

ECEN 5823-001 / -001B

Internet of Things Embedded Firmware

Lecture #7

19 September 2017

Agenda

- Class announcements
- Reading assigned / Quiz 4 assigned
- Quiz 3 review
- ESD diodes
- Load Power Management
- BMA280 – Accelerometer / Tap Sensor
- TIMERN peripheral
- Interrupt Priority

Class Announcements

- Quiz #4 is due at 11:59pm on Sunday, September 24th, 2017
- Low Energy ADC Assignment is due at 11:59pm on Wednesday, September 20th, 2017

SWE CU Boulder Presents: FALL NETWORKING NIGHT



Monday, September 25th from 6:30-8:30 pm.
Student Check-in begins at 6:00.

@ Discovery Learning Center Collaboratory

All STEM majors/backgrounds welcome!

Companies Attending: CH2M, Medtronic, Google, Seagate, Harris Corporation, Emerson, First RF, NetApp, Arrow Electronics, Sandia National Laboratories, Ball Aerospace, Intel, Xcel Energy, Northrup Grumman, and more!

Registration: *Please register online at: <https://www.eventbrite.com/e/swe-networking-night-2017-tickets-37787619788> (Online: \$7, At Door: \$10)*

Registration will cover catered dinner. NATIONAL SWE MEMBERS: email Vanika Hans at cuswevp@gmail.com for discount PROMO code.

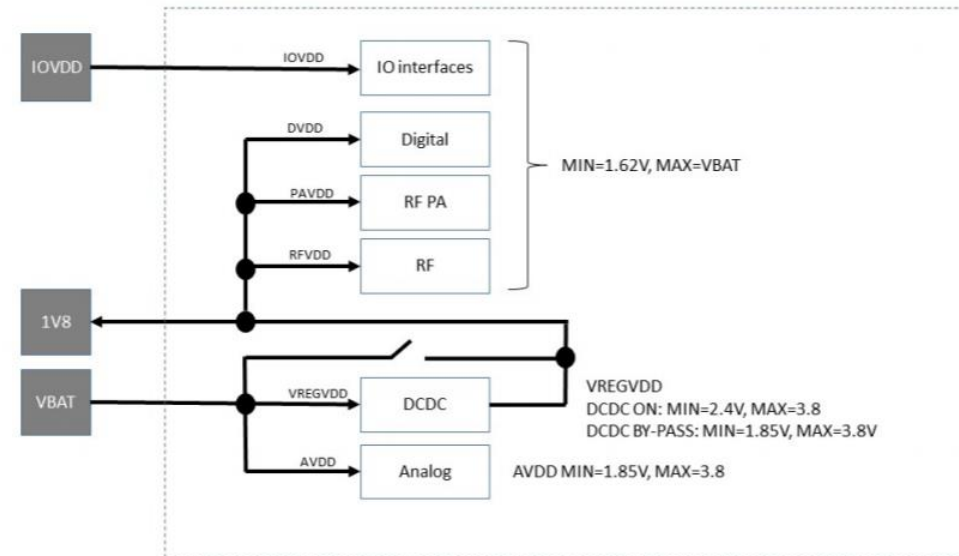
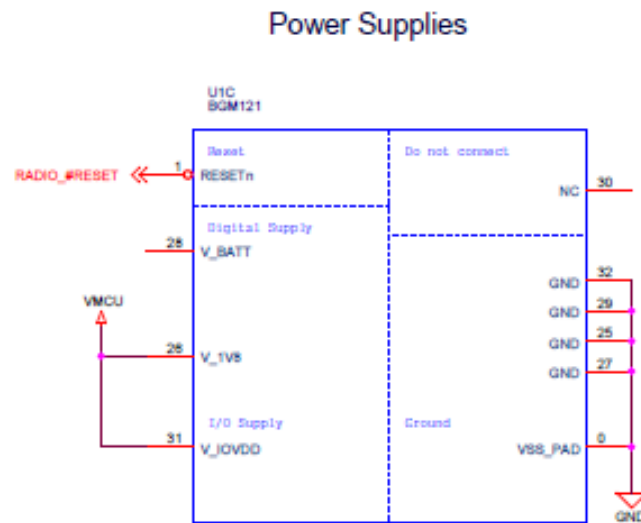
Come network with company representatives, learn about industry, and make connections for internships and job opportunities! Business casual.

Blue Gecko's ADC – voltage reference

- $V_{FS} = 1.25\text{ V}$ using internal VBGR as the reference source
- $V_{FS} = 2.5\text{ V}$ using internal VBGR as the reference source
- $V_{FS} = AVDD$ using AVDD as the reference source ($AVDD \leq 3.6\text{ V}$)
- $V_{FS} = 5\text{ V}$ using internal VBGR as the reference source
- $V_{FS} = \text{ADCn_EXTP}$ external pin as a single-ended reference source (1.2 V - 3.6 V)
- $V_{FS} = \text{ADCn_EXTP} - \text{ADCn_EXTN}$ external pins as a differential reference source. (1.2 V - 3.6 V difference)
- $V_{FS} = 2 \times AVDD$ using AVDD as the reference source ($AVDD \leq 3.6\text{ V}$)
- The maximum and minimum input voltage which the ADC can recognize at any external pin

Blue Gecko's ADC – voltage reference

- Two possible ADC references to measure a 3.3v input voltage using the full range of the ADC



- First: $V_{FS} = AVDD$ using $AVDD$ as the reference source ($AVDD \leq 3.6 \text{ V}$)

Blue Gecko's ADC – voltage reference

- Two possible ADC references to measure a 3.3v input voltage using the full range of the ADC
 - VBGR: An internal 0.83 V bandgap reference voltage
 - $VFS = 2 \times VREF \times ATTVREF / ATTVIN$
 - The VFS must allow a full 3.3v input sampling through a combination of attenuating the input and attenuating the reference

Blue Gecko's ADC – voltage reference

11:8	VINATT	0x0	RW	Code for VIN attenuation factor. Used to set the VIN attenuation factor.
7:4	VREFATT	0x0	RW	Code for VREF attenuation factor when VREFSEL is 1, 2 or 5 Used to set VREF attenuation factor.
3	VREFATTFIX	0	RW	Enable fixed scaling on VREF Enables fixed scaling on VREF
		Value	Description	
		0	VREFATT setting is used to scale VREF when VREFSEL is 1, 2 or 5.	
		1	A fixed VREF attenuation is used to cover a large reference source range. When VREFATT = 0, the scaling factor is 1/4. For non-zero values of VREFATT, the scaling factor is 1/3.	
2:0	VREFSEL	0x0	RW	Single Channel Reference Selection Select reference VREF to ADC single channel mode.
		Value	Mode	Description
		0	VBGR	Internal 0.83V Bandgap reference
		1	VDDXWATT	Scaled AVDD: $AVDD \times (\text{the VREF attenuation factor})$
		2	VREFPWATT	Scaled singled ended external Vref: $ADCn_EXTP \times (\text{the VREF attenuation factor})$
		3	VREFP	Raw single ended external Vref: $ADCn_EXTP$
		4	VENTROPY	Special mode used to generate ENTROPY.
		5	VREFPNWATT	Scaled differential external Vref from : $(ADCn_EXTP - ADCn_EXTN) \times (\text{the VREF attenuation factor})$
		6	VREFPN	Raw differential external Vref from : $(ADCn_EXTP - ADCn_EXTN)$
		7	VBGRLOW	Internal Bandgap reference at low setting 0.78V



Blue Gecko's ADC – voltage reference

- Two possible ADC references to measure a 3.3v input voltage using the full range of the ADC
 - VBGR: An internal 0.83 V bandgap reference voltage
 - $VFS = 2 \times VREF \times \text{ATT}VREF / \text{ATT}VIN$, due to using VBGR
 - Now VFS will equal 2 x 0.83 volts or 1.66v
 - The input attenuation must be 1/2 to scale the 3.3v input to within the range of VFS, 1.66v
 - $ATTVIN = VINATT / 12$ for $VINATT \geq 3$ (settings 0, 1, and 2 are not allowable values for VINATT)
 - VINATT must be 6 for ATTVIN of 1/2

Quiz 3 review

In a typical Thread network, which address is not a sleepy end device?

☐ 0x0C40

☐ 0xFA01

☒ 0x0800

☐ 0xAA00

Quiz 3 review

There are a number of variables that impact the actual power consumption during the sleep end point wake sleep cycle. Select all that are implementation specific.

☒ Time for the device to wake from deep sleep and be ready with the radio

☒ Active currents during the various operations

☐ Time waiting for the ACK packet

☐ Time on the air transmitting (size of packet)

Quiz 3 review

Based on the Blue Gecko data sheet and reference manual, which energy mode would be specified for the ACMP peripheral while operational and is measuring its input by a call to the blockSleepMode() routine?

☐ EM0

☐ EM1

☐ EM2

☒ EM3

☐ EM4

Quiz 3 review

For the Blue Gecko STK, which instruction would be used to configure the input of the analog joy stick?

- ☐ GPIO_PinModeSet(gpioPortA, 0, gpioModePushPull, 0);
- ☐ GPIO_PinModeSet(gpioPortA, 0, gpioModeInput , 1);
- ☐ GPIO_PinModeSet(gpioPortF, 4, gpioModePushPull, 0);
- ☒ GPIO_PinModeSet(gpioPortA, 0, gpioModeDisabled, 0);

Quiz 3 review

To disable Tailgate on the Analog to Digital Converter of the Blue Gecko peripheral, complete the following c line of code. Use best coding practices.

ADC0->CTRL &=



Quiz 3 review



Complete the following line of c code for the Blue Gecko to read the status of the Blue Gecko Analog to Digital Converter;

```
int ADCStatus;
```

```
ADCStatus = (ADC0->STATUS;, ADC0 -> STATUS;, ADC0-> STATUS;, ADC0 ->STATUS;)
```





Quiz 3 review

If the maximum count of the LETIMER0 is 65,536, what would the minimum LETIMER0 LFXO prescaler need to be to enable the LETIMER0 to count to 48 seconds based on the LFXO set to the frequency of 32,768 to interrupt on the underflow condition only when 48 seconds have past.  What count would be required to be stored in the LETIMER0->CNT register to equal 48 seconds based on the above prescaler which is indicated upon an underflow event? 

