MSP430 Teaching Materials

Introductory Overview
Week2

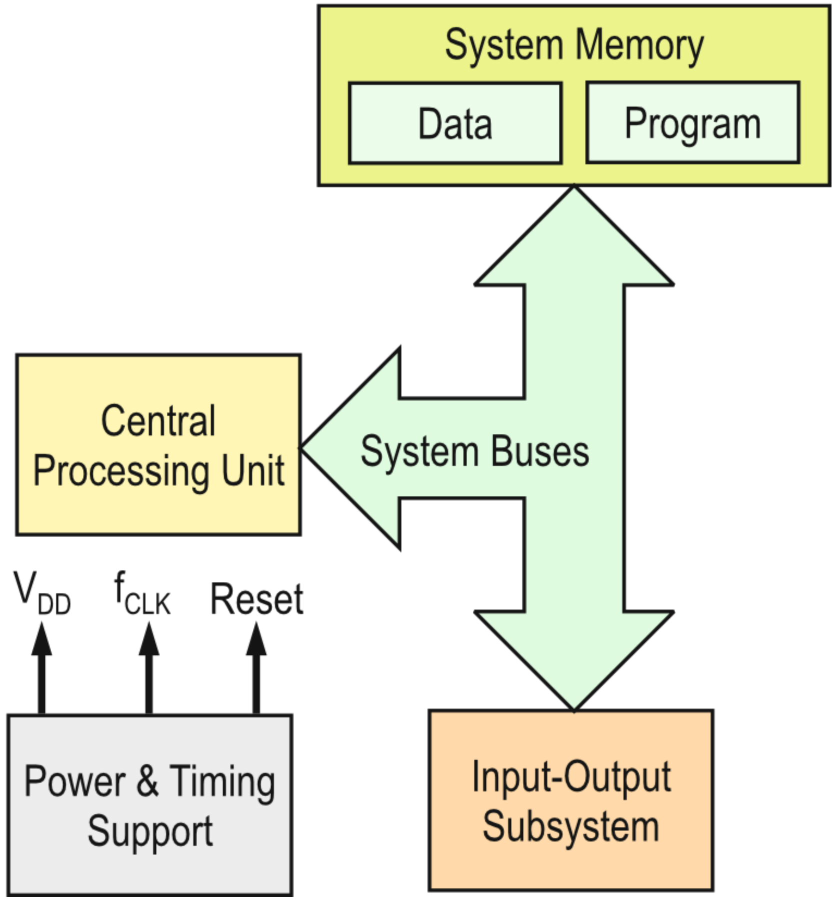
Hacettepe University

Outline

- ❑ Base Microcomputer Structure
- ❑ Microcontrollers Versus Microprocessors
- ❑ Central Processing Unit
- ❑ System Buses
- ❑ Memory Organization
- ❑ I/O Subsystem Organization

Fig. 3.1 General architecture of a microcomputer system

- Central Processing Unit
- System Memory
- Program Memory
- Data Memory
- Input-Output Subsystem
- System Buses
- Address bus
- Data bus
- Control bus
- Power & Support Logic



Microcontrollers vs Microprocessors

- Abbreviated as MPUs
- Contain a General Purpose CPU
  - ALU, CU, Registers, & BIU
- Require External Components to form a basic system
  - Buses
  - Memory
  - I/O interfaces & devices
- Additional Characteristics
  - Architecture optimized for accelerating data processing
  - Include elements to accelerate instruction execution ?

Microcontrollers vs Microprocessors

- Abbreviated as MCUs
- Contain a CPU
- Usually less complex than that of an MPU
- Include memory and peripherals in a single chip
- Denominated computer-on-a-chip
- Most MCUs do not provide external buses
- On-chip Memory
- Includes both program and data memory
- Typical Peripherals
- Timers
- I/O ports
- Data converters

Microcontrollers vs Microprocessors

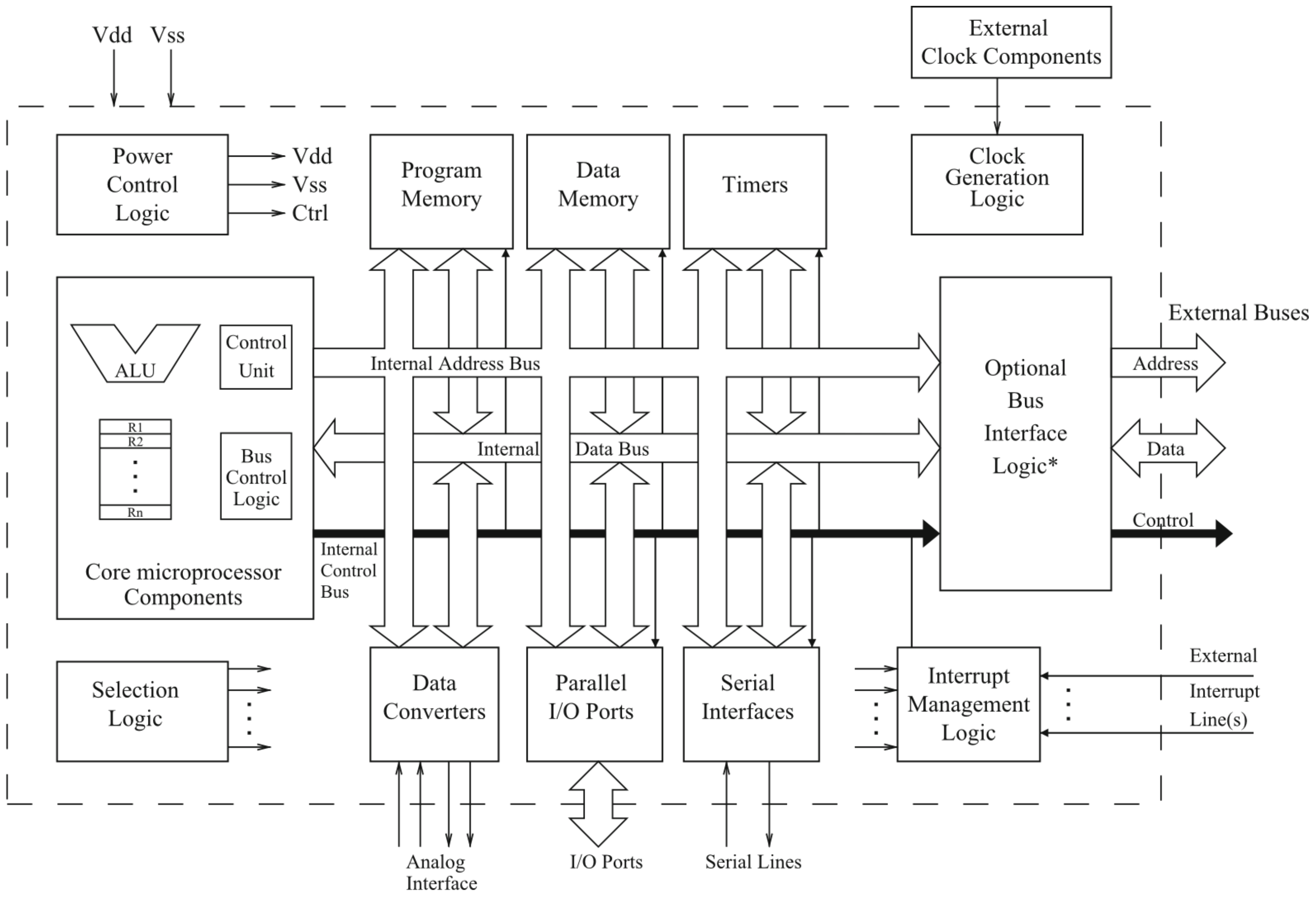


Fig. 3.2 Structure of a typical microcontroller

Table 3.1 A sample of MCU families/series

Company	4-bits	8-bits	16-bits	32-bits
EM Microelectronic	EM6807	EM6819		
Samsung	S3P7xx	S3F9xxx	S3FCxx	S3FN23BXZZ
Freescale semiconductor		68HC11	68HC12	
Toshiba		TLCS-870	TLCS-900/L1	TLCS-900/H1
Texas instruments			MSP430	TMS320C28X
			TMS320C24X	Stellaris line
Microchip		PIC1X	PIC2x	PIC32

RISC VS CISC

- CISC (Complex Instruction Set Computer)
- Variable length instructions
- Large instruction set
- Focuses in accomplishing as much as possible with each instruction
    - Helps programmer's tasks
    - Augments hardware complexity
- RISC (Reduced Instruction Set Computer)
- Fixed length instructions
- Short (reduced) instruction set
- Focuses on simple instructions
    - Makes programming harder
    - Simplifies the hardware structure

PROGRAMMER'S AND HW MODEL (1/2)

- General-purpose, High-level Programming
  - Do not require knowledge of the underlying hardware
  - Centered on compiler-level abstraction
- Embedded Systems Programming
  - Need to consider both, hardware and software models
    - Requires awareness of underlying hardware infrastructure
    - Needs to use software programmer's model
  - Might require some assembly-level programming
    - Allows mixed assembly- C-language programming
    - Other languages might be available

PROGRAMMER'S AND HW MODEL (2/2)

- Hardware Model
  - Knowledge of hardware characteristics is indispensable
    - Structure, protocols, timing, power, limitations
  - Supports the programmer's model
- Programmer's Model Focus
  - Instruction set
  - Syntax
  - Addressing modes
  - Memory map
  - Transfers and execution time

CPU COMPONENTS

- Hardware
  - Control unit (CU)
  - Arithmetic Logic Unit (ALU)
  - Register set
  - Bus interface logic (BIL)
- Software
  - Instruction set
  - Addressing modes

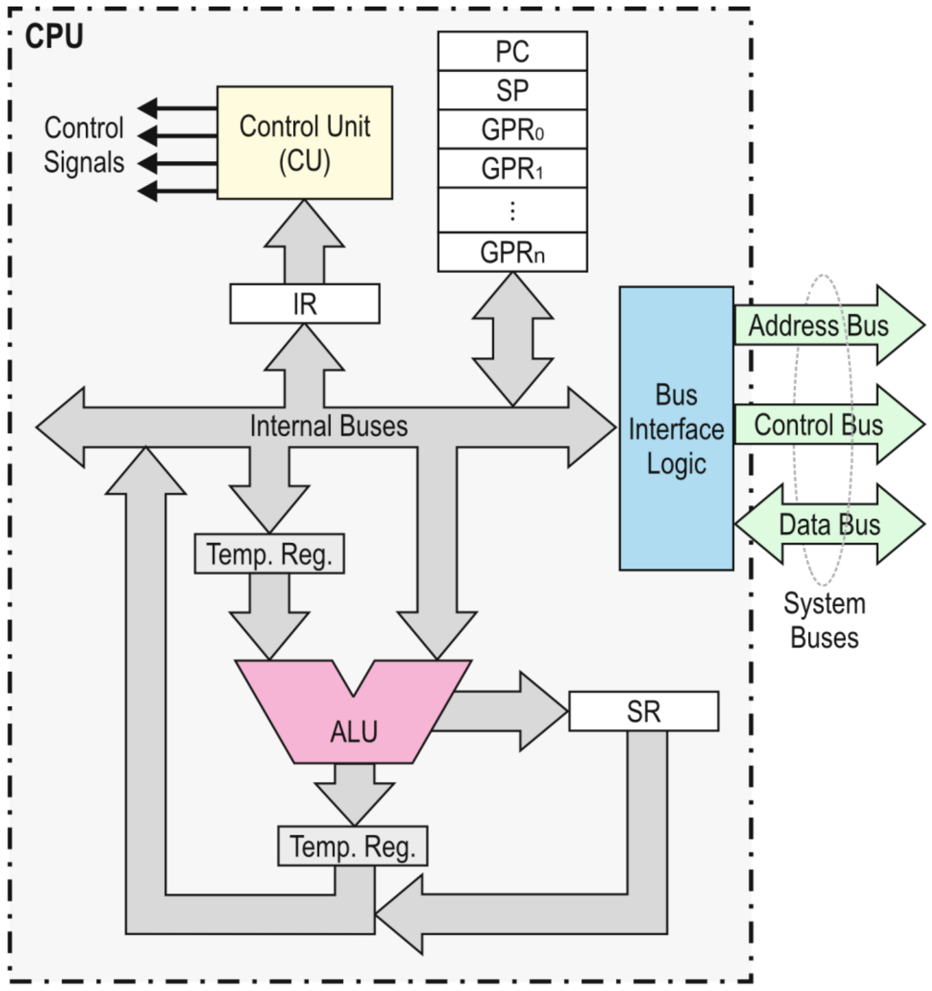
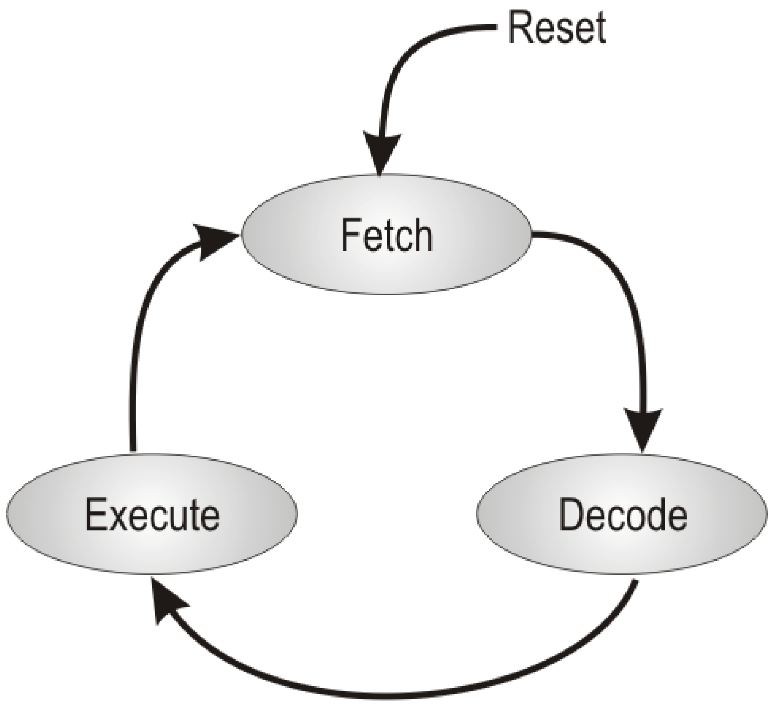


Fig. 3.3 Minimal architectural components in a simple CPU

CONTROL UNIT

- **Operation**
- **Governs the CPU working like a finite state machine**
- **Cycles forever through three states:**
    - **Fetch**
    - **Decode, and**
    - **Execute**
- **The fetch-decode-execute cycle is also known as the instruction cycle or CPU cycle**



CONTROL UNIT

- **Fetch**
  - **The program counter (PC) provides the address of the instruction to be fetched from memory**
  - **The instruction pointed by the PC is brought from memory into the CPU's instruction register (IR)**
- **Decode**
  - **The instruction meaning is deciphered**
  - **The decoded information is used to send signals to the appropriate CPU components to execute the instruction**
- **Execute**
  - **The CU commands the corresponding functional units to perform the actions specified by the instruction**
  - **At the end of the execution phase, the PC has been incremented to point to the address of the next instruction in memory**

ARITHMETIC LOGIC UNIT

- Performs Supported Logic and Arithmetic Operations
  - Logic: AND, OR, NOT, XOR, SHIFT, ROTATE
  - Arithmetic: ADD, SUB, CMP
- Datapath Width
  - Established by the ALU operand width
  - Also sets the width of the data bus and data registers
- Example: A 16-bit CPU
  - 16-bit ALU operands
  - 16-bit Data bus
  - 16-bit Registers

