Lab 2 Basic Clock Module +

ReadMeFirst

Lab Folder Content

1) ReadMeFirst
2) BasicClockModulePlus – Lecture material
3) IOModule
4) PinOutSummary
5) Use_RigolPowerSupply_wt_Launchpad

Overview

Like all sequential circuits, microcontrollers (MCUs) need a clock to keep different parts of the system synchronized. Modern day microcontrollers contain a number of clocks; each clock can be sourced from external sources (that lie outside of the microcontroller chip) or from internal sources which are integrated within the MCU. In former days the clock of a microcontroller was mostly sourced from a single crystal or an RC oscillator with limited flexibility to change the speed of the clock. Modern day microcontrollers generally provide more than one choice for the source of the clocks which allows users to achieve the best balance between performance and power consumption.

Select Clock Source

In general, clock sources can be divided into two groups: fast clocks and slow clocks. Fast clock sources, typically in the MHz range with a short start-up time and are used for driving the CPU and some peripherals. For example, how much time your coded instructions take to execute is counted in terms of the number of CPU clock cycles consumed by your code. The faster the clock, the faster your instructions will execute. However the aim is not always to have the fastest CPU, the power consumed by the MCU is
also an important factor to consider. We will give you some idea about this balance in this lab. The slow clock, usually with a frequency in the range of tens of kHz, is used to drive peripherals or to run continuously to monitor real time. Therefore the slow clock needs to be low in power consumption and stable over time, temperature and other external influences; after all you do not want your cell phone to spend all its battery power in a few hours while you are not using it, neither would you want your watch to run faster in summer than in winter and mess up your time schedule.

RC oscillators are cheap sources for clocks that can be integrated within the MCU. They are not very accurate but can be started quickly. Crystals are accurate sources for clocks. They are stable over time and temperature. However crystals are relatively expensive and delicate. They draw relatively large amount of current at high frequencies and takes a long time to start, about $10^5$ cycles

**MSP430G2553 Clock Module – Basic Clock Module +**

The clock module provided in the MSP430G2553 microcontroller (or the family of MSP430x2xx family in general) is called the “Basic Clock Module+”. This clock module addresses the conflicts between high performance, low power consumption and precision in frequency by introducing three different possible clocks to choose from:

- Master Clock (MCLK) – used by the CPU and some peripherals
- Subsystem Master Clock (SMCLK) – used by peripherals
- Auxiliary Clock (ACLK) – also used by peripherals

The G2553 provides three possible clock sources users can choose from to source the clocks:

- Very Low-Frequency Oscillator (VLO) – Very low frequency (about 12kHz) and low power consumption (μW range)
- Digitally Controlled Oscillator (DCO) – High frequency (MHz range), versatile in frequency selection, fast start-up and suitable to be used to drive the CPU.
• External Crystal Oscillator (LFXT1CLK)

**IMPORTANT:**

Below is a block diagram of the Basic Clock Module+. Block diagrams try to detail all possible settings and which bits in control registers the settings corresponds to. The bits are given easy to understand names such as the ones marked out in yellow in the diagram using header files. For example, at the top right hand side corner of the diagram, “DIVAx” means divide Auxiliary clock (ACLK) frequency. Possible dividing values are 1, 2, 4 and 8. What the block diagram in the data sheet does not specify is that these bits (which are controlling how much to divide by) are the 4-5th bits in an 8 bit special function register named BCSCTL1 (Basic Clock System Control Register 1, located at 0x0057). If you are programming a microcontroller you would not need to know this information in most of the cases. All you really care is which register you need to set and the name of the function bits. Setting 4-5th bit to (00)₂ allows you to divide ACLK by 1, setting to (01)₂ gives you a divide by 2 etc. As you can see, block the diagram is very important as a reference to how to setup a complicated microcontroller.

There are two different naming convention for MSP430. Take divide ACLK as an example, you can access this function by

a) Set DIVA0 and DIVA1, which is equivalent to setting bit 4 and bit 5 of BCSCTL1
   e.g. To divide ACLK by 4
   
   bis.b  #DIVA1, &BCSCTL1
   bic.b  #DIVA0. &BCSCTL1

b) Or if the function is set by more than 1 bits and less than 5 bits, you can use DIVA₀, DIVA₁, DIVA₂, DIVA₃ to set the 4-5th bits to (00)₂, (01)₂, (10)₂ or (11)₂
   e.g. To divide ACLK by 4
   
   bis.b  #DIVA₂, &BCSCTL1
Reading block diagrams is very important because block diagrams provides you with valuable information about the peripheral modules and how to configure them at a glance.

One more thing to point out is that “modules” are not device specific. For example, Basic Clock Module is available in MSP430x2xxx series (where G2553 belongs to), MSP430x1xxx. Some other clock module for MSP430 are Unified Clock System, Real Time Clock System and Clock System etc. Each of them can be found in various devices or families.

