CPE/EE 421/521

The MSP430 Family

Dr. Rhonda Kay Gaede

UAH

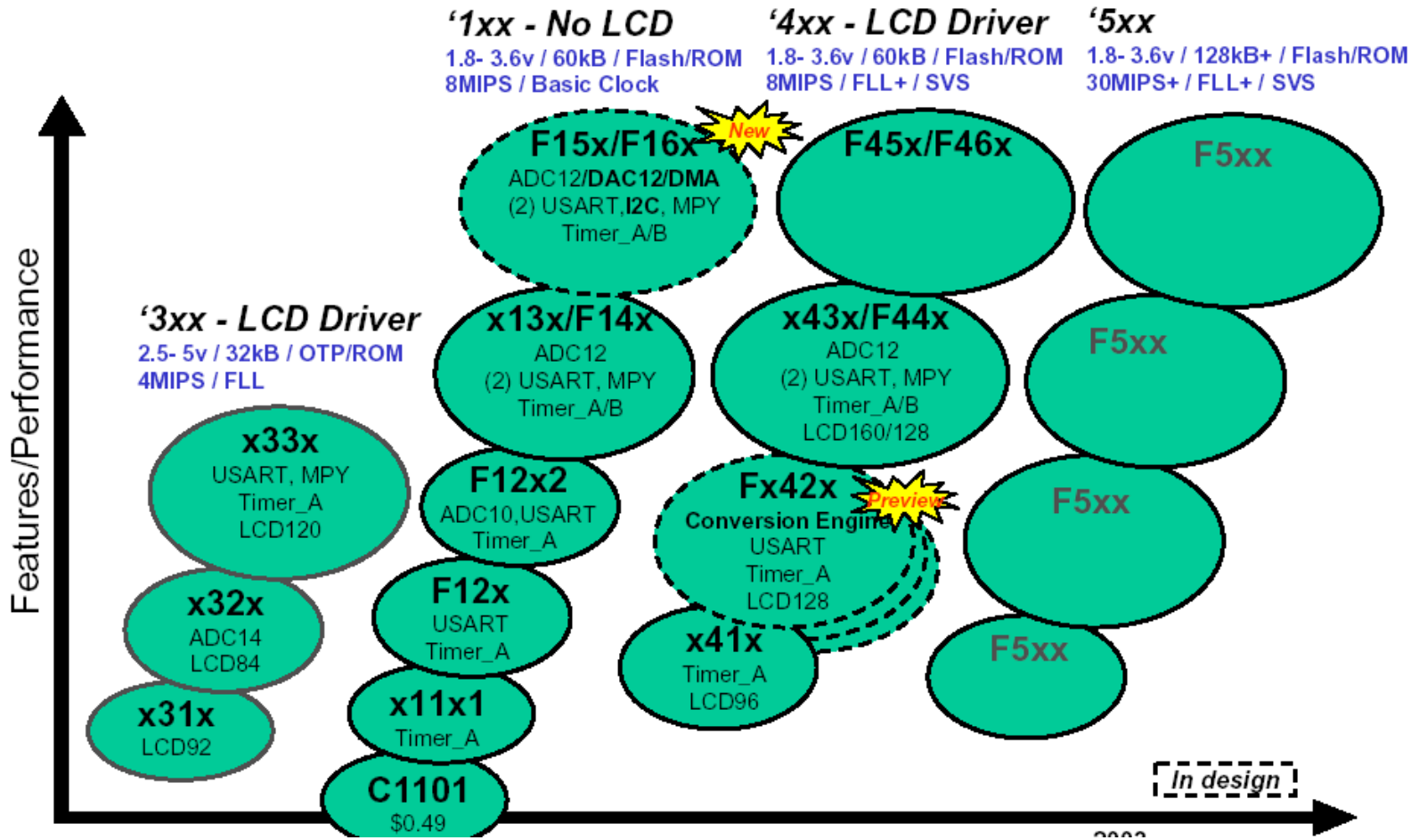
The MSP430 Family

- Broad family of TI's 16-bit microcontrollers
  - from 1Kbytes ROM, 128 bytes RAM (approx. $1)
  - to 60Kbytes ROM, 2K RAM ($10)
- Three subfamilies
  - MSP430x1xx: basic unit
  - MSP430x3xx: more features
  - MSP430x4xx: built-in LCD driver

The MSP430 Family – Part Numbering Convention

- MSP430$M_t F_a F_b M_c$
  - Mt : Memory type
    - C – ROM, F – Flash, P – OTP, E – EPROM
  - F_a, F_b
    - 10, 11 – basic
    - 12, 13 – HW UART
    - 14 – HW UART, HW multiplier
    - 31, 32 – LCD Controller
    - 33 – LCD controller, HW UART, HW multiplier
    - 41 – LCD controller
    - 43 - LCD controller, HW UART
    - 44 - LCD controller, HW UART, HW multiplier
  - M_c : Memory capacity
    - 0: 1 Kb ROM, 128 b RAM
    - 1: 2 KB ROM, 128 b RAM
    - 2: 4 KB ROM, 256 b RAM
    - ....
    - 9: 60 KB ROM, 2 Kb RAM

The MSP430 Family – Roadmap



The MSP430 Family – Typical Applications

Handheld Measurement
- Air Flow measurement
- Alcohol meter
- Barometer
- Data loggers
- Emission/Gas analyser
- Humidity measurement
- Temperature measurement
- Weight scales

Medical Instruments
- Blood pressure meter
- Blood sugar meter
- Breath measurement
- EKG system

Utility Metering
- Gas Meter
- Water Meter
- Heat Volume Counter
- Heat Cost Allocation
- Electricity Meter
- Meter reading system (RF)

