BME3321:Introduction to Microcontroller Programming

Topic 2: Embedded systems, Microcontroller architectures, Memory mapping

Asst. Prof. Dr. İsmail Cantürk

The Definitive Guide to ARM® Cortex®-M0 and Cortex-M0+ Processors (ch1,2,3,4,7)

Welcome to the World of Embedded systems

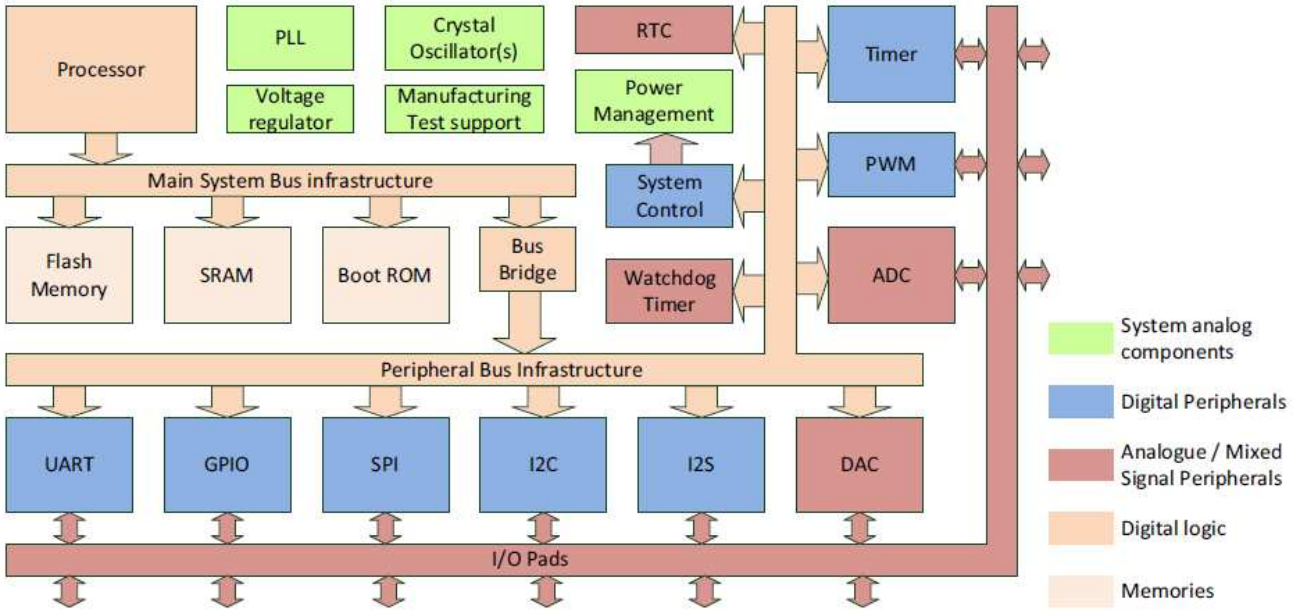
- Microcontrollers are used in majority of electronic products. For example, your mobile phones, televisions, washing machines....

- **Some definitions related to embedded systems:**

Processor core/CPU core: typically refer to the processor inside a microcontroller product, excluding the memory system, peripherals...
The terms “Microprocessor” and “CPU” can be interchangeable.

Microcontroller: a chip device containing processor to handle control and computational tasks. This chip typically contains a memory system and a number of peripherals

What Is Inside a Microcontroller



A simple microcontroller.

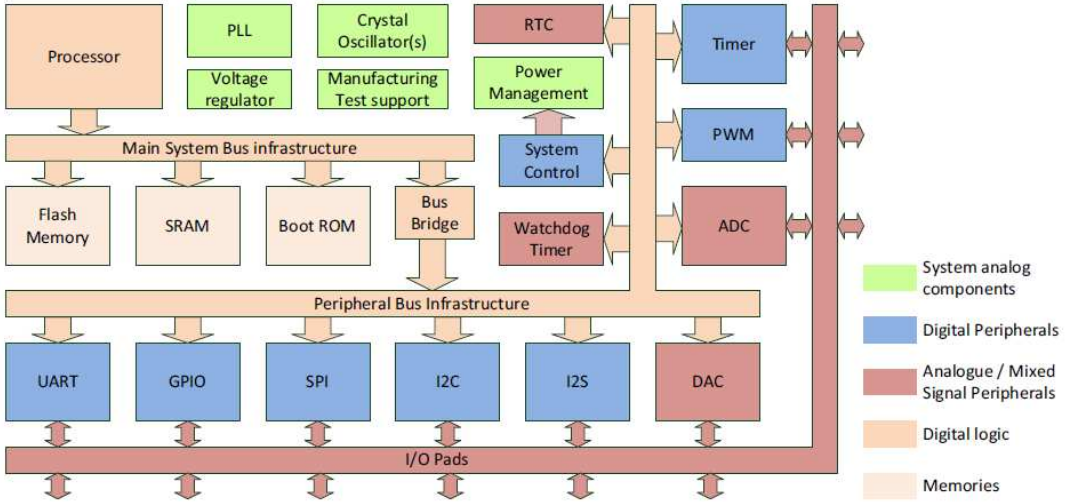
ROM Read Only Memory, Nonvolatile memory storage for program code.

Flash memory A special type of ROM, which can be reprogrammed many times, typically for storing program code.

SRAM Static Random Access Memory for data storage (volatile)

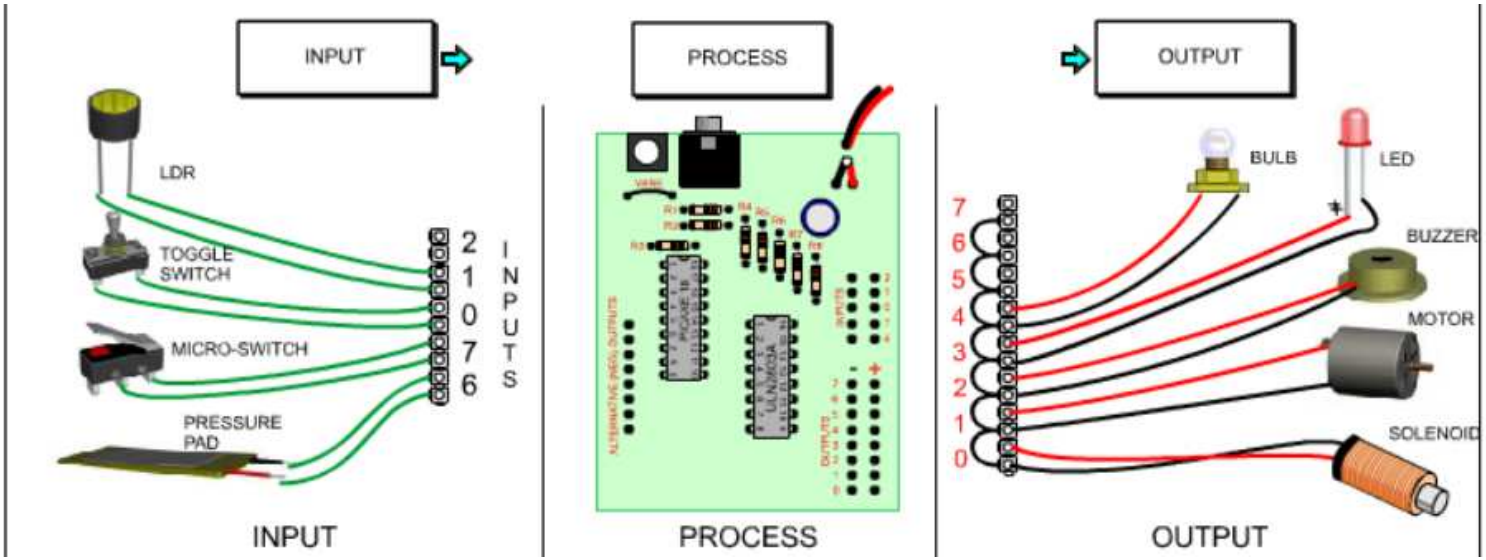
Memories

What Is Inside a Microcontroller

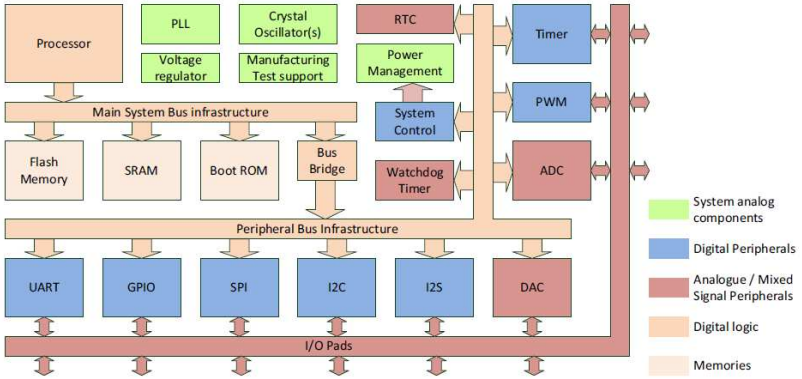


A simple microcontroller.

