Timer and I/O

Chapter 4

Embedded System Architecture

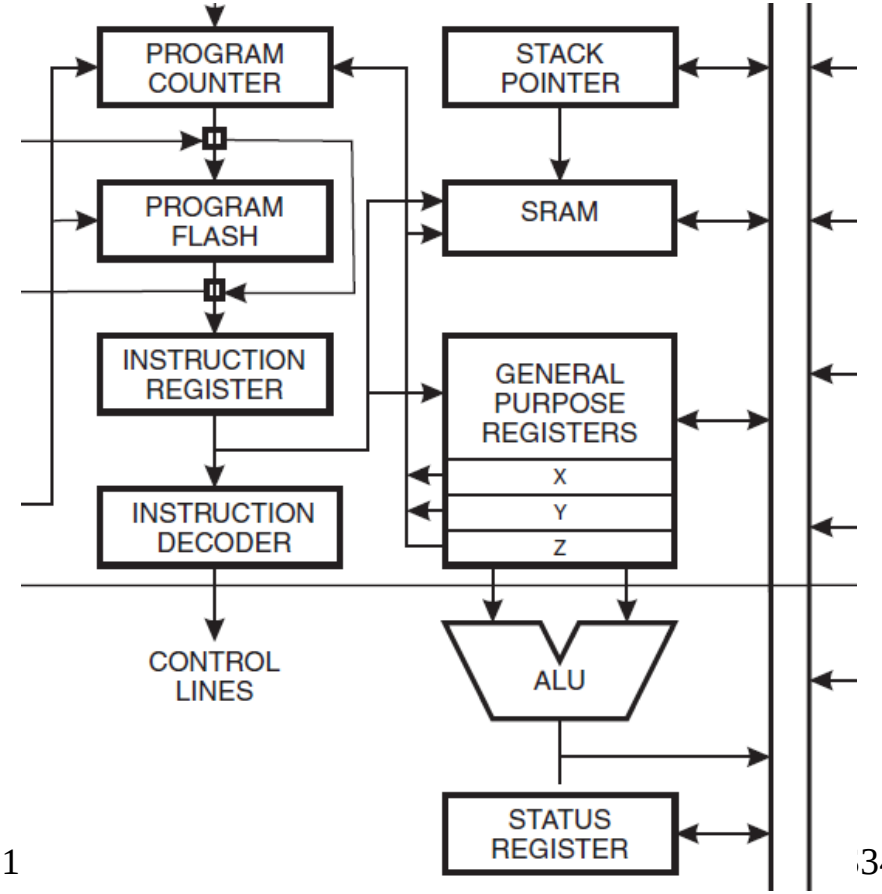
- von Neumann
  - A processing unit
  - A memory to hold both instructions and data
  - CPU can either read an instruction or access data from the memory, but not at the same time
- Harvard
  - A processing unit
  - Two memories to hold instructions and data separately
  - CPU can both read an instruction and access data at the same time

IO Architecture

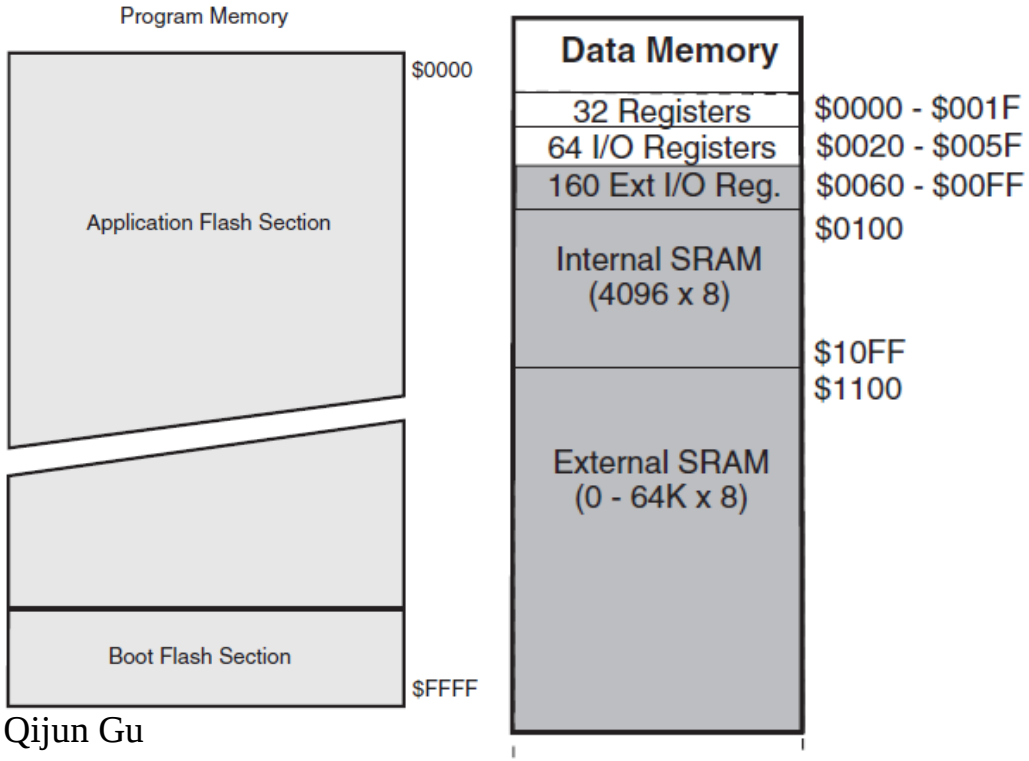
- Port IO
  - IO address is not memory address
  - IO data path is separated from memory data path
  - Special IO instructions
  - Simultaneous access to IO and memory
- Memory mapped IO
  - IO address is a part of memory address
  - IO and memory share the same data path
  - No special IO instructions
  - Exclusive access to IO or memory

