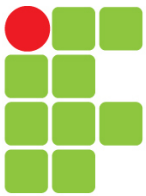
Microcontroladores

Introdução ao microcontrolador Texas MSP430

Prof. Renan Augusto Starke

Instituto Federal de Santa Catarina – IFSC
Campus Florianópolis
renan.starke@ifsc.edu.br

13 de fevereiro de 2020



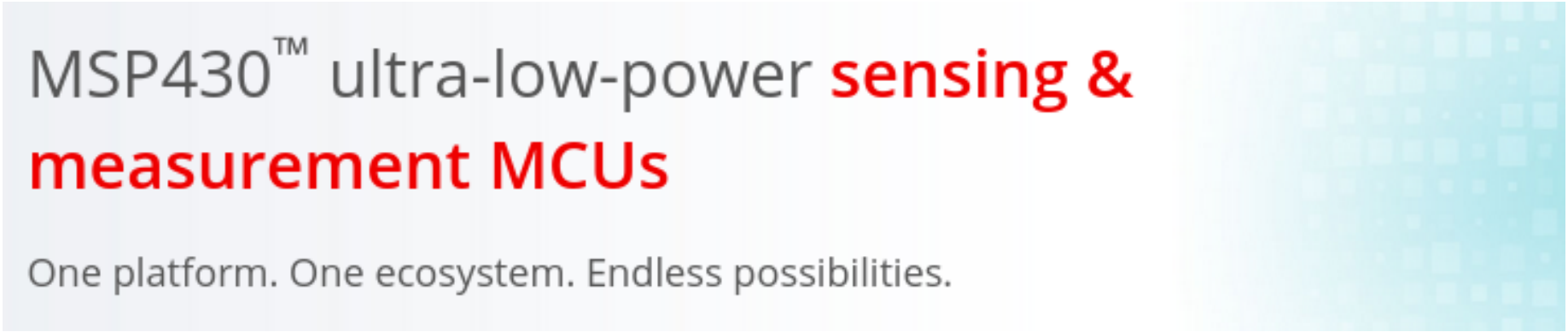
INSTITUTO FEDERAL
SANTA CATARINA

Ministério da Educação
Secretaria de Educação Profissional e Tecnológica
INSTITUTO FEDERAL DE SANTA CATARINA

Tópicos

Tópico

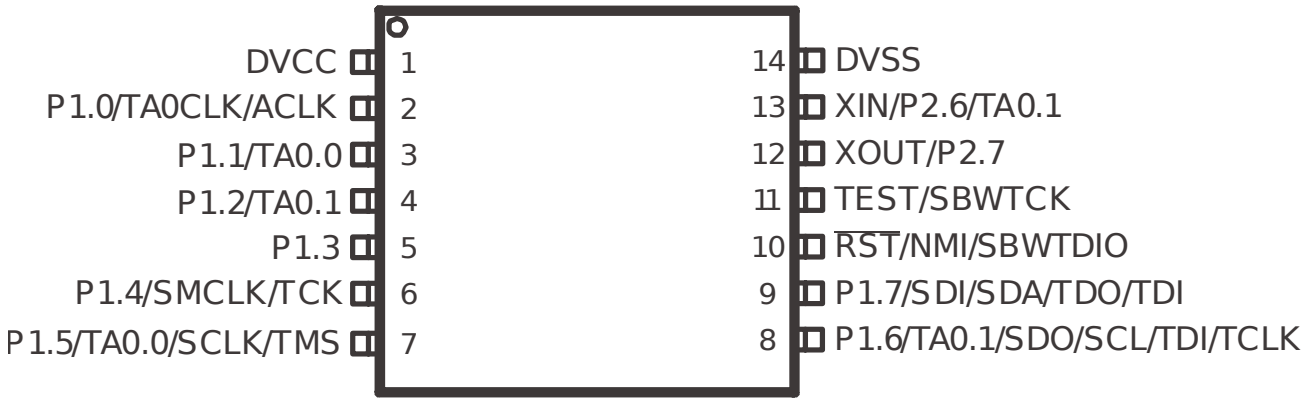
Texas Instruments MSP430

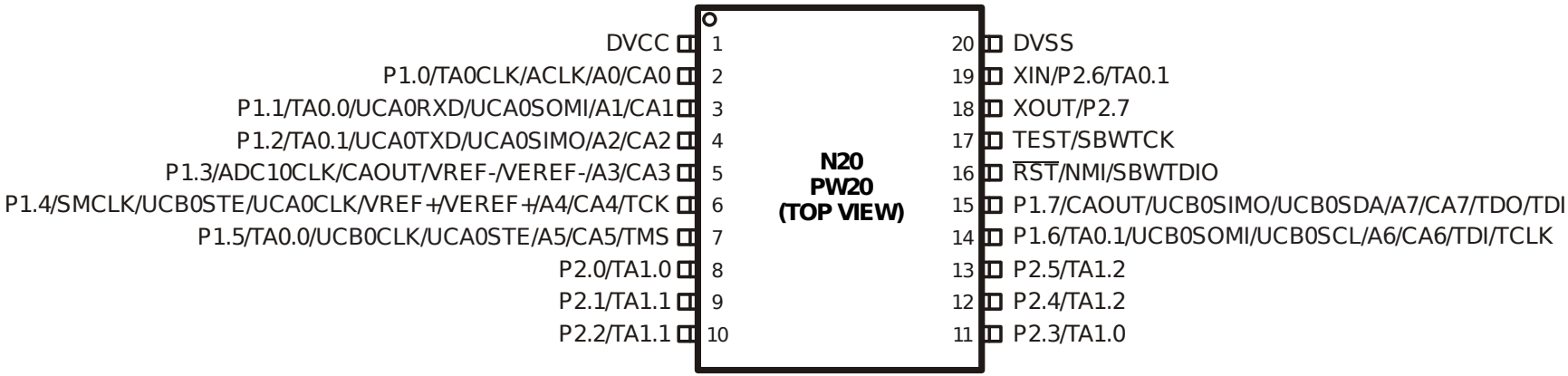


- MSP430 é muito utilizado em aplicações embarcadas com restrições de energia.
- Corrente em *idle* pode ser menor que $\mathbf{1}\ \mu A$
- Velocidade do núcleo até 25 MHz.
- **Microcontroladores criados para aplicações específicas**.

