Teaching and classroom laboratories based on the “eZ430” and "Experimenter’s board" MSP430 microcontroller platforms and Code Composer Essentials

Collection Editors:

Pedro Dinis

António Espírito Santo

Teaching and classroom laboratories based on the “eZ430” and "Experimenter’s board" MSP430 microcontroller platforms and Code Composer Essentials

Collection Editors:

Pedro Dinis
António Espírito Santo

Authors:

Pedro Dinis
António Espírito Santo
Bruno Ribeiro

Online:

< http://cnx.org/content/col10706/1.3/ >

C O N N E X I O N S

Rice University, Houston, Texas

Collection structure revised: May 19, 2009
PDF generated: October 26, 2012

Table of Contents

Chapter 1

MSP430 Overview

1.1 Introduction[1]

Introduction

The types of devices such as microprocessor, microcontroller, processor, digital signal processor (DSP), amongst others, in a certain manner, are related to the same device – the ASIC (Application Specific Integrated Circuit). Each processing device executes instructions, following a determined program applied to the inputs and shares architectural characteristics developed from the first microprocessors created in 1971. In the three decades after the development of the first microprocessor, huge developments and innovations have been made in this engineering field. Any of the terms used at the beginning of this section are correct to define a microprocessor, although each one has different characteristics and applications.

The definition of a microcontroller is somewhat difficult due to the constantly changing nature of the silicon industry. What we today consider a microcontroller with medium capabilities is several orders of magnitude more powerful, than the computer used on the first space missions. Nevertheless, some generalizations can be made as to what characterizes a microcontroller. Typically, microcontrollers are selected for embedded systems projects, i.e., control systems with a limited number of inputs and outputs where the controller is *embedded* into the system.

The programmable SoC (system-on-chip) concept started in 1972 with the 4-bit TMS1000 microcomputer developed by Texas Instruments (TI), and in those days it was ideal for applications such as calculators and ovens. This term was changed to Microcontroller Unit (MCU), which was more descriptive of a typical application. Nowadays, MCUs are at the heart of many physical systems, with higher levels of integration and processing power at lower power consumption.

The following list presents several qualities that define a microcontroller:

- Cost: Usually, the microcontrollers are high-volume, low cost devices;
- Clock frequency: Compared with other devices (microprocessors and DSPs), microcontrollers use a low clock frequency. Microcontrollers today can run up to 100 MHz/ 100 Million Instructions Per Second (MIPS)
- Power consumption: orders of magnitude lower than their DSP and MPU cousins;
- Bits: 4 bits (older devices) to 32 bits devices;
- Memory: Limited available memory, usually less than 1 MByte;
- Input/Output (I/O): Low to high (8-150) pin-out count.

Request the MSP430 Teaching ROM Materials here https://www-a.ti.com/apps/dspuniv/teaching_rom_request.asp[2]

[1] This content is available online at <http://cnx.org/content/m23492/1.1/>.

[2] https://www-a.ti.com/apps/dspuniv/teaching_rom_request.asp

1.2 MSP430 Main characteristics[3]

MSP430 Main characteristics
Although there are variants in devices in the family, a MSP430 microcontroller can be characterized by:
- Low power consumption:

- 0.1 μA for RAM data retention;
- 0.8 μA for real time clock mode operation;
- 250 μA/MIPS at active operation.

Low operation voltage (from 1.8 V to 3.6 V).
$<$ 1 μs clock start-up.
$<$ 50 nA port leakage.
Zero-power Brown-Out Reset (BOR).
On-chip analogue devices:

- 10/12/16-bit Analogue-to-Digital Converter (ADC);
- 12-bit dual Digital-to-Analogue Converter (DAC);
- Comparator-gated timers;
- Operational Amplifiers (OP Amps);
- Supply Voltage Supervisor (SVS).

16 bit RISC CPU:

- Instructions processing on either bits, bytes or words;
- Compact core design reduces power consumption and cost;
- Compiler efficient;
- 27 core instructions;
- 7 addressing modes;
- Extensive vectored-interrupt capability.

Flexibility:

- Up to 256 kB In-System Programmable (ISP) Flash;
- Up to 100 pin options;
- USART, I2C, Timers;
- LCD driver;
- Embedded emulation.

The microcontroller's performance is directly related to the 16-bit data bus, the 7 addressing modes and the reduced instructions set, which allows a shorter, denser programming code for fast execution. These microcontroller families share a 16-bit CPU (Central Processing Unit) core, RISC type, intelligent peripherals, and flexible clock system that interconnects using a Von Neumanncommon memory address bus (MAB) and memory data bus (MDB) architecture.