Example: ATmega128

- CPU architecture
  - page 3, datasheet

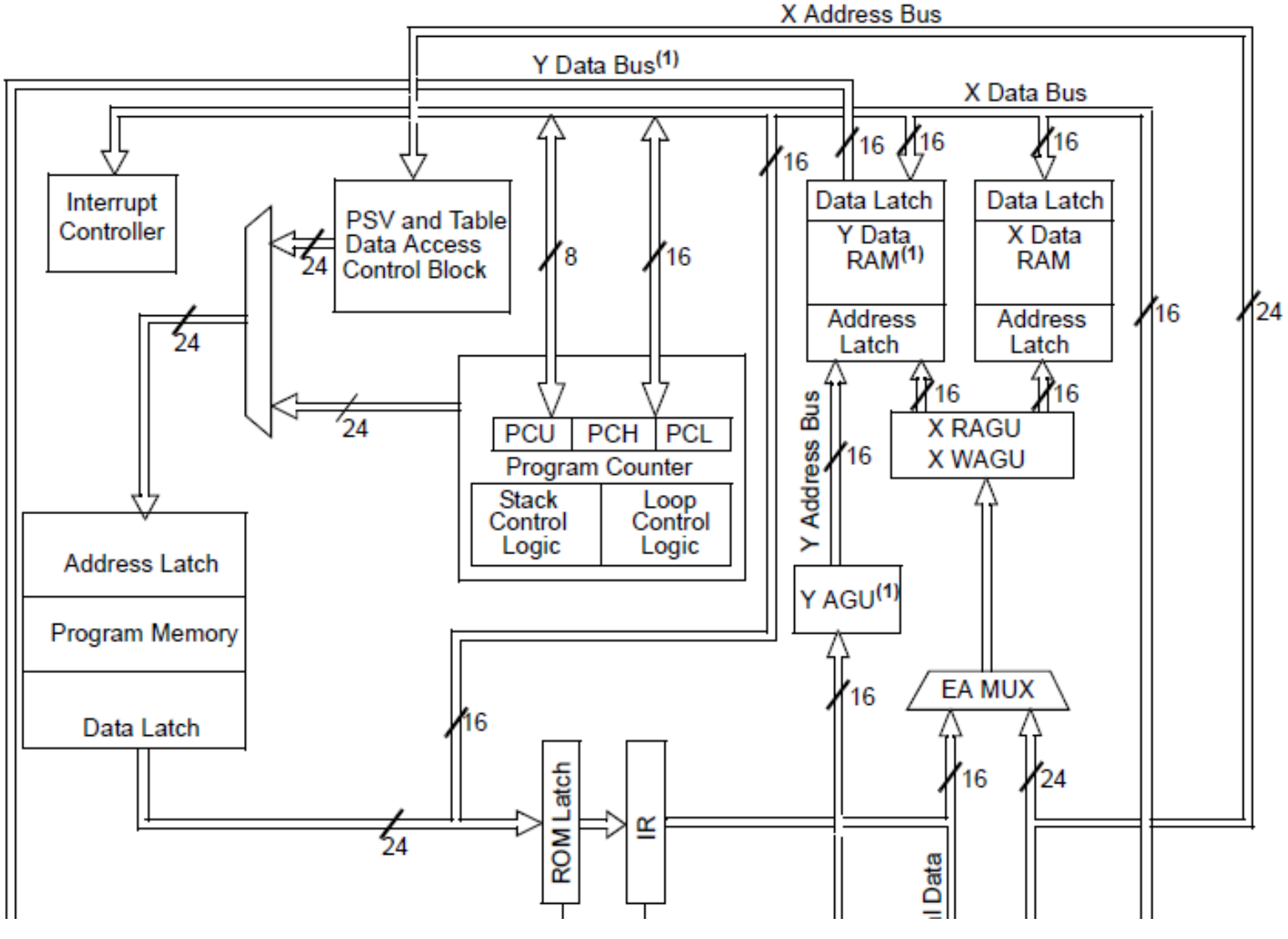


- IO architecture
  - Both port and memory IO spaces
  - page 18, 20, datasheet



