Universidad Nacional de Ingeniería
Facultad de Ciencias

Arquitectura de computadores

INT. AL MSP430 DE TEXAS INSTRUMENTS

Prof.: Lic. César Martín Cruz S.
ce.cruz@gmail.com

2015 - I

Microcontroladores de TI

Microcontrollers (MCU)				Application (MPU)		
MSP430	**C2000**	**Tiva C**	**Hercules**	**Sitara**	**DSP**	**Multicore**
16-bit Ultra Low Power & Cost	32-bit Real-time	32-bit All-around MCU	32-bit Safety	32-bit Linux Android	16/32-bit All-around DSP	32-bit Massive Performance
MSP430 ULP RISC MCU	• Real-time C28x MCU • ARM M3+C28	**ARM Cortex-M4F**	**ARM Cortex-M3 Cortex-R4**	**ARM Cortex-A8 Cortex-A9**	**DSP C5000 C6000**	• C66 + C66 • A15 + C66 • A8 + C64 • ARM9 + C674
• Low Pwr Mode ▪ 0.1 µA ▪ 0.5 µA (RTC) • Analog I/F • USB and RF	• Motor Control • Digital Power • Precision Timers/PWM	• 32-bit Float • Nested Vector Int Ctrl (NVIC) • Ethernet (MAC+PHY)	• Lock step Dual-core R4 • ECC Memory • SIL3 Certified	• $5 Linux CPU • 3D Graphics • PRU-ICSS industrial subsys	• C5000 Low Power DSP • 32-bit fix/float C6000 DSP	• Fix or Float • Up to 12 cores 4 A15 + 8 C66x • DSP MMAC's: 352,000
TI-RTOS	TI-RTOS (k)	TI-RTOS	3rd Party (only)	Linux, Android, TI-RTOS Kernel	C5x: DSP/BIOS C6x: TI-RTOS (k)	Linux TI-RTOS (k)
Flash: 512K FRAM: 64K	512K Flash	512K Flash	256K to 3M Flash	L1: 32K x 2 L2: 256K	L1: 32K x 2 L2: 256K	L1: 32K x 2 L2: 1M + 4M
25 MHz	300 MHz	80 MHz	220 MHz	1.35 GHz	800 MHz	1.4 GHz
$0.25 to $9.00	$1.85 to $20.00	$1.00 to $8.00	$5.00 to $30.00	$5.00 to $25.00	$2.00 to $25.00	$30.00 to $225.00

LA SERIE MSP430

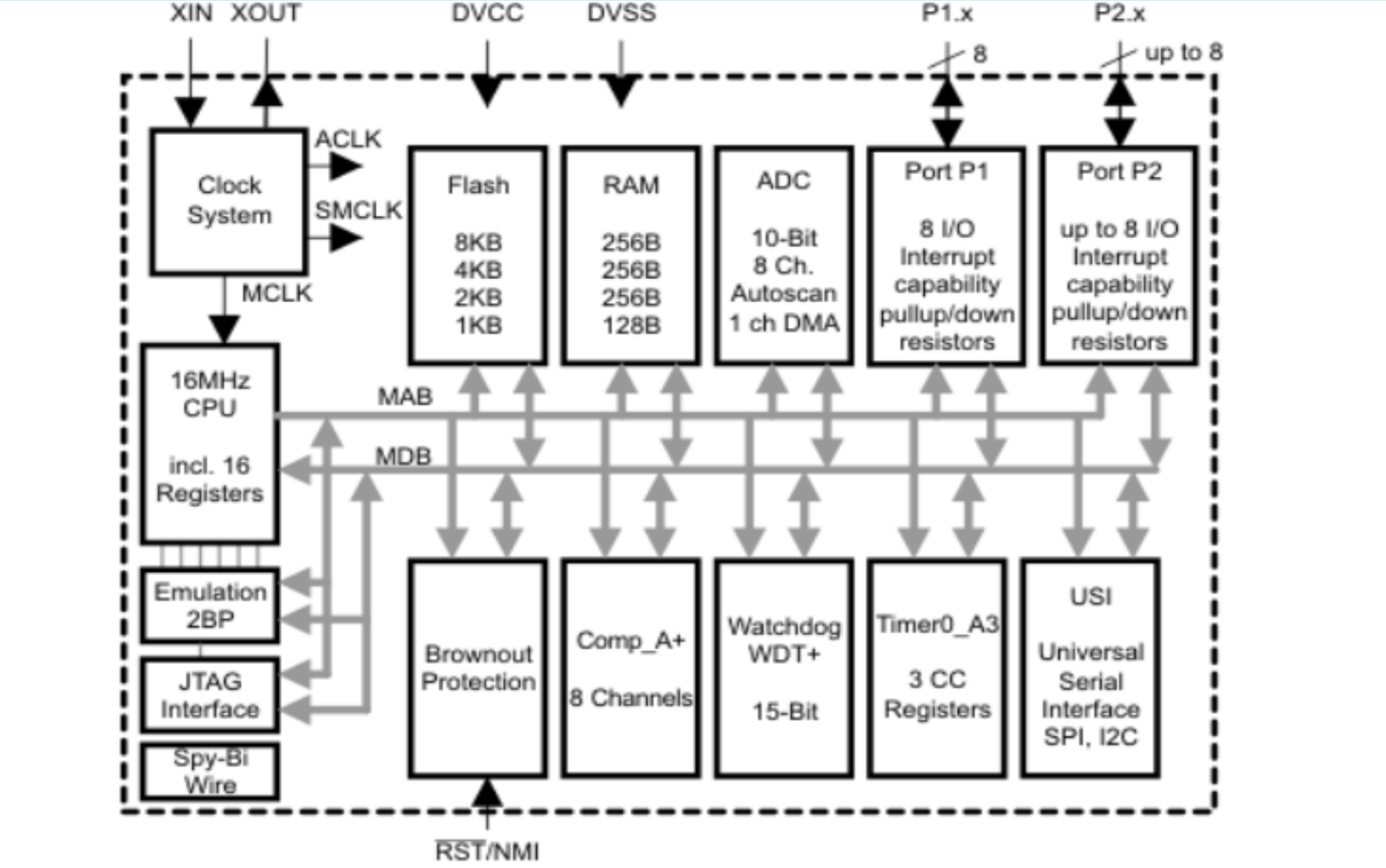
Microcontrolador de Texas Instruments **RISC** de **16 bits** de Arquitectura Von Neumann. Estos dispositivos tienen entre otras características:

- **Temporizadores** de **16 bits** y Watchdog Timer
- Ultra bajo consumo de energía
- Rango de voltaje de 1.8v a 3.6v
  - **27 instrucciones** básicas para aprender **(+24 emuladas)**. Hacen un total de **51 instrucciones**. La serie **MSP430G2xx** tiene:
  - Velocidad a una frecuencia de **16Mhz**. Flash de **0.5 – 16kB**
  - RAM de **256Bytes** o **512Bytes.**
  - Convertidor **Análogo a Digital(A/D)** de **10 bits** a 200ksps (miles de muestras por segundo), con referencia interna. Sensor de Temperatura interna en algunos modelos.
  - Puertos de Entrada/Salida de propósito general de **10 a 16 pines**
  - Interfaz Serie Universal (USI) que soporta SPI y I2C.

