Kuwait University
College of Engineering and Petroleum

جامعة الكويت
KUWAIT UNIVERSITY

ME319 MECHATRONICS

Part I: The Brains – Microcontrollers, Software and Digital Logic

Lecture 7: Interrupts

Spring 2021

Ali AlSaibie

Lecture Plan

- Objectives:
  - Introduce the concepts of timers
  - Overview some of different ways interrupts are used on a microcontroller

Interrupts

- Whole purpose of MCU devices is to respond to external stimuli by controlling mechanical device
- Question arises: How do we program device to monitor and/or respond to external stimulus?
- Examples:
  - *Run a motor every time someone presses a button*
  - *Each time a sensor signal goes high, sound an alarm*
  - *Turn a stepper motor until a limit switch activates*
  - *Each time a GPS signal is received, log data to flash memory*

جامعة الكويت
KUWAIT UNIVERSITY

Polling vs Interrupts

- One mechanism to do this is called polling or busy-wait synchronization
- Idea: Continuously check a variable in a loop to see if it has changed
  - *Combine with if statement to take action if it has*
- Example:
  - *Loop and continuously check if user presses button*
  - *If so, do something*
  - *Then wait until user releases button*

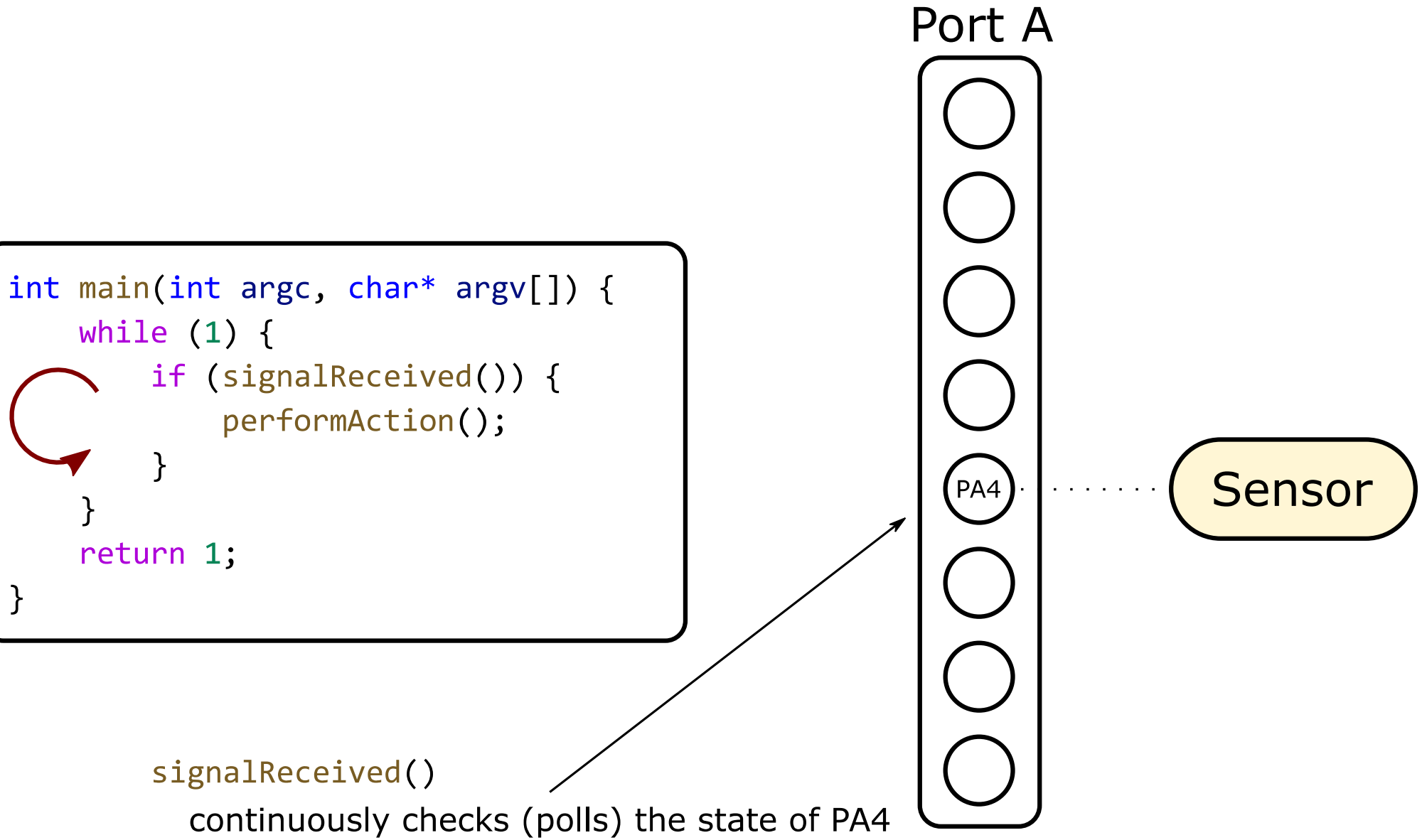
جامعة الكويت
KUWAIT UNIVERSITY

Polling

- What are the drawbacks of polling?
  - Software is tied up checking a certain address and cannot do other tasks (inefficient computation)
  - If we are waiting on multiple possible events, cumbersome to establish priority
  - If software is waiting for long periods, extremely inefficient use of power (inefficient power consumption)
    - *Ideally, would like to be able to put MCU in low power mode while waiting*

Polling

```
int main(int argc, char* argv[]) {
    while (1) {
        if (signalReceived()) {
            performAction();
        }
    }
    return 1;
}
```



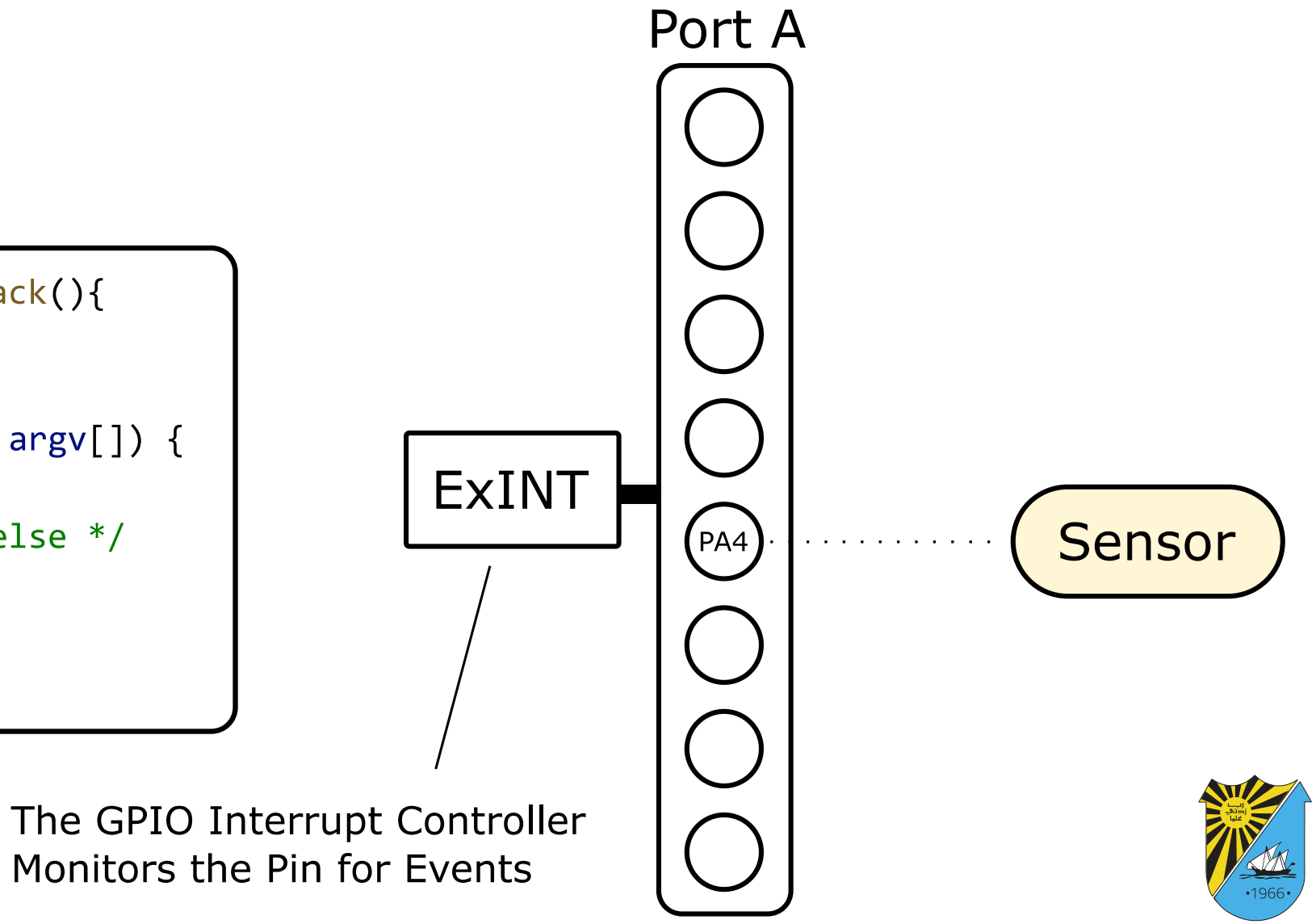
`signalReceived()`
continuously checks (polls) the state of PA4