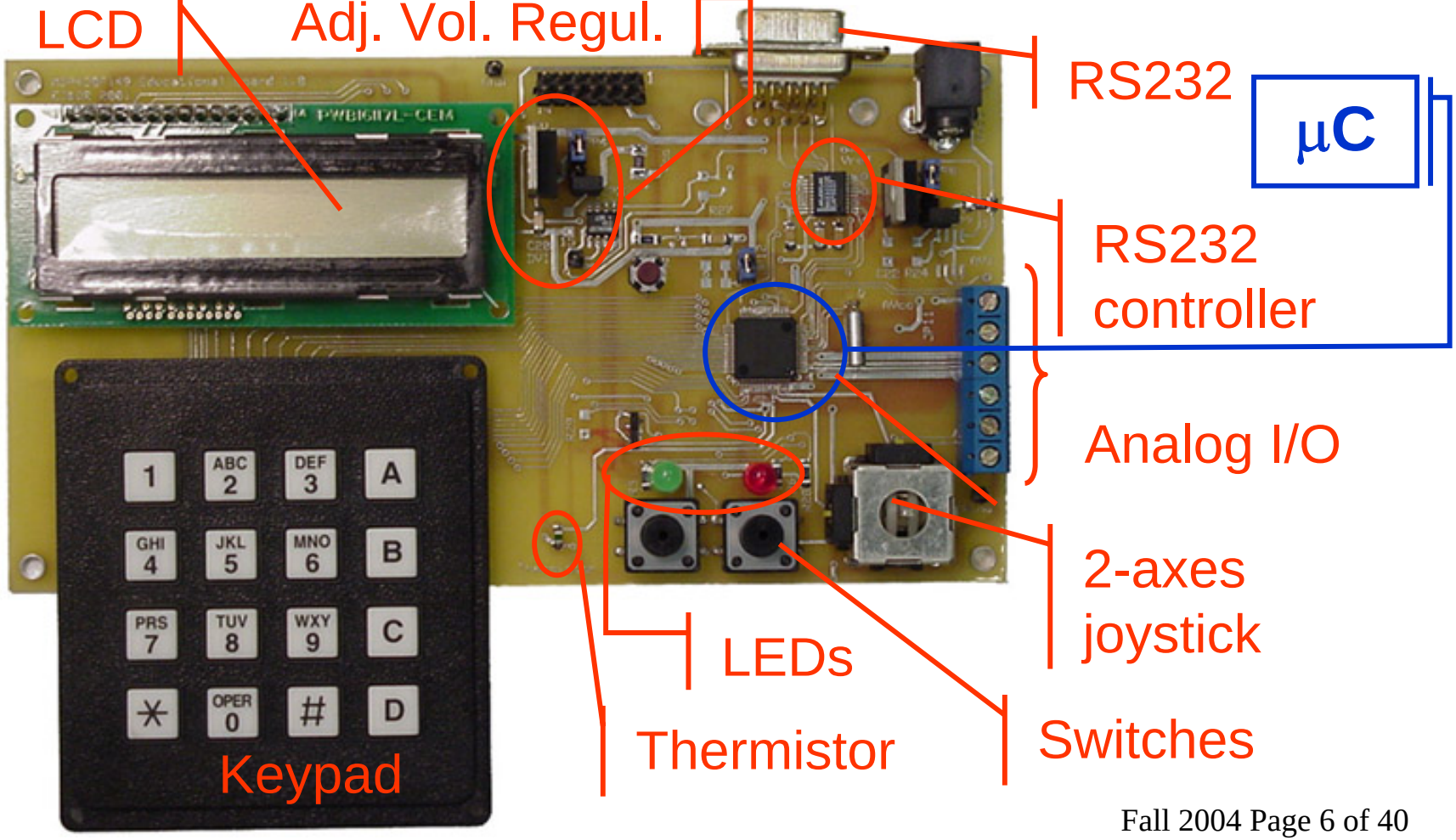
Sports equipment
- Altimeter
- Bike computer
- Diving watches

Security
- Glass break sensors
- Door control
- Smoke/fire/gas detectors

Home environment
- Air conditioning
- Control unit
- Thermostat
- Boiler control
- Shutter control
- Irrigation system
- White goods (Washing machine,..)

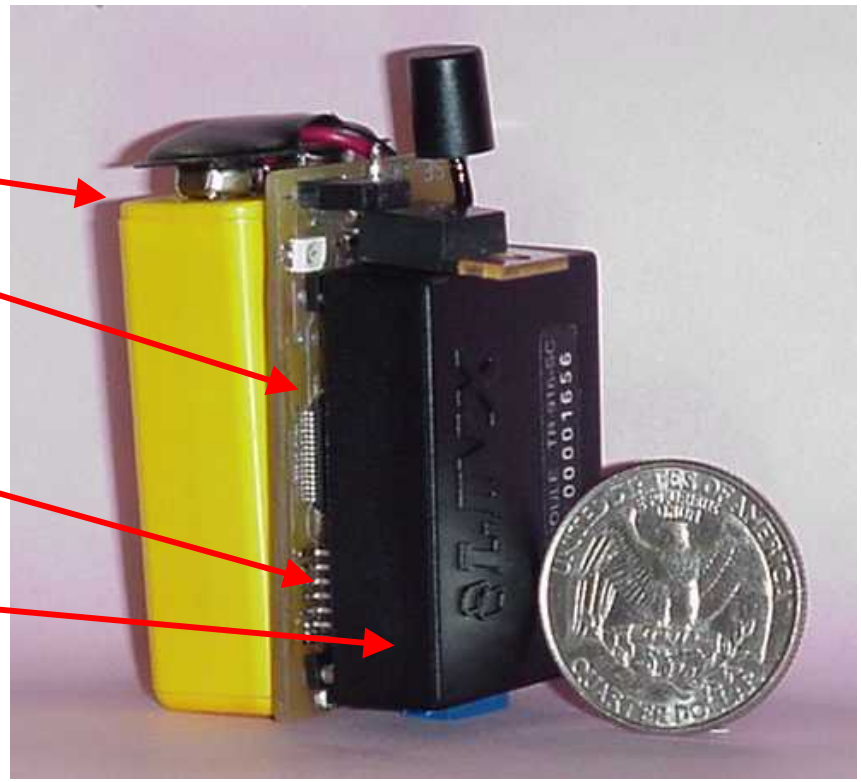
Misc
- Smart card reader
- Taxi meter
- Smart Batteries

