Low Level Programming by Example

Considerations, Techniques and Examples for Low Level Programming in C on Small Processors like the ARM Cortex-M Family

Mike Stitt

July 26, 2017

This talk describes the techniques I like when controlling small microprocessors like the ARM Cortex-M.

- With a small example program.
- We'll walk through the full initialization of the small example program.

“Once upon a time, two programmers walked into a bar and agreed on the best way to program a Kraken.”

Yes – that’s a fable.

Tools I use

- OS X
- Ubuntu/Parallels
- gnu make
- make menuconfig – using kconfig-frontends
  - By Yann E. MORIN http://ymorin.is-a-geek.org/projects/kconfig-frontends
- arm-none-eabi-gcc, arm-none-eabi-ld, arm-none-eabi-gdb
  - All the standard gnu gcc compiler and linker tools, with a “arm-none-eabi-” prefix.
  - looking at clang
- Python, Ruby
- Segger jlink
- Volt meter
- Saleae Logic Pro 16
- FTDI serial cables

ARM eabi – what is it?

- ARM Embedded Application Binary Interface
  - https://stackoverflow.com/questions/8060174/what-are-the-purposes-of-the-arm-abi-and-eabi
  - https://wiki.debian.org/ArmEabiPort
  - Defines register calling conventions for bare-metal C.

Libraries

- Newlib – comes with arm-none-eabi-gcc
- Nanopb
- lwip – Light Weight IP
- Printf2 by Georges Menie and Daniel D. Miller
    - https://github.com/spindance/printf-stdarg-float/blob/master/source/printf2/printf2.c
- Vendor libraries like
    - STM32F4 DSP and standard peripherals library
        - STMicroelectronics
        - http://www.st.com/content/st_com/en/products/embedded-software/mcus-embedded-software/stm32-embedded-software/stm32-standard-peripheral-libraries/stsw-stm32065.html
    - CMSIS
        - ARM
        - https://developer.arm.com/embedded/cmsis

Important embedded c keywords

- volatile
  - An instruction to the compiler, not to the processor to access the memory location as coded.
    - -It is not and instruction to the processors.
  - CMSIS
    - #defines __IO volatile
    - When reading CMSIS register definitions, a prefix of __IO means volatile
- ARM Cortex: __DBM(), __DSB(), __ISB()
  - Data Memory Barrier, Data Syncrhronization Barrier, Instruction Syncrhonization Barrier
    - Rarely Needed on the ARM Cortex-M
  - Instructions to the processor, not the compiler
  - See http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.dui0489c/CIHGHHIE.html

Important embedded c keywords

- const
  - const on the input to blocks of code
    - is not a contract that the input will not change at some time
    - Is a contract that the function will not change the input
  - const on global variables and local static variables places initial values for the variables in ROM
- A constant pointer to a volatile register
  - uint32_t volatile * const registerPtr = (uint32_t *) 0x2000000;
- A pointer to a constant and volatile register
  - uint32_t const volatile * ptr = (uint32_t *) 0x2000000;
- I like this description.
  - http://embeddedgurus.com/barr-code/2012/01/combining-cs-volatile-and-const-keywords/

Thoughts

Low level programming is moving the right bits to the right spots at the right time

Bits start out in flash (rom) and then move to cpu registers, ram, peripherals, to external interfaces (gpios, LEDs, serial, Ethernet).

We’ll walk through how to get the right bits in the right spots.

Useful gcc and make options

- Warn about all avoidable and questionable constructions
  - -Wall
- Make all warnings into errors
  - -Werror
- I like -pedantic, but it is too strict for many libraries
- Verbose Mode – Print out gcc's version number and report the final form of the include path
  - -v
- Print the name of each header file used.
  - -H

Useful gcc and make options

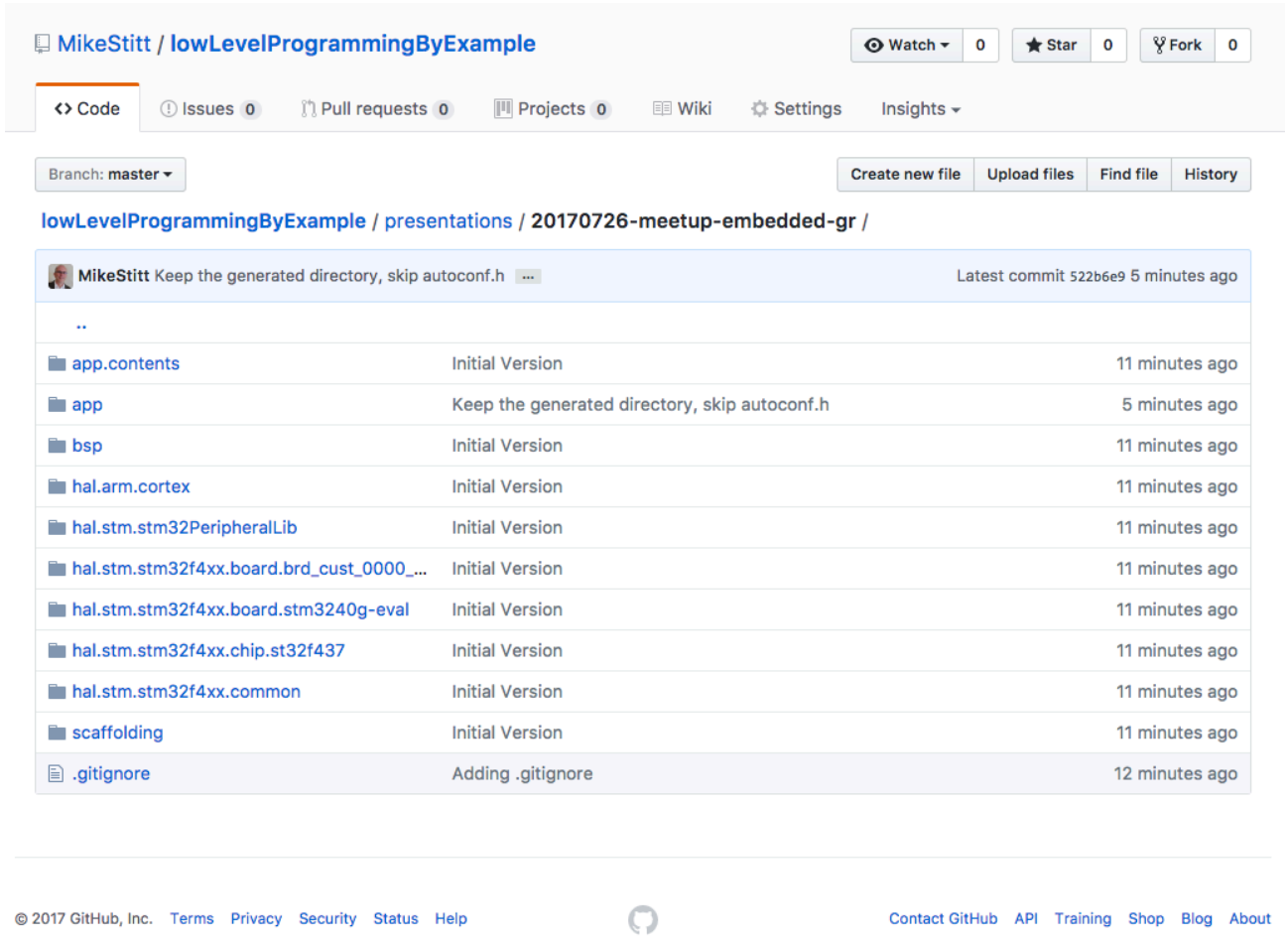
- Stop after the preprocessor
  - -E
- Inhibit the generation of line markers in the preprocessor output (useful with –P)
  - -P
- instead of the normal output, generate a list of ‘#define’ directives for all the macros defined during the execution of the preprocessor, including predefined macros. This gives you a way of finding out what is predefined in your version of the preprocessor.
  - arm-none-eabi-gcc -dM -E - < /dev/null

Useful gcc and make options

- Make - quiet down the build so that errors pop-up
  - Prefix make lines with $(Q)
  - Options to quiet the build
    - make Q=@
  - Options to unquiet the build
    - make Q=#@

- See
  https://github.com/MikeStitt/lowLevelProgrammingByExample/blob/master/presentations/20170726-meetup-embedded-gr/scaffolding/MakeToolDefs#L86