Interrupts

- A way for software/processor/peripheral to flag/notify each other
- Critical and powerful feature of processors
  - *Allow the implementation of real-time computing*
- Commonly used to “interrupt” the processor
  - *The processor would then have to service the interrupt*
- Peripherals can use interrupts to indicate status
  - *Can be polled by software*
  - *Or, configured to interrupt the processor.*
- An interrupt is an asynchronous switch in processor execution
  - *Asynchronous means that it occurs on demand, not with some specific timing (like polling)*

Interrupt Execution

```
void interrupt_PA4_callback(){
    performAction();
}
int main(int argc, char* argv[]) {
    while (1) {
        /* Do something else */
    }
    return 1;
}
```

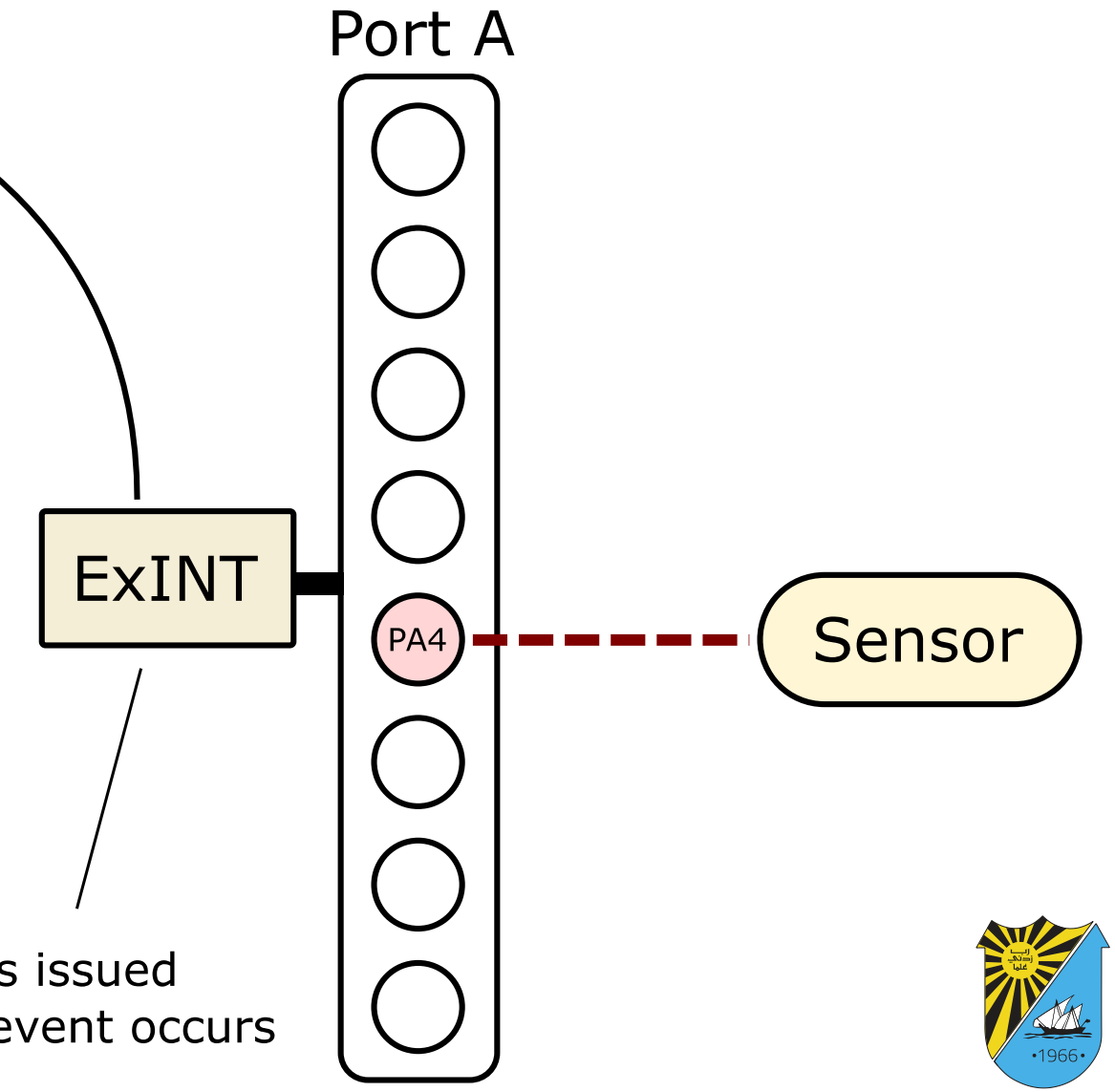


The GPIO Interrupt Controller
Monitors the Pin for Events

Interrupt Execution

A callback function is executed in response to the interrupt flag

```
void interrupt_PA4_callback(){
    performAction();
}
int main(int argc, char* argv[]) {
    while (1) {
        /* Do something else */
    }
    return 1;
}
```



GPIO Switch Example Using Interrupts

```
/* Example: Interrupt Based LED Switch (Soft-Latching)
*  Using the arduino API, it's very convenient to setup an external event interrupt
*/
#include <Arduino.h>
/* Callback function for whenever the User Button is pressed */
void switch_callback(void) { digitalToggle(LED_BUILTIN); }

void setup() {
    pinMode(USER_BTN, INPUT);
    pinMode(LED_BUILTIN, OUTPUT);
