4 Memory Organization

The MSP430 family's memory space is configured in a "von-Neumann Architecture" and has code memory (ROM, EPROM, RAM) and data memory (RAM, EEPROM, ROM) in one address space using a unique address and data bus.

All the physically separated memory areas, the internal areas for ROM, RAM, SFRs and peripheral modules, and the external memory, are mapped into the common address space. The total addressable memory space provided is 64KB in the small memory model and 1MB in the large memory model. The small memory model uses a linear address space, while in the large memory model the address space is arranged in sixteen segments of 64KB at code access, and 16 pages of 64KB at data access.

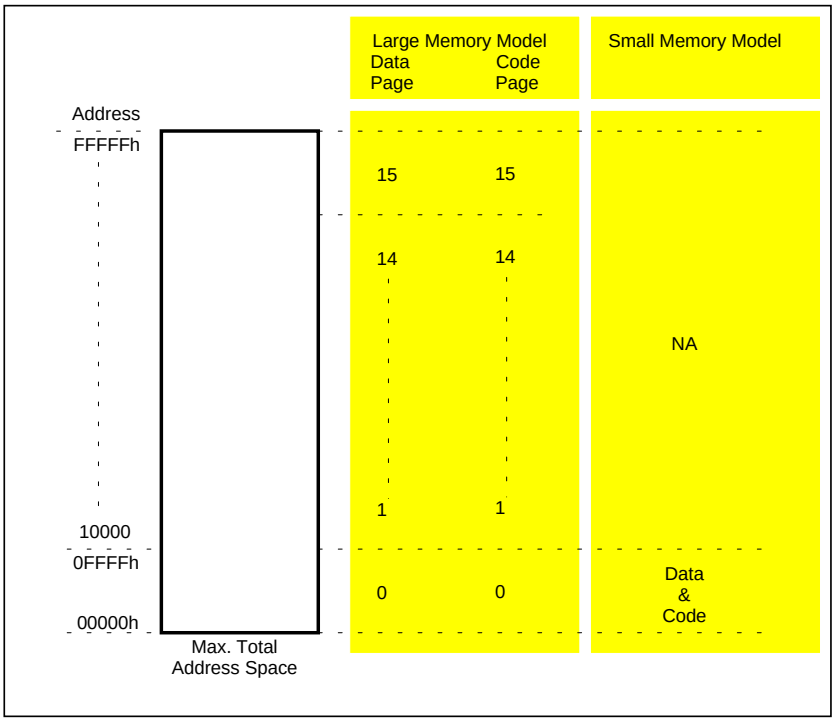


Figure 4.1: Total Memory Address Space

Devices with a memory configuration of 64KB or less use the small memory model with basic address range of the lowest 64KB, and do not care about code segments and data pages.

The configuration according to the small memory model and data bus width is shown below:

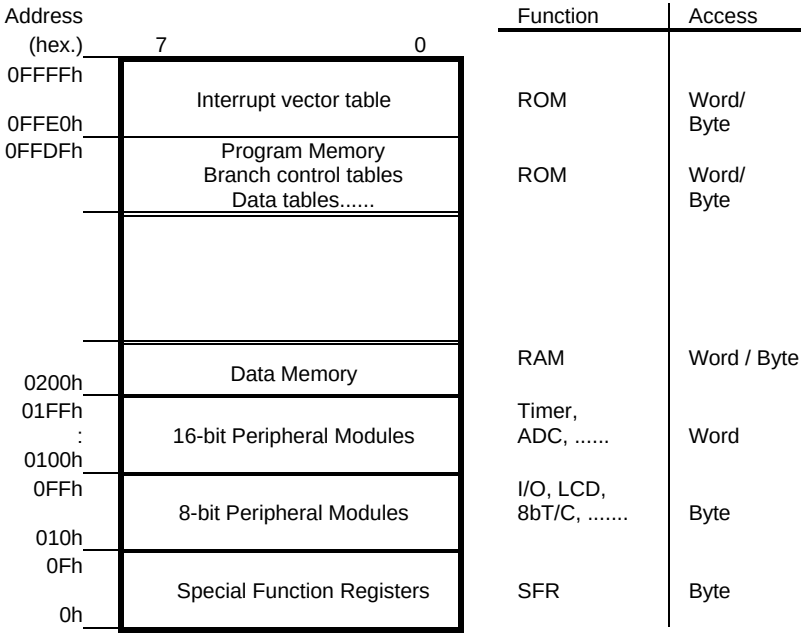
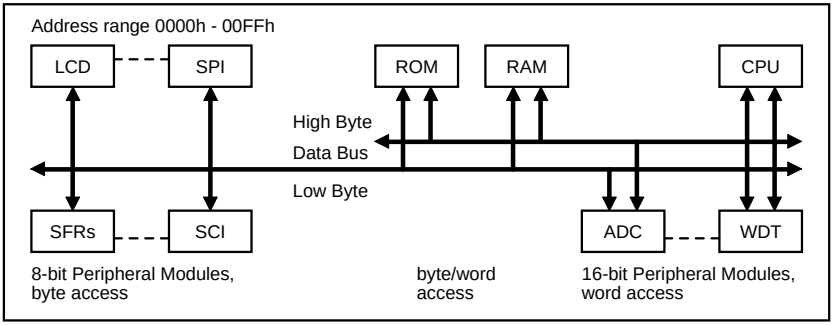


