



---

---

## 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 Register .....                               | 4-4  |
| 4.3 | Program Counter .....                                | 4-6  |
| 4.4 | Program Memory Access Using Table Instructions ..... | 4-7  |
| 4.5 | Program Space Visibility from Data Space.....        | 4-12 |
| 4.6 | Program Memory Writes .....                          | 4-16 |
| 4.7 | Register Maps.....                                   | 4-17 |
| 4.8 | Related Application Notes.....                       | 4-18 |
| 4.9 | Revision History .....                               | 4-19 |

**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 dsPIC33F/PIC24H devices.

Please consult the note at the beginning of the “**Program 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>

## 4.1 PROGRAM MEMORY ADDRESS MAP

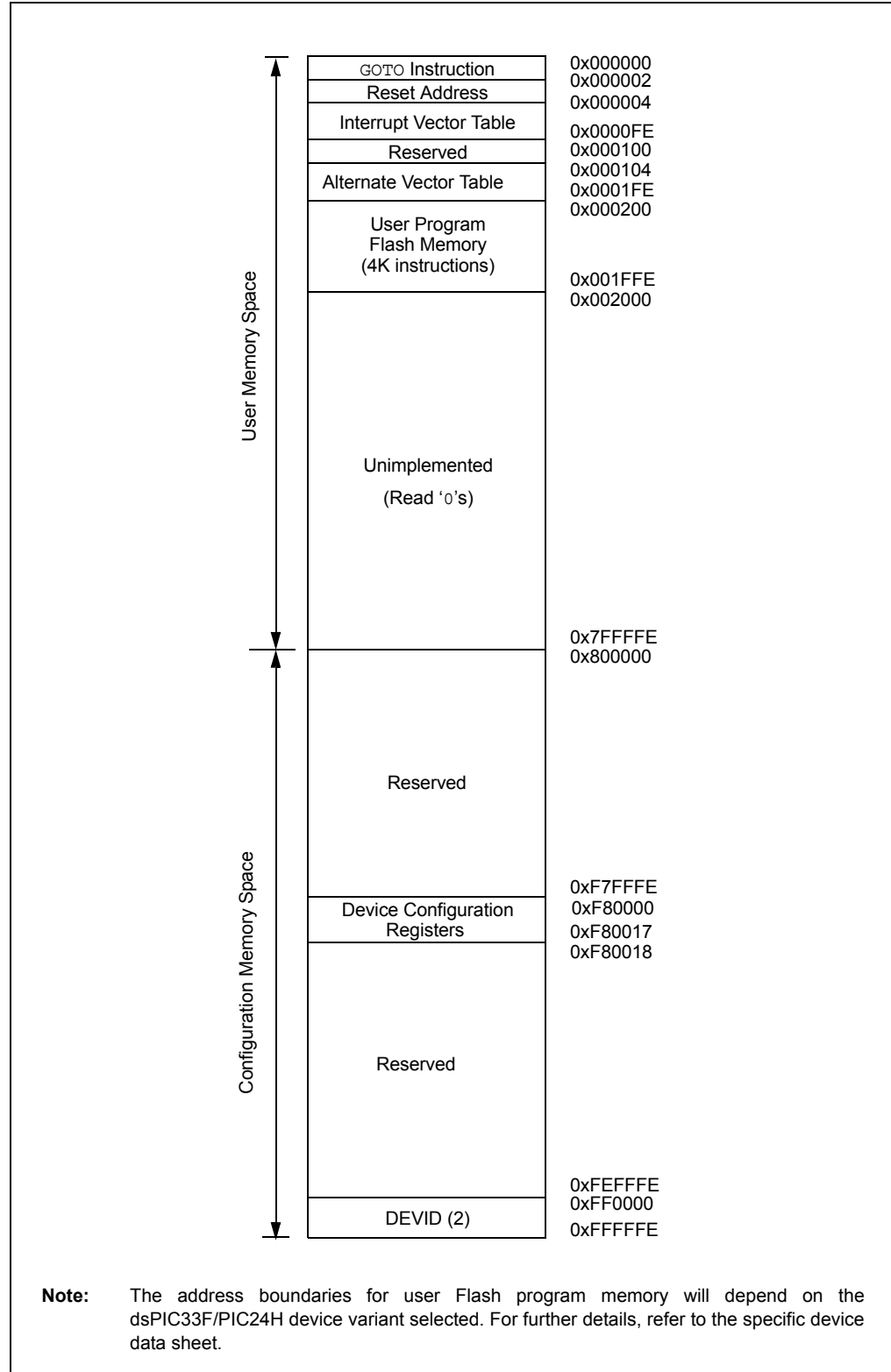
Figure 4-1 illustrates that the dsPIC33F/PIC24H devices have a 4M x 24-bit program memory address space. Three methods are available for accessing program space.

