Topics

Contents

[MCU 0 – Absolute Beginners 3](#_Toc83319996)

[1. Introduction 3](#_Toc83319997)

[2. Type qualifiers in C 3](#_Toc83319998)

[3. Bitwise operators in C 3](#_Toc83319999)

[4. Structures in C 3](#_Toc83320000)

[5. Unions in C 3](#_Toc83320001)

[6. Turn ON LED - Coding 3](#_Toc83320002)

[7. Volatile type qualifier 3](#_Toc83320003)

[8. Pointers 3](#_Toc83320004)

[9. MISC Topics 3](#_Toc83320005)

[ARM M3/M4 Processor - Programming 4](#_Toc83320006)

[1. Introduction 4](#_Toc83320007)

[2. Embedded Hello World 4](#_Toc83320008)

[3. Access levels and Operation modes 4](#_Toc83320009)

[4. ARM GCC Inline assembly coding 4](#_Toc83320010)

[5. Reset Sequence of the Processor 4](#_Toc83320011)

[6. Access level and T-bit 4](#_Toc83320012)

[7. Memory map and bus interfaces of ARM Cortex Mx processor 4](#_Toc83320013)

[8. Stack memory and placement 4](#_Toc83320014)

[9. Exception model of ARM Cortex Mx processor 4](#_Toc83320015)

[10. Interrupt priority and configuration 4](#_Toc83320016)

[11. Exception entry and exit sequences 4](#_Toc83320017)

[12. Fault handling & Analysis 5](#_Toc83320018)

[13. Exception for system level services 5](#_Toc83320019)

[14. Implementation of Task Scheduler 5](#_Toc83320020)

[15. Bare metal embedded and linker scripts 5](#_Toc83320021)

[MCU-1: Mastering Microcontroller and Embedded Driver Development 6](#_Toc83320022)

[1. Introduction 6](#_Toc83320023)

[2. MCU Clocks 6](#_Toc83320024)

# MCU 0 – Absolute Beginners

## Introduction

* How Printf works in SWO?
* SWD and JTAG
* ITM console
* Build process
* Pointer definition

## Type qualifiers in C

* Const and volatile
* Case studies of const

## Bitwise operators in C

* Testing of bits
* Setting of bits
* Clearing of bits
* Toggle the bits
* Shift operators
* Bit extraction

## Structures in C

* Structure definition/syntax
* Aligned/unaligned data access
* Typedef with structure
* Bit fields
* See all programs

## Unions in C

* Union definition/syntax
* Applicability of Unions in C
* See all programs

## Turn ON LED - Coding

* This is in Section 18.
* Memory mapped I/O
* Memory map of the processor
* STM32 memory map
* Memory mapped peripheral registers and I/O access
* Procedure to turn ON LED
* How to enable peripheral clock?
* Calculating peripheral register addresses
* PIN Read Exercise

## Volatile type qualifier

* Definition/info.
* When to use volatile?
* Volatile case studies

## Pointers

* Section 13
* See all pointer exercises code.

## MISC Topics

* Button ISR
* Usage of bit fields in embedded code
* Keypad interfacing
* Why pull up resistors are needed?

# ARM M3/M4 Processor - Programming

## Introduction

* Section 1
* Motivation
* Processor core VS Processor
* Processor VS Microcontroller
* Build process
* Pointer definition

## Embedded Hello World

* Section 4
* Embedded Target
* Settings in STM32Cube IDE
* To view printf statements in console
* Printf using SWO

## Access levels and Operation modes

* Section 5
* Features of the Cortex M processor
* Operational Modes – Thread and Handler
* Example demo program for operational modes
* Access Levels – Privileged and Non-privileged
* Cortex M Processor – Core Registers
* Memory mapped and Non-memory mapped registers

## ARM GCC Inline assembly coding

* Section 6

## Reset Sequence of the Processor

* Section 7
* <http://fastbitlab.com/arm-cortex-m-processor/> - Blog page 33

## Access level and T-bit

* Section 8
* Switching between access levels
* T-bit of EPSR

## Memory map and bus interfaces of ARM Cortex Mx processor

* Section 9
* Memory map – different regions
* Bus protocols and bus interfaces – AHB/APB
  + <http://fastbitlab.com/bus-protocols-bus-interfaces-cortex-m3-m4-prcoessor/> - Blog page 32
* Bit Banding
* Calculation of bit band alias address

## Stack memory and placement

* Section 10
* Introduction to stack memory
* Stack memory uses
* Stack operation models
* Stack placement
* Banked stack pointer registers of ARM Cortex Mx
* Stack exercise
* AAPCS standard – Scope
* Stack activities during interrupt and exception
* Stack initialization

