ECEN 5823-001
Internet of Things Embedded Firmware

Lecture #5

11 September 2018

Be Boulder.

University of Colorado Boulder

Agenda

- Class Announcements
- Attributing IP
- Reading Assigned
- Quiz 2 review
- ESD diodes
- Load Power Management
- Sensors and Communication buses for the low energy application
- Bluetooth Classic

Class Announcements

- Quiz #3 is due at 11:59 on Sunday, September 16th, 2018
- Assignment #1: Managing Energy Modes is due, Saturday, September 15th, at 11:59pm via Canvas
- The ESE coding style document has been uploaded onto campus under the Course Material folder
  - Assignments and Projects will need to adhere to this coding style
  - Hint: A part of the grade will be based on using the coding style

Attributing IP

```
/***************************************************************************//**
 * @file sleep.c
 *******************************************************************************
 * @section License
 * <b>(C) Copyright 2015 Silicon Labs, http://www.silabs.com</b>
 *******************************************************************************
 *
 * Permission is granted to anyone to use this software for any purpose,
 * including commercial applications, and to alter it and redistribute it
 * freely, subject to the following restrictions:
 *
 * 1. The origin of this software must not be misrepresented; you must not
 *    claim that you wrote the original software.
 * 2. Altered source versions must be plainly marked as such, and must not be
 *    misrepresented as being the original software.
 * 3. This notice may not be removed or altered from any source distribution.
 *
 * DISCLAIMER OF WARRANTY/LIMITATION OF REMEDIES: Silicon Labs has no
 * obligation to support this Software. Silicon Labs is providing the
 * Software "AS IS", with no express or implied warranties of any kind,
 * including, but not limited to, any implied warranties of merchantability
 * or fitness for any particular purpose or warranties against infringement
 * of any proprietary rights of a third party.
 *
 * Silicon Labs will not be liable for any consequential, incidental, or
 * special damages, or for any other relief, or for any claim by any third party,
 * arising from your use of this Software.
 *
 ******************************************************************************/
```

Reading Assigned

Note: Quiz for week 3 will be from the readings below as well as the lectures since August 28th, 2018.

1. Texas Instruments application note SWRY007 – Three flavors of Bluetooth: Which one to choose?
   http://www.ti.com/lit/wp/swry007/swry007.pdf

2. Adafruit Learning System: Introduction to Bluetooth Low Energy
   https://learn.adafruit.com/downloads/pdf/introduction-to-bluetooth-low-energy.pdf

3. Bluetooth in Wireless Communications
   - Article can be found in the Canvas Reading assignment folder for week 3

4. TI application note SWRA349 - Coin cells and peak current draw
   http://www.ti.com/lit/wp/swra349/swra349.pdf

5. Silicon Labs' Blu2 Gecko Reference Manual – I2C section
   https://www.silabs.com/documents/public/reference-manuals/EFR32xG1-ReferenceManual.pdf

6. (Recommended for assignment) AN607: Si70XX HUMIDITY AND TEMPERATURE SENSOR DESIGNER 'S GUIDE
   https://www.silabs.com/Support%20Documents/TechnicalDocs/AN607.pdf

Reading Assigned

Recommended reading:

1. Silicon Labs' I2C application note - AN0011
   http://www.silabs.com/Support%20Documents/TechnicalDocs/AN0011.pdf

Quiz 2 Review

Complete the below C line of code to start LETIMER0 by writing directly to its register.

LETIMER0->CMD =

Quiz 2 Review

Which version of Bluetooth Smart introduced Forward Error Correction (FEC) to extend the range of the Bluetooth radio?

- [ ] 5.0
- [ ] 4.1
- [ ] 4.0
- [ ] 5
- [ ] 4.2

Quiz 2 Review

Using the below information from the Energy Profiler, order the devices from lowest average energy / current to the most. The below information is for a period, and all of the periods repeat indefinitely.

Period of repletion is 0.20S - In EM2 at 1.4uA for 0.198s - In EM2 at 87.0uA for 0.002s	[Choose] ▾
Period of repetition is 4s - In EM3 at 1uA for 3.995s - In EM0 at 3.1mA for 0.005s	[Choose] ▾
Period is indefinite - Continuous at 5uA	[Choose] ▾
Period of repetition is 3.5s - In EM3 at 1uA for 3.45s - In EM0 at 3.1mA for 0.005s - In EM1 at 1.4mA for 0.045s	[Choose] ▾

Quiz 2 Review

The Bluetooth 5 2 Mbps PHY can save energy due all the following. Select all that apply.

- [ ] Device can be in sleep mode longer
- [ ] Radio lower peak power
- [ ] Radio lower average power
- [ ] Radio on for a shorter period of time

Quiz 2 Review

Which clock sources can the LETIMER0 in the Blue Gecko use in EM3?

- LFXO
- LFRCO
- HFRCO
- ULFRCO
- HFXO

Quiz 2 Review

Which version of Bluetooth introduced ultra-low-power-consumption into the specifications?

- 4.1
- 5
- 5.2
- 4.0
- 1.0

Quiz 2 Review

Match the following sleep calls to the appropriate function or interrupt type handler

Enabling a peripheral	[Choose] ▾
Interrupt handler of a "single operation" peripheral	[Choose] ▾
Interrupt handler of a "re-occurring" peripheral	[Choose] ▾

Quiz 2 Review

Select all the C lines of code that would just disable the LETIMER0 underflow interrupt.

- [ ] LETIMER0->IEN = ~LETIMER_IEN_UF;
- [ ] LETIMER0_IEN = LETIMER0->IEN & ~LETIMER_IEN_UF;
- [ ] LETIMER_IntDisable(LETIMER0, LETIMER_IEN_UF);
- [ ] LETIMER0->IEN &= ~LETIMER_IEN_UF;

Quiz 2 Review

Write the C-code to remove the Vertical Front Porch Interrupt Enable, VFPORCH, of the External Bus Interface, EBI, peripheral interrupt enable register, IEN.

Quiz 2 Review

Complete the C line of code to clear the LETIMER0 UF interrupt bit.

```
[          ] = LETIMER_IFC_UF;
```

Quiz 2 Review

Bluetooth 5's 2 Mbps PHY enables transmission rates of Bluetooth 5 to go from 800 Kbps with the 1 Mbps PHY to ...

- 1.6 Mbps
- 0.80 Mbps
- 1.4 Mbps
- 2 Mbps

Quiz 2 Review

Match the action resulting from the signal type.

