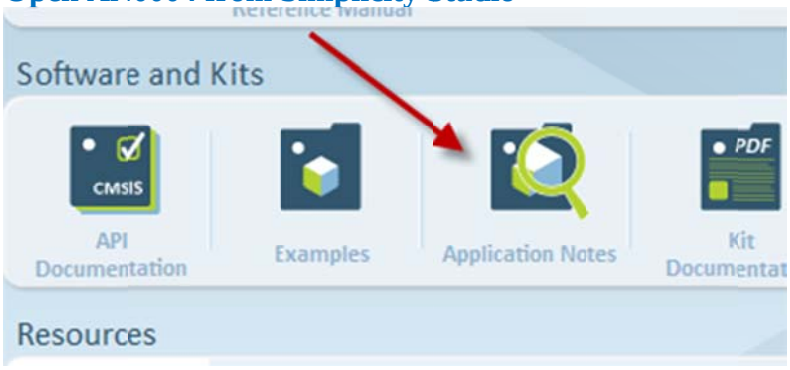
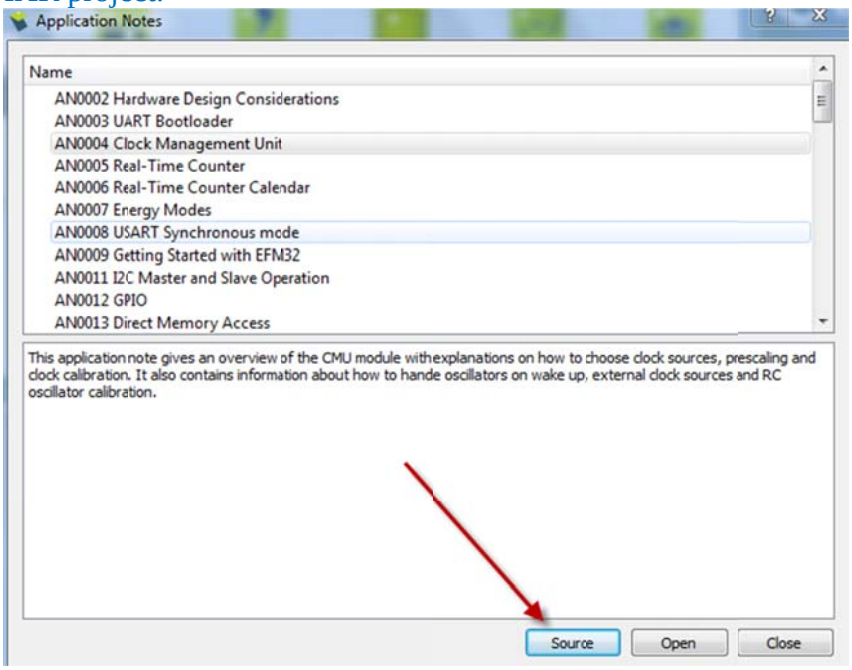
Scripps/Energy Micro Hands-On Training
Frank Roberts
February 20, 2013

Clock Management Unit(AN0004 CMU): Demo for clock switching: Add in custom functionality for PB0/PB1, LCD Functionality, Port Toggle for measuring HFXO startup time, modify HFXO Timeout via CMU Control register.

1) **Open AN0004 from Simplicity Studio**



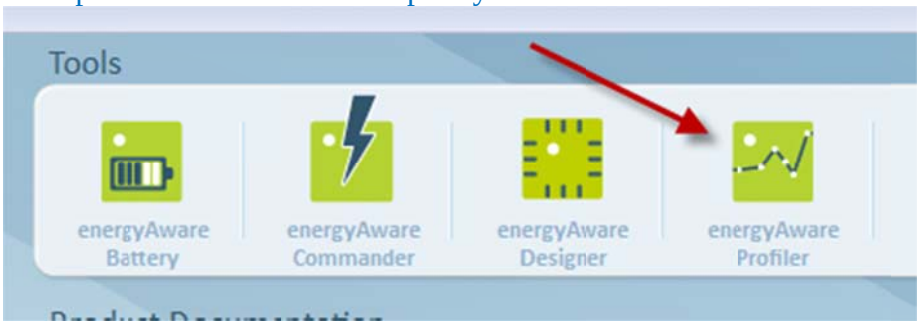
Click 'Source' button when highlighting AN0004. If you have installed IAR Kickstart it will open the IAR project.



