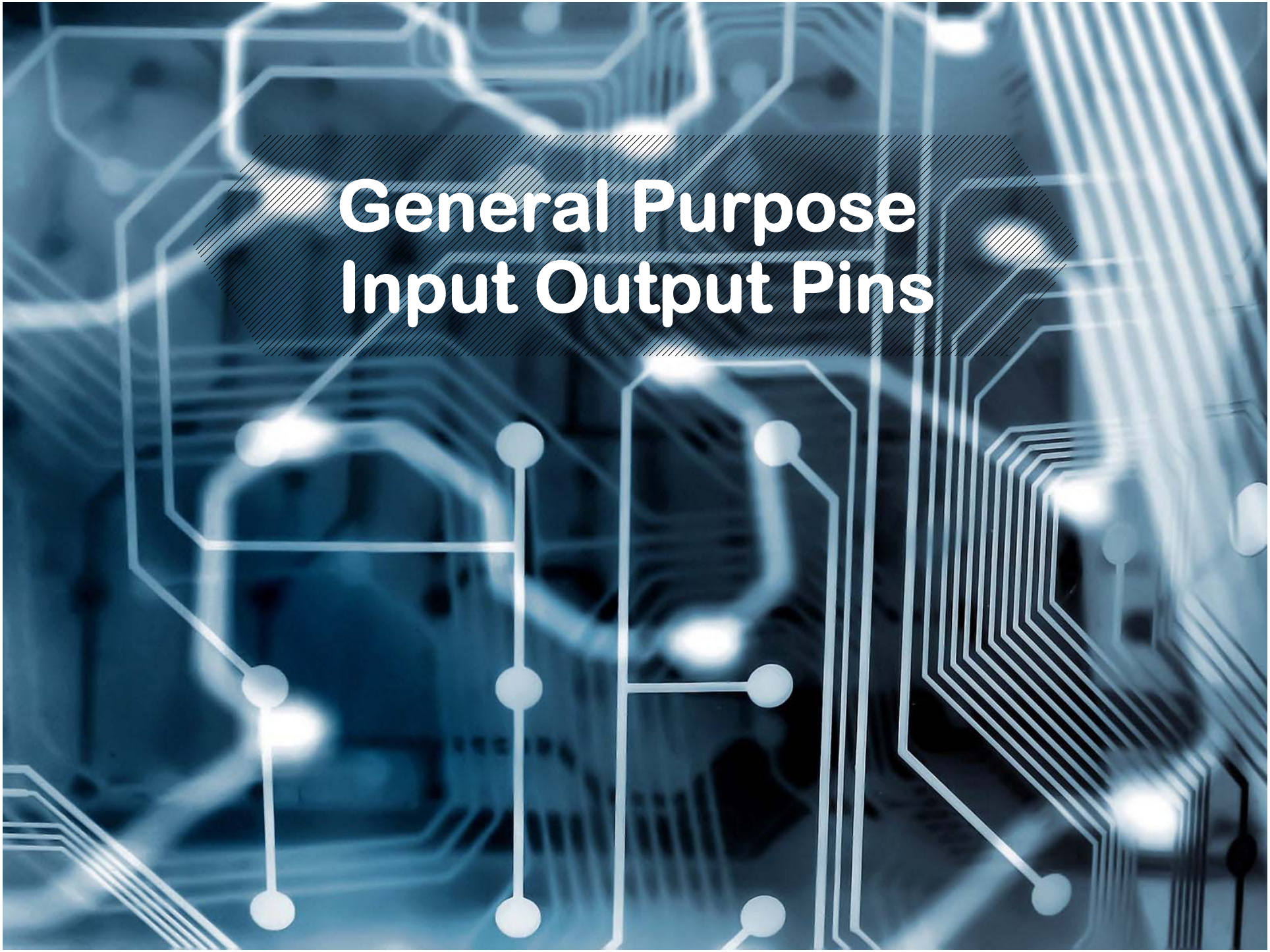
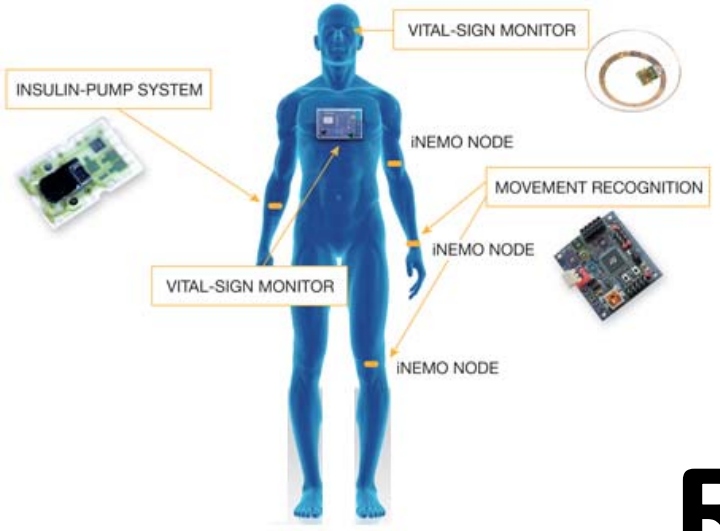