The MSP430 Family – An MSP430-Based System



The MSP430 Family – Another MSP430-Based System: Basic WISE

- Battery
- Microcontroller
  - TI MSP430F149
  - 8-channel 12-bit AD conv.
- Accelerometer
  - Movement detection
  - Analog Device ADXL202
- Transceiver
  - LINX 916 MHz

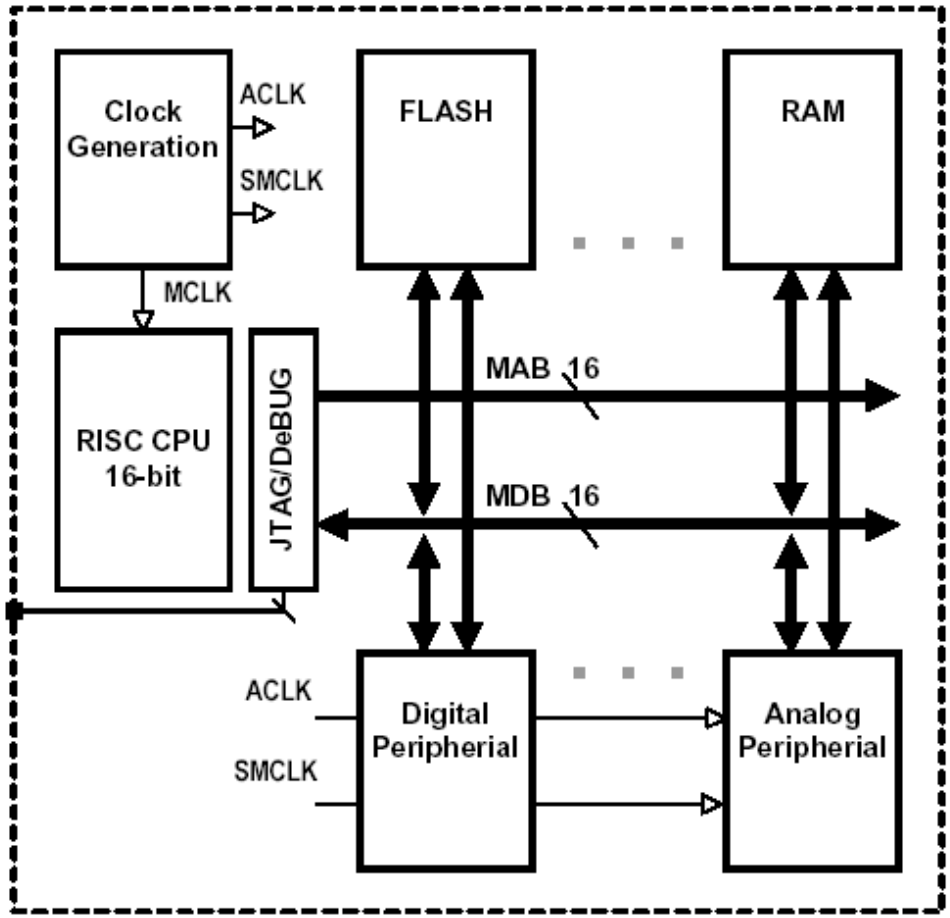


The MSP430 Family – Documentation

- MSP430 home page (TI)
  - www.ti.com/msp430
- User's manual
  - http://www.ece.uah.edu/~gaede/manuals/slau049c.pdf
- Datasheet
  - http://www.ece.uah.edu/~gaede/manuals/slas272c.pdf
- TI Workshop document
  - http://www.ece.uah.edu/~gaede/manuals/430_2002_atc_workshop.pdf
- IAR Workbench Tutorial
  - http://www.ece.uah.edu/~gaede/manuals/TUTOR.pdf

The MSP430 Family – Modular Architecture

von-Neumann common bus connects CPU to all memory and peripherals



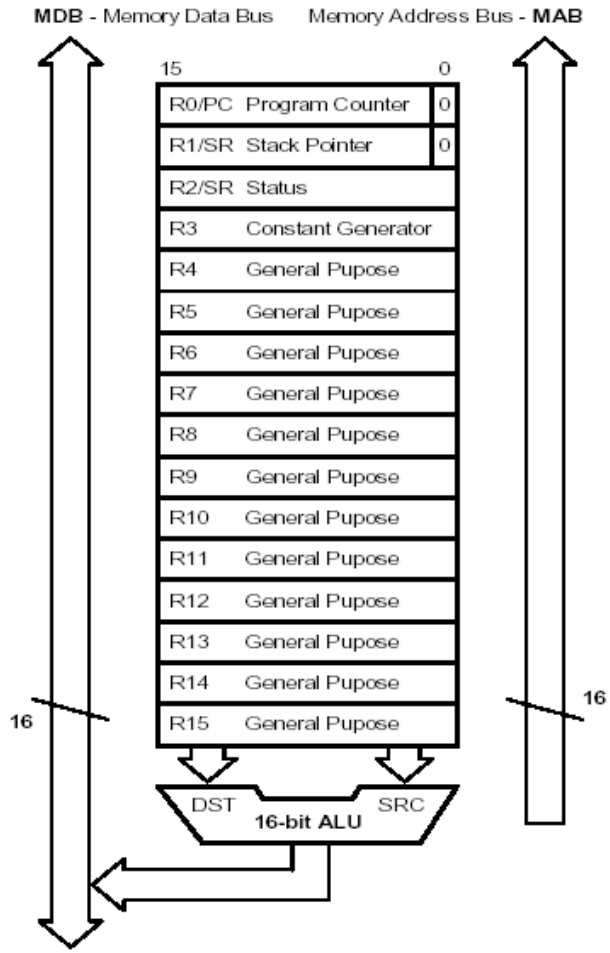
Architecture reduces power consuming, noise generating fetches to memory