Comparación del MSP430G2452 y el MSP430G2553

	MSP430G2452	MSP430G2553
Frecuencia (MHz)	16	16
Flash Rom (KBytes)	8	16
SRAM (Bytes)	256	512
Timers de 16bits	1	2
USCI_A (UART/LIN/IrDA/SPI)		1
USCI_B (I2C & SPI)		1
USI: (I2C/SPI)	1	
Sensor de Temperatura	Si	Si
ADC (convertidor análogo a digital)	10 bits	10 bits
Canales de entrada del ADC	8	8

Diagrama de bloques del MSP430G2x52



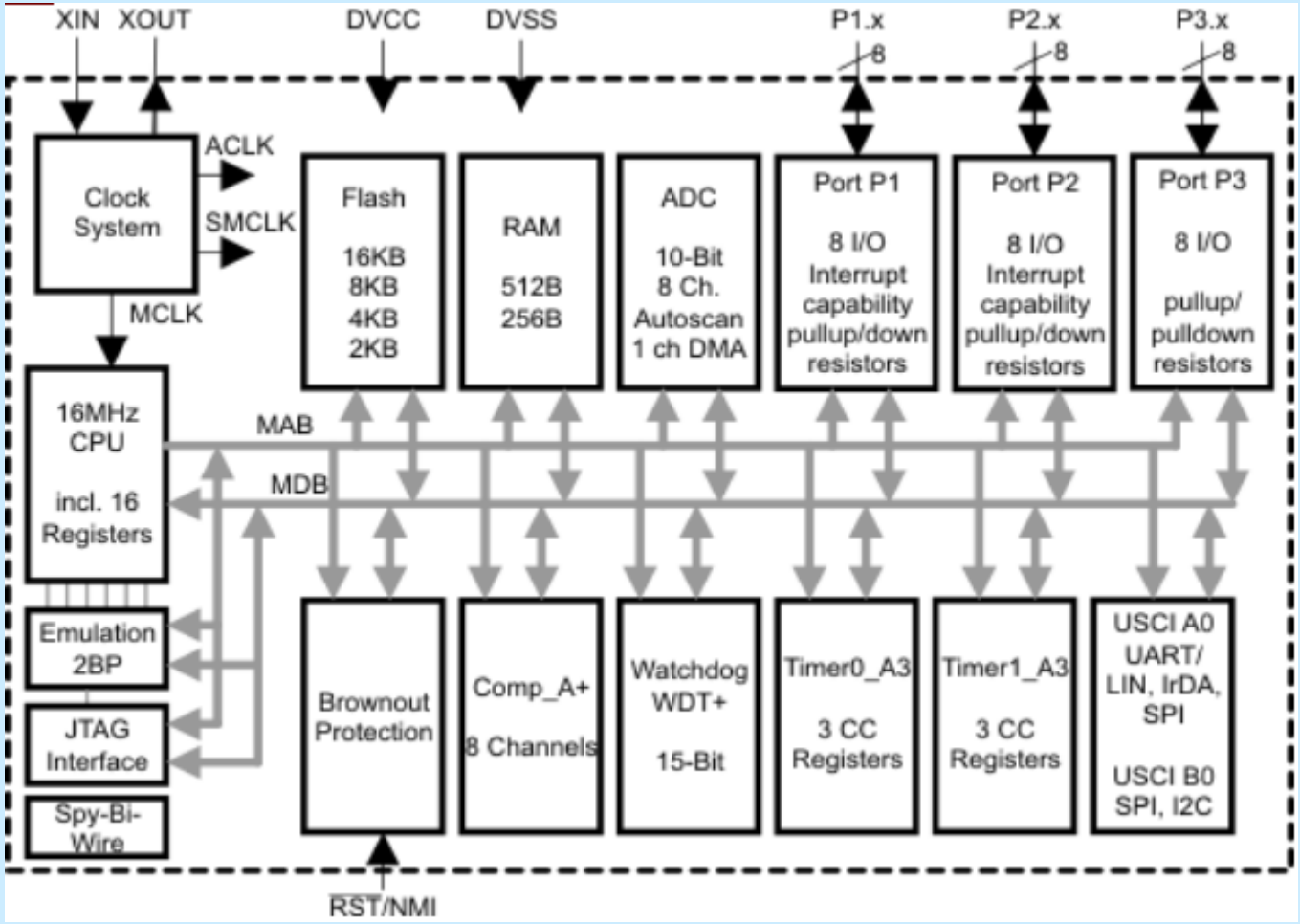
Note: Memory sizes, supported peripherals, and ports may differ depending on the device.

Descripción del MSP430G2x53

Microcontroladores de la familia MSP430 consisten de varios dispositivos que ofrecen diferentes conjuntos de periféricos específicos para diversas aplicaciones. La arquitectura, se ha optimizado para lograr una vida prolongada de la batería en aplicaciones de uso portátiles. El dispositivo cuenta con una poderosa **CPU RISC *de 16 bits***, registros **de 16-bits** que contribuyen a la eficiencia máxima del código.

La serie **MSP430G2x53** contiene ***timers de 16 bits***, hasta ***24 puertos de E/S***, una comunicación integrada que utiliza la ***interfaz universal de comunicación serie***. Convertidor ***analógico a digital*** de 10 bits. Las aplicaciones típicas incluyen sistemas de bajo costo con sensores que captan señales analógicas, estas se convierten a valores digitales, y luego de procesarlos son visualizados o transmitidos a una PC de escritorio.

Diagrama de bloques del MSP430G2x53

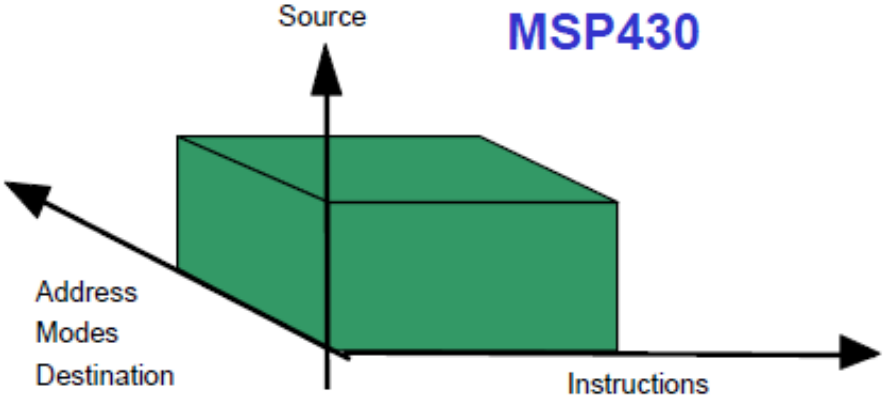


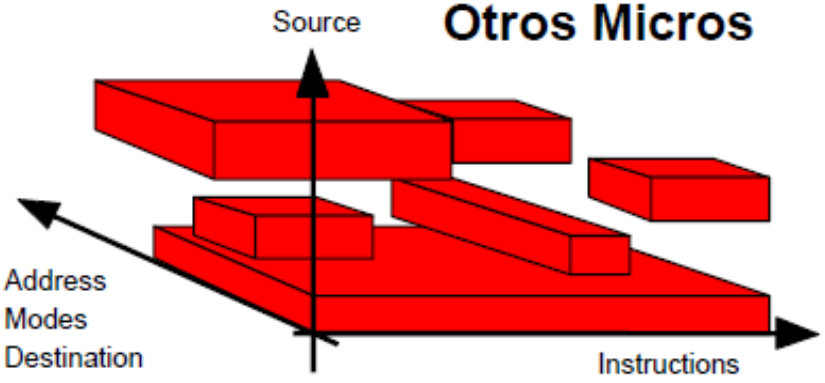
NOTA: El puerto P3 es solamente disponible en dispositivos de 28 y 32 pines

El CPU del MSP430 posee una arquitectura ortogonal

Ortogonal = todas las instrucciones funcionan con todos los modos de direccionamiento.