Specific action that is generated by the program.	[Choose] ▾
Actions outside the control of the process that receives them and may arrive at unpredictable times.	[Choose] ▾

Quiz 2 Review

Select all that are atomic in nature

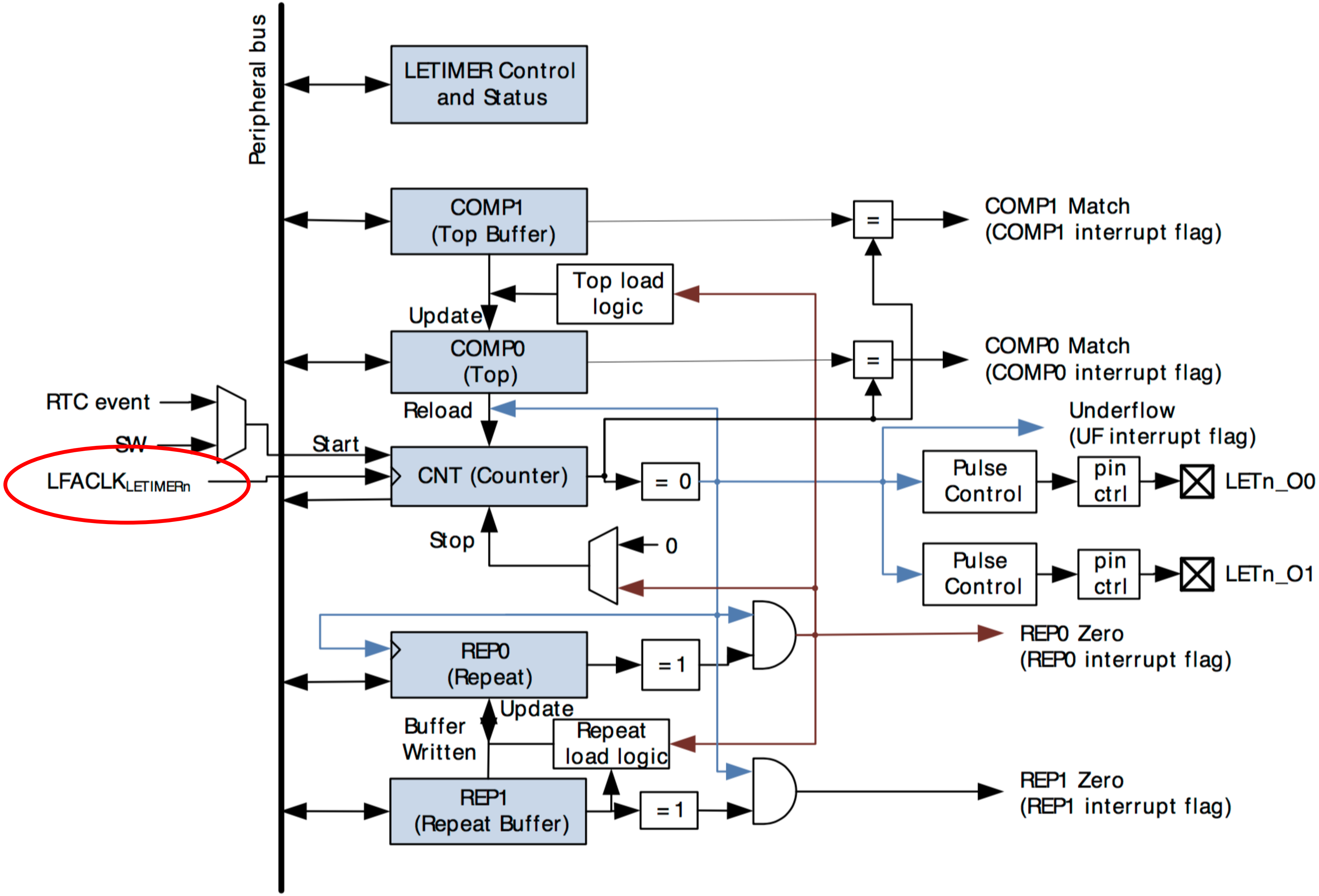
- ```
  for (i = 0; i < 1000; i++){

  }
  ```
- `ACD0->IEN |= ADC_IEN_SINGLE;`
- `i++;`
- ```
  sub r1, r2, r3
  ```

Quiz 2 Review

Match the following ...

Way to share data between a program and its event handler	[Choose] ▾
Data races	[Choose] ▾
A considerable amount of its computational is initiated and influenced by external events.	[Choose] ▾

Low Energy Timer



Setting up the LETIMER0

- What should we first do in setting up a peripheral?
- The clock tree to the LETIMER0 must be established
  - Why must we first set up the clock tree?
  - Without establishing the clock tree, all writes to the LETIMER0 registers will not occur
  - Pseudo code in the CMU setup routine to enable the LETIMER0 clock tree:
    - If using LFXO, enable the LFXO using the CMU_OscillatorEnable routine
    - Select the appropriate Low Frequency clock for the LFA clock tree depending on lowest energy mode for the LETIMER0
      - If EM0 – EM2, use CMU_ClockSelectSet to select the LFXO for LFA
      - If EM3, use the CMU_ClockSelectSet to select ULFRCO for LFA
    - Enable the Low Frequency clock tree by using the CMU_ClockEnable for CORELE
    - Lastly, enable the LFA clock tree to the LETIMER0 using the CMU_ClockEnable for the LETIMER0

Setting up the LETIMER0

- What should we do next in setting up a peripheral?
  - Define all variables in the LETIMER_Init_TypeDef to configure the LETIMER0 to perform as desired (disable the LETIMER0 at this time)
  - Then initialize the LETIMER0 using the LETIMER_Init command
  - If required, writing directly to the CMU->LFAPRESCO register, update the LFA prescaler
  - Program the COMP0 and COMP1 register with the values required to obtain the functionality desired using LETIMER_CompareSet command
  - Wait for the LETIMER0 synch bit is cleared before proceeding by accessing the register LETIMER0->SYNCBUSY

Setting up the LETIMER0

- After setting up the peripheral, what should we do next in setting up a peripheral?
- The LETIMER0 interrupts must be enabled
  - Clear all interrupts from the LETIMER0 to remove any interrupts that may have been set up inadvertently by accessing the LETIMER0->IFC register or the emlib routine
  - Enable the appropriate LETIMER0 interrupts by setting the appropriate bits in the LETIMER0->IEN register or using an emlib routine
  - Set the appropriate BlockSleep mode for this peripheral based on the system configuration such as going to EM3 or limiting to a higher EM level such as EM1 or EM2
  - Enable interrupts to the CPU by enabling the LETIMER0 in the Nested Vector Interrupt Control register using NVIC_EnableIRQ(LETIMER0_IRQn);

Setting up the LETIMER0

- With the peripheral interrupt now configured, what is the next step in setting up the LETIMER0?
- The LETIMER0 interrupt handler must be written
  - Routine name must match the vector table name:

    ```
    Void LETIMER0_IRQHandler(void) {
    }
    ```
  - Inside this routine, you add the functionality that is desired for the LETIMER0 interrupts
  - Note: Most timers are meant to repeat, so an unBlockSleep call most likely will not be needed in the LETIMER0 interrupt handler

Setting up the LETIMER0

- There is one more step in setting up the LETIMER0. What is this step?
- Lastly, enable the LETIMER0 when it is desired to have the peripheral to start operation
  - Can enable the LETIMER0 by writing directly to the LETIMER0 or using the emlib routin LETIMER_Enable(LETIMER0, true);

LETIMER0 Compare Registers

- The LETIMER has two compare match registers, LETIMERn_COMP0 and LETIMERn_COMP1
  - Each of these compare registers are capable of generating an interrupt when the counter value LETIMERn_CNT becomes equal to their value.
  - When LETIMERn_CNT becomes equal to the value of LETIMERn_COMP0, the interrupt flag COMP0 in LETIMERn_IF is set, and when LETIMERn_CNT becomes equal to the value of LETIMERn_COMP1, the interrupt flag COMP1 in LETIMERn_IF is set.
    - Setting the correct count value with the known period of the clock used by LETIMER, the period of the LETIMER can be divided into an On Duty Cycle and an Off Duty Cycle

