MACQUARIE University
SYDNEY·AUSTRALIA

Introduction to Embedded Systems

ELEC3042 EMBEDDED SYSTEMS

Lectorial 1

Unit Description

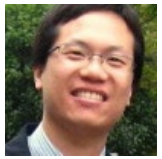
- Project-based unit.
- Students complete a major project that emphasize aspects of digital computing systems, including state machines, digital data processing, arithmetic processing, timing, internal and external peripherals.
- Students will design a program for a microcontroller that will perform processing of real world data to achieve a defined aim. This programming exercise will be used to explore the complexities that make up digital hardware designs.

Learning Outcomes

MACQUARIE University
SYDNEY·AUSTRALIA

1. Describe the various components that comprise a modern embedded system, including those that are essential and those that are optional
2. Distinguish between the different external and internal interfaces and select which is most appropriate for a given circumstance
3. Interface a CPU with both internal and external functional units
4. Program an embedded system in either the assembly or C languages
5. Construct state machines on an embedded system

Teaching Staff



Unit Convenor
Dr Alan Kan

Unit Convenor
Dr Rex Di Bona

Sessional Academic
Richard Vu

Sessional Academic
Gerry O'Connor

Project-based learning

- Working on the projects should raise questions
- Learning how to learn
- Find answers by:
  1. Attending lectorials and asking questions
  2. Ask in lab
  3. Look in the textbooks, datasheets
  4. Posting questions to the Peer-Assisted iLearn Forum – answer other student's questions
  5. Ask Google
- Use Important Private Messages to Unit Contacts for personal messages only

Weekly Schedule

MACQUARIE University
SYDNEY·AUSTRALIA

Week	Lectorial	Practical	Textbook*	Datasheet	Assessment
1	Introduction to Embedded Systems	MPLAB X	1, 2, 3	1, 2, 7, 14	
2	Bit Logic and Interrupts	Hardware Interaction & Interrupts	4, 6, 8	12, 13, 14	
3	Timers	Timers	9	9, 15, 16, 17, 18	
4	Embedded Systems Design	7-segment display & State Machines			Quiz 1
5	Interfacing with an analog world	ADC & PWM	7, 10	15, 16, 17, 18, 24	
6	Debugging, Testing and UART	MPLAB X Simulator & UART	5	20	
7	SPI & I2C	SPI & I2C	16, 17	19, 21, 22	Minor Project Due
8	Event Driven Programming	Work on Major Project		10	Quiz 2
9	Memory	Work on Major Project	18, 19	26, 27, 18	Major Project Design Review
10	Compilers & Assembly	Work on Major Project			
11	Q&A				Major Project Due
12					Major Project Defence
13					Major Project Defence

* Make: AVR programming by Elliot Williams – available online through library

Assessments

- All assessments are due at 11:55 pm (except those designated as in-class).
- Check unit guide for late submission and special consideration policies
- Project submission boxes have multiple components – use the right parts

ASSESSMENT NAME	WEIGHT	DUE DATE	SUBMISSION METHOD	RETURN DATE	LEARNING OUTCOMES
Quiz 1	5%	15/03/2023	iLearn Quiz	16/03/2023	1, 2, 3
Minor Project - Report & Code	8%	5/04/2023	Turnitin	24/04/2023	3, 4, 5
Minor Project - Demonstration	12%	6/04/2023	In Class Demonstration	6/04/2023	3, 4, 5
Quiz 2	15%	26/04/2023	iLearn Quiz	27/04/2023	1, 2, 3, 4
Major Project - Design Review	15%	04/05/2023	On Zoom during class time	13/05/2023	2, 3, 5
Major Project - Answers to Defence Questions & Code	-	17/05/2023	Turnitin	-	2, 3, 4, 5
Major Project - Demonstration	20%	18/05/2023	In Class Demonstration	19/05/2023	3, 4, 5
Major Project - Defence	25%	Week 12 & 13	On Zoom during class time	8/07	2, 3, 4, 5

Assessment Tasks: Projects

MACQUARIE University
SYDNEY·AUSTRALIA

- Two Projects
  - Minor (15-20 hours) – apply basic skills to a small design
  - Major (50-60 hours) – complex system that utilises full range of internal and external units
- Read specification documents carefully
  - Post questions to the Peer Assisted Learning Forum (NOT email or private message to Unit contacts)
  - Do not make assumptions
- Discussion among students is encouraged but must be individual work
  - You must not share your code or written reports with other students