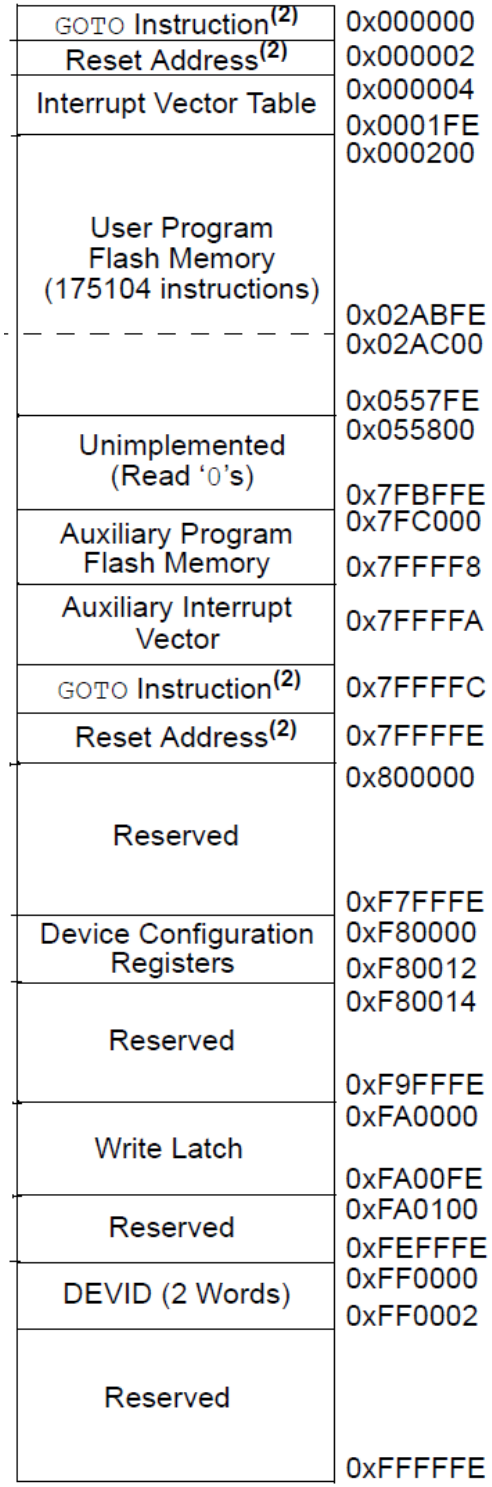
Example: Pic24ep

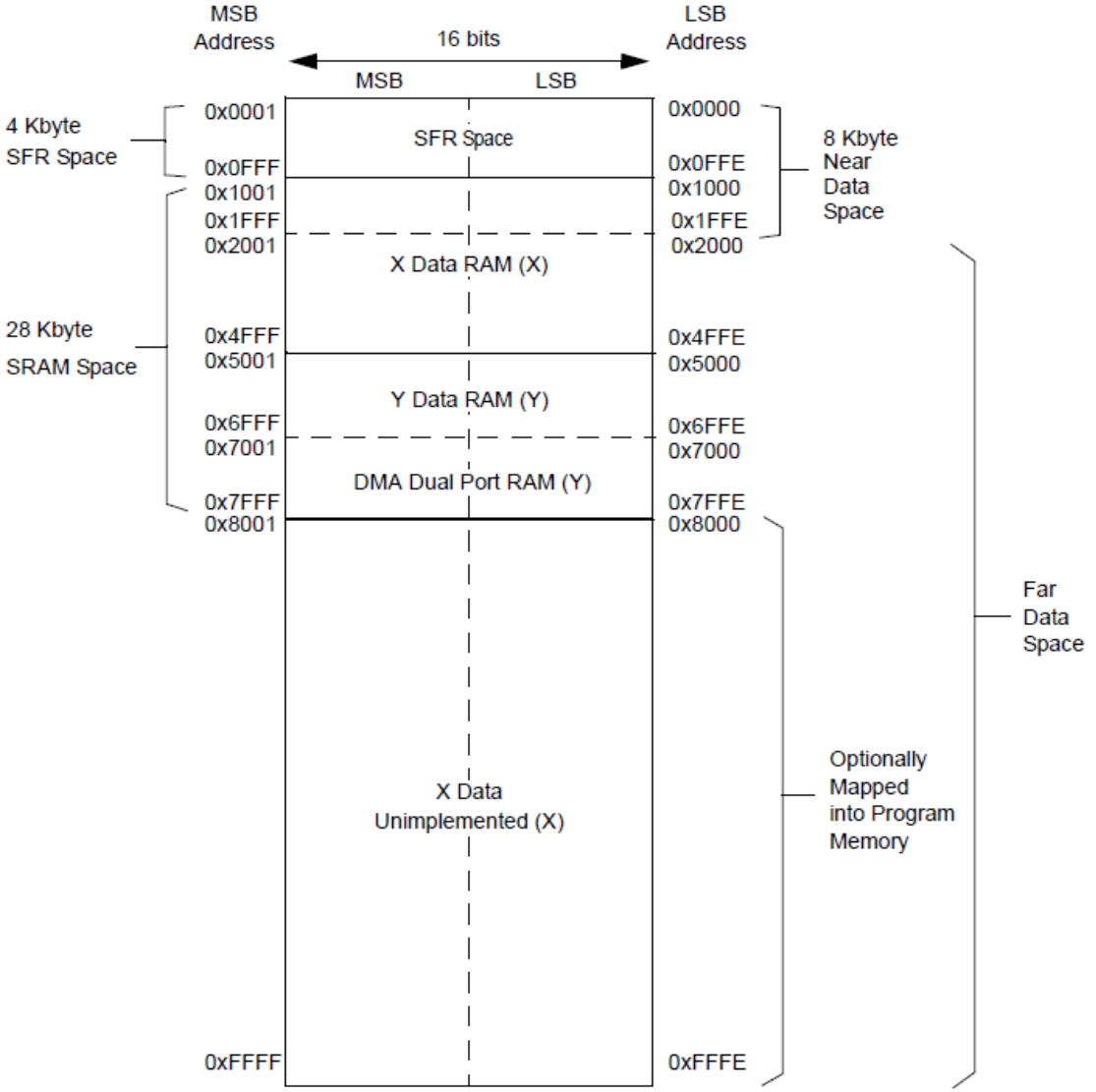
- Page 38, datasheet



nple: Pic24ep

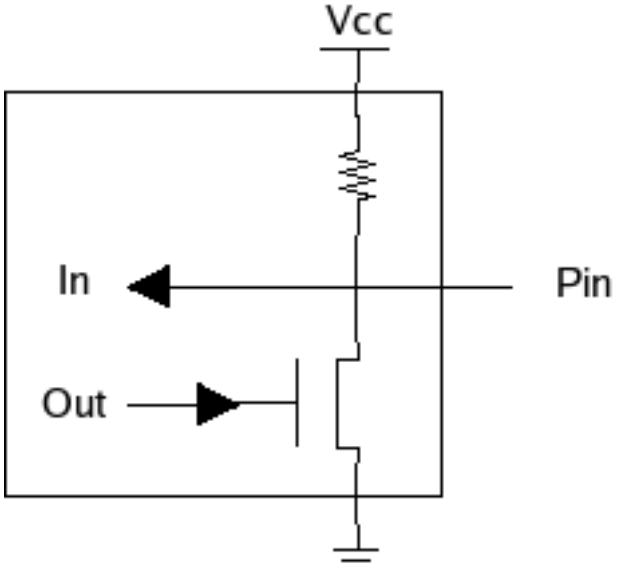
Page 47, 50, datasheet





