الباب الثانى CHAPTER (2)

برمجة الـميكروكونترلر بإستخدم لغة السى

Microcontroller Programming by Using C Language

يوجد العديد من الـ C compilers فى السوق للـ PIC microcontroller بشكل عام. وتستخدم هذه الـ Compilers لكتابة برامج الـ PIC microcontroller بلغة الـ C وهى لغة High level بدلاً من كتابة البرنامج بلغة الـ Assembly أو التجميع. حيث أن الـ Assembly language تعتبر من اللغات الـ Low level القريبة من الـ Hardware الداخلى للـ microcontroller والتى يصعب إستخدامها لكتابة البرامج المعقدة والتى تحتوى على عمليات حسابية فى الغالب. ومن أشهر الـ C compiler النستخدمة لبرمجة الـ PIC microcontroller والمتوفرة فى السوق حالياً هى:

1. CCS PICC.
2. Hi-Tech PICC.
3. mikroC.
4. MPLAB C18.

ولكننا فى هذا الكتاب سوف نستخدم برنامج الـ mikroC نظراً لسهولة التعامل معه وإنتشارة بشكل واسع بين الهواة ومبرمجين الـ Embedded system. ويتميز هذا الـ Compiler بأشياء كثيرة مثل المكتبات Libraries الغزيرة والتى تغنى المستخدم عن التعامل بشكل مباشر عن تفاصيل الـ Hardware الداخلية للـ Microcontroller. كما يحتوى على محاكى داخلى Built-in simulator وغيرها من الأشياء التى تسهل للمستخدم التعامل مع الـ PIC microcontroller. ويمكن تحميل نسخه Demo من موقع الشركة www.microe.com ولكن لكتابة برامج سعتها بحد أقصى 2kB.

وبدلاً من عمل دائرة علميه لإختبار دائرة الـ Microcontroller يمكنك إستعمال برنامج محاكاة يسمى Proteus simulator والذى يستخدم فى المقام الأول لمحاكاة الدوائر الـ Analog والـ Digital. ويمكن تحميل هذا البرنامج عن طريق موقع www.labcenter.co.uk.

2.1- تركيب ومواصفات البرنامج المكتوب فى الـ MikroC

Structure of a mikroC Program

الكود التالى يوضح أبسط برنامج مكتوب فى الـ MikroC compiler. ويستخدم هذا البرنامج لعمل Flash للـ LED أى لتشغيل وإطفاء هذا LED كل 1 ثانية والموصل على الطرف RB0 . ويلاحظ أن البرنامج يتكون من ثلاثة أشياء أساسية وهم:

1. التعليقات Comments.
2. قالب البرنامج والذى بداخل الـ void main().
3. بعض الأوامر الإضافية مثل أمر delay_ms.

```
/*
*************Flasher program *************
Written by:
Dr. Hesham Gad
Faculty of engineering
Mansoura university
*****************************************
*/

void main()
{

TRISB=0;                        //PORTB pins will be output

while(1)                        //Loop forever
        {
        PORTB.b0=0;             //Turn ON LED connected to RB0
        delay_ms(1000);         //Wait 1 second
        PORTB.b0=1;             //Turn OFF LED connected to RB0
        delay_ms(1000);         //Wait 1 second
        }
}
```

2.1.1- التعليقات Comments

تستخدم الـ Comments لعمل تعليقات على أجزاء البرنامج وذلك لكى يسهل قراءة البرنامج فيما بعد. وهذه التعليقات يتم تجاهلها عن طريق الـ Compiler أى انها لا تدخل فى تنفيذ أوامر البرنامج. ويوجد نوعين من الـ Comments :

1. الـ Single line comment: ويستخدم لعمل Comment على كل سطر ويبدأ بهذا الرمز // كما هو موضح،

```
LED=1;          //Turn ON the LED
```

