



Introduction to Embedded Systems, Lecture 6

Architecture of modern (micro) controllers

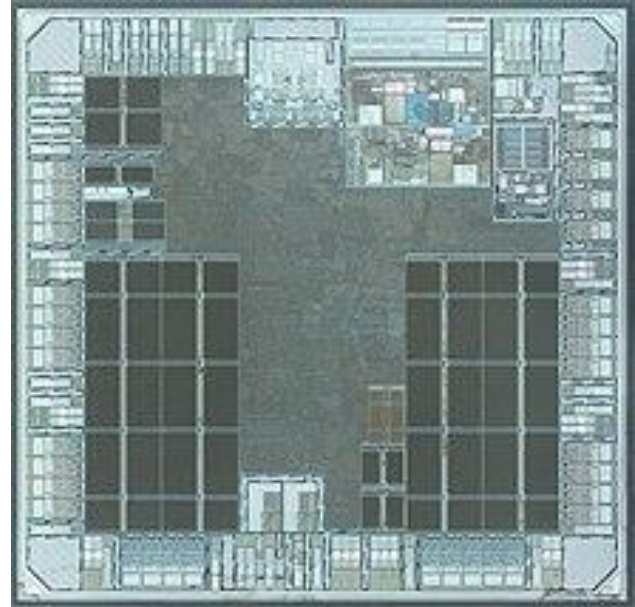
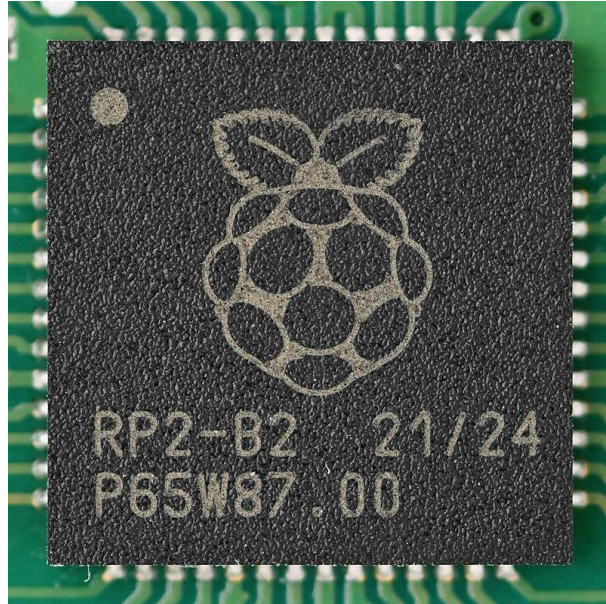
Dmitry Zaitsev

<http://daze.ho.ua>

Microcontroller

- A microcontroller is a compact integrated circuit designed to govern a specific operation in an embedded system.
- A typical microcontroller includes a processor, memory and input/output (I/O) peripherals on a single chip.

Microcontroller on chip RP2040



High performance, low cost, and ease of use

- a large on-chip memory
- symmetric dual-core processor complex
- deterministic bus fabric
- rich peripheral set augmented with unique Programmable I/O (PIO) subsystem
- polished MicroPython port
- UF2 bootloader in ROM

RP2040 brief specification

- 32-bit dual ARM Cortex-M0+ microcontroller integrated circuit
- 21 January 2021
- 1\$
- 40 nm silicon in a 7×7 mm surface-mount device (SMD)
- <https://github.com/raspberrypi/pico-bootrom>

Key features of RP2040

- 133 MHz dual ARM Cortex-M0+ cores (supports overclocking)
- Each core has an integer divider peripheral, and two interpolators.
- 264 KB SRAM in six independent banks (four 64 KB, two 4 KB)
- No internal flash or EEPROM memory (after reset, the boot-loader loads firmware from either external flash memory or USB into internal SRAM)
- QSPI bus controller, supporting up to 16 MB of external flash memory
- DMA controller
- AHB crossbar, fully-connected
- On-chip programmable low-dropout regulator (LDO) to generate core voltage
- 2 on-chip PLLs to generate USB and core clocks
- 30 GPIO pins, of which 4 can optionally be used as analog inputs

