
Section 4. Program Memory

HIGHLIGHTS

This section of the manual contains the following topics:

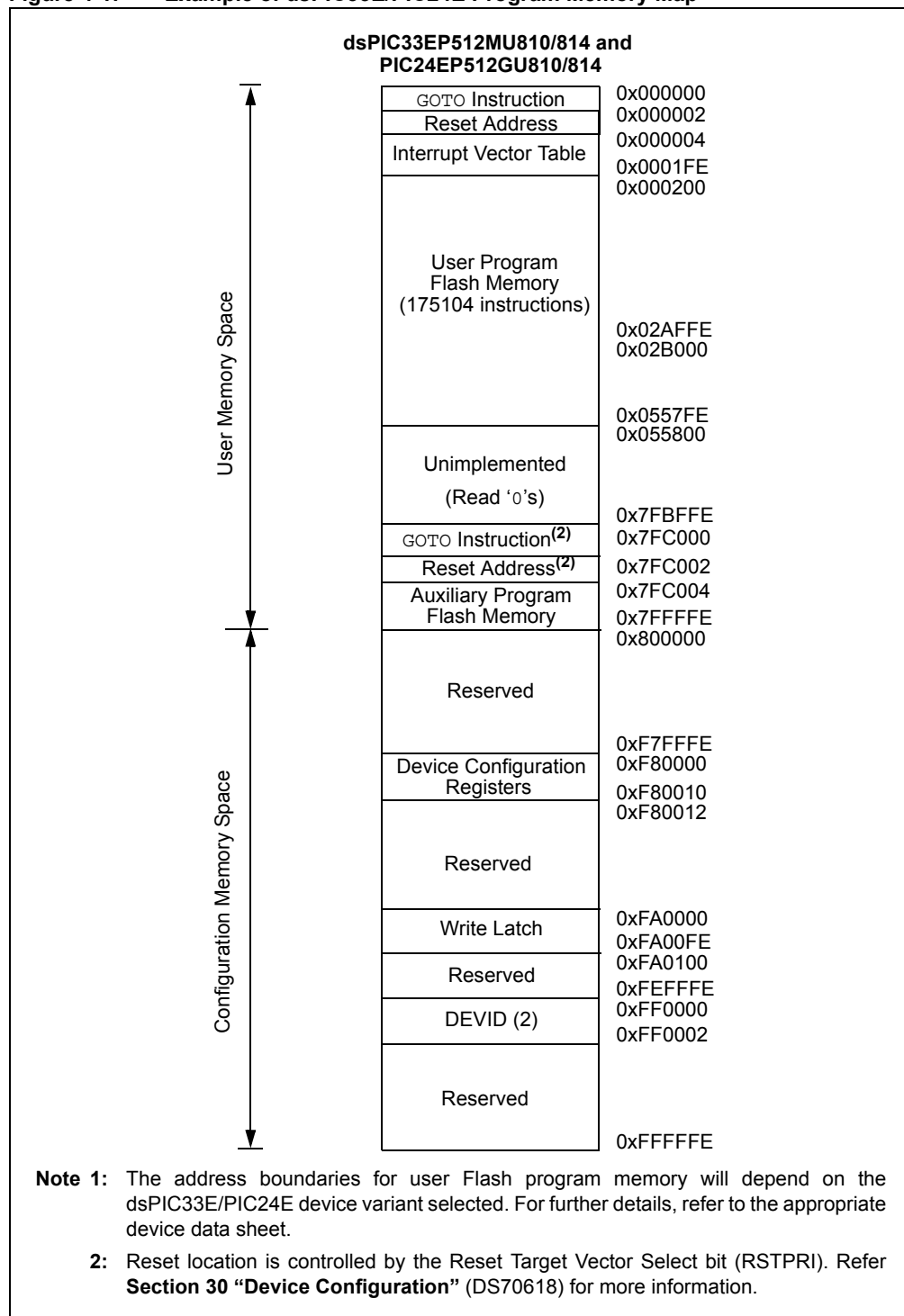
4.1	Program Memory Address Map	4-2
4.2	Control Registers	4-4
4.3	Program Counter	4-5
4.4	Program Memory Access Using Table Instructions	4-6
4.5	Program Space Visibility from Data Space.....	4-12
4.6	Program Memory Writes	4-19
4.7	Program Memory Low-Power Mode	4-19
4.8	Register Map.....	4-20
4.9	Related Application Notes.....	4-21
4.10	Revision History	4-22

4.1 PROGRAM MEMORY ADDRESS MAP

dsPIC33E/PIC24E devices have a 4M x 24-bit program memory address space. An example of the program memory map is presented in Figure 4-1. The program memory space can be accessed through the following methods:

- the 23-bit program counter (PC)
- table read (TBLRD) and table write (TBLWT) instructions
- mapping any 32-Kbyte segment of program memory into the data memory address space

Figure 4-1: Example of dsPIC33E/PIC24E Program Memory Map



Arrows on the left side of the program memory map in Figure 4-1 illustrate the division of the program memory address space in dsPIC33E/PIC24E devices into the User Memory Space and the Configuration Memory Space.

The User Memory Space is comprised of the following areas:

- Reset Vector Address
- Interrupt Vector Table
- User Program Flash Memory
- Auxiliary Program Flash Memory (if applicable; refer to the data sheet for a specific device)

Instructions in auxiliary program Flash memory can be executed by the CPU, without stalling it, while the user program memory is being erased and/or programmed. Similarly, instructions in the user program memory can be executed by the CPU while the auxiliary program memory is being erased and/or programmed, without stalls.

The Configuration Memory Space contains the nonvolatile Configuration bits, for setting device options, and the device ID locations.

dsPIC33E/PIC24E Family Reference Manual

4.2 CONTROL REGISTERS

Register 4-1: TBLPAG: Table Page Register

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-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
TBLPAG<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 15-8 **Unimplemented:** Read as '0'

bit 7-0 **TBLPAG<7:0>**: Table Address Page bits

The 8-bit Table Address Page bits are concatenated with the W register to form a 23-bit effective program memory address plus a Byte Select bit.

Register 4-2: DSRPAG: Data Space Read Page Register^(1,2)

U-0	U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0
—	—	—	—	—	—	DSRPAG<9:8>	
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-1
DSRPAG<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 15-10 **Unimplemented:** Read as '0'

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

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

2: DSRPAG is reset to 0x001.