- Through the 23-bit (Program Counter) PC
- Through table read (TBLRD) and table write (TBLWT) instructions
- By mapping a 32-Kbyte segment of program memory into the data memory address space

The program memory map is divided into the user program space and the user configuration space. The user program space contains the Reset vector, interrupt vector tables, and program memory. The user configuration space contains nonvolatile configuration bits for setting device options and the device ID locations.

**Note:** If the RETURN instruction is placed at the end of the program memory, an Illegal Address Error Trap will be generated by the device during run-time, as a result of the instruction prefetch operation which would try to preload the next instruction from an unimplemented memory location. The solution is to leave 2 extra instruction words available after the RETURN instruction so that the compiler can place NOP and RESET instructions at the end of the program memory.

**Figure 4-1: Example Program Memory Map**



# dsPIC33F/PIC24H Family Reference Manual

## 4.2 CONTROL REGISTER

**Register 4-1: CORCON: Core Control Register**

|        |     |     |       |       |         |     |       |
|--------|-----|-----|-------|-------|---------|-----|-------|
| U-0    | U-0 | U-0 | R/W-0 | R/W-0 | R-0     | R-0 | R-0   |
| —      | —   | —   | US    | EDT   | DL<2:0> |     |       |
| bit 15 |     |     |       |       |         |     | bit 8 |

|       |       |       |        |       |       |       |       |
|-------|-------|-------|--------|-------|-------|-------|-------|
| R/W-0 | R/W-0 | R/W-1 | R/W-0  | R/C-0 | R/W-0 | R/W-0 | R/W-0 |
| SATA  | SATB  | SATDW | ACCSAT | IPL3  | PSV   | RND   | IF    |
| 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 15-3      **Not used by the Program Memory**  
 For full description of the CORCON bits, refer to **Section 2. "CPU"** (DS70204).
- bit 2      **PSV: Program Space Visibility in Data Space Enable bit**  
 1 = Program space visible in data space  
 0 = Program space not visible in data space
- bit 1-0      **Not used by the Program Memory**  
 For full description of the CORCON bits, refer to **Section 2. "CPU"** (DS70204).

**Register 4-2: PSVPAG: PSV 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 |
| PSV Address Page bits |       |       |       |       |       |       |       |
| 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      **PSV Address Page bits**  
 The 8-bit PSV Address Page bits are concatenated with the 15 Least Significant bits (LSBs) of the W register, to form a 23-bit effective program memory address.

## Section 4. Program Memory

**Register 4-3: 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 |
| Table Address Page bits |       |       |       |       |       |       |       |
| 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      **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.

