LẬP TRÌNH PIC
SỬ DỤNG CCS

PIC Product Selector

http://www.microchip.com/productselector/MCUProductSelector.html

Product Family	PIC16F84	PIC16F87	PIC16F690	PIC16F887	dsPIC33FJ128GP P202
Architecture	8	8	8	8	16
5K $ Pricing	3.11	2.06	1.20	1.78	3.44
Flash (KB)	1.75	7	7	14	128
EEPROM (Bytes)	64	256	256	256	0
RAM (KB)	0.06	0.36	0.25	0.36	8.00
CPU Speed (MHz, MPS)	[20,5]	[20,5]	[20,5]	[20,5]	[80,40]
Low Power	No	Yes	Yes	Yes	Yes
Comparators	0	2	2	2	2
ADC Channels	0	0	12	14	10
ADC Bits	-	-	10	10	12
Total UART	-	1	1	1	2

PIC Product Selector

http://www.microchip.com/productselector/MCUProductSelector.html

Product Family	PIC16F84	PIC16F87	PIC16F690	PIC16F887	dsPIC33FJ128GP P202
SPI	0	1	1	1	2
I2C	0	1	1	1	1
USB	-	-	-	-	-
Ethernet	-	-	-	-	-
LIN	-	-	Yes	Yes	-
CAN	-	-	-	-	-
Total Timers	1	3	3	3	7
Input Capture	0	1	1	2	4
PWM Channels	0	1	1	2	4
Parallel Port	-	-	-	-	PMP
Segment LCD	0	0	0	0	0
Supply Voltage	2 to 6	2 to 5.5	2 to 5.5	2 to 5.5	3 to 3.6

Một chương trình trong CCS

```
#include < 16F877 .h >              // Các chỉ thị tiền xử lý
#device PIC6f877 *=16 ADC=10
#use delay(clock=20000000)
. . . .
Int a,b                            // Các khai báo biến
. . . .
Void thuc_hien_ADC ( )             // Các hàm con
{ . . .
. . .
}
#INT_TIMER1                        // Các hàm phục vụ ngắt
Void phuc_vu_ngat_timer ( )
{ . . .
. . .
}
Main ( )                           //Chương trình chính
{ . . .
. . .
}
```

4

Hàm

1. Hàm không trả về giá trị

```
Void tinh_toan ( )
{
z= x+y ;
}
```

2. Hàm có trả về giá trị

```
int tinh_toan (int a, int b)
{
. . . . . .
Return (a+b) ;
}
```

Ví dụ

```
int tinh_toan (int a ,int b)
{
Return (a+b) ;
}
Main ( )
{
Int c, d, e ;
c = 2 ;
d = 4;
e = tinh_toan(c ,d );
}
```


5

Biến

- int1 số 1 bit
- int8 số nguyên 1 byte (8 bit)
- int16 số nguyên 16 bit
- int32 số nguyên 32 bit
- float32 số thực 32 bit

C standard type	Default type
short	Int1
char	unsigned int8
Int	Int8
long	int16
long long	int32
float	float32

- Số có dấu: thêm signed vào phía trước
- Số không dấu: mặc nhiên, hoặc thêm unsigned vào phía trước

Tầm giá trị

int1	0, 1 (true, false)
int 8	$0 \rightarrow 2^8 - 1$
int16	$0 \rightarrow 2^{16} - 1$
int32	$0 \rightarrow 2^{32} - 1$
signed int8	$-2^7 \rightarrow 2^7 - 1$
signed int16	$-2^{15} \rightarrow 2^{15} - 1$
signed int32	$-2^{31} \rightarrow 2^{31} - 1$
float32	$-1.5 \times 10^{45} \rightarrow 3.4 \times 10^{38}$

Ví dụ:

```
int a,b,c;
signed int d,e;
char f;
int x = 1; //biến x loại int
           //và có giá trị đầu là 1
int16 y[100]; //biến mảng 101 phần tử
```

6

Hằng số

- **int const a=12;**
- **int16 const b=65535;**
- **int const c[5]={2,4,15,0,155};**
- **int16 const d[3]={0,345,12,430};**

7

Phát biểu lệnh (Statement)