General Purpose
Input Output Pins



Real Time Embedded Systems

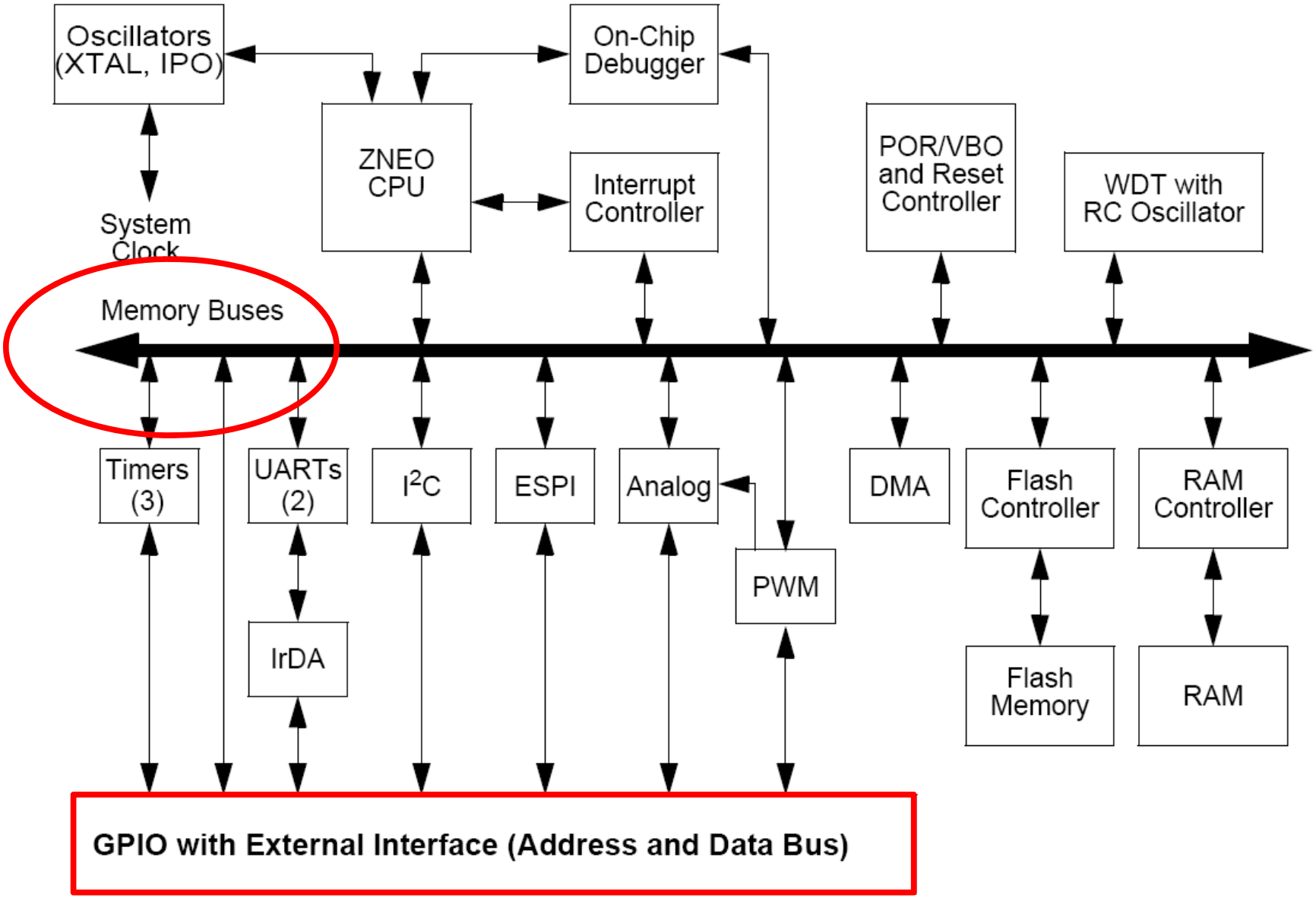
www.atomicrhubarb.com/embedded

Lecture 9 – January 31, 2012

General Purpose IO

Section Topic

- Where in the books
  - Catsoulis chapter 1, pg 28
  - Simon chapter 4
  - Zilog UM197 (ZNEO Z16F Series Flash Microcontroller Contest Kit User Manual)
  - Zilog UM171 (ZiLOG Developer Studio II—ZNEO User Manual)
  - Zilog PS220 (ZNEO Z16F Series Product Specification)
  - Zilog UM188 (ZNEO CPU Core User Manual)
  - Assorted datasheets


Oscillators (XTAL, IPO)
On-Chip Debugger
ZNEO CPU
Interrupt Controller
POR/VBO and Reset Controller
WDT with RC Oscillator
System Clock
Memory Buses
Timers (3)
UARTs (2)
I2C
ESPI
Analog
DMA
Flash Controller
RAM Controller
PWM
IrDA
Flash Memory
RAM
GPIO with External Interface (Address and Data Bus)

Address Space

Special Function Registers

Internal RAM

Internal FLASH memory

128KB

64KB

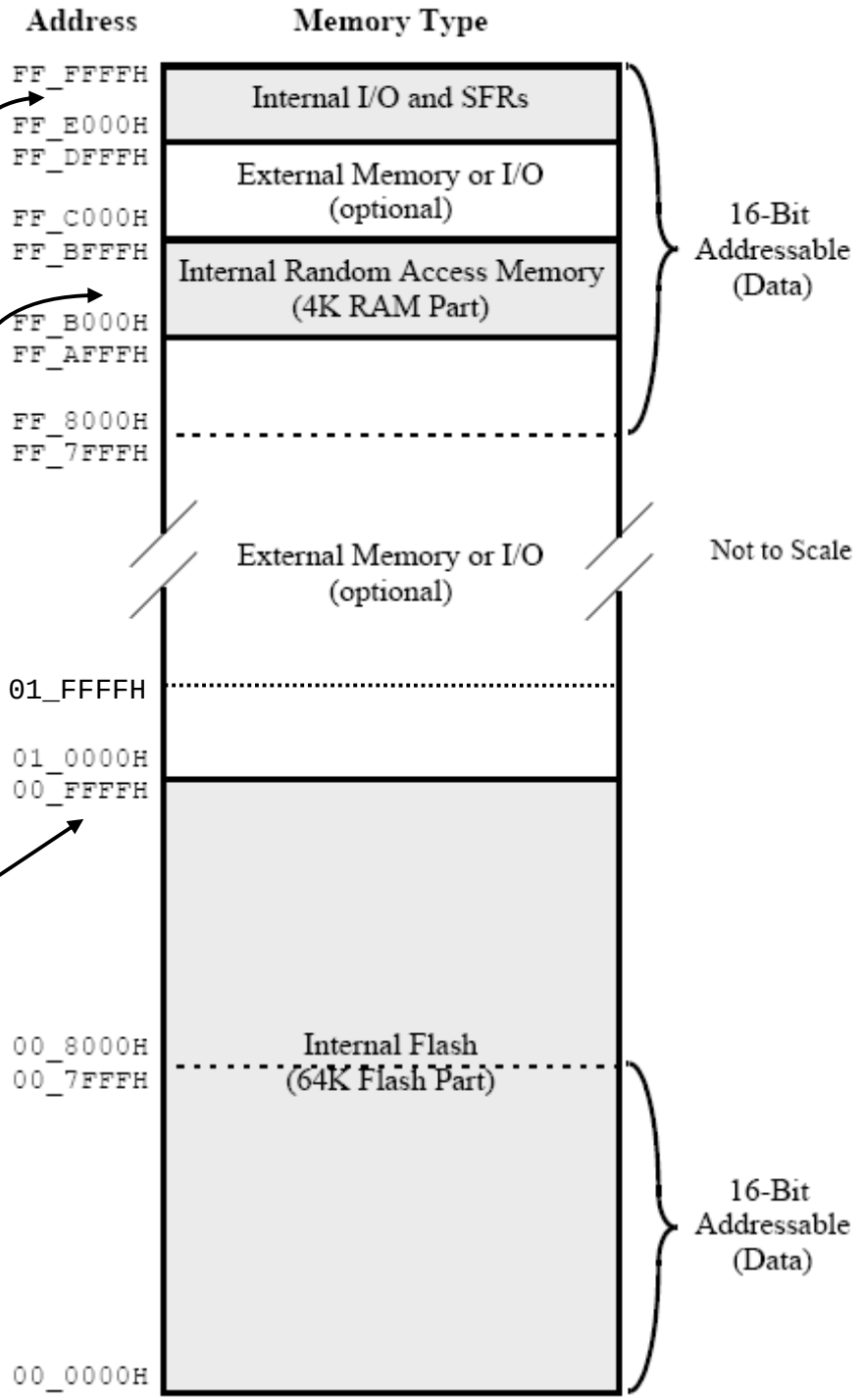


Table 6. Register File Address Map

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page No
ZNEO CPU Base Address = FF_E000				
FF_E004-FF_E007	Program Counter Overflow	PCOV	00FFFFFF	Refer to the ZNEO CPU User Manual
FF_E00C-FF_E00F	Stack Pointer Overflow	SPOV	00000000	
FF_E010	Flags	FLAGS	XX	
FF_E012	CPU Control	CPUCTL	00	
ZNEO Trace Address = FF_E014				
FF_E013	Trace Control	TRACECTL	00	321
FF_E014-FF_E017	Trace Address	TRACEADDR	XXXXXXXX	322
Interrupt Controller Base Address = FF_E020				
FF_E020	System Exception Status High	SYSEXCPH	0000	84
FF_E021	System Exception Status Low	SYSEXCPL	0000	84
FF_E022 - FF_E02F	Reserved	—	XX	
FF_E030	Interrupt Request 0	IRQ0	00	85
FF_E031	Interrupt Request 0 Set	IRQ0SET	xx	85
FF_E032	IRQ0 Enable High Bit	IRQ0ENH	00	89
FF_E033	IRQ0 Enable Low Bit	IRQ0ENL	00	89
FF_E034	Interrupt Request 1	IRQ1	00	87
FF_E035	Interrupt Request 1Set	IRQ1SET	XX	87
FF_E036	IRQ1 Enable High Bit	IRQ1ENH	00	91
FF_E037	IRQ1 Enable Low Bit	IRQ1ENL	00	91
FF_E038	Interrupt Request 2	IRQ2	00	88
FF_E039	Interrupt Request 2 Set	IRQ2SET	xx	88
FF_E03A	IRQ2 Enable High Bit	IRQ2ENH	00	92
FF_E03B	IRQ2 Enable Low Bit	IRQ2ENL	00	92
FF_E03C-FF_E03F	Reserved	—	XX	