2) **Let's Review the Project and code for changing clocks, Review Project Settings**

3) **Add SWO function to be able to use AE Profiler**

a. Open EA Profiler from Simplicity Studio



b. Copy the ***void setupSWO(void)*** function from EA Profiler/Code View and paste into the top of your project.

c. Add the function prototype at the top of main_cmu_conf.c

d. In main() after CHIP_Init(); add the setupSWO(); function call to configure the SWO output.

e. Compile the project.

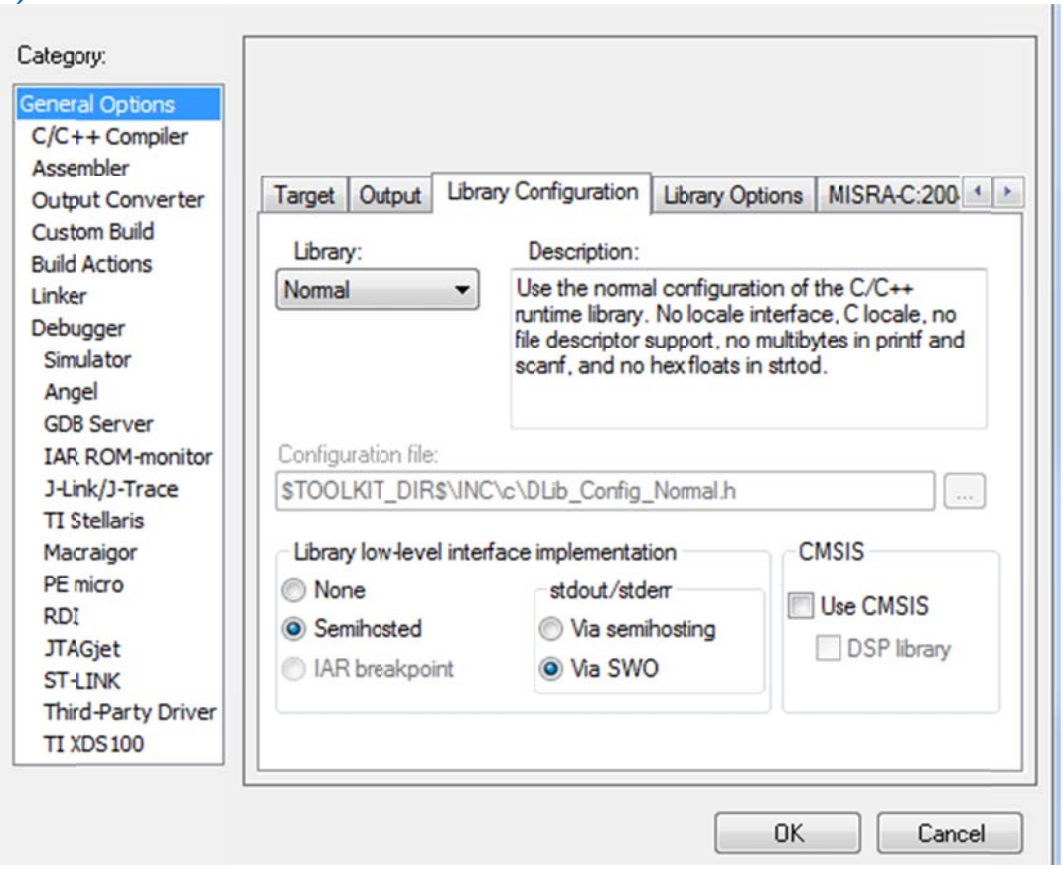
f. Now we will add printf() statements in various places in the firmware for debug checks. Let's place Printf statements inside of the while(1) to indicate the count value and indicate the clock frequency changes as shown below.

```
if (count > 5 && !prescChange)
{
  printf("Count =  %d\n", count);
  printf("HFRCO = 14MHz\n");
  /* Remove the prescaling for the HFPERCLK,
   * TIMER now running at 14Mhz */
  CMU_ClockDivSet(cmuClock_HFPER, cmuClkDiv_1);
  /* signal that prescaler has been changed
   * so it doesn't go inside the if clause again */
  prescChange = true;
}
/* If 10 togglings occured and banf
 * hans't changed */
else if (count > 10 && !bandChange)
{
  /* Change HFRCO band to 21Mhz */
  CMU_HFRCOBandSet(cmuHFRCOBand_21MHz);
  /* signal that band has been changed to 21Mhz
   * so it doesn't go inside the if clause again */
  bandChange = true;
  printf("Count =  %d\n", count);
  printf("HFRCO = 21MHz\n");
  __NOP();
}
/* If 15 togglings occured and crystal
 * hans't been enabled */
else if (count > 15 && !xtalChange)
{
  /* Enable HFXO without waiting for HFXORDY */
  CMU_OscillatorEnable(cmuOsc_HFXO, true, false);
  /* signal that the crystal (HFXO) has been enables
   * so it doesn't go inside the if clause again */
  xtalChange = true;
  printf("Count =  %d\n", count);
  printf("HFRCO = 48MHz\n");
  __NOP();
}
```

