25 June 2010

Linux development environment for ez430-RF2500

This document describes how to set up a development environment on Linux for the ez430-RF2500. Tested on Gentoo Linux, kernel version 2.6.34.

Overview

Basically we need to:

- Configure the kernel so that we can view the serial port output from the board.
- Setup the MSPGCC toolchain so that we can compile source code into a binary executable that runs on the MSP430.
- Install MSPDebug so that we can flash the binary executable onto the RF2500 board.
- Download the SimpliciTI library, which is needed to use the CC2500 radio, and modify some code so that we can compile the library using MSPGCC. Note that we don't need to use the entire library but we need 2 entities: the Minimal RF Interface (MRFI) and the Board Support Package (BSP). The MRFI and the BSP are described in [5].
- Create a Makefile for development.

Configuring the kernel

The recommended kernel version is >2.6.31, it seems [1]. When configuring the kernel, enable USB Serial Converter support, USB TI 3410/5052 Serial Driver, and USB Modem (CDC ACM) support.

```
Device Drivers --->
  [*] USB support
      <M> USB Modem (CDC ACM) support
      <M> USB Serial Converter support --->
          <M> USB TI 3410/5052 Serial Driver
```

Setting up the MSPGCC cross-compile toolchain

1. Download radhermit's MSP430 overlay [2] and add it to the Gentoo system. Untar it into /var/lib/pizu-overlay and add to /etc/make.conf:

   ```
   # must be below sourcing of layman's make.conf
   PORTDIR_OVERLAY="${PORTDIR_OVERLAY} /var/lib/pizu-portage-overlay"
   ```

2. Emerge sys-devel/crossdev from this overlay.
3. Install the MSPGCC toolchain with GDB support:
   `sudo crossdev --ex-gdb msp430`
4. After crossdev successfully creates the toolchain, symlink the `ldscripts` folder into a place where the linker will find it. Run:

   `sudo ln -s /usr/$CHOST/msp430/lib/ldscripts /usr/msp430/lib/ldscripts`

   where `$CHOST` is something like `x86_64-pc-linux-gnu` or `i686-pc-linux-gnu` for standard amd64 or x86 architectures running Gentoo.

Setting up MSPDebug

1. Download MSPDebug from [3].
2. Unpack, compile and install the source:

```
tar xvfz mspdebug-version.tar.gz
cd mspdebug-version
make
make install
```

3. Create a new udev rule named /etc/udev/rules.d/99-msp430.rules:

```
ATTRS{product}=="Texas Instruments MSP-FET430UIF", MODE="0660",
GROUP="plugdev"
```

4. Now all users in the plugdev group should be able to access the device.

Download and modify the simpliciTI libraries

Information here is from [6].

1. Download the latest release of SimpliciTI from [7]. Choose the version for IAR.
2. Create a development folder where the source code and modified SimpliciTI will reside.
3. Copy the bsp/ and mrfi/ subfolders from Components/ into the development folder.
4. *Optional*: Some of the code is unused and can be removed. CC2500 is a Family 1 radio, so remove unused radios (Family 2 to Family 5) from mrfi/radios/. Remove unused boards from bsp/boards/.
5. *Optional*: For the complete SimpliciTI stack, copy the simpliciti/ subfolder from Components/ into the development folder.
6. Modify bsp/mcus/bsp_msp430_defs.h.
   Replace the line:
   #error "ERROR: Unknown compiler."
   with:

```
#include <io.h>
#include <signal.h>
#include <iomacros.h>
#define __bsp_ISTATE_T__                 uint16_t
#define __bsp_ISR_FUNCTION__(f,v)        interrupt (v) f(void)
#define __bsp_ENABLE_INTERRUPTS__()      eint()
#define __bsp_DISABLE_INTERRUPTS__()     dint()
#define __bsp_INTERRUPTS_ARE_ENABLED__() (READ_SR & 0x8)
#define __bsp_GET_ISTATE__()             (READ_SR & 0x8)
#define __bsp_RESTORE_ISTATE__(x)        st(if((x&GIE))_BIS_SR(GIE);)
```

7. Modify bsp/drivers/code/bsp_generic_buttons.h.
   Remove the line:

```
#error "ERROR: Debounce delay macro is missing."
```

8. Optional: If the simpliciti/ folder was copied, modify simpliciti/nwk/nwk_QMgmt.c.
   Remove the line:

```
#include <intrinsics.h>
```

9. Modify mrfi/mrfi_defs.h.
   Change all occurrences of

```
#define __mrfi_MAX_PAYLOAD_SIZE__       20
```

   to

```
#define __mrfi_MAX_PAYLOAD_SIZE__       53
```

Putting it all together

To compile for the MSP430F2274, do:

```
msp430-gcc -o zzz -mmcu=msp430x2274 zzz.c
```

To upload the program to the board:

```
mspdebug -R "prog zzz"
```

Watch the serial output using minicom. The device is /dev/ttyACM0, 9600 baudrate.

Sample Makefile

```
# Adapted from Makefile example at
# http://senstools.gforge.inria.fr/doku.php?id=lib:simpliciti:example
# Includes only BSP and MRFI entities, not the whole SimpliciTI stack.

SIMPLICITI_COMPONENTS_PATH = ./
BSP_PATH     = ${SIMPLICITI_COMPONENTS_PATH}/bsp
MRFI_PATH    = ${SIMPLICITI_COMPONENTS_PATH}/mrfi

BOARD    = EZ430RF
CPU      = msp430x2274
RF       = -DMRFI_CC2500

INCLUDES = -I${MRFI_PATH}/                  \
           -I${BSP_PATH}/                   \
           -I${BSP_PATH}/drivers/           \
           -I${BSP_PATH}/boards/${BOARD}/   \

OBJECTS  = bsp.o mrfi.o
CC       = msp430-gcc
CFLAGS   = -DMAX_HOPS=3 ${RF} ${SMPL_NWK_CONFIG} -mmcu=${CPU} -O2 -Wall -g $
{INCLUDES}

all: zzz

clean:
      rm -f ${OBJECTS} zzz

%.o: %.c
      $(CC) ${CFLAGS} -c -o $@ $<

mrfi.o:     ${MRFI_PATH}/mrfi.c
      ${CC} ${CFLAGS} -c $< -o $@
      @echo CC $<

bsp.o: ${BSP_PATH}/bsp.c
      ${CC} ${CFLAGS} -c $< -o $@
      @echo CC $<

zzz: zzz.c ${OBJECTS}
      $(CC) ${CFLAGS} -o zzz zzz.c ${OBJECTS}
```

References

[1] http://www.koka-in.org/~kensyu/handicraft/diary/20100419.html
[2] Radhermit's MSP430 Gentoo overlay - github.com/radhermit/msp430-overlay
[3] MSPDebug - http://mspdebug.sourceforge.net/
[4] MSPGCC - http://mspgcc.sourceforge.net/ and http://mspgcc4.sourceforge.net

[5] SimpliciTI Developers Notes
[6] Setting up the MSP430 environment - http://www.tooz.us/projects/MSP430/
[7] SimpliciTI TI Software Folder - http://focus.ti.com/docs/toolsw/folders/print/simpliciti.html