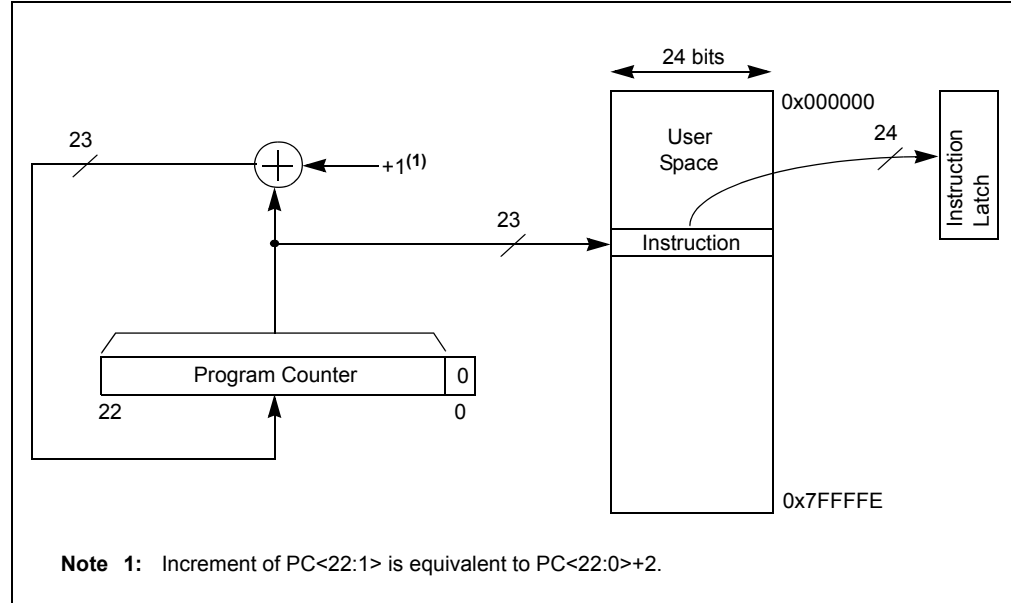
## 4.3 PROGRAM COUNTER

The Program Counter (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 or table instructions. For instruction fetches via the PC, the byte select bit is not required. Therefore, PC<0> is always set to '0'.

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

**Figure 4-2: Instruction Fetch Example**



## 4.4 PROGRAM MEMORY ACCESS USING TABLE INSTRUCTIONS

The TBLRDL and TBLWTL instructions offer a direct method of reading or writing the least significant word (lsw) of any address within program space without going through data space, which is preferable for some applications. The TBLRDH and TBLWTH instructions are the only method whereby the upper 8 bits of a program word can be accessed as data.

### 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.

**Note:** Detailed code examples using table instructions are found in **Section 5. “Flash Programming”** (DS70191).

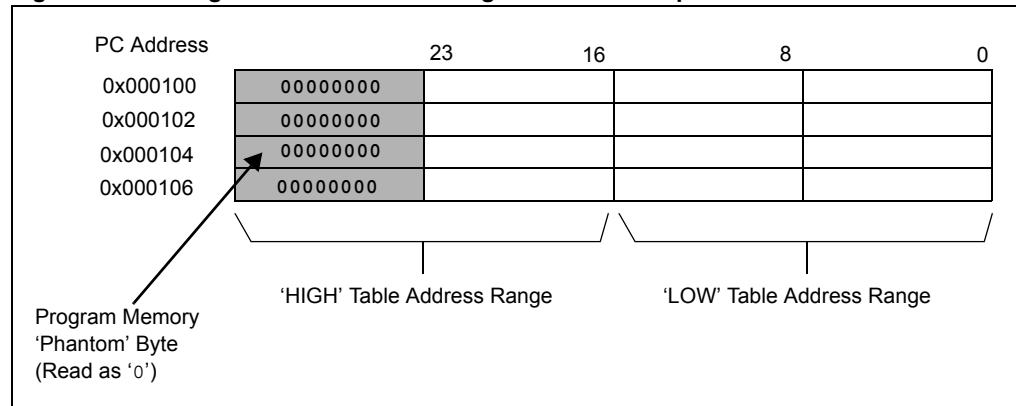
The four available table instructions are:

- 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. As 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.

