The CoinForth Documentation

Arduino Simple BLE Peripheral

Version 0.2

Table of Contents

Abstract

This started out as a Docbook example within Syntext Serna Free 4.3.0-20110207.0, which is the latest **FREE** version I found at http://syntext-serna-free.software.informer.com/download/?ca336a2. I've started adding my coinForth notes, with the eventual goal of becoming some useful documentation.

BLE-STACK 1.4.1

This software used to be available at http://www.ti.com/tool/ble-stack, but I only see **BLE-STACK-2-1** right now, since the CC2540 is pretty old at this point. I have installed it here, so all other file paths will be relative to this:

C:\Texas Instruments\BLE-CC254x-1.4.1.43908\

It requires IAR's EW8051 v9.10.3 with a full license, but all I did was change the device from **CC2540F256** to **CC2540F128** and compiled the **CC2540EM** configuration for using the P0 serial port. I saved the HostTestReleaseCC2540.hex output file, which should work better than the CC2540_SmartRF_HostTestRelease_All.hex file TI included.

TI's SmartRF Flash Programmer Ver. 1.12.7 can program those files, but both files give the following error message:

```
CC2540 - ID1188: HEX file content at address 0x26CEE exceeds chip's 128 kB flash size
```

Even IAR, when debugging, gives the following warning:

```
Warning: Possible IDATA stack overflow detected.
To see the instruction that caused the possible overflow, choose Debug>Break and close this message box. To continue execution, just close
```

The Disassembly shows:

```
?BANKED_ENTER_XDATA:
 001F20  65 0C         XRL   A,V4
>001F22  45 0D         ORL   A,V5
```

That is somewhere in TI's library (e.g. no source code).

I've also change the baud rate in Projects\ble\common\npi\npi_np\npi.h from 115,200 to 19,200 to fit with 328eForth v2.20's existing serial port driver.

```
#define NPI_UART_BR                   HAL_UART_BR_19200
```

I have also found a "better" Project -> Options -> Linker -> Linker configuration file from:

.\Projects\ble\common\cc2540\ti_51ew_cc2540b.xcl to .\Projects\ble\common\cc2540\ti_51ew_cc2540f128b.xcl

However, now I get the following error when compiling:

```
Error[e16]: Segment BLENV_ADDRESS_SPACE (size: 0x1000 align: 0) is too long for segment definition. At least 0x1000 more bytes needed. The
problem occurred while processing the segment placement command
"-Z(CODE)BLENV_ADDRESS_SPACE=_BLENV_ADDRESS_SPACE_START-_BLENV_ADDRESS_SPACE_END", where at the moment of placement
the available memory ranges were "-none-"
   Reserved ranges relevant to this placement:
   CODE:3de6d-3fb30     BANKED_CODE
   BIT:0-7              BREG
   BIT:80-97            SFR_AN
   BIT:a0-af            SFR_AN
   BIT:b8-c7            SFR_AN
   BIT:e8-ef            SFR_AN
   BIT:f8-ff            SFR_AN
Error while running Linker
```

.\Projects\ble\SimpleBLEPeripheral\CC2540DB\SimpleBLEPeripheral.eww compiles and

.\Projects\ble\SimpleBLEPeripheral\CC2540DB\CC2540F128DK-MINI Keyfob\Exe\SimpleBLEPeripheral.hex can be flashed, so let's start there.

The CC2540's serial port is connected to the ATA6614Q pins PD2 (RX) and PD3 (TX), which require a "soft" serial port, like what Arduino provides in their **SoftwareSerial**. https://www.arduino.cc/en/Reference/SoftwareSerial and http://arduiniana.org/libraries/newsoftserial/.

http://ross-arduinoprojects.blogspot.com/2014/04/setting-up-coin-ble-dev-kit.html has: Tool: Select AVRISP mkII and Device: ATA6614Q for Atmel Stdio setup, but can't get the mkII to show up there yet. https://atmelsupport.force.com/customers/500G000000ohYDZ. Fixed!

```
C:\Users\Dennis\Documents>atprogram -t avrispmk2 selftest
Firmware check OK
[ERROR] No self tests to perform for this tool. (TCF Error code: 1)
```

Under the Arduino source, burn the following bootloader:

./hardware/arduino/avr/bootloaders/atmega/ATmegaBOOT_168_atmega328_pro_8MHz.hex

Change the fuse: HIGH: 0XDA from 0XD9, which is used by eForth.

Inside Arduino 1.6.5, File -> Examples -> 01.Basics -> Blink, then Sketch -> Upload and the LED under the reset button starts blinking.

https://github.com/sdwood68/YAFFA Yet Another Forth For Arduino

Using Arduino's Serial Monitor @ 19200 baud, I get:

```
 YAFFA - Yet Another Forth For Arduino, Version 0.6
 Copyright (C) 2012 Stuart Wood
 This program comes with ABSOLUTELY NO WARRANTY.
 This is free software, and you are welcome to
 redistribute it under certain conditions.

 Terminal Echo is On
 Pre-Defined Words : 157
 Input Buffer: Starts at $6AB, Ends at $70A
 Token Buffer: Starts at $68B, Ends at $6AA
 Forth Space: Starts at $140, Ends at $63F
 315 ($13B) bytes free
>>
>>
>> words5 .
```

Looks like there is some work to do.

