

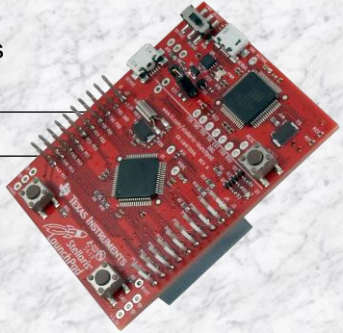
Hibernation Module

Introduction

In this chapter we'll take a look at the hibernation module and the low power modes of the M4F. The lab will show you how to place the device in sleep mode and you'll measure the current draw as well.

Agenda

- Introduction to ARM® Cortex™-M4F and Peripherals
- Code Composer Studio
- Introduction to StellarisWare, Initialization and GPIO
- Interrupts and the Timers
- ADC12
- Hibernation Module**
- USB
- Memory
- Floating-Point
- BoosterPacks and grLib
- Synchronous Serial Interface
- UART
- μDMA



Key Features...

Chapter Topics


Hibernation Module	6-1
<i>Chapter Topics.....</i>	<i>6-2</i>
<i>Low Power Modes.....</i>	<i>6-3</i>
<i>Lab 6: Low Power Modes</i>	<i>6-5</i>
Objective.....	6-5
Procedure.....	6-6

Low Power Modes

Key Features

- ◆ Real Time Clock is a 32-bit seconds counter with a 15-bit sub seconds counter & add-in trim capability
- ◆ Dedicated pin for waking using an external signal
- ◆ RTC operational and hibernation memory valid as long as V_{BAT} is valid
- ◆ GPIO pins state retention provided during VDD3ON mode
- ◆ Two mechanisms for power control
 - System Power Control for CPU and other on-board hardware
 - On-chip Power Control for CPU only

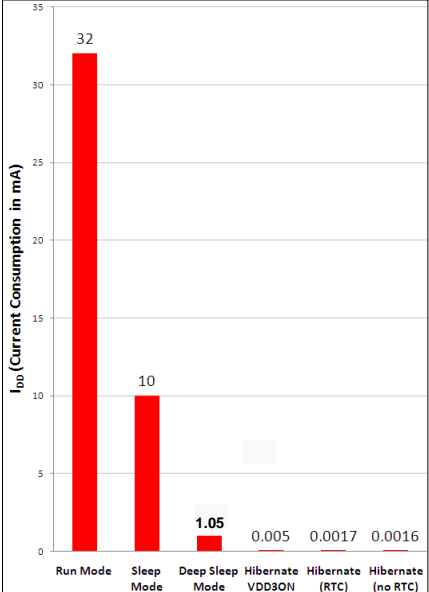
- ◆ Low-battery detection, signaling, and interrupt generation, with optional wake on low battery
- ◆ 32,768 Hz external crystal or an external oscillator clock source
- ◆ 16 32-bit words of battery-backed memory are provided for you to save the processor state to during hibernation
- ◆ Programmable interrupts for RTC match, external wake, and low battery events.



Low Power Modes...

Power Modes

- ◆ Run mode
- ◆ Sleep mode stops the processor clock
 - 2 SysClk wakeup time
- ◆ Deep Sleep mode stops the system clock and switches off the PLL and Flash
 - 1.25 – 350 μ S wakeup time
- ◆ Hibernate mode with only hibernate module powered (VDD3ON, RTC and no RTC)
 - ~500 μ S wakeup time



Power Mode	I_{BAT} (Current Consumption in mA)
Run Mode	32
Sleep Mode	10
Deep Sleep Mode	1.05
Hibernate VDD3ON	0.005
Hibernate (RTC)	0.0017
Hibernate (no RTC)	0.0016

Power Mode Comparison...

Power Mode Comparison

Mode →	Run Mode	Sleep Mode	Deep Sleep Mode	Hibernation (VDD3ON)	Hibernation (RTC)	Hibernation (no RTC)
Parameter ↓						
I_{DD}	32 mA	10 mA	1.05 mA	5 μA	1.7 μA	1.6 μA
V_{DD}	3.3 V	3.3 V	3.3 V	3.3 V	0 V	0 V
V_{BAT}	N.A.	N.A.	N.A.	3 V	3 V	3 V
System Clock	40 MHz with PLL	40 MHz with PLL	30 kHz	Off	Off	Off
Core	Powered On	Powered On	Powered On	Off	Off	Off
	Clocked	Not Clocked	Not Clocked	Not Clocked	Not Clocked	Not Clocked
Peripherals	All On	All Off	All Off	All Off	All Off	All Off
Code	while{1}	N.A.	N.A.	N.A.	N.A.	N.A.