GPIO Port A

GPIO Base Address = FF_E100				
GPIO Port A Base Address = FF_E100				
FF_E100	Port A Input Data	PAIN	xx	72
FF_E101	Port A Output Data	PAOUT	00	72
FF_E102	Port A Data Direction	PADD	00	73
FF_E103	Port A High Drive Enable	PAHDE	00	74
FF_E104	Port A Alternate Function High	PAAFH	00	74
FF_E105	Port A Alternate Function Low	PAAFL	00	75
FF_E106	Port A Output Control	PAOC	00	75
FF_E107	Port A Pull-Up Enable	PAPUE	00	76
FF_E108	Port A Stop Mode Recovery Enable	PASMRE	00	76
FF_E109 - FF_E10B	Port A Reserved			
FF_E10C	Port A Irq Mux1	PAIMUX1	00	77
FF_E10D	Port A Reserved			
FF_E10E	Port A Irq Mux	PAIMUX	00	77
FF_E10F	Port A Irq Edge	PAIEDGE	00	78

Peripheral Address Map

- Many pages of this in the manual

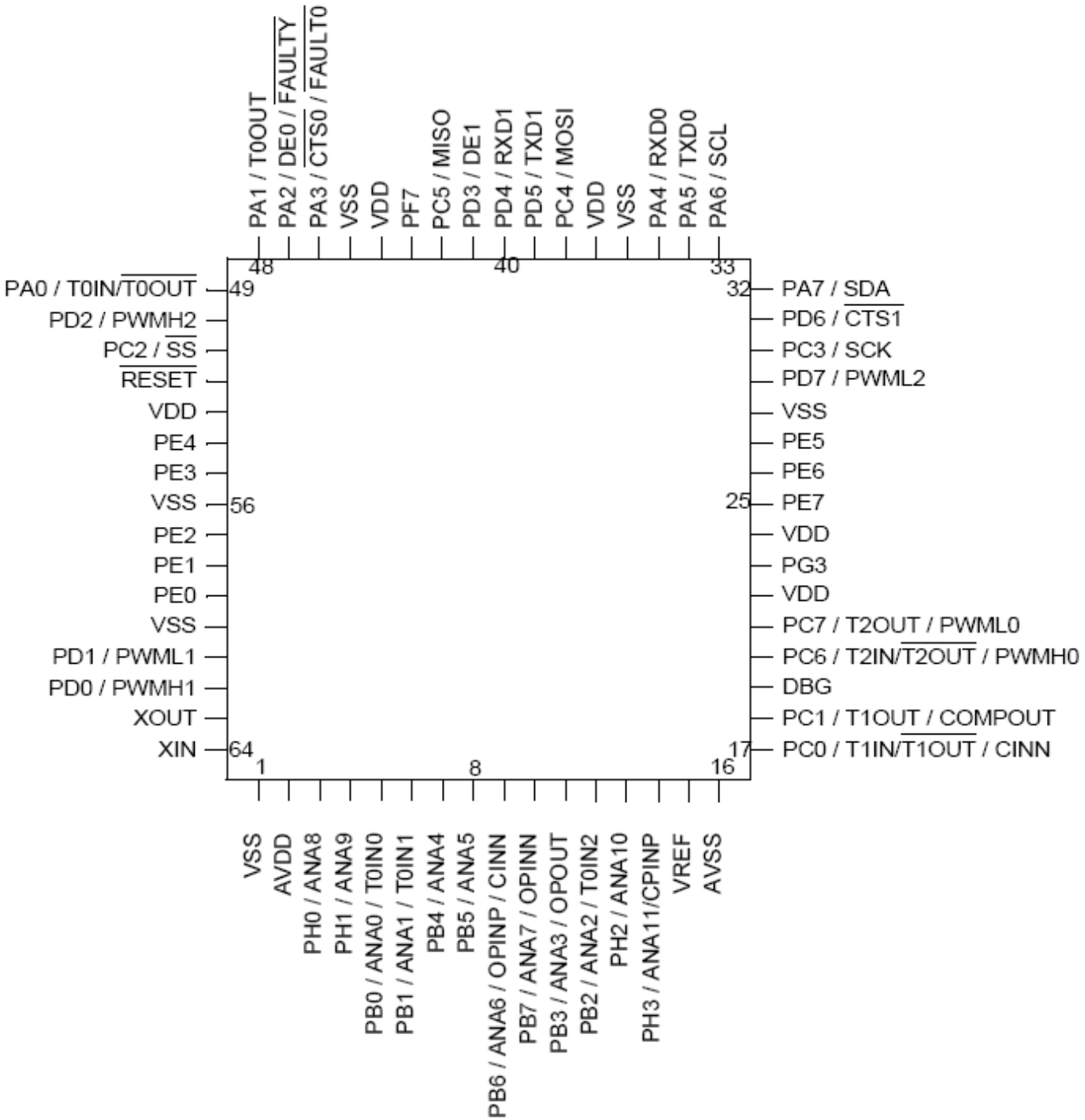
Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page No
GPIO Port F Base Address = FF_E150				
FF_E150	Port F Input Data	PFIN	XX	72
FF_E151	Port F Output Data	PFOUT	00	72
FF_E152	Port F Data Direction	PFDD	00	73
FF_E153	Port F High Drive Enable	PFHDE	00	74
FF_E154	Reserved			
FF_E155	Port F Alternate Function Low	PFAFL	00	75
FF_E156	Port F Output Control	PFOC	00	75
FF_E157	Port F Pull-Up Enable	PFPUE	00	76
FF_E158	Port F Stop Mode Recovery Enable	PFSMRE	00	76
FF_E159 - FF_E15F	Port F Reserved			

GPIO

- General Purpose Input Output ports
- Maximum of 10 ports
  - 8-bit ports (A – K)
  - Less available on physically smaller parts (not enough pins)
  - Some not accessible or partially accessible

