存取內部 EEPROM

讀取
清除
寫入

© 2004 Microchip Technology Incorporated. All Rights Reserved.

dsPIC30F Data EEPROM

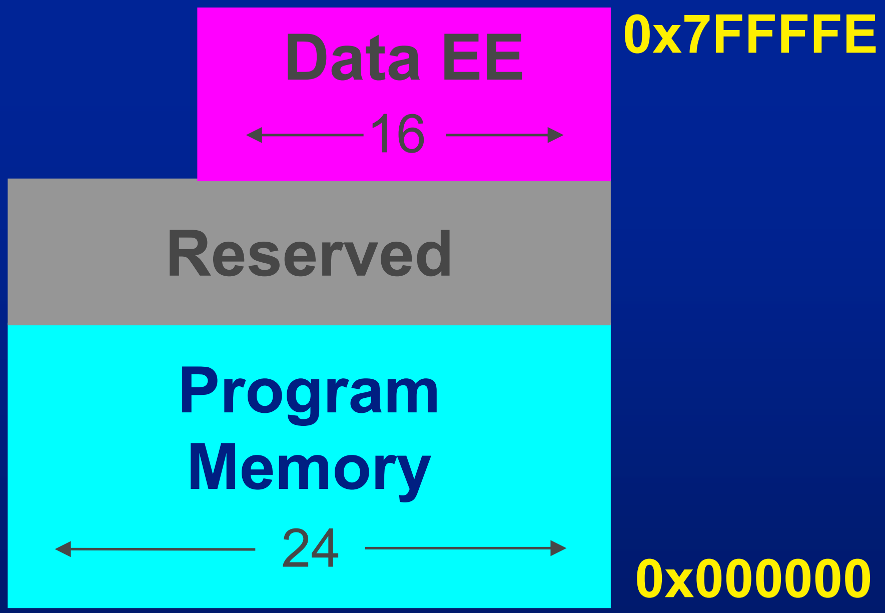
- 大部分的 dsPIC® 擁有內建的 Data EEPROM
  - 記憶容量從 1 KByte 到 4 Kbyte (因元件而異)
- 兩種燒錄方式
  - 利用 ICSP™ 方式直接燒錄 HEX 的檔案資料
  - 程式執行自我燒錄功能 (Self Programming)
- 清除 / 寫入的時間規格
  - 單一個 word 需 2mS
  - 1 列 (16 words) 需 2mS
  - 清除整個 EEPROM 需 2mS

© 2004 Microchip Technology Incorporated. All Rights Reserved.

Data EEPROM in Program Memory

- Data EEPROM 的位址是對映到程式記憶體的空間 (0x7FF000 ~ 0x7FFFFE)
- 資料寬度爲16位元 – 允許以 BYTE 或 WORD 的方式存取

