ECEN 5023-001, -001B, -740

Mobile Computing & IoT Security

Lecture #6

2 February 2017

Be Boulder.

University of Colorado Boulder

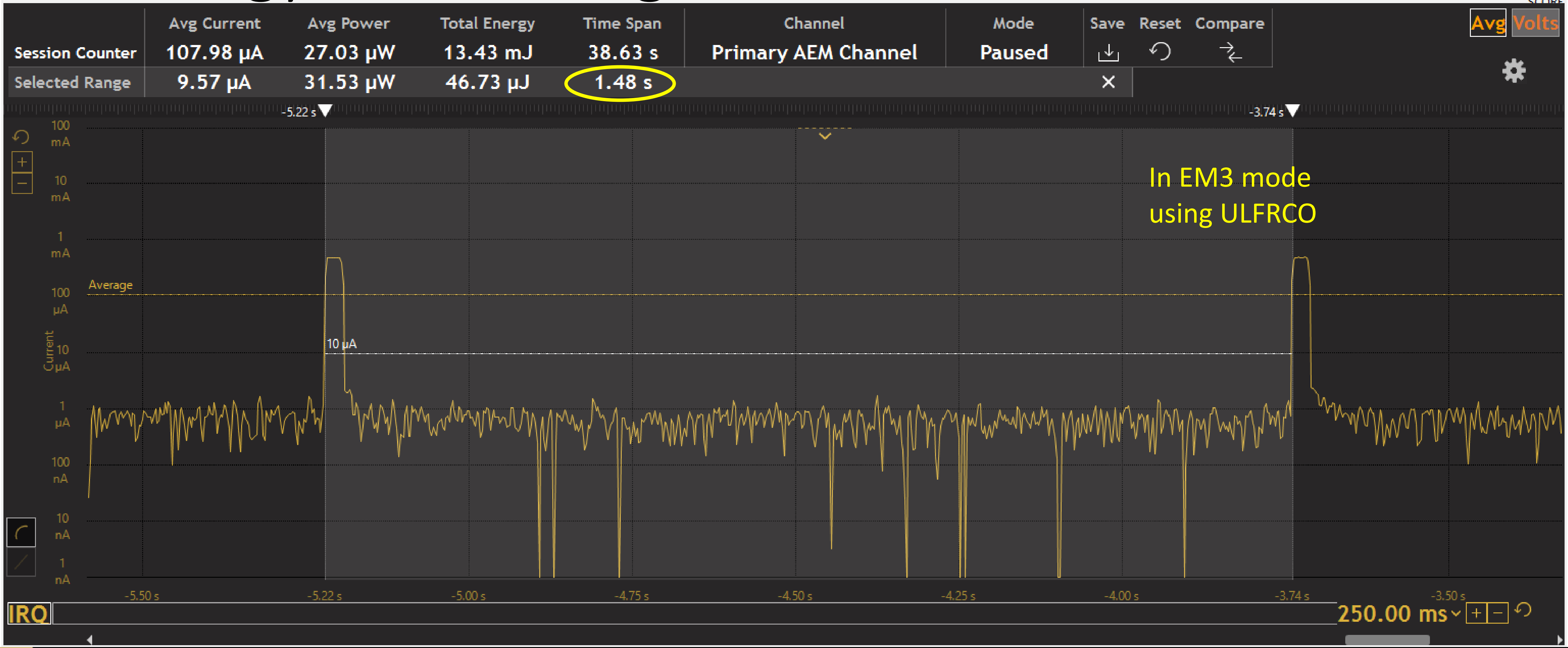
Agenda

- Review of Managing Energy Modes programming assignment
- Self-calibrating ULFRCO programming assignment
- ACMP peripheral
- TIMERn peripheral
- Self-calibrating ULFRCO
- Mobile Computing Adaptation

Class Announcements

- Quiz #3 is due at 11:59pm on Sunday, February 5th, 2017
- Self-calibrating ULFRCO is due at 11:59pm on Wednesday, February 8th, 2017

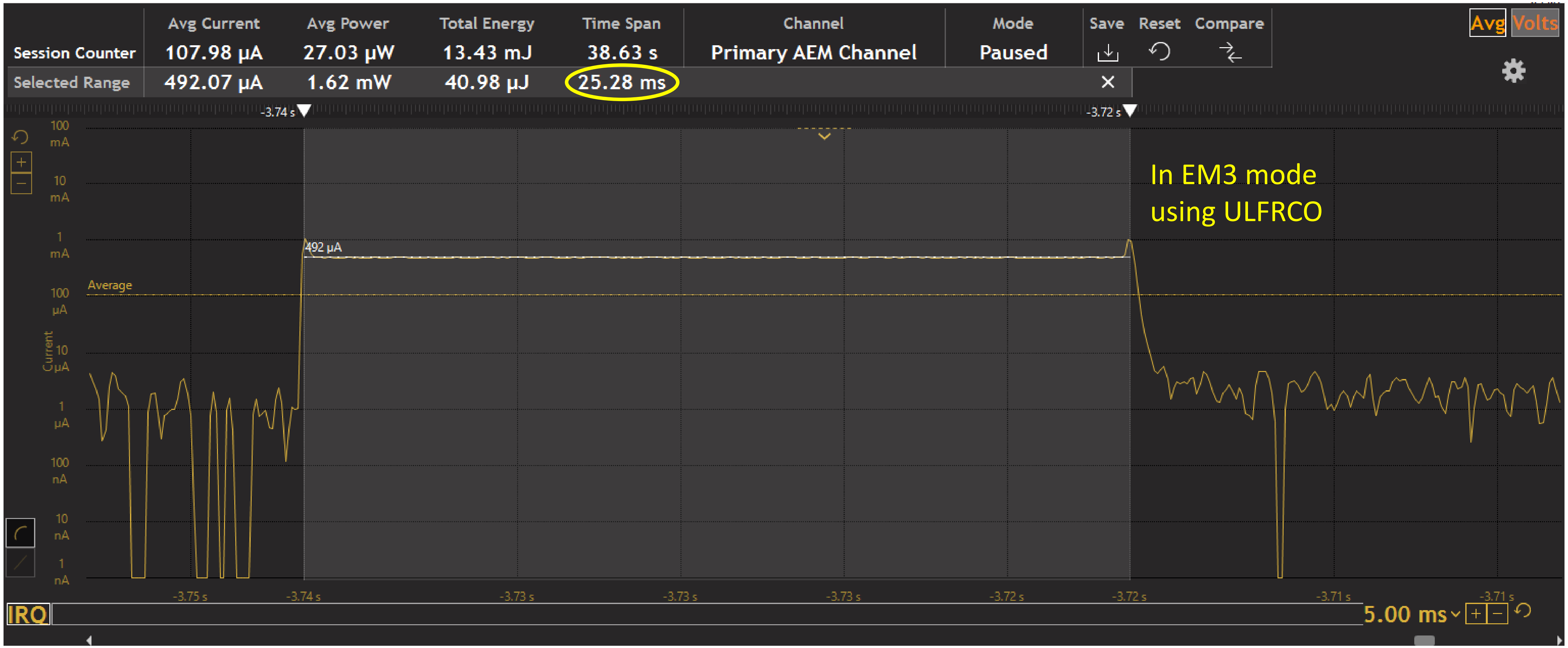
Energy Mode Assignment #1 review



	Avg Current	Avg Power	Total Energy	Time Span	Channel	Mode
Session Counter	107.98 µA	27.03 µW	13.43 mJ	38.63 s	Primary AEM Channel	Paused
Selected Range	9.57 µA	31.53 µW	46.73 µJ	1.48 s		