Embedded emulation accessed in-application with JTAG

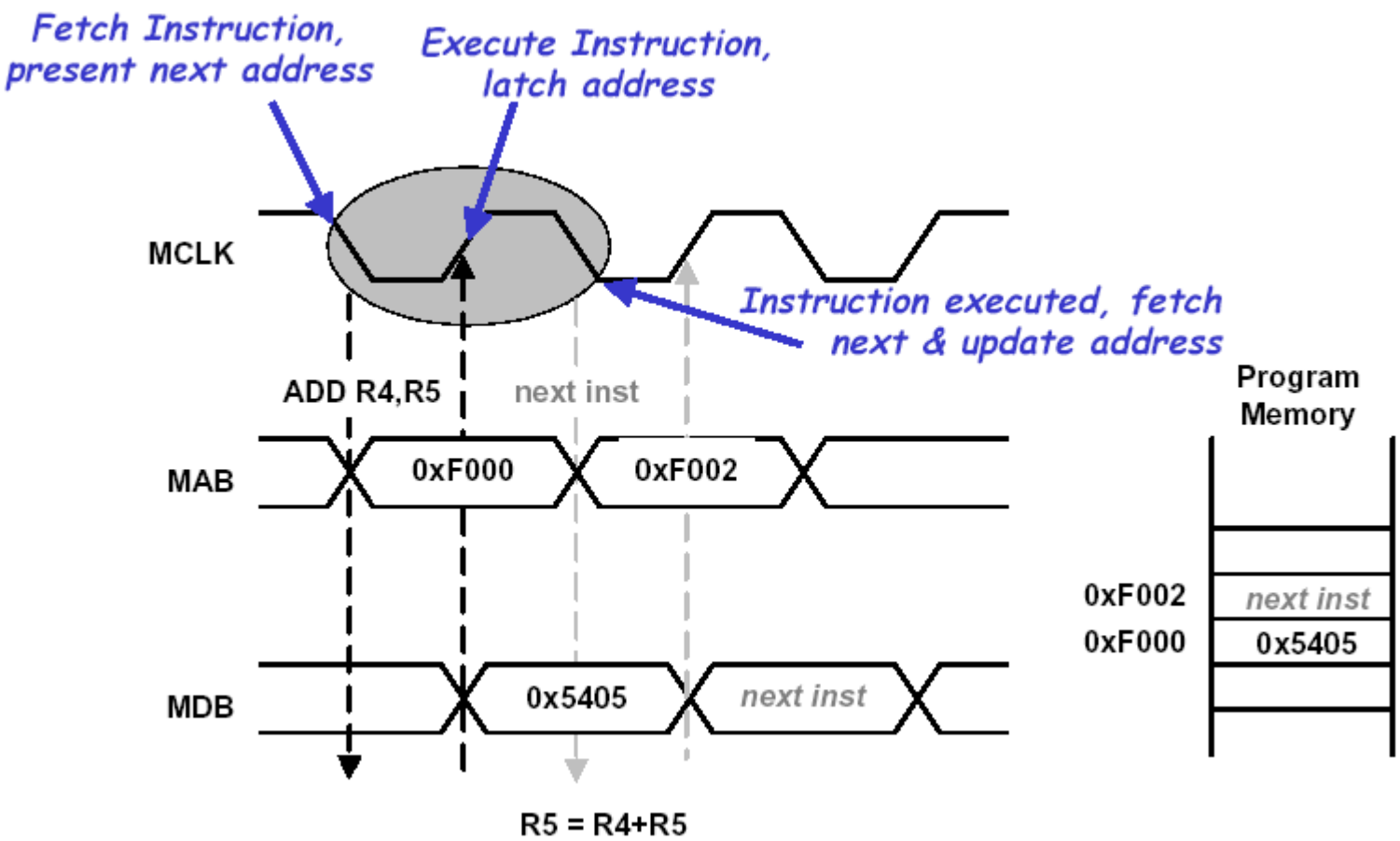
16-bit bus handles wide-width data much more effectively

The MSP430 Family – 16-Bit RISC

- Large 16-bit register file eliminates single accumulator bottleneck
- High-bandwidth 16-bit data and address bus with no paging
- RISC architecture with 27 instructions and 7 addressing modes
- Single-cycle register operations with full-access
- Direct memory-memory transfer designed for modern programming
- Compact silicon 30% smaller than an '8051 saves power and cost