Assessment Tasks: Quizzes

- Designed to be milestone assessments
- Provides self-evaluation of understanding of basic knowledge
- Multiple choice & short answers
- If you fail quiz 1 => you are falling behind
- If you fail quiz 2 (hurdle) => you are advised to withdraw
- Pre-lab quizzes are good practice

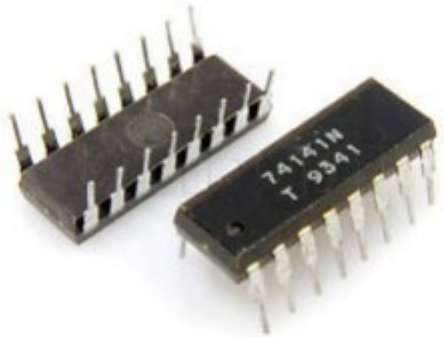
Required Hardware/Software

MACQUARIE University
SYDNEY·AUSTRALIA

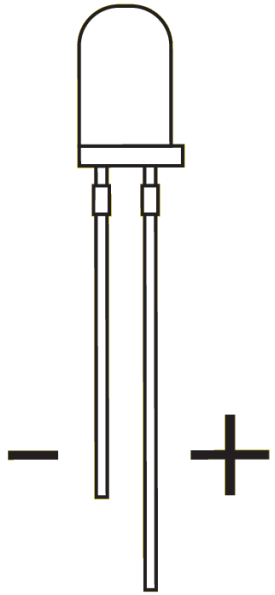
- Arduino Uno Kit
  - Buy from University
  - Build your own (see Arduino Kit Component List under Resource Materials on iLearn)
- Laptop computer
  - Preferably running Windows
  - Install MPLAB X and Arduino IDE (must be install version, not app) – follow instructions on iLearn

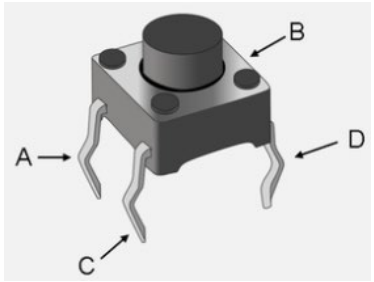
Safety

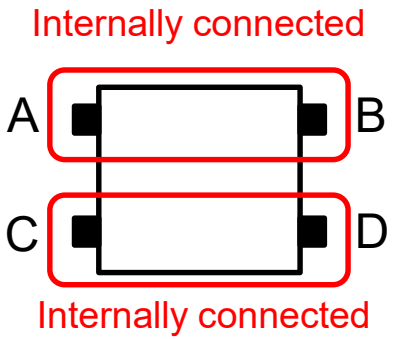
- The components you will be using have sharp points. Always handle them appropriately
  - Hold ICs by the plastic body
  - Avoid touching the legs
  - Do not touch the sharp points of ICs, LEDs, resistors or wires for both electrostatic discharge and personal safety reasons

