PINAGEM

PW or N PACKAGE
(TOP VIEW)

V_{CC} — 1	14 — V_{SS}
P1.0/TACLK/ACLK/A0+ — 2	13 — XIN/P2.6/TA1
P1.1/TA0/A0−/A4+ — 3	12 — XOUT/P2.7
P1.2/TA1/A1+/A4− — 4	11 — TEST/SBWTCK
P1.3/VREF/A1− — 5	10 — $\overline{RST}$/NMI/SBW TDIO
P1.4/SMCLK/A2+/TCK — 6	9 — P1.7/A3−/SDI/SDA/TDO/TDI
P1.5/TA0/A2−/SCLK/TMS — 7	8 — P1.6/TA1/A3+/SDO/SCL/TDI/TCLK

Figure 2.1: Pin-out of the MSP430F2003 and F2013, taken from the data sheet.

PINAGEM

PW or N PACKAGE
(TOP VIEW)

V_{CC}	1	14	V_{SS}
P1.0/TACLK/ACLK/A0+	2	13	XIN/P2.6/TA1
P1.1/TA0/A0−/A4+	3	12	XOUT/P2.7
P1.2/TA1/A1+/A4−	4	11	TEST/SBWTCK
P1.3/VREF/A1−	5	10	RST /NMI/SBW TDIO
P1.4/SMCLK/A2+/TCK	6	9	P1.7/A3−/SDI/SDA/TDO/TDI
P1.5/TA0/A2−/SCLK/TMS	7	8	P1.6/TA1/A3+/SDO/SCL/TDI/TCLK

TACLK pode ser usado como entrada do clock do Timer_A. TA0 e TA1 podem ser entradas ou saídas do Timer_A.

PINAGEM

PINAGEM

PW or N PACKAGE
(TOP VIEW)

V_{CC}	1	14	V_{SS}
P1.0/TACLK/ACLK/A0+	2	13	XIN/P2.6/TA1
P1.1/TA0/A0−/A4+	3	12	XOUT/P2.7
P1.2/TA1/A1+/A4−	4	11	TEST/SBWTCK
P1.3/VREF/A1−	5	10	$\overline{RST}$/NMI/SBWTDIO
P1.4/SMCLK/A2+/TCK	6	9	P1.7/A3−/SDI/SDA/TDO/TDI
P1.5/TA0/A2−/SCLK/TMS	7	8	P1.6/TA1/A3+/SDO/SCL/TDI/TCLK

A0-, A0+, ..., A3-, A3+ são
entradas do conversor A/D

PINAGEM

PINAGEM

PW or N PACKAGE
(TOP VIEW)

V_{CC}	1 ○ — 14	V_{SS}
P1.0/TACLK/ACLK/A0+	2 — 13	XIN/P2.6/TA1
P1.1/TA0/A0−/A4+	3 — 12	XOUT/P2.7
P1.2/TA1/A1+/A4−	4 — 11	TEST/SBWTCK
P1.3/VREF/A1−	5 — 10	$\overline{RST}$/NMI/SBW TDIO
P1.4/SMCLK/A2+/TCK	6 — 9	P1.7/A3−/SDI/SDA/TDO/TDI
P1.5/TA0/A2−/SCLK/TMS	7 — 8	P1.6/TA1/A3+/SDO/SCL/TDI/TCLK

VREF é a tensão de referência
para o conversor A/D

PINAGEM

PW or N PACKAGE
(TOP VIEW)

```
              V_CC  [] 1 O   14 []  V_SS
  P1.0/TACLK/ACLK/A0+ [] 2   13 []  XIN/P2.6/TA1
     P1.1/TA0/A0−/A4+ [] 3   12 []  XOUT/P2.7
  P1.2/TA1/A1+/A4−    [] 4   11 []  TEST/SBWTCK
     P1.3/VREF/A1−    [] 5   10 []  RST/NMI/SBW TDIO
  P1.4/SMCLK/A2+/TCK  [] 6    9 []  P1.7/A3−/SDI/SDA/TDO/TDI
  P1.5/TA0/A2−/SCLK/TMS [] 7  8 []  P1.6/TA1/A3+/SDO/SCL/TDI/TCLK
```

SMCLK e ACLK são saídas para o sinal de clock
do MSP430, para serem aproveitados por
outros componentes e para testar o MSP430