Quiz 3 review

While debugging the following routine, it was determined a line of code is incorrect. Using the line numbers to the left, which line of code is incorrect?

```
1. void ADC0_UpdateCtrl(void) {  
2.  
3.   if (ADC->STATUS) {  
4.  
5.     ADC0->IEN = ADC0->IEN | ADC_IEN_SINGLE; }  
6.  
7.   else {  
8.  
9.     ADC0->CTRL &= ~ADC_CTRL_SINGLEDMAWU; }  
10.  
11. }
```

 Write the proper c-code instruction of the incorrect line of code? (do not add the line item to the c-code instruction. The numbers are for reference only.) 

Reading Assignment

Below is a list of required reading for this course. Questions from these readings plus the lectures from August 28th onward can be on the weekly quiz.

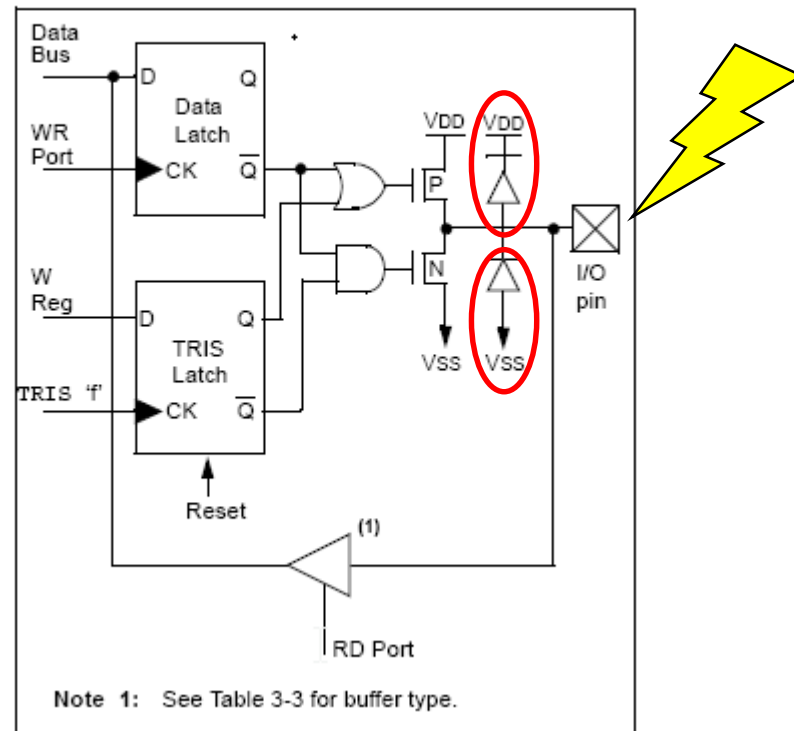
1. Circuit Cellar: Electronic Compass: Tilt Compensation & Calibration
http://cache.nxp.com/files/sensors/doc/reports_presentations/ARTICLE_REPRINT.pdf
2. AN607: Si70XX HUMIDITY AND TEMPERATURE SENSOR DESIGNER 'S GUIDE
<https://www.silabs.com/Support%20Documents/TechnicalDocs/AN607.pdf>
3. AN580: Infrared Gesture Sensing
<https://www.silabs.com/Support%20Documents/TechnicalDocs/AN580.pdf>

Recommended readings. These readings will not be on the weekly quiz, but will be helpful in the class programming assignments and course project.

4. AN0008: Silicon Lab's USART / SPI application note
 - a. May be found in the course D2L folder Week 3 / Reading Assignment

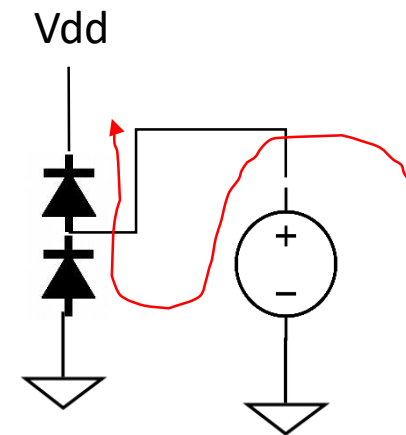
Why an ESD diode to protect the I/O pin?

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



Normal Operation

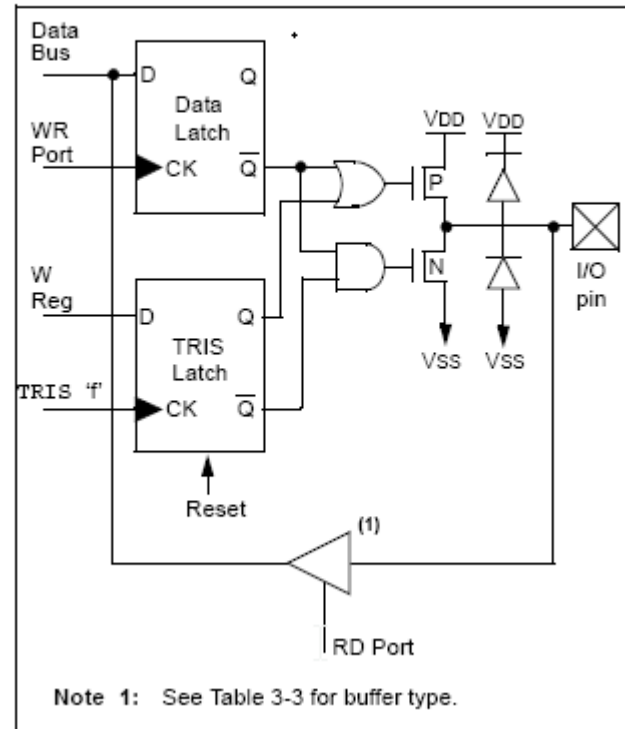
- $V_{DD} = 3.3\text{V}$ and $V_{SS} = 0.0\text{V}$
- An Electro Static Discharge event occurs



- The ESD event is much greater voltage than V_{DD}
- Current will flow from the ESD event through the top ESD diode
- This diode clamps the voltage to the IC at $V_{CC} + V_{diode}$
- Protecting the IC

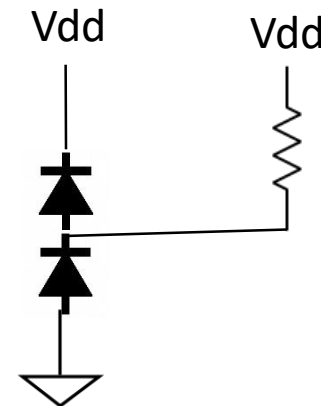
Modeling an IC that is connected to an I2C device

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



IC Vdd is turned on

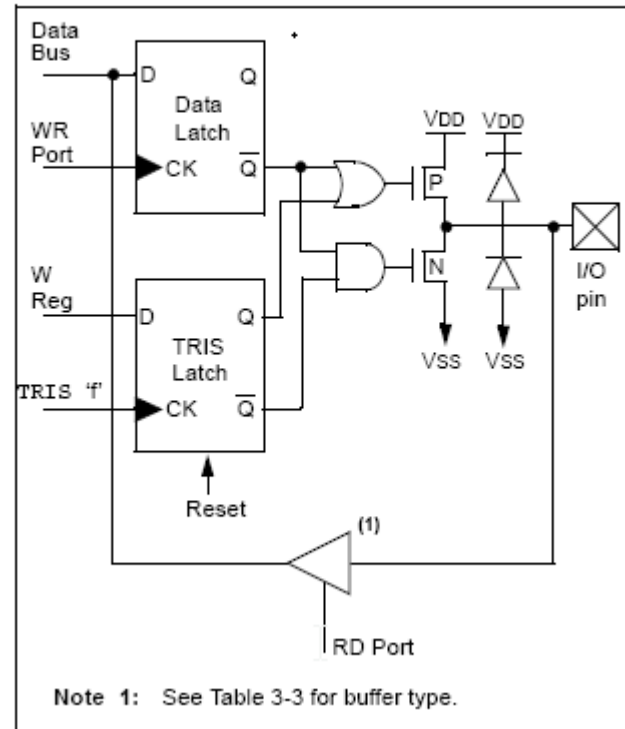
- Vdd = 3.3v and Vss = 0.0v



- 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

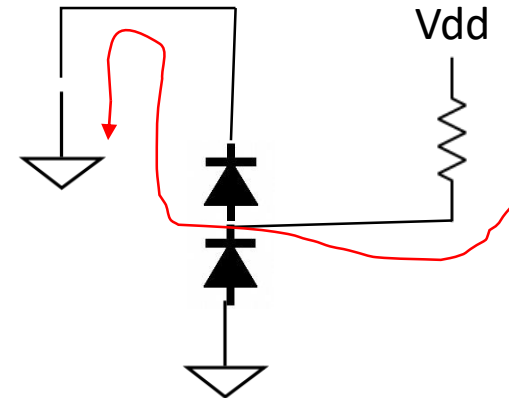
What happens when just the IC's Vdd is turned off?

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



IC Vdd is turned off

- $V_{dd} = 0.0\text{v}$ and $V_{ss} = 0.0\text{v}$



- When V_{dd} is 0.0v , the I2C signal is continuously pulled to ground through the upper ESD diode
- I2C voltage is now continuously equal to $0 + V_{diode}$
- I2C bus is now not operational
- And, each I2C line is pulling current equal to $(V_{dd} - V_{diode}) / R_{pull-up}$
- This continuous current can damage the I/O pin

Modeling an IC with two standard I/Os

- Normal Operation
 - Left IC output: $V_{dd} = 3.3\text{v}$ and $V_{ss} = 0.0\text{v}$
 - Right IC input: $V_{dd} = 3.3\text{v}$ and $V_{ss} = 0.0\text{v}$
 - Output can drive 6mA