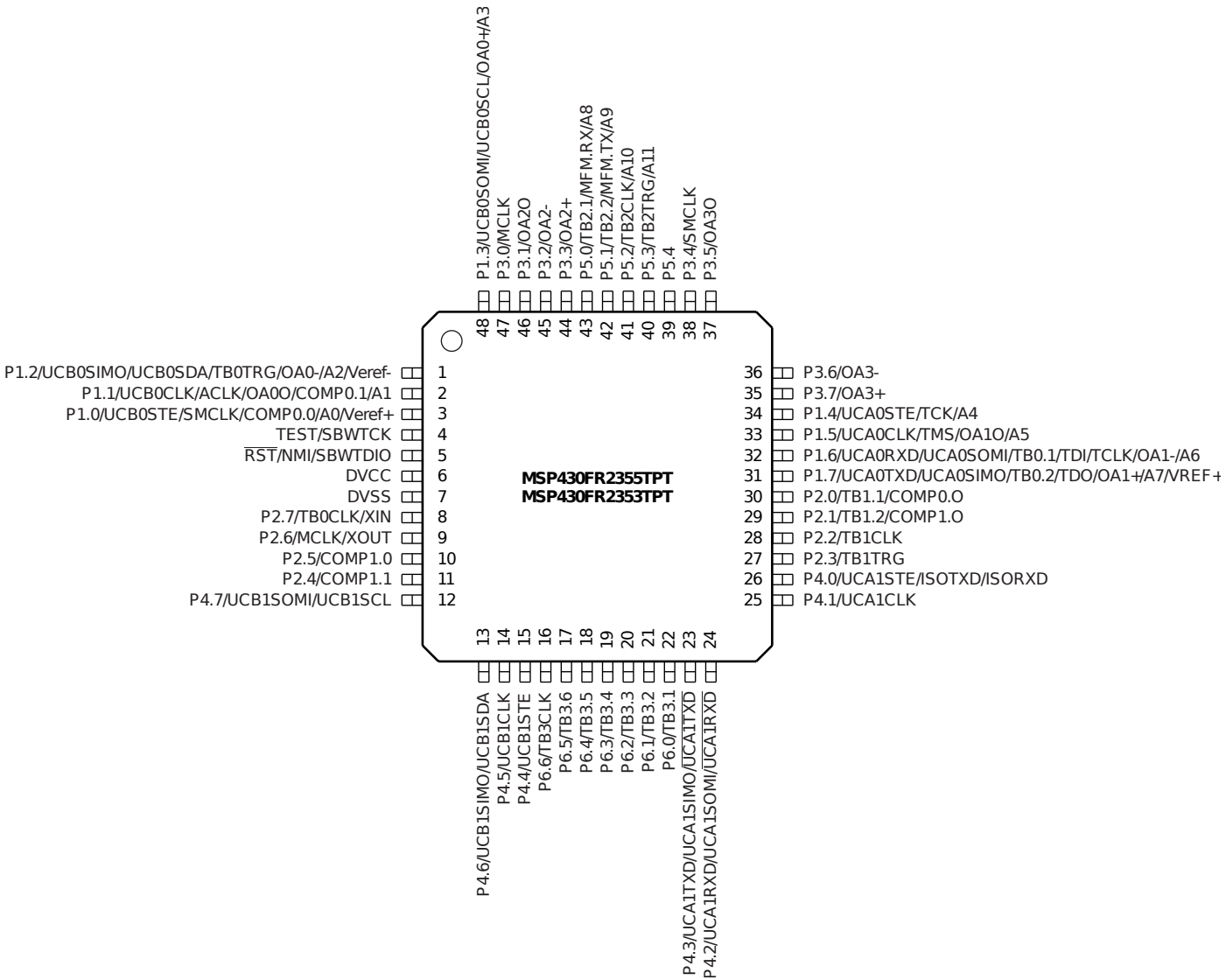
`http://www.ti.com/microcontrollers/msp430-ultra-low-power-mcus/overview.html`

Texas Instruments MSP430





Texas Instruments MSP430



Recursos didáticos

- 10 kits Lauchpad MSP430FR2355
- 2 kits Lauchpad MSP430FR5739
- 5 kits Lauchpad MSP430G2553
- 5 kits Lauchpad MSP430G2231/MSP430G2452/MSP430G2211
- Kits suportam depuração em hardware
- Microcontroladores mais simples possuem modelos simuláveis no Proteus

Energīa

Experimenter Board with MSP430FR5739

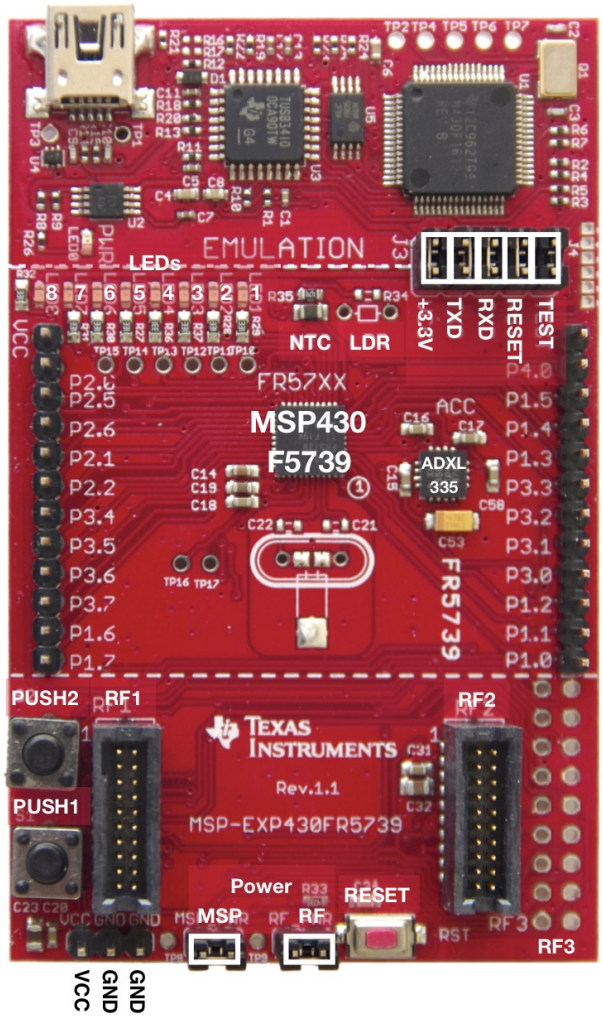
Revision 1.1

FRAM	16	KB
SRAM	1	KB

Serial	hardware	
ADC	10	bits
Use pins numbers only!		
PWM 2 or 23		
PWM 5 or 9		
PWM 6 or 10		
PWM 7 or 11		
PWM 8 or 12		

+3.3V					1
	TXD			P2_0	2
				P2_5	3
				P2_6	4
	RXD			P2_1	5
		SCK (B0)		P2_2	6
LED5				P3_4	7
LED6				P3_5	8
LED7				P3_6	9
LED8				P3_7	10
		MOSI (B0)	SDA	P1_6	11
		MISO (B0)	SCL	P1_7	12

LED1	RED_LED			PJ_0	25
LED2				PJ_1	26
LED3				PJ_2	27
LED4	GREEN_LED			PJ_3	28



Hardware
Pin number
Other pin number

I²C
Serial UART
SPI

analogRead()
digitalRead() and digitalWrite()
digitalRead(), digitalWrite() and analogWrite()