Table 23. GPIO Port Availability by Device

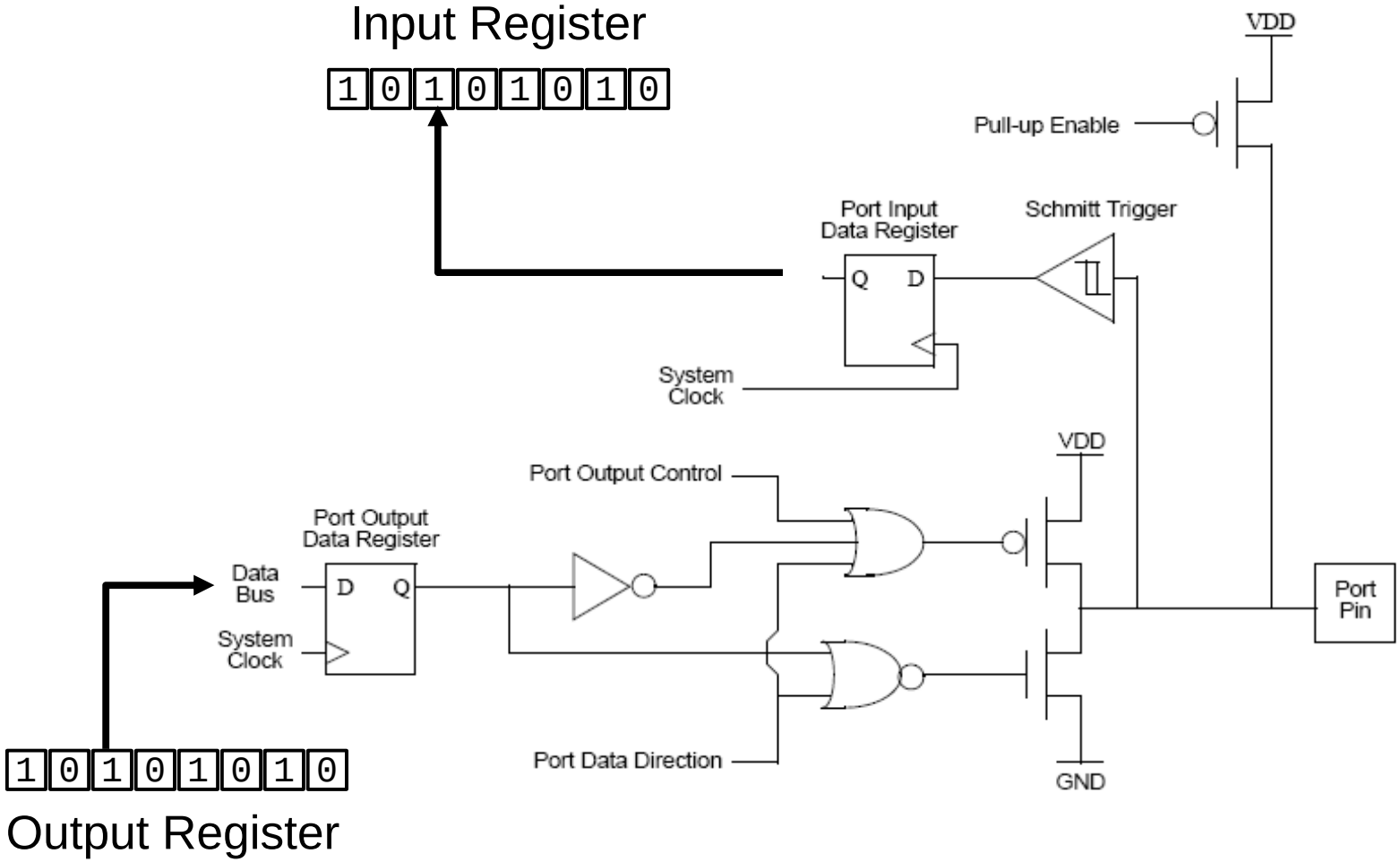
Device	Pin-Count	Port A	Port B	Port C	Port D	Port E	Port F	Port G	Port H	Port J	Port K
Z16F2811	100-pin	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[3:0]	[7:0]	[7:0]
	80-pin	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[3:0]	-	-
Z16F6411	100-pin	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[3:0]	[7:0]	[7:0]
	80-pin	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[3:0]	-	-
Z16F3211	100-pin	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[3:0]	[7:0]	[7:0]
	80-pin	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[3:0]	-	
Z16F2810	80-pin	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[3:0]	-	-
	68-pin	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[7]	[3]	[3:0]	-	-
	64-pin	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[7]	[3]	[3:0]	-	-


PA1 / T0OUT
PA2 / DE0 / FAULTY
PA3 / CTS0 / FAULT0
VSS
VDD
PF7
PC5 / MISO
PD3 / DE1
PD4 / RXD1
PD5 / TXD1
PC4 / MOSI
VDD
VSS
PA4 / RXD0
PA5 / TXD0
PA6 / SCL
48
40
33
PA0 / T0IN/T0OUT
49
PD2 / PWMH2
PC2 / SS
RESET
VDD
PE4
PE3
VSS
56
PE2
PE1
PE0
VSS
PD1 / PWML1
PD0 / PWMH1
XOUT
XIN
64
1
8
16
32
PA7 / SDA
PD6 / CTS1
PC3 / SCK
PD7 / PWML2
VSS
PE5
PE6
25
PE7
VDD
PG3
VDD
PC7 / T2OUT / PWML0
PC6 / T2IN/T2OUT / PWMH0
DBG
PC1 / T1OUT / COMPOUT
17
PC0 / T1IN/T1OUT / CINN
VSS
AVDD
PH0 / ANA8
PH1 / ANA9
PB0 / ANA0 / T0IN0
PB1 / ANA1 / T0IN1
PB4 / ANA4
PB5 / ANA5
PB6 / ANA6 / OPINP / CINN
PB7 / ANA7 / OPINN
PB3 / ANA3 / OPOUT
PB2 / ANA2 / T0IN2
PH2 / ANA10
PH3 / ANA11/CPINP
VREF
AVSS

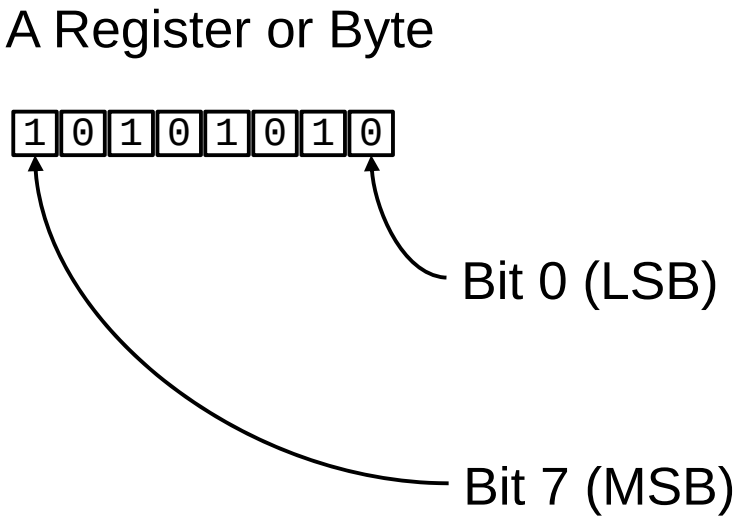
Alternate Function

Port	Pin	Alternate Function 1	Alternate Function 2	Alternate Function 3	External Interface
Port A	PA0	T0IN / $\overline{\text{T0OUT}}$	DMA0REQ	T0INPB	
	PA1	T0OUT	DMA0ACK		
	PA2	DE0	FAULTY		
	PA3	$\overline{\text{CTS0}}$	FAULT0		
	PA4	RXD0	CS1		
	PA5	TXD0	CS2		
	PA6	SCL	CS3		
	PA7	SDA	CS4		

GPIO


Input Register
1 0 1 0 1 0 1 0
VDD
Pull-up Enable
Port Input
Data Register
Schmitt Trigger
Q D
System
Clock
VDD
Port Output Control
Port Output
Data Register
Data
Bus
D Q
System
Clock
Port
Pin
Port Data Direction
GND
1 0 1 0 1 0 1 0
Output Register

Bits



- SET a bit (turn it on or “1”)
- CLEAR a bit (turn it off or “0”)
- Some folks say they RESET a bit to turn it off.

