F28035 ControlCard Lab1_4

External XINT1 Interrupt Latency

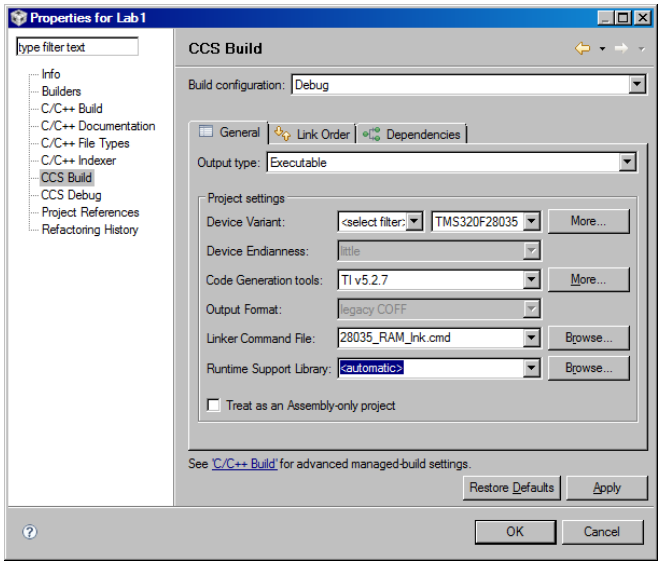
1. Project Dependencies

The project expects the following support files:

- Support files of controlSUITE installed in:

  C:\TI\controlSUITE\device_support\f2803x\v122

- Code Generation Tools Version 5.2.7:



- CodeComposerStudio Version 4.1.3:

2. Project Objective

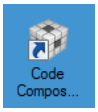
- GPIO02 is used as digital input. A falling edge at GPIO02 will trigger an XINT1 Interrupt Service.
- to trigger the interrupt, use a wire to connect GPIO02 to ground (GND)
- The XINT1 – ISR will copy register "XIntruptRegs.XINT1CTR" into the global variable "counter".
- Register "XIntruptRegs.XINT1CTR" is cleared by the hardware interrupt event of XINT1 and clocked by the CPU frequency of 60 MHz.
- As a result, the value in "counter" gives the interrupt latency in clock cycles (1/60MHz). **Note: The result is 18...20 clock cycles!**
- CPU – Timer 0 runs at 100 ms in background but is not used in this exercise. It is active just to have a 2nd active interrupt source.
- The watchdog unit is active; it is serviced in the main-loop and in the CPU Timer 0 interrupt service function.
- Hint: Use the "Real-Time Debug" mode to display variable "counter" in the watch window

3. Hardware Setup

No special setup required. Connect the F28035 Docking Station with a USB cable to your computer. Make sure the main switch "SW1" is in position "USB". The LED LD1 (green) of the F28035 controlCARD should be "ON".

To test the code, a wire is required to connect GPIO02 to GND to produce a falling edge.

4. Start Code Composer Studio V4

Start Code Composer Studio V4: 

When CCS loads, a dialog box will prompt you for the location of a workspace folder. Use the default location for the workspace and click OK. This folder contains all CCS custom settings, which includes project settings and views when CCS is closed so that the same projects and settings will be available when CCS is opened again. The workspace is saved automatically when CCS is closed.

Note: The first time CCS opens a "Welcome to Code Composer Studio v4" page appears. Close the page by clicking on the CCS icon in the upper right or by clicking the X on the "Welcome" tab. You should now have an empty workbench.