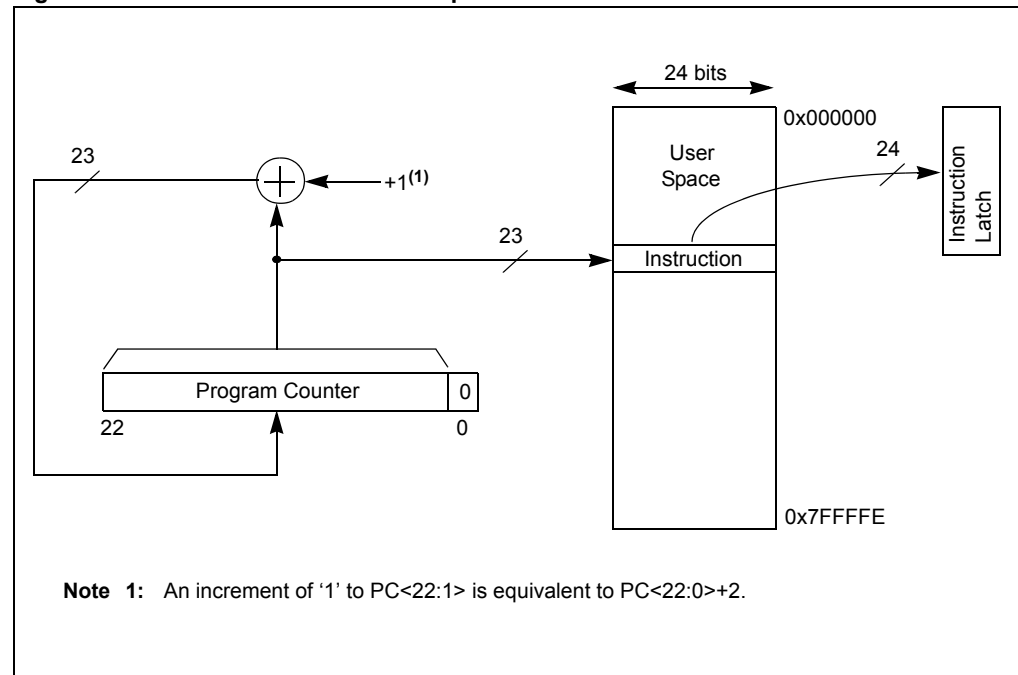
4.3 PROGRAM COUNTER

The PC increments by two with the Least Significant bit (LSb) set to '0' to provide compatibility with data space addressing. Sequential instruction words are addressed in the 4M program memory space by PC<22:1>. Each instruction word is 24 bits wide.

The LSb of the program memory address (PC<0>) is reserved as a Byte Select bit for program memory accesses from data space, that use Program Space Visibility (PSV) or table instructions. For instruction fetches via the PC, the Byte Select bit is not required, so PC<0> is always set to '0'. For more information on the PSV mode of operation, see **4.5 “Program Space Visibility from Data Space”**.

Figure 4-2 illustrates an instruction fetch example. Note that incrementing PC<22:1> by '1' is equivalent to adding 2 to PC<22:0>.

Figure 4-2: Instruction Fetch Example



4.4 PROGRAM MEMORY ACCESS USING TABLE INSTRUCTIONS

The TBLRDx and TBLWTx instructions offer a direct method of reading or writing the least significant word (lsw) and the Most Significant Byte (MSB) of any address within program space without going through data space, which is preferable for some applications.

4.4.1 Table Instruction Summary

A set of table instructions is provided to move byte- or word-sized data between program space and data space. The table read instructions are used to read from the program memory space into data memory space. The table write instructions allow data memory to be written to the program memory space using write latches.

Note: Detailed code examples using table write instructions can be found in **Section 5. “Flash Programming”** (DS70609).

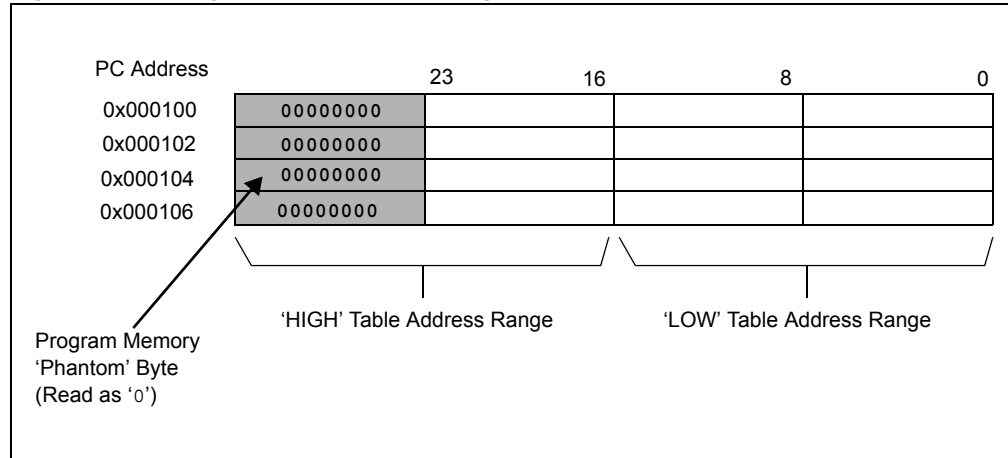
The four available table instructions are:

- TBLRDL: Table Read Low
- TBLWTL: Table Write Low
- TBLRDH: Table Read High
- TBLWTH: Table Write High

For table instructions, program memory can be regarded as two 16-bit, word-wide address spaces residing side by side, each with the same address range (as illustrated in Figure 4-3). This allows program space to be accessed as byte or aligned word addressable, 16-bit-wide, 64-Kbyte pages (i.e., same as data space).

TBLRDL and TBLWTL access the least significant data word of the program memory, and TBLRDH and TBLWTH access the upper word. Because program memory is only 24 bits wide, the upper byte from this latter space does not exist, although it is addressable. It is, therefore, termed the “phantom” byte.