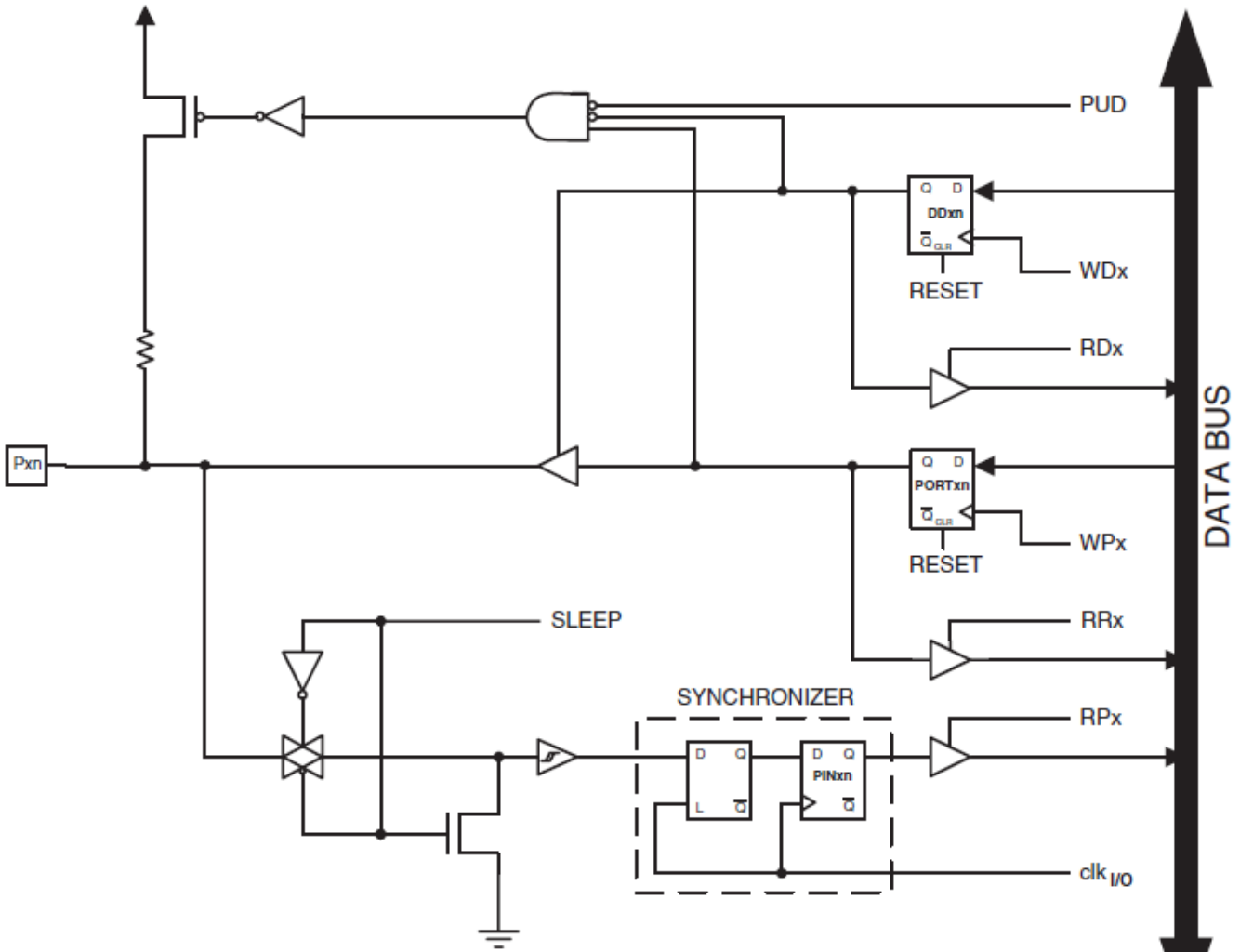
I/O Pins

- Bidirectional (in and out)
- Digital or analog
- Programmable
- Registers
  - Direction control
  - Data
    - Input
    - Output