## Exception model of ARM Cortex Mx processor

* Section 11
* Exception model
* System exceptions
* System exception vector addresses
* System exception control registers – SCB/SLB
* Default S/m exception settings
* NVIC
* Enable/Disable/Pend various interrupts using NVIC Registers
* Peripheral interrupt exercise
* Steps to program an MCU Peripheral Interrupt

## Interrupt priority and configuration

* Section 12
* Interrupt priority explanation
* Pre-empt and sub priority
* Priority grouping
* Priority grouping case study
* Interrupt priority configuration exercise
* Pending interrupt behavior – Single pended & Double pended interrupts

## Exception entry and exit sequences

* Section 13
* Exception entry sequence
* Exception exit sequence
* EXC\_RETURN – when it is generated?

## Fault handling & Analysis

* Section 14
* Why Faults happen?
* Different types of Fault exceptions
* Hard fault – causes, HFSR register
* Mem manage fault – causes
* Bus fault – causes
* Usage fault – causes
* Analyzing stack frame
* Error reporting when fault happens

## Exception for system level services

* Section 15
* SVC exception
* Methods to trigger SVC exception
* How to extract SVC number?
* SVC number exercise
* PendSV exception - Uses
* Context switching
* How pendsv is used for context switching in FreeRTOS?
  + <http://fastbitlab.com/category/blog/page/33/> - Blog page 33
  + <http://fastbitlab.com/free-rtos/>

## Implementation of Task Scheduler

* Section 16
* Introduction
* Creating user tasks
* Stack pointer selection/Stack assessment
* Scheduling policy selection
* Context switching
* Case study of Context Switching
* Systick count value calculation
* Task’s stack area init and storing of dummy SF
* Initialization of stack
* LED toggling using multiple tasks
* Blocking a task for given no: of systicks

## Bare metal embedded and linker scripts

* Section 17
* Bare metal embedded
* Cross compilation and toolchains
* Installing GCC ARM cross toolchain
* Build process
* Compilation and compiler flags
* Makefile
* Analyzing relocatable obj files
* Code and data of a program
* Linker and locator
* Different data and sections of a program
* . bss Vs data
* Startup file of Microcontroller
* Writing startup file of Microcontroller from scratch part - 1, 2, 3
* Writing linker script from scratch part – 1,2
* Location counter
* Linker script symbols
* Writing linker script from scratch part-3
* Linking and Linker flags
* Analyzing ELF file
* Implementing reset handler
* OpenOCD and debug adapters
* Using GDB client
* C standard library integration
* Integrating system calls
* Section merging of standard library
* Fixing linker script to resolve hardfault
* Semi-hosting

# MCU-1: Mastering Microcontroller and Embedded Driver Development

## Introduction

* Section 6 – Embedded code debugging tips and tricks
* Section 7 – Understanding MCU Memory map (covered in section 9 of the ARM M3/M4 course)
* Section 8 – MCU Bus Interfaces – I Bus, D Bus, S-Bus of the processor, AHB, APB1, APB2
* Q/A section
* MCU Bus Matrix

## MCU Clocks

* Section 9 – MCU Clocks
* Section 10 – Understanding MCU Clock tree
* HSI and RCC registers
* Peripheral clock configuration
* HSI measurement exercise
* HSE measurement exercise
* System/AHB/APB clocks of STM32 MCU - <http://fastbitlab.com/exploring-system-ahb-apb-stm32-mcu/> - Blog page 33

## MCU Topics

* Section 11 - MCU Vector Table
* Section 12 – MCU Interrupt design, NVIC, Interrupt handling
* Button interrupt
* EXTI Registers
* Section 13 – Volatile qualifier in C – Already covered in MCU 0 – section 2 and 7 in this document

## GPIO - Concepts

* Section 14 – GPIO Must Know Concepts
* GPIO pin and GPIO port - <http://fastbitlab.com/stm32-gpio-lecture-1-gpio-pin-ports/>
* GPIO behind the scene - <http://fastbitlab.com/gpio-behind-scene/>
* GPIO input mode with high impedance  
  state - <http://fastbitlab.com/gpio-input-mode-high-impedance-state/>
* GPIO input mode with pull-up/down state - <http://fastbitlab.com/gpio-input-mode-pull-pull-state/>
* GPIO output mode with open drain state - <http://fastbitlab.com/gpio-output-mode-open-drain-state/>
* GPIO output mode with push pull state - <http://fastbitlab.com/gpio-output-mode-push-pull-state/>
* Optimizing I/O power consumption - <http://fastbitlab.com/optimizing-o-power-consumption/>