RP2040 peripherals

- 2 UARTs - Universal Asynchronous Receiver-Transmitter
- 2 SPI - Serial Peripheral Interface - controllers
- 2 I²C - Inter-Integrated Circuit - controllers
- 16 PWM - Pulse-width modulation - channels
- USB 1.1 controller and PHY, with host and device support
- 8 programmable input-output (PIO) state machines
- 4 channel ADC with internal temperature sensor, 500ksps, 12-bit conversion

Raspberry Pi



Pico



Pico H



Pico W



Pico WH

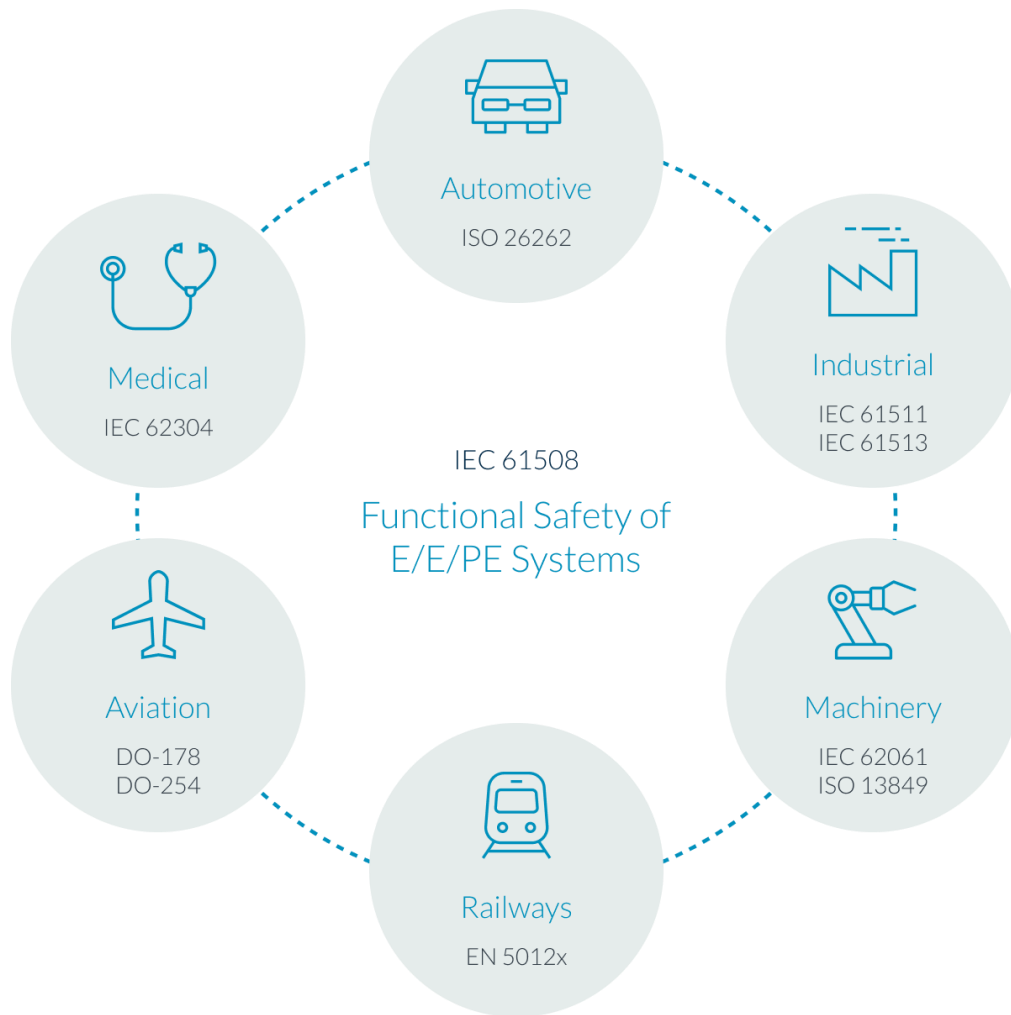
Basic manufactures of controllers

- ARM: <https://www.arm.com>
- Microchip: <https://www.microchip.com>
- Qualcomm: <https://www.qualcomm.com>
- NXP: <https://www.nxp.com>
- Renesas: <https://www.renesas.com>
- Silicon Labs: <https://www.silabs.com/mcu>