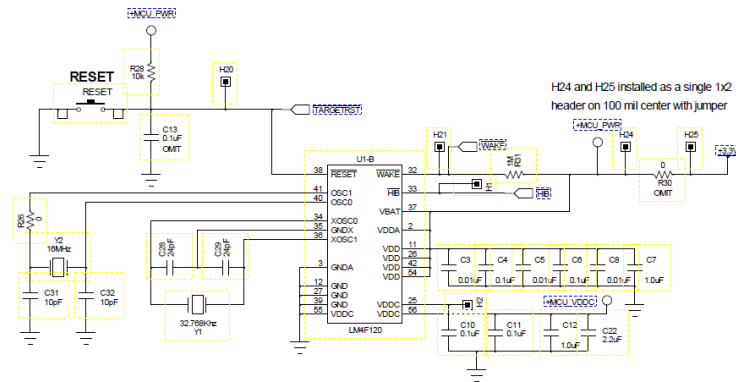


Box denotes power modes available on LaunchPad board

LaunchPad Considerations ...

LaunchPad Considerations

- ◆ The low-cost LaunchPad board does not have a battery holder
- ◆ VDD and VBAT are wired together on the board (this disables battery-only powered low-power modes)
- ◆ Device current is measured between test points H24 and H25




Lab ...

Lab 6: Low Power Modes

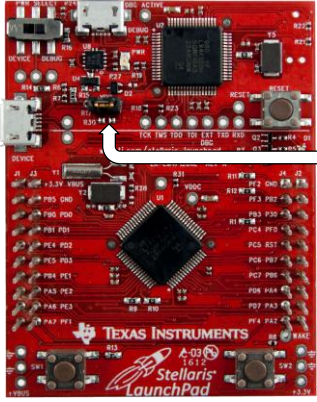
Objective

In this lab we'll use the hibernation module to place the device in a low power state. Then we'll wake up from both the wake-up pin and the Real-Time Clock (RTC). We'll also measure the current draw to see the effects of the different power modes.


Lab 6: Low Power Modes



USB Emulation Connection



Power Measurement Jumper



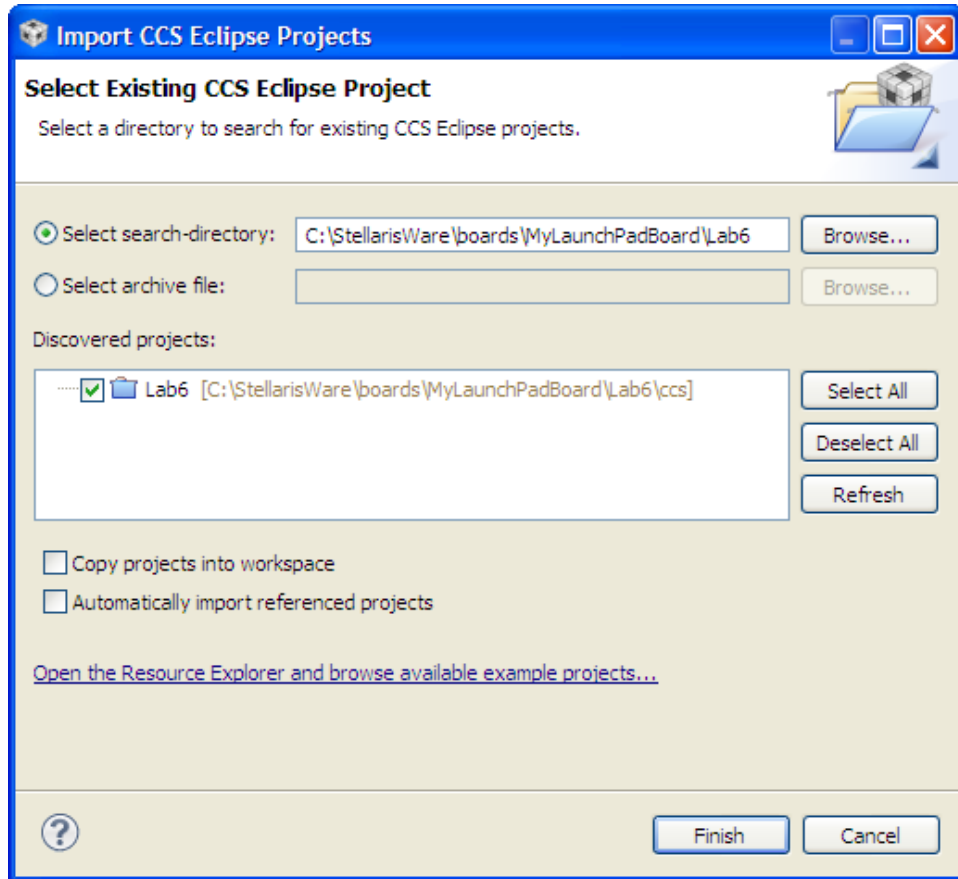
- ◆ Place device in low power modes
- ◆ Wake from pin
- ◆ Wake from RTC
- ◆ Measure current
- ◆ No battery holder on board

Agenda ...

