Lecture 9: General Purpose I/O

ME/AE 6705

Introduction to Mechatronics

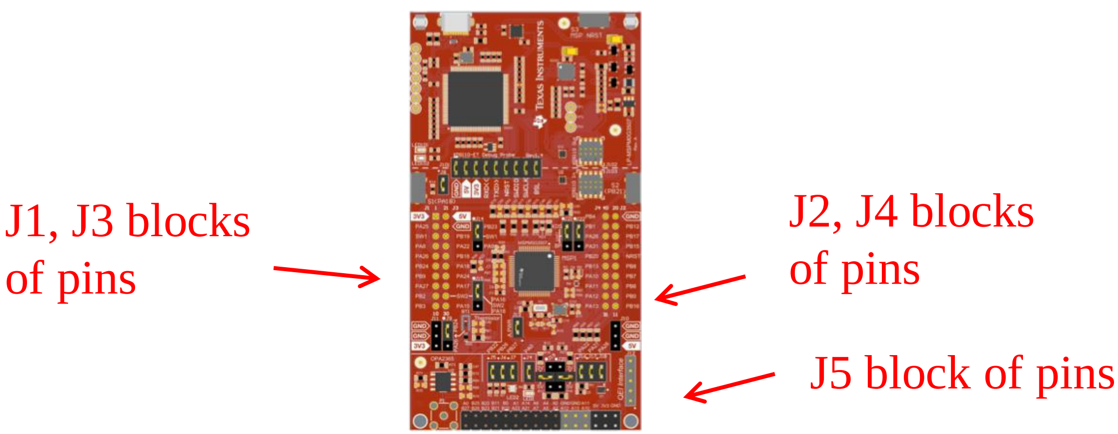
Dr. Jonathan Rogers

Lesson Objectives

- Become familiar with GPIO ports and functionality
- Be able to configure ports for desired functionality
  - Input vs output, drive strength, etc.
- Understand concept of pull-up and pull-down resistors
- Be able to interface external devices (switches & LEDs) with microcontroller
- Be able to configure floating point unit for desired operation

General Purpose Input/Output

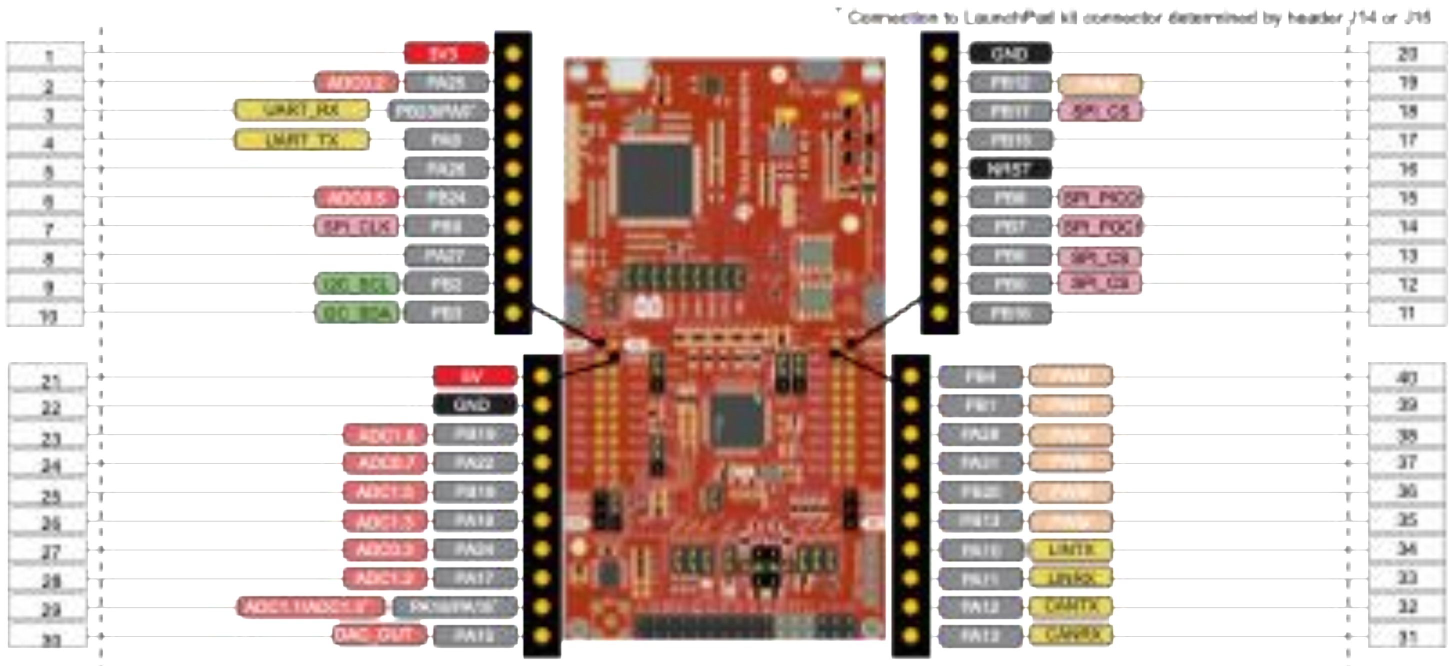
- Incorporation of GPIO ports (pins) is really what separates a microcontroller from a microprocessor
- These ports allow MCU to interface with an extremely wide array of external devices
- MSPMG3507 has total of 60 pins which can be configured for digital input/output



GPIO

- I/O ports on MSPM0 may serve variety of purposes:
  - UART (Universal Asynchronous receiver/transmitter)
  - SPI (Serial Peripheral Interface)
  - I^2C (Inter-Integrated Circuit)
  - TimerA (Periodic interrupts, input capture, output compare)
  - Timer32 (Periodic interrupts)
  - ADC14 (Analog to digital converter to measure analog signals)
  - Analog Comparator (compare two analog signals)
  - Etc…