[3]This content is available online at <http://cnx.org/content/m23490/1.2/>.

MSP430 architecture.

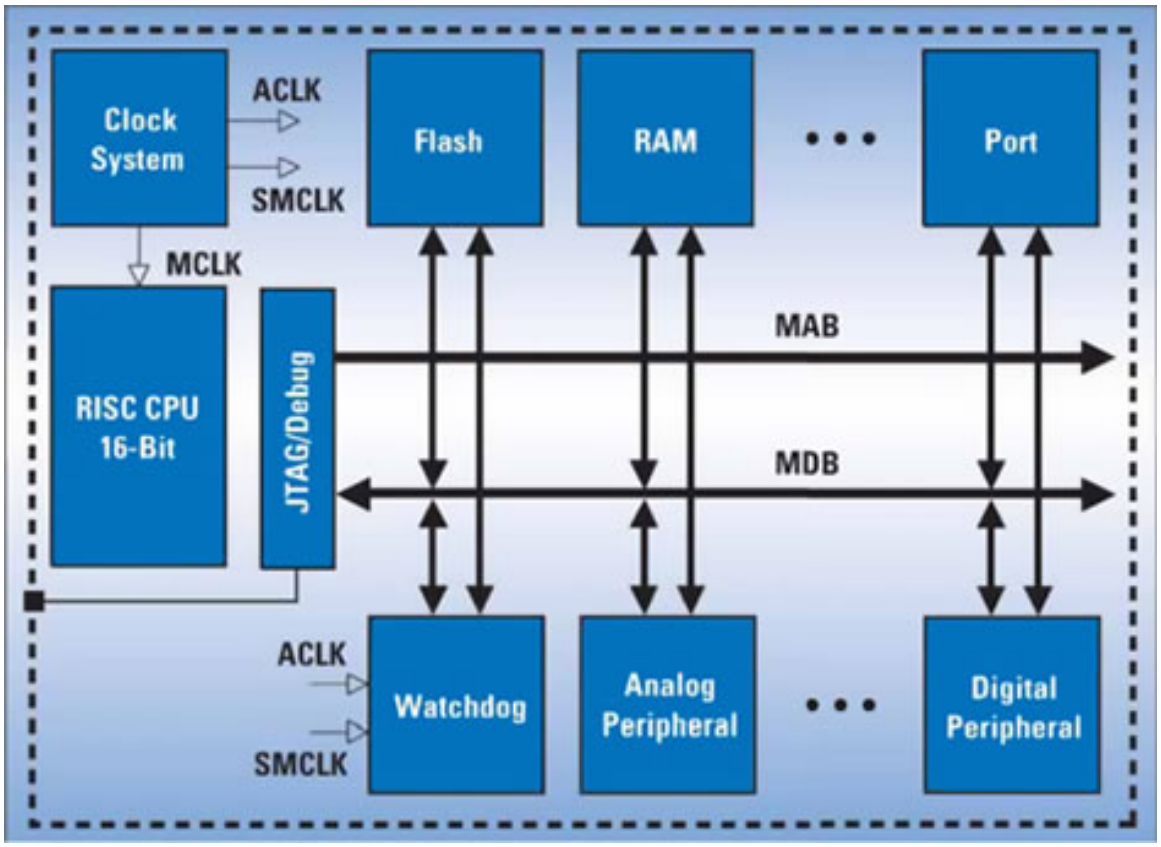

Figure 1.1

Request the MSP430 Teaching ROM Materials here https://www-a.ti.com/apps/dspuniv/teaching_rom_request.asp[4]

1.3 Address space[5]

Address space

All memory, including RAM, Flash/ROM, information memory, special function registers (SFRs), and peripheral registers are mapped into a single, contiguous address space.

Note: See the device-specific datasheets for specific memory maps. Code access is always performed on even addresses. Data can be accessed as bytes or words.

The MSP430 is available with either Flash or ROM memory types. The memory type is identified by the letter immediately following "MSP430" in the part numbers.

Flash devices: Identified by the letter "F" in the part numbers, having the advantage that the code space can be erased and reprogrammed.

[4] https://www-a.ti.com/apps/dspuniv/teaching_rom_request.asp
[5] This content is available online at <http://cnx.org/content/m23495/1.1/>.