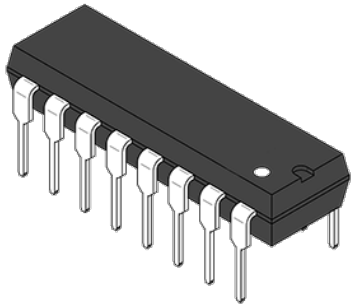


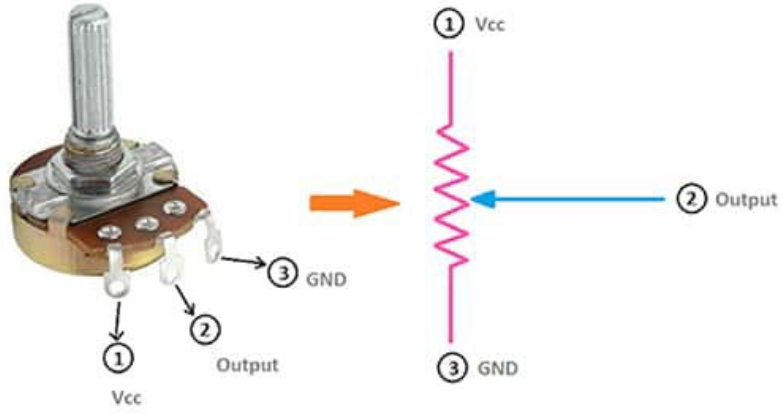
Components



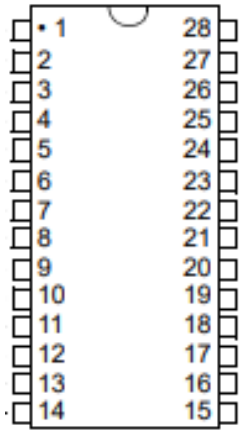








MCP23S17



Cohort Survey

https://forms.office.com/r/YUG9uay8dH

- We'd like to get to know you better as a cohort
- Understand your concerns and worries
- Survey is anonymous

Questions?

Embedded System

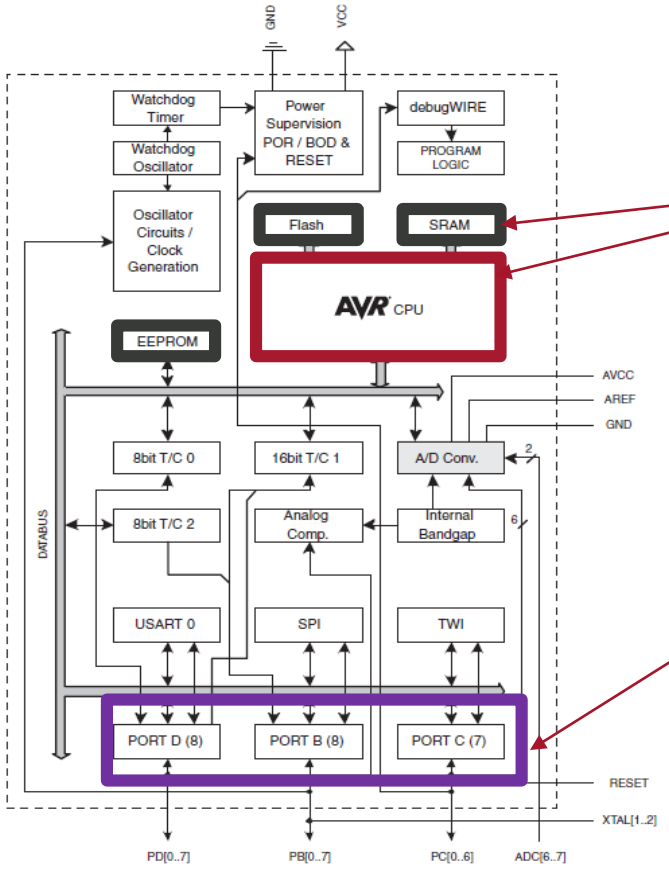
MACQUARIE University
SYDNEY·AUSTRALIA

- A computer system that runs ONE fixed program
- Part of a larger system that has a non computational focus
- Rarely, if ever, updated or maintained
- Examples: swipe card locks, alarm systems, smart lights

ATmega328p Microcontroller

MACQUARIE University
SYDNEY·AUSTRALIA

Figure 2-1. Block Diagram



Microcontrollers are complete computers, with **processor**, **memory** and **I/O subsystems** on a single chip