BUS INTERFACE UNIT

- Coordinates the interaction between the internal buses and the system buses, if externally accessible
- Defines how the external address, data, and control buses operate
- Present even in MCUs with no external buses to coordinate internal bus activity
- Transparent to the programmer

SYSTEM BUS

- Data bus
  - Set of lines carrying data and instructions
  - Data bus lines are bidirectional to allow for reads & writes
  - READ: A transfer into a CPU register from memory or I/O
  - WRITE: A transfer or from a CPU register to memory or I/O
- Address bus
  - Set of lines transporting the address information which uniquely identifies a data cell in memory or peripheral device
- Control bus
  - Set of lines carrying the signals that regulate the system activity

REGISTERS

- Provide temporary storage for:
  - Data & operands
  - Memory addresses
  - Control words
- Fastest form of storage
- Smallest Capacity
- Volatile Contents
  - Contents lost when CPU is de-energized
- Register Types
  - General Purpose
  - Special Purpose

GP REGISTERS

- Are not tied to specific functions
  - Are available for programmer’s general usage
- Can hold data, variables, or addresses
  - Usage depend on addressing mode and programmer’s designation
- Number of registers depend on CPU architecture
  - Accumulator architectures have only a few
  - Some as little as two GP registers
  - RISC CPUs use a register file with dozens of registers

SP REGISTERS

- **Instruction Register (IR)**

Holds the instruction being currently decoded and executed

- **Program Counter (PC)**
  - **Holds the address of the next instruction to be fetched from memory**
- **Stack Pointer (SP)**
  - **Holds the address of the current top-of-stack (TOS)**
- **Status Register (SR)**
  - **Holds the current CPU status**
  - **Status is indicated by a set of *flags***
  - **A Flag: an individual bit indicating some condition**

Flag REGISTERS

- **Zero Flag (ZF)**
  - Set when the result of an ALU operation is zero, and cleared otherwise
- ***Carry Flag (CF)***
  - Set when an ALU arithmetic operation produces a carry
- ***Negative*** **or** ***sign flag*** **(*NF*)**
  - Set if the result of an ALU operation is negative and cleared otherwise
- ***Overflow Flag (VF)***
  - This flag signals overflow in ALU addition or subtraction operations with signed numbers
- ***Interrupt Flag (IF)***
  - Also called the *Global Interrupt Enable* (GIE)
  - Is not associated to the ALU status
  - Indicates whether or not the CPU would accept interrupt

Example 3.1 *The following operations are additions performed by the ALU using 8-bit data. For each one, determine the Carry, Zero, Negative, and Overflow flags.*

$$\begin{array}{lcccccc}
\ 01001010+ & \quad & \ 10110100+ & \quad & \ 10011010+ & \quad & \ 11001010+ \\
\ 01111001= & & \ 01001100= & & \ 10111001= & & \ 00011011= \\
\hline
0\ 11000011 & & 1\ 00000000 & & 1\ 01010011 & & 0\ 11100101 \\
\uparrow\ \uparrow & & \uparrow\ \uparrow & & \uparrow\ \uparrow & & \uparrow\ \uparrow \\
\mathrm{C}\ \mathrm{N} & & \mathrm{C}\ \mathrm{N} & & \mathrm{C}\ \mathrm{N} & & \mathrm{C}\ \mathrm{N}
\end{array}$$

Solution: *The operands have eight bits, so this length is our reference for the flags when we look at the result. The most significant bit in this group is flag* N. *The bit to the left is* C. *In hex form, these additions are, respectively,* 4Ah + 79h = C3h*;* B4h + 4Ch = 100h*;* 9Ah + B9h = 153h*; and* CAh + 1Bh = E5h. *The zero flag is set if the result is 0, discarding the carry, and the overflow flag is set if the addition of numbers of the same sign (that is, with equal most significant bit) yield a result of different sign (signaled by N). With this information we have then:*

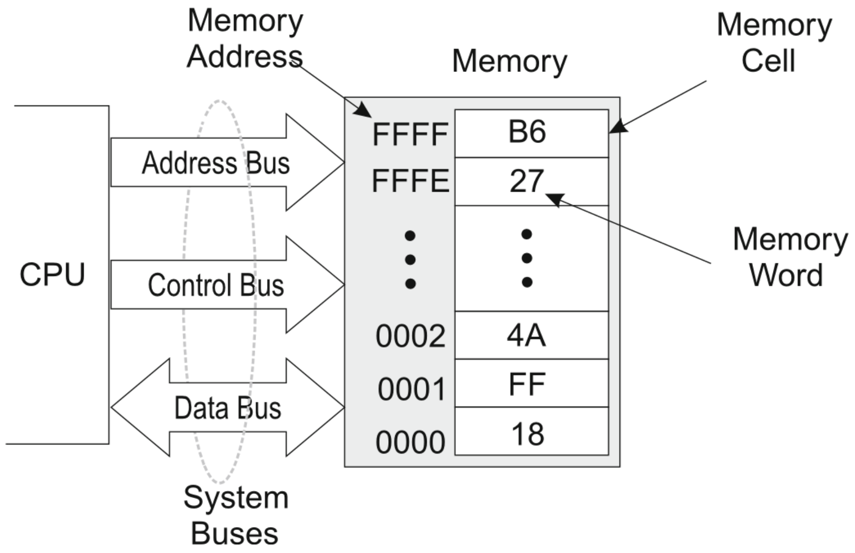
Operation 4Ah + 79h = C3h : $\mathrm{C} = 0,\ \mathrm{N} = 1,\ \mathrm{Z} = 0$ *and* $\mathrm{V} = 1$.
Operation B4h + 4Ch = 100h : $\mathrm{C} = 1,\ \mathrm{N} = 0,\ \mathrm{Z} = 1$ *and* $\mathrm{V} = 0$.
Operation 9Ah + B9h = 153h : $\mathrm{C} = 1,\ \mathrm{N} = 0,\ \mathrm{Z} = 0$ *and* $\mathrm{V} = 1$.
Operation CAh + 1Bh = E5h : $\mathrm{C} = 0,\ \mathrm{N} = 1,\ \mathrm{Z} = 0$ *and* $\mathrm{V} = 0$.

MEMORY ORGANIZATION

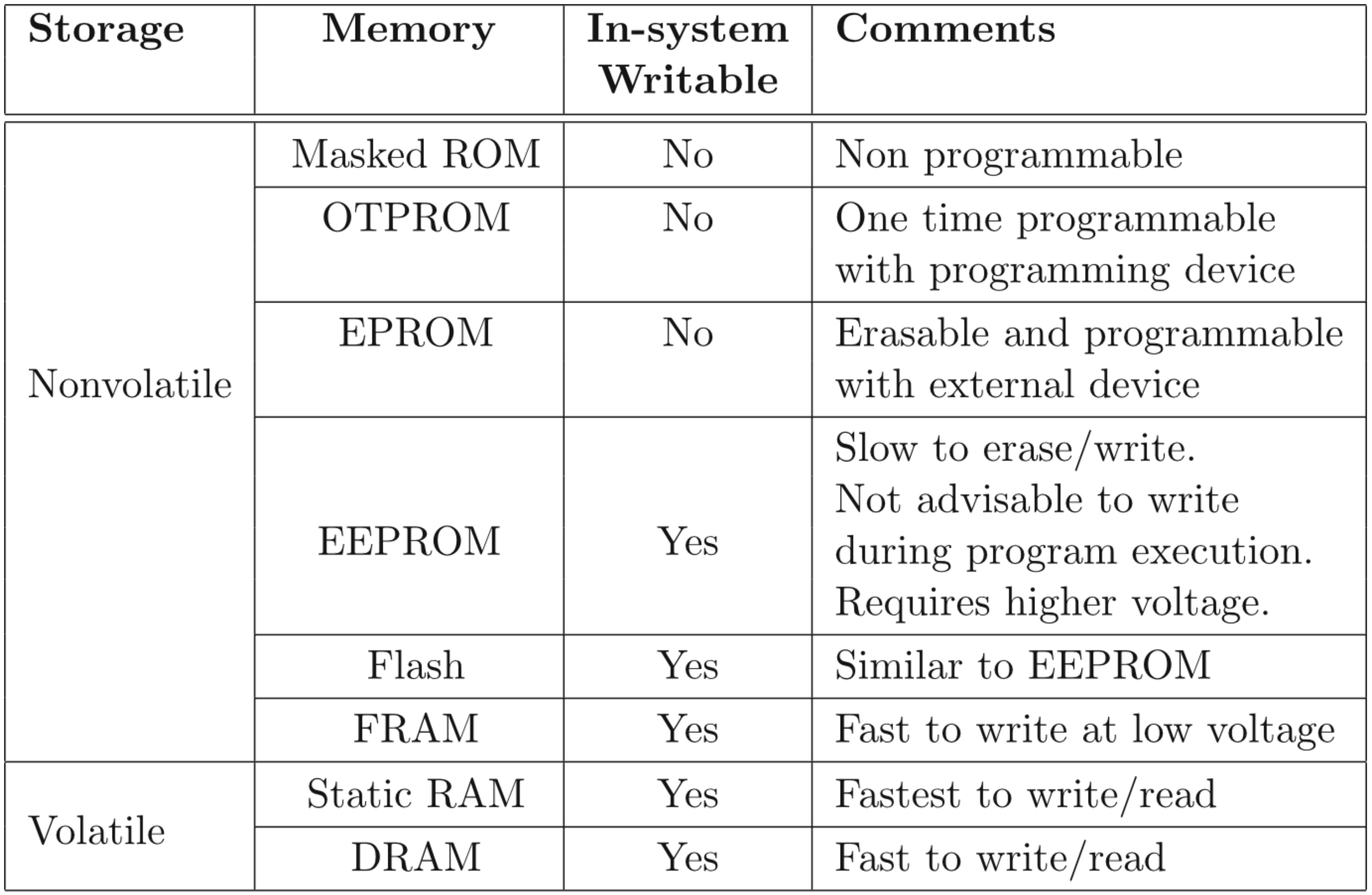
- Data Address: Little and Big Endian Conventions
- Program and Data Memory
- Von Neumann and Harvard Architectures
- Memory and CPU Data Exchange: An Example
- Memory Map
- MSP430 Memory Organization

MEMORY ORGANIZATION

- The memory subsystem is organized as an array of cells or locations
- Each memory location is identified by an address
- The contents of a memory cell is a word
  - A memory word may store instructions or data



MEMORY TYPES



Storage	Memory	In-system Writable	Comments
Nonvolatile	Masked ROM	No	Non programmable
	OTPROM	No	One time programmable with programming device
	EPROM	No	Erasable and programmable with external device
	EEPROM	Yes	Slow to erase/write. Not advisable to write during program execution. Requires higher voltage.
	Flash	Yes	Similar to EEPROM
	FRAM	Yes	Fast to write at low voltage
Volatile	Static RAM	Yes	Fastest to write/read
	DRAM	Yes	Fast to write/read

Fig. 3.7 Memory types

Memory

- **Volatile:** Loses its contents when power is removed. RAM
- **Nonvolatile:** Retains its contents when power is removed and is therefore used for the program and constant data. Usually slower than writing to RAM
  - **Masked ROM:** The data are encoded into one of the masks used for photolithography and written into the IC during manufacture. This memory really is read-only. Used for the high-volume production of stable products, because any change to the data requires a new mask to be produced at great expense. Some MSP430 devices can be ordered with ROM, shown by a *C* in their part number. For ex: MSP430CG4619.
  - **EPROM (electrically programmable ROM):** As its name implies, it can be programmed electrically but not erased. Devices must be exposed to ultraviolet (UV) light for about ten minutes to erase them. The usual black epoxy encapsulation is opaque, so erasable devices need special packages with quartz windows, which are expensive. These were widely used for development before flash memory was widely available.
  - **OTP (one-time programmable memory):** This is just EPROM in a normal package without a window, which means that it cannot be erased. Devices with OTP ROM are still widely used and the first family of the MSP430 used this technology.

Memory

- **Flash memory:** This can be both programmed and erased electrically and is now by far the most common type of memory. It has largely superseded electrically erasable, programmable ROM (EEPROM). The practical difference is that individual bytes of EEPROM can be erased but flash can be erased only in blocks. Most MSP430 devices use flash memory, shown by an *F* in the part number

EXAMPLE MEMORY BANK CONNECTION

MEMORY BANK: Memory blocks of one byte are called bank
MEM SEGMENT: A set of memory words with continuous addressing

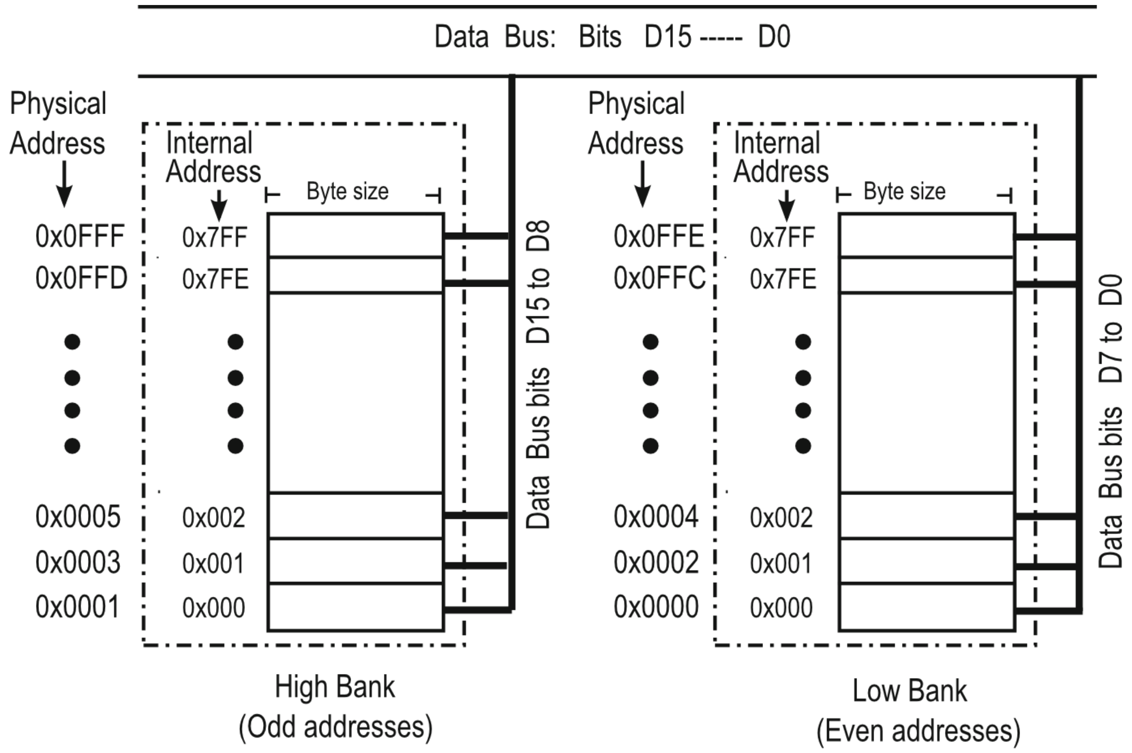
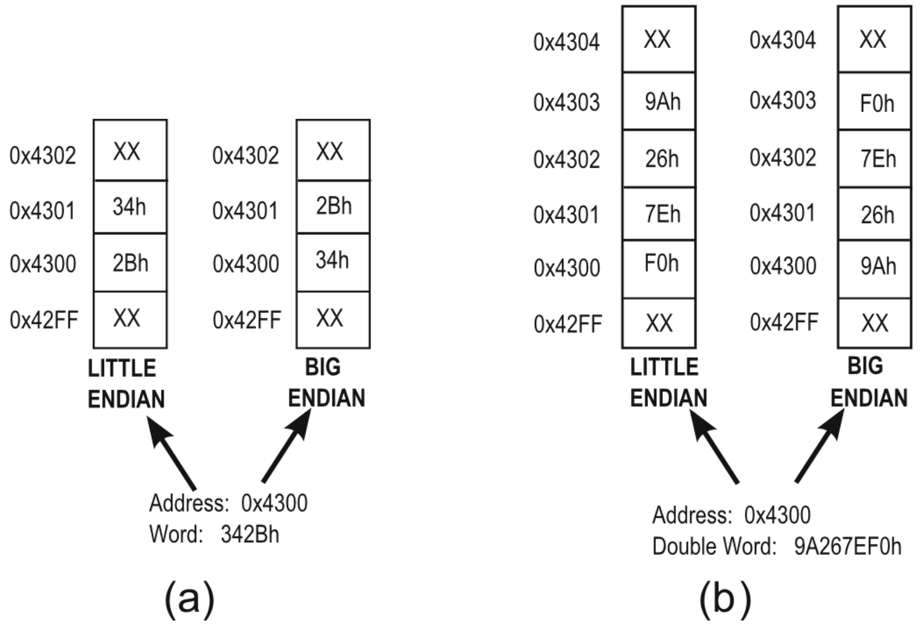


Fig. 3.8 Example of bank connection

40BF003A0120 has the address F8AAh ?