---

Please watch the screencast – Basic Clock Module Plus

Proceed after watching screencast:

Here we provide a basic summary for setting up the clock module.

1) Decide what frequencies are needed
   a. What clock frequency and power consumption requirements are needed is determined by your application’s specifications.
   b. DCO can be configured to be from about 90kHz to 20MHz. VLO frequency and crystal frequency are non-configurable. VLO is about 12kHz with variation from device to device. The frequency of the Low Frequency crystal is fixed at 32,768 Hz = 2^{15}. 
c. Each clock source before connecting to a clock can be further divided by 1,2,4 and 8 in frequency.

2) Configure DCO (if you decide to use the DCO)
   a. Please refer to the handout *Basic Clock Module Plus* for more details on selecting the DCO frequency
   b. You can use pre-calibrated frequencies: 1MHz, 8MHz, 12MHz and 16MHz
   c. Otherwise, select DCO frequency and range via DCOx and RSELx. This gives you 8*16 different clock frequency to choose from.
   d. For fine tuning DCO frequency, select modulation of DCO frequency via MODx bits to generate (effective) frequencies in between the options provided in the previous step.

3) Decide what clocks are needed and what source to use for each clock
   a. MCLK, SMCLK and ACLK are used for different purposes
   b. Consider the feature of each clock source if they are good enough to use
   c. For a slow clock source used in the system, you can select between VLO or the crystal via LFXT1Sx.
   d. Note that for some microcontrollers (but not the G2553) there is room for a second crystal to be used as a clock source. The 2nd crystal is not available for the G2553 so you can have either VLO or a crystal but not both.
   e. ACLK is connected to the slow clock source selected. MCLK and SMCLK can be configured via SELM and SELS bits respectively to connect to either DCO or the selected slow clock source.

Note that some peripherals that require clock input may have their own dividers to further reduce the frequency of the clock.

MCLK and SMCLK clock sources are selected via associated multiplexers separately. The clock source selection multiplexers as marked by blue color in the block diagram are controlled via SELM (select MCLK) and SELS (select SMCLK) respectively.
• Please make sure you understand the content of the document *BasicClockModulePlus* before proceeding with lab experiments.

• Please attach the code you write in this lab at the end of your lab report. The code should be nicely commented for you to use in later labs. We encourage you to write modular programs by using subroutines. Missing course code will cost you points in your lab report grades.
Lab Experiments

In this lab we are going to inspect the specifications of each of the clocks, how to setup the clock sources, how to connect the source to the clocks and how to measure and inspect the clock waveforms and their power consumption behavior. In order to have control to adjust the supply voltage, we are going to use the power supply on the lab bench to supply power to your MCU in some of the lab procedures.

We can think of the LaunchPad as being composed of two sections; one side that contains the USB port which connects the PC to the debugger, and the other side which contains the microcontroller's socket. These sections are separated by a dashed white line on the PCB board. The connections between the debugger section and the MCU section are via five jumpers shown in the figure below (yellow rectangle)
The debugger programs the LaunchPad via the RST and TEST pins and supplies power to the MCU side of the board via the VCC pin. In order to provide external power to the MCU using the power supply on your lab bench you should remove the jumper labeled VCC (see the figure above) and then connect V+ and gnd from the power supply directly to the VCC and GND pin on the MCU side of the LaunchPad as shown in the figures below.
Please read through the document:

*Use_RigolPowerSupply_wt_Launchpad*
Experiment 1 – Very Low Frequency Clock (VLO)

The internal very low power, low-frequency oscillator (VLO) approximately provides a frequency of 12 kHz (see device-specific data sheet for parameters). In this experiment you are going to measure your device VLO frequency, and inspect basic properties of the VLO clock.

Power the MCU from the USB connector
Select VLO to be used by the system

Select LFXT1 to be low-frequency mode (XTS, bit 6 in register BCSCTL1)
Set Low Frequency Clock Select (LFXT1S, bit 5-4 in register BCSCTL3) to be sourced from VLO (10)

Output ACLK via GPIO pins

ACLK is at pin 1.0 (Pin #2)
Configure P1.0 to output ACLK by setting bit0 in register P1DIR, P1SEL and clear bit0 in register P1SEL2

1) Measure and record the ACLK frequency using the oscilloscope.
   ○ Provide a oscilloscope screenshot of ACLK frequency (1pt)

2) Use the power supply to power the MCU. Set over current protection to 0.1A.
   Change the power supply voltage of the MCU.
   ○ Compare the frequency of VLO at 2.5 V and 3.5V. (No screenshots required)
   ○ Compare the ACLK frequency you obtained with the results of two other groups.
   ○ Comment on the results you found.
   (1pt)
Experiment 2 – Digitally Controlled Oscillator (DCO Clock)

After a PUC (Power Up Clear), the default settings are: RSELx = 7 and DCOx = 3, allowing the DCO to start at a mid-range frequency. You can get the PUC setting by click on the button in CCS and look at the registers before running any code. **We are going to use 3V external power for this experiment.** Set over current protection to 0.1A.