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

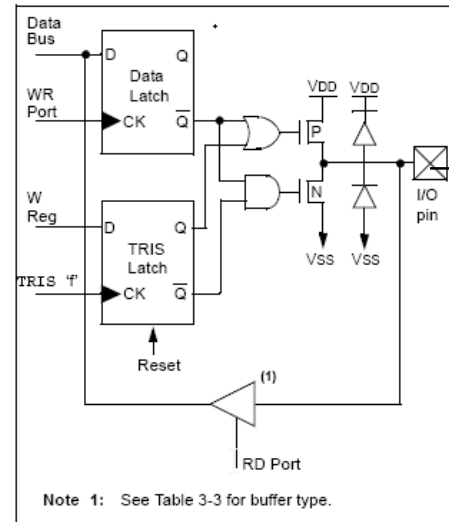
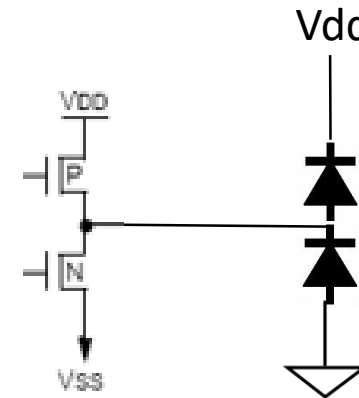
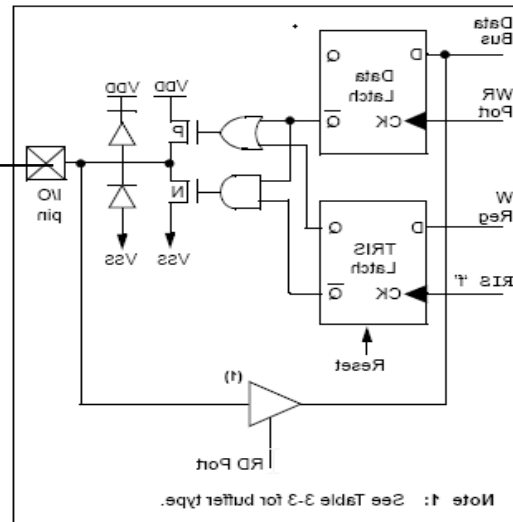


FIGURE 5-1: PIC12F508/509/16F505
EQUIVALENT CIRCUIT
FOR A SINGLE I/O PIN



Modeling an IC when 1 IC is turned off

1 IC is turned off

- Left IC output: $V_{dd} = 3.3\text{v}$ and $V_{ss} = 0.0\text{v}$
- Right IC input: $V_{dd} = 0.0\text{v}$ and $V_{ss} = 0.0\text{v}$
- Output can drive 6mA

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

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

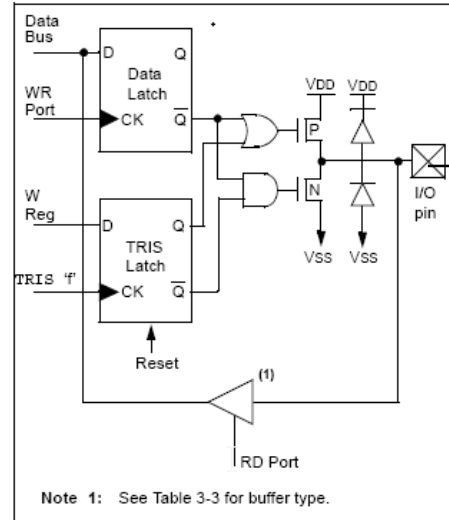
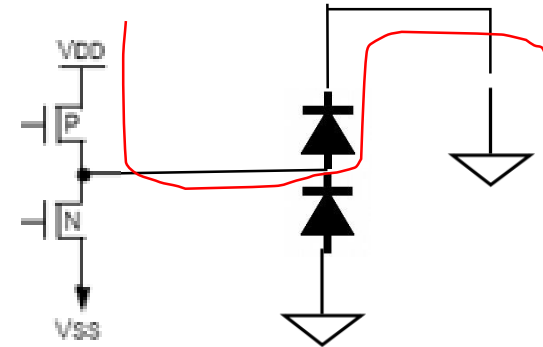
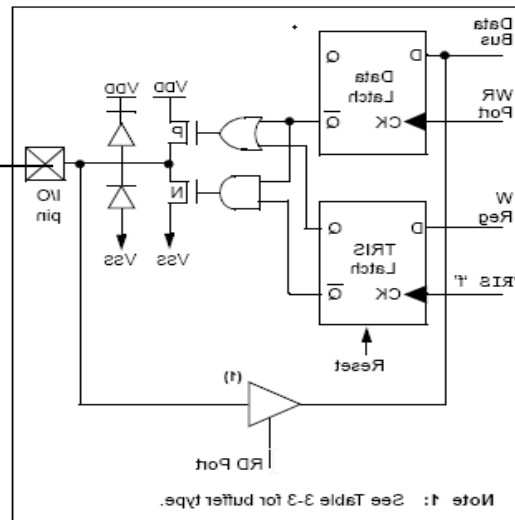


FIGURE 5-1: PIC12F508/509/16F505 EQUIVALENT CIRCUIT FOR A SINGLE I/O PIN



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
- $\pm 2g/\pm 4g/\pm 8g$ dynamically selectable full-scale
- Output Data Rates (ODR) from 1.56 Hz to 800 Hz
- $99 \mu g/\sqrt{Hz}$ noise
- 12-bit and 8-bit digital output
- I²C 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 of Load Power Management via GPIO pin

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.

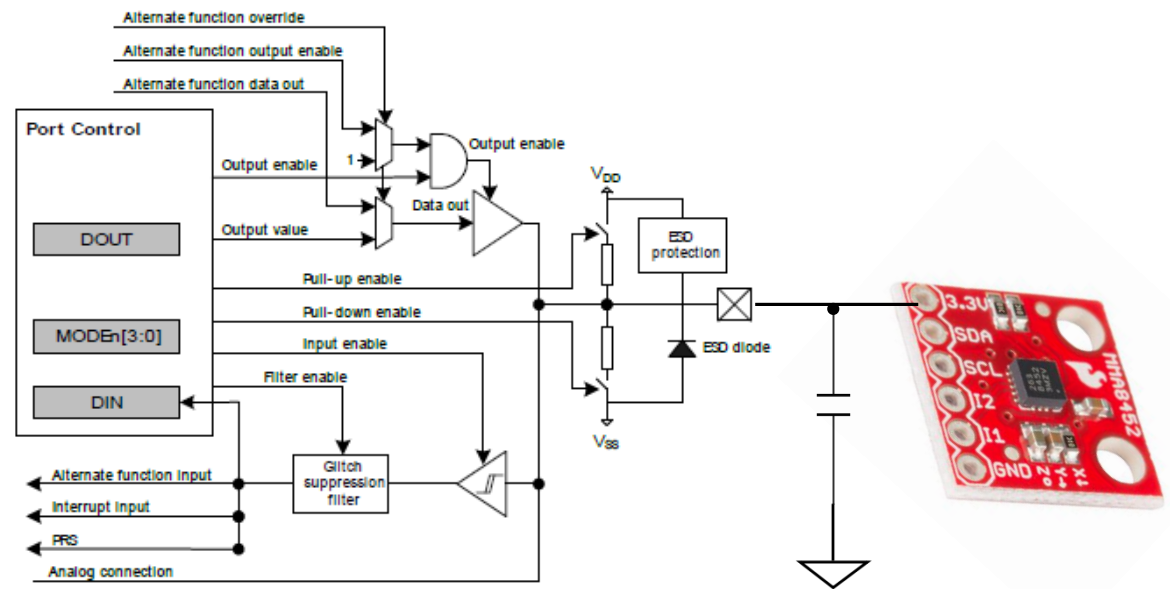
Enumerator	
gpioDriveStrengthWeakAlternateWeak	GPIO weak 1mA and alternate function weak 1mA
gpioDriveStrengthWeakAlternateStrong	GPIO weak 1mA and alternate function strong 10mA
gpioDriveStrengthStrongAlternateWeak	GPIO strong 10mA and alternate function weak 1mA
gpioDriveStrengthStrongAlternateStrong	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 V_{dd} 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



Load Power Management via GPIO pin

The MMA8452Q has two power connections

- One to the VDD, digital logic
- The second, to VDDIO
- This enables the digital logic to be powered off without causing issues with the ESD diodes on the I2C bus
- Total capacitance on the VDD line is 4.7 μ F

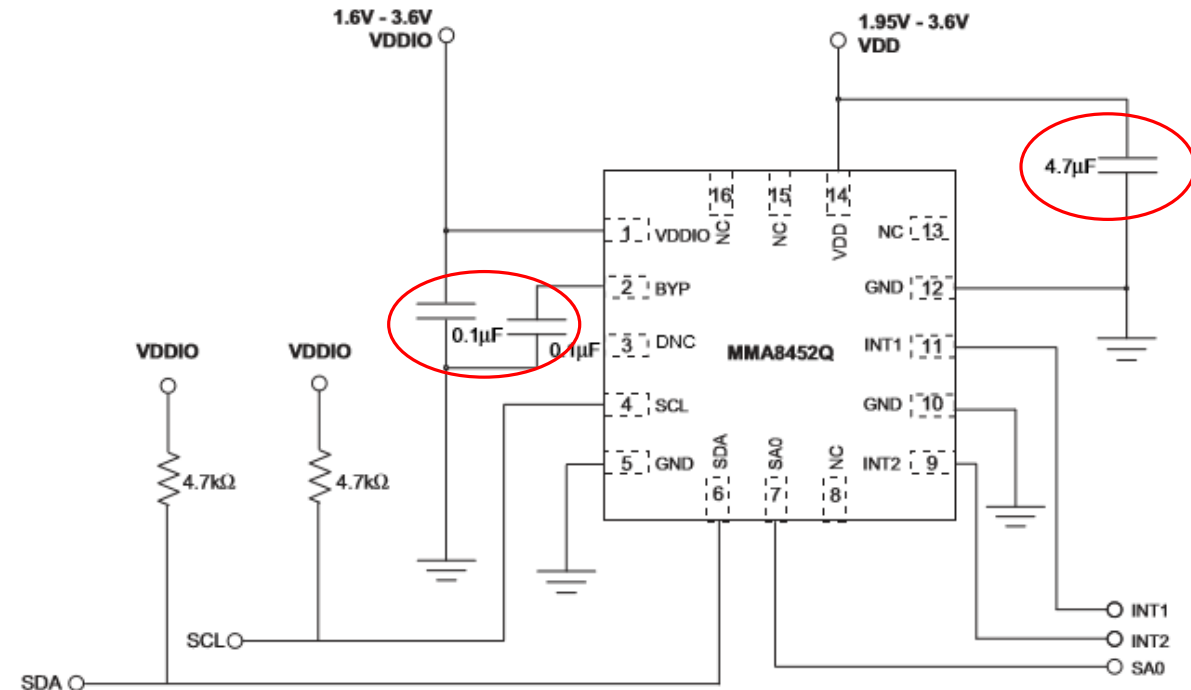
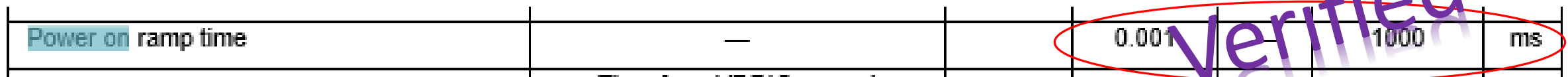
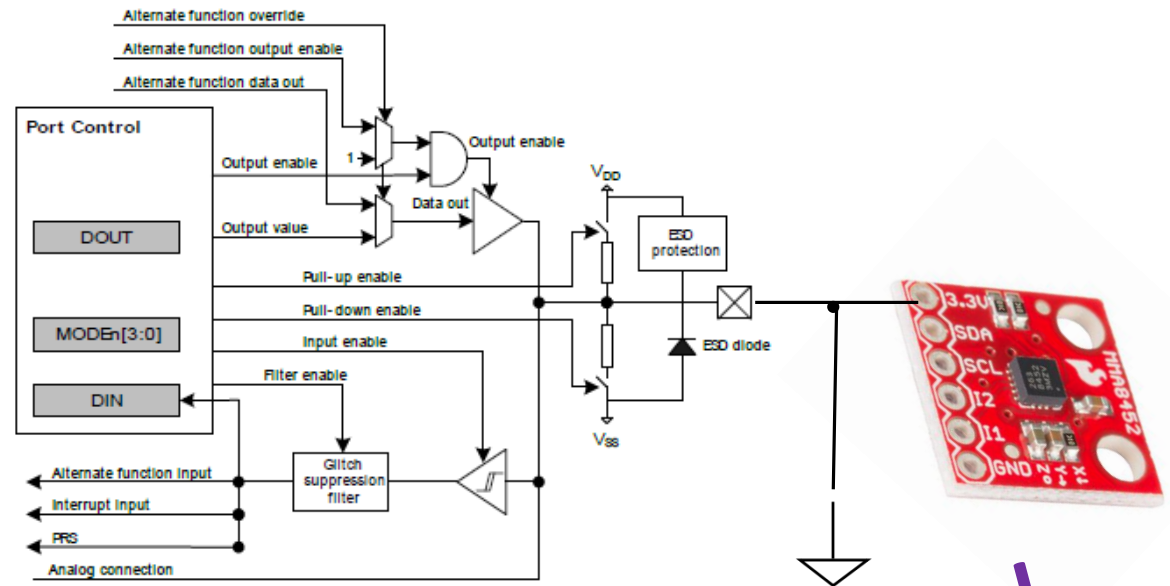