In EM3 mode using ULFRCO

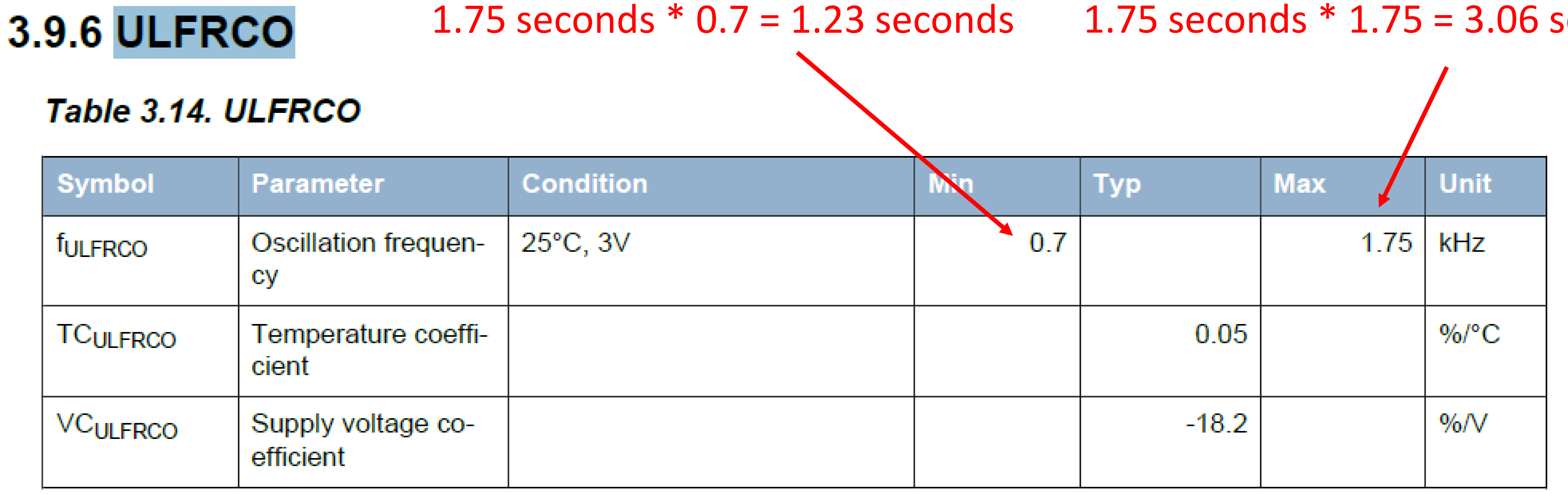
Energy Mode Assignment #1 review



	Avg Current	Avg Power	Total Energy	Time Span	Channel	Mode
Session Counter	107.98 µA	27.03 µW	13.43 mJ	38.63 s	Primary AEM Channel	Paused
Selected Range	492.07 µA	1.62 mW	40.98 µJ	25.28 ms		

In EM3 mode using ULFRCO

Energy Mode Assignment #1 review



1.75 seconds * 0.7 = 1.23 seconds

1.75 seconds * 1.75 = 3.06 seconds

3.9.6 ULFRCO

Table 3.14. ULFRCO

Symbol	Parameter	Condition	Min	Typ	Max	Unit
f_{ULFRCO}	Oscillation frequency	25°C, 3V	0.7		1.75	kHz
TC_{ULFRCO}	Temperature coefficient			0.05		%/°C
VC_{ULFRCO}	Supply voltage coefficient			-18.2		%/V

Did anyone get a period less than 1.40 seconds or greater than 2.1 seconds?

Managing Energy Mode Assignment Rubric Questions

- Question 1: Energy score of 2.3 or 2.4 and average current <= 4.5mA (0.5pts for Energy Score & 0.5 pts for average current)
- Question 2: Energy score of 2.8, 2.9, or 3.0 and average current <= 1.5mA (0.5pts for Energy Score & 0.5 pts for average current)
- Question 3: Energy score of 5.3, 5.4, or 5.5 and average current <= 1.4uA (0.5pts for Energy Score & 0.5 pts for average current)
- Question 4: Period measured 1.70 to 1.80 seconds for ½ point and On-duty cycle time of 29.90 to 30.10 mS for ½ point
- Question 5: Energy score 5.4 or 5.5 and average current <= 1.0 uA (0.5pts for Energy Score & 0.5 pts for average current)
- Question 6: EM3 Period should be measured between 1.2 and 2.5 seconds for ½ point. The other half of point is if the on duty time proportionally has changed as the period from 30mS.

LETIMERn in EM0-EM2 greater than 1.9999 sec using the LFXO oscillator

- The LETIMERn is only a 16-bit counter, so to time greater than 2 seconds, the LETIMERn prescaler is required
- The prescaler can be programmed directly by writing to the appropriate CMU register
  - CMU->LFAPRESCO
  - The prescaler value is 4 bits in width and needs to be shifted 8-bits to the left to be placed in the proper position in the CMU_LFAPRESCO register

LETIMERn in EM0-EM2 greater than 1.9999 sec using the LFXO oscillator

- A good way to calculate the required prescaler value is the following:

```
Desired_Period = LETIMER0_period * LETIMER0_LFXO_count;
LETIMER0_prescaler = 0;
temp = Desired_Period / 1;
Prescaled_two_power = 1;
while (temp > LETIMER0_Max_Count){
        LETIMER0_prescaler++;
        Prescaled_two_power = Prescaled_two_power * 2;
        temp = Desired_Period / Prescaled_two_power;
}
```

- With LFA prescaler set, you must then adjust your times in the LETIMERn count and compare registers accordingly to the new effective LETIMER clock frequency

ULFRCO self-calibrating assignment

Objective: Develop the code for the Leopard Gecko to indicate whether the ambient is lit or in darkness using a passive light sensor, and to self-calibrate the ULFRCO oscillator for more accurate timing in EM3 state

Instructions:

1. Make any corrections to assignment #1 that are necessary

2. Develop a self-calibrating routine for the ULFRCO. This routine should have an accuracy of +/- 3%
   a. Enabling or disabling self-calibration should be done via a #define statement
   b. Only perform self-calibration once at the start of the program before entering sleep() for the first time

ULFRCO self-calibrating assignment

3. Using the ACMP0 peripheral and the Leopard Gecko STK3600 passive light sensor, turn on LED0 when the passive light sensor indicates darkness, and turn it off when the light sensor indicates light per the hysteresis defined below.
   a. Use the LETIMER0 in EM2 and then EM3 to Excite the Ambient Light Sensor and then to measure / decode the result
   b. Every 2.5 seconds, the Ambient Light Sensor should be excited and evaluated
   c. After the Ambient Light Sensor has been excited for a minimum of 4mS, but a short as possible, the Leopard Gecko should make a decision on whether LED0 should be turned ON indicating "darkness" or turned OFF indicating light
   d. The reference for the ACMP should be set to 2/63rds of the reference voltage to indicated darkness if the current state is lit.
      i. The value to set the darkness reference should be in the program's #defined statements so that experiments can be made using different references
   e. The reference for the ACMP should be set to 61/63rds of the reference voltage to indicated light if the current state is dark.
      i. The value to set the lit reference should be in the program's #defined statements so that experiments can be made using different references
   f. The program should go to sleep while the Ambient Light Sensor is not in use as well as between Ambient Light Sensor being excited and waiting to be measured / decoded