LETIMER0 Top Value

- If COMP0TOP in LETIMERn_CTRL is set, the value of LETIMERn_COMP0 acts as the top value of the timer, and LETIMERn_COMP0 is loaded into LETIMERn_CNT on timer underflow.
  - A specific period of the LETIMER0 can be set by COMP0TOP being set and the correct count value programmed into COMP0 if the clock period is known for the LETIMER
- Else, the timer wraps around to 0xFFFF. The underflow interrupt flag UF in LETIMERn_IF is set when the timer reaches zero
  - The period with COMP0TOP is defined by 0xFFFF * the clock period used for LETIMER

LETIMER Buffered Top Value

- If BUFTOP in LETIMERn_CTRL is set, the value of LETIMERn_COMP0 is buffered by LETIMERn_COMP1
- In this mode, the value of LETIMERn_COMP1 is loaded into LETIMERn_COMP0 every time LETIMERn_REP0 is about to decrement to 0
- This can for instance be used in conjunction with the buffered repeat mode to generate continually changing output waveforms
- Write operations to LETIMERn_COMP0 have priority over buffer loads

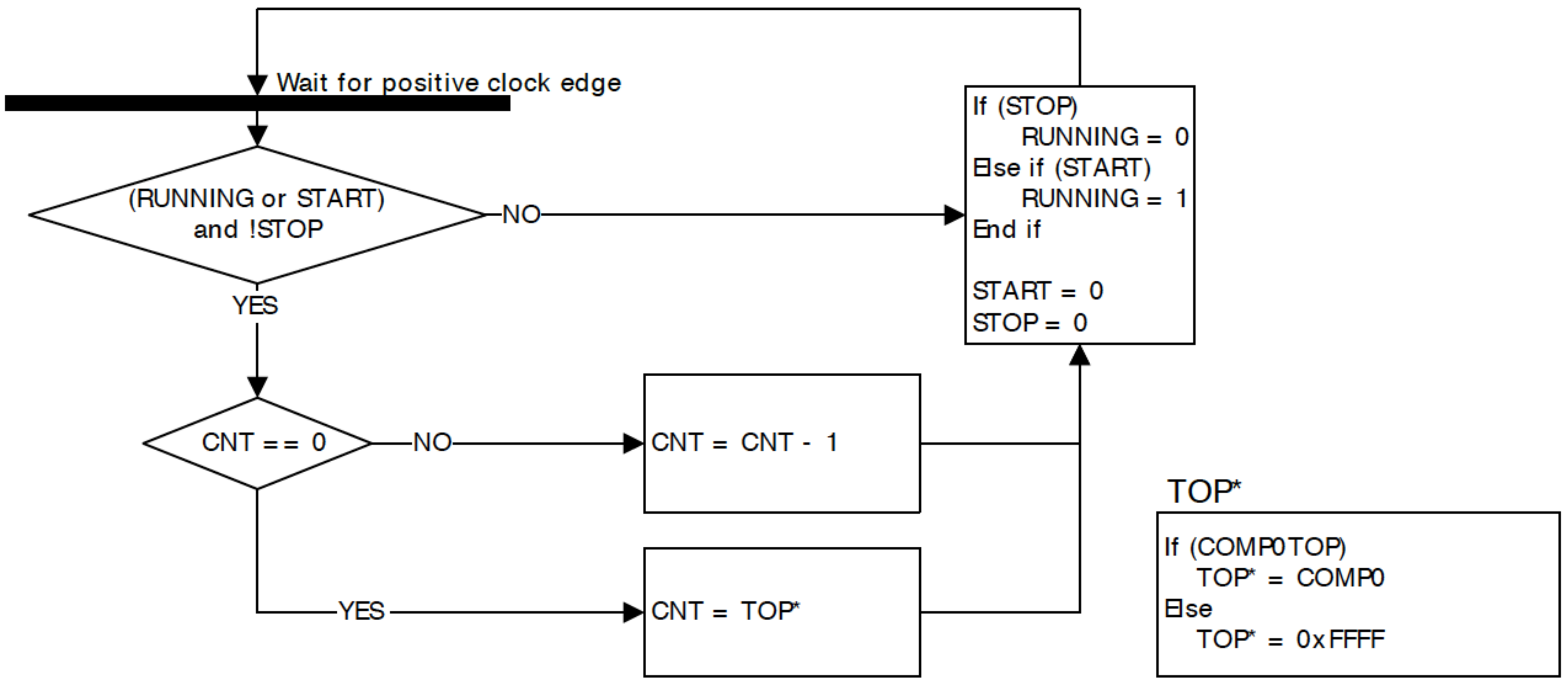
LETIMER Repeat Modes

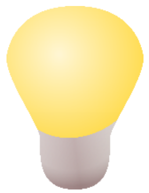
Table 2.1. LETIMER Repeat Modes

REPMODE	Mode	Description
00	Free	The timer runs until it is stopped
01	One-shot	The timer runs as long as LETIMERn_REP0 != 0. LETIMERn_REP0 is decremented at each timer underflow.
10	Buffered	The timer runs as long as LETIMERn_REP0 != 0. LETIMERn_REP0 is decremented on each timer underflow. If LETIMERn_REP1 has been written, it is loaded into LETIMERn_REP0 when LETIMERn_REP0 is about to be decremented to 0.
11	Double	The timer runs as long as LETIMERn_REP0 != 0 or LETIMERn_REP1 != 0. Both LETIMERn_REP0 and LETIMERn_REP1 are decremented at each timer underflow.

