Open-Source Cortex-M Development and Debugging

May 20, 2015

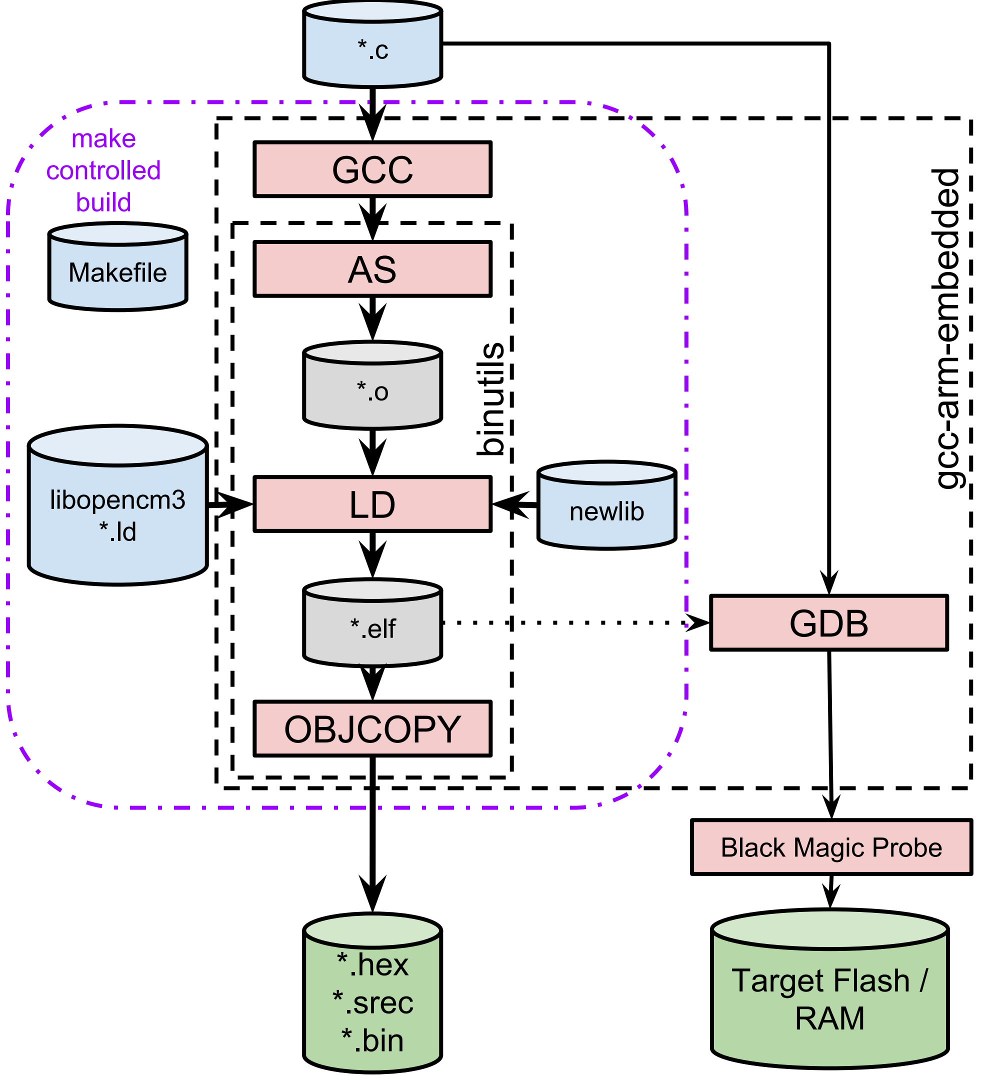
Gareth McMullin

Overview

- Introduction to open-source bare-metal ARM stack
- Highlight differences between Unix and embedded
- Introduction to remote debugging with GDB
- Demonstrate complete workflow with an example using Newlib's `<stdio.h>`
- Some simple techniques for reducing binary size

ARM Toolchain

- Your fav text editor
- make
- gcc-arm-embedded
  - gcc
  - binutils (as, ld, objcopy, …)
  - newlib
  - gdb
- libopencm3
- Black Magic Probe



