BLG351E-Microcomputer Lab.
Fall 2024-2025

General Information

- Days and hours:
  - Monday: 12:30-14:30 , CRN13599,
  - Tuesday: 8:30-10:30 , CRN13600,
  - Friday: 14:30-16:30 , CRN13603.
- Course Web Page: Ninova path: BLG351/Microcomputer Lab.
- Instructors: İlkay Öksüz
- Teaching Assistants:

- Kadir Özlem (kadir.ozlem@itu.edu.tr)
- Mustafa İzzet Muştu (mustu18@itu.edu.tr)
- Sadettin Fidan (sadettinf@itu.edu.tr)
- Talip Tolga Sarı (sarita@itu.edu.tr)
- İlknur Çelik (celikil17@itu.edu.tr)
- Onur Can Koyun (okoyun@itu.edu.tr)
- M. Abdullah Hakkoz(mustafa.hakkoz@gmail.com)
- Ziya Kağan Zeydan (zeydan20@itu.edu.tr)

Evaluation Criteria

Grading & Criteria:

- 8 Experiments (56%)

  You should participate the experiments in Hardware Lab 2 (103). Coding for each experiment has a time limit of 1.5 hours. The last 0.5 hour will be used for grading the remaining groups. No extension will be given.
- 8 Reports (24%)

  You should use the report template under Class Files.
- 1 Quiz Exam (20%)

  After the experiments, there will be a quiz exam.

VF Conditions:

- Attendance: 7/8 Experiments. Minimum grade for an experiment is 20/100.
- Experiments + Reports: Minimum 40 points out of 80 (without the last Quiz Exam).

Make-Up Conditions:

- To attend a make-up, you should have a valid excuse. The lecturers of the course will decide whether you could participate to the make-up experiment.

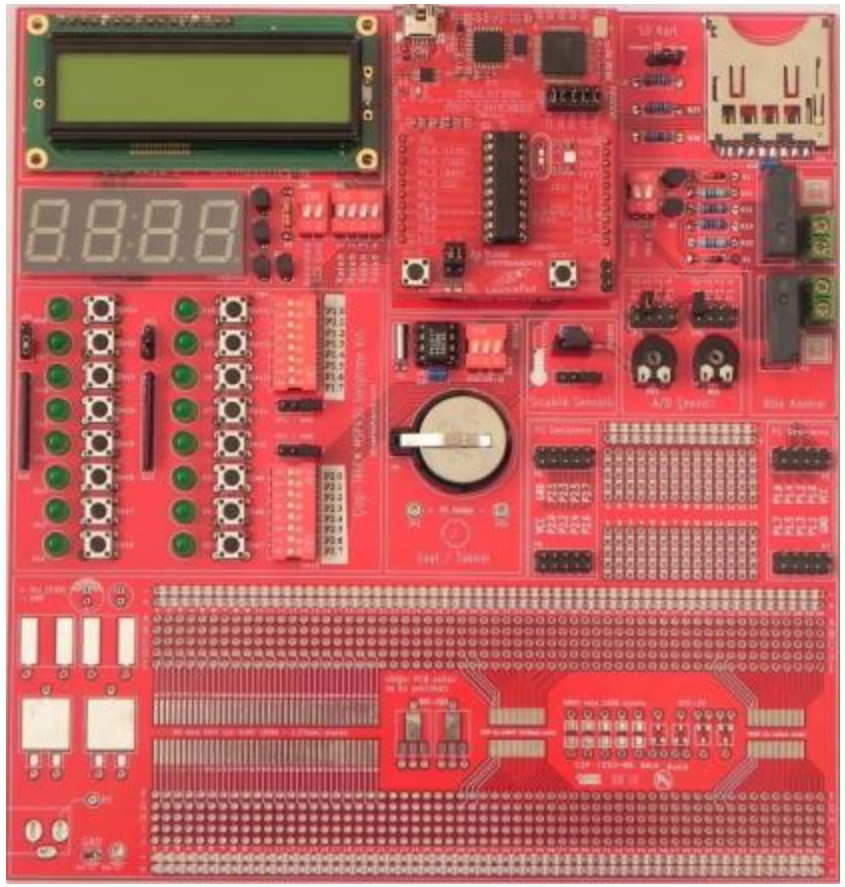
Schedule

Week	Monday (CRN:13599)	Tuesday (CRN:13600)	Friday (CRN:13603)	Activity
1	30.09.2024	01.10.2024	04.10.2024	No Lecture
2	07.10.2024	08.10.2024	11.10.2024	No Lecture
3	14.10.2024	15.10.2024	18.10.2024	Introduction Lecture
4	21.10.2024	22.10.2024	25.10.2024	Experiment 1
5	28.10.2024	29.10.2024	01.11.2024	Republic Day Holiday
6	04.11.2024	05.11.2024	08.11.2024	Experiment 2
7	11.11.2024	12.11.2024	15.11.2024	Experiment 3
8	18.11.2024	19.11.2024	22.11.2024	Midterm Break
9	25.11.2024	26.11.2024	29.11.2024	Experiment 4
10	02.12.2024	03.12.2024	06.12.2024	Experiment 5
11	09.12.2024	10.12.2024	13.12.2024	Experiment 6
12	16.12.2024	17.12.2024	20.12.2024	Experiment 7
13	23.12.2024	24.12.2024	27.12.2024	Experiment 8
14	30.12.2024	31.12.2024	03.01.2025	Make-up Experiment
15	06.01.2025			Quiz Exam