ULFRCO self-calibrating assignment

Questions:

In a separate document to be placed in the drop box with the program code, please answer the following questions:

1. With self-calibration disabled, what is the period of the LETIMER0 in EM3 using the Energy Profiler to measure once the program has begun to sense the Ambient Light Sensor?
2. With self-calibration enabled, what is the period of the LETIMER0 in EM3 using the Energy Profiler to measure once the program has begun to sense the Ambient Light Sensor?
3. What is the accuracy of the ULFRCO calibration compared to the target of 2.5s in percent?
4. Wil self-calibration enabled and in EM3, reset the session counter and wait 60 seconds. What is the Energy Profiler score of the Ambient Light Sensor program when it is indicating light, LED0 is OFF? What is the average current when no measurement is being taken?

ULFRCO self-calibrating assignment

5. Wil self-calibration enabled in EM3, reset the session counter and wait 60 seconds. What is the Energy Profiler score of the Ambient Light Sensor program when it is indicating darkness, LED0 turned ON? What is the average current when no measurement is being taken?

6. Does the LED0 turn on when the Ambient Light Sensor is blocked, mimicking darkness or the lights turned off?

7. Does the LED0 turn off when the Ambient Light Sensor is unblocked, mimicking lights being turned on?

Deliverables:

1. One document that provides the answers to Assignment #2

2. Another document that contains your .c project source file with LETIMER0 set to EM3 mode

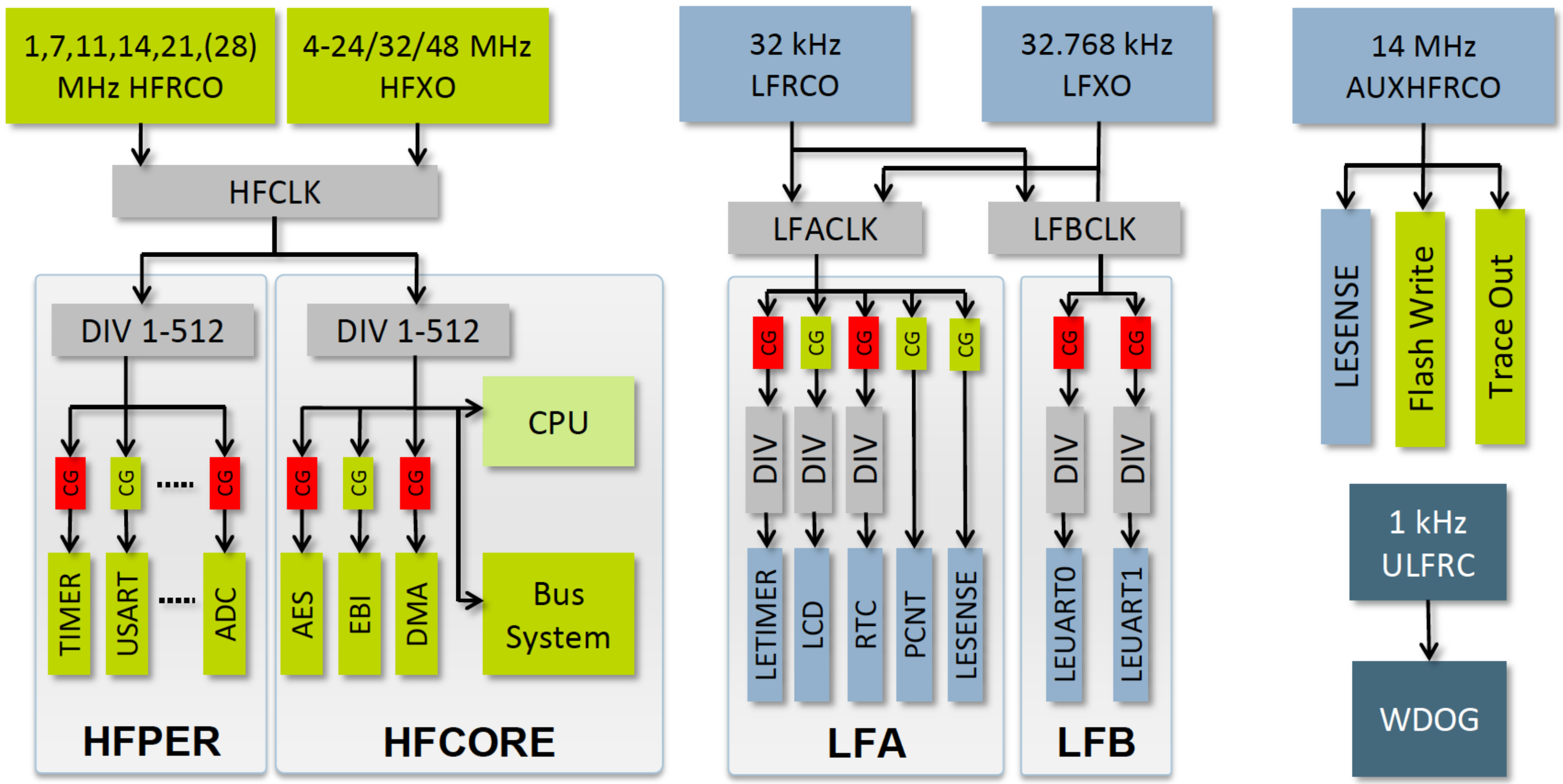
ULFRCO self-calibrating assignment

- BONUS: If you can beat the professors Energy Score on question 4 running on the same STK3600 dev kit, you will get 1 bonus points.

Clocks and Oscillators

ACMP is connected to the HFPER, which is always running in EM0, so what does this mean?

No need to enable an oscillator or to tie the oscillator to a clock branch



But, you still need to enable the clock to the ACMP!