ARM Toolchain Links

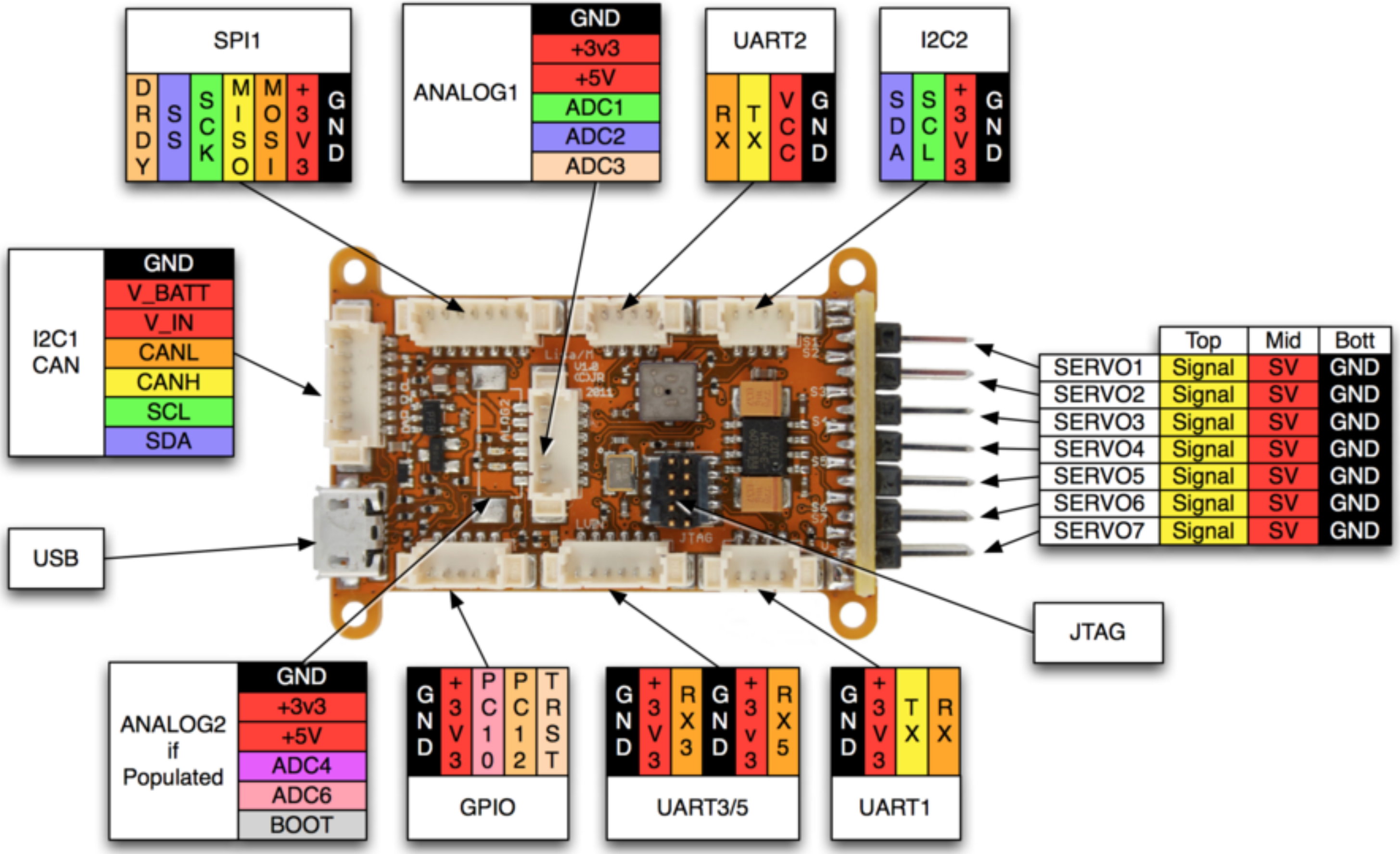
- gcc-arm-embedded
  https://launchpad.net/gcc-arm-embedded/
- libopencm3
  https://github.com/libopencm3/libopencm3
- Black Magic Probe
  https://github.com/blacksphere/blackmagic

Unix Demo Program

Github: tree

```c
/* demo.c */
#include <stdio.h>

int main(void)
{
  int counter = 0;
  while (1) {
    int a, b;
    printf("Iteration %d\n", ++counter);
    printf("Give me two numbers to add.\n");
    scanf("%d %d", &a, &b);
    printf("%d + %d = %d\n\n", a, b, a + b);
  }
  return 0;
}
```

```make
# Makefile
CFLAGS = -Wall -Wextra -g3 -O0 -MD

CSRC = demo.c
OBJ = $(CSRC:.c=.o)

demo: $(OBJ)
	$(CC) $(LDFLAGS) -o $@ $(OBJ) $(LDLIBS)

.PHONY: clean
clean:
	-rm -rf *.o demo

-include *.d
```