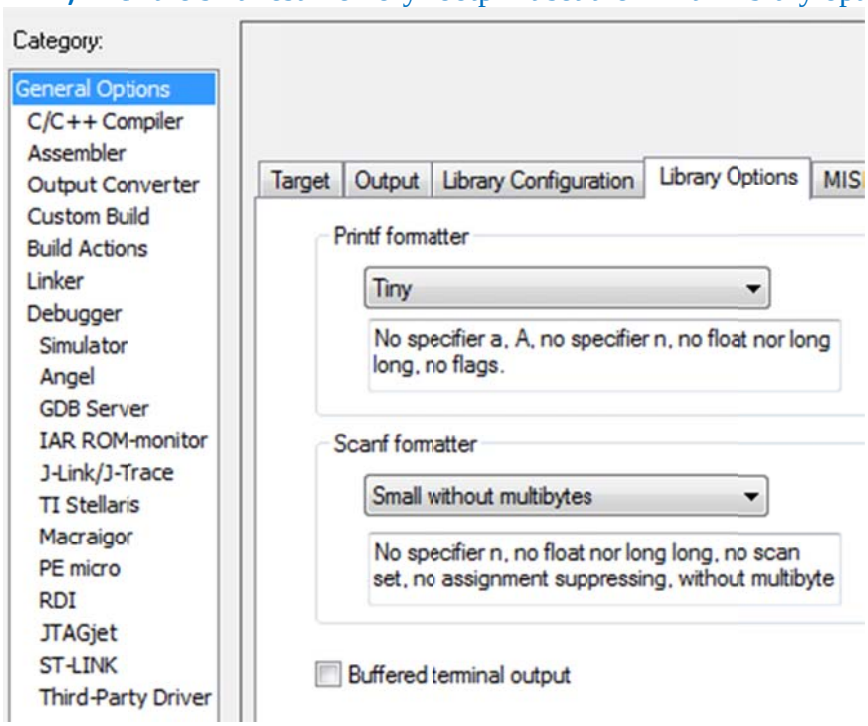
g. Now we need to adjust our project settings to have this work. ***Right-click on the Project and choose Options…***