這個起始位址會因不同的元件而異



Data EE

16

Reserved

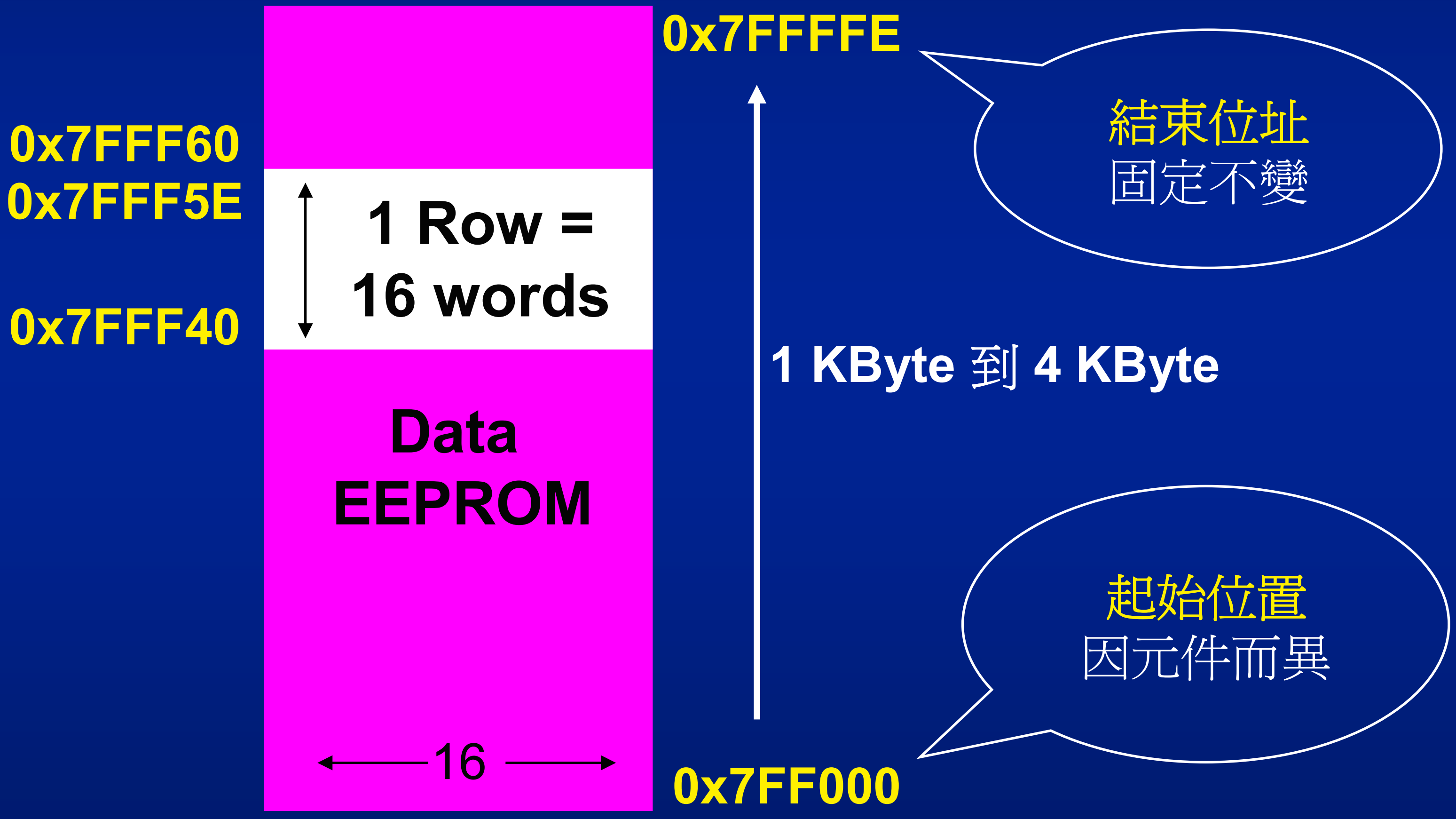
Program Memory

24

0x7FFFFE

0x000000

Data EEPROM 配置



結束位址
固定不變

起始位置
因元件而異

Flash PM vs. EEPROM
兩者的差異性

- 寫入 / 清除 Flash Memory 時，CPU 會暫時停止工作直到燒錄完成
  - 區塊的清除爲 32 個指令空間，寫入爲 32 個指令
- 寫入 / 清除 EEPROM 時，CPU 能繼續工作
  - 可單獨清除 1 word，寫入爲 1 word
  - 或清除區塊爲 16 words，寫入爲 16 words
- EEPROM 可有百萬次的寫入/清除次數
- Flash PM 有十萬次的寫入/清除次數

© 2004 Microchip Technology Incorporated. All Rights Reserved.

利用 C30 直接建立 EEPROM 的資料

- C30 可以直接在程式建立 EEPROM 的資料
- C30 提供巨集宣告 **_EEDATA(N)**方便使用
  - 巨集宣告在 P30Fxxxx.H

```
#define _EEDATA(N)  __attribute__((section(".eedata,r"),aligned(N)))
```

  - (N) 指定記憶位址的邊界值(alignment)以Byte爲單位
- 所建立的EEPROM資料編譯後會存在於Hex檔
- 可以用MPLAB® SIM 檢查 EEPROM Window

© 2004 Microchip Technology Incorporated. All Rights Reserved.

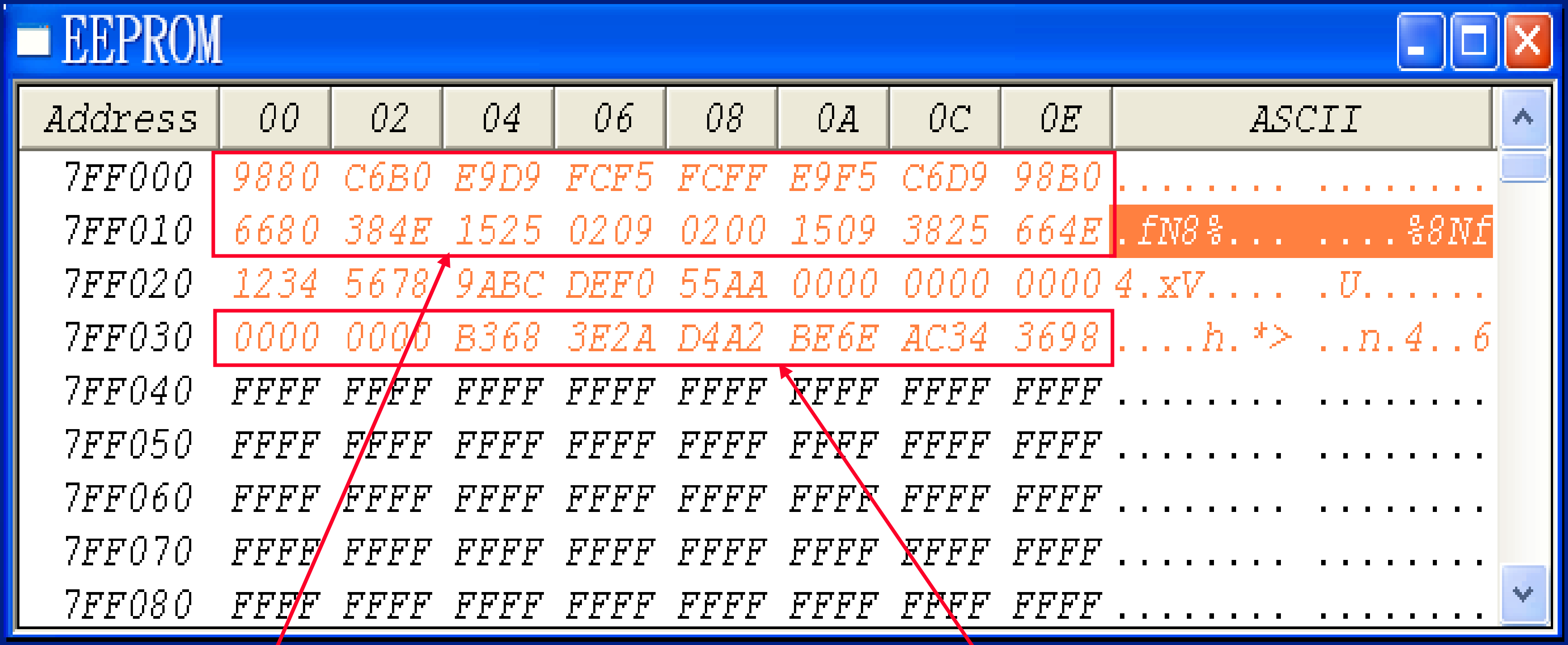
建立 EEPROM 的資料

範例程式:

```
const char _EEDATA(1) sine_table[ ] =
   {128,152,176,198,217,233,245,252,255,252
   ,245,233,217,198,176,152,128,102,78,56,37,21
   ,9,2,0,2,9,21,37,56,78,102};
const int _EEDATA(2) ramp[ ] =
         {0x1234, 0x5678, 0x9ABC,0xDEF0,0x55AA};
const double _EEDATA(16) k_factors[4] =
          {0.0, 0.1667, -0.233233, .00000455};
```

© 2004 Microchip Technology Incorporated. All Rights Reserved.