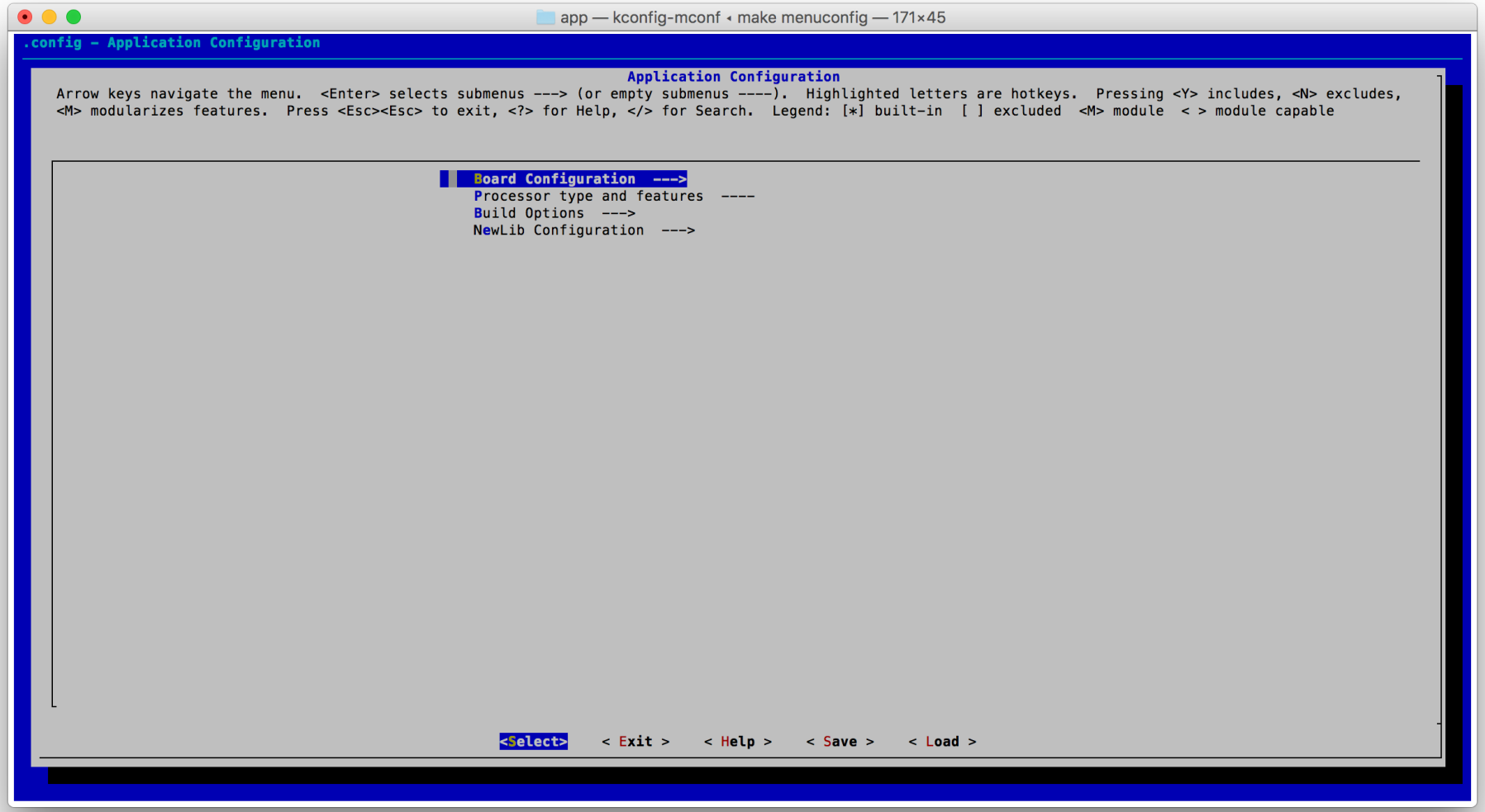
Example source code



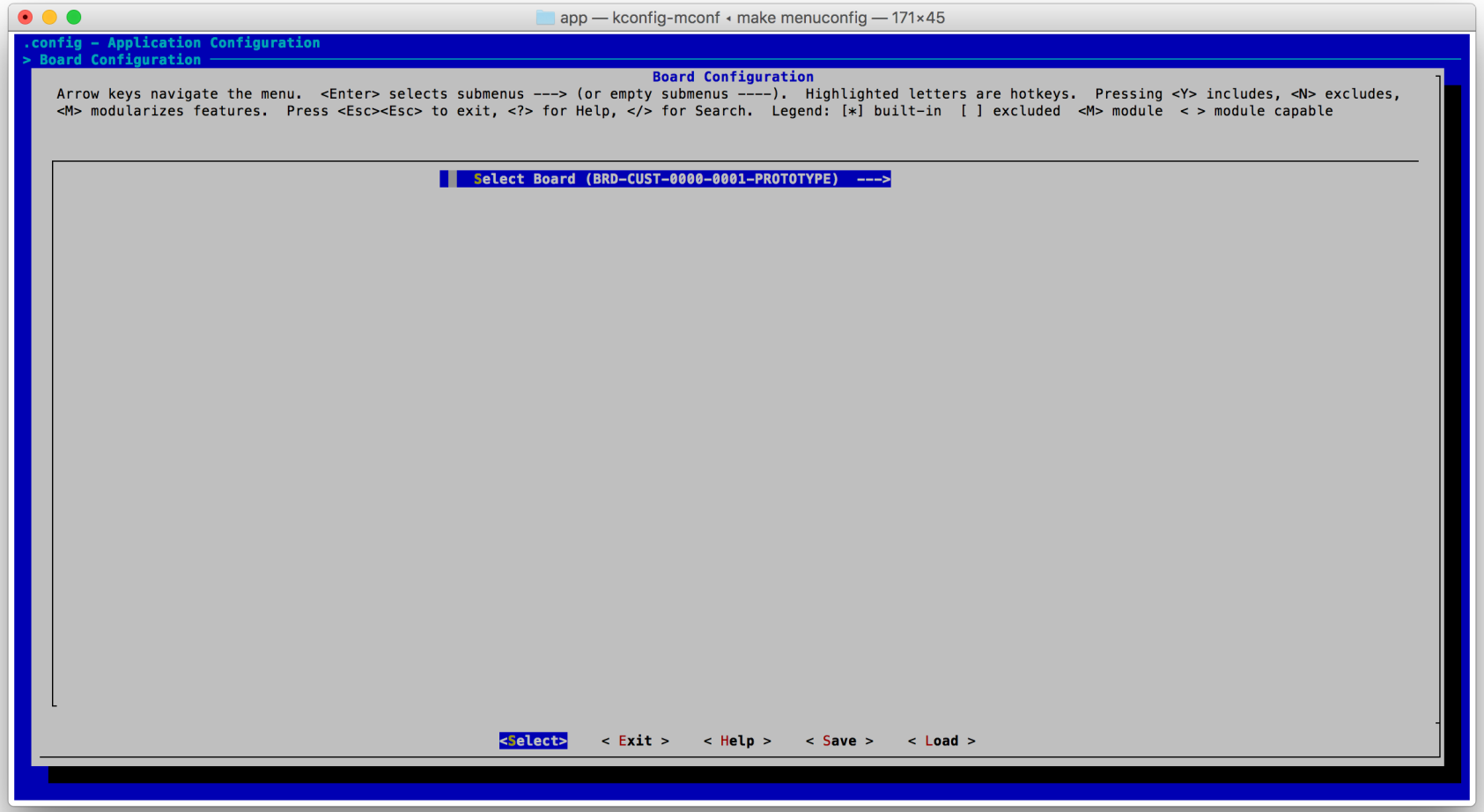
https://github.com/MikeStitt/lowLevelProgrammingByExample/tree/master/presentations/20170726-meetup-embedded-gr

How to build

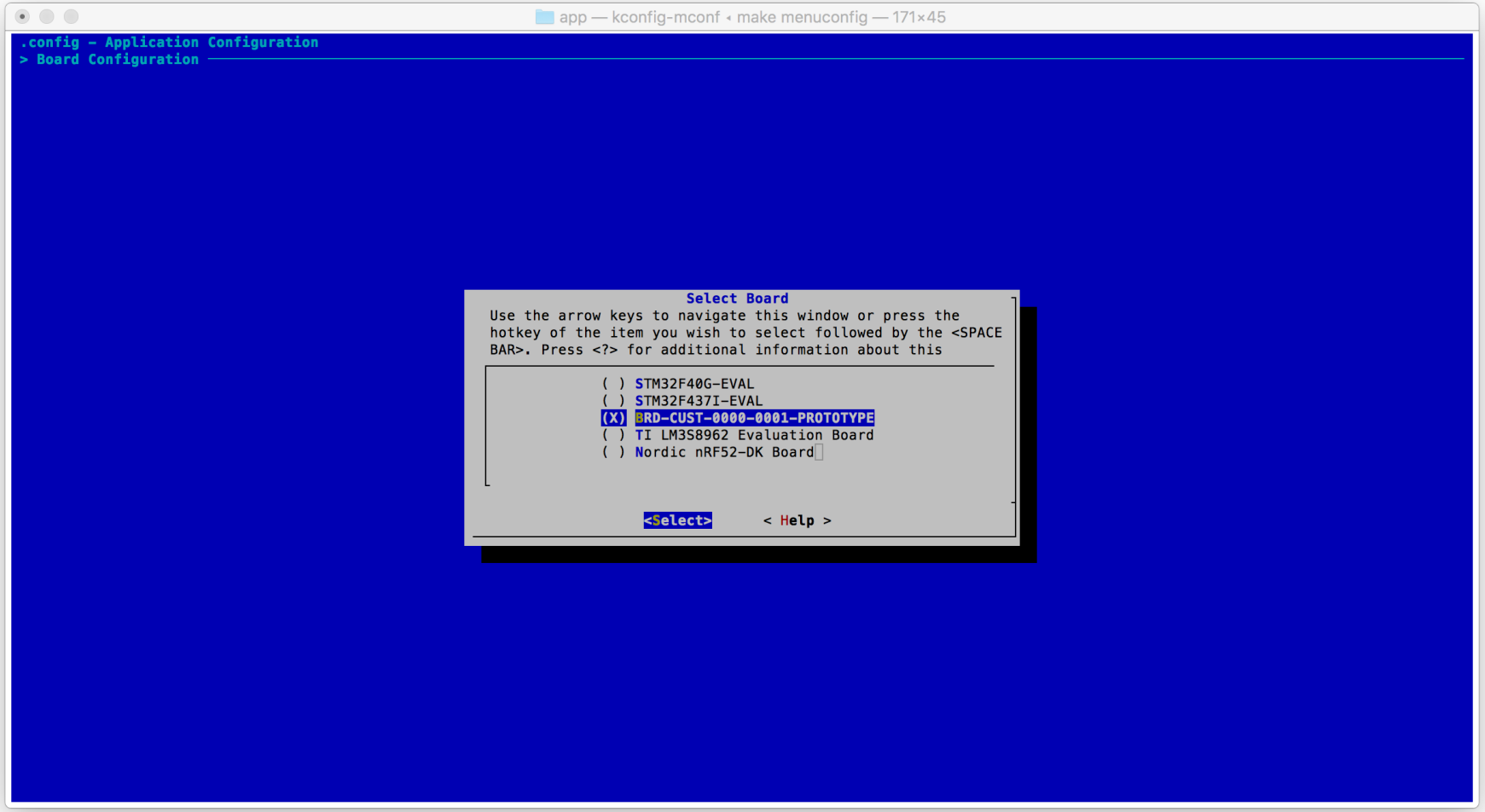
```
cd lowLevelProgrammingByExample/presentations/20170726-meetup-embedded-gr
cd app
make menuconfig
```