ACMP – Warm Up

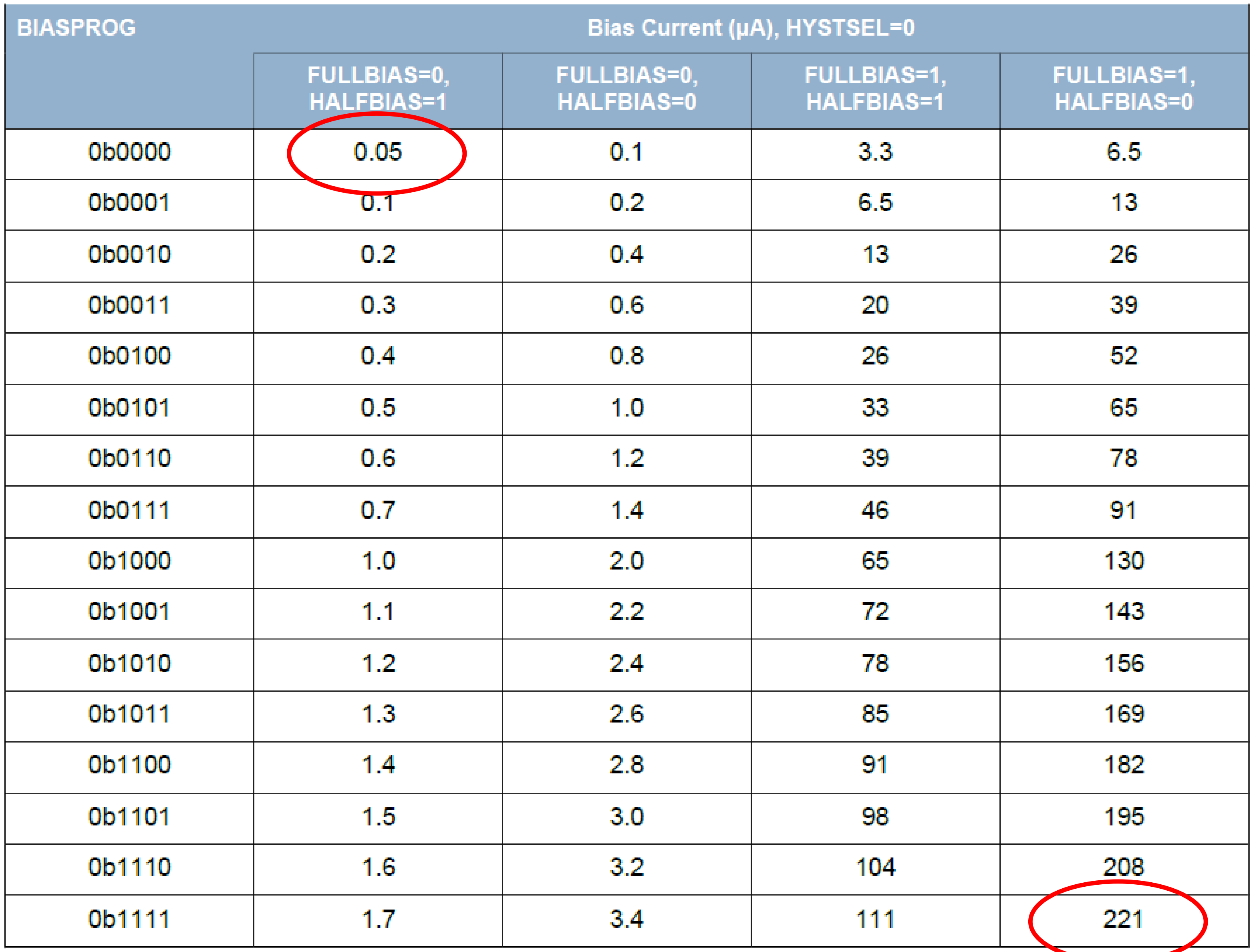
- When this Enable bit is set in the ACMPn_CTRL, the comparator must stabilize before becoming active and the outputs can be used. This time period is called the warm-up time.
- The warm-up time is a configurable number of peripheral clock (HFPERCLK) cycles, set in WARMTIME, which should be set to at least 10 µs but lengthens to up to 1ms if LPREF is enabled.
- The ACMP should always start in active mode and then enable the LPREF after warm-up time.
- One should wait until the warm-up period is over before entering EM2 or EM3, otherwise no comparator
  - When "warmed up," the ACMPACT bit in the ACMP STATUS register will become 1
- Interrupts will be detected. EM1 can still be entered during warm-up. After the warm-up period is completed, interrupts will be detected in EM2 and EM3.

ACMP – Response Time

- The delay from when the actual input voltage changes polarity, to when the output toggles is called the response time and can be altered by increasing or decreasing the bias current to the comparator through the BIASPROG, FULLBIASPROG and HALFBIAS fields in the ACMPn_CTRL.
- Setting the HALFBIAS bit in ACMPn_CTRL effectively halves the current. Setting a lower bias current will result in lower power consumption, but a longer response time.
- If the FULLBIAS bit is set, the highest hysteresis level should be used to avoid glitches on the output.

ACMP – Bias Current

Table 26.1. Bias Configuration



BIASPROG	Bias Current (µA), HYSTSEL=0			
	FULLBIAS=0, HALFBIAS=1	FULLBIAS=0, HALFBIAS=0	FULLBIAS=1, HALFBIAS=1	FULLBIAS=1, HALFBIAS=0
0b0000	0.05	0.1	3.3	6.5
0b0001	0.1	0.2	6.5	13
0b0010	0.2	0.4	13	26
0b0011	0.3	0.6	20	39
0b0100	0.4	0.8	26	52
0b0101	0.5	1.0	33	65
0b0110	0.6	1.2	39	78
0b0111	0.7	1.4	46	91
0b1000	1.0	2.0	65	130
0b1001	1.1	2.2	72	143
0b1010	1.2	2.4	78	156
0b1011	1.3	2.6	85	169
0b1100	1.4	2.8	91	182
0b1101	1.5	3.0	98	195
0b1110	1.6	3.2	104	208
0b1111	1.7	3.4	111	221