EEPROM 的顯示



```
Address   00   02   04   06   08   0A   0C   0E         ASCII
 7FF000  9880 C6B0 E9D9 FCF5 FCFF E9F5 C6D9 98B0 ........ ........
 7FF010  6680 384E 1525 0209 0200 1509 3825 664E .fN8%... ....%8Nf
 7FF020  1234 5678 9ABC DEF0 55AA 0000 0000 0000 4.xV.... .U......
 7FF030  0000 0000 B368 3E2A D4A2 BE6E AC34 3698 ....h.*> ..n.4..6
 7FF040  FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF ........ ........
 7FF050  FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF ........ ........
 7FF060  FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF ........ ........
 7FF070  FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF ........ ........
 7FF080  FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF ........ ........
```

sine_table[ ]

k_factors[4]

© 2004 Microchip Technology Incorporated. All Rights Reserved.

讀取 EEPROM 的方式

- 使用 Table Read 指令讀取
  - TBLRDL 指令：讀取程式記憶體的16位元資料及 EEPROM 的資料
  - TBLRDH 指令：讀取程式記憶體的最高的16位元資料，無法使用於讀取 EEPROM
- 使用 PSV 方式讀取
  - 快而有效率的讀取EEPROM的方式，操作起來就好像直接讀取 RAM
  - 需做一些特別的設定以啓動此功能

© 2004 Microchip Technology Incorporated. All Rights Reserved.

Table 指令的操作對象

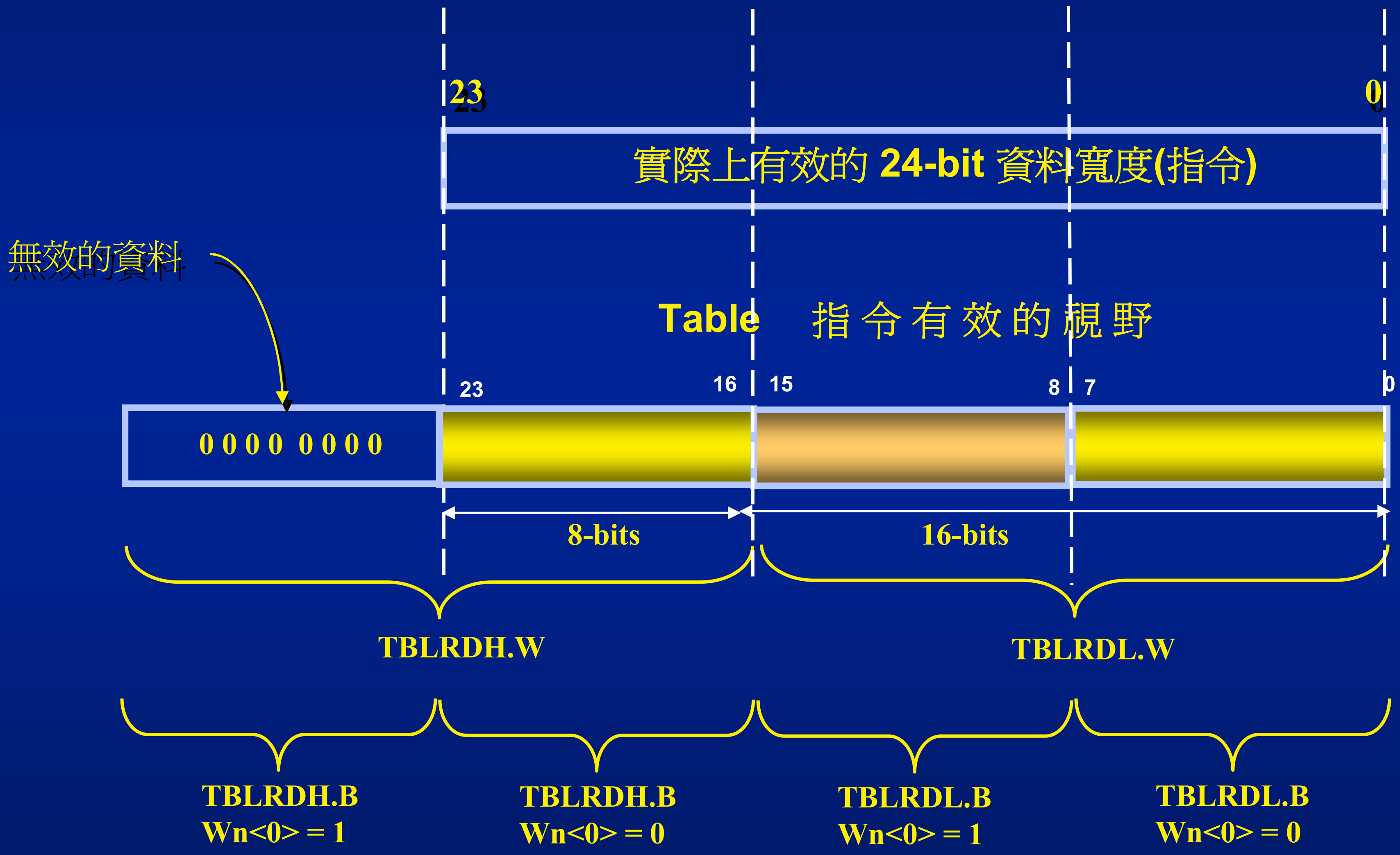
- Data EEPROM
- Flash Program Memory
- Configuration Word
  - 只能用 Table 指令存取

Table 指令操作

- TBLRDL，TBLWTL 指令
  - 讀取程式記憶體或 EEPROM 的 bit<15:0>
  - 可操作於 Byte 及 Word 兩種模式
- TBLRDH，TBLWTH
  - 只能讀取程式記憶體的 bit<31:16>
  - 可操作於 Byte 及 Word 兩種模式
  - 在 Word 模式下，只有 LSB 有效，MSB 永遠爲零

© 2004 Microchip Technology Incorporated. All Rights Reserved.