GPIO Pins on the MSPM0



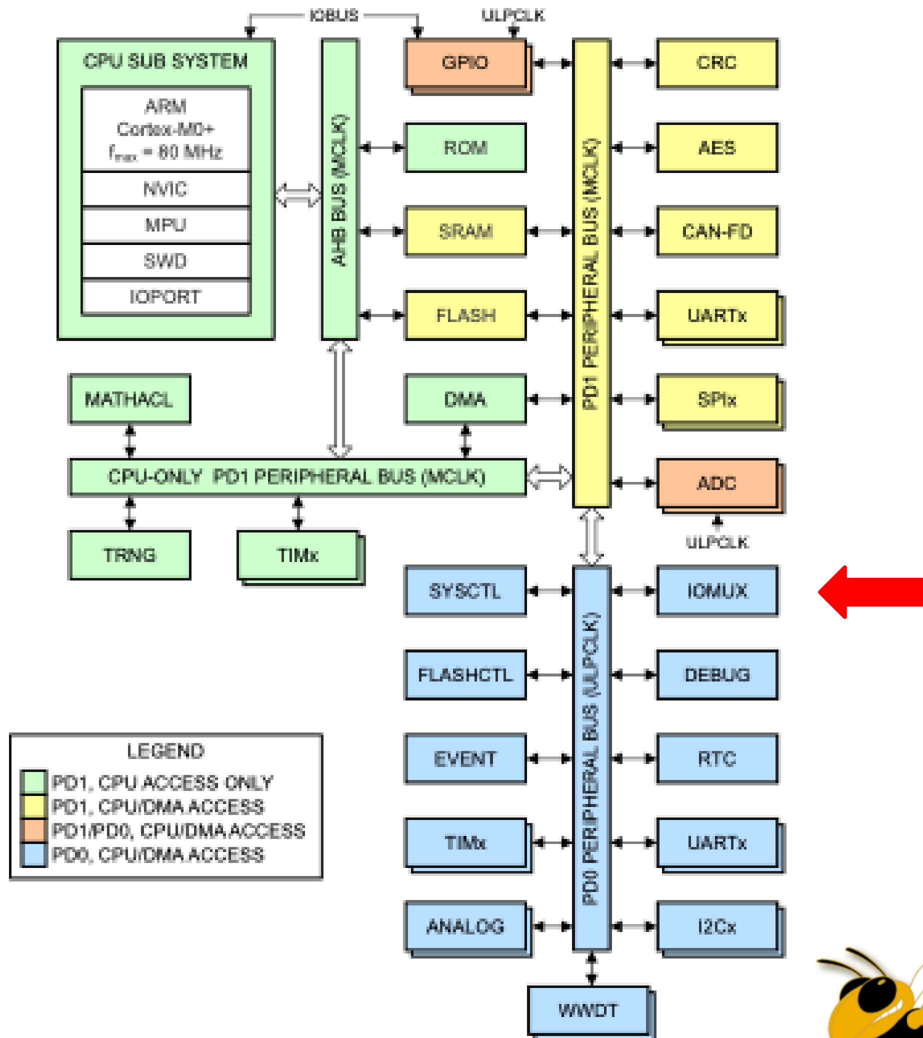
Pins can be configured on an individual basis and can serve multiple purposes

Example: PA18 can be configured for either digital I/O or an Analog-to-Digial Input (channel 1.3).

GPIO Pin Functionality

- Pin functionality is listed in table in MSPM0 datasheet

- Pin functionality set in software by configuring IO Mux (mux = multiplexer)
  - Will get to this shortly



GPIOA vs GPIOB

- MSPM0 has 60 total pins that can be configured for digital I/O
  - Divided into 2 groups GPIOA (32 pins) and GPIOB (28 pins)
  - This is done so they can be mapped to single bits in a 32-bit register
- Pin 22 on port A is called PA22, etc.

Table 9-1. GPIO Port and Device Pin Mapping

GPIO Port and Bit Name	Device Pin Signal Name
GPIO Port A Bit 0 (DIO0)	PA0
GPIO Port A Bit 1 (DIO1)	PA1
GPIO Port B Bit 0 (DIO0)	PB0
GPIO Port B Bit 1 (DIO1)	PB1
GPIO Port x Bit y (DIOy)	Pxy

GPIO Pin Functionality

- Functions of different pins and associated configurations of IOMUX can be found in Table 6.1 of MSPM0G3507 datasheet (p. 10)