GPIO General Purpose Input/Output, a peripheral to control external devices and to read back external signals status.

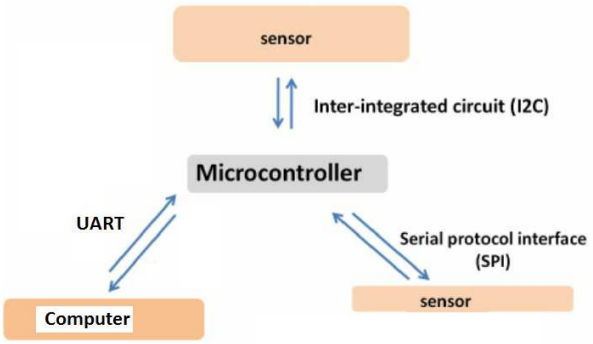


What Is Inside a Microcontroller



A simple microcontroller.

Communication Peripherals



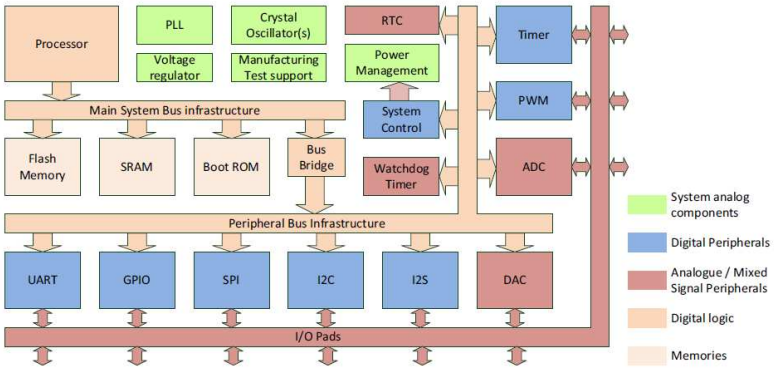
UART Universal Asynchronous Receiver/Transmitter, a peripheral to handle data transfers in a simple serial data protocol.

I2C Inter-Integrated Circuit, a peripheral to handle data transfers in a serial data protocol. Unlike UART, a clock signal is required and can provide higher data rate.

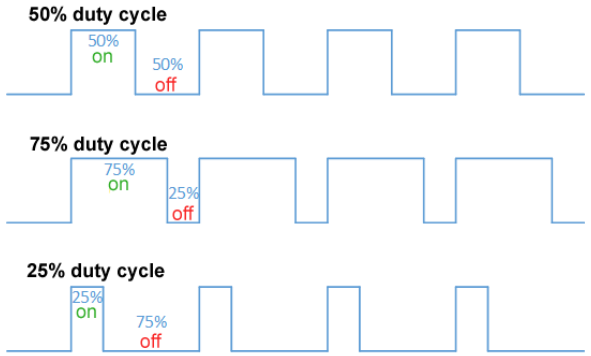
SPI Serial Peripheral Interface, another serial communication interface for off-chip peripherals.

I2S Inter-IC Sound, a serial data communication interface specifically for audio information.

What Is Inside a Microcontroller

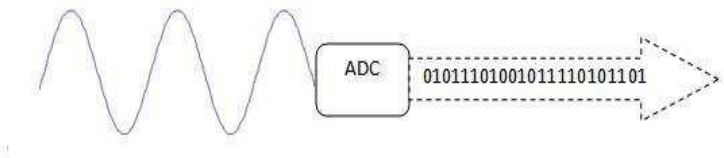


A simple microcontroller.

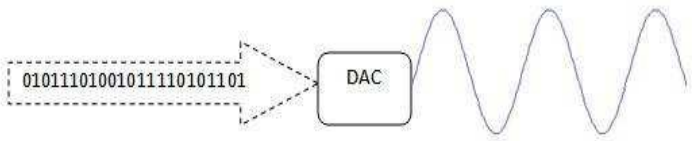


PWM Pulse Width Modulator, a peripheral to output waveform with programmable duty cycle.

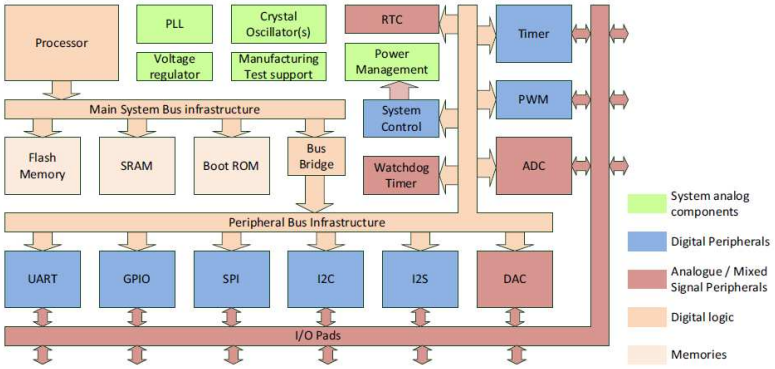
ADC Analog to Digital Converter, a peripheral to convert analog signal-level information into digital form.



DAC Digital to Analog Converter, a peripheral to convert data values into analog signal level.



What Is Inside a Microcontroller



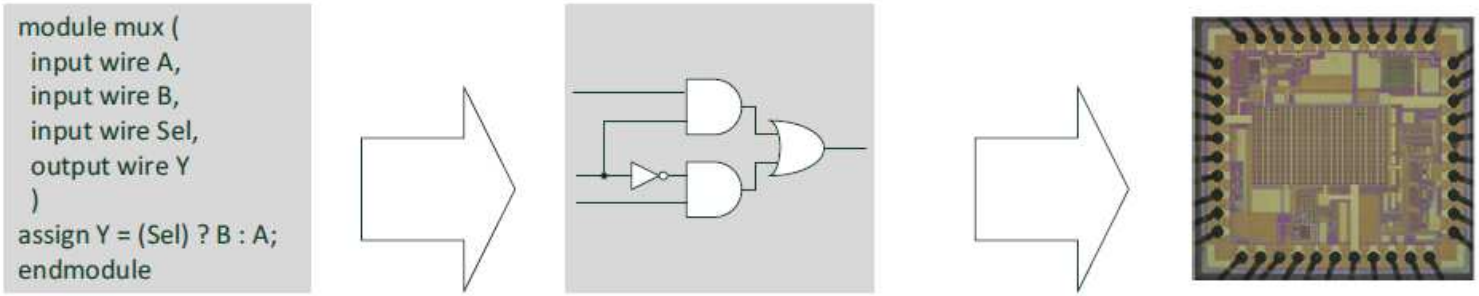
A simple microcontroller.

Watchdog timer A programmable timer device for ensuring the processor is running program. When enabled, the program running needs to update the watchdog timer within a certain time gap. If the program crashed, the watchdog timed out and this can be used to trigger a reset or a critical interrupt event.