Figure 4. Application diagram

Load Power Management via GPIO pin

- How long will it take the power line to stabilize
 - Using the recommended decoupling capacitance, 4.7uF
 - Calculate time to achieve VDD
 - $i = C \frac{dV}{dT}$
 - $dT = C \frac{dV}{i}, 4.7\mu F \frac{3.3V}{6mA}$
 - $dT = 2.59mS$
 - Verify that the power ramp meets the specifications of the external device

Figure 32.1. Pin Configuration



Load Power Management via GPIO pin

Enabling the external device pseudo code

- Turn power onto the external device
 - Set GPIO Power pin to 1
- Wait for power to stabilize + external boot time
 - For the MMA8452Q
 - 2.59mS + 500uS
 - 3.09mS

Boot time	Time from VDDIO on and VDD > VDD min until I ² C is ready for operation, Cbyp = 100 nF	Tbt	—	350	500	μs
-----------	---	-----	---	-----	-----	----

- Enable GPIO I/O pins, such as SCL and SDA for I2C, on the MCU after peripheral to protect ESD diodes
- **Initialize the device for operation**
- Enable Interrupts if required
- Device is ready to be used!

Load Power Management via GPIO pin

- Disabling the external device pseudo code
 - Disable Interrupts if used
 - Disable GPIO I/O pins, such as SCL and SDA for I2C, to protect ESD diodes
 - Turn off power to the GPIO Power pin by clearing the pin
 - Do not disable, but clear the pin to 0
 - Device is now deactivated and you are saving energy!



STK6101C – BGM121 Blue Gecko dev kit

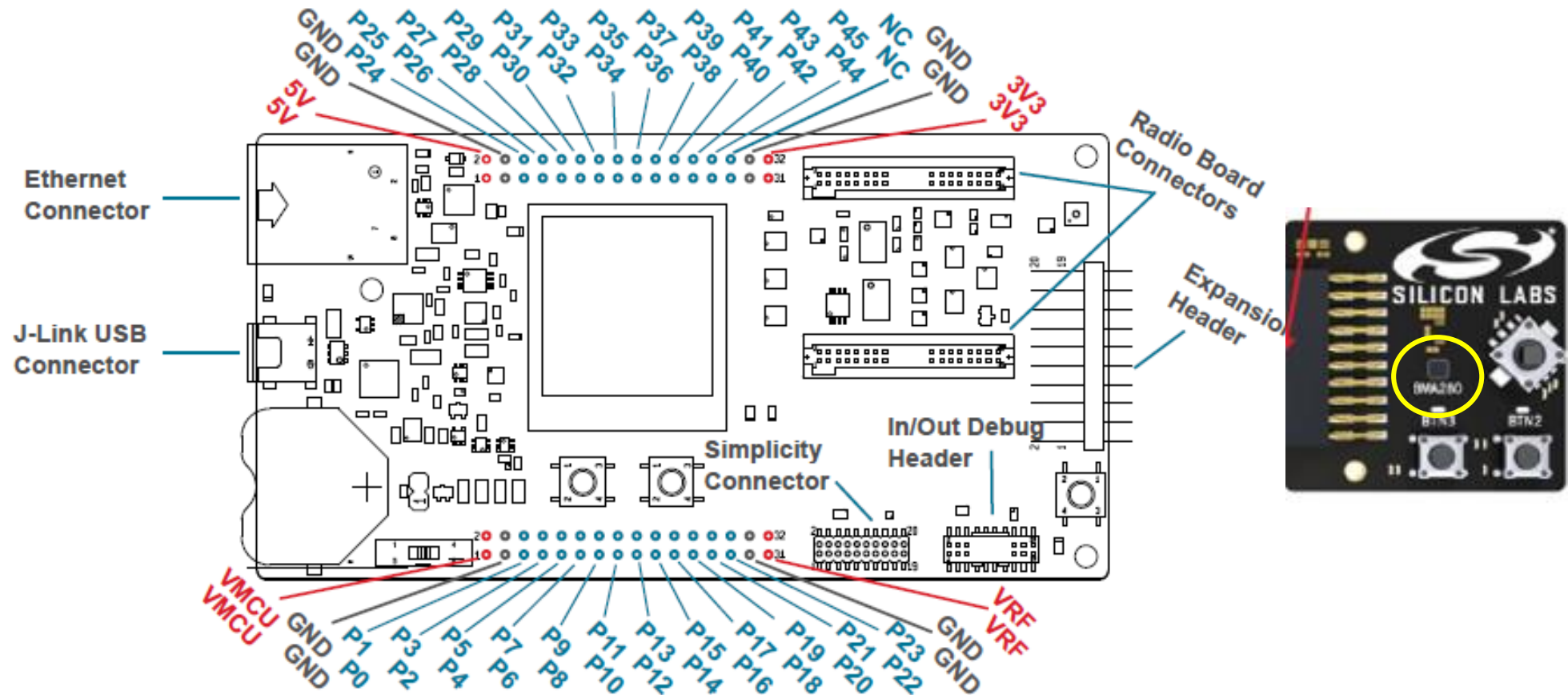
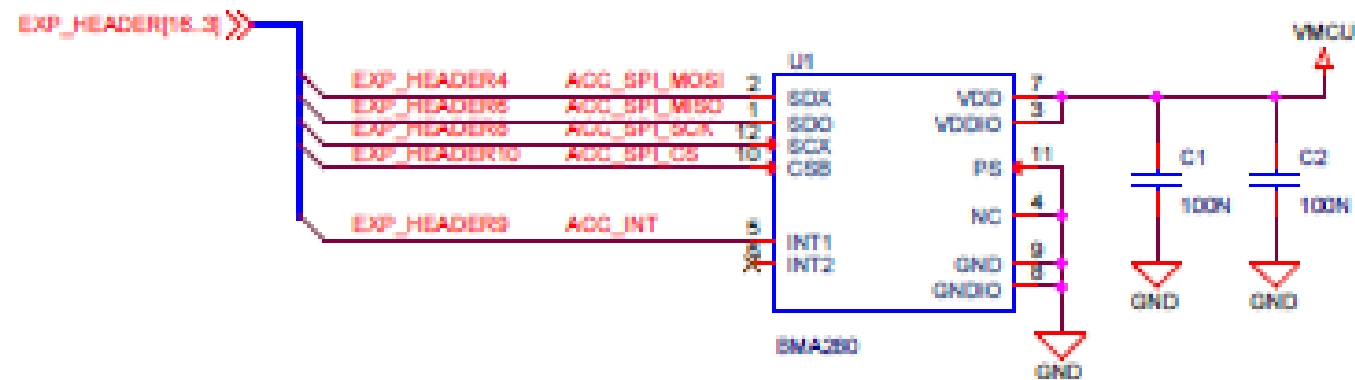


Figure 3.1. Mainboard Connector Layout

BMA280

BMA280 Accelerometer



BMA280

Typical applications

- Display profile switching
- Menu scrolling, tap / double tap sensing
- Gaming
- Pedometer / step counting
- Free-fall detection
- E-compass tilt compensation
- Drop detection for warranty logging
- Advanced system power management for mobile applications

BMA280

BMA280

14 BIT, DIGITAL, TRIAXIAL ACCELERATION SENSOR WITH INTELLIGENT ON-CHIP MOTION-TRIGGERED INTERRUPT CONTROLLER

Key features

- Ultra-Small package LGA package (12 pins), footprint 2mm x 2mm, height 0.95mm
- Digital interface SPI (4-wire, 3-wire), I²C, 2 interrupt pins
V_{DDIO} voltage range: 1.2V to 3.6V
- Programmable functionality Acceleration ranges $\pm 2g/\pm 4g/\pm 8g/\pm 16g$
Low-pass filter bandwidths 500Hz - <8Hz
up to an max. output data read out of 2kHz (unfiltered)
Integrated FIFO with a depth of 32 frames
Motion-triggered interrupt-signal generation for
 - new data
 - any-motion (slope) detection
 - tap sensing (single tap / double tap)
 - orientation recognition
 - flat detection
 - low-g/high-g detection
 - no-motion / inactivity detection
- Ultra-low power Low current consumption, short wake-up time, advanced features for system power management
- Temperature sensor
- RoHS compliant, halogen-free

BMA280

4.2 Power modes

The BMA280 has six different power modes. Besides normal mode, which represents the fully operational state of the device, there are five energy saving modes: deep-suspend mode, suspend mode, standby mode, low-power mode 1 and low-power mode 2.