Table 6-1. Pin Attributes

PINCMx	PIN NAME	SIGNAL NAMES: ANALOG	SIGNAL NAMES: DIGITAL *[PIN FUNCTION]* [(1)]	PIN NUMBER: 64 LQFP	PIN NUMBER: 48 LQFP, VQFN	PIN NUMBER: 32 VQFN	PIN NUMBER: 28 VSSOP	IO STRUCTURE
N/A		VDD		40	6	4	7	Power
N/A		VSS		41	7	5	8	Power
N/A		VCORE		32	48	32	3	Power
N/A		NRST		38	4	3	6	Reset
1	PA0		UART0_TX *[2]* / I2C0_SDA *[3]* / TIMA0_C0 *[4]* / TIMA_FAL1 *[5]* / TIMG8_C1 *[6]* / FCC_IN *[7]* /(Default BSL I2C_SDA)	33	1	1	4	5V Tol. Open-Drain
2	PA1		UART0_RX *[2]* / I2C0_SCL *[3]* / TIMA0_C1 *[4]* / TIMA_FAL2 *[5]* / TIMG8_IDX *[6]* / TIMG8_C0 *[7]* /(Default BSL I2C_SCL)	34	2	2	5	5V Tol. Open-Drain
7	PA2	ROSC	TIMG8_C1 *[2]* / SPI0_CS0 *[3]* / TIMG7_C1 *[4]* / SPI1_CS0 *[5]*	42	8	6	9	Standard
8	PA3	LFXIN	TIMG8_C0 *[2]* / SPI0_CS1 [3] / UART2_CTS [4] / TIMA0_C2 [5] / COMP1_OUT [6] / TIMG7_C0 [7] / TIMA0_C1 [8] / I2C1_SDA [9]	43	9	7	10	Standard
9	PA4	LFXOUT	TIMG8_C1 *[2]* / SPI0_POCI *[3]* / UART2_RTS *[4]* / TIMA0_C3 *[5]* / LFCLK_IN *[6]* / TIMG7_C1 *[7]* / TIMA0_C1N *[8]* / I2C1_SCL *[9]*	44	10	8	11	Standard
10	PA5	HFXIN	TIMG8_C0 *[2]* / SPI0_PICO *[3]* / TIMA_FAL1 *[4]* / TIMG0_C0 *[5]* / TIMG6_C0 *[6]* / FCC_IN *[7]*	45	11	9	12	Standard
11	PA6	HFXOUT	TIMG8_C1 *[2]* / SPI0_SCK *[3]* / TIMA_FAL0 *[4]* / TIMG0_C1 *[5]* / HFCLK_IN *[6]* / TIMG6_C1 *[7]* / TIMA0_C2N *[8]*	46	12	10	13	Standard

GPIO Pin Functionality

IO Mux pin index

GPIO pin name

Pin number on actual chip

PINCM = pin configuration module

Table 6-1. Pin Attributes

PINCMx	PIN NAME	SIGNAL NAMES		PIN NUMBER				IO STRUCTURE
		ANALOG	DIGITAL *[PIN FUNCTION]* (1)	64 LQFP	48 LQFP, VQFN	32 VQFN	28 VSSOP	
N/A		VDD		40	6	4	7	Power
N/A		VSS		41	7	5	8	Power
N/A		VCORE		32	48	32	3	Power
N/A		NRST		38	4	3	6	Reset
1	PA0		UART0_TX *[2]* / I2C0_SDA *[3]* / TIMA0_C0 *[4]* / TIMA_FAL1 *[5]* / TIMG8_C1 *[6]* / FCC_IN *[7]* /(Default BSL I2C_SDA)	33	1	1	4	5V Tol. Open-Drain
2	PA1		UART0_RX *[2]* / I2C0_SCL *[3]* / TIMA0_C1 *[4]* / TIMA_FAL2 *[5]* / TIMG8_IDX *[6]* / TIMG8_C0 *[7]* /(Default BSL I2C_SCL)	34	2	2	5	5V Tol. Open-Drain
7	PA2	ROSC	TIMG8_C1 *[2]* / SPI0_CS0 *[3]* / TIMG7_C1 *[4]* / SPI1_CS0 *[5]*	42	8	6	9	Standard
8	PA3	LFXIN	TIMG8_C0 *[2]* / SPI0_CS1 [3] / UART2_CTS [4] / TIMA0_C2 [5] / COMP1_OUT [6] / TIMG7_C0 [7] / TIMA0_C1 [8] / I2C1_SDA [9]	43	9	7	10	Standard
9	PA4	LFXOUT	TIMG8_C1 *[2]* / SPI0_POCI *[3]* / UART2_RTS *[4]* / TIMA0_C3 *[5]* / LFCLK_IN *[6]* / TIMG7_C1 *[7]* / TIMA0_C1N *[8]* / I2C1_SCL *[9]*	44	10	8	11	Standard
10	PA5	HFXIN	TIMG8_C0 *[2]* / SPI0_PICO *[3]* / TIMA_FAL1 *[4]* / TIMG0_C0 *[5]* / TIMG6_C0 *[6]* / FCC_IN *[7]*	45	11	9	12	Standard
11	PA6	HFXOUT	TIMG8_C1 *[2]* / SPI0_SCK *[3]* / TIMA_FAL0 *[4]* / TIMG0_C1 *[5]* / HFCLK_IN *[6]* / TIMG6_C1 *[7]* / TIMA0_C2N *[8]*	46	12	10	13	Standard

GPIO Pin Functionality

- To set a certain pin to be a digital input/output, set bits of IOMUX->SECCFG.PINCM register

8.3.1 PINCM (Offset = 4h) [Reset = X]

Pin Control Management Register

Figure 8-3. PINCM

31	30	29	28	27	26	25	24
RESERVED			WCOMP	WUEN	INV	HIZ1	RESERVED
R/W-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
RESERVED			DRV	HYSTEN	INENA	PIPU	PIPD
R/W-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RESERVED		WAKESTAT	RESERVED				
R/W-0h		R-0h	R/W-0h				
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PC	RESERVED	PF					
R/W-0h	R/W-0h	R/W-0h					

GPIO Pin Functionality

- Important bits:

8.3.1 PINCM (Offset = 4h) [Reset = X]

Pin Control Management Register

Figure 8-3. PINCM

31	30	29	28	27	26	25	24
RESERVED			WCOMP	WUEN	INV	HIZ1	RESERVED
R/W-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
RESERVED			DRV	HYSTEN	INENA	PIPU	PIPD
R/W-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
RESERVED		WAKESTAT	RESERVED				
R/W-0h		R-0h	R/W-0h				
7	6	5	4	3	2	1	0
PC	RESERVED	PF					
R/W-0h	R/W-0h	R/W-0h					

PF (Bits 0-5): Sets pin functionality. Digital IO function is always 0x01.

PC (Bit 7): Peripheral connect. Must be 1 for pin to be connected to desired peripheral.