Free Running flow diagram

Figure 23.2. LETIMER State Machine for Free-running Mode



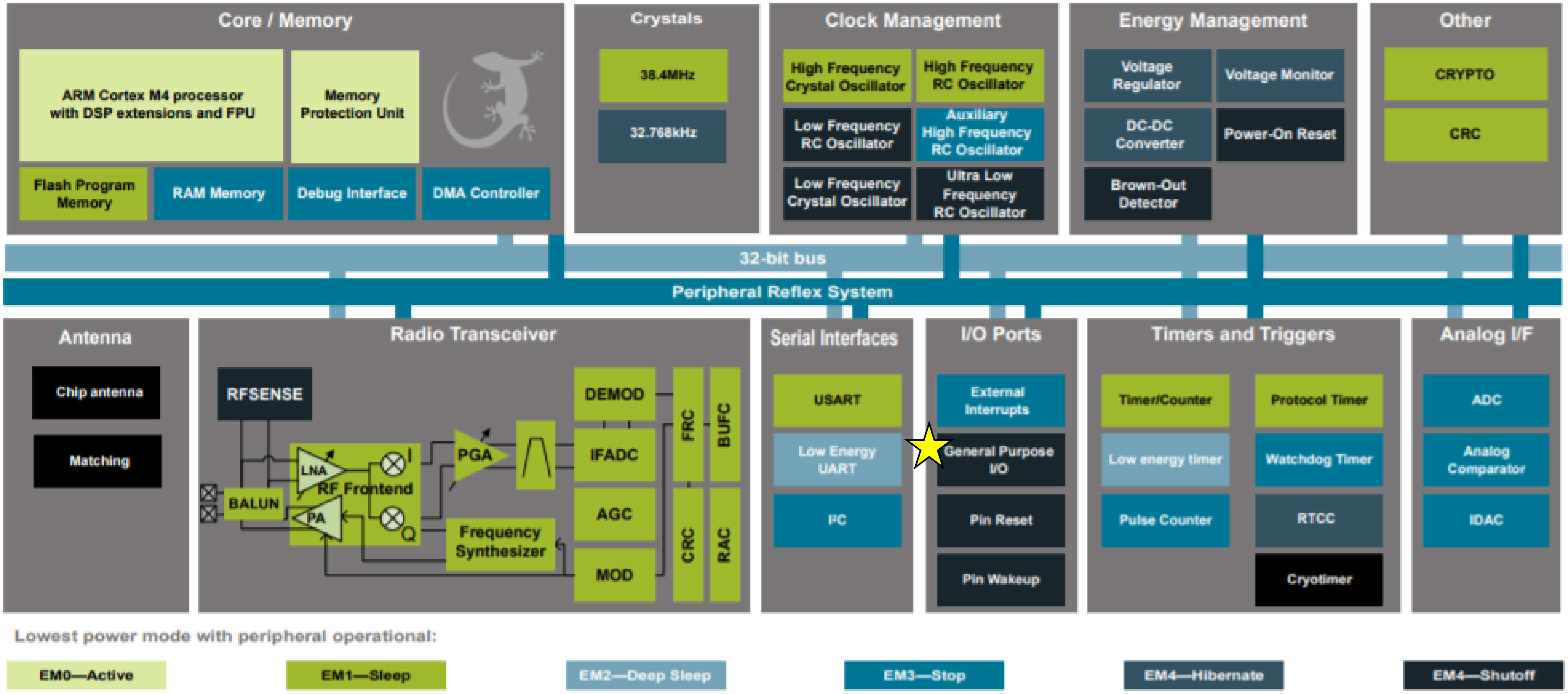


LETIMER interrupt emlib routine examples

- There are 5 interrupts available for LETIMER0
  - REP0, REP1, COMP0, COMP1, and UL
- emlib routine to enable interrupts
  - LETIMER_IntEnable(LETIMER_TypeDef *letimer, unit32_flags);
- emlib routine to disable interrupts
  - LETIMER_IntDisable(LETIMER_TypeDef *letimer, unit32_flags);
- emlib routine to clear interrupts
  - LETIMER_IntClear(LETIMER_TypeDef *letimer, unit32_flags);
- example

```
__STATIC_INLINE void LETIMER_IntEnable(LETIMER_TypeDef *letimer, uint32_t flags)
{
  letimer->IEN |= flags;
}
```

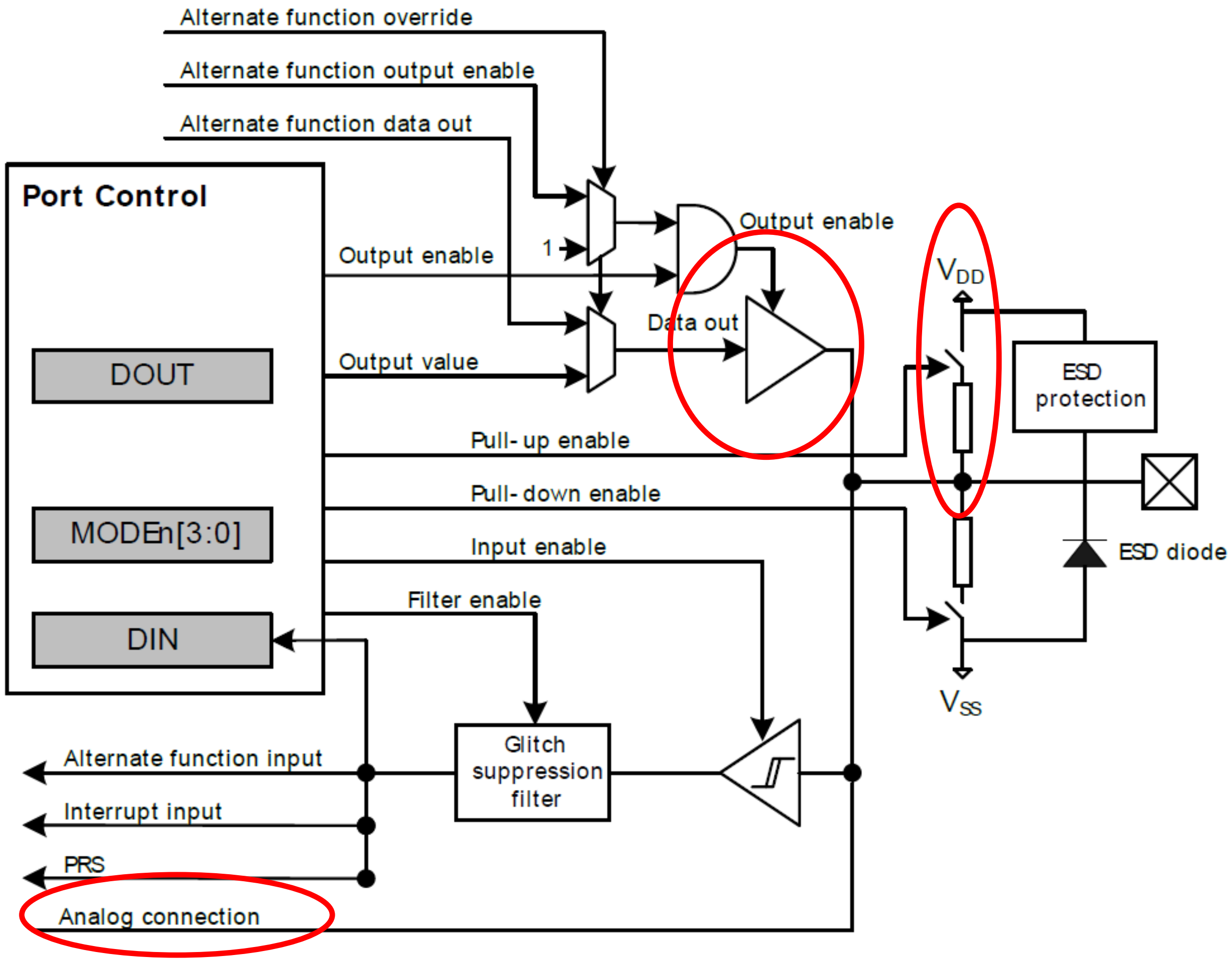
LETIMER0 will be used for LETIMER_TypeDef for the Blue Gecko