    - Data change interrupt
  - D/A selection



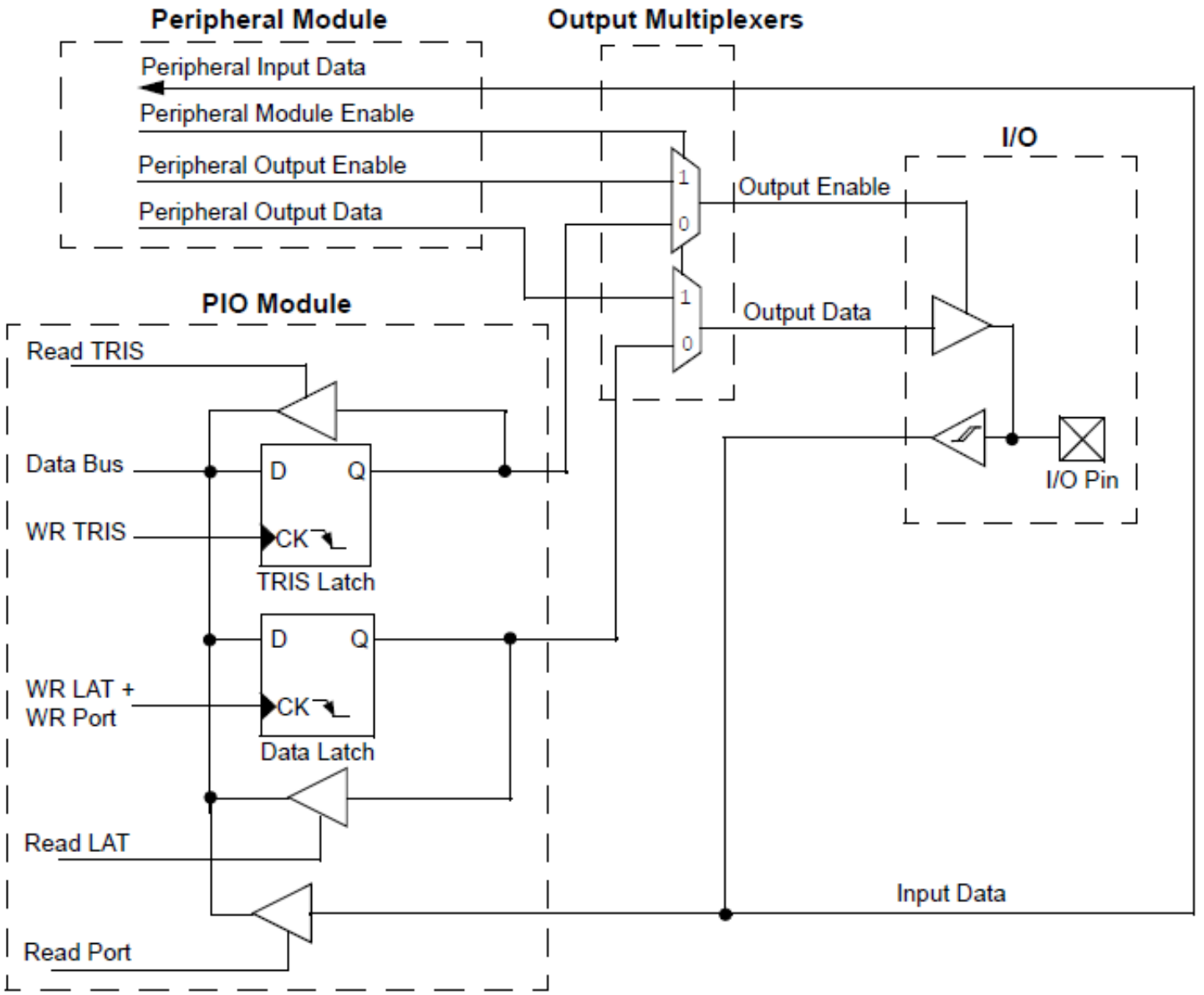
Example: Atmega128

Page 66, datasheet



Example: Pic24ep

Page 205, datasheet



Output (Atmega128)