## GPIO – Programming structure/Registers

* GPIO programming structure - <http://fastbitlab.com/gpio-programming-structure/>
* Exploring GPIO PORT and pins of  
  STM32F4xx Discovery board - <http://fastbitlab.com/exploring-gpio-port-pins/>
* GPIO Mode register(used to set mode for a  
  pin) - <http://fastbitlab.com/gpio-mode-register/>
* Input configuration of a Microcontroller's  
  GPIO Pin - <http://fastbitlab.com/input-configuration-microcontrollers/>
* Output Configuration of a GPIO Pin in Push  
  pull mode - <http://fastbitlab.com/output-configuration-gpio-pin-push/>
* Output Configuration of a GPIO Pin in open  
  drain mode - <http://fastbitlab.com/output-configuration-gpio-pin-open-drain/>
* Input stage of a GPIO pin during output  
  configuration
* Alternate functionality Configuration of a GPIO pin - <http://fastbitlab.com/alternate-functionality-configuration/>
* GPIO output type register explanation

## GPIO Registers : SPEED, PULL UP/DOWN, IDR and ODR

* Section 16: GPIO Registers : SPEED, PULL  
  UP/DOWN, IDR and ODR
* GPIO output speed register and its applicability - <http://fastbitlab.com/gpio-output-speed-register-applicability/>
* GPIO Pull up and Pull down register - <http://fastbitlab.com/gpio-pull-pull-registers/>
* GPIO input data register - <http://fastbitlab.com/gpio-input-data-register/>
* GPIO output data register and summary of  
  various modes discussed - <http://fastbitlab.com/gpio-output-data-register/>

## GPIO – AFR and Peripheral clock

* Section 17: GPIO Alternate functionality  
  register and example of usage
* Alternate functionality settings of a GPIO pin with example - <http://fastbitlab.com/alternate-functionality-setting-gpio/>
* Quiz 4: Find Out I/O Alt Functionality
* Section 18: GPIO peripheral clock control
* Enabling and disabling GPIO peripheral clock - <http://fastbitlab.com/enabling-disabling-gpio-peripheral-clock/>

## GPIO – Driver Development

* ***Look at the code while studying these topics***
* Overview and project creation – section 19
* MCU specific header file contents
* Base addresses of various memories such as Flash, SRAM1, SRAM2, ROM – section 20
* Base address of bus domains – AHB, APB – section 20
* Structuring peripheral registers – section 21
* Clock management macros – RCC, Peripheral clock enable – section 22
* GPIO handle and configuration structure – section 23
* GPIO driver API requirements - section 23
* GPIO driver API – clock control – section 24
* GPIO Init and De-Init – section 25
* GPIO data read and write – section 26
* LED exercise on GPIO – section 27
* GPIO Pin Interrupt configuration – section 28
* GPIO Interrupts exercise – section 29
* MCU I/O pin spec, current characteristics, logic levels

## SPI – Intro

* Introduction to SPI Bus - <http://fastbitlab.com/introduction-spi-bus-2/>
* SPI comparison with other protocols - <http://fastbitlab.com/spi-comparison-protocols/>
* Importance of SPI slave select pin - <http://fastbitlab.com/importance-spi-slave-select-pin/>
* SPI Minimum bus configuration - <http://fastbitlab.com/spi-minimal-bus-configuration/>
* SPI behind the scene data communication  
  principle - <http://fastbitlab.com/spi-behind-scene-data-communication-principle/>

## SPI – Details

* Section 32: SPI bus configuration and functional block diagram
* SPI bus configuration discussion : full duplex, half duplex and simplex - <http://fastbitlab.com/spi-bus-configuration-discussion-full-duplex-half-duplex-simplex/>
* SPI functional block diagram explanation - <http://fastbitlab.com/spifunctional-block-diagram-explanation/>
* Section 33: STM32 NSS pin settings and management
* NSS settings in STM32 master and slave modes - <http://fastbitlab.com/nss-setting-stm32-master-slave-mode/>
* STM32 SPI hardware and software slave managements - <http://fastbitlab.com/stm32-spi-hardware-software-slave-managements/>
* Section 34: SPI CPOL and CPHA - <http://fastbitlab.com/spi-cpol-cpha-discussion/>
* Section 35: SPI Serial clock

## SPI – Driver Development

* ***Look at the code while studying these topics***
* SPI driver API requirements and configuration structure – section 36
* SPI driver API – Clock control – section 37
* SPI driver API – SPI Init – section 38
* SPI driver API – Send Data – section 39
* Exercise – SPI send data – section 40