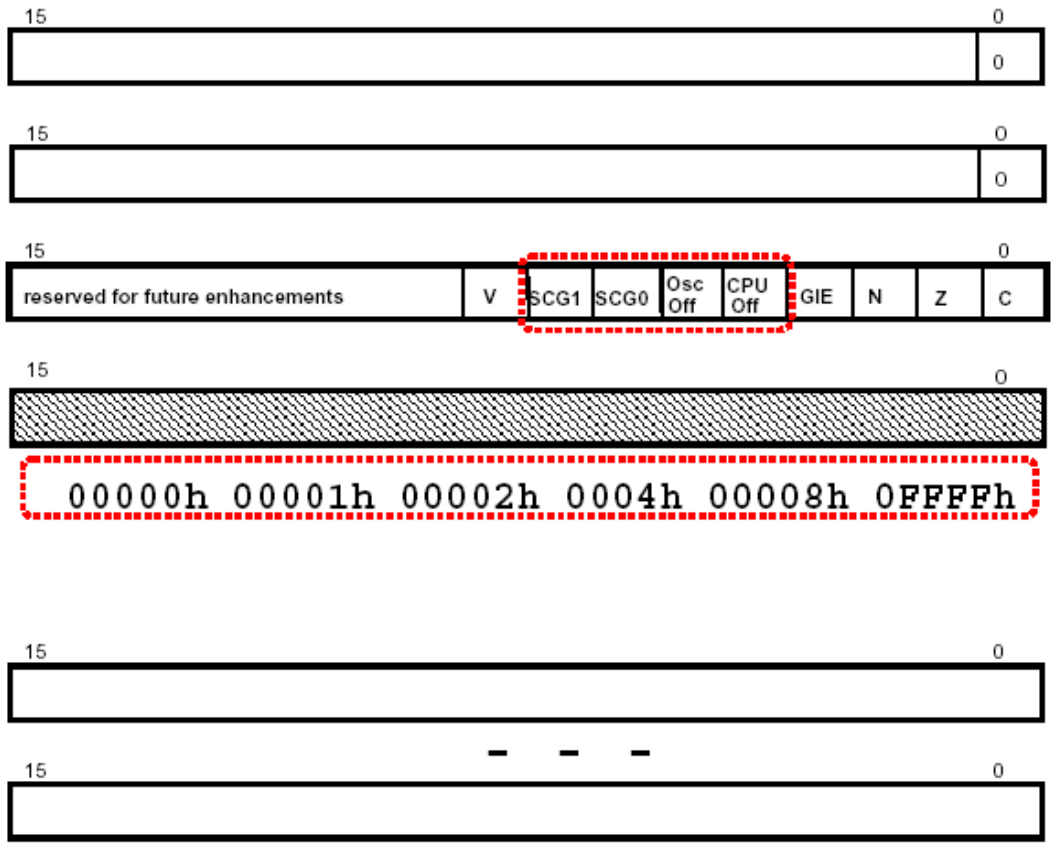
The MSP430 Family – Double Data Fetch Technology (DDFT)



The MSP430 Family – CPU Introduction

- RISC architecture with 27 instructions and 7 addressing modes.
- Orthogonal architecture with every instruction usable with every
- addressing mode.
- Full register access including program counter, status registers, and stack pointer.
- Single-cycle register operations.
- Large 16-bit register file reduces fetches to memory.
- 16-bit address bus allows direct access and branching throughout entire memory range.
- 16-bit data bus allows direct manipulation of word-wide arguments.
- Constant generator provides six most used immediate values and reduces code size.
- Direct memory-to-memory transfers without intermediate register holding.
- Word and byte addressing and instruction formats.

The MSP430 Family – CPU Registers



R0 - PC Program Counter
16-bit = no paging

R1 - SP Stack Pointer
Addressable = great "C" code

R2 - SR Status Register
Define LPMx

R3/R2 - CG Constant Generator
automatic generation of common used values reduces code size 30%

R4 - General Purpose

R15 - General Purpose

R4 through R15 are single-cycle, general purpose and identical in all respects - used for math, storage, and addressing modes.

The MSP430 Family – Registers: PC(R0)

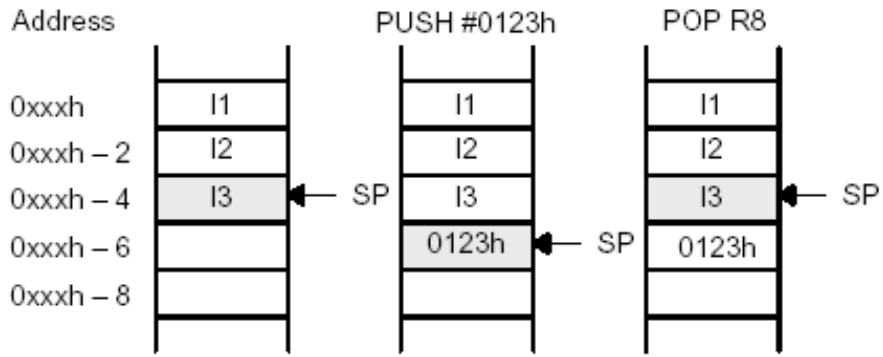
- Each instruction uses an even number of bytes (2, 4, or 6)
- PC is word aligned (the LSB is 0)

```
MOV #LABEL,PC ; Branch to address LABEL
MOV LABEL,PC ; Branch to address contained in LABEL
MOV @R14,PC ; Branch indirect, indirect R14
```

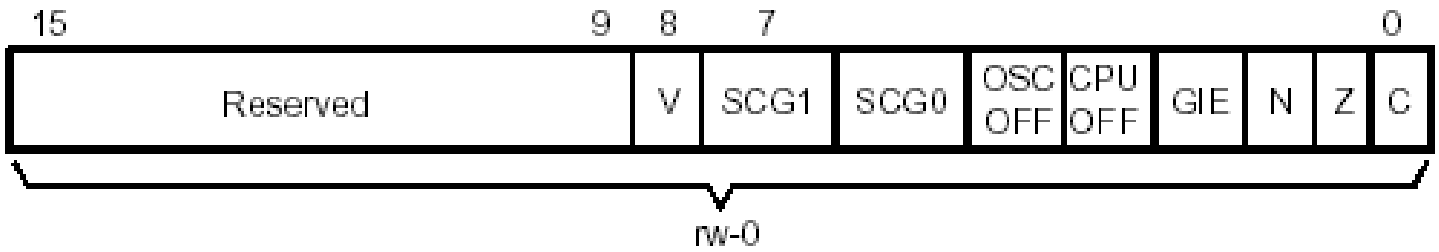
The MSP430 Family – Registers: SP(R1)

- Stack pointer for return addresses of subroutines and interrupts
- SP is word aligned (the LSB is 0)
- Pre-decrement/post-increment scheme

```
MOV 2(SP),R6 ; Item I2 -> R6
MOV R7,0(SP) ; Overwrite TOS with R7
PUSH #0123h ; Put 0123h onto TOS
POP R8 ; R8 = 0123h
```