- LEDs (case 12)
  - From schematic: Port D0..2
  - Three register bits of Port D
    - Table 25, page 67, datasheet
    - DDRD : direction control
      - 0 : input, 1 : output
    - PORTD : data output
      - 0 : off, 1 : on
    - PIND : data input

Table 25. Port Pin Configurations

DDxn	PORTxn	PUD (in SFIOR)	I/O	Pull-up	Comment
0	0	X	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	Pxn will source current if ext. pulled low.
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	X	Output	No	Output Low (Sink)
1	1	X	Output	No	Output High (Source)

Output (Atmega128)

- Debug
  - IO View
    - Debug/Windows/IO
  - Observe in debug
    - DDRA (direction)
    - PORTA (output)
    - PINA (input)
  - IO functions
    - Init
    - Toggle
    - Set
    - On/off

Output (Pic24ep)

- LEDs (case 12)
  - From schematic: Port D1..3
  - Addition doc on IO: S10-IO
  - Eight registers of Port D
    - ANSEL: analog/digianl section
      - By default 1: analog
      - Page of 206 of datasheet
    - TRISD : direction control
      - 1 : input, 0 : output
      - Page 3 of S10-IO
    - LATD : data output
    - PORTD: data input

Output (Pic24ep)

- Debug
  - IO View
    - Window/PIC Memory Views/SFRs
    - Memory:SFR, Format:Peripherals
  - Observe in debug
    - ANSELD (analog/debug)
    - TRISD (direction)
    - PORTD (input)
    - LATD (output)
  - IO functions
    - Init
    - Toggle
    - Set
    - On/off

Timer (Case 13)

- Delay
  - <util/delay.h>: https://www.nongnu.org/avr-libc/user-manual/group__util__delay.html
  - Define cpu frequency
    - Example: #define F_CPU 4000000UL // 4MHz
  - _delay_ms()
    - Atmega128's max delay is 262.14ms/f_cpu in MHz.
    - Example: f_cpu=4MHz, max delay is 262.14/4=65.535ms
  - _delay_us()
    - Max delay is 768us/f_cpu in MHz.
- Compile: only work with optimization -Os or -O3
- Debug: Processor/Stopwatch

Timer (Case13)

- Interrupt
  - <avr/interrupt.h>: https://www.nongnu.org/avr-libc/user-manual/group__avr__interrupts.html
  - Interrupt service routine
    - ISR(TIMER1_COMPA_vect)
  - Configure timer interrupt
  - Timer interrupt
    - Set timerFired
  - Timer task
    - Clear timerFired
    - Usually, do not execute big tasks in the interrupt.

Timer (Atmega128)

- Interrupt
  - Three timer control registers
    - Compare mode
    - Clock select and prescaler
  - Timer counter register
    - Store the counter incremented per prescaled clock tick
  - Output compare register
    - Store the value to be compared
  - Interrupt mask register
    - Enable the interrupt
  - 16-bit counter
    - Max delay: 2^16*1024/f

Timer

- Formula
  - Max delay
    - Fcpu=4MHz
    - Scaler=256
    - Count=65536
    - Max delay is 4.19s
  - Specifiy count
    - Fcpu=4MHz
    - Scaler=256
    - Delay=0.5s
    - Count is 7812

$$Delay_{Max} = \frac{Count_{Max} \times Scaler}{F_{CPU}}$$

$$Count = \frac{Delay \times F_{CPU}}{Scaler}$$

Timer (Pic24ep)

- Interrupt
  - #define __PIC24E__ to enable timer functions
  - Single or combination mode
  - Timer control register
    - Compare mode
    - Clock select and prescaler
  - Timer counter register
    - Store the counter incremented per prescaled clock tick
  - Interrupt mask register
    - Enable the interrupt
  - 16 or 32-bit counter
    - Max delay: 2^32*256/f