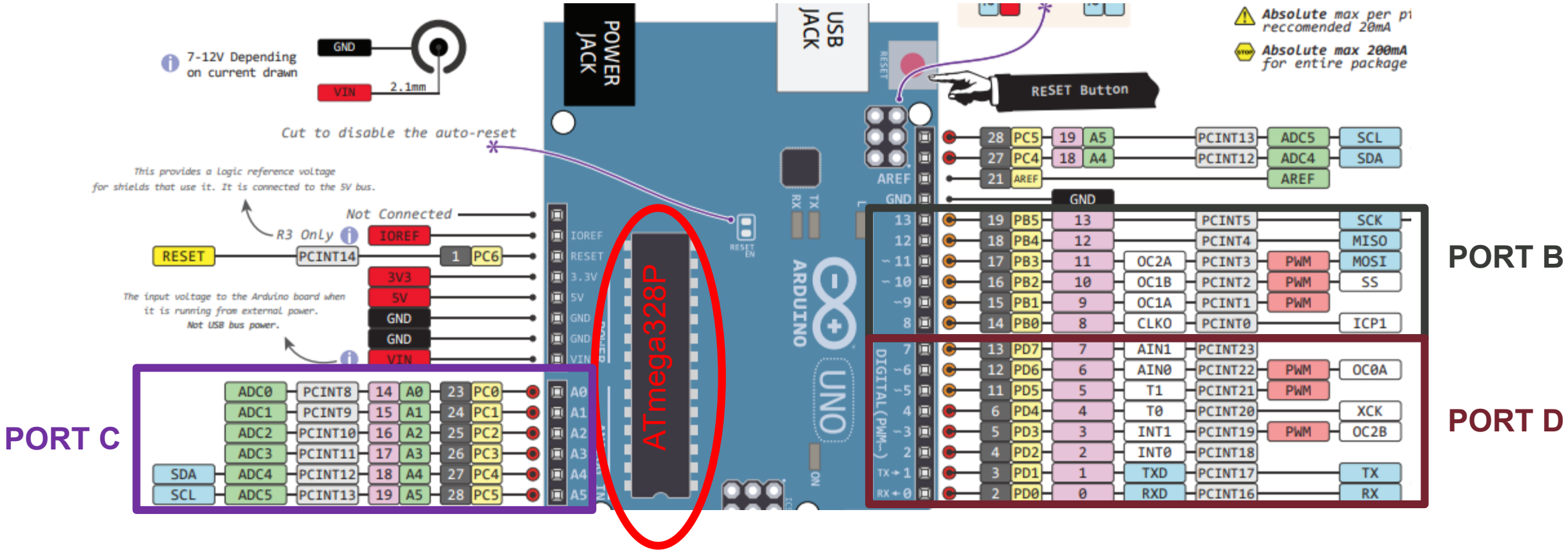
Table 2-1. Memory Size Summary

Device	Flash	EEPROM	RAM	Interrupt Vector Size
ATmega48A	4KBytes	256Bytes	512Bytes	1 instruction word/vector
ATmega48PA	4KBytes	256Bytes	512Bytes	1 instruction word/vector
ATmega88A	8KBytes	512Bytes	1KBytes	1 instruction word/vector
ATmega88PA	8KBytes	512Bytes	1KBytes	1 instruction word/vector
ATmega168A	16KBytes	512Bytes	1KBytes	2 instruction words/vector
ATmega168PA	16KBytes	512Bytes	1KBytes	2 instruction words/vector
ATmega328	32KBytes	1KBytes	2KBytes	2 instruction words/vector
ATmega328P	32KBytes	1KBytes	2KBytes	2 instruction words/vector

Program Storage Non-volatile storage Variables

ATmega48A-PA-88A-PA-168A-PA-328-P-DS-DS40002061B.pdf

Arduino Uno



Arduino_Uno_pinout_V2.pdf

Basic Embedded Systems Program in C

MACQUARIE University
SYDNEY·AUSTRALIA

```c
#include <xc.h>

/**
 * The delay routine is calibrated for an Arduino Uno 16MHz
 *
 * @param num number of ms to delay
 */
void delay_ms(uint16_t num) {
    while (num--) {
        for (volatile long x = 0; x < 468; x++) {
            ;
        }
    }
}

/*
 * We set the Data direction register to specify which pins are inputs and
 * which are outputs. A 0 bit indicates the corresponding pin is an input,
 * a 1 indicates the corresponding pin is an output
 */
void setup() {
    DDRB |= 0b00100000;     // PORTB pin 5 (D13) output
    PORTB &= 0b11011111;    // turn off LED
}

int main(void) {
    setup();    // set up the physical hardware
    uint16_t delay = 1000;
    while (1) {
        delay_ms(delay);
        PORTB ^= 0b00100000;    // invert LED
    }
}
```

