
dsPIC33/PIC24 Program Memory

HIGHLIGHTS

This section of the manual contains the following topics:

1.0	Program Memory Address Map	1-2
2.0	Control Registers	1-5
3.0	Program Counter	1-6
4.0	Reading Program Memory Using Table Instructions.....	1-7
5.0	Program Space Visibility from Data Space.....	1-11
6.0	Program Memory Writes	1-15
7.0	Error Correcting Code (ECC).....	1-15
8.0	Program Memory Low-Power Mode	1-16
9.0	Register Map.....	1-17
10.0	Related Application Notes.....	1-18
11.0	Revision History	1-19

dsPIC33/PIC24 Family Reference Manual

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 dsPIC33/PIC24 devices.

Please consult the note at the beginning of the “**Memory Organization**” and “**Flash Program Memory**” chapters 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 Website at: <http://www.microchip.com>.

1.0 PROGRAM MEMORY ADDRESS MAP

dsPIC33/PIC24 devices have a 4M x 24-bit program memory address space. Figure 1-1 shows a typical program memory map for dsPIC33/PIC24 family devices. Figure 1-2 provides an example of the program memory map for devices that also implement auxiliary memory.

The program memory space can be accessed through the following methods:

- 23-bit Program Counter (PC)
- Table Read (TBLRD) instruction
- Program Space Visibility (PSV) mapping any 32-Kbyte segment of program memory into the data memory address space

The program memory address space in dsPIC33/PIC24 devices is divided into two equal halves, referred to as the User Memory Space and the Configuration Memory Space.

The User Memory Space is comprised of the following areas:

- User Program Flash Memory
- Flash Configuration Bytes (if applicable; refer to the “**Special Features**” chapter of the specific device data sheet for availability)
- Auxiliary Program Flash Memory (if applicable; refer to the “**Memory Organization**” chapter of the specific device data sheet for availability)

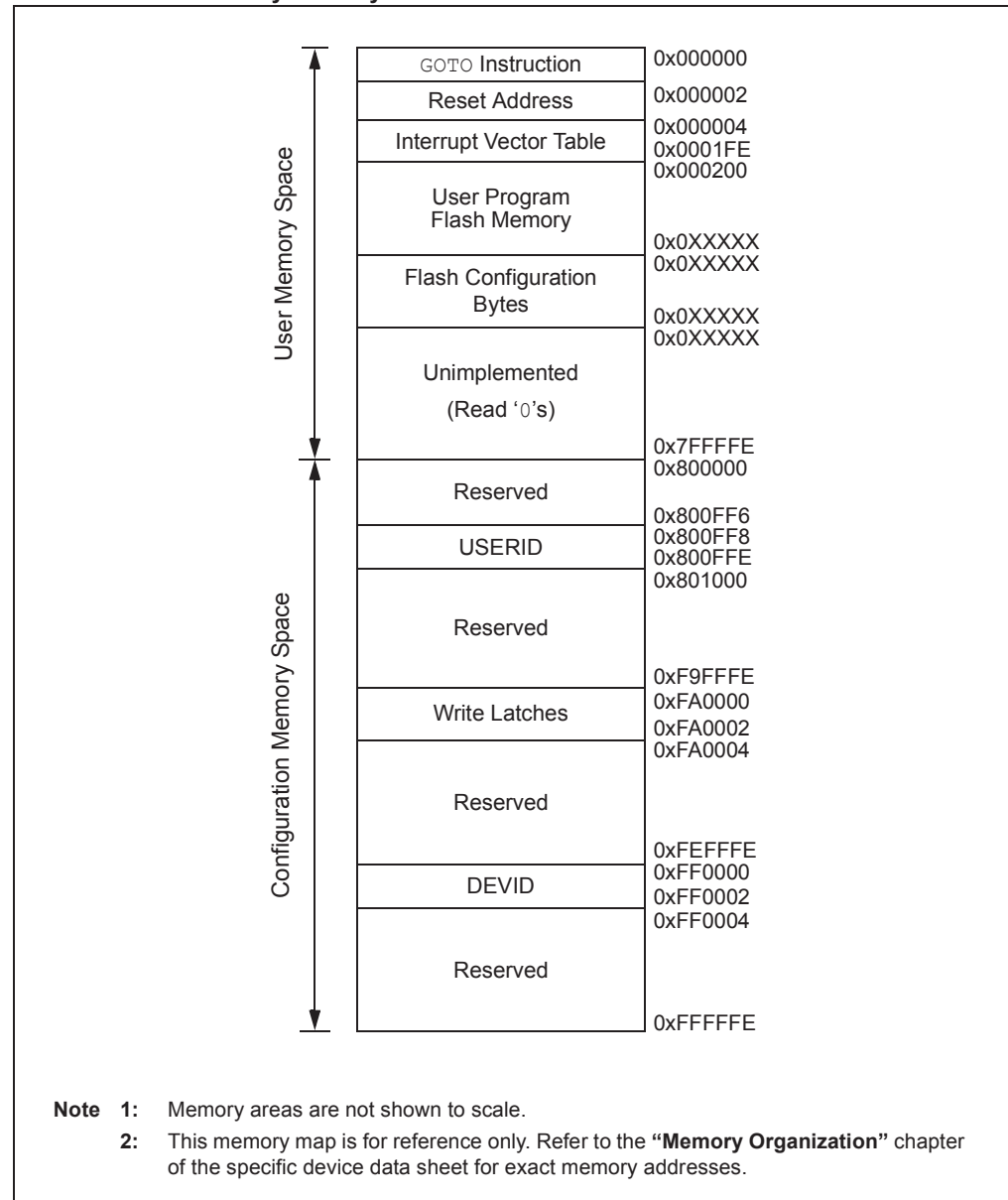
For devices that support auxiliary program Flash memory, instructions in the auxiliary program Flash memory can be executed by the CPU, without stalling it, while the user program memory is being erased and/or programmed. Similarly, instructions in the user program memory can be executed by the CPU while the auxiliary program memory is being erased and/or programmed, without Stalls.

The Configuration Memory Space consists of the following areas:

- Device Configuration registers (if applicable; refer to the “**Special Features**” chapter of the specific device data sheet for availability)
- Either USERID or One-Time-Programmable (OTP) locations to store serialization and other application-specific data (if applicable; refer to the “**Special Features**” chapter of the device data sheet for specific implementation details)
- Write latches, which are used for programming user and auxiliary Flash memory (the number of latches is device-dependent; refer to the “**Memory Organization**” chapter of the specific device data sheet for the number of available write latches)
- DEVID locations, which contain the device ID and revision ID. Refer to the “*Programming Specification*” for your device, which is available for download from the Microchip Website (www.microchip.com) for more information