The MSP430 Family – Registers: SR(R2)



- C: SR(0)
- Z: SR(1)
- N: SR(2)
- GIE (Global interrupt enable): SR(3)
- CPUOff: SR(4)
- OSCOff: SR(5)
- SCG1, SCG0: SR(7), SR(6)
- V: SR(8)

The MSP430 Family – Status Bits

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. This bit, when set, turns off the SMCLK.	
SCG0	System clock generator 0. This bit, when set, turns off the DCO dc generator, if DCOCLK is not used for MCLK or SMCLK.	
OSCOFF	Oscillator Off. This bit, when set, turns off the LFXT1 crystal oscillator, when LFXT1CLK is not use for MCLK or SMCLK	
CPUOFF	CPU off. This bit, when set, turns off the CPU.	
GIE	General interrupt enable. This bit, when set, enables maskable interrupts. When reset, all maskable interrupts are disabled.	
N	Negative bit. This bit is 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. This bit is set when the result of a byte or word operation is 0 and cleared when the result is not 0.	
C	Carry bit. This bit is set when the result of a byte or word operation produced a carry and cleared when no carry occurred.	

The MSP430 Family – Constant Generators

- As – source register addressing mode in the instruction word

Register	As	Constant	Remarks
R2	00	– – – – –	Register mode
R2	01	(0)	Absolute address 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

The MSP430 Family – CISC/RISC Instruction Set

- **Three instruction formats**
  - Source, destination
  - Destination
  - Jumping
- **Fifty-one instructions available in assembler**
  - 27 basic instructions ⇨ *RISC*
  - 24 emulated instructions ⇨ *CISC*
- **Seven addressing modes for source, four for destination**

	source	destination
Register Mode	✓	✓
Indexed Mode	✓	✓
Symbolic Mode	✓	✓
Absolute Mode	✓	✓
Indirect Mode	✓	
Indirect-autoincrement Mode	✓	
Immediate Mode	✓	

- **Bit, byte and word processing**

The MSP430 Family – Memory Map

- Special function registers
  - memory locations 0000h – 000Fh
  - 0000h, 0001h: interrupt enables
  - 0002h, 0003h: interrupt flags
  - 0004h, 0005h: module enable flags
- Peripheral registers
  - byte addressable: 0010h – 00FFh
  - word addressable: 0100h – 01FFh
- RAM: 0200h

The MSP430 Family – 27 Core RISC Instructions

Format I Source,Destination	Format II Single Operand	Format III +/- 9bit Offset
add(.b)	call	jmp
addc(.b)	swpb	jc
and(.b)	sxt	jnc
bic(.b)	push(.b)	jeq
bis(.b)	reti	jne
bit(.b)	rra(.b)	jge
cmp(.b)	rrc(.b)	jl
dadd(.b)		jn
mov(.b)		
sub(.b)		
subc(.b)		
xor(.b)		

The MSP430 Family – Emulated Instructions

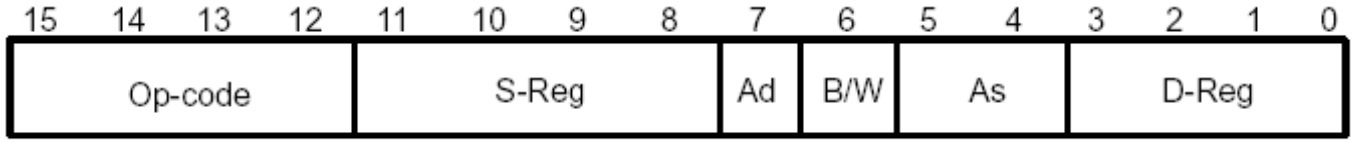
- Simply easier to understand with no code size or speed penalty
- Replaced by assembler with core instructions using *CG*, *PC* and *SP*

```
clrc                    ; Clear carry (emulated)
bic.w    #01h,SR        ; Core instruction

dec.w    R4             ; Decrement (emulated)
sub.w    #01,R4         ; Core instruction

ret                     ; Return (emulated)
mov.w    @SP+,PC        ; Core instruction
```

The MSP430 Family – Full Instruction Set

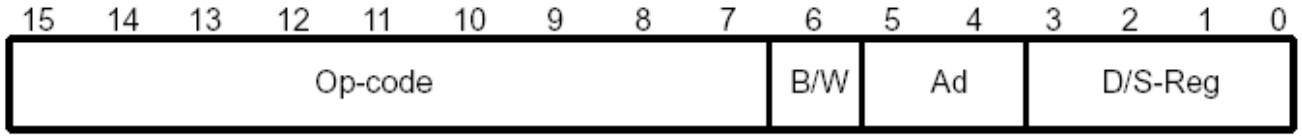
Format I Source,Destination	Format II Single Operand	Format III +/- 9bit Offset	Support
add(.b)	**br**	jmp	**clrc**
addc(.b)	call	jc	**setc**
and(.b)	swpb	jnc	**clrz**
bic(.b)	sxt	jeq	**setz**
bis(.b)	push(.b)	jne	**clrn**
bit(.b)	**pop.(b)**	jge	**setn**
cmp(.b)	rra(.b)	jl	**dint**
dadd(.b)	rrc(.b)	jn	**eint**
mov(.b)	**inv(.b)**		**nop**
sub(.b)	**inc(.b)**		**ret**
subc(.b)	**incd(.b)**		reti
xor(.b)	**dec(.b)**		
	decd(.b)		
	adc(b)		
	sbc(.b)		
	clr(.b)		
	dadc(.b)		
	rla(.b)		
	rlc(.b)		
	tst(.b)		