PINAGEM

PINAGEM

PW or N PACKAGE
(TOP VIEW)

V_{CC}	1	14	V_{SS}
P1.0/TACLK/ACLK/A0+	2	13	XIN/P2.6/TA1
P1.1/TA0/A0−/A4+	3	12	XOUT/P2.7
P1.2/TA1/A1+/A4−	4	11	TEST/SBWTCK
P1.3/VREF/A1−	5	10	$\overline{RST}$/NMI/SBW TDIO
P1.4/SMCLK/A2+/TCK	6	9	P1.7/A3−/SDI/SDA/TDO/TDI
P1.5/TA0/A2−/SCLK/TMS	7	8	P1.6/TA1/A3+/SDO/SCL/TDI/TCLK

SCLK, SDO e SCL são usados pela interface
serial universal (comunicação SPI e I2C)

PINAGEM

PW or N PACKAGE
(TOP VIEW)

V_{CC}	1 ○	14	V_{SS}
P1.0/TACLK/ACLK/A0+	2	13	XIN/P2.6/TA1
P1.1/TA0/A0−/A4+	3	12	XOUT/P2.7
P1.2/TA1/A1+/A4−	4	11	TEST/SBWTCK
P1.3/VREF/A1−	5	10	$\overline{RST}$/NMI/SBW TDIO
P1.4/SMCLK/A2+/TCK	6	9	P1.7/A3−/SDI/SDA/TDO/TDI
P1.5/TA0/A2−/SCLK/TMS	7	8	P1.6/TA1/A3+/SDO/SCL/TDI/TCLK

XIN e XOUT são as conexões para o cristal
externo, para gerar um clock estável

Little Wing - YouTube | Microcontroladores/Aula | Microcontroladores/Que | Microcontroladores/uCo | João

GitHub, Inc. [US] | https://github.com/DiogoCaetanoGarcia/Microcontroladores/blob/master/Aulas/uControl%2002%20Overview%20MSP430.pdf

Apps | W3Schools Online W | Curso de HTML5 – 01 | O que é Ciência da C | HTML, CSS e Javascri | GitHub – jhu-ep-cour | Aulas de Introdução | Introdução — Aulas

PINAGEM

PW or N PACKAGE
(TOP VIEW)

V_{CC}	1 ○ 14	V_{SS}
P1.0/TACLK/ACLK/A0+	2 13	XIN/P2.6/TA1
P1.1/TA0/A0−/A4+	3 12	XOUT/P2.7
P1.2/TA1/A1+/A4−	4 11	TEST/SBWTCK
P1.3/VREF/A1−	5 10	RST/NMI/SBW TDIO
P1.4/SMCLK/A2+/TCK	6 9	P1.7/A3−/SDI/SDA/TDO/TDI
P1.5/TA0/A2−/SCLK/TMS	7 8	P1.6/TA1/A3+/SDO/SCL/TDI/TCLK

_RST reseta o MSP430 quando
levado ao nível baixo

PINAGEM

PINAGEM

PW or N PACKAGE
(TOP VIEW)

V_{CC}	1	14	V_{SS}
P1.0/TACLK/ACLK/A0+	2	13	XIN/P2.6/TA1
P1.1/TA0/A0−/A4+	3	12	XOUT/P2.7
P1.2/TA1/A1+/A4−	4	11	TEST/SBWTCK
P1.3/VREF/A1−	5	10	$\overline{RST}$/NMI/SBWTDIO
P1.4/SMCLK/A2+/TCK	6	9	P1.7/A3−/SDI/SDA/TDO/TDI
P1.5/TA0/A2−/SCLK/TMS	7	8	P1.6/TA1/A3+/SDO/SCL/TDI/TCLK

NMI é a entrada para interrupção não-mascarável, que permite um sinal externo interromper o programa principal

Not Falling Asleep // × | Microcontroladores/Aula × | Microcontroladores/Que × | Microcontroladores/uCo ×

GitHub, Inc. [US] | https://github.com/DiogoCaetanoGarcia/Microcontroladores/blob/master/Aulas/uControl%2002%20Overview%20MSP430.pdf

Apps W3Schools Online W Curso de HTML5 – 01 O que é Ciência da C HTML, CSS e Javascri GitHub - jhu-ep-cour Aulas de Introdução Introdução — Aulas