Core / Memory
ARM Cortex M4 processor with DSP extensions and FPU
Memory Protection Unit
Flash Program Memory
RAM Memory
Debug Interface
DMA Controller
Crystals
38.4MHz
32.768kHz
Clock Management
High Frequency Crystal Oscillator
High Frequency RC Oscillator
Low Frequency RC Oscillator
Auxiliary High Frequency RC Oscillator
Low Frequency Crystal Oscillator
Ultra Low Frequency RC Oscillator
Energy Management
Voltage Regulator
Voltage Monitor
DC-DC Converter
Power-On Reset
Brown-Out Detector
Other
CRYPTO
CRC
32-bit bus
Peripheral Reflex System
Antenna
Chip antenna
Matching
Radio Transceiver
RFSENSE
LNA
RF Frontend
BALUN
PA
I
Q
PGA
Frequency Synthesizer
DEMOD
IFADC
AGC
MOD
FRC
BUFC
CRC
RAC
Serial Interfaces
USART
Low Energy UART
I²C
I/O Ports
External Interrupts
General Purpose I/O
Pin Reset
Pin Wakeup
Timers and Triggers
Timer/Counter
Protocol Timer
Low energy timer
Watchdog Timer
Pulse Counter
RTCC
Cryotimer
Analog I/F
ADC
Analog Comparator
IDAC
Lowest power mode with peripheral operational:
EM0—Active
EM1—Sleep
EM2—Deep Sleep
EM3—Stop
EM4—Hibernate
EM4—Shutoff

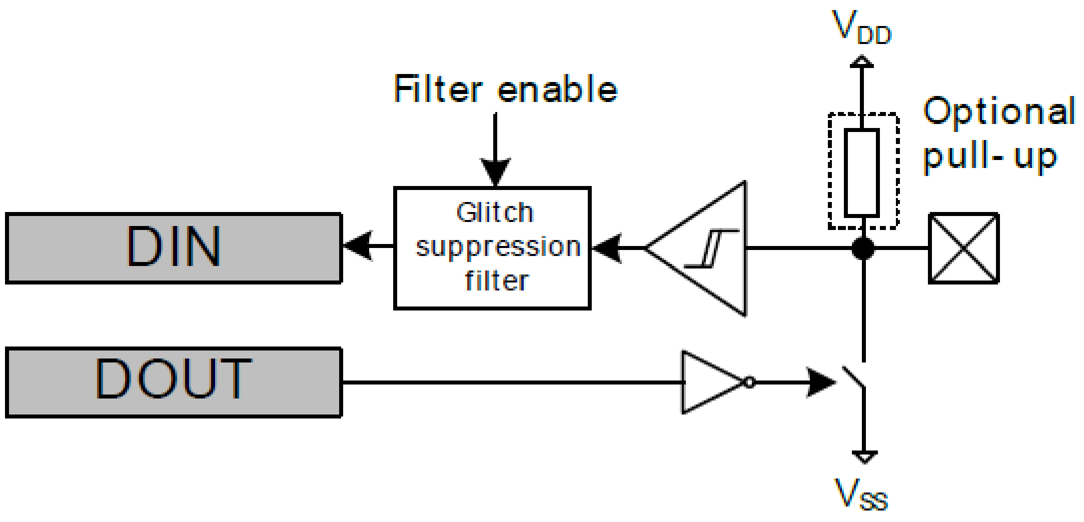
GPIO Peripheral

- Individual configuration for each pin
  - Tristate (reset state)
  - Push-pull
  - Open-drain
  - Pull-up resistor
  - Pull-down resistor
  - Four drive strength modes
    - HIGH
    - STANDARD
    - LOW
    - LOWEST



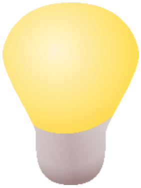
GPIO – Open drain

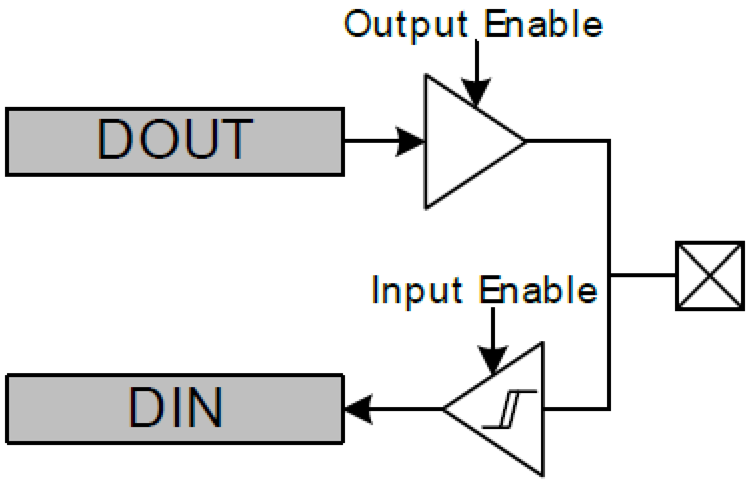
- What is a common bus that utilizes open drain ports?
  - Common configuration when multiple sources may drive the bus
  - I2C



GPIO – Push/Pull configuration

- When would you want to use a push/pull port?
- Common Output or input configuration when a single source is driving the net
- In output mode, it can be used to:
  - Source current to an LED by driving the positive node of the LED
  - Or, sink current from an LED by connecting to ground the negative/ground node of the LED





