

C4SX Environment, Version 03
Document Version 03d
09/05/2007
Bob Senser

Overview

C4SX is an extremely low-cost/no-cost environment for running C code on the SX, supporting both the SX28 and the SX48 devices. It uses the editor of your choice (we suggest several), an SX C compiler called 'CC1B', the Parallax SX-Key software and hardware, and some specialized utility programs and C source modules. C4SX is a grassroots project, born out of necessity, facilitating the use C code on the SX chips, via the Parallax SX-Key.

This document describes the C4SX environment and how to install and use it. All of the software components can be downloaded at no cost. Some of these components are free software, others are open source and the CC1B compiler itself is available as a downloadable beta.

This document's first major section is a Quick Start that helps you quickly get C4SX installed and running. It shows how to run a small C sample program, which blinks an LED attached to port B, pin 0. The remainder of this document covers more details about the various components of C4SX, and covers some options and information you might find useful.

Feedback is important. The preferred method for providing C4SX feedback is via the Parallax SX Forum. You can also contact the C4SX coordinator at c4sx@rwsenser.com. If you are thinking of using the CC1B compiler, one of the main C4SX components, for more than one commercial product then you should contact sales@bkndcom (see the CC1B free beta readme.txt for details).

IMPORTANT: Upgrading from C4SX Version 01 or 02

Please install the latest version of CC1B compiler, version 0.7A or better. This new CC1B version contains important fixes. If you have already installed C4SX V01 or V02 then please rename your current CC1B directory to CC1Bv01 or CC1Bv02 before you install this new version of C4SX. Once you have installed this new version, you can manually copy your C programs from your old directory to the new CC1B directory. These steps are necessary because the C4SX.bat script, the sample programs and the CC1B2SX utility/reformatter have changed in subtle ways and conflicts will arise if the two versions are merged. You might need to make some minor changes to the C programs you have written. See the "Changes in C4SX Version 03" Section in the Appendix.

IMPORTANT: Known CC1B Bugs

All the bugs found in C4SX V01 have been fixed by the CC1B author in CC1B 0.7A. In addition, CC1B 0.7A (or better) has some new, handy features. Please see the "Known CC1B Bugs" section in the Appendix for a list of the currently known bugs.

Quick Start

Here are the steps to quickly install C4SX and run a sample SX C program.

Needed Software

1. You will need an editor. We suggest installing "notepadpp", noted as "notepad++" in the remainder of this document. It is a simple editor with a C "styling mode". It presents the C module being edited and a list of the functions in the C module. It is not visually "cluttered" like many IDE editors. Any text-oriented editor will also work. If you are a relatively new programmer and choose to use the standard Windows "notepad" editor, be sure that your C program files end up with a file extension of ".c" and not ".txt". The notepad++ download URL is: <http://notepad-plus.sourceforge.net/uk/about.php>. See the Appendix for some additional notepad++ information. Other good editors and IDEs include Microsoft VC, Eclipse and Code::Blocks.
2. You will need the CC1B compiler, version 0.7A or newer. Download it from <http://www.bknd.com/cc1b/index.shtml>. There are many CC1B editions that can be downloaded. For this Quick Start, we suggest using the "cc1bfree.exe" edition because it comes with an installer.
3. You will need the Parallax SX-Key software. Download it from: <http://www.parallax.com/sx/downloads.asp>. We are currently using the SX-Key 3.2.3 version. Other versions may also work.
4. You will need the C4SX utilities, documents, sample code and SXIO.H headers. These can be downloaded from: <http://www.rwsenser.com/c4sx/C4SXv03.zip>.