PINAGEM

PW or N PACKAGE
(TOP VIEW)

V_{CC}	1	14	V_{SS}
P1.0/TACLK/ACLK/A0+	2	13	XIN/P2.6/TA1
P1.1/TA0/A0−/A4+	3	12	XOUT/P2.7
P1.2/TA1/A1+/A4−	4	11	TEST/SBWTCK
P1.3/VREF/A1−	5	10	RST /NMI/SBW TDIO
P1.4/SMCLK/A2−/TCK	6	9	P1.7/A3−/SDI/SDA/TDO/TDI
P1.5/TA0/A2−/SCLK/TMS	7	8	P1.6/TA1/A3+/SDO/SCL/TDI/TCLK

TCK, TMS, TCLK, TDI e TDO formam
a interface JTAG, usada para
programar e debugar o MSP430

PINAGEM

programar e debugar o MSP430

PINAGEM

PW or N PACKAGE
(TOP VIEW)

V_{CC}	1	14	V_{SS}
P1.0/TACLK/ACLK/A0+	2	13	XIN/P2.6/TA1
P1.1/TA0/A0−/A4+	3	12	XOUT/P2.7
P1.2/TA1/A1+/A4−	4	11	TEST/SBWTCK
P1.3/VREF/A1−	5	10	RST/NMI/SBWTDIO
P1.4/SMCLK/A2+/TCK	6	9	P1.7/A3−/SDI/SDA/TDO/TDI
P1.5/TA0/A2−/SCLK/TMS	7	8	P1.6/TA1/A3+/SDO/SCL/TDI/TCLK

SBWTCK e SBWTDIO formam a interface
Spy-Bi-Wire, uma alternativa ao JTAG

PINAGEM

PW or N PACKAGE
(TOP VIEW)

V_{CC}	1	14	V_{SS}
P1.0/TACLK/ACLK/A0+	2	13	XIN/P2.6/TA1
P1.1/TA0/A0−/A4+	3	12	XOUT/P2.7
P1.2/TA1/A1+/A4−	4	11	TEST/SBWTCK
P1.3/VREF/A1−	5	10	RST /NMI/SBW TDIO
P1.4/SMCLK/A2+/TCK	6	9	P1.7/A3−/SDI/SDA/TDO/TDI
P1.5/TA0/A2−/SCLK/TMS	7	8	P1.6/TA1/A3+/SDO/SCL/TDI/TCLK

Únicos pinos que não podem alterados

VISÃO INTERNA

Operadores binários em linguagem C

A linguagem C apresenta uma grande variedade de operadores. A seguir são apresentados os operadores para operações binarias, bit a bit:

Operador	Função
&	E (AND)
		OU (OR)
^	OU-EXCLUSIVO (XOR)
~	NOT ou Complemento de 1
<<	Deslocamento à esquerda
>>	Deslocamento à direita

Os operadores **&**, |, ^ e ~ funcionam como apresentado anteriormente e têm as saidas conforme exibidos em suas tabelas verdades. Os operadores de deslocamento fazem o shift de bits para esquerda ou direita.

Exemplo:

Vamos assumir que **x** incialmente tenha o valor 0b00000001, valor 1 em decimal.

```
1  x = (x<<2);
```

Após a operação acima, será atribuido a *x* o valor 0b000000100, valor 4 em decimal.

Se em seguida realizarmos uma operação de deslocamento à direita:

Nibble = 4 bits

1	() [] -> . ::	Grouping, scope, array/member access
2	! ~ - + * & sizeof *type cast* ++x --x	(most) unary operations, sizeof and type casts
3	* / %	Multiplication, division, modulo
4	+ -	Addition and subtraction
5	<< >>	Bitwise shift left and right
6	< <= > >=	Comparisons: less-than, ...
7	== !=	Comparisons: equal and not equal
8	&	Bitwise AND
9	^	Bitwise exclusive OR
10	|	Bitwise inclusive (normal) OR
11	&&	Logical AND
12	||	Logical OR
13	?:	Conditional expression (ternary operator)
14	= += -= *= /= %= &= |= ^= <<= >>=	Assignment operators
15	,	Comma operator

PINAGEM

PW or N PACKAGE
(TOP VIEW)