Port Configuration

- PxDD = Data Direction - Is the port we are interested in an input port or output port.
- PxAFH,PxAFL - Alternate Function – GPIO or something else
- PxOC- Output Control – Standard logic or open drain
- PxHDE - High Drive – Increased drive current on outputs

Port Configuration

- PxSMRE - Stop Mode Recovery – Wake up processor on signal change
- PxIN – Data Input
- PxOUT – Data Output
- PxPUE – Pull Up enable

PxIN

Table 25. Port A-K Input Data Registers (PxIN)

BITS	7	6	5	4	3	2	1	0
FIELD	PIN7	PIN6	PIN5	PIN4	PIN3	PIN2	PIN1	PIN0
RESET	X	X	X	X	X	X	X	X
R/W	R	R	R	R	R	R	R	R
ADDR	FF_E100, FF_E110, FF_E120, FF_E130, FF_E140, FF_E150, FF_E160, FF_E170, FF_E180, FF_E190							

PIN[7:0]—Port Input Data

Sampled data from the corresponding port pin input.

0 = Input data is logical 0 (Low).

1 = Input data is logical 1 (High).

PxDD

Table 27. Port A-K Data Direction Registers (P*x*DD)

BITS	7	6	5	4	3	2	1	0
FIELD	DD7	DD6	DD5	DD4	DD3	DD2	DD1	DD0
RESET	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	FF_E102, FF_E112, FF_E122, FF_E132, FF_E142, FF_E152, FF_E162, FF_E172, FF_E182, FF_E192							

DD[7:0]—Data Direction

These bits control the direction of the associated port pin. Port alternate function operation overrides the data direction register setting.

0 = Output

Data in the Port A-K output data register is driven onto the port pin.

1 = Input

The port pin is sampled and the value written into the Port A-K input data register. The output driver is high impedance.

Dan's Helpful Hint

```
1 looks like I(input)
0 looks like O(output)
```

PxAFH, PxAFL

Table 29. Port A-K Alternate Function High Registers (PxAFH)

BITS	7	6	5	4	3	2	1	0
FIELD	AFH[7]	AFH[6]	AFH[5]	AFH[4]	AFH[3]	AFH[2]	AFH[1]	AFH[0]
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	FF_E104, FF_E124, FF_E134, FF_E174							

BITS	7	6	5	4	3	2	1	0
FIELD	AFL[7]	AFL[6]	AFL[5]	AFL[4]	AFL[3]	AFL[2]	AFL[1]	AFL[0]
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	FF_E105, FF_E115, FF_E125, FF_E135, FF_E155, FF_E165, FF_E175, FF_E195							

PxAFH,PxAFL

Port A-K Alternate Function High and Low Registers

The Port A-K alternate function high and low registers (see Table 29 and Table 30 on page 74) select the alternate functions for the selected pins. To determine the alternate function associated with each port pin, see GPIO Alternate Functions on page 68. When changing alternate functions, it is recommended to use word data mode instructions to perform simultaneous Writes to the port alternate function high and low registers.

Caution: Do not enable alternate function for GPIO port pins which do not have an associated alternate function. Failure to follow this guideline will result in undefined operation.

AFH[*x*]	AFL[*x*]	Priority
0	0	No Alternate Function enabled
0	1	Alternate Function 1 Enabled
1	0	Alternate Function 2 Enabled
1	1	Alternate Function 3 Enabled

Note: *x* indicates the register bits from 0 through 7.

IN, OUT

Read from IN

```
int i = PDIN;
```

Write to OUT

```
PDOUT = i;
```

And reading from OUT and writing to IN

```
i = PDOUT;  // what we put there

PDIN = i;       // ?
```

For every port

```
#define PAIN    (*(unsigned char volatile near*)0xE100)
#define PAOUT   (*(unsigned char volatile near*)0xE101)
#define PADD    (*(unsigned char volatile near*)0xE102)
#define PAHDE   (*(unsigned char volatile near*)0xE103)
#define PAAF    (*(unsigned short volatile near*)0xE104)
#define PAAFH   (*(unsigned char volatile near*)0xE104)
#define PAAFL   (*(unsigned char volatile near*)0xE105)
#define PAOC    (*(unsigned char volatile near*)0xE106)
```

Example

- Set port A
  - Bit 0,2,4,6 as INPUTS
    - Bit 0 as Timer0 input
  - Bit 1,3,5,7 as OUTPUTS
    - Bit 1 as open drain
    - Bit 3 as high drive

Example

```
PADD  = 0x55;
PAAF  = 0x01;
PAOC  = 0x02;
PAHDE = 0x04;
```

Another Example

- Set port B, bit 3 for ANALOG input
  - But leave other bits alone as they are already
    set for something else.

Example

```
PBAF |= 0x04;  // set bit3 (00001000)
```

The magic of &= and |=

```
PDOUT |= 0x0F;      // set the lower 4 bits
PADD  |= 0x10;      // set bit 4

PDOUT &= 0xF0;      // clear the lower 4 bits
PADD  &= 0xEF;      // clear bit 4
PADD  &= ~0x10;     // clear bit 4
```

