Section 3. Data Memory

HIGHLIGHTS

This section of the manual contains the following topics:

Note: This family reference manual section is meant to serve as a complement to device data sheets. Depending on the device variant, this manual section may not apply to all dsPIC33E/PIC24E devices.

Please consult the note at the beginning of the **"Data Memory"** chapter in the current device data sheet to check whether this document supports the device you are using.

Device data sheets and family reference manual sections are available for download from the Microchip Worldwide Web site at: http://www.microchip.com

3.1 INTRODUCTION

The dsPIC33E/PIC24E data width is 16 bits. All internal registers and data space memory are organized as 16 bits wide. The data spaces can be accessed as one 64 Kbyte linear address range or 4 Mbyte pseudo-linear paged address range. The data memory is accessed using Address Generation Units (AGUs) and separate data paths.

3.2 DATA SPACE

Data memory addresses between 0x0000 and 0x0FFF are reserved for the device Special Function Registers (SFRs). The SFRs include control and status bits for the CPU and peripherals on the device.

An example data space memory map is shown in Figure 3-1.

The RAM begins at address 0x1000 and is split into two blocks, X and Y data space. For data writes, the X and Y data memory spaces are always accessed as a single, linear data space. For data reads, the X and Y data memory spaces can be accessed independently or as a single, linear space. Data reads for MCU class instructions always access the X and Y data spaces as a single combined data space. Dual source operand DSP instructions, such as the `MAC` instruction, access the X and Y data spaces separately to support simultaneous reads for the two source operands.

MCU instructions can use any W register as an address pointer for a data read or write operation.

During data reads, the DSP class of instructions isolates the Y address space from the total data space. W10 and W11 are used as address pointers for reads from the Y data space. The remaining data space is referred to as X space, but could more accurately be described as "X minus Y" space. W8 and W9 are used as address pointers for data reads from the X data space in DSP class instructions.

Figure 3-2 shows how the data memory map functions for both MCU class and DSP class instructions. Note that it is the W register number and type of instruction that determines how address space is accessed for data reads. In particular, MCU instructions treat the X and Y memory as a single combined data space. The MCU instructions can use any W register as an address pointer for reads and writes in combination with read and write data space page registers. The DSP instructions that can simultaneously pre-fetch two data operands, split the data memory into two spaces. Specific W registers must be used for read address pointers in this case.

Some DSP instructions have the ability to store the accumulator that is not targeted by the instruction to data memory. This function is called "accumulator write back". W13 must be used as an address pointer to the combined data memory space for accumulator write back operations.

For DSP class instructions, W8 and W9 should point to implemented X memory space for all memory reads. If W8 or W9 points to Y memory space, zeros will be returned. If W8 or W9 points to an unimplemented memory address, an address error trap will be generated.

For DSP class instructions, W10 and W11 should point to implemented Y memory space for all memory reads. If W10 or W11 points to implemented X memory space, all zeros will be returned. If W10 or W11 points to an unimplemented memory address, an address error trap will be generated. For additional information on address error traps, refer to **Section 6. "Interrupts"** (DS70602).

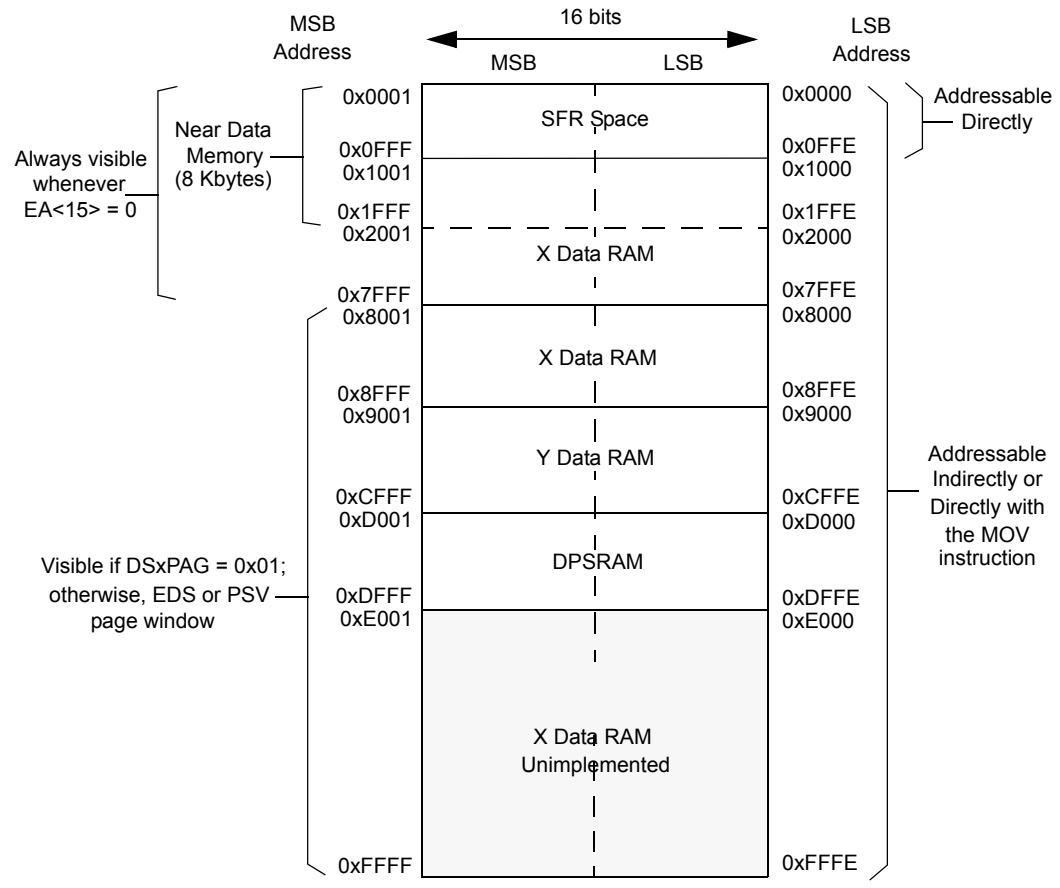
Note:	The data memory map and the partition between the X and Y data spaces is device specific. Refer to the specific dsPIC33E/PIC24E device data sheet for further details.

In addition, some dsPIC33E/PIC24E devices contain DMA and dual-ported SRAM memory (DPSRAM). Both the CPU and DMA controller can write and read to/from addresses within the DPSRAM without interference, such as CPU stalls, resulting in maximized, real-time performance. The DMA can address all of the data memory space including the Extended Data Space (EDS), excluding the SFR space. Refer to **Section 22. "Direct Memory Access (DMA)"** (DS70348) for more information.

Note:	The presence and size of DPSRAM is device specific. Refer to the specific dsPIC33E/PIC24E device data sheet for further details.

The data memory space is extended to support additional RAM and an optional external bus interface. The extended data memory space is a pseudo-linear address space across DS, EDS and Program Space Visibility (PSV) spaces.

Figure 3-1: Example Data Memory Map



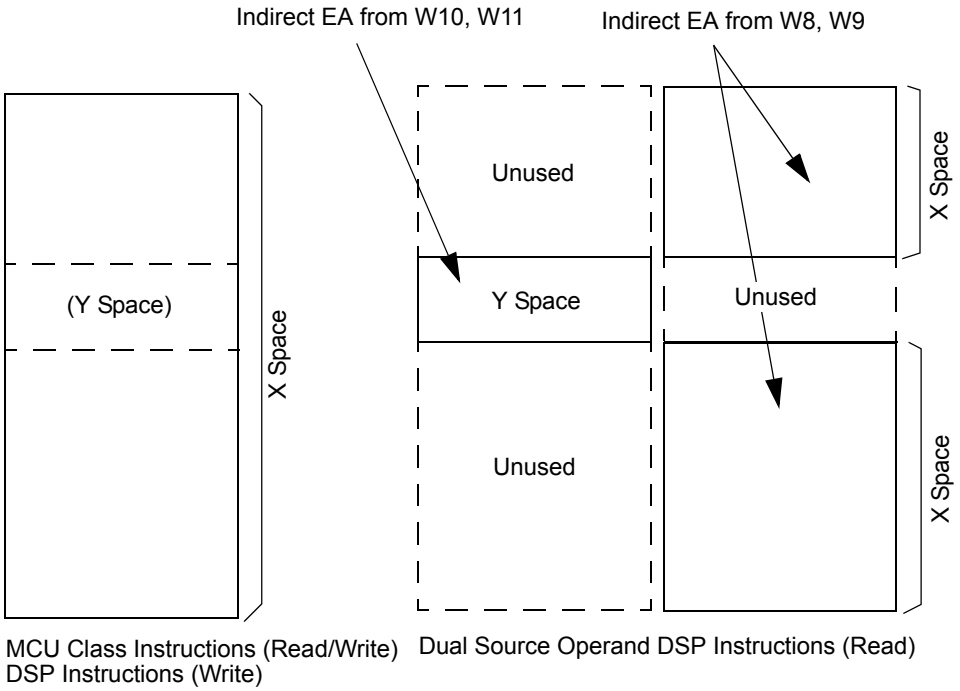
Note 1: The size of the X and Y data spaces is device specific. Refer to the appropriate device data sheet for further details. The data space boundaries indicated here are used for example purposes only.

2: Near data memory can be accessed directly via file register instructions that encode a 13-bit address into the opcode.

3: All data memory can be accessed indirectly via W registers or directly using the `MOV` instruction.

4: Upper half of data memory map can be mapped into a segment of program memory space for PSV.

Figure 3-2: Data Spaces for MCU and DSP Instructions



Note: Data writes for DSP instructions consider the entire data memory as one combined space. DSP instructions that perform an accumulator write back use W13 as an address pointer for writes to the combined data spaces.