GPIO Pin Functionality

- Important bits:

8.3.1 PINCM (Offset = 4h) [Reset = X]

Pin Control Management Register

Figure 8-3. PINCM

31	30	29	28	27	26	25	24
RESERVED			WCOMP	WUEN	INV	HIZ1	RESERVED
R/W-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
RESERVED			DRV	HYSTEN	INENA	PIPU	PIPD
R/W-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RESERVED		WAKESTAT	RESERVED				
R/W-0h		R-0h	R/W-0h				
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PC	RESERVED	PF					
R/W-0h	R/W-0h	R/W-0h					

INENA (Bit 18): Sets pin as an input pin. If zero, it is treated as output pin.

PIPU or PIPD (Bits 16, 17): Sets whether pin is connected to pull-down or pull-up resistor.

Bit functions explained in Table 8.3 of Tech. Ref. Manual.

GPIO Pin Functionality

- To set up pins as other functions that are not digital input/output, consult table 6.1 in Datasheet

Table 6-1. Pin Attributes

PINCMx	PIN NAME	SIGNAL NAMES: ANALOG	SIGNAL NAMES: DIGITAL *[PIN FUNCTION]* (1)	PIN NUMBER: 64 LQFP	PIN NUMBER: 48 LQFP, VQFN	PIN NUMBER: 32 VQFN	PIN NUMBER: 28 VSSOP	IO STRUCTURE
N/A		VDD		40	6	4	7	Power
N/A		VSS		41	7	5	8	Power
N/A		VCORE		32	48	32	3	Power
N/A		NRST		38	4	3	6	Reset
1	PA0		UART0_TX *[2]* / I2C0_SDA *[3]* / TIMA0_C0 *[4]* / TIMA_FAL1 *[5]* / TIMG8_C1 *[6]* / FCC_IN *[7]* /(Default BSL I2C_SDA)	33	1	1	4	5V Tol. Open-Drain
2	PA1		UART0_RX *[2]* / I2C0_SCL *[3]* / TIMA0_C1 *[4]* / TIMA_FAL2 *[5]* / TIMG8_IDX *[6]* / TIMG8_C0 *[7]* /(Default BSL I2C_SCL)	34	2	2	5	5V Tol. Open-Drain
7	PA2	ROSC	TIMG8_C1 *[2]* / SPI0_CS0 *[3]* / TIMG7_C1 *[4]* / SPI1_CS0 *[5]*	42	8	6	9	Standard
8	PA3	LFXIN	TIMG8_C0 *[2]* / SPI0_CS1 [3] / UART2_CTS [4] / TIMA0_C2 [5] / COMP1_OUT [6] / TIMG7_C0 [7] / TIMA0_C1 [8] / I2C1_SDA [9]	43	9	7	10	Standard
9	PA4	LFXOUT	TIMG8_C1 *[2]* / SPI0_POCI *[3]* / UART2_RTS *[4]* / TIMA0_C3 *[5]* / LFCLK_IN *[6]* / TIMG7_C1 *[7]* / TIMA0_C1N *[8]* / I2C1_SCL *[9]*	44	10	8	11	Standard
10	PA5	HFXIN	TIMG8_C0 *[2]* / SPI0_PICO *[3]* / TIMA_FAL1 *[4]* / TIMG0_C0 *[5]* / TIMG6_C0 *[6]* / FCC_IN *[7]*	45	11	9	12	Standard
11	PA6	HFXOUT	TIMG8_C1 *[2]* / SPI0_SCK *[3]* / TIMA_FAL0 *[4]* / TIMG0_C1 *[5]* / HFCLK_IN *[6]* / TIMG6_C1 *[7]* / TIMA0_C2N *[8]*	46	12	10	13	Standard

For this pin, set PC field to 0x01 for digital input/output, 0x02 for UART receive, 0x03 for I2C SCL, etc.

Other pins will be different.

GPIO Pin Functionality

- Each pin has its own 32-bit PINCM register
  - Can be accessed in software using the following syntax
  - Pins are indexed 1-60, register offsets are 0-59

Set pin functionality for Pin 11:

```
IOMUX->SECCFG.PINCM[10] = ...
```

Set pin functionality for Pin 48:

```
IOMUX->SECCFG.PINCM[47] = ...
```

GPIO Pin Functionality

- Pins can be configured to be either <u>input</u> or <u>output</u>
  - Cannot be configured to be both at the same time
- **Input** pins allow us to read external signals. For instance:
  - Read an analog signal using ADC
  - Check whether a pin is set high
  - Check whether a button is being pressed
- **Output** pins allow us to send data out from MCU. For instance:
  - PWM control of motor
  - Sending serial data

GPIO Pin Functionality

- When setting up pin, need to configure following bits in PINCM register (at a minimum):
  - Connect pin to peripheral by setting PC bit (bit 7)
  - Set pin functionality by selecting appropriate number for PF bits (bits 0-4)
    - *E.g., 00001 for digital input/output functionality*
  - If pin will be input pin, set INENA bit (bit 18)
  - If using pull up or pull down resistor, set PIPU or PIPD bit (bit 16 or 17)
- Bits can be set simultaneously by combining macros using bitwise OR

GPIO Pin Functionality

- Example: Set up pin PA17 for UART transmit function

39	PA17	A1_2 / OPA1_IN1- / COMP0_IN1-	UART1_TX *[2]* / SPI1_SCK *[3]* / I2C1_SCL *[4]* / TIMA0_C3 *[5]* / TIMG7_C0 *[6]* / TIMA1_C0 *[7]*	10	32	21	20	Standard with wake[2]

```
IOMUX->SECCFG.PINCM[IOMUX_PINCM39] = IOMUX_PINCM_PC_CONNECTED | 0x00000002 ;
```

Macro to decimal 38 (mspm0g350x.h)

Macro to 0x00000080 – to set the peripheral connect (PC) bit (hw_iomux.h)

Sets pin function (PF) field to 2 (binary 00010, or 0x02). Need 32-bit number because register is 32 bit.