    /* attach the Low/Falling event (Negative logic on Nucleo Button)
     * to a callback function.
     * */
    attachInterrupt(USER_BTN, switch_callback, LOW);
}
void loop() {/* Nothing else to be done */}
```

Interrupts on Peripherals

- Most hardware features on the MCU can generate interrupts and can have an associated ISR
  - Analog-to-digital converter
  - Flash memory controller
  - Timers
  - UARTs
  - GPIO ports
  - I2C
  - Serial Peripheral Interface
- Basically, any peripheral that would benefit from flagging the CPU on specific events, likely has an interrupt issuing capability.

جامعة الكويت
KUWAIT UNIVERSITY

Interrupts

- By default, if you do not explicitly enable an interrupt, code executes serially, and no interrupts will occur
  - *Except for system level interrupts (faults, power loss, stack overflow, etc)*
- Setting up an interrupt and defining appropriate ISR to run when interrupt occurs is your responsibility
- Deciding when to use interrupts, and when to use polling, is your choice as software developer
  - *Use polling when I/O structure is simple and fixed*
  - *Use interrupts when I/O timing is variable and/or structure is complex*

جامعة الكويت
KUWAIT UNIVERSITY

Interrupts Priority

- Potentially, your code may have several interrupts enabled at one time
  - i.e., you have one interrupt for ADC and one interrupt for edge-triggered GPIO
  - What happens if both events happen at same time? While ISR runs?
    - Or, if GPIO event happens when ADC ISR is executing?
- This is why you must define interrupt priority when enabling an interrupt