Lab. Rules

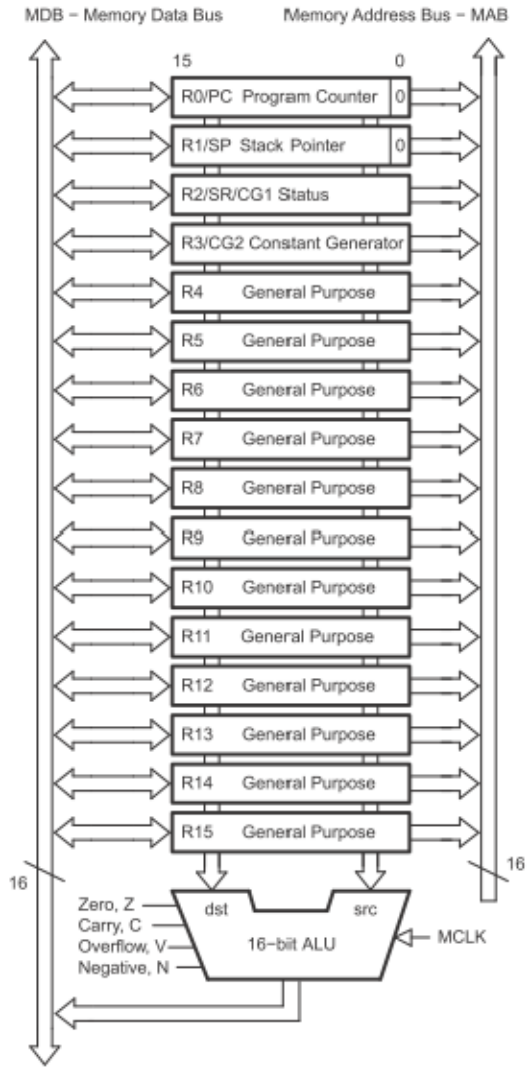
- No drinks or food is allowed to be consumed in the laboratory (except for water in a closed bottle).
- Experiment kits are given in a closed box to the students and all of groups have to return the kits in the same way to the lab. assistants.
- Static electricity load on humans can damage the kits. Therefore before touching the kits, you have to unload the possible static electricity on your body. You can do this by touching to ground pins of the outlets. **Be aware that your fingers should not be wet when touching to an outlet**.
- Before modifying the positions of the jumpers on a kit, make sure that you unplugged the USB cable of the kit from the computer. After cutting the power source, you can do the necessary changes on jumper configurations.
- If your project is not working as you expected, firstly debug your code and make sure everything is working part by part. You **must not** ask for help from the lab assistants without doing these previously.
- If you can not find the problem after debugging and self controls, then you are allowed to ask for help from lab assistants.
  You should delete the project files of the experiment on the computer before leaving the lab.

MSP430 Education Board



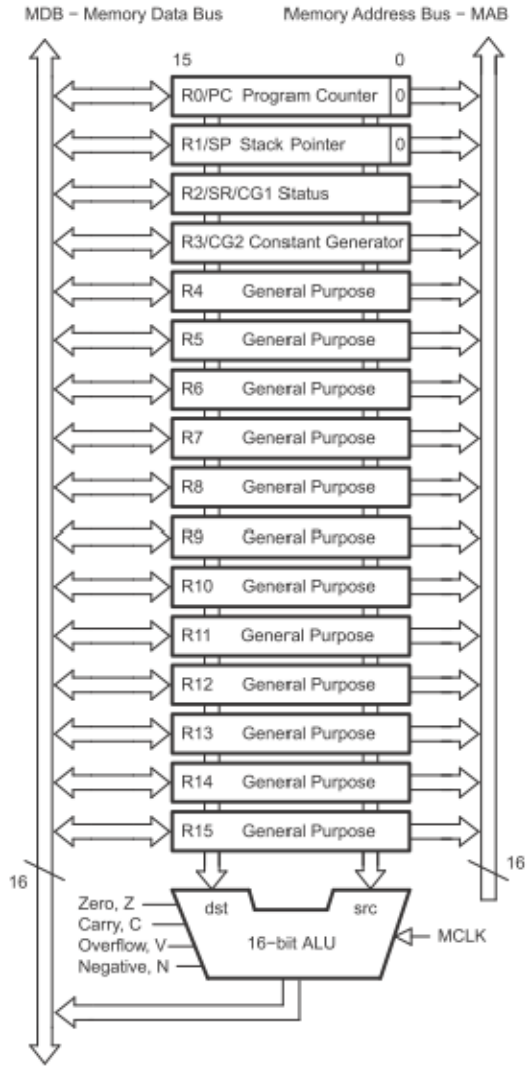
- MSP430 is a widely used low power low lost microcontroller from Texas Instruments (TI),
- TI MSP430 launchpad + external peripherals,
- TI Code Composer Studio will be used for development purposes,
- Experiments will be in assembly language,
- Architecture of the MSP430 is important.