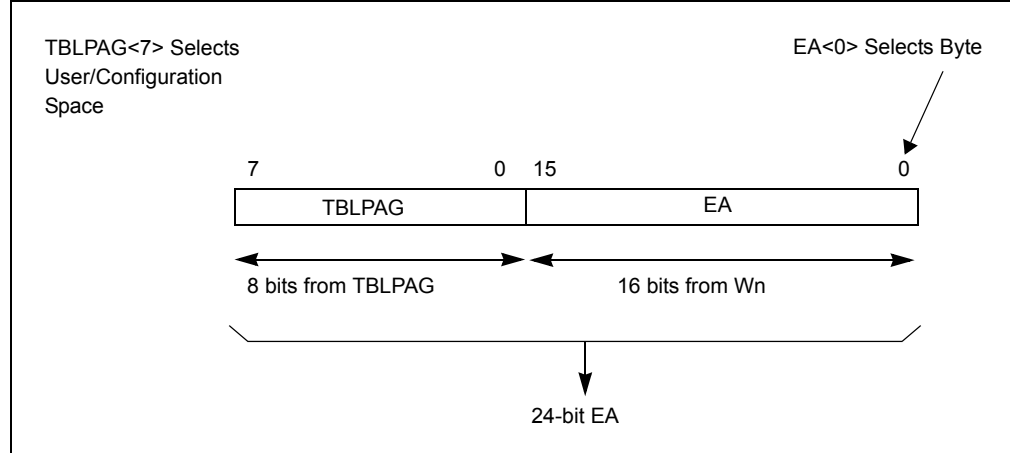
**Figure 4-3: High and Low Address Regions for Table Operations**



## 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 Data Table Page register (TBLPAG), to form a 23-bit effective program space address plus a byte select bit. As there are 15 bits of program space address provided from the W register, the data table page size in program memory is, therefore, 32K words.

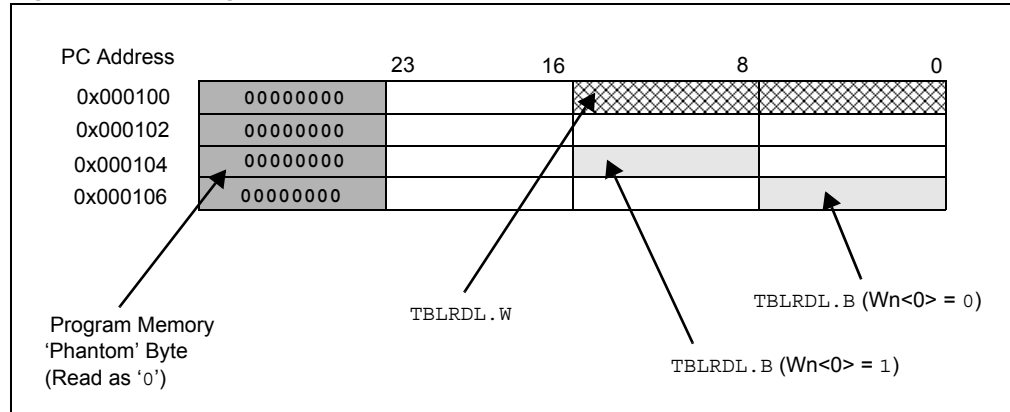
**Figure 4-4: Address Generation for Table Operations**



## 4.4.3 Program Memory Low Word Access

The TBLRDL and TBLWTL instructions are used to access the lower 16 bits of program memory data. The LSB of the W register address 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 TBLRDL and TBLWTL instructions.