GPIO Pin Functionality

- Example: Set up pin PB7 as digital output

24	PB7		UART1_RX *[2]* / SPI1_POCI *[3]* / SPI0_CS2 *[4]* / TIMG8_C1 *[5]* / UART2_RTS *[6]* / TIMG6_C1 *[7]* / TIMA1_C1N *[8]*	59	21	–	–	Standard

```
IOMUX->SECCFG.PINCM[IOMUX_PINCM24] = IOMUX_PINCM_PC_CONNECTED | 0x00000001 ;
```

Macro to decimal 23 (mspm0g350x.h)

Macro to 0x00000080 – to set the peripheral connect (PC) bit (hw_iomux.h)

Sets pin function (PF) field to 1 (binary 00001, or 0x01). Need 32-bit number because register is 32 bit.

GPIO Pin Functionality

- Example: Set up pin PB21 as digital input with pull-up resistor

49	PB21	COMP2_IN0+	SPI1_POCI *[2]* / TIMG8_C0 *[3]*	20	–	–	–	Standard

Macro to 0x00000080 – to set the peripheral connect (PC) bit

Sets pin function (PF) field to 1 (binary 00001, or 0x01).

```
IOMUX->SECCFG.PINCM[IOMUX_PINCM49] = IOMUX_PINCM_PC_CONNECTED | 0x00000001 |
                    IOMUX_PINCM_INENA_ENABLE |  0x00020000 ;
```

Macro to decimal 48 (mspm0g350x.h)

Macro to 0x00040000 - sets input enable bit (bit 18) to 1.

Set pull up resistor bit (bit 17) to 1.

Reading from an Input Pin

- Read from a pin by creating a 32-bit variable and setting it equal to register GPIOA->DIN31_0 or GPIOB->DIN31_0
  – GPIOA->DIN31_0 or GPIOB->DIN31_0 is hardware address of input port, can be read just like memory location

Figure 9-56. DIN31_0

31	30	29	28	27	26	25	24
DIO31	DIO30	DIO29	DIO28	DIO27	DIO26	DIO25	DIO24
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
23	22	21	20	19	18	17	16
DIO23	DIO22	DIO21	DIO20	DIO19	DIO18	DIO17	DIO16
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
15	14	13	12	11	10	9	8
DIO15	DIO14	DIO13	DIO12	DIO11	DIO10	DIO9	DIO8
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1	DIO0
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

Reading from an Input Pin

- Example: Read bits from digital input register on GPIOA

```
uint32_t gpioaValue = 0 ;          // declare variable

gpioaValue = GPIOA->DIN31_0 ;      // read from GPIOA
```

  - Suppose pins PA4 and PA13 were high and all others were low at this time. What decimal value would gpioaValue be?

Reading from an Input Pin

- Example: Read bits from digital input register on GPIOA

```
uint32_t gpioaValue = 0 ;          // declare variable

gpioaValue = GPIOA->DIN31_0 ;      // read from GPIOA
```

  - Suppose pins PA4 and PA13 were high and all others were low at this time. What decimal value would gpioaValue be?

$2^4 + 2^{13}$ = 8208 (or 0x00002010)

Reading from an Input Pin

- Oftentimes you only use some pins on a given port as inputs
  - To read only these input pins and ignore the rest, you use a <u>bitmask</u>
- Example: Read only pins PA1 and PA4

```
uint32_t pinValues_1_4 = 0 ;                              // declare variable

pinValues_1_4 = GPIOA->PIN31_0 & 0x00000012 ;  // read from GPIOA
```

Bitmask of 0x00000012 = 00000000000000000000000000010010 in binary

(will only let current value of pins 1 and 4 be reflected in output to variable)

Example: Reading from a Pin

- Consider example where we read only pins PA1 and PA4 using a bitmask:

```
uint32_t pinValues_1_4 = 0 ;                          // declare variable

pinValues_1_4 = GPIOA->PIN31_0 & 0x00000012 ;  // read from GPIOA
```

  – How many different values can variable pinValues_1_4 take on? What are they?

Reading from a Pin

- MSPM0 also has “alias registers” which provide byte-level access to individual bits
  - Makes using bitmasks unnecessary if you know how to use them

Figure 9-48. DIN3_0

31	30	29	28	27	26	25	24
RESERVED							DIO3
R-0h							R-0h
23	22	21	20	19	18	17	16
RESERVED							DIO2
R-0h							R-0h
15	14	13	12	11	10	9	8
RESERVED							DIO1
R-0h							R-0h
7	6	5	4	3	2	1	0
RESERVED							DIO0
R-0h							R-0h

DIN3_0 register provides bits associated with PA0–PA3 or PB0-PB3

Example: Reading from a Pin

- Example: Getting value of PA3 and checking whether it is 1

```
if( GPIOA->DIN0_3 == 0x01000000 ) {

...

}
```

  - When reading from DIN0_3, DIN4_7, DIN_8_11, etc. registers only need to compare against simple mask of 0x01000000, 0x00010000, 0x00000100, or 0x00000001 to see if any of 4 bits are high

Writing to an Output Pin

- Writing to an output port can be done in similar manner using the assignment operator
- To write to output port, we set bits of **GPIOA->DOUT31_0 or GPIOB->DOUT31_0** register
- Like reading from an input port, we must write all 32 bits of output port at once
  - Or we can use bitwise operator techniques to keep current values constant and only flip certain bits (using |= or &=)

Writing to an Output Pin

- Can also use "alias registers" when writing to an output to make things easier
  - DOUT0_3, DOUT4_7, etc.