ROM devices: Identified by the letter "C" in the part numbers. They have the advantage of being very inexpensive because they are shipped pre-programmed, which is the best solution for high-volume designs.

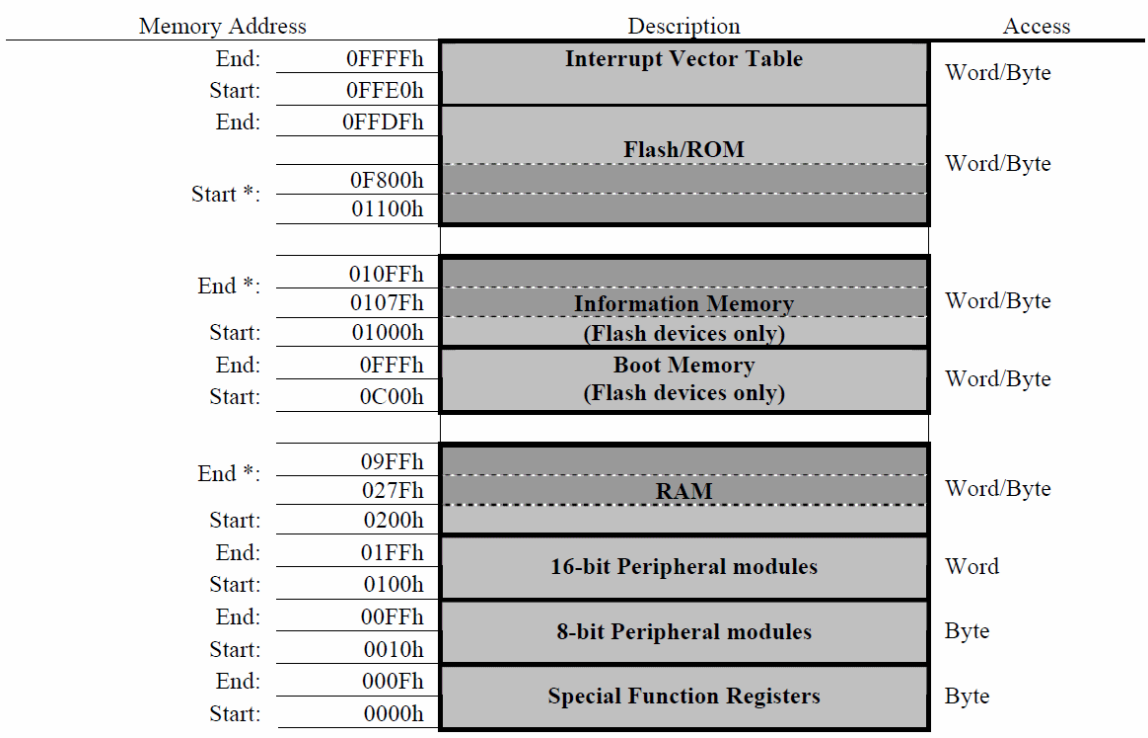

Figure 1.2

* Depending on device family.

For all devices, each memory location is formed by 1 data byte. The CPU is capable of addressing data values either as bytes (8 bits) or words (16 bits). Words are always addressed at an even address, which contain the least significant byte, followed by the next odd address, which contains the most significant byte. For 8-bit operations, the data can be accessed from either odd or even addresses, but for 16-bit operations, the data values can only be accessed from even addresses.

1.3.1 Interrupt vector table

The interrupt vector table is mapped at the very end of memory space (upper 16 words of Flash/ROM), in locations 0FFE0h through to 0FFFEh (see the device-specific datasheets). The priority of the interrupt vector increases with the word address.

Interrupt vector table for MSP430 families.