利用 Table 指令讀取程式記憶空間的資料



23

0

實際上有效的 24-bit 資料寬度(指令)

無效的資料

Table 指令有效的視野

23

16

15

8

7

0

0000 0000

8-bits

16-bits

TBLRDH.W

TBLRDL.W

TBLRDH.B
Wn<0> = 1

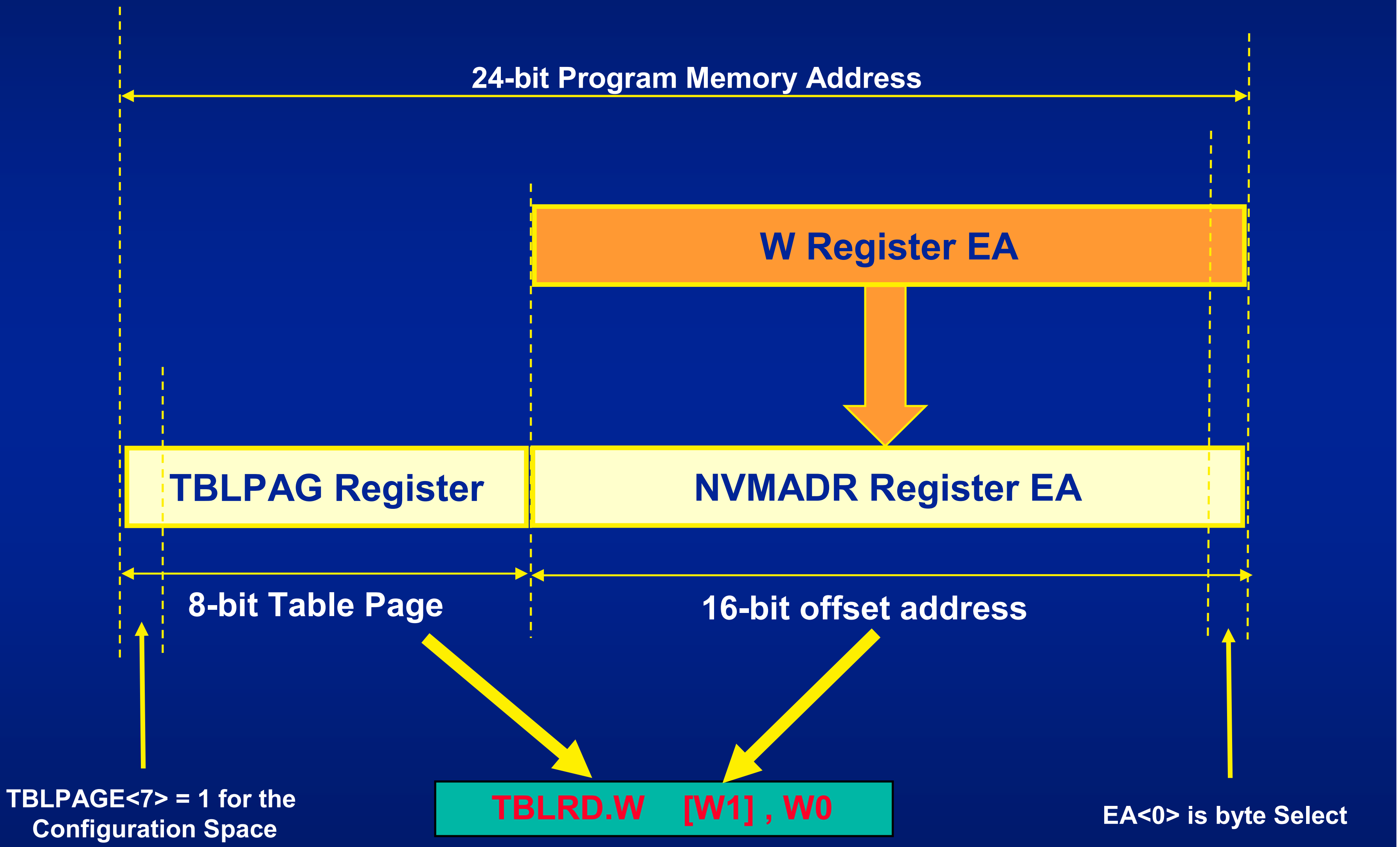
TBLRDH.B
Wn<0> = 0

TBLRDL.B
Wn<0> = 1

TBLRDL.B
Wn<0> = 0

© 2004 Microchip Technology Incorporated. All Rights Reserved.

NVM Address and Page



24-bit Program Memory Address

W Register EA

TBLPAG Register

NVMADR Register EA

8-bit Table Page

16-bit offset address

TBLPAGE<7> = 1 for the
Configuration Space

`TBLRD.W [W1] , W0`

EA<0> is byte Select

© 2004 Microchip Technology Incorporated. All Rights Reserved.

Set Table Page and Address using ASM30

```
;------------------------------------------------------------------------------
;Tone table is placed as a loopup table in EEPROM at 0x7FFxxx

         .section .eedata,"r"
ToneTable:
         .hword   0x1370,0x1398,0x13B0,0x13C6,0x13D9,0x13E9,0x13F5,0x13FC
         .hword   0x13FF,0x13FC,0x13F5,0x13E9,0x13D9,0x13C6,0x13B0,0x1398
         .hword   0x137F,0x1366,0x134E,0x1338,0x1325,0x1315,0x1309,0x1302
         .hword   0x1300,0x1302,0x1309,0x1315,0x1325,0x1338,0x134E,0x1356;
;
         .section  .text, “x”
;
         :
         :
         :
;
         mov      #tblpage(ToneTable),W0       ;Get upper address (page)
         mov      W0,TBLPAG                    ;Load address into TBLPAG
         mov      #tbloffset(ToneTable),W0     ;Get lower address (offset)
         tblrdl    [W0++],W1                   ; Get ToneTable into the W1
```

Set Table Page and Address using C30 (PSV LAB1.c)

- Use __ builtin function to get both TBLPAG and Offset address form the data area
- Table Read using the In-Line Assembly method

Example:

```
const unsigned char _EEDATA(1) sine_table[ ] =
                    {128,152,176,198,217,233,245,252,255,252
                    ,245,233,217,198,176,152,128,102,78,56,37,21
                    ,9,2,0,2,9,21,37,56,78,102};   // Define Byte Data

TBLPAG = __builtin_tblpage (sine_table) ;
EEPROM_Offset =  __builtin_tbloffset (sine_table) ;
```