Then again, using my 328eForth setup in HyperAccess (or whatever you favorite terminal emulator is, setup for 19200 @ 8-None-1), it's working fine!

```
>> 5 .
5  OK
>> : junk 5 0 do [char] * emit loop ;
 OK
>> junk
***** OK
```

Now, to line up the UART pins with each other, I reference C:\Texas Instruments\BLE-CC254x-1.4.0\Documents\swru191f.pdf which is available here http://www.ti.com/lit/ug/swru191f/swru191f.pdf. Table 7-1. Peripheral I/O Pin Mapping tells me that what is called AR_TX on the schematic is connected to P0_5 (pin 14), is USART 1 RX Alt. 1 configuration. The same is true for AR_RX on P0_4 (pin 15), which is TX for that same configuration. However, this means that, in C:\Texas Instruments\BLE-CC254x-1.4.1.43908\Components\hal\target\CC2540EB_hal_uart_isr.c, this is wrong:

```
#define HAL_UART_PERCFG_BIT        0x02         // USART1 on P1, Alt-2; so set this bit.
```

It needs to be:

```
#define HAL_UART_PERCFG_BIT        0x02         // USART1 on P1, Alt-1; so clear this bit.
```

So, I added a BLE_ARDUINO configuration option to switch and keep track of these changes to TI's code.

Since I'm starting to make some significant changes to the BLE-CC254x stack, I've added it to my coinForth repository https://github.com/DRuffer/coinForth. This also allowed me to update to BLE-CC254x-1.4.1.43908b which is the latest release from TI. Opening this in IAR's EW8051 v 9.10.3 reminded me that you have to select the **CC2540F128DK-MINI Keyfob** workspace.

Warning

The rest of this doc are the original contents, which I'll delete from the final doc, but for now, provide easy reference for things I want to do in my content.

Draft Areas

Note the presence of gray "Draft Areas" in the document. They are necessary because Docbook stylesheet rules are often intricate. For instance, `title` in `section` can be specified within `section` itself and within `sectioninfo`. If you specify both, one of them becomes hidden. To avoid this, all such meta-information is shown also in Draft Areas. They can be turned off by changing value of *`show-preamble-editing`* parameter to 0 in `parameters.xsl` file in Docbook stylesheet.

Serna Docbook stylesheet also takes special care of empty content. For example, when you make new article, it provides you with "Title: " inscription where you can enter article title.

Basic editing

Editing of Docbook documents in Serna is quite straightforward, much like in a traditional word-processor. One difference is that you must use "InsertElement" command (**Ctrl-Enter**) to insert new elements. Serna will suggest you a list of elements which you can insert at any given location. Other element operations are listed in "Element" menu.

By default **ENTER** splits the current element. For example, if you are within `a para`, it will be split in two. If you are at the end of paragraph, new paragraph will be added.

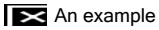
You can see current editing context in the bottom status bar. Navigation commands from "Go" menu should be use for easier navigation in "tagless" mode. Also, pay attention to the two modes of selection: *balanced* and *unbalanced* (they can be toggled from Edit menu or with **Ctrl-B**). In unbalanced mode, selection is more distinct, but it sometimes can be difficult to correctly place ends of selection. In balanced mode selection is automatically adjusted, so it is easier to select list items, etc.

To edit element attributes, press **Ctrl-Enter**.

Images

Inserting images is easy: just insert `figure` or `graphic` elements, invoke *Element Attributes Dialog* for corresponding element, and choose an image file by pressing Browse button for the `fileref` attribute in Element Attributes Dialog.

Figure 1. An example figure



Program listings

Serna supports whitespace stripping policies, as defined by the stylesheet. Editing behavior within whitespace-preserved ares like Docbook `programlisting` is different. Within those elements **ENTER** means newline, and you can mix white-spaces and newlines freely.

```
 SubscriberPtr(SubscriberPtrWatcher* watcher, T* ptr)
      : SubscriberPtrBase(watcher, ptr), P(ptr) {}
SubscriberPtr<T>& operator=(T* ptr)
{
      remove();
      P::operator=(ptr);
      if (!P::isNull())
          P::pointer()->registerSubscriber(this);
    return *this;
}
SubscriberPtr(const SubscriberPtr<T>& other)
      : SubscriberPtrBase(other.watcher(), other.pointer()),
        P(other.pointer()) {}
```

Lists and tables

There are two types of lists in Docbook:

Ordered list. A list may have optional title.

1. First item.
2. Second item.
3. Third item.

Itemized list. Optional title is also available.

- First item.
- Second item.
- Third item.

In Serna, CALS tables are supported by Docbook stylesheet.

Table 1. An example of complex table

Title 1		Title 2	Title 3		Title 4		Title 5	
Sub1	Sub2		Sub3	Sub4	Sub5	Sub6	Sub7	Sub8
A B C D E F G		1. This is item1 2. This is item2			Content		Cells with vertical span.	
		Contents....	This is another horizontal span.					

Localization

It is possible to localize your docbook documents or their parts by simply changing *`lang`* parameter of the compound element. For example, this section's attribute *`lang`* is set to `de`, that is why you see German inscriptions for this section.