- Blinks LED connected to PB5

Basic Embedded Systems Program in C

MACQUARIE University
SYDNEY·AUSTRALIA

```c
#include <xc.h>

/**
 * The delay routine is calibrated for an Arduino Uno 16MHz
 *
 * @param num number of ms to delay
 */
void delay_ms(uint16_t num) {
    while (num--) {
        for (volatile long x = 0; x < 468; x++) {
            ;
        }
    }
}

/*
 * We set the Data direction register to specify which pins are inputs and
 * which are outputs. A 0 bit indicates the corresponding pin is an input,
 * a 1 indicates the corresponding pin is an output
 */
void setup() {
    DDRB |= 0b00100000;     // PORTB pin 5 (D13) output
    PORTB &= 0b11011111;    // turn off LED
}

int main(void) {
    setup();    // set up the physical hardware
    uint16_t delay = 1000;
    while (1) {
        delay_ms(delay);
        PORTB ^= 0b00100000;    // invert LED
    }
}
```

#include

- Header files
  - xc.h
  - avr/io.h
  - avr/interrupt.h
  - avr/sleep.h

Basic Embedded Systems Program in C

MACQUARIE University
SYDNEY·AUSTRALIA

```c
#include <xc.h>

/**
 * The delay routine is calibrated for an Arduino Uno 16MHz
 *
 * @param num number of ms to delay
 */
void delay_ms(uint16_t num) {
    while (num--) {
        for (volatile long x = 0; x < 468; x++) {
            ;
        }
    }
}

/*
 * We set the Data direction register to specify which pins are inputs and
 * which are outputs. A 0 bit indicates the corresponding pin is an input,
 * a 1 indicates the corresponding pin is an output
 */
void setup() {
    DDRB |= 0b00100000;     // PORTB pin 5 (D13) output
    PORTB &= 0b11011111;    // turn off LED
}

int main(void) {
    setup();    // set up the physical hardware
    uint16_t delay = 1000;
    while (1) {
        delay_ms(delay);
        PORTB ^= 0b00100000;    // invert LED
    }
}
```

Main function

- First function to be executed

Basic Embedded Systems Program in C

MACQUARIE University
SYDNEY·AUSTRALIA

```c
#include <xc.h>

/**
 * The delay routine is calibrated for an Arduino Uno 16MHz
 *
 * @param num number of ms to delay
 */
void delay_ms(uint16_t num) {
    while (num--) {
        for (volatile long x = 0; x < 468; x++) {
            ;
        }
    }
}

/*
 * We set the Data direction register to specify which pins are inputs and
 * which are outputs. A 0 bit indicates the corresponding pin is an input,
 * a 1 indicates the corresponding pin is an output
 */
void setup() {
    DDRB |= 0b00100000;     // PORTB pin 5 (D13) output
    PORTB &= 0b11011111;    // turn off LED
}

int main(void) {
    setup();    // set up the physical hardware
    uint16_t delay = 1000;
    while (1) {
        delay_ms(delay);
        PORTB ^= 0b00100000;    // invert LED
    }
}
```

Sub functions

- Encapsulating code for re-use or setup
- Must be defined above main()

Basic Embedded Systems Program in C

MACQUARIE University
SYDNEY·AUSTRALIA

```c
#include <xc.h>

/**
 * The delay routine is calibrated for an Arduino Uno 16MHz
 *
 * @param num number of ms to delay
 */
void delay_ms(uint16_t num){
    while (num--) {
        for (volatile long x = 0; x < 468; x++) {
            ;
        }
    }
}

/*
 * We set the Data direction register to specify which pins are inputs and
 * which are outputs. A 0 bit indicates the corresponding pin is an input,
 * a 1 indicates the corresponding pin is an output
 */
void setup() {
    DDRB |= 0b00100000;     // PORTB pin 5 (D13) output
    PORTB &= 0b11011111;    // turn off LED
}

int main(void) {
    setup();    // set up the physical hardware
    uint16_t delay = 1000;
    while (1) {
        delay_ms(delay);
        PORTB ^= 0b00100000;    // invert LED
    }
}
```

Variables

- Scope
- Data type is important

Variable sizes

MACQUARIE University
SYDNEY·AUSTRALIA

Data type	Size
int8_t / uint8_t	8-bit
int16_t / uint16_t	16-bit
int32_t / uint32_t	32-bit
int64_t / uint64_t	64-bit