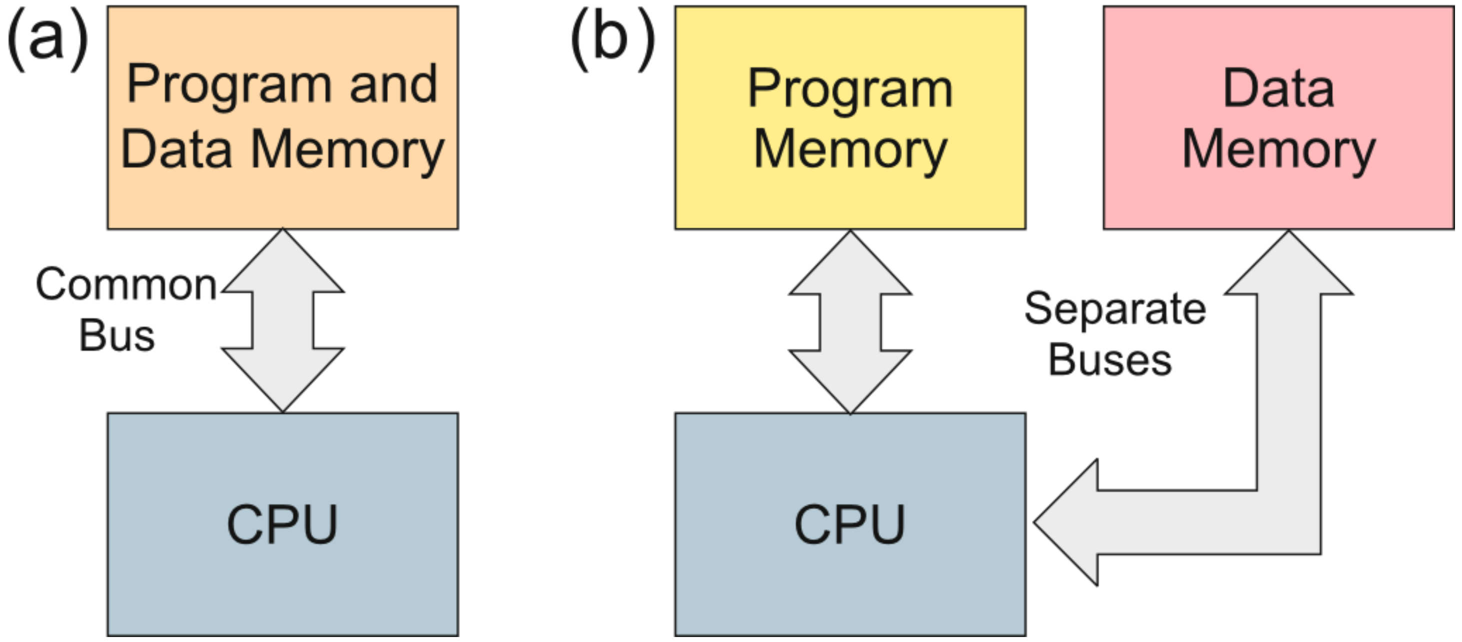
ENDIANNESS

- **Big Endian:**
  - **Data is stored with the most significant byte in the lowest address and the least significant byte in the highest address**
- **Little Endian**
  - **Data is stored the the least significant byte in the lowest address and the most significant byte in the highest address**



MCU ARCHITECTURES

- Von Neumann
  - A single set of buses for accessing both programs and data and a single address space
- Harvard
  - Uses separate buses for accessing programs and data and has separate address spaces



Harvard and von Neumann Architectures

- The volatile (data) and nonvolatile (program) memories are treated as separate systems, each with its own address and data bus.
  - **Simulataneous Access!**
  - **Individually Optimized.**
  - **Microchip PICs, the Intel 8051 and the ARM9.**
- Von Neumann
  - Easier architecture.
  - MSP430, Freescale HCS08, and the ARM7.

CONCEPTUAL MEMORY CONNECTION

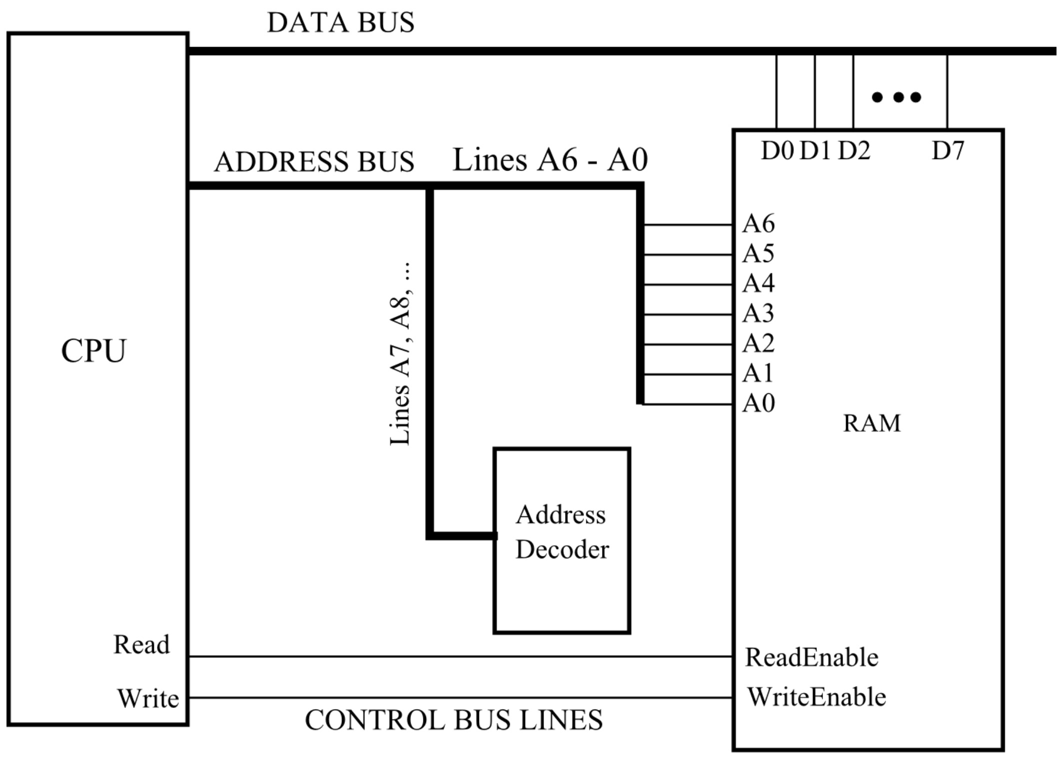


Fig. 3.11 CPU to memory connection: a conceptual scheme

256B Memory Example

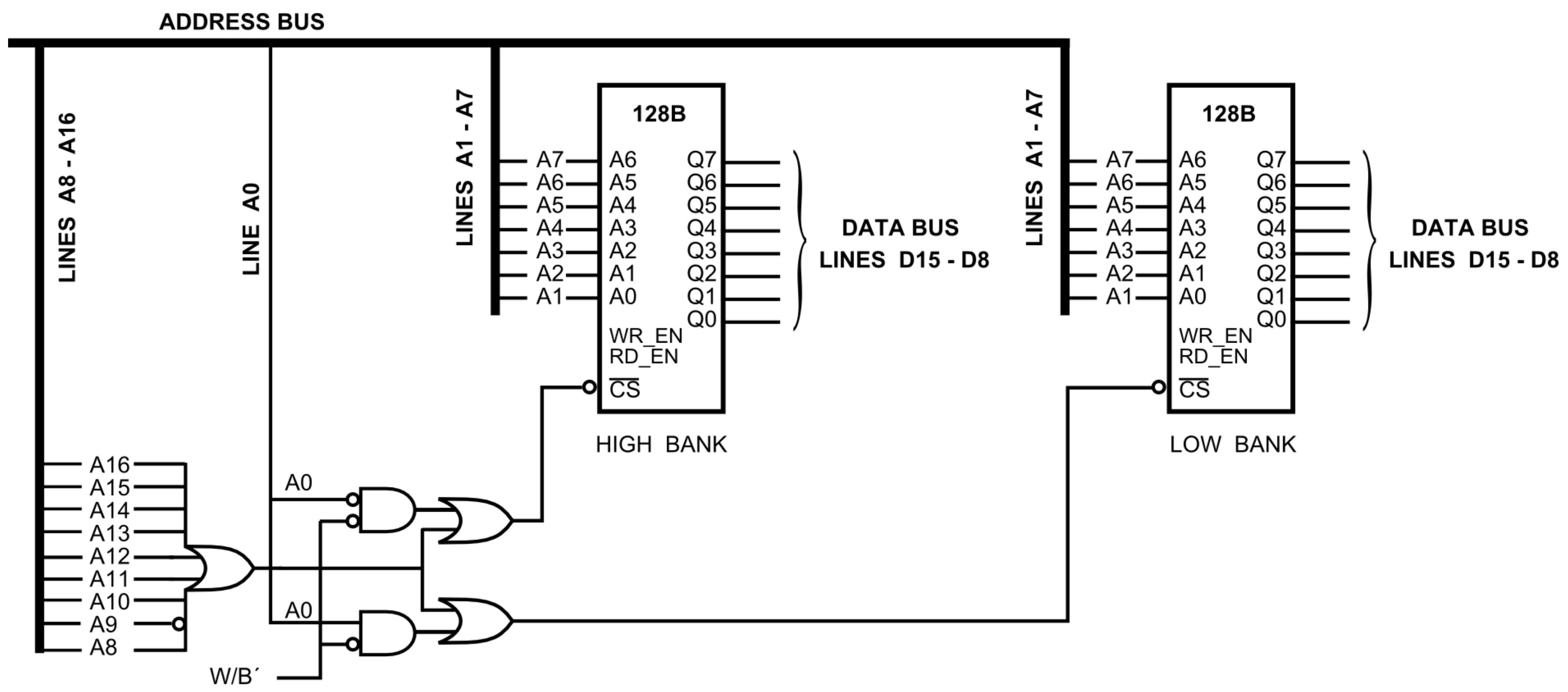


Fig. 3.12 256B memory bank example

Memory Map Example

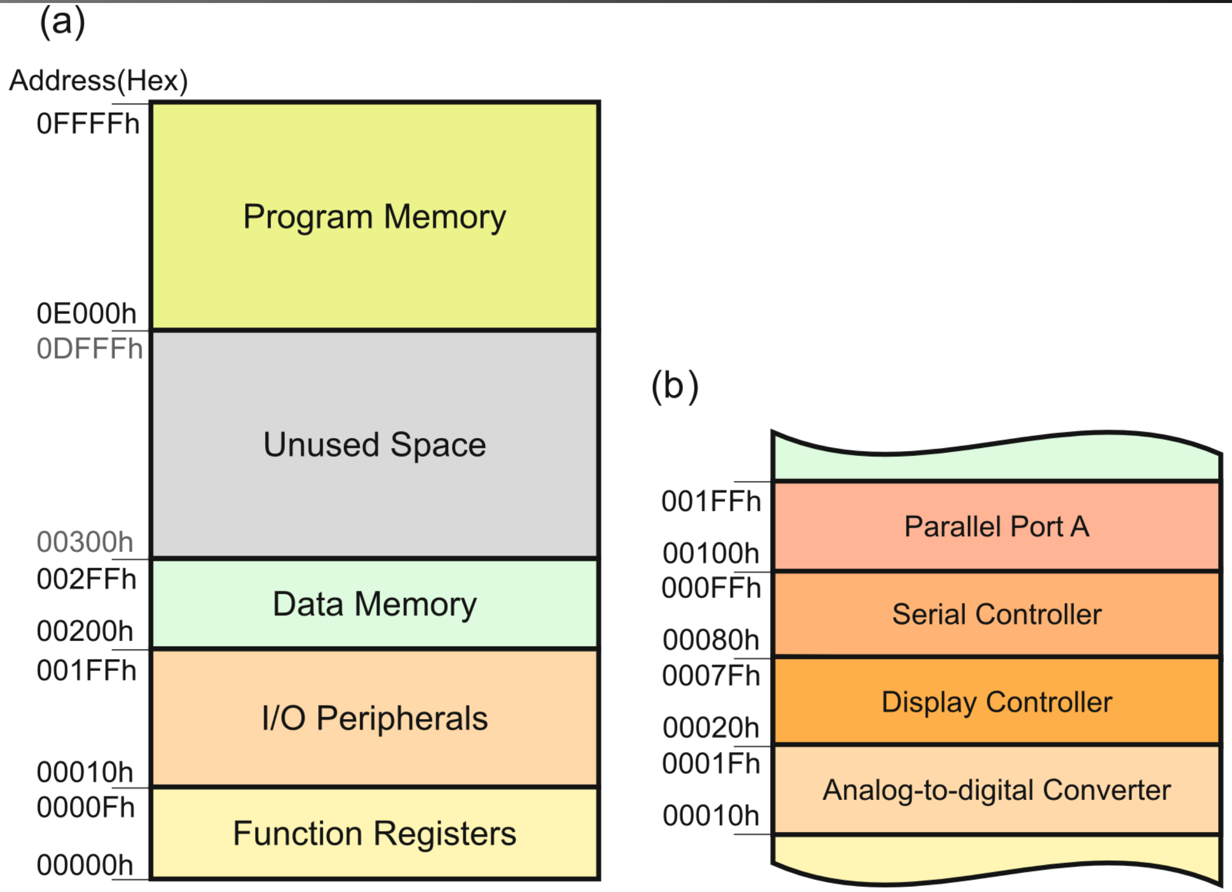


Fig. 3.13 Example memory map for a microcomputer with a 16-bit address bus. **a** Global memory gap, **b** Partial memory map