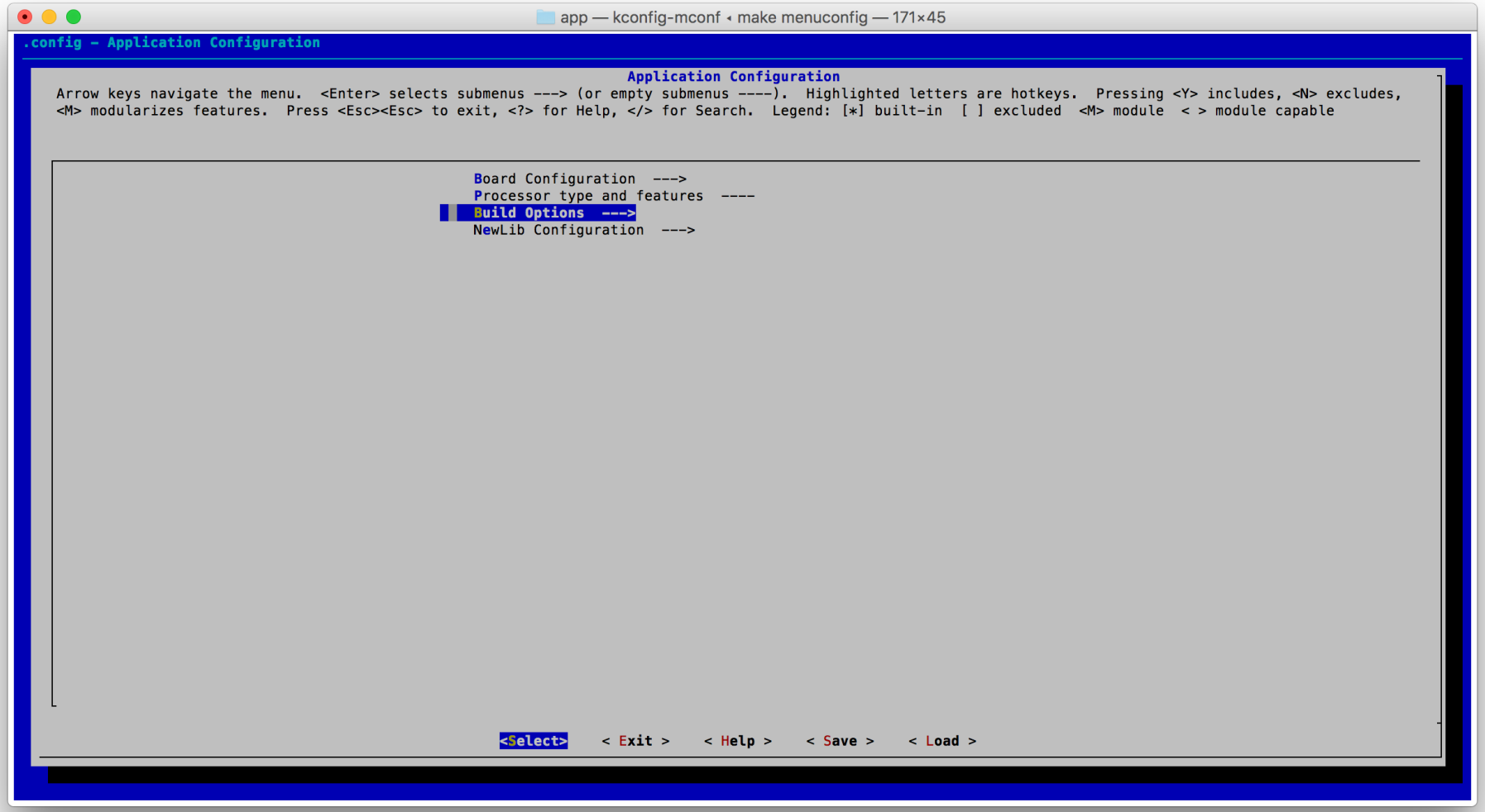
Board configuration



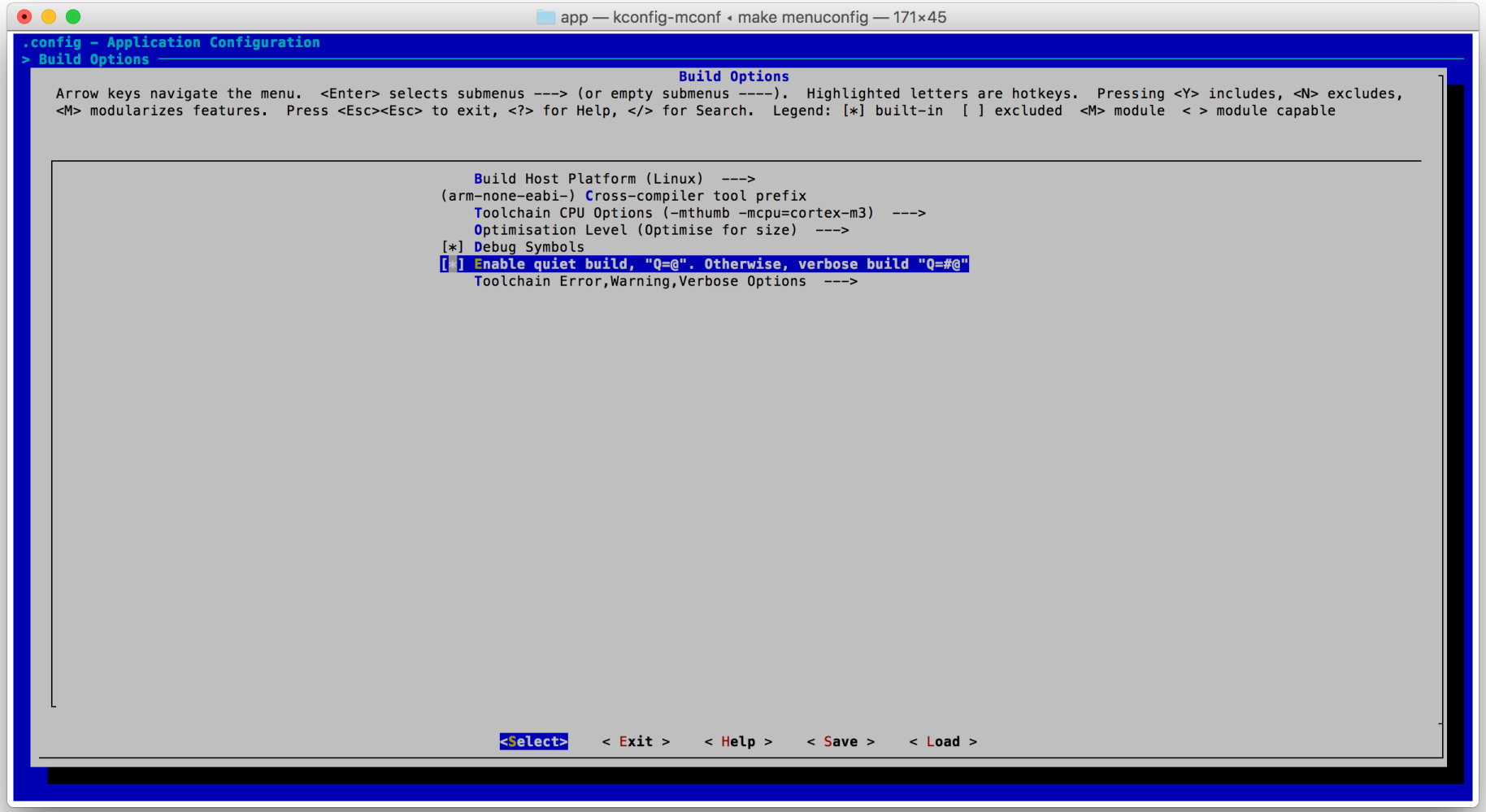
Select the board to be built



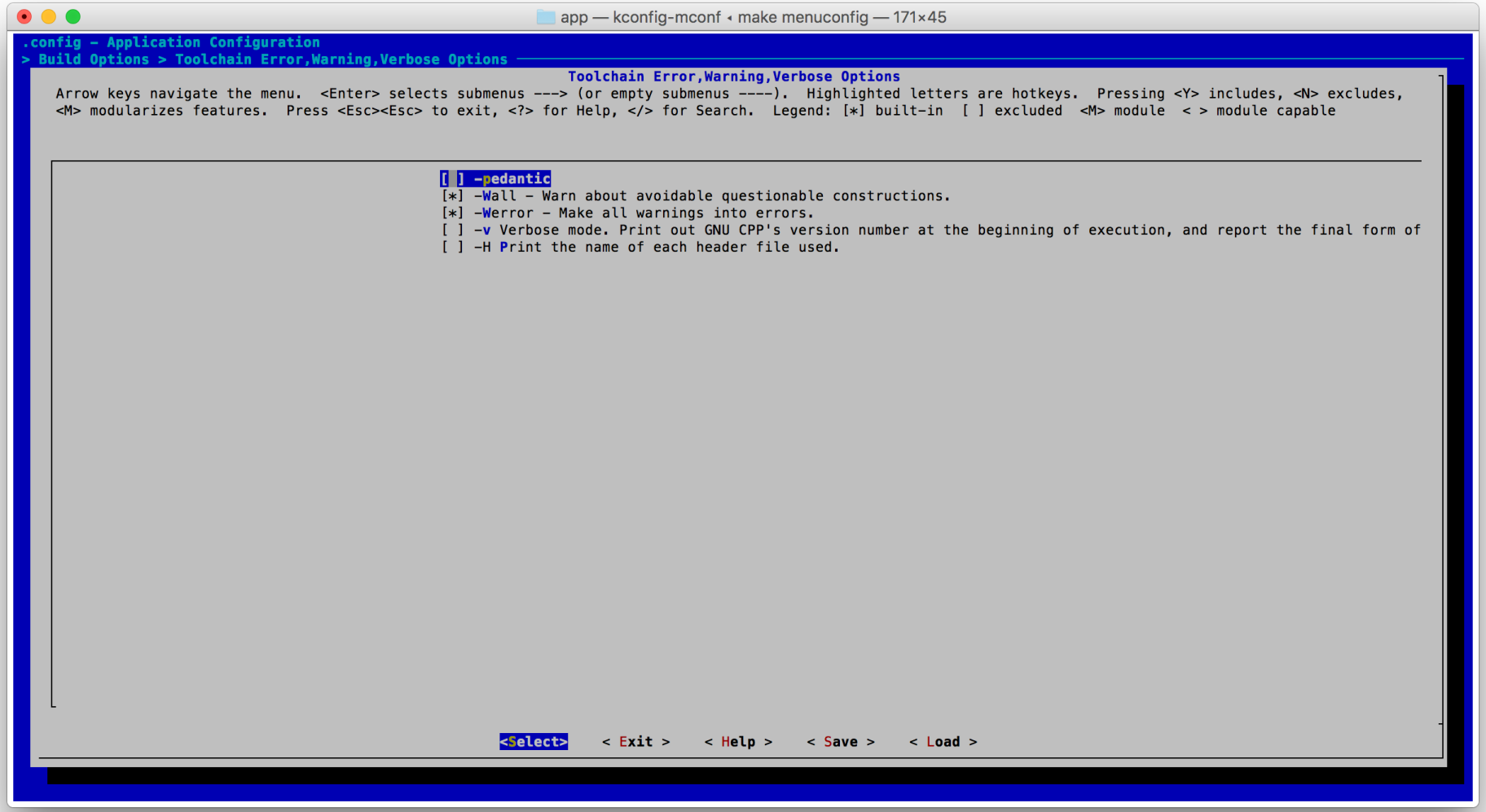
Back up to the main menu, going to build options