Arm Cortex-M0+ processor

- the most energy-efficient Arm processor
- builds on Cortex-M0 processor
- retains full instruction set and tool compatibility
- small silicon area
- low power and minimal code footprint

Safety Package



General scheme

arm CORTEX[®]-M0+

Nested vectored
interrupt controller

Wake-up interrupt
controller

CPU
Armv6-M

Memory protection unit

AHB-Lite

Data
watchpoint

JTAG

Breakpoint
unit

Fast I/O
port

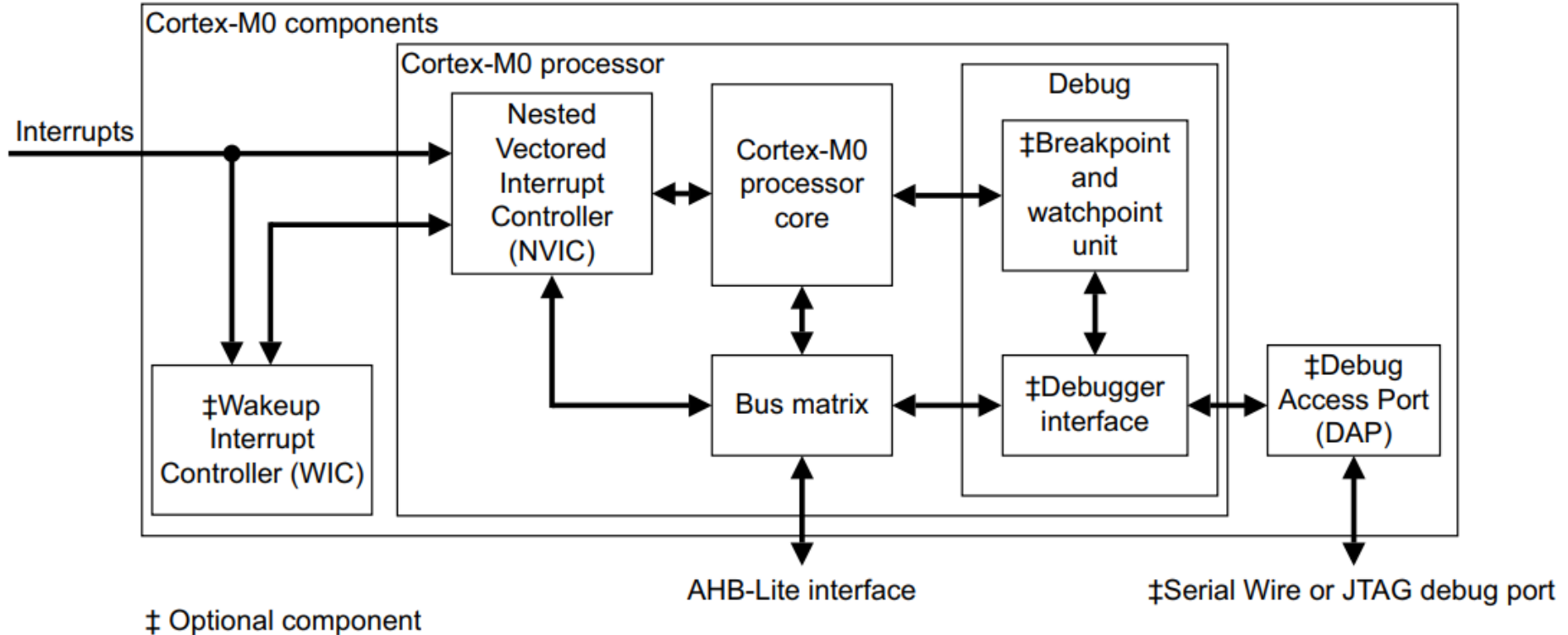