- Set de instrucciones reducido
- Instrucciones fáciles de aprender.
- Sin instrucciones especiales
- Bajo consumo
- Código eficiente
- Código compacto

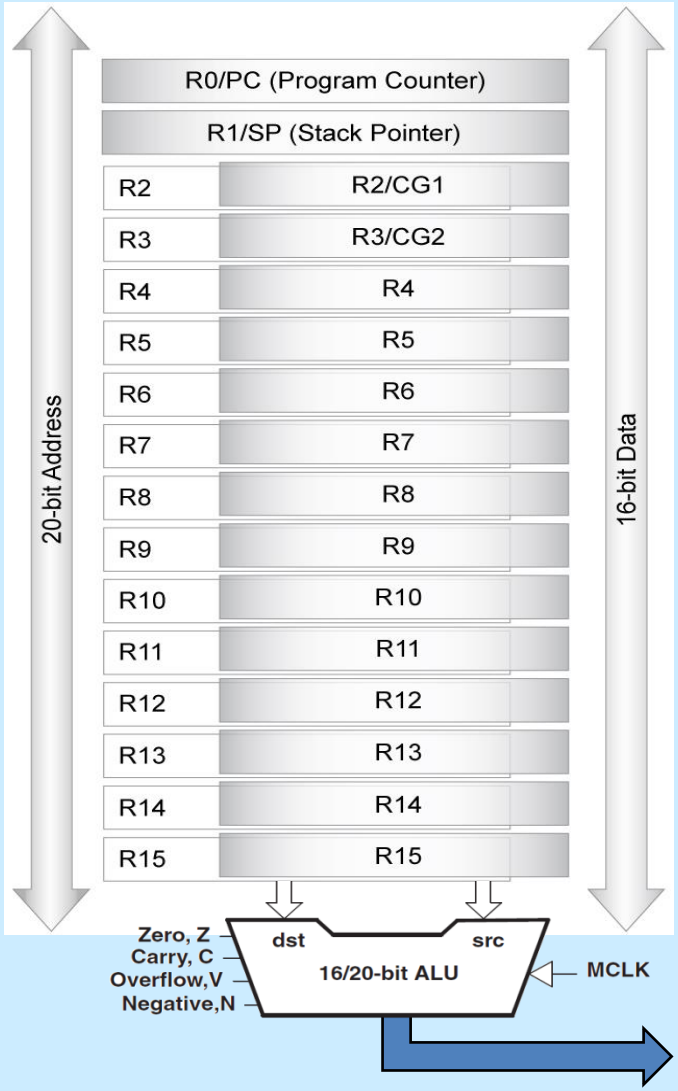




- Set de instrucciones complejo
- Instrucciones especiales a aprender
- Poco eficiente
- Mayor área de silicio.

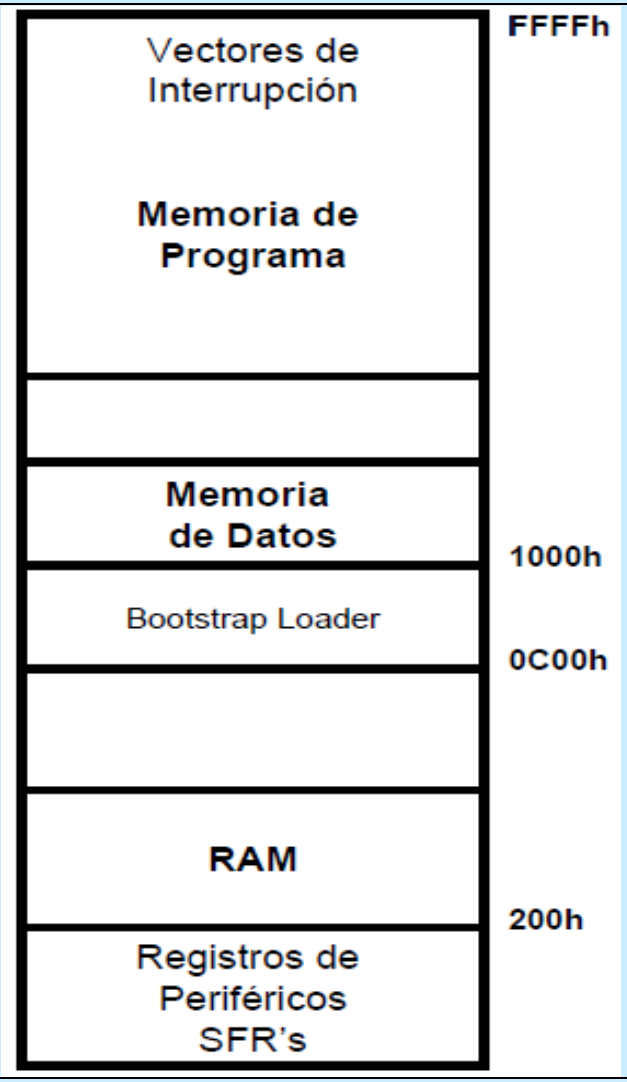
El CPU del MSP430

- ALU de 16 bits
- Buses de datos de 16 bits y direcciones de 20 bits
- 16 registros multipropósito de 16 bits(acumuladores)
  - R0/PC (Program Counter)
  - R1/SP (Stack Pointer)
  - R2/SR/CG1 (Status Register)
  - R3/CG2 (Generador Constante)
  - R4 - R15 (Propósito general)
- Acceso total a todos los registros Incluyendo PC, SP, SR y CG.
- Operaciones en un ciclo



Mapa de memoria

- Vectores de interrupción de 16 bits
- Memoria Flash, OTP o ROM para memoria de código.
- 128/256 Bytes de memoria Flash de información.
- ROM Bootstrap loader
- SRAM
- Registros de periféricos (SFRs)



Organización de la memoria para el MSP430G2553

		MSP430G2553
Memoria Flash	Tamaño	16kB
Vectores de Interrupción	Tipo Flash	0xFFFF a 0xFFC0
Memoria de código	Tipo Flash	0xFFFF a 0xC000
Información de memoria	Tamaño Tipo Flash	256 Byte 010FFh a 01000h
RAM estática	Tamaño	512 Bytes 0x03FF a 0x0200
Periféricos	16 bits	01FFh a 0100h
	8 bits	0FFh a 010h
	SFR de 8 bits	0Fh a 00h