The MSP430 Family – Double Operand Instructions



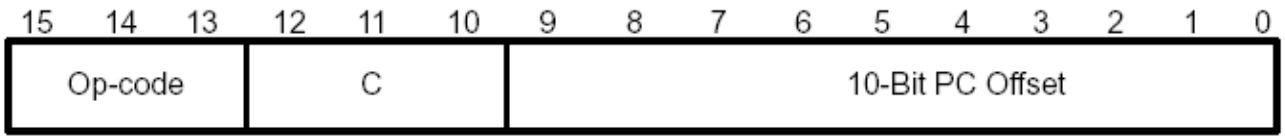
Mnemonic	S-Reg, D-Reg	Operation	Status Bits			
			V	N	Z	C
MOV(.B)	src,dst	src → dst	–	–	–	–
ADD(.B)	src,dst	src + dst → dst	*	*	*	*
ADDC(.B)	src,dst	src + dst + C → dst	*	*	*	*
SUB(.B)	src,dst	dst + .not.src + 1 → dst	*	*	*	*
SUBC(.B)	src,dst	dst + .not.src + C → dst	*	*	*	*
CMP(.B)	src,dst	dst – src	*	*	*	*
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 MSP430 Family – 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	*	*	*

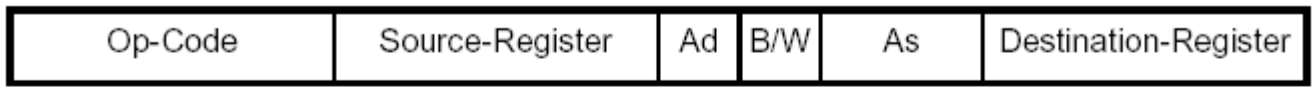
The MSP430 Family – Jump Instructions



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

The MSP430 Family – Instruction Formats

```
; Format I Source and Destination
```




```
5405            add.w   R4,R5                   ; R4+R5=R5 xxxx
5445            add.b   R4,R5                   ; R4+R5=R5 00xx
```

```
; Format II Destination Only
```


```
6404            rlc.w   R4                      ;
6444            rlc.b   R4                      ;
```

```
; Format III There are 8(Un)conditional Jumps
```


```
3c28            jmp     Loop_1                  ; Goto Loop_1
```

The MSP430 Family – 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.

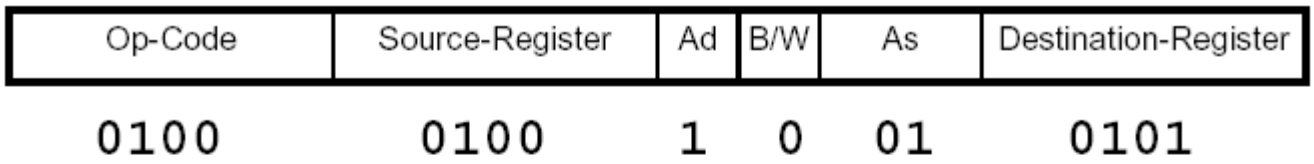
The MSP430 Family – Register Addressing

Op-Code	Source-Register	Ad	B/W	As	Destination-Register
0100	0100	0	0	00	0101

```
4405          mov.w    R4,R5          ;

4445          mov.b    R4,R5          ;
```

Valid for Source and destination As=00, Ad=0
The operand is contained in one of the CPU registers R0 to R15.
This is the fastest addressing mode and needs the least memory .

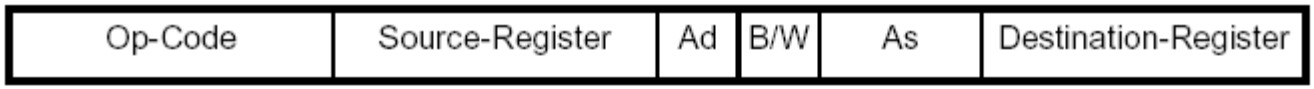
The MSP430 Family – Register-Indexed Addressing




```
4495 01000200   mov.w   100h(R4),200h(R5) ;

44150100        mov.w   100h(R4),R5       ;
```

Valid for Source and destination As=01, Ad=1
The address of the operand is the sum of the index and the contents of the register.

The MSP430 Family – Symbolic Addressing



0100 0000 1 0 01 0000

```
4090ffa80006   mov.w    EDE,TONI          ;

4015ffac       mov.w    EDE,R5            ;
```

Source and destination As=01, Ad=1
The content of the addresses EDE / TONI are used for the operation. The source or destination address is computed as a difference from the PC and uses the PC in indexed addressing mode. Any address in the 64k memory space is addressable.

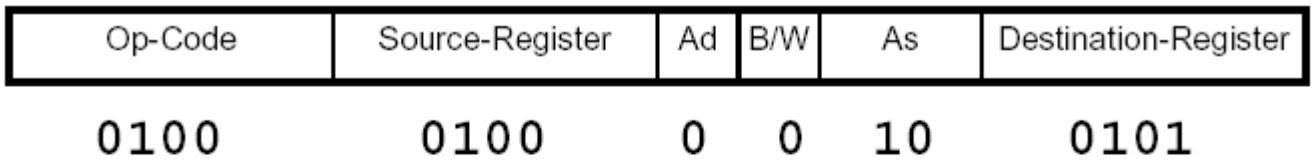
The MSP430 Family – Absolute Addressing

Op-Code	Source-Register	Ad	B/W	As	Destination-Register
0100	0010	1	0	01	0010

```
4292201720174   mov.w    &CCR0,&CCR1          ;

42150172        mov.w    &CCR0,R5             ;
```

Source and destination As=01, Ad=1
The contents of the fixed addresses are used for the operation. The SR is used in the indexed mode to create an absolute 0. Use for hardware peripherals located at an absolute address that can never be relocated.