MSP430 MEMORY ORGANIZATION

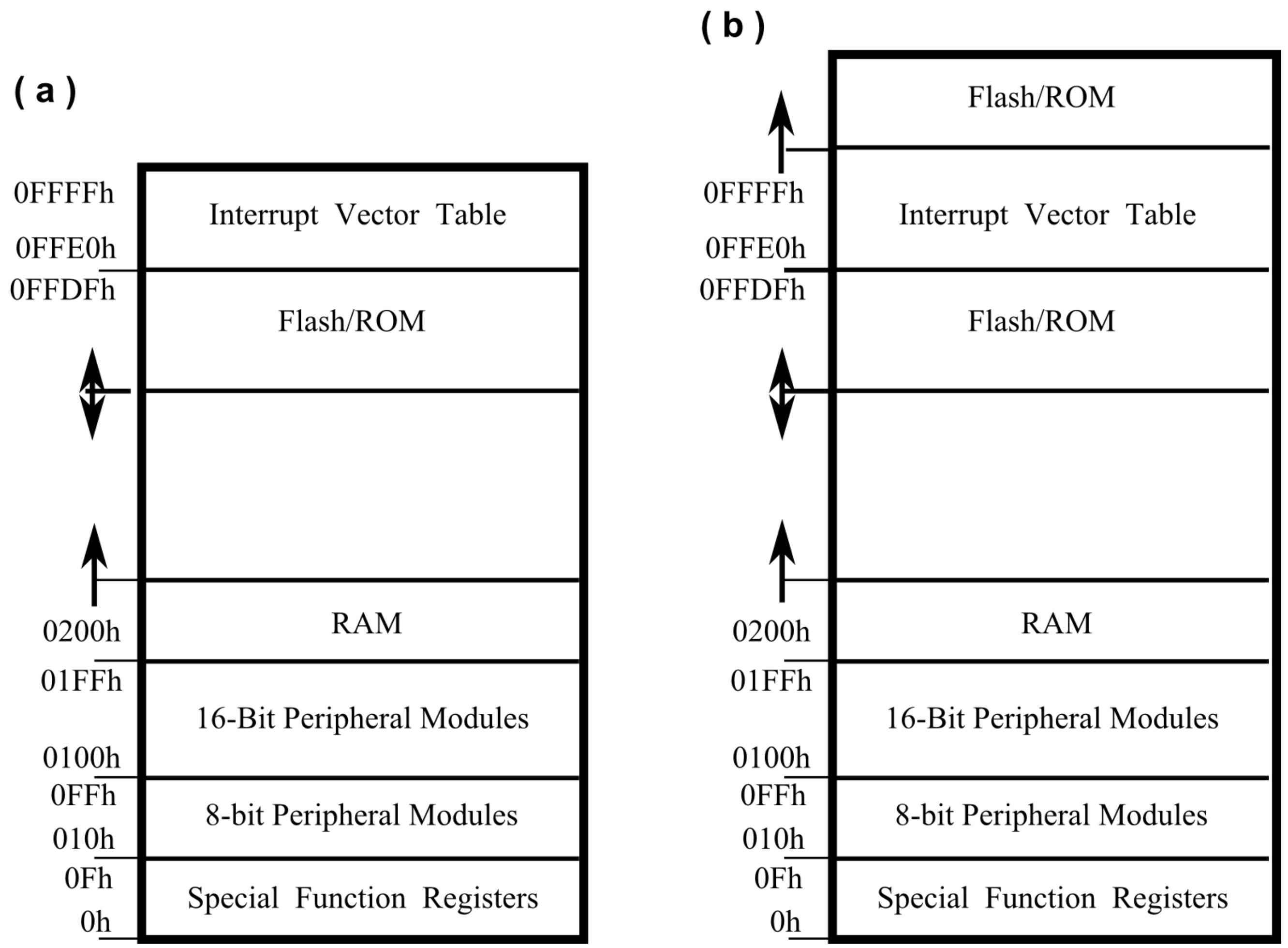
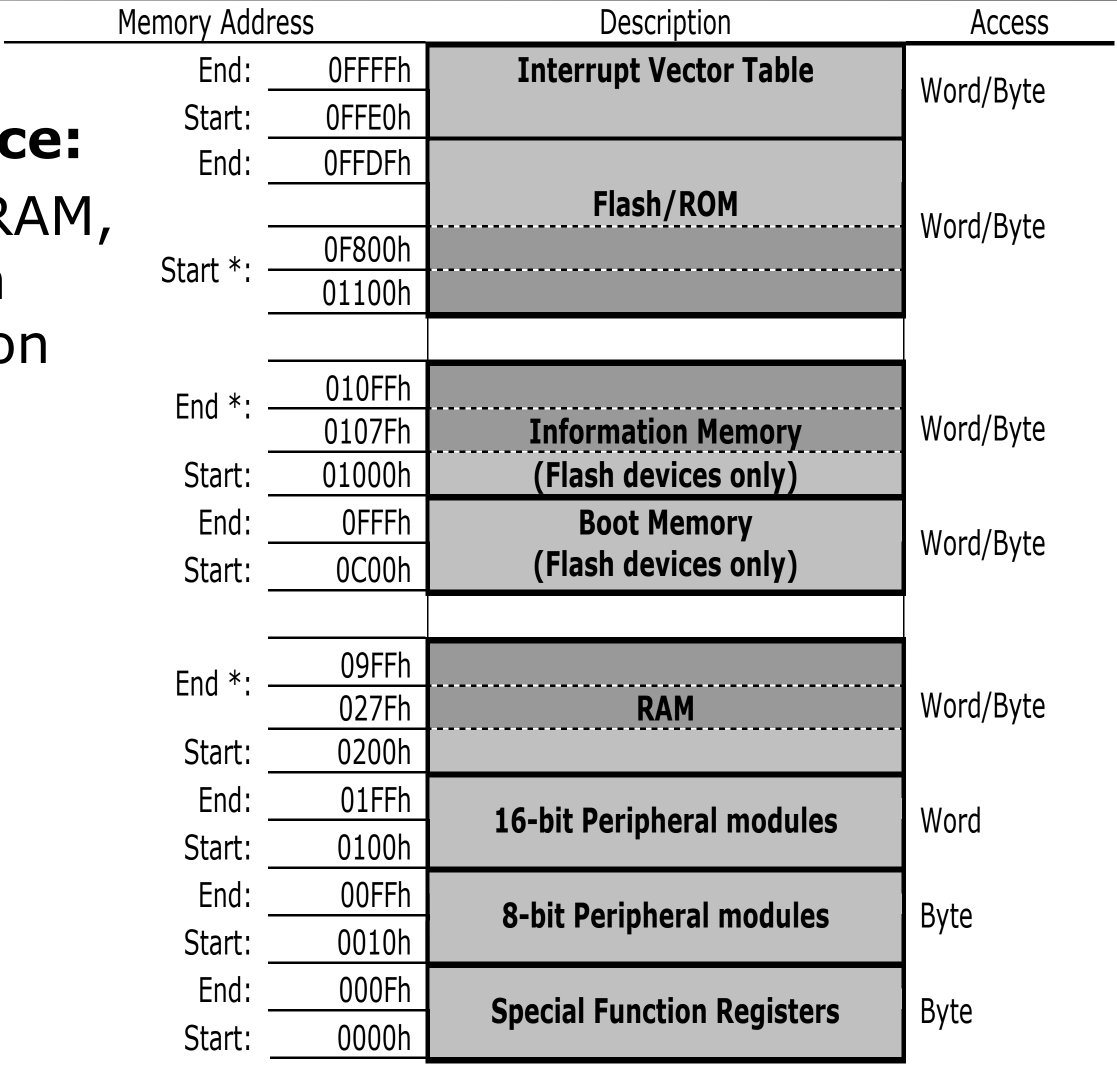


Fig. 3.14 MSP430 global memory maps (*Courtesy of Texas Instruments, Inc*). **a** 16-bit address bus memory map, **b** 20-bit address bus memory map

Address Space

- **Mapped into a single, contiguous address space:**
  - All memory, including RAM, Flash/ROM, information memory, special function registers (SFRs), and peripheral registers.
  - **Memory Map:**



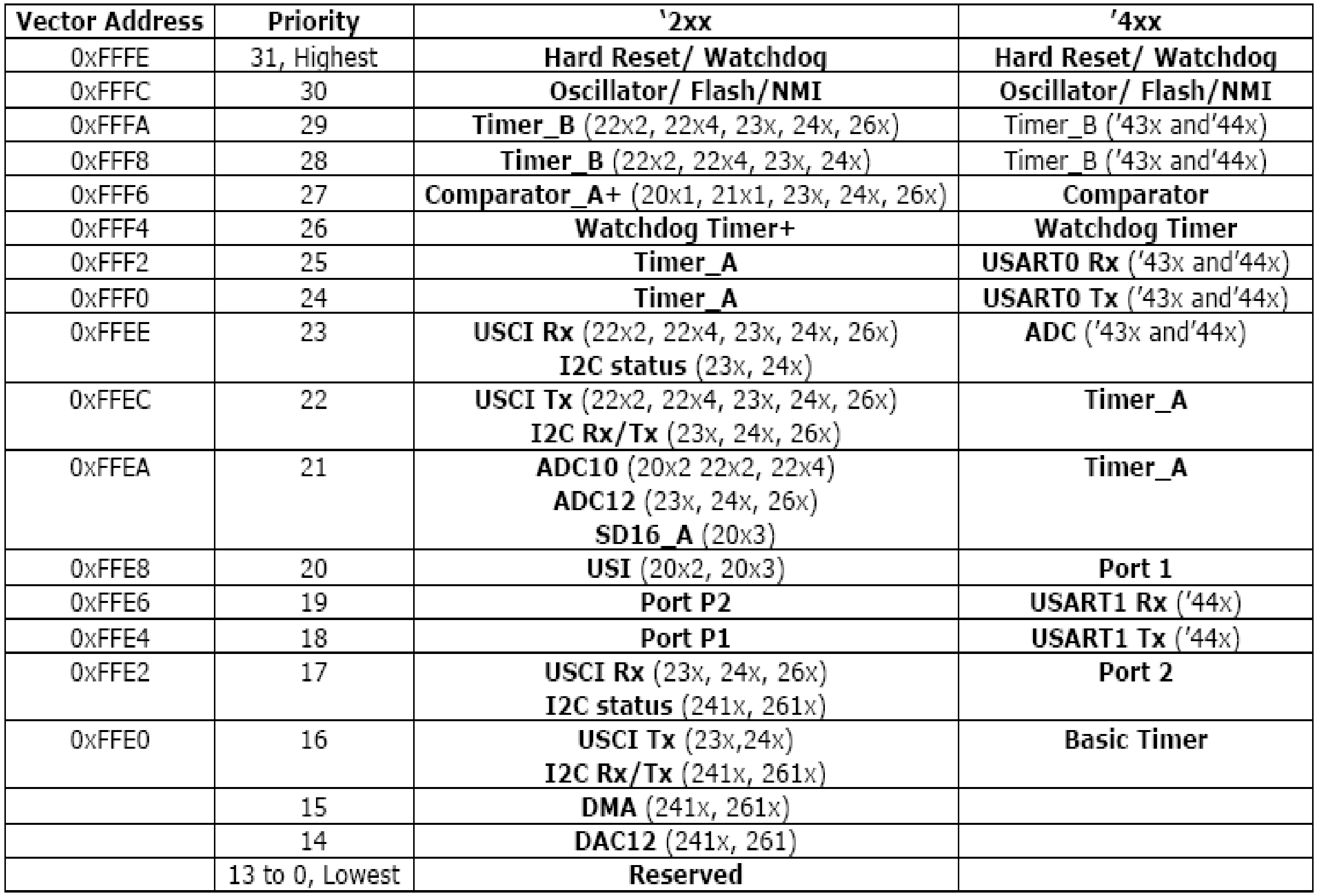
Memory Address		Description	Access
End:	0FFFFh	**Interrupt Vector Table**	Word/Byte
Start:	0FFE0h		
End:	0FFDFh	**Flash/ROM**	Word/Byte
Start *:	0F800h		
	01100h		
End *:	010FFh	**Information Memory (Flash devices only)**	Word/Byte
	0107Fh		
Start:	01000h		
End:	0FFFh	**Boot Memory (Flash devices only)**	Word/Byte
Start:	0C00h		
End *:	09FFh	**RAM**	Word/Byte
	027Fh		
Start:	0200h		
End:	01FFh	**16-bit Peripheral modules**	Word
Start:	0100h		
End:	00FFh	**8-bit Peripheral modules**	Byte
Start:	0010h		
End:	000Fh	**Special Function Registers**	Byte
Start:	0000h		

Address Space

- **Random access memory:** Used for variables. This always starts at address 0x0200 and the upper limit depends on the size of the RAM. The F2013 has 128 B.
- **Bootstrap loader:** Contains a program to communicate using a standard serial protocol, often with the COM port of a PC. This can be used to program the chip but improvements in other methods of communication have made it less important than in the past, particularly for development. All MSP430s had a bootstrap loader until the F20xx, from which it was omitted to improve security.
- **Information memory:** A 256B block of flash memory for storage of nonvolatile data. Such as an address for a network, or variables that should be retained even when power is removed. For example, a printer might remember the settings from when it was last used and keep a count of the total number of pages printed.
- **Code memory:** Holds the program, including the executable code itself and any constant data. The F2013 has 2KB but the F2003 only 1KB.
- **Interrupt and reset vectors:** Used to handle "exceptions," when normal operation of the processor is interrupted or when the device is reset. This table was smaller and started at 0xFFE0 in earlier devices.

Interrupt vector table

- **Mapped at the very end of memory space (upper 16 words of Flash/ROM):** `0FFE0h - 0FFFEh` (4xx devices);
- **Priority of the interrupt vector increases with the word address.**



Vector Address	Priority	'2xx	'4xx
0xFFFE	31, Highest	**Hard Reset/ Watchdog**	**Hard Reset/ Watchdog**
0xFFFC	30	**Oscillator/ Flash/NMI**	**Oscillator/ Flash/NMI**
0xFFFA	29	**Timer_B** (22x2, 22x4, 23x, 24x, 26x)	Timer_B ('43x and'44x)
0xFFF8	28	**Timer_B** (22x2, 22x4, 23x, 24x)	Timer_B ('43x and'44x)
0xFFF6	27	**Comparator_A+** (20x1, 21x1, 23x, 24x, 26x)	**Comparator**
0xFFF4	26	**Watchdog Timer+**	**Watchdog Timer**
0xFFF2	25	**Timer_A**	**USART0 Rx** ('43x and'44x)
0xFFF0	24	**Timer_A**	**USART0 Tx** ('43x and'44x)
0xFFEE	23	**USCI Rx** (22x2, 22x4, 23x, 24x, 26x) **I2C status** (23x, 24x)	**ADC** ('43x and'44x)
0xFFEC	22	**USCI Tx** (22x2, 22x4, 23x, 24x, 26x) **I2C Rx/Tx** (23x, 24x, 26x)	**Timer_A**
0xFFEA	21	**ADC10** (20x2 22x2, 22x4) **ADC12** (23x, 24x, 26x) **SD16_A** (20x3)	**Timer_A**
0xFFE8	20	**USI** (20x2, 20x3)	**Port 1**
0xFFE6	19	**Port P2**	**USART1 Rx** ('44x)
0xFFE4	18	**Port P1**	**USART1 Tx** ('44x)
0xFFE2	17	**USCI Rx** (23x, 24x, 26x) **I2C status** (241x, 261x)	**Port 2**
0xFFE0	16	**USCI Tx** (23x,24x) **I2C Rx/Tx** (241x, 261x)	**Basic Timer**
	15	**DMA** (241x, 261x)	
	14	**DAC12** (241x, 261)	
	13 to 0, Lowest	**Reserved**	

MSP430

- The CPU is identical for all members of the ’430 family.
- 3-stage instruction pipeline, instruction decoding, a 16-bit ALU, four dedicated-use registers, and twelve working (or scratchpad) registers
- The CPU is connected to its memory through two 16-bit busses, one for addressing, and the other for data
- All memory, including RAM, ROM, information memory, special function registers, and peripheral registers are mapped into a single, contiguous address space.