Figure 4.2: Memory Map of Basic Address Space

The Data Bus is 16-bit or 8-bit wide. For those modules that can be accessed with word data, the width is always 16 bits, and for the other modules 8 bits; they should only be accessed with byte instructions. The Program Memory (ROM) and the Data Memory (RAM) can be accessed with byte or word instructions. Parts of peripheral modules are realized as 16-bit wide or 8-bit wide modules. The access should use the proper instructions, either byte or word.

Many peripheral modules are connected to the CPU with an 8-bit Memory Data Bus (MDB), with the 5 least significant bits of the Memory Address Bus (MAB) plus two Module Enable signals (ME), two interrupt control/request lines, and a power-up signal.

The access to these modules should be always performed using byte instruction formats. Other 16-bit peripheral modules are connected to the 16-bit MDB with full supporting word processing, and should use word instruction format for any access.



4.1 Data in the Memory

Bytes are located at even or odd addresses. Words are located in the ascending memory locations aligned to even addresses: the low byte is at the even address, followed by the high byte at the next odd address.

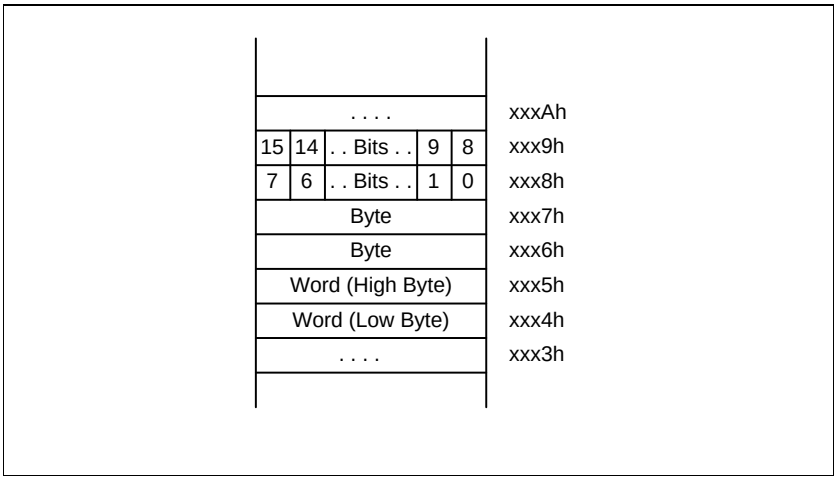


Figure 4.3: Bit, Byte and Word in a byte organized Memory

4.2 Internal ROM Organization

Various sizes of ROM up to 64K bytes are possible. The common address space is shared with special function registers, peripheral module registers, data and code memory. The special function registers and peripheral modules are mapped into the address range, starting with 0 and up to 01FFh. The remaining address space 0200h to 0FFFFh is shared by data and code memory.

