Internship @ Mindtribe:
Progress report 08-14-2015

RETROSPECTIVE: WEEK 1 – 08-03

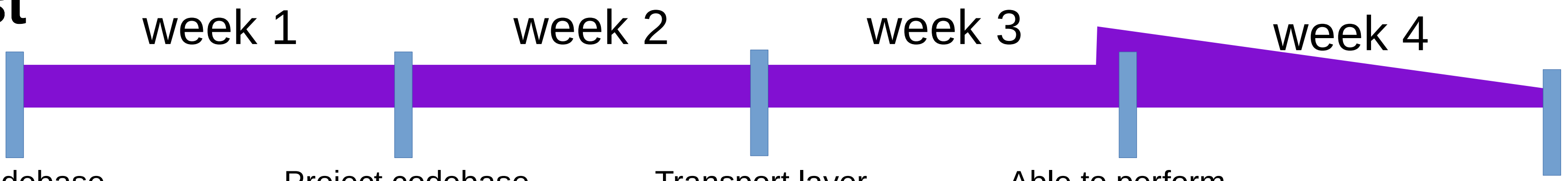
- **Arrival at Mindtribe:**
  - Warm welcome and information sessions (company structure, policies and project approach)
- **Project definition:**
  - A JTAG debug adapter for, and implemented on, TI CC3200 SoC (ARM Cortex-M4)
  - Wi-Fi TCP connection to host GDB session (no intermediate remote debugging tool such as OpenOCD required)
  - Goal: be a convenient internal development tool, and possibly a nice open-source showcase project for Mindtribe.
  - Secondary goal: architect the project neatly, such that it may later be portable to other platforms and/or target debuggee chips

RETROSPECTIVE: WEEK 1 – 08-03 (cont.)

- **Project planning:**
  - Rough timeline planning put together with Mike Ho (next page)
  - Day-to-day guidance from Jerry Ryle
- **Preliminary reading/feasibility study**
  - Verified availability of necessary documentation:
    - TI (CC3200's JTAG interface and internal routing, application notes on Wi-Fi)
    - ARM (ARM JTAG debug interface, ARMv7 Debug features, Cortex-M4 embodiment of those)
    - GDB (GDB's serial protocol for remote debugging – reasonably well-documented)
    - OpenOCD – a great example solution to inspect when all else fails.
- **First implementation steps:**
  - *Isolated+adapted TI's “Blinky LED” example* to a stand-alone blank project for GCC compilation
  - *Set-up OpenOCD toolchain* to “debug the debugger”
  - I*mplemented JTAG transport layer code* – was able to read TI's manufacturer ID from the debuggee chip using the CC3200 debugger.

PROJECT PLANNING (JTAG → GDBserver → WiFi)

August



Project codebase set up, toolchain configured

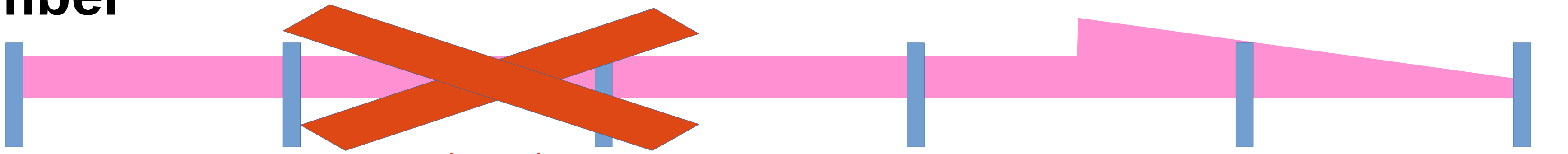
Project codebase set up, toolchain configured

Transport layer done, able to read man. ID

Able to perform one debugging funtion (e.g. halt)

Able to perform most relevant debugging functions

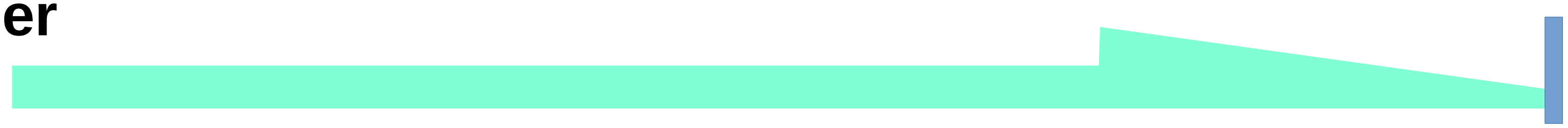
September



Focus on getting a serial link to GDB host (UART or similar) and implementing the GDBserver stub for communicating to GDB. Meanwhile, finish implementing a minimal set of JTAG debugging functions (halt, step, continue, reset, memory read/write, register read/write).

A limited but working UART debugger

October



Focus on implementing a serial link over WiFi through TCP sockets, configuring the CC3200 as an access point. Once done, existing GDBserver stub should be easy to adapt to use this.
Other than that, polish and add any extra features that time may allow (breakpoints, watchpoints? Serial communication to debuggee? Etc)
Also: knowledge transfer, evaluation/code review, open-sourcing the project.

A (reasonably) complete WiFi debugger.

RETROSPECTIVE: WEEK 2 – 08-10

- **Have been focusing on implementation so far, and reading/learning as I go.**
- **Off to a good start (one week ahead of planning)**
  - JTAG transport layer implemented (simple GPIO)
  - JTAG control layer implemented (scan-chain operations, TMS state machine)
  - Implemented abstraction layers analogously to the “maze” of logical entities encountered in the debuggee SoC:
    - TI ICEPICK router, which re-routes JTAG traffic
    - ARM “JTAG-DP”, standardized access port with a proprietary register read/write protocol
    - ARMv7 MEM-AP, standardized second-tier access port to interface to the system memory (in which various debug feature registers are mapped), with another “nested” register read/write protocol
    - Played with a few features accessible through the MEM-AP.
  - Implemented a small “debuggee-under-test” program to run on the debuggee: blinks an LED (easy to see when processor is halted) and continuously writes a specific memory address to another LED (easy to verify memory poke success, by disabling this LED using the debugger)
  - Successfully performed two different debug operations: halting the processor and poking a memory location.

LOOKING AHEAD: WEEK 3 – 08-17

- **The bulk of abstraction layers for debugging features is there.** Time to broaden the focus a little bit:
  - Make a comprehensive list of all the debug features this Cortex-M4 implementation actually supports
  - Estimate workload for leveraging each feature
  - Decide on importance of each feature (keeping in mind typical GDB usage patterns)
  - Decide on the minimal required feature subset for this project, and start tackling them.
- **Will give a small presentation about JTAG learnings** during Mindtribe's weekly soft-/firmware team meeting (08-18)

QUESTION TO KEES:

- You mentioned you would like to see a small report after the first month – literature study.
  - Project is very specific: have learned a lot of technical details (e.g. JTAG protocol operation, ARM architecture, OpenOCD internals and workflow, Makefile-based codebase, etc), but not very “academically interesting”
  - What would you like to see in this report? (e.g. A broader view on recent advances/initiatives in debugging?)
  - Could you give a rough indication of the level of detail you expect?

- these points will help determine how we can fit it into the project.