Installations

1. If you have not already done so, install the editor of your choice. If you are unsure what to do, install Notepad++.
2. Install CC1B. Using the 'cc1bfree.exe' install is the easiest way to get CC1B going. Word to the wise: If you allow the install to put CC1B in the default "C:\Program Files\bknd\CC1B" folder then you will save yourself some work later in this Quick Start.
3. If you have not already done so, install the SX-Key software and hardware.
4. Unzip the C4SX zip file into a directory. We strongly suggest using "C:\C4SX". If you are upgrading from an older C4SX version, be sure to rename your current "C:\C4SX" directory. After the unzip operation is done, this directory should contain at least these files: c4sx.bat, CC1B2SX.EXE, led28.src and ledc28.c. The last file is a SX C sample program, 'led28.src' is a sample SX assembler program and the other files are briefly discussed later in this document. Be sure the version of CC1B you are using matches the C4SX you are using. If you are unsure, be sure both of them are the newest versions.

Hardware Setup

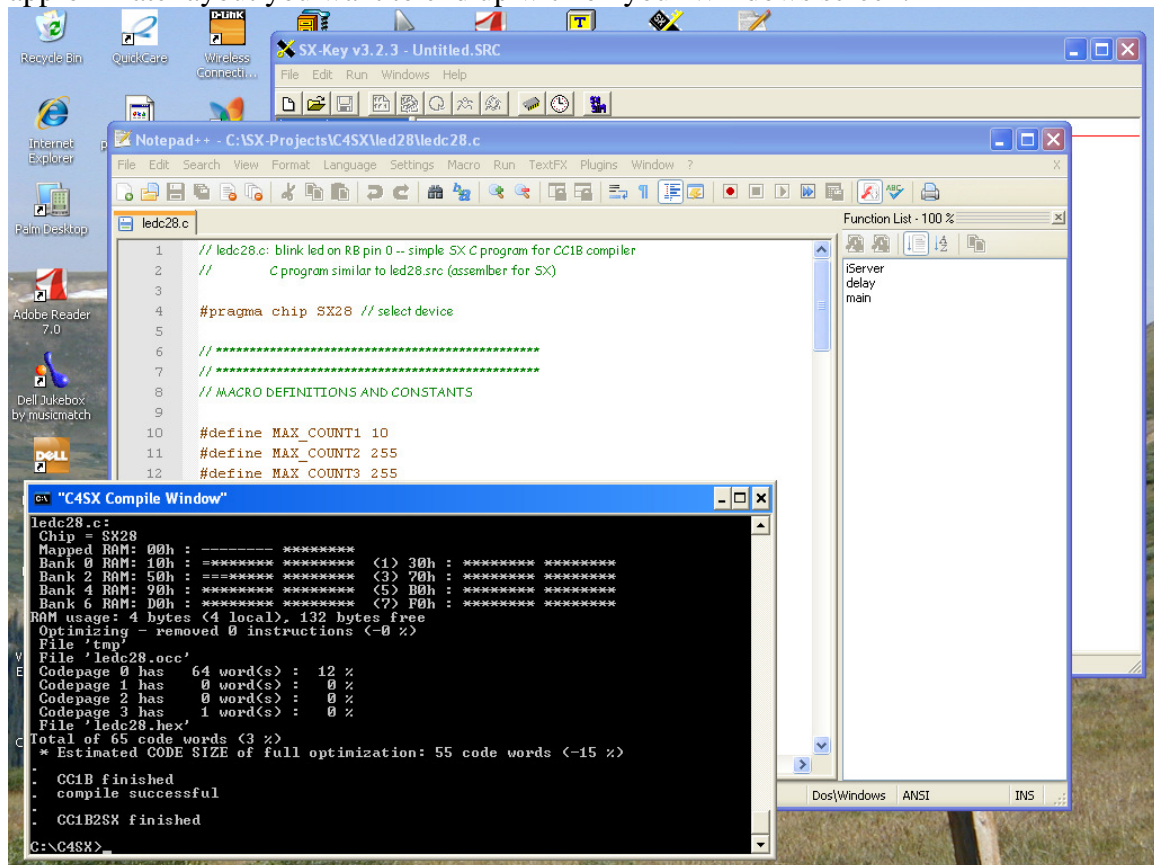
You will need an SX Tech Board and associated hardware (SX-Key, cables, etc) to run the sample C program. Here is a Tech Board picture:
http://www.parallax.com/images/prod_jpg/45205.jpg. You will want to wire your board as this picture shows, EXCEPT you will want the LED on pin B0 and NOT pin B7.