PLL Phase Lock Loop device to generate programmable clock frequency based on a reference clock.

RTC Real Time Clock a low power timer for counting seconds (typically runs on a low power oscillator)

What ARM Does in ARM-Cortex Microcontrollers



Simplified chip design flow.

ARM produces processor designs and sell them to Microcontroller producers.

ARM does not sell hardware. Only software.

Embedded software development- Basic concepts

Reset:

A microcontroller needs to be reset to get to a known state before program execution. Reset is typically generated by hardware (reset button) and sometimes by software

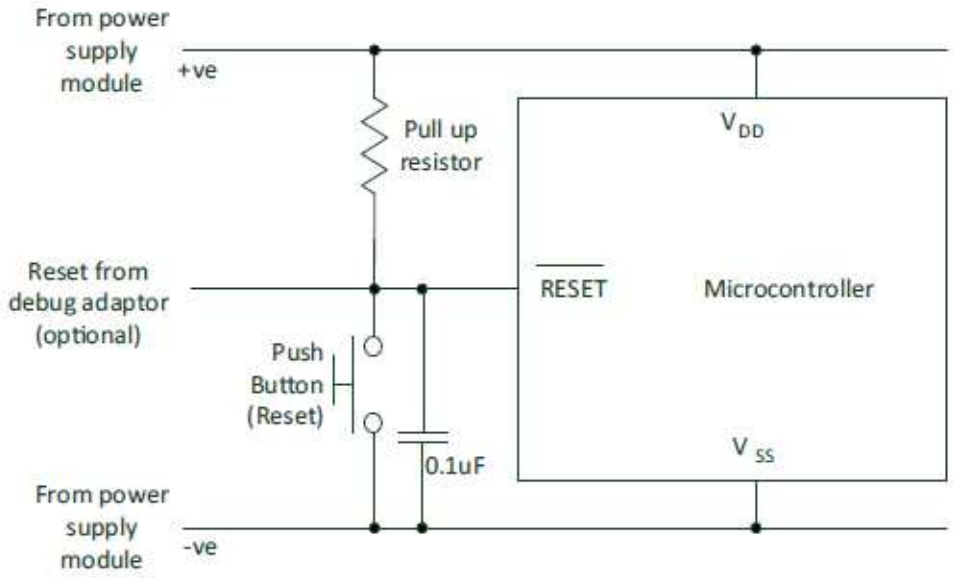
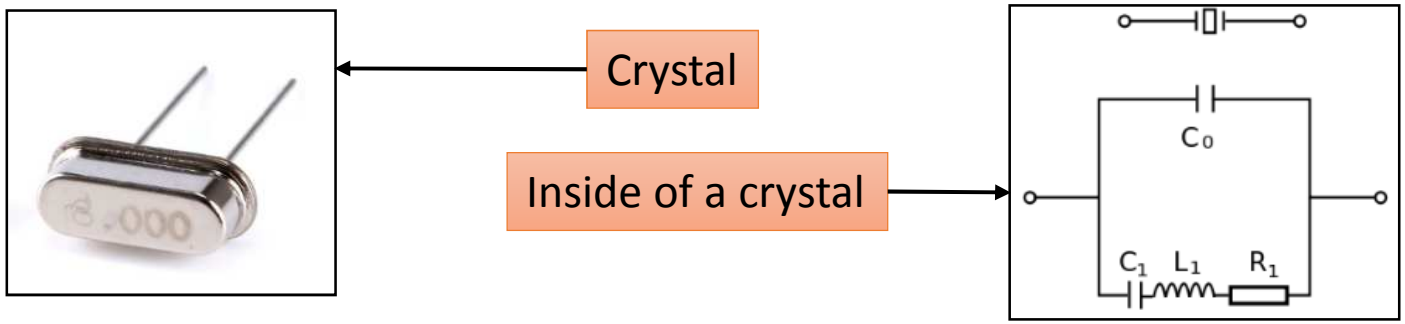


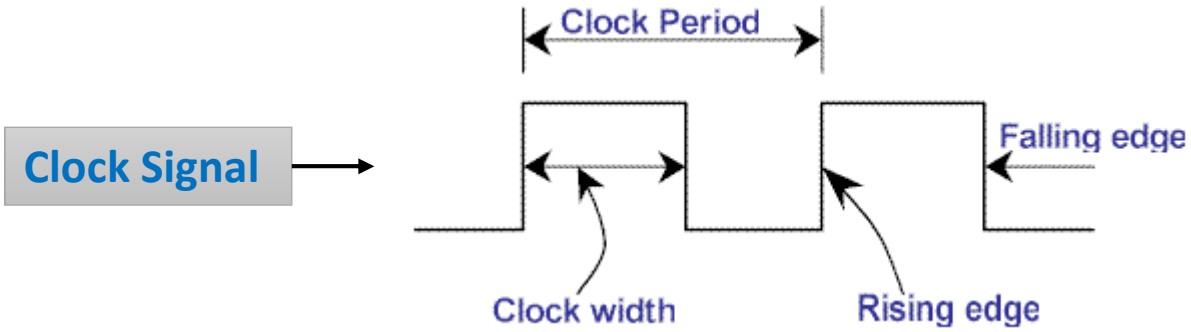
Figure 3.1

Example reset connection in low-cost microcontroller board (assumed that the reset pin is active low).

Clocks:

Almost all processors and digital circuits need clock signals to operate. Microcontrollers typically support external crystal for reference clock generation.





Basic concepts

Voltage level:

- All microcontrollers need power to run, so you will find power supply pins (VDD, GND) on a microcontroller.

- Check user manual of MCU for voltage levels

- STM32 MCUs can be powered with USB connection.

Inputs and Outputs:

- Unlike personal computers, most embedded systems have no display, no keyboard, and mouse.

- The available inputs and outputs can be limited to simple electronic interfaces like digital and analog inputs and outputs (I/Os), UARTs, I2C, SPI, etc.

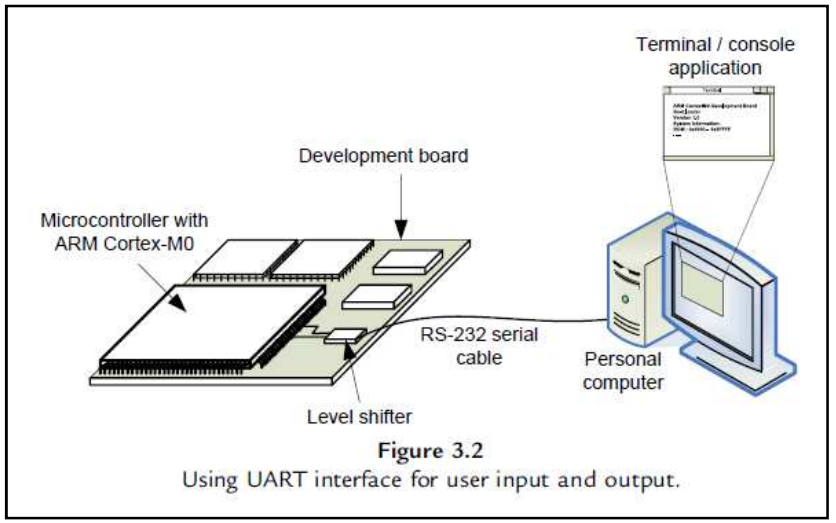


Figure 3.2
Using UART interface for user input and output.

Embedded Software Program Flows: Polling or Interrupt based CODING

Polling: polling is easy to set up and works fairly well for simple tasks. However, when the application gets complicated and demands higher processing performance, polling is not suitable.