Tipos de Instrucciones

51 instrucciones disponibles en assembler:

27 instrucciones básicas → **RISC**

24 instrucciones emuladas →**CISC**

Formato de la Instrucción	Ejemplo	Operación
Operandos dobles, *fuente - destino*	ADD R4,R5	R4 + R5 ---> R5
Único operando, *destino solamente*	CALL R8	PC -->(TOS), R8--> PC
Salto relativo, *in/condicional*	JNE	Jump-no-equal si bit Z = 0

Modos de Direccionamiento	F	D	Sintaxis	Ejemplo	Operación
Registro	✓	✓	MOV Rs,Rd	MOV R10,R11	R10 → R11
Indexado	✓	✓	MOV X(Rn),Y(Rm)	MOV 2(R5),6(R6)	M(2+R5) → M(6+R6)
Simbólico	✓	✓	MOV EDE,TONI		M(EDE) →M(TONI)
Absoluto	✓	✓	MOV &MEM,&TCDAT		M(MEM) → M(TCDAT)
Indirecto	✓		MOV @Rn,Y(Rm)	MOV @R10,Tab(R6)	M(R10) → M(Tab+R6)
Indirecto con Auto incremento	✓		MOV @Rn+,Rm	MOV @R10+,R11	M(R10) → R11, R10 + 2 → R10
Inmediato	✓		MOV #X,TONI	MOV #45,TONI	#45 → M(TONI)

Nota: F=*Fuente*, **D**=*Destino*. **S**=*Source*, **D**=*Destination*

Modo Registro

Operaciones en este modo trabajan directamente sobre los registros del procesador **R4** a **R15**, o sobre registros de funciones especiales tal como el contador de programa o registro de estado.

Ejm.: Mover el contenido del registro fuente R6 al registro destino R7. Registro R6 no es afectado.

Antes de la operación: R6=AB03h, R7=FE9Bh PC=PC_0
Operación: mov R6,R7
Después de la operación: R6=AB03h, R7=AB03h PC=PC_0 + 2

Modo Indexado

En este modo se utiliza X(Rn), donde X es una constante y Rn es uno de los registros del procesador. La posición de memoria absoluta X + Rn es direccionada.

Ejm.: Mover el contenido de la dirección (F000h + R8) al registro de destino R9.

Antes de la operación: R9=B002h, R8=050Ah 0xF50A=0345h
Operación: mov F000h(R8), R9
Después de la operación: R9=0345h, R8=050Ah 0xF50A=0345h

Modo Simbólico

En este caso el contador de programa PC es usado como la dirección base, así la constante es el offset a los datos de la PC.

Por ejemplo, supongamos que un programa usa la variable LoopCtr, que ocupa una palabra. La siguiente instrucción almacena el contenido de LoopCtr en R6:

mov.w LoopCtr,R6 *;carga el contenido de LoopCtr en R6, modo simbólico.*

El ensamblador remplaza esto por la forma indexada:

mov.w X(PC) , R6 *;carga el contenido de LoopCtr=X + PC en R6, modo simbólico.*

Modo Absoluto

Similar al modo simbólico, con la diferencia que la etiqueta es precedida por “&”.
Se usa un tipo de direccionamiento indexado cuya dirección es la dirección absoluta del dato. Esto es la dirección completa es requerida que se añade a un registro que contiene 0.
El MSP430 usa el registro de estado SR, que en este caso tiene como contenido 0. Esto es, X(SR).

Por Ejm.:
mov &XPT, &YPT ;mueve el contenido de la dirección absoluta XPT a YPT
Otro ejemplo, mover el contenido del registro del puerto P1 de entrada al registro R6.
mov.b &P1IN, R6 ;carga byte en P1IN a R6, modo absoluto
El ensamblador remplaza esto por la forma indexada
mov.b P1IN(SR),R6 ;carga byte en P1IN a R6, modo absoluto

P1IN es la dirección absoluta del registro.

Modo Indirecto con registro

La palabra direccionada es localizada en la posición de memoria apuntada por Rn.

Ejm.:
Mueve el contenido de la dirección que es contenido de R4 al registro destino R5.
Antes de la operación: R4=A002h, R5=050Ah, (0xA002)=0345h
Operación: mov.w @R4,R5
Después de la operación: R4=A002h, R5=0345h, (0xA002)=0345h

Modo indirecto con autoincremento de registro

Es disponible solamente para la fuente y es mostrado por el símbolo @ en frente de un registro con un signo + después de él, tal como @Rn+. Usa el valor de Rn como un puntero y automáticamente se incrementa en 1 si un byte ha sido buscado o por un 2 si fue una palabra.

Antes de la operación: R4=A002h, R5=050Ah (0xA002)=0567h
Operación: mov @R4+,R5
Después de operación: R4=A004h, R5=0567h (0xA002)=0567h

Modo Inmediato

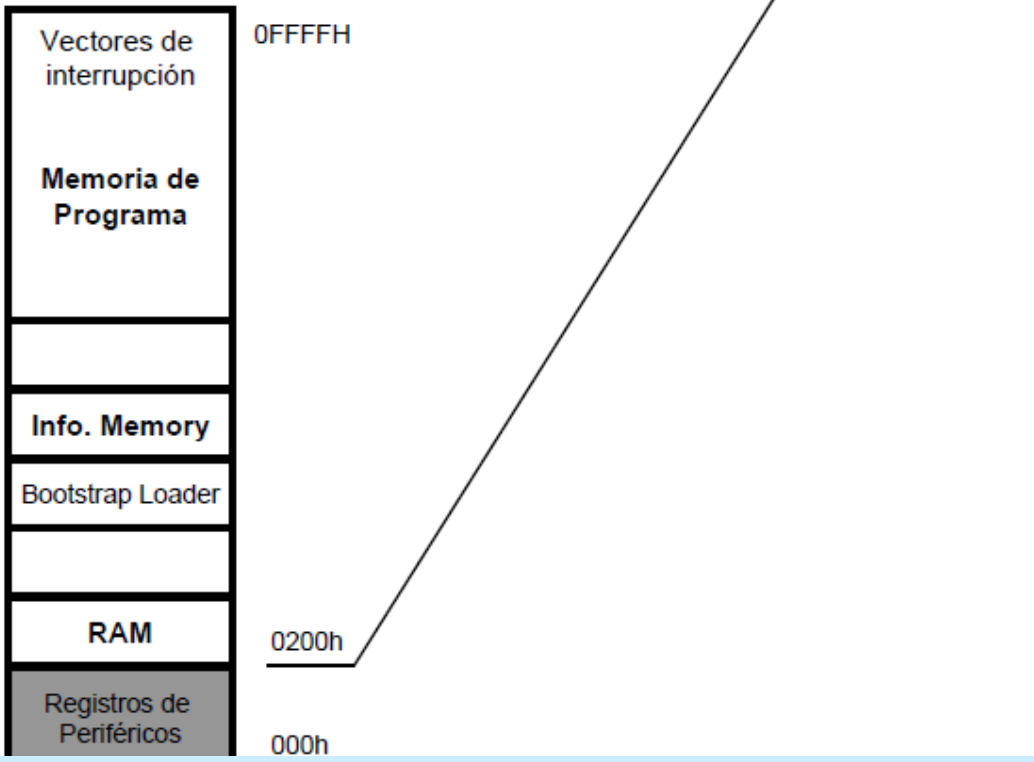
El modo inmediato es usado para asignar los valores constantes a registros o posiciones de memoria.