Vector Address	Priority	**' 11xx and ' 12xx**	**' 13x and ' 14x**	'2xx	**' 3xx**	**' 4xx**
0xFFFE	31, Highest	Hard Reset/ Watchdog	Hard Reset/ Watchdog	Hard Reset/ Watchdog	Hard Reset/ Watchdog	Hard Reset/ Watchdog
0xFFFC	30	Oscillator/ Flash/NMI	Oscillator/ Flash/NMI	Oscillator/ Flash/NMI	Oscillator/ Flash/NMI	Oscillator/ Flash/NMI
0xFFFA	29	Unused	Timer_B	Timer_B (22x2, 22x4, 23x, 24x, 26x only)	Dedicated I/O	Timer_B('43x and'44x only)
0xFFF8	28	Unused	Timer_B	Timer_B (22x2, 22x4, 23x, 24x only)	Dedicated I/O	Timer_B('43x and'44x only)
0xFFF6	27	Comparator	Comparator	Comparator_A (20x1 only, 21x1, 23x, 24x, 26x)	Unused	Comparator
0xFFF4	26	Watchdog Timer	Watchdog Timer	Watchdog Timer+	Watchdog Timer	Watchdog Timer
0xFFF2	25	Timer_A	USART Rx	Timer_A	Timer_A	USART0 Rx('43x and'44x only)
0xFFF0	24	Timer_A	USART0 Tx	Timer_A	Timer_A	USART0 Tx('43x and'44x only)
0xFFEE	23	USART0 Rx ('12xx only)	ADC	USCI Rx(22x2, 22x4, 23x, 24x, 26x only)I2C status (23x, 24x)	USART Rx	ADC('43x and'44x only)

continued on next page

0xFFEC	22	USART0 Tx ('12xx only)	Timer_A	USCI Tx(22x2, 22x4, 23x, 24x, 26x only)I2C Rx/Tx (23x, 24x, 26x only)	USART Tx	Timer_A
0xFFEA	21	ADC10	Timer_A	ADC10 (20x2 22x2, 22x4 only)ADC12 (23x, 24x, 26x only)SD16_A (20x3 only)	ADC('32x and '33x) Timer/Port ('31x)	Timer_A
0xFFE8	20	Unused	Port 1	USI(20x2, 20x3 only)	Timer/Port('32x and '33x)	Port 1
0xFFE6	19	Port 2	USART1 Rx	Port P2	Port 2	USART1 Rx('44x only)
0xFFE4	18	Port 1	USART1 Tx	Port P1	Port 1	USART1 Tx('44x only)
0xFFE2	17	Unused	Port 2	USCI Rx (23x, 24x, 26x only)I2C status (241x, 261x only)	Basic Timer	Port 2
0xFFE0	16	Unused	Unused	USCI Tx (23x,24x only)I2C Rx/Tx (241x, 261x only)	Port 0	Basic Timer

continued on next page

	15			DMA (241x, 261x only)		
	14			DAC12 (241x, 261 only)		
	13 to 0Lowest			Reserved		

Table 1.1

1.3.2 Flash/ROM

The start address of Flash/ROM depends on the amount of Flash/ROM present on the device. The start address varies between 01100h (60k devices) to 0F800h (2k devices) and always runs to the end of the address space at location 0FFFFh. Flash can be used for both code and data. Word or byte tables can also be stored and read by the program from Flash/ROM.

All code, tables, and hard-coded constants reside in this memory space.

1.3.3 Information memory (Flash devices only)

The MSP430 flash devices contain an address space for information memory. It is like an onboard EEPROM, where variables needed for the next power up can be stored during power down. It can also be used as code memory. Flash memory may be written one byte or word at a time, but must be erased in segments. The information memory is divided into two 128-byte segments. The first of these segments is located at addresses 01000h through to 0107Fh (Segment B), and the second is at address 01080h through to 010FFh (Segment A). This is the case in 4xx devices. It is 256 bytes (4 segments of 64 bytes each) in 2xx devices.

1.3.4 Boot memory (Flash devices only)

The MSP430 flash devices contain an address space for boot memory, located between addresses 0C00h through to 0FFFh. The "bootstrap loader" is located in this memory space, which is an external interface that can be used to program the flash memory in addition to the JTAG. This memory region is not accessible by other applications, so it cannot be overwritten accidentally. The bootstrap loader performs some of the same functions as the JTAG interface (excepting the security fuse programming), using the TI data structure protocol for UART communication at a fixed data rate of 9600 baud.

1.3.5 RAM

RAM always starts at address 0200h. The end address of RAM depends on the amount of RAM present on the device. RAM is used for both code and data.

1.3.6 Peripheral Modules

Peripheral modules consist of all on-chip peripheral registers that are mapped into the address space. These modules can be accessed with byte or word instructions, depending if the peripheral module is 8-bit or 16-bit respectively. The 16-bit peripheral modules are located in the address space from addresses 0100 through to 01FFh and the 8-bit peripheral modules are mapped into memory from addresses 0010h through to 00FFh.

1.3.7 Special Function Registers (SFRs)

Some peripheral functions are mapped into memory with special dedicated functions. The Special Function Registers (SFRs) are located at memory addresses from 0000h to 000Fh, and are the specific registers for:

- Interrupt enables (locations 0000h and 0001h);
- Interrupt flags (locations 0002h and 0003h);
- Enable flags (locations 0004h and 0005h).