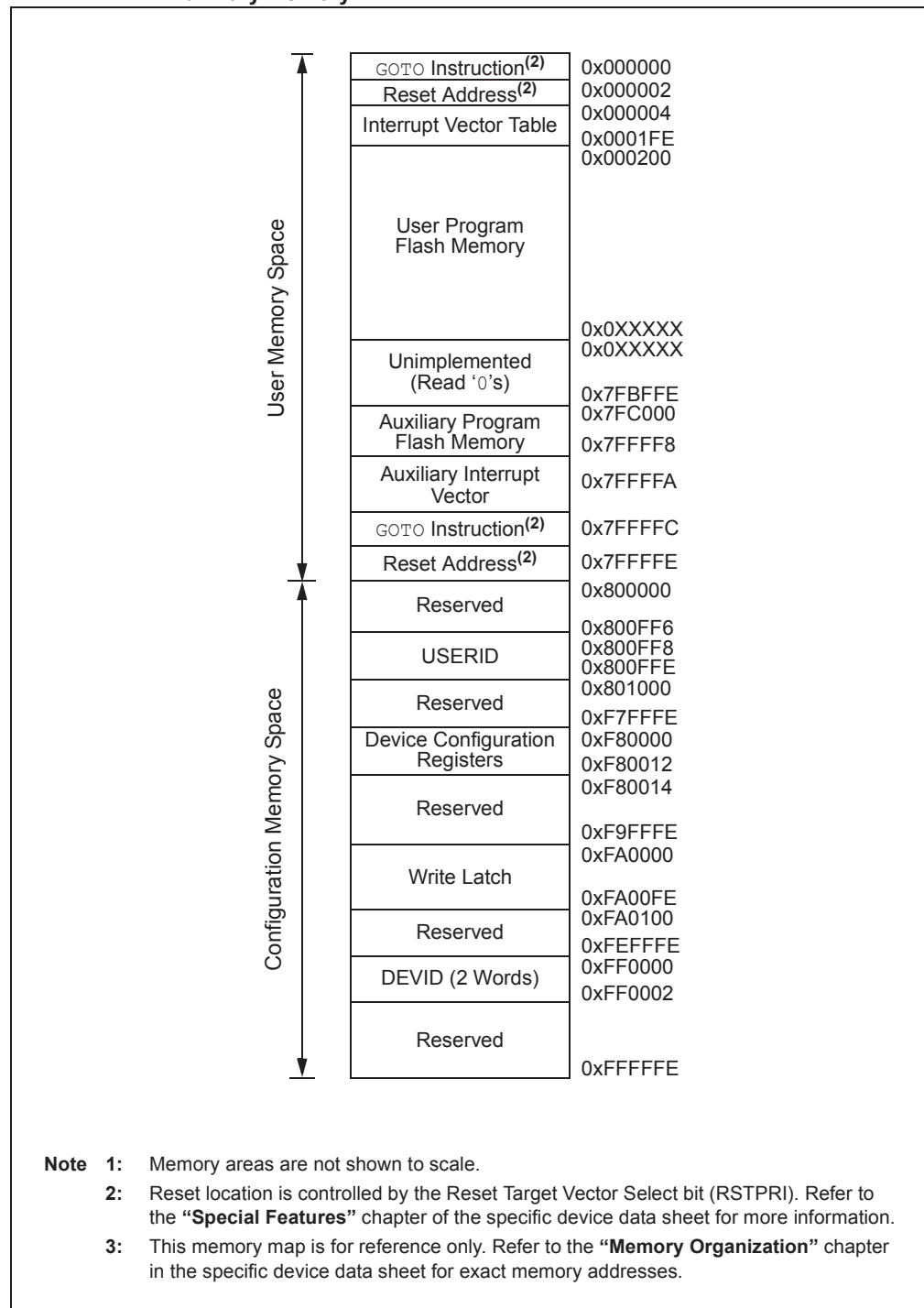
dsPIC33/PIC24 Program Memory

Figure 1-1: dsPIC33/PIC24 Program Memory Map for Devices without Auxiliary Memory



dsPIC33/PIC24 Family Reference Manual

Figure 1-2: dsPIC33/PIC24 Program Memory Map for Devices with Auxiliary Memory



dsPIC33/PIC24 Program Memory

2.0 CONTROL REGISTERS

There are two registers that can be used to manage the program Flash:

- TBLPAG: Table Page Register
- DSRPAG: Data Space Read Page Register

Register 2-1: 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 Page Address 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 2-2: DSRPAG: Data Space Read Page Register^(1,2,3)

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 Data Space (DS) window will cause an address error trap.

2: DSRPAG is reset to 0x001.

3: The Program Space (PS) can be read using DSRPAG values of 0x200 or greater.

dsPIC33/PIC24 Family Reference Manual

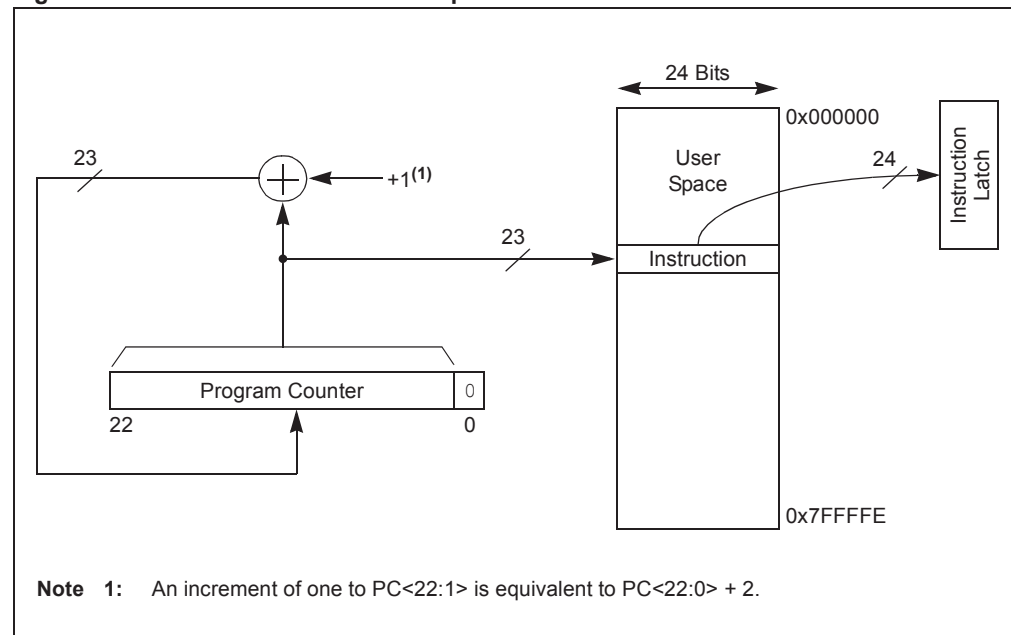
3.0 PROGRAM COUNTER

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 [Section 5.0 "Program Space Visibility from Data Space"](#).

[Figure 3-1](#) illustrates an instruction fetch example. Note that incrementing PC<22:1> by one is equivalent to adding two to PC<22:0>.

Figure 3-1: Instruction Fetch Example



dsPIC33/PIC24 Program Memory

4.0 READING PROGRAM MEMORY USING TABLE INSTRUCTIONS

The Table Read instruction offers a direct method of reading the least significant word (lsword) and the Most Significant Byte (MSB) of any instruction word, within Program Space, without going through Data Space, which is preferable for some applications. For information on programming Flash memory, refer to the “dsPIC33/PIC24 Family Reference Manual”, “Flash Programming” (DS70000609), which is available from the Microchip website (www.microchip.com).

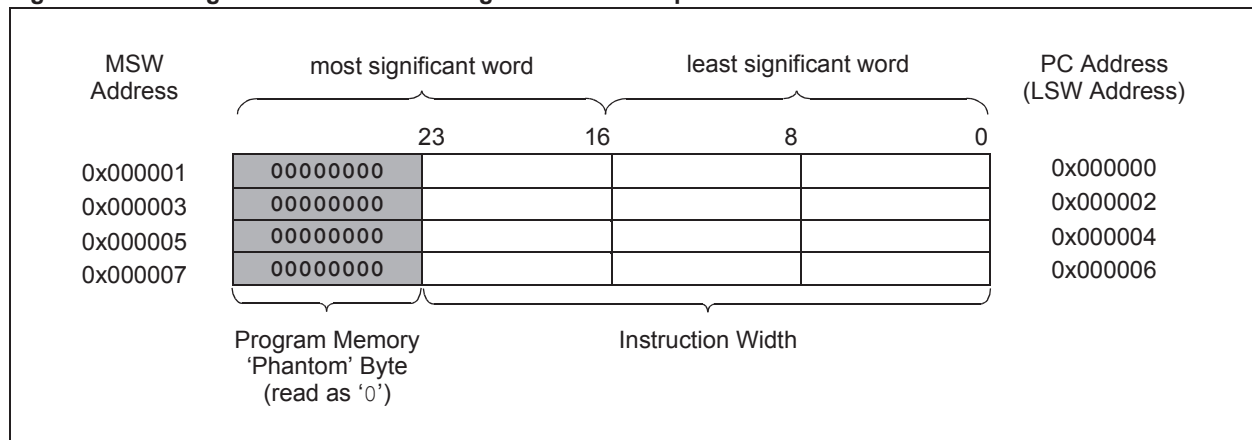
4.1 Table Instruction Summary

A set of table instructions is provided to move byte-sized or word-sized data between Program Space and Data Space. The Table Read instructions, in conjunction with the TBLPAG register, are used to read from the program memory space into data memory space. There are two Table Read instructions: TBLRDL (Table Read Low) and TBLRDH (Table Read 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 4-1). 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).

The TBLRDL instruction accesses the least significant data word of the program memory and TBLRDH accesses 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 4-1: High and Low Address Regions for Table Operations



4.2 Table Address Generation

Figure 4-2 illustrates how for all table instructions, a W register address value is concatenated with the 8-bit Table Page (TBLPAG) register 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. Figure 4-3 shows the organization of the table pages in the Program Space.