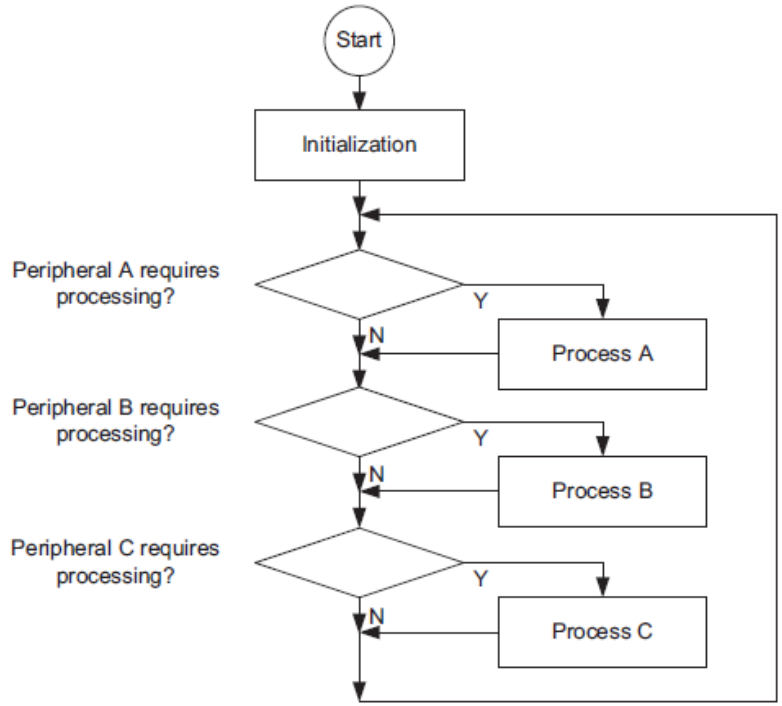


Figure 3.3
Polling method for simple application processing.

Just like checking door periodically...

Embedded Software Program Flows: Polling or Interrupt based CODING

Interrupt based: processing is carried out in interrupt service routines so that the processor can enter sleep mode when no processing is required.

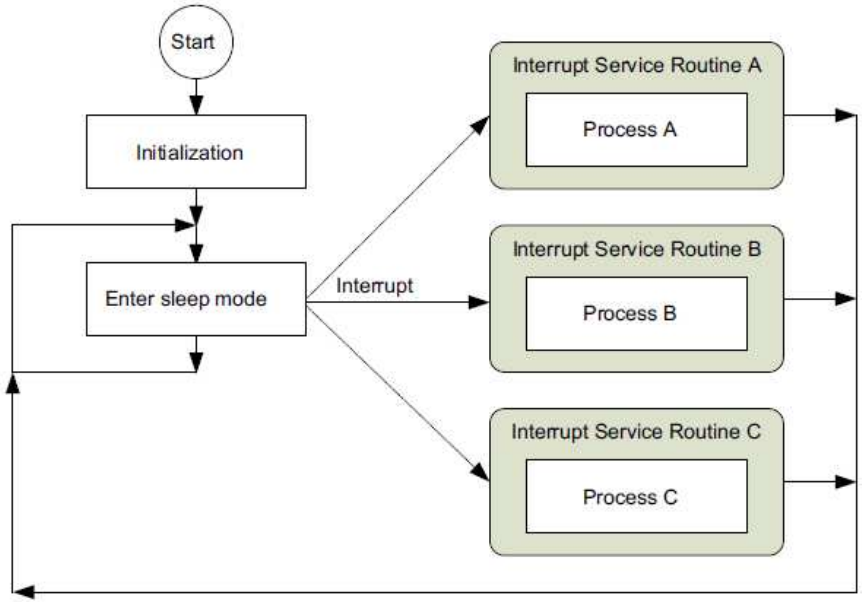
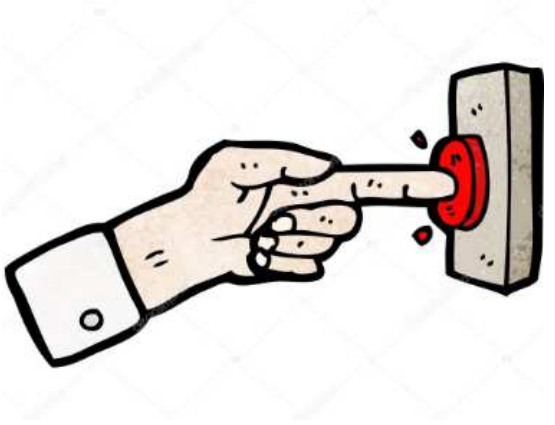


Figure 3.4
Interrupt-driven application.



Doorbell is interrupt ...

Combination of Polling or Interrupt

- In many cases, polling and interrupt methods are combined.

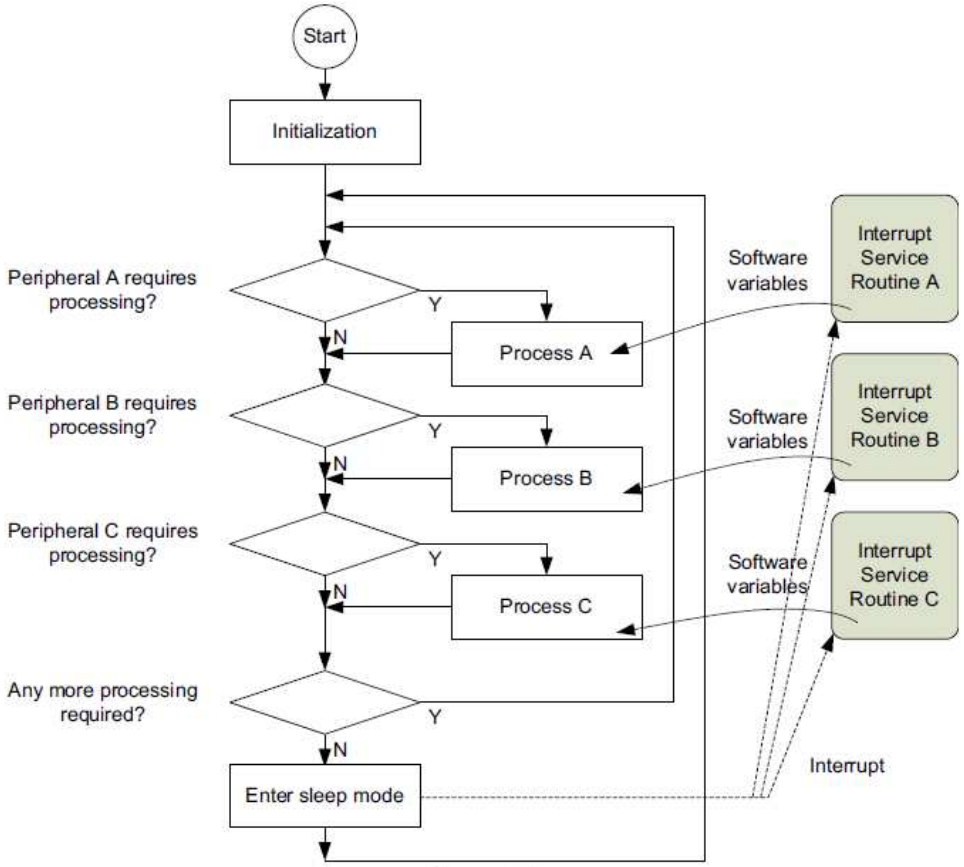
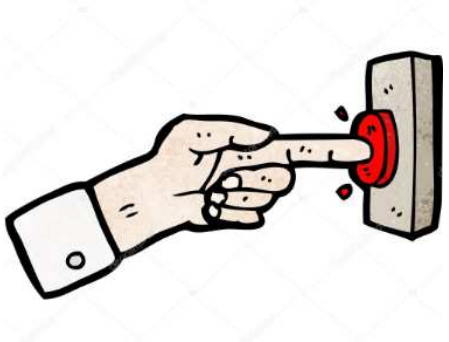


Figure 3.5
Combination of polling and interrupt-driven application.

Checking food is pooling...



Doorbell is interrupt ...