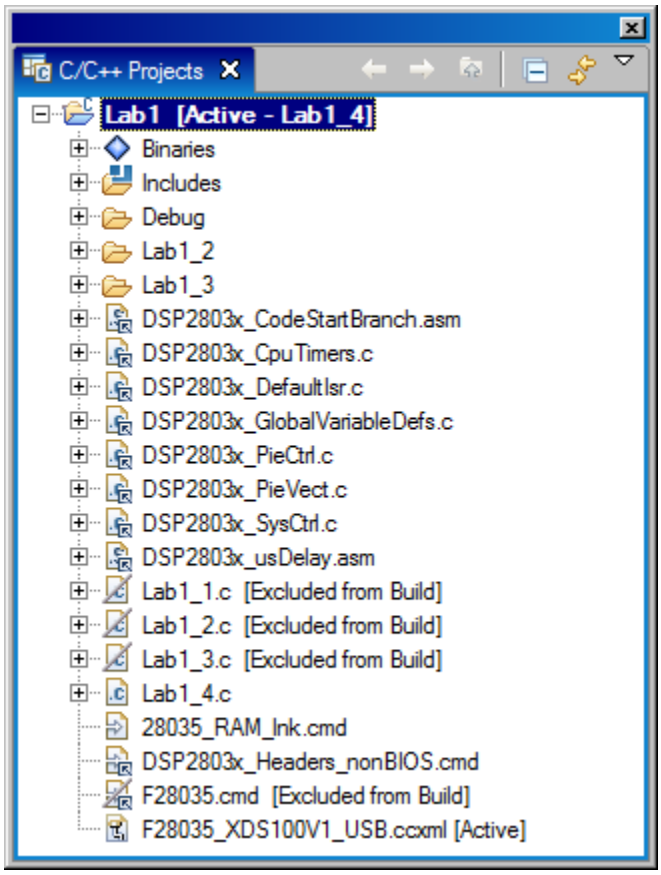
The term workbench refers to the desktop development environment. The workbench will open in the "C/C++ Perspective" view. Notice the C/C++ icon in the upper right-hand corner. A perspective defines the initial layout views of the workbench windows, toolbars, and menus which are appropriate for a specific type of task (i.e. code development or debugging). The "C/C++ Perspective" is used to create or build C/C++ projects. A "Debug Perspective" view will automatically be enabled when the debug session is started.

5. Project Modification

We will use the project “Lab1” as starting point. If not still open in CCSV4, open this project again.

5.1. Create a new code file

- Open file “Lab1_2.c” and save it as “Lab1_4.c”.
- Right click at file “Lab1_3.c” and select “Exclude File(s) from Build”. Make sure that from the files “Lab1_x.c” only “Lab1_4.c” is active.
- Also make sure that the file “28035_RAM_lnk.cmd” is included and that the file “F28035.cmd” is excluded from build:



6. Code Modification

6.1. Modify file “Lab1_4.c”

At the beginning of the file add a function prototype for the XINT1 interrupt service routine:

```
interrupt void GPIO02_isr(void);
```

Add a new global variable “counter”. Later we will use this variable to store the value from the interrupt counter of XINT1

```
unsigned int counter;
```

Inside the “EALLOW - EDIS” block, add the following lines to select GPIO02 as source signal for XINT1, to enable the XINT1 interrupt service, to request an interrupt service with the falling edge of GPIO02 and to replace the interrupt vector address for XINT1:

```
EALLOW;
GpioCtrlRegs.GPAPUD.bit.GPIO2 = 0;// enable pull-up
GpioIntRegs.GPIOXINT1SEL.bit.GPIOSEL = 2; // GPIO02
XIntruptRegs.XINT1CR.bit.ENABLE = 1;      // XINT1 – ISR
XIntruptRegs.XINT1CR.bit.POLARITY = 0; // falling edge
PieVectTable.XINT1 = &GPIO02_isr;
EDIS;
```

Finally in “main()” we have to enable the XINT1 – interrupt for the PIE – unit, additionally to the CPU – Timer 0 ISR:

```
PieCtrlRegs.PIEIER1.bit.INTx4 = 1; // enable XINT1 isr
```

After the end of function “main()” we have to define the interrupt service function “GPIO02_isr()”. We have to execute code to acknowledge the interrupt service and to copy the value from the XINT1 clock counter to variable “counter”:

```
interrupt void GPIO02_isr(void)
{
      counter = XIntruptRegs.XINT1CTR;  // copy value
      PieCtrlRegs.PIEACK.all = 1;       // clear INT1
      asm(" ESTOP0");        // software breakpoint)
}
```

Note: The “asm(“ ESTOP0”)” – instruction is a software breakpoint, which will stop the execution. You can use this option to halt the code execution and inspect the value in variable “counter”.

Another test option is to delete this inline assembly instruction and to use the real-time mode debug to inspect variable “counter”.

6.2. Rebuild Project

➔Project ➔ Rebuild Active Project (Alt + Shift + P)

Look for any error messages or warnings and make changes, if necessary.

➔ Target ➔ Debug Active Project

After a successful programming of the device, the “Debug” perspective should come up and the blue arrow in window “Lab1_4.c” should point to the beginning of “main()”.