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

Figure 4-2: Address Generation for Table Operations

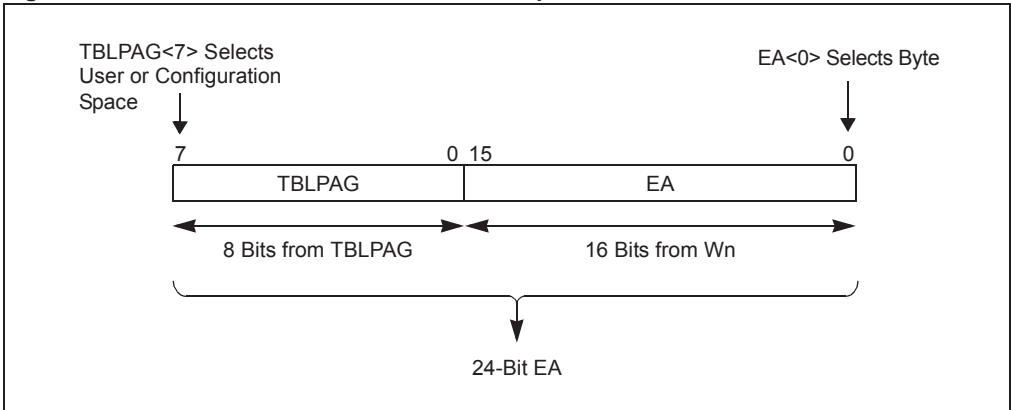


Figure 4-3: Table Page Memory Map

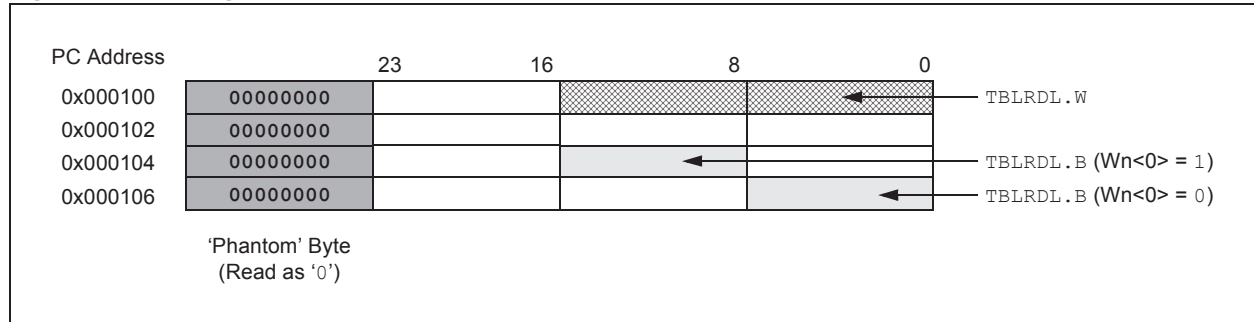
TBLRDH MSB Access Enabled	TBLRDL LSW Access Enabled	24-Bit Program Space Address [TBLPAG<7:0>:Wn<15:0>]
TABLE PAGE 0x00	TABLE PAGE 0x00	0x000000
TABLE PAGE 0x01	TABLE PAGE 0x01	0x010000
TABLE PAGE 0x02	TABLE PAGE 0x02	0x020000
TABLE PAGE 0x03	TABLE PAGE 0x03	0x030000
TABLE PAGE 0x04	TABLE PAGE 0x04	0x040000
TABLE PAGE 0x05	TABLE PAGE 0x05	0x050000
TABLE PAGE 0x06	TABLE PAGE 0x06	0x060000
TABLE PAGE 0x07	TABLE PAGE 0x07	0x070000
TABLE PAGE 0x08	TABLE PAGE 0x08	0x080000
TABLE PAGE 0x09	TABLE PAGE 0x09	0x090000
TABLE PAGE 0x0A	TABLE PAGE 0x0A	0x0A0000
TABLE PAGE 0x0B	TABLE PAGE 0x0B	0x0B0000
TABLE PAGE 0x0C	TABLE PAGE 0x0C	0x0C0000
TABLE PAGE 0x0D	TABLE PAGE 0x0D	0x0D0000
TABLE PAGE 0x0E	TABLE PAGE 0x0E	0x0E0000
TABLE PAGE 0x0F	TABLE PAGE 0x0F	0x0F0000
TABLE PAGE 0x10	TABLE PAGE 0x10	0x100000
TABLE PAGE 0x11	TABLE PAGE 0x11	0x110000
TABLE PAGE 0x12	TABLE PAGE 0x12	0x120000
TABLE PAGE 0x13	TABLE PAGE 0x13	0x130000
TABLE PAGE 0x14	TABLE PAGE 0x14	0x140000
TABLE PAGE 0x15	TABLE PAGE 0x15	0x150000
TABLE PAGE 0x16	TABLE PAGE 0x16	0x160000
TABLE PAGE 0x17	TABLE PAGE 0x17	0x170000
TABLE PAGE 0x18	TABLE PAGE 0x18	0x180000
TABLE PAGE 0x19	TABLE PAGE 0x19	0x190000
TABLE PAGE 0x1A	TABLE PAGE 0x1A	0x1A0000
TABLE PAGE 0x1B	TABLE PAGE 0x1B	0x1B0000
TABLE PAGE 0x1C	TABLE PAGE 0x1C	0x1C0000
TABLE PAGE 0x1D	TABLE PAGE 0x1D	0x1D0000
TABLE PAGE 0x1E	TABLE PAGE 0x1E	0x1E0000
TABLE PAGE 0x1F	TABLE PAGE 0x1F	0x1F0000
TABLE PAGE 0x20	TABLE PAGE 0x20	0x200000
TABLE PAGE 0x21	TABLE PAGE 0x21	0x210000
TABLE PAGE 0x22	TABLE PAGE 0x22	0x220000
TABLE PAGE 0x23	TABLE PAGE 0x23	0x230000
TABLE PAGE 0x24	TABLE PAGE 0x24	0x240000
TABLE PAGE 0x25	TABLE PAGE 0x25	0x250000
TABLE PAGE 0x26	TABLE PAGE 0x26	0x260000
TABLE PAGE 0x27	TABLE PAGE 0x27	0x270000
TABLE PAGE 0x28	TABLE PAGE 0x28	0x280000
TABLE PAGE 0x29	TABLE PAGE 0x29	0x290000
TABLE PAGE 0x2A	TABLE PAGE 0x2A	0x2A0000
TABLE PAGE 0x2B	TABLE PAGE 0x2B	0x2B0000
TABLE PAGE 0x2C	TABLE PAGE 0x2C	0x2C0000
TABLE PAGE 0x2D	TABLE PAGE 0x2D	0x2D0000
TABLE PAGE 0x2E	TABLE PAGE 0x2E	0x2E0000
TABLE PAGE 0x2F	TABLE PAGE 0x2F	0x2F0000
TABLE PAGE 0x30	TABLE PAGE 0x30	0x300000
TABLE PAGE 0x31	TABLE PAGE 0x31	0x310000
TABLE PAGE 0x32	TABLE PAGE 0x32	0x320000
TABLE PAGE 0x33	TABLE PAGE 0x33	0x330000
TABLE PAGE 0x34	TABLE PAGE 0x34	0x340000
TABLE PAGE 0x35	TABLE PAGE 0x35	0x350000
TABLE PAGE 0x36	TABLE PAGE 0x36	0x360000
TABLE PAGE 0x37	TABLE PAGE 0x37	0x370000
TABLE PAGE 0x38	TABLE PAGE 0x38	0x380000
TABLE PAGE 0x39	TABLE PAGE 0x39	0x390000
TABLE PAGE 0x3A	TABLE PAGE 0x3A	0x3A0000
TABLE PAGE 0x3B	TABLE PAGE 0x3B	0x3B0000
TABLE PAGE 0x3C	TABLE PAGE 0x3C	0x3C0000
TABLE PAGE 0x3D	TABLE PAGE 0x3D	0x3D0000
TABLE PAGE 0x3E	TABLE PAGE 0x3E	0x3E0000
TABLE PAGE 0x3F	TABLE PAGE 0x3F	0x3F0000
TABLE PAGE 0x40	TABLE PAGE 0x40	0x400000
TABLE PAGE 0x41	TABLE PAGE 0x41	0x410000
TABLE PAGE 0x42	TABLE PAGE 0x42	0x420000
TABLE PAGE 0x43	TABLE PAGE 0x43	0x430000
TABLE PAGE 0x44	TABLE PAGE 0x44	0x440000
TABLE PAGE 0x45	TABLE PAGE 0x45	0x450000
TABLE PAGE 0x46	TABLE PAGE 0x46	0x460000
TABLE PAGE 0x47	TABLE PAGE 0x47	0x470000
TABLE PAGE 0x48	TABLE PAGE 0x48	0x480000
TABLE PAGE 0x49	TABLE PAGE 0x49	0x490000
TABLE PAGE 0x4A	TABLE PAGE 0x4A	0x4A0000
TABLE PAGE 0x4B	TABLE PAGE 0x4B	0x4B0000
TABLE PAGE 0x4C	TABLE PAGE 0x4C	0x4C0000
TABLE PAGE 0x4D	TABLE PAGE 0x4D	0x4D0000
TABLE PAGE 0x4E	TABLE PAGE 0x4E	0x4E0000
TABLE PAGE 0x4F	TABLE PAGE 0x4F	0x4F0000
TABLE PAGE 0x50	TABLE PAGE 0x50	0x500000
TABLE PAGE 0x51	TABLE PAGE 0x51	0x510000
TABLE PAGE 0x52	TABLE PAGE 0x52	0x520000
TABLE PAGE 0x53	TABLE PAGE 0x53	0x530000
TABLE PAGE 0x54	TABLE PAGE 0x54	0x540000
TABLE PAGE 0x55	TABLE PAGE 0x55	0x550000
TABLE PAGE 0x56	TABLE PAGE 0x56	0x560000
TABLE PAGE 0x57	TABLE PAGE 0x57	0x570000