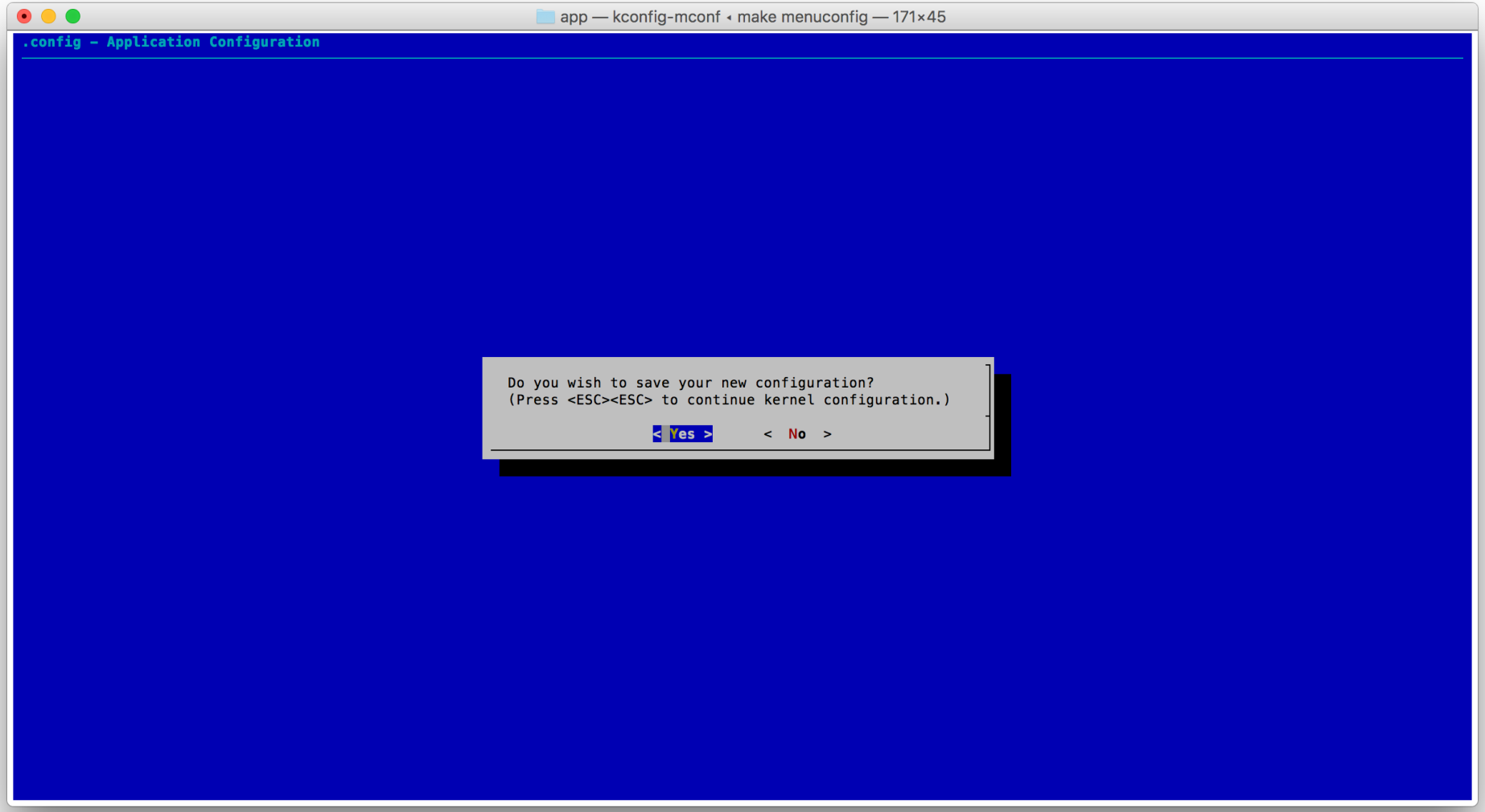
Build Options



Toolchain Error, Warning, Verbose Options



exit, exit, exit … yes to save



How to build

```
make help
```

app — -bash — 171×45

```
*** Execute 'make' to start the build or try 'make help'.

tomato:app mikes$ make help
+--Building include/generated/autoconf.h
+--Generating Dependency obj/app/../bsp/source/board/serialPortDriver.d
+--Generating Dependency obj/app/../bsp/source/board/gpioDriver.d
+--Generating Dependency obj/app/../bsp/source/newlib/syscalls.d
+--Generating Dependency obj/app/../bsp/source/startup/main.d
+--Generating Dependency obj/app/../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_rng.d
+--Generating Dependency obj/app/../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_cryp.d
+--Generating Dependency obj/app/../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_tim.d
+--Generating Dependency obj/app/../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_usart.d
+--Generating Dependency obj/app/../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_syscfg.d
+--Generating Dependency obj/app/../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_spi.d
+--Generating Dependency obj/app/../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_sdio.d
+--Generating Dependency obj/app/../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_rtc.d
+--Generating Dependency obj/app/../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_rcc.d
+--Generating Dependency obj/app/../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_pwr.d
+--Generating Dependency obj/app/../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_iwdg.d
+--Generating Dependency obj/app/../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_i2c.d
+--Generating Dependency obj/app/../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_gpio.d
+--Generating Dependency obj/app/../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_flash.d
+--Generating Dependency obj/app/../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_exti.d
+--Generating Dependency obj/app/../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_dma.d
+--Generating Dependency obj/app/../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_crc.d
+--Generating Dependency obj/app/../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_adc.d
+--Generating Dependency obj/app/../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/misc.d
+--Generating Dependency obj/app/../hal.stm.stm32PeripheralLib/Libraries/CMSIS/Device/ST/STM32F4xx/Source/Templates/system_stm32f4xx.d
+--Generating Dependency obj/app/../hal.stm.stm32f4xx.common/source/startup/halStartupBeforeMain.d
+--Generating Dependency obj/app/../hal.stm.stm32f4xx.common/source/board/serialPortPrivateDriver.d
+--Generating Dependency obj/app/../hal.stm.stm32f4xx.common/source/board/gpioDriverImplement.d
+--Generating Dependency obj/app/../hal.stm.stm32f4xx.chip.st32f437/source/startup/resetVectorTable.d
+--Generating Dependency obj/app/../hal.stm.stm32f4xx.board.brd_cust_0000_0001/source/board/serialPortConfig.d
+--Generating Dependency obj/app/../hal.stm.stm32f4xx.board.brd_cust_0000_0001/source/board/gpioConfig.d
+--Generating Dependency obj/app/../app.contents/source/startup/main.d

make help             - Print this message

make all              - Build the executable software - same as "make"
make clean            - Clean build products
make distclean        - Clean to repository distribution clean. Removes configuration files.
make printincludedirs - Show the -I include dir list.
make printdefines     - Show a list of default compiler defines.
make findinclude      - make findinclude INAME=*.h will list the .h files in the include dir list.
tomato:app mikes$
```

How to build

`make`

app — -bash — 171×45

```
make all              - Build the executable software - same as "make"
make clean            - Clean build products
make distclean        - Clean to repository distribution clean. Removes configuration files.
make printincludedirs - Show the -I include dir list.
make printdefines     - Show a list of default compiler defines.
make findinclude      - make findinclude INAME=*.h will list the .h files in the include dir list.
tomato:app mikes$ make
+--arm-none-eabi-gcc ../hal.arm.cortex/source/startup/armCortexCommonIntHandlers.c
+--arm-none-eabi-gcc -o ../app.contents/source/startup/main.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32f4xx.board.brd_cust_0000_0001/source/board/gpioConfig.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32f4xx.board.brd_cust_0000_0001/source/board/serialPortConfig.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32f4xx.chip.st32f437/source/startup/resetVectorTable.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32f4xx.common/source/board/gpioDriverImplement.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32f4xx.common/source/board/serialPortPrivateDriver.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32f4xx.common/source/startup/halStartupBeforeMain.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/CMSIS/Device/ST/STM32F4xx/Source/Templates/system_stm32f4xx.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/misc.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_adc.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_crc.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_dma.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_exti.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_flash.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_gpio.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_i2c.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_iwdg.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_pwr.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_rcc.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_rtc.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_sdio.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_spi.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_syscfg.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_usart.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_tim.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_cryp.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_rng.c
+--arm-none-eabi-gcc -o ../bsp/source/startup/main.c
+--arm-none-eabi-gcc -o ../bsp/source/newlib/syscalls.c
+--arm-none-eabi-gcc -o ../bsp/source/board/gpioDriver.c
+--arm-none-eabi-gcc -o ../bsp/source/board/serialPortDriver.c
------------------------------------------------------------------------+-
+--arm-none-eabi-ld obj/example.app.axf
------------------------------------------------------------------------+-
+--arm-none-eabi-objcopy -O binary obj/example.app.axf obj/example.app.bin
+--cp obj/example.app.bin deviceLoad/example.app.bin
tomato:app mikes$
```

How to build

```
make
```

app — -bash — 171×45

```
make all              - Build the executable software - same as "make"
make clean            - Clean build products
make distclean        - Clean to repository distribution clean. Removes configuration files.
make printincludedirs - Show the -I include dir list.
make printdefines     - Show a list of default compiler defines.
make findinclude      - make findinclude INAME=*.h will list the .h files in the include dir list.
tomato:app mikes$ make
+--arm-none-eabi-gcc ../hal.arm.cortex/source/startup/armCortexCommonIntHandlers.c
+--arm-none-eabi-gcc -o ../app.contents/source/startup/main.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32f4xx.board.brd_cust_0000_0001/source/board/gpioConfig.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32f4xx.board.brd_cust_0000_0001/source/board/serialPortConfig.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32f4xx.chip.st32f437/source/startup/resetVectorTable.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32f4xx.common/source/board/gpioDriverImplement.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32f4xx.common/source/board/serialPortPrivateDriver.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32f4xx.common/source/startup/halStartupBeforeMain.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/CMSIS/Device/ST/STM32F4xx/Source/Templates/system_stm32f4xx.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/misc.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_adc.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_crc.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_dma.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_exti.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_flash.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_gpio.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_i2c.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_iwdg.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_pwr.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_rcc.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_rtc.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_sdio.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_spi.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_syscfg.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_usart.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_tim.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_cryp.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_rng.c
+--arm-none-eabi-gcc -o ../bsp/source/startup/main.c
+--arm-none-eabi-gcc -o ../bsp/source/newlib/syscalls.c
+--arm-none-eabi-gcc -o ../bsp/source/board/gpioDriver.c
+--arm-none-eabi-gcc -o ../bsp/source/board/serialPortDriver.c
------------------------------------------------------------------+-
+--arm-none-eabi-ld obj/example.app.axf
------------------------------------------------------------------+-
+--arm-none-eabi-objcopy -O binary obj/example.app.axf obj/example.app.bin
+--cp obj/example.app.bin deviceLoad/example.app.bin
tomato:app mikes$
```