Alimentação — Terra

P1.0/TACLK/ACLK/A0+	2	13	XIN/P2.6/TA1
P1.1/TA0/A0−/A4+	3	12	XOUT/P2.7
P1.2/TA1/A1+/A4−	4	11	TEST/SBWTCK
P1.3/VREF/A1−	5	10	$\overline{RST}$/NMI/SBW TDIO
P1.4/SMCLK/A2+/TCK	6	9	P1.7/A3−/SDI/SDA/TDO/TDI
P1.5/TA0/A2−/SCLK/TMS	7	8	P1.6/TA1/A3+/SDO/SCL/TDI/TCLK

(Pin 1 / 14 at top)

PINAGEM

PW or N PACKAGE
(TOP VIEW)

V_{CC}	1 ○	14	V_{SS}
P1.0/TACLK/ACLK/A0+	2	13	XIN/P2.6/TA1
P1.1/TA0/A0−/A4+	3	12	XOUT/P2.7
P1.2/TA1/A1+/A4−	4	11	TEST/SBWTCK
P1.3/VREF/A1−	5	10	$\overline{RST}$/NMI/SBW TDIO
P1.4/SMCLK/A2+/TCK	6	9	P1.7/A3−/SDI/SDA/TDO/TDI
P1.5/TA0/A2−/SCLK/TMS	7	8	P1.6/TA1/A3+/SDO/SCL/TDI/TCLK

P1.0-P1.7 e P2.6-P2.7 são entradas e saídas digitais

PINAGEM

MSP430 Instructions

MPS430 Instruction Formats

- **Format I: Instructions with two operands:**

15 14 13 12	11 10 9 8	7	6	5 4	3 2 1 0
Op-code	S-reg	Ad	b/w	As	D-reg

- **Format II: Instruction with one operand:**

15 14 13 12 11 10 9 8 7	6	5 4	3 2 1 0
Op-code	b/w	Ad	D/S-reg

- **Format III: Jump instructions:**

15 14 13	12 11 10	9 8 7 6 5 4 3 2 1 0
Op-code	Condition	10-bit, 2's complement PC offset

Lecture 03 - MSP430 ISA   14

Format I: Double Operand

■ **Double operand instructions (12 instructions):**

Mnemonic	Operation	Description
Arithmetic instructions		
ADD(.B or .W) src,dst	src+dst →dst	Add source to destination
ADDC(.B or .W) src,dst	src+dst+C →dst	Add source and carry to destination
DADD(.B or .W) src,dst	src+dst+C →dst (dec)	Decimal add source and carry to destination
SUB(.B or .W) src,dst	dst+.not.src+1 →dst	Subtract source from destination
SUBC(.B or .W) src,dst	dst+.not.src+C →dst	Subtract source and not carry from destination
Logical and register control instructions		
AND(.B or .W) src,dst	src.and.dst →dst	AND source with destination
BIC(.B or .W) src,dst	.not.src.and.dst →dst	Clear bits in destination
BIS(.B or .W) src,dst	src.or.dst →dst	Set bits in destination
BIT(.B or .W) src,dst	src.and.dst	Test bits in destination
XOR(.B or .W) src,dst	src.xor.dst →dst	XOR source with destination
Data instructions		
CMP(.B or .W) src,dst	dst-src	Compare source to destination
MOV(.B or .W) src,dst	src→dst	Move source to destination

Lecture 03 - MSP430 ISA   15

Single Operand Instructions

Format II: Single Operand

■ Single operand instructions:

Mnemonic	Operation	Description
Logical and register control instructions		
RRA(.B or .W) dst	MSB→MSB→… LSB→C	Roll destination right
RRC(.B or .W) dst	C→MSB→…LSB→C	Roll destination right through carry
SWPB(or .W) dst	Swap bytes	Swap bytes in destination
SXT dst	bit 7→bit 8…bit 15	Sign extend destination
PUSH(.B or .W) src	SP-2→SP, src→@SP	Push source on stack
Program flow control instructions		
CALL(.B or .W) dst	SP-2→SP, PC+2→@SP dst→PC	Subroutine call to destination
RETI	@SP+→SR, @SP+→SP	Return from interrupt

Lecture 03 - MSP430 ISA   17

Jump Instructions

Jump Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Op-code			Condition			10-bit, 2's complement PC offset									