TABLE PAGE 0x58	TABLE PAGE 0x58	0x580000
TABLE PAGE 0x59	TABLE PAGE 0x59	0x590000
TABLE PAGE 0x5A	TABLE PAGE 0x5A	0x5A0000
TABLE PAGE 0x5B	TABLE PAGE 0x5B	0x5B0000
TABLE PAGE 0x5C	TABLE PAGE 0x5C	0x5C0000
TABLE PAGE 0x5D	TABLE PAGE 0x5D	0x5D0000
TABLE PAGE 0x5E	TABLE PAGE 0x5E	0x5E0000
TABLE PAGE 0x5F	TABLE PAGE 0x5F	0x5F0000
TABLE PAGE 0x60	TABLE PAGE 0x60	0x600000
TABLE PAGE 0x61	TABLE PAGE 0x61	0x610000
TABLE PAGE 0x62	TABLE PAGE 0x62	0x620000
TABLE PAGE 0x63	TABLE PAGE 0x63	0x630000
TABLE PAGE 0x64	TABLE PAGE 0x64	0x640000
TABLE PAGE 0x65	TABLE PAGE 0x65	0x650000
TABLE PAGE 0x66	TABLE PAGE 0x66	0x660000
TABLE PAGE 0x67	TABLE PAGE 0x67	0x670000
TABLE PAGE 0x68	TABLE PAGE 0x68	0x680000
TABLE PAGE 0x69	TABLE PAGE 0x69	0x690000
TABLE PAGE 0x6A	TABLE PAGE 0x6A	0x6A0000
TABLE PAGE 0x6B	TABLE PAGE 0x6B	0x6B0000
TABLE PAGE 0x6C	TABLE PAGE 0x6C	0x6C0000
TABLE PAGE 0x6D	TABLE PAGE 0x6D	0x6D0000
TABLE PAGE 0x6E	TABLE PAGE 0x6E	0x6E0000
TABLE PAGE 0x6F	TABLE PAGE 0x6F	0x6F0000
TABLE PAGE 0x70	TABLE PAGE 0x70	0x700000
TABLE PAGE 0x71	TABLE PAGE 0x71	0x710000
TABLE PAGE 0x72	TABLE PAGE 0x72	0x720000
TABLE PAGE 0x73	TABLE PAGE 0x73	0x730000
TABLE PAGE 0x74	TABLE PAGE 0x74	0x740000
TABLE PAGE 0x75	TABLE PAGE 0x75	0x750000
TABLE PAGE 0x76	TABLE PAGE 0x76	0x760000
TABLE PAGE 0x77	TABLE PAGE 0x77	0x770000
TABLE PAGE 0x78	TABLE PAGE 0x78	0x780000
TABLE PAGE 0x79	TABLE PAGE 0x79	0x790000
TABLE PAGE 0x7A	TABLE PAGE 0x7A	0x7A0000
TABLE PAGE 0x7B	TABLE PAGE 0x7B	0x7B0000
TABLE PAGE 0x7C	TABLE PAGE 0x7C	0x7C0000
TABLE PAGE 0x7D	TABLE PAGE 0x7D	0x7D0000
TABLE PAGE 0x7E	TABLE PAGE 0x7E	0x7E0000
TABLE PAGE 0x7F	TABLE PAGE 0x7F	0x7F0000
TABLE PAGE 0x80	TABLE PAGE 0x80	0x800000
TABLE PAGE 0x81	TABLE PAGE 0x81	0x810000
TABLE PAGE 0x82	TABLE PAGE 0x82	0x820000
TABLE PAGE 0x83	TABLE PAGE 0x83	0x830000
TABLE PAGE 0x84	TABLE PAGE 0x84	0x840000
TABLE PAGE 0x85	TABLE PAGE 0x85	0x850000
TABLE PAGE 0x86	TABLE PAGE 0x86	0x860000
TABLE PAGE 0x87	TABLE PAGE 0x87	0x870000
TABLE PAGE 0x88	TABLE PAGE 0x88	0x880000
TABLE PAGE 0x89	TABLE PAGE 0x89	0x890000
TABLE PAGE 0x8A	TABLE PAGE 0x8A	0x8A0000
TABLE PAGE 0x8B	TABLE PAGE 0x8B	0x8B0000
TABLE PAGE 0x8C	TABLE PAGE 0x8C	0x8C0000
TABLE PAGE 0x8D	TABLE PAGE 0x8D	0x8D0000
TABLE PAGE 0x8E	TABLE PAGE 0x8E	0x8E0000
TABLE PAGE 0x8F	TABLE PAGE 0x8F	0x8F0000
TABLE PAGE 0x90	TABLE PAGE 0x90	0x900000
TABLE PAGE 0x91	TABLE PAGE 0x91	0x910000
TABLE PAGE 0x92	TABLE PAGE 0x92	0x920000
TABLE PAGE 0x93	TABLE PAGE 0x93	0x930000
TABLE PAGE 0x94	TABLE PAGE 0x94	0x940000
TABLE PAGE 0x95	TABLE PAGE 0x95	0x950000
TABLE PAGE 0x96	TABLE PAGE 0x96	0x960000
TABLE PAGE 0x97	TABLE PAGE 0x97	0x970000
TABLE PAGE 0x98	TABLE PAGE 0x98	0x980000
TABLE PAGE 0x99	TABLE PAGE 0x99	0x990000
TABLE PAGE 0x9A	TABLE PAGE 0x9A	0x9A0000
TABLE PAGE 0x9B	TABLE PAGE 0x9B	0x9B0000
TABLE PAGE 0x9C	TABLE PAGE 0x9C	0x9C0000
TABLE PAGE 0x9D	TABLE PAGE 0x9D	0x9D0000
TABLE PAGE 0x9E	TABLE PAGE 0x9E	0x9E0000
TABLE PAGE 0x9F	TABLE PAGE 0x9F	0x9F0000
TABLE PAGE 0xA0	TABLE PAGE 0xA0	0xA00000
TABLE PAGE 0xA1	TABLE PAGE 0xA1	0xA10000
TABLE PAGE 0xA2	TABLE PAGE 0xA2	0xA20000
TABLE PAGE 0xA3	TABLE PAGE 0xA3	0xA30000
TABLE PAGE 0xA4	TABLE PAGE 0xA4	0xA40000
TABLE PAGE 0xA5	TABLE PAGE 0xA5	0xA50000
TABLE PAGE 0xA6	TABLE PAGE 0xA6	0xA60000
TABLE PAGE 0xA7	TABLE PAGE 0xA7	0xA70000
TABLE PAGE 0xA8	TABLE PAGE 0xA8	0xA80000
TABLE PAGE 0xA9	TABLE PAGE 0xA9	0xA90000
TABLE PAGE 0xAA	TABLE PAGE 0xAA	0xAA0000
TABLE PAGE 0xAB	TABLE PAGE 0xAB	0xAB0000
TABLE PAGE 0xAC	TABLE PAGE 0xAC	0xAC0000
TABLE PAGE 0xAD	TABLE PAGE 0xAD	0xAD0000
TABLE PAGE 0xAE	TABLE PAGE 0xAE	0xAE0000
TABLE PAGE 0xAF	TABLE PAGE 0xAF	0xAF0000
TABLE PAGE 0xB0	TABLE PAGE 0xB0	0xB00000
TABLE PAGE 0xB1	TABLE PAGE 0xB1	0xB10000
TABLE PAGE 0xB2	TABLE PAGE 0xB2	0xB20000
TABLE PAGE 0xB3	TABLE PAGE 0xB3	0xB30000
TABLE PAGE 0xB4	TABLE PAGE 0xB4	0xB40000
TABLE PAGE 0xB5	TABLE PAGE 0xB5	0xB50000
TABLE PAGE 0xB6	TABLE PAGE 0xB6	0xB60000
TABLE PAGE 0xB7	TABLE PAGE 0xB7	0xB70000