STATEMENT	EXAMPLE

if (expr) **stmt;** [else **stmt;**]	`if (x==25)`
	`  x=1;`
	`else`
	`  x=x+1;`

while **(expr) stmt;**	`while (get_rtcc()!=0)` `  putc('n');`

do **stmt** while **(expr);**	`do {`
	`  putc(c=getc());`
	`} while (c!=0);`

for **(expr1;expr2;expr3) stmt;**	`for (i=1;i<=10;++i)` `  printf("%u\r\n",i);`

switch (expr) {
case **cexpr: stmt; //one or more case**
[default:stmt]
... }
	`switch (cmd) {`
	`  case 0:  printf("cmd 0");`
	`  break;`
	`  case 1: printf("cmd 1");`
	`  break;`
	`  default: printf("bad cmd");`
	`  break; }`

8

Phát biểu lệnh (Statement)

return **[expr];**	return (5);
goto **label;**	goto loop;
label: stmt;	loop: I++;
break;	break;
continue;	continue;
expr;	i=1;
;	;
{[stmt]} Zero or more	{a=1; b=1;}

- **return dùng để trả giá trị về cho hàm (ví dụ: return (5); return (x); return (a+b), nếu không cần trả giá trị thì chỉ dùng return;**
- **break thoát khỏi vòng lặp while**
- **continue quay trở về đầu vòng lặp while**

9

Toán tử (Operators)

+	Addition Operator
+=	Addition assignment operator, x+=y, is the same as x=x+y
&=	Bitwise and assignment operator, x&=y, is the same as x=x&y
&	Address operator
&	Bitwise and operator
^=	Bitwise exclusive or assignment operator, x^=y, is the same as x=x^y
^	Bitwise exclusive or operator
I=	Bitwise inclusive or assignment operator, xl=y, is the same as x=xly
I	Bitwise inclusive or operator
?:	Conditional Expression operator
- -	Decrement
/=	Division assignment operator, x/=y, is the same as x=x/y
/	Division operator
==	Equality
>	Greater than operator
>=	Greater than or equal to operator
++	Increment
*	Indirection operator
!=	Inequality

10

Toán tử (Operators)

<<=	Left shift assignment operator, x<<=y, is the same as x=x<<y
<	Less than operator
<<	Left Shift operator
<=	Less than or equal to operator
&&	Logical AND operator
!	Logical negation operator
ll	Logical OR operator
%=	Modules assignment operator x%=y, is the same as x=x%y
%	Modules operator
*=	Multiplication assignment operator, x*=y, is the same as x=x*y
*	Multiplication operator
~	One's complement operator
>>=	Right shift assignment, x>>=y, is the same as x=x>>y
>>	Right shift operator
->	Structure Pointer operation
-=	Subtraction assignment operator
-	Subtraction operator
sizeof	Determines size in bytes of operand

11

Các phép toán

- sin(x)
- cos(x)
- tan(x)
- asin(x)
- acos(x)
- atan(x)
- ceil(x) làm tròn tăng
- floor(x) làm tròn giảm
- exp(x) e^x
- log(x)
- log10(x)
- pow(x,y) x^y
- sqrt(x) căn bậc hai của x

- bit_clear(var,bit) xóa vị trí *bit* của biến *var*
- bit_set(var,bit) set vị trí *bit* của biến *var*
- bit_set(var,bit) trả về giá trị của vị trí *bit* của biến *var*
- swap(var) hoán chuyển 4 bit thấp và 4 bit cao
- make8(var,offset) trả về 1 byte trích từ biến *var*
  - *var:* biến 16 hay 32 bit
  - *offset*: vị trí byte cần trích (0,1,2,3)
- make16(varhigh,varlow) trả về giá trị 2 byte kết hợp từ *varhigh* và *varlow*
- make32(var1,var2,var3,var4) trả về giá trị 4 byte kết hợp từ *var1, var2, var3,* và *var4*

12

Delay

- Để dùng hàm delay, cần có khai báo
  #use delay (clock=20000000)
  ở đầu file (ví dụ cho fosc=20 MHz)
- delay_cycles(x) delay *x* (hằng số từ 1→255) chu kỳ lệnh
  - 1 chu kỳ lệnh = 4 chu kỳ máy
- delay_us(x) delay *x* µs
  - x là biến (int16) hoặc hằng từ 0→65535
- delay_ms(x) delay *x* ms
  - x là biến (int16) hoặc hằng từ 0→65535