The start address for all different sizes of ROM is at the same address 0FFFEh. The interrupt vector table also starts with highest priority at this highest ROM word address. The program counter, and hence the flow of instructions, is in the opposite direction - from lower addresses towards higher addresses. The program counter is increased by two, four or six according to the address mode used - program flow control instruction Jumps, branches and calls excluded.

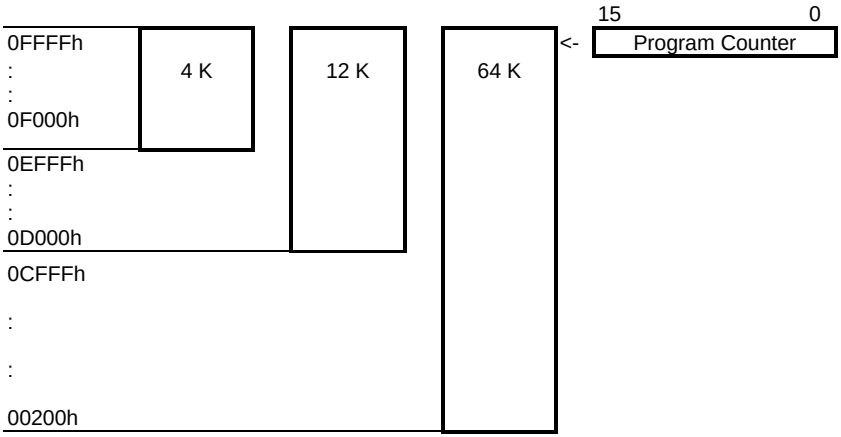


Figure 4.4: ROM Organization

The interrupt vectors and the power-up vector are located in the ROM, starting at address 0FFFEh. The vectors contain the 16-bit addresses of the appropriate interrupt handler instruction sequence.

4.2.1 Processing of ROM Tables

The MSP430 architecture allows the storage of large tables in the ROM. To access these tables, all word and byte instructions can be used. This offers various advantages with regard to flexible and ROM saving programming:

- Storage of an Output-PLA for display character conversion inside the ROM
- As many OPLA-terms as needed (no restriction on n terms)

- OTP version automatically includes OPLA programmability
- Computed table accesses (e.g. for a bar graph display)
- Table supported program flows.

The processing of tables is a very important feature, which allows very fast and clear programming. Especially for sensor applications, it is advantageous to have the sensor data in tables e.g. for linearization, compensation etc.

4.2.2 Computed Branches and Calls

Computed branches and subroutine calls are possible using standard instructions. The CALL and BR instructions use the same addressing modes as the other instructions (see programming examples).

The addressing modes allow indirect-indirect addressing, ideally suited for computed branches and calls. The full use of this programming technique permits a program structure different to conventional 8- and 16-bit controllers. A lot of routines can be handled easily using software status handling, instead of 'Flag' type program flow control.

The computed branches and subroutine calls are valid within a 64KB code segment.

4.3 RAM and Peripheral Organization

The entire RAM can be accessed in byte or word data, using the appropriate instruction suffix. The peripheral modules are located in two different address spaces:

- the special function registers are byte oriented by hardware and mapped into the address space from 0h up to 0Fh
- the peripheral modules that are byte oriented by hardware are mapped into the address space from 010h up to 0FFh
- and peripheral modules that are word oriented by hardware are mapped into the address space from 100h up to 01FFh

4.3.1 RAM

The RAM can be used for both code and data memory. Code accesses are always made on even byte addresses.

The suffix at the instruction memonic defines the access of the data as being word or byte data.

Example:

```
ADD.B    &TCDATA,TCSUM_L                        ;Byte acess
ADDC.B   TCSUM_H                                 Byte acess
ADD      R5,SUM_A     ≡     ADD.W    R5,SUM_A;  ;Word acess
ADDC     SUM_B        ≡     ADDC.W   SUM_A      ;Word acess
```

A Word consists of two bytes, a Highbyte (bit 15 to bit 8) and a Lowbyte (bit 7 to bit 0) and should always be aligned to even addresses.