3.2.1 Near Data Memory

An 8 Kbyte address space, referred to as near data memory, is reserved in the data memory space between 0x0000 and 0x1FFF. Near data memory is directly addressable through a 13-bit absolute address field within all file register instructions.

The memory regions included in the near data region will depend on the amount of data memory implemented for each dsPIC33E/PIC24E device variant. At a minimum, the near data region will include all of the SFRs and some of the X data memory. For devices that have smaller amounts of data memory, the near data region can include all of X memory space and possibly some or all of Y memory space. Refer to Figure 3-1 for more details.

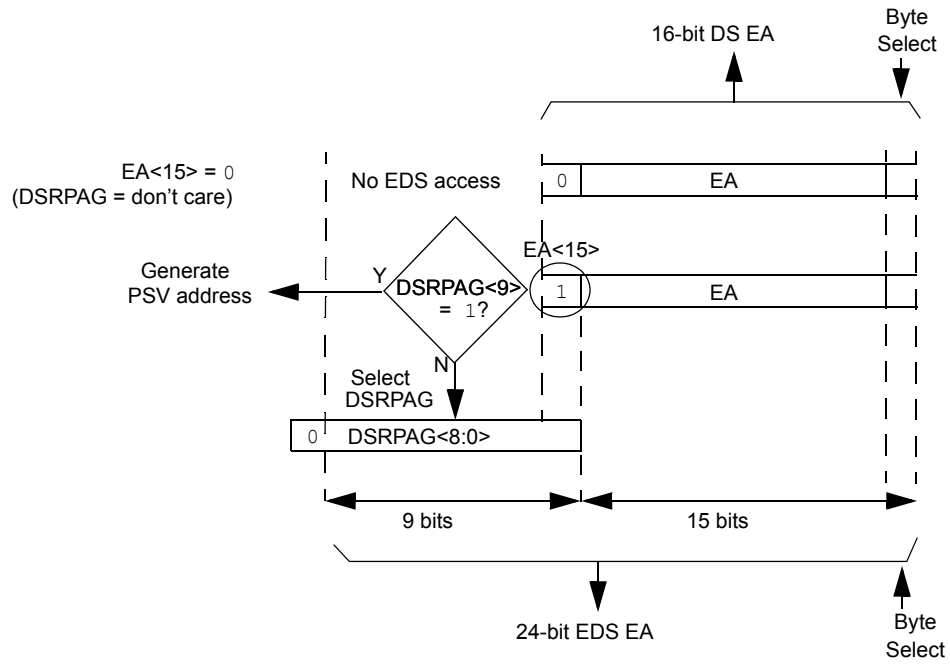
Note: The entire 64K data space can be addressed directly using the `MOV` instruction. Refer to the *"16-bit MCU and DSC Programmer's Reference Manual"* (DS70157) for further details.

3.2.2 Paged Memory Scheme

The dsPIC33E/PIC24E architecture extends the available data space through a paging scheme, which allows the available data space to be accessed using `MOV` instructions in a linear fashion for pre- and post-modified effective addresses (EAs). The base data space address pointer is used in conjunction with the data space page registers, the 10-bit read page register (DSRPAG) or the 9-bit write page register (DSWPAG), to form an EDS address or PSV address. The data space page registers are located in the SFR space.

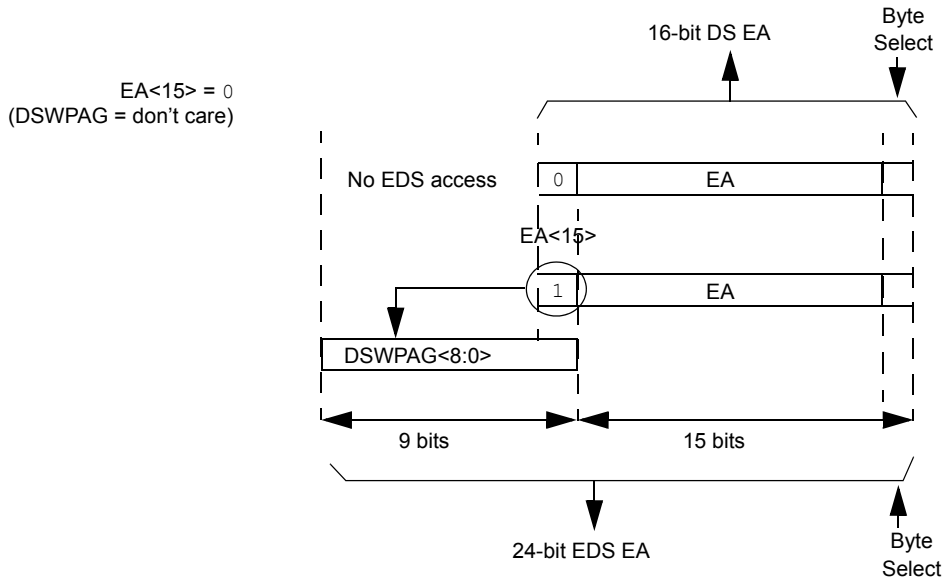
Construction of the EDS address is shown in Figure 3-3 and Figure 3-4. When DSRPAG<9> = `0` and base address bit, EA<15> = `1`, DSRPAG<8:0> is concatenated onto EA<14:0> to form the 24-bit EDS read address. Similarly, when base address bit, EA<15> = `1`, DSWPAG<8:0> is concatenated onto EA<14:0> to form the 24-bit EDS write address.

Figure 3-3: Extended Data Space (EDS) Read Address Generation



Note: DS read access when DSRPAG = 0x000 will force an Address Error trap.

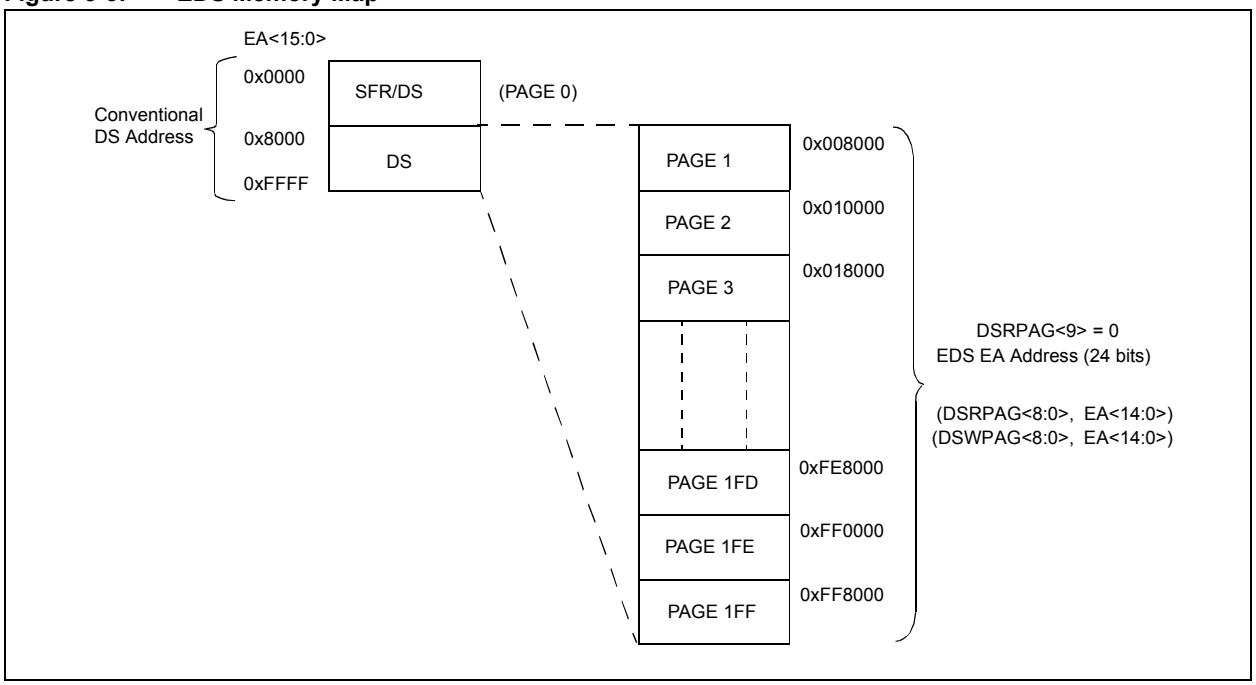
Figure 3-4: Extended Data Space (EDS) Write Address Generation



Note: DS write access when DSWPAG = 0x000 will force an Address Error trap.

The paged memory scheme provides access to multiple 32 Kbyte windows in the EDS and PSV memory. The data space page registers DSxPAG, in combination with the upper half of data space address can provide up to 16 Mbytes of additional address space in the EDS and 8 Mbytes (DSRPAG only) of PSV address space. The EDS memory map is shown in Figure 3-5.

Figure 3-5: EDS Memory Map



Note: The program space (PS) can be read using DSRPAG values of 0x200 or greater. Writes to PS are not supported, so DSWPAG is dedicated to EDS writes only. Refer to **Section 4.0 "Program Memory"** (DS70613) for details on PSV access.

The usage of the page registers (DSRPAG and DSWPAG) for read and write access allows data movement between different pages in data memory. This is accomplished by setting the DSRPAG register value to the page from which you want to read, and configuring the DSWPAG register to the page to which it needs to be written. Data can also be moved from different PSV to EDS pages, by configuring the DSRPAG and DSWPAG registers to address PSV and EDS space, respectively. The data can be moved between pages by a single instruction.

Example 3-1, Example 3-2 and Example 3-3 provide code examples for managing EDS access.