Programming Language Choices: Assembly or C

- C is close to problem domain →

```
while( eax < ebx)
   eax = eax + 1;
```

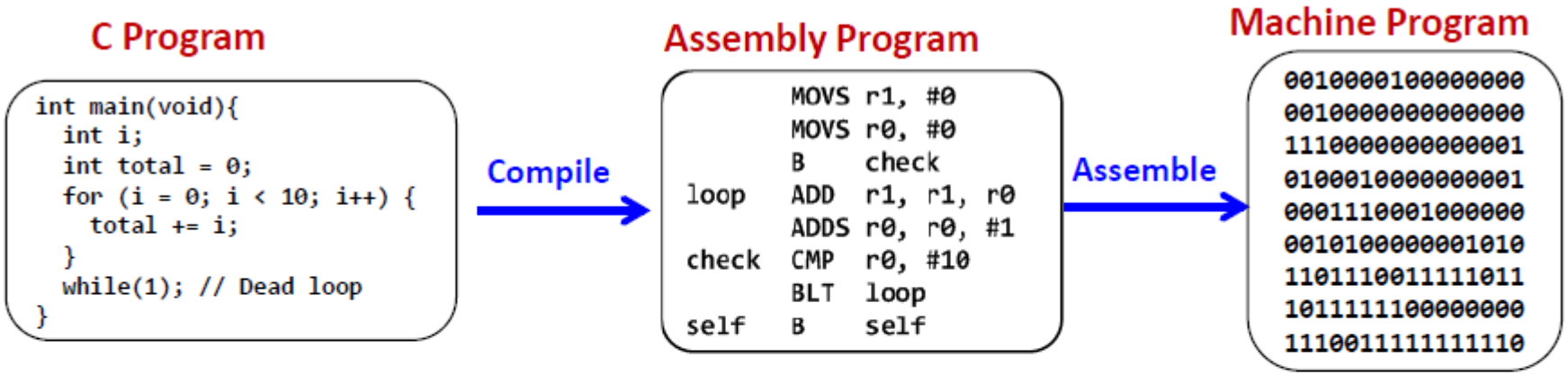
This is a possible implementation:

- Assembly is close to machine language →

```
top: cmp eax,ebx          ; check loop condition
     jae next             ; false? exit loop
     inc eax              ; body of loop
     jmp top              ; repeat the loop
next:
```

Language	Pros and cons
C/C++	Pros: • easy to learn • portable • easy handling of complex data structures Cons: • limited/no direct access to core register and stack • no direct control over instruction sequence generate • no direct control over stack usage
Assembly	Pros: • allows direct control to each instruction step and all memory operations • allows direct access to instructions that cannot be generated with C Cons: • takes longer time to learn • difficult to manage data structure • less portable (syntax of assembly language in different tool chains can be different)

Levels of program code



- **High-level language**
  - Level of abstraction closer to problem domain
  - Provides for productivity and portability
- **Assembly language**
  - Textual representation of instructions
- **Hardware representation**
  - Binary digits (bits)
  - Encoded instructions and data

Data types and size definitions in ARM-CORTEX-M processors

Table 3.2: Size of data types in Cortex®-M processors

C and C99 (stdint.h) data type	Number of bits	Range (Signed)	Range (Unsigned)
char, int8_t, uint8_t	8	−128 to 127	0 to 255
short, int16_t, uint16_t	16	−32768 to 32767	0 to 65535
int, int32_t, uint32_t	32	−2147483648 to 2147483647	0 to 4294967295
long	32	−2147483648 to 2147483647	0 to 4294967295
long long, int64_t, uint64_t	64	−(2^63) to (2^63−1)	0 to (2^64−1)
float	32	$-3.4028234 \times 10^{38}$ to 3.4028234×10^{38}	
double	64	$-1.7976931348623157 \times 10^{308}$ to $1.7976931348623157 \times 10^{308}$	
long double	64	$-1.7976931348623157 \times 10^{308}$ to $1.7976931348623157 \times 10^{308}$	
pointers	32	0x0 to 0xFFFFFFFF	
enum	8/16/32	Smallest possible data type, except when overridden by compiler option	
bool (C++ only), _Bool (C only)	8	True or false	
wchar_t	16	0 to 65535	

Table 3.3: Data size definition in ARM® processors

Terms	Size
Byte	8-bit
Half word	16-bit
Word	32-bit
Double word	64-bit

Software Development Flow

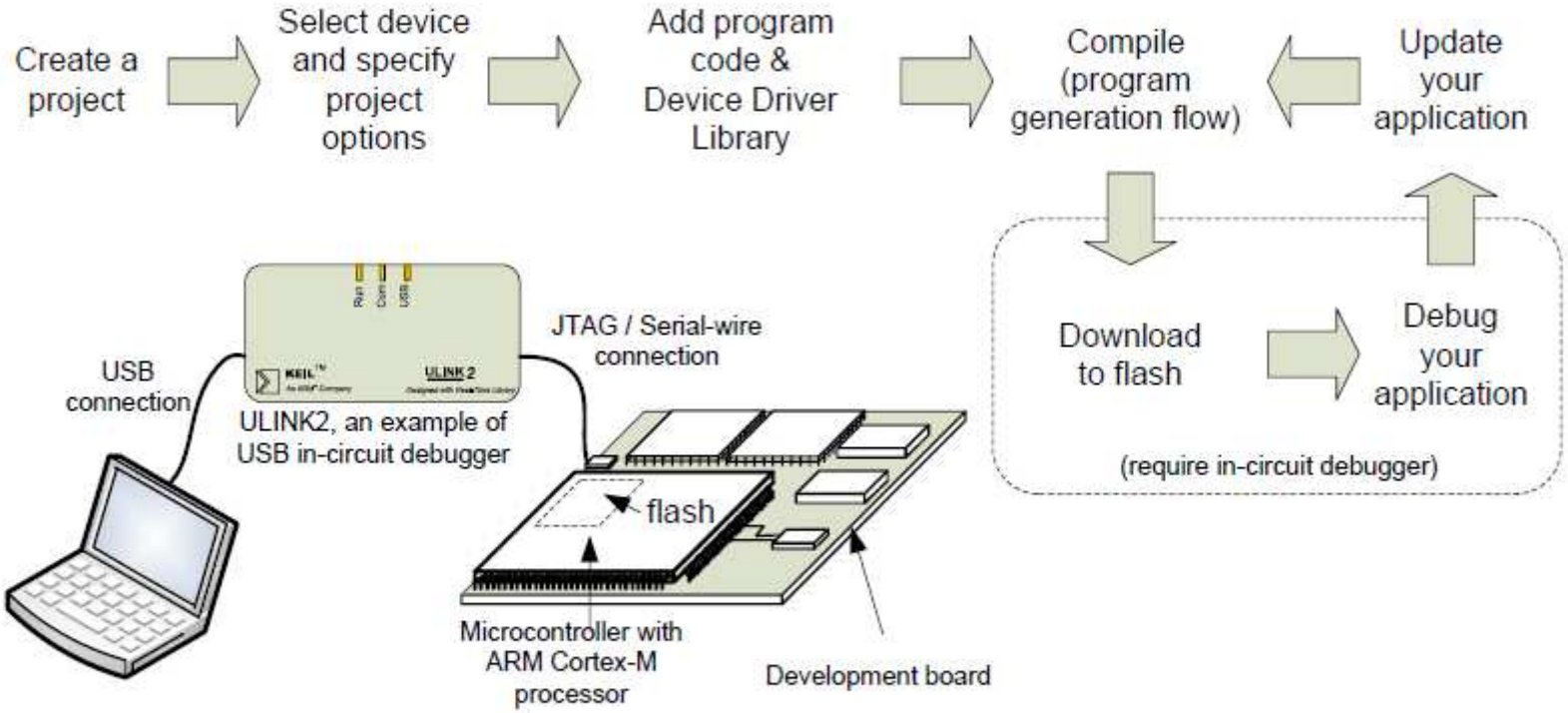
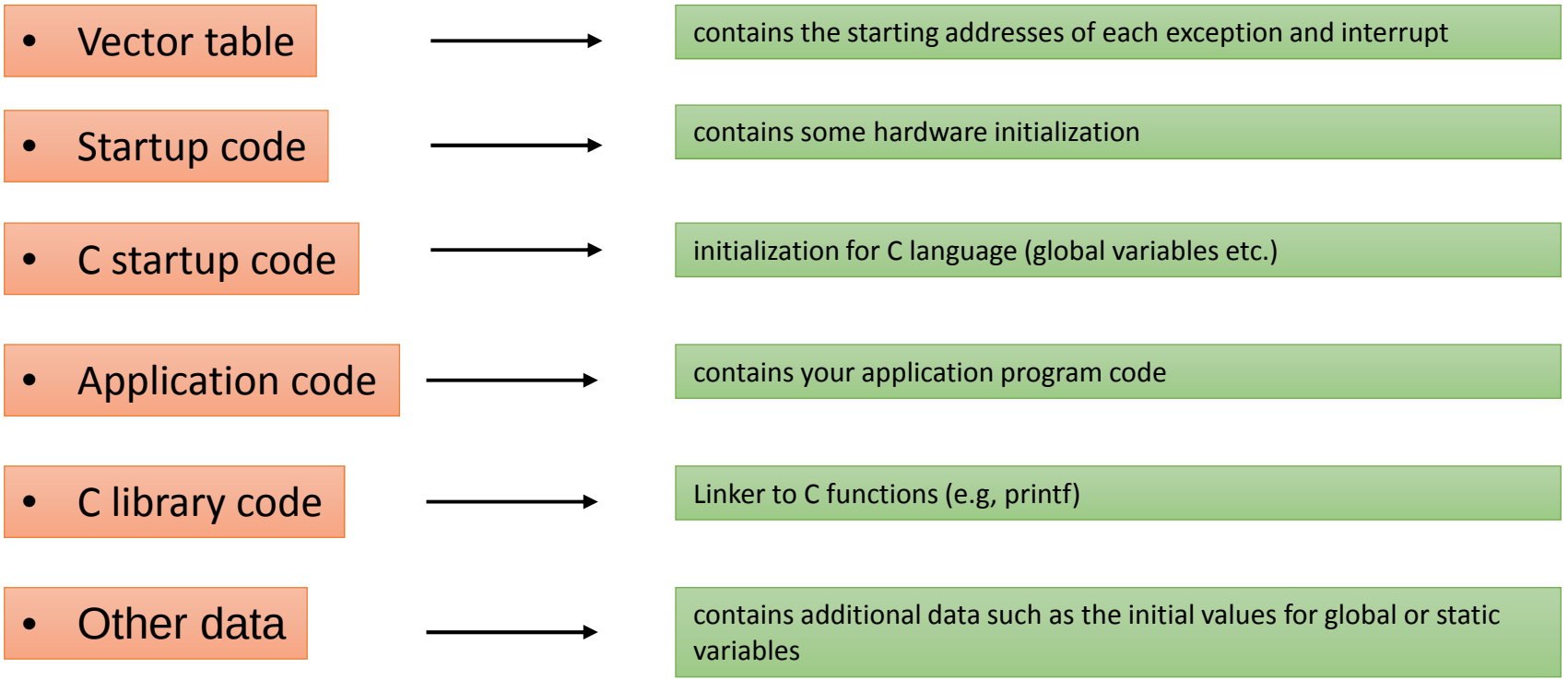