**Figure 4-5: Program Data Table Access (lsbw)**

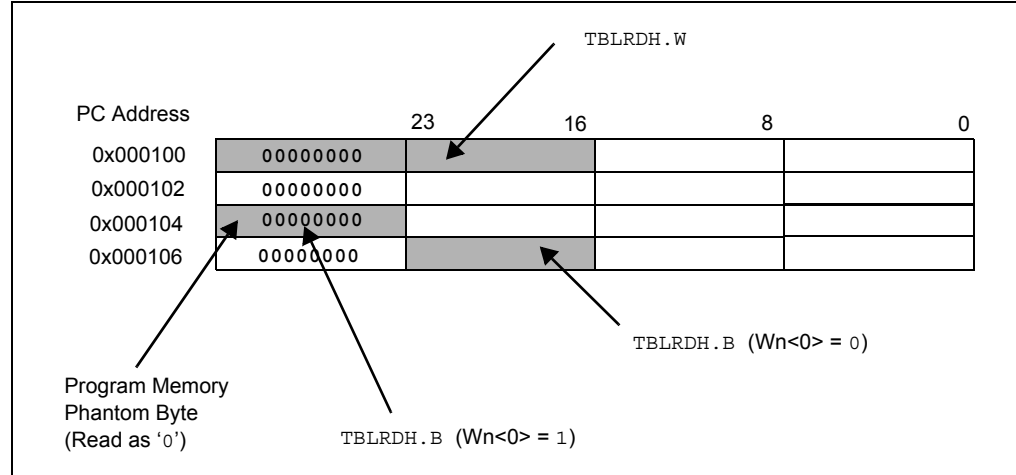




## 4.4.4 Program Memory High Word Access

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 (MS Byte)**



## 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 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 primarily provided for array program/verification purposes and for applications that require compressed data storage.

## 4.4.6 Program Memory Access Using Table Instructions Example

Example 4-1 uses table instructions to erase the program memory page starting at the address 0x12000, and programs the values 0x123456 and 0x789ABC into addresses 0x12000 and 0x12002, respectively.

**Note:** For more information on the unlocking sequence, refer to **Section 5. "Flash Programming"** (DS70191).

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

```
#define PM_ROW_ERASE 0x4042
#define PM_ROW_WRITE 0x4001
#define CONFIG_WORD_WRITE0X4000

unsigned long Data;

/* Erase 512 instructions starting at address 0x12000 */
MemWriteLatch(0x1, 0x2000,0x0,0x0);
MemCommand(PM_ROW_ERASE);

/* Write 0x12345 into program address 0x12000 */
MemWriteLatch(0x1, 0x2000,0x0012,0x3456);
MemCommand(PM_ROW_WRITE);

/* Write 0x789ABC into program address 0x12002 */
MemWriteLatch(0x1, 0x2002,0x0078,0x9ABC);
MemCommand(PM_ROW_WRITE);

/* Read program addresses 0x12000 and 0x12002 */
Data = MemReadLatch(0x1, 0x2000);
Data = MemReadLatch(0x1, 0x2002);

;*****
;_MemWriteLatch:
;
;W0 = TBLPAG
;W1 = Wn
;W2 = WordHi
;W3 = WordLo
;no return values
_MemWriteLatch:
    mov W0, TBLPAG
    tblwtl W3, [W1]
    tblwth W2, [W1]
    return
;*****
;_MemReadLatch:
;
;W0 = TBLPAG
;W1 = Wn
;return: data in W1:W0
_MemReadLatch:
    mov W0, TBLPAG
    tblrdl [W1], W0
    tblrdh [W1], W1
    return
;*****
;_MemCommand:
;
;W0 = NVMCON
;no return values
_WriteCommand:
    mov W0, NVMCON
    mov #0x55, W0; Unlock sequence
    mov W0, NVMKEY
    mov #0xAA, W0
    mov W0, NVMKEY
    bset NVMCON, #WR
    nop ; Required
    nop
Loop: btsc NVMCON, #WR; Wait for write end
    bra Loop
return
```

Example 4-2 uses the `space(prog)` attribute to allocate the buffer in program memory. The MPLAB® C30 built-in functions, such as `builtin_tblpage` and `builtin_tbloffset`, can be used to access the buffer.

### Example 4-2: Using MPLAB® C30 Built-in Functions to Access Program Memory

```
#include <p33fxxxx.h>

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

unsigned lowWord, tbloffset;

int main(void)
{
    TBLPAG = __builtin_tblpage(&prog_data);
    tbloffset = __builtin_tbloffset(&prog_data);
    lowWord = __builtin_tblrdl(tbloffset + 6); // Load 0x3333 into lowWord

    for(;;);
    return 0;
}
```

## 4.5 PROGRAM SPACE VISIBILITY FROM DATA SPACE

The upper 32 Kbytes of the dsPIC33F/PIC24H data memory address space can optionally be mapped into any 16K word program space page. This mode of operation, called Program Space Visibility (PSV), 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

Program Space Visibility is enabled by setting the PSV bit in the Core Control register (`CORCON<2>`). A description of the `CORCON` register is found in **Section 2. “CPU”** (DS70204).