Procedure

Import Lab6

1. We have already created the Lab6 project for you with an empty `main.c`, a startup file and all necessary project and build options set. Maximize Code Composer and click Project → Import Existing CCS Eclipse Project. Make the settings shown below and click Finish. **Make sure that the “Copy projects into workspace” checkbox is unchecked.**



Limitations

2. In order to keep the cost of the LaunchPad board ultra-low, the battery holder was omitted. We will be evaluating the following power modes and wake events:
 - Run
 - Hibernate (VDD3ON)
 - Wake from pin (no RTC)
 - Wake from RTC

Header Files

- Open `main.c` for editing and delete the current contents. Type (or copy/paste) the following lines into `main.c` to include the header files needed to access the StellarisWare APIs :

```
#include "utils/ustdlib.h"
#include "inc/hw_types.h"
#include "inc/hw_memmap.h"
#include "driverlib/sysctl.h"
#include "driverlib/pin_map.h"
#include "driverlib/debug.h"
#include "driverlib/hibernate.h"
#include "driverlib/gpio.h"
#include "driverlib/systick.h"
```

Error Function

- Performing error checking on function calls is good programming practice. Skip a line after the previous includes, and add the following code. `DEBUG` has already been added to the project's pre-defined symbols.

```
#ifdef DEBUG
void __error__(char *pcFilename, unsigned long ulLine)
{
}
#endif
```

Main Function

- Skip a line and add this `main()` template after the error function:

```
int main(void)
{

}
```

Clock Setup

- Configure the system clock to 40MHz again. Add this line as the first line of code in `main()` :

```
SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAIN);
```

GPIO Configuration

7. We're going to use the green LED (2=red=pin1, 4=blue=pin2 and 8=green=pin3) as an indicator that the device is in hibernation (off for hibernate and on for wake). Add a line for spacing and add these lines of code after the last:

```
SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);  
GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);  
GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0x08);
```

Hibernate Configuration

8. We want to set the wake condition to the wake pin. Take a look at the board schematics and see how the WAKE pin is connected to user pushbutton 2 (SW2) on the LaunchPad board.

The code below has the following functions:

Line 1: enable the hibernation module

Line 2: defines the clock supplied to the hibernation module

Line 3: Calling this function enables the GPIO pin state to be maintained during hibernation and remain active even when waking from hibernation.

Line 4: delay 4 seconds for you to observe the LED

Line 5: set the wake condition to the wake pin

Line 6: turn off the green LED before the device goes to sleep

Add a line for spacing and add these lines after the last ones in `main()` :

```
SysCtlPeripheralEnable(SYSCTL_PERIPH_HIBERNATE);  
HibernateEnableExpClk(SysCtlClockGet());  
HibernateGPIORetentionEnable();  
SysCtlDelay(64000000);  
HibernateWakeSet(HIBERNATE_WAKE_PIN);  
GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_3, 0x00);
```


Hibernate Request

- Finally we need to go into hibernation mode. The `HibernateRequest()` function requests the Hibernation module to disable the external regulator, removing power from the processor and all peripherals. The Hibernation module remains powered from the battery or auxiliary power supply. If the battery voltage is low (or off) or if interrupts are currently being serviced, the switch to hibernation mode may be delayed. If the battery voltage is not present, the switch will never occur.

The `while()` loop acts as a trap while any pending peripheral activities shut down (or other conditions exist). Add a line for spacing and add these lines after the last ones in `main()`:

```
HibernateRequest();
while(1)
{
}
```

Click the Save button to save your work. Your code should look something like this:

```
#include "utils/ustdlib.h"
#include "inc/hw_types.h"
#include "inc/hw_memmap.h"
#include "driverlib/sysctl.h"
#include "driverlib/pin_map.h"
#include "driverlib/debug.h"
#include "driverlib/hibernate.h"
#include "driverlib/gpio.h"
#include "driverlib/systick.h"

#ifdef DEBUG
void __error__(char *pcFilename, unsigned long ulLine)
{
}
#endif

int main(void)
{
    SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAIN);


    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);
    GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0x08);

    SysCtlPeripheralEnable(SYSCTL_PERIPH_HIBERNATE);
    HibernateEnableExpClk(SysCtlClockGet());
    HibernateGPIORetentionEnable();
    SysCtlDelay(64000000);
    HibernateWakeSet(HIBERNATE_WAKE_PIN);
    GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_3, 0x00);


    HibernateRequest();
    while(1)
    {
    }
}
```