Figure 3.14
An example of development flow.

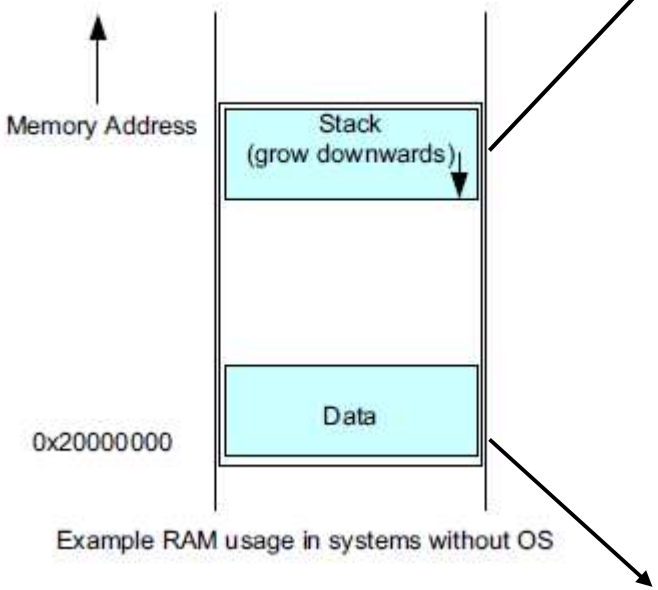
What Is Inside a Program Image?

- In addition to the program code you created, there are a range of software components inside a program image:



Most of these parts are provided by either microcontroller vendors or development toolchains. Mostly, you will deal application code only.

Data in SRAM



stack memory includes temporary data storage memory space for local variables, parameter passing in function
calls, register saving during an exception sequence

PUSH data are placed in the stack, grow downwards

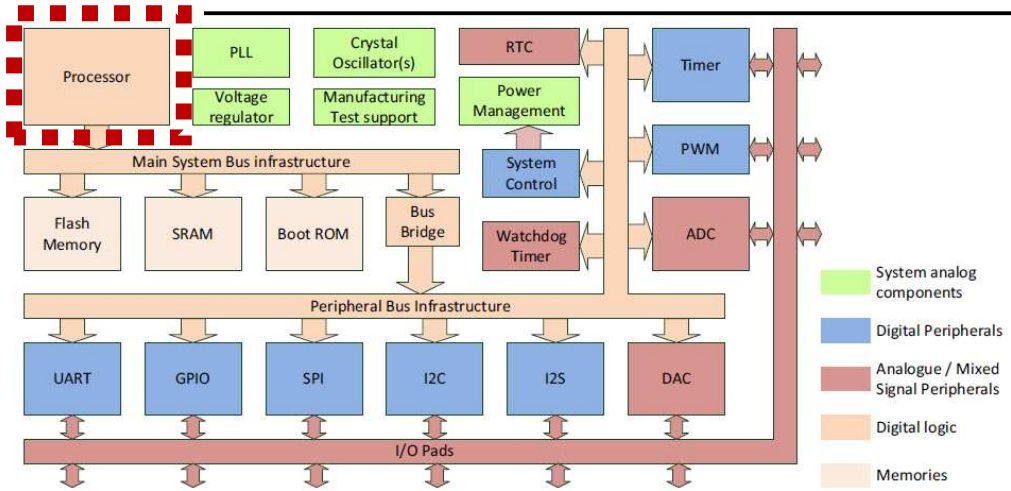
POP data are removed, stack memory size decrements

contains global and static variables

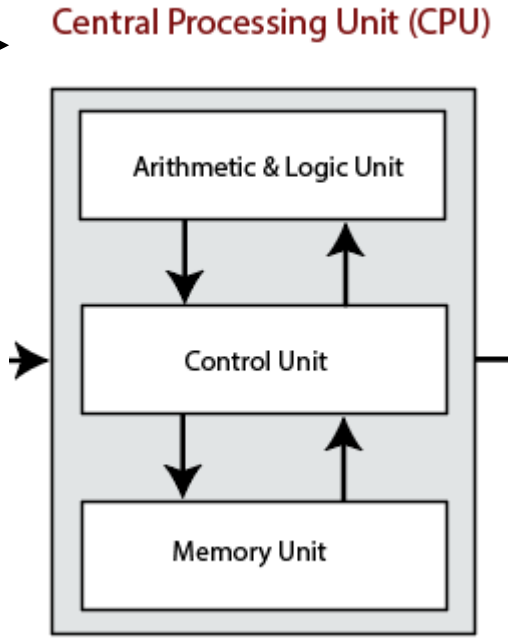
Too many push operations without popping may cause stack overflow. Stack and data regions crash.