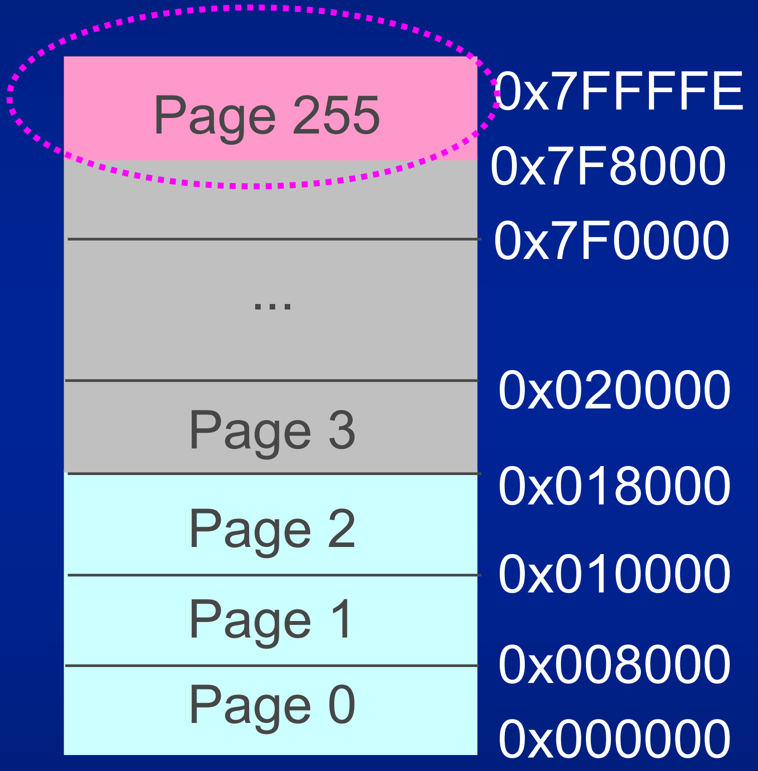
Read EEPROM Function (Table Read) in the PSV LAB1.c

- Use both W1 & W2 register for source and destination point, this is a danger operation, need to push both W1 & W2 into Stack

```
void Table_Read (unsigned int EE_Point)
{
     asm ("push.s") ;
     WREG1= EE_Point ;
     // Use the WREG1 for the ROM source Index
     WREG2 = (int) &(Sine_temp[0]) ;
     // Set the WREG2 for the destination RAM area buffer
     asm ("repeat     #3                        ; repeat 4 time , 8 bytes");\
     asm   ("tblrdl.w [W1++],[W2++] ; Table Read data from ROM to RAM ");\
     asm ("pop.s");
}
```

PSVPAG/TBLPAG
切換頁的選擇

PSVPAG=0xFF 時就可對應到 EEPROM 位址



PSV 切換頁爲
32K Bytes

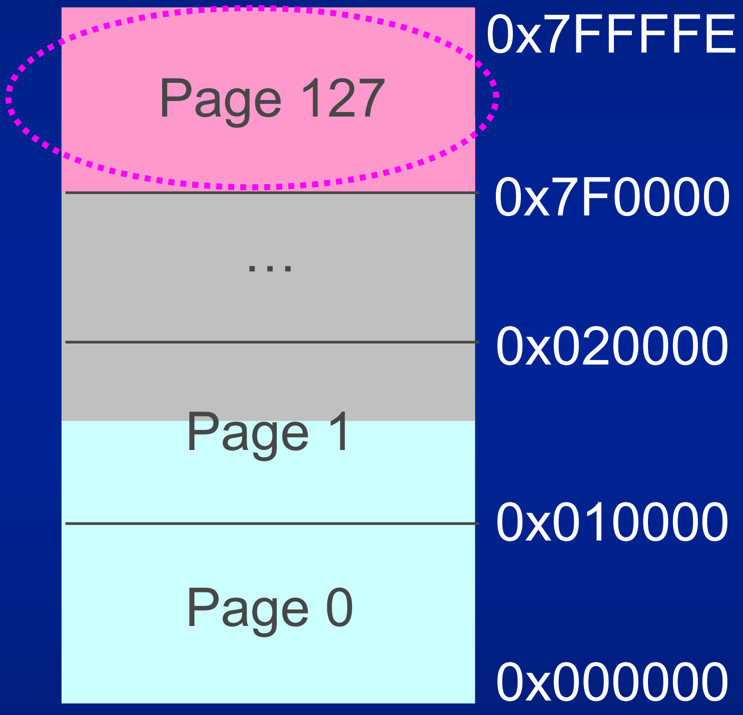
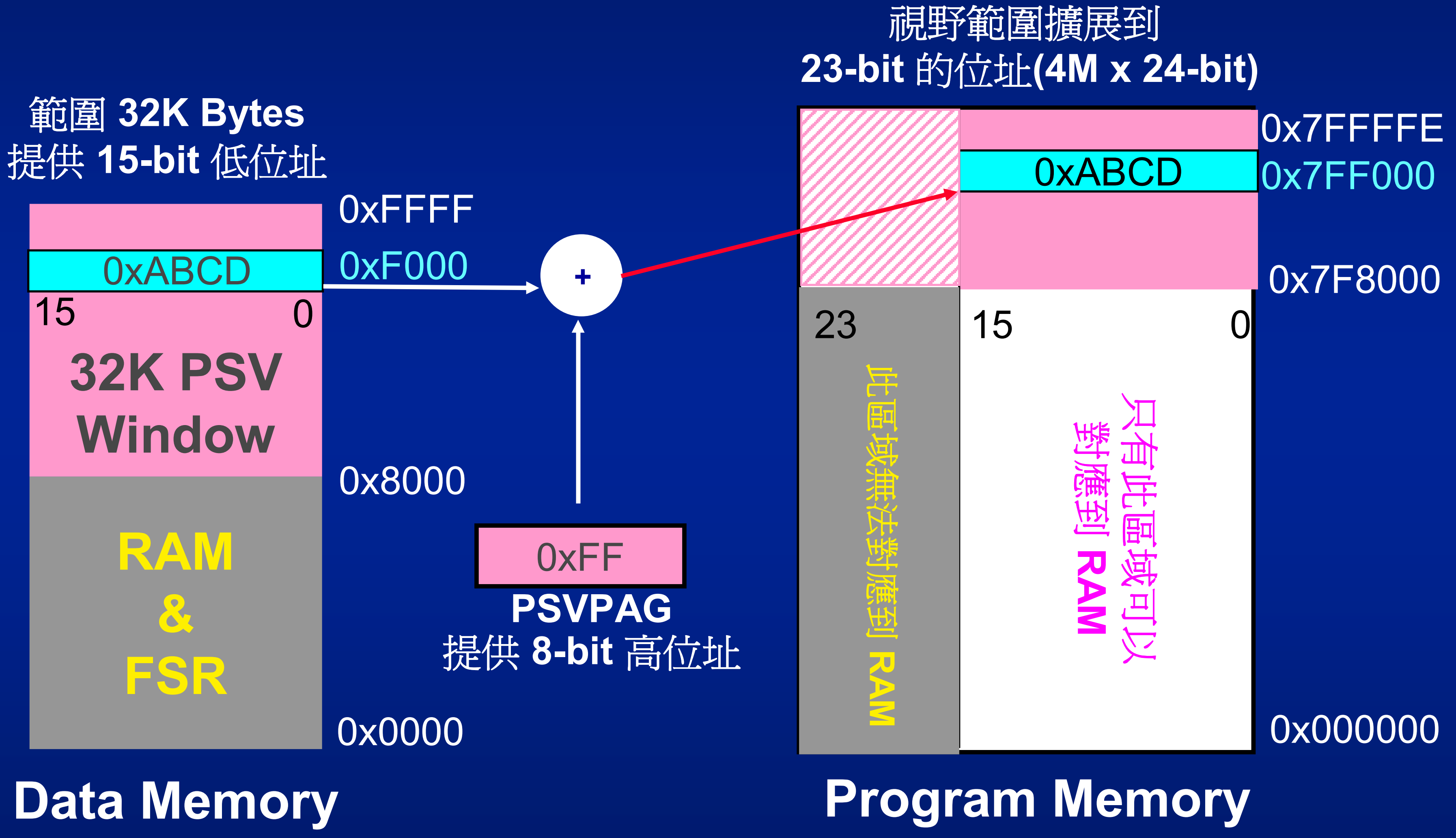