Interrupts Priority

- Priority rules:
  - *If an ISR of higher priority is running and lower priority interrupt is triggered, ISR of lower priority will wait for the higher priority ISR to finish, then start afterwards.*
  - *If an ISR of lower priority is running and higher priority interrupt is triggered, ISR of higher priority takes over, runs to completion, and returns execution to lower priority ISR*
- ISR is akin to an interrupt **callback** function. The reason it is called an "interrupt service routine" is that once it is executed the interrupt flag is cleared: *it has been serviced, and then it lives happily ever after, until it is flagged again that is.*

جامعة الكويت
KUWAIT UNIVERSITY

Nested vectored interrupt controller (NVIC)

NVIC features

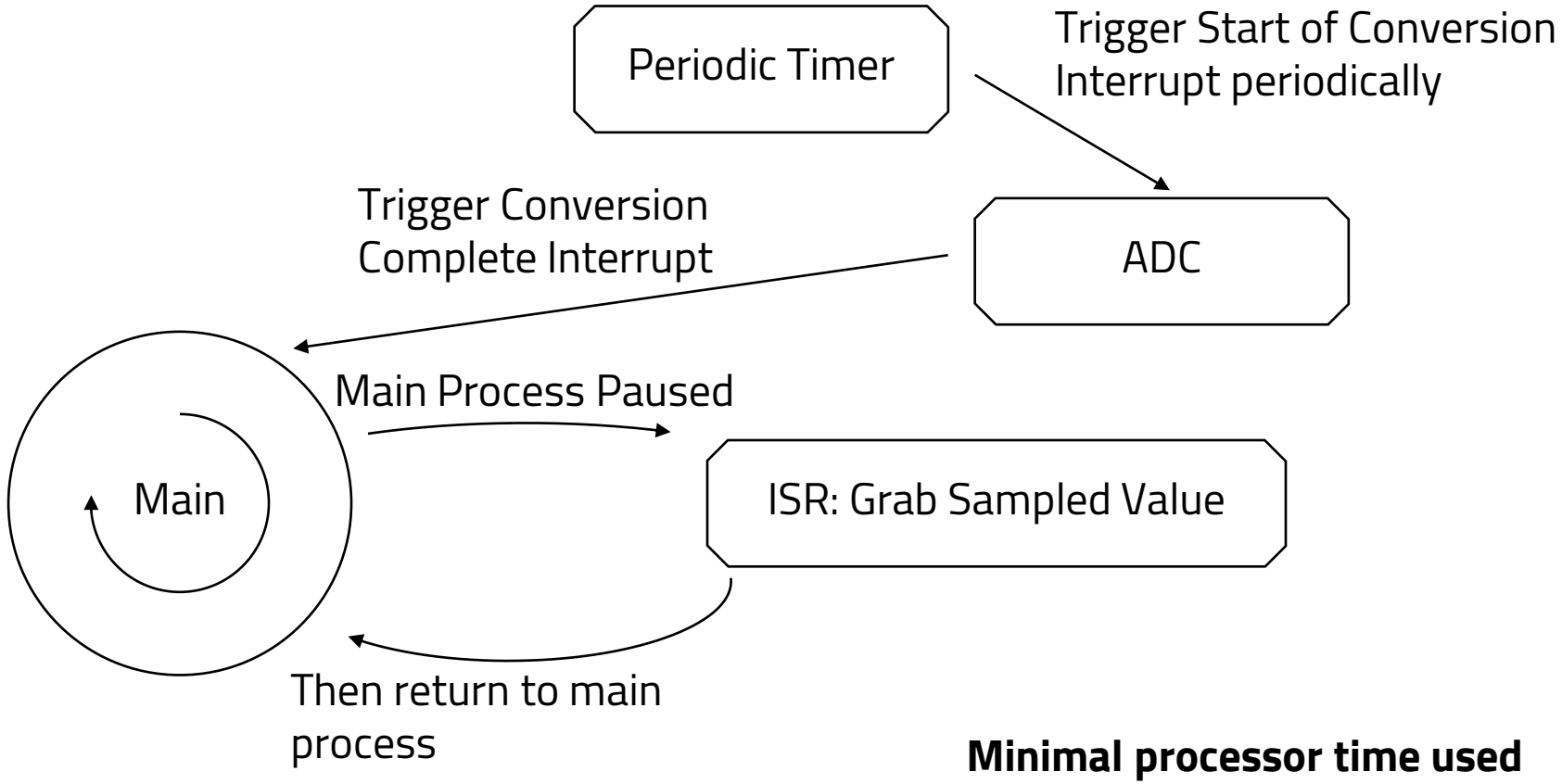
The nested vector interrupt controller NVIC includes the following features:

