Z1 Sampler

Programming the Device

At the present, programming the Z1, on the `driver-sampler` branch, will require the use of the MSP-FET430UIF. There exists an issue when programming with `make upload` transmitting the wrong password.

Using the MSP-FET430UIF

`mspdebug` is used to communicate with the Z1's MSP430F2617 using JTAG. The orientation of the JTAG connector can be seen in Figure 1. If a pin header is not available, it is necessary to hold the pins against the board's connector carefully for the duration of the programming operation.

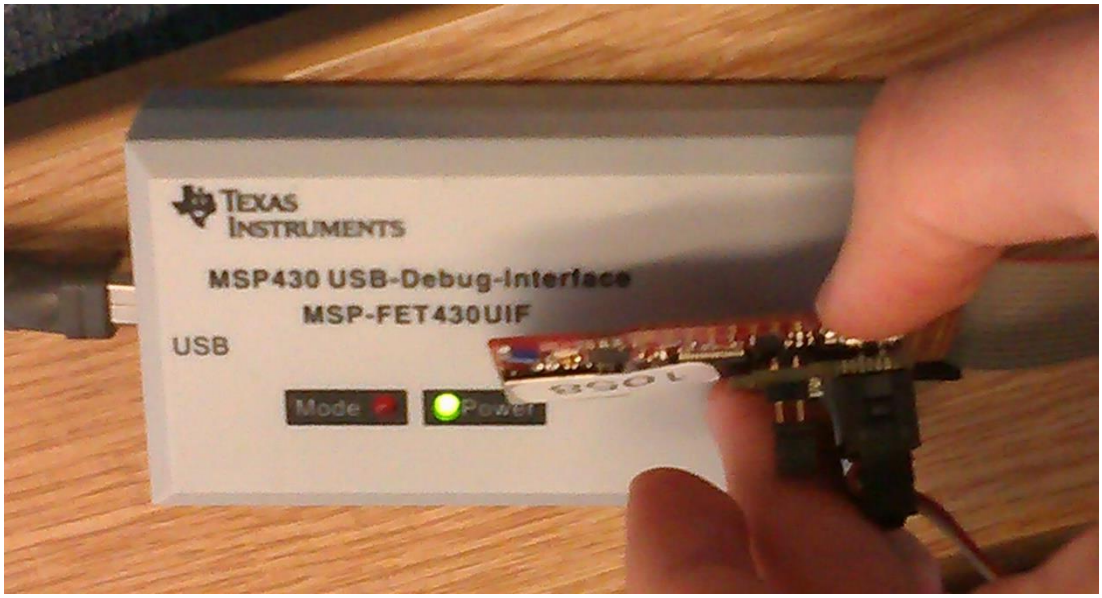

Figure 1 - JTAG connector orientation for the Z1

`mspdebug` is available with apt-get. Once installed, the typical command for use is `mspdebug (driver) -d (device) -j`. An example for the Z1 is

```
mspdebug tilib -d /dev/ttyACM0 -j
```

If this is the first time the debugger has been used, it may be required to run with the flag

```
--allow-fw-update
```

Once mspdebug is run, it will attempt to find the device, which should be attached/held on the JTAG connector. If not connected properly, an unknown device error will be thrown. If successful, the device model will be shown and a list of available commands will become apparent.

Programming the Z1 with the debugger

In order to program the Z1, an ELF file should be made, this will come in the guise of a .z1 file. Before the device can be programmed, the ROM must be erased. This can be done with **`erase all`** within the `mspdebug` command line. Once complete, the device can be programmed with **`prog (filename)`**. Once complete, the device can put removed from the debugger and powered with USB.

The Basics of Configuring the Z1 Sampler

The Z1 Sampler can be configured via its web interface. The web interface is accessible on port 80 of the device.

WARNING

Access to the webserver is slow and in some cases will fail. The choice of browser will also affect performance significantly. Firefox tends to give up if the download take too long, even with a high timeout in the config. Chromium seems to work quite well and unlike Firefox, will show the page as it downloads, giving reassurance that something is happening. Lynx is good for a CLI browser and copes at least as well as Chromium. **It is strongly advised to have Chromium and Lynx available.**

If the page fails to load, or is not making any progress downloading, try again. In lab tests, it was not uncommon for the page to just not load or for the connection to hang and no progress made.