24				GROUND
23	P4_0			PUSH1
22	P1_5	A0		
21	P1_4	A1		NTC
20	P1_3	A2	CS (B0)	
19	P3_3	A3		LDR
18	P3_2	A4		ACC_Z
17	P3_1	A5		ACC_Y
16	P3_0	A6		ACC_X
15	P1_2	A7		
14	P1_1	A8		
13	P1_0	A9		

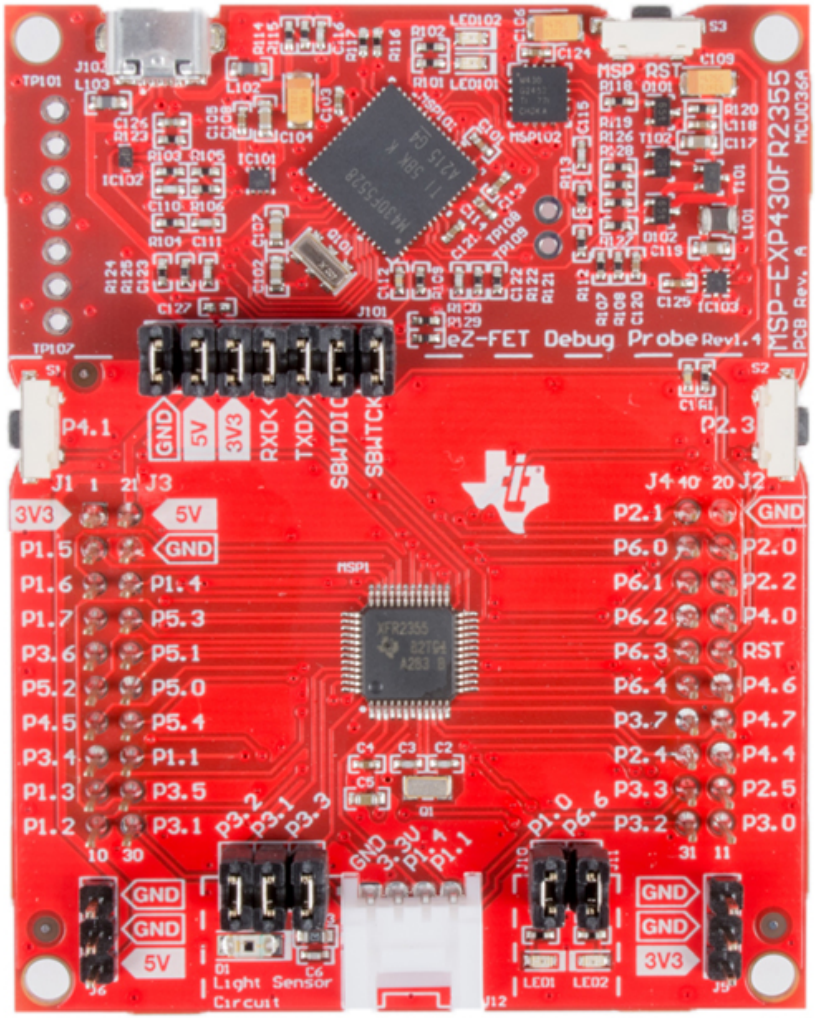
29	P4_1			PUSH2
30	P2_7		NTC_ENABLE	ACC_ENABLE

Headers are not compatible with LaunchPad!

embeddedcomputing.weebly.com

version 2.1 2015-09-13

MSP430FR2355



Energīa

LaunchPad with MSP430G2553

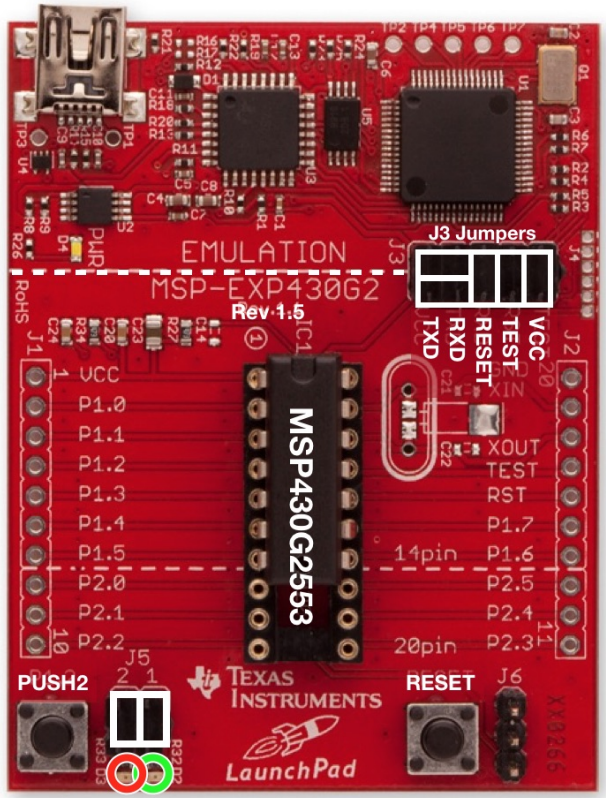
Revision 1.5

Flash	16	KB
RAM	512	B

Serial	Hardware
ADC	10 bits
Use pins numbers only!	
Default I²C = (1)	
Software I²C (1) master only	
PWM 4 or 14 or 19	
PWM 9 or 10	
PWM 12 or 13	

+3.3V				1
RED_LED		A0	P1_0	2
	RXD	A1	P1_1	3
	TXD	A2	P1_2	4
PUSH2		A3	P1_3	5
		A4	P1_4	6
	SCK (B0)	A5	P1_5	7
	CS (B0)		P2_0	8
	SCL (1)		P2_1	9
	SDA (1)		P2_2	10

temperature		A10



20				GROUND
19	P2_6			XIN
18	P2_7			XOUT
17				TEST
16				RESET
15	P1_7	A7	SDA (0)	MOSI (B0)
14	P1_6	A6	SCL (0)	MISO (B0) GREEN_LED
13	P2_5			
12	P2_4			
11	P2_3			