When PSV is enabled, each data space address in the upper half of the data memory map will map directly into a program address (see Figure 4-7). The PSV window allows access to the lower 16 bits of the 24-bit program word. The upper 8 bits of the program memory data should be programmed to force an illegal instruction, or a `NOB` instruction, to maintain machine robustness. Table instructions provide the only method of reading the upper 8 bits of each program memory word.

Figure 4-8 illustrates how the PSV address is generated. The 15 LSb 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 space or a normal access from data memory space. If a W register effective address of 0x8000 or greater is used, the data access will occur from program memory space when PSV is enabled. All data access occurs from data memory when the W register effective address is less than 0x8000.

Figure 4-8 illustrates how the remaining address bits are provided by the Program Space Visibility Page Address register (`PSVPAG<7:0>`). The `PSVPAG` bits are concatenated with the 15 LSb of the W register, holding the effective address to form a 23-bit program memory address. PSV can only be used to access values in 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.

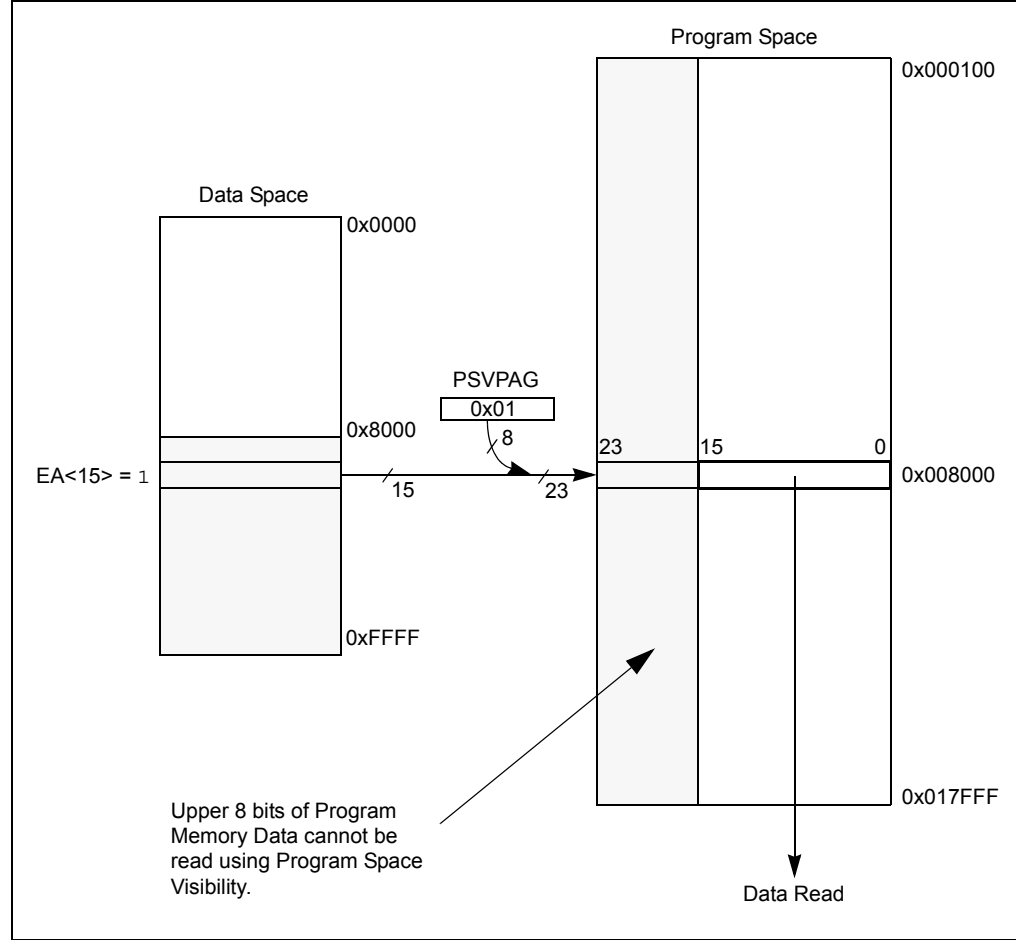
### 4.5.2 PSV Mapping with X and Y Data Spaces

The Y data space is located outside of the upper half of data space for most dsPIC33F/PIC24H variants, such that the PSV area will map into X data space. The X and Y mapping affect the way PSV is used in algorithms.