TABLE PAGE 0xB8	TABLE PAGE 0xB8	0xB80000
TABLE PAGE 0xB9	TABLE PAGE 0xB9	0xB90000
TABLE PAGE 0xBA	TABLE PAGE 0xBA	0xBA0000
TABLE PAGE 0xBB	TABLE PAGE 0xBB	0xBB0000
TABLE PAGE 0xBC	TABLE PAGE 0xBC	0xBC0000
TABLE PAGE 0xBD	TABLE PAGE 0xBD	0xBD0000
TABLE PAGE 0xBE	TABLE PAGE 0xBE	0xBE0000
TABLE PAGE 0xBF	TABLE PAGE 0xBF	0xBF0000
TABLE PAGE 0xC0	TABLE PAGE 0xC0	0xC00000
TABLE PAGE 0xC1	TABLE PAGE 0xC1	0xC10000
TABLE PAGE 0xC2	TABLE PAGE 0xC2	0xC20000
TABLE PAGE 0xC3	TABLE PAGE 0xC3	0xC30000
TABLE PAGE 0xC4	TABLE PAGE 0xC4	0xC40000
TABLE PAGE 0xC5	TABLE PAGE 0xC5	0xC50000
TABLE PAGE 0xC6	TABLE PAGE 0xC6	0xC60000
TABLE PAGE 0xC7	TABLE PAGE 0xC7	0xC70000
TABLE PAGE 0xC8	TABLE PAGE 0xC8	0xC80000
TABLE PAGE 0xC9	TABLE PAGE 0xC9	0xC90000
TABLE PAGE 0xCA	TABLE PAGE 0xCA	0xCA0000
TABLE PAGE 0xCB	TABLE PAGE 0xCB	0xCB0000
TABLE PAGE 0xCC	TABLE PAGE 0xCC	0xCC0000
TABLE PAGE 0xCD	TABLE PAGE 0xCD	0xCD0000
TABLE PAGE 0xCE	TABLE PAGE 0xCE	0xCE0000
TABLE PAGE 0xCF	TABLE PAGE 0xCF	0xCF0000
TABLE PAGE 0xD0	TABLE PAGE 0xD0	0xD00000
TABLE PAGE 0xD1	TABLE PAGE 0xD1	0xD10000
TABLE PAGE 0xD2	TABLE PAGE 0xD2	0xD20000
TABLE PAGE 0xD3	TABLE PAGE 0xD3	0xD30000
TABLE PAGE 0xD4	TABLE PAGE 0xD4	0xD40000
TABLE PAGE 0xD5	TABLE PAGE 0xD5	0xD50000
TABLE PAGE 0xD6	TABLE PAGE 0xD6	0xD60000
TABLE PAGE 0xD7	TABLE PAGE 0xD7	0xD70000
TABLE PAGE 0xD8	TABLE PAGE 0xD8	0xD80000
TABLE PAGE 0xD9	TABLE PAGE 0xD9	0xD90000
TABLE PAGE 0xDA	TABLE PAGE 0xDA	0xDA0000
TABLE PAGE 0xDB	TABLE PAGE 0xDB	0xDB0000
TABLE PAGE 0xDC	TABLE PAGE 0xDC	0xDC0000
TABLE PAGE 0xDD	TABLE PAGE 0xDD	0xDD0000
TABLE PAGE 0xDE	TABLE PAGE 0xDE	0xDE0000
TABLE PAGE 0xDF	TABLE PAGE 0xDF	0xDF0000
TABLE PAGE 0xE0	TABLE PAGE 0xE0	0xE00000
TABLE PAGE 0xE1	TABLE PAGE 0xE1	0xE10000
TABLE PAGE 0xE2	TABLE PAGE 0xE2	0xE20000
TABLE PAGE 0xE3	TABLE PAGE 0xE3	0xE30000
TABLE PAGE 0xE4	TABLE PAGE 0xE4	0xE40000
TABLE PAGE 0xE5	TABLE PAGE 0xE5	0xE50000
TABLE PAGE 0xE6	TABLE PAGE 0xE6	0xE60000
TABLE PAGE 0xE7	TABLE PAGE 0xE7	0xE70000
TABLE PAGE 0xE8	TABLE PAGE 0xE8	0xE80000
TABLE PAGE 0xE9	TABLE PAGE 0xE9	0xE90000
TABLE PAGE 0xEA	TABLE PAGE 0xEA	0xEA0000
TABLE PAGE 0xEB	TABLE PAGE 0xEB	0xEB0000
TABLE PAGE 0xEC	TABLE PAGE 0xEC	0xEC0000
TABLE PAGE 0xED	TABLE PAGE 0xED	0xED0000
TABLE PAGE 0xEE	TABLE PAGE 0xEE	0xEE0000
TABLE PAGE 0xEF	TABLE PAGE 0xEF	0xEF0000
TABLE PAGE 0xF0	TABLE PAGE 0xF0	0xF00000
TABLE PAGE 0xF1	TABLE PAGE 0xF1	0xF10000
TABLE PAGE 0xF2	TABLE PAGE 0xF2	0xF20000
TABLE PAGE 0xF3	TABLE PAGE 0xF3	0xF30000
TABLE PAGE 0xF4	TABLE PAGE 0xF4	0xF40000
TABLE PAGE 0xF5	TABLE PAGE 0xF5	0xF50000
TABLE PAGE 0xF6	TABLE PAGE 0xF6	0xF60000
TABLE PAGE 0xF7	TABLE PAGE 0xF7	0xF70000
TABLE PAGE 0xF8	TABLE PAGE 0xF8	0xF80000
TABLE PAGE 0xF9	TABLE PAGE 0xF9	0xF90000
TABLE PAGE 0xFA	TABLE PAGE 0xFA	0xFA0000
TABLE PAGE 0xFB	TABLE PAGE 0xFB	0xFB0000
TABLE PAGE 0xFC	TABLE PAGE 0xFC	0xFC0000
TABLE PAGE 0xFD	TABLE PAGE 0xFD	0xFD0000
TABLE PAGE 0xFE	TABLE PAGE 0xFE	0xFE0000
TABLE PAGE 0xFF	TABLE PAGE 0xFF	0xFF0000
TABLE PAGE 0x100	TABLE PAGE 0x100	0x100000
TABLE PAGE 0x101	TABLE PAGE 0x101	0x101000
TABLE PAGE 0x102	TABLE PAGE 0x102	0x102000
TABLE PAGE 0x103	TABLE PAGE 0x103	0x103000
TABLE PAGE 0x104	TABLE PAGE 0x104	0x104000
TABLE PAGE 0x105	TABLE PAGE 0x105	0x105000
TABLE PAGE 0x106	TABLE PAGE 0x106	0x106000
TABLE PAGE 0x107	TABLE PAGE 0x107	0x107000
TABLE PAGE 0x108	TABLE PAGE 0x108	0x108000
TABLE PAGE 0x109	TABLE PAGE 0x109	0x109000
TABLE PAGE 0x10A	TABLE PAGE 0x10A	0x10A000
TABLE PAGE 0x10B	TABLE PAGE 0x10B	0x10B000
TABLE PAGE 0x10C	TABLE PAGE 0x10C	0x10C000
TABLE PAGE 0x10D	TABLE PAGE 0x10D	0x10D000
TABLE PAGE 0x10E	TABLE PAGE 0x10E	0x10E000
TABLE PAGE 0x10F	TABLE PAGE 0x10F	0x10F000
TABLE PAGE 0x110	TABLE PAGE 0x110	0x110000
TABLE PAGE 0x111	TABLE PAGE 0x111	0x111000
TABLE PAGE 0x112	TABLE PAGE 0x112	0x112000
TABLE PAGE 0x113	TABLE PAGE 0x113	0x113000
TABLE PAGE 0x114	TABLE PAGE 0x114	0x114000
TABLE PAGE 0x115	TABLE PAGE 0x115	0x115000
TABLE PAGE 0x116	TABLE PAGE 0x11	0x116000