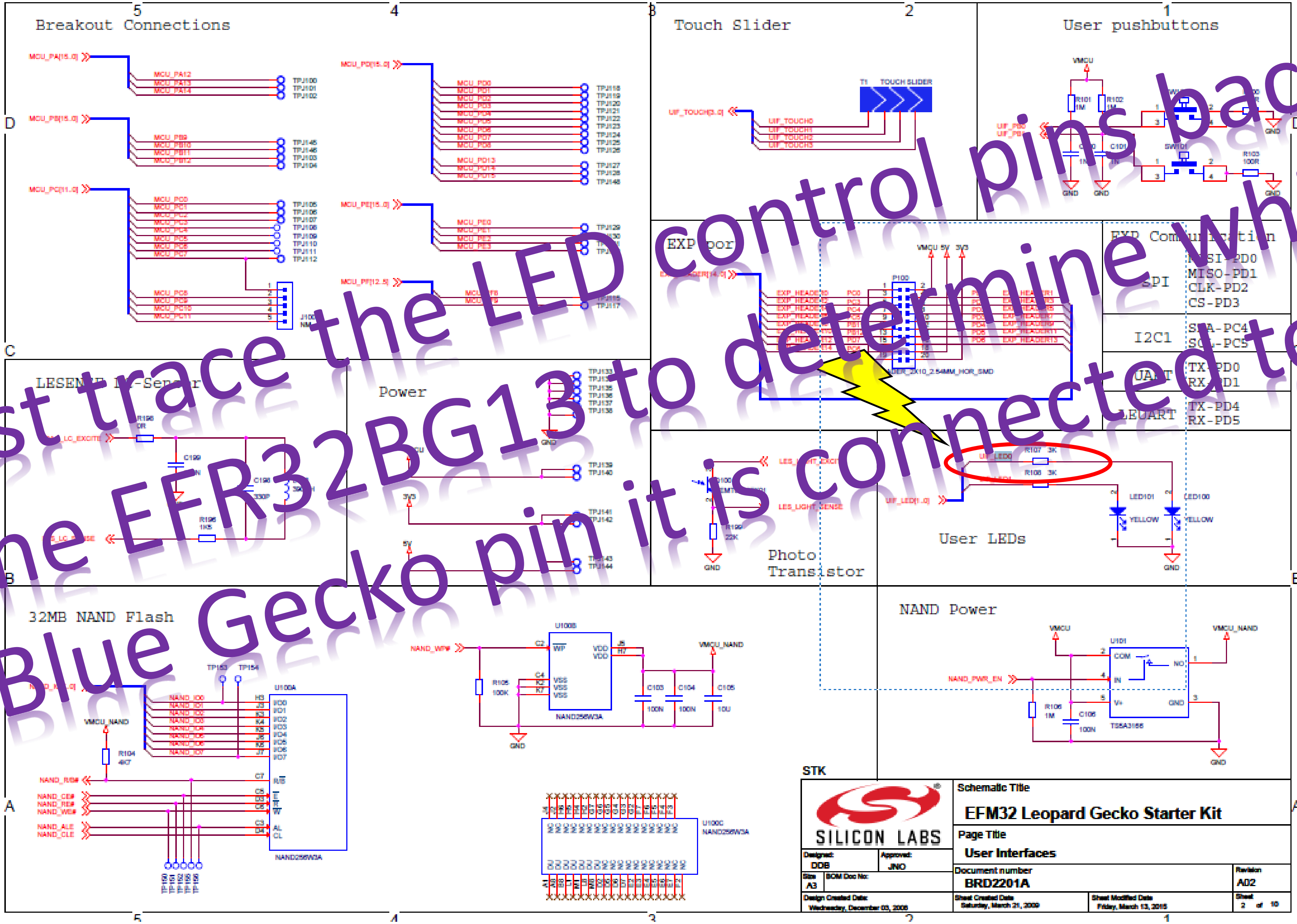
Setting up the GPIO

- What should we first do in setting up a peripheral?
- The clock tree to the GPIO must be established
  - Without establishing the clock tree, all writes to the GPIO registers will not occur
  - Pseudo code in the CMU setup routine to enable the GPIO clock tree:
    - Lastly, enable the GPIO clocking using the CMU_ClockEnable for the GPIO

Setting up the GPIO

- After the GPIO clock tree has been established, what is the next step?
- The GPIO peripheral must be set up
  - Specifying the pins
      - Both the Port and Pin # is required
      - For the Blue Gecko, use the schematic from Simplicity
    - What is the function of the pins?
      - Push-Pull
      - Open drain
      - Etc.
    - Program the functionality of the GPIO pin using GPIO_PinModeSet
    - Program drive strength of the GPIO pin using GPIO_DriveModeSet

Must trace the LED control pins back to the EFR32BG13 to determine which Blue Gecko pin it is connected to



Setting up the GPIO

- After the GPIO peripheral has been configured, what is the next step?
- The GPIO interrupts must be enabled if needed
  - Clear all interrupts from the GPIO to remove any interrupts that may have been set up inadvertently by accessing the GPIO->IFC register or the emlib routine
  - GPIO_IntConfig emlib command to set GPIO interrupts
  - Enable the appropriate GPIO interrupts by setting the appropriate bits in the GPIO->IEN register or using an emlib routine
    - There are two interrupt vectors (handlers) for the GPIO
      - Even GPIO pins
      - Odd GPIO pins
  - No need to set BlockSleep mode since GPIO pins work EM0 thru EM3
  - A subset of GPIO pins works down to EM4
  - Enable interrupts to the CPU by enabling the GPIO in the Nested Vector Interrupt Control register using NVIC_EnableIRQ(GPIO_EVEN_IRQn); or NVIC_EnableIRQ(GPIO_ODD_IRQn);

Setting up the GPIO

- After the GPIO interrupts have been configured, what is the next step?
- The GPIO interrupt service handler must be included
  - Routine name must match the vector table name:

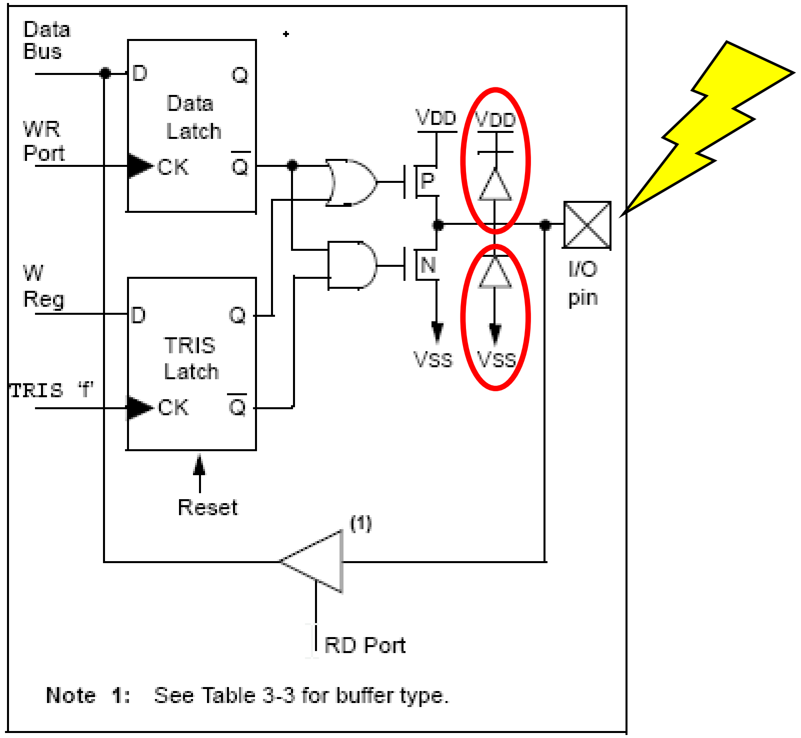
    Void GPIO_EVEN_IRQHandler(void) {
    }
    Or
    Void GPIO_ODD_IRQHandler(void) {
    }
  - Inside this routine, you add the functionality that is desired for the GPIO interrupts

GPIO – Input / Output

- To read a GPIO pin, direct register access can be used or emlib routine GPIO_PinInGet
- Setting a GPIO pin output:
    - Programming a one or “high” with GPIO_PinOutSet
    - Programming a zero or “low” with GPIO_PinOutClear

Why an ESD diode to protect the I/O pin?

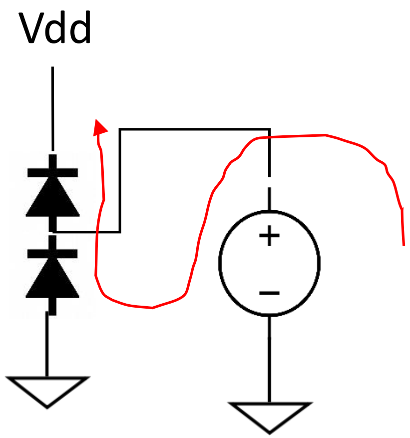
FIGURE 5-1: PIC12F508/509/16F505 EQUIVALENT CIRCUIT FOR A SINGLE I/O PIN



Note 1: See Table 3-3 for buffer type.