Example 3-1: Compiler Managed EDS Access

```
#include <p33Exxxx.h>

_FOSCSEL(FNOSC_FRC);
_FOSC(FCKSM_CSECMD & OSCIOFNC_OFF & POSCMD_NONE);
_FWDT(FWDTEN_OFF);

__eds__ int __attribute__((space(xmemory),eds)) m[5] = {1, 2, 3, 4, 5};
__eds__ int __attribute__((space(ymemory),eds)) x[5] = {10, 20, 30, 40, 50};
int sum;

int vectorDotProduct (int *, int *);

int main(void)
{
    // Compiler-managed EDS accesses
    sum = vectorDotProduct (__eds__ int *,__eds__ int *);

    while(1);
}

int vectorDotProduct (__eds__ int * m,__eds__ int * x)
{
    int i, sum = 0;

    for (i = 0; i < 5; i ++)
        sum += (*m++) * (*x++);
    return (sum);
}
```

Example 3-2: User Managed EDS Access

```
#include <p33Exxxx.h>

_FOSCSEL(FNOSC_FRC);
_FOSC(FCKSM_CSECMD & OSCIOFNC_OFF & POSCMD_NONE);
_FWDT(FWDTEN_OFF);

int m[5] = {1, 2, 3, 4, 5};
int __attribute__((space(xmemory),eds)) m1[5] = {2, 4, 6, 8, 10};
int __attribute__((space(ymemory),eds)) m2[5] = {3, 6, 9, 12, 15};
int x[5] = {10, 20, 30, 40, 50};
int sum[5];
int __attribute__((space(xmemory),eds)) sum1[5];
int __attribute__((space(ymemory),eds)) sum2[5];

void vectorMultiply (int *, int *, int *);

int main(void)
{
    int temp1, temp2;

    // Save original EDS page values
    temp1 = DSRPAG;
    temp2 = DSWPAG;

    DSRPAG = __builtin_edspage (m1);
    DSWPAG = __builtin_edspage (sum1);
    vectorMultiply ((int *) m1, x, (int *) sum1);

    DSRPAG = __builtin_edspage (m2);
    DSWPAG = __builtin_edspage (sum2);
    vectorMultiply ((int *) m2, x, (int *) sum2);

    // Restore original EDS page values
    DSRPAG = temp1;
    DSWPAG = temp2;

    vectorMultiply ((int *) m, x, (int *) sum);

    while(1);
}

void vectorMultiply (int *m, int *x, int *sum)
{
    int i;

    for (i = 0; i < 5; i ++)
        (*sum++) = (*m++) * (*x++);
}
```

Example 3-3: EDS Code Example in Assembly

```
    .section .data, eds
    .global fib_data
    .global fib_sum
fib_data:
    .word 0, 1, 2, 3, 5, 8, 13
fib_sum:
    .space 2

    ; Start of code section
    .text

    .global _main
_main:

    ; Set DSRPAG to the page that contains the “fib_data” array
    MOVPAG #edspage(fib_data), DSRPAG

    ; Set DSWPAG to the page that contains the “fib_sum” array
    MOVPAG #edspage(fib_sum), DSWPAG

    ; Set up W0 as a pointer to “fib_data”
    MOV #edsoffset(fib_data), W0

    ; Set up W1 as a pointer to “fib_sum”
    MOV #edsoffset(fib_sum), W1

    ; Clear register W2 as it will contain a running sum
    CLR     W2

    ; Add all the data values in register W2
    REPEAT #6
    ADD W2, [W0++], W2

    ; Store the final result in variable “fib_sum”
    MOV     W2, [W1]

done:
    BRA done

    RETURN
```

When an EDS or PSV page overflow or underflow occurs, the register indirect EA calculation results in clearing EA<15>. An overflow or underflow of the EA in the EDS or PSV pages can occur at the page boundaries when:

- the initial address, prior to modification, addresses an EDS or PSV page
- the EA calculation uses pre- or post-modified register indirect addressing. However, this does not include register offset addressing.

In general, when an overflow is detected, the DSxPAG register is incremented, and the resultant EA<15> bit is set to keep the base address within the EDS or PSV window. When an underflow is detected, the DSxPAG register is decremented, and the EA<15> bit is set to keep the base address within the EDS or PSV window. This creates a linear EDS and PSV address space, but only when using register indirect addressing modes.

Exceptions to the operation described above arise when entering and exiting the boundaries of page 0, EDS and PSV spaces. Table 3-1 shows the effects of overflow and underflow scenarios at different boundaries.

In the following cases, when overflow or underflow occurs, the EA<15> bit is set and the DSxPAG is not modified; therefore, the EA will wrap to the beginning of the current page:

- Register indirect with register offset addressing
- Modulo addressing
- Bit-reversed addressing
- The table address used in any TBLRDx or TBLWTx operations (source address in TBLRDx and the destination address in TBLWTx)

Table 3-1: Overflow and Underflow Scenarios at Page 0, EDS and PSV Space Boundaries

O/U, R/W	Operation	Before			After		
		DSxPAG	DS EA<15>	Page Description	DSxPAG	DS EA<15>	Page Description
O, Read	[++Wn] or [Wn++]	DSRPAG = 0x1FF	1	EDS: Last page	DSRPAG = 0x1FF	0	See **Note 1**
O, Read		DSRPAG = 0x2FF	1	PSV: Last lsw page	DSRPAG = 0x300	1	PSV: First MSB page
O, Read		DSRPAG = 0x3FF	1	PSV: Last MSB page	DSRPAG = 0x3FF	0	See **Note 1**
O, Write		DSWPAG = 0x1FF	1	EDS: Last page	DSWPAG = 0x1FF	0	See **Note 1**
U, Read	[--Wn] or [Wn--]	DSRPAG = 0x001	1	EDS page	DSRPAG = 0x001	0	See **Note 1**
U, Read		DSRPAG = 0x200	1	PSV: First lsw page	DSRPAG = 0x200	0	See **Note 1**
U, Read		DSRPAG = 0x300	1	PSV: First MSB page	DSRPAG = 0x2FF	1	PSV: Last lsw page

Legend: O = Overflow, U = Underflow, R = Read, W = Write

Note 1: The register indirect address now addresses a location in the base data space (0x0000-0x8000).

2: An EDS access with DSxPAG = 0x000 will generate an address error trap.

3: Only reads from PS are supported using DSRPAG. An attempt to write to PS using DSWPAG will generate an address error trap.

4: Pseudo-linear addressing is not supported for large offsets.

3.2.3 Extended X Data Space

The lower half of the base address space range between 0x0000 and 0x7FFF is always accessible regardless of the contents of the data space page registers. It is indirectly addressable through the register indirect instructions. It can be regarded as being located in the default EDS page 0 (i.e., EDS address range of 0x000000 to 0x007FFF with the base address bit EA<15> = 0 for this address range). However, page 0 cannot be accessed through upper 32 Kbytes, 0x8000 to 0xFFFF, of base data space in combination with DSRPAG = 0x00 or DSWPAG = 0x00. Consequently, DSRPAG and DSWPAG are initialized to 0x001 at Reset.

Note 1: DSxPAG should not be used to access page 0. An EDS access with DSxPAG set to 0x000 will generate an Address Error trap.

2: Clearing the DSxPAG in software has no effect.

The remaining pages including both EDS and PSV pages are only accessible using the DSRPAG or DSWPAG registers in combination with the upper 32 Kbytes, 0x8000 to 0xFFFF, of the base address, where base address bit EA<15> = 1.

For example, when DSRPAG = 0x01 or DSWPAG = 0x01, accesses to the upper 32 Kbytes, 0x8000 to 0xFFFF, of the data space will map to the EDS address range of 0x008000 to 0x00FFFF. When DSRPAG = 0x02 or DSWPAG = 0x02, accesses to the upper 32 Kbytes of the data space will map to the EDS address range of 0x010000 to 0x017FFF and so on, as shown in the EDS memory map in Figure 3-5.

3.2.4 EDS Arbitration and Bus Master Priority

EDS accesses from bus masters in the system are arbitrated.

The arbiter for data memory (including EDS) arbitrates between the CPU, the DMA, and up to three other bus masters, one of which will be the ICD module. In the event of coincidental access to a bus by the bus masters, the arbiter determines which bus master access has the highest priority. The other bus masters are suspended and processed after the access of the bus by the bus master with the highest priority.

Note: DMA accesses to DPSRAM are not subject to arbitration, and occur without any stalls.

By default, the CPU is bus master 0 (M0) with the highest priority, and the ICD is bus master 4 (M4) with the lowest priority. The remaining bus masters (USB and DMA Controllers) are allocated to M2 and M3, respectively (in this default configuration, M1 is reserved and cannot be used). The user application may raise or lower the priority of the masters to be above that of the CPU by setting the appropriate bits in the EDS Bus Master Priority Control (MSTRPR) register. All bus masters with raised priorities will maintain the same priority relationship relative to each other (i.e., M1 being highest and M3 being lowest with M2 in between). Also, all the bus masters with priorities below that of the CPU maintain the same priority relationship relative to each other. The priority schemes for bus masters with different MSTRPR values are tabulated in Table 3-2.

This bus master priority control allows the user application to manipulate the real-time response of the system, either statically during initialization, or dynamically in response to real-time events.