Ejm.:
Antes de la operación: R4=A002h, R5=050Ah
Operación: mov.w #E2h, R5
Después de la operación: R4=A002h, R5=00E2h

Registros de Periféricos

MSP430 usa arquitectura Von Newan

- Todos los periféricos están mapeados en memoria.
- Se puede utilizar cualquier instrucción con estos registros



PERIPHERALS WITH WORD ACCESS			
Timer_A	Reserved		017Eh
	Reserved		017Ch
	Reserved		017Ah
	Reserved		0178h
	Capture/compare register	CCR2	0176h
	Capture/compare register	CCR1	0174h
	Capture/compare register	CCR0	0172h
	Timer_A register	TAR	0170h
	Reserved		016Eh
	Reserved		016Ch
	Reserved		016Ah
	Reserved		0168h
	Capture/compare control	CCTL2	0166h
	Capture/compare control	CCTL1	0164h
	Capture/compare control	CCTL0	0162h
	Timer_A control	TACTL	0160h
	Timer_A interrupt vector	TAIV	012Eh
Flash Memory	Flash control 3	FCTL3	012Ch
	Flash control 2	FCTL2	012Ah
	Flash control 1	FCTL1	0128h
Watchdog	Watchdog/timer control	WDTCTL	0120h

PERIPHERALS WITH BYTE ACCESS			
Comparator_A	Comparator_A port disable	CAPD	05Bh
	Comparator_A control2	CACTL2	05Ah
	Comparator_A control1	CACTL1	059h
System Clock	Basic clock sys. control2	BCSCTL2	058h
	Basic clock sys. control1	BCSCTL1	057h
	DCO clock freq. control	DCOCTL	056h
Port P2	Port P2 selection	P2SEL	02Eh
	Port P2 interrupt enable	P2IE	02Dh
	Port P2 interrupt edge select	P2IES	02Ch
	Port P2 interrupt flag	P2IFG	02Bh
	Port P2 direction	P2DIR	02Ah
	Port P2 output	P2OUT	029h
	Port P2 input	P2IN	028h
Port P1	Port P1 selection	P1SEL	026h
	Port P1 interrupt enable	P1IE	025h
	Port P1 interrupt edge select	P1IES	024h
	Port P1 interrupt flag	P1IFG	023h
	Port P1 direction	P1DIR	022h
	Port P1 output	P1OUT	021h
	Port P1 input	P1IN	020h
Special Function	SFR interrupt flag2	IFG2	003h
	SFR interrupt flag1	IFG1	002h
	SFR interrupt enable2	IE2	001h
	SFR interrupt enable1	IE1	000h

Registros de control del puerto P1

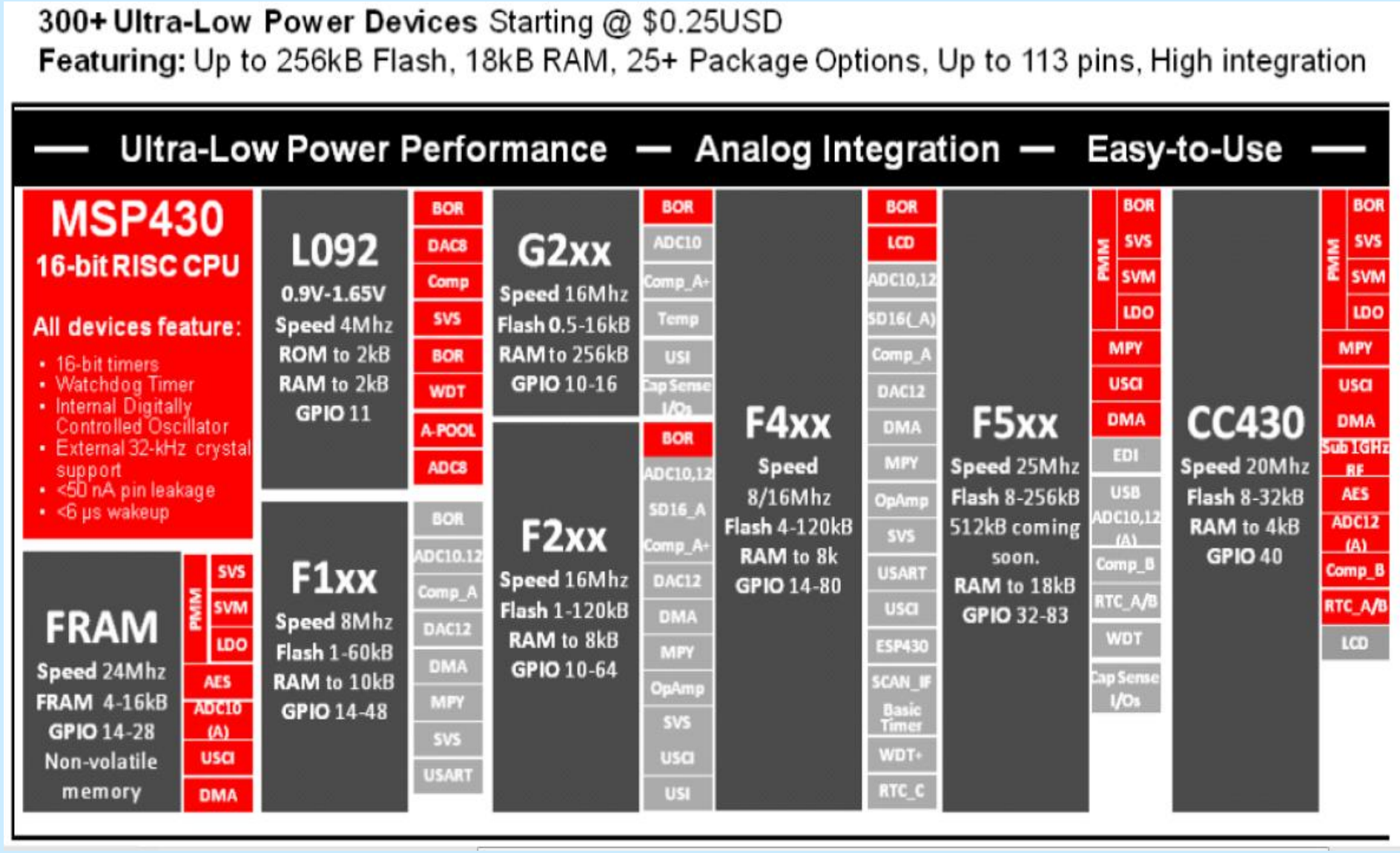
1. **P1DIR:** Se usa para configurar si un pin del puerto se toma como entrada o salida. “1” en la posición del pin lo configura como salida y “0” como entrada.
2. **P1IN:** Registro de solo lectura, su valor es el de la entrada del puerto.