Architecture-Registers in the Cortex-M0 and Cortex-M0+ processors

In order to perform data processing and controls, a number of registers are required inside the processor core.



A simple microcontroller.



Architecture-Registers in the Cortex-M0 and Cortex-M0+ processors

In order to perform data processing and controls, a number of registers are required inside the processor core.

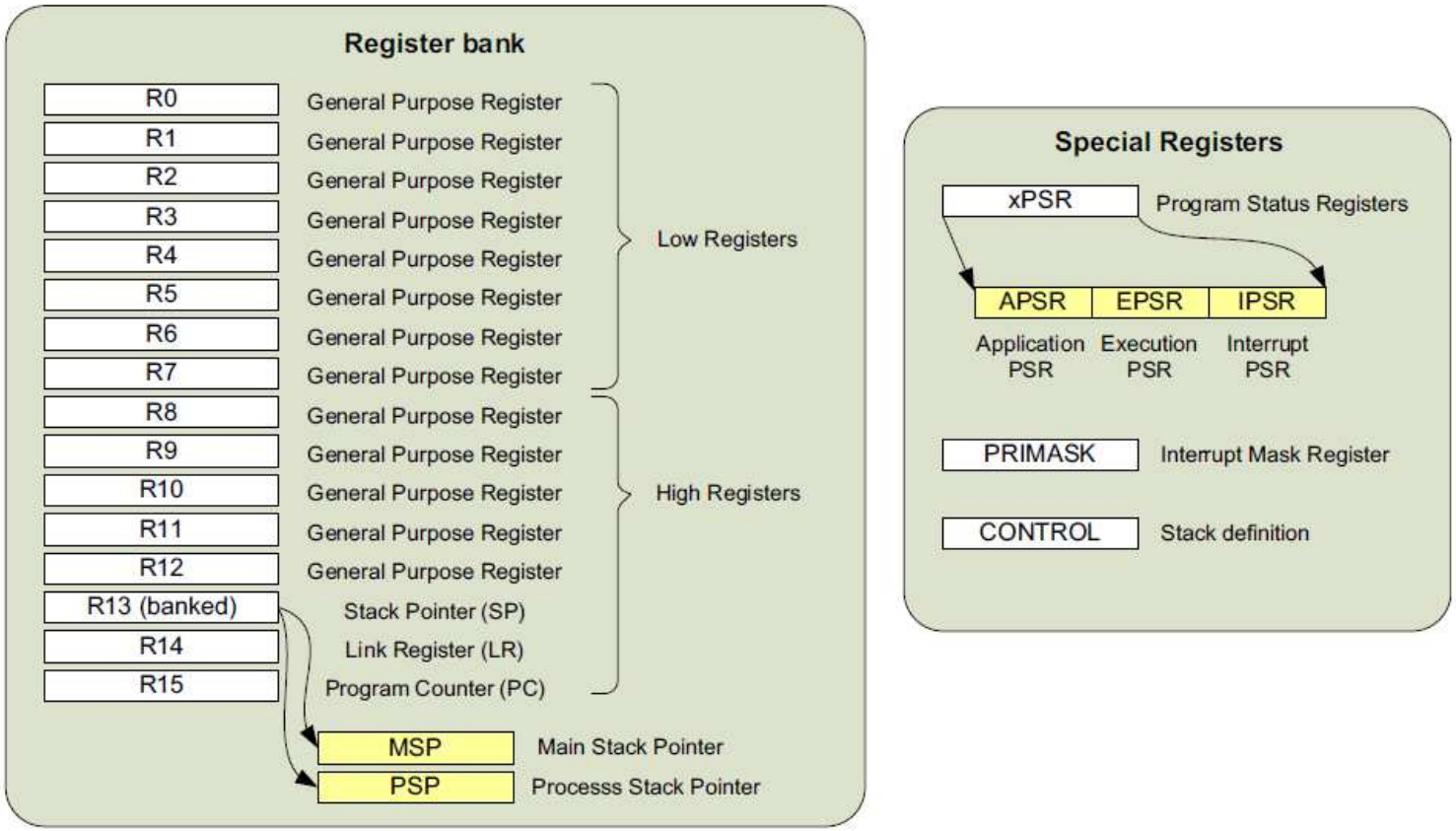


Figure 4.3
Registers in the Cortex®-M0 and Cortex-M0+ processors.

Registers in the Cortex-M0 and Cortex-M0+ processors

- R0-R12 registers are for general uses.
- R13 is stack pointer. It is used for accessing the stack memory via PUSH and POP operations. MSP is the default one and used while dealing interrupts. PSP is used in thread mode(when not handling exceptions).
- R14 is link register. It stores the return address of a subroutine or function call. At the end of the subroutine or function, the return address stored in LR is loaded into the program counter (PC) so that the execution of the calling program can be resumed.
- R15 is program counter. Current instruction address is stored here.
- xPSR is combined program status register. It provides information about program execution and the ALU flags. ALU is a unit in processor.

Registers in the Cortex-M0 and Cortex-M0+ processors

- PRIMASK register is a 1-bit wide interrupt mask register. When set, it blocks all interrupts apart from the Non-Maskable Interrupt (NMI) and the HardFault exception

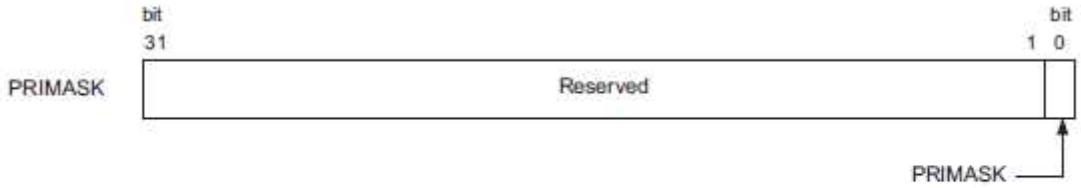


Figure 4.6
PRIMASK.

- CONTROL register is used to select one of the stack pointer. MSP or PSP. The stack pointer selection is determined by the processor.