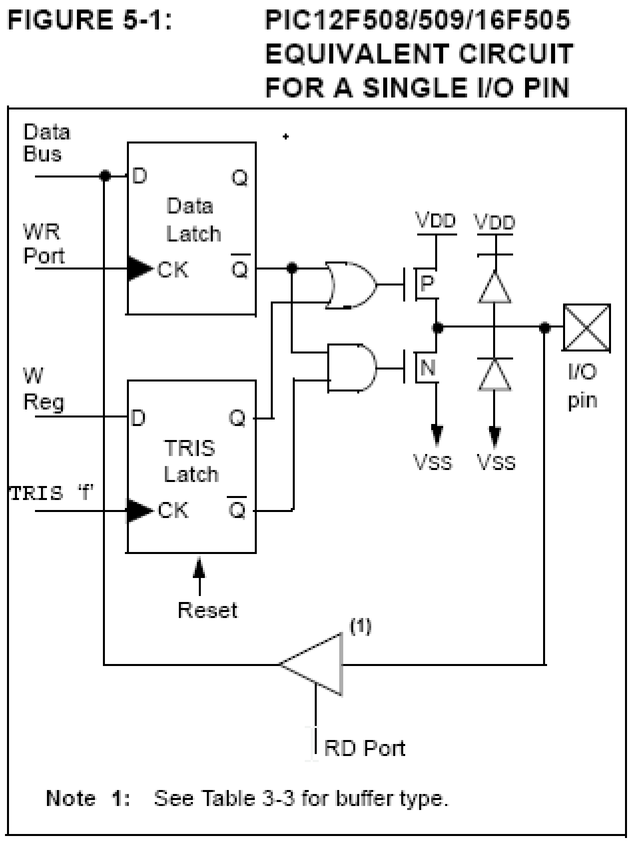
Normal Operation

- Vdd = 3.3v and Vss = 0.0v
- An Electro Static Discharge event occurs



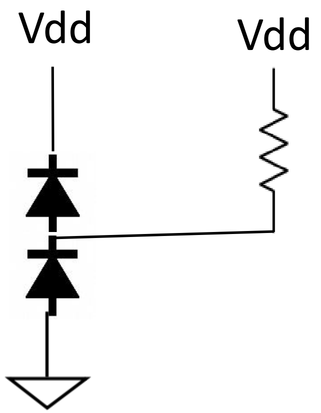
- The ESD event is much greater voltage than Vdd
- Current will flow from the ESD event through the top ESD diode
- This diode clamps the voltage to the IC at Vcc + Vdiode
- Protecting the IC

Modeling an IC that is connected to an I2C device



IC Vdd is turned on

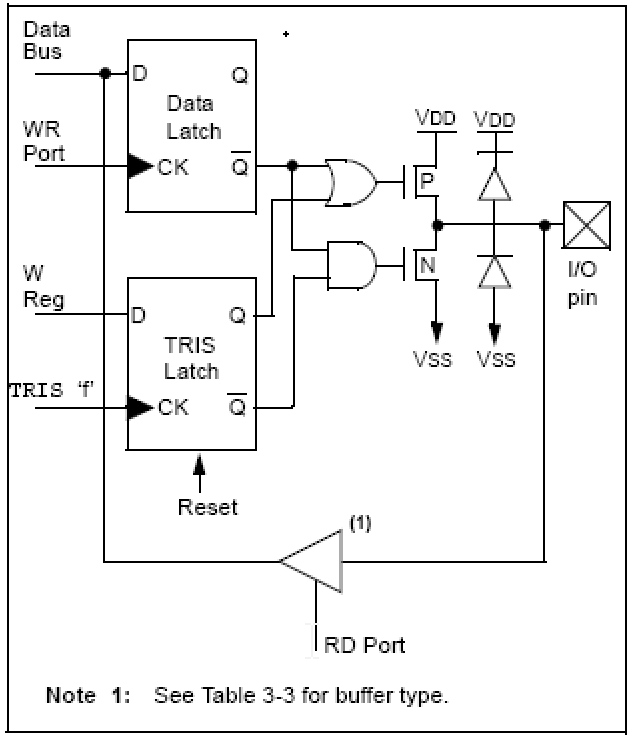
- Vdd = 3.3v and Vss = 0.0v



- If the I/O pin is not pulling the I/O low, the pull-up resistor will pull the I2C line high, to Vdd = 3.3v

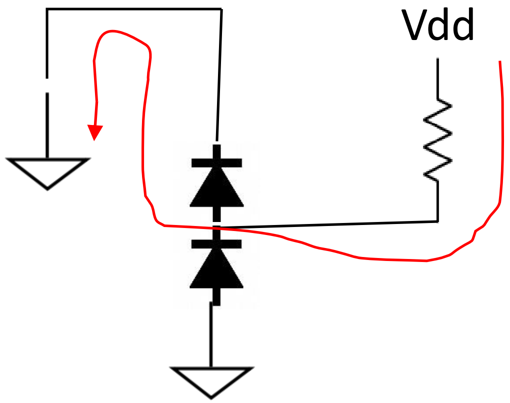
What happens when just the IC's Vdd is turned off?

FIGURE 5-1: PIC12F508/509/16F505 EQUIVALENT CIRCUIT FOR A SINGLE I/O PIN



IC Vdd is turned off

- Vdd = 0.0v and Vss = 0.0v

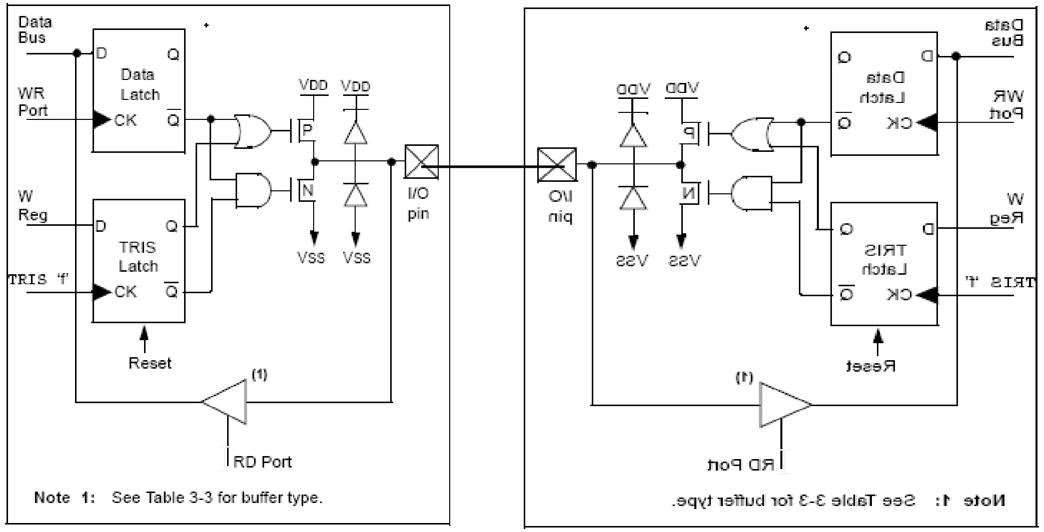


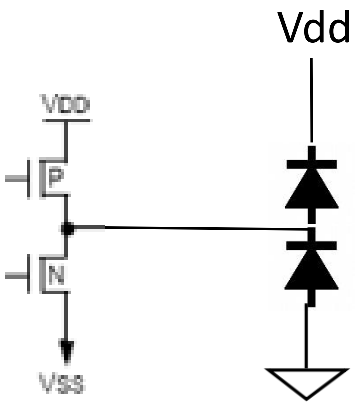
- When Vdd is 0.0v, the I2C signal is continuous pulled to ground through the upper ESD diode
- I2C voltage is now continuously equal to 0 + Vdiode
- I2C bus is now not operational
- And, each I2C line is pulling current equal to (Vdd – Vdiode) / Rpull-up
- This continuous current can damage the I/O pin