Figure 3-6: Arbiter Architecture

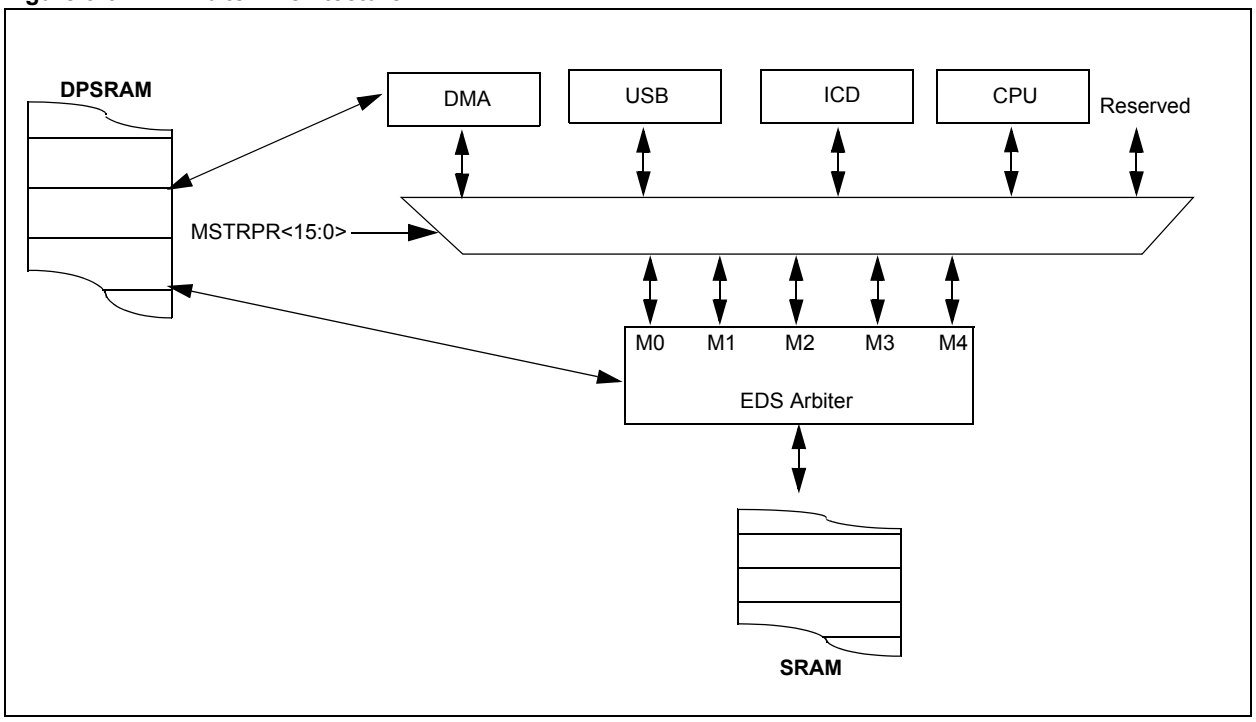


Table 3-2: EDS Bus Arbiter Priority

Priority	MSTRPR<15:0> Bit Setting			
	0x0000	0x0008	0x0020	0x0028
M0 (highest)	CPU	USB	DMA	USB
M1	Reserved	CPU	CPU	DMA
M2	USB	Reserved	Reserved	CPU
M3	DMA	DMA	USB	Reserved
M4 (lowest)	ICD	ICD	ICD	ICD

Note: All other values of MSTRPR<15:0> are reserved.

3.3 DATA SPACE ADDRESS GENERATION UNITS (AGUs)

The dsPIC33E/PIC24E contains an X AGU and a Y AGU for generating data memory addresses. Both X and Y AGUs can generate any Effective Address (EA) within the available data memory range. However, EAs that are outside the physical memory provided return all zeros for data reads and data writes to those locations, have no effect. Furthermore, an address error trap will be generated. For more information on address error traps, refer to **Section 6. "Interrupts"** (DS70602).

3.3.1 X Address Generation Unit

The X AGU is used by all instructions and supports all addressing modes. The X AGU consists of a read AGU (X RAGU) and a write AGU (X WAGU), which operate independently on separate read and write buses during different phases of the instruction cycle. The X read data bus is the return data path for all instructions that view data space as combined X and Y address space. It is also the X address space data path for the dual operand read instructions (DSP instruction class). The X write data bus is the only write path to the combined X and Y data space for all instructions.

The X AGU supports pseudo-linear addressing through the base data space address range into all of the EDS and PSV address space. It can therefore generate EAs within the range 0x000000 to 0xFFFFFF.

The X RAGU starts its effective address calculation during the prior instruction cycle, using information derived from the just pre-fetched instruction. The X RAGU EA is presented to the address bus at the beginning of the instruction cycle.

The X WAGU starts its effective address calculation at the beginning of the instruction cycle. The EA is presented to the address bus during the write phase of the instruction.

Both the X RAGU and the X WAGU support modulo addressing.

Bit-reversed addressing is supported by the X WAGU only.

3.3.2 Y Address Generation Unit (dsPIC33E Devices Only)

The Y data memory space has one AGU that supports data reads from the Y data memory space. The Y memory bus is never used for data writes. The function of the Y AGU and Y memory bus is to support concurrent data reads for DSP class instructions.

The Y AGU can only generate an EA within the base data space address range 0x0000 to 0xFFFF and cannot extend into EDS.

The Y AGU timing is identical to that of the X RAGU, in that its effective address calculation starts prior to the instruction cycle, using information derived from the pre-fetched instruction. The EA is presented to the address bus at the beginning of the instruction cycle.

The Y AGU supports Modulo Addressing and Post-modification Addressing modes for the DSP class of instructions that use it.

Note:	The Y AGU does not support data writes. All data writes occur via the X WAGU to the combined X and Y data spaces. The Y AGU is only used during data reads for dual source operand DSP instructions.

3.3.3 Address Generation Units and DSP Class Instructions (dsPIC33E Devices Only)

The Y AGU and Y memory data path are used in concert with the X RAGU by the DSP class of instructions to provide two concurrent data read paths. For example, the MAC instruction can simultaneously pre-fetch two operands to be used in the next multiplication.

The DSP class of instructions dedicates two W register pointers, W8 and W9, to always operate through the X RAGU and address X data space independently from Y data space, plus two W register pointers, W10 and W11, to always operate through the Y AGU and address Y data space independently from X data space. Any data write performed by a DSP class instruction takes place in the combined X and Y data space and the write occurs across the X-bus. Consequently, the write can be to any address regardless of where the EA is directed.

The Y AGU only supports Post-modification Addressing modes associated with the DSP class of instructions. For more information on Addressing modes, please refer to the *"16-bit MCU and DSC Programmer's Reference Manual"* (DS70157). The Y AGU also supports modulo addressing for automated circular buffers. All other (MCU) class instructions can access the Y data address space through the X AGU when it is regarded as part of the composite linear space.

3.3.4 Data Alignment

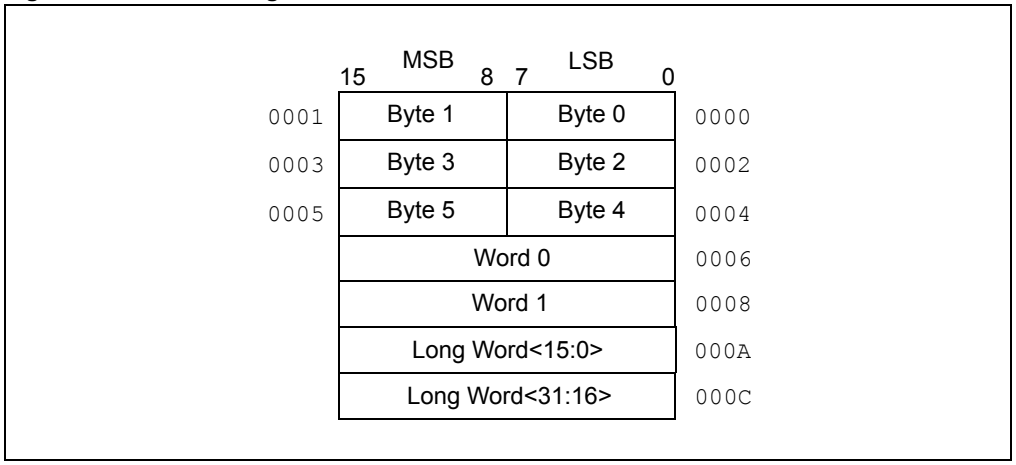
The ISA supports both word and byte operations for all MCU instructions that access data through the X memory AGU. The Least Significant bit (LSb) of a 16-bit data address is ignored for word operations. Word data is aligned in the little-endian format with the Least Significant Byte (LSB) at the even address (LSb = 0) and the Most Significant Byte (MSB) at the odd address (LSb = 1).

For byte operations, the LSb of the data address is used to select the byte that is accessed. The addressed byte is placed on the lower 8 bits of the internal data bus.

All effective address calculations are automatically adjusted depending on whether a byte or a word access is performed. For example, an address is incremented by 2 for a word operation that post-increments the address pointer.

Note:	All word accesses must be aligned to an even address (LSb = 0). Misaligned word data fetches are not supported, so care must be taken when mixing byte and word operations or translating code from existing 8-bit PIC® microcontrollers. Should a misaligned word read or write be attempted, an address error trap will occur. A misaligned read data will be discarded and a misaligned write will not take place. The trap will then be taken, allowing the system to examine the machine state prior to execution of the address Fault.

Figure 3-7: Data Alignment



3.4 MODULO ADDRESSING (dsPIC33E DEVICES ONLY)

Modulo, or circular addressing provides an automated means to support circular data buffers using hardware. The objective is to remove the need for software to perform data address boundary checks when executing tightly looped code as is typical in many DSP algorithms.

Any W register, except W15, can be selected as the pointer to the modulo buffer. The modulo hardware performs boundary checks on the address held in the selected W register and automatically adjusts the pointer value at the buffer boundaries, when required.

dsPIC33E/PIC24E modulo addressing can operate in either data or program space (since the data pointer mechanism is essentially the same for both). One circular buffer can be supported in each of the X (which also provides the pointers into Program space) and Y data spaces.

The modulo data buffer length can be any size up to 32K words. The modulo buffer logic supports buffers using word or byte-sized data. However, the modulo logic only performs address boundary checks at word address boundaries, so the length of a byte modulo buffer must be even. In addition, byte-sized modulo buffers cannot be implemented using the Y AGU because byte access is not supported via the Y memory data bus.

3.4.1 Modulo Start and End Address Selection

Four address registers are available for specifying the modulo buffer start and end addresses:

- XMODSRT: X AGU Modulo Addressing Start Register
- XMODEND: X AGU Modulo Addressing End Register
- YMODSRT: Y AGU Modulo Addressing Start Register
- YMODEND: Y AGU Modulo Addressing End Register

The start address for a modulo buffer must be located at an even byte address boundary. The LSb of the XMODSRT and YMODSRT registers is fixed at '0' to ensure the correct modulo start address. The end address for a modulo buffer must be located at an odd byte address boundary. The LSb of the XMODEND and YMODEND registers is fixed to '1' to ensure the correct modulo end address.