Figure 4-3: High and Low Address Regions for Table Operations



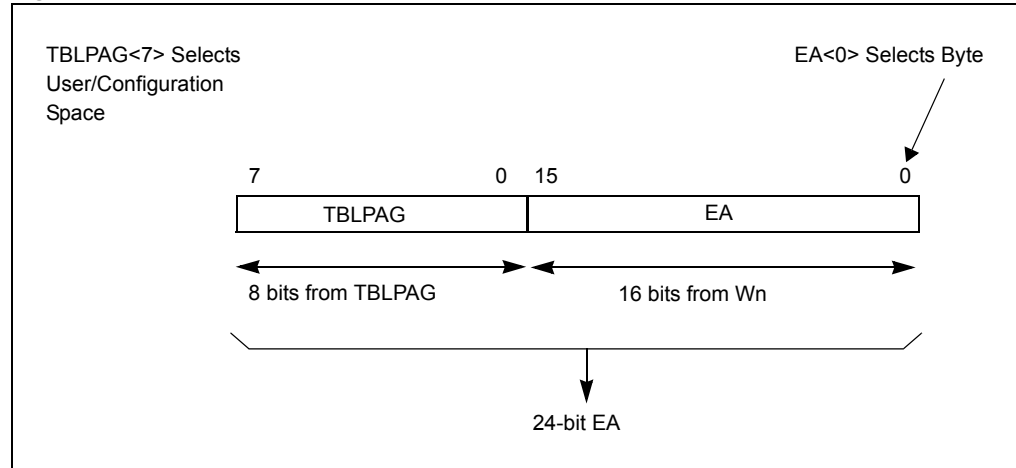
Note: The TBLPAG register can be written using the MOV_{PAG} #lit8, TBLPAG instruction or the MOV_{PAGW} Wn, TBLPAG instruction. Refer to the “16-bit MCU and DSC Programmer’s Reference Manual” (DS70157) for details.

4.4.2 Table Address Generation

Figure 4-4 illustrates how for all table instructions, a W register address value is concatenated with the 8-bit Table Page register (TBLPAG), to form a 24-bit effective program space address, including a Byte Select bit (bit 0). Because there are 16 bits of program space address provided from the W register, the data table page size in program memory is 32K words.

Note: In the event of an overflow or underflow, the Effective Address (EA) will wrap to the beginning of the current page.

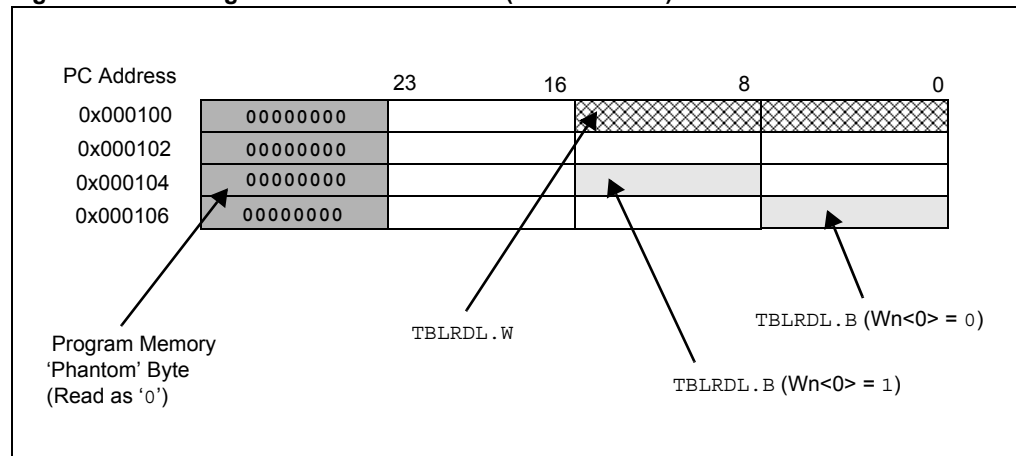
Figure 4-4: Address Generation for Table Operations



4.4.3 Program Memory Low Word Access

The TBLRD_L and TBLWT_L instructions are used to access the lower 16 bits of program memory data. The LSb of the W register, which is used as a pointer, is ignored for word-wide table accesses. For byte-wide accesses, the LSb of the W register address determines which byte is read. Figure 4-5 demonstrates the program memory data regions accessed by the TBLRD_L and TBLWT_L instructions.

Figure 4-5: Program Data Table Access (Lower 16 bits)



4.4.4 Program Memory High Word Access

The TBLRDH and TBLWTH instructions are used to access the upper 8 bits of the program memory data. Figure 4-6 illustrates how these instructions also support Word or Byte Access modes for orthogonality, but the high byte of the program memory data will always return '0'.

Figure 4-6: Program Data Table Access (Upper 8 bits)