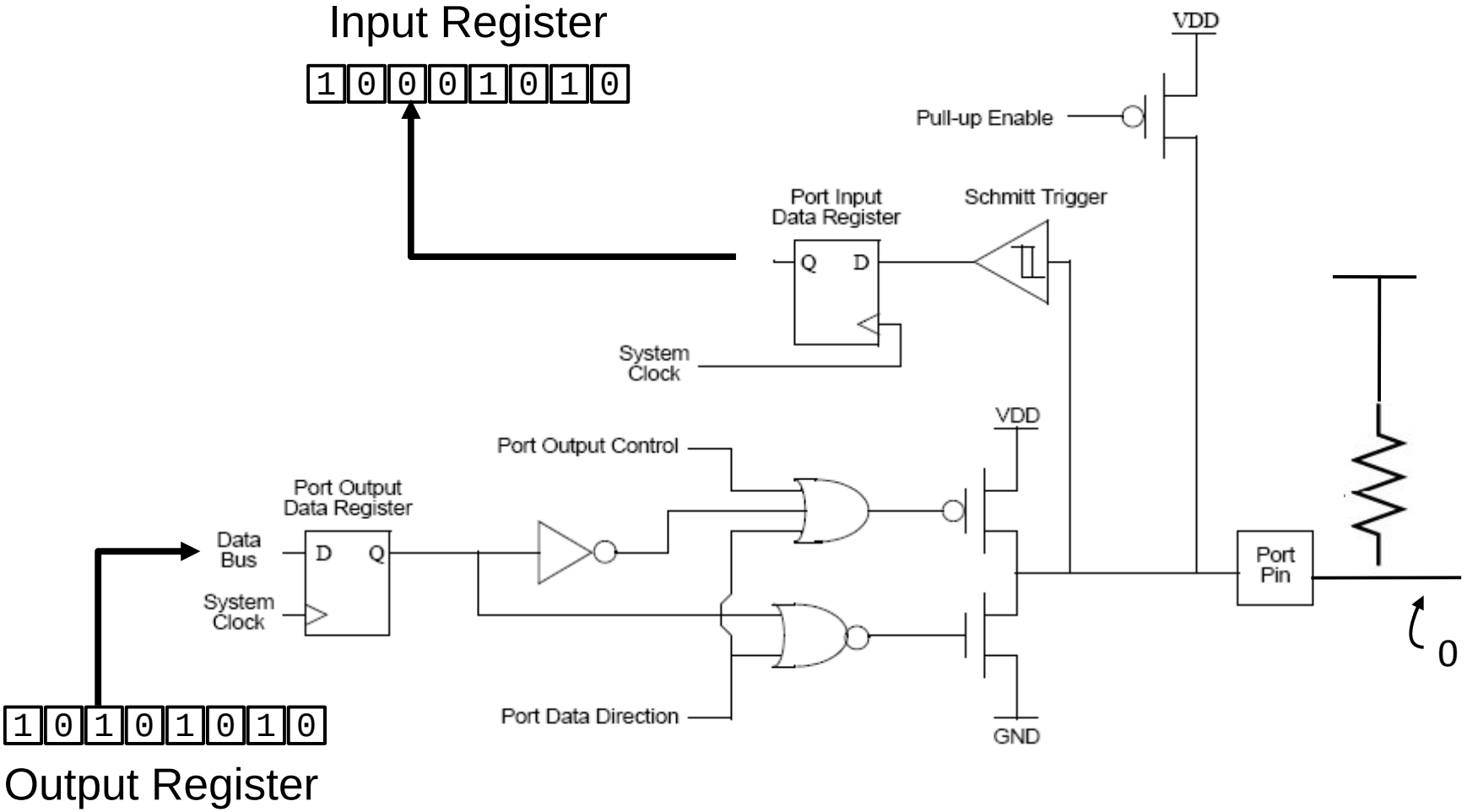
Output Direction

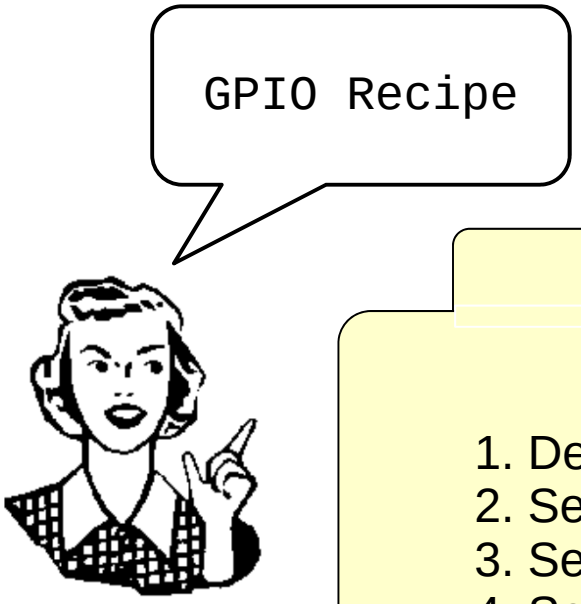
- Would it ever make sense to: read from an OUTPUT port or write to an INPUT port?

Open Drain Output

- Set port to OUTPUT as OPEN DRAIN
- Write a “1”, but external device is writing a “0”
- We can test for a collision by reading from the output port and comparing that to the input port

Write a 1, read a 0!





GPIO

1. Determine which bits need to get set and how
2. Set port data direction
3. Set port alternate function
4. Set port output control
5. Set port drive
6. Set port stop mode recovery

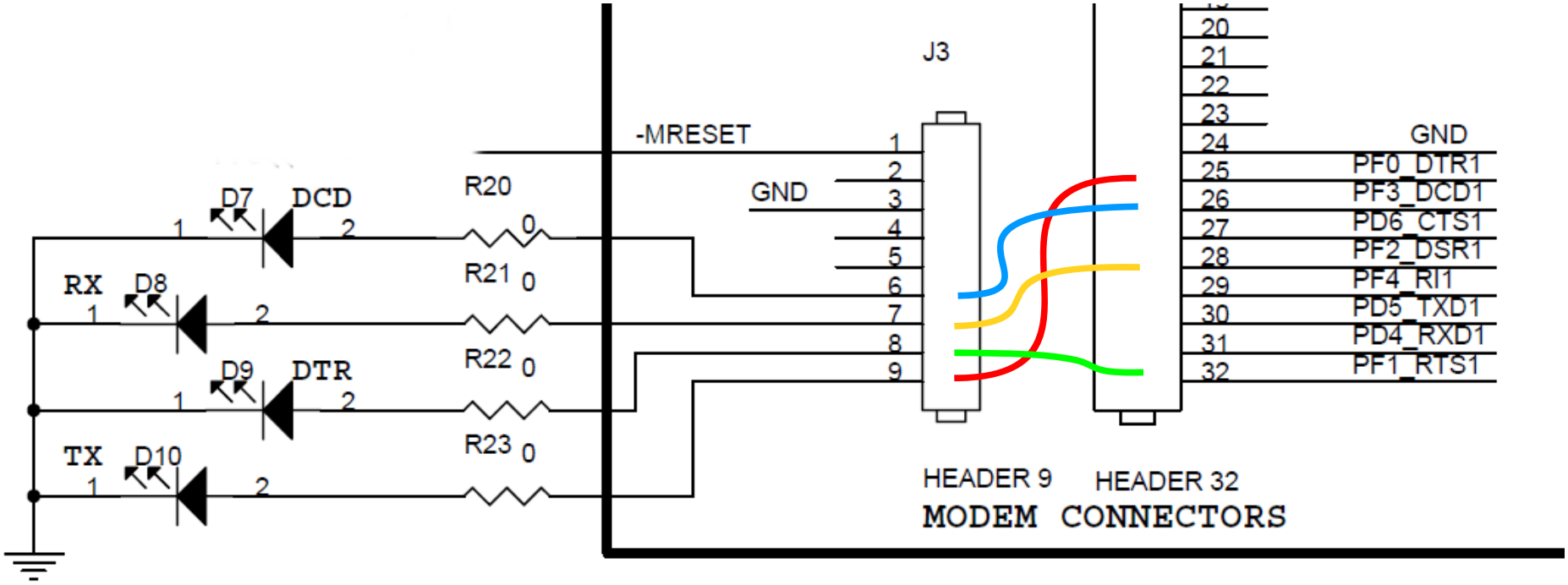
Port A/D and C are special

- have some additional configuration
- PAIMUX1 – Configures port A or D for use as comparator input or debug interrupt
- PAIMUX – Configure port A or D for interrupt source
- PAIEDGE – Selected port A or D interrupt as negative or positive edge
- PCIMUX – Selects if port C is interrupt source or DMA interrupt source

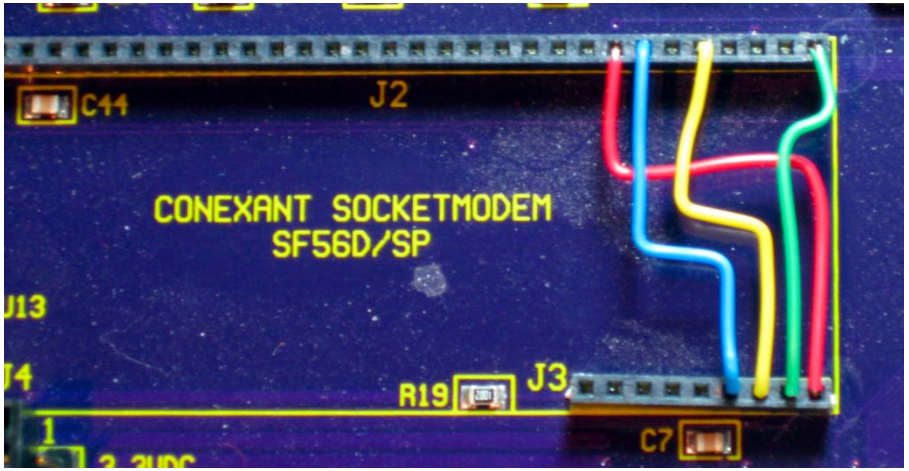
Example

SimpleBlinky

- Show how to count on 4 bits with silly delay
- 3 Versions


J3
-MRESET
GND
D7
DCD
R20
0
RX
D8
R21
0
D9
DTR
R22
0
TX
D10
R23
0
1
2
3
4
5
6
7
8
9
20
21
22
23
24
25
26
27
28
29
30
31
32
GND
PF0_DTR1
PF3_DCD1
PD6_CTS1
PF2_DSR1
PF4_RI1
PD5_TXD1
PD4_RXD1
PF1_RTS1
HEADER 9
HEADER 32
MODEM CONNECTORS