Input

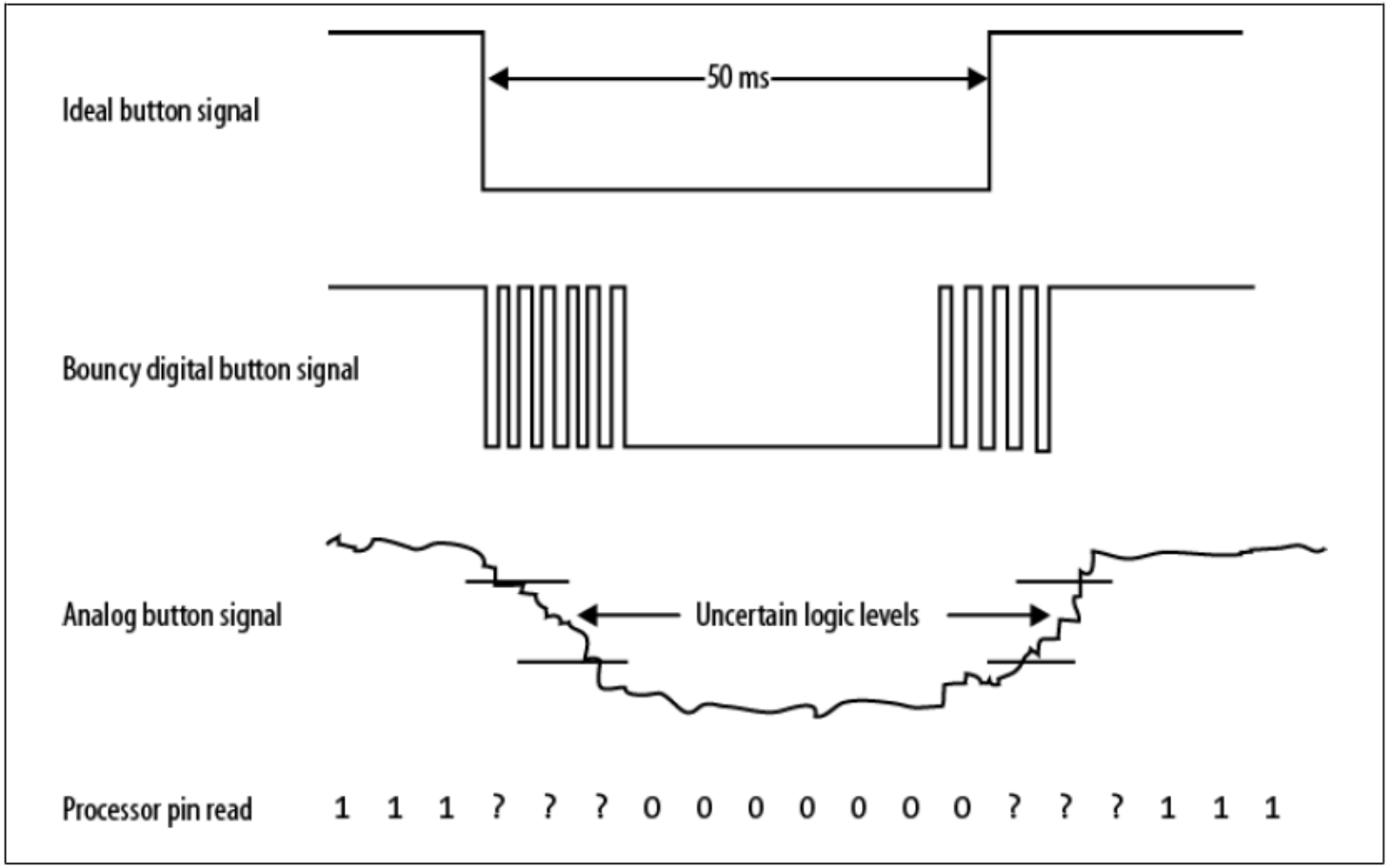
- Polling
  - When an input does not trigger an interrupt
  - Configure IO pin as input
  - Case 15, atmega128
- Interrupt
  - When an input change triggers an interrupt
  - Configure IO pin as input
  - Enable interrupt
  - Case 15, pic24ep
- Two states are needed for bouncing button.
- Stimulus is needed for simulating button.

Input Simulation

- AVR Tool
  - Debug/Simulator/Stimuli
  - Break, then Debug/Execute Stimulifile
  - Stimuli file format
    - #delay_cycles
    - assignment (e.g. PINA |= 0x80)
  - 4MHz CPU clock
- MPLab X
  - Window/Simulator/Stimulus
  - Pin/Register Actions
  - Time/Pin/Value

```
#0
PINA = 00
#1000
PINA |= 0x80
#10000
PINA = 00
#20000
PINA |= 0x80
#100000
PINA = 00
#150000
PINA |= 0x80
#10000
PINA = 00
```

Bouncing Button



Applications

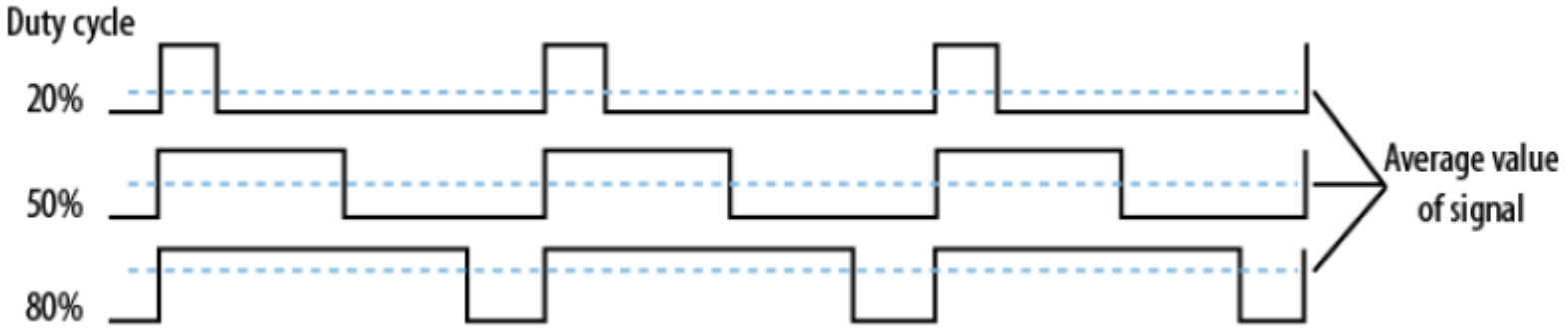
- Basic application needs
  - Clock (**timer**)
  - Input (**digital**/analog)
  - Output (**digital**/analog)
- Applications
  - Blinking/flashing (output)
  - Pulse Width Modulation (PWM) (output)
    - LED dimmer
    - Motor control
      - RPM control (rotation per minute)
      - Direction control
- Debug: logic analyzer
  - MPLab: Window/Simulator/Analyzer