	Top	Mid	Bott
SERVO1	Signal	SV	GND
SERVO2	Signal	SV	GND
SERVO3	Signal	SV	GND
SERVO4	Signal	SV	GND
SERVO5	Signal	SV	GND
SERVO6	Signal	SV	GND
SERVO7	Signal	SV	GND

Lisa/M v1.0 Autopilot from https://wiki.paparazziuav.org/wiki/Lisa/M_v1.0

Makefile for cross-compile

Github: tree / diff

```diff
+CROSS_COMPILE = arm-none-eabi-
+CC = $(CROSS_COMPILE)gcc
+CPUFLAGS = -mcpu=cortex-m3 -mthumb
-CFLAGS = -Wall -Wextra -g3 -O0 -MD
+CFLAGS = -Wall -Wextra -g3 -O0 -MD $(CPUFLAGS)
+LDFLAGS = $(CPUFLAGS)
+LDLIBS = -lc -lnosys
```

CPUFLAGS from gcc-arm-embedded readme.txt

Newlib libc and Unix syscalls

- https://sourceware.org/newlib/libc.html#sprintf
- https://sourceware.org/newlib/libc.html#Syscalls

  (Nice PDF documentation included with gcc-arm-embedded)

- Newlib provides dummy syscalls in libnosys:

```
LDLIBS = -lc -lnosys
```

Generating Hex files

```
OBJCOPY = $(CROSS_COMPILE)objcopy

%.hex: %.elf
    $(OBJCOPY) -O ihex $< $@

%.srec: %.elf
    $(OBJCOPY) -O srec $< $@
```

What’s still missing?

We have a binary using ARM instructions, but no OS loader:

```
$ arm-none-eabi-nm demo.elf | sort | head -n 3
00008000 T _init
00008010 T exit
00008030 t register_fini
```

- Code at arbitrary addresses
- No interrupt vector table

Introducing libopencm3

- **Newlib** provides a device independent C library.
- **libopencm3** provides device dependent support.
    - Peripheral MMIO register definitions
    - Interrupt vector table definition
    - Convenience routines for peripheral access
    - Linker scripts

Git Submodule: libopencm3

Github: tree / diff

```
$ git submodule add https://github.com/libopencm3/libopencm3.git
```

```
+.PHONY: libopencm3
+libopencm3:
+	if [ ! -f libopencm3/Makefile ]; then \
+		git submodule init; \
+		git submodule update; \
+	fi
+	$(MAKE) -C libopencm3 lib/stm32/f1
+
 .PHONY: clean
 clean:
+	-$(MAKE) -C libopencm3 clean
```