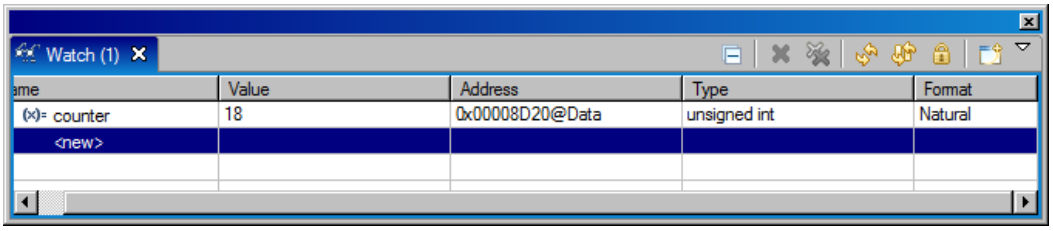
Perform a Run (F8). The LED should blink with a period of 100 milliseconds because it is toggled by CPU – Timer 0 Interrupt Service.

Now use a wire to connect GPIO02 to GND, which will produce a falling edge at GPIO02.

The code execution should stop automatically at the position of the software breakpoint:

```
51 interrupt void GPIO02_isr(void)
52 {
53         counter = XIntruptRegs.XINT1CTR;     // copy value
54         PieCtrlRegs.PIEACK.all = 1;      // clear INT1
55         asm(" ESTOP0");      // software breakpoint)
56 }
57
```

The value in variable “counter” shows the time difference between the falling edge of the signal at GPIO02 and the instruction, which has copied the value from register “XIntruptRegs.XINT1CTR” in number of clock cycles:



7. Solution for Lab1_4.c

```
#include "DSP2803x_Device.h"
extern void InitSysCtrl(void);
extern void InitPieCtrl(void);
extern void InitPieVectTable(void);
extern void InitCpuTimers(void);
extern void ConfigCpuTimer(struct CPUTIMER_VARS *,float,float);
interrupt void cpu_timer0_isr(void);
interrupt void GPIO02_isr(void);
unsigned int counter;
void main(void)
{
      InitSysCtrl();//60MHz, Disable Watchdog, all clocks enabled
      InitPieCtrl();
      InitPieVectTable(); // default addresses in PIE table
      EALLOW;                          // Open secure block
      GpioCtrlRegs.GPADIR.bit.GPIO31 = 1; // output for LED LD2
      GpioCtrlRegs.GPBDIR.bit.GPIO34 = 1; // output for LED LD3
      SysCtrlRegs.WDCR = 0x00AF;          // re - enable watchdog
      PieVectTable.TINT0 = &cpu_timer0_isr;
      GpioCtrlRegs.GPAPUD.bit.GPIO2 = 0;  // enable pull-up
      GpioIntRegs.GPIOXINT1SEL.bit.GPIOSEL = 2; // GPIO02
      XIntruptRegs.XINT1CR.bit.ENABLE = 1;        // XINT1 – ISR
      XIntruptRegs.XINT1CR.bit.POLARITY = 0; // falling edge
      PieVectTable.XINT1 = &GPIO02_isr;
      EDIS;
      InitCpuTimers();
      ConfigCpuTimer(&CpuTimer0, 60, 100000);
      GpioDataRegs.GPACLEAR.bit.GPIO31 = 1;
      GpioDataRegs.GPBSET.bit.GPIO34 = 1;
      PieCtrlRegs.PIEIER1.bit.INTx7 = 1; //enable CPU_Timer0 ISR
      PieCtrlRegs.PIEIER1.bit.INTx4 = 1; // enable XINT1 isr
      IER = 1;                 // enable core line INT1
      EINT;                    // enable global interrupt INTM
      ERTM;                    // Enable global real time DBGM
      CpuTimer0Regs.TCR.bit.TSS = 0;      // start T0
      while(1){
            EALLOW;
            SysCtrlRegs.WDKEY = 0x55;     // 1st WD service instr
            EDIS;
      }
}
interrupt void cpu_timer0_isr(void)
{
      GpioDataRegs.GPATOGGLE.bit.GPIO31 = 1;  // toggle LD2
      GpioDataRegs.GPBTOGGLE.bit.GPIO34 = 1;  // toggle LD3
      PieCtrlRegs.PIEACK.all = 1;             // ack PIE-group 1
      EALLOW;
      SysCtrlRegs.WDKEY = 0xAA;               // 2nd WD service instr
      EDIS;
}
interrupt void GPIO02_isr(void)
{
      counter = XIntruptRegs.XINT1CTR;        // copy value
      PieCtrlRegs.PIEACK.all = 1;             // clear INT1
      asm(" ESTOP0");                         // software breakpoint
}
```