The start and end address selected for each modulo buffer have certain restrictions, depending on whether an incrementing or decrementing buffer is to be implemented. For an incrementing buffer, a W register pointer is incremented through the buffer address range. When the end address of the incrementing buffer is reached, the W register pointer is reset to point to the start of the buffer. For a decrementing buffer, a W register pointer is decremented through the buffer address range. When the start address of a decrementing buffer is reached, the W register pointer is reset to point to the end of the buffer.

Note:	The user must decide whether an incrementing or decrementing modulo buffer is required for the application. Certain address restrictions depend on whether an incrementing or decrementing modulo buffer is to be implemented.

3.4.1.1 MODULO START ADDRESS

The data buffer start address is arbitrary, but must be at a 'zero' power of two boundary for incrementing modulo buffers. The modulo start address can be any value for decrementing modulo buffers and is calculated using the chosen buffer end address and buffer length.

For example, if the buffer length for an incrementing buffer is chosen to be 50 words (100 bytes), then the buffer start byte address must contain 7 Least Significant zeros. Valid start addresses may, therefore, be 0xNN00 and 0xNN80, where 'N' is any hexadecimal value.

3.4.1.2 MODULO END ADDRESS

The data buffer end address is arbitrary but must be at a 'ones' boundary for decrementing buffers. The modulo end address can be any value for an incrementing buffer and is calculated using the chosen buffer start address and buffer length.

For example, if the buffer size (modulus value) is chosen to be 50 words (100 bytes), the buffer end byte address for decrementing modulo buffer must contain 7 Least Significant ones. Valid end addresses can, therefore, be 0xNNFF and 0xNN7F, where 'N' is any hexadecimal value.

Note: If the required modulo buffer length is an even power of 2, the modulo start and end addresses that are chosen must satisfy the requirements for incrementing and decrementing buffers.

3.4.1.3 MODULO ADDRESS CALCULATION

The end address for an incrementing modulo buffer must be calculated from the chosen start address and the chosen buffer length in bytes. Equation 3-1 can be used to calculate the end address.

Equation 3-1: Modulo End Address for Incrementing Buffer

$$End\ Address = Start\ Address + Buffer\ Length - 1$$

The start address for a decrementing modulo buffer is calculated from the chosen end address and the buffer length, as shown in Equation 3-2.

Equation 3-2: Modulo Start Address for Decrementing Buffer

$$Start\ Address = End\ Address - Buffer\ Length + 1$$

3.4.1.4 DATA DEPENDENCIES ASSOCIATED WITH MODULO ADDRESSING SFRs

A write operation to the Modulo and Bit-Reversed Addressing Control (MODCON) register, should not be immediately followed by an indirect read operation using any W register. The code segment shown in Example 3-4 will thus lead to unexpected results.

Note 1: Using a `POP` instruction to pop the contents of the top-of-stack (TOS) location into the MODCON register, also constitutes a write to MODCON register. The instruction immediately following a write to MODCON cannot be any instruction performing an indirect read operation.

2: Some instructions perform an indirect read operation, implicitly. These are: `POP`, `RETURN`, `RETFIE`, `RETLW` and `ULNK`.

Example 3-4: Incorrect MODCON Initialization

```
MOV #0x8FF4, w0     ;Initialize MODCON
MOV w0, MODCON
MOV [w1], w2        ;Incorrect EA generated here
```

Instead, use any Addressing mode other than indirect reads in the instruction that immediately follows the initialization of MODCON. Alternately, add a `NOP` instructions after initializing the MODCON register, as shown in Example 3-5.

Example 3-5: Correct MODCON Initialization

```
MOV #0x8FF4, w0     ;Initialize MODCON
MOV w0, MODCON
NOP                 ;See Note below
MOV [w1], w2        ;Correct EA generated here
```

Note: Alternately, execute other instructions that do not perform indirect read operations, using the W register designated for modulo buffer access.

An additional condition exists for indirect read operations performed immediately after writing to the modulo address SFRs:

- XMODSRT: X AGU Modulo Addressing Start Register
- XMODEND: X AGU Modulo Addressing End Register
- YMODSRT: Y AGU Modulo Addressing Start Register
- YMODEND: Y AGU Modulo Addressing End Register

If modulo addressing has already been enabled in MODCON, then a write to the X (or Y) modulo address SFRs should not be immediately followed by an indirect read, using the W register designated for modulo buffer access from X-data space (or Y-data space). The code segment in Example 3-6 shows how initializing the modulo SFRs associated with the X-data space, could lead to unexpected results. A similar example can be made for initialization in Y-data space.

Example 3-6: Incorrect Modulo Addressing Setup

```
MOV #0x8FF4,w0      ;Modulo addressing enabled
MOV w0, MODCON      ;in X-data space using w4
                    ;for buffer access
MOV #0x1200,w4      ;XMODSRT is initialized
MOV w4, XMODSRT
MOV #0x12FF,w0      ;XMODEND is initialized
MOV w0, XMODEND
MOV [w4++], w5      ;Incorrect EA generated
```

To avoid this condition, insert a NOP, or perform any operation other than an indirect read that uses the W register designated for modulo buffer access, after initializing the modulo address SFRs. This is demonstrated in Example 3-7. Another alternative would be to enable modulo addressing in MODCON after initializing the modulo start and end address SFRs.

Example 3-7: Correct Modulo Addressing Setup

```
MOV #0x8FF4,w0      ;Modulo addressing enabled
MOV w0, MODCON      ;in X-data space using w4
                    ;for buffer access
MOV #0x1200,w4      ;XMODSRT is initialized
MOV w4, XMODSRT
MOV #0x12FF,w0      ;XMODEND is initialized
MOV w0, XMODEND
NOP                 ;See Note below
MOV [w4++], w5      ;Correct EA generated here
```

Note: Alternately, execute other instructions that do not perform indirect read operations, using the W register designated for modulo buffer access.

3.4.2 W Address Register Selection

The X address space pointer W register to which modulo addressing is to be applied, is stored in the XWM bits of the Modulo and Bit-Reversed Addressing Control register (MODCON<3:0>). The XMODSRT, XMODEND and the XWM register selection are shared between the X RAGU and X WAGU. Modulo addressing is enabled for X data space when XWM<3:0> is set to any value other than 15 and the XMODEN bit (MODCON<15>) is set. W15 cannot be used as the pointer for modulo addressing because it is the dedicated software stack pointer.

The Y address space pointer W register to which modulo addressing is to be applied, is stored in the YWM bits in the Modulo and Bit-Reversed Addressing Control register (MODCON<7:4>). Modulo addressing is enabled for Y data space when YWM<3:0> is set to any value other than 15 and the YMODEN bit (MODCON<14>) is set.

Note: A write to the MODCON register should not be followed by an instruction that performs an indirect read operation using a W register. Unexpected results may occur. Some instructions perform an implicit indirect read. These are: POP, RETURN, RETFIE, RETLW and ULNK.

3.4.3 Modulo Addressing Applicability

Modulo addressing can be applied to the effective address calculation associated with the selected W register. It is important to realize that the address boundary tests look for addresses equal to or greater than the upper address boundary for incrementing buffers and equal to or less than the lower address boundary for decrementing buffers. Address changes can, therefore, jump over boundaries and still be adjusted correctly. The automatic adjustment of the W register pointer by the modulo hardware is unidirectional. That is, the W register pointer may not be adjusted correctly by the modulo hardware when the W register pointer for an incrementing buffer is decremented and vice versa. The exception to this rule is when the buffer length is an even power of 2 and the start and end addresses can be chosen to meet the boundary requirements for both incrementing and decrementing modulo buffers.

A new EA can exceed the modulo buffer boundary by up to the length of the buffer and still be successfully corrected. This is important to remember when the Register Indexed (`[Wb + Wn]`) and Literal Offset (`[Wn + lit10]`) Addressing modes are used. In addition, the Register Indexed and Literal Offset Addressing modes do not change the value held in the W register. Only the indirect with Pre- and Post-modification Addressing modes (`[Wn++]`, `[Wn--]`, `[++Wn]`, `[--Wn]`) will modify the W register address value.

Modulo addressing operates within any EDS or PSV memory space. However, it will not operate across page boundaries. The only exception to this rule is when crossing a boundary into or out of page 0.

3.4.4 Modulo Addressing Initialization for Incrementing Modulo Buffer

The following steps describe the setup procedure for an incrementing circular buffer. The steps are similar whether the X AGU or Y AGU is used.

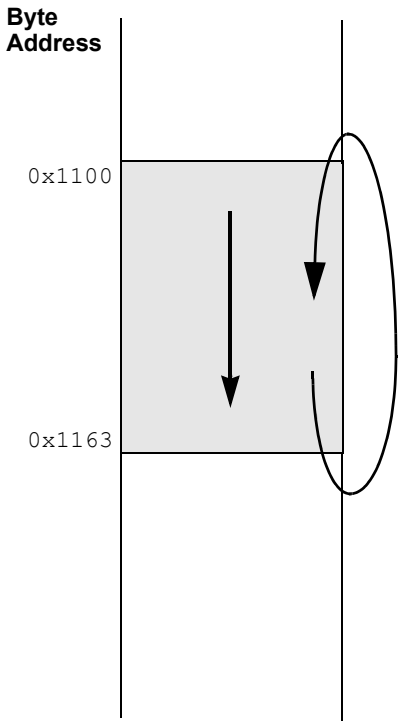
1. Determine the buffer length in 16-bit data words. Multiply this value by 2 to get the length of the buffer in bytes.
2. Select a buffer starting address that is located at a binary ‘zeros’ boundary based on the desired length of the buffer. Remember that the buffer length in words must be multiplied by 2 to obtain the byte address range. For example, a buffer with a length of 100 words (200 bytes) could use 0xXX00 as the starting address.
3. Calculate the buffer end address using the buffer length chosen in Step 1 and the buffer start address chosen in Step 2. The buffer end address is calculated using Equation 3-1.
4. Load DSxPAG with the appropriate page value.
5. Load the XMODSRT or YMODSRT register with the buffer start address chosen in step 2.
6. Load the XMODEND or YMODEND register with the buffer end address calculated in step 3.
7. Write to the XWM bit (MODCON<3:0>) or the YWM bits (MODCON<7:4>) to select the W register that will be used to access the circular buffer.
8. Set the XMODEN bit (MODCON<15>) or the YMODEN bit (MODCON<14>) to enable the circular buffer.
9. Load the selected W register with an address that points to the buffer.