13

Xuất nhập I/O

- Để sử dụng Port A và Port B, cần có khai báo
  #use fast_io(A)
  #use fast_io(B)
  ở đầu file
- Hoặc #use fast_io(ALL)
- set_tris_a(value) xác lập Port A (0: output, 1: input)
- set_tris_b(value) xác lập Port B

Ví dụ:

```
SET_TRIS_B( 0x0F );
   // B7,B6,B5,B4 are outputs
   // B3,B2,B1,B0 are inputs
```

- output_a(value) xuất ra Port A
- output_b(value) xuất ra Port B

Ví dụ: OUTPUT_B(0xf0);

- output_high(pin) xuất mức 1 ra một chân port
- output_low(pin) xuất mức 0 ra một chân port

Ví dụ:

```
output_high(PIN_A0);
output_low(PIN_A1);
```

(Pin constants are defined in the devices .h file)

- output_bit(pin,value) xuất *value (0 hay 1)* ra *pin*

Ví dụ:

```
output_bit( PIN_B0, 0);
// Same as output_low(pin_B0);

output_bit( PIN_B0,input( PIN_B1 ) );
// Make pin B0 the same as B1
```

- output_float(pin) tạo cực thu hở

14

Xuất nhập I/O

- input_a() nhập từ Port A
- input_b() nhập từ Port B

Ví dụ:

```
data = input_b();
```

- input(pin) nhập từ một chân port

Ví dụ:

```
while ( !input(PIN_B1) );
// waits for B1 to go high

if( input(PIN_A0) )
  printf("A0 is now high\r\n");
```

15

Tạo xung vuông

Ví dụ: Tạo xung vuông f=1 KHz tại chân RB0 (Cách 1)

```
#include <16F84.h>
#use delay(clock=20000000)
Main()
{
  while(1)
    {
      output_high(pin_B0);
      delay_us(500); // delay 250us
      output_low (pin_B0);
      delay_us (500 );
    }
}
```

Ví dụ: Tạo xung vuông f=1 KHz tại chân RB0 (Cách 2)

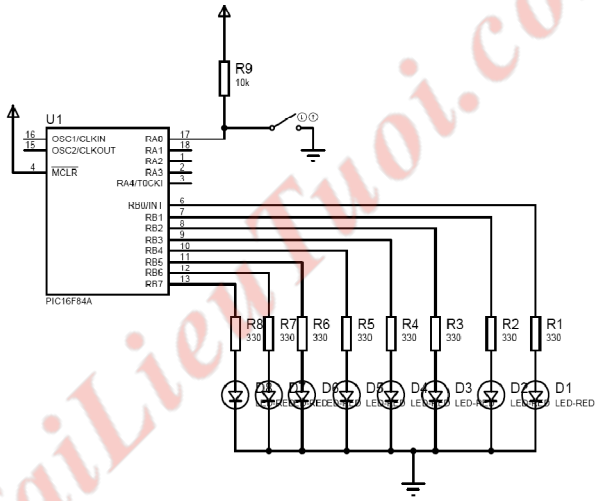
```
#include <16F84.h>
#use delay(clock=20000000)
Main()
{
  int1 x;
  while(1)
    {
      output_bit(pin_B0,!x);
      delay_us(500);
    }
}
```

16

LED chạy đuổi (LED chaser)

1 LED sáng được chạy từ trái qua phải ở port B [khi chân RA0=1] hoặc từ phải sang trái [khi chân RA0=0])

Sơ đồ mạch: (Giả sử phím nhấn không bị nẩy [rung])



17

LED chạy đuổi (LED chaser)