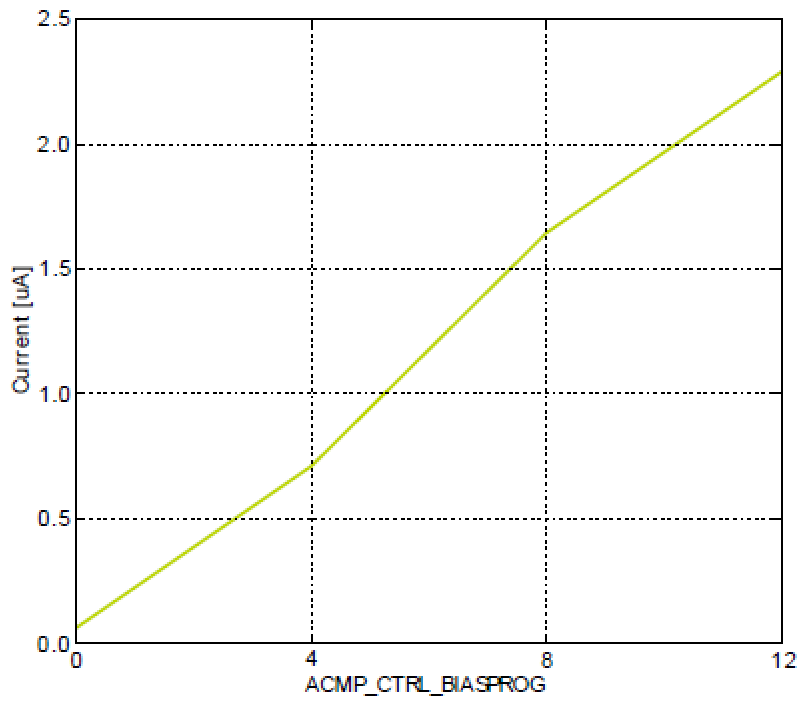
Silicon Labs' EFM32LG reference manual

ACMP - Hysteresis

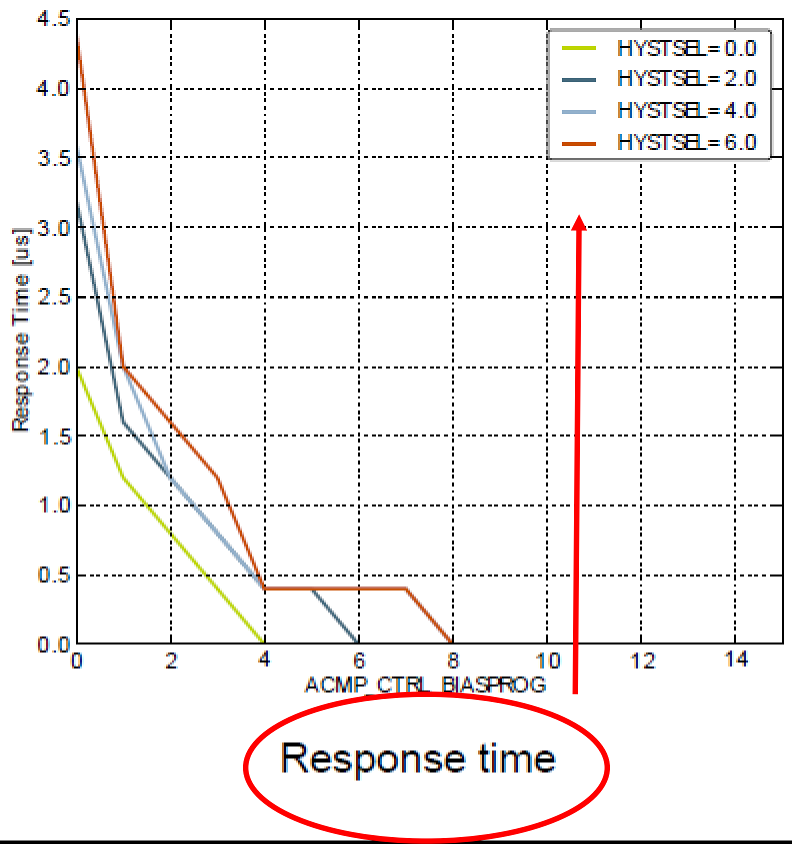
- The ACMP can be configured to 8 different hysteresis levels, including off which is level 0, through the HYSTSEL field in ACMPn_CTRL.
- When the hysteresis level is set above 0, the digital output will not toggle until the positive input voltage is at a voltage equal to the hysteresis level above or below the negative input voltage.
- This feature can be used to filter out uninteresting input fluctuations around zero and only show changes that are big enough to breach the hysteresis threshold.
- Note that the ACMP current consumption will be influenced by the selected hysteresis level and in general decrease with increasing HYSTSEL values.

ACMP hysteresis versus Response Time

Figure 3.37. ACMP Characteristics, Vdd = 3V, Temp = 25°C, FULLBIAS = 0, HALFBIAS = 1



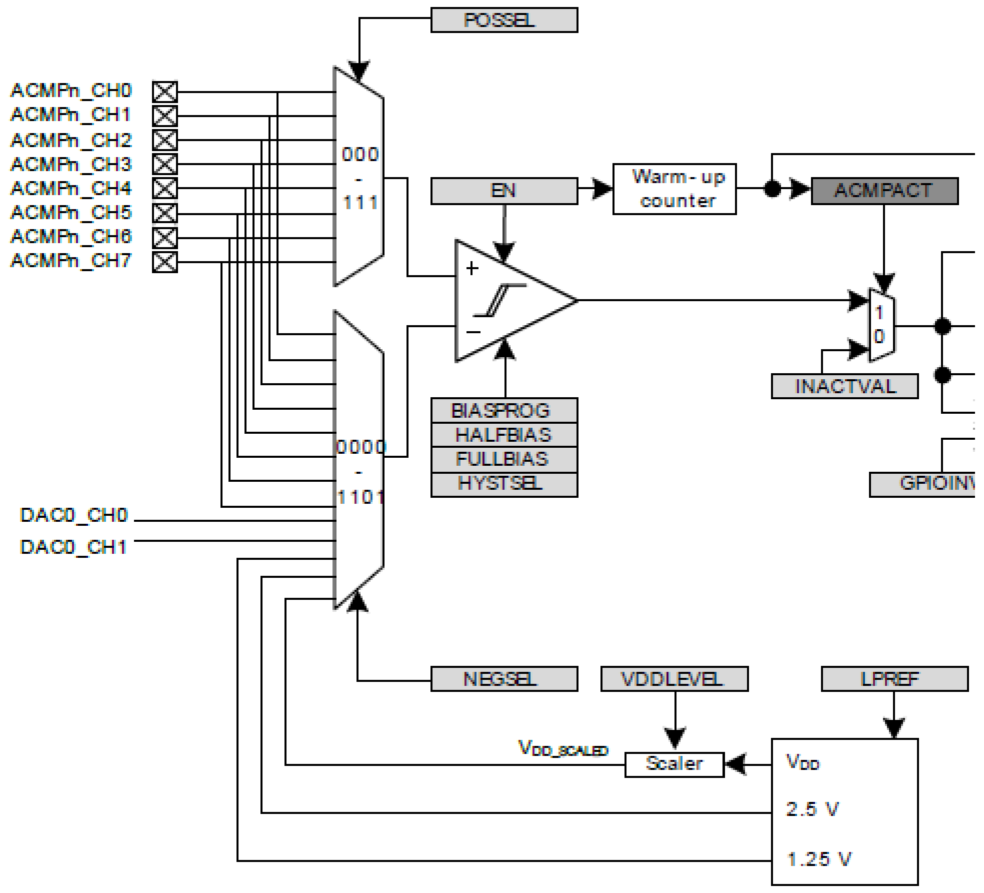
Current consumption, HYSTSEL = 4



Silicon Labs' EFM32LG data sheet

ACMP – Input Selection

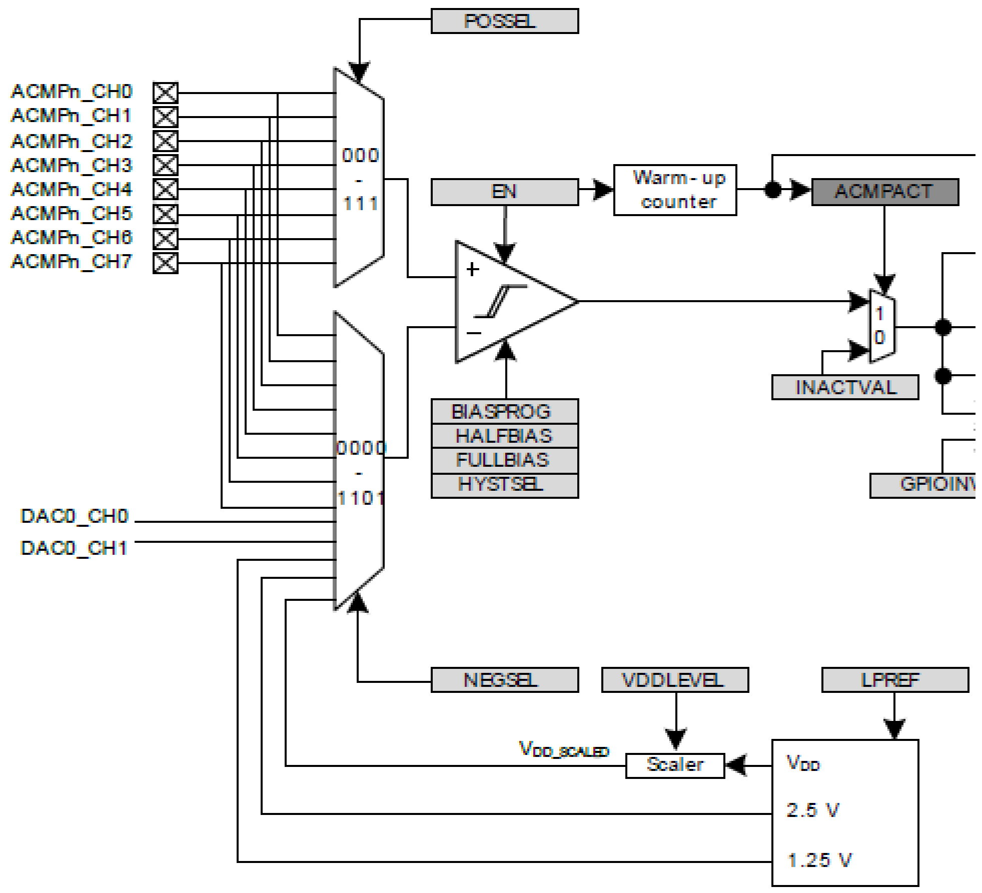
- The POSSEL and NEGSEL fields in ACMPn_INPUTSEL controls which signals are connected to the two inputs of the comparator. 8 external pins are available for both the negative and positive input.