dsPIC33/PIC24 Program Memory

4.3 Program Memory Low Word Access

The `TBLRD L` instruction is 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 4-4](#) demonstrates the program memory data regions accessed by the `TBLRD L` instruction.

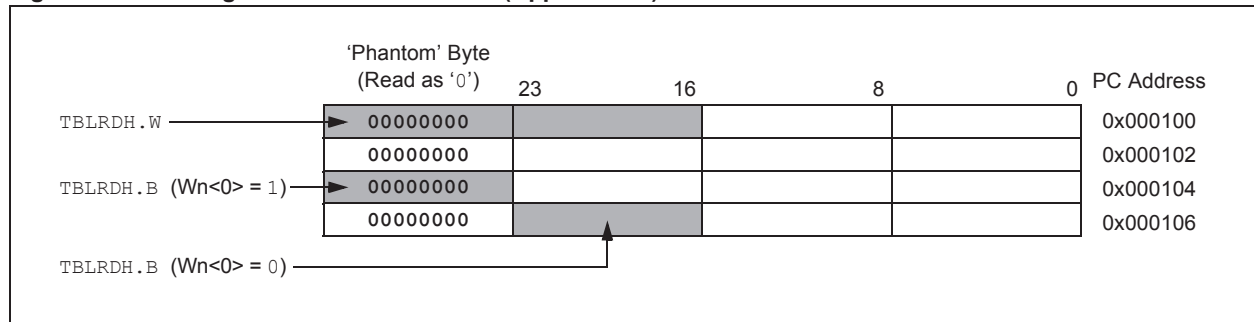
Figure 4-4: Program Data Table Access (Lower 16 Bits)



4.4 Program Memory High Word Access

The `TBLRD H` instruction is used to access the upper eight bits of the program memory data. [Figure 4-5](#) 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 4-5: Program Data Table Access (Upper 8 Bits)



dsPIC33/PIC24 Family Reference Manual

4.5 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 built-in functions, `__builtin_tblpage` and `__builtin_tbloffset`, that are provided by the MPLAB® XC16 C compiler.

[Example 4-2](#) uses the `space(prog)` attribute to allocate the buffer in program memory. The MPLAB XC16 Compiler also has built-in functions, such as `__builtin_tblpage` and `__builtin_tbloffset`, that can be used to access the buffer. For more information, refer to the “MPLAB XC16 C Compiler User’s Guide” (DS50002071).

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

```
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
    TBLRDL  [W1], W0
    TBLRDH  [W1], W1
    RETURN
```

Example 4-2: Using MPLAB® XC16 C Compiler to Access Program Memory

```
int prog_data[10] __attribute__((space(prog))) = {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);
}
```

dsPIC33/PIC24 Program Memory

5.0 PROGRAM SPACE VISIBILITY FROM DATA SPACE

The upper 32 Kbytes of the dsPIC33/PIC24 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).

5.1 PSV Configuration

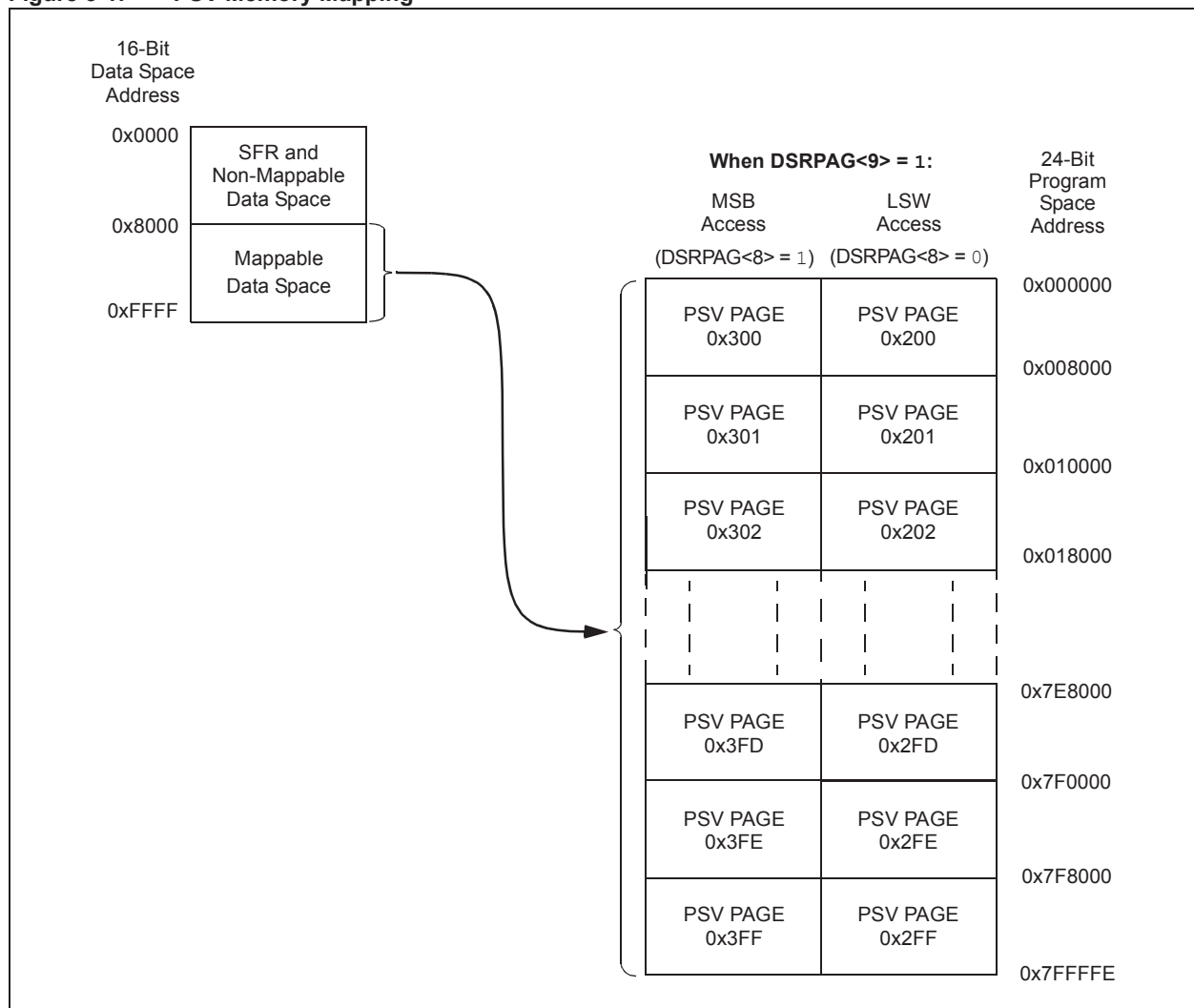
The dsPIC33/PIC24 core extends the available Data Space through a paging scheme to make it appear linear for pre-modified and post-modified Effective Addresses.

The upper half of the base Data Space address (0x8000 to 0xFFFF) is used with the 10-bit Data Space Read Page (DSRPAG) register to form a PSV address and can address eight 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 5-1.

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 (DSWPAG) register is dedicated exclusively to Data Space (DS), including Extended Data Space (EDS).

For more information on the paged memory scheme, refer to the “dsPIC33/PIC24 Family Reference Manual”, “Data Memory” (DS70595).