Do the following…

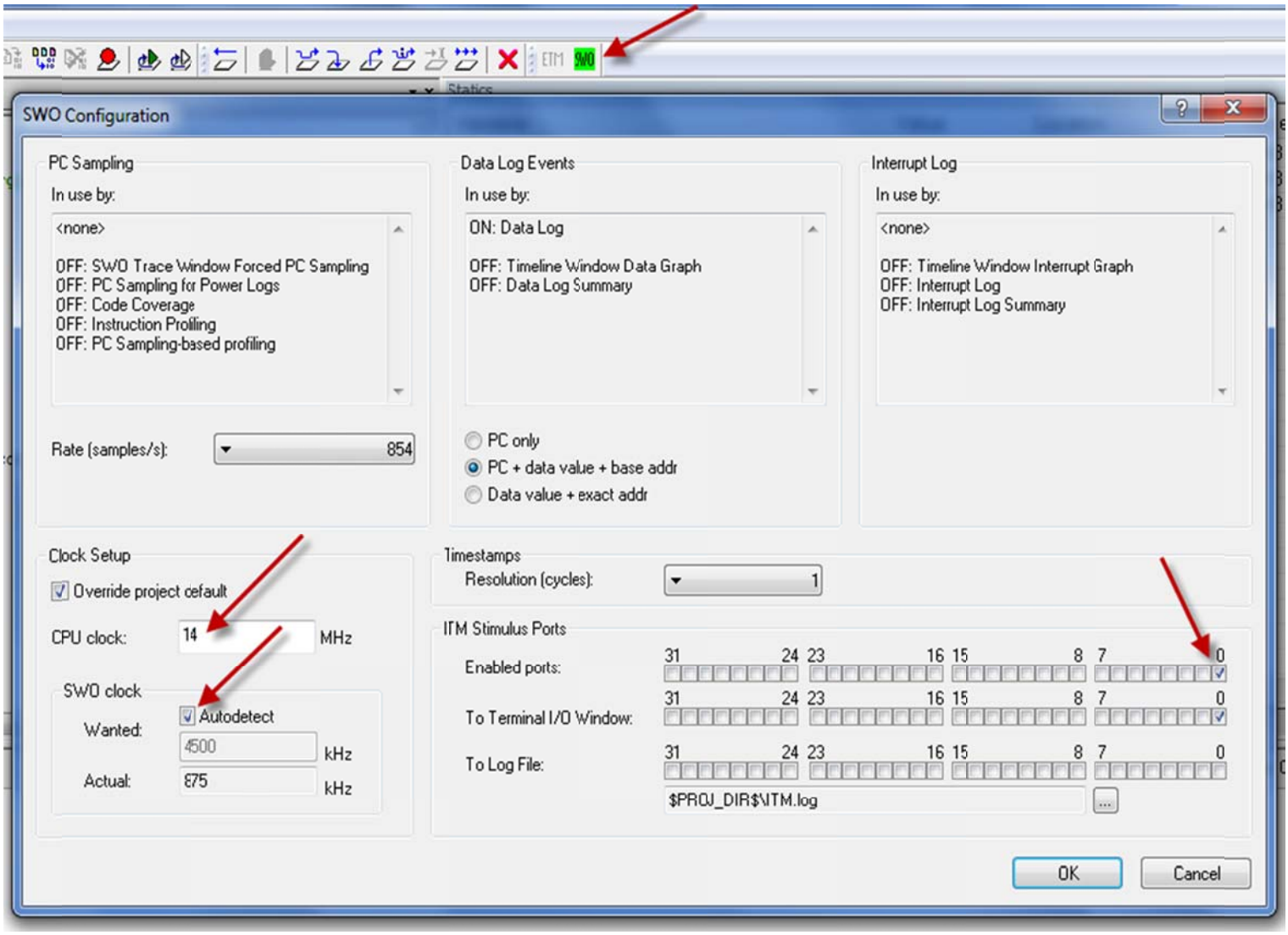
1)



2) For the smallest memory footprint set the Printf Library options as shown below.



3) After Downloading project to the board you will need to configure the SWO settings to ensure the following…
   a. The correct SWO Channel is enabled Channel 0 in this case
   b. The correct Processer speed is provided in this example 14MHz from the Internal OSC
   c. Select AutoDetect for SWO



4) **Add in GPIO Push-button control, call *GPIO_IRQInit()***

a. Open EFM32 Train_Functions.c

b. Copy ***GPIO_IRQInit()***, GPIO Interrupt functions into main_cmu.c

```
// ***************************************************************
void GPIO_ODD_IRQHandler(void)
{
  GPIO_IntClear(1 << 9);
}

//****************************************************************
// * @brief GPIO Interrupt handler (PB10)
// ***************************************************************
void GPIO_EVEN_IRQHandler(void)
{
  GPIO_IntClear(1 << 10);
}

//****************************************************************
// * @brief Initialize GPIO interrupt PB9/PB10
// ***************************************************************
void GPIO_IRQInit(void)
{
  // Enable GPIO in CMU
  CMU_ClockEnable(cmuClock_GPIO, true);

  // Configure PB9 and PB10 as input for PB0/1
  GPIO_PinModeSet(gpioPortB, 9, gpioModeInput, 0);
  GPIO_PinModeSet(gpioPortB, 10, gpioModeInput, 0);

  // Set falling edge interrupt for both ports
  GPIO_IntConfig(gpioPortB, 9, false, true, true);
  GPIO_IntConfig(gpioPortB, 10, false, true, true);

  // Enable interrupt in core for even and odd gpio interrupts
  NVIC_ClearPendingIRQ(GPIO_EVEN_IRQn);
  NVIC_EnableIRQ(GPIO_EVEN_IRQn);

  NVIC_ClearPendingIRQ(GPIO_ODD_IRQn);
  NVIC_EnableIRQ(GPIO_ODD_IRQn);
}
```

