

بسم الله الرحمن الرحيم
الفريق العربي للبرمجة



برمجة الميكروكنترولار بواسطة لغة ال C

المحتويات:

1.1 مقدمة

CISC and RISC 1.1.1

2.1 دوائر أعداد ال PIC

Power Supply Circuit 2.1.1

External reset circuits 2.1.2

External Oscillator 2.1.3

Crystal Operation 2.1.4

3.1 ال PIC 16F84

3.2 تنظيم الذاكرة

3.2.1 ذاكرة البرنامج

3.2.2 ذاكرة البيانات

3.2.3 ال Registers

4.1 المخارج و المداخل

PORTA and TRISA Registers 4.1.1

PORTB and TRISB Registers 4.1.2

5.1 برمجة ال PIC بواسطة ال C

PICC Lite Compiler 5.1.1

PICC LITE VARIABLE TYPES 5.1.2

5.1.3 المصفوفات

ASCII CONSTANTS 5.1.4

ARITHMETIC AND LOGIC OPERATORS 5.1.5

PROGRAM FLOW CONTROL 5.1.6

Built-in Functions 5.1.7

ACCESSING THE EEPROM MEMORY 5.1.8

DELAYS IN C PROGRAMS 5.1.9

PIC MICROCONTROLLER INPUT-OUTPUT INTERFACE 6.1

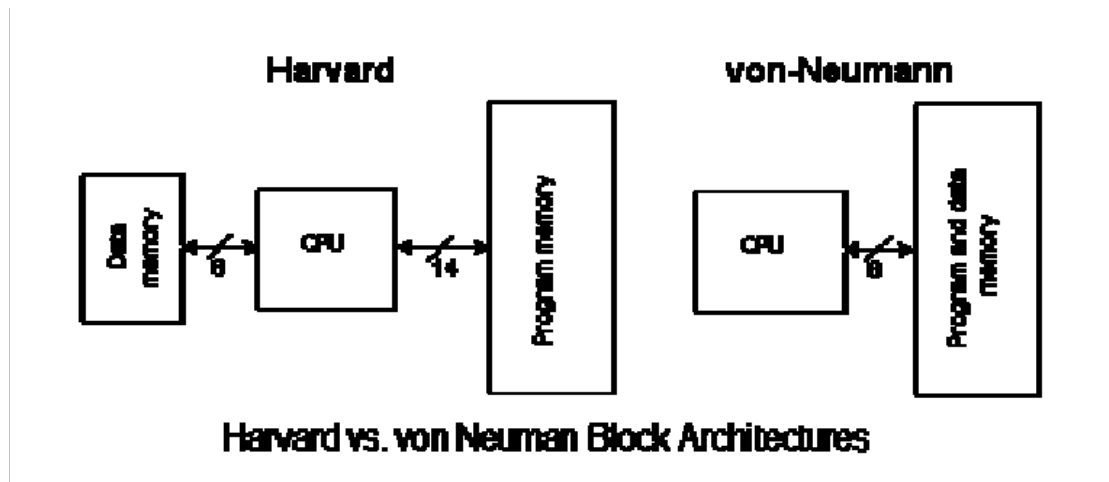
HARDWARE AND SOFTWARE REQUIREMENTS 7.1

8.1 تجارب عملية

1.1 مقدمة

CISC and RISC 1.1.1

يوجد نوعين من بنية الميكروكنترولار هما Harvard و von-Neumann . و نوع ال Harvard هو الاحدث في بنية الميكروكنترولار وفي هذا النوع يكون ال Data Bus و ال Address Bus منفصلين اي المسار الذي تسري فيه البيانات منفصل عن مسار العناوين. وهذا يزيد من سرعة الميكروكنترولار. وتسمى الميكروكنترولار المبنية بال Harvard ب RISC microcontrollers حيث ان كلمة RISC تعني *Reduced Instruction Set Computer* اما الميكروكنترولار المبنية بنظام ال Von-Neumann تسمى CISC microcontrollers حيث ان كلمة CISC تعني *Complex Instruction Set Computer*



و ال PIC نوع 16f84 مبنية بنظام ال RISC microcontroller هذا يعني انها تحتوي علي عدد قليل من التعليمات حوالي 35 تعليمة بينما هناك ميكروكنترولار بواسطة شركات مثل Intel و موتورولا تحتوي علي اكثر من مئة تعليمة. وكل هذه التعليمات يتم تنفيذها في Cycle واحدة.

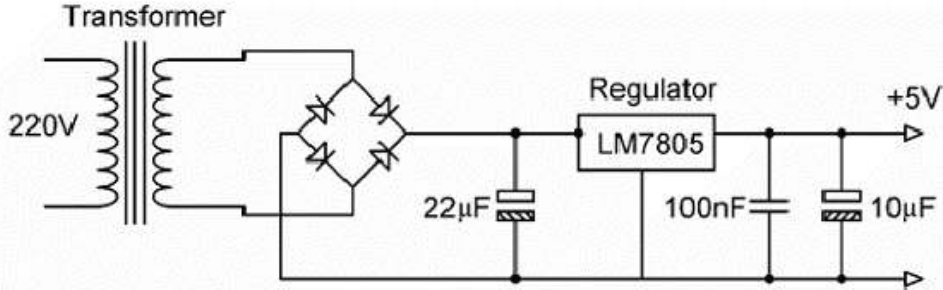
2.1 دوائر أعداد ال PIC

كل دائرة تحتوي علي PIC يجب ان يوجد معها دوائر اخري مساعدة لكي تعمل ال PIC ويختلف عدد لدوائر المساعدة و مدي تعقيدها مع اختلاف ال PIC المستخدمة. وبشكل عام تحتاج ال PIC الي هذه الدوائر :

- دائرة مصدر الجهد A power supply
- Reset circuit
- Oscillator (دائرة الذبذبات)

Power Supply Circuit 2.1.1

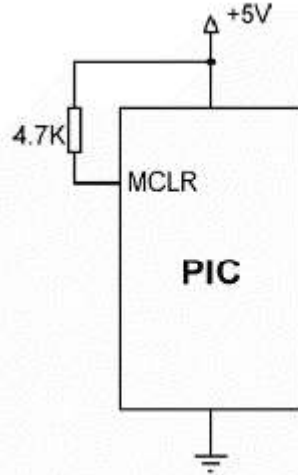
ال power supply يكون عادة 5 فولت وتتكون دائرة كما بالصورة :



تتكون من transformer و rectifier circuit و power regulator وهو لتنظيم الجهد و ضمان دخول جهد ثابت علي ال PIC حتي لا تتلف ويمكن استخدام نوع منة وهو LM78L05 ومتوافر بكثرة.

External reset circuits 2.1.2