MSP430 Architecture – I



- 16-bit RISC CPU,
- 16 16-bit registers:
  - R0-R3 dedicated registers: Program Counter, Stack Pointer, Status Register, Constant Generator,
  - R4-R15 general purpose registers.
    - Can be used for programming purposes.
- Instruction set consists of 51 instructions (27 core + 24 emulated),
  - Instruction processing on bits, bytes or words.
- 7 addressing modes for source operand and 4 addressing modes for destination operand.

MSP430 Architecture – II : Status Register



15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							V	SCG1	SCG0	OSC OFF	CPU OFF	GIE	N	Z	C
rw-0							rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Description of the bits in SR are shown below:

Bit	Description
V	Overflow bit. This bit is set when the result of an arithmetic operation overflows the signed-variable range. `ADD(.B),ADDC(.B)` Set when: Positive + Positive = Negative Negative + Negative = Positive Otherwise reset `SUB(.B),SUBC(.B),CMP(.B)` Set when: Positive – Negative = Negative Negative – Positive = Positive Otherwise reset
SCG1	System clock generator 1. When set, turns off the SMCLK.
SCG0	System clock generator 0. When set, turns off the DCO dc generator, if DCOCLK is not used for MCLK or SMCLK.
OSCOFF	Oscillator Off. When set, turns off the LFXT1 crystal oscillator, when LFXT1CLK is not use for MCLK or SMCLK.
CPUOFF	CPU off. When set, turns off the CPU.
GIE	General interrupt enable. When set, enables maskable interrupts. When reset, all maskable interrupts are disabled.
N	Negative bit. Set when the result of a byte or word operation is negative and cleared when the result is not negative. Word operation: N is set to the value of bit 15 of the result. Byte operation: N is set to the value of bit 7 of the result.
Z	Zero bit. Set when the result of a byte or word operation is 0 and cleared when the result is not 0.
C	Carry bit. Set when the result of a byte or word operation produced a carry and cleared when no carry occurred.

MSP430 Architecture – III : Addressing Modes

As/Ad	Addressing Mode	Syntax	Description
00/0	Register mode	Rn	Register contents are operand
01/1	Indexed mode	X(Rn)	(Rn + X) points to the operand. X is stored in the next word.
01/1	Symbolic mode	ADDR	(PC + X) points to the operand. X is stored in the next word. Indexed mode X(PC) is used.
01/1	Absolute mode	&ADDR	The word following the instruction contains the absolute address. X is stored in the next word. Indexed mode X(SR) is used.
10/-	Indirect register mode	@Rn	Rn is used as a pointer to the operand.
11/-	Indirect autoincrement	@Rn+	Rn is used as a pointer to the operand. Rn is incremented afterwards by 1 for .B instructions and by 2 for .W instructions.
11/-	Immediate mode	#N	The word following the instruction contains the immediate constant N. Indirect autoincrement mode @PC+ is used.

ADDRESS MODE	S	D	SYNTAX	EXAMPLE	OPERATION
Register	✓	✓	MOV Rs,Rd	MOV R10,R11	R10 -- --> R11
Indexed	✓	✓	MOV X(Rn),Y(Rm)	MOV 2(R5),6(R6)	M(2+R5) -- --> M(6+R6)
Symbolic (PC relative)	✓	✓	MOV EDE,TONI		M(EDE) -- --> M(TONI)
Absolute	✓	✓	MOV &MEM,&TCDAT		M(MEM) -- --> M(TCDAT)
Indirect	✓		MOV @Rn,Y(Rm)	MOV @R10,Tab(R6)	M(R10) -- --> M(Tab+R6)
Indirect autoincrement	✓		MOV @Rn+,Rm	MOV @R10+,R11	M(R10) -- --> R11 R10 + 2-- --> R10
Immediate	✓		MOV #X,TONI	MOV #45,TONI	#45 -- --> M(TONI)

MSP430 Architecture – IV : Instruction Set

- 27 core, 24 emulated instructions,
- Three types of instructions: dual operand, single operand, relative jump/unconditional.

INSTRUCTION FORMAT	EXAMPLE	OPERATION
Dual operands, source-destination	ADD R4,R5	R4 + R5 ---> R5
Single operands, destination only	CALL R8	PC -->(TOS), R8--> PC
Relative jump, un/conditional	JNE	Jump-on-equal bit = 0

- All single-operand and dual-operand instructions can be a byte instruction or a word instruction by using **.B** or **.W** extensions,
  - ADD.W R4, R5
  - ADD.B R4, R5

- If no extension is used, then the instruction is a word instruction.
  - ADD R4, R5

MSP430 Architecture – V : Dual-operand Instructions

Mnemonic	S-Reg, D-Reg	Operation	Status Bits			
			V	N	Z	C
MOV(.B)	src,dst	src → dst	-	-	-	-
ADD(.B)	src,dst	src + dst → dst	*	*	*	*
ADDC(.B)	src,dst	src + dst + C → dst	*	*	*	*
SUB(.B)	src,dst	dst + .not.src + 1 → dst	*	*	*	*
SUBC(.B)	src,dst	dst + .not.src + C → dst	*	*	*	*
CMP(.B)	src,dst	dst - src	*	*	*	*
DADD(.B)	src,dst	src + dst + C → dst (decimally)	*	*	*	*
BIT(.B)	src,dst	src .and. dst	0	*	*	*
BIC(.B)	src,dst	not.src .and. dst → dst	-	-	-	-
BIS(.B)	src,dst	src .or. dst → dst	-	-	-	-
XOR(.B)	src,dst	src .xor. dst → dst	*	*	*	*
AND(.B)	src,dst	src .and. dst → dst	0	*	*	*