GND
GND
+3.3V

Hardware
Pin number

I²C
Serial UART
SPI

analogRead()
digitalRead() and digitalWrite()
digitalRead(), digitalWrite() and analogWrite()

embeddedcomputing.weebly.com

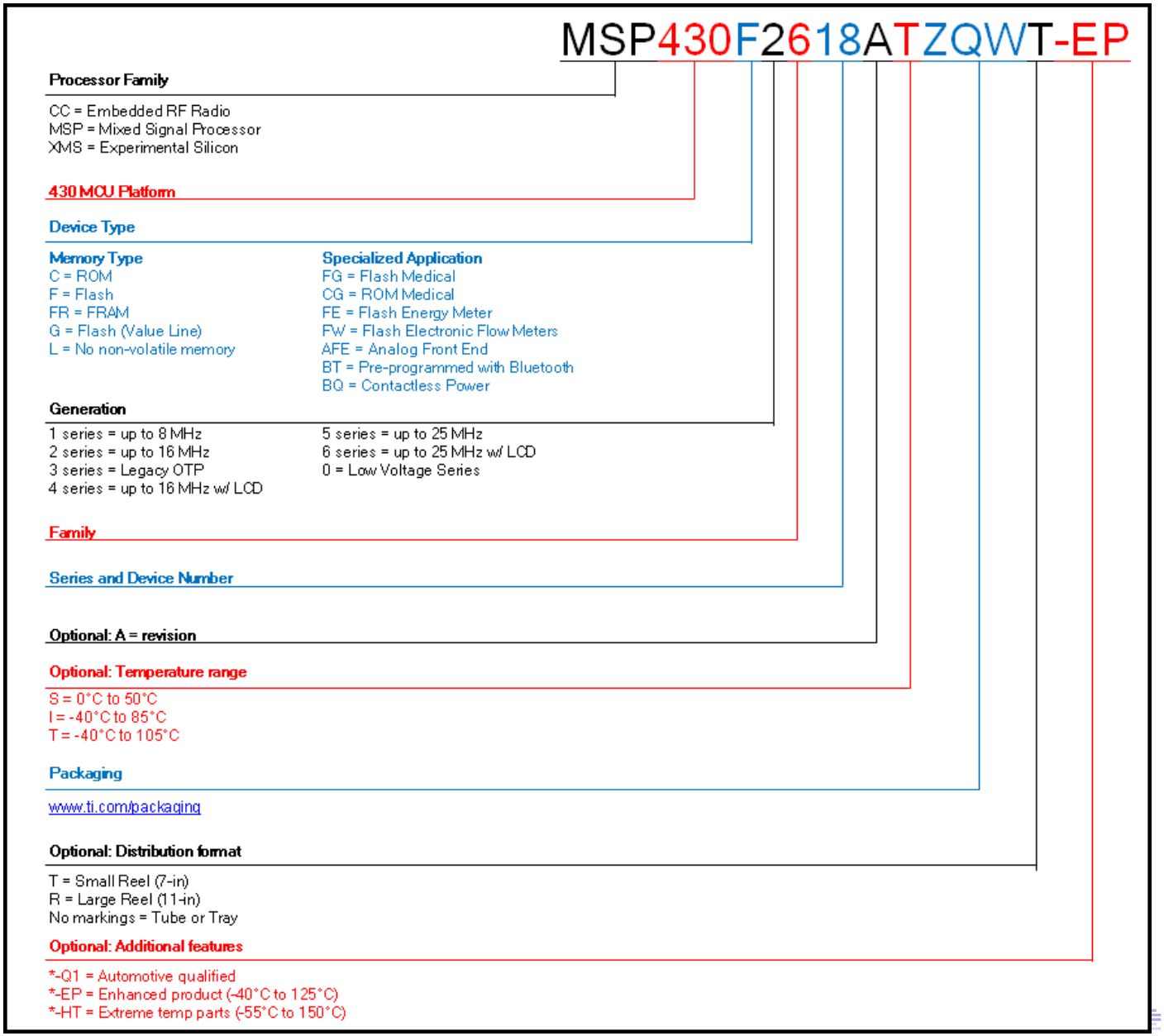
version 2.1 2015-09-13

Tópico

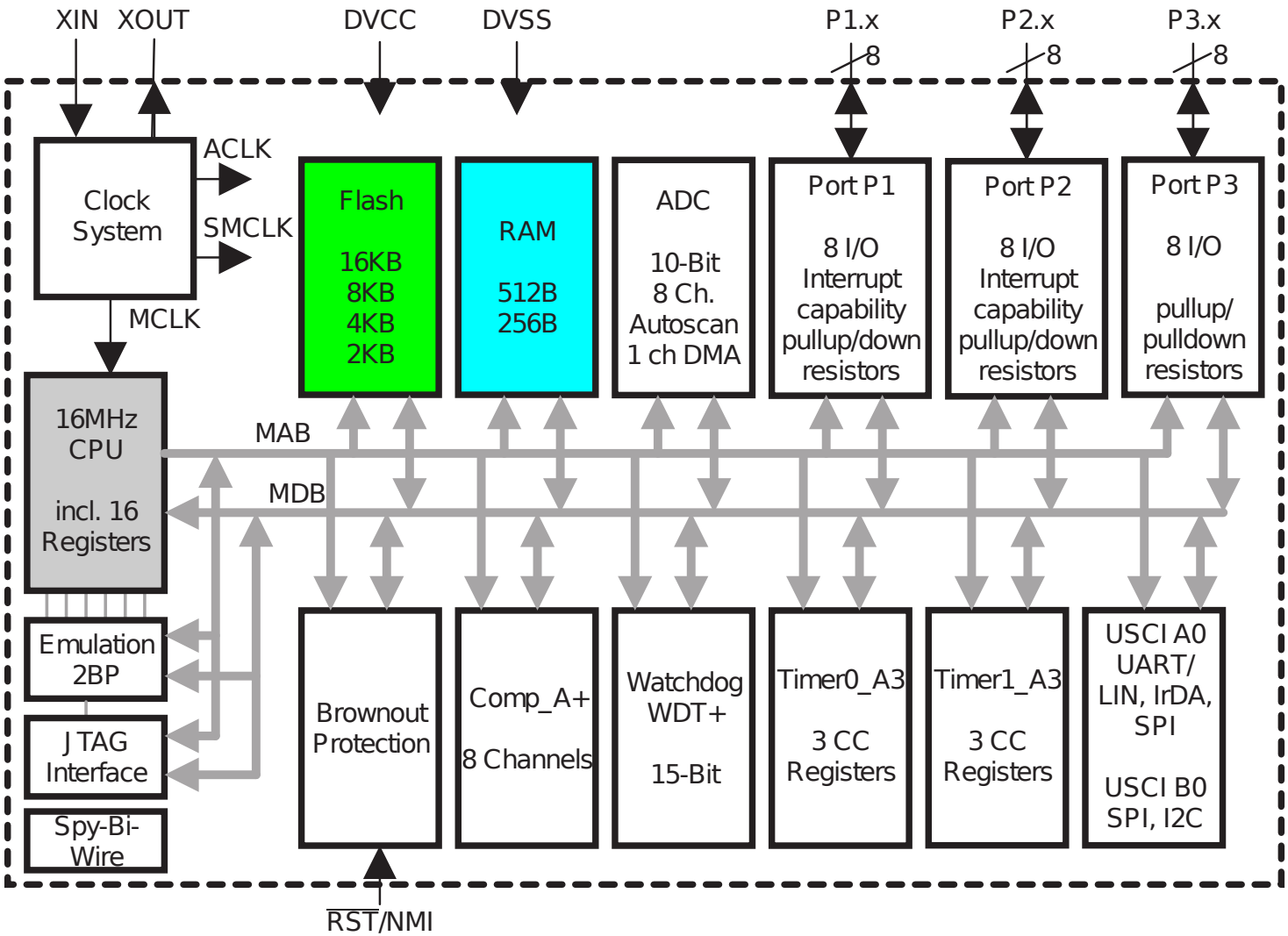
Família MSP430

- A famílilia MSP430 foi desenvolvida pela Texas Instruments e introduzida em 1992.
- O objetivo sempre foi o mercado de aplicações embarcadas portáveis alimentadas por baterias.
- Há centenas de configurações diferentes. Novas famílias são compatíveis com os CPUs mais antigas mantendo a filosofia original.
- Todos os microcontroladores MSP430 utiliza uma CPU RISC de 16 bits de arquitetura Von-Neumann.

