ChipCon Tiny Loader, a 1KB serial bootloader for CC1110/CC1111

29 commits 1 branch 0 releases 2 contributors GPL-2.0

Branch: master New pull request Find file Clone or download

tobyjaffey Merge pull request #1 from KryoStoffer/master Latest commit 9b9bfda on 5 Nov 2012

cchl	CCHL, a hardware ChipCon programmer following swra124, running on CC1...	8 years ago
ccpil	CCPIL, a CC1110 hardware loader for Raspberry Pi - to solve the initi...	7 years ago
cctl-prog	Higher timeout when programing via wireless.	7 years ago
cctl-rf	Small documentation changes	7 years ago
cctl	CCHL, a hardware ChipCon programmer following swra124, running on CC1...	8 years ago
cygwin	Builds and runs under cygwin	8 years ago
example_payload	Initial version of CCTL, 1KB serial bootloader for CC1110/CC1111	8 years ago
LICENSE	Initial version of CCTL, 1KB serial bootloader for CC1110/CC1111	8 years ago
Makefile	CCHL, a hardware ChipCon programmer following swra124, running on CC1...	8 years ago
README.markdown	CCPIL, a CC1110 hardware loader for Raspberry Pi - to solve the initi...	7 years ago
cctl-prog.exe	Added --console, built in terminal emulator to talk to device once lo...	8 years ago
prebuilt-cctl.hex	Initial version of CCTL, 1KB serial bootloader for CC1110/CC1111	8 years ago

README.markdown

CC Tiny Loader

CCTL is a serial bootloader for the Chipcon CC1110/CC1111 using only one 1KB page of flash. It allows update of the the microcontroller firmware over its serial port.

Included is CCPIL, a ChipCon Hardware Loader which runs on the Raspberry Pi, which can be used to program the initial bootloader.

Included is also CCHL, a ChipCon Hardware Loader which runs on the CC111x and can program a slave device over the ChipCon debug interface (http://focus.ti.com/lit/ug/swra124/swra124.pdf)

The bootloader consists of two components, a piece of firmware that is flashed onto the device and a utility for downloading code and manipulating the flash memory. The client program, `cctl-prog` runs on Linux, OSX/Darwin and Windows.

Authors

Toby Jaffey (c) 2012 toby-cctl@hodgepig.org

Portions originally from CC Bootloader, Fergus Noble (c) 2011

Usage

The first step is to flash your device with the `cctl.hex` firmware file. A prebuilt version `prebuilt-cctl.hex` is provided.

Once the device is flashed with this firmware it will identify itself over the serial interface with the message "CCTL" on reset.

The microcontroller is configured for 115200, 8N1 transmitting on P0.3, receiving on P0.2. Note that the CC111x is not 5v tolerant, so be sure to use a 3v serial adapter.

You can now use cctl-prog to download your payload code. A prebuilt version of cctl-prog.exe is provided for Windows. For Linux or OSX, type "make".

For usage instructions, run cctl-prog with no arguments:

```
./cctl-prog
```

```
ChipCon Tiny Loader Programmer
cctl-prog -d /dev/ttyXYZ [-c] [-f file.hex]
--help          -h          This help
--console       -c          Connect console to serial port on device
--flash=file.hex -f file.hex Reflash device with intel hex file
--timeout=n     -t n        Search for bootload string for n seconds
--passthrough   -p          Program remote device over passthrough
```

If both `--console` and `--flash` are specified, then the device will be reflashed first, then the console will connect.

Before flashing, `cctl-prog` sends the string "+++", which firmware can detect and reset automatically.

Preparing your user code for usage with the bootloader is very simple. All you need to do is set your linker to start the code section at 0x400. For an example of this see the `Makefile` file in the `example_payload` subdirectory. This is the relevant line:

```
LDFLAGS_FLASH = ... --code-loc 0x400 ...
```

Building

This requires sdcc (Small Device C Compiler). Then it should be as simple as issuing

```
make
```

from the root directory of the project.

The code has been tested with SDCC version 3.0.0

Serial protocol

CCTL uses a custom binary serial protocol, running at 115200bps, 8 bits, no party, 1 stop bit.

On reset, the bootloader prints "\r\nCCTL\r\n" followed by "B" up to 8 times with a delay between each. If the bootloader receives any character before printing "B" 8 times, it will enter upgrade mode.

If no character is received, the bootloader will attempt to launch user code from 0x400.

The bootloader enables the watchdog with a 1s timeout while running. It does not engage the hardware watchdog when jumping to user code (as the watchdog cannot be disabled making it incompatible with applications which remain in deep sleep for long periods).

Once in upgrade mode, the bootloader expects to receive at least one character per second, else it will reset using the hardware watchdog.

In upgrade mode, the following commands are available:

Jump to user code

Jumps to user code. On failure, the device will reset.

`-> j`

Erase page

Erase a 1KB page of flash. On completion, `\0` is sent

`-> e, uint8_t page` (0-31)

`<- \0`

Read page

Read a 1KB page from flash. Sends 1024 raw bytes. On completion, '\0' is sent

`-> r, uint8_t page` (0-31)

`<- uint8_t data[1024], \0`

Load page

Loads 1KB page from serial into a RAM buffer. Receives 1024 raw bytes. On completion, '\0' is sent

`-> l, uint8_t data[1024]`

`<- \0`

Program page

Program a 1KB page of flash from RAM buffer. On completion, `\0` is sent

`-> p, uint8_t page` (0-31)

`<- \0`

Interrupts

CCTL resides in the first flash page, 0x0000 to 0x0400. On reset, the CC1110 begins executing from address 0x0000 where it finds the CCTL reset vector.

CCTL provides a vector table which jumps to the relevant vector in application code at 0x400 + offset.

For interrupts which are used by both CCTL and application code, CCTL looks at the F1 user flag in the PSW register. When this flag is 0, the bootloader isr is called, when 1, the application isr is called.

Application code should not modify PSW.F1.

How do I get CCTL into my flash?

The CC111x is programmed with a SPI like protocol using a hardware programmer. The protocol is given in detail in http://focus.ti.com/lit/ug/swra124/swra124.pdf

(CCTL is placed in the flash using this protocol, it is different to the serial protocol).

If you have access to one of the devices below, you can bootstrap out of it, by programming CCHL into a CC1110. CCHL allows the CC1110 to program virgin chips over the debug interface.

To reflash a slave device, connect P1_6 to DD, P1_5 to DC and P1_4 to RESET, load `cchl.hex` onto a CC1110 running CCTL and run `cctl-prog` using the `--passthrough` flag. Eg. to program the `cctl` bootloader into a device:

```
`./cctl-prog -p -d /dev/ttyUSB0 -f cctl.hex`
```

Official hardware programmer

- TI's CC-Debugger http://www.ti.com/tool/cc-debugger (Windows only)

Third-party hardware programmers

- Travis Goodspeed's GoodFET http://goodfet.sourceforge.net/

Open source implementations of protcol

- GoodFET (python) http://goodfet.sourceforge.net/clients/goodfet.cc/
- Teensy (C) https://github.com/jkerdels/open_imme/tree/master/tools/teensy-prog
- Linux GPIO sysfs (C) https://github.com/ffainelli/cc2530prog

Building for Windows

`cctl-prog` is currently built for Windows using the mingw32 cross compiler from Linux.

```
make -f Makefile.mingw32
```

The Linux/OSX version of `cctl-prog` compiles and runs under cygwin.

Run the createdevices.sh script in cygwin to populate /dev.

In cygwin COM ports are available as /dev/ttySX. BEWARE, THEY ARE NUMBERED FROM 0 ie. COM7 = /dev/ttyS6