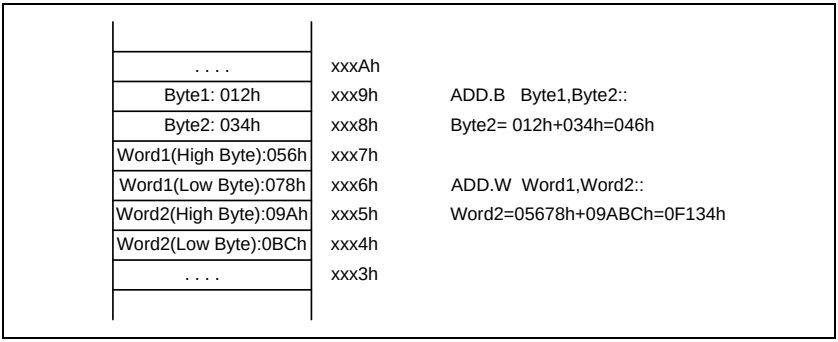


Figure 4.5: Byte and Word Operation

All operations on Stack and PC are word operations, and use even aligned memory addresses.

Word-to-word and byte-to-byte operations are performed completely correctly, both the results of the operation and the status bit information.

Word-word operation:

R5 = 0F28Eh
EDE .EQU 0212h
Mem(0F28Eh) = 0FFFEh
Mem(0212h) = 00112h

ADD @R5,&EDE

Mem(0212h) = 00110h
C = 1, Z = 0, N = 0

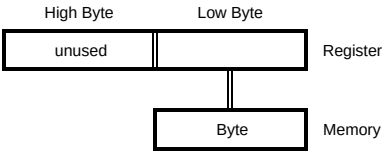
Byte-byte operation

R5 = 0223h
EDE .EQU 0202h
Mem(0223h) = 05Fh
Mem(0202h) = 043h

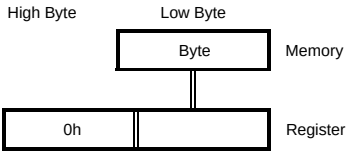
ADD.B @R5,&EDE

Mem(0202h) = 0A2h
C = 0, Z = 0, N = 1

Register-Byte operation:



Byte-Register operation:



Example Register-Byte operation	Example Byte-Register operation
R5 = 0A28Fh	R5 = 01202h
R6 = 0203h	R6 = 0223h
Mem(0203h) = 012h	Mem(0223h) = 05Fh
ADD.B R5,0(R6)	**ADD.B @R6,R5**
08Fh	05Fh
+ 012h	+ 002h ;Lowbyte of R5
0A1h	061h ;-> store into R5 -
Highbyte is 0	
Mem(0203h) = 0A1h	R5 = 061h
C = 0, Z = 0, N = 1	C = 0, Z = 0, N = 0
(Lowbyte of register)	(addressed byte)
+ (addressed byte)	+ (Lowbyte of register)
->(addressed byte)	->(Lowbyte of register, zero to Highbyte)

Note: Word-Byte operation

Word-Byte or Byte-Word operations on memory data are n o t supported. Each register-byte and byte-register operation is performed as a byte operation.

4.3.2 Peripheral Modules - Address Allocation

All peripheral modules are accessed and controlled by the software. All instructions are approved for the data interchange operation. Since there are modules physically using the MDB with its word construction, and modules that use only the eight least significant bits, the address space from 0100 to 01FFh is reserved for word modules and the address space from 00h to 0FFh is reserved for byte modules.

Peripheral modules mapped into the word address space should be accessed with word instructions (e.g. MOV R5,&WDTCTL). Peripheral modules mapped into the word address space should be accessed with byte instructions (MOV.B #1,&TCCTL).

The addressing of both is made via the absolute addressing mode, or via the 16-bit working registers, using the indexed, indirect or indirect autoincrement addressing mode.

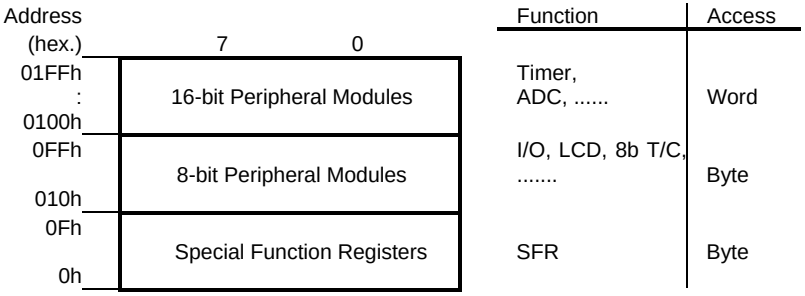


Figure 4.6: Example of RAM/peripheral organization

Word modules

Word modules are peripherals that are connected to the complete 16-bit MDB.