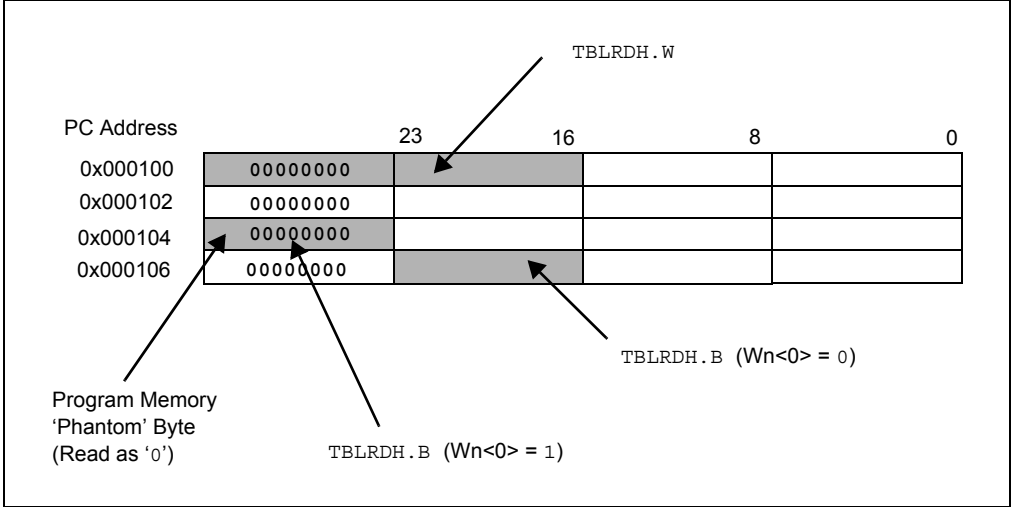
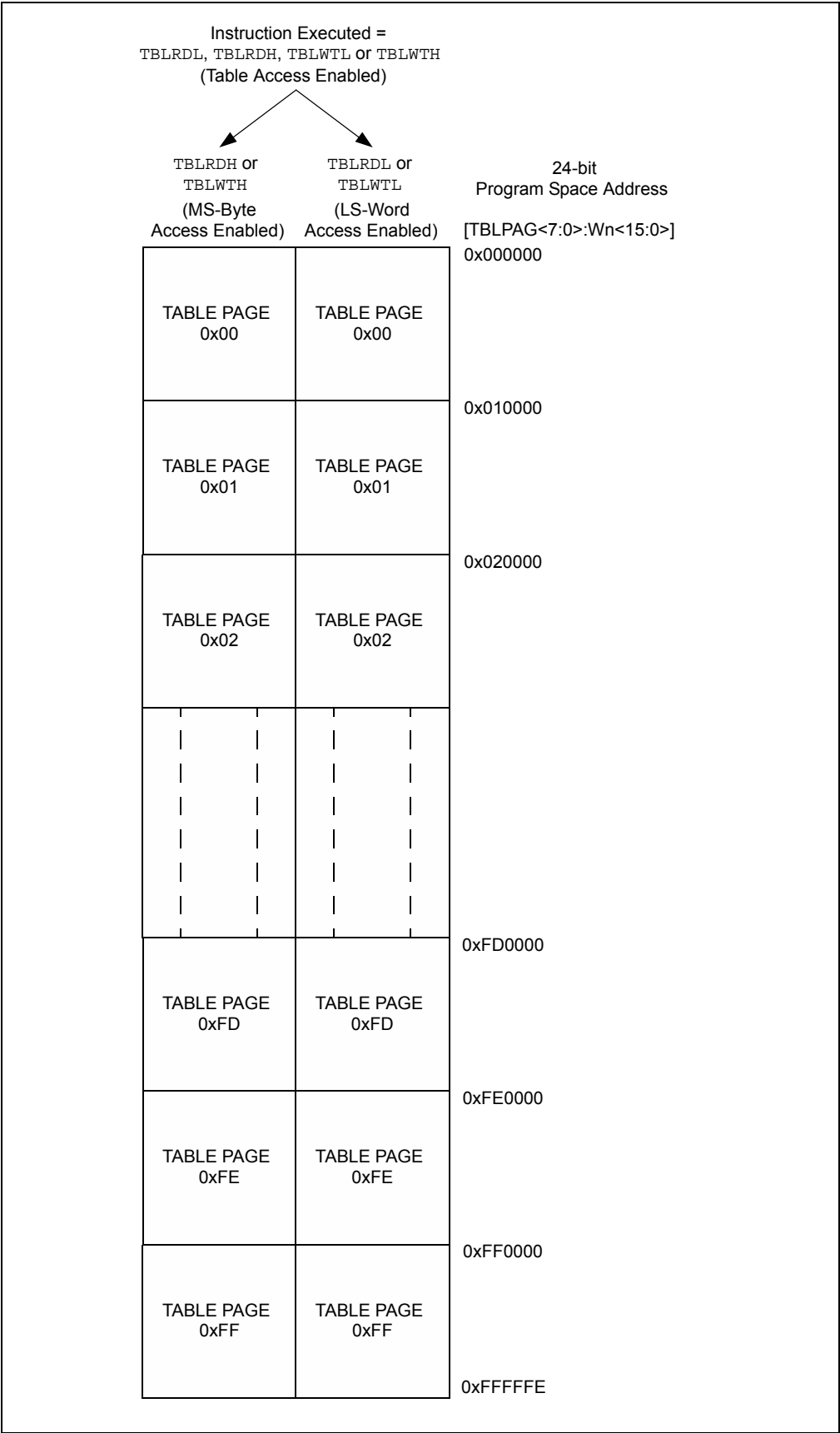


Figure 4-7: Table Memory Map



4.4.5 Data Storage in Program Memory

It is assumed that for most applications, the high byte (PC<23:16>) is not used for data, making the program memory appear to be 16 bits wide for data storage. It is recommended that the upper byte of program data be programmed either as a NOP instruction, or as an illegal opcode value, to protect the device from accidental execution of stored data. The TBLRDH and TBLWTH instructions are provided primarily for array program/verification purposes, and for applications that require compressed data storage.

4.4.6 Accessing Program Memory Using Table Instructions

In Example 4-1, table instructions are used to access the program memory using an assembly language subroutine. In Example 4-2, program memory is accessed using the `_builtin_tblpage` and `_builtin_tbloffset` built-in functions provided by the MPLAB® C30 compiler.

Note: For more information on the unlocking sequence, refer to **Section 5. “Flash Programming”** (DS70609).

Example 4-1: Using Table Instructions to Access Program Memory

```
#include <p33Exxxx.h>

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

#define PM_ROW_ERASE 0x4003
#define PM_ROW_WRITE 0x4002
#define CONFIG_WORD_WRITE 0x4000

extern long MemRead (unsigned int TablePage, unsigned int TableOffset);

unsigned long Data1, Data2, Data3;

int main(void)
{
    /* Read Configuration Register addresses 0xF80000 and 0xF80002 */
    Data1 = MemRead (0xF8, 0x0006);
    Data2 = MemRead (0xF8, 0x0008);
    Data3 = MemRead (0xF8, 0x000A);

    while(1);
}

.section .text
.global _MemRead

;*****
; Function _MemRead:
;
; W0 = TBLPAG value
; W1 = Table Offset
; Return: Data in W1:W0
;*****
_MemRead:
    MOV W0, TBLPAG
    NOP
    TBLRDH [W1], W0
    TBLRDH [W1], W1
    RETURN
```

Example 4-2 uses the `space(prog)` attribute to allocate the buffer in program memory. Also, the upper byte of each element in the constant array is filled with a value of 0xAB, using the `fillupper()` attribute. The MPLAB® C30 Compiler has built-in functions, such as `builtin_tblpage` and `builtin_tbloffset`, that can be used to access the buffer.

Example 4-2: Using MPLAB® C Compiler to Access Program Memory

```
#include <p33Exxxx.h>

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

int prog_data[10] __attribute__((space(prog),fillupper(0xAB))) = {0x0000,
0x1111, 0x2222, 0x3333, 0x4444, 0x5555, 0x6666, 0x7777, 0x8888, 0x9999};

unsigned int lowWord[10], highWord[10];
unsigned int tableOffset, loopCount;

int main(void)
{
    TBLPAG = __builtin_tblpage (prog_data);
    tableOffset = __builtin_tbloffset (prog_data);

    /* Read all 10 constants into the lowWord and highWord arrays */
    for (loopCount = 0; loopCount < 10; loopCount++)
    {
        lowWord[loopCount] = __builtin_tblrld (tableOffset);
        highWord[loopCount] = __builtin_tblrhd (tableOffset);
        tableOffset +=2;
    }

    while(1);
}
```

4.5 PROGRAM SPACE VISIBILITY FROM DATA SPACE

The upper 32 Kbytes of the dsPIC33E/PIC24E data memory address space can optionally be mapped into any 16K word program space page. The PSV mode of operation provides transparent access of stored constant data from X data space without the need to use special instructions (i.e., TBLRD, TBLWT instructions).

4.5.1 PSV Configuration

The dsPIC33E/PIC24E core extends the available data space through a paging scheme to make it appear linear for pre- and post-modified effective addresses.

The upper half of the base data space address (0x8000 to 0xFFFF) is used with the 10-bit Data Space Read Page register (DSRPAG) to form a PSV address, and can address 8 Mbytes of PSV address space.