2. Multiple line comment ويستخدم فى حالة وجود أكثر من سطر للتعليقات وتكون سطور التعليقات بين الرمزين /* */ كما هو موضح

```
/*
*************Flasher program *************
Written by:
Dr. Hesham Gad
Faculty of engineering
Mansoura university
*****************************************

*/
```

2.1.2- بداية ونهاية برنامج الميركروسى Beginning and ending of the program

بشكل عام فى لغة الـ C ، فإن بداية البرنامج تبدأ عن طريق كتابة الجملة الآتية:

```
void main()
```

وبعد ذلك، يتم تحديد بداية ونهاية البرنامج عن طريق الـ Brackets كما هو موضح بالآتى:

```
void main()
{
    يتم كتابة البرنامج هنا
}
```

ملحوظات مهمة:

1. من المهم أن تعرف أن لغة السى بالرغم من أنها Case sensitive أى أنها تفرق بين الحروف الـ Capital أو Small مثل:

   Motor, motor, Lamp, lamp, LAMP

   إلا أن برنامج الـ mikroC لا يهتم بالـ Case sensitivity أى أنه لا يتأثر بالكلمات من حيث كونها تحتوى على حروف Capital أو Small.

2. بما أن البرنامج مكتوب بلغة السى، فإن كل سطر يجب أن ينتهى بعلامة الـ semicolon أو ;
3. لا يجوز تسمية المتغيرات Variables بأسماء محجوزة Reserved فى الـ mikroC والجدول التالى يوضح الكلمات الممنوع إستخدامها كأسماء متغيرات.

asm	enum	signed
auto	extern	sizeof
break	float	static
case	for	struct
char	goto	switch
const	if	typedef
continue	int	union
default	long	unsigned
do	register	void
double	return	volatile
else	short	while

2.2- أنواع المتغيرات Variable types

مثل كل الـ C compilers يحتوى برنامج الـ mikroC على أنواع مختلفة من المخازن أو المتغيرات variables وكل نوع له قيمة قصوى Maximum value وقيمة صغرى Minimum value يمكن تخزينها كما هو موضح فى الجدول التالى.

Type	Size (bits)	Range
unsigned char	8	0 to 255
unsigned short int	8	0 to 255
unsigned int	16	0 to 65535
unsigned long int	32	0 to 4294967295
signed char	8	−128 to 127
signed short int	8	−128 to 127
signed int	16	−32768 to 32767
signed long int	32	−2147483648 to 2147483647
float	32	±1.17549435082E-38 to ±6.80564774407E38
double	32	±1.17549435082E-38 to ±6.80564774407E38
long double	32	±1.17549435082E-38 to ±6.80564774407E38

يُلاحظ فى الجدول أنه عندما يتم وضع كلمة signed أو unsigned أمام نوع المتغير (بإستثناء الـ float والـ double) فإنه يتم تحديد نوع المخزن من حيث كونه يقبل الأرقام الموجبة فقط أو الأرقام الموجبة والسالبة.

ملحوظة مهمة:

إذا لم يتم وضع unsigned او signed أما نوع الـ Variable فإن الـ mikroC سوف يفرض أن المتغير نوعه signed.

بالنسبة لمتغيرات الـ char فهذا النوع من النمتغيرات يقبل أرقام صحيحة (8 bits) تتراوح من 0 إلى 255 إذا كانت unsigned مثل:

```
unsigned char x, y, z;
x=150;
y=100;
z=255;
```

//or

```
char x=150, y=100, z=255;
```

أو أرقام موجبة وسالبة تتراوح من 128- إلى 127 مثل:

```
char x, y, z;
x=0;
y=-100;
z=-50;
```

أما لمخازن الـ int هى تقبل أرقام صحيحة (16 bits) تتراوح من 0 إلى 65535 إذا كانت unsigned مثل:

```
unsigned int x, y, z;
x=3542;
y=5005;
```

أو أرقام صحيحة موجبة وسالبة إذا كانت signed

```
int x=-12585, y=5005, z=-4520;
```