The possible transitions between the power modes are illustrated in figure 2:

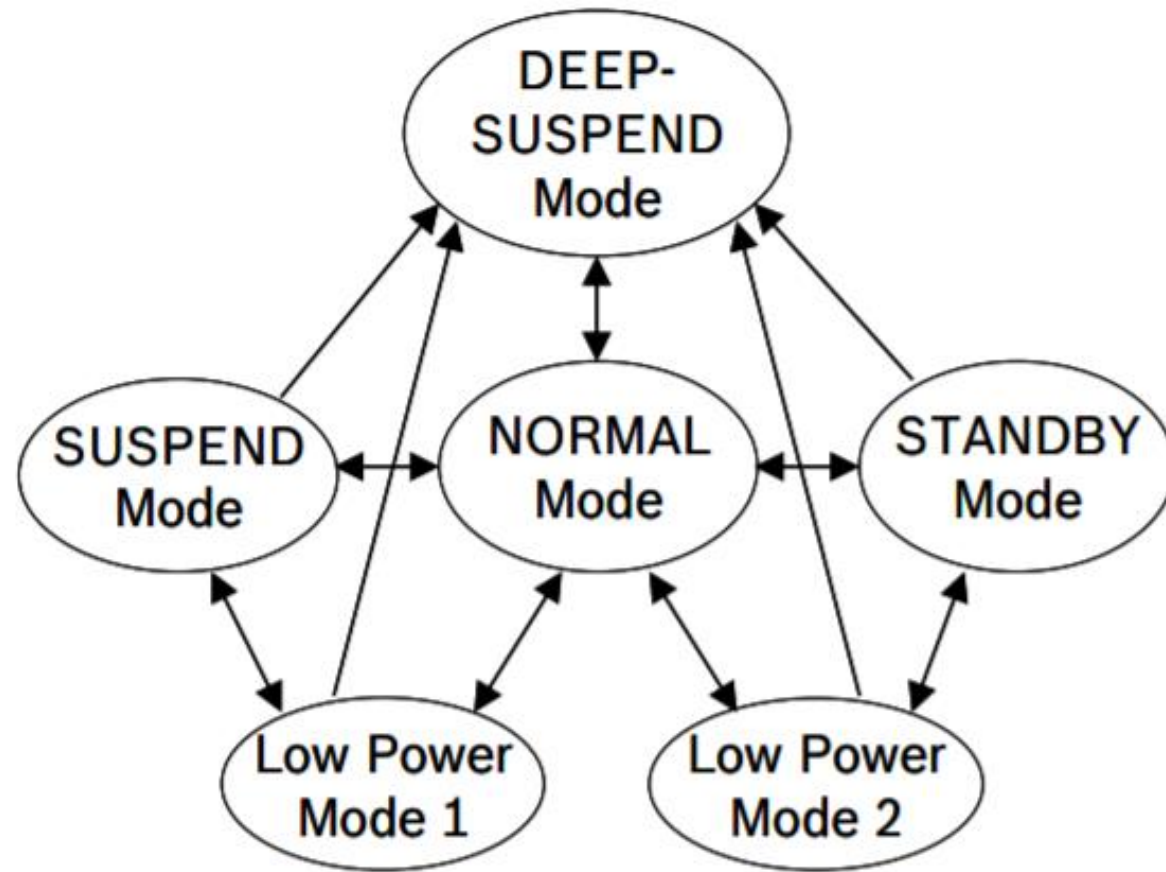


Figure 2: Power mode transition diagram

BMA280

- In the upcoming assignment, you will be enabling (taking out of Deep Suspend Mode, Low-Power Mode) and disabling the BMA280 via the joy stick
- Even though you will not be turning power on or off to the BMA280, you are performing a Load Power Management Function.

Load Power Management via GPIO pin

Enabling the external device pseudo code

- Turn power onto the external device
 - Set GPIO Power pin to 1
- Wait for power to stabilize + external boot time
 - For the MMA8452Q
 - 2.59mS + 500uS
 - 3.09mS

Boot time	Time from VDDIO on and VDD > VDD min until I ² C is ready for operation, Cbyp = 100 nF	Tbt	—	350	500	μs
-----------	---	-----	---	-----	-----	----

- Enable GPIO I/O pins, such as SCL and SDA for I2C, on the MCU after peripheral to protect ESD diodes
- **Initialize the device for operation**
- Enable Interrupts if required
- Device is ready to be used!

BMA280

- In the upcoming assignment, you will be enabling (taking out of Deep Suspend Mode, Low-Power Mode) and disabling the BMA280 via the joy stick
- Even though you will not be turning power on or off to the BMA280, you are performing a Load Power Management Function.

Wake-Up Time 1	$t_{w,up1}$	from Low-power Mode 1 or Suspend Mode or Deep Suspend Mode, ODR_{max} .		1.3	1.8	ms
----------------	-------------	---	--	-----	-----	----

- How will you insure that the BMA280 will have 1.8mS to wake-up when you re-enable the BMA280?

Delay loops

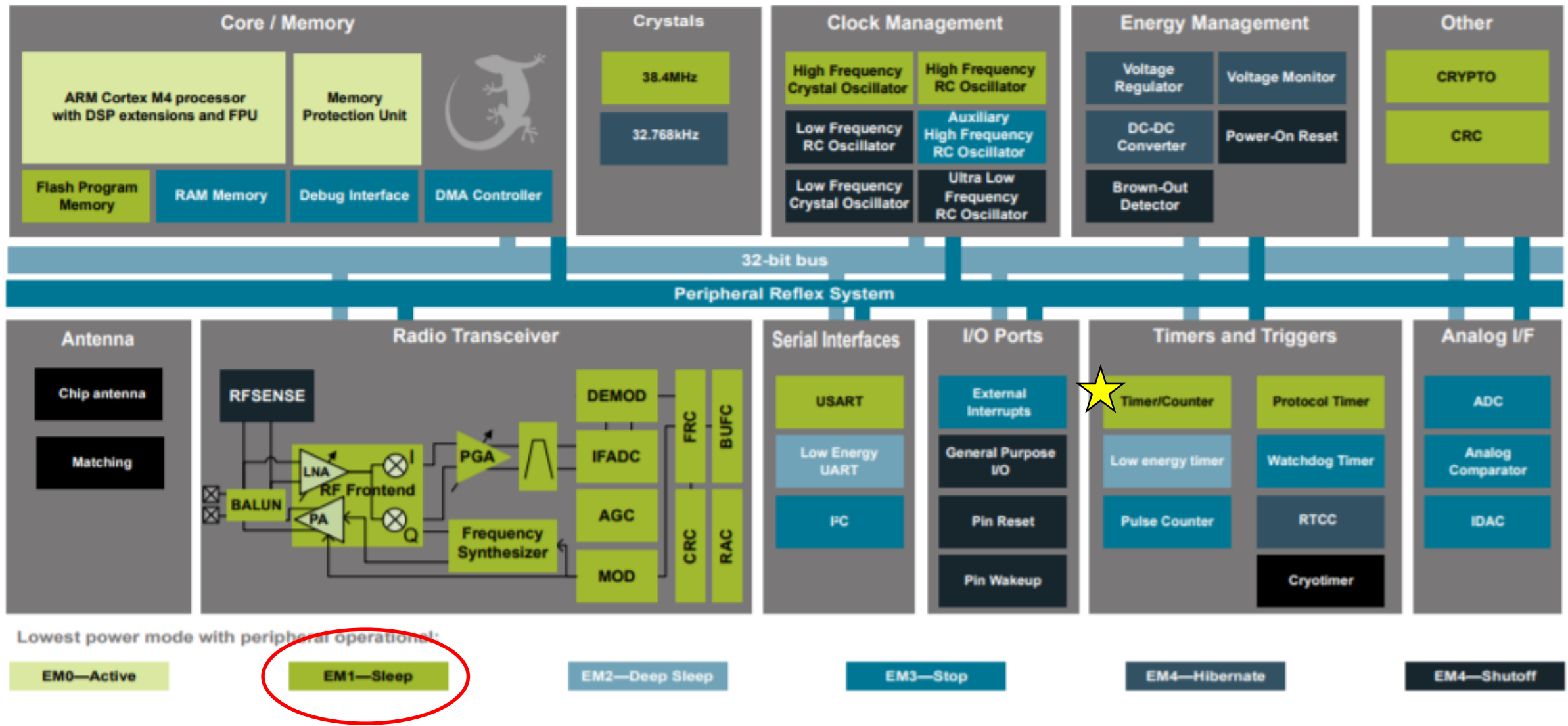
- For loops

- `for (int j = 0; j < 1000000; i++);`
 - None deterministic – compiler dependent
 - Optimizer may optimize the for loop delay