Access to word modules is always in word format, and byte access is not supported since the hardware is constructed for word operation only.

The peripheral file address space is organized in sixteen frames, and each frame represents eight words.

Address	**Description**
1F0h - 1FFh	reserved
1E0h - 1EFh	reserved
1D0h - 1DFh	reserved
1C0h - 1CFh	reserved
1B0h - 1BFh	reserved
1A0h - 1aFh	reserved
190h - 19Fh	reserved
180h - 18Fh	reserved
170h - 17Fh	**Timer_A**
160h - 16Fh	**Timer_A**
150h - 15Fh	reserved
140h - 14Fh	reserved
130h - 13Fh	**Multiplier**
120h - 12Fh	**Watchdog Timer**
110h - 11Fh	**Analog-to-Digital Converter**
100h - 10Fh	reserved

Figure 4.7: Peripheral File Address Map - Word Modules

Byte modules

Byte modules are peripherals that are connected to the reduced (eight LSB) MDB. The access to byte modules is always a byte access. The hardware in the peripheral byte modules takes the LowByte - the least significant bits - along with a write operation.

Byte instructions operate on byte modules without any restriction. Read access to the data of a peripheral byte module with word instructions results in unpredictable data on the Highbyte. Word data are written into a byte module by writing the LowByte to the appropriate peripheral register, and ignoring the HighByte.

The peripheral file address space is organized in sixteen frames.

Address	Description
00F0h - 00FFh	reserved
00E0h - 00EFh	reserved
00D0h - 00DFh	reserved
00C0h - 00CFh	reserved
00B0h - 00BFh	reserved
00A0h - 00AFh	reserved
0090h - 009Fh	reserved
0080h - 008Fh	reserved
0070h - 007Fh	**USART registers**
0060h - 006Fh	reserved
0050h - 005Fh	**System Clock Generator registers**
0040h - 004Fh	**Basic Timer, 8-bit Timer/Counter, Timer/Port registers**
0030h - 003Fh	**LCD registers**
0020h - 002Fh	**Digital I/O Port P3 and P4 control registers**
0010h - 001Fh	**Digital I/O Port P0, P1 and P2 control registers**
0000h - 000Fh	**Special Function Registers**

Figure 4.8: Peripheral File Address Map - Byte Modules

4.3.3 Peripheral Modules - Special Function Registers SFRs

The system configuration and the individual reaction of the peripheral modules to processor operation modes are mainly defined in Special Function Registers. The Special Function Registers are located in the lower address range, and are realized in **byte** manner. SFRs should be only accessed with byte instructions. Even if specific SFR bits share the same address space, they can be implemented physically within the associated module.

Address	Data Bus 7 … 0
000Fh	Not defined / implemented yet
000Eh	:
000Dh	:
000Ch	:
000Bh	:
000Ah	:
0009h	:
0008h	:
0007h	:
0006h	:
0005h	Module enable 2; ME2.2
0004h	Module enable 1; ME1.1
0003h	Interrupt flag reg. 2; IFG2.x
0002h	Interrupt flag reg. 1; IFG1.x
0001h	Interrupt enable 2; IE2.x
0000h	Interrupt enable 1; IE1.x

Figure 4.9: Special Function Register Address Map

The different devices of the MSP430 Family support SFRs with the correct logic and function within the individual modules. Each module can be enabled individually, to access the interrupt function and the operation. Full software control of these configuration bits enables the application software to react to system requirements on interrupt enable mask.

The power consumption of the system is influenced by the number of the enabled modules, and their function. Disabling a module from the actual operation mode reduces power consumption while other parts of the controller remain fully active. Two parts can not be disabled: ROM and RAM. The processor core can be switched to disabled mode - CPUOff Mode - with all internal functions disabled: CPU and bus activities are stopped.