وإذا أردت أن تخزن أرقام صحيحة كبيرة (32 bits) فيمكنك إستعمال المتغيرات من نوع الـ Long int كالآتى:

```
long int x=125855241, y=-5002414, z=44544520;
```

بالنسبة لمخازن الـ float فهذا النوع من المخازن يقبل الأرقام الكسرية Floating numbers ولا يوضع أمامة signed أو unsigned نظراً لأنه يتعامل مع الأرقام الكسرية الموجبة والسالبة مباشرة مع العلم أن حجم المخزن 32 bit أى أنه يقبل أرقام كبيرة جداً كما موضح فى الجدول السابق. ويمكن تعريف المخزن كالآتى:

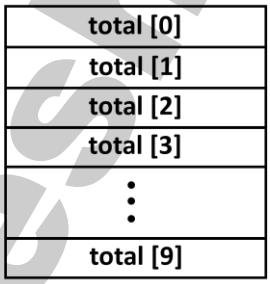
```
float a= -125.85, b=50.05, c= -4.52;
```

2.3- المصفوفات Arrays

الـ Arrays هى عبارة عن مجموعة من القيم المترابطة والتى يتم تخزينها فى الـ Memory تحت إسم واحد ونوع واحد Same type. فعلى سبيل المثال إذا أردنا تخزين 10 أرقام من نوع int بحيث يحملوا الأسم total، فإننا نكتب الآتى فى بداية البرنامج:

```
int total[10];
```

وهذا يعنى أنه تم حجز 10 اماكن فى الذاكرة بأسم total وبنوع int كما هو موضح فى شكل (1-2).



شكل (1-2): حجز Array فى الذاكرة بإسم total.

ولتخزين قيمة فى أى مخزن بهذا الـ Array فكل ما عليك فعله كتابة الآتى:

```
total[3]=120;
```

ويمكن تعريف الـ Array وتخزين القيم بداخله فى خطوه واحدة كالآتى:

```
int  total[ ]={125,251,487,100};
```

فى السطر السابق لم يتم تحديد عدد الأماكن المطلوبة حيث أن الـ Compiler بعمل بشكل آلى على تحديد عدد هذه اللأماكن وتخزين القيم بداخلها كما هو موضح فى شكل (2-2).

total [0]=125
total [1]=251
total [2]=487
total [3]=100

شكل (2-2): تعريف الـ Array والتخزين بداخلة فى خطوة واحدة.

ومن أكثر تطبيقات الـ Arrays هى إستخدامه كـ Character array. فعلى سبيل المثال، إذا أردنا إنشاء مخزن بإسم name حيث أننا نريد تخزين أسم الـ User فى هذا المخزن، فإننا نكتب الآتى:

```
char  name[ ]="Hesham";
```

وبذلك نكون أنشأنا array فى الذاكرة نوعه char وتم تخزين كلمة "Hesham" بداخلة كما هو موضح فى شكل (2-3). وهذا النوع من الـ Arrays مفيد جداً وخاصتاً عند عرض الرسائل Messages على شاشات الـ LCD أو عند إرسال أو أستقبال الرسائل فى الـ Serial communication.

name
'H'
'e'
's'
'h'
'a'
'm'

شكل (2-3): تعريف الـ Character array والتخزين بداخلة فى خطوة واحدة.

كما يمكن فتح مخزن بعدد معين من الحروف لكى يتم تخزين رسالة بداخلها فيما بعد كالآتى:

```
char  txt[15];
```

ويمكن إنشاء Two dimension array بنفس طريقة الـ One dimension كالآتى:

```
int  total [2][2]={ {2,3},{0,1} };
```

total

2	3
0	1

شكل (2-3): تعريف الـ Two dim. Array والتخزين بداخلة فى خطوة واحدة.