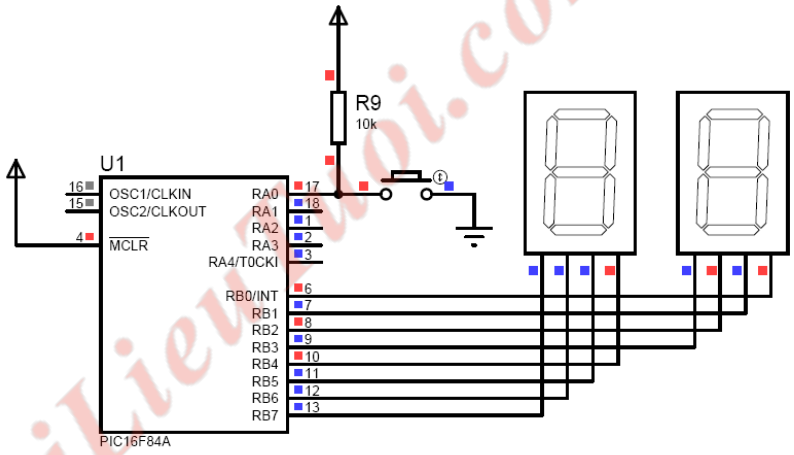
```
#include <16F84.h>
#use delay(clock=20000000)
#use fast_io(A)
#use fast_io(B)
 void main()
{
  int pattern=1;
  set_tris_a(0x01);
          /* Chân A0 là ngõ nhập */
  set_tris_b(0x00); /* Port B xuất */
  while (1)
  {
    output_b(pattern);
    delay_ms(20);
    if (input(PIN_A0)==1)
      /* Rotate Left */
      if (pattern != 0x80)
         pattern <<= 1;
else
         pattern = 1;
    else
      /* Rotate Right */
      if (pattern != 1)
         pattern >>= 1;
      else
         pattern = 0x80;
  }
}
```

18

Mạch đếm lên

Mạch đếm lên thập phân 2 ký số với xung nhịp kích cạnh xuống

Sơ đồ mạch: (Giả sử phím nhấn không bị nẩy [rung] và LED 7 đoạn có sẵn mạch giải mã)



19

Mạch đếm lên

```
#include <16F84.h>
#use delay(clock=20000000)
#use fast_io(A)
#use fast_io(B)
 int bin2BCD(int bin)
{ /* CT đổi từ 1 số nhị phân ra số BCD 2
ký số, chỉ đúng cho số nhị phân này có trị
<= 99 */
  int BCD;
  BCD = ((bin/10) << 4) + bin % 10;
  return BCD;
}
void main()
{
  int counter=0,counter_BCD=0;
  set_tris_a(0x01); /* Pin A0 is Input port
pin */
  set_tris_b(0x00); /* Output Port B
configuration*/
while (1)
  {
    output_b(counter_BCD);
    while(!input(PIN_A0));  // đợi cho
đến khi A0 = 1
    while(input(PIN_A0));   // đợi cho đến
khi A0 = 0 → phát hiện cạnh xuống
    counter++;
    if (counter  == 100) counter = 0;
    counter_BCD=bin2BCD(counter);
  }
}
```

20

Timer

- **setup_counters (*rtcc_state*, *ps_state*)**
  - ***rtcc_state*:** RTCC_INTERNAL, RTCC_EXT_L_TO_H or RTCC_EXT_H_TO_L
  - ***ps_state*:** RTCC_DIV_2, RTCC_DIV_4, RTCC_DIV_8, RTCC_DIV_16, RTCC_DIV_32, RTCC_DIV_64, RTCC_DIV_128, RTCC_DIV_256, WDT_18MS, WDT_36MS, WDT_72MS, WDT_144MS, WDT_288MS, WDT_576MS, WDT_1152MS, WDT_2304MS

Ví dụ: `setup_counters (RTCC_INTERNAL, WDT_2304MS);`

`setup_counters (RTCC_EXT_H_TO_L, RTCC_DIV_1);`

setup_timer_0 and setup_WDT are the recommended replacements when possible

- **setup_timer0(mode)**
  - ***mode*** may be one or two of the constants defined in the devices .h file.

  RTCC_INTERNAL, RTCC_EXT_L_TO_H or RTCC_EXT_H_TO_L

  RTCC_DIV_2, RTCC_DIV_4, RTCC_DIV_8, RTCC_DIV_16, RTCC_DIV_32, RTCC_DIV_64, RTCC_DIV_128, RTCC_DIV_256

One constant may be used from each group or'ed together with the | operator.