MTB

Serial wire

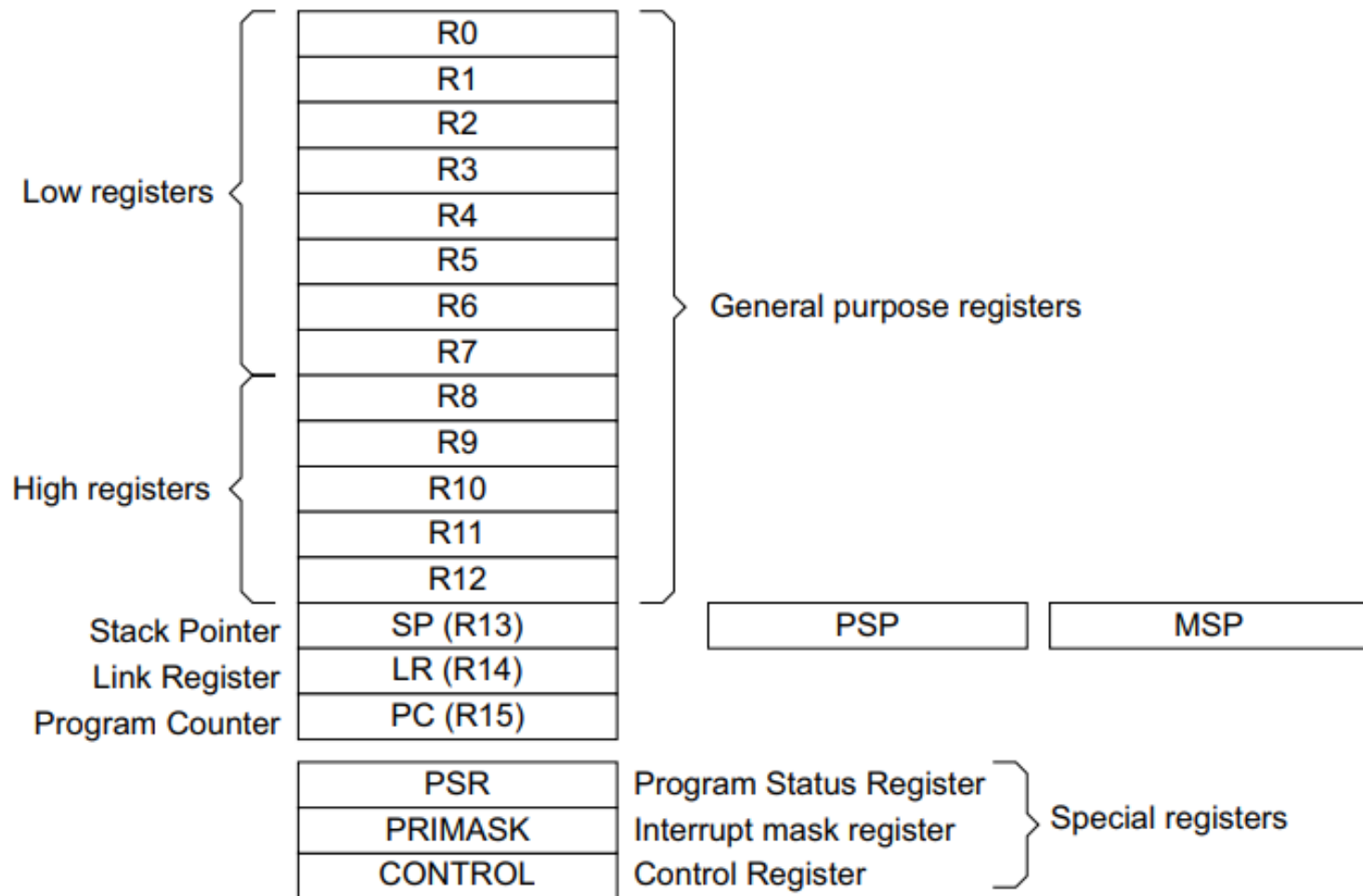
Programmers model

- Processor modes: Thread mode; Handler mode
- Stacks: the main stack and the process stack
- Registers: R0-R12 are 32-bit general-purpose registers for data operations; R13 Stack Pointer (SP); R14 Link Register (LR); R15 Program Counter (PC); Program Status Register (PSR) – Flags NZCV; Mask register; Control register
- Exceptions and interrupts – Nested Vectored Interrupt Controller (NVIC)