2.4- التعاملات فى لغة السى Operators in C

يتم تطبيق الـ Operators على الـ Variables وذلك قد يكون لغرض العمليات الحسابية Arithmetic operation أو للتحقق من شرط ما Condition. وبرنامج الـ mikroC يدعم العديد من الـ operators، ولكننا سوف تناول أهمها وهى:

1. الـ Arithmetic operators والتى تستخدم للعمليات الحسابية (مثل الجمع والطرح والقسمة والضرب). ويتم إجراء العمليات من اليسار إلى اليمين، كما أن الناتج النهائى يكون رقم Numerical result. والجدول التالى يبين أنواع العمليات الحسابية التى يمكن إجراءها عن طريق لغة الـ mikroC.

Operator	Operation
+	Addition
–	Subtraction
*	Multiplication
/	Division
%	Remainder (integer division)
++	Auto increment
––	Auto decrement

مثال على ذلك:

```
int x,y,z;
x=10;
y=6;
z=x+y;          //Result will be 16
```

2. الـ Relational operators والتى تستخدم لمقارنة المخازن أو المتغيرات (مثل أكبر من "<" أو أقل من ">"). ويستخدم هذا النوع من الـ Operators مع المعمليات الحسابية Arithmetic

operations والمنطقية Logical operations. وناتج هذا النوع من العمليات يكون True أو False فقط أو (1 و 0). وفيما يلى الجدول الخاص بالـ Relational operators.

Operator	Operation
= =	Equal to
!=	Not equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

مثال على ذلك:

```
int x,y;
x=10;
y=6;

x>y          //Returns 1 (True)
x<y          //Returns 0 (False)
x==y         //Returns 0 (False)
x==10        //Returns 1 (True)
```

3. الـ Logical operators وتستخدم مع الـ Relational للتحقق من أكثر Relation فى نفس الوقت (مثل a>b AND c<d) . والجدول التالى يبين أنواع الـ Logical operators.

Operator	Operation
&&	AND
||	OR
!	NOT

مثال على ذلك:

```
int x,y;
x=10;
y=6;
x>0 && x<20          //Returns 1 (True)
```

```
x<y || x>15          //Returns 0 (False)
x==y && y!=6         //Returns 0 (False)
```

4. الـ Bitwise operators وتستخدم لإجراء العمليات المنطقية Logical operationsعلى القيم أو Bits. والجدول التالى يبين أنواع العمليات المنطقية التى يمكن إجراءها على الـقيم.

Operator	Operation
&	Bitwise AND
|	Bitwise OR
^	Bitwise EXOR
~	Bitwise complement
<<	Shift left
>>	Shift right

مثال على ذلك:

```
x=0xFA;          //x=1111 1010
y=0xEE;          //y=1110 1110

z=x & y;         //z=1110 1010          (AND)

z=x | y;         //z=1111 1110          (OR)

z=~x;            //z=0000 0101          (NOT)

z=x>>2           //z=0011 1110          (Shift x to the right two times)

z=x<<3           //z=1101 000       (Shift x to the left three times)
```

2.5- التحكم فى سير البرنامج Program flow control

فى أى لغة برمجة، يحتاج المبرمج إلى مجموعة من الأوامر Instructions والجمل statements للتحكم بسير البرنامج. وهذه الجمل تشمل الآتى:

1. جمل الأختيار Selection statements (تستخدم لإختيار تنفيذ أكواد محددة طبقاً لشروط معينة) وتشمل:
    1. *if-else* statement.
    2. *Switch-case* statement.
2. جمل التكرار Iteration statements (تستخدم لإختيار تنفيذ أكواد معين عدد من المرات طبقاً لشرط محدد) وتشمل:
    1. *for* statement.
    2. *while* statement.
    3. *do* statement.
    4. *goto* statement.

وفيما يلى شرح كل جملة بشكل مبسط وسريع.