Silicon Labs' EFM32LG reference manual

ACMP – Input Selection

- Important note that there are 5 negative inputs that can be set to compare an input to a fixed voltage available from the processor, 1.25 V bandgap, 2.5V bandgap,
- DAC channel 0, DAC channel 1, and VDD. The VDD reference can be scaled by a configurable factor,
- which is set in VDDLEVEL (in ACMPn_INPUTSEL) according to the following formula:



V_{DD} Scaled

$$V_{DD\_SCALED} = V_{DD} \times VDDLEVEL/63$$

ACMP – Low Power Reference

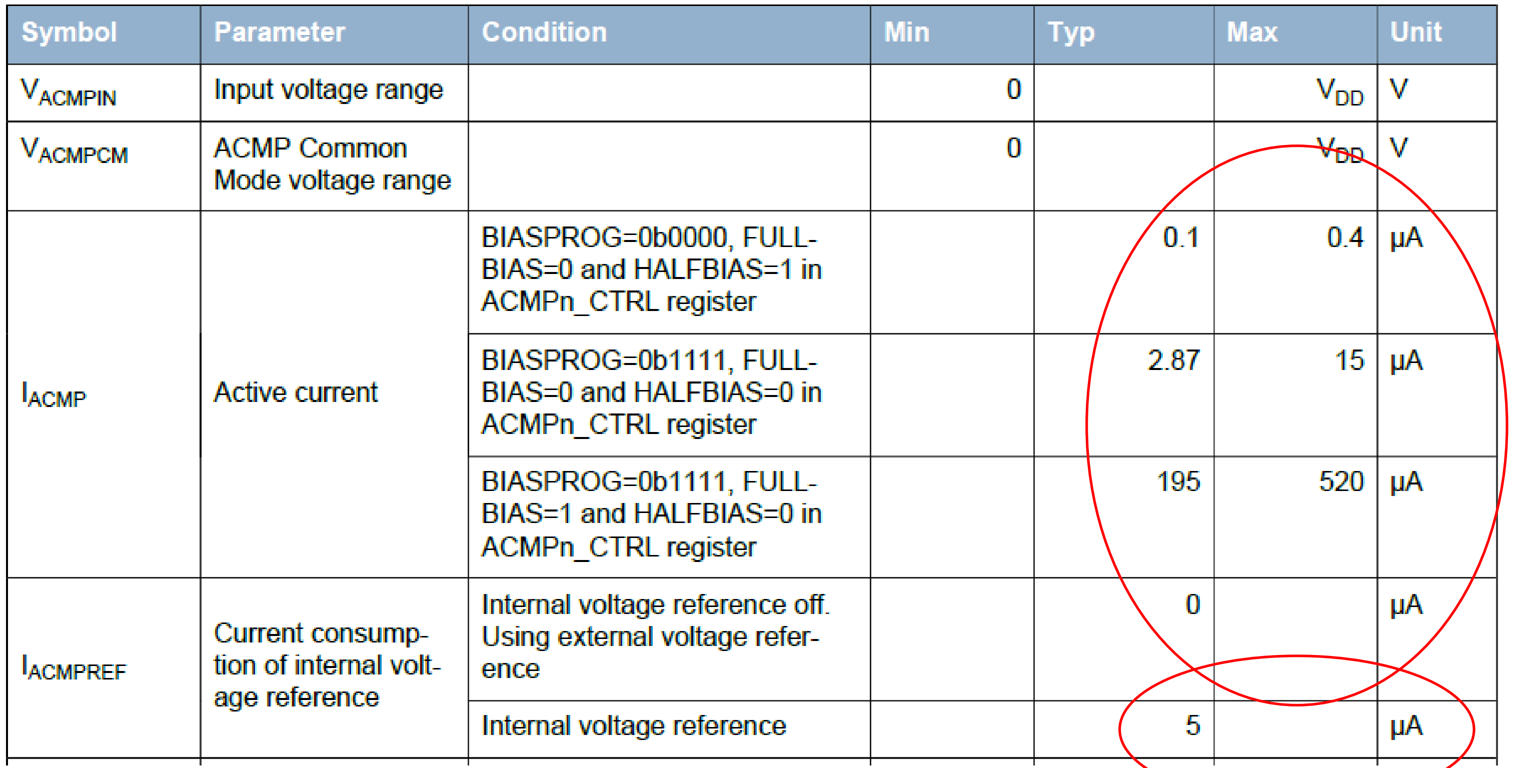
- A low power reference mode can be enabled by setting the LPREF bit in ACMPn_INPUTSEL.
- In this mode, the power consumption in the reference buffer (VDD and bandgap) is lowered at the cost of accuracy.
- Low power mode will only save power if VDD with VDDLEVEL higher than 0 or a bandgap reference is selected.

ACMP – Interrupts and PRS Out

- There are edge triggered interrupt flag (EDGE in ACMPn_IF).
- If either IRISE and/or IFALL in ACMPn_CTRL is set, the EDGE interrupt flag will be set on rising and/or falling edge of the comparator output, respectively.
- An interrupt request will be sent if the EDGE interrupt flag in ACMPn_IF is set and enabled through the EDGE bit in ACMPn_IEN.
- The analog comparator also includes an interrupt flag, WARMUP in ACMPn_IF, which is set when a warm-up sequence has finished
- The comparator output is also available as a PRS signal.