Table 切換頁爲
64K Bytes

Program Space Visibility



範圍 **32K Bytes**
提供 **15-bit** 低位址

視野範圍擴展到
23-bit 的位址**(4M x 24-bit)**

Data Memory

Program Memory

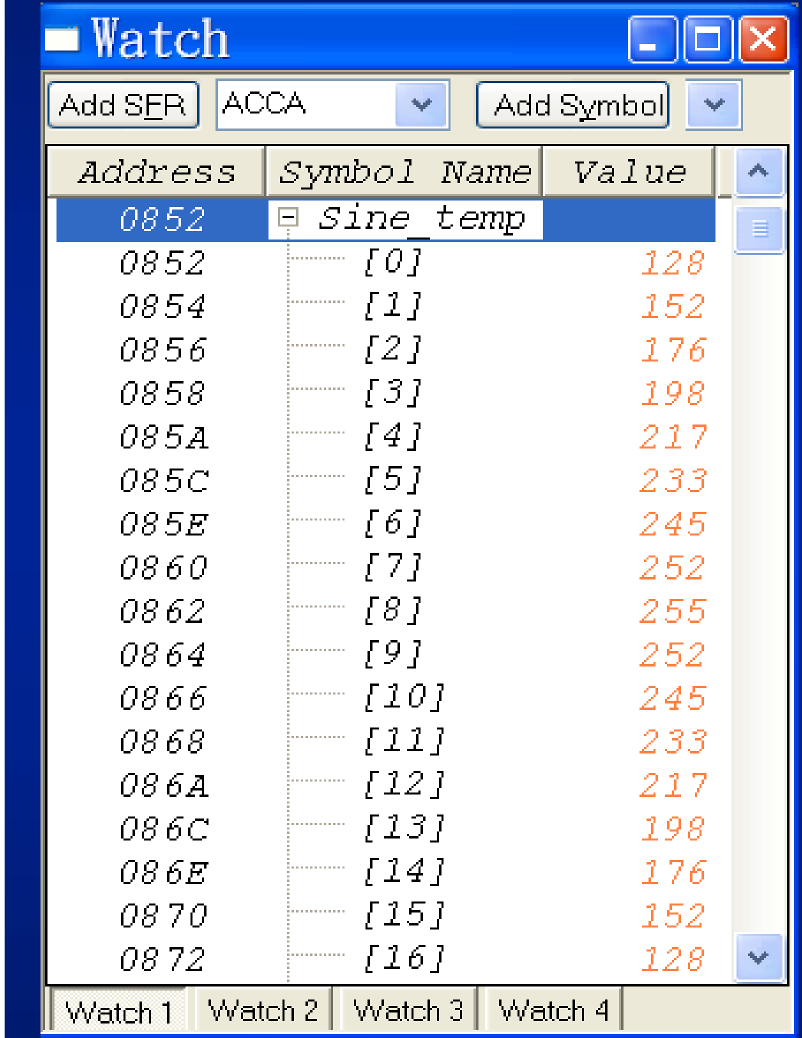
© 2004 Microchip Technology Incorporated. All Rights Reserved.

使用 PSV 方式讀取 EEPROM

```
const unsigned char _EEDATA(1) sine_table[ ] =
        {128,152,176,198,217,233,245,252,255,252
        ,245,233,217,198,176,152,128,102,78,56,37,21
        ,9,2,0,2,9,21,37,56,78,102};
CORCONbits.PSV = 1;         // 起用 PSV 功能
PSVPAG= __builtin_psvpage (sine_table); // 設定 PSVPAG = 0xFF
while(1)
    {
        if (Index > 31) Index=0;
        Sine_temp[I] = (sine_table[Index]); //讀取 EEPROM
        Index ++;
        if (I>255) while (1);           // 讀取完畢 ?
        I ++;
    }
```

© 2004 Microchip Technology Incorporated. All Rights Reserved.

使用 PSV 注意事項



Address	Symbol Name	Value
0852	⊟ Sine_temp	
0852	[0]	128
0854	[1]	152
0856	[2]	176
0858	[3]	198
085A	[4]	217
085C	[5]	233
085E	[6]	245
0860	[7]	252
0862	[8]	255
0864	[9]	252
0866	[10]	245
0868	[11]	233
086A	[12]	217
086C	[13]	198
086E	[14]	176
0870	[15]	152
0872	[16]	128

用 **PSV** 讀取 **EEPROM** 的值到 **RAM** 陣列

注意！

- 如果其它程式已經使用 PSV，PSV 將也會被設爲 1。
- 你必須在更改 PSVPAG 的值之前，將 PSVPAG 存起來程式結束後將原先的 PSVPAG 取回。
- PSV所指到的區域在MPLAB IDE下無法用 Watch 視窗檢視，使用時請務必確定PSV切換動作正常。

© 2004 Microchip Technology Incorporated. All Rights Reserved.