c. Add ***GPIO_IRQInit()*** function call in ***main()*** as shown below…

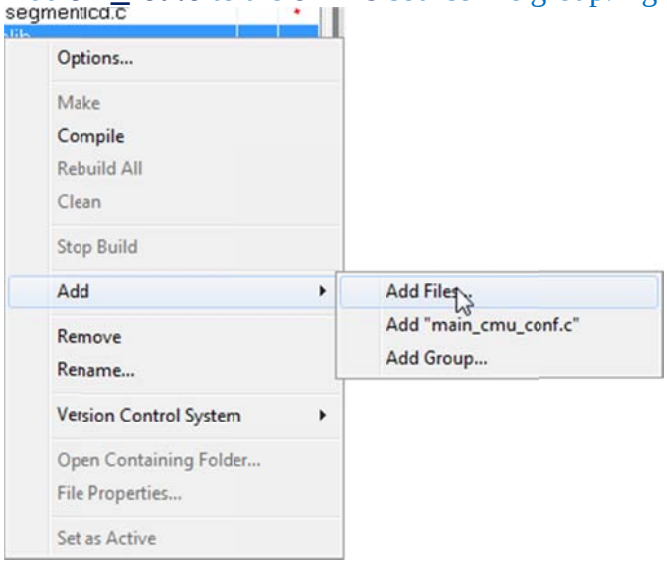
```
/**************************************************************************
int main(void)
{
  /* Initialize chip */
  CHIP_Init();

  /* Prescale the HFPERCLK -> HF/2 = 14/2 = 7Mhz */
  CMU_ClockDivSet(cmuClock_HFPER, cmuClkDiv_2);

  /* Enable clock for GPIO module */
  CMU_ClockEnable(cmuClock_GPIO, true);

  GPIO_IRQInit();
```

d. Now test Push-button functionality by setting breakpoints in the GPIO_ODD/EVEN Interrupts. **Double click on the line to set the breakpoint**.

```
62 //***************************************************************
63 // * @brief GPIO Interrupt handler (PB9)
64 // ***************************************************************
65 void GPIO_ODD_IRQHandler(void)
66 {
67   GPIO_IntClear(1 << 9);
68 }
69
70 //***************************************************************
71 // * @brief GPIO Interrupt handler (PD10)
72 // ***************************************************************
73 void GPIO_EVEN_IRQHandler(void)
74 {
75   GPIO_IntClear(1 << 10);
76 }
```

5) Add in LCD support files:

Add ***em_lcd.c*** to the **emLib** source file group: right-click on folder and choose: Add->Add Files…

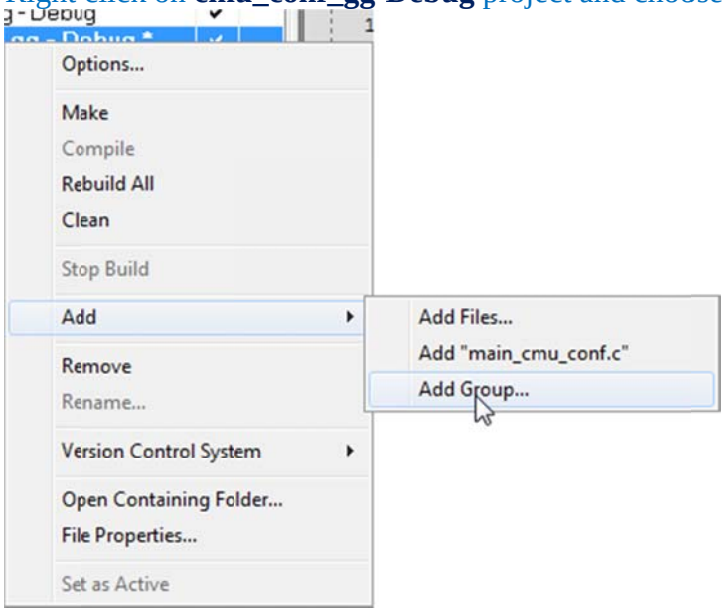


Browse to the directory below…
C:\Users\'YOUR NAME'\AppData\Roaming\energymicro\emlib\src

and choose ***em_lcd.c.***

a. Add ***segmentlcd.c*** to the project as shown below.