The paged memory scheme provides access to multiple 32-Kbyte windows in the PSV memory. The PSV in the paged data memory space is illustrated in Figure 4-8.

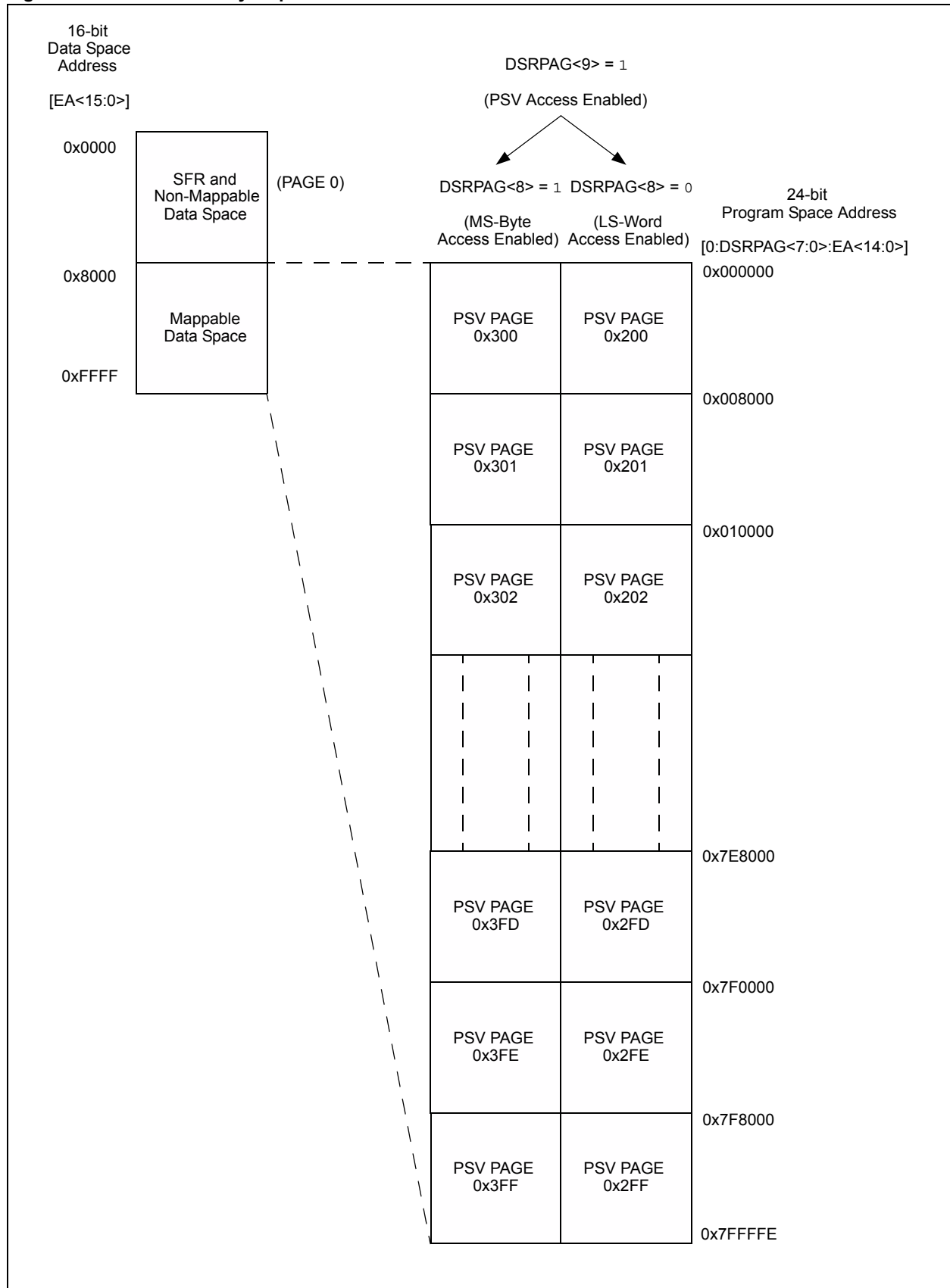
Program space (PS) can be read with a DSRPAG register of 0x200 or greater.

Reads from PS are supported using the DSRPAG register. Writes to PS are not supported; therefore, the Data Space Write Page register (DSWPAG) is dedicated exclusively to data space (DS), including extended data space (EDS).

For more information on the paged memory scheme refer to **Section 3. “Data Memory”** (DS70595).

Note: The DSRPAG register can be written using the <code>MOVPAG #lit10, DSRPAG</code> instruction or the <code>MOVPAG Wn, DSRPAG</code> instruction. Refer to the “16-bit MCU and DSC Programmer’s Reference Manual” (DS70157) for details.
--