3. **P1OUT:** De lectura y escritura, su valor es el de la salida del puerto.

4. **P1IFG:** Flags de interrupción de las correspondientes patillas, indica qué patillas están solicitando interrupción. Es de lectura y escritura.
   Este registro es propio de los puertos uno y dos.

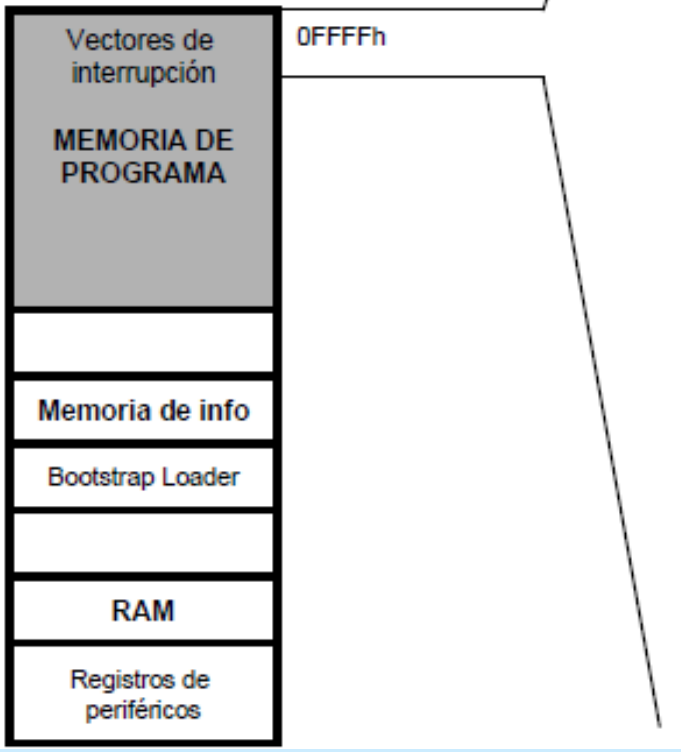
5. **P1IES:** Sirve para seleccionar individualmente si las interrupciones se solicitarán con flancos de subida o de bajada. Es propio sólo de los puertos uno y dos, siendo de lectura y escritura.

Dispositivos MSP430



Vectores de Interrupción MSP430

- 16 vectores disponibles
- Los vectores se cargan en el PC
- PC y SR se respaldan en el stack en forma automática.



Fuente	Flag	Interr.	Direccion	Prior.
Pwr.Up. Wdt	WDTIFG	RESET	0FFFEh	15
NMI, OF, FV.	NMIIFG	No masc	0FFFCh	14
Timer B	TBCCR0	Masc	0FFFAh	13
Timer B	TBCCR1-6	Masc	0FFF8h	12
Comparador A	CAIFG	Masc	0FFF6h	11
WDT (interv)	WDTIFG	Masc	0FFF4h	10
USART0 RX	URXIFG0	Masc	0FFF2h	9
USART0 TX	UTXIFG0	Masc	0FFF0h	8
ADC12	ADC12IFG	Masc	0FFEEh	7
Timer A	TACCR0	Masc	0FFECh	6
Timer B	TACCR1-2	Masc	0FFEAh	5
I/O port P1	P1IFG0-7	Masc	0FFE8h	4
USART1 RX	URXIFG1	Masc	0FFE6h	3
USART1 TX	UTXIFG1	Masc	0FFE4h	2
I/O port P2	P2IFG0-7	Masc	0FFE2h	1
			0FFE0h	0

Watchdog Timer del MSP430

- Puede operar como Watchdog Timer o temporizador.
- Cualquier acceso al registro WDTCTL esta protegido por contraseña.
- 8 Opciones de post-scaller configurables por software.
- Fuente de reloj seleccionable por software *(dependiente)*.
- Registro de control mapeado en ram.
- Completamente configurable por software.
- Puede ser detenido para ahorrar energía.

Conjunto de Instrucciones

Instrucciones con operando dobles

Nemónico	Operación	Descripción
Instrucciones Aritméticas		
ADD(.B or .W) src,dst	src+dst→dst	*Suma fuente a destino*
ADDC(.B or .W) src,dst	src+dst+C→dst	*Suma fuente y carry a destino*
DADD(.B or .W) src,dst	src+dst+C→dst (dec)	*Suma decimal fuente y carry a destino*
SUB(.B or .W) src,dst	dst+.not.src+1→dst	*Resta source de destination*
SUBC(.B or .W) src,dst	dst+.not.src+C→dst	*Resta source y not carry de destino*
Instrucciones lógicas y de control		
AND(.B or .W) src,dst	src.and.dst→dst	*AND fuente con destino*
BIC(.B or .W) src,dst	not.src.and.dst→dst	*Clear bits en destino*
BIS(.B or .W) src,dst	src.or.dst→dst	*Set bits en destino*
BIT(.B or .W) src,dst	src.and.dst	*Test bits en destino*
XOR(.B or .W) src,dst	src.xor.dst→dst	*XOR fuente con destino*
Instrucciones de datos		
CMP(.B or .W) src,dst	dst-src	*Compara fuente a destino*
MOV(.B or .W) src,dst	src→dst	*Mueve fuente a destino*

Conjunto de Instrucciones

Instrucciones de un único operando

Nemónico	Operación	Descripción
Instrucciones Lógicas y de control		
RRA(.B or .W) dst	MSB→MSB→…LSB→C	*Roll destination right*
RRC(.B or .W) dst	C→MSB→…LSB→C	*Roll destino right atravez del carry*
SWPB(.B or .W) dst	Swap bytes	*Swap bytes en destino*
SXT dst	bit 7→bit 8…bit	*Sign extend destination*
PUSH(.B or .W) src	SP-2→SP, src→@SP	*Push fuente a la pila*
Instrucciones de control de flujo de programa		
CALL(.B or .W)dst	SP-2→SP, PC+2→@SP Dst→PC	*llama subrutina en destino*
RETI	TOS→SR, SP+2→SP TOS→PC, SP+2→SP	*Retorna de la interrupción*

Conjunto de Instrucciones
Instrucciones de control de flujo de programa

Nemónico	Descripción
Instrucciones de control de flujo de programa	
JEQ/JZ label	*Salta a label si flag Z es uno*
JNE/JNZ label	*Salta a label si flag Z es cero*
JC label	*Salta a label si flag carry es uno*
JNC label	*Salta a label si flag carry es cero*
JN label	*Salta a label si flag negative es uno*
JGE label	*Salta a label si más grande que o igual*
JL label	*Salta a label si menor que*
JMP label	*Salta a label incondicionalmente*

Conjunto de Instrucciones

Instrucciones emuladas

Mnemonic	Operation	Emulation	Description
Arithmetic instructions			
ADC(.B or .W) dst	dst+C→dst	ADDC(.B or .W) #0,dst	Add carry to destination
DADC(.B or .W) dst	dst+C→dst (decimally)	DADD(.B or .W) #0,dst	Decimal add carry to destination
DEC(.B or .W) dst	dst-1→dst	SUB(.B or .W) #1,dst	Decrement destination
DECD(.B or .W) dst	dst-2→dst	SUB(.B or .W) #2,dst	Decrement destination twice
INC(.B or .W) dst	dst+1→dst	ADD(.B or .W) #1,dst	Increment destination
INCD(.B or .W) dst	dst+2→dst	ADD(.B or .W) #2,dst	Increment destination twice
SBC(.B or .W) dst	dst+0FFFFh+C→dst dst+0FFh→dst	SUBC(.B or .W) #0,dst	Subtract source and borrow /.NOT. carry from dest.