Basic Embedded Systems Program in C

MACQUARIE University
SYDNEY·AUSTRALIA

```c
#include <xc.h>

/**
 * The delay routine is calibrated for an Arduino Uno 16MHz
 *
 * @param num number of ms to delay
 */
void delay_ms(uint16_t num) {
    while (num--) {
        for (volatile long x = 0; x < 468; x++) {
            ;
        }
    }
}

/*
 * We set the Data direction register to specify which pins are inputs and
 * which are outputs. A 0 bit indicates the corresponding pin is an input,
 * a 1 indicates the corresponding pin is an output
 */
void setup() {
    DDRB |= 0b00100000;     // PORTB pin 5 (D13) output
    PORTB &= 0b11011111;    // turn off LED
}

int main(void) {
    setup();    // set up the physical hardware
    uint16_t delay = 1000;
    while (1) {
        delay_ms(delay);
        PORTB ^= 0b00100000;    // invert LED
    }
}
```

Main program loop

- Runs forever

Basic Embedded Systems Program in C

MACQUARIE University
SYDNEY·AUSTRALIA

```
#include <xc.h>

/**
 * The delay routine is calibrated for an Arduino Uno 16MHz
 *
 * @param num number of ms to delay
 */
void delay_ms(uint16_t num) {
    while (num--) {
        for (volatile long x = 0; x < 468; x++) {
            ;
        }
    }
}

/*
 * We set the Data direction register to specify which pins are inputs and
 * which are outputs. A 0 bit indicates the corresponding pin is an input,
 * a 1 indicates the corresponding pin is an output
 */
void setup() {
    DDRB |= 0b00100000;     // PORTB pin 5 (D13) output
    PORTB &= 0b11011111;    // turn off LED
}

int main(void) {
    setup();    // set up the physical hardware
    uint16_t delay = 1000;
    while (1) {
        delay_ms(delay);
        PORTB ^= 0b00100000;    // invert LED
    }
}
```

Pin configuration

- Each pin can be set up for input and output via corresponding Data Direction Register

Data Direction Register

- For setting direction of a pin as input or output

14.4.3 DDRB – The Port B Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x04 (0x24)	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

14.4.6 DDRC – The Port C Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x07 (0x27)	–	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	DDRC
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

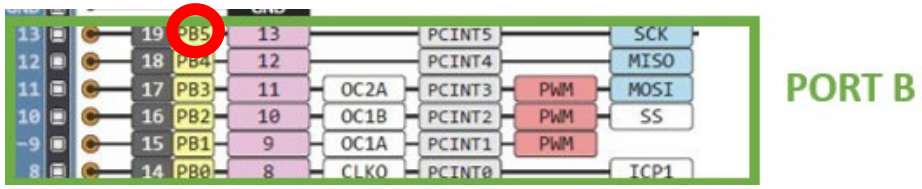
14.4.9 DDRD – The Port D Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x0A (0x2A)	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	DDRD
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- Input – write a 0 to the bit corresponding to the pin
- Output – write a 1

Example

- Set PB5 as an output pin
  - Write a 1 to bit 5 of DDRB
  - Leave all other bits in DDRB unchanged



PORT B

14.4.3 DDRB – The Port B Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x04 (0x24)	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

OR

0	0	1	0	0	0	0	0

Bitwise Operations

MACQUARIE University
SYDNEY·AUSTRALIA

Operator	Function
&	AND
|	OR
^	XOR
~	NOT
<< X	Left-shift by X bits
>> X	Right-shift by X bits

Operator	Function
&=	AND with existing values and assign
|=	OR with existing values and assign
^=	XOR with existing values and assign

Example

- Set PB5 as an output pin
  - Write a 1 to bit 5 of DDRB
  - Leave all other bits in DDRB unchanged

```
DDRB |= 0b00100000;
```

0b = following digits are binary

Numbers

MACQUARIE University
SYDNEY·AUSTRALIA

- 0b00001111 -> binary number
- 0x0F -> hexadecimal number
- 017 -> octal number
- 15 -> decimal number