ACMP – Current requirements

Table 3.18. ACMP



Symbol	Parameter	Condition	Min	Typ	Max	Unit
V_{ACMPIN}	Input voltage range		0		V_{DD}	V
V_{ACMPCM}	ACMP Common Mode voltage range		0		V_{DD}	V
I_{ACMP}	Active current	BIASPROG=0b0000, FULL-BIAS=0 and HALFBIAS=1 in ACMPn_CTRL register		0.1	0.4	µA
		BIASPROG=0b1111, FULL-BIAS=0 and HALFBIAS=0 in ACMPn_CTRL register		2.87	15	µA
		BIASPROG=0b1111, FULL-BIAS=1 and HALFBIAS=0 in ACMPn_CTRL register		195	520	µA
$I_{ACMPREF}$	Current consumption of internal voltage reference	Internal voltage reference off. Using external voltage reference		0		µA
		Internal voltage reference		5		µA

Total ACMP Active Current

$$I_{ACMPTOTAL} = I_{ACMP} + I_{ACMPREF}$$

Setting up the ACMP

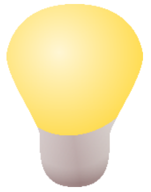
- First, the clock tree to the ACMP must be established
  - Without establishing the clock tree, all writes to the ACMPn registers will not occur
  - Pseudo code in the CMU setup routine to enable the ACMPn clock tree:
    - Lastly, enable the GPIO clocking using the CMU_ClockEnable for the ACMPn

Setting up the ACMP

- Second, the ACMP must be set up
  - Inputs to the ACMP must be specified
    - Specify the Positive Input source to the Analog Comparator
      - If external to the MCU, the appropriate GPIO pin must be configured
      - ACMP external GPIO pins should be set to gpioModeDisabled
      - You will need to look at the board schematic to determine which positive channel to use
    - Specify the Negative Input source to the Analog Comparator
      - What is the scaling of the reference?
    - Set up ACMP channels using the ACMP_ChannelSet library routine
  - What bias level to configure the Analog Comparator
    - Higher the bias level, the more quickly the comparison
    - Set to lowest bias level to achieve application goals to enable lowest power
  - Set the Hysteresis
    - The higher the hysteresis, generally lower power
  - Program the functionality of the ACMP using ACMP_init()

Setting up the ACMP

- Third, the ACMP interrupts must be enabled if needed
  - Clear all interrupts from the ACMP to remove any interrupts that may have been set up inadvertently by accessing the ACMPn->IFC register or the emlib routine
  - Enable the desired interrupts by setting the appropriate bits in ACMPn->IEN
  - Set BlockSleep mode to the desired Energy Mode
    - ACMP only uses HFPERCLK to write to registers and not required to operate
    - ACMP can be set to operate down to EM3
  - Enable interrupts to the CPU by enabling the ACMP in the Nested Vector Interrupt Control register using NVIC_EnableIRQ(ACMPn_IRQn);



Setting up the ACMP

- Fourth, the ACMPn interrupt handler must be included
  - Routine name must match the vector table name:

    ```
    Void ACMPn_IRQHandler(void) {
    }
    ```

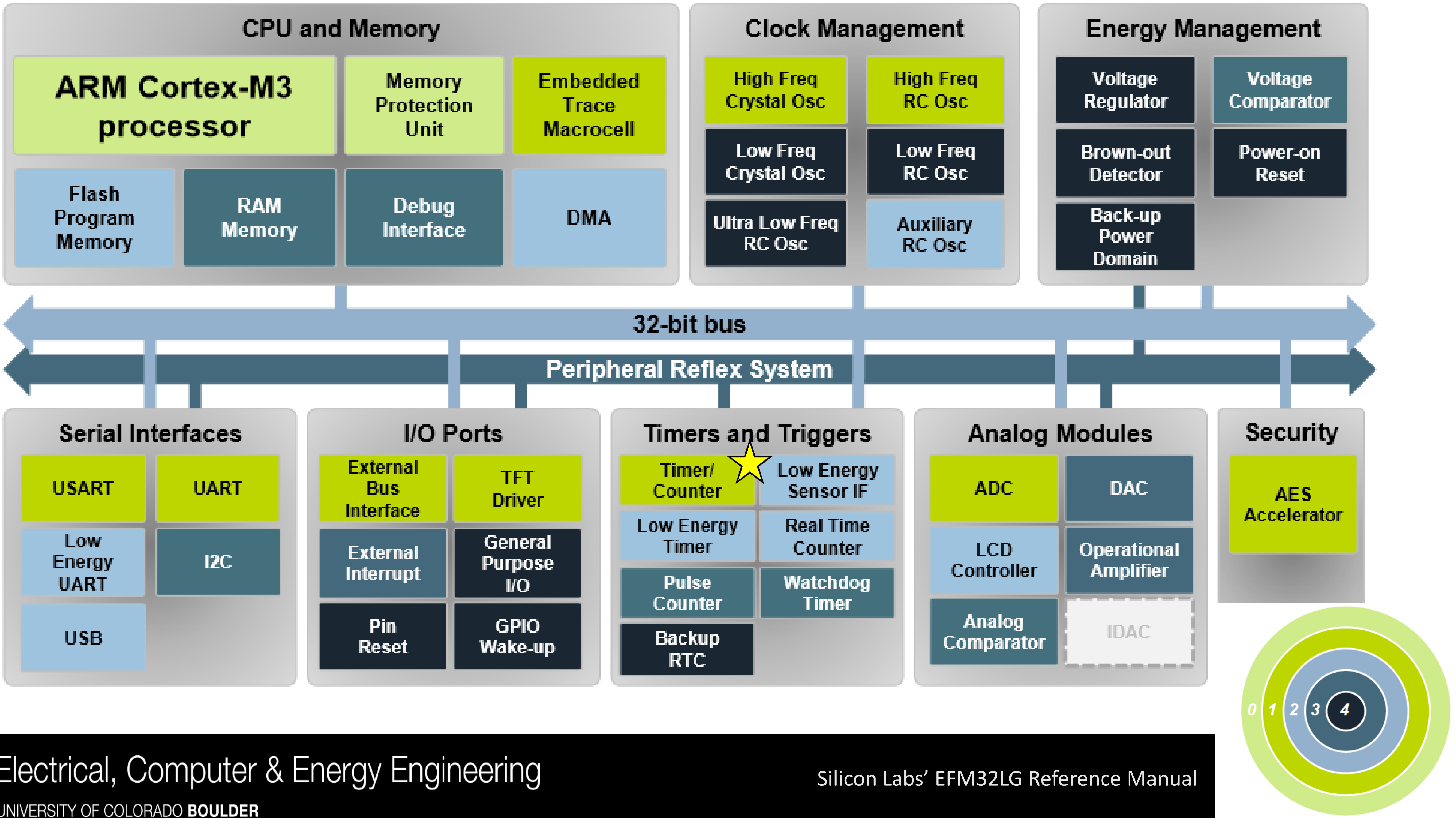
  - Inside this routine, you add the functionality that is desired for the ACMPn interrupts

Calibrating ULFRCO

- What to you require to calibrate the ULFRCO?
  - A know precision time base running for a specific number of ticks
    - HFXO oscillator
  - A "third party" reference that can be used to compare:
    - ULFRCO
    - LFXO