The W register address will be adjusted automatically at the end of the buffer when an indirect access with pre/post increment is performed (see Figure 3-8).

Note: The start address of an incrementing buffer can be suitably aligned to a binary 0’s boundary using the `aligned` (alignment) attribute provided by the MPLAB® C30 compiler, as follows:

```
int x __attribute__((aligned(256)));
```

Figure 3-8: Incrementing Buffer Modulo Addressing Operation Example




```
MOVPAG  #0x001,DSWPAG
MOVPAG  #0x001,DSRPAG
MOV     #0x1100,W0
MOV     W0,XMODSRT      ;set modulo start address
MOV     #0x1163,W0
MOV     W0,XMODEND      ;set modulo end address
MOV     #0x8001,W0
MOV     W0,MODCON       ;enable W1, X AGU for modulo
MOV     #0x0000,W0      ;W0 holds buffer fill value
MOV     #0x1100,W1      ;point W1 to buffer
REPEAT  #49             ;fill the 50 buffer locations
MOV     W0,[W1++]       ;fill the next location

;W1 = 0x1100 when DO loop completes
```

Start Addr = 0x9100
End Addr = 0x9163
Length = 50 Words

Note: The DSRPAG value can be different from that of DSWPAG. Refer to **3.2.2 "Paged Memory Scheme"** for details.

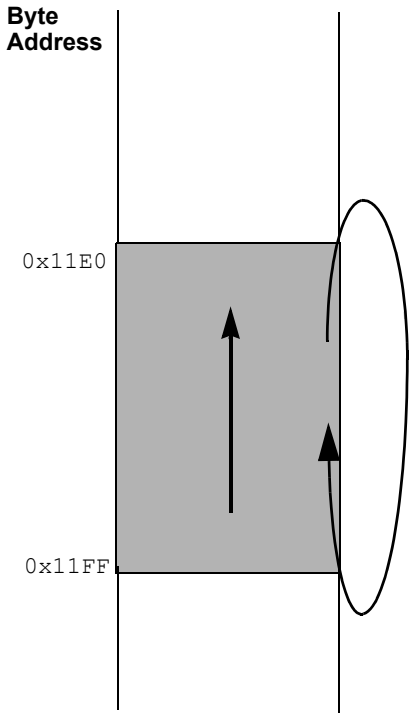
3.4.5 Modulo Addressing Initialization for Decrementing Modulo Buffer

The following steps describe the setup procedure for a decrementing circular buffer. The steps are similar whether the X AGU or Y AGU is used.

1. Determine the buffer length in 16-bit data words. Multiply this value by 2 to get the length of the buffer in bytes.
2. Select a buffer end address that is located at a binary 'ones' boundary, based on the desired length of the buffer. Remember that the buffer length in words must be multiplied by 2 to obtain the byte address range. For example, a buffer with a length of 128 words (256 bytes) could use 0xXXFF as the end address.
3. Calculate the buffer start address using the buffer length chosen in step 1 and the end address chosen in step 2. The buffer start address is calculated using Equation 3-2.
4. Load DSxPAG with the appropriate page value.
5. Load the XMODSRT or YMODSRT register with the buffer start address chosen in step 3.
6. Load the XMODEND or YMODEND register with the buffer end address chosen in step 2.
7. Write to the XWM bit (MODCON<3:0>) or the YWM bits (MODCON<7:4>) to select the W register that will be used to access the circular buffer.
8. Set the XMODEN bit (MODCON<15>) or the YMODEN bit (MODCON<14>) to enable the circular buffer.
9. Load the selected W register with an address that points to the buffer.

The W register address will be adjusted automatically at the end of the buffer when an indirect access with pre/post-decrement is performed (see Figure 3-9).

Figure 3-9: Decrementing Buffer Modulo Addressing Operation Example




```
MOVPAG  #0x001,DSWPAG
MOVPAG  #0x001,DSRPAG
MOV     #0x11E0,W0
MOV     W0,XMODSRT      ;set modulo start address
MOV     #0x11FF,W0
MOV     W0,XMODEND      ;set modulo end address
MOV     #0x8001,W0
MOV     W0,MODCON       ;enable W1, X AGU for modulo
MOV     #0x000F,W0      ;W0 holds buffer fill value
MOV     #0x11FE,W1      ;point W1 to buffer
DO      #15,FILL        ;fill the 16 buffer locations
MOV     W0,[W1--]       ;fill the next location

FILL:
DEC     W0,W0           ;decrement the fill value

; W1 = 0x11FE when DO loop completes
```

Start Addr = 0x91E0
End Addr = 0x91FF
Length = 16 Words

Note: The DSRPAG value can be different from that of DSWPAG. Refer to **3.2.2 "Paged Memory Scheme"** for details.

3.5 BIT-REVERSED ADDRESSING (dsPIC33E DEVICES ONLY)

3.5.1 Introduction to Bit-Reversed Addressing

Bit-reversed addressing is a special addressing mode that supports fast data reordering for radix-2 FFT algorithms. It is supported through the X WAGU only. Bit-reversed addressing is accomplished by effectively creating a mirror image of an address pointer by swapping the bit locations around the center point of the binary value, as shown in Figure 3-10. An example bit-reversed sequence for a 4-bit address field is shown in Table 3-3.

Figure 3-10: Bit-Reversed Address Example

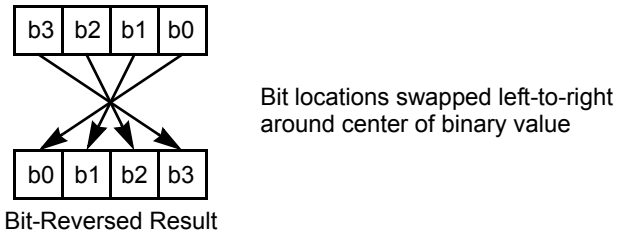


Table 3-3: Bit-Reversed Address Sequence (16-Entry)

Normal Address					Bit-Reversed Address				
A3	A2	A1	A0	Decimal	A3	A2	A1	A0	Decimal
0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0	8
0	0	1	0	2	0	1	0	0	4
0	0	1	1	3	1	1	0	0	12
0	1	0	0	4	0	0	1	0	2
0	1	0	1	5	1	0	1	0	10
0	1	1	0	6	0	1	1	0	6
0	1	1	1	7	1	1	1	0	14
1	0	0	0	8	0	0	0	1	1
1	0	0	1	9	1	0	0	1	9
1	0	1	0	10	0	1	0	1	5
1	0	1	1	11	1	1	0	1	13
1	1	0	0	12	0	0	1	1	3
1	1	0	1	13	1	0	1	1	11
1	1	1	0	14	0	1	1	1	7
1	1	1	1	15	1	1	1	1	15

3.5.2 Bit-Reversed Addressing Operation

Bit-reversed addressing is supported only by the X WAGU and is controlled by the MODCON and X Write AGU Bit-Reversal Addressing Control (XBREV) SFRs. Bit-reversed addressing is invoked as follows:

1. Bit-reversed addressing is assigned to one of the W registers using the BWM control bits (MODCON<11:8>).
2. Bit-reversed addressing is enabled by setting the BREN control bit (XBREV<15>).
3. The X AGU bit-reverse modifier is set via the XB control bits (XBREV<14:0>).

When enabled, the bit-reversed addressing hardware will generate bit-reversed addresses, only when the register indirect with Pre- or Post-increment Addressing modes are used (`[Wn++]`, `[++Wn]`). Furthermore, bit-reverse addresses are only generated for Word mode instructions. It will not function for all other Addressing modes or Byte mode instructions (normal addresses will be generated).

Note 1: A write to the MODCON register must not be followed by an instruction that performs an indirect read operation using a W register. Unexpected results may occur. Some instructions perform an implicit indirect read. These are: `POP`, `RETURN`, `RETFIE`, `RETLW` and `ULNK`.

2: If bit-reversed addressing has already been enabled by setting the BREN bit (XBREV<15>), a write to the XBREV register must not be followed by an indirect read operation using the W register, designated as the bit-reversed address pointer.

Bit-reversed addressing operates within any EDS or PSV memory space. However, it will not operate across page boundaries. The only exception to this rule is when crossing a boundary into or out of page 0.

3.5.2.1 MODULO ADDRESSING AND BIT-REVERSED ADDRESSING

Modulo addressing and bit-reversed addressing can be enabled simultaneously using the same W register, but bit-reversed addressing operation will always take precedence for data writes when enabled. As an example, the following setup conditions would assign the same W register to modulo and bit-reversed addressing:

- X modulo addressing is enabled (XMODEN = `1`)
- Bit-reverse addressing is enabled (BREN = `1`)
- W1 assigned to modulo addressing (XWM<3:0> = `0001`)
- W1 assigned to bit-reversed addressing (BWM<3:0> = `0001`)

For data reads that use W1 as the pointer, modulo address boundary checking will occur. For data writes using W1 as the destination pointer, the bit-reverse hardware will correct W1 for data reordering.