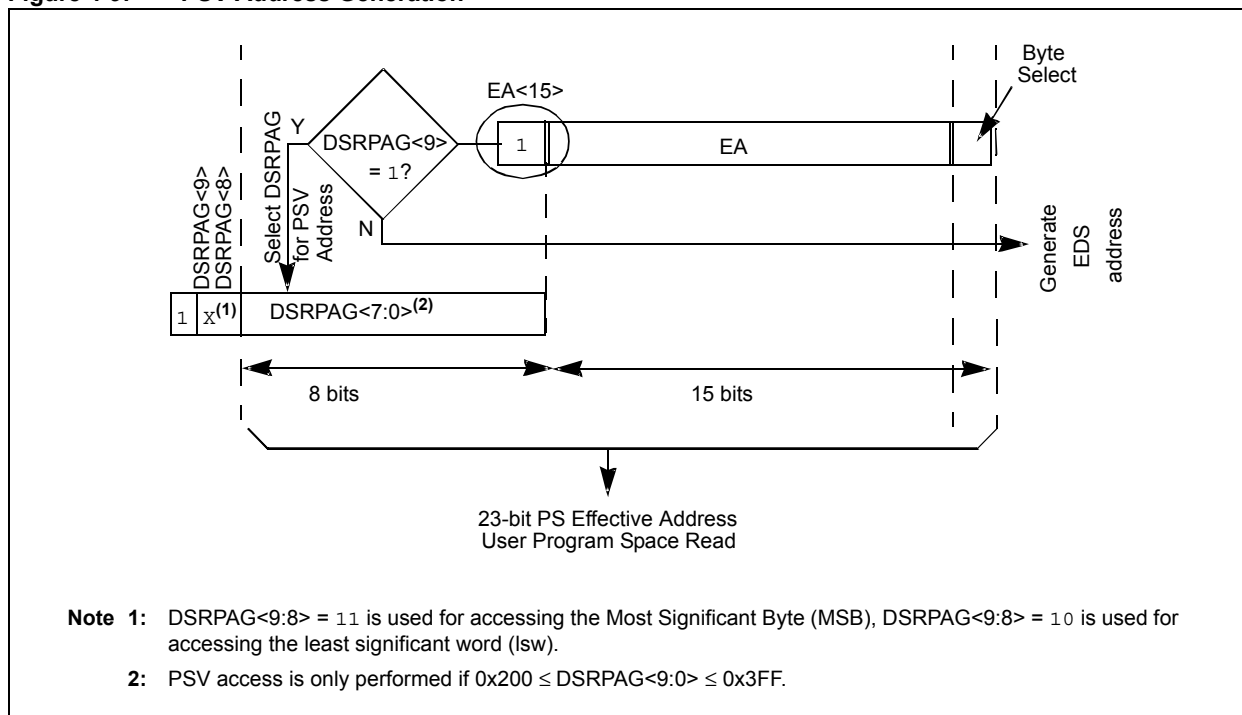
Figure 4-8: PSV Memory Map



Allocating different page registers for read and write access allows the architecture to support data movement from different PSV to EDS pages, by configuring DSRPAG and DSWPAG to address PSV and EDS space, respectively. The data can be moved from PSV to EDS space by a single instruction.

Figure 4-9 illustrates the generation of the PSV address. The 15 Least Significant bits (LSBs) of the PSV address are provided by the W register that contains the effective address. The Most Significant bit (MSb) of the W register is not used to form the address. Instead, the MSb specifies whether to perform a PSV access from program memory space or a normal access from the data memory space. If the effective address of the W register is 0x8000 or greater, the data access will occur from program memory space, depending on the page selected by the DSRPAG register. All data access occurs from the data memory when the effective address of the W register is less than 0x8000.

Figure 4-9: PSV Address Generation



The remaining address bits are provided by the 8 LSb of the Read Data Space Page register (DSRPAG<7:0>). The DSRPAG<7:0> bits are concatenated with the 15 LSb of the W register holding the effective address, and the MSb is forced to '0', thereby forming a 24-bit program memory address.

Note: PSV can only be used to access values in the program memory space. Table instructions must be used to access values in the user configuration space.

The LSb of the W register value is used as a Byte Select bit, which allows instructions using PSV to operate in Byte or Word mode.

The PSV address is split into lsw and MSB. When DSRPAG<9:8> = 0b10, the lsw 16 bits of the 24-bit PS word can be accessed using PSV. When DSRPAG<9:8> = 0b11, the MSB of the 24-bit PS word can be accessed using PSV. This is illustrated in Figure 4-10 and Figure 4-11. The range of valid DSRPAG values for a lsw read starts at DSRPAG = 0x200 and the range of valid DSRPAG values for a MSB read starts at DSRPAG = 0x300.