Figure 9-34. DOUT7_4

31	30	29	28	27	26	25	24
RESERVED							DIO7
W-0h							W-0h
23	22	21	20	19	18	17	16
RESERVED							DIO6
W-0h							W-0h
15	14	13	12	11	10	9	8
RESERVED							DIO5
W-0h							W-0h
7	6	5	4	3	2	1	0
RESERVED							DIO4
W-0h							W-0h

Writing to an Output Port

- Example: Write all bits of GPIOA

```
GPIOA->DOUT31_0 = 0x0F7A3005 ;     // write all 32 bits of GPIOA
```

- Example: Set PA3 and PA5 high, keep all others at their current state

```
GPIOA->DOUT31_0 |= 0x00000028 ;      // write PA3 and PA5 high
```

- Example: Set PA6 and PA7 low

```
GPIOA->DOUT31_0 &= ~0x000000C0 ;     // write P2.3 and P2.5 low
```

Writing to an Output Port

- Example: Set pin PB17 high

```
GPIOB->DOUT19_16 |= 0x00000100 ;  // write PB17 high, leave bits 16, 18, 19 as is
```

Figure 9-37. DOUT19_16

31	30	29	28	27	26	25	24
RESERVED							DIO19
W-0h							W-0h
23	22	21	20	19	18	17	16
RESERVED							DIO18
W-0h							W-0h
15	14	13	12	11	10	9	8
RESERVED							DIO17
W-0h							W-0h
7	6	5	4	3	2	1	0
RESERVED							DIO16
W-0h							W-0h

Writing to an Output Port

- Example: Clear pin PB15 (set to zero)

```
GPIOB->DOUT15_12 &= ~0x01000000 ;  // clear PB15, leave bits 12, 13, 14 as is
```

Figure 9-36. DOUT15_12

31	30	29	28	27	26	25	24
RESERVED							DIO15
W-0h							W-0h
23	22	21	20	19	18	17	16
RESERVED							DIO14
W-0h							W-0h
15	14	13	12	11	10	9	8
RESERVED							DIO13
W-0h							W-0h
7	6	5	4	3	2	1	0
RESERVED							DIO12
W-0h							W-0h

Writing to an Output Pin

- To write to output pin, must enable output by setting appropriate bit of **GPIOA->DOE31_0 or GPIOB->DOE31_0** register
  – Done after configuring pin and after setting pin to appropriate state

Figure 9-45. DOE31_0

31	30	29	28	27	26	25	24
DIO31	DIO30	DIO29	DIO28	DIO27	DIO26	DIO25	DIO24
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
DIO23	DIO22	DIO21	DIO20	DIO19	DIO18	DIO17	DIO16
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
DIO15	DIO14	DIO13	DIO12	DIO11	DIO10	DIO9	DIO8
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1	DIO0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

Writing to an Output Pin

- Example: Set up pins PA3 and PA5 as output pins

```
// Configure pins PA3 and PA5 as digital output pins
IOMUX->SECCFG.PINCM[IOMUX_PINCM8] = IOMUX_PINCM_PC_CONNECTED | 0x00000001 ;

IOMUX->SECCFG.PINCM[IOMUX_PINCM10] = IOMUX_PINCM_PC_CONNECTED | 0x00000001 ;

GPIOA->DOUT31_0 &= ~0x00000028 ;      // set PA3 and PA5 low

GPIOA->DOE31_0 |= 0x00000028 ;        // enable output on PA3 and PA5
```

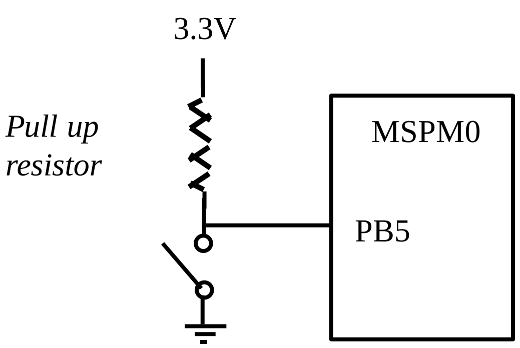
Notes on Input and Output

- Writing to an input pin will have no effect
  - Compiler will not care, operation just will not do anything
- You can read from an output pin
  - Simply use assignment (=) operator as for input pin
  - This can be helpful if you aren't sure what state of pin is (high or low)
- Some pins on single port (GPIOA or GPIOB) may be set as input, some as output
  - Thus you need to be careful and keep track

Pull Up and Pull Down Resistors

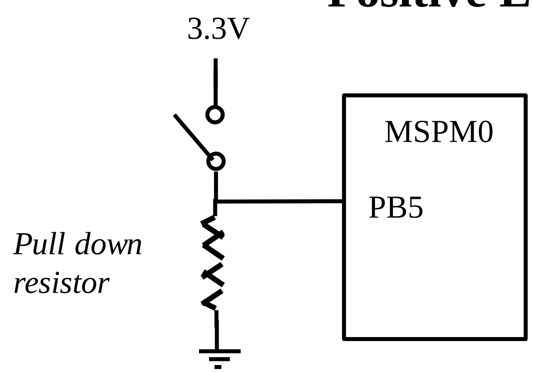
- Suppose you want to interface a switch to one of the GPIO pins on the MSPM0
- Two possible ways to do it: positive logic and negative logic

Negative Logic



When switch is open (off),
MCU pin reads high.

Positive Logic

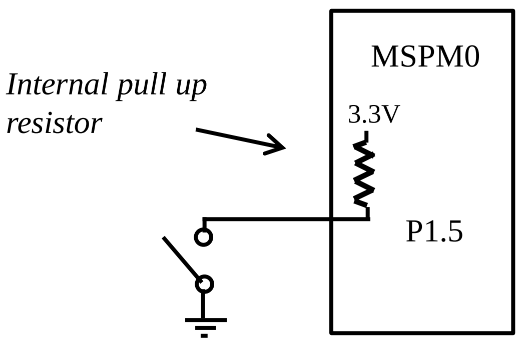


When switch is open (off),
MCU pin reads low.