Accessing the Configuration Pages

The root of the webserver contains the index of all the configuration pages, allowing for easy navigation. By default, all non-existent pages will be served as the root index.

/clock

This allows configuration of the real time clock. Input is taken in the year, month, etc. Once the form is submitted, a “Success!” will be sent back to acknowledge the submission and setting of the RTC. If this acknowledgement is not received, it can be assumed the form did not submit and it will be necessary to try again.

/sample

This allows configuration of the sampling.

Sample Interval

The time in seconds between samples. Example: `900`

AVR IDs

The IDs of the AVR sensors (0-255) separated by . (periods). Example: `14.102.255`

Rain? ADC1? ADC2?

Select these if the device should read the appropriate sensor.

/comms

This allows configuration of the communications.

Comms interval

The time in seconds between POST sessions. Note that the comms interval is synchronised to the RTC, so if all devices have the same interval, they will all POST at the same time, regardless of when they were set.
Example: `3600`

Gateway IP

The IPv6 address of the POST gateway. Each field contains 4 hexadecimal numbers.
Example: `2001 630 d0 f111 224 e8ff fe38 6cf2`

Gateway Port

The port of the POST gateway. Example: `8080`

Hardware

The Z1 Sampler is designed to be run on the Z1 mote.

Sensors

Universal Internal Sensors

These are sensors that are assumed to be on every mote. As they are standard for the Z1, this was assumed to be a safe assumption. As such, basic sensors will *always* be sampled. The universal internal are:

- Battery Voltage
- Internal Temperature
- Internal Accelerometer (x, y and z)

To keep processing on the mote to a minimum where possible, accelerometer readings are read in the direction of the 3 spatial axes. As conversion to pitch and roll requires the use of the `sqrt` and `atan` functions, it would introduce excessive wasted resources on the mote.

Universal External Sensors

These are sensors that are assumed will be attached to every device that are not part of the Z1 mote as standard. These sensors will *always* be sampled. They are:

- Real Time Clock

The RTC is used throughout the Z1 Sampler to help prevent clock drift in timers.

Optional Sensors and Sensor Interfaces

- ADC1
- ADC2
- Rain
- AVR

For the ADC1 and ADC2, the values are read off as unsigned 16-bit integers and stored in protocol buffers as such. They have no human-readable labels.

Software

The Z1 Sampler consists of 3 main elements, the webserver, the sampler and the POSTer.

The Webserver

The webserver is non-standard code to replace the bloated example one. It makes use of protosockets and has IPv6-only support, running on port 80. Access will be slow, but if all fails, use of netcat to diagnose the problem or download the webpage may prove successful:

```
netcat -6 2001:630:d0:f200:c30c:0:0:fda 80
>     GET / HTTP/1.1

<     HTTP/1.1 200 OK
<
<     <html><body><p>…
```

Although attempts have been made to optimise for size, some pages are still around 600 bytes in size, and with speeds measured in bytes per second, this will take a while to download.

The Sampler

The sampler takes a reading every sampler interval, as set in the sample configuration. This is then encoded with protocol buffers and appended to a file. Sample files are at most 256 bytes in size (due to limitations of POST success over the lossy network). Sample files are named r_0, r_1, etc. If a sample cannot be appended without going over the 256 byte limit, it is placed in a new file.

The POSTer

The POSTer will attempt to connect to the POST gateway every POST interval. It should be noted that the POST interval will account for clock drift to ensure the network wakes up at the same time.

Various Findings and Interesting Bits

Contiki-OS contains a lot of bloated code, such as non-debug `printf`, creating a lot of wasted space on the device.

Floating point numbers create a problem in that they also have to use software for floating point operations as opposed to a dedicated FPU. This makes their use slow and space inefficient. A simple test estimated the space for the FP code to be around 3kB in size.

Protothreads don't behave in a linear fashion which can be very confusing when dealing with some more complicated functions. This is because they use Duff's device, which effectively allows PTs to resume execution by returning from a function and later return to that point when called again. Hence, what appears to be a blocking function is actually non-blocking.

POST Gateway

Installation

The POST gateway **will require python with IPv6 enabled**. Unfortunately, this means Python must have been configured with IPv6 when built.