Exemplos de part numbers



Estrutura interna



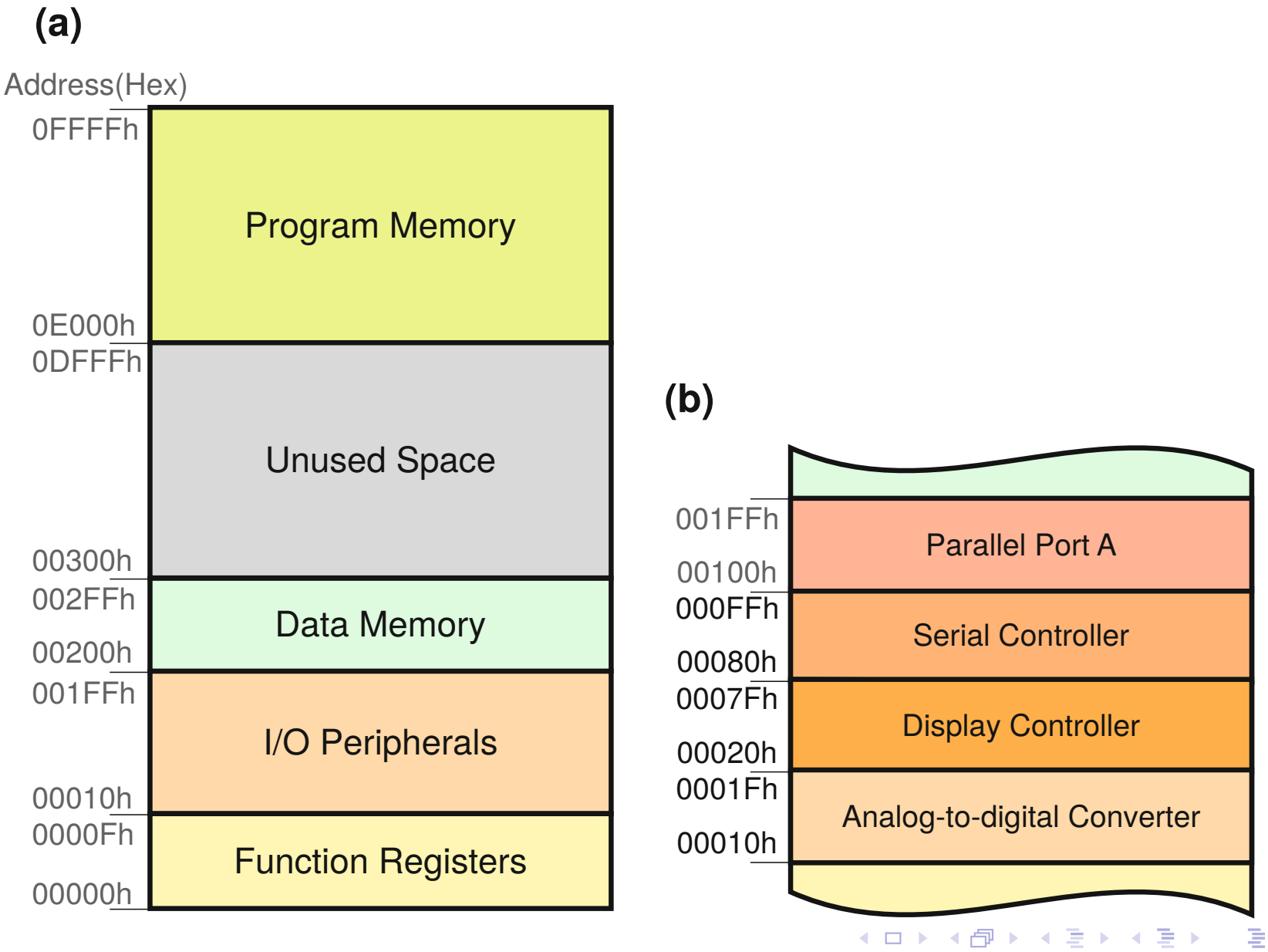
Características gerais

- Arquitetura 16 bits padronizada. Todos utilizam mesmo núcleo e conjunto de instruções. Extensão de 20-bits CPUX utilizam mesma arquitetura.
- Modos de operações de baixo consumo: **ultra low-power operation modes**.
- Modelo de programação flexível. Diversos modos de endereçamento, vetor de interrupções com prioridades e automaticamente aninhado.
- Periféricos extensíveis mapeados em memória: conversores AD, temporizadores, PWM, USART, drivers LCD, temporizador Watchdog, oscilador programável, entre outros.
- **Depurador JTAG para agilizar desenvolvimento/gravação.**