利用程式燒錄 EEPROM 的方式

- dsPIC30F EEPROM 除了可以用前面所提及使用 C30 提供的巨集宣告 **_EEDATA(N)** 的方式設定EEPROM 資料
- 或直接定 EEPROM 的位址
  - `const unsigned char __attribute__((space(eedata)), address(0x7fff00), aligned(1))) sine_table[ ] = {…}`
- 利用程式的執行來對 EEPROM 資料進行存、取的動作
  - 寫入前須確定目標位址的內容是否已清除
  - 寫入時間約 2 mSec

© 2004 Microchip Technology Incorporated. All Rights Reserved.

PSV LAB1

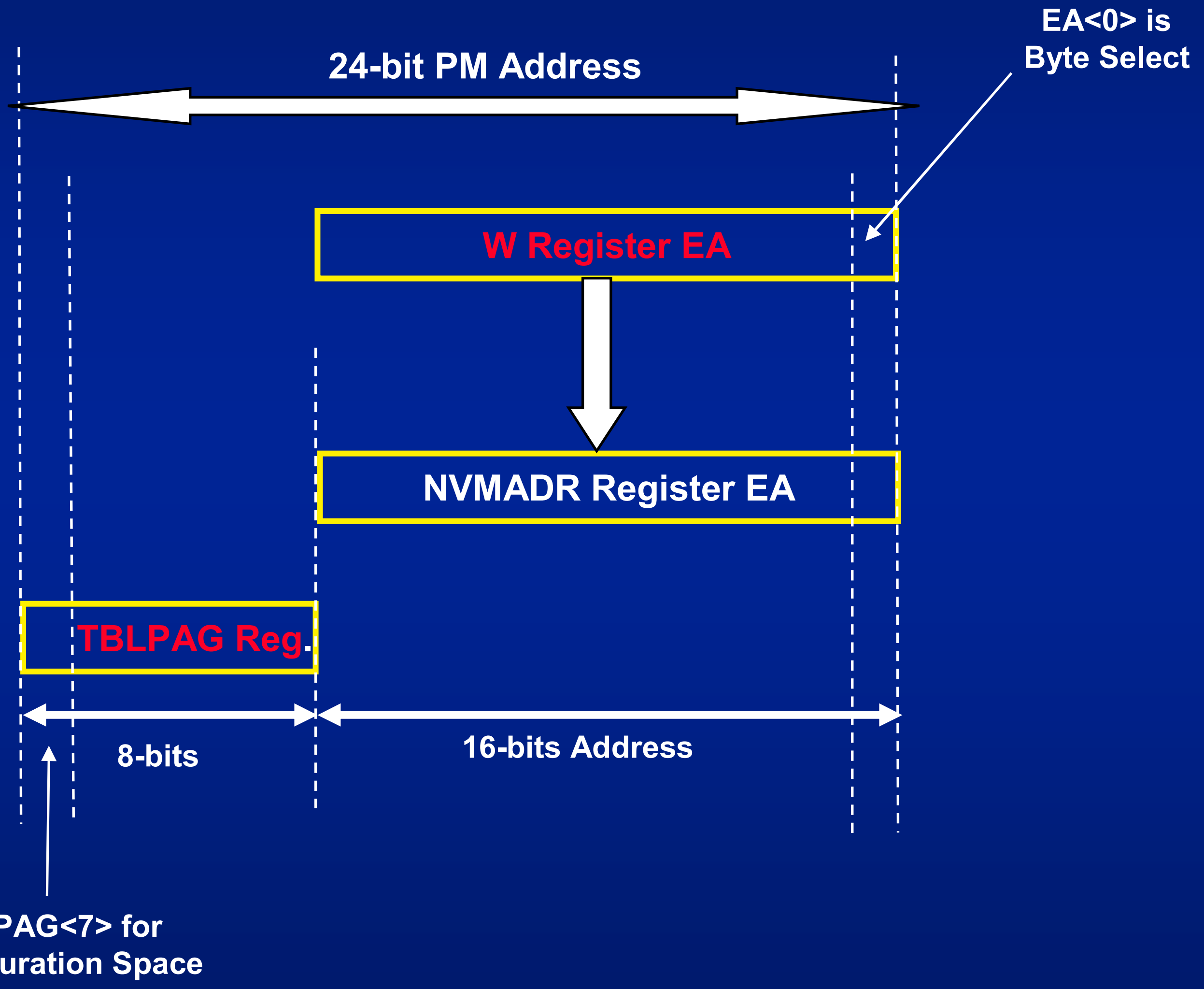
- Use Marco _EEDATA(N) to define the initialize EEPROM Data in the program code
- Use Table Read Method to read the EEPROM Data and show on LCD
- Use the PSV Method to read the EEPROM Data and show on LCD
- Compare two Method and which one is more easily
- Why still need to use the Table Read ?

© 2004 Microchip Technology Incorporated. All Rights Reserved.

Non-Volatile Memory 暫存器

- **NVMCON** – NVM 控制暫存器
  - 定義 NVM 工作模式
- **NVMADR** - NVM 位址暫存器
  - A15 ~ A0 位址從 Wn 取得
  - A23 ~ A16 自 TBLPAG 取得
- **NVMKEY** - NVM 解鎖暫存器
  - 清除或寫入資料的解鎖

NVM 位址的產生



Using NVMADR Addressing

Programming Operation

- NVMCON register (NVM 控制暫存器)
  - Erase Operations
    - ✓ = 0x4041 : Erase 1 row (32 instruction words) from 1 panel of program FLASH
    - ✓ = 0x4044 : Erase 1 data words from EEPROM
    - ✓ = 0x4045 : Erase 1 row (16 data words) from EEPROM
  - Programming Operations
    - ✓ = 0x4001 : Program 4 instruction words into FLASH
    - ✓ = 0x4006 : Program 1 data word into Configuration
    - ✓ = 0x4004 : Program 1 data word into data EEPROM
    - ✓ = 0x4005 : Program 1 row (16 words) into EEPROM

© 2004 Microchip Technology Incorporated. All Rights Reserved.