This code is saved in the `Lab6/ccs` folder as `main1.txt`. Don't forget that you can auto-correct the indentations:

Build, Download and Run the VDD3ON (no RTC) Code

10. Compile and download your application by clicking the Debug button  on the menu bar. If you have any issues, correct them, and then click the Debug button again. After a successful build, the CCS Debug perspective will appear.
11. Delete the watch expressions by right-clicking in the Expressions pane and clicking Remove All, then click Yes.

Note: Code Composer Studio has some issues connecting to hibernating devices (and re-connecting) since they essentially power off in the middle of the debugging process. We'll try to step around those issues, but you may see CCS terminate abruptly. If this happens, you can restart CCS and try again, or you can use the LM Flash Programmer to reprogram the device with the qs-rgb (non-hibernation) program. In either case, you need to hold SW2 down to keep the LM4F device awake in order for either tool to connect.

12. We're going to step around those hibernate/CCS issues now. Press the Terminate  button in CCS to return to the editing mode. When you do this the LaunchPad will be issued a reset. Observe the LED on the board and cycle power on the LaunchPad by removing/replacing the USB emulator cable. The green LED should light.



After about 4 seconds the green LED will go out. Press the SW2 button located at the lower right corner of the LaunchPad board. The processor will wake up and start the code again, lighting the green LED.

Note that this wakeup process is the same as powering up. We will not be using the battery-backed memory in this lab, but that feature is essential to applications that need to know how they "woke up". Your code can save/restore the processor state to that memory. When your code starts, you can determine that the processor woke from sleep and restore the processor state from the battery-backed memory.

13. Now that we know the code is running properly, we can take some current measurements. Before we do, let's comment out the line of code that lights the green LED so that the LED current won't be part of our measurement. In `main.c`, comment out the line of code shown below:

```
22 SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);  
23 GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);  
24 // GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0x08);
```

Save your work.

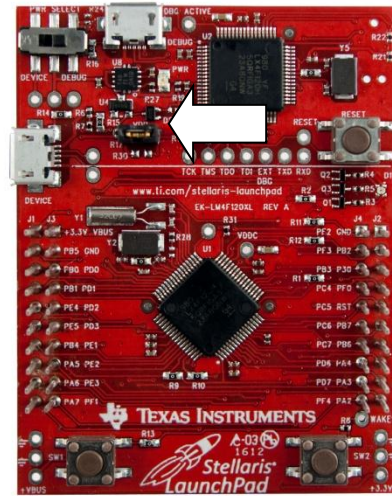
14. Press and hold SW2 on the LaunchPad board (to make sure it is awake), then compile and download your application by clicking the Debug button  on the menu bar. Press the Terminate  button in CCS to return to the editing mode. Remove the USB emulator cable from the LaunchPad board.

Measure the Current

- Remove the jumper located on the LaunchPad board near the DEVICE USB port and put it somewhere for safekeeping.

Connect your Digital Multi-Meter (DMM) test leads to the pins with the positive lead nearest the DEVICE USB port. Double check the lead connections on the meter. Switch the meter to measure DC current around 20mA.

- Watch the meter display and plug the USB emulator cable into the LaunchPad. During the first four seconds the LM4F device is in Run mode (in the software delay loop). Record this reading in the first row of the chart below.



- After four seconds the device goes into the VDD3ON hibernate mode. Switch your meter to measure 10uA and record your reading in the second row of the chart below.
- Remove the USB emulator cable from the LaunchPad board, disconnect your DMM and replace the jumper on the power measurement pins.

Mode	Workbook Step	Your Reading	Our Reading
Run	15	mA	21.4 mA
VDD3ON (no RTC)	17	μA	6.0 μA
VDD3ON (RTC)	27	μA	6.3 μA

Wake Up on RTC

19. Plug the USB emulator cable into the LaunchPad board.
20. In `main.c`, find this line of code: `HibernateWakeSet(HIBERNATE_WAKE_PIN);`
Right above that line of code, enter the three lines below. These lines configure the RTC wake-up parameters; reset the RTC to 0, turn the RTC on and set the wake up time for 5 seconds in the future.

```
HibernateRTCSet(0);  
HibernateRTCEnable();  
HibernateRTCMatch0Set(5);
```

21. We also need to change the wake-up parameter from just the wake-up pin to add the RTC. Find:

```
HibernateWakeSet(HIBERNATE_WAKE_PIN);
```

and change it to:

```
HibernateWakeSet(HIBERNATE_WAKE_PIN | HIBERNATE_WAKE_RTC);
```

22. Uncomment the line of code that turns on the green LED, as shown below:

```
22   SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);  
23   GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);  
24   GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0x08);  
25
```

Save your changes.