>> Contents

Central Processing Unit (MSP430 CPU) (1/7)

- **RISC (Reduced Instructions Set Computing) architecture:**
  - Instructions are reduced to the basic ones (short set):
    - 27 physical instructions;
    - 24 emulated instructions.
  - This provides simpler and faster instruction decoding;
  - Interconnect by a using a common memory address bus (MAB) and memory data bus (MDB) - Von Neumann architecture:
    - Makes use of only one storage structure for data and instructions sets.
    - The separation of the storage processing unit is implicit;
    - Instructions are treated as data (programmable).

Central Processing Unit (MSP430 CPU) (2/7)

- **RISC (Reduced Instructions Set Computing) type architecture:**
  - Uses a 3-stage instruction pipeline containing:
    - Instruction decoding;
    - 16 bit ALU;
    - 4 dedicated-use registers;
    - 12 working registers.

- **Address bus has 16 bit so it can address 65 kB (including RAM + Flash + Registers);**

- **Arithmetic Logic Unit (ALU):**
  - Addition, subtraction, comparison and logical (AND, OR, XOR) operations;
  - Operations can affect the overflow, zero, negative, and carry flags of the SR (Status Register).

Data Bus

- The set of lines carrying data and instructions to or from the CPU is called the *data bus*.
- A *read* operation occurs when information is being transferred *into* the CPU.
- A data bus transfer *out from* the CPU into memory or into a peripheral device, is called a *write* operation.
- Note that the designation of a transfer on the data bus as read or write is always made with respect to the CPU.
- Data bus lines are generally bi-directional because the same set of lines allows us to carry information to or from the CPU.
- One transfer of information is referred to as *data bus transaction*.
- The number of lines in the data bus determines the maximum *data width* the CPU can handle in a single transaction; wider data transfers are possible, but require multiple data bus transactions

MSP430 REGISTERS

- Based on a 16-bit RISC architecture
  - CPUX: Extends base architecture to 20-bit wide
- Register Structure
  - Organized as a 16, 16-bit register file (R0 – R15)
- Special Purpose Registers
  - Program Counter (PC): named R0 or PC
  - Stack Pointer (SP): named R1 or SP
  - Status Register (SR): named R2 or SR
  - Has a dual function as (SR) and as Constant Generator 1 (CG1)
  - Constant Generator 2, named CG2 or R3
- PC & SP always point to even addresses
  - Their LSB is always 0

MSP430 REGISTERS

- **Status Flags**
  - Carry (C), Zero (Z), Sign or Negative (N), overflow (V), and global interrupt enable (GIE)
  - Other flags in the PS
    - CPUOFF, OSCOFF, SCG1 and SCG0
    - Are used to configure the CPU, oscillator and low power modes
- **MSP430 ALU**
  - 16-bit wide (20-bit in CPUX)
  - Arithmetic ADD, SUB, CMP operations
  - BCD arithmetic
  - Bitwise logic operations
  - Does not provide for multiplication, division, or FP
  - Some MSP430 feature a hardware multiplier peripheral device

Working Registers

- ❑ Twelve 16-bit working registers, R4 through R15
- ❑ Use these registers as much as possible. Any variable which is accessed often should reside in one of these locations, for the sake of efficiency.
- ❑ Generally, you may select any of these registers for any purpose, either data or address.
- ❑ However, some development tools will reserve R4 and R5 for debug information. Different compilers will use these registers in different fashions, as well.
- ❑ Understand your tools.

Central Processing Unit (MSP430 CPU) (3/7)

- **Incorporates sixteen 16-bit registers:**
  - 4 registers (R0, R1, R2 and R3) have dedicated functions;
  - 12 register are working registers (R4 to R15) for general use.

- **R0: Program Counter (PC):**
  - Points to the next instruction to be read from memory and executed by the CPU.

Program Counter

- The Program Counter is located in R0.
- Individual memory location addresses are 8-bit, but all instructions are 16 bit, the PC is constrained to even numbers (i.e. the LSB of the PC is always zero).
- Avoid direct manipulation of the PC except following:
- **Example: Switch Statement via Manual PC Control**

```
mov value,R15      ; Put the Switch value into R15
cmp R15,#8         ; range checking
jge outofrange     ; If R15>7, do not use PC Switch
cmp #0,R15         ; more range checking
jn outofrange
rla R15            ; Multiply R15 by two, since PC is always even
rla R15            ; Double R15 again, since symbolic jmp is 2 words long
add R15,PC         ; PC goes to proper jump
jmp value0
jmp value1
jmp value2
jmp value3
jmp value4
jmp value5
jmp value6
jmp value7
outofrange
jmp RangeError
```

Central Processing Unit (MSP430 CPU) (4/7)

- **R1: Stack Pointer (SP):**
  - 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.

Stack Pointer

- The Stack Pointer is implemented in R1. Like the Program Counter, the LSB is fixed as a zero value, so the value is always even.
- The stack is implemented in RAM, and it is common practice to start the SP at the top (TOS- highest valid value) of RAM.
- Asymmetric push/pop combinations.
- Stack encroachment problem

Central Processing Unit (MSP430 CPU) (5/7)

- **R2: Status Register (SR):**
  - Stores status and control bits;
  - System flags are changed automatically by the CPU;
  - Reserved bits are used to support the constant generator.

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

Bit		Description
8	V	Overflow bit. V = 1 ⇒ Result of an arithmetic operation overflows the signed-variable range.
7	SCG1	System clock generator 1. SCG1 = 1 ⇒ DCO generator is turned off – if not used for MCLK or SMCLK
6	SCG0	System clock generator 0. 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.

- CPUOFF, OSCOFF, SCG0, and SCG1, which control the mode of operation of the MCU. Setting various combinations of these bits puts the MCU into one of its low-power modes, which is described in the section “Low-Power Modes of Operation”

- All bits are clear in the full-power, active mode. This is the main effect of setting each bit in the MSP430F2xx:
- **CPUOFF** disables MCLK, which stops the CPU and any peripherals that use MCLK.
- **SCG1** disables SMCLK and peripherals that use it.
- **SCG0** disables the DC generator for the DCO (disables the FLL in the MSP430x4xx family).
- **OSCOFF** disables LFXT1.

Central Processing Unit (MSP430 CPU) (6/7)

- **R2/R3: Constant Generator Registers (CG1/CG2):**
  - Depending of the source-register addressing modes (As) value, six constants can be generated without 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 implement 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

Constant Generators

- R2 and R3 function as constant generators, so that register mode may be used instead of immediate mode for some common constants.
- R2 is a dual use register. It serves as the Status Register, as well.
- The use of a constant generator saves a word in the machine instruction.
- For example, mov #1,R7 translates into **4317** (0010 **0011** 00**01** 0111), and
- mov.b #4,R7 into **4267** (0010 **0001** 01**10** 0111).

W(S)	**Value in R2**	**Value in R3**
00	——	0000h
01	(0) (absolute mode)	0001h
10	0004h	0002h
11	0008h	0FFFFh

Status Register

- The Status Register is implemented in R2, and is comprised of various system flags.
- The bits of the SR are:
- The Carry Flag (C)
  - Location: SR(0) (the LSB)
  - Function: Identifies when an operation results in a carry. Can be set or cleared by software, or automatically.
  - 1=Carry occurred
  - 0=No carry occurred

Status Register

- The Zero Flag (Z)
- Location: SR(1)
- Function: Identifies when an operation results in a zero. Can be set or cleared by software, or automatically.
- 1=Zero result occurred
- 0=Nonzero result occurred

- The Negative Flag (N)
- Location: SR(2)
- Function: Identifies when an operation results in a negative. Can be set or cleared by software, or automatically. This flag reflects the value of the MSB of the operation result (Bit 7 for byte operations, and bit 15 for word operations).
- 1=Negative result occurred
- 0=Positive result occurred

Status Register

- The Global Interrupt Enable (GIE)
  Location: SR(3)
  Function: Enables or disables all maskable interrupts. Can be set or cleared by software, or automatically.
- Interrupts automatically reset this bit, and the reti instruction automatically sets it.
- 1=Interrupts Enabled
- 0=Interrupts Disabled

- The CPU off bit (CPUOff)
  Location: SR(4)
  Function: Enables or disables the CPU core. Can be cleared by software, and is reset by enabled interrupts.
- None of the memory, peripherals, or clocks are affected by this bit. This bit is used as a power saving feature.
- 1=CPU is on
- 0=CPU is off

Status Register

- The Oscillator off bit (OSCOff)
  Location: SR(5)
  Function: Enables or disables the crystal oscillator circuit (LFXT1).
- Can be cleared by software, and is reset by enabled external interrupts.
- OSCOff shuts down everything, including peripherals. RAM and register contents are preserved. This bit is used as a power saving feature.
- 1=LFXT1 is on
- 0=LFXT1 is off
- The System Clock Generator (SCG1,SCG0)
  Location: SR(7),SR(6)
  Function: These bits, along with OSCOff and CPUOff define the power mode of the device.
- More on this later

Status Register

- The Overflow Flag (V)
  Location: SR(8)
  Function: Identifies when an operation results in an overflow.
- Can be set or cleared by software, or automatically.
- Overflow occurs when two positive numbers are added together, and the result is negative, or when two negative numbers are added together, and the result is positive.
- 1=Overflow result occurred
- 0=No overflow result occurred
- Four of these flags (Overflow, Negative, Carry, and Zero) drive program control, via instructions such as cmp (compare) and jz (jump if Zero flag is set).

Central Processing Unit (MSP430 CPU) (7/7)

- **R4 - R15: General–Purpose Registers:**
  - These general-purpose registers are adequate to store data registers, address pointers, or index values and can be accessed with byte or word instructions.

Copyright 2008
Texas Instruments

Special Function Registers

- Special function registers are, memory-mapped registers with special dedicated functions
- There are, sixteen of these registers, at memory locations 0000h through 000Fh.
- Only the first six are used.
- Locations 0000h and 0001h contain interrupt enables
- Locations 0002h and 0003h contain interrupt flags
- Locations 0004h and 0005h contain module enable flags Only two bits are implemented in each byte. These bits are used for the USARTs.

Peripheral Registers

- All on-chip peripheral registers are mapped into memory, immediately after the special function registers.
- Byte-addressable, which are mapped in the space from 010h to 0FFh, and
- Word-addressable, which are mapped from 0100h to 01FFh.

RAM

- RAM always begins at location 0200h, and is contiguous up to its final address.
- RAM is used for all scratchpad variables, global variables, and the stack
- The developer needs to be careful that scratchpad allocation and stack usage do not encroach on each other, or on global variables
- Accidental sharing of RAM is a very common bug, and can be difficult to chase down
- Be consistent about use. Locate the stack at the very end of the RAM space, and place your most commonly used globals at the beginning.

Boot Memory (flash devices only)

- Boot memory is implemented in flash devices only, located in memory locations 0C00h through 0FFFh.
- It is the only hard-coded ROM space in the flash devices.
- This memory contains the bootstrap loader, which is used for programming of flash blocks, via a USART module.

Information Memory (flash devices only)

- ❑ Flash devices in the ’430 family have the added feature of information memory.
- ❑ This information memory acts as onboard EEPROM, allowing critical variables to be preserved through power down.
- ❑ It is divided into two 128-byte segments.
- ❑ The first of these segments is located at addresses 01000h through 0107Fh, and the second is at 01080h through 010FFh.

Code Memory

- ❑ Code memory is always contiguous at the end of the address space (i.e. always runs to location 0FFFFh).
- ❑ The code runs from 01100h to 0FFE0h for 60K devices
- ❑ The code runs from 0E000 to 0FFE0h for 8K devices.
- ❑ All code, tables, and hard-coded constants reside in this memory space.

Interrupt Vectors

- Interrupt vectors are located at the very end of memory space, in locations 0FFE0h through 0FFFEh.

01B0h-01FFh	Unused (All devices)
01A0h-01Afh	ADC Control
0180h-019Fh	Timer B
0160h-017Fh	Timer A
0140h-015Fh	ADC Conversion
0130h-013Fh	Multiplier
0120h-012Fh	Watchdog timer
0110h-011Fh	ADC
0100h-010Fh	Unused (All devices)
00B0h-00FFh	Unused (All devices)
0090h-00Afh	LCD
0080h-008Fh	ADC memory control

0080h-008Fh	ADC memory control
0070h-007Fh	USART
0060h-006Fh	Unused
0050h-005Fh	System Clock (Byte addressable, All devices) / Comparator
0040h-004Fh	Basic Timer and 8-bit Counter
0030h-003Fh	I/O ports 5 and 6 control
0020h-002Fh	I/O ports 1 and 2 control
0010h-001Fh	I/O ports 3 and 4 control
0006h-000Fh	Unused
0005h	Module Enables 2
0004h	Module Enables 1
0003h	Interrupt Flags 2
0002h	Interrupt Flags 1
0001h	Interrupt Enables 2
0000h	Interrupt Enables 1

IO SUBSYSTEM

- All the components other than the CPU & memory connected to the system buses
    - Timers & Watchdog timers
    - Communication interfaces
    - Analog to Digital Converter (ADC)
    - Digital to Analog Converter (DAC)
    - Development peripherals
- Its organization resembles that of memory
- Memory mapped I/O
    - Address space inside the memory space, uses same instructions used to access memory
- I/O mapped I/O
    - Separate address space, instructions, and signals for I/O (rarely used in modern processors)

CONCEPTUAL IO INTERFACE

- Serves as a bridge between a device & the buses
- Has one or more registers each with their own addresses
    - Data, control, and status registers
- Accessed like memory (read/write)

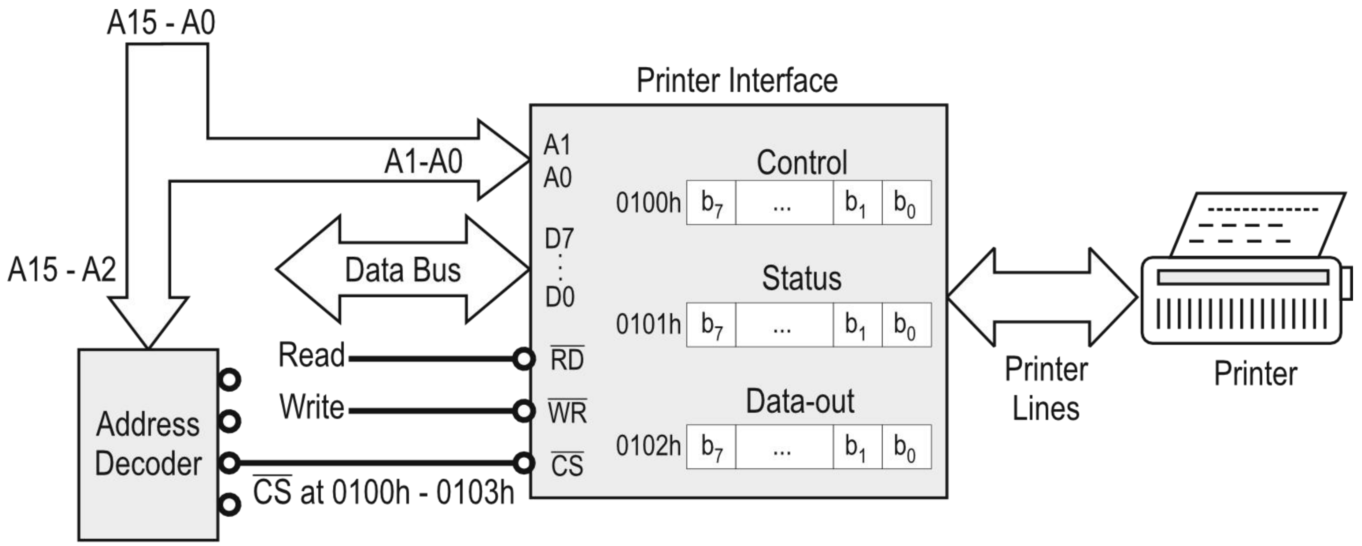


Fig. 3.17 Connection of printer interface to CPU buses and printer itself

CONCEPTUAL IO INTERFACE

- Includes lines to connect to the system buses, I/O device connection lines, and a set of internal registers
- Internal register types
  - Control: to configure the operation of the device & interface
  - Status: to allow inquiries about the device & interface status
  - Data: for exchanging data with the device