.axf has ROM and Symbols to be loaded by gdb

How to build

`make`

app — -bash — 171×45

```
make all              - Build the executable software - same as "make"
make clean            - Clean build products
make distclean        - Clean to repository distribution clean. Removes configuration files.
make printincludedirs - Show the -I include dir list.
make printdefines     - Show a list of default compiler defines.
make findinclude      - make findinclude INAME=*.h will list the .h files in the include dir list.
tomato:app mikes$ make
+--arm-none-eabi-gcc ../hal.arm.cortex/source/startup/armCortexCommonIntHandlers.c
+--arm-none-eabi-gcc -o ../app.contents/source/startup/main.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32f4xx.board.brd_cust_0000_0001/source/board/gpioConfig.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32f4xx.board.brd_cust_0000_0001/source/board/serialPortConfig.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32f4xx.chip.st32f437/source/startup/resetVectorTable.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32f4xx.common/source/board/gpioDriverImplement.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32f4xx.common/source/board/serialPortPrivateDriver.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32f4xx.common/source/startup/halStartupBeforeMain.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/CMSIS/Device/ST/STM32F4xx/Source/Templates/system_stm32f4xx.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/misc.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_adc.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_crc.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_dma.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_exti.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_flash.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_gpio.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_i2c.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_iwdg.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_pwr.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_rcc.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_rtc.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_sdio.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_spi.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_syscfg.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_usart.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_tim.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_cryp.c
+--arm-none-eabi-gcc -o ../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/src/stm32f4xx_rng.c
+--arm-none-eabi-gcc -o ../bsp/source/startup/main.c
+--arm-none-eabi-gcc -o ../bsp/source/newlib/syscalls.c
+--arm-none-eabi-gcc -o ../bsp/source/board/gpioDriver.c
+--arm-none-eabi-gcc -o ../bsp/source/board/serialPortDriver.c
------------------------------------------------------------------------+-
+--arm-none-eabi-ld obj/example.app.axf
------------------------------------------------------------------------+-
+--arm-none-eabi-objcopy -O binary obj/example.app.axf obj/example.app.bin
+--cp obj/example.app.bin deviceLoad/example.app.bin
tomato:app mikes$
```

.bin is the binary image without a load address or symbols. It's the exact size of the ROM image.

All of the build products go into ./obj

```
------------------------------------------------------------------------+-
+--arm-none-eabi-ld obj/example.app.axf
------------------------------------------------------------------------+-
+--arm-none-eabi-objcopy -O binary obj/example.app.axf obj/example.app.bin
+--cp obj/example.app.bin deviceLoad/example.app.bin
tomato:app mikes$ ls -Flas obj
total 528
  0 drwxr-xr-x  14 mikes  staff     476 Jul 24 22:55 ./
  0 drwxr-xr-x  11 mikes  staff     374 Jul 24 22:54 ../
  0 drwxr-xr-x   2 mikes  staff      68 Jul 24 22:55 app/
  0 drwxr-xr-x   3 mikes  staff     102 Jul 24 22:55 app.contents/
  0 drwxr-xr-x   3 mikes  staff     102 Jul 24 22:55 bsp/
184 -rwxr-xr-x   1 mikes  staff   90622 Jul 24 22:55 example.app.axf*
  8 -rwxr-xr-x   1 mikes  staff    3380 Jul 24 22:55 example.app.bin*
  8 -rw-r--r--   1 mikes  staff    1601 Jul 24 22:55 example.app.ld
328 -rw-r--r--   1 mikes  staff  165201 Jul 24 22:55 example.app.map
  0 drwxr-xr-x   3 mikes  staff     102 Jul 24 22:55 hal.stm.stm32PeripheralLib/
  0 drwxr-xr-x   3 mikes  staff     102 Jul 24 22:55 hal.stm.stm32f4xx.board.brd_cust_0000_0001/
  0 drwxr-xr-x   3 mikes  staff     102 Jul 24 22:55 hal.stm.stm32f4xx.chip.st32f437/
  0 drwxr-xr-x   3 mikes  staff     102 Jul 24 22:55 hal.stm.stm32f4xx.common/
  0 drwxr-xr-x   3 mikes  staff     102 Jul 24 22:55 source/
tomato:app mikes$ pwd
/Users/mikes/Documents/family/programming/lowLevelProgrammingByExample/presentations/20170726-meetup-embedded-gr/app
tomato:app mikes$
```

All of the build products go into ./obj

```
------------------------------------------------------------------------+-
+--arm-none-eabi-ld obj/example.app.axf
------------------------------------------------------------------------+-
+--arm-none-eabi-objcopy -O binary obj/example.app.axf obj/example.app.bin
+--cp obj/example.app.bin deviceLoad/example.app.bin
tomato:app mikes$ ls -Flas obj
total 528
  0 drwxr-xr-x  14 mikes  staff     476 Jul 24 22:55 ./
  0 drwxr-xr-x  11 mikes  staff     374 Jul 24 22:54 ../
  0 drwxr-xr-x   2 mikes  staff      68 Jul 24 22:55 app/
  0 drwxr-xr-x   3 mikes  staff     102 Jul 24 22:55 app.contents/
  0 drwxr-xr-x   3 mikes  staff     102 Jul 24 22:55 bsp/
184 -rwxr-xr-x   1 mikes  staff   90622 Jul 24 22:55 example.app.axf*
  8 -rwxr-xr-x   1 mikes  staff    3380 Jul 24 22:55 example.app.bin*
  8 -rw-r--r--   1 mikes  staff    1601 Jul 24 22:55 example.app.ld
328 -rw-r--r--   1 mikes  staff  165201 Jul 24 22:55 example.app.map
  0 drwxr-xr-x   3 mikes  staff     102 Jul 24 22:55 hal.stm.stm32PeripheralLib/
  0 drwxr-xr-x   3 mikes  staff     102 Jul 24 22:55 hal.stm.stm32f4xx.board.brd_cust_0000_0001/
  0 drwxr-xr-x   3 mikes  staff     102 Jul 24 22:55 hal.stm.stm32f4xx.chip.st32f437/
  0 drwxr-xr-x   3 mikes  staff     102 Jul 24 22:55 hal.stm.stm32f4xx.common/
  0 drwxr-xr-x   3 mikes  staff     102 Jul 24 22:55 source/
tomato:app mikes$ pwd
/Users/mikes/Documents/family/programming/lowLevelProgrammingByExample/presentations/20170726-meetup-embedded-gr/app
tomato:app mikes$
```

This program is just 3380 bytes.

All of the build products go into ./obj

```
------------------------------------------------------------------------+-
+--arm-none-eabi-ld obj/example.app.axf
------------------------------------------------------------------------+-
+--arm-none-eabi-objcopy -O binary obj/example.app.axf obj/example.app.bin
+--cp obj/example.app.bin deviceLoad/example.app.bin
tomato:app mikes$ ls -Flas obj
total 528
  0 drwxr-xr-x  14 mikes  staff     476 Jul 24 22:55 ./
  0 drwxr-xr-x  11 mikes  staff     374 Jul 24 22:54 ../
  0 drwxr-xr-x   2 mikes  staff      68 Jul 24 22:55 app/
  0 drwxr-xr-x   3 mikes  staff     102 Jul 24 22:55 app.contents/
  0 drwxr-xr-x   3 mikes  staff     102 Jul 24 22:55 bsp/
184 -rwxr-xr-x   1 mikes  staff   90622 Jul 24 22:55 example.app.axf*
  8 -rwxr-xr-x   1 mikes  staff    3380 Jul 24 22:55 example.app.bin*
  8 -rw-r--r--   1 mikes  staff    1601 Jul 24 22:55 example.app.ld
328 -rw-r--r--   1 mikes  staff  165201 Jul 24 22:55 example.app.map
  0 drwxr-xr-x   3 mikes  staff     102 Jul 24 22:55 hal.stm.stm32PeripheralLib/
  0 drwxr-xr-x   3 mikes  staff     102 Jul 24 22:55 hal.stm.stm32f4xx.board.brd_cust_0000_0001/
  0 drwxr-xr-x   3 mikes  staff     102 Jul 24 22:55 hal.stm.stm32f4xx.chip.st32f437/
  0 drwxr-xr-x   3 mikes  staff     102 Jul 24 22:55 hal.stm.stm32f4xx.common/
  0 drwxr-xr-x   3 mikes  staff     102 Jul 24 22:55 source/
tomato:app mikes$ pwd
/Users/mikes/Documents/family/programming/lowLevelProgrammingByExample/presentations/20170726-meetup-embedded-gr/app
tomato:app mikes$
```

Debug symbols and load addresses balloon the debugger load file to 90622 bytes.

We had to tell the linker to put the right bits in the right spots in flash (ROM), and to use the right bits in the right spots in SRAM.

app — less obj/example.app.ld — 139×70

```
MEMORY
{
    FLASH_READ (rx) : ORIGIN = 0x00000000, LENGTH = 0x00070000
    RAM (rw) : ORIGIN = ((0x20000000 + 0x000011C0) + 0x00000800), LENGTH = (0x20030000 - ((0x20000000 + 0x000011C0) + 0x00000800))
    RAM_KEEP (rw) : ORIGIN = 0x20000000, LENGTH = 0x000011C0
    RAM_RESET_STACK(rw) : ORIGIN = (0x20000000 + 0x000011C0), LENGTH = 0x00000800
}
ENTRY(Reset_Handler)
_Min_Heap_Size = 0x0;
SECTIONS
{
    .text :
    {
        _startIsrVector = .;
        KEEP(*(.isr_vector))
        . = 0x200;
        *(.progHeader)
        *(.text*)
        *(.rodata*)
        *(.glue_7)
        *(.glue_7t)
        *(.eh_frame)
        KEEP (*(.init))
        KEEP (*(.fini))
    } > FLASH_READ
    .ARM :
    {
        . = ALIGN(4);
        __exidx_start = .;
        *(.ARM.exidx* .gnu.linkonce.armexidx.*)
        __exidx_end = .;
    } > FLASH_READ
    . = ALIGN(4);
      _startInitSourceData = .;
     .ram_keep (NOLOAD) :
     {
         *(.ram_keep)
     } > RAM_KEEP
     .reset_stack (NOLOAD) :
     {
         *(.reset_stack)
     } > RAM_RESET_STACK
    .data : AT ( _startInitSourceData )
    {
        . = ALIGN(4);
        _startInitDestinationData = .;
        *( data*)
```

example.app.ld provided instructions to the linker (arm-none-eabi-ld) where to put the bits in the program.

ldCfg/configLoadModule.ld.preProc is configured by memoryMap.h.