Your code should look like this:

```
#include "utils/ustdlib.h"
#include "inc/hw_types.h"
#include "inc/hw_memmap.h"
#include "driverlib/sysctl.h"
#include "driverlib/pin_map.h"
#include "driverlib/debug.h"
#include "driverlib/hibernate.h"
#include "driverlib/gpio.h"
#include "driverlib/systick.h"

#ifdef DEBUG
void __error__(char *pcFilename, unsigned long ulLine)
{
}
#endif


int main(void)
{
    SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAIN);

    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);
    GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0x08);

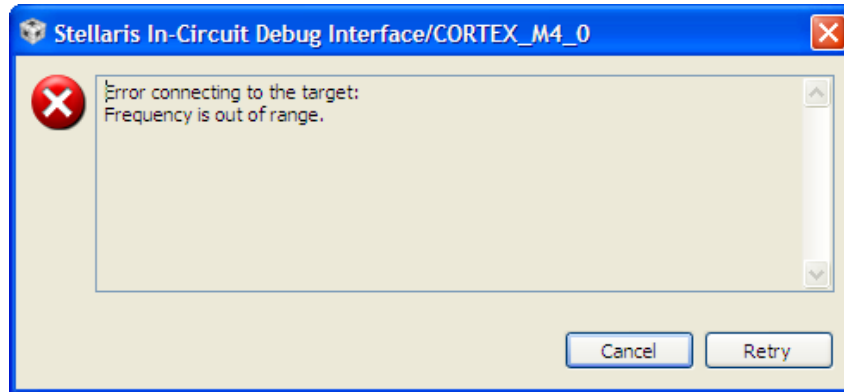
    SysCtlPeripheralEnable(SYSCTL_PERIPH_HIBERNATE);
    HibernateEnableExpClk(SysCtlClockGet());
    HibernateGPIORetentionEnable();
    SysCtlDelay(64000000);
    HibernateRTCSet(0);
    HibernateRTCEnable();
    HibernateRTCMatch0Set(5);
    HibernateWakeSet(HIBERNATE_WAKE_PIN | HIBERNATE_WAKE_RTC);
    GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_3, 0x00);

    HibernateRequest();
    while(1)
    {
    }
}
```


If you're having problems, this code is saved as `main2.txt` in your `Lab6/ccs` folder.

23. Press and hold the SW2 button on your evaluation board to assure the LM4F120 is awake. Compile and download your application by clicking the Debug button  on the menu bar.

CCS can't talk to the device while it's asleep. If you accidentally do this, you'll see the following when CCS attempts to communicate:



If this happens, press and hold the SW2 button and click Retry. Release the SW2 button when the debug controls appear in CCS.

24. Press the Terminate  button in CCS to return to the editing mode. When the Debugger terminated, it reset the LM4F120. You should see the green LED turning on for 4 seconds, then off for about 5 seconds. The real-time-clock (RTC) is waking the device up from hibernate mode after 5 seconds. Also note that you can wake the device with SW2 at any time.
25. Disconnect your LaunchPad board from the USB emulator cable.
26. Remove the jumper located on the LaunchPad board near the DEVICE USB port and put it somewhere for safekeeping.
- Connect your DMM test leads to the pins with the positive lead nearest the DEVICE USB port. Double check the lead connections on the DMM itself. Switch the DMM to measure DC current around 20mA.
27. Plug the USB emulator cable into your LaunchPad board. When the green LED goes off, quickly switch the DMM to measure 10uA and record your reading in the last row of the chart in step 18. The equivalent series resistance on most DMMs will be too high in the low current mode to allow the device to go back to run mode.
28. Remove the USB emulator cable from the LaunchPad board, disconnect and turn off your DMM, then replace the jumper on the power measurement pins.

29. To make things easier for the next lab, use the LM Flash Programmer to program the `qs-rgb` bin file into the device (as shown in lab 2). Don't forget to hold SW2 down while this process completes.
30. Close the Lab6 project and minimize Code Composer Studio.
31. **Homework Idea:** Experiment with the RTC to create a time-of-day clock at the lowest possible power.



You're done.