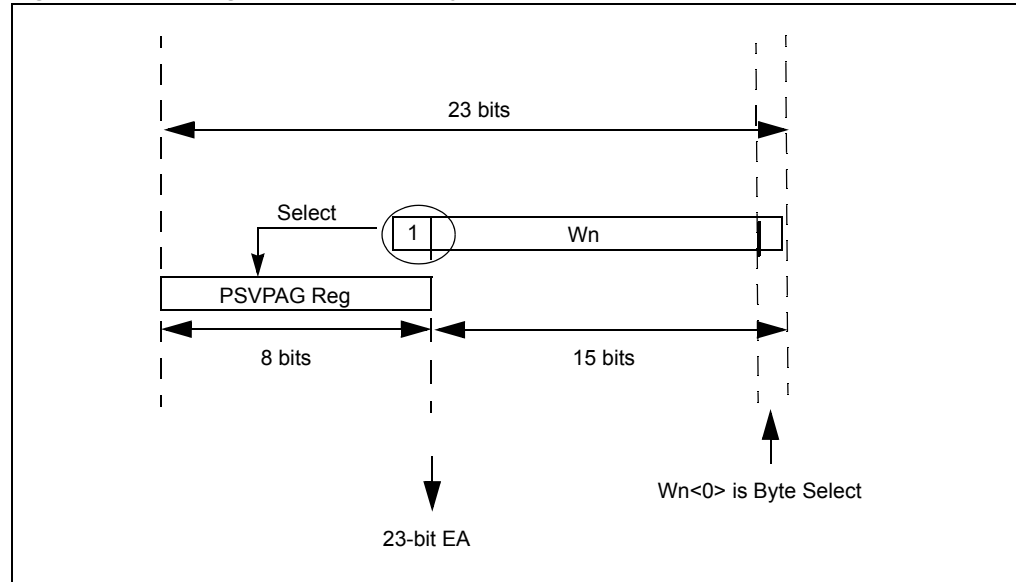
For example, the PSV mapping can be used to store coefficient data for Finite Impulse Response (FIR) filter algorithms. The FIR filter multiplies each value of a data buffer containing historical filter input data with elements of a data buffer that contains constant filter coefficients. The FIR algorithm 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.

**Figure 4-7: Program Space Visibility Operation**



**Figure 4-8: Program Space Visibility Address Generation**



## 4.5.3 PSV Timing

Instructions that use PSV require two extra instruction cycles to complete execution, except for the following instructions that require only one extra cycle to complete execution:

- The MAC class of instructions with data prefetch operands
- All MOV instructions including the MOV.D instruction

The additional instruction cycles are used to fetch the PSV data on the program memory bus.

### 4.5.3.1 USING PSV IN A REPEAT LOOP

Instructions that use PSV within a REPEAT loop eliminate the extra instruction cycle(s) required for the data access from program memory, therefore incurring no overhead in execution time. However, the following iterations of the REPEAT loop incur an overhead of two instruction cycles to complete execution:

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

### 4.5.3.2 PSV AND INSTRUCTION STALLS

For more information about instruction stalls using PSV, refer to **Section 2. “CPU”** (DS70204).

## 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. If the size of this section exceeds 32K, the linker will give an error. The tool chain will arrange for the PSVPAG to be correctly set at program start-up. By default, the compiler places all 'const' qualified variables into the `auto_psv` space.