- 52 maskable interrupt channels (not including the 16 interrupt lines of Cortex®-M4 with FPU)
- 16 programmable priority levels (4 bits of interrupt priority are used)
- low-latency exception and interrupt handling
- power management control
- implementation of system control registers

- Sometimes you really do not want main code to pause execution
  - For instance, if you are doing some sort of critical task
  - Examples:
    - *Sending a control input to a mechanism*
    - *Sending a serial packet via UART*
    - *Initializing port configurations*
- Then disable all interrupts in this code section, and enable again when finished with critical section

جامعة الكويت
KUWAIT UNIVERSITY

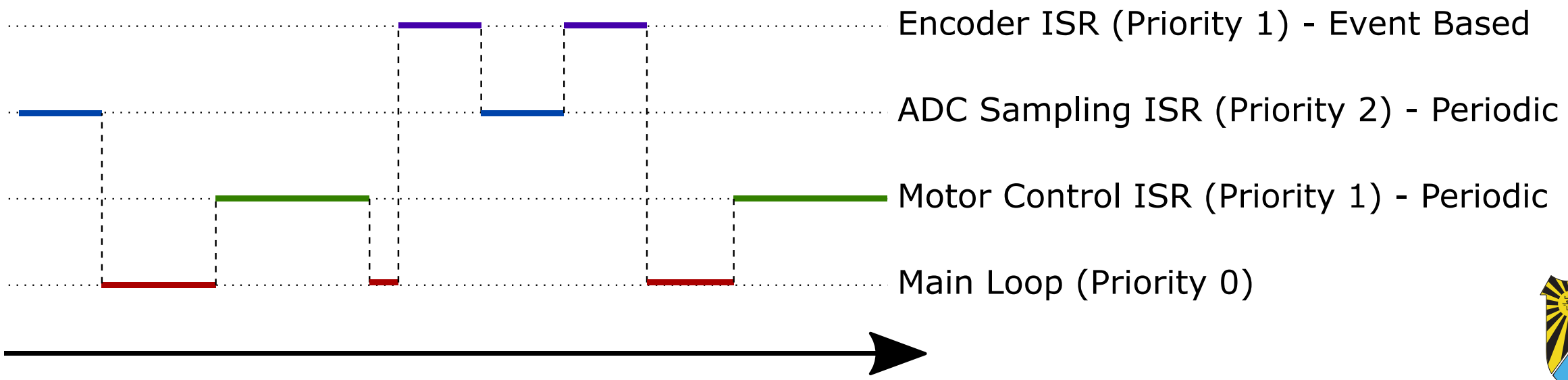
- Using interrupts to trigger ADC Conversion



Minimal processor time used

Interrupts: Processor Interrupt-Based Threading

- Single core CPUs can only do one instruction at a time. One way to parallelize the program is through the use of scheduled (periodic) and event-based interrupts, with different priority levels.
- Higher priority ISRs can pause lower priority ISRs.
  - Except for Non-Maskable interrupts (usually reserved for hardware errors)



Interrupts from TIM Peripheral

- The TIM peripheral covered previously can generate several types of interrupts, including:
  - Overflow Interrupt: Every time the counter completes one lap (period completed).
    - *This is the interrupt used to issue a periodic callback function*
  - Input Capture Interrupt: Every time an edge is detected from an input.
    - *This can be used to measure input signal frequency.*
  - Output Compare Interrupt: Every time the compare value is reached.

جامعة الكويت
KUWAIT UNIVERSITY