Basic Embedded Systems Program in C

MACQUARIE University
SYDNEY·AUSTRALIA

```c
#include <xc.h>

/**
 * The delay routine is calibrated for an Arduino Uno 16MHz
 *
 * @param num number of ms to delay
 */
void delay_ms(uint16_t num) {
    while (num--) {
        for (volatile long x = 0; x < 468; x++) {
            ;
        }
    }
}

/*
 * We set the Data direction register to specify which pins are inputs and
 * which are outputs. A 0 bit indicates the corresponding pin is an input,
 * a 1 indicates the corresponding pin is an output
 */
void setup() {
    DDRB |= 0b00100000;     // PORTB pin 5 (D13) output
    PORTB &= 0b11011111;    // turn off LED
}

int main(void) {
    setup();    // set up the physical hardware
    uint16_t delay = 1000;
    while (1) {
        delay_ms(delay);
        PORTB ^= 0b00100000;    // invert LED
    }
}
```

Write Output

- Set initial value
- Update value

Code is an example of a state machine

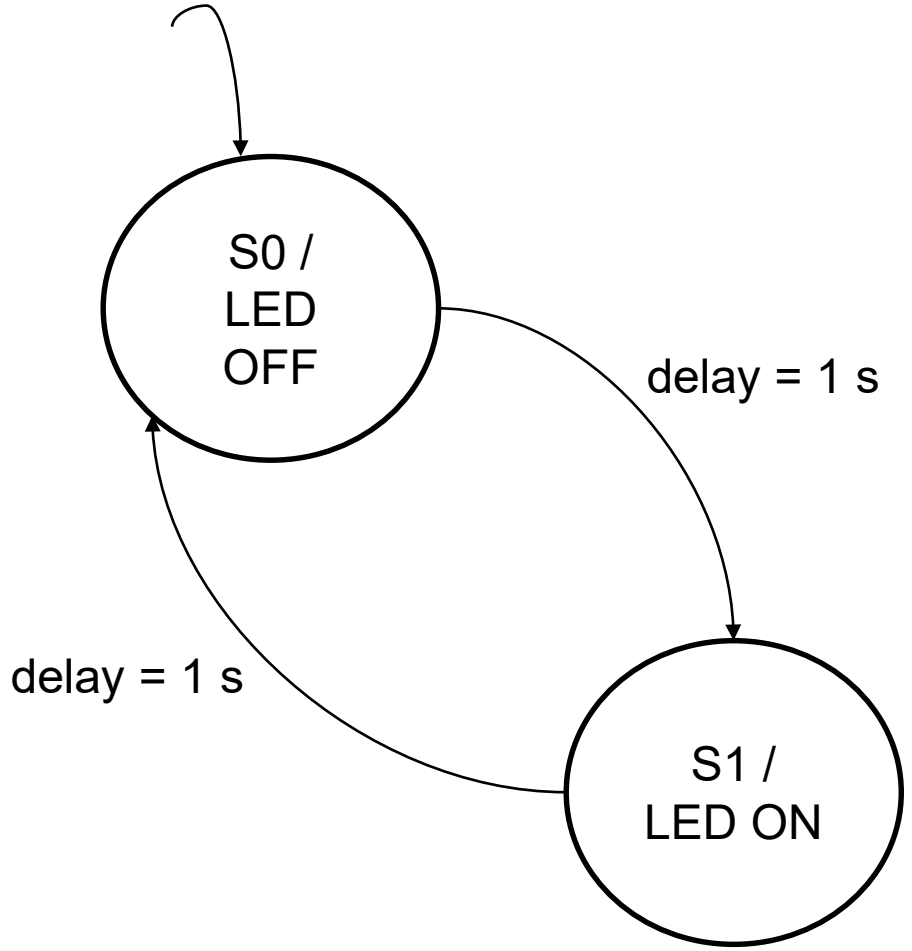
MACQUARIE University
SYDNEY·AUSTRALIA

```
#include <xc.h>

/**
 * The delay routine is calibrated for an Arduino Uno 16MHz
 *
 * @param num number of ms to delay
 */
void delay_ms(uint16_t num) {
    while (num--) {
        for (volatile long x = 0; x < 468; x++) {
            ;
        }
    }
}

/*
 * We set the Data direction register to specify which pins are inputs and
 * which are outputs. A 0 bit indicates the corresponding pin is an input,
 * a 1 indicates the corresponding pin is an output
 */
void setup() {
    DDRB |= 0b00100000;     // PORTB pin 5 (D13) output
    PORTB &= 0b11011111;    // turn off LED
}

int main(void) {
    setup();    // set up the physical hardware
    uint16_t delay = 1000;
    while (1) {
        delay_ms(delay);
        PORTB ^= 0b00100000;    // invert LED
    }
}
```