- Jump instructions are used to direct program flow to another part of the program.
- The condition on which a jump occurs depends on the Condition field consisting of 3 bits:
  - 000: jump if not equal
  - 001: jump if equal
  - 010: jump if carry flag equal to zero
  - 011: jump if carry flag equal to one
  - 100: jump if negative (N = 1)
  - 101: jump if greater than or equal (N = V)
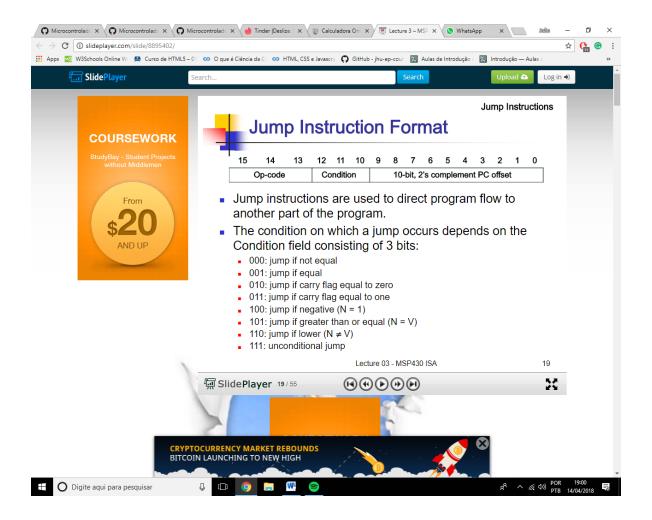 - 110: jump if lower (N ≠ V)
  - 111: unconditional jump

Lecture 03 - MSP430 ISA

19

Link foda: http://slideplayer.com/slide/8895402/

DESVIO CONDICIONAL

cmp R6, R8

seta ou reseta os bits N, Z, C e V no registrador SR (*status register*) de acordo com os valores de R6 e R8.

DESVIO CONDICIONAL

DESVIO CONDICIONAL

cmp R6, R8

seta ou reseta os bits N, Z, C e V no registrador SR (*status register*) de acordo com os valores de R6 e R8.

N = 1 se R8<R6 (R8-R6<0),
    0 caso contrário (R8>=R6).
Z = 1 se R8=R6 (R8-R6 = 0),
    0 caso contrário (R8 diferente de R6).
C = 1 se houve *carry* no cálculo R8-R6, 0 caso contrário.
V = 1 se houve *overflow* no cálculo R8-R6, 0 caso contrário.

DESVIO CONDICIONAL

cmp R6, R8
jeq LABEL_1

A primeira instrução (*cmp*) compara R6 e R8.

Se R6 for igual a R8, a segunda instrução (*jeq*) faz com que a CPU comece a executar instruções a partir da linha de código marcada com o *label LABEL_1*.

Se R6 for diferente de R8, a próxima instrução é executada normalmente.

DESVIO CONDICIONAL

cmp R6, R8
jne LABEL_2

A primeira instrução (*cmp*) compara R6 e R8.

Se R6 for diferente a R8, a segunda instrução (*jne*) faz com que a CPU comece a executar instruções a partir da linha de código marcada com o *label LABEL_2*.

Se R6 for igual a R8, a próxima instrução é executada normalmente.

DESVIO CONDICIONAL

de código marcada com o *label LABEL_2*.

Se R6 for igual a R8, a próxima instrução é executada normalmente.

DESVIO CONDICIONAL

jmp LABEL_3

Esta instrução faz com que a CPU comece a executar instruções a partir da linha de código marcada com o *label LABEL_3*, **independente de qualquer resultado anterior**.

de código marcada com o *label LABEL_2*.

Se R6 for igual a R8, a próxima instrução é executada normalmente.

DESVIO CONDICIONAL

jmp LABEL_3

Esta instrução faz com que a CPU comece a executar instruções a partir da linha de código marcada com o *label LABEL_3*, **independente de qualquer resultado anterior**.

de código marcada com o *label LABEL_2*.

Se R6 for igual a R8, a próxima instrução é executada normalmente.

DESVIO CONDICIONAL

jmp LABEL_3

Esta instrução faz com que a CPU comece a executar instruções a partir da linha de código marcada com o *label LABEL_3*, **independente de qualquer resultado anterior**.