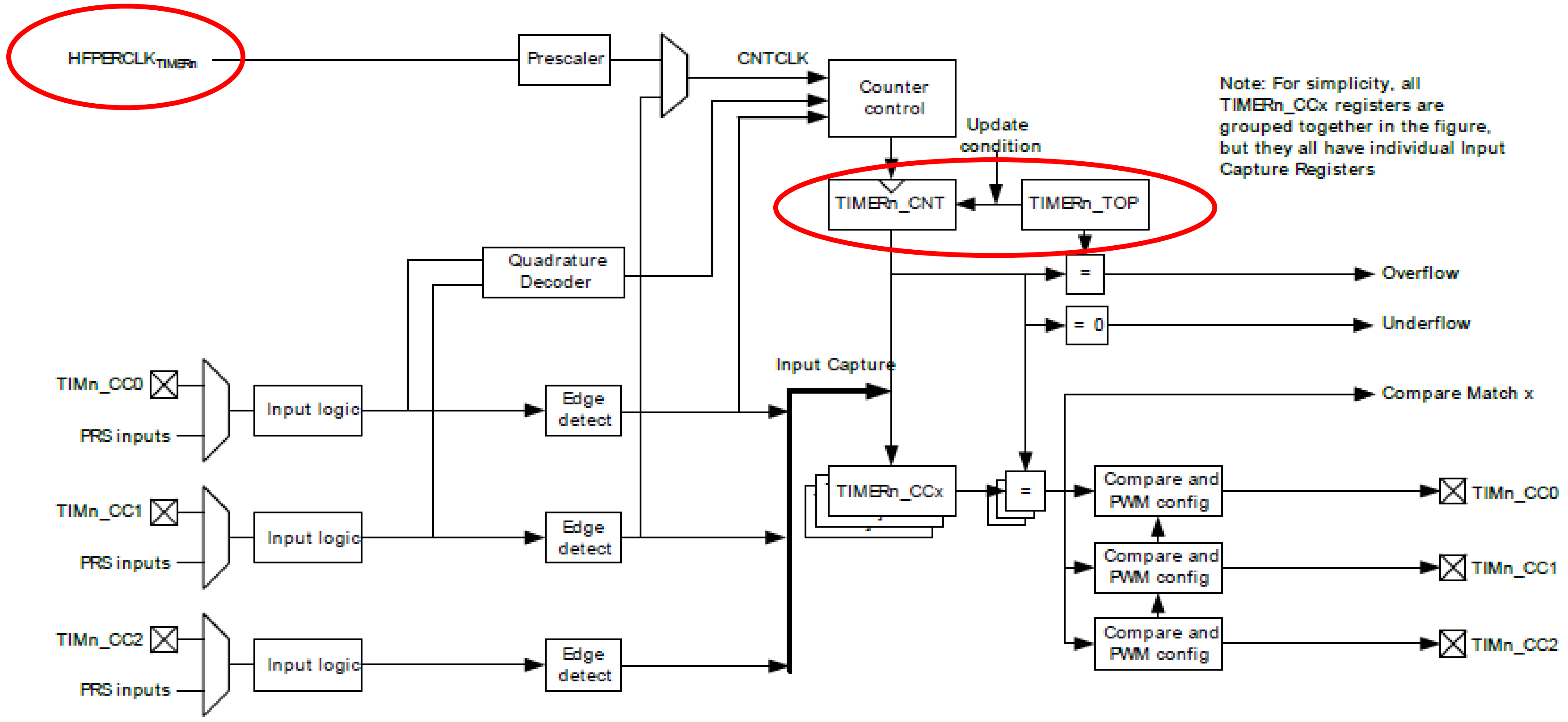
TIMERn peripherals

  - And, the ULFRCO running for the "equivalent" number of ticks
    - "Equivalent" is used since each oscillator has a different number of ticks per second
  - Perform a ratio to the "third party" reference to adjust the number of ULFRCO ticks per second
    - If the ULFRCO is not running the 1,000 ticks per second, then the "calibrate" ticks per second must be used as the time base


CPU and Memory
ARM Cortex-M3 processor
Memory Protection Unit
Embedded Trace Macrocell
Flash Program Memory
RAM Memory
Debug Interface
DMA
Clock Management
High Freq Crystal Osc
High Freq RC Osc
Low Freq Crystal Osc
Low Freq RC Osc
Ultra Low Freq RC Osc
Auxiliary RC Osc
Energy Management
Voltage Regulator
Voltage Comparator
Brown-out Detector
Power-on Reset
Back-up Power Domain
32-bit bus
Peripheral Reflex System
Serial Interfaces
USART
UART
Low Energy UART
I2C
USB
I/O Ports
External Bus Interface
TFT Driver
External Interrupt
General Purpose I/O
Pin Reset
GPIO Wake-up
Timers and Triggers
Timer/ Counter
Low Energy Sensor IF
Low Energy Timer
Real Time Counter
Pulse Counter
Watchdog Timer
Backup RTC
Analog Modules
ADC
DAC
LCD Controller
Operational Amplifier
Analog Comparator
IDAC
Security
AES Accelerator
0
1
2
3
4

Timer0

Figure 20.1. TIMER Block Overview

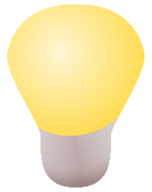
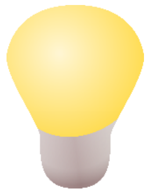


Setting up TIMER0

- 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

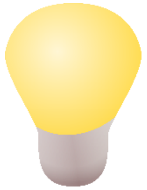
Setting up the TIMERn

- 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
    - 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

Calibrating ULFRCO

- Suggestions for calibrating the ULFRCO
  - Create routine that is ran after generic CMU set up routine
  - From LETIMERn calibration routine, call TIMER0 and TIMER1 setup routines based on the calibration requirements
    - Timers must be in sync
    - And, combined to become a 32-bit counter
    - Timers should be configured to be up counters
    - Disable Timers
    - Initialize the Timer count registers to be zero
  - Set up LETIMER to be one-shot and using LFXO
    - Count should be loaded with 1 second, 32768
  - Enable LETIMER and TIMERs
  - Poll LETIMER0 Count register and stop TIMER registers when Count = 0

Calibrating ULFRCO (continue)

- Suggestions for calibrating the ULFRCO
  - Save 32 bit Timer count value into a variable, LFXO count
  - Re-initialize Timer counter registers to 0
  - Set up LETIMER to be one-shot and using ULFRCO
    - Count should be loaded with 1 second, 1000
  - Enable LETIMER and TIMERs
  - Poll LETIMER0 Count register and stop TIMER registers when Count = 0
  - Save 32 bit Timer count value into a variable, ULFRCO count
- Now divide LFXO count by ULFRCO count and save in Osc ratio
  - If ULFRCO is running faster than 1000 Hz, Osc ratio will be less than 1
  - If ULFRCO is running slower than 1000 Hz, Osc ratio will be greater than 1

Calibrating ULFRCO (continue)

- The Osc ratio value will be used to adjust the ULFRCO count value per second
  - ULFRCO count per second value = 1000 * Osc ratio
- This modified ULFRCO ticks per second value should now be the basis of timing calculations using the ULFRCO
- Return to normal setup procedures including the LETIMER0_Setup() routine for the program which will now be accessing the modified ULFRCO ticks per second if going into EM3 mode

Beware of Integer Math

Firmware Best Practices

- Int a;
- Are these three expression equivalent in c-code? NO

a = (5/2) * 4;	a = 5 * (4/2);	a = (5 * 4) / 2;
a = (2) * 4;	a = 5 * 2;	a = 20 / 2;
a = 8;	a = 10;	a = 10;

Integer Math

Firmware Best Practices

- Another example
- In “real” math, (5/2) * 3 = 7.5
- In Integer math, ordering matters towards precision

a = (5/2) * 3;	a = 5 * (3/2);	a = (5 * 3) / 2;
a = (2) * 3;	a = 5 * 1;	a = 15 / 2;
a = 6;	a = 5;	a = 7;

If staying strictly in Integer Math, always do the multiplication first. Will limit the loss of accuracy