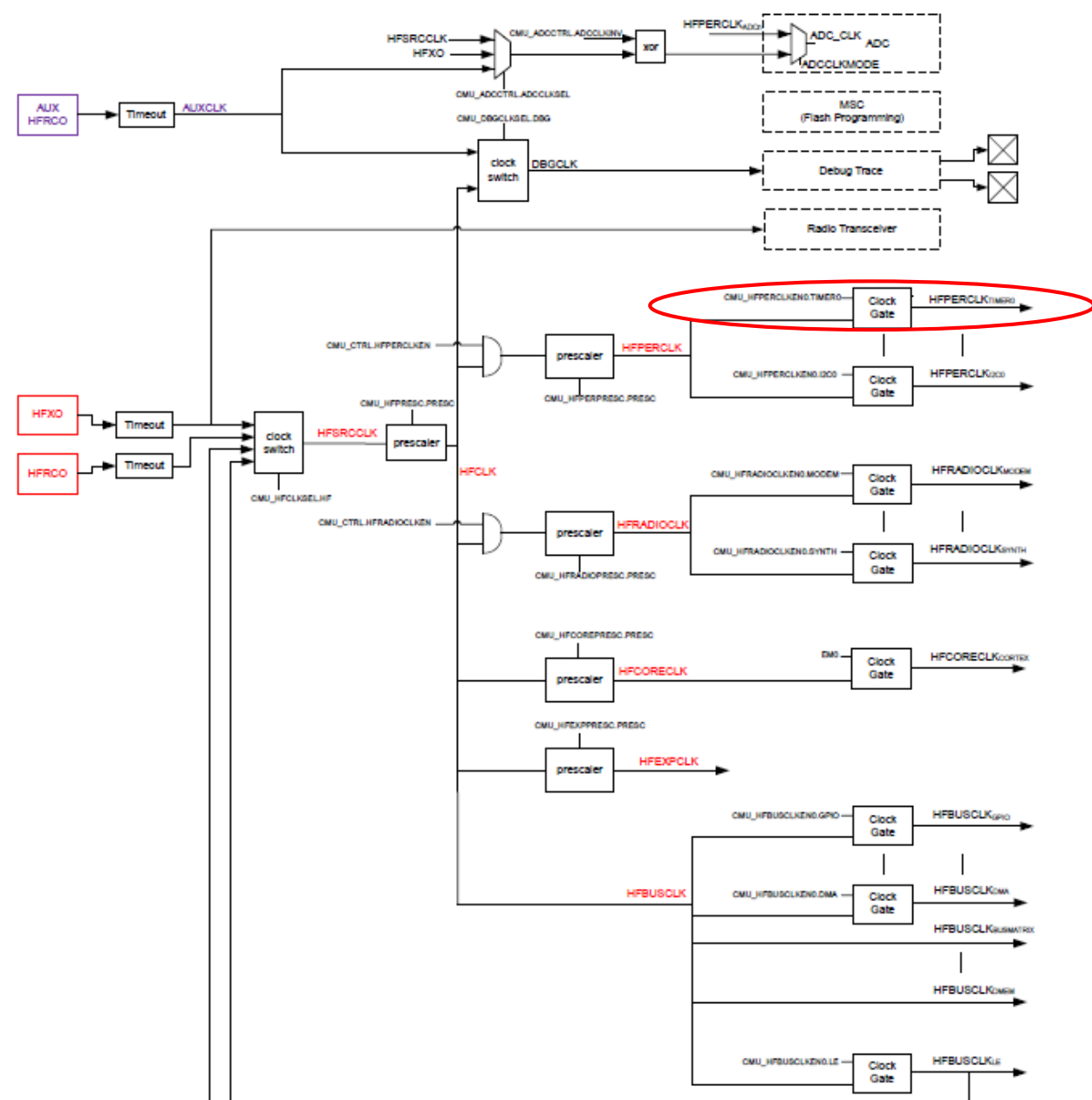
- On board Timers



- Deterministic
 - Cannot be compiled or optimized out
 - Can reduce energy by putting the processor asleep while counting the delay



Blue Gecko High Frequency Clock Tree



Setting up TIMER0

- First, the clock tree to the TIMER0 must be established
 - Without establishing the clock tree, all writes to the TIMERN registers will not occur
 - Pseudo code in the CMU setup routine to enable the TIMERN clock tree:
 - The HFPERCO must be enabled using [CMU_ClockEnable](#) for the HFPER
 - Lastly, enable the TIMERN clocking using the [CMU_ClockEnable](#) for the TIMERN

Timer0

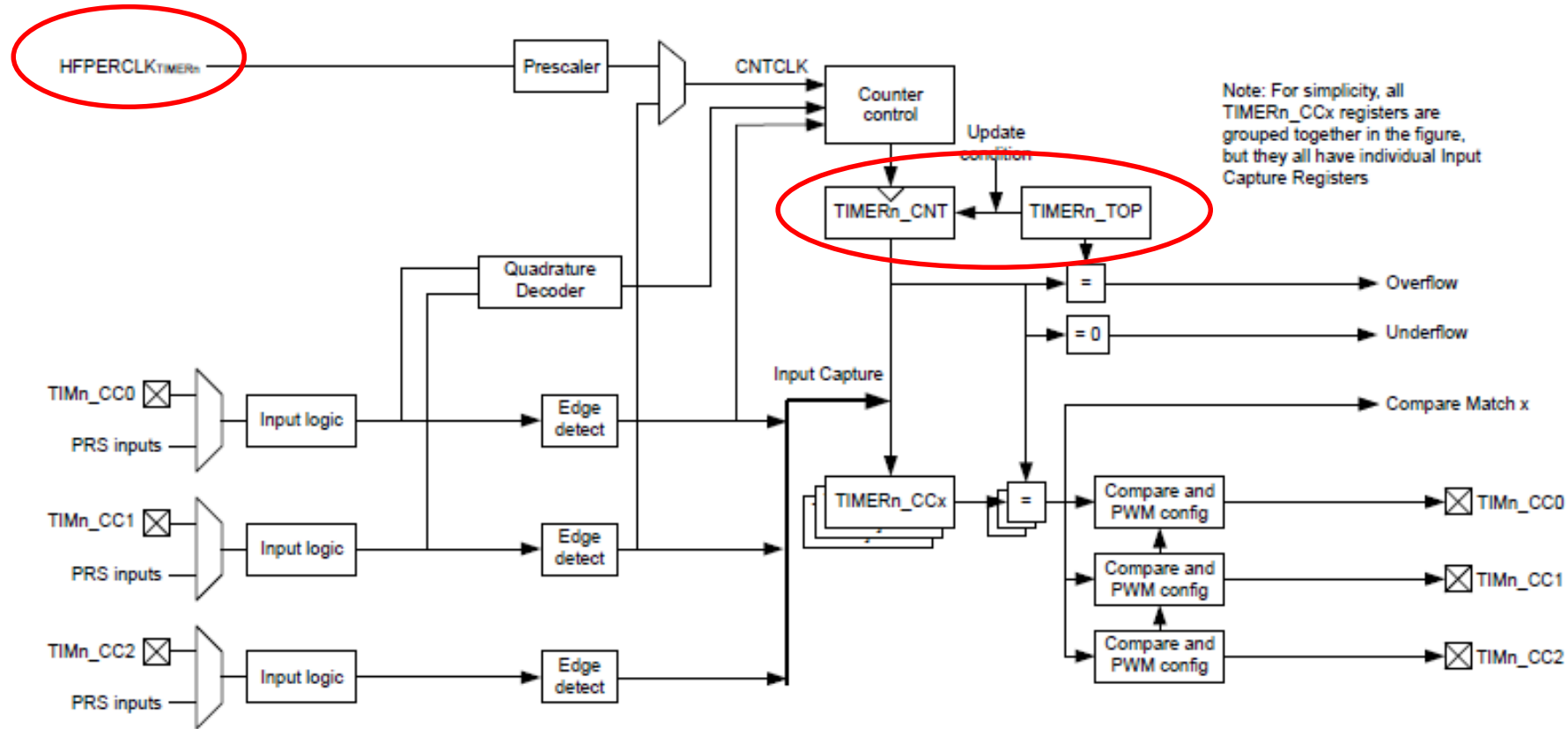


Figure 20.1. TIMER Block Overview

Setting up the TIMERN

- Second, the TIMERN must be set up
 - TIMER_Init_TypeDef structure must be programmed
 - Specify clock source
 - HFPERCLK
 - Compare/Capture Channel 1 input
 - Or, the overflow or underflow of the next lower TIMERN
 - This feature can be used to make the 16-bit timers become a 32-bit timer!
 - The direction of the TIMERN
 - UP, DOWN, or UP & DOWN
 - Set whether a particular timer is in sync with another timer
 - Set whether the TIMERN will be one time or repeat
 - There are many more functions in the TIMER_Init_TypeDef that may need to be considered
- Program the functionality of the TIMERN using [TIMER_init\(\)](#)

Setting up the TIMERN

- Third, the TIMERN interrupts must be enabled if needed
 - Clear all interrupts from the TIMERN to remove any interrupts that may have been set up inadvertently by accessing the `TIMERN->IFC` register or the `emlib` routine
 - Enable the desired interrupts by setting the appropriate bits in `TIMERN->IEN`
 - Set **BlockSleep** mode to the desired Energy Mode
 - TIMERN only uses HFPERCLK
 - TIMERN when enabled must have BlockSleepMode set to EM1
 - Enable interrupts to the CPU by enabling the ACMP in the Nested Vector Interrupt Control register using `NVIC_EnableIRQ(TIMERN_IRQn);`



Setting up the TIMERN

- Fourth, the TIMERN interrupt handler must be included
 - Routine name must match the vector table name:
`Void TIMERN_IRQHandler(void) {
}`
 - Inside this routine, you add the functionality that is desired for the TIMERN interrupts

BMA280

- In the upcoming assignment, you will be enabling (taking out of Deep Suspend Mode, Low-Power Mode) and disabling the BMA280 via the joy stick
- Even though you will not be turning power on or off to the BMA280, you are performing a Load Power Management Function.

Wake-Up Time 1	$t_{w,up1}$	from Low-power Mode 1 or Suspend Mode or Deep Suspend Mode, ODR_{max} .		1.3	1.8	ms
----------------	-------------	---	--	-----	-----	----

- How will you insure that the BMA280 will have 1.8mS to wake-up when you re-enable the BMA280? **TIMERn Peripheral**

BMA280

- You will be enabling (Load Power Management) the BMA280 in the ADC Interrupt handler when the Joy Stick is pushed to its “north” position

Calling BMA280 Enable routine from an interrupt

Load Power Management wake up pseudo code

```
void ADC0_IRQHandler(void) {  
    CORE_ATOMIC_IRQ_DISABLE();  
    Read ADC0 Interrupt Flag Register;  
    Clear ADC0 Interrupt Flag Register;  
    If ADC0 compare match;  
        Enable BMA20;  
    CORE_ATOMIC_IRQ_ENABLE(); }  
}
```

BMA280 Enable routine

```
void BMA280 Enable pseudo code(void) {  
    Return BMA280 to normal power mode from suspend mode;  
    Call delay_TIMER0 for Wake Up-1 Time (1.8ms);  
  
    Reset BMA280 Interrupt hardware to clear any pending interrupts;  
    Set BG GPIO pin as Input to interrupt BG on BMA280 interrupt;  
    Clear appropriate GPIO interrupt pins;  
    Enable BG BMA280 GPIO interrupt pin; }
```

Delay Timer routine

```
void Delay Timer pseudo code(time in micro seconds) {  
    Determine TIMERN count based on time;  
    Set TIMERN_cnt;  
  
    Set block sleep mode for TIMERN;  
    Enter Sleep;  
    Unblock sleep mode or TIMERN;}
```

TIMERn Interrupt Handler

```
void TIMERn_IRQHandler(void) {  
    CORE_ATOMIC_IRQ_DISABLE();  
    Determine TIMER Interrupt source;  
    Clear Interrupt source  
    CORE_ATOMIC_IRQ_ENABLE();}
```