3.5.3 Bit-Reverse Modifier Value

The value loaded into the XBREV register is a constant that defines the size of the bit-reversed data buffer. The XB modifier values used with common bit-reversed buffers are summarized in Table 3-4.

Table 3-4: Bit-Reversed Address Modifier Values

Buffer Size (Words)	XB Bit-Reversed Address Modifier Value
32768	0x4000
16384	0x2000
8192	0x1000
4096	0x0800
2048	0x0400
1024	0x0200
512	0x0100
256	0x0080
128	0x0040
64	0x0020
32	0x0010
16	0x0008
8	0x0004
4	0x0002
2	0x0001

Note: Only the bit-reversed modifier values shown will produce valid bit-reversed address sequences.

The bit-reverse addressing hardware modifies the W register address by performing a “reverse-carry” addition of the W contents and the XB modifier constant. A reverse-carry addition is performed by adding the bits from left-to-right instead of right-to-left. If a carry-out occurs in a bit location, the carry out bit is added to the next bit location to the right. Example 3-8 demonstrates the reverse-carry addition and subsequent W register values using 0x0008 as the XB modifier value. Note that the XB modifier is shifted one bit location to the left to generate word address values.

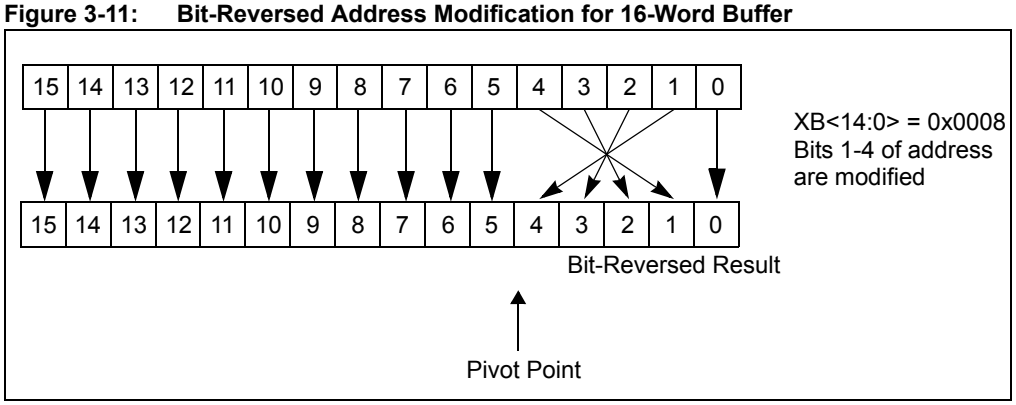
Example 3-8: XB Address Calculation

```
0000 0000 0000 0000     Wn points to word 0
           +1 0000      Wn = Wn + XB
-------------------

0000 0000 0001 0000     Wn points to word 8
           +1 0000      Wn = Wn + XB
-------------------

0000 0000 0000 1000     Wn points to word 4
           +1 0000      Wn = Wn + XB
-------------------

0000 0000 0001 1000     Wn points to word 12
           +1 0000      Wn = Wn + XB
-------------------

0000 0000 0000 0100     Wn points to word 2
           +1 0000      Wn = Wn + XB
-------------------

0000 0000 0001 0100     Wn points to word 10
```

When XB<14:0> = 0x0008, the bit-reversed buffer size will be 16 words. Bits 1-4 of the W register will be subject to bit-reversed address correction, but bits 5-15 (outside the pivot point) will not be modified by the bit-reverse hardware. Bit 0 is not modified because the bit-reverse hardware operates only on word addresses.

The XB modifier controls the pivot point for the bit-reverse address modification. Bits outside of the pivot point will not be subject to bit-reversed address corrections.

Figure 3-11: Bit-Reversed Address Modification for 16-Word Buffer



3.5.4 Bit-Reversed Addressing Code Example

The code shown in Example 3-9 reads a series of 16 data words and writes the data to a new location in bit-reversed order. W0 is the read address pointer and W1 is the write address pointer subject to bit-reverse modification.

Example 3-9: Bit-Reversed Addressing Code Example

```
    .section .data
    .global Input_Buf
Input_Buf:
    .word 0x0000, 0x1111, 0x2222, 0x3333, 0x4444, 0x5555, 0x6666, 0x7777,
0x8888, 0x9999, 0xAAAA, 0xBBBB, 0xCCCC, 0xDDDD, 0xEEEE, 0xFFFF

    .section .bss
    .global Bit_Rev_Buf
    .align 32
Bit_Rev_Buf:
    .space 32

    ; Start of code section
    .text

    .global _main
_main:

    ; Set XBREV for 16-word buffer with bit-reversed addressing
    MOV #0x8008, W0
    MOV W0, XBREV

    ; Set up W1 as a pointer for bit-reversed addressing
    MOV #0x01FF, W0
    MOV W0, MODCON

    ; W0 points to sequential input data buffer
    MOV #Input_Buf, W0

    ; W1 points to bit-reversed output data buffer
    MOV #Bit_Rev_Buf, W1

    ; Re-order the data from Input_buf into Bit_Rev_Buf
    REPEAT #15
    MOV [W0++], [W1++]

done:
    BRA done

    RETURN
```

3.6 DMA RAM

Some dsPIC33E/PIC24E devices contain DMA and dual-ported SRAM memory (DPSRAM). Only the DMA controller can write and read to/from addresses within the DPSRAM without arbitration and CPU stalls, resulting in maximized real-time performance. In addition, DMA can address all of data memory and EDS excluding SFR space. Access to data memory and EDS is arbitrated by the EDS arbiter, and therefore, may be subject to stalls.

Note:	The presence and size of DPSRAM is device specific. Refer to the specific dsPIC33E/PIC24E device data sheet for further details.

Figure 3-6 shows a block diagram that demonstrates how the DMA integrates into the dsPIC33E/PIC24E internal architecture.

The DMA channels communicate with Port 2 of the DPSRAM and the DMA port of each of the DMA-ready peripherals across a dedicated DMA bus. Refer to **Section 22. "Direct Memory Access (DMA)"** (DS70348) for more information.

3.7 CONTROL REGISTER DESCRIPTIONS

The EDS Bus Master Priority Control register (MSTRPR) sets the access priority of the bus masters to the data memory.

Register 3-1: MSTRPR: EDS Bus Master Priority Control Register

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
MSTRPR<15:8>							
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
MSTRPR<7:0>							
bit 7							bit 0

Legend:	r = Reserved		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 15-0 **MSTRPR<15:0>:** EDS Bus Master Priority bits
Refer to **3.2.4 “EDS Arbitration and Bus Master Priority”** for additional details.

Note 1: Default priority at reset is CPU (M0, highest), M1, M2, M3, ICD (M4, always lowest priority).

2: All raised priority bus masters maintain the same priority relationship relative to each other: M1 (highest), M2, M3 (lowest).

3: All bus masters whose priority remains below that of the CPU, maintain the same priority relationship relative to each other: M1 (highest), M2, M3 (lowest).

4: The specific bus masters that are available varies depending on the device used. Refer to the specific device data sheet for further details.

The following eight registers are used to control the data space page, modulo addressing and bit-reversed addressing.

Register 3-2: DSRPAG: Data Space Read Page Register

U-0	U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0
—	—	—	—	—	—	DSRPAG<9:8>(1)	
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1
DSRPAG<7:0>(1)							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-10 **Unimplemented:** Read as '0'

bit 9-0 **DSRPAG<9:0>:** Data Space Read Page Pointer bits(1)

Note 1: Attempting to read from the paged DS window when DSRPAG = 0x000, will cause an address error trap.

Register 3-3: DSWPAG: Data Space Write Page Register

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0
—	—	—	—	—	—	—	DSWPAG<8>(1)
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1
DSWPAG<7:0>(1)							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-9 **Unimplemented:** Read as '0'

bit 8-0 **DSWPAG<8:0>:** Data Space Write Page Pointer bits(1)

Note 1: Attempting to write from the paged DS window when DSWPAG = 0x000, will cause an address error trap.

Register 3-4: MODCON: Modulo and Bit-Reversed Addressing Control Register

R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
XMODEN	YMODEN	—	—	BWM<3:0>			
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
YWM<3:0>				XWM<3:0>			
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15 **XMODEN:** X RAGU and X WAGU Modulus Addressing Enable bit
1 = X AGU modulus addressing enabled
0 = X AGU modulus addressing disabled

bit 14 **YMODEN:** Y AGU Modulus Addressing Enable bit
1 = Y AGU modulus addressing enabled
0 = Y AGU modulus addressing disabled

bit 13-12 **Unimplemented:** Read as '0'

bit 11-8 **BWM<3:0>:** X WAGU Register Select for Bit-Reversed Addressing bits
1111 = Bit-reversed addressing disabled
1110 = W14 selected for bit-reversed addressing
1101 = W13 selected for bit-reversed addressing
•
•
•
0000 = W0 selected for bit-reversed addressing

bit 7-4 **YWM<3:0>:** Y AGU W Register Select for Modulo Addressing bits
1111 = Modulo addressing disabled
1010 = W10 selected for modulo addressing
1011 = W11 selected for modulo addressing

All other settings of the YWM<3:0> control bits are reserved and should not be used.

bit 3-0 **XWM<3:0>:** X RAGU and X WAGU W Register Select for Modulo Addressing bits
1111 = Modulo addressing disabled
1110 = W14 selected for modulo addressing
•
•
•
0000 = W0 selected for modulo addressing

Note: A write to the MODCON register should not be followed by an instruction that performs an indirect read operation using a W register. Unexpected results may occur. Some instructions perform an implicit indirect read. These are: POP, RETURN, RETFIE, RETLW and ULNK.