*	The status bit is affected
–	The status bit is not affected
0	The status bit is cleared
1	The status bit is set

MSP430 Architecture – VI : Single-operand Instructions

Mnemonic	S-Reg, D-Reg	Operation	Status Bits			
			V	N	Z	C
RRC(.B)	dst	C → MSB →.....LSB → C	*	*	*	*
RRA(.B)	dst	MSB → MSB →....LSB → C	0	*	*	*
PUSH(.B)	src	SP – 2 → SP, src → @SP	-	-	-	-
SWPB	dst	Swap bytes	-	-	-	-
CALL	dst	SP – 2 → SP, PC+2 → @SP dst → PC	-	-	-	-
RETI		TOS → SR, SP + 2 → SP TOS → PC,SP + 2 → SP	*	*	*	*
SXT	dst	Bit 7 → Bit 8.....Bit 15	0	*	*	*

*	The status bit is affected
–	The status bit is not affected
0	The status bit is cleared
1	The status bit is set

MSP430 Architecture – VII : Jump Instructions

Mnemonic	S-Reg, D-Reg	Operation
JEQ/JZ	Label	Jump to label if zero bit is set
JNE/JNZ	Label	Jump to label if zero bit is reset
JC	Label	Jump to label if carry bit is set
JNC	Label	Jump to label if carry bit is reset
JN	Label	Jump to label if negative bit is set
JGE	Label	Jump to label if (N .XOR. V) = 0
JL	Label	Jump to label if (N .XOR. V) = 1
JMP	Label	Jump to label unconditionally

- For the complete instruction set and detailed explanations check:
  - *MSP430_Instruction_Set_short*,
  - *MSP430_Instruction_Set_in_detail* files in Ninova.

MSP430 Architecture – VIII : Memory Organization

		MSP430G2153	MSP430G2253 MSP430G2213	MSP430G2353 MSP430G2313	MSP430G2453 MSP430G2413	MSP430G2553 MSP430G2513
Memory	Size	1kB	2kB	4kB	8kB	16kB
Main: interrupt vector	Flash	0xFFFF to 0xFFC0	0xFFFF to 0xFFC0	0xFFFF to 0xFFC0	0xFFFF to 0xFFC0	0xFFFF to 0xFFC0
Main: code memory	Flash	0xFFFF to 0xFC00	0xFFFF to 0xF800	0xFFFF to 0xF000	0xFFFF to 0xE000	0xFFFF to 0xC000
Information memory	Size	256 Byte	256 Byte	256 Byte	256 Byte	256 Byte
	Flash	010FFh to 01000h	010FFh to 01000h	010FFh to 01000h	010FFh to 01000h	010FFh to 01000h
RAM	Size	256 Byte	256 Byte	256 Byte	512 Byte	512 Byte
		0x02FF to 0x0200	0x02FF to 0x0200	0x02FF to 0x0200	0x03FF to 0x0200	0x03FF to 0x0200
Peripherals	16-bit	01FFh to 0100h	01FFh to 0100h	01FFh to 0100h	01FFh to 0100h	01FFh to 0100h
	8-bit	0FFh to 010h	0FFh to 010h	0FFh to 010h	0FFh to 010h	0FFh to 010h
	8-bit SFR	0Fh to 00h	0Fh to 00h	0Fh to 00h	0Fh to 00h	0Fh to 00h

MSP430G2553 has 16KB of memory, where:

- Interrupt vectors are located between 0xFFFF to 0xFFC0,
- Code memory is located between 0xFFFF to 0xC000,
- Information memory between 0x10FF to 0x1000,
- RAM is between 0x03FF to 0x0200.

- For more detailed information on MSP430 architecture check:
  - *Supplementary_Chapter_4_MSP430_ Architecture*.

MSP430 Architecture – IX: General Purpose Input/Output

- There are two general purpose Input/Output (I/O) ports: **P1** and **P2**,
  - Each of the ports can be configured interruptible or not,
  - Each I/O lines of each port can be configured as either input or output,
  - Configuration of the ports and their operation can be defined by registers:

    - Direction Registers (**PxDIR**),
    - Input Registers (**PxIN**),
    - Output Registers (**PxOUT**),
    - Pull-up/Pull-down Resistor Enable Registers (**PxREN**),
    - Function Select Registers: (**PxSEL**) and (**PxSEL2**).

- **Direction Registers (PxDIR)**:
  - 8-bit read-write register,
  - Selects the direction of the I/O pin of the port,
  - Configuration:
    - Bit: 1 => the pin is set up as an output,
    - Bit: 0 => the pin is set up as an input.

- **Input Registers (PxIN)**:
  - 8-bit read-only register,
  - Reflects the input signal at the I/O pin of the port,
    - Bit: 1 => input is high,
    - Bit: 0 => input is low.