BMA280

- You will be enabling (Load Power Management) the BMA280 in the ADC Interrupt handler when the Joy Stick is pushed to its “north” position
- What are the potential issues that may arise in the interrupt handler when calling the Load Power Management routine to enable the BMA280?
 - Not interrupting on the TIMERN interrupt to indicate the 1.8mS delay has completed
 - NVIC interrupts have not been re-enabled before enabling / calling the BMA280 Load Power Management routine

Calling BMA280 Enable routine from an interrupt

Load Power Management wake up pseudo code

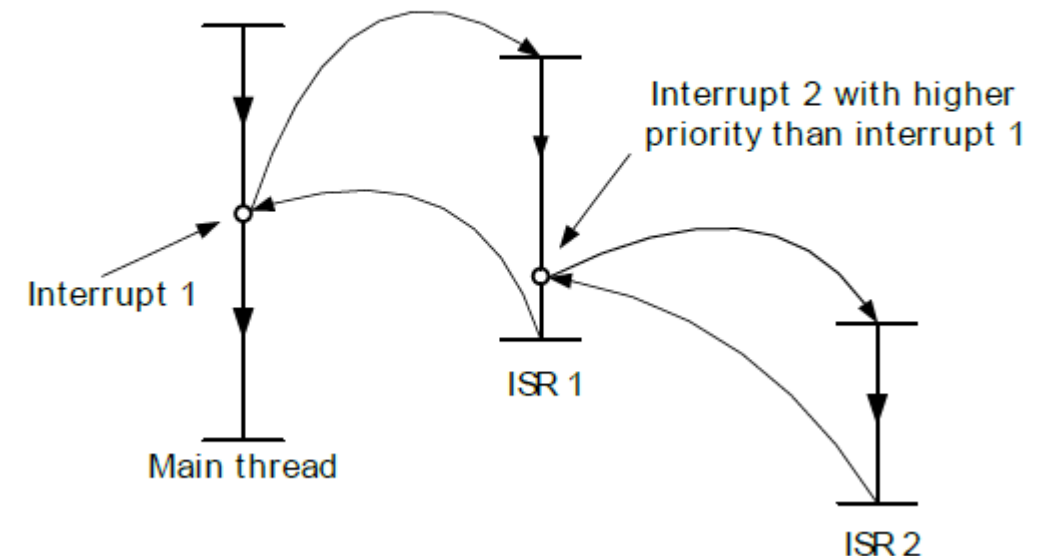
```
void ADC0_IRQHandler(void) {  
    CORE_ATOMIC_IRQ_DISABLE();  
    Read ADC0 Interrupt Flag Register;  
    Clear ADC0 Interrupt Flag Register;  
    CORE_ATOMIC_IRQ_ENABLE();  
    If ADC0 compare match;  
        Enable BMA20;}  
}
```

Enabling the nesting
TIMERn interrupt to the
ADC0_IRQHandler()

Interrupts

- Interrupt priorities can be assigned to the different IRQs, thus allowing lower latency for the most important interrupts for real-time control etc
- In interrupt controllers that support nesting, it is also possible for a high priority interrupt handler to start immediately even if another lower priority handler is executing
- The CPU will then continue where it left off in the lower priority handler once it is done servicing the higher priority interrupt. Figure 1.2 (p. 3) shows an example where a higher priority interrupt (2) is serviced in the middle of the lower priority interrupt handler (1).

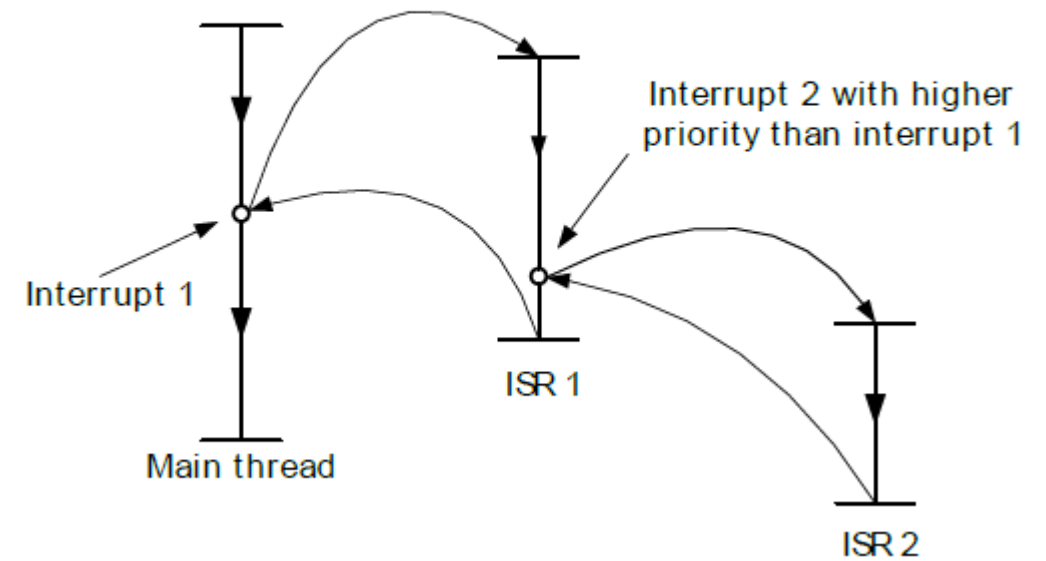
Figure 1.2. Nested Interrupts



Interrupts

- To increase priority control in systems with interrupts, the NVIC supports priority grouping. This divides each interrupt priority register entry into two fields:
 - an upper field that defines the group priority
 - a lower field that defines a sub priority within the group.

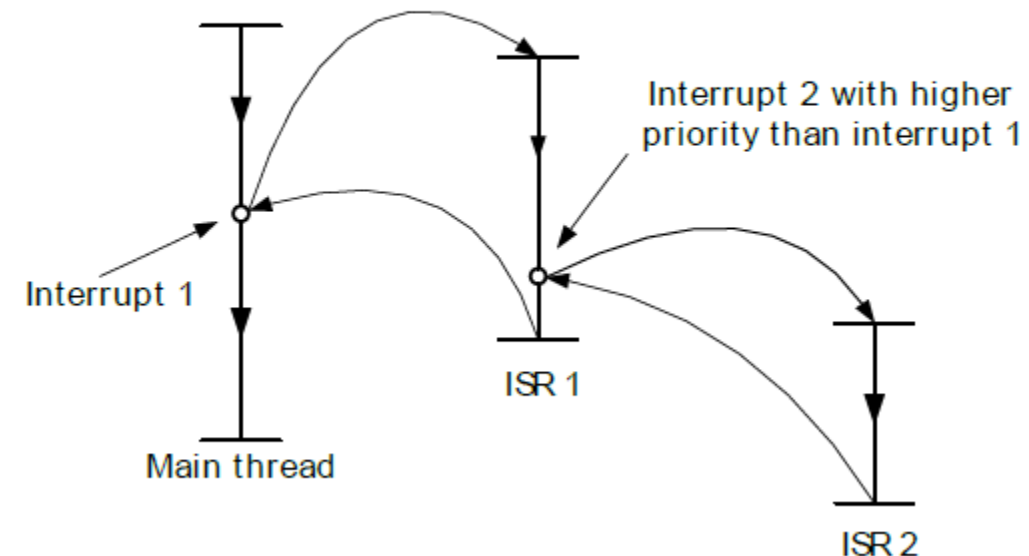
Figure 1.2. Nested Interrupts



Interrupts

- Only the group priority determines preemption of interrupt exceptions.
- When the processor is executing an interrupt exception handler, another interrupt with the same group priority as the interrupt being handled **does not** preempt the handler. If multiple pending interrupts have the same group priority

Figure 1.2. Nested Interrupts



NVIC_SetPriority

Table 4-3 CMSIS access NVIC functions

CMSIS function	Description
<code>void NVIC_EnableIRQ(IRQn_Type IRQn)^a</code>	Enables an interrupt or exception.
<code>void NVIC_DisableIRQ(IRQn_Type IRQn)^a</code>	Disables an interrupt or exception.
<code>void NVIC_SetPendingIRQ(IRQn_Type IRQn)^a</code>	Sets the pending status of interrupt or exception to 1.
<code>void NVIC_ClearPendingIRQ(IRQn_Type IRQn)^a</code>	Clears the pending status of interrupt or exception to 0.
<code>uint32_t NVIC_GetPendingIRQ(IRQn_Type IRQn)^a</code>	Reads the pending status of interrupt or exception. This function returns non-zero value if the pending status is set to 1.
<code>void NVIC_SetPriority(IRQn_Type IRQn, uint32_t priority)^a</code>	Sets the priority of an interrupt or exception with configurable priority level to 1.
<code>uint32_t NVIC_GetPriority(IRQn_Type IRQn)^a</code>	Reads the priority of an interrupt or exception with configurable priority level. This function return the current priority level.

a. The input parameter `IRQn` is the IRQ number, see [Table 2-16 on page 2-22](#) for more information.