Register 3-5: XMODSRT: X AGU Modulo Addressing Start Register

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
XS<15:8>							
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0
XS<7:1>							0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-1 **XS<15:1>:** X RAGU and X WAGU Modulo Addressing Start Address bits

bit 0 **Unimplemented:** Read as '0'

Note: A write to the XMODSRT register should not be followed by an instruction that performs an indirect read operation using a W register. Unexpected results may occur. Some instructions perform an implicit indirect read. These are: POP, RETURN, RETFIE, RETLW and ULNK.

Register 3-6: XMODEND: X AGU Modulo Addressing End Register

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
XE<15:8>							
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-1
XE<7:1>							1
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-1 **XE<15:1>:** X RAGU and X WAGU Modulo Addressing End Address bits

bit 0 **Unimplemented:** Read as '1'

Note: A write to the XMODEND register should not be followed by an instruction that performs an indirect read operation using a W register. Unexpected results may occur. Some instructions perform an implicit indirect read. These are: POP, RETURN, RETFIE, RETLW and ULNK.

Register 3-7: YMODSRT: Y AGU Modulo Addressing Start Register

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
YS<15:8>							
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0
YS<7:1>							0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-1 **YS<15:1>:** Y AGU Modulo Addressing Start Address bits

bit 0 **Unimplemented:** Read as '0'

Note: A write to the YMODSRT register should not be followed by an instruction that performs an indirect read operation using a W register. Unexpected results may occur. Some instructions perform an implicit indirect read. These are: POP, RETURN, RETFIE, RETLW and ULNK.

Register 3-8: YMODEND: Y AGU Modulo Addressing End Register

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
YE<15:8>							
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-1
YE<7:1>							1
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-1 **YE<15:1>:** Y AGU Modulo Addressing End Address bits

bit 0 **Unimplemented:** Read as '1'

Note: A write to the YMODEND register should not be followed by an instruction that performs an indirect read operation using a W register. Unexpected results may occur. Some instructions perform an implicit indirect read. These are: POP, RETURN, RETFIE, RETLW and ULNK.

Register 3-9: XBREV: X Write AGU Bit-Reversal Addressing Control Register

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
BREN	XB<14:8>						
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
XB<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15 **BREN:** Bit-Reversed Addressing (X AGU only) Enable bit
1 = Bit-reversed addressing enabled
0 = Bit-reversed addressing disabled

bit 14-0 **XB<14:0>:** X AGU Bit-Reversed Modifier bits
0x4000 = 32768 word buffer
0x2000 = 16384 word buffer
0x1000 = 8192 word buffer
0x0800 = 4096 word buffer
0x0400 = 2048 word buffer
0x0200 = 1024 word buffer
0x0100 = 512 word buffer
0x0080 = 256 word buffer
0x0040 = 128 word buffer
0x0020 = 64 word buffer
0x0010 = 32 word buffer
0x0008 = 16 word buffer
0x0004 = 8 word buffer
0x0002 = 4 word buffer
0x0001 = 2 word buffer

Note: If the BREN bit is set, a write to the XB<14:0> bits should not be followed by an instruction that performs an indirect read operation using a W register. Unexpected results may occur. Some instructions perform an implicit indirect read. These are: POP, RETURN, RETFIE, RETLW and ULNK.

3.8 REGISTER MAPS

A summary of the registers associated with the dsPIC33E/PIC24E Data Memory module is provided in Table 3-5.

Table 3-5: Data Memory Control Register Map

File Name	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	All Resets
DSRPAG	—	—	—	—	—	—	DSRPAG<9:0>										0001
DSWPAG	—	—	—	—	—	—	—	DSWPAG<8:0>									0001
MODCON	XMODEN	YMODEN	—	—	BMW3	BWM2	BWM1	BWM0	YWM3	YWM2	YWM1	YWM0	XWM3	XWM2	XWM1	XWM0	0000
XMODSRT	XS15	XS14	XS13	XS12	XS11	XS10	XS9	XS8	XS7	XS6	XS5	XS4	XS3	XS2	XS1	—	xxxx
XMODEND	XE15	XE14	XE13	XE12	XE11	XE10	XE9	XE8	XE7	XE6	XE5	XE4	XE3	XE2	XE1	—	xxxx
YMODSRT	YS15	YS14	YS13	YS12	YS11	YS10	YS9	YS8	YS7	YS6	YS5	YS4	YS3	YS2	YS1	—	xxxx
YMODEND	YE15	YE14	YE13	YE12	YE11	YE10	YE9	YE8	YE7	YE6	YE5	YE4	YE3	YE2	YE1	—	xxxx
XBREV	BREN	XB14	XB13	XB12	XB11	XB10	XB9	XB8	XB7	XB6	XB5	XB4	XB3	XB2	XB1	XB0	0000
MSTRPR	MSTRPR<15:0>																0000

Legend: x = unknown value on Reset, — = unimplemented, read as '0'. Reset values are shown in hexadecimal.

3.9 RELATED APPLICATION NOTES

This section lists application notes that are related to this section of the manual. These application notes may not be written specifically for the dsPIC33E/PIC24E product family, but the concepts are pertinent and could be used with modification and possible limitations. The current application notes related to the Data Memory module are:

Title	Application Note #
No related application notes at this time.	N/A

Note: Please visit the Microchip web site (www.microchip.com) for additional Application Notes and code examples for the dsPIC33E/PIC24E family of devices.

3.10 REVISION HISTORY

Revision A (June 2009)

This is the initial released version of this document.

Revision B (August 2010)

This revision includes the following updates:

- Minor changes to text and formatting have been incorporated throughout the document
- The Preliminary document status has been removed
- The following sections pertain to dsPIC33E devices only:
  - **3.4 “Modulo Addressing (dsPIC33E Devices Only)”**
  - **3.5 “Bit-Reversed Addressing (dsPIC33E Devices Only)”**
- All references to DMA RAM (with the exception of the section with the same name) have been changed to: DPSRAM
- Note 2 was removed from the Example Data Memory Map (see Figure 3-1)
- Added new examples on managing EDS access (see Example 3-1, Example 3-2, and Example 3-3)
- The fourth paragraph in **3.2.2 “Paged Memory Scheme”** has been updated and converted to a shaded note
- The Paged Data Memory Space (Figure 3-5) has been replaced with the EDS Memory Map (formerly Figure 3-6)
- The Pseudo-Linear Addressing Truth Table (Table 3-1) has been replaced with an entirely new table named Overflow and Underflow Scenarios at Page 0, EDS and PSV Space Boundaries
- **3.2.4 “EDS Arbitration and Bus Master Priority”** has been updated in its entirety
- A shaded note referencing the aligned (alignment) attribute was added to **3.4.4 “Modulo Addressing Initialization for Incrementing Modulo Buffer”**
- The first two instructions and the Start and End Addr values in Figure 3-8 and Figure 3-9 have been changed
- The paragraph in 3.5.2.2 DATA DEPENDENCIES ASSOCIATED WITH XBREV has been moved to Note 2 in the shaded note in **3.5.2 “Bit-Reversed Addressing Operation”**
- The Bit-Reversed Addressing Code Example (Example 3-9) has been updated in its entirety
- The EDS Bus Mater Priority Control Register has been updated (see Register 3-1)
- The notes in the Data Space Read Page (DSRPAG) and Data Space Write Page (DSWPAG) registers have been changed (see Register 3-2 and Register 3-3)
- A note regarding writes to a register have been added to the following registers:
  - XMODSRT: X AGU Modulo Addressing Start Register (see Register 3-5)
  - XMODEND: X AGU Modulo Addressing End Register (see Register 3-6)
  - YMODSRT: Y AGU Modulo Addressing Start Register (see Register 3-7)
  - YMODEND: Y AGU Modulo Addressing End Register (see Register 3-8)
  - XBREV: X Write AGU Bit-Reversal Addressing Control Register (see Register 3-9)
- The MSTRPR register has been updated in the Data Memory Control Register Map (see Table 3-5)

Revision C (June 2011)

This revision includes the following updates:

- Updated the EDS Access code examples (see Example 3-1 and Example 3-2)
- Changes to formatting and minor text updates were incorporated throughout the document

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Trademarks

ISBN: 978-1-61341-306-7

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
ISO/TS 16949:2009

Microchip received ISO/TS-16949:2002 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

Worldwide Sales and Service

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://www.microchip.com/support
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Cleveland
Independence, OH
Tel: 216-447-0464
Fax: 216-447-0643

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Farmington Hills, MI
Tel: 248-538-2250
Fax: 248-538-2260

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

Santa Clara
Santa Clara, CA
Tel: 408-961-6444
Fax: 408-961-6445

Toronto
Mississauga, Ontario,
Canada
Tel: 905-673-0699
Fax: 905-673-6509

ASIA/PACIFIC

Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing
Tel: 86-10-8569-7000
Fax: 86-10-8528-2104

China - Chengdu
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

China - Chongqing
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

China - Hangzhou
Tel: 86-571-2819-3180
Fax: 86-571-2819-3189

China - Hong Kong SAR
Tel: 852-2401-1200
Fax: 852-2401-3431

China - Nanjing
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

China - Qingdao
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

China - Shanghai
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

China - Shenyang
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

China - Shenzhen
Tel: 86-755-8203-2660
Fax: 86-755-8203-1760

China - Wuhan
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

China - Xian
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

China - Xiamen
Tel: 86-592-2388138
Fax: 86-592-2388130

China - Zhuhai
Tel: 86-756-3210040
Fax: 86-756-3210049

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

India - Pune
Tel: 91-20-2566-1512
Fax: 91-20-2566-1513

Japan - Yokohama
Tel: 81-45-471- 6166
Fax: 81-45-471-6122

Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

Taiwan - Hsin Chu
Tel: 886-3-6578-300
Fax: 886-3-6578-370

Taiwan - Kaohsiung
Tel: 886-7-213-7830
Fax: 886-7-330-9305

Taiwan - Taipei
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

UK - Wokingham
Tel: 44-118-921-5869
Fax: 44-118-921-5820

05/02/11