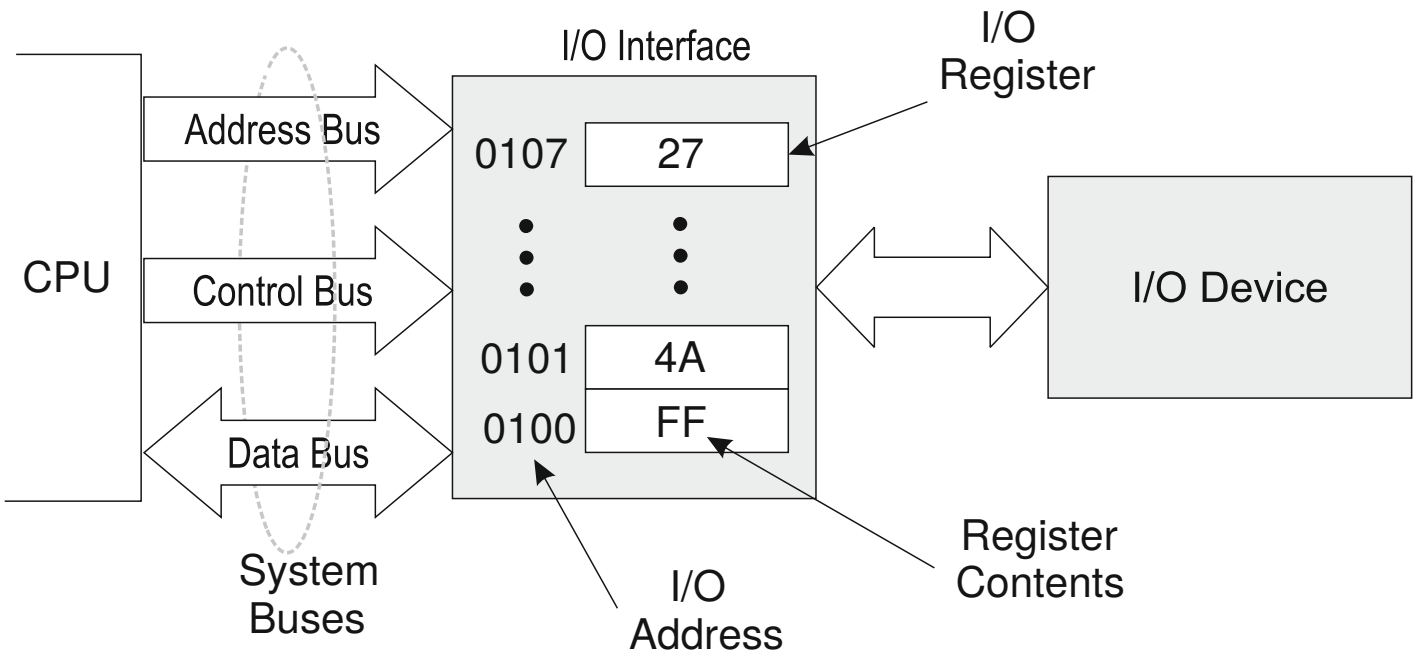
Registradores

Registrador	**Nome/Função**
R0	Contador de programa (PC)
R1	Apontador de pilha (SP)
R2	Status/Gerador de constantes 1
R3	Gerador de constantes 2
R4	Propósito geral (GPR)
R5	Propósito geral (GPR)
R6	Propósito geral (GPR)
R7	Propósito geral (GPR)
R8	Propósito geral (GPR)
R9	Propósito geral (GPR)
R10	Propósito geral (GPR)
R11	Propósito geral (GPR)
R12	Propósito geral (GPR)
R13	Propósito geral (GPR)
R14	Propósito geral (GPR)
R15	Propósito geral (GPR)

Estrutura típica da memória



Estrutura típica da memória



Instruções principais

Type	Instruction	Description	V	N	Z	C
Data	`mov src,dest`	Loads destination with source	-	-	-	-[1]
Transfer	`push src`	Pushes source onto top of stack	-	-	-	-
	`swpb dest`	Swap bytes in destination word	-	-	-	-
	`add src,dest`	Adds source to destination	*	*	*	*
	`addc src,dest`	Adds source and carry to destination	*	*	*	*
	`sub src,dest`	Adds $\overline{\text{source}} + 1$ to destination	*	*	*	*
Arithmetic		(subtract source from destination)				
	`subc src,dest`	Adds $\overline{\text{source}} + CF$ to destination	*	*	*	*
		(subtract with borrow)				
	`dadd src,dest`	Adds source and carry to destination	*	*	*	*
		in Decimal (BCD) form[2]				
	`cmp src,dest`	$dest - source$, but only affects flags[3]	*	*	*	*
	`sxt dest`	Sign extend LSB to 16-bit word	0	*	*	*

[1]: For Flags: − means there is no effect; ∗there is an effect; “0”, f ag is reset.
[2]: Result is irrelevant if operands are not in format BCD
[3]: Used to compare numbers, usually followed by a conditional jump
[4]: Used to test if bits are set, usually followed by a conditional jump

Instruções principais

Type	Instruction	Description	V	N	Z	C
	`and src,dest`	"AND"s source to destination bitwise	0	∗	∗	∗
	`xor src,dest`	"XOR"s source to destination bitwise	∗	∗	∗	∗
	`bit src,dest`	Like `and`, but only affects flags[4]	0	∗	∗	∗
Logic	`bic src,dest`	Resets bits in destination	-	-	-	-
and bit	`bis src,dest`	Sets bits in destination.	-	-	-	-
management	`rra dest`	Roll bits to right arithmetically, i.e., $B_n \rightarrow B(n-1) \ldots B_1 \rightarrow B_0 \rightarrow C$	0	∗	∗	∗
	`rrc dest`	Roll destinations to right through Carry, $C \rightarrow B_n \rightarrow B(n-1) \ldots B_1 \rightarrow B_0 \rightarrow C$	∗	∗	∗	∗
	`jz/jeq label`	Jump if zero/equal (Z = 1)	-	-	-	-
	`jnz/jne label`	Jump not zero/equal (Z = 0)	-	-	-	-
	`jc/jhe label`	Jump if carry (C = 1) – if higher or equal–- (≥, for unsigned numbers)	-	-	-	–
	`jnc/jlo label`	Jump if not carry (C = 0)– if lower,– - (<, for unsigned numbers)	-	-	-	–
Program	`jn label`	Jump if negative (N = 1)	-	-	-	-
Flow	`jge label`	Jump if $V = N$ (≥, for signed numbers)	-	-	-	-
	`jl label`	Jump if V=N (if <, signed numbers)	-	-	-	-
	`jmp label`	Jump to label unconditionally	-	-	-	-
	`call dest`	Call subroutine at destination	-	-	-	-
	`reti`	Return from interrupt	-	-	-	-

[1]: For Flags: – means there is no effect; ∗there is an effect; "0", f ag is reset.
[2]: Result is irrelevant if operands are not in format BCD

