CMPE-013/L

Introduction to “C” Programming

Gabriel Hugh Elkaim
Winter 2015

Hardware Peripherals

Digital pins
Timers
ADC
Hardware Peripherals

- Communications
- Pin change notification
- DMA
- Output compare
- Input capture
- Digital pins
- Timers
- ADC

Special function registers

- Peripherals are controlled by hardware registers
  - Referred to as Special Function Registers (SFRs)
- Memory-mapped unsigned 16-bit integers
- Accessible as global variables
  - Included from the `<xc.h>` header
Hardware Peripherals

Special function registers

- Declaration of Interrupt Flag SFR 0 register
- `volatile` qualifier indicates value can change outside of the code in this program
- `__attribute__` is a compiler directive to specify additional compiler parameters
  - `__sfr__` indicates that it's a memory-mapped SFR

```c
extern volatile unsigned int IFS0 __attribute__((__sfr__));
```

Hardware Peripherals

Digital pins

- Voltage
  - High
  - Low
- Direction
  - Input
  - Output
- Polling interface

\( TRIS_x < \quad 0 - \text{output} \quad 1 - \text{input} \)
Hardware Peripherals

Digital pins

- **TRIS** – TRIState register. Sets pin direction.
  - Pin is an output when corresponding bit is 0, input when corresponding bit is 1
- **LAT** – LATch register. Sets pin value/gets pin's desired value
  - Desired output value of the pin
- **PORT** – PORT register. Sets pin value/gets pin's actual value
  - Actual value of the pin
Hardware Peripherals

Timers

- **Multiple 16-bit timers**
  - 5 total
- **Interrupt-based**
  - ISR is called every X seconds
- **Configurable periodicity**
  - Range from 20MHz to 305Hz
Hardware Peripherals

Timer SFRs

- **TMRx** - Timer counter
  - `uint16`
  - Ticks every instruction clock cycle (20MHz)

- **PRx** - Timer x prescalar
  - Limit for when to trigger the timer interrupt.
  - Valid values are [1, INT16_MAX]
  - 0 is a special value, disables peripheral.

Hardware Peripherals

Timers

- To modify timer interrupt period, set PRx register.
- To set a period of the timer interrupt:
  - 20MHz / PRx = periodicity
- PRx of 20000 -> 1kHz interrupts
Hardware Peripherals

Timers

- Analog to Digital Converter
- Measures the voltage of a processor pin
- Used to read analog sensors
  - Temperature
  - Power
  - Battery levels
Hardware Peripherals

ADC SFRs

- ADCxBUFy: Buffer for holding samples
  - x is the ADC
  - y is the sample [0, 7]
  - 16-bit unsigned value
  - Only lowest 10-bits matter

Hardware Peripherals

ADC

- The input signal is continuously sampled
- Every 8th sample triggers an interrupt
Hardware Peripherals

ADC

- Voltage range from $V_{ref^-}$ to $V_{ref^+}$
  - 0V to 3.3V
- Values are unsigned 10-bits, from [0, 1023]
- Units are in $V_{ref}/1023 = 0.0032V$

Hardware Peripherals

ADC
Hardware Peripherals

ADC

event

ADC1BUF0 = 2
ADC1BUF1 = 146
ADC1BUF2 = 288
ADC1BUF3 = 420
ADC1BUF4 = 563
ADC1BUF5 = 691
ADC1BUF6 = 829
ADC1BUF7 = 987

event

Hardware Peripherals

ADC

event

ADC1BUF0 = 950
ADC1BUF1 = 600
ADC1BUF2 = 100
ADC1BUF3 = 65
ADC1BUF4 = 81
ADC1BUF5 = 93
ADC1BUF6 = 107
ADC1BUF7 = 122
Event-driven Programming

- Real-time programming paradigm
- Build around the concept of events
- Events are then handled by specific event handlers
- Works well with systems with multiple inputs that need to be handled in a timely manner
  - Real-time system
- Integrates well with interrupts
Event-driven Programming

Events

- Any temporally-short sensor occurrence
- Usually the derivative of a signal
  - Button was pressed down
  - The mouse was clicked
  - This sensor value changed
  - This interrupt triggered

Event-driven Programming

The event loop

- A continual loop that checks for and processes events
- The core of an event-driven program

Check Events()
Event-driven Programming

Event priorities

```c
while (1) {
    // Check for event 3
    // Process event 3

    // Check for event 1
    // Process event 1

    // Check for event 2
    // Process event 2
}
```

Event-driven Programming

Real-world example

```c
while (1) {
    if (buttonsEvent) {
        // Update fixed LED mask
    }
    if (adcEvent) {
        // Update OLED
    }
    if (timerEvent) {
        // Update bouncing LED mask
    }
    if (ledEvent) {
        // Update LEDs
    }
}
```
Event-driven Programming
Real-world example

```c
static uint8_t buttonsEvent;

void main()
{
    while (1) {
        if (buttonsEvent) {
            // Event loop
        }
    }
}

void _ISR Timer1Int(void)
{
    buttonsEvent = ButtonsCheckEvents();
    IFSO &= ~(1 << 3);
}
```

Bit manipulation

- Bit masking
- Bit flags
- Bit fields
Bit manipulation

Bit packing

- Data is commonly packed into larger unsigned integers on embedded systems
- Generally a tie in to hardware or when space is critical
  - Hardware
  - Storage
  - Binary formats

C1CTRL1 – dsPIC33EP256MC502