Documentation on Git Submodules

The linking stage

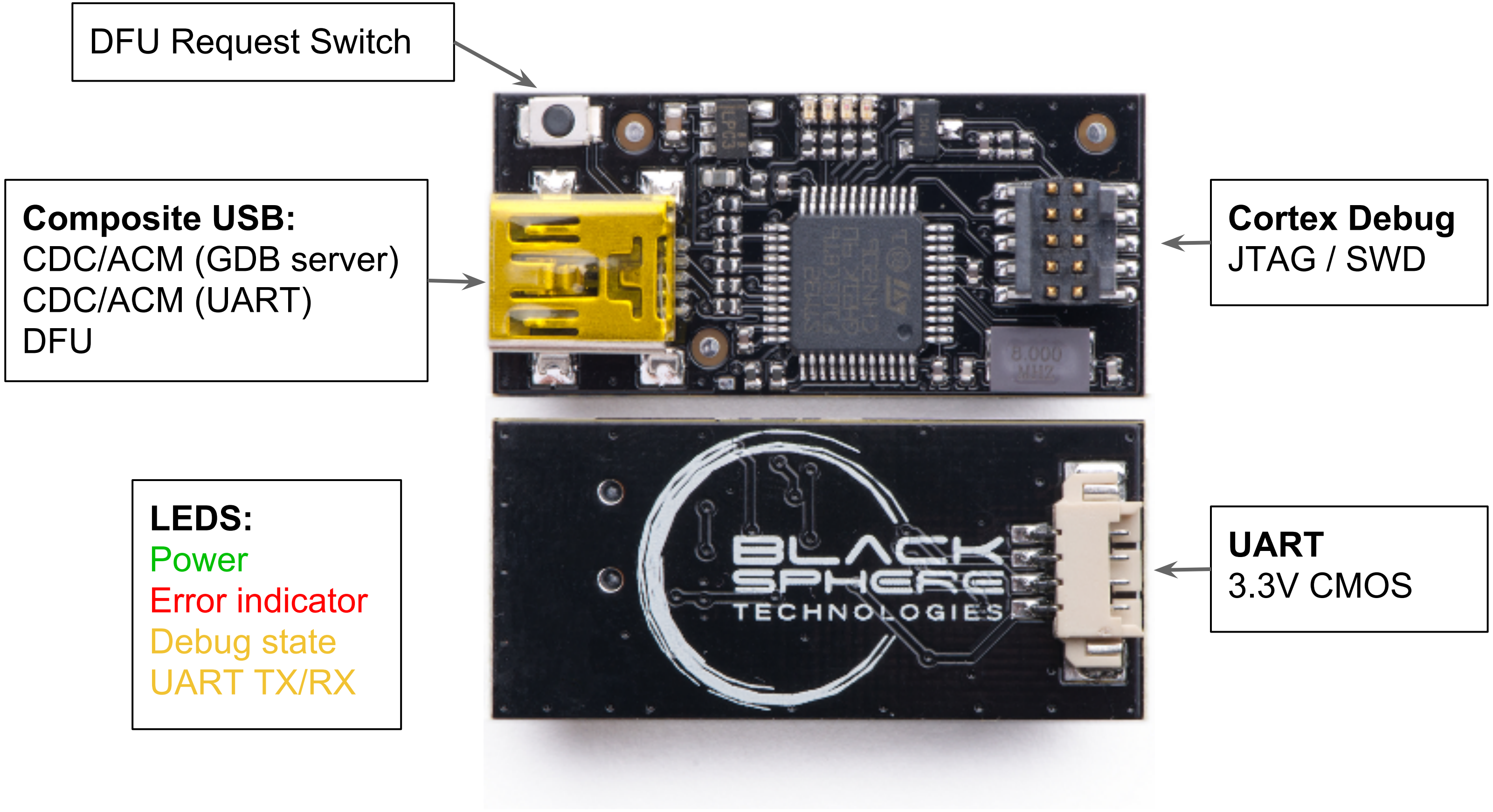
Github: tree / diff

```
/* lisa-m.ld - Linker script for Lisa-M
*/

/* Define memory regions. */
MEMORY
{
    rom (rx) : ORIGIN = 0x08000000,
               LENGTH = 128K
    ram (rwx) : ORIGIN = 0x20000000,
                LENGTH = 20K
}

/* Include the common ld script. */
INCLUDE libopencm3_stm32f1.ld
```

```
-CFLAGS = -Wall -Wextra -g3 -O0 -MD $(CPUFLAGS)
+CFLAGS = -Wall -Wextra -g3 -O0 -MD $(CPUFLAGS) \
+         -DSTM32F1 -Ilibopencm3/include
-LDFLAGS = $(CPUFLAGS)
+LDFLAGS = $(CPUFLAGS) -nostartfiles \
+         -Llibopencm3/lib -Wl,-T,lisa-m.ld
-LDLIBS = -lc -lnosys
+LDLIBS = -lopencm3_stm32f1 -lc -lnosys
```

Remote Debugging with GDB

- GDB offers remote debugging using a serial protocol over TCP, pipes, serial ports, etc.
  https://sourceware.org/gdb/current/onlinedocs/gdb/Remote-Protocol.html#Remote-Protocol
- Remote server implementations:
  - gdbserver
  - gdb-stubs
  - kgdb
  - Black Magic Probe
  - OpenOCD / texane/stlink / PyOCD / Segger J-Link GDB server
- Connect with `target extended-remote ...`
- Send commands to server with `monitor ...`



Black Magic Probe (Mini)