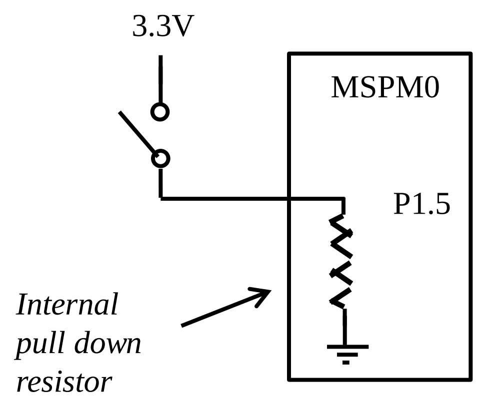
Pull Up and Pull Down Resistors

- These circuits can be constructed externally
  - i.e., by manually incorporating pull up or pull down resistors
- MSPM0 (and many other MCU’s) have pull up and pull down resistors internally that can be used for this purpose

Negative Logic (internal)



Positive Logic (internal)



Note: All internal resistors have 40 kΩ resistance.

Pull Up and Pull Down Resistors

- To activate internal pull up/down resistors, PIPU (pull up) or PIPD (pull down) bits in PINCM register

8.3.1 PINCM (Offset = 4h) [Reset = X]

Pin Control Management Register

Figure 8-3. PINCM

31	30	29	28	27	26	25	24
RESERVED			WCOMP	WUEN	INV	HIZ1	RESERVED
R/W-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
RESERVED			DRV	HYSTEN	INENA	PIPU	PIPD
R/W-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RESERVED		WAKESTAT	RESERVED				
R/W-0h		R-0h	R/W-0h				
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PC	RESERVED	PF					
R/W-0h	R/W-0h	R/W-0h					

- Example: Set PB5 as input with pull-down resistor

```
IOMUX->SECCFG.PINCM[IOMUX_PINCM18] = IOMUX_PINCM_PC_CONNECTED | 0x00000001 |
                    IOMUX_PINCM_INENA_ENABLE |  0x00010000 ;
```

Set PIPD bit (bit 16) high

Drive Strength

- On the MSPM0, the maximum current a pin can output is **6 mA**
  - This is called *Standard Drive Strength*
  - Note: This is not enough to run any mechanical device (e.g., motor, etc). This is what transistors are used for.
- Pins PA10, PA11, PA28, PA31 can be operated in High Drive Strength mode
  - In this mode their current limit increases to 20 mA
  - All other pins are limited to the 6 mA value

Drive Strength

- Drive strength can be set by setting DRV bit in PINCM register

8.3.1 PINCM (Offset = 4h) [Reset = X]

Pin Control Management Register

Figure 8-3. PINCM

31	30	29	28	27	26	25	24
RESERVED			WCOMP	WUEN	INV	HIZ1	RESERVED
R/W-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
RESERVED			DRV	HYSTEN	INENA	PIPU	PIPD
R/W-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
RESERVED		WAKESTAT	RESERVED				
R/W-0h		R-0h	R/W-0h				
7	6	5	4	3	2	1	0
PC	RESERVED	PF					
R/W-0h	R/W-0h	R/W-0h					

Set DRV bit (bit 20) high

```
IOMUX->SECCFG.PINCM[IOMUX_PINCM21] = IOMUX_PINCM_PC_CONNECTED | 0x00000001 |
                    0x00100000 ;
```

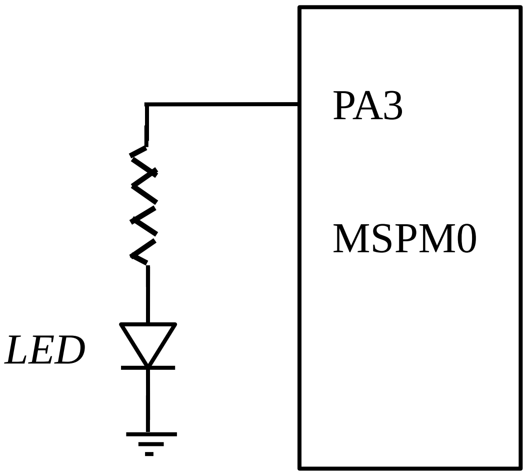
- Typical current levels for LED's – 2-15 mA
- Typical current levels for brushed motors – 1-4 A

Port Configuration Registers

- Summary of registers for digital IO pin configuration, reading, and writing

Register Name	Purpose
IOMUX->SECCFG.PINCM[IOMUX_PINCMXX]	Configure pin as digital input or output, set direction (output or input), set drive strength, connect pull up or pull down resistor
GPIOA->DOUT31_0, GPIOB->DOUT31_0	Holds digital output values on each pin (write)
GPIOA->DIN31_0, GPIOB->DIN31_0	Holds digital input values on each pin (read)
GPIOA->DOE31_0, GPIOB->DOE31_0	Enables pins as digital outputs

LED Circuit Examples

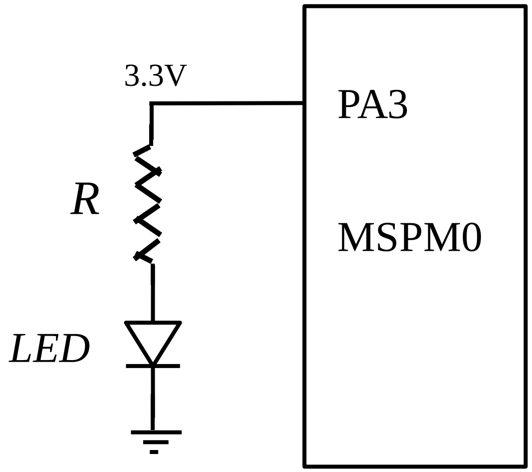



```
void initialize_pins(){

  // Configure pin PA3 as output pin
  IOMUX->SECCFG.PINCM[IOMUX_PINCM8] =
                IOMUX_PINCM_PC_CONNECTED | 0x00000001 ;

  // Set output high initially to turn LED off
  GPIOA->DOUT31_0 = 0x000000080 ;

  // Enable output
  GPIOA->DOE31_0 = 0x000000080 ;
}

void main(){

  int i = 0 ;
  initialize_pins() ; // Initialize port for LED output

  while(1){

    // Toggle LED
    GPIOA->DOUT31_0 ^= 0x000000080 ;

    // Delay
    for(i=0; i < 10000; i++){}
  }
}
```