To test your SK-Key and Tech Board setup, please run the led28.src file. This is a very simple SX assembler program that will blink the LED on pin B0. Until you have your SX board's LED blinking, there is no point in proceeding with the next steps in this Quick Start!! Sorry.

Edit, C Compile and Execution

Once you know your SX configuration is correctly executing led28.src and you see the blinking LED, you are ready to proceed and try C4SX.

With the Command Window, Notepad++ and SX-Key all running, this is the approximate layout you want to end up with on your Windows screen:



The top-most window is from the Parallax SX-Key IDE. The middle window is from Notepad++ and the bottom black window is the Windows Command Window ("cmd.exe").

To get these windows all opened, start the SX-Key IDE and Notepad++ (or your chosen editor) from your Windows program menu or desktop. The Windows Command Window can be started from "All Programs | Accessories | Command Prompt" or by entering "cmd.exe" at the Windows "Run" option. Your Command Window will be titled "C4SX Compile Window" after the c4sx bat file (script) is run the first time.

If you installed CC1B in the location mentioned earlier and put the C4SX contents in "C:\C4SX", you are ready to begin. If you put either of these in a different location, you will need to modify the c4sx.bat file (script).

The basic approach used is to edit the C program in the editor (notepad++ in this example), to compile the program and automatically reformat it with the c4sx.bat script and then to run the program with the normal SX-KEY IDE. The c4sx.bat also has some special options, which are beyond the scope of this Quick Start (see the comments in c4sx.bat for details). Here we go:

Load the ledc28.c program into you editor. Please don't change it. Once you have looked at it, then go to the Command Window and enter:

```
cd c:\c4sx
```

This will put you in the C4SX directory. From here, enter:

```
c4sx ledc28
```

Do not put a ".c" and the end of the program name. If all has gone well, you will see something like this appear:

```
C:\C4SX>c4sx ledc28
CC1B Ver 0.6I beta, Copyright (c) B Knudsen Data, Norway 2002-2007
--> SMALL edition, Ubicom SX, 4k code, reduced optim.
ledc28.c:
Chip = SX28
Mapped RAM: 00h : ----- *****
Bank 0 RAM: 10h : ===== (1) 30h : =====
Bank 2 RAM: 50h : ===== (3) 70h : =====
Bank 4 RAM: 90h : ===== (5) B0h : =====
Bank 6 RAM: D0h : ===== (7) F0h : =====
RAM usage: 4 bytes (4 local), 132 bytes free
Optimizing - removed 0 instructions (-0 %)
File 'tmp'
File 'ledc28.occ'
Codepage 0 has 64 word(s) : 12 %
Codepage 1 has 0 word(s) : 0 %
Codepage 2 has 0 word(s) : 0 %
Codepage 3 has 1 word(s) : 0 %
File 'ledc28.hex'
Total of 65 code words (3 %)
* Estimated CODE SIZE of full optimization: 55 code words (-15 %)
.
. CC1B finished
. compile successful
.
. CC1B2SX finished
```

The "compile successful" and "CC1B2SX finished" output lines mean that everything has worked successfully.

The next step is to load the generated ledc28.src file into the SX-Key and to execute it. Please load this now and then pause a minute to look at the source code. The assembler source code will have the C code included as comments and the assembler code, generated by CC1B, is there ready to run. At this point, with your SX Tech Board attached and powered up, run or debug the program in the SX-Key IDE. When you run the program, you should see the LED blinking.

If you want to experiment, you can go back into the editor and change the timing constants for the C delay() function. After saving the change, rerun the c4sx script, reload the generated program into the SX-Key (it will ask to do this anyway) and then re-execute.

If your SX setup has serial capabilities then you can run the sertstb.sxb and sertstc.c programs. These both require a terminal (or Windows Hyperterm) setup at 19,200 baud, no parity and no handshaking. Pin A2 is the serial input and pin A3 is the serial output. Please run sertstb.sxb first to verify your serial configuration and then run sertstc.c. This C program uses the sxio.h library to communicate serially. Note that this library does use the SX interrupts.