Stack Memory Operations

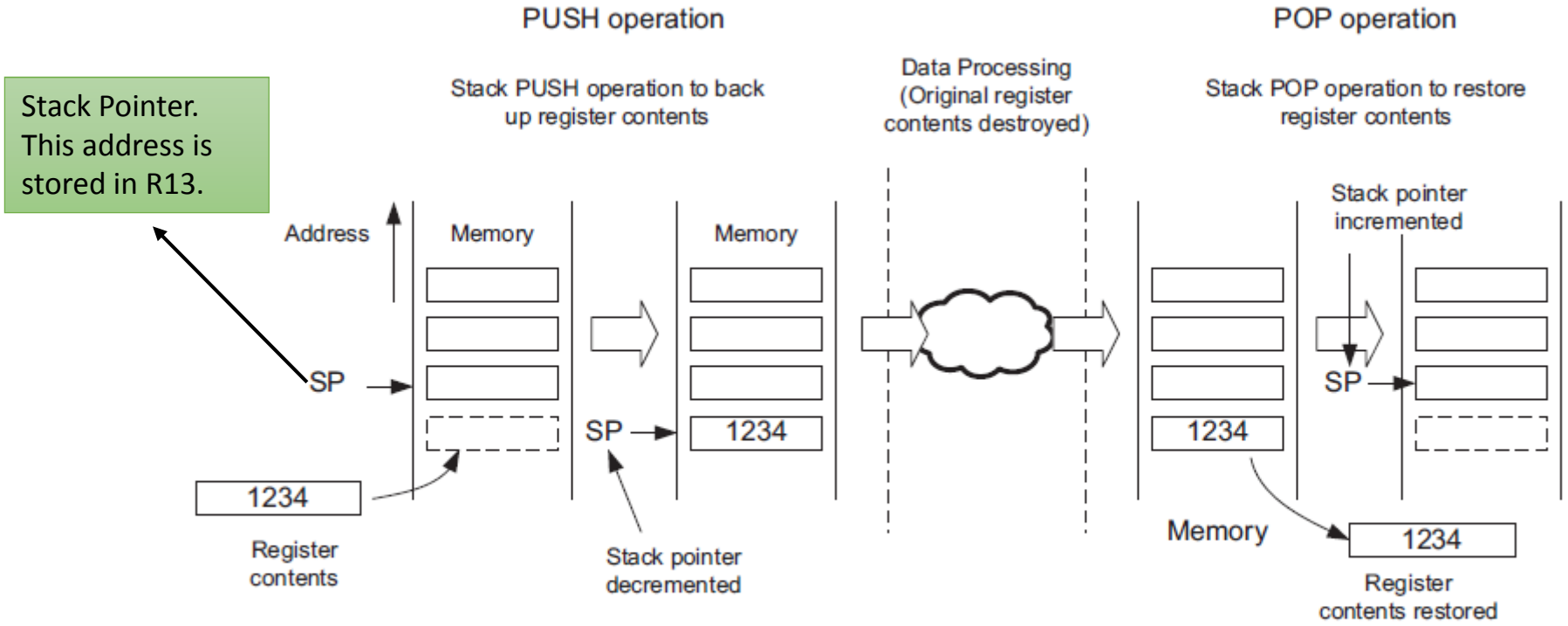
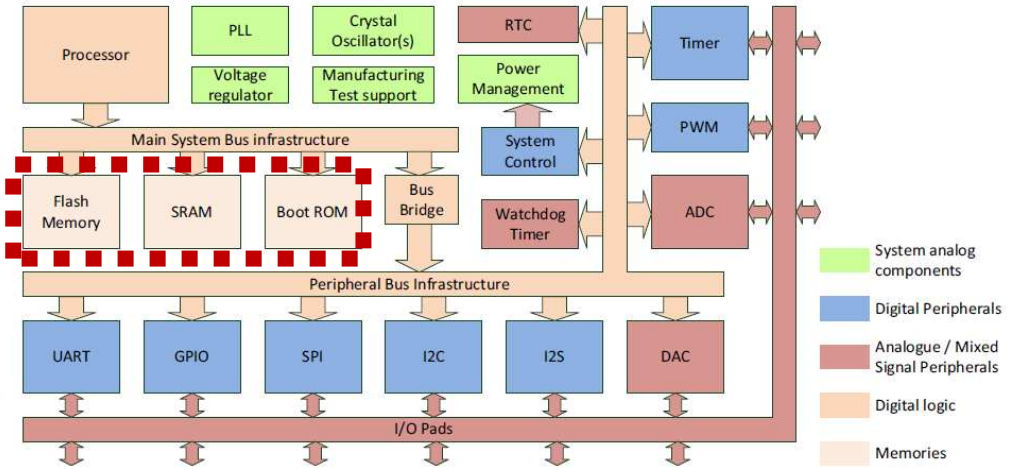


Figure 4.12
Stack PUSH and POP in the Cortex®-M processors.

- Stack memory is a temporary data storage mechanism.
- Last in- First out buffer.

Memory Systems

- All microcontroller systems need memories for storage of program code, data variables, stack memory...
- Cortex-M0 and Cortex-M0+ processors allow to have 4 GB memory space
- This memory space is architecturally divided into a number of regions (Memory mapping)



A simple microcontroller.

Memory Mapping

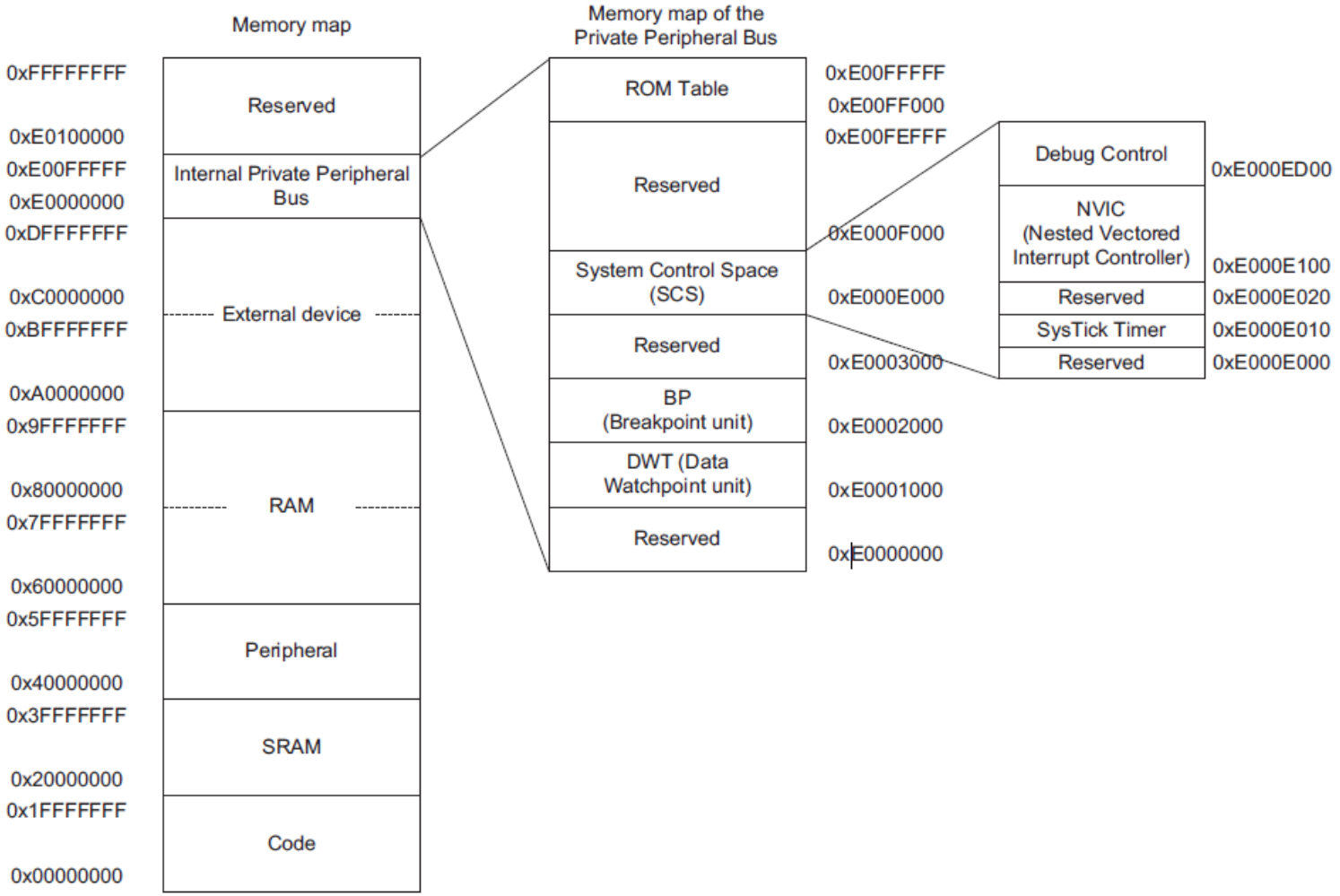


Figure 7.2
Architecturally defined memory map of the Cortex®-M0/M0+ processor.

Memory Mapping

Code Region: The size of the code region is 512 MB. It is primarily used to store program code, including the initial exception vector table at address 0x00000000 which is a part of the program image.

SRAM Region: The SRAM region is the located in the next 512 MB of the memory map. It is primarily used to store data, including stack.

Peripheral Region: The Peripheral region also has the size of 512 MB. It is primarily used for peripherals (e.g., GPIO, USART...)

RAM Region: 1 GB space, primarily used to store data.

External Device: 1 GB space, primarily used for peripherals and I/O usages.

Internal Private Peripheral Bus: The internal Private Peripheral Bus (PPB) memory space is allocated for peripherals inside the processor, such as the interrupt controller Vectored Interrupt Controller (NVIC) and debugging.