The MSP430 Family – Register-Indirect Addressing




```
4425          mov.w    @R4,R5          ;

4465          mov.b    @R4,R5          ;
```

Source only As=10, Ad=n/a
The registers are used as a pointer to the operand.
The indexed mode with zero index may be used for "indirect register addressing" of the destination operand.

```
44a50000      mov.w    @R4,0(R5)       ;
```

The MSP430 Family – Register Indirect Autoincrement Addressing

Op-Code	Source-Register	Ad	B/W	As	Destination-Register
0100	0100	0	0	11	0101

```
4435          mov.w    @R4+,R5              ;

4475          mov.b    @R4+,R5              ;
```

Source only As=11, Ad=n/a
The registers are used as a pointer to the operand. The registers are incremented afterwards - by 1 in byte mode, by 2 in word mode.

The MSP430 Family – Immediate Addressing

Op-Code	Source-Register	Ad	B/W	As	Destination-Register
0100	0000	0	0	11	0101

```
40351234        mov.w   #1234h,R5           ; Any 16-bit value
```

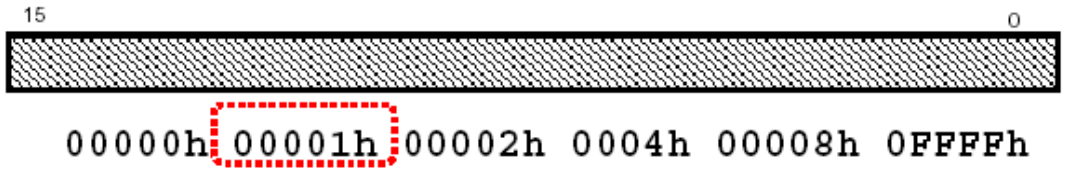
Source only As=11, Ad=n/a

Any immediate 8 or 16 bit constant can be used with the instruction. The PC is used in autoincrement mode to emulate this addressing mode.

The MSP430 Family – Code Reduction Effect of Constant Generator

Op-Code	Source-Register	Ad	B/W	As	Destination-Register
0100	0011	0	0	01	0100

```
4314          mov.w   #0001h,R4               ;
```



R3/R2 - *CG Constant Generator*

automatic generation of commonly used values reduces code size 30%

The MSP430 Family – Machine Cycles for Format I Instructions

Address Mode As	Address Mode Ad	#-of-Cycles	Length of Instruction [words]	Examples
00, Rn	0, Rm	1	1	MOV R5,R8
	0, PC	2	1	BR R9
00, Rn	1, x(Rm)	4	2	ADD R5,2(R6)
	1, EDE			XOR R8,EDE
	1,&EDE			MOV R5,&EDE
01,x(Rn)	0, Rm	3	2	MOV 2(R5),R7
01, EDE				AND EDE,R6
01,x(Rn)	1,x(Rm)	6	3	ADD 4(R4),6(R9)
01,EDE	1,TONI			CMP EDE,TONI
01,&EDE	1,&EDE			MOV R5,&EDE
10,@Rn	0, Rm	2	1	AND @R4,R5
10,@Rn	1,x(Rm)	5	2	XOR @R5,8(R6)
	1, EDE			MOV @R5,EDE
	1,&EDE			XOR @R5,&EDE
11,@Rn+	0,Rm	2	1	ADD @R5+,R6
	0, PC	3		BR @R5+
11,#N	0,Rm	2	2	MOV #20,R9
	0,PC	3		BR #2AEh
11,@Rn+	1,x(Rm)	5	2	MOV @R9+,2(R4)
11,#N	1,EDE		3	ADD #33,EDE
11,@Rn+	1,&EDE		2	MOV @R9+,&EDE
11,#N			3	ADD #33,&EDE

The MSP430 Family – Machine Cycles for Format II/III Instructions

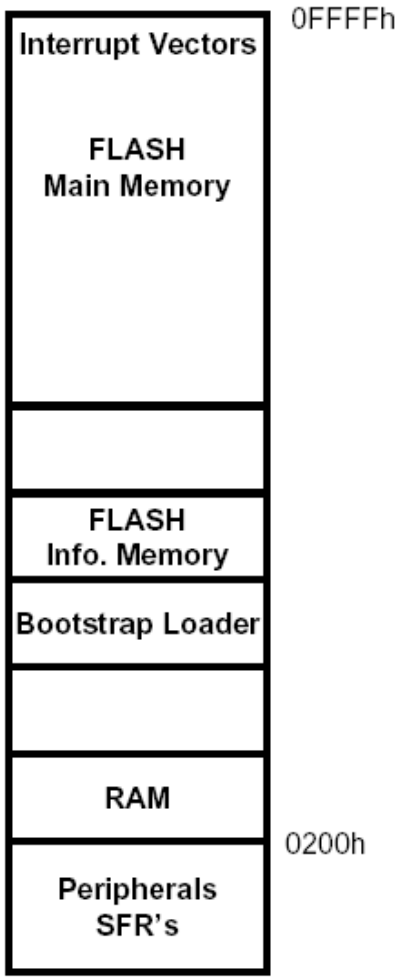
Address Mode As/Ad	#-of-Cycles		Length of Instruction [words]	Examples
	RRA RRC SWPB SXT	PUSH/ CALL		
00, Rn	1	3/4	1	SWPB R5
01, x(Rn) 01, EDE	4	5/5	2	CALL Table(R7) PUSH EDE
10, @Rn	3	4/4	1	RRC @R9
11, @Rn+ 11, #N	3	4/5	1 2	SWPB @R10+ CALL 2(R7)

Machine Cycles for Format III Instructions

All Jxx - instructions need the same #-of-cycles independent of executing a Jump

- Clock Cycles: 2
- Length of Instruction: 1 word

The MSP430 Family – MSP430 Memory Model



- Unified 64kB continuous memory map
- Same instructions for data and peripherals
- Program and data in Flash or RAM with no restrictions
- Easy to understand with no paging
- Designed for modern programming techniques such as pointers and fast look-up tables

The MSP430 Family – Memory Organization