SFRs must be accessed using byte instructions only. See the device-specific data sheets for the applicable SFR bits.

Request the MSP430 Teaching ROM Materials here https://www-a.ti.com/apps/dspuniv/teaching_rom_request.asp[6]

1.4 Central Processing Unit (MSP430 CPU)[7]

1.4.1 Central Processing Unit (MSP430 CPU)

The RISC type architecture of the CPU is based on a short instruction set (27 instructions), interconnected by a 3-stage instruction pipeline for instruction decoding. The CPU has a 16-bit ALU, four dedicated registers and twelve working registers, which makes the MSP430 a high performance microcontroller suitable for low power applications. The addition of twelve working general purpose registers saves CPU cycles by allowing the storage of frequently used values and variables instead of using RAM.

The orthogonal instruction set allows the use of any addressing mode for any instruction, which makes programming clear and consistent, with few exceptions, increasing the compiler efficiency for high-level languages such as C.

[6] https://www-a.ti.com/apps/dspuniv/teaching_rom_request.asp
[7] This content is available online at <http://cnx.org/content/m23497/1.1/>.

MSP430 CPU block diagram.

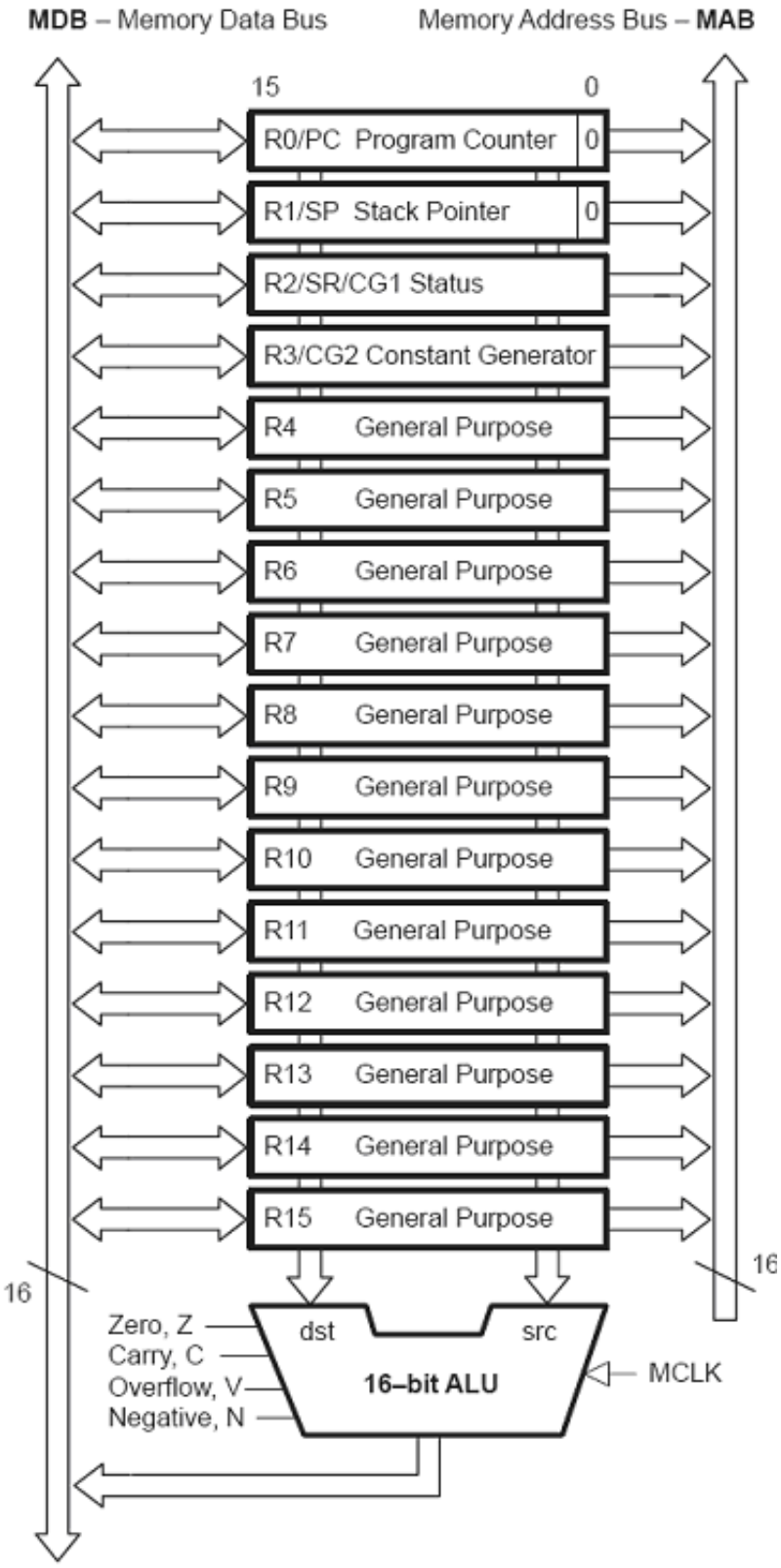

Figure 1.3

1.4.1.1 Arithmetic Logic Unit (ALU)

The MSP430 CPU includes an arithmetic logic unit (ALU) that handles addition, subtraction, comparison and logical (AND, OR, XOR) operations. ALU operations can affect the overflow, zero, negative, and carry flags in the status register.

1.4.1.2 MSP430 CPU registers

The CPU incorporates sixteen 16-bit registers:
- Four registers (R0, R1, R2 and R3) have dedicated functions;
- There are 12 working registers (R4 to R15) for general use.

1.4.1.2.1 R0: Program Counter (PC)

The 16-bit Program Counter (PC/R0) points to the next instruction to be read from memory and executed by the CPU. The Program counter is implemented by the number of bytes used by the instruction (2, 4, or 6 bytes, always even). It is important to remember that the PC is aligned at even addresses, because the instructions are 16 bits, even though the individual memory addresses contain 8-bit values.

1.4.1.2.2 R1: Stack Pointer (SP)

The Stack Pointer (SP/R1) is located in R1.

1st: stack can be used by user to store data for later use (instructions: store by PUSH, retrieve by POP);