Ví dụ: `setup_timer_0 (RTCC_DIV_2|RTCC_EXT_L_TO_H);`

21

Timer

- **set_timer0(value) bộ Timer0 đếm lên từ giá trị *value*, khi đến 255 sẽ đếm lên 0, 1, 2, …**

Ví dụ:

```
// 20 mhz clock, no prescaler, set timer 0
// to overflow in 35us
   set_timer0(81);      // 256-(.000035/(4/20000000)) = 81
```

- **get_timer0() trả về giá trị thời gian thực của bộ đếm**

Ví dụ:

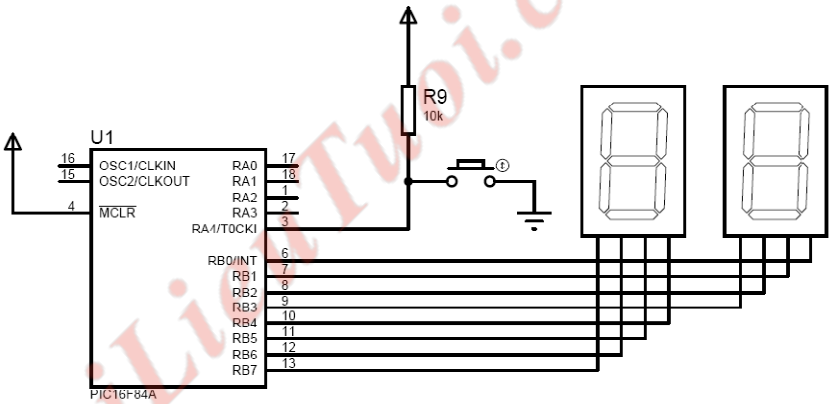
```
   set_timer0(0);
   while ( get_timer0() < 200 ) ;

   int8 counter;
   counter = get_timer();
```

22

Mạch đếm lên dùng Timer

Mạch đếm lên thập phân 2 ký số với xung nhịp kích cạnh xuống (dùng Timer của PIC)

Sơ đồ mạch: (Giả sử phím nhấn không bị nẩy [rung] và LED 7 đoạn có sẵn mạch giải mã)



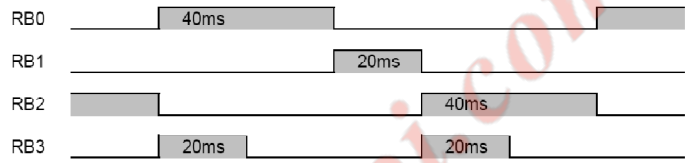
Mạch đếm lên

```
#include <16F84.h>
#use delay(clock=20000000)
#use_fast_io(A)
#use_fast_io(B)
int bin2BCD(int bin)
{ // Chương trình đổi từ 1 số nhị phân ra
số BCD 2 ký số, chỉ đúng cho số nhị phân
này có trị <= 99
  int BCD;
  BCD = ((bin/10) << 4) + bin % 10;
  return BCD;
}
 void main()
{
  int counter=0,counter_BCD=0;
  set_tris_a(0x10); /* Pin A4 (external
          Counter) is Input port pin */
  set_tris_b(0x00); /* Output Port B
          configuration*/
  setup_counters(RTCC_EXT_H_TO_L,
   RTCC_DIV_1);   /* Set up Timer 0 */
  set_timer0(0); /* Initial value of
                    Counter */
   while (1)
  {
    output_b(counter_BCD);
    counter=get_timer0();
    if (counter  == 100)
       {
          counter = 0;
          set_timer0(0); /* Initial value of
                            Counter */
       }
    counter_BCD=bin2BCD(counter);
   }
}
```

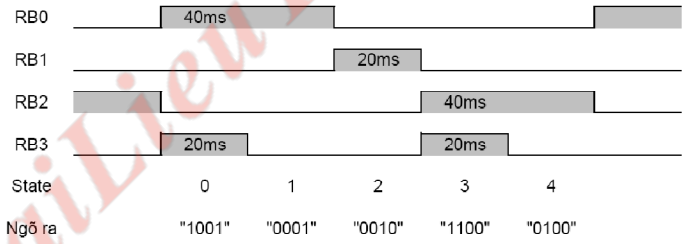
24

Tạo dạng sóng (dùng bảng)

Tạo dạng sóng tuần hoàn có dạng sau ở Port B: (mức 1 thì LED sáng ở chân đó)



Từ dạng sóng trên ta có thể phân tích thành máy trạng thái Moore với thời gian tồn tại của 1 trạng thái là 20ms và trạng thái bắt đầu là trạng thái 0. Ta có thể dùng bảng để chứa trị số ra tương ứng với mỗi trạng thái.