Conjunto de Instrucciones
Instrucciones emuladas

Mnemonic	Operation	Emulation	Description
Logical and register control instructions			
INV(.B or .W) dst	.NOT.dst→dst	XOR(.B or .W) #0(FF)FFh,dst	Invert bits in destination
RLA(.B or .W) dst	C←MSB←MSB-1 LSB+1←LSB←0	ADD(.B or .W) dst,dst	Rotate left arithmetically
RLC(.B or .W) dst	C←MSB←MSB-1 LSB+1←LSB←C	ADDC(.B or .W) dst,dst	Rotate left through carry
Program flow control			
BR dst	dst→PC	MOV dst,PC	Branch to destination
DINT	0→GIE	BIC #8,SR	Disable (general) interrupts
EINT	1→GIE	BIS #8,SR	Enable (general) interrupts
NOP	None	MOV #0,R3	No operation
RET	@SP→PC SP+2→SP	MOV @SP+,PC	Return from subroutine

Conjunto de Instrucciones

Instrucciones emuladas

Mnemonic	Operation	Emulation	Description
Data instructions			
CLR(.B or .W) dst	0→dst	MOV(.B or .W) #0,dst	Clear destination
CLRC	0→C	BIC #1,SR	Clear carry flag
CLRN	0→N	BIC #4,SR	Clear negative flag
CLRZ	0→Z	BIC #2,SR	Clear zero flag
POP(.B or .W) dst	@SP→temp SP+2→SP temp→dst	MOV(.B or .W) @SP+,dst	Pop byte/word from stack to destination
SETC	1→C	BIS #1,SR	Set carry flag
SETN	1→N	BIS #4,SR	Set negative flag
SETZ	1→Z	BIS #2,SR	Set zero flag
TST(.B or .W) dst	dst + 0FFFFh + 1 dst + 0FFh + 1	CMP(.B or .W) #0,dst	Test destination

El LaunchPad MSP430

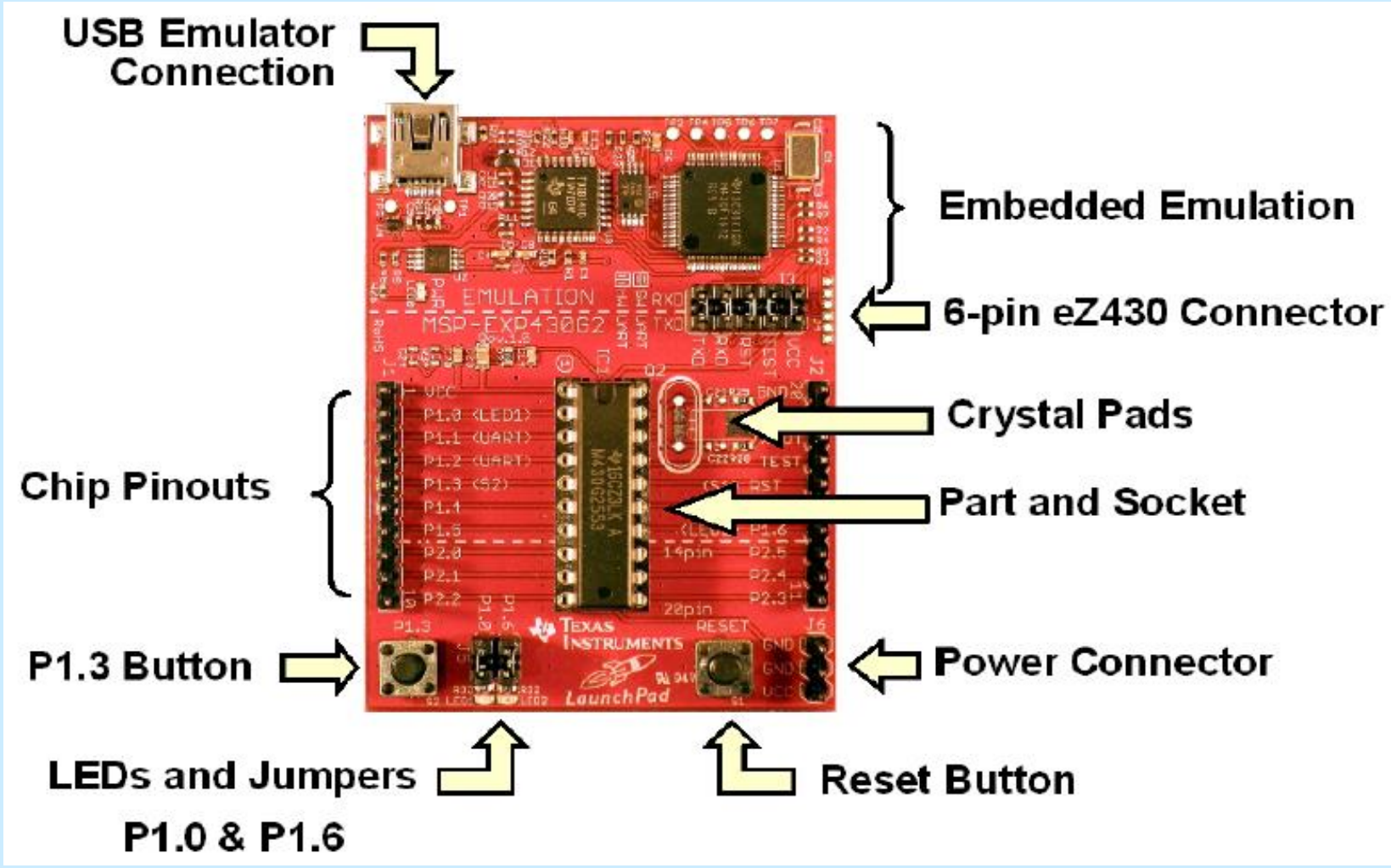
Herramienta de bajo costo para la evaluación y el aprendizaje de la serie MSP430.