POWER & RS232


C44
J2
CONEXANT SOCKETMODEM
SF56D/SP
R19
J3
C7

3 Versions

- V1 = Basic code. It works.
- V2 = Better. More comments. Functions. Function prototypes. Proper datatype.
- V3 = Modular (use of .h and .c files)

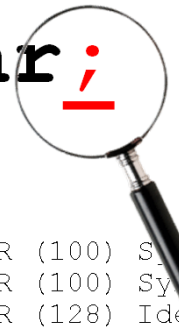
Whats wrong with this?

```
#define byte unsigned char;
```

```
------- example_SimpleBlinky  Configuration: Debug -------
E:\Dan\Projects\ZNEO_programs\example_SimpleBlinky\main2.c
E:\DAN\PROJECTS\ZNEO_PROGRAMS\EXAMPLE_SIMPLEBLINKY\MAIN2.C  (72,31) : ERROR (100) Syntax error
E:\DAN\PROJECTS\ZNEO_PROGRAMS\EXAMPLE_SIMPLEBLINKY\MAIN2.C  (73,33) : ERROR (100) Syntax error
E:\DAN\PROJECTS\ZNEO_PROGRAMS\EXAMPLE_SIMPLEBLINKY\MAIN2.C  (80,22) : ERROR (128) Identifier "i"
    not defined within current scope
Linking...
WARNING (731) --> File ".\main2.obj" is not found.
Build completed.
```

Whats wrong with this?

```
#define byte unsigned char;
```



```
------- example_SimpleBlinky  Configuration: Debug -------
E:\Dan\Projects\ZNEO_programs\example_SimpleBlinky\main2.c
E:\DAN\PROJECTS\ZNEO_PROGRAMS\EXAMPLE_SIMPLEBLINKY\MAIN2.C  (72,31) : ERROR (100) Syntax error
E:\DAN\PROJECTS\ZNEO_PROGRAMS\EXAMPLE_SIMPLEBLINKY\MAIN2.C  (73,33) : ERROR (100) Syntax error
E:\DAN\PROJECTS\ZNEO_PROGRAMS\EXAMPLE_SIMPLEBLINKY\MAIN2.C  (80,22) : ERROR (128) Identifier "i"
    not defined within current scope
Linking...
WARNING (731) --> File ".\main2.obj" is not found.
Build completed.
```

Whats wrong with this?

```
#ifndef SLEEP_H
#define SLEEP_H
      void sleep(int ms)
#endif
```

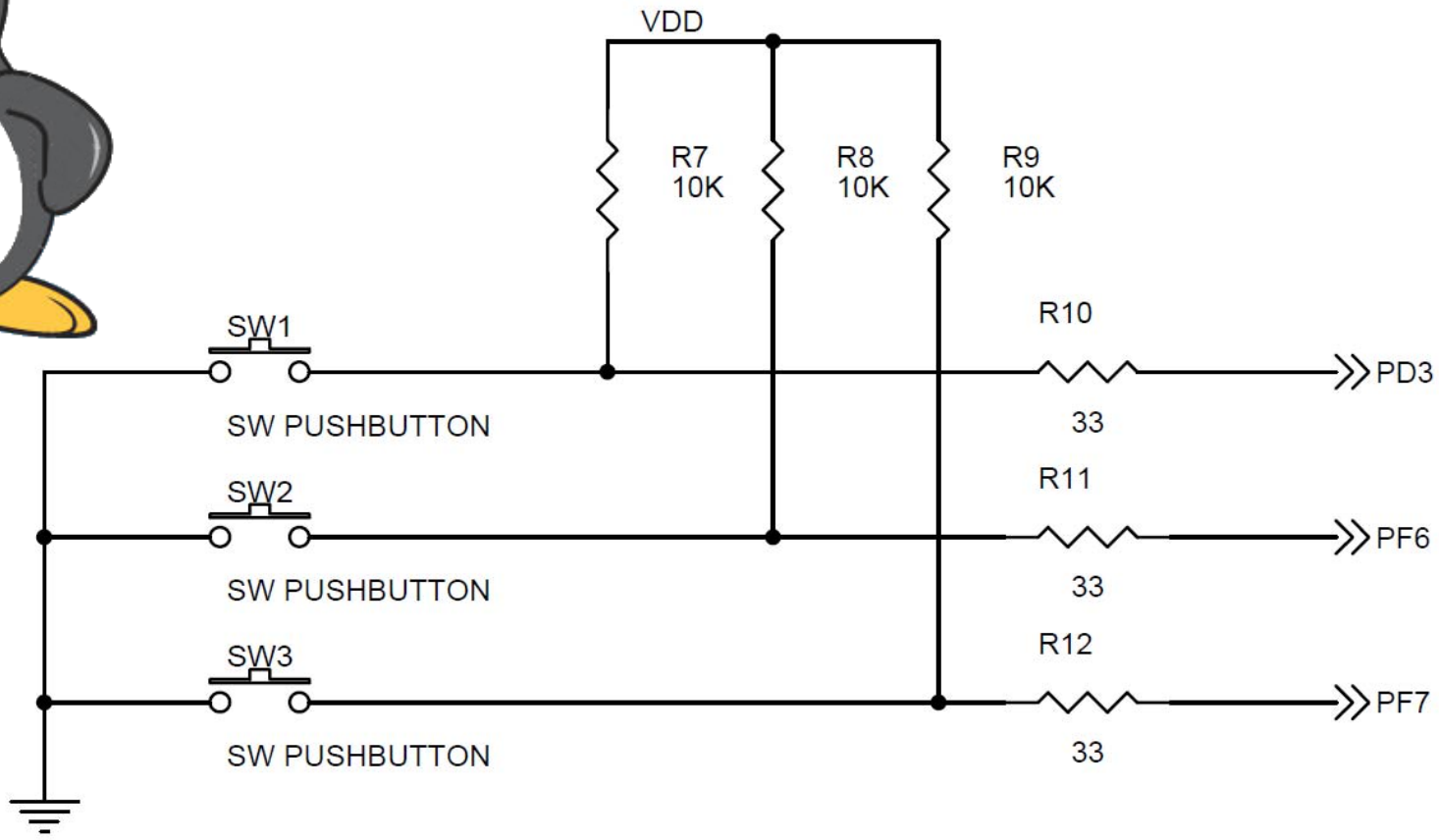
```
------- SimpleBlinky3  Configuration: Debug -------
E:\Dan\Projects\ZNEO_programs\example_SimpleBlinky\sleeps.c
E:\Dan\Projects\ZNEO_programs\example_SimpleBlinky\leds.c
E:\Dan\Projects\ZNEO_programs\example_SimpleBlinky\main3.c
E:\DAN\PROJECTS\ZNEO_PROGRAMS\EXAMPLE_SIMPLEBLINKY\MAIN3.C  (68,18) : ERROR (100) Syntax error
E:\DAN\PROJECTS\ZNEO_PROGRAMS\EXAMPLE_SIMPLEBLINKY\MAIN3.C  (69,13) : ERROR (121) Illegal declaration sp
E:\DAN\PROJECTS\ZNEO_PROGRAMS\EXAMPLE_SIMPLEBLINKY\MAIN3.C  (71,16) : ERROR (100) Syntax error
E:\DAN\PROJECTS\ZNEO_PROGRAMS\EXAMPLE_SIMPLEBLINKY\MAIN3.C  (73,4)  : ERROR (100) Syntax error
E:\DAN\PROJECTS\ZNEO_PROGRAMS\EXAMPLE_SIMPLEBLINKY\MAIN3.C  (74,7)  : ERROR (100) Syntax error
E:\DAN\PROJECTS\ZNEO_PROGRAMS\EXAMPLE_SIMPLEBLINKY\MAIN3.C  (76,15) : ERROR (125)
        Identifier "i" already declared within current scope
E:\DAN\PROJECTS\ZNEO_PROGRAMS\EXAMPLE_SIMPLEBLINKY\MAIN3.C  (76,15) : ERROR (193) Missing "62" detected
E:\DAN\PROJECTS\ZNEO_PROGRAMS\EXAMPLE_SIMPLEBLINKY\MAIN3.C  (76,18) : ERROR (100) Syntax error
E:\DAN\PROJECTS\ZNEO_PROGRAMS\EXAMPLE_SIMPLEBLINKY\MAIN3.C  (78,18) : ERROR (100) Syntax error
E:\DAN\PROJECTS\ZNEO_PROGRAMS\EXAMPLE_SIMPLEBLINKY\MAIN3.C  (79,10) : ERROR (100) Syntax error
E:\DAN\PROJECTS\ZNEO_PROGRAMS\EXAMPLE_SIMPLEBLINKY\MAIN3.C  (81,26) : ERROR (100) Syntax error
E:\DAN\PROJECTS\ZNEO_PROGRAMS\EXAMPLE_SIMPLEBLINKY\MAIN3.C  (82,14) : ERROR (100) Syntax error
E:\DAN\PROJECTS\ZNEO_PROGRAMS\EXAMPLE_SIMPLEBLINKY\MAIN3.C  (84,6) : ERROR (100) Syntax error
Linking...
WARNING (731) --> File ".\main3.obj" is not found.
Build completed.
```