Right click on **cmu_conf_gg-Debug** project and choose: Add->Add Group…

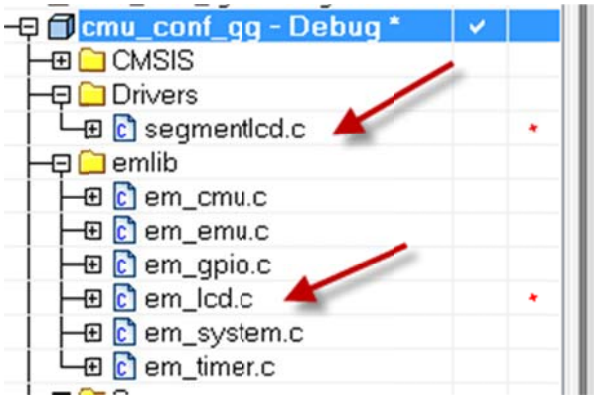


Label new group as **Drivers.**

Right click on the Drivers group and choose: Add->Add Files… as you did earlier

Browse to the directory below…
C:\Users\'YOUR NAME'\AppData\Roaming\energymicro\kits\common\drivers

and choose ***segmentlcd.c.*** Your project should now look like what is shown below.



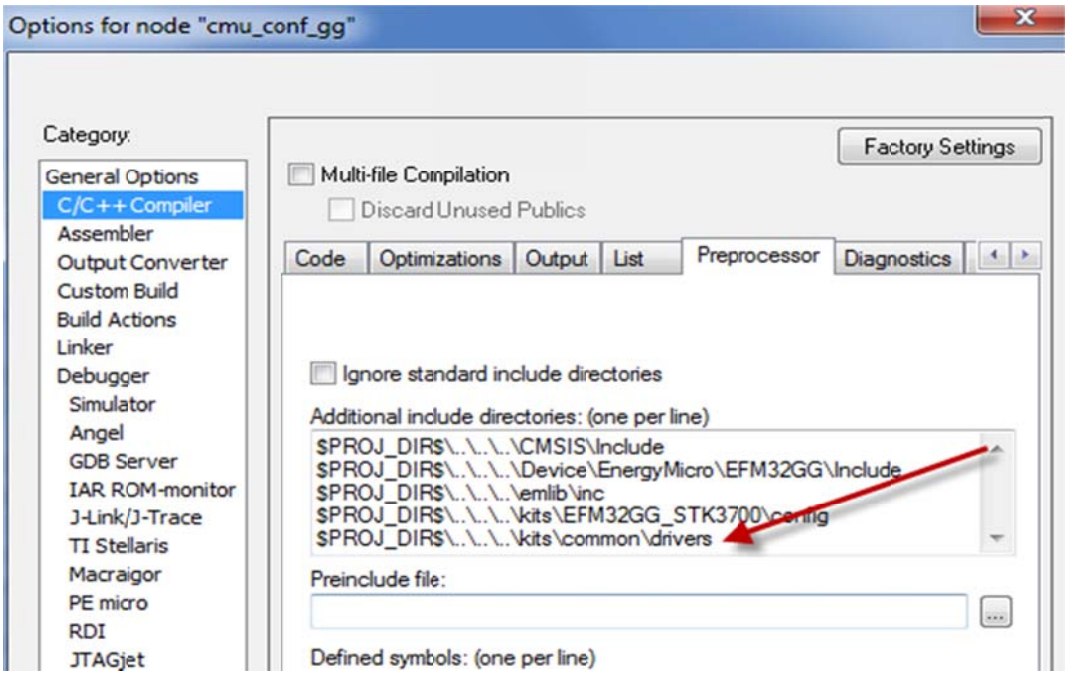
6) **Add LCD Initialization Function:** ***SegmentLCD_Init(false);***
After *CMU_ClockEnable(cmuClock_TIMER0, true);*

```
int main(void)
{
  /* Initialize chip */
  CHIP_Init();

  /* Prescale the HFPERCLK -> HF/2 = 14/2 = 7Mhz */
  CMU_ClockDivSet(cmuClock_HFPER, cmuClkDiv_2);

  /* Enable clock for GPIO module */
  CMU_ClockEnable(cmuClock_GPIO, true);

  GPIO_IRQInit();

  /* Enable clock for TIMER0 module */
  CMU_ClockEnable(cmuClock_TIMER0, true);

  SegmentLCD_Init(false);
```

7) **Add LCD Write Functoin:**
SegmentLCD_Write("EM1"); After *EMU_EnterEM1();* in main **while(1)** as shown below.

```
while (1)
{
  /* Go to EM1 */
  EMU_EnterEM1();
  SegmentLCD_Write("EM1");
```