2nd: stack can be used by user or by compiler for subroutine parameters (PUSH, POP in calling routine; addressed via offset calculation on stack pointer (SP) in called subroutine);

3rd: used by subroutine calls to store the program counter value for return at subroutine's end (RET);

4th: used by interrupt - system stores the actual PC value first, then the actual status register content (on top of stack) on return from interrupt (RETI) the system get the same status as just before the interrupt happened (as long as none has changed the value on TOS) and the same program counter value from stack.

1.4.1.2.3 R2: Status Register (SR)

The Status Register (SR/R2) stores the state and control bits. The system flags are changed automatically by the CPU depending on the result of an operation in a register. The reserved bits of the SR are used to support the constants generator. See the device-specific data sheets for more details.

SR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved for CG1							V	SCG1	SCG0	OSCOFF	CPUOFF	GIE	N	Z	C

Table 1.2

Bit		Description
8	V	Overflow bit.V = 1 ⇒ Result of an arithmetic operation overflows the signed-variable range.

continued on next page

7	SCG1	System clock generator 0.SCG1 = 1 ⇒ DCO generator is turned off – if not used for MCLK or SMCLK.
6	SCG0	System clock generator 1.SCG0 = 1 ⇒ FLL+ loop control is turned off.
5	OSCOFF	Oscillator Off.OSCOFF = 1 ⇒ turns off LFXT1 when it is not used for MCLK or SMCLK.
4	CPUOFF	CPU off.CPUOFF = 1 ⇒ disable CPU core.
3	GIE	General interrupt enable.GIE = 1 ⇒ enables maskable interrupts.
2	N	Negative flag.N = 1 ⇒ result of a byte or word operation is negative.
1	Z	Zero flag.Z = 1 ⇒ result of a byte or word operation is 0.
0	C	Carry flag.C = 1 ⇒ result of a byte or word operation produced a carry.

Table 1.3

R2/R3: Constant Generator Registers (CG1/CG2)

Depending of the source-register addressing modes (As) value, six commonly used constants can be generated without a code word or code memory access to retrieve them.

This is a very powerful feature, which allows the implementation of emulated instructions, for example, instead of implementing a core instruction for an increment, the constant generator is used.

Register	As	Constant	Remarks
R2	00	-	Register mode
R2	01	(0)	Absolute mode
R2	10	00004h	+4, bit processing
R2	11	00008h	+8, bit processing
R3	00	00000h	0, word processing
R3	01	00001h	+1
R3	10	00002h	+2, bit processing
R3	11	0FFFFh	-1, word processing

Table 1.4