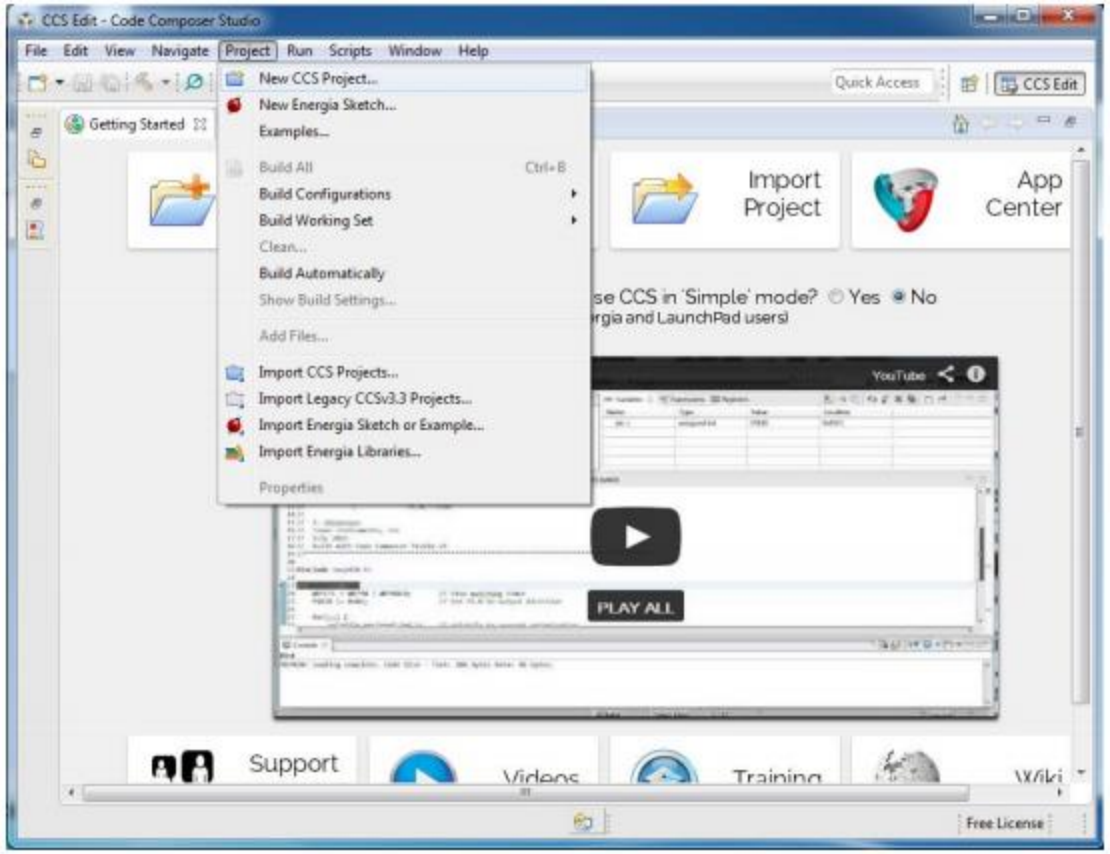
- **Output Registers (PxOUT)**:
  - 8-bit read-write register,
  - Reflects the value written to the output pin,
    - Bit: 1 => output is high,
    - Bit: 0 => output is low.

- For detailed information, check:
  - *Supplementary_Chapter_6_General _purpose _IO* at Ninova.

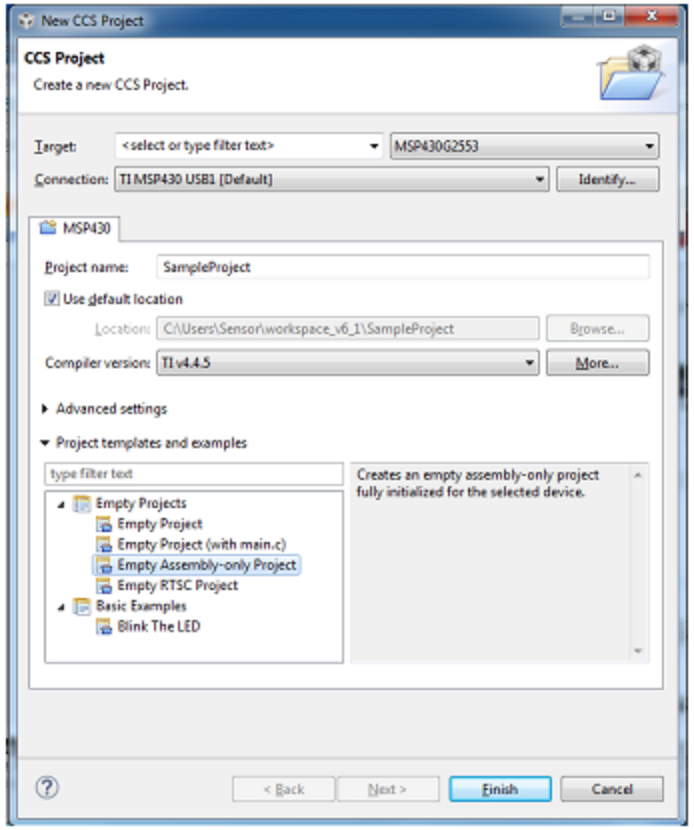
Please read *MSP430_introduction* document on Ninova!