Whats wrong with this?

```
#ifndef SLEEP_H
#define SLEEP_H
      void sleep(int ms);
#endif
```

```
------- SimpleBlinky3  Configuration: Debug -------
E:\Dan\Projects\ZNEO_programs\example_SimpleBlinky\sleeps.c
E:\Dan\Projects\ZNEO_programs\example_SimpleBlinky\leds.c
E:\Dan\Projects\ZNEO_programs\example_SimpleBlinky\main3.c
E:\DAN\PROJECTS\ZNEO_PROGRAMS\EXAMPLE_SIMPLEBLINKY\MAIN3.C  (68,18) : ERROR (100) Syntax error
E:\DAN\PROJECTS\ZNEO_PROGRAMS\EXAMPLE_SIMPLEBLINKY\MAIN3.C  (69,13) : ERROR (121) Illegal declaration sp
E:\DAN\PROJECTS\ZNEO_PROGRAMS\EXAMPLE_SIMPLEBLINKY\MAIN3.C  (71,16) : ERROR (100) Syntax error
E:\DAN\PROJECTS\ZNEO_PROGRAMS\EXAMPLE_SIMPLEBLINKY\MAIN3.C  (73,4)  : ERROR (100) Syntax error
E:\DAN\PROJECTS\ZNEO_PROGRAMS\EXAMPLE_SIMPLEBLINKY\MAIN3.C  (74,7)  : ERROR (100) Syntax error
E:\DAN\PROJECTS\ZNEO_PROGRAMS\EXAMPLE_SIMPLEBLINKY\MAIN3.C  (76,15) : ERROR (125)
        Identifier "i" already declared within current scope
E:\DAN\PROJECTS\ZNEO_PROGRAMS\EXAMPLE_SIMPLEBLINKY\MAIN3.C  (76,15) : ERROR (193) Missing "62" detected
E:\DAN\PROJECTS\ZNEO_PROGRAMS\EXAMPLE_SIMPLEBLINKY\MAIN3.C  (76,18) : ERROR (100) Syntax error
E:\DAN\PROJECTS\ZNEO_PROGRAMS\EXAMPLE_SIMPLEBLINKY\MAIN3.C  (78,18) : ERROR (100) Syntax error
E:\DAN\PROJECTS\ZNEO_PROGRAMS\EXAMPLE_SIMPLEBLINKY\MAIN3.C  (79,10) : ERROR (100) Syntax error
E:\DAN\PROJECTS\ZNEO_PROGRAMS\EXAMPLE_SIMPLEBLINKY\MAIN3.C  (81,26) : ERROR (100) Syntax error
E:\DAN\PROJECTS\ZNEO_PROGRAMS\EXAMPLE_SIMPLEBLINKY\MAIN3.C  (82,14) : ERROR (100) Syntax error
E:\DAN\PROJECTS\ZNEO_PROGRAMS\EXAMPLE_SIMPLEBLINKY\MAIN3.C  (84,6) : ERROR (100) Syntax error
Linking...
WARNING (731) --> File ".\main3.obj" is not found.
Build completed.
```

What does the Z16 “see” when you press a button?



Example

ButtonCounter

- Show how to count on 4 bits on modem status lights. Count button presses.

Whats wrong with this?

```
PFDD |= C0;
```

```
-------- ButtonCounter  Configuration: Debug -------
E:\Dan\Projects\ZNEO_programs\example_ButtonCounter\sleeps.c
E:\Dan\Projects\ZNEO_programs\example_ButtonCounter\leds.c
E:\Dan\Projects\ZNEO_programs\example_ButtonCounter\main.c
Linking...
Build completed.
```

Whats wrong with this?

```
PFDD |= 0xC0;
```

```
-------- ButtonCounter  Configuration: Debug -------
E:\Dan\Projects\ZNEO_programs\example_ButtonCounter\sleeps.c
E:\Dan\Projects\ZNEO_programs\example_ButtonCounter\leds.c
E:\Dan\Projects\ZNEO_programs\example_ButtonCounter\main.c
Linking...
Build completed.
```

Whats wrong with this (to test for button presses)?

```
if ((PDIN & 0x08) |
    (PFIN & 0xC0) == 0xC8)
```

```
-------- ButtonCounter  Configuration: Debug -------
E:\Dan\Projects\ZNEO_programs\example_ButtonCounter\sleeps.c
E:\Dan\Projects\ZNEO_programs\example_ButtonCounter\leds.c
E:\Dan\Projects\ZNEO_programs\example_ButtonCounter\main.c
Linking...
Build completed.
```

Whats wrong with this (to test for button presses)?

```
if ((PDIN & 0x08) |
    (PFIN & 0xC0) == 0xC8)
```

== has higher precedence than |

- Is that what we wanted?

```
(PDIN&0x08) |
((PFIN&0xC0)==0xC8)
```

- Will only succeed when SW1 or (SW2 & SW3) are pressed.

What if we didn't sleep?

```
void main(void) {
    byte i;i=0;
    status_init();buttons_init();
    while(1) {
        if (buttons_pressed()) {
            i++;
        }
        status_set(i);
        if (i>0x0F) i=00;
        sleep(250);
    }
}
```

End of Section Reminder