8) **Compile the Project now.** You will get a compile error. Why? Keep in mind we have to ensure the compiler knows where the respective header files such as **segmentlcd.h** are that are referenced from the source files we added.
Why do we not need to do this for em_lcd.h? To resolve this add the following line in the Preprocessor tab



9) **Add in Clock Outputs to GPIO pins.** Want to observe LFXO on CLK1 and HFCLK2 on CLK0

a. Looking at the GG990F1024 Datasheet Section 4.2 Alternate functionality pinout we see the following

Alternate	LOCATION							
Functionality	0	1	2	3	4	5	6	Description
CMU_CLK0	PA2		PD7					Clock Management Unit, clock output number 0.
CMU_CLK1	PA1	PD8	PE12					Clock Management Unit, clock output number 1.

Based on reviewing the STK Schematic we decide we can use PD7 for CLK0 and PD8 for CLK1.

CLK0/CLK1 have multiple clock options that can be output. Refer to the reference manual for the available options on each of these. These options are all set in the CMU_CTRL register. To have CLK0/CLK1 output to PD7/PD8 insert the code below into your project.

Put this section of code after the ***SegmentLCD_Init(false);*** function in ***Main()***

```
//-------------------------------------------------------------//
  /* Starting LFXO and waiting until it is stable */
  CMU_OscillatorEnable(cmuOsc_LFXO, true, true);
  /* Enabling clock to the interface of the low energy modules */
//  CMU_ClockEnable(cmuClock_CORELE, true);

  CMU->CTRL |= CMU_CTRL_CLKOUTSEL1_LFXO;              //Configure CLKOUTSEL1 to be LFXO
  GPIO_PinModeSet(gpioPortD, 8, gpioModePushPull, 0);   //Config PD8 as Output for LFXO,
  CMU->ROUTE |= (CMU_ROUTE_CLKOUT1PEN | CMU_ROUTE_LOCATION_LOC1);

  CMU->CTRL |= CMU_CTRL_CLKOUTSEL0_HFCLK2;
  GPIO_PinModeSet(gpioPortD, 7, gpioModePushPull, 0);
  CMU->ROUTE |= (CMU_ROUTE_CLKOUT0PEN | CMU_ROUTE_LOCATION_LOC2);
//-------------------------------------------------------------//
```

Energy Modes: Use EFM32TG_STK3300 Emode Example. Find in Simplicity Studio and open click 'source' button Project is already configured for TG-STK

1) Notate how various energy modes are entered
2) Use Energy Profiler to view current waveforms for various modes
3) Enable PORTD.2 as output: *GPIO_PinModeSet(gpioPortD, 2, gpioModePushPull, 0);* add after *gpioSetup()* in *main()*
4) Using *case 5:* EM2 + RTC add in *GPIO_PinOutToggle(gpioPortD, 2);* to observe Sleep/Wake time in while(1) loop in case 5:

```
while (1)
  {
   GPIO_PinOutToggle(gpioPortD, 2);
   RTC_Trigger(2000, NULL);
   EMU_EnterEM2(false);
  }
```

5) Why is the wakeup time much shorter than the expected 2seconds? Hint: System Tick timer
6) How can we clear the Systick timer when we enter case 5:?

Low Energy Timer: AN0026 Low Energy Timer-> Open from Simplicity Studio 'Appnotes'->AN0026 'Source'

PART I: PWM_PULSE

1) Modify Project for EFM32TG840F32
   a) *Right Click on letimer_pwm_pulse-Debug* choose *Options…*
   b) General Options -> Change to TG840F32
   c) C/C++ Compiler: Preprocessor \EFM32TG\Include, Defined Symbols: EFM32TG840F32
   d) Linker: Change to EFM32TG840F32.icf
   e) Add startup_efm32tg.s and system_efm32tg.c in CMSIS folder
      C:\Users\Frank Roberts\AppData\Roaming\energymicro\Device\EnergyMicro\EFM32TG\Source\IAR
      C:\Users\Frank Roberts\AppData\Roaming\energymicro\Device\EnergyMicro\EFM32TG\Source
2) Notate how LETimer is configured
3) Look at Outputs on PD6(PWM) and PD7(Pulse) outputs
4) Add Energy Profiler capability: Open EA Profiler from Simplicity Studio, copy the function below to the top of your main_letimer_pwm_pulse

```
void setupSWO(void)
{
uint32_t *dwt_ctrl = (uint32_t *) 0xE0001000;
uint32_t *tpiu_prescaler = (uint32_t *) 0xE0040010;
uint32_t *tpiu_protocol = (uint32_t *) 0xE00400F0;
CMU->HFPERCLKEN0 |= CMU_HFPERCLKEN0_GPIO;
/* Enable Serial wire output pin */
GPIO->ROUTE |= GPIO_ROUTE_SWOPEN;
#if defined(_EFM32_GIANT_FAMILY
.
.
.
}
```