BMA280

- You will be enabling (Load Power Management) the BMA280 in the ADC Interrupt handler when the Joy Stick is pushed to its “north” position
- What are the potential issues that may arise in the interrupt handler when calling the Load Power Management routine to enable the BMA280?
 - Not interrupting on the TIMERN interrupt to indicate the 1.8mS delay has completed
 - NVIC interrupts have not been re-enabled before enabling / calling the BMA280 Load Power Management routine
 - The ADCn and TIMERN interrupts are set at the same interrupt level

BMA280

- For nested interrupts, the following is required:
 - Initial Interrupt Service Handler must have enabled NVIC interrupts
 - `CORE_ATOMIC_IRQ_ENABLE();`
 - The nested interrupt must have a higher priority interrupt, a lower priority number, than the current Interrupt Service Handler
 - `NVIC_SetPriority(IRQn, Priority);`

SPI

- BMA280 data sheet's SPI write and read timing

Table 22: SPI timing

Parameter	Symbol	Condition	Min	Max	Units
Clock Frequency	f_{SPI}	Max. Load on SDI or SDO = 25pF, $V_{DDIO} \geq 1.62V$		10	MHz
		$V_{DDIO} < 1.62V$		7.5	MHz

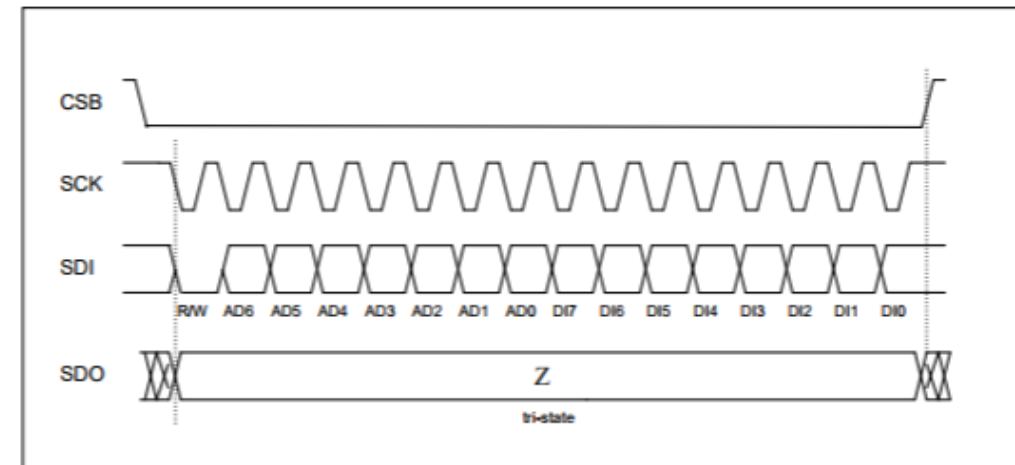


Figure 14: 4-wire basic SPI write sequence (mode '11')

The basic read operation waveform for 4-wire configuration is depicted in figure 15:

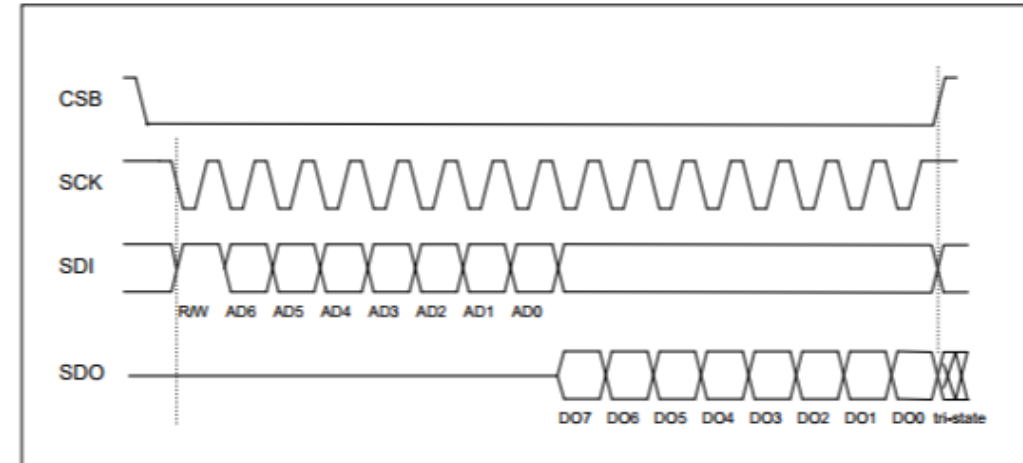
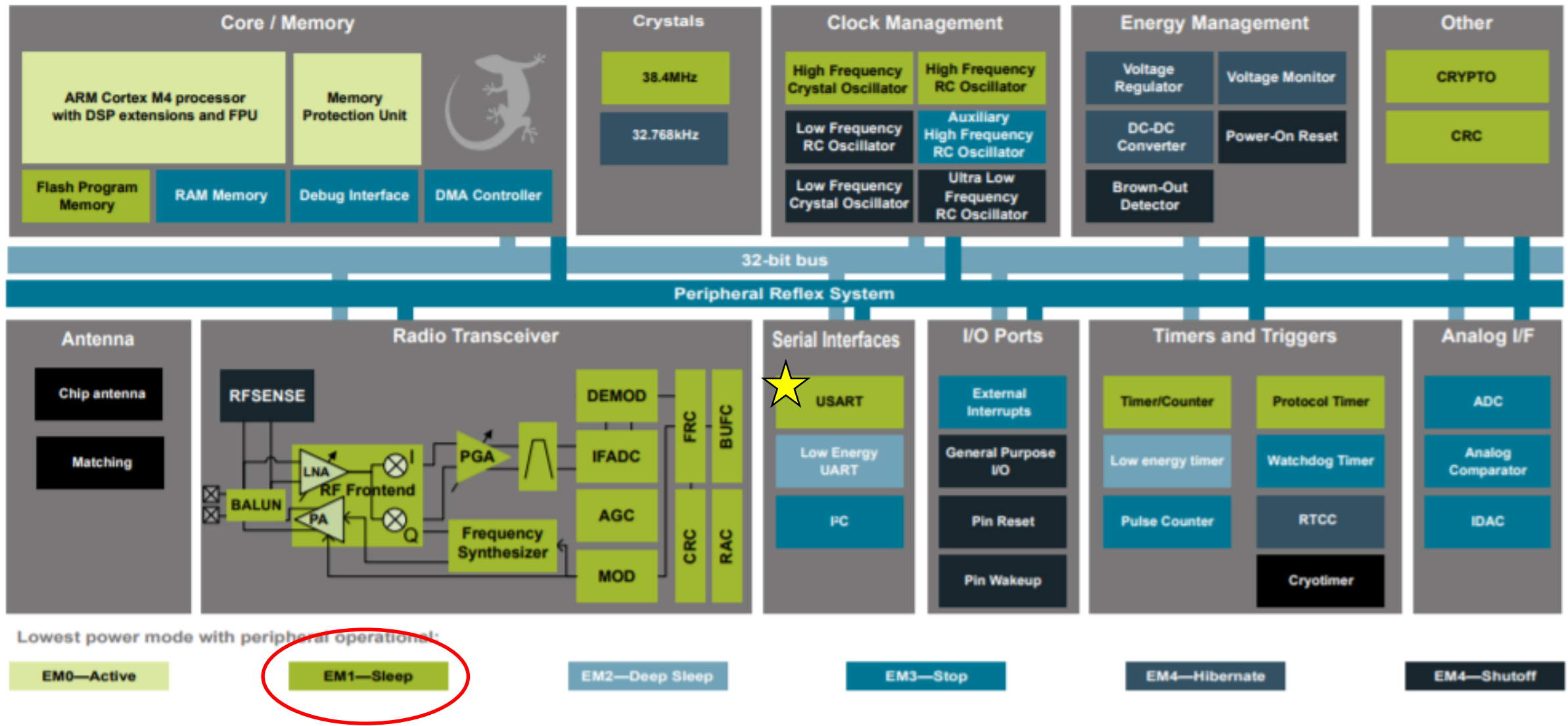
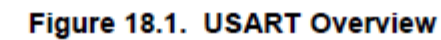


Figure 15: 4-wire basic SPI read sequence (mode '11')



- Blue Gecko's USART peripheral



Setting up USART

- First, the clock tree to the USARTn must be established
 - Without establishing the clock tree, all writes to the USARTn registers will not occur
 - Pseudo code in the CMU setup routine to enable the USARTn clock tree:
 - The HFPERCO must be enabled using [CMU_ClockEnable](#) for the HFPER
 - Lastly, enable the TIMERN clocking using the [CMU_ClockEnable](#) for the USARTn

Setting up the USARTn

- Second, the USARTn must be set up for SPI
 - USART_InitSync_TypeDef structure must be programmed for a synchronous connection
 - Specify whether the use of auto CS, Chip Select, assertion / deassertion
 - autoCsEnable, autoCsHold, and autoCsSetup
 - When would you not use auto Chip Select option?
 - When you are using the USART to control multiple SPI devices. Then, SW knowing which device it is communicating with controls that individual SPI CS line

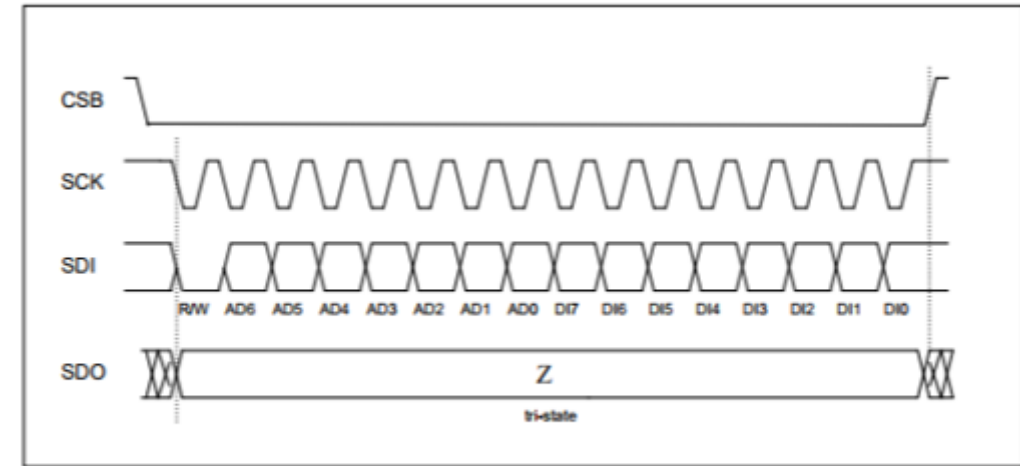


Figure 14: 4-wire basic SPI write sequence (mode '11')

Setting up the USARTn

- Second, the USARTn must be set up for SPI
 - USART_InitSync_TypeDef structure must be programmed for a synchronous connection
 - Specify whether the use of auto CS, Chip Selection, assertion / deassertion
 - autoCsEnable, autoCsHold, and autoCsSetup
 - Specify clocking and baud rate information
 - autoTX, baudrate, and clockMode
 - Specify data format information
 - ddatabits, master, enable, and msbf (most significant bit first)
 - Remaining synchronous set up information
 - prsRxCh, prsRxEnable, and refFreq
 - `USART_InitSync(spi, &spi_init);`

Setting up the USARTn

- Second, the USARTn must be set up for SPI
 - With the SPI /USARTn communicating of chip, you will need to set up the pins accordingly to their function in the GPIO peripheral
 - CS is output, Push-Pull
 - SCLK is output, Push-Pull
 - MISO is input, Input
 - MOSI is output, Push-Pull
 - With the GPIO pins initialized to their function, these GPIO pins must be routed to the SPI / USARTn peripheral
 - `spi->ROUTELOC0`
 - The pins now must be enabled to the SPI / USARTn peripheral
 - `spi->ROUTEPEN`