1) Verify the PUC setting in the debugger.
   Provide a screenshot in your lab report (DCO, RSEL and MOD). (0.5pt)

2) Use calibration to set DCO frequency to 12MHz.

   Move content of CALBC1_12MHZ to BCSTCL1
   Move content of CALDCO_12MHZ to DCOTL
   Pay attention that MHZ are all upper case

Measure the actual DCO frequency under this calibrated setting.

- Provide a screenshot of the 12MHz calibration settings from the debugger (DCO, RSEL and MOD). (0.5pt)
- Find the two frequencies mixed into this calibration. Provide screenshots of the oscilloscope clearly showing the two frequencies. Record the two frequencies. (1pt)
- Provide an oscilloscope screenshot measuring 10 period of DCO frequency using markers. You can measure more than 10 period if need to.
- Justify what is your calibrated DCO frequency. Provide any calculation if necessary. (1pt)
- Justify the calibration equation. Comment on your result (1pt)

\[ 32t = (32 - \text{MOD}) \times t_{\text{DCO,RSEL}} + \text{MOD} \times t_{\text{DCO+1,RSEL}} \]

\[ f = 1/t \text{ is the effective frequency.} \]
3) Find the minimum and maximum frequency of your DCO clock,
   - record your result (1pt)

   *Set DCO Range Select (RSELx, bit3-0 in register BCSCTL1) and DCO Frequency Select (DCOx, bit7-5 in register DCOCTL) to 0 for minimum frequency*

   *Set RSEL to 15 and DCO to 7 for maximum frequency*

   *To measure DCO clock, configure P1.4 for SMCLK*

**Experiment 3 – External Clock Source**

Besides the on-chip VLO and DCO, Basic Clock Module also supports external oscillators. The module connecting to external oscillators is called LFXT1 Oscillator. Diagram below is taken from the Basic Clock Module Plus block diagram showing the LFXT1 Oscillator:
To use LFXT1, you can solder external crystals or resonators onto the Launchpad. The location of external crystal/resonator in the block diagram and on the Launchpad is marked out in the diagrams below:
Another use of LFXT1 is to connect an external clock source, from a signal generator for example. The signal generator can be another MCU, DSP, or a bench top signal generator.

******************************************************************************

By default our ACLK is connected to the LFXT1 Oscillator. When missing a source, you will observe almost noise on ACLK. A special bit called oscillator-fault (LFXT1OF) will be set if the LFXT1 oscillator is connected but not operating. An interrupt enable (OFIE) and interrupt flag (OFIFG) is associated with this false condition as well.

******************************************************************************

In this experiment we are going to use our Rigol signal generator to supply 1MHz clocking signal via LFXT1 to ACLK. Couple of settings need to be made:

Select high frequency mode by setting XTS (bit 6 in BCSCTL1) to 1

Select digital external clock source by setting LFXT1Sx (bits 5-4 in BCSCTL3) to (11)b

Select oscillator capacitor to be the minimum ~1pF by setting XCAPx (Bits 3-2 in BCSCTL3) to (00)b

Configure XIN (P2.6) to be input pin

Use USB to power the microcontroller. Use the waveform generator to generate a square wave with amplitude from 0V to 3V at frequency 1MHz.

- Output and measure this external clock source via ACLK. Comment on the result. (1pt)
Short Answer Questions

1. Among DCO, VLO and Crystal, which one is the least precise? Which one starts the fastest? Which one has the lowest power consumption? (1pt)

2. What is the lowest and highest frequency you can generate internally using G2553? Recall that the frequencies can be further divided before being sent to the clocks. (0.5pt)

3. Check the datasheet, which low power mode has only ACLK clock running? What will happen to your system if you set to this low power mode? (0.5pt)

(Optional) Experiment 4 – Design: 1 Hz Square Wave Generation

(Bonus 2 pts for successful completion)

In this experiment you will be using software to produce a 1sec delay.

Set your DCO to 1MHz calibrated frequency and divide it by 8. By default your DCO clock is sourcing your MCLK. Follow the pseudo code below:
Outer loop
{
    Initialize a counter
    While (counter > 0)
    {
        Decrement counter
    }
    Toggle LED
}

If we consider the code in red should consume 1 sec. What counter value should you use? To find your counter value, you should use the **clock tool** in Code Composer Studio to determine how many clock cycles each iteration of your loop is consuming.

To enable the clock tool, go to debug view and click on Run -> Clock -> Enable

Toggle an LED every time you have wasted 1 second as shown in the pseudo-code above. How many LED blinks do you get in 10 seconds? Is the result consistent with the red code producing a 1 sec delay?

Try to get as close to 1Hz as possible by tweaking your clock frequency and counter value. Probe your pin out using the digital probe of oscilloscope. Include a screenshot of the waveform in the report.