Demonstration and Assembly Information

Code Composer Studio Project Creation Step 1



Code Composer Studio Project Creation Step II



Code Composer Studio Empty Assembly Project

```
;-------------------------------------------------------------------------------
; MSP430 Assembler Code Template for use with TI Code Composer Studio
;
;-------------------------------------------------------------------------------
            .cdecls C,LIST,"msp430.h"       ; Include device header file
;-------------------------------------------------------------------------------
            .def    RESET                   ; Export program entry-point to
                                            ; make it known to linker.
;-------------------------------------------------------------------------------
            .text                           ; Assemble into program memory.
            .retain                         ; Override ELF conditional linking
                                            ; and retain current section.
            .retainrefs                     ; And retain any sections that have
                                            ; references to current section.
;-------------------------------------------------------------------------------
RESET       mov.w   #__STACK_END,SP         ; Initialize stackpointer
StopWDT     mov.w   #WDTPW|WDTHOLD,&WDTCTL  ; Stop watchdog timer


;-------------------------------------------------------------------------------
; Main loop here
;-------------------------------------------------------------------------------

;-------------------------------------------------------------------------------
; Stack Pointer definition
;-------------------------------------------------------------------------------
            .global __STACK_END
            .sect   .stack
;-------------------------------------------------------------------------------
; Interrupt Vectors
;-------------------------------------------------------------------------------
            .sect   ".reset"                ; MSP430 RESET Vector
            .short  RESET
```

- MSP430 source code programs are a sequence of statements,
  - Assembly directives,
  - Assembly instructions,
  - Macros,
  - Comments.
- A line can have four fields in the following syntax:

  [label[:]] mnemonic [operand list] [;comment]
- Comments start by: **;**

Code Composer Guidelines

```
;-------------------------------------------------------------------------------
; MSP430 Assembler Code Template for use with TI Code Composer Studio
;
;-------------------------------------------------------------------------------
            .cdecls C,LIST,"msp430.h"       ; Include device header file
;-------------------------------------------------------------------------------
            .def    RESET                   ; Export program entry-point to
                                            ; make it known to linker.
;-------------------------------------------------------------------------------
            .text                           ; Assemble into program memory.
            .retain                         ; Override ELF conditional linking
                                            ; and retain current section.
            .retainrefs                     ; And retain any sections that have
                                            ; references to current section.
;-------------------------------------------------------------------------------
RESET       mov.w   #__STACK_END,SP         ; Initialize stackpointer
StopWDT     mov.w   #WDTPW|WDTHOLD,&WDTCTL  ; Stop watchdog timer


;-------------------------------------------------------------------------------
; Main loop here
;-------------------------------------------------------------------------------

;-------------------------------------------------------------------------------
; Stack Pointer definition
;-------------------------------------------------------------------------------
            .global __STACK_END
            .sect   .stack
;-------------------------------------------------------------------------------
; Interrupt Vectors
;-------------------------------------------------------------------------------
            .sect   ".reset"                ; MSP430 RESET Vector
            .short  RESET
```

- The general guidelines for writing the code are:
  - All statements must begin with a label, a blank, an asterisk, or a semicolon,
  - Labels are optional and if used, they must begin in column 1,
  - One or more blanks must separate each field. Tab and space characters are blanks. You must separate the operand list from the preceding field with a blank,
  - Comments are optional. Comments that begin in column 1 can begin with an asterisk or a semicolon (* or ;), but comments that begin in any other column must begin with a semicolon,
  - A mnemonic cannot begin in column 1, as it will be interpreted as a label.

Code Composer Guidelines

```
;-------------------------------------------------------------------------------
; MSP430 Assembler Code Template for use with TI Code Composer Studio
;
;-------------------------------------------------------------------------------
            .cdecls C,LIST,"msp430.h"       ; Include device header file
;-------------------------------------------------------------------------------
            .def    RESET                   ; Export program entry-point to
                                            ; make it known to linker.
;-------------------------------------------------------------------------------
            .text                           ; Assemble into program memory.
            .retain                         ; Override ELF conditional linking
                                            ; and retain current section.
            .retainrefs                     ; And retain any sections that have
                                            ; references to current section.
;-------------------------------------------------------------------------------
RESET       mov.w   #__STACK_END,SP         ; Initialize stackpointer
StopWDT     mov.w   #WDTPW|WDTHOLD,&WDTCTL  ; Stop watchdog timer


;-------------------------------------------------------------------------------
; Main loop here
;-------------------------------------------------------------------------------

;-------------------------------------------------------------------------------
; Stack Pointer definition
;-------------------------------------------------------------------------------
            .global __STACK_END
            .sect   .stack
;-------------------------------------------------------------------------------
; Interrupt Vectors
;-------------------------------------------------------------------------------
            .sect   ".reset"                ; MSP430 RESET Vector
            .short  RESET
```

- The general guidelines for writing the code are:
  - All statements must begin with a label, a blank, an asterisk, or a semicolon,
  - Labels are optional and if used, they must begin in column 1,
  - One or more blanks must separate each field. Tab and space characters are blanks. You must separate the operand list from the preceding field with a blank,
  - Comments are optional. Comments that begin in column 1 can begin with an asterisk or a semicolon (* or ;), but comments that begin in any other column must begin with a semicolon,
  - A mnemonic cannot begin in column 1, as it will be interpreted as a label.