Figure 5-1: PSV Memory Mapping



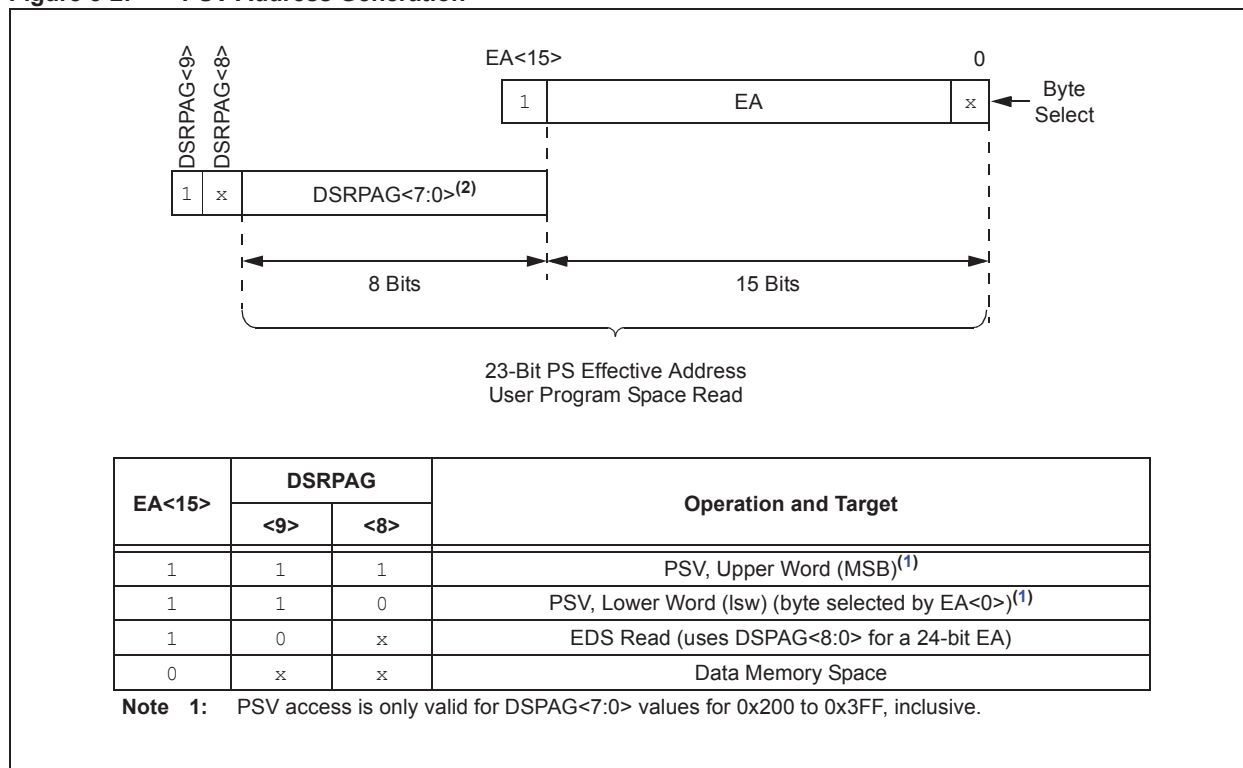
dsPIC33/PIC24 Family Reference Manual

5.1.1 PSV ADDRESS GENERATION

Allocating different Page registers for read and write access allows the architecture to support data movement from different PSV pages 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 5-2 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 5-2: PSV Address Generation



The remaining address bits are provided by the eight LSbs of the Data Space Read Page register (DSRPAG<7:0>). The DSRPAG<7:0> bits are concatenated with the 15 LSbs 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> = 10, the lsw 16 bits of the 24-bit PS word can be accessed using PSV. When DSRPAG<9:8> = 11, the MSB of the 24-bit PS word can be accessed using PSV. 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.

dsPIC33/PIC24 Program Memory

5.2 PSV Timing

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

5.2.1 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 TBLRDL/H instruction requires five instruction cycles for each iteration.

5.2.2 PSV AND INSTRUCTION STALLS

For more information about instruction Stalls using PSV, refer to the appropriate “dsPIC33/PIC24 Family Reference Manual” chapter, “CPU” or “dsPIC33 Enhanced CPU” (DS70359 or DS70005158, respectively) specified in the device data sheet.

5.3 PSV Code Examples

[Example 5-1](#) 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 start-up code.

Example 5-1: Compiler-Managed PSV Access

```
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 Kbytes of RAM.

dsPIC33/PIC24 Family Reference Manual

[Example 5-2](#) illustrates buffer placement and access in the user-managed PSV section. The `psv` space is the user-managed PSV section. [Example 5-3](#) illustrates the placement of constant data in program memory and accesses this data through the PSV data window using an assembly program.

Example 5-2: User-Managed PSV Access