Closing note: Depending on the acceptance of C4SX within the SX community, we expect to work with Parallax to be able to access the SX-Key directly from C4SX. We are also thinking about producing a "plug in" for at least one common editor/IDE. This would make it possible to request the compile and run the program from within the editor/IDE.

User's Guide

Currently a "work in progress." but some subsections are available:

CC1B RAM usage

The SX28 and SX48, especially the SX28, have a very limited amount of RAM. In addition to being limited, not all of the RAM on these chips can be used in the same manner. The CC1B compiler makes allowances for these differences and tries to optimize the use of RAM. However, if the C programmer helps by manually placing variables and giving the compiler hints via the C #pragma statements then the memory can be used more effectively and the size of the actual SX program can be made much smaller. Section 2.2, "Defining Variables" of the CC5X Manual, obtainable from the CC1B website, covers the details.

In a nutshell, small C programs that only have a main() function, that have fewer than 5 to 8 variable and that do not use the string.h, stdlib.h, ctype.h and/or sxio.h libs likely do not need to use these special C constructs (#pragma, bank0, shrbank, etc.). Larger programs very likely will.

Components

C4SX Utility and Modules

The C4SX utility/converter is CC1B2SX. It converts the output from CC1B to an SX assembler program, styled for the SX-Key assembler. Note that the CC1B 0.7A version now produces assembler output that is SX-Key usable. The CC1B2SX utility/converter is still used to handle the optional `/*<?asm >?*/` construct. The C4SX modules consist basically of the c4sx.bat script, CC1B2SX, the sxio.h header/lib file and some sample C programs. The sxio.h lib file provides some C-like I/O functions such as `getchar()` and `putchar()`.

In addition to sxio.h, there are other libraries available. C4SX Version 03 includes three additional libraries in the base directory: `stdlib.h`, `string.h` and `ctype.h`. These libs/includes provide many of the 'normal' C string functions, among other items. These libs should be considered as 'experimental' as their implementations could change in the next release of C4SX. The directory 'pvlbs' includes other libraries made available on the Parallax SX Forum. Many thanks, to the various authors, for making these libs available.

Notepad++ Editor

As stated earlier, about any textual editor can be used with C4SX. Notepad++ seems to be a good choice. It is easy to use, makes good use of the screen and does not bring unneeded complexity into the programming process.

CC1B Compiler

This compiler is the backbone of C4SX. To date, it has done a very good job of converting C code into reasonable, optimized SX code. We hope that B. Knudson will make it a full commercial product at some point.

SX-KEY IDE and SX Programmer

What's to say? This is the SX IDE and programmer of choice for SX/B and SX assembler programs.

Troubleshooting and Known CC1B Limits

We have found the following limits with the free CC1B:

1. *Beta software limits:* The size of the resulting program is limited and some of the fancy C data types are not supported. We have never had these limits impact our work. See the CC1B readme file for details.
2. *The limit to one Commercial Product:*
The CC1B free version readme file says:
" RESTRICTIONS

The free edition can be used to generate code for all prototype and non-commercial systems without restrictions. Permission is

given to use the generated code in ONE commercial system.

No restriction applies to source code written by the user."

3. Coding limits: See the sxlimits.c program in the src\samples directory for examples of a few other limits and workarounds.

Future Plans for C4SX

1. As mentioned in the Quick Start, we hope to utilize the SX-Key directly from C4SX.
2. Also as mentioned in the Quick Start, we are considering an editor "plug in" for C4SX. Do you have strong feelings about which editor should have a C\$SX "plug in"? If so, make your feelings known!
3. More documentation and more SX C sample programs.
4. More sxio.h improvements (mainly support for more than 19.2 kb).
5. Combine the current multiple versions of CC1B SX libraries (header files) into one set.