Figure 4-10: Program Space Visibility Operation Isw Access

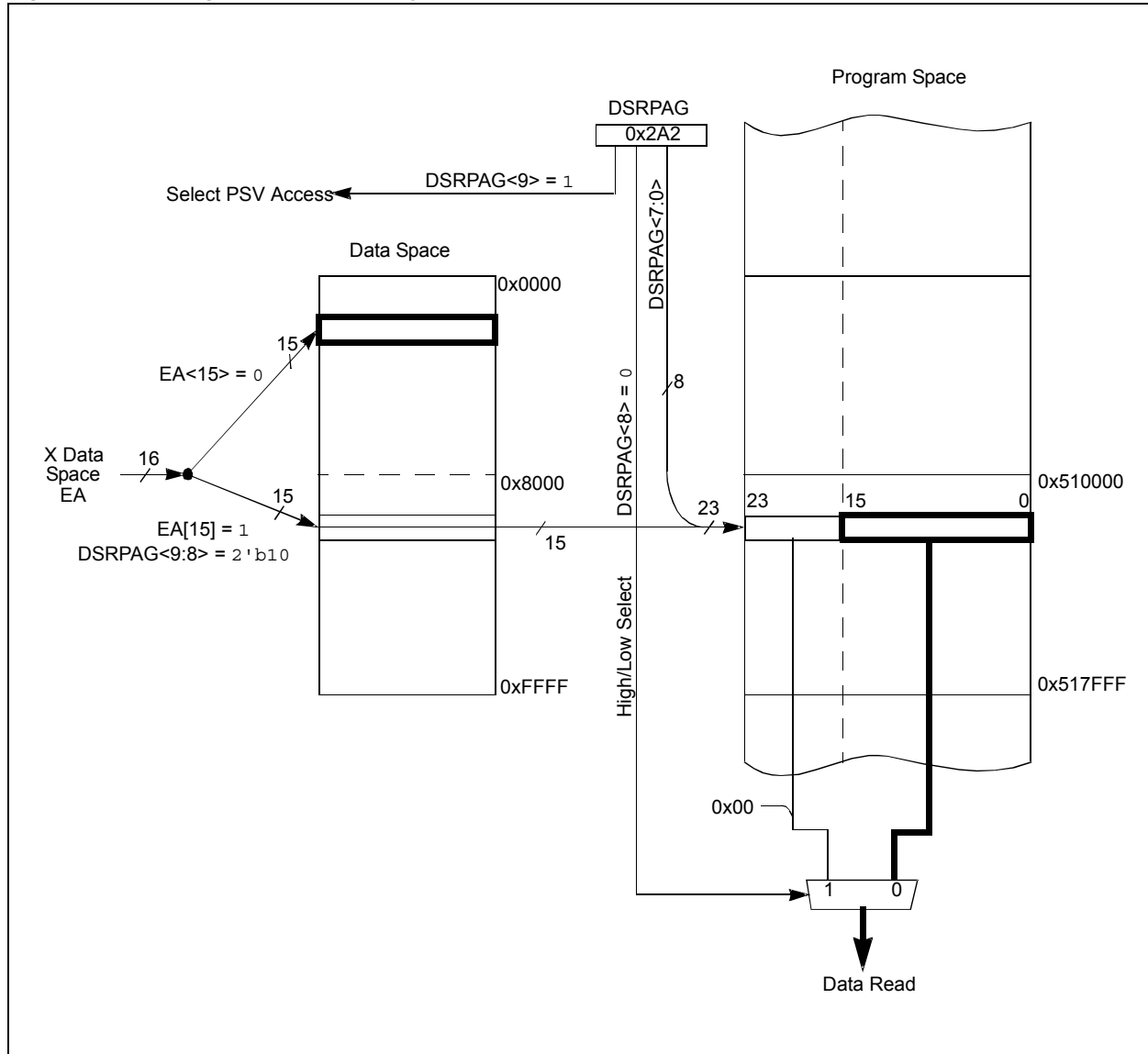
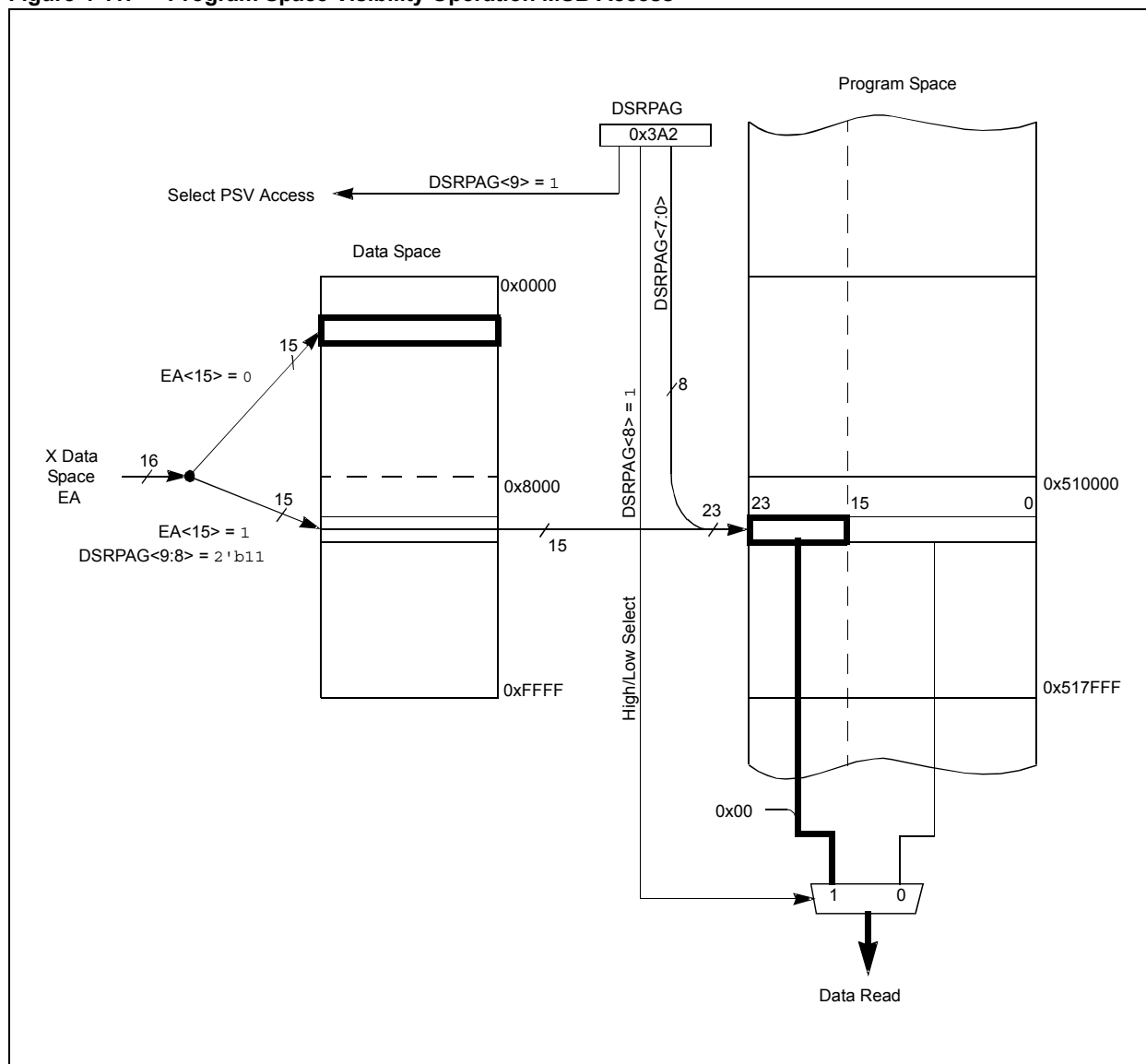


Figure 4-11: Program Space Visibility Operation MSB Access



4.5.2 PSV Mapping with X and Y Data Space

The Y data space is located outside the upper half of data space for most dsPIC33E/PIC24E variants, such that the PSV area will map into X data space. The X and Y mapping affects the way PSV is used in algorithms.

For example, the PSV mapping can be used to store coefficient data for Finite Impulse Response (FIR) filter algorithms. The FIR filter multiplies each value of a data buffer that contains historical filter input data with elements of a data buffer that contains constant filter coefficients. The FIR algorithm is executed using the `MAC` instruction within a `REPEAT` loop. Each iteration of the `MAC` instruction prefetches one historical input value and one coefficient value to be multiplied in the next iteration. One of the prefetched values must be located in X data memory space and the other must be located in Y data memory space.

To satisfy the PSV mapping requirements for the FIR filter algorithm, the user application must locate the historical input data in the Y memory space, and the filter coefficients in X memory space.

4.5.3 PSV Timing

All instructions that use PSV require five instruction cycles to complete execution.

4.5.3.1 USING PSV IN A REPEAT LOOP

Instructions that use PSV with Indirect Addressing mode using a post-modification offset of +2 or -2 within a REPEAT loop eliminate some of the cycle count overhead required for the instruction access from program memory. These instructions have an effective execution throughput of one instruction cycle per iteration. However the following iterations of the REPEAT loop will execute in five instruction cycles:

- First iteration
- Instruction execution prior to exiting the loop due to an interrupt
- Instruction execution upon re-entering the loop after an interrupt is serviced

The last iteration of the REPEAT loop will execute in six instruction cycles.

If the PSV Addressing mode uses an offset range other than +2 or -2 within a REPEAT loop, five instruction cycles are needed to execute each iteration of the loop.

Note: Unlike PSV accesses, a TBLRDx instruction requires five instruction cycles for each iteration.

4.5.3.2 PSV AND INSTRUCTION STALLS

For more information about instruction stalls using PSV, refer to **Section 2. “CPU”** (DS70359).

4.5.4 PSV Code Examples

Example 4-3 illustrates how to create a buffer, and access the buffer in the compiler-managed PSV section. The `auto_psv` space is the compiler-managed PSV section. Sections greater than 32K are allowed and automatically managed. By default, the compiler places all 'const' qualified variables into the `auto_psv` space.

When `auto_psv` is used, the compiler will save/restore the DSRPAG register dynamically, as needed. The tool chain will arrange for the DSRPAG to be correctly initialized in the compiler run-time startup code.