Data Register

- If output, used for setting pin value
- If input, used for setting pullup resistor

14.4.2 PORTB – The Port B Data Register

Bit	7	6	5	4	3	2	1	0	
0x05 (0x25)	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

14.4.5 PORTC – The Port C Data Register

Bit	7	6	5	4	3	2	1	0	
0x08 (0x28)	–	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	PORTC
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

14.4.8 PORTD – The Port D Data Register

Bit	7	6	5	4	3	2	1	0	
0x0B (0x2B)	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	PORTD
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Input pin pull up

- All port pins have individually selectable pull-up resistors
  - Write 1 to activate pull-up
- For an input, it is desirable to set the pull-up resistor HIGH
  - Implies input is active LOW, saves on external components
  - An external switch connected to input pin will pull to ground
  - 0 = Pressed, 1 = Not Pressed

```
DDRB &= 0b11111011;       // set PORT B, Pin 2 as input
PORTB |= 0b00000100;      // pull up PORT B, Pin 2
```

Input registers

- For reading input from pins

14.4.4 PINB – The Port B Input Pins Address[1]

Bit	7	6	5	4	3	2	1	0	
0x03 (0x23)	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

14.4.7 PINC – The Port C Input Pins Address[1]

Bit	7	6	5	4	3	2	1	0	
0x06 (0x26)	–	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	PINC
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

14.4.10 PIND – The Port D Input Pins Address[1]

Bit	7	6	5	4	3	2	1	0	
0x09 (0x29)	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	PIND
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Quick Summary – I/O Pins

MACQUARIE University
SYDNEY·AUSTRALIA

Register	Input	Output
Data Direction (DDRB, DDRC, DDRD)	Write 0	Write 1
Port (PORTB, PORTC, PORTD)	Write 1 for pull-up	Value for output
Input (PINB, PINC, PIND)	Read value of input	

Assignment	Usage
|=	Assign a 1 and leave other bits unchanged
&=	Assign a 0 and leave other bits unchanged

Multiple ways of setting bits

MACQUARIE University
SYDNEY·AUSTRALIA

```
DDRB |= 0b00100000;          // set pin 5 as output
PORTB &= 0b11011111;         // set output LOW

DDRB |= _BV(DDB5);           // set pin 5 as output
PORTB &= ~_BV(PORTB5);       // set output LOW

DDRB |= (1<<DDB5);           // set pin 5 as output
PORTB &= ~(1<<PORTB5);       // set output LOW
```

Multiple pins can be set at once

MACQUARIE University
SYDNEY·AUSTRALIA

```
DDRB |= 0b00100100;                    // set pin 2 & 5 as output
PORTB &= 0b11011011;                   // set both output LOW

DDRB |= _BV(DDB2) | _BV(DDB5);         // set pin 2 & 5 as output
PORTB &= ~_BV(PORTB2) & ~_BV(PORTB5);  // set both output LOW

DDRB |= (1<<DDB2) | (1<<DDB5);         // set pin 2 & 5 as output
PORTB &= ~(1<<PORTB2) & ~(1<<PORTB5);  // set both output LOW
```

Reading inputs

MACQUARIE University
SYDNEY · AUSTRALIA

```
// setup
DDRB  |= 0b00100000;                    // PORTB pin 5 (D13) output
PORTB &= 0b11011111;                    // turn off LED
DDRC  &= 0b11111011;                    // set PORT C, Pin 2 as input
PORTC |= 0b00000100;                    // pull up PORT C, Pin 2

while (1) {
    if ((PINC & 0b00000100) == 0) {     // button is pressed
        PORTB |= 0b00100000;
    } else {                            // button not pressed
        PORTB &= 0b11011111;
    }
}
```

Homework

- Before practical session
  1. Install MPLab and Arduino IDE
  2. Make sure you have an Arduino UNO
  3. Read Atmega328P Datasheet Sections: 1, 2, 7
- Before next lectorial
  - Read Atmega328P Datasheet Sections: 12, 13, 14