This way C code can know about the top level memory configuration.

lowLevelProgrammingByExample / presentations / 20170726-meetup-embedded-gr / app / ldCfg /
configLoadModule.ld.preProc


MikeStitt Initial Version — 9ba66ae an hour ago

1 contributor

107 lines (87 sloc) | 2.2 KB — Raw | Blame | History

```
1   #include "memoryMap.h"
2
3   MEMORY
4   {
5       FLASH_READ (rx)     : ORIGIN = SW_FLASH_ADR, LENGTH = SW_FLASH_LEN
6       RAM (rw)            : ORIGIN = SW_RAM_ADR, LENGTH = SW_RAM_LEN
7   #ifdef BUILD_AN_APP
8       RAM_KEEP (rw)       : ORIGIN = RAM_KEEP_THROUGH_RESET_ADR, LENGTH = RAM_KEEP_THROUGH_RESET_LEN
9       RAM_RESET_STACK(rw) : ORIGIN = RAM_APP_RESET_STACK, LENGTH = RAM_APP_RESET_STACK_LEN
10  #endif
11  }
12
13  ENTRY(Reset_Handler)
14
15  //  _Min_Heap_Size is the heap size for newLib. See include/FreeRTOSconfig.h for the FreeRTOS heap.
16
17  _Min_Heap_Size = 0x0;
18
19  SECTIONS
```

MikeStitt Initial Version 9ba66ae an hour ago

1 contributor

32 lines (20 sloc) | 789 Bytes

```
1   #ifndef __MEMORY_MAP_H
2   #define __MEMORY_MAP_H
3
4   #define APP_ADR      0x00000000
5   #define APP_LEN      0x00070000
6   #define APP_RN       4
7
8   #define VECTORS_OFF         0x00000000
9   #define VECTORS_LEN         0x00000200
10
11  #define RAM_KEEP_THROUGH_RESET_ADR 0x20000000
12  #define RAM_KEEP_THROUGH_RESET_LEN 0x000011C0
13
14  #define RAM_APP_RESET_STACK         (RAM_KEEP_THROUGH_RESET_ADR + RAM_KEEP_THROUGH_RESET_LEN)
15  #define RAM_APP_RESET_STACK_LEN     0x00000800
16
17  #define RAM_ADR                     (RAM_APP_RESET_STACK + RAM_APP_RESET_STACK_LEN)
18  #define RAM_LEN                     (0x20030000 - RAM_ADR)
19
20  #define RAM_RN                      6
21
```

The c preprocessor
does textual substitution to define
the math to place the
top level memory regions of the processor.

The gnu linkers, ld or in this case arm-none-eabi-ld are able to do simple math to calculate the ORIGIN and LENGTH of the MEMORY regions.

app — less obj/example.app.ld — 139×70

```
MEMORY
{
    FLASH_READ (rx) : ORIGIN = 0x00000000, LENGTH = 0x00070000
    RAM (rw) : ORIGIN = ((0x20000000 + 0x000011C0) + 0x00000800), LENGTH = (0x20030000 - ((0x20000000 + 0x000011C0) + 0x00000800))
    RAM_KEEP (rw) : ORIGIN = 0x20000000, LENGTH = 0x000011C0
    RAM_RESET_STACK(rw) : ORIGIN = (0x20000000 + 0x000011C0), LENGTH = 0x00000800
}
ENTRY(Reset_Handler)
_Min_Heap_Size = 0x0;
SECTIONS
{
    .text :
    {
        _startIsrVector = .;
        KEEP(*(.isr_vector))
        . = 0x200;
        *(.progHeader)
        *(.text*)
        *(.rodata*)
        *(.glue_7)
        *(.glue_7t)
```

We tell the linker about 4 top level regions of memory that correspond to spots on the chip **where the processor can access** the type of memory we need.

FLASH_READ (rx) – The permanent, sort-of-read-only memory for instructions and constants.

RAM (rw) – General purpose read/write memory.

RAM_KEEP (rw) – General purpose read/write memory, where we don't intend to follow the C language initialization rules.

RAM_RESET_STACK (rw) – General purpose read/write memory, which we want to use for the processor stack.

app — less obj/exar

```
MEMORY
{
    FLASH_READ (rx) : ORIGIN = 0x00000000, LENGTH = 0x00070000
    RAM (rw) : ORIGIN = ((0x20000000 + 0x000011C0) + 0x00000800), LENG
    RAM_KEEP (rw) : ORIGIN = 0x20000000, LENGTH = 0x000011C0
    RAM_RESET_STACK(rw) : ORIGIN = (0x20000000 + 0x000011C0), LENGTH =
}
ENTRY(Reset_Handler)
_Min_Heap_Size = 0x0;
SECTIONS
{
    .text :
    {
        _startIsrVector = .;
        KEEP(*(.isr_vector))
        . = 0x200;
        *(.progHeader)
        *(.text*)
        *(.rodata*)
        *(.glue_7)
        *(.glue_7t)
        *(.eh_frame)
        KEEP (*(.init))
        KEEP (*(.fini))
    } > FLASH_READ
    .ARM :
    {
```

We place a ”section” of memory labeled “.reset_stack” at the start of RESET_STACK at address 0x200011C0.

We did this because we’re going to use the reset stack for ‘C’ code while we initialize other RAM.

```
    KEEP (*(.fini))
} > FLASH_READ
.ARM :
{
    . = ALIGN(4);
    __exidx_start = .;
    *(.ARM.exidx* .gnu.linkonce.armexidx.*)
    __exidx_end = .;
} > FLASH_READ
. = ALIGN(4);
  _startInitSourceData = .;
 .ram_keep (NOLOAD) :
 {
     *(.ram_keep)
 } > RAM_KEEP
 .reset_stack (NOLOAD) :
 {
     *(.reset_stack)
 } > RAM_RESET_STACK
.data : AT ( _startInitSourceData )
{
    . = ALIGN(4);
    _startInitDestinationData = .;
    *(.data*)
    . = ALIGN(4);
    _endInitDestinationData = .;
} > RAM
.bss (NOLOAD):
{
    . = ALIGN(4);
    _startBss = .;
    *(.bss*)
    *(COMMON)
    . = ALIGN(4);
    _endBss = .;
```

lowLevelProgrammingByExample / presentations / 20170726-meetup-embedded-gr / hal.stm.stm32f4xx.chip.st32f437 / source / startup / **resetVectorTable.c**

MikeStitt Initial Version 9ba66ae 2 days ago

1 contributor

288 lines (279 sloc) | 14.8 KB

Raw Blame History

```
1   #include "app/commonHeaders.h"
2   #include "startup/armCortexCommonIntHandlers.h"
3
4   void WEAK Default_Handler_Relay(void)
5   {
6     Default_Handler();
7   }
8
9   void WEAK WWDG_IRQHandler(void);
10  void WEAK PVD_IRQHandler(void);
11  void WEAK TAMP_STAMP_IRQHandler(void);
12  void WEAK RTC_WKUP_IRQHandler(void);
```

The interrupt vector table is different is slightly different for each ARM Cortex System-On-Chip device. But common interrupts like “reset” are always in the same place.

lowLevelProgrammingByExample / presentations / 20170726-meetup-embedded-gr / hal.stm.stm32f4xx.chip.st32f437 / source / startup / **resetVectorTable.c**

```
169  #pragma weak OTG_HS_IRQHandler = Default_Handler_Relay
170  #pragma weak DCMI_IRQHandler = Default_Handler_Relay
171  #pragma weak CRYP_IRQHandler = Default_Handler_Relay
172  #pragma weak HASH_RNG_IRQHandler = Default_Handler_Relay
173  #pragma weak FPU_IRQHandler = Default_Handler_Relay
174
175
176  RESET_STACK_SECTION int resetStack[RAM_APP_RESET_STACK_LEN / sizeof(int)];
177
178
179  ISR_VECTOR_SECTION void (* WEAK const vectors[])(void) =
180  {
181    (void (*)(void))((unsigned long)resetStack+sizeof(resetStack)),
182                                        /* 0000 - The initial stack pointer */
183          Reset_Handler,                /* 0004 - Reset Handler */
184          NMI_Handler,                  /* 0008 - NMI Handler */
185          HardFault_Handler,            /* 000C - Hard Fault Handler */
186          MemManage_Handler,            /* 0010 - Mpu Fault Handler */
187          BusFault_Handler,             /* 0014 - Bus Fault Handler */
188          UsageFault_Handler,           /* 0018 - Usage Fault Handler */
189          Default_Handler,              /* 001C - Reserved */
190          Default_Handler,              /* 0020 - Reserved */
191          Default_Handler,              /* 0024 - Reserved */
192          Default_Handler,              /* 0028 - Reserved */
193          SVC_Handler,                  /* 002C - SVCall Handler */
194          DebugMon_Handler,             /* 0030 - Debug Monitor Handler */
195          Default_Handler,              /* 0034 - Reserved */
196          PendSV_Handler,               /* 0038 - PendSV Handler */
197          SysTick_Handler,              /* 003C - SysTick Handler */
198
```

We're defining

the reset stack
and
interrupt vector table

in "C"!

```
169  #pragma weak OTG_HS_IRQHandler = Default_Handler_Relay
170  #pragma weak DCMI_IRQHandler = Default_Handler_Relay
171  #pragma weak CRYP_IRQHandler = Default_Handler_Relay
172  #pragma weak HASH_RNG_IRQHandler = Default_Handler_Relay
173  #pragma weak FPU_IRQHandler = Default_Handler_Relay
174
175
176  RESET_STACK_SECTION int resetStack[RAM_APP_RESET_STACK_LEN / sizeof(int)];
177
178
179  ISR_VECTOR_SECTION void (* WEAK const vectors[])(void) =
180  {
181    (void (*)(void))((unsigned long)resetStack+sizeof(resetStack)),
182                                        /* 0000 - The initial stack pointer */
183          Reset_Handler,                /* 0004 - Rese
184          NMI_Handler,                  /* 0008 - NMI
185          HardFault_Handler,            /* 000C - Hard
186          MemManage_Handler,            /* 0010 - Mpu
187          BusFault_Handler,             /* 0014 - Bus
188          UsageFault_Handler,           /* 0018 - Usag
189          Default_Handler,              /* 001C - Rese
190          Default_Handler,              /* 0020 - Rese
191          Default_Handler,              /* 0024 - Rese
192          Default_Handler,              /* 0028 - Rese
193          SVC_Handler,                  /* 002C - SVCa
194          DebugMon_Handler,             /* 0030 - Debu
195          Default_Handler,              /* 0034 - Rese
196          PendSV_Handler,               /* 0038 - Pend
197          SysTick_Handler,              /* 003C - SysT
198
```