```
const int m[5] = {1, 2, 3, 4, 5};
const int m1[5] __attribute__((space(psv))) = {2, 4, 6, 8, 10};
const int m2[5] __attribute__((space(psv))) = {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;
    temp = DSRPAG; // Save original PSV page value
    DSRPAG = __builtin_psvpage (m1);
    sum1 = vectordot ((int *) m1, x);
    DSRPAG = __builtin_psvpage (m2);
    sum2 = vectordot ((int *) m2, x);
    DSRPAG = temp; // Restore original PSV page value
    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 5-3: 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
MOVPAG #psvpage(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
```

6.0 PROGRAM MEMORY WRITES

There are two methods by which the user application can program Flash memory:

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

For more information on RTSP, refer to the “*dsPIC33/PIC24 Family Reference Manual*”, “**Flash Programming**” (DS70000609). For more information on ICSP, refer to the specific “*Flash Programming Specification*” for your device, which can be obtained from the Microchip website (www.microchip.com).

7.0 ERROR CORRECTING CODE (ECC)

In order to improve program memory performance and durability, select dsPIC33 and PIC24 devices include Error Correcting Code (ECC) functionality as an integral part of the Flash memory controller. ECC can determine the presence of single-bit errors in program data, including which bit is in error, and correct the data without user intervention. When implemented, ECC is automatic and cannot be disabled.

When data is written to program memory, ECC generates a 7-bit Hamming code parity value for every two (24-bit) instruction words. The data is stored in blocks of 48 data bits and seven parity bits; parity data is not memory-mapped and is inaccessible. When the data is read back, the ECC calculates parity on it and compares it to the previously stored parity value. If a parity mismatch occurs, there are two possible outcomes:

- Single-bit errors are automatically identified and corrected on read back. An optional device-level interrupt (ECCSBEIF) is also generated.
- Double-bit errors will generate a generic hard trap. If special exception handling for the trap is not implemented, a device Reset will also occur.

To use the single-bit error interrupt, set the ECC Single-Bit Error Interrupt Enable (ECCSBEIE) bit and configure the ECCSBEIPx bits to set the appropriate interrupt priority.

On some devices, Fault injection and additional error information are made available. Refer to the device-specific data sheet for availability. In addition to the ECCSBEIF flag, the ECCSTATL register contains the parity information for single-bit errors. The SECOUT<7:0> bits field contains the expected calculated SEC parity and the SECIN<7:0> bits contain the actual value from a Flash read operation. The SECSYNDx bits (ECCSTATH<7:0>) indicate the bit position of the single-bit error within the 48-bit pair of instruction words. When no error is present, SECINx equals SECOUTx and SECSYNDx is zero.

Double-bit errors result in a generic hard trap. The ECCDBE bit (INTCON4<1>) will be set to identify the source of the hard trap. If no Interrupt Service Routine is implemented for the hard trap, a device Reset will also occur. The ECCSTATH register contains double-bit error status information. The DEDOUT bit is the expected calculated DED parity and DEDIN is the actual value from a Flash read operation. When no error is present, DEDIN equals DEDOUT.

7.1 ECC Fault Injection

To test Fault handling, an EEC error can be generated. Both single and double-bit errors can be generated in both the read and write data paths. Read path Fault injection first reads the Flash data and then modifies it prior to entering the ECC logic. Write path Fault injection modifies the actual data prior to it being written into the target Flash and will cause an EEC error on a subsequent Flash read. The following procedure is used to inject a Fault:

1. Load the Flash target address into the ECCADDR register.
2. Select 1st Fault bit determined by FLT1PTRx (ECCCONH<7:0>). The target bit is inverted to create the Fault.
3. If a double Fault is desired, select the 2nd Fault bit determined by FLT2PTRx (ECCCONH<15:8>); otherwise, set to all '1's.
4. Write the NVMKEY unlock sequence (see specific device data sheet for sequence).
5. Enable the ECC Fault injection logic by setting the FLTINJ bit (ECCCONL<0>).
6. Perform a read or write to the Flash target address.

8.0 PROGRAM MEMORY LOW-POWER MODE

The voltage regulator for the program Flash memory can be placed in Standby mode when the device is in Sleep mode, resulting in a reduction in device Power-Down Current (IPD).

When the VREGSF bit (RCON<11>) is equal to '0', the Flash memory voltage regulator goes into Standby mode during Sleep. When the VREGSF bit is equal to '1', the Flash memory voltage regulator is active during Sleep mode; however, this mode increases the device wake-up delay.

9.0 REGISTER MAP

A summary of the registers associated with the dsPIC33/PIC24 Program Memory is provided in [Table 9-1](#).

Table 9-1: CPU Core 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
PCL	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
PCH	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
DSRPAG	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0001
DSWPAG	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0001
TBLPAG	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
ECCONL	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
ECCONH	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
ECCADDRL	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
ECCADDRH	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
ECCSTATL	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
ECCSTATH	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000

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

dsPIC33/PIC24 Family Reference Manual

10.0 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 dsPIC33/PIC24 Product Families, but the concepts are pertinent and could be used with modification and possible limitations. The current application notes related to the dsPIC33/PIC24 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 dsPIC33/PIC24 families of devices, visit the Microchip website (www.microchip.com).

11.0 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 5-3](#))
- Updated the Program Memory Map (see [Figure 1-2](#))
- Updated the first paragraph and the shaded note in [Section 4.1 “Table Instruction Summary”](#)
- Added a shaded note after [Figure 4-1](#) with information on writing to the TBLPAG register
- Updated [Section 4.2 “Table Address Generation”](#)
- Updated the second sentence in [Section 4.3 “Program Memory Low Word Access”](#)
- Added the new figure [Table Page Memory Map](#) (see [Figure 4-3](#)) in [Section 4.4 “Program Memory High Word Access”](#)
- Added a shaded note and updated the last paragraph in [Section 5.1 “PSV Configuration”](#)
- Updated the Paged Data Memory Space (see [Figure 5-1](#))
- Updated the PSV Address Generation (see [Figure 5-2](#))
- Changed the number of required instruction cycles from two to five throughout [Section 5.2 “PSV Timing”](#)
- Added a shaded note after [Example 5-1](#) with information on using the `auto_psv` option
- Added a reference to the “*dsPIC33/PIC24 Flash Programming Specification*” (DS70619) to [Section 6.0 “Program Memory Writes”](#)

Revision C (December 2011)

This revision includes the following updates:

- Updated [Section 1.0 “Program Memory Address Map”](#)
- Updated the existing Program Memory Map for devices with auxiliary memory (see [Figure 1-2](#))
- Added a new Program Memory Map for devices without auxiliary memory (see [Figure 1-1](#))
- Updated Using Table Instructions to Access Program Memory (see [Example 4-1](#))
- Updated Using MPLAB® C Compiler to Access Program Memory (see [Example 4-2](#))
- Removed [4.4.5 “Data Storage in Program Memory”](#)
- Removed [4.5.2 “PSV Mapping with X and Y Data Space”](#)
- Updated Compiler-Managed PSV Access (see [Example 5-1](#))
- Updated User-Managed PSV Access (see [Example 5-2](#))
- Updated [Section 6.0 “Program Memory Writes”](#)
- Updated [Section 8.0 “Program Memory Low-Power Mode”](#)
- Updated the Register Map table (see [Table 9-1](#))
- Minor updates to text and formatting were incorporated throughout the document

dsPIC33/PIC24 Family Reference Manual

Revision D (November 2014)

Updates the document format and removes the previously assigned master section number as part of the realignment of dsPIC33 technical documentation. The document reference number format is also updated.

Updates the document title to “dsPIC33/PIC24 Program Memory” for clarity.

Adds **Section 7.0 “Error Correcting Code (ECC)”**. Subsequent sections are renumbered accordingly.

Updates **Section 1.0 “Program Memory Address Map”** to mention OTP locations, in addition to USERID locations

Reorganizes **Section 4.0 “Reading Program Memory Using Table Instructions”** to include the “Table Memory Map” (formerly Figure 4-5) with the text of **Section 4.2 “Table Address Generation”**, as Figure 4-3.

Updates **Section 5.0 “Program Space Visibility from Data Space”** by revising Figures 5-1 and 5-2 for clarity, and removing Figures 5-3 and 5-4 as redundant. Adds the subhead, **Section 5.1.1 “PSV Address Generation”** to delineate topics without changing the previously existing text.

Other minor changes to text and typographic changes throughout the document.

Revision E (October 2018)

Updates **Section 7.0 “Error Correcting Code (ECC)”**.

Updates **Table 9-1**.

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 unless otherwise stated.

Microchip received ISO/TS-16949:2009 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, KEELQ® 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 ==

Trademarks

The Microchip name and logo, the Microchip logo, AnyRate, AVR, AVR logo, AVR Freaks, BitCloud, chipKIT, chipKIT logo, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, Heldo, JukeBlox, KeeLoq, Klear, LANCheck, LINK MD, maXStylus, maXTouch, MediaLB, megaAVR, MOST, MOST logo, MPLAB, OptoLyzer, PIC, picoPower, PICSTART, PIC32 logo, Prochip Designer, QTouch, SAM-BA, SpyNIC, SST, SST Logo, SuperFlash, tinyAVR, UNI/O, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

ClockWorks, The Embedded Control Solutions Company, EtherSynch, Hyper Speed Control, HyperLight Load, IntelliMOS, mTouch, Precision Edge, and Quiet-Wire are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, BodyCom, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, EtherGREEN, In-Circuit Serial Programming, ICSP, INICnet, Inter-Chip Connectivity, JitterBlocker, KlearNet, KlearNet logo, memBrain, Mindi, MiWi, motorBench, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICKit, PICtail, PowerSmart, PureSilicon, QMatrix, REAL ICE, Ripple Blocker, SAM-ICE, Serial Quad I/O, SMART-I.S., SQI, SuperSwitcher, SuperSwitcher II, Total Endurance, TSHARC, USBCheck, VariSense, ViewSpan, WiperLock, Wireless DNA, 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.

Silicon Storage Technology is a registered trademark of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

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

© 2018, Microchip Technology Incorporated, All Rights Reserved.

ISBN: 978-1-5224-3699-7

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

Austin, TX
Tel: 512-257-3370

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

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

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

Detroit
Novi, MI
Tel: 248-848-4000

Houston, TX
Tel: 281-894-5983

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453
Tel: 317-536-2380

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608
Tel: 951-273-7800

Raleigh, NC
Tel: 919-844-7510

New York, NY
Tel: 631-435-6000

San Jose, CA
Tel: 408-735-9110
Tel: 408-436-4270

Canada - Toronto
Tel: 905-695-1980
Fax: 905-695-2078

ASIA/PACIFIC

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

China - Beijing
Tel: 86-10-8569-7000

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

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

China - Dongguan
Tel: 86-769-8702-9880

China - Guangzhou
Tel: 86-20-8755-8029

China - Hangzhou
Tel: 86-571-8792-8115

China - Hong Kong SAR
Tel: 852-2943-5100

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

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

China - Shanghai
Tel: 86-21-3326-8000

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

China - Shenzhen
Tel: 86-755-8864-2200

China - Suzhou
Tel: 86-186-6233-1526

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

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

China - Xiamen
Tel: 86-592-2388138

China - Zhuhai
Tel: 86-756-3210040

ASIA/PACIFIC

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

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

India - Pune
Tel: 91-20-4121-0141

Japan - Osaka
Tel: 81-6-6152-7160

Japan - Tokyo
Tel: 81-3-6880-3770

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

Korea - Seoul
Tel: 82-2-554-7200

Malaysia - Kuala Lumpur
Tel: 60-3-7651-7906

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

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

Singapore
Tel: 65-6334-8870

Taiwan - Hsin Chu
Tel: 886-3-577-8366

Taiwan - Kaohsiung
Tel: 886-7-213-7830

Taiwan - Taipei
Tel: 886-2-2508-8600

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

Vietnam - Ho Chi Minh
Tel: 84-28-5448-2100

EUROPE

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

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

Finland - Espoo
Tel: 358-9-4520-820

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

Germany - Garching
Tel: 49-8931-9700

Germany - Haan
Tel: 49-2129-3766400

Germany - Heilbronn
Tel: 49-7131-67-3636

Germany - Karlsruhe
Tel: 49-721-625370

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

Germany - Rosenheim
Tel: 49-8031-354-560

Israel - Ra'anana
Tel: 972-9-744-7705

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

Italy - Padova
Tel: 39-049-7625286

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

Norway - Trondheim
Tel: 47-7288-4388

Poland - Warsaw
Tel: 48-22-3325737

Romania - Bucharest
Tel: 40-21-407-87-50

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

Sweden - Gothenberg
Tel: 46-31-704-60-40

Sweden - Stockholm
Tel: 46-8-5090-4654

UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820