- Positive Logic – setting pin high turns LED on
- Note: This will work only if LED needs less then 6 mA

LED Circuit Examples

- Suppose LED requires 2 mA to be lit
  - Forward biased voltage of 1.7 V (HLMP-4700)
- Design resistor such that current through LED is 2 mA when PA3 is set high (3.3V)



Voltage drop across resistor:

$$V_R = 3.3 - 1.7 = 1.6\,\text{V}$$

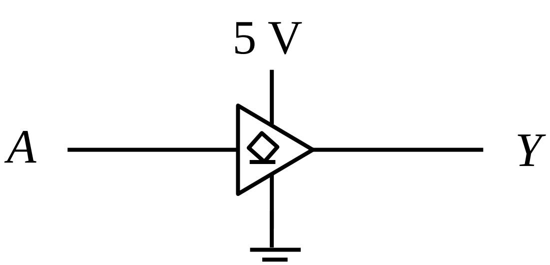
To achieve a current of 2 mA:

$$V_R = (2\text{ mA})R$$

$$\rightarrow R = 1.6\text{ V} / 2\text{ mA} = 800\ \Omega$$

LED Circuit Example: High Current

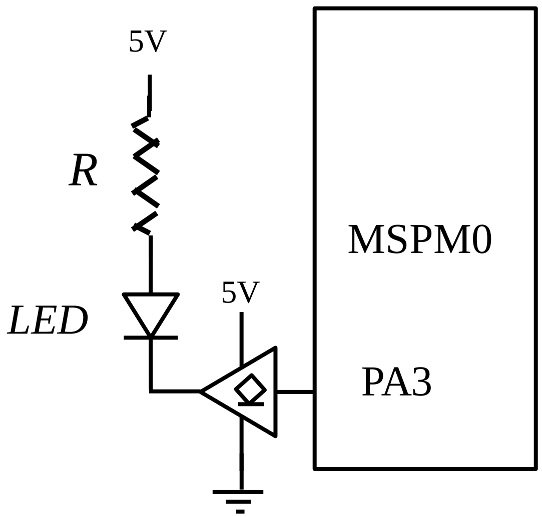
- Now suppose our LED needs more current (say 10 mA) and we want to use normal drive pin
- Make use of a so-called “hex buffer” logic gate
  - Consider 7407 chip
  - Used to connect logic level devices to higher voltage devices



- When A is low (0V), Y is low (0.5V) and can transmit current up to 40 mA
- When A is high (3.3V), Y is high (5V)
- Thus it performs logic $Y = A$

LED Circuit Example: High Current

- Create circuit using negative logic, incorporating 7407 IC



- When PA3 goes low, output of logic gate is low (0.5V) and current flows through LED
- When PA3 goes high, output of gate is high (5V) and no current flows

- Suppose the LED is designed to operate at 2 V, 10 mA
- Voltage drop across resistor is:

$$V_R = 5 - 2 - 0.5 = 2.5\,\text{V}$$

- To achieve current of 10 mA:

$$V_R = (10\ \text{mA})R$$

$$\rightarrow R = 2.5\,\text{V} / 10\ \text{mA} = 250\ \Omega \approx 220\Omega$$

Driverlib vs Direct Register Access

- So far we have discussed GPIO configuration using registers
  - This is known as "direct register access"
  - This is the way you configure most microcontrollers
- TI produces a software library for their MCU's called driverlib that abstracts this register manipulation
  - You call functions with readable names and arguments
  - ***Makes code more readable, but does not add any additional functionality or abstraction.***

Driverlib vs Direct Register Access

- Some examples:

```
void initialize_pins(){

  // Configure pin PA3 as output pin
  IOMUX->SECCFG.PINCM[IOMUX_PINCM8] =
                IOMUX_PINCM_PC_CONNECTED |
                0x00000001 ;

  // Set output high initially to turn LED off
  GPIOA->DOUT31_0 = 0x000000080 ;

  // Enable output
  GPIOA->DOE31_0 = 0x000000080 ;
}

void main(){

  int i = 0 ;
  initialize_pins() ;

  while(1){

    // Toggle LED
    GPIOA->DOUT31_0 ^= 0x000000080 ;

    // Delay
    for(i=0; i < 10000; i++){}
  }
}
```

```
void initialize_pins(){

  // Configure pin PA3 as output pin
  DL_GPIO_initDigitalOutput(IOMUX_PINCM8);

  // Set output high initially (LED off)
  DL_GPIO_setPins(GPIOA, DL_GPIO_PIN3);

  // Enable output
  DL_GPIO_enableOutput(GPIOA, DL_GPIO_PIN3);
}

void main(){

  int i = 0 ;
  initialize_pins() ;

  while(1){

    // Toggle LED
    DL_GPIO_togglePins(GPIOA, DL_GPIO_PIN3);

    // Delay
    for(i=0; i < 10000; i++){}
  }
}
```

Driverlib vs Direct Register Access

- Note that driverlib functions are just doing exact same bit manipulations in registers “under the covers”
  - Still need to understand the datasheet and register purposes
- All driverlib functions are documented in Driverlib Users Guide – please use accordingly
  - Available at this link
- You can use either Driverlib or direct register access in your codes for this class – your choice

Review Example Code

- Lecture 9 example code will:
  - Set up GPIO pins for toggling LEDs