Conexiones en el LaunchPad MSP430

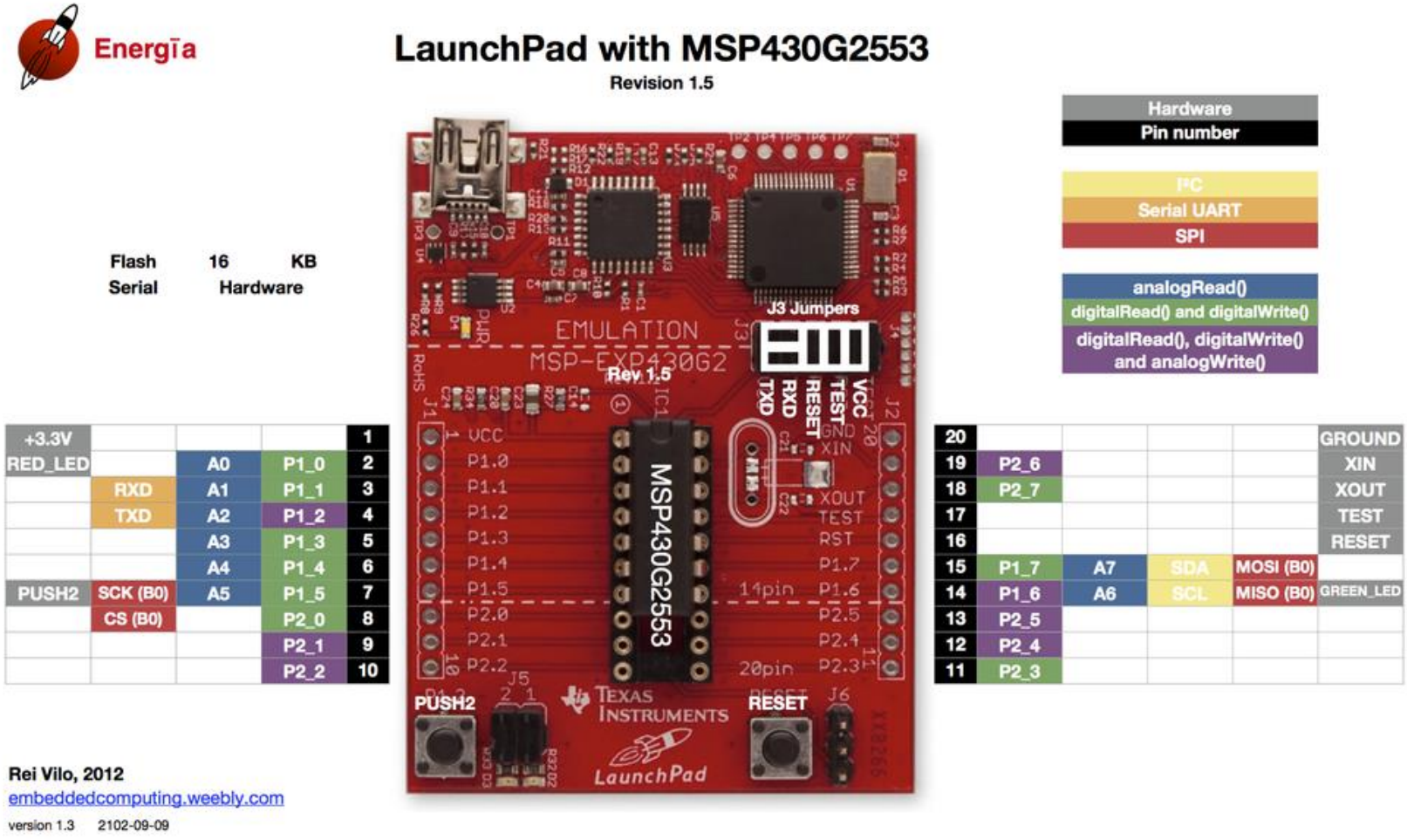
Sobre la placa algunos pines están conectados de la siguiente forma:

- P1.0 =LED1(color rojo)
- P1.6=LED2(color verde)
- P1.3=SWITCH2(Botón 2(S2))
- RESET=SWITCH1 (Botón 1(S1))
- P1.1=Transmisión serie UART
- P1.2=Recepción serie UART

Disposición de componentes en el LaunchPad MSP430



LaunchPad con MSP430G2553



Para apagar y encender un LED en el LaunchPad MSP430

- LaunchPad Development Tool
  - P1.0 Red LED
- Habilita el bit del puerto para salida escribiendo un 1 al puerto en el registro de dirección

```
bis.b #0x01,&P1DIR    ; P1.0 como salida
```

- Apaga LED escribiendo un 0 al pin del puerto

```
bic.b #0x01,&P1OUT    ; apaga LED
```

- Enciende LED escribiendo un 1 al pin del puerto

```
bis.b #0x01,&P1OUT    ; enciende LED
```

- Conmutar el LED by XOR'ing a 1 al pin del puerto

```
xor.b #0x01,&P1OUT    ; CONMUTA el LED
```

Para apagar y encender un LED….

```
;-------------------------------------------------------------------------
        .cdecls C,LIST,"msp430g2553.h"       ; Include device header file
;-------------------------------------------------------------------------
        .text                           ; Assemble into program memory
        .retain                         ; Override ELF conditional linking
                                        ; and retain current section
        .retainrefs                     ; Additionally 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
            bis.b   #0x01,&P1DIR            ; P1.0 como salida
            bis.b   #0x01,&P1OUT            ; enciende LED rojo
repite      xor.b   #0x01,&P1OUT            ; conmuta el LED rojo
espera      mov.w #65000, R15
otra_vez    dec.w R15
```

```
        jne otra_vez
        jmp repite
;-------------------------------------------------------------------------
;           Stack Pointer definition
;-------------------------------------------------------------------------
```

Otro ejemplo de programa

Lee el estado del switch en P1.3 (Notar que P1.3 es “1” cuando el botón no está presionado y “0” cuando el botón es presionado).
Se enciende el led rojo si el botón no es presionado (P1.0).
Se enciende el led verde si el botón es presionado (P1.6).

```
;-------------------------------------------------------------------------
        .cdecls C,LIST,"msp430g2553.h"       ; Include device header file
;-------------------------------------------------------------------------
        .text                           ; Assemble into program memory
        .retain                         ; Override ELF conditional linking
                                        ; and retain current section
        .retainrefs                     ; Additionally retain any sections
                                        ; that have references to current
                                        ; section
;-------------------------------------------------------------------------
```

Ejemplo de un programa ...

```
RESET       mov.w   #__STACK_END,SP         ; Initialize stackpointer
StopWDT     mov.w   #WDTPW|WDTHOLD,&WDTCTL  ; Stop watchdog timer
;-------------------------------------------------------------------------
                                            ; Main loop here
            bis.b #01000001b, &P1DIR        ; P1.0 y P1.6 son salidas
            mov.b #8h,&P1OUT                ; Se fija en 1 el pin P1.3
            bis.b #8h,&P1REN                ; Se fija PULL UP en el pin P1.3
repite      bit.b #00001000b, &P1IN         ; Se ve el estado del pin P1.3
            jc off
on          bis.b #01000000b,&P1OUT             ; Se enciende el LED verde
            bic.b #00000001b,&P1OUT             ; Se apaga el LED rojo
            jmp salir
off         bis.b #00000001b,&P1OUT             ; Se enciende el LED rojo
            bic.b #01000000b,&P1OUT             ; Se apaga el LED verde
salir       jmp repite
;-------------------------------------------------------------------------
;-------------------------------------------------------------------------
;           Stack Pointer definition
```

Ejemplo de un programa ...

```
;-------------------------------------------------------------------------
        .global __STACK_END
        .sect       .stack

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