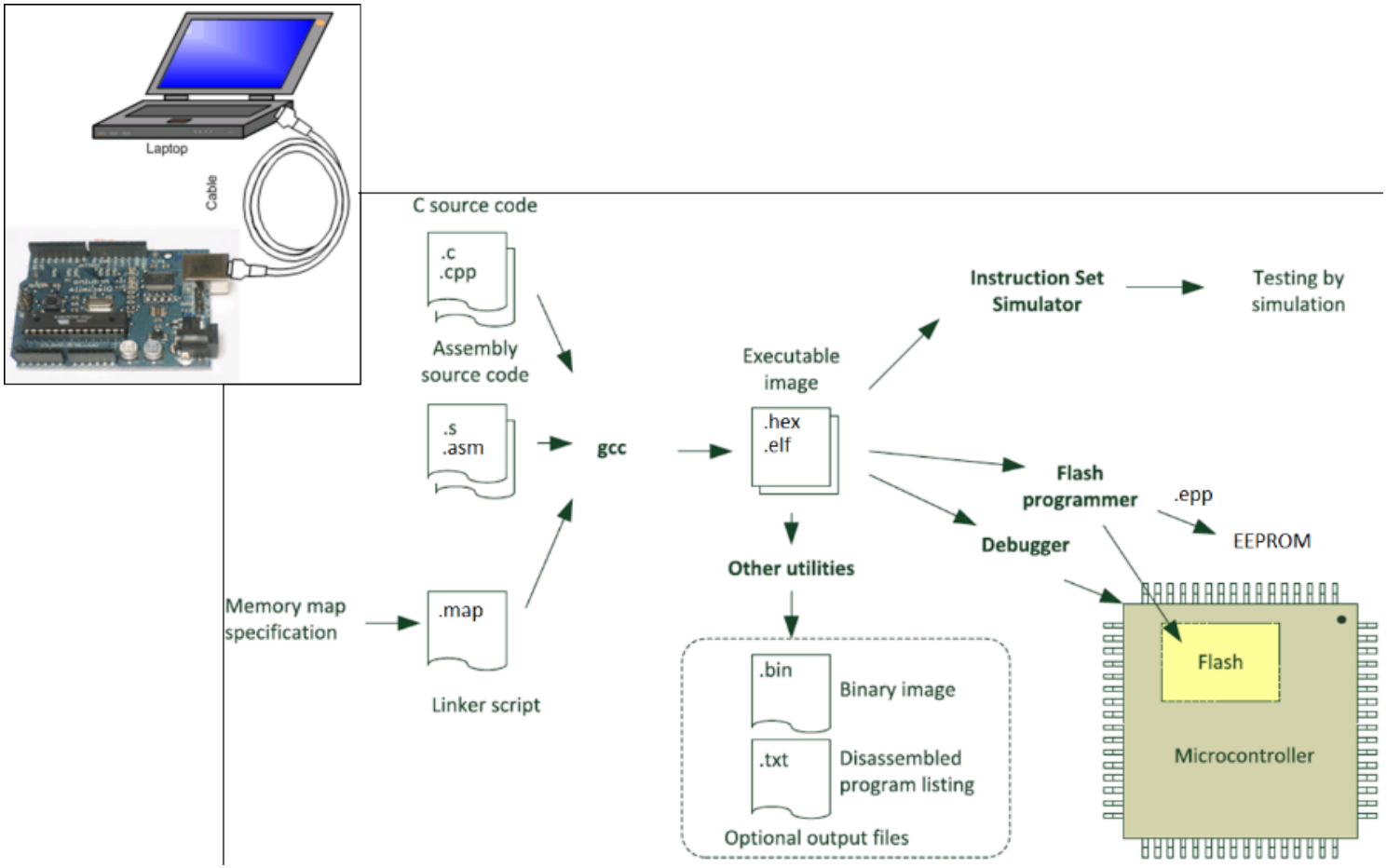
Pseudo-instruções

Table 4.5 Emulated instructions in the MSP430

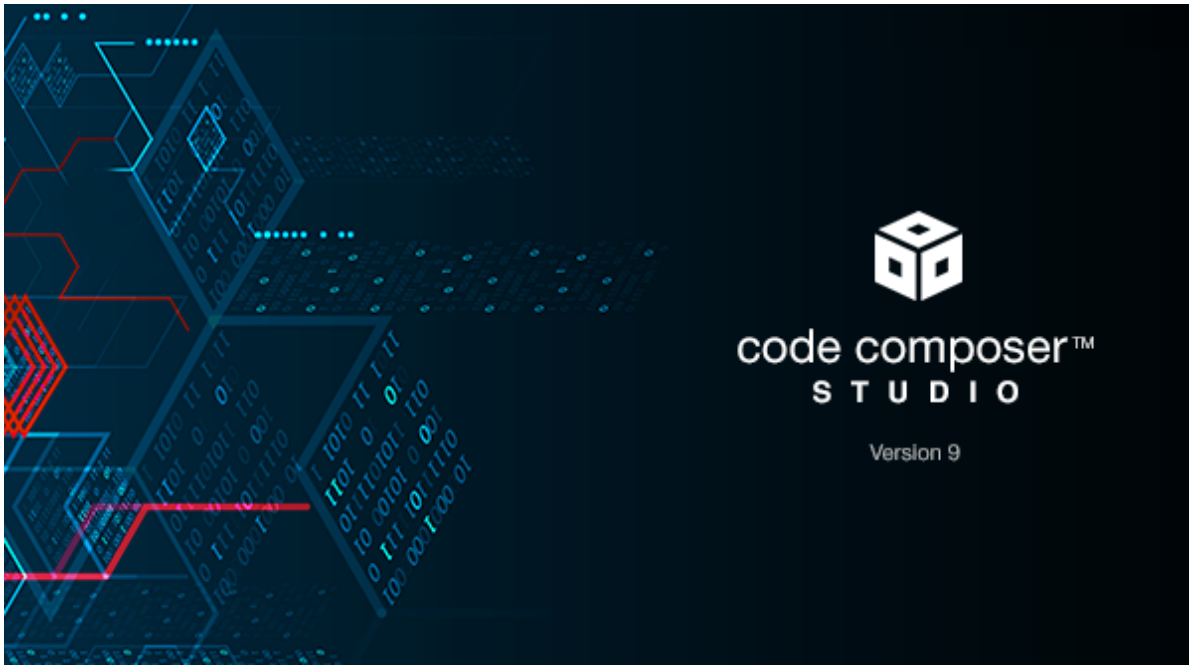
Type	Instruction	Description	Core Inst.
Data Transfer	`pop dest`	Loads destination from TOS	`mov @SP+,dest`
	`adc dest`	Add carry to destination	`addc #0,dest`
	`dadc src,dest`	Decimal add Carry to destination	`addc #0,dest`
	`dec dest`	Decrement destination	`sub #1,dest`
Arithmetic	`decd dest`	Decrement destination twice	`sub #2,dest`
	`inc dest`	Increment destination	`add #1,dest`
	`incd dest`	Increment destination twice	`add #2,dest`
	`sbc dest`	Subtract Carry from destination	`subc #0,dest`
	`tst dest`	Test destination	`cmp #0,dest`
	`inv dest`	Invert bits in destination	`xor #0FFFFh,dest`
	`rla dest`	Roll (shift) bits to left	`add dest,dest`
Logic	`rlc dest`	Roll bits left through carry	`addc dest,dest`
and bit	`clr dest`	Clear destination	`mov #0,dest`
Management	`clrc`	Clear carry f ag	`bic #1,SR`
	`clrz`	Clear zero f ag	`bic #2,SR`
	`clrn`	Clear negative f ag	`bic #4,SR`
	`setc`	Clear carry f ag	`bis #1,SR`
	`setz`	Clear zero f ag	`bis #2,SR`
	`setn`	Clear negative f ag	`bis #4,SR`
	`br dest`	Branch to destination	`mov dest,PC`
Program	`dint`	Disable interrupts	`bic #8,SR`
Flow	`eint`	Enable interrupts	`bis #8,SR`
	`nop`	no operation	`mov R3,R3`
	`ret`	Return from subroutine	`mov @SP+,PC`