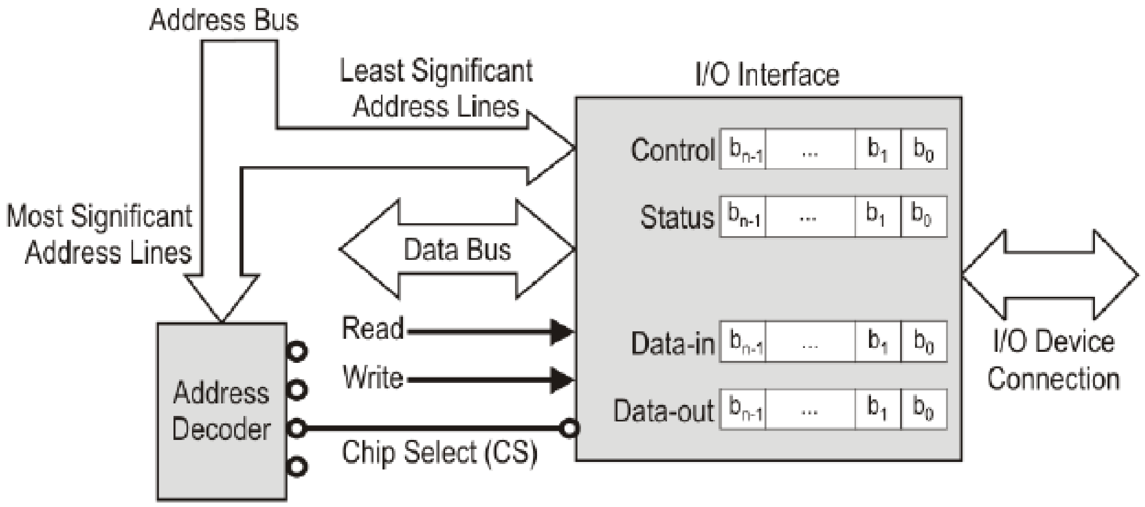


Fig. 3.16 Anatomy of an input/output (I/O) interface

INTERFACE EXAMPLE

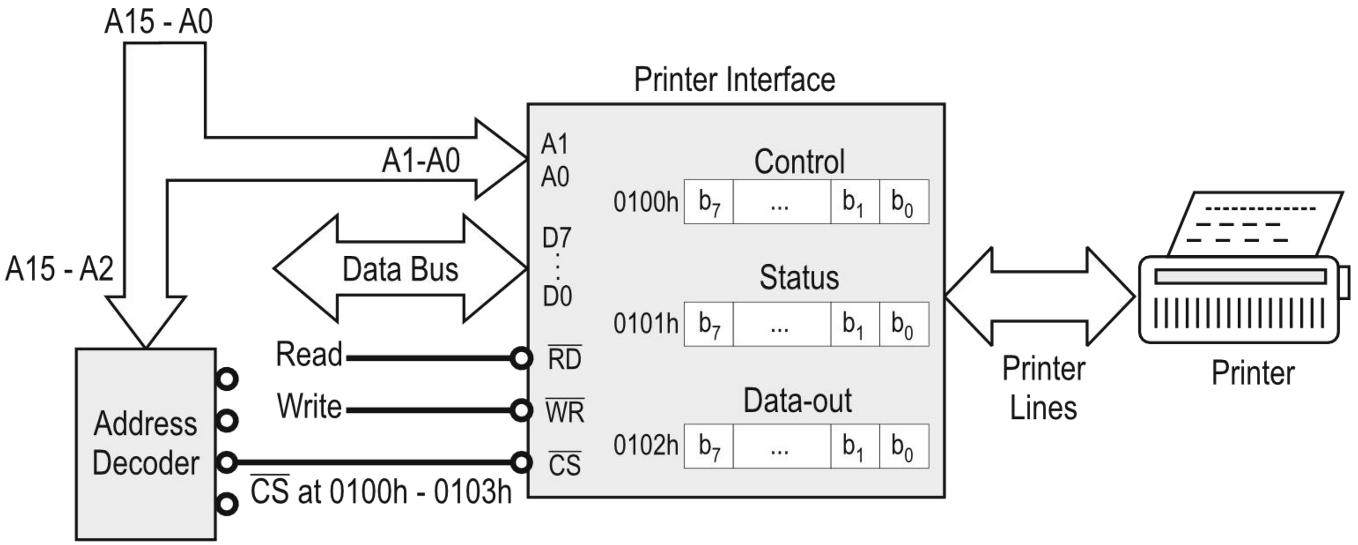


Fig. 3.17 Connection of printer interface to CPU buses and printer itself

Table 3.3 Internal addresses for sample printer interface

A1	A0	Register
0	0	Control
0	1	Status
1	0	Data-out
1	1	Not used

INTERFACE EXAMPLE

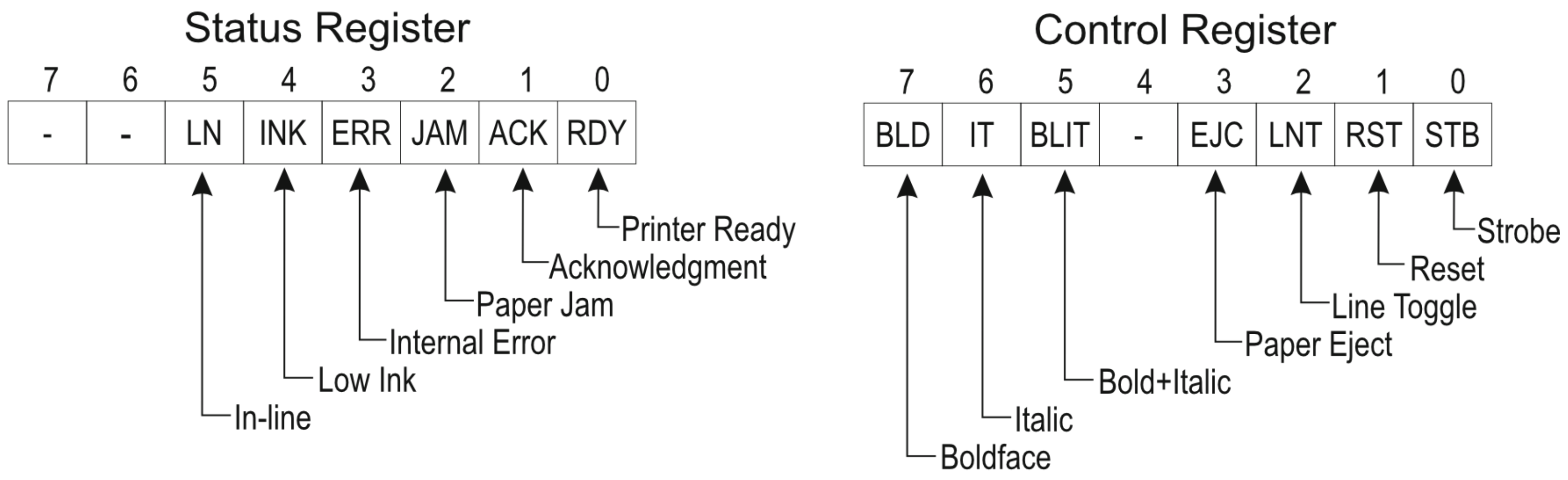


Fig. 3.18 Status and control registers in printer interface example

Input and Output

- Every system you will ever design has one thing in common: use of input and/or output
- Ports, which are fundamentally the same across all device families, can be divided into two categories:
  - **interruptible (meaning that interrupts can be generated via these ports)**
  - **non-interruptible.**
- In all devices, ports 1 and 2 are interruptible, and the higher numbered ports are not.
- Along with basic I/O functions, the port pins can be individually configured as special function I/Os, such as USARTs, Comparator signals, and ADCs.

Non-Interruptible I/O

- I/O ports 3,4,5 and 6 are non-interruptible data ports. These ports are not implemented on all devices.
- Use of non-interruptible I/O is simple and straightforward. Each bit is individually controllable, so inputs, outputs, and dedicated function I/O can be mixed in a single port.
- Port pins are controlled by four byte-addressable registers: direction, input, output, and function select.

MSP430 IO Subsystem

- Memory mapped from address 0x0000h to 0x01FFh
- Information stored in special function registers (SFR)
    - Located from 0x0000 to 0x000F
    - Control the operation of MSP430 peripherals
- Available Peripherals (Specific list is device dependent)
    - General Purpose I/O Pins
        - Arranged as multiple 8-bit I/O ports
    - 16-bit Timers
        - General purpose and Watchdog
    - Communication Support
        - USART, SPI, I2C
    - Data Converters
    - ADCs and DACs
    - System Support
        - POR and brownout reset circuitry
        - Clock generators and supply voltage supervisors

IO pin HARDWARE CONFIGURATION

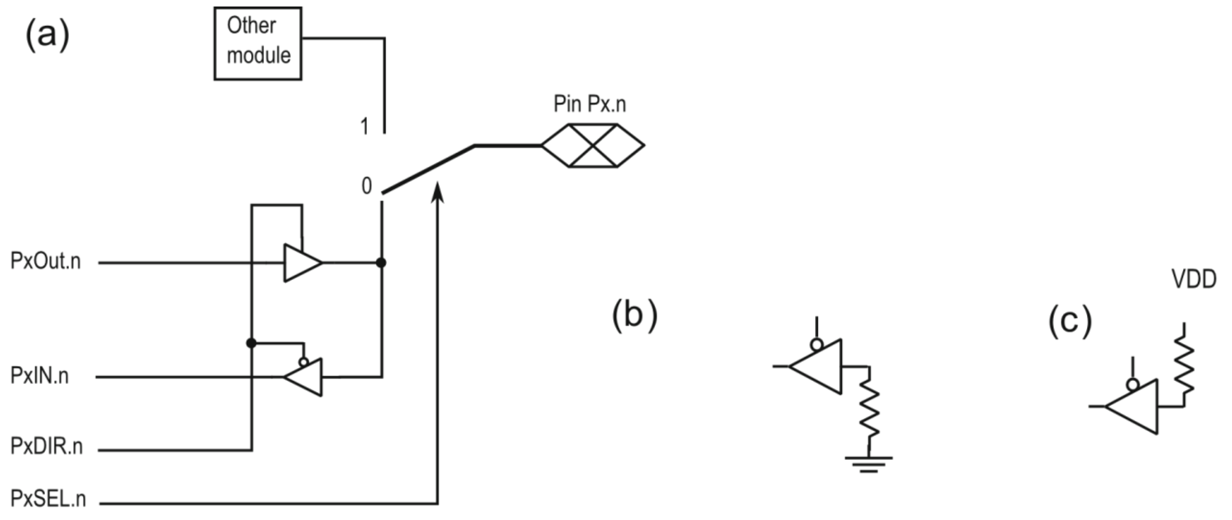


Fig. 3.19 Basic IO Pin hardware configuration: **a** Basic I/O register's functions; **b** pull-down resistor for inputs; **c** pull-up resistors

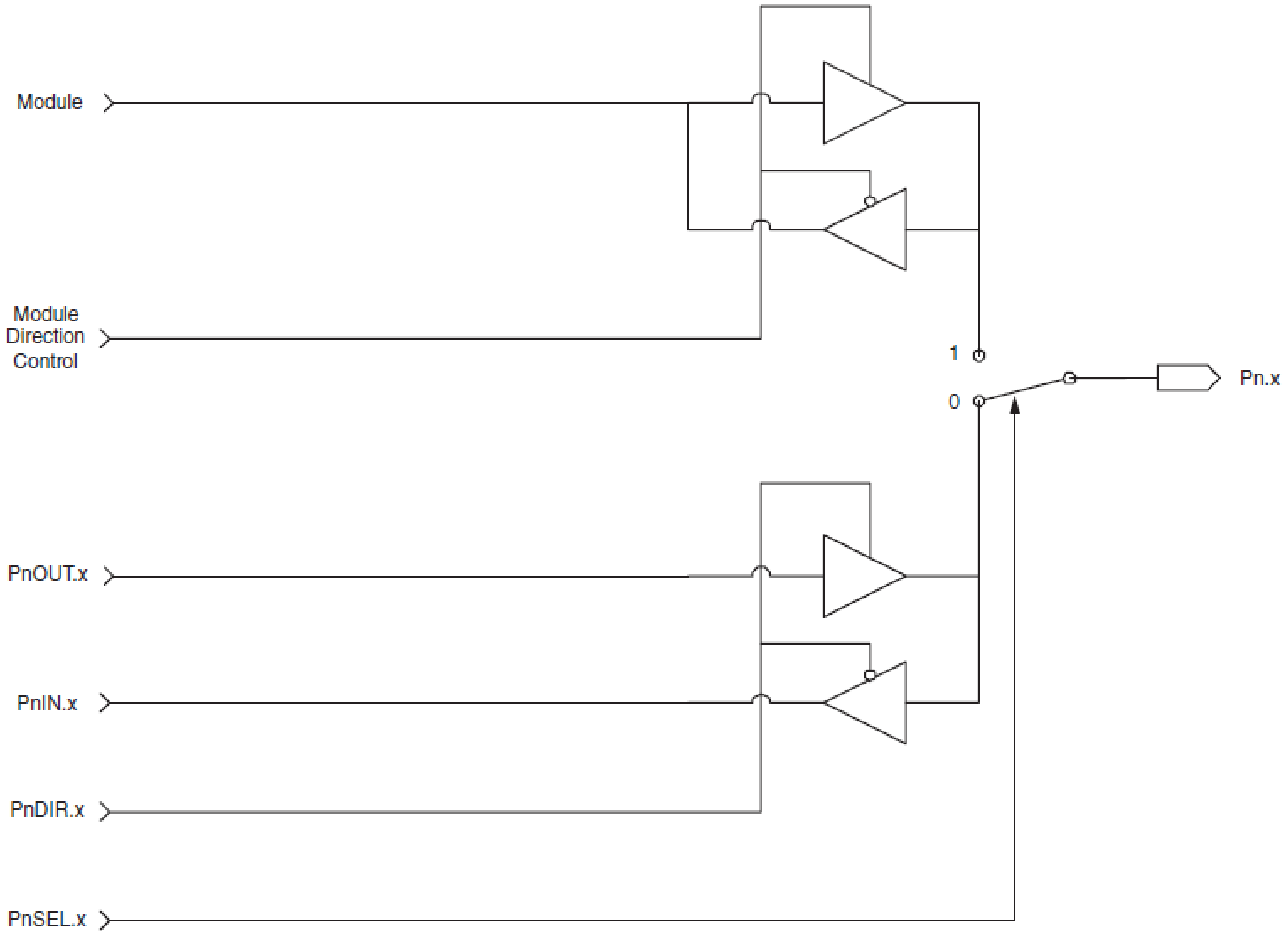
Direction Register (PxDIR): *Selects in or out direction function for pin, with 1 for output direction and 0 for input direction.*

Input Register (PxIN): *This is a read-only register. The value changes automatically when the input itself changes.*

Output Register (PxOUT): *to write signal to output. This is a read-and-write register.*

Function Select Register (PxSEL): *Used to select between I/O port or peripheral module function. With PxSEL.n = 0, pin Px.n operates as an I/O pin port; with PxSEL.n = 1, as a module pin.*

Non-Interruptible I/O



Direction Registers.