Code Composer Studio Sample Assembly Project

- Place the code below starting from the ";Main loop here" line

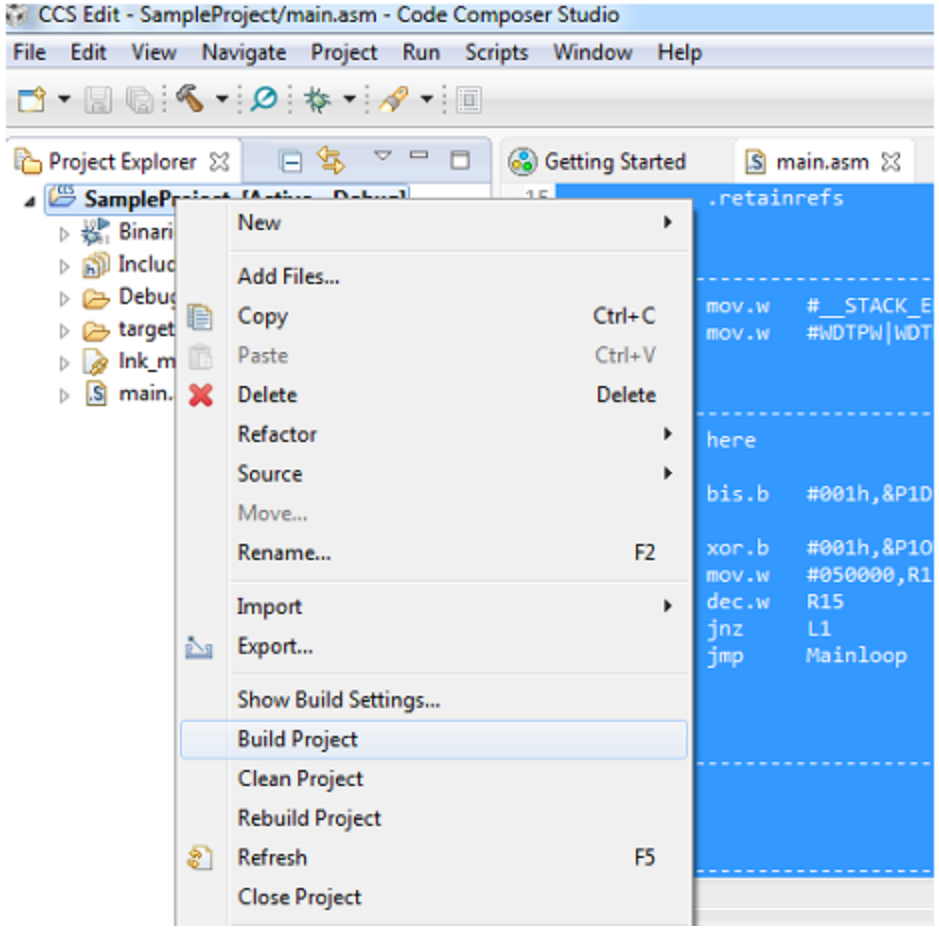
```
SetupP1     bis.b #001h,&P1DIR  ; P1.0 output
                                ;
Mainloop    xor.b #001h,&P1OUT  ; Toggle P1.0
Wait        mov.w #050000,R15   ; Delay to R15
L1          dec.w R15           ; Decrement R15
            jnz L1              ; Delay over?
            jmp Mainloop        ; Again
```

- Disable the compiler optimizations from Project's Properties->Build->MSP430 Compiler->Optimization