Contributors and Credits for C4SX

Thanks to the following people and organizations:

| Item(s) | Contributor |
|---|--|
| CC1B Compiler | B. Knudsen at www.bknd.com |
| CC1B2SX utility, CCB script and VC6 IDE usage information | Bruce Ray |
| dwutil utility, sxio.h, sample C programs and updated c4sx script | Bob Senser |
| Many C libraries, and much valuable research and feedback | Peter Verkaik |

Appendix

C4SX 'Guiding Principles'

C4SX is not a large project and is being done with limited resources. Therefore the following general principles are being used to guide the project.

1. Simple and clear is better than complex, even if it is slightly less powerful.
Rationale: Things that are clear and simple are normally easy to use and require little documentation and more-limited testing.
2. Generally, only one solution to a given problem is provided.
Rationale: Extra options often lead to complexity, which requires extra documentation and extra testing.
3. C4SX should use standard C constructs whenever possible.
Rationale: Existing (embedded) C programmers should be able to understand and use C4SX without lots of documentation or retraining.
4. 'Field' changes to CC1B should not be made.
Meaning: Unless a major bug is found, which must be fixed, CC1B will be used as it is released.
Rationale: This tends to keep things simple and also makes it possible for CC1B

to be supported by its author. If we change it/enhance it, it is no longer the product the author released.

VC6 IDE Usage

Microsoft VC6 can be used with C4SX. Here is the Custom Build information:

MSVC custom build notes for using CC1B/C4SX with MSVC IDE
edit *.c file, output *.src file.
directory names will change with specific system setup.
\tmp directory used in sample is arbitrary

<file>.c source file
<file>.
Custom build

File settings:

General

Always use custom build step

Custom Build

Commands:

\cc1b\rev_I\task\cc1b -I\cc1b\rev_I\task -a\$(InputDir)\tmp
\$(InputPath)
\projectx\cc1b\cc1b2sx.exe < \$(InputDir)\tmp > \$(InputPath).src

Outputs:

nfuse.src

InputDir=\projectx\firmware\nfusion

InputPath=..\NFUSE.C

"nfuse.src" : \$(SOURCE) "\$(INTDIR)" "\$(OUTDIR)"
\cc1b\task\cc1b -I\cc1b\task -a\$(InputDir)\tmp \$(InputPath)
\projectx\cc1b\cc1b2sx.exe < \$(InputDir)\tmp > \$(InputPath).src

End Custom Build

NotePad++ Advanced Usage

NotePad++ has "plug in" support and one of these is "NppExec". From a running NotePad++ window, if you follow "Plug Ins | NppExec | Execute..." and enter "cmd.exe" in the "command(s)" window then you will be able to run C4SX from within NotePad++. Just give it the same commands as outline for the Command Window in the Quick Start.

Library: sxio.h

The sxio.h is a C serial I/O library for the SX. If you open this include file, you will see comments describing the baud rate it uses and the I/O pins. It provides several functions including standard C-like _kbhit(), getchar(), putchar(), and puts() functions. Note that this C-based function library uses SX interrupts. To use these functions, you will need a Parallax USB2SER or MAX RS-232 chip, and RS-232 device (terminal or PC with HyperTerm, etc.) attached to your SX Tech Board. See sertstc.c for a usage example.

Library: vplibs

Experimental versions of the string.h, ctype.h and stdlib.h libraries are included in the base C4SX directory. These may change internally in the next version of C4SX. In addition, the 'vplibs' directory contains additional C libraries from the Parallax SX Forum. To use the vplibs, unzip the file, rename the existing string.h, ctype.h and stdlib.h files (in your C4SX directory) and then copy the new libs from the 'vplibs' directory to your C4SX directory.

Sample C Programs

The C4SX "src" directory supplies a set of sample C programs. These programs show how to code past some CC1B limits and how the SX-Key break debugger can be used. Version 03 of SX4C includes well-tested sample programs, some that use sxio.h.

Changes in C4SX Version 03

Version 03 contains changes in these areas:

1) CC1B has been updated by the author to fix all the bugs found in C4SX Versions 01/02.