- These read/write registers control the signal direction for port pins. When a bit in the direction register is set, the corresponding port pin is set as an output, and when the bit in the direction register is cleared, the port pin is set as an input. The direction registers need to be configured properly if the port pin is selected as a general purpose I/O or as a special function I/O.
- *Direction registers* are cleared on reset.

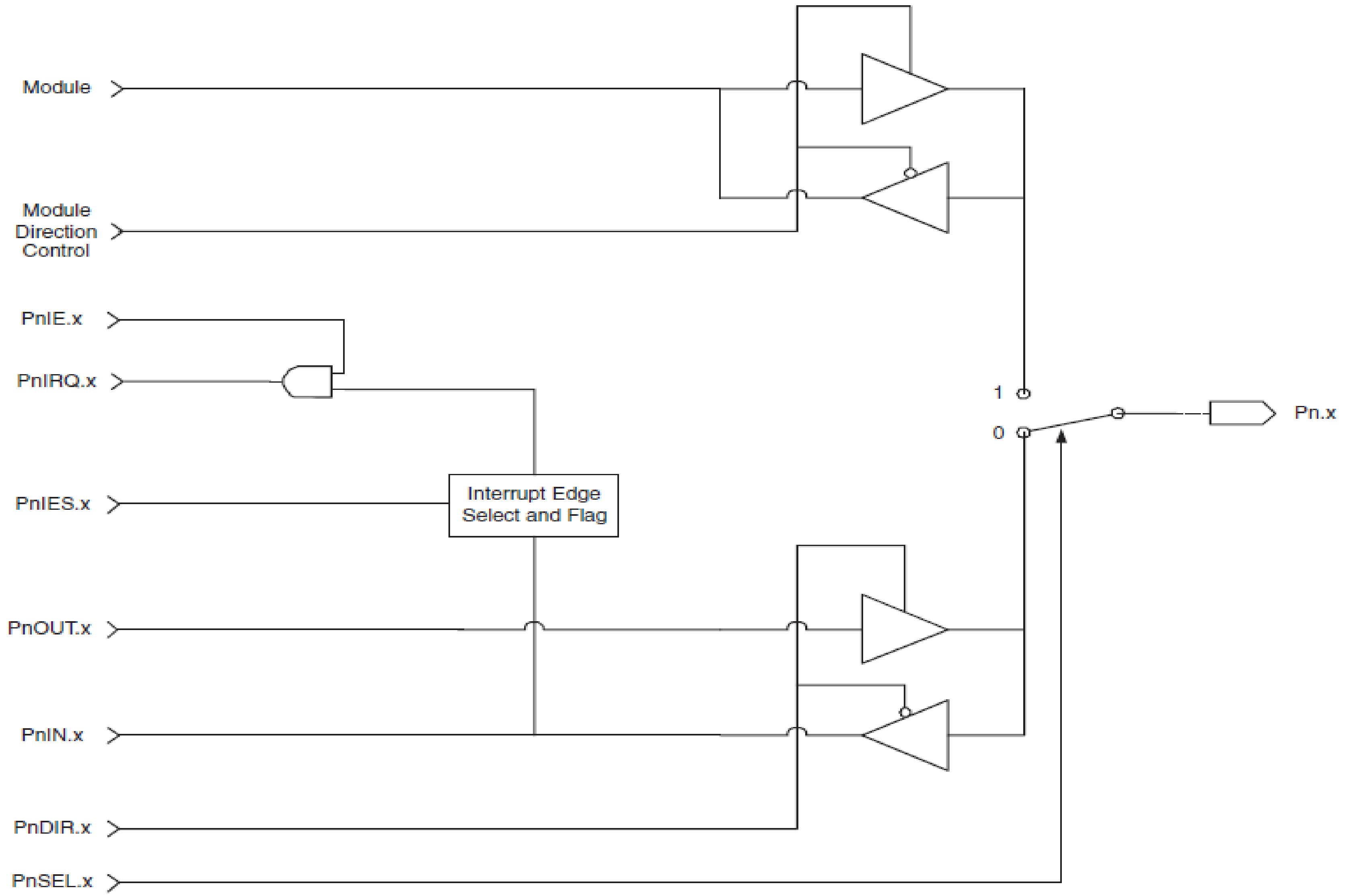
Input Output Function Regs

- *Input Registers*. These are read-only registers, which reflect the input value on the port.
- *Output Registers*. These registers are used to write to output ports, and can be read as well. When reading these registers, they will reflect the last value written to them. However, if the port is configured as an input, the output register will be in an indeterminate state. It will not necessarily reflect the input value on the associated pin.
- *Function Select Registers*. These read/write registers determine the use of the individual pins on the I/O port. When the bit in the select register is set, the port pin is set as a function I/O, and when the bit in the direction register is cleared, the port pin is set as general purpose I/O.

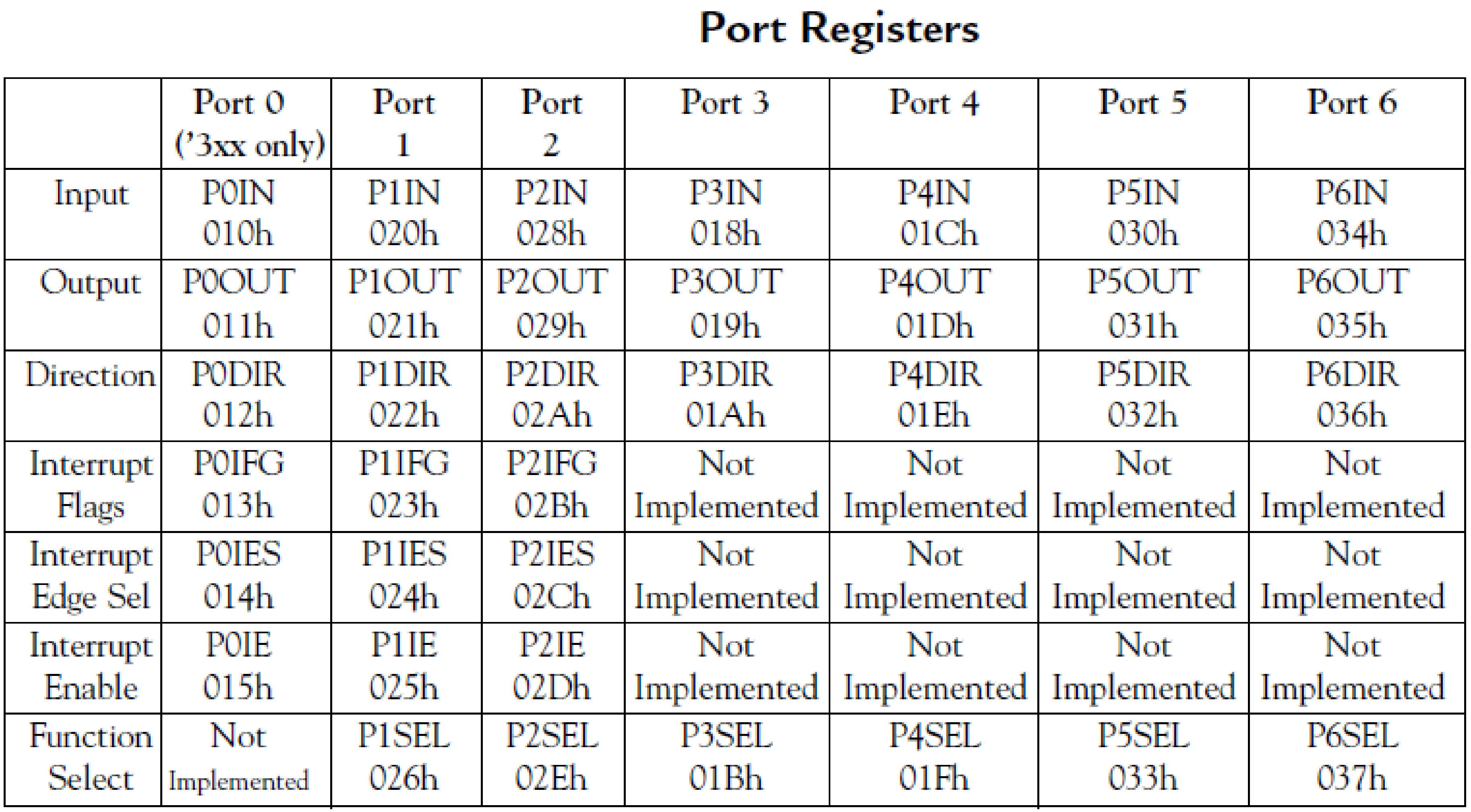
Interruptible I/O

- Ports 1 and 2 (and port 0 on ’3xx devices) are interruptible ports. They contain all of the same control registers as non-interruptible ports (described in the previous section), along with three other byte-addressable registers:
  - **interrupt enable,**
  - **interrupt edge select,**
    - *Interrupt Edge Select.* This read-write register selects the transition on which an interrupt occurs. If set, an interrupt occurs on a high-to-low transition on the corresponding pin. If cleared, an interrupt occurs on a low-to-high transition on the corresponding pin.
  - **interrupt flags**
    - The corresponding bit in this read-write register is set automatically when an interrupt is generated. This register can be written to, and will generate an interrupt when a high level is written. When an interrupt occurs, this flag needs to be cleared before the reti instruction, or the same interrupt will call the ISR a second time

Interruptible I/O



Port Registers



Port Registers

	Port 0 ('3xx only)	Port 1	Port 2	Port 3	Port 4	Port 5	Port 6
Input	P0IN 010h	P1IN 020h	P2IN 028h	P3IN 018h	P4IN 01Ch	P5IN 030h	P6IN 034h
Output	P0OUT 011h	P1OUT 021h	P2OUT 029h	P3OUT 019h	P4OUT 01Dh	P5OUT 031h	P6OUT 035h
Direction	P0DIR 012h	P1DIR 022h	P2DIR 02Ah	P3DIR 01Ah	P4DIR 01Eh	P5DIR 032h	P6DIR 036h
Interrupt Flags	P0IFG 013h	P1IFG 023h	P2IFG 02Bh	Not Implemented	Not Implemented	Not Implemented	Not Implemented
Interrupt Edge Sel	P0IES 014h	P1IES 024h	P2IES 02Ch	Not Implemented	Not Implemented	Not Implemented	Not Implemented
Interrupt Enable	P0IE 015h	P1IE 025h	P2IE 02Dh	Not Implemented	Not Implemented	Not Implemented	Not Implemented
Function Select	Not Implemented	P1SEL 026h	P2SEL 02Eh	P3SEL 01Bh	P4SEL 01Fh	P5SEL 033h	P6SEL 037h

>> Contents

Using I/O

- All port registers can be changed in software. This allows software to turn interrupts on and off, and to use a single pin for both input and output.
- When port pins are configured as inputs, they also function as highimpedance inputs. When being used as a high-impedance input, no pulling resistor is necessary, as long as any externally applied voltage is at the ground or supply rail. When configured this way, the leakage current is typically on the order of tens of nanoamperes.
- Interrupts on this device are edge triggered, and very susceptible to noise. A bit of filtering on interrupt lines will go a long way in noisy designs.

MSP430MtFaFbMc

- Mt: Memory Type
  - C: ROM
  - F: Flash
  - P: OTP
  - E: EPROM (for developmental use. There are few of these.)
- Fa, Fb: Family and Features
  - 10, 11: Basic
  - 12, 13: Hardware UART
  - 14: Hardware UART, Hardware Multiplier
  - 31, 32: LCD Controller
  - 33: LCD Controller, Hardware UART, Hardware Multiplier
  - 41: LCD Controller
  - 43: LCD Controller, Hardware UART
  - 44: LCD Controller, Hardware UART, Hardware Multiplier
- Mc: Memory Capacity
  - 0: 1kb ROM, 128b RAM
  - 1: 2kb ROM, 128b RAM
  - 2: 4kb ROM, 256b RAM
  - 3: 8kb ROM, 256b RAM
  - 4: 12kb ROM, 512b RAM
  - 5: 16kb ROM, 512b RAM
  - 6: 24kb ROM, 1kb RAM
  - 7: 32kb ROM, 1kb RAM
  - 8: 48kb ROM, 2kb RAM
  - 9: 60kb ROM, 2kb RAM

Example

- The MSP430F435 is a Flash memory device with an LCD controller, a hardware UART, 16 kb of code memory, and 512 bytes of RAM.

Microcontroller characteristics

- **Integration: Able to implement a whole design onto a single chip.**
- **Cost: Are usually low-cost devices (a few $ each);**
- **Clock frequency: Compared with other devices (microprocessors and DSPs), MCUs use a low clock frequency:**
  - MCUs today run up to 100 MHz/100 MIPS (Million Instructions Per Second).
- **Power consumption: Low power (battery operation);**
- **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 to 150) pin-out count.**

MSP430 main characteristics (1/3)

- **Low power consumption:**
  - 0.1 µA for RAM data retention;
  - 0.8 µA for real-time clock mode operation;
  - 250 µA/MIPS during 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).**

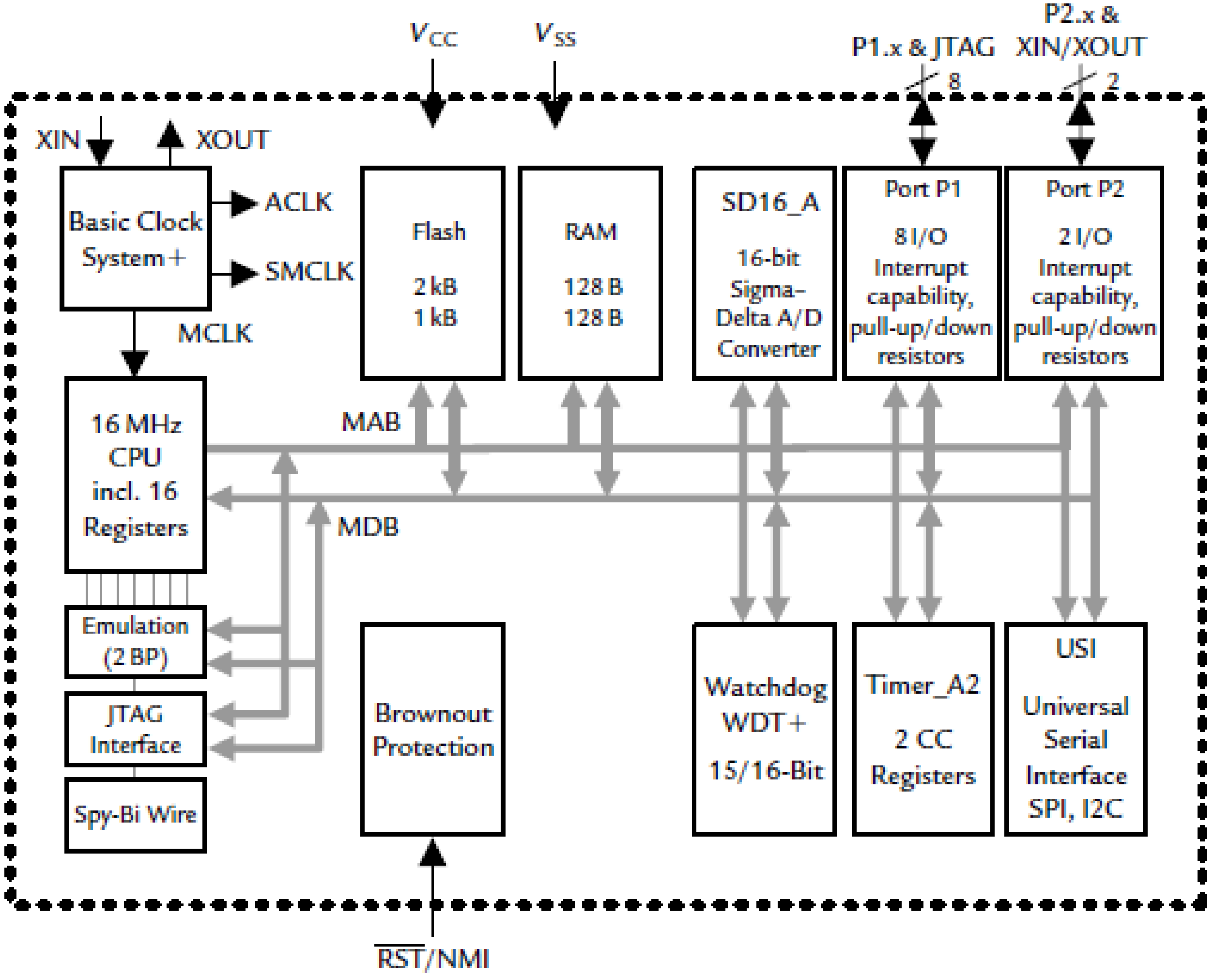
MSP430 main characteristics (3/3)

