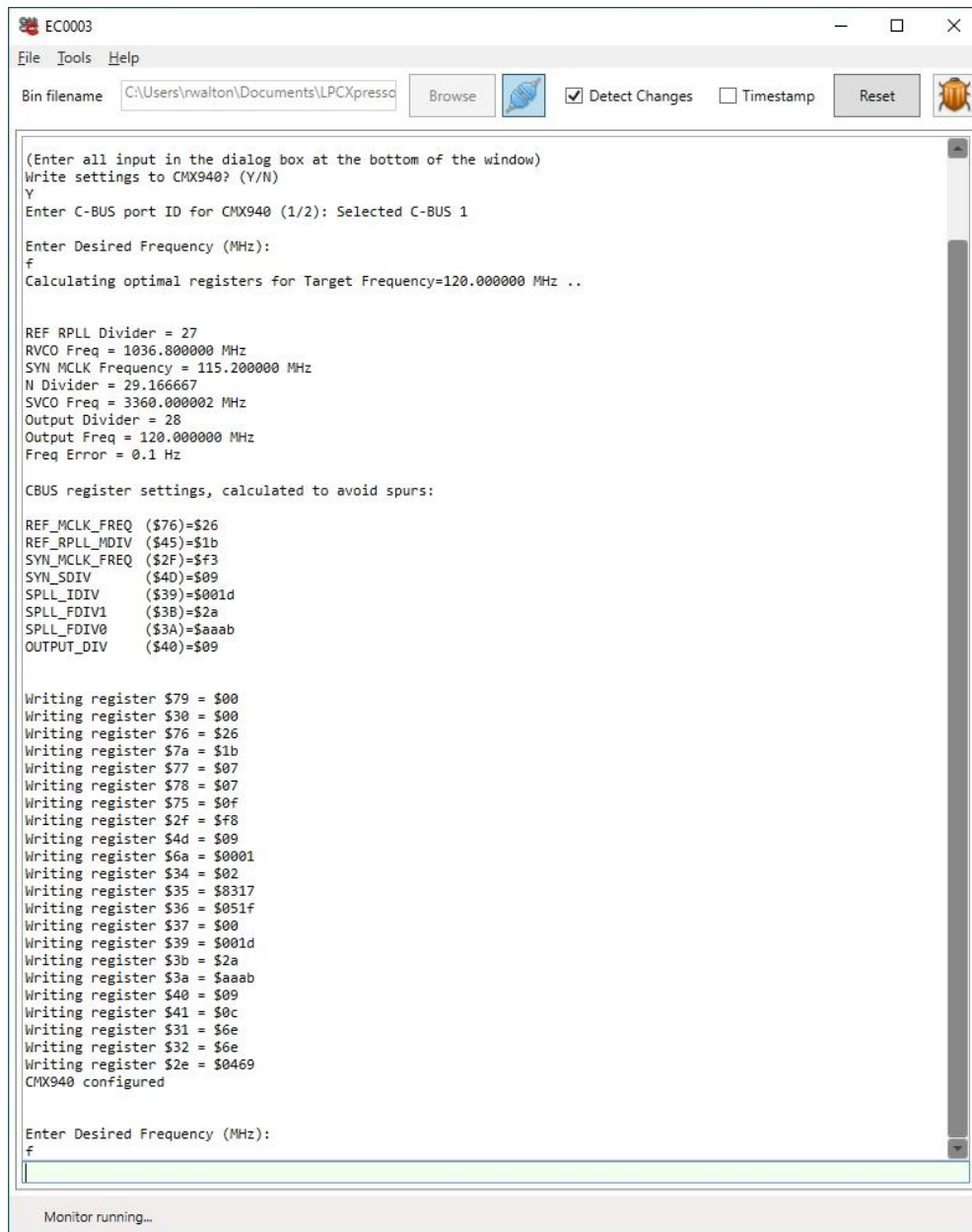
If “Y” is entered to the first question (“Write Settings to CMX940? (Y/N)”) the values will be written to a connected device via the C-BUS serial interface, otherwise the suggested register settings will just be displayed in the output window.

If the previous selection was Y, the user will be asked to provide the PE0003 C-BUS interface which is connected to the EV9400 (“Enter C-BUS port ID for CMX940 (1/2)”).