Modeling an IC with two standard I/Os

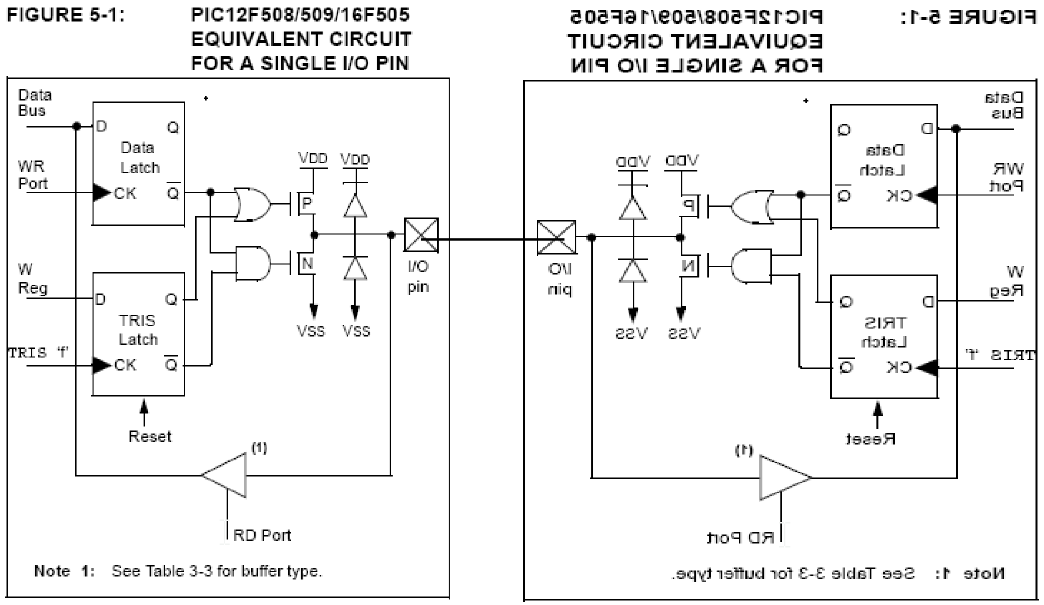
- Normal Operation
  - Left IC output: Vdd = 3.3v and Vss = 0.0v
  - Right IC input: Vdd = 3.3v and Vss = 0.0v
  - Output can drive 6mA

FIGURE 5-1: PIC12F508/509/16F505 EQUIVALENT CIRCUIT FOR A SINGLE I/O PIN



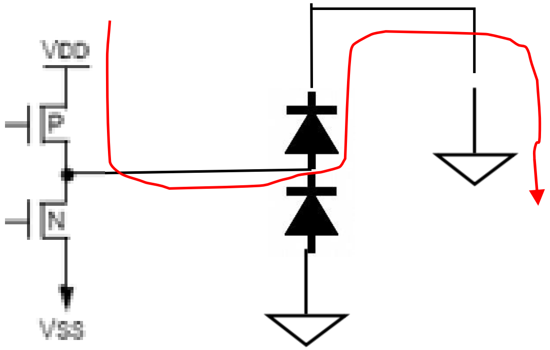


Modeling an IC when 1 IC is turned off



1 IC is turned off

- Left IC output: Vdd = 3.3v and Vss = 0.0v
- Right IC input: Vdd = 0.0v and Vss = 0.0v
- Output can drive 6mA



- When left IC wants to drive the output high, the left IC Vdd drives current through its P-channel FET and the right input pin Vdd ESD diode
- Instead of a high output, the output goes to 0v + Vdiode
- The current through the diode could equal the drive strength of the output, 6mA
- Possibly damaging the IC

Microchip PICF508 family equivalent I/O circuit

Load Power Management

- What is it?
  - Turning off a peripheral when not needed to save energy
  - Common technique used in notebooks, computers, embedded systems, and battery powered products
  - You are already doing it!!!!
    - By not turn on peripherals that are not in use
    - And, by disabling the ACMP0 when not required
    - And, by disabling the ADC0 when not in use

Load Power Management

- Now, lets take a look at load power management of a non-MCU peripheral
- Basic steps include the following:
  - Enable power to the device
    - Via GPIO control instead of CMU_ClockEnable()
    - It will take some time for the GPIO power pin to stabilize
  - Wait for external device to complete its Power On Reset (POR)
  - Initialize the device
  - Enable Interrupts if will be used

Sensor examples

- 3-axis accelerometer
  - SparkFun Triple Axis Accelerometer Breakout - MMA8452Q

MMA8452Q

Features

- 1.95V to 3.6V supply voltage
- 1.6V to 3.6V interface voltage
- ±2g/±4g/±8g dynamically selectable full-scale
- Output Data Rates (ODR) from 1.56 Hz to 800 Hz
- 99 μg/√Hz noise
- 12-bit and 8-bit digital output
- I^2C digital output interface
- Two programmable interrupt pins for six interrupt sources
- Three embedded channels of motion detection
  - — Freefall or Motion Detection: 1 channel
  - — Pulse Detection: 1 channel
  - — Transient Detection: 1 channel
    - – Orientation (Portrait/Landscape) detection with set hysteresis
    - – Automatic ODR change for Auto-WAKE and return to SLEEP
    - – High-Pass Filter Data available real-time
    - – Self-Test
    - – RoHS compliant
    - – Current Consumption: 6 μA to 165 μA

- Separate power for I2C and digital logic
- Enabling ease or Load Power Management via GPIO pin

Photo from the Sparkfun.com and the MMA8452Q features from its NXP datasheet

Load Power Management via GPIO pin

- For the MMA8452Q, any of the gpio Drive Mode settings should be sufficient
  - To insure that the Vdd to the external IC can support the transients required by the IC, the GPIO Power pin should be decoupled at the IC
  - The power setting of the gpio power pin should be set high enough to drive the capacitive load in a reasonable time to power up the IC in the time required for the application

enum GPIO_DriveStrength_TypeDef

GPIO drive strength.

Enumerator	
gpioDriveStrengthWeakAlternateWeak	GPIO weak 1mA and alternate function weak 1mA
gpioDriveStrengthWeakAlternateStrong	GPIO weak 1mA and alternate function strong 10mA
gpioDriveStrengthStrongAlternateWeak	GPIO strong 10mA and alternate function weak 1mA
gpioDriveStrengthStrongAlternateStrong	GPIO strong 10mA and alternate function strong 10mA

Load Power Management via GPIO pin

Setting up LPM via GPIO pin

1. Connect GPIO pin from output pin to Vdd of external peripheral
2. Add appropriate decoupling capacitors
   a. Refer to the external peripheral IC recommended decoupling capacitors
3. Configure the GPIO output to be a Push-Pull output
   a. Set the default output setting to 0, turned off

Figure 32.1. Pin Configuration