- **Flexibility:**
  - Up to 256 kByte Flash;
  - Up to 100 pins;
  - USART, I2C, Timers;
  - LCD driver;
  - Embedded emulation;
  - And many more peripherals modules…

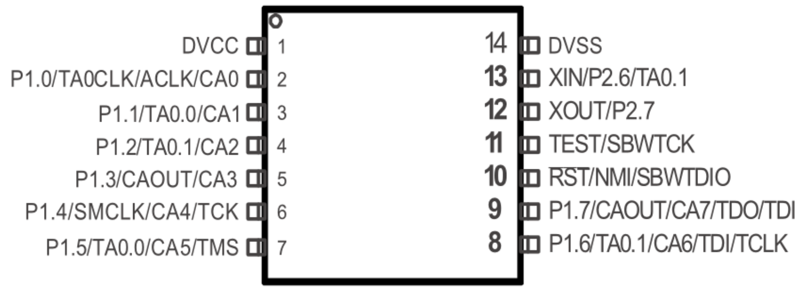
- **Microcontroller performance:**
  - Instruction processing on either bits, bytes or words
  - Reduced instructions set;
  - Compiler efficient;
  - Wide range of peripherals;
  - Flexible clock system.
- **1.8–3.6V operation**

MSP430 Architecture

- Block diagram:



MSP430 Package Pinout



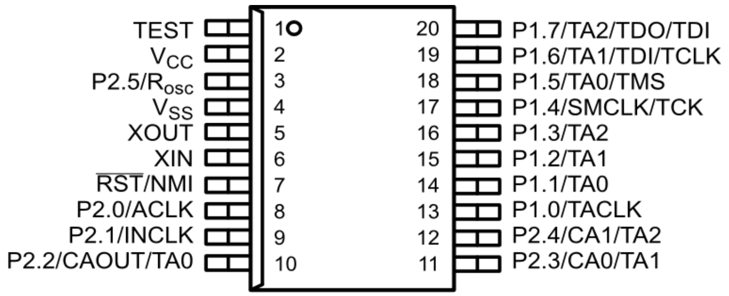


Fig. 3.22 Package pinout of two MSP430 microcontroller models: **a** MSPG2231 Pin description, **b** MSP430x1xx Pin description (*Courtesy of Texas Instruments, Inc.)*

How to Read Datasheets (1/6)

- **Manufacturers of electronic components provide datasheets containing the specifications detailing the part/device characteristics;**

- **Datasheets give the electrical characteristics of the device and the pin-out functions, but without detailing the internal operation;**

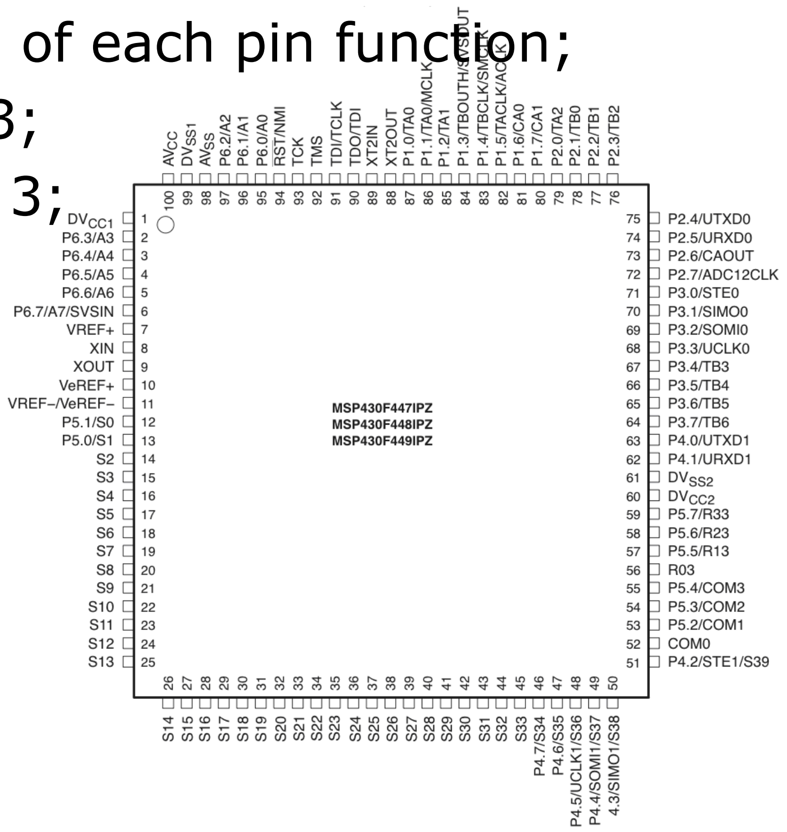
- **More complex devices are provided with documents that aid the development of applications, such as:**
  - Application notes;
  - User's guides;
  - Designer's guides;
  - Package drawings, etc…

How to Read Datasheets (2/6)

- **Datasheets include information:**
  - Concerning the part number of the device or device series;
  - Electrical characteristics:
    - Maximum and minimum operating voltages;
    - Operating temperature range e.g. 0 to 70 degrees C;
    - Output drive capacity for each output pin, as well as an overall limit for the entire chip;
    - Clock frequencies;
    - Pin out electrical characteristics (capacitance, inductance, resistance);
    - Noise tolerance or the noise generated by the device itself;
    - Physical tolerances…

How to Read Datasheets (3/6)

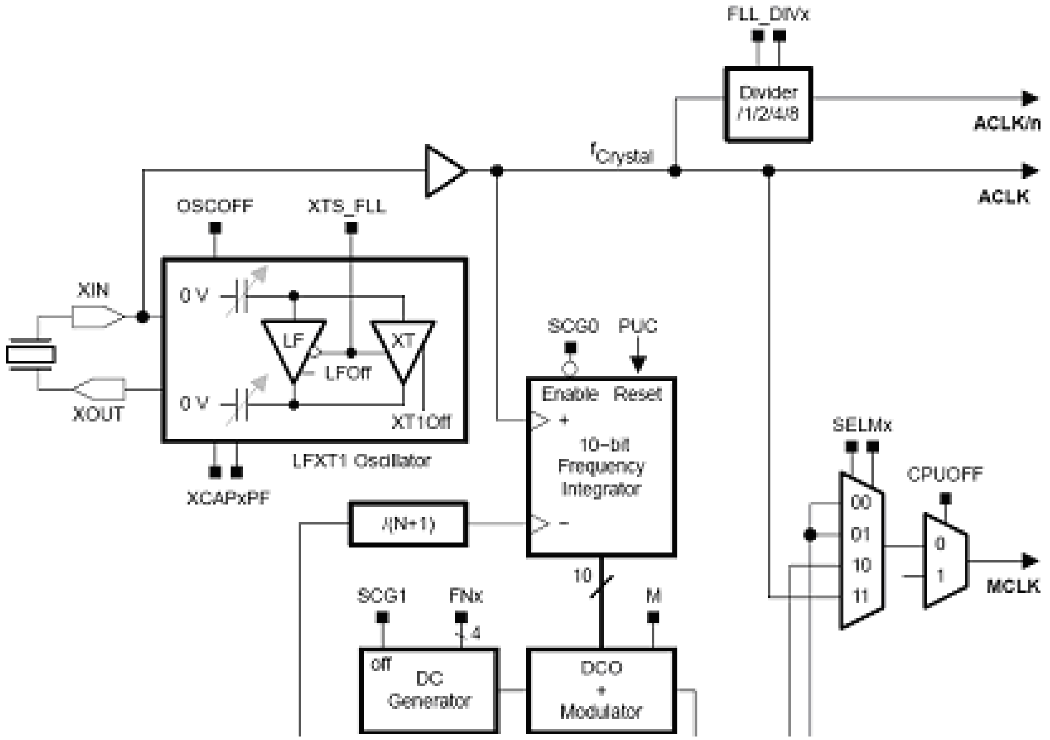
- **MSP430 device datasheet:**
  - Device has a large number of peripherals;
  - Each input/output pin usually has more than one function;
  - It has a table with the description of each pin function;
  - Example, Pin number 2 = P6.3/A3;
    - Digital Input/Output Port 6 bit 3;
    - 3rd analogue input.



>> Contents

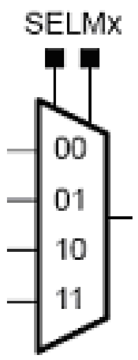
How to Read Datasheets (4/6)

- **MSP430 User's Guide:**
  - Most peripherals are represented by Block Diagrams.
  - Example: Part of the MSP430F44x clock module block diagram:



How to Read Datasheets (5/6)

- **MSP430 User's Guide:**
  - Example: Part of the MSP430F44x clock module block diagram: Detail SELMx;



  - Multiplexed block: (4 inputs and 1 output):
    - SELMx = 00: Output routed from the first input to the multiplexer output;
    - SELMx = 01: Output routed from the second one and so on;
    - SELMx is a 2-bit *mnemonic*: SELM1 (MSB), SELM0 (LSB).
  - To use the peripheral, it is necessary to find out how the register(s) need to be configured:
    - SELMx is in the FLL_CTL1 register.

How to Read Datasheets (6/6)

- **MSP430 User's Guide:**
  - SELMx are the 3rd and 4th bits of FLL_CTL1 control register.

FLL_CTL1, FLL+ control register 1

7	6	5	4	3	2	1	0
-	SMCLKOFF	XT2OFF	SELMx		SELS	FLL_DIVx	

Bit		Description

6	SMCLKOFF	Disable the submain clock signal (SMCLK):
		SMCLKOFF = 0 ⇒ SMCLK active
		SMCLKOFF = 1 ⇒ SMCLK inactive

5	XT2OFF	Disable the second crystal oscillator (XT2):
		XT2OFF = 0 ⇒ XT2 active
		XT2OFF = 1 ⇒ XT2 inactive

4-3	SELMx	Select the master clock (MCLK) source: SELM1 SELM0 = 0 0 ⇒ DCO SELM1 SELM0 = 0 1 ⇒ DCO SELM1 SELM0 = 1 0 ⇒ XT2 SELM1 SELM0 = 1 1 ⇒ LFXT1

2	SELS	Select the submain clock (SMCLK) source:
		SELS = 0 ⇒ DCO
		SELS = 1 ⇒ XT2

1-0	FLL_DIVx	Select the auxiliary clock (ACLK) signal divider:
		FLL_DIV_0 = 0 0 ⇒ Divider factor: /1
		FLL_DIV_1 = 0 1 ⇒ Divider factor: /2
		FLL_DIV_2 = 1 0 ⇒ Divider factor: /4
		FLL_DIV_3 = 1 1 ⇒ Divider factor: /8

Quiz (1/7)

- **1. The number and types of instructions used by the MSP430 CPU are:**

  (a) 27 core instructions;

  (b) 20 core instructions and 14 emulated ones;

  (c) 27 core instructions and 24 emulated ones;

  (d) 24 core instructions.

- **2. The MSP430 RISC type CPU is:**

  (a) Based on a reduced instruction set;

  (b) Based on pure pattern matching and absence of instructions;

  (c) Based on a complex instruction set;

  (d) A CPU without peripherals connections.

Quiz (2/7)

- **3. The von Neumann architecture used for the MSP430:**

  (a) Has the data storage entirely contained within the data processing unit;

  (b) Has physically separate storage and signal pathways for instructions and data;

  (c) Has a separate bus just for peripherals;

  (d) Has program, data memory and peripherals all sharing a common bus structure.

Quiz (3/7)

- **4. The ALU in the MSP430 CPU handles:**

  (a) Addition, subtraction, multiplication and division operations;

  (b) Addition, subtraction, comparison and logical (AND, OR, XOR) operations;

  (c) Addition, subtraction, multiplication and comparison operations;

  (d) Addition, subtraction, multiplication and logical (AND, OR, XOR) operations.

Quiz (4/7)

- **5. The MSP430 CPU incorporates:**

  (a) 14 registers (2 for dedicated functions and 12 for work);

  (b) 16 registers (6 for dedicated functions and 10 for work);

  (c) 18 registers (4 for dedicated functions and 14 for work);

  (d) 16 registers (4 for dedicated functions and 12 for work).

- **6. The Program Counter (PC):**

  (a) Stores the return addresses of subroutine calls and interrupts;

  (b) Points to the next instruction to be read from memory and executed by CPU;

  (c) Stores state and control bits;

  (d) Points to the next instruction to be written in memory.

Quiz (5/7)

- **7. The result of the Status Register SR = 0x0104 indicates:**

  (a) Arithmetic operation result overflows the signed-variable range and produced a carry;

  (b) Arithmetic operation result overflows the signed-variable range which result is negative, when maskable interrupts are enabled;

  (c) Arithmetic operation result is negative and produced a carry;

  (d) CPU is disabled and the maskable interrupts are enabled.

- **8. The MSP430 Status Register (SR) bit:**

  (a) V is set when the result of a byte or word operation overflows;

  (b) Z is set when the result of a byte or word operation is zero;

  (c) all of the above;

  (d) none of the above.

Quiz (6/7)

- **9. The MSP430 supports on two-address-instructions:**

  (a) Seven addressing modes for the source operand and three addressing modes for the destination operand;

  (b) Six addressing modes for the source operand and four addressing modes for the destination operand;

  (c) Seven addressing modes for the source operand and four addressing modes for the destination operand;

  (d) Six addressing modes for the source operand and three addressing modes for the destination operand.

Quiz (7/7)

- Answers
  - 1. (c) 27 core instructions and 24 emulated instructions.
  - 2. (a) Based on a reduced instruction set.
  - 3. (d) has program, data memory and peripherals all sharing a common bus structure.
  - 4. (b) Addition, subtraction, comparison and logical (OR, AND, XOR) operations.
  - 5. (d) 16 registers (4: dedicated functions and 12 working).
  - 6. (b) Points to the next instruction to be read from memory and executed by the CPU.
  - 7. (b) Arithmetic operation result overflows the signed-variable range when result is negative, when maskable interrupts are enabled.
  - 8. (c) all of the above.
  - 9. (c) Seven for the source operand and four addressing modes for the destination operand.