25

Tạo dạng sóng (dùng bảng)

```
#include <16F877.h>
#use delay(clock=20000000)
#use_fast_io(B)

void main()
{
int state[5]={0x09, 0x01, 0x02, 0x0C, 0x04};

set_tris_b(0x00); // Đặt cấu hình xuất cho Port B

while(1)
{
        for (i = 0; i <5; i++)
        {
                output_b(state[i]);
                delay_ms(20);
                }
        }
}
```

26

Tạo dạng sóng (dùng switch … case)

```
#include <16F877.h>
#use delay(clock=20000000)
#use_fast_io(B)
 void main()
{
 int state=0;
 set_tris_b(0x00); // Đặt cấu hình xuất cho Port B
 while(1)
        {
           switch(state){
                    case 0: state =1; output_b(0x09); break;
                    case 1: state =2; output_b(0x01); break;
                    case 2: state =3; output_b(0x02); break;
                    case 3: state =4; output_b(0x0C); break;
                    case 4: state =0; output_b(0x04); break;
                    default: state =0; output_b(0x00);
                    } // end of switch-case
                    delay_ms(20);
           } // end of while
}
```

27

Interrupt

Khai báo ngắt

#int_ext external interrupt (RB0/INT)

#int_timer0 timer0 overflow

#int_rb Port B any change on B4-B7

#int_eeprom write complete

- enable_interrupts (*level*) cho phép ngắt

Ví dụ:

```
enable_interrupts(global);
enable_interrupts(int_ext);
enable_interrupts(int_timer0);
enable_interrupts(int_rb);
enable_interrupts(int_eeprom);
```

- ext_int_edge (source,edge)
  - source=0, 1, 2 (default=0)
  - edge= L_TO_H or H_TO_L

Ví dụ:

```
ext_int_edge(H_TO_L);
```

- disable_interrupts (*level*) cấm ngắt

Ví dụ:

```
disable_interrupts(timer0);
```

- clear_interrupt (*level*) xóa cờ ngắt

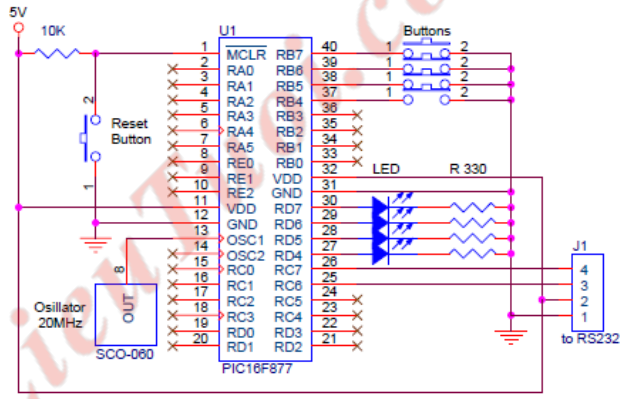
Ví dụ:

```
disable_interrupts(timer0);
```

28

Ví dụ dùng ngắt INT_RB

Điều khiển LED on/off bằng nút nhấn tương ứng



29

Ví dụ dùng ngắt INT_RB

```
#include<16F877.h>
#use fast_io(B)
#use fast_io(D)
#int_rb
void RB_LED( )
{
  output_d(input_b());
}
main( )
{
  set_tris_b(0xF0);
  set_tris_d(0x00);
  enable_interrupts(GLOBAL);
  enable_interrupts(INT_RB);
  while(1)
  {
    …
  }
}
```