Blinking Leds

- Case 14
- Modularize program
- Led module
  - Led.c
  - Led.h
- Timer module
  - Timer.c
  - Timer.h
- Interrupt or delay based

Pulse Width Modulation (PWM)

- LED brightness control
- Motor speed and direction control
- Two timing signals
  - One interrupt-based timer for period: 20ms
  - Delay for duty cycle: 0% to 100% of period
  - Approaches
    - Two timers (not very accurate due to timer overhead)
    - PWM module (better than two timers)



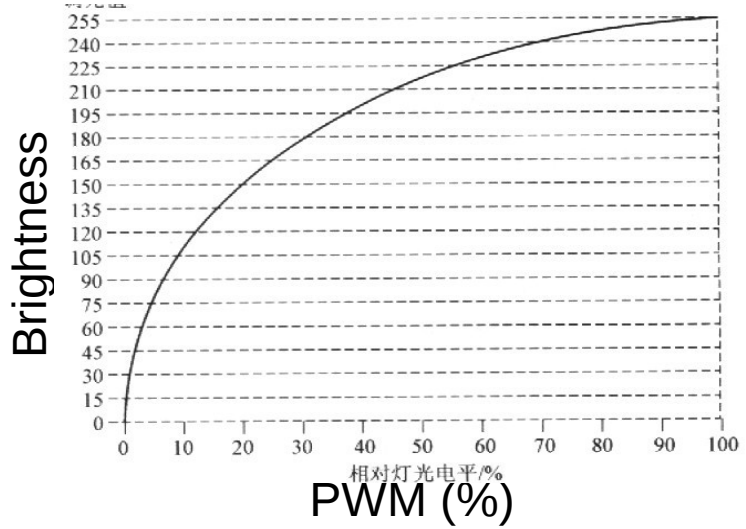
PWM Applications (Case31)

- LED dimmer (video)
  - Provide levels of brightness
    - Short duty of brightness
    - Longer duty means brighter
- Motor speed controller (video)
  - Reduce speed from 5000rpm to 5rpm
    - Short duty of rotation
    - Longer duty means faster
- Servo (video, pdf)
  - Control directions (-90 to 90)
    - Ratio of duty to period (5%-10%)
    - Longer duty means clockwise

LED Brightness

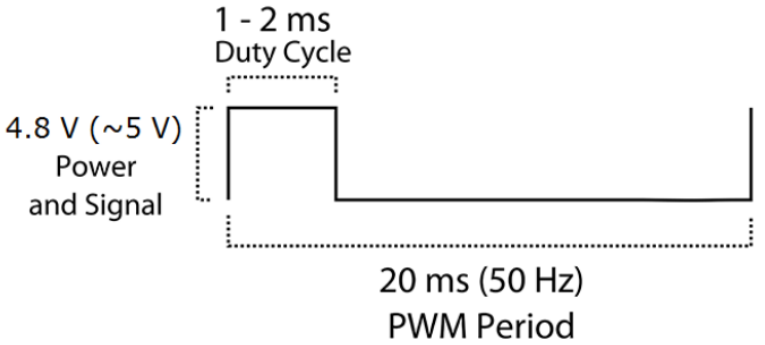
- Logarithmic dimmer curve
  - Non-linear brightness perception
  - PWM table of discrete brightness levels
- PWM parameters
  - 20 ms periodic duration
  - 200 us pwm step size
  - 100 pwm steps

PWM	Bright
100	10
71	9
58	8
42	7
30	6
21	5
14	4
9	3
4	2
1	1
0	0



Motor Control

- Very low RPM (rotation per minute) motor
  - https://www.youtube.com/watch?v=H36oAtkAb7w
  - A normal motor has >1000rpm
  - PWM: motor rotates only in duty cycles
- Servo motor
  - https://www.youtube.com/watch?v=KDfhXd2kelI
  - 180 degree, bidirectional, direction control
  - Direction = -90+(duty-1)*180
  - If accuracy is 1 degree,
  - Then duty step size is 1/180ms or 5.556us



Position "0" (1.5 ms pulse) is middle, "90" (~2 ms pulse) is all the way to the right, "-90" (~1 ms pulse) is all the way to the left.