Now add the *setupSWO()* call in main() after CHIP_Init()

You now have Serial Wire Output capability that can be used for all sorts of debug functionality included with the Jlink Debugger.

5) Compile/Download w/IAR and disconnect, now hit the 'PLAY' button on EA Profiler, Hit Reset on the STK.
6) Can LETIMER be ran in EM3? How?
Modify the following instruction to have the LFA Clock branch run from the ULFRCO 1kHz clock source

```
void LETIMER_setup(void)
{
 /* Enable necessary clocks */
 CMU_ClockSelectSet(cmuClock_LFA, cmuSelect_ULFRCO); //cmuSelect_LFXO);
 Adjust COMPMAX to 100 to compensate for a slower clock
```

7) Add the following instruction in the while loop:

```
 while(1)
 {
  CMU_ClockEnable(cmuClock_CORELE, false);
  /* Go to EM2 */
  EMU_EnterEM3(false);
//   EMU_EnterEM2(false);
 }
}
```

What happens do you still see an output?
8) Remove LED Blink to further reduce current spikes: To do this we will modify the Output location for LETIMER from Location 0 to Location 1:
   a) Modify the gpioOutput pins to the following:

```
/* Configure PD6 and PD7 as push pull so the
    LETIMER can override them */
#ifdef LED
 GPIO_PinModeSet(gpioPortD, 7, gpioModePushPull, 0);
 GPIO_PinModeSet(gpioPortD, 6, gpioModePushPull, 0);
#else
  GPIO_PinModeSet(gpioPortB, 11, gpioModePushPull, 0);
  GPIO_PinModeSet(gpioPortB, 12, gpioModePushPull, 0);
#endif
```

   b) Now modify the LETIMER0 ROUTE register as such:

```
 /* Route LETIMER to location 0 (PD6 and PD7) and enable outputs */
#ifdef LED
 LETIMER0->ROUTE = LETIMER_ROUTE_OUT0PEN | LETIMER_ROUTE_OUT1PEN |
LETIMER_ROUTE_LOCATION_LOC0;
#else
 LETIMER0->ROUTE = LETIMER_ROUTE_OUT0PEN | LETIMER_ROUTE_OUT1PEN |
LETIMER_ROUTE_LOCATION_LOC1;
#endif
```

   c) Verify this makes sense from opening the TG840 datasheet. Look at Section *4 Pinout and Package* find *LETIMO_OUT0#1* and *LETIMO_OUT1#1* are these *PB11,12*?
   d) Compile/Download, Disconnect and view on Energy Profiler, notice a difference?

PART II: LETIMER PWM w/RTC Trigger

1) Modify Project for EFM32TG840F32
   a) *Right Click on letimer_pwm_pulse-Debug* choose *Options…*
   b) General Options -> Change to TG840F32
   c) C/C++ Compiler: Preprocessor \EFM32TG\Include, Defined Symbols: EFM32TG840F32
   d) Linker: Change to EFM32TG840F32.icf
   e) Add startup_efm32tg.s and system_efm32tg.c in CMSIS folder
   C:\Users\Frank Roberts\AppData\Roaming\energymicro\Device\EnergyMicro\EFM32TG\Source\IAR
   C:\Users\Frank Roberts\AppData\Roaming\energymicro\Device\EnergyMicro\EFM32TG\Source

2) Read AN Software Example section, notate how LETIMER is configured.

Backup Power Domain: Go through BURTC and EMU Ref Sections on BU Domain

EFM32 Energy Debugging: Adding SWO() Function to observe on Energy Aware Profiler

Real-Time Counter: Incorporate LETIMER, Energy Modes into RTC AN

Low Energy UART: Show with 2 Eval Boards have groups join together. Add SWO() function for Energy Profile

Direct Memory Access: Reference DMA AN0013

UART Bootloader: AN0003 and USB/UART Bootloader AN0042