Demo with Black Magic Probe

For OS X: `/dev/cu.usbmodem*1`
For Windows: `COM?`

For JTAG, use `monitor jtag_scan`

```
(gdb) target extended-remote /dev/ttyACM0
Remote debugging using /dev/ttyACM0
(gdb) monitor swdp_scan
Target voltage: 3.4V
Available Targets:
No. Att Driver
 1        STM32, High density.
(gdb) attach 1
Attaching to program: /home/gareth/Projects/embedded-demo/demo.elf, Remote target
reset_handler () at ../../cm3/vector.c:67
(gdb) load
Loading section .text, size 0xc5d8 lma 0x8000000
Loading section .ARM.exidx, size 0x8 lma 0x800c5d8
Loading section .data, size 0x8c8 lma 0x800c5e0
Start address 0x80001a8, load size 52904
Transfer rate: 18 KB/sec, 944 bytes/write.
(gdb) start
Temporary breakpoint 1 at 0x8000156: file demo.c, line 5.
Starting program: /home/gareth/Projects/embedded-demo/demo.elf
Note: automatically using hardware breakpoints for read-only addresses.

Temporary breakpoint 1, main () at demo.c:5
5       int counter = 0;
```

GDB Host I/O and ARM Semihosting

- ARM defines a host system call interface: http://infocenter.arm.com/help/topic/com.arm.doc.dui0471c/DUI0471C_developing_for_arm_processors.pdf (Chapter 8)
- GDB provides a host system call interface: https://sourceware.org/gdb/current/onlinedocs/gdb/File_002dI_002fO-Remote-Protocol-Extension.html
- Unfortunately these don't match. Black Magic Probe provides the translation.
- Newlib provides ARM compliant semihosting syscalls enabled with linker command line argument `--specs=rdimon.specs`.
- Must initialise file descriptors with `initialise_monitor_handles()`.

Newlib’s semihosting syscalls

Github: tree / diff

```
          -DSTM32F1 -Ilibopencm3
 LDFLAGS = $(CPUFLAGS) -nostartfiles \
          -Llibopencm3/lib -Wl,-T,lisa-m.ld
-LDLIBS = -lopencm3_stm32f1 -lc -lnosys
+LDLIBS = -lopencm3_stm32f1 \
+         --specs=rdimon.specs

 CSRC = demo.c
 OBJ = $(CSRC:.c=.o)
```

```
 int main(void)
 {
   int counter = 0;
+
+  void initialise_monitor_handles(void);
+  initialise_monitor_handles();
+
    while (1) {
                int a, b;
                printf("Iteration %d\n",
```

Note: Semihosting calls use BKPT instructions for communication with the debugger. If no debugger is connected these will trigger faults.

STM32 USART with libopencm3

Github: tree / diff

```
int _read(int file, char *ptr, int len)
{
  int i;

  if (file != STDIN_FILENO) {
    errno = EIO;
    return -1;
  }

  for (i = 0; i < len; i++) {
    ptr[i] = usart_recv_blocking(USART2);
    usart_send_blocking(USART2, ptr[i]);
    if (ptr[i] == '\r') {
      ptr[i] = '\n';
      usart_send_blocking(USART2, ptr[i]);
      return i + 1;
    }
  }
  return i;
}
```

```
int _write(int file, const char *ptr, int len)
{
  int i;

  if (file != STDOUT_FILENO) {
    errno = EIO;
    return -1;
  }

  for (i = 0; i < len; i++) {
    if (ptr[i] == '\n')
      usart_send_blocking(USART2, '\r');
    usart_send_blocking(USART2, ptr[i]);
  }

  return i;
}
```

libopencm3 API reference

Why is the binary so big?

Github: tree / diff

Binary size before:

```
$ arm-none-eabi-size demo.elf
  text       data     bss      dec       hex    filename
 54880       2260     72      57212      df7c    demo.elf
```

Some low-hanging fruit:

```
CFLAGS += -Os
LDFLAGS += --specs=nano.specs
CFLAGS += -ffunction-sections -fdata-sections
LDFLAGS += -Wl,--gc-sections
```

After:

```
$ arm-none-eabi-size demo.elf
  text       data     bss      dec       hex    filename
  8820       116      16       8952      22f8    demo.elf
```

More examples

- libopencm3 Examples repository:
  https://github.com/libopencm3/libopencm3-examples.git
- Black Magic Probe firmware:
  https://github.com/blacksphere/blackmagic.git
- Swift Navigation Piksi firmware:
  https://github.com/swift-nav/piksi_firmware.git
- Projects based on libopencm3:
  http://libopencm3.org/wiki/Projects_based_on_libopencm3

Further discussion

- The pub afterwards?
- libopencm3 on Gitter or #libopencm3 on irc.freenode.
  net
- Black Magic Probe on Gitter