Next enter the desired output frequency that the script should produce at the output. This is expressed as a floating-point number in MHz, (“Enter Desired Frequency (MHz): f”).

The following output will be displayed, and a basic configuration will be applied to the EV9400 if enabled.

Note: Some of the other register values which are applied to configure the CMX940 may still require further optimisations for other characteristics, for example phase noise.



If the EV9400 is being configured the full list of registers being written is also displayed for information. This includes the registers included in the calculator output, but also a minimum configuration to allow the output signal to be generated by the CMX940.

5 Other use

If the spur avoidance calculator functionality is required in a new project only the source files `spurcalc.c` and `spurcalc.h` are required to implement the calculator itself. The “baseproject.c” source example may be used as an example for how to call the function, or follow the steps below:

- 1) Declare an instance of the structure “tSpurCalcIO”.
- 2) Populate the following fields in the tSpurCalcIO structure:

Variable name	Description	Type
freq_target	Desired output freq (MHz)	double (input)
ref_in	On board reference (MHz)	double (input) = 38.4 (See Note)
fdiv_16bit_mode	Fractional resolution mode (0 = 24 bit, 1 = 16 bit)	int16_t (input)

Note: this algorithm has only been evaluated using a reference clock of 38.4 MHz, the default reference clock on the EV9400. The input is passed into the code in this structure to possibly allow for further developments. On many microcontrollers it may be necessary to compute this value to produce an accurate representation in double precision format. A constant value may only be a single precision float, limiting accuracy in the calculated clock values.

e.g.

```
double ref_in = (double) 384 / (double) 10;
```

- 3) Call `SpurCalc()`, passing the address of this structure as the lone argument.
- 4) The spur calculator avoidance algorithm will populate the settings in the remainder of the structure with values for the clocks and divider registers to produce “freq_target” MHz. These are then able to be written to the CMX940 C-BUS registers.

Variable name	Description	Type	CMX940 C-BUS address
reg_REF_MCLK_FREQ	Ref Main Clock Freq	uint8_t	\$76 (8 bits)
reg_REF_PLL_MDIV	Ref PLL Main Divider	uint8_t	\$7A (8 bits)
reg_SYN_MCLK_FREQ	Synth Main Clock Freq	uint8_t	\$2F (8 bits)
reg_SYN_SDIV	Synth Source Divider	uint8_t	\$4D (8 bits)
reg_SPLL_IDIV	Integer divider value	uint16_t	\$39 (16 bits)
reg_SPLL_FDIV0	Lower 16 bits for 24 bit, or 16 bit fractional divider value	uint16_t	\$3A (16 bits)
reg_SPLL_FDIV1	Upper 8 bits for 24 bit fractional mode	uint8_t	\$3B (8 bits)
reg_OUTPUT_DIV	Output divider setting	uint8_t	\$40 (8 bits)

(In addition to the registers above the SPLL_CON register (\$34) must be set to match the requested divider mode)

- 5) Optionally – the function “GetClocksInfo()” can produce additional detail of the various clocks within the system. This is not required for configuration and is provided for feedback only. Note: This function expects the primary structure to have been fully populated by calling “SpurCalc()” beforehand.
- 6) The code is written in floating point C so the target must be capable of processing (or emulating) floating point operations. In the BaseProject/LPCXpresso example a software emulation of floating point (Soft-ABI) is used to demonstrate that the hardware FPU is not required, but it might be more efficient to use a hardware FPU if it is available.

CML does not assume any responsibility for the use of any algorithms, methods or circuitry described. No IPR or circuit patent licenses are implied. CML reserves the right at any time without notice to change the said algorithms, methods and circuitry and this product specification. CML has a policy of testing every product shipped using calibrated test equipment to ensure compliance with this product specification. Specific testing of all circuit parameters is not necessarily performed.



CML
Microcircuits

United Kingdom	p: +44 (0) 1621 875500	e: sales@cmlmicro.com techsupport@cmlmicro.com
Singapore	p: +65 62888129	e: sg.sales@cmlmicro.com sg.techsupport@cmlmicro.com
United States	p: +1 336 744 5050	e: us.sales@cmlmicro.com us.techsupport@cmlmicro.com

www.cmlmicro.com