2) CC1B new feature: The "RESET main" is automatically generated. You should remove this line from your C source code.

3) CC1B new feature: The "pragma asmDir: xxxxxx" support has been added. For example:

```
#pragma asmDir:    DEVICE OSCHS3
#pragma asmDir:    IRC_CAL IRC_FAST
#pragma asmDir:    FREQ 50_000_000
#pragma asmDir:    ID    'project'
```

This feature can replace the existing /*<?asm <whatever> >?*/ functionality. New C programs should use this #pragma technique instead of the /*<?asm ... ?>*/ technique.

4) C4SX.bat script improvement: The -V option has been added to the CC1B call. This causes a ".var" file to be generated, showing the allocation of variables in the SX RAM.

5) C4SX.bat script improvement: The results of the compiler are more closely checked to verify that the compile is correct before the CC1B2SX converter is called.

Changes in C4SX Version 02

Version 02 contains changes in three basic areas:

1) The /*<?asm whatever.... ?>*/ notation was added as a method to pass SX assembler directives and instructions 'around' CC1B to the SX-Key. In other words, the 'whatever' in the previous sentence is not seen by CC1B but is seen the SX-Key software. This enables the various DEVICE, FREQ, etc. SX directives to be put in the C program.

This removes the need for the CC1B2SX.HDR and TRL files. They are no longer used. In addition, this usage of `/*<?asm break ?>*/` makes it possible to put an SX-Key 'break' in the C program.

2) The second change is the modification of the CC1B2SX program. As mentioned above, it no longer uses the CC1B2SX.HDR file. This file was often the location of the SX-Key directives. Now these need to be put directly in the C source code. Here is an example:

```
/*<?asm
    DEVICE OSCHS3
    IRC_CAL IRC_FAST
    FREQ 50_000_000
    RESET      main
    ID      'project'
?>*/
```

Note: This `/*<?asm whatever ?>*/` syntax is a bit odd -- but it makes it so that these instructions look like comments to CC1B and makes it easy for CC1B2SX utility to find these special areas since "<?asm" and ">?" are unusual literals in a C context. If you have used ASP or PHP, you have seen these constructs before.

3) The sxio.h library is better documented. It is used in the sertstc.c program. If you are going to run this C program, consider running sertstb.sxb first to verify your serial I/O configuration. The sxio.c library contains basic C-style low-level I/O functions (getchar(), putchar(), puts(), _kbhit()) and a delay function (delay_milliseconds()). There are also serial_on() and serial_off() function. These two functions configure resources and start and stop the interrupts, respectively. The serial_off() function should always be called before a program terminates or the last character sent may get truncated, due to the use of asynchronous I/O.

Known CC1B Bugs

Complex software tends to have bugs and CC1B appears to be no exception. As bugs are found and verified, they are passed to the CC1B author. The CC1B author adds them to his CC1B 'bug list'. As CC1B is a free beta, fixes may take some time. Here is a list of known, open bugs, fixed bugs and workarounds:

Open:

As of 9/4/2007, no known bugs!

Fixed (in CC1B 0.7A and C4SXv03):

1) FIXED: Issue with 'const char * pointers'. C example: `int puts(const char *in_str)`

The root issue is with the pointers that access the const char data. Simple solution is not to use 'const char' coding techniques. If you must use them then be sure the pointer to the const char data is in the lower 4 pages of RAM. This approach allows the sertstc.c and sxstring.c sample programs to work. This bug appears on the SX28 and is likely also on the SX48.

2) FIXED: Issue with setting TRIS<x>, PLP_<x> and LVL_<x> mode values for I/O on the SX48. The first on of these registers gets set in a C code sequence, sometimes the others do not. The SX48 sxstring48.c program works around this by only setting TRISA and not setting the other two mode values as they are not really needed.

3) FIXED: Suspected issue with the correct setting of RAM banks (the FSR register). In some limited cases, it appears the FSR register is not set correctly when the C code is changing from one RAM bank to another. As of 08/11/2007 this bug is not verified. Simple C programs are not likely to experience this bug.