Tópico

Ferramentas de projeto

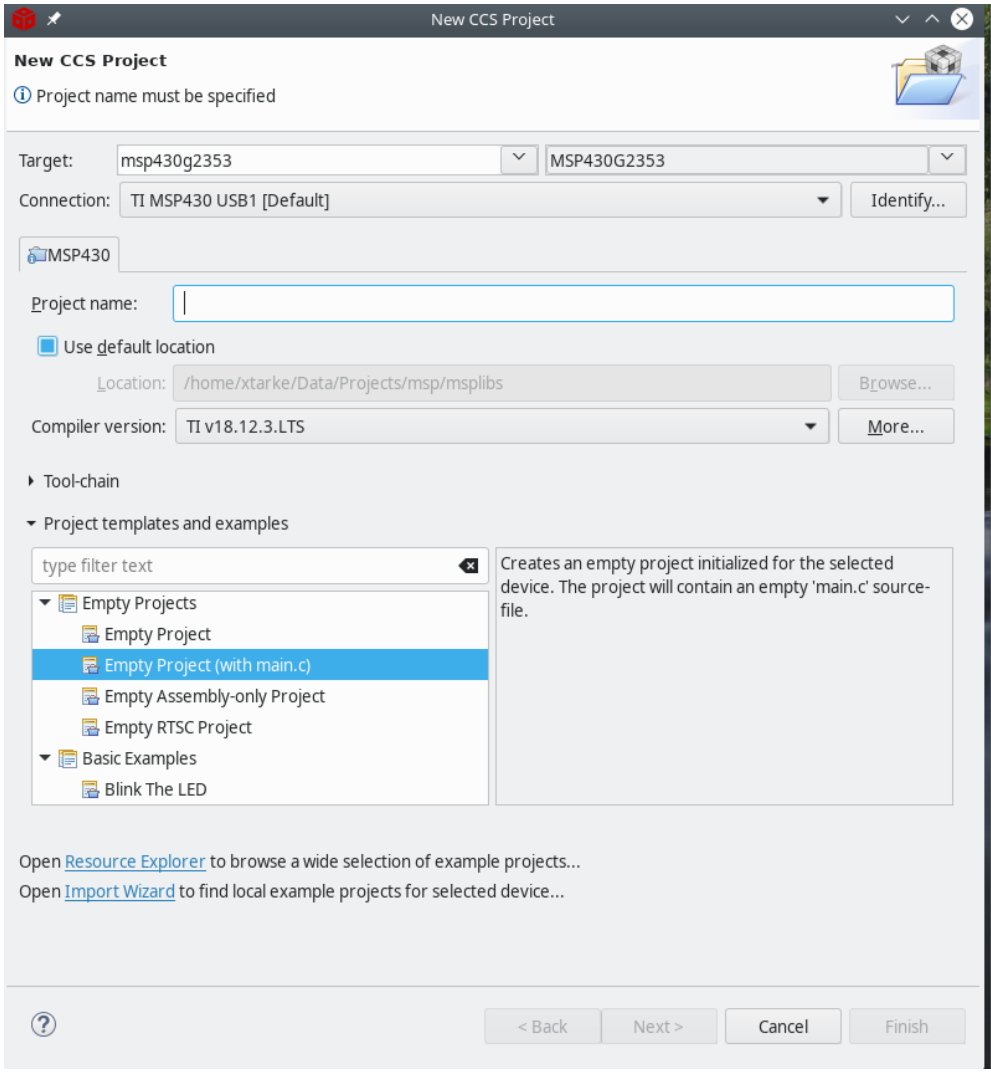


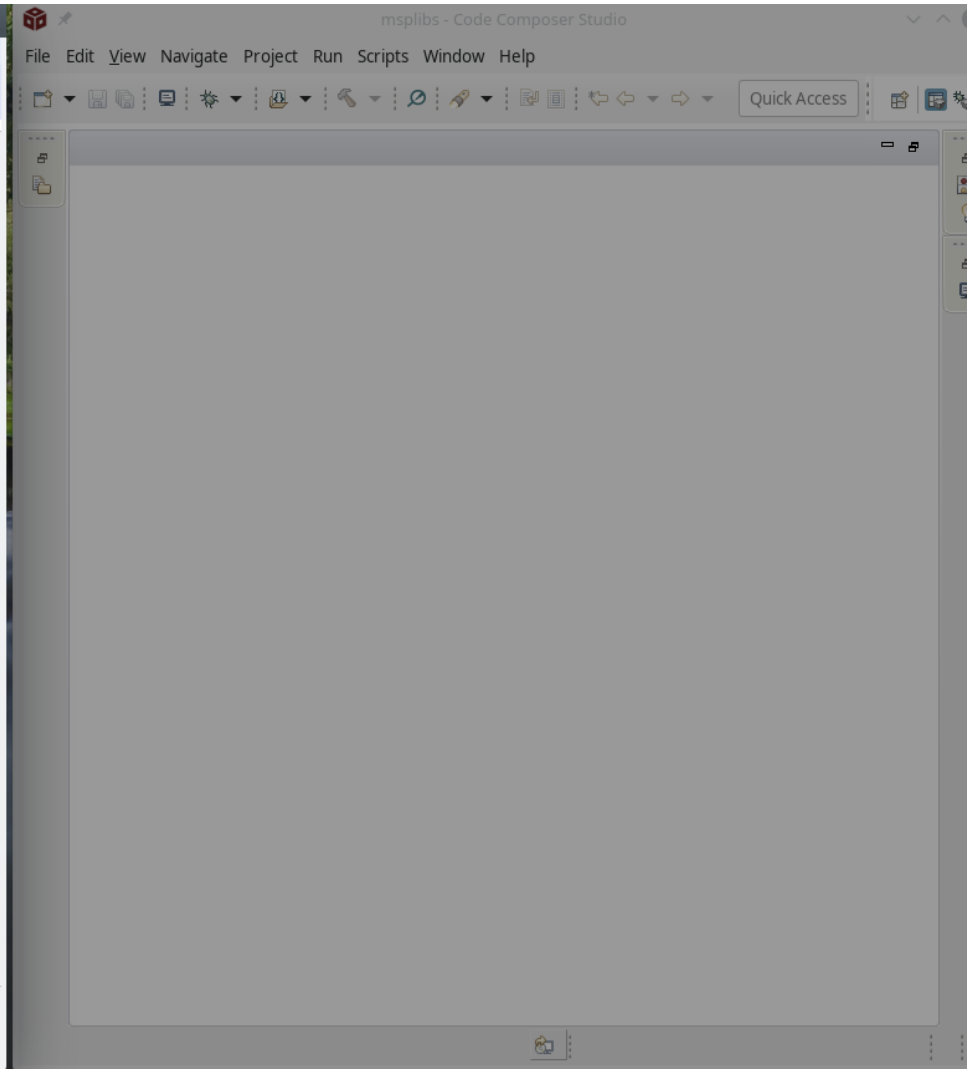
Ferramentas de projeto



`http://software-dl.ti.com/ccs/esd/documents/ccs_downloads.html`

Ferramentas de projeto






DANGER
MAN AT WORK