30

16F84.h (1)

```
//Standard Header file for the PIC16F84 device //
#device PIC16F84
#nolist
//Program memory: 1024x14
//Data RAM: 68  Stack: 8
/////// I/O: 13   Analog Pins: 0
/////// Data EEPROM: 64
/////// C Scratch area: 0C   ID Location: 2000
/////// Fuses:
LP,XT,HS,RC,NOWDT,WDT,NOPUT,PUT,PROTECT,NOP
ROTECT
// I/O
// Discrete I/O Functions: SET_TRIS_x(),
OUTPUT_x(), INPUT_x(),
//                PORT_x_PULLUPS(), INPUT(),
//                OUTPUT_LOW(), OUTPUT_HIGH(),
//                OUTPUT_FLOAT(), OUTPUT_BIT()
// Constants used to identify pins in the above are:
#define PIN_A0  40
#define PIN_A1  41
#define PIN_A2  42
#define PIN_A3  43
#define PIN_A4  44
#define PIN_B0  48
#define PIN_B1  49
#define PIN_B2  50
#define PIN_B3  51
#define PIN_B4  52
#define PIN_B5  53
#define PIN_B6  54
#define PIN_B7  55
// Useful defines
#define FALSE 0
#define TRUE 1

#define BYTE int8
#define BOOLEAN int1

#define getc getch
#define fgetc getch
#define getchar getch

#define putc putchar
#define fputc putchar
#define fgets gets
#define fputs puts
```

31

16F84.h (2)

```
// Control
// Control Functions:  RESET_CPU(), SLEEP(),
RESTART_CAUSE()
// Constants returned from RESTART_CAUSE() are:
#define WDT_FROM_SLEEP  3
#define WDT_TIMEOUT     11
#define MCLR_FROM_SLEEP 19
#define MCLR_FROM_RUN   27
#define NORMAL_POWER_UP 25
#define BROWNOUT_RESTART 26

// Timer 0
// Timer 0 (AKA RTCC)Functions:
SETUP_COUNTERS() or SETUP_TIMER_0(),
//                SET_TIMER0() or SET_RTCC(),
//                GET_TIMER0() or GET_RTCC()
// Constants used for SETUP_TIMER_0() are:
#define T0_INTERNAL   0
#define T0_EXT_L_TO_H 32
#define T0_EXT_H_TO_L 48

#define T0_DIV_1      8
#define T0_DIV_2      0
#define T0_DIV_4      1
#define T0_DIV_8      2
#define T0_DIV_16     3
#define T0_DIV_32     4
#define T0_DIV_64     5
#define T0_DIV_128    6
#define T0_DIV_256    7

#define T0_8_BIT      0

#define RTCC_INTERNAL   0     // The following are
provided for compatibility
#define RTCC_EXT_L_TO_H 32    // with older
compiler versions
#define RTCC_EXT_H_TO_L 48
#define RTCC_DIV_1      8
#define RTCC_DIV_2      0
#define RTCC_DIV_4      1
#define RTCC_DIV_8      2
#define RTCC_DIV_16     3
#define RTCC_DIV_32     4
#define RTCC_DIV_64     5
#define RTCC_DIV_128    6
#define RTCC_DIV_256    7
#define RTCC_8_BIT      0
```

32

16F84.h (3)

```
// Constants used for SETUP_COUNTERS() are the
above
// constants for the 1st param and the following for
// the 2nd param:

// WDT
// Watch Dog Timer Functions: SETUP_WDT() or
SETUP_COUNTERS() (see above)
//                    RESTART_WDT()
// WDT base is 18ms
//

#define WDT_18MS      8
#define WDT_36MS      9
#define WDT_72MS      10
#define WDT_144MS     11
#define WDT_288MS     12
#define WDT_576MS     13
#define WDT_1152MS    14
#define WDT_2304MS    15
```

```
// INT
// Interrupt Functions: ENABLE_INTERRUPTS(),
DISABLE_INTERRUPTS(),
//                CLEAR_INTERRUPT(),
INTERRUPT_ACTIVE(),
//                EXT_INT_EDGE()
//
// Constants used in EXT_INT_EDGE() are:
#define L_TO_H          0x40
#define H_TO_L          0
// Constants used in
ENABLE/DISABLE_INTERRUPTS() are:
#define GLOBAL              0x0B80
#define INT_RTCC            0x000B20
#define INT_RB              0x00FF0B08
#define INT_EXT_L2H         0x50000B10
#define INT_EXT_H2L         0x60000B10
#define INT_EXT             0x000B10
#define INT_EEPROM          0x000B40
#define INT_TIMER0          0x000B20

#list
```

33