ISR_VECTOR_SECTION is

#define ISR_VECTOR_SECTION __attribute__ ((section(".isr_vector")))

We've tied the "C" code to the linker control file to tell the linker to place this table at address 0.

lowLevelProgrammingByExample / presentations / 20170726-meetup-embedded-gr / hal.stm.stm32f4xx.chip.st32f437 / source / startup / **resetVectorTable.c**

```
169  #pragma weak OTG_HS_IRQHandler = Default_Handler_Relay
170  #pragma weak DCMI_IRQHandler = Default_Handler_Relay
171  #pragma weak CRYP_IRQHandler = Default_Handler_Relay
172  #pragma weak HASH_RNG_IRQHandler = Default_Handler_Relay
173  #pragma weak FPU_IRQHandler = Default_Handler_Relay
174
175
176  RESET_STACK_SECTION int resetStack[RAM_APP_RESET_STACK_LEN / sizeof(int)];
177
178
179  ISR_VECTOR_SECTION void (* WEAK const vectors[])(void) =
180  {
181    (void (*)(void))((unsigned long)resetStack+sizeof(resetStack)),
182                                        /* 0000 - The initial stack pointer */
183          Reset_Handler,
184          NMI_Handler,
185          HardFault_Handler,
186          MemManage_Handler,
187          BusFault_Handler,
188          UsageFault_Handler,
189          Default_Handler,
190          Default_Handler,
191          Default_Handler,
192          Default_Handler,
193          SVC_Handler,
194          DebugMon_Handler,
195          Default_Handler,
196          PendSV_Handler,
197          SysTick_Handler,
198
```

RESET_STACK_SECTION is

```
#define RESET_STACK_SECTION __attribute__ ((section(".reset_stack")))
```

Yeah! The linker now knows that resetStack is at 0x200011C0. This is good. We need our stack to be in RAM which is in 0x2000mmmm.

lowLevelProgrammingByExample / presentatio

source / startup / **resetVectorTable.c**

```
169   #pragma weak OTG_HS_IRQHandler = D
170   #pragma weak DCMI_IRQHandler = Def
171   #pragma weak CRYP_IRQHandler = Def
172   #pragma weak HASH_RNG_IRQHandler =
173   #pragma weak FPU_IRQHandler = Defa
174
175
176   RESET_STACK_SECTION int resetStack
177
178
179   ISR_VECTOR_SECTION void (* WEAK const vectors[])(void) =
180   {
181     (void (*)(void))((unsigned long)resetStack+sizeof(resetStack)),
182                                       /* 0000 - The initial stack pointer */
183           Reset_Handler,              /* 0004 - Reset Handler */
184           NMI_Handler,                /* 0008 - NMI Handler */
185           HardFault_Handler,          /* 000C - Hard Fault Handler */
186           MemManage_Handler,          /* 0010 - Mpu Fault Handler */
187           BusFault_Handler,           /* 0014 - Bus Fault Handler */
188           UsageFault_Handler,         /* 0018 - Usage Fault Handler */
189           Default_Handler,            /* 001C - Reserved */
190           Default_Handler,            /* 0020 - Reserved */
191           Default_Handler,            /* 0024 - Reserved */
192           Default_Handler,            /* 0028 - Reserved */
```

Some fancy math to place a pointer to the the address just past the end `resetStack` in the first uint32_t in the `vectors` table. It's ok. The CPU subtracts 4 prior to writing to the stack.

lowLevelProgrammingByExample / presentatio

source / startup / **resetVectorTable.c**

```
169   #pragma weak OTG_HS_IRQHandler = D
170   #pragma weak DCMI_IRQHandler = Def
171   #pragma weak CRYP_IRQHandler = Def
172   #pragma weak HASH_RNG_IRQHandler =
173   #pragma weak FPU_IRQHandler = Defa
174
175
176   RESET_STACK_SECTION int resetStack
177
178
179   ISR_VECTOR_SECTION void (* WEAK const vectors[])(void) =
180   {
181     (void (*)(void))((unsigned long)resetStack+sizeof(resetStack)),
182                                     /* 0000 - The initial stack pointer */
183           Reset_Handler,            /* 0004 - Reset Handler */
184           NMI_Handler,              /* 0008 - NMI Handler */
185           HardFault_Handler,        /* 000C - Hard Fault Handler */
186           MemManage_Handler,        /* 0010 - Mpu Fault Handler */
187           BusFault_Handler,         /* 0014 - Bus Fault Handler */
188           UsageFault_Handler,       /* 0018 - Usage Fault Handler */
189           Default_Handler,          /* 001C - Reserved */
190           Default_Handler,          /* 0020 - Reserved */
191           Default_Handler,          /* 0024 - Reserved */
192           Default_Handler,          /* 0028 - Reserved */
```

The second uint32_t in the `vectors` table is a pointer to the `Reset_Handler` function.

We have placed a vector table and reset stack at the right addresses in memory.

We've loaded the first 2 entries in the vector table with pointers to the stack and the reset handler.

We can release the KRAKEN.

lowLevelProgrammingByExample / presentations / 20170726-meetup-embedded-gr / hal.arm.cortex / source / startup /

armCortexCommonIntHandlers.c

MikeStitt Initial Version — 9ba66ae 2 days ago

1 contributor

210 lines (175 sloc) | 6.81 KB

```
1   #include "app/commonHeaders.h"
2   #include "startup/armCortexCommonIntHandlers.h"
3   #include "startup/main.h"
4   #include "startup/halStartupBeforeMain.h"
5
6   #include <stdint.h>
7
8   //  ------------------------------------------------------------
9   /// Entry point upon processor reset release.
10  ///
11  /// When reset is released, the processor loads the stack pointer
12  /// with the value at vectors[0] and the program counter with
13  /// the value at vectors[1] to start execution at Reset_Handler.
14  //  ------------------------------------------------------------
15  void WEAK INTERRUPT Reset_Handler(void)
16  {
17      extern unsigned long _startInitSourceData;
18      extern unsigned long _startInitDestinationData;
19      extern unsigned long  endInitDestinationData;
```

This reset handler can be common to all ARM Cortex-M processors.

It's in "C".

Now days, when interviewing, talking about low level programming in assembly is not the best. Just saying.

lowLevelProgrammingByExample / presentations / 20170726-meetup-embedded-gr / hal.arm.cortex / source / startup /

armCortexCommonIntHandlers.c

MikeStitt Initial Version 9ba66ae 2 days ago

1 contributor

```
15  void WEAK INTERRUPT Reset_Handler(void)
16  {
17      extern unsigned long _startInitSourceData;
18      extern unsigned long _startInitDestinationData;
19      extern unsigned long _endInitDestinationData;
20      extern unsigned long _startBss;
21      extern unsigned long _endBss;
22
23      extern void SystemInit(void);
24      extern void __libc_init_array(void);
25
26      unsigned long *destPtr;
27      unsigned long *srcPtr;
28
29      /// Copy the data segment initializers from flash to SRAM in ROM mode.
30      /// When (&_startInitSourceData != &_startInitDestinationData) we are linked
31      /// to boot from ROM and we need to copy.  Otherwise we are linked to boot
32      /// from RAM and we don't need to copy.
33
34      if (&_startInitSourceData != &_startInitDestinationData) {
35          srcPtr = (unsigned long *)&_startInitSourceData;
36          for (destPtr = &_startInitDestinationData;
37               destPtr < &_endInitDestinationData;) {
38              *(destPtr++) = *(srcPtr++);
39          }
40      }
```

The “C” language allows global variables, global statics, and local statics to take non-zero initial values.

Copy the initial values out of Flash (ROM) at addresses defined by the link control and into RAM to addresses controlled by the linker.

lowLevelProgrammingByExample / presentations / 20170726-meetup-embedded-gr / hal.arm.cortex / source / startup /
armCortexCommonIntHandlers.c

```
41
42      /// Zero fill the bss segment in RAM.
43      for (destPtr = &_startBss; destPtr < &_endBss; destPtr++) {
44          *destPtr = 0;
45      }
46
47      /// Start the CPU (PLL, FLASH)
48      SystemInit();
49
50      /// Initialize newLib.
51      __libc_init_array();
52
53      /// The last couple of hardware things we
54      halStartupBeforeMain();
55
56      // Call the application's entry point.
57      main();
58
59      /// We should not get here.
60      while (1) {};
61  }
```

The “C” language requires global variables, global statics, and local statics that are not specifically initialized to default to zero. Historically, the compiler places these in the “.bss” section.

To conform (and not be cast out ☺) we zero the “.bss”.

Great News!

C code that uses global and static variables, but doesn’t call libraries, will now work.

The KRAKEN is getting stronger.

lowLevelProgrammingByExample / presentations / 20170726-meetup-embedded-gr / hal.arm.cortex / source / startup /
armCortexCommonIntHandlers.c

```
41
42      /// Zero fill the bss segment in RAM.
43      for (destPtr = &_startBss; destPtr < &_endBss; destPtr++) {
44          *destPtr = 0;
45      }
46
47      /// Start the CPU (PLL, FLASH)
48      SystemInit();
49
50      /// Initialize newLib.
51      __libc_init_array();
52
53      /// The last couple of hardware things we
54      halStartupBeforeMain();
55
56      // Call the application's entry point.
57      main();
58
59      /// We should not get here.
60      while (1) {};
61  }
```

Call the vendor specific initialization code to get the processor phase lock loop and clock up to full speed for the chip. Perhaps from a handful of MHz to 188 MHz.

Wait.

We weren’t initializing RAM at full speed?

Nope.

We do it in this order because `SystemInit()` accesses global variables.

lowLevelProgrammingByExample / presentations / 20170726-meetup-embedded-gr / hal.arm.cortex / source / startup /
armCortexCommonIntHandlers.c

```
41
42      /// Zero fill the bss segment in RAM.
43      for (destPtr = &_startBss; destPtr < &_endBss; destPtr++) {
44          *destPtr = 0;
45      }
46
47      /// Start the CPU (PLL, FLASH)
48      SystemInit();
49
50      /// Initialize newLib.
51      __libc_init_array();
52
53      /// The last couple of hardware things we
54      halStartupBeforeMain();
55
56      // Call the application's entry point.
57      main();
58
59      /// We should not get here.
60      while (1) {};
61  }
```

Initialize the newLib “C” library.