Data EEPROM Prog.

- EEPROM is accessed by Table Read/Write
- Programming Operation
  - Erase one word
  - Erase one row (16 words)
  - Program one word
  - Program one row (16 words)
- Erase the data of target address before you write data into the same location

© 2004 Microchip Technology Incorporated. All Rights Reserved.

Erase one EEPROM word

1. Erase one EEPROM word.
   - Setup NVMCON register to erase one EEPROM word.
   - Write address of word to be erased into TBLPAG, NVMADR registers.
   - Clear NVMIF status bit and enable NVM interrupt (optional).
   - Write the key sequence to NVMKEY.
   - Set the WR bit. This will begin erase cycle.
   - Either poll the WR bit or wait for the NVM interrupt.
2. Write data word into data EEPROM write latch.
3. Program the data word into the EEPROM.
   - Setup the NVMCON register to program one EEPROM word.
   - Clear NVMIF status bit and enable NVM interrupt (optional).
   - Write the key sequence to NVMKEY.
   - Set the WR bit. This will begin the program cycle.
   - Either poll the WR bit or wait for the NVM interrupt.

© 2004 Microchip Technology Incorporated. All Rights Reserved.

EEPROM Row Programming

1. Read one row of data EEPROM (16 words) and store into data RAM as a data "image". The section of EEPROM to be modified must fall on an even 16-word address boundary.
2. Update the data image with the new data.
3. Erase the EEPROM row.
   - Setup the NVMCON register to erase one row of EEPROM.
   - Clear NVMIF status bit and enable NVM interrupt (optional).
   - Write the key sequence to NVMKEY.
   - Set the WR bit. This will begin the erase cycle.
   - Either poll the WR bit or wait for the NVM interrupt.
4. Write the 16 data words into the data EEPROM write latches.
5. Program a row into data EEPROM.
   - Setup the NVMCON register to program one row of EEPROM.
   - Clear NVMIF status bit and enable NVM interrupt (optional).
   - Write the key sequence to NVMKEY.
   - Set the WR bit. This will begin the program cycle.
   - Either poll the WR bit or wait for the NVM interrupt.

© 2004 Microchip Technology Incorporated. All Rights Reserved.

Set up the Point to EEPROM

- Assembly Code:
  - Remember the EEPROM start address from
    - 0x7FF000 ~ 0x7FFFFF (Fixed on 0x7FFFFF)

Example Code :

```
; Set up a pointer to the EEPROM location to be erased.
                MOV #0x7F,W0
                MOV W0,TBLPAG
                MOV #tbloffset(EE_ADDR),W0
                MOV W0,NVMADR
; Setup NVMCON to erase one word of data EEPROM
                MOV #0x4044,W0
                MOV W0,NVMCON
                :
                :
```

© 2004 Microchip Technology Incorporated. All Rights Reserved.

Define the Constant for NVMCON (eeprom.h)

- Data EEPROM Operations:
  - Define the constant value for NVMCON

```
/* Data EEPROM Erase Operations */
#define EE_ERS_DATA_WORD       0x4044
#define EE_ERS_DATA_ROW        0x4045

/* Data EEPROM Program Operations */
#define EE_PRG_DATA_WORD       0x4004
#define EE_PRG_DATA_ROW        0x4005
```

© 2004 Microchip Technology Incorporated. All Rights Reserved.

EEPROM Lab2

Erasing Data EEPROM

1. Set the **NVMCON** for desired operation

   (word or row erase , defined in the EEPROM.h)

2. Load the **NVMADR** and **NVMADRU** for target Addr

```
NVMADRU = 0x7F;   // TBLPAG for EEPROM
NVMADR = & mu;    // address to erase
```

3. Write 0x55 and 0xAA to **NVMKEY** (unlock EEPROM)
4. Set the “WR” bit in **NVMCON** (start the erase)
5. Poll the “WR” bit in **NVMCON** (wait for completion)

© 2004 Microchip Technology Incorporated. All Rights Reserved.

EEPROM Lab2

Programming 1 Row of Data EEPROM

1. Setup **NVMCON** for desired operation (row or word)
2. Load the **TBLPAG** and write latches for target location

```
TBLPAG = 0x7F;
WREG6 = source_address;
WREG7 = destination_address;
/*even address boundaries (5 LSBs are no effect)*/
REPEAT #15
TBLWTL [W6++], [W7++]
```

3. Write 0x55 and 0xAA to **NVMKEY** (unlock EEPROM)
4. Set the “WR” bit in **NVMCON** (start the programming)
5. Poll the “WR” bit in **NVMCON** (wait for completion)

EEPROM Lab2

Unlocked the Key Sequence

1. Must always ERASE before PROGRAMMING!
2. Interrupts must be disabled (or masked)
    - Set all "IE"s to 0 or set CPU IPL to 7
3. Row operations must start on row boundary
    - Target address must be a multiple of 0x20
4. NVMKEY sequence must be followed:

Use "C" macro called

`START_PROGRAMMING`

```
MOV   #0x55, W7
MOV   W7, NVMKEY
MOV   #0xAA, W7
MOV   W7, NVMKEY
BSET NVMCON, #WR
```

© 2004 Microchip Technology Incorporated. All Rights Reserved.

EEPROM Lab2

- Test Erase and Program EEPROM function using MPLAB SIM
- Open the "EEPROM Lab2.mcp", use the break point function to check with :
  - Erase All Function
  - Erase a word
  - Erase a row
  - Program a word
  - Program a row
  - Read EEPROM data with PSV

© 2004 Microchip Technology Incorporated. All Rights Reserved.

Other RTSP Operations

- Program Memory
  - Program 1 row (32 words x 24-bits)
  - Erase 1 row (32 words x 24-bits)
- Configuration Memory
  - FOSC, WDT, FBORPOR, FGS
  - Program 1 word (not erasable)
- Use same technique as Data EEPROM
  - Self-timed operation

© 2004 Microchip Technology Incorporated. All Rights Reserved.