Example 4-3: Compiler Managed PSV Access

```
#include <p33Exxxx.h>

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

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

int vectordot (int *, int *);

int main(void)
{
    // Compiler-managed PSV
    sum = vectordot ((int *) m, x);

    while(1);
}

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

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

    return (sum);
}
```

Note: The `auto_psv` option must be used if the user application is using both PSV and EDS accesses on a device with more than 28 KB of RAM.

Example 4-4 illustrates buffer placement and access in the user-managed PSV section. The `psv` space is the user-managed PSV section.

Example 4-4: User Managed PSV Access

```
#include <p33Exxxx.h>

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

const int m[5] = {1, 2, 3, 4, 5};
const int m1[5] __attribute__((address(0x10000))) = {2, 4, 6, 8, 10};
const int m2[5] __attribute__((address(0x20000))) = {3, 6, 9, 12, 15};
int x[5] = {10, 20, 30, 40, 50};
int sum, sum1, sum2;

int vectordot (int *, int *);

int main(void)
{
    int temp;

    // Save original PSV page value
    temp = DSRPAG;

    DSRPAG = __builtin_psvpage (m1);
    sum1 = vectordot ((int *) m1, x);

    DSRPAG = __builtin_psvpage (m2);
    sum2 = vectordot ((int *) m2, x);

    // Restore original PSV page value
    DSRPAG = temp;

    sum = vectordot ((int *) m, x);

    while(1);
}

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

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

    return (sum);
}
```

Example 4-5 illustrates the placement of constant data in program memory, and accesses this data through the PSV data window using an assembly program.

Example 4-5: PSV Code Example in Assembly

```
.section .const, psv
fib_data:
    .word 0, 1, 2, 3, 5, 8, 13

    ; Start of code section
    .text

    .global __main
__main:

    ; Set DSRPAG to the page that contains the "fib_data" array
    MOVPAW #psvpag(fib_data), DSRPAG

    ; Set up W0 as a pointer to "fib_data" through the PSV data window
    MOV #psvoffset(fib_data), W0

    ; Load the data values into registers W1 - W7
    MOV [W0++], W1
    MOV [W0++], W2
    MOV [W0++], W3
    MOV [W0++], W4
    MOV [W0++], W5
    MOV [W0++], W6
    MOV [W0++], W7

done:
    BRA done

    RETURN
```

4.6 PROGRAM MEMORY WRITES

The dsPIC33E/PIC24E families of devices contain internal program Flash memory for executing user code. There are two methods by which the user application can program this memory:

- Run-Time Self-Programming (RTSP)
- In-Circuit Serial Programming™ (ICSP™)

RTSP is accomplished using TBLWT instructions. ICSP is accomplished using the SPI interface and integral bootloader software. For further details on RTSP, refer to **Section 5. "Flash Programming"** (DS70609). For further details on ICSP, refer to the "*dsPIC33E/PIC24E Flash Programming Specification*" (DS70619), which can be obtained from the Microchip web site (www.microchip.com).

4.7 PROGRAM MEMORY LOW-POWER MODE

The voltage regulator for the program Flash memory can be placed in Stand-by mode when the device is in Sleep mode, resulting in a significant reduction in device power-down current (IPD).

When the VREGSF bit (RCON<11>) is equal to '0', the Flash memory voltage regulator goes into Stand-by mode during Sleep. When the VREGSF bit is equal to '1', the Flash memory voltage regulator is active during Sleep mode.

4.8 REGISTER MAP

A summary of the registers associated with Program Memory is provided in Table 4-1.

Table 4-1: Program Memory Registers

SFR	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
TBLPAG	—	—	—	—	—	—	—	—	TBLPAG<7:0>								0000
DSRPAG	—	—	—	—	—	—	DSRPAG<9:0>										0001

Legend: — = unimplemented, read as '0'. Shaded bits are not used in the operation of Program Memory.

4.9 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 Families, but the concepts are pertinent and could be used with modification and possible limitations. The current application notes related to the Program Memory module are:

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

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

4.10 REVISION HISTORY

Revision A (September 2009)

This is the initial released version of this document.

Revision B (July 2010)

This revision includes the following updates:

- All code examples have been updated (see Example 4-1 through Example 4-5)
- Updated the Program Memory Map (see Figure 4-1)
- Updated the first paragraph and the shaded note in **4.4.1 “Table Instruction Summary”**
- Added a shaded note after Figure 4-3 with information on writing to the TBLPAG register
- Updated **4.4.2 “Table Address Generation”**
- Updated the second sentence in **4.4.3 “Program Memory Low Word Access”**
- Added the new figure Table Memory Map (see Figure 4-7) in **4.4.4 “Program Memory High Word Access”**
- Added a shaded note and updated the last paragraph in **4.5.1 “PSV Configuration”**
- Updated the Paged Data Memory Space (see Figure 4-8)
- Updated the PSV Address Generation (see Figure 4-9)
- Changed the number of required instruction cycles from two to five throughout **4.5.3 “PSV Timing”**
- Added a shaded note after Example 4-3 with information on using the `auto_psv` option
- Added a reference to the “*dsPIC33E/PIC24E Flash Programming Specification*” (DS70619) to **4.6 “Program Memory Writes”**

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.

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights.

Trademarks

The Microchip name and logo, the Microchip logo, dsPIC, KEELOQ, KEELOQ logo, MPLAB, PIC, PICmicro, PICSTART, PIC³² logo, rfPIC and UNI/O are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.


FilterLab, Hampshire, HI-TECH C, Linear Active Thermistor, MXDEV, MXLAB, SEEVAL and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Analog-for-the-Digital Age, Application Maestro, CodeGuard, dsPICDEM, dsPICDEM.net, dsPICworks, dsSPEAK, ECAN, ECONOMONITOR, FanSense, HI-TIDE, In-Circuit Serial Programming, ICSP, Mindi, MiWi, MPASM, MPLAB Certified logo, MPLIB, MPLINK, mTouch, Octopus, Omniscient Code Generation, PICC, PICC-18, PICDEM, PICDEM.net, PICkit, PICTail, REAL ICE, rLAB, Select Mode, Total Endurance, TSHARC, UniWinDriver, WiperLock and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.

© 2010, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

 Printed on recycled paper.

ISBN: 978-1-60932-386-8

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.

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
== ISO/TS 16949:2002 ==



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://support.microchip.com>
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

Kokomo

Kokomo, IN
Tel: 765-864-8360
Fax: 765-864-8387

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: 86-10-8528-2100
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 - 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: 886-7-536-4818
Fax: 886-7-536-4803

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 - Druenen

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