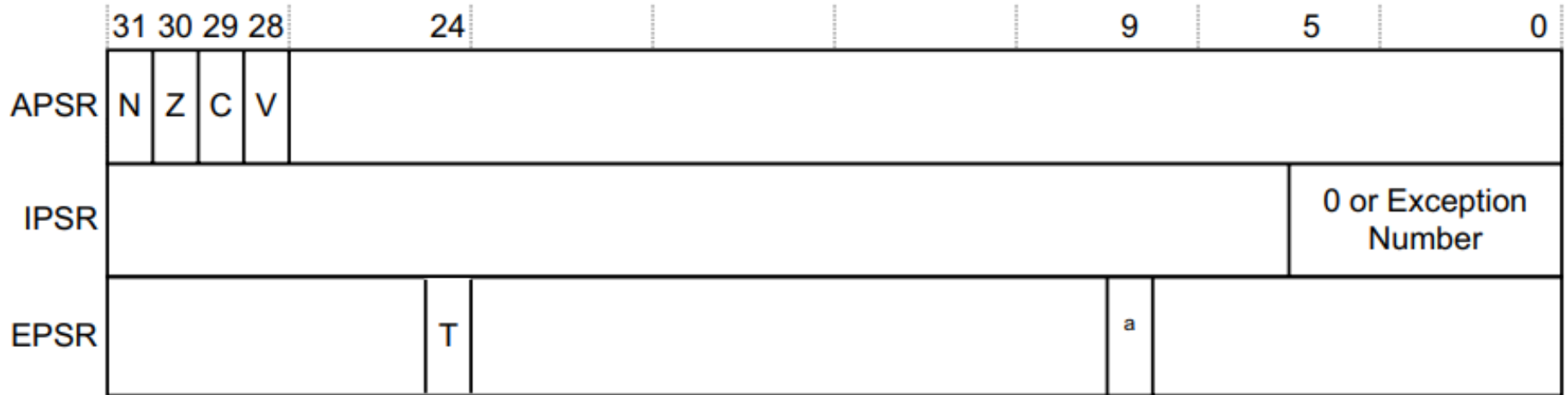
Functional blocks



Processor core registers



xPSR register layout



Exception numbers

Exception number	Exception
1	Reset
2	NMI
3	HardFault
4-10	Reserved
11	SVCall
12-13	Reserved
14	PendSV
15	SysTick, optional
16	External Interrupt(0)

Memory model

Device	511MB	0xFFFFFFF
Private peripheral bus	1MB	0xE0100000 0xE00FFFFFF 0xE0000000 0xDFFFFFFF
External device	1.0GB	
External RAM	1.0GB	0xA0000000 0x9FFFFFFF
Peripheral	0.5GB	0x60000000 0x5FFFFFFF
SRAM	0.5GB	0x40000000 0x3FFFFFFF
Code	0.5GB	0x20000000 0x1FFFFFFF
		0x00000000

Memory map

Address	Name	Device type	XN?	Cache	Description
0x00000000-0x1FFFFFFF	Code	Normal	-	WT	Typically ROM or flash memory. Memory required from address 0x0 to support the vector table for system boot code on reset.
0x20000000-0x3FFFFFFF	SRAM	Normal	-	WBWA	SRAM region typically used for on-chip RAM.
0x40000000-0x5FFFFFFF	Peripheral	Device	XN	-	On-chip peripheral address space.
0x60000000-0x7FFFFFFF	RAM	Normal	-	WBWA	Memory with writeback, write allocate cache attribute for L2/L3 cache support.
0x80000000-0x9FFFFFFF	RAM	Normal	-	WT	Memory with Write-Through cache attribute.
0xA0000000-0xBFFFFFFF	Device	Device, shareable	XN	-	Shared device space.
0xC0000000-0xDFFFFFFF	Device	Device, Non-shareable	XN	-	Non-shared device space.
0xE0000000-0xFFFFFFFF	System	See <i>Description</i>	XN	-	System segment for the PPB and vendor system peripherals, see Table B3-2 .