2.5.1- جمل الإختيار Selection statements

بالنسبة للـ IF-ELSE يوجد منها ثلاثة أشكال وهم:

Syntax 1:

```
If ( الشرط)
{
    .....
    الكود الذى سوف يتم تنفيذه فى حالة تحقق الشرط
    ......
}
```

Example:

```
if (temperature>25.0)          //إذا كنت قيمة درجة الحرارة أكبر من 25
{
   Led=1;                      //يتم تشغيل الليد والمروحة
   fan=1;
}
```

Syntax 2:

```
if ( الشرط)
{
   …..
   الكود الذى سوف يتم تنفيذه فى حالة تحقق الشرط
   .....
}
else
{
   …..
   الكود الذى سيتم تنفيذه فى حالة عدم تحقق الشرط
   .....
}
```

Example:

```
if (temperature>25.0)          //إذا كنت قيمة درجة الحرارة أكبر من 25
{
   Led=1;                      //يتم تشغيل الليد والمروحة
   fan=1;
}
else                           //إذا لم يتحقق الشرط السابق
{
   led=0;                      //يتم إطفاء الليد والمروحة
   fan=0;
}
```

Syntax 3:

```
if (الشرط الأول)
{
    .....
    الكود الذى سيتم تنفيذه فى حلة تحقق الشرط الأول
    .....
}
else if (الشرط الثانى)
{
    .....
  إذا لم يتحقق الشرط الأول سوف يتم التحقق من الشرط الثانى وإذا تحقق الشرط الثانى سيتم
                        تنفيذ الكود الذى يوجد هنا
    .....
}
else
{
    ……
  إذا لم يتحقق أى شرط من الشروط السابقة سوف يتم التحقق سيتم تنفيذ الكود الذى يوجد هنا
    .....
}
```

Example:

```
if (temperature>25.0)          إذا كنت قيمة درجة الحرارة أكبر من 25//
{
    Led=1;                     يتم تشغيل الليد والمروحة//
    fan=1;
}
elseif (temperature =25.0)     إذا لم يتحقق الشرط السابق يتم التحقق من هذا الشرط//
{
    led=1;                     يتم تشغيل الليد فقط//
    fan=0;
 }
else                           إذا لم يتحقق أياً من الشروط السابقة//
{
    led=0;                     لا يتم تشغيل أى شئ//
    fan=0;
 }
```

بالنسبة للـ Switch-case statement، فإنه يستخدم لتنفيذ كود محدد من مجموعة أكواد بناءً على قيمة محددة لمخزن ما وله شكل واحد كالآتى:

Syntax:

```
switch (المخزن الذى نريد التحقق من قيمته)
{
   case القيمة الأولى:
                .....
                الكود الذى سيتم تنفيذه فى حالة تحقق القيمة الأول
                break;

   case القيمة الثانية:
                .....
                الكود الذى سيتم تنفيذه فى حالة تحقق القيمة الثانية
                break;

   ……................

   default:
                .....
                الكود الذى سيتم تنفيذه فى حالة تحقق أياً من القيم السابقة
}
```

Example:

```
switch (cnt)                    سيتم التحقق من مخزن عداد//
{
   case 10:                     فى حالة وصول العداد إلى 10//
                motor=1;        يتم تشغيل الماتور//
                break;
   case 20:                     فى حالة وصول العداد إلى 20//
                fan=1;          يتم تشغيل المروحة//
                break;
   case 30:                     فى حالة وصول العداد إلى 30 //
                heater=1;       يتم تشغيل السخان//
                break;
```

```
    default:
                fan=0;        //إذا لم يتحقق أى شئ من السابق لا يتم تشغيل أى شئ
                motor=0;
                heater=0;
}
```

2.5.2- جمل التكرار Iteration statements

تستخدم جمل الـ Iteration لعمل Loops فى البرنامج، حيث أنه هذه الـ Loops تستخدم لكى يتم تنفيذ كود ما عدد معين من المرات تبعاً لشرط ما. ويوجد فى لغة الـ C أربعة طرق لعمل الـ Loops وهم:

1. *for* statement.
2. *while* statement.
3. *do-while* statement.