Code Composer Studio Sample Assembly Project

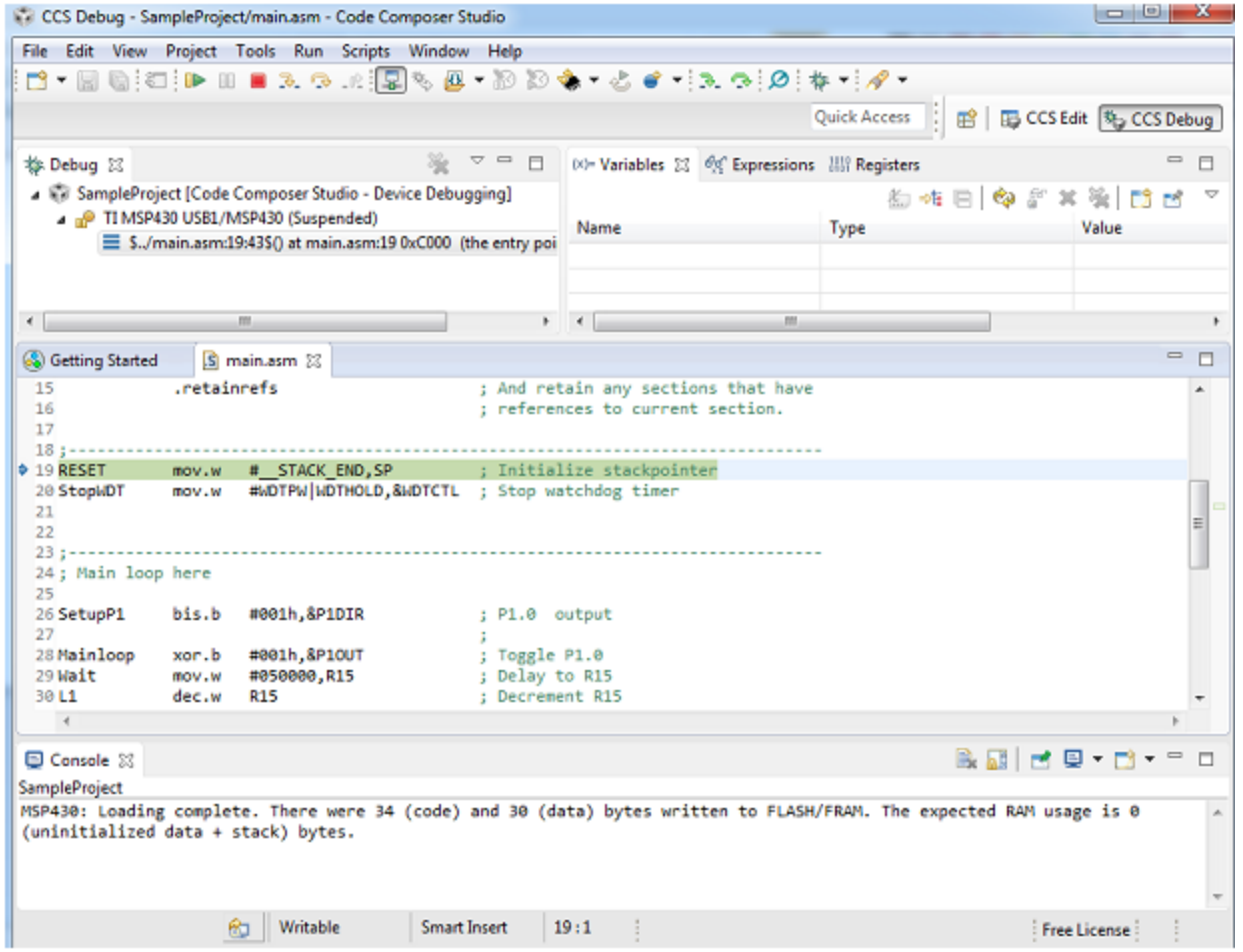
```
;-------------------------------------------------------------------------------
; MSP430 Assembler Code Template for use with TI Code Composer Studio
;-------------------------------------------------------------------------------
            .cdecls C,LIST,"msp430.h"       ; Include device header file

;-------------------------------------------------------------------------------
            .def    RESET                   ; Export program entry-point to
                                            ; make it known to linker.
;-------------------------------------------------------------------------------
            .text                           ; Assemble into program memory.
            .retain                         ; Override ELF conditional linking
                                            ; and retain current section.
            .retainrefs                     ; And retain any sections that have
                                            ; references to current section.

;-------------------------------------------------------------------------------
RESET       mov.w   #__STACK_END,SP         ; Initialize stackpointer
StopWDT     mov.w   #WDTPW|WDTHOLD,&WDTCTL  ; Stop watchdog timer

;-------------------------------------------------------------------------------

SetupP1     bis.b   #001h,&P1DIR            ; P1.0  output
                                            ;
Mainloop    xor.b   #001h,&P1OUT            ; Toggle P1.0
Wait        mov.w   #050000,R15             ; Delay to R15
L1          dec.w   R15                     ; Decrement R15
            jnz     L1                      ; Delay over?
            jmp     Mainloop                ; Again

;-------------------------------------------------------------------------------

;-------------------------------------------------------------------------------
; Stack Pointer definition
;-------------------------------------------------------------------------------
            .global __STACK_END
            .sect   .stack

;-------------------------------------------------------------------------------
; Interrupt Vectors
;-------------------------------------------------------------------------------
            .sect   ".reset"                ; MSP430 RESET Vector
            .short  RESET
```

Code Composer Studio Sample Assembly Project Building

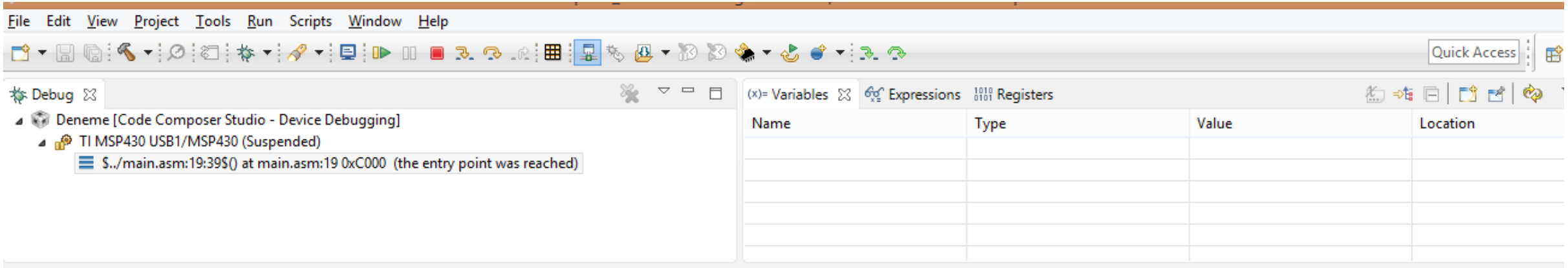


Code Composer Studio Sample Assembly Project Debugging and Loading

- In order to load your program and start debugging Press F11 or select Run->Debug



Code Composer Studio Sample Assembly Project Debugging and Loading



- When the program is loaded to the board, then we have chance run the assembly code step by step by either pressing F5 on the keyboad or using the debugging menu (i.e., step into, step over),
- You can view the contents of the all registers by Registers tab,
- You can add breakpoints by either menü on the right or by double clicking the line number within the assembly code.