Cortex-M0 Instruction Set

- Memory access instructions
- General data processing instructions
- Branch and control instructions
- Miscellaneous instructions.

Memory access instructions

Mnemonic	Brief description
ADR	Generate PC-relative address
LDM	Load Multiple registers
LDR{type}	Load Register using immediate offset
LDR{type}	Load Register using register offset
LDR	Load Register from PC-relative address
POP	Pop registers from stack
PUSH	Push registers onto stack
STM	Store Multiple registers
STR{type}	Store Register using immediate offset
STR{type}	Store Register using register offset

Data processing instructions

Mnemonic	Brief description
-----------------	--------------------------

ADCS	Add with Carry
ADD{S}	Add
ANDS	Logical AND
ASRS	Arithmetic Shift Right
BICS	Bit Clear
CMN	Compare Negative
CMP	Compare
EORS	Exclusive OR
LSLS	Logical Shift Left
LSRS	Logical Shift Right
MOV{S}	Move
MULS	Multiply
MVNS	Move NOT

Mnemonic	Brief description
-----------------	--------------------------

ORRS	Logical OR
REV	Reverse byte order in a word
REV16	Reverse byte order in each halfword
REVSH	-// - and sign extend
RORS	Rotate Right
RSBS	Reverse Subtract
SBCS	Subtract with Carry
SUBS	Subtract
SXTB	Sign extend a byte
SXTH	Sign extend a halfword
UXTB	Zero extend a byte
UXTH	Zero extend a halfword
TST	Test

Branch and control instructions

Mnemonic

Brief description

B{cc}

Branch {conditionally}

BL

Branch with Link

BLX

Branch indirect with Link

BX

Branch indirect

Addressing modes

- Register: r3
- Indirect register: [r3]
- Immediate: =0xF00DF00D, #1
- Indirect with offset with respect to PC or SP
 strb r1, [sp, #4]

Example: eternal increment

@ Vector table start

.long 0x20001000

.long _start

@ Vector table end

_start:

loop:

ADD R0, R0, #1

b loop

.global _start

Example: blink LED (beginning)

.thumb

@ Register addresses

.equ PERIPH_BASE, (0x40000000)

.equ AHBPERIPH_BASE, (PERIPH_BASE + 0x00020000)

.equ AHB2PERIPH_BASE, (PERIPH_BASE + 0x08000000)

.equ RCC_BASE , (AHBPERIPH_BASE + 0x00001000)

.equ GPIOC_BASE, (AHB2PERIPH_BASE + 0x00000800)

@ Make start function global

.global _start

@ Vector table

.word 0x20001000 @ Stack

.word _start @ Code

.thumb_func

Example: blink LED (continuation)

_start:

@ Enable clock for GPIOC peripheral in RCC registers

LDR r0, =(RCC_BASE + 0x14)

LDR r1, =(1 << 19)

STR r1, [r0]

@ Enable GPIOC pin 9 as output

LDR r0, =(GPIOC_BASE + 0x00)

LDR r1, =(1 << (9*2))

STR r1, [r0]

Example: blink LED (completion)

loop:

@ Write high to pin 9

LDR r0, =(GPIOC_BASE + 0x14)

LDR r1, =(1 << 9)

STR r1, [r0]

@ Dummy counter for delay

LDR R0, =0

LDR R1, =200000

loop0:

ADD R0, R0, #1

cmp R0, R1

bne loop0

@ Write low to 9

LDR r0, =(GPIOC_BASE + 0x14)

LDR r1, =(0)

STR r1, [r0]

@ Dummy counter for delay

LDR R0, =0

LDR R1, =200000

loop1:

ADD R0, R0, #1

cmp R0, R1

bne loop1

b loop