بالنسبة للـ *for* statement فإنه شكل الكود الخاص به كالآتى:

Syntax:

```
for (معدل الزيادة أو النقصان ; شرط الإستمرارية ; القيمة الأبتدائية)
{
      .....
      for الكود الذى نريد تنفيذه عدد من المرات طبقاً للشرط الذى بداخل الـ
}
```

Example:

```
for (i=1;i<=10;i++)        // من قيمة 1 إلى 10 وبمعدل زيادة 1 فى كل مرة
{
                           // سيتم تنفيذ الكود الموجود هنا 10 مرات
}
```

Example:

```
for (i=0;i<=10;i++)          // من قيمة 0 إلى 10 وبمعدل زيادة 1 فى كل مرة
{
                             // سيتم تنفيذ الكود الموجود هنا 11 مرة
}
```

Example:

```
for (;;)          // لم يتم تحديد البداية والنهاية
{
                  //سيتم تنفيذ الكود هنا عدد لا نهائى من المرات
}
```

بالنسبة للـ *while* statement فإن الفرق بينه وبين الـ *for* statement أنه لا توجد به قيمة إبتدائية ولا معدل زيادة، أى أن يظل فى حالة الـ Loop طالما أن شرط الإستمرارية مازال فعال أو كان True. وبذلك فإن شكل الكود الخاص به سوف يكون كالآتى:

Syntax:

```
while (شرط الإستمرارية)
{
      .....
   while الكود الذى نريد تنفيذه عدد من المرات طبقاً للشرط الذى بداخل الـ
}
```

Example:

```
i=0;                  // القيمة الأبتدائية
while (i<10)          //طالما أن قيمة المخزن لم تصل إلى 10 فإن الكود سيستمر بالعمل
{
   i=i+1;             // سيتم تنفيذ زيادة المخزن بمعدل 1 فى كل لفة
   .....              //الكود الموجود هنا سيتم تنفيذه 10 مرات
}
```

كما يمكن عمل Infinite loop عن طريق وضع TRUE أو 1 كالأتى:

```
while (1)
{
    .....                    //الكود الموجود هنا سيتم تنفيذه عدد لا نهائى من المرات
}
```

وأفضل تطبيق للـ *while* statement هى الإنتظار لحدوث حدث ما. فعلى سبيل المثال، إذا أردنا إنتظار الضغط على Button، فإننا نكتب الآتى:

```
while (button==0);          //إنتظر هنا حتى تصبح قيمة الطرف تساوى 1
```

حيث أنه سيظل البرنامج عالقاً فى هذا السطر طالما أن قيمة المفتاح تساوى الصفر إلى أن يتم الضغط على الـ Button وبذلك سينتقل البرنامج بشكل آلى إلى السطر الذى يليه.

بالنسبة للـ *do-while* statement فإن مشابه للـ *while* statement ولكن بإستثناء أن شرط الإستمرارية يتم التحقق منه فى نهاية الـ Loop وليس فى بدايته كالآتى:

Syntax:

```
do
{
  الكود الذى نريد تنفيذه عدد من المرات طبقاً للشرط الذى بداخل الـ while
}
while (شرط الإستمرارية)
```

يُلاحظ أن الشرط فى نهاية الكود وليس فى بدايته، وهذا يعنى أن الكود سيتم تنفيذه أول مره سواء أكان الشرط True أو False.

Example:

```
i=0;                          // القيمة الأبتدائية

do
{
      i=i+1;                  // سيتم تنفيذ زيادة المخزن بمعدل 1 فى كل لفة
      …..                     // الكود الموجود هنا سيتم تنفيذه 11 مرات
}
while (i<10)                  // طالما أن قيمة المخزن لم تصل إلى 10 فإن الكود سيستمر بالعمل
```

كما يمكن عمل Infinite loop عن طريق عدم وضع TRUE أو 1 كالأتى:

```
do
{
      …..                     // الكود الموجود هنا سيتم تنفيذه عدد لا نهائى من المرات
}
while (1)
```