We can start calling some “C” library functions.

lowLevelProgrammingByExample / presentations / 20170726-meetup-embedded-gr / hal.arm.cortex / source / startup /
armCortexCommonIntHandlers.c

```
41
42      /// Zero fill the bss segment in RAM.
43      for (destPtr = &_startBss; destPtr < &_endBss; destPtr++) {
44          *destPtr = 0;
45      }
46
47      /// Start the CPU (PLL, FLASH)
48      SystemInit();
49
50      /// Initialize newLib.
51      __libc_init_array();
52
53      /// The last couple of hardware things we need to be intialzed before main can run.
54      halStartupBeforeMain();
55
56      // Call the application's entry point.
57      main();
58
59      /// We should not get here.
60      while (1) {};
61  }
```

A couple of house keeping hardware initializations that we need to do that are not part of the standard `SystemInit()` go here.

lowLevelProgrammingByExample / presentations / 20170726-meetup-embedded-gr / hal.arm.cortex / source / startup /
armCortexCommonIntHandlers.c

```
41
42      /// Zero fill the bss segment in RAM.
43      for (destPtr = &_startBss; destPtr < &_endBss; destPtr++) {
44          *destPtr = 0;
45      }
46
47      /// Start the CPU (PLL, FLASH)
48      SystemInit();
49
50      /// Initialize newLib.
51      __libc_init_array();
52
53      /// The last couple of hardware things we need to be intialzed before main can run.
54      halStartupBeforeMain();
55
56      // Call the application's entry point.
57      main();
58
59      /// We should not get here.
60      while (1) {};
61  }
```

We've made it to `main()` !!!!

`main()` should not return, but get stuck here to minimize confusion if it does.

lowLevelProgrammingByExample / presentations / 20170726-meetup-embedded-gr / app.contents / source / startup / main.c

MikeStitt Initial Version 9ba66ae 2 days ago

1 contributor

72 lines (55 sloc) | 1.48 KB Raw Blame History

```
1  #include "app/commonHeaders.h"
2  #include "startup/main.h"
3
4  void sendRepeatedSerialMessage(void)
5  {
6      static int index = 0;
7      char const* message = "Hello World!\n\rWhatttttt's ha
8
9      if (0==message[index]) {
10        index = 0;
11     }
12
13     serialPortWaitWhileFull(SERIAL0);
14     serialPortTxByte(SERIAL0, (uint8_t)message[index]);
15     index++;
16 }
17
18 int nextGpioState(int current)
19 {
20     int next=0;
```

Great news!

Our processor is running (Air Quotes) “application” code in main.c .

```
52  // This main does not have a WEAK prefix so it overrides the default main.
53  int main(void) {
54
55      gpioInit();
56      serialPortInit();
57
58      while (1) {
59          gCounter++;
60
61          toggleDiscretes();
62          sendRepeatedSerialMessage();
63      }
64
65      // An infinite loop for safety
66      while (1) {
67      }
68
69      // We can't get passed the infinite loop to here
70  }
```

A very simple low level application.

It toggles General Purpose IOs (GPIOs) and sends messages out a serial port.

lowLevelProgrammingByExample / presentations / 20170726-meetup-embedded-gr / app.contents / source / startup / **main.c**

```
52  // This main does not have a WEAK prefix so it overrides the default main.
53  int main(void) {
54
55      gpioInit();
56      serialPortInit();
57
58      while (1) {
59          gCounter++;
60
61          toggleDiscretes();
62          sendRepeatedSerialMessage();
63      }
64
65      // An infinite loop for safety
66      while (1) {
67      }
68
69      // We can't get passed the infinite loop to here
70  }
```

Yes. I'm paranoid. Or experienced. Running a KRAKEN off the end the stack can run your whole day.

I tend to drop these loops at the bottom of any function that should never return.

Let’s talk about the GPIO and Serial drivers and the main infinite loop.

Here's some handy links to browse the GPIO code:

https://github.com/MikeStitt/lowLevelProgrammingByExample/blob/master/presentations/20170726-meetup-embedded-gr/hal.stm.stm32f4xx.board.brd_cust_0000_0001/source/board/gpioConfig.c
https://github.com/MikeStitt/lowLevelProgrammingByExample/blob/master/presentations/20170726-meetup-embedded-gr/app.contents/source/startup/main.c#L53

https://github.com/MikeStitt/lowLevelProgrammingByExample/blob/master/presentations/20170726-meetup-embedded-gr/bsp/source/board/gpioDriver.c#L1

https://github.com/MikeStitt/lowLevelProgrammingByExample/blob/master/presentations/20170726-meetup-embedded-gr/hal.stm.stm32f4xx.common/source/board/gpioDriverImplement.h#L10

https://github.com/MikeStitt/lowLevelProgrammingByExample/blob/master/presentations/20170726-meetup-embedded-gr/bsp/source/board/gpioDriver.c#L10

Here's some handy links to browse the Serial code:

https://github.com/MikeStitt/lowLevelProgrammingByExample/blob/master/presentations/20170726-meetup-embedded-gr/hal.stm.stm32f4xx.board.brd_cust_0000_0001/source/board/serialPortConfig.c

https://github.com/MikeStitt/lowLevelProgrammingByExample/blob/master/presentations/20170726-meetup-embedded-gr/bsp/source/board/serialPortDriver.c#L3

https://github.com/MikeStitt/lowLevelProgrammingByExample/blob/master/presentations/20170726-meetup-embedded-gr/bsp/source/board/serialPortDriver.h#L40

https://github.com/MikeStitt/lowLevelProgrammingByExample/blob/master/presentations/20170726-meetup-embedded-gr/hal.stm.stm32f4xx.common/source/board/serialPortPrivateDriver.c

And back to main:

https://github.com/MikeStitt/lowLevelProgrammingByExample/blob/master/presentations/20170726-meetup-embedded-gr/app.contents/source/startup/main.c

Generate mixed c and assembly output

- Add
  - -Wa,-adhln –g

```
arm-none-eabi-gcc    -D"STM32F427_437xx" -D"USE_STDPERIPH_DRIVER" -D GCC_ARM_CM3 -D
ALIGN_STRUCT_END=__attribute\(\(aligned\(4\)\)\) -D DEPRECATED -D WDT_ENABLE= -D INCLUDE_AUTOCONF -D
ARM_MATH_CM4 -D __FPU_PRESENT=1     -I../app/include -I../app/include/generated
-I../app.contents/source -I../scaffolding/source -I../hal.stm.stm32f4xx.board.brd_cust_0000_0001/source
-I../hal.stm.stm32f4xx.chip.st32f437/include -I../hal.stm.stm32f4xx.chip.st32f437/source
-I../hal.arm.cortex/source -I../hal.stm.stm32f4xx.common/include -I../hal.stm.stm32f4xx.common/source
-I../hal.stm.stm32PeripheralLib/Libraries/CMSIS/Include
-I../hal.stm.stm32PeripheralLib/Libraries/CMSIS/Device/ST/STM32F4xx/Include
-I../hal.stm.stm32PeripheralLib/Libraries/STM32F4xx_StdPeriph_Driver/inc
-I../hal.stm.stm32PeripheralLib/Utilities/STM32_EVAL/Common
-I../hal.stm.stm32PeripheralLib/Utilities/STM32_EVAL/STM324x7I_EVAL -I../bsp/source   -mthumb
-mcpu=cortex-m3   -std=c99 -mcpu=cortex-m3    -Os -ffunction-sections -fdata-sections -Wall -Werror
-Wa,-adhln -g    -c -o obj/app/../app.contents/source/startup/main.o
../app.contents/source/startup/main.c > main.lst
```

```
int nextGpioState(int current)
{
    int next=0;
    if (0==current) {
       next = 1;
    }
    return next;
}

void toggleDiscretes(void)
{
    static uint32_t toggleCount = 0;
    const uint32_t symbolsPerSecond=115200;
    const uint32_t symbolsPerChar = 10;
    const uint32_t oneHz = symbolsPerSecond / symbolsPerChar;
    const uint32_t fiveHz = (symbolsPerSecond / symbolsPerChar) / 5;

        if (0==toggleCount) {
        int current = gpioGetOutCmd(GPIOLED1);
        int next = nextGpioState(current);
        gpioSetOutCmd(GPIOLED1, next);
    }

    if (0==toggleCount%fiveHz) {
        int current = gpioGetOutCmd(GPIOLED2);
        int next = nextGpioState(current);
        gpioSetOutCmd(GPIOLED2, next);
    }

    toggleCount = (toggleCount+1) % oneHz;
}
```

```
42:../app.contents/source/startup/main.c ****         int current = gpioGetOutCmd(GPIOLED2);
229                             .loc 2 42 0
230 002c 0120                   movs    r0, #1
231 002e FFF7FEFF               bl      gpioGetOutCmd
232                     .LVL15:
233                     .LBB34:
234                     .LBB35:
 21:../app.contents/source/startup/main.c ****     if (0==current) {
235                             .loc 2 21 0
236 0032 D0F10101               rsbs    r1, r0, #1
237 0036 38BF                   it      cc
238 0038 0021                   movcc   r1, #0
239                     .LBE35:
240                     .LBE34:
 43:../app.contents/source/startup/main.c ****         int next = nextGpioState(current);
 44:../app.contents/source/startup/main.c ****         gpioSetOutCmd(GPIOLED2, next);
241                             .loc 2 44 0
242 003a 0120                   movs    r0, #1
243                     .LVL16:
244 003c FFF7FEFF               bl      gpioSetOutCmd
245                     .LVL17:
246                     .L19:
247                     .LBE33:
 45:../app.contents/source/startup/main.c ****     }
 46:../app.contents/source/startup/main.c ****
 47:../app.contents/source/startup/main.c ****     toggleCount = (toggleCount+1) % oneHz;
248                             .loc 2 47 0
249 0040 2268                   ldr     r2, [r4]
250 0042 4FF43453               mov     r3, #11520
251 0046 0132                   adds    r2, r2, #1
252 0048 B2FBF3F1               udiv    r1, r2, r3
253 004c 03FB1123               mls     r3, r3, r1, r2
254 0050 2360                   str     r3, [r4]
255 0052 38BD                   pop     {r3, r4, r5, pc}
```

nextGpioState was pulled in to toggleDiscretes()

Thanks!