### Example 4-3: Compiler Managed PSV Access

```
#include "p33fxxxx.h"

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

main()
{
    // Compiler Managed PSV
    sum=vectordot(m,x);
}

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

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

    return(sum);
}
```

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 "p33fxxxx.h"

const int m[5] = {1, 2, 3, 4, 5};
int m1[5] __attribute__((space(psv))) = {1, 2, 3, 4, 5};
int m2[5] __attribute__((space(psv),address(0xA000))) = {1, 2, 3, 4, 5};
int x[5] = {10, 20, 30, 40, 50};
int sum, sum1, sum2;

main()
{
    int temp;

    // User Managed PSV
    temp=PSVPAG; // Save auto_psv page

    PSVPAG = __builtin_psvpage(&m1);
    CORCONbits.PSV = 1;
    sum1=vectordot(m1,x);

    PSVPAG = __builtin_psvpage(&m2);
    sum2=vectordot(m2,x);

    PSVPAG=temp; // Restore auto_psv page

    // Compiler Managed PSV
    sum=vectordot(m,x);
}

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 constant data placement in program memory and performs accessing of this data through the PSV data window using an assembly program.

### Example 4-5: PSV Code Example in Assembly

```
.include "p33fxxxx.inc"
.section .const,psv
fib_data:
    .word 0, 1, 2, 3, 5, 8, 13

;Start of Code section
.text

.global __reset
__reset:
    ; Enable Program Space Visibility
    bset.b CORCONL, #PSV

    ; Set PSVPAG to the page that contains the fib_data array
    mov #psvpag(fib_data), w0
    mov w0, __PSVPAG

    ; Make a pointer to fib_data in the PSV data window
    mov #psvoffset(fib_data), w0

    ; Load the first data value
    mov [w0++], w1
```

## 4.6 PROGRAM MEMORY WRITES

The dsPIC33F/PIC24H family of devices contains 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 about RTSP, refer to **Section 5. “Flash Programming”** (DS70191). ICSP specifications can be downloaded from the Microchip Technology web site ([www.microchip.com](http://www.microchip.com)).



## 4.7 REGISTER MAPS

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 | —      | —      | —      | —      | —      | —       | —     | —     | Table Page Address Pointer Register                     |       |       |        |       |       |       |       | 0000       |
| PSVPAG | —      | —      | —      | —      | —      | —       | —     | —     | Program Memory Visibility Page Address Pointer Register |       |       |        |       |       |       |       | 0000       |
| CORCON | —      | —      | —      | US     | EDT    | DL<2:0> |       |       | SATA  | SATB  | SATDW | ACCSAT | IPL3  | PSV   | RND   | IF    | 0000       |

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

## 4.8 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 dsPIC33F/PIC24H device family, 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. |                    |

**Note:** For additional Application Notes and code examples for the dsPIC33F/PIC24H family of devices, visit the Microchip web site ([www.microchip.com](http://www.microchip.com)).

### 4.9 REVISION HISTORY

#### Revision A (March 2007)

This is the initial released version of this document.

#### Revision B (April 2007)

Minor updates made to document.

#### Revision C (July 2008)

This revision incorporates the following updates:

- Rearranged the following sections: (see 4.2 “Control Register”), (see 4.6 “Program Memory Writes”)
- Registers:
  - PSVPAG: PSV Page Register (see Register 4-2)
  - TBLPAG: Table Page Register (see Register 4-3)
- Examples:
  - Using MPLAB C30 Built-in Functions to Access Program Memory (see Example 4-2)
  - Compiler Manager PSV Access (see Example 4-3)
  - User Manager PSV Access (see Example 4-4)
  - PSV Code Example in Assembly (see Example 4-5)
- Added Register Maps section (see **Example 4-5: “PSV Code Example in Assembly”**)
- Additional minor corrections such as language and formatting updates are incorporated in the entire document

#### Revision D (July 2010)

This revision includes the following updates:

- Updated the code in Example 4-1 and Example 4-2
- Additional minor updates to text and formatting have been incorporated throughout the document

NOTES:

---

**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, PIC<sup>32</sup> logo, rPIC 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: 978-1-60932-367-7

*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<sup>®</sup> MCUs and dsPIC<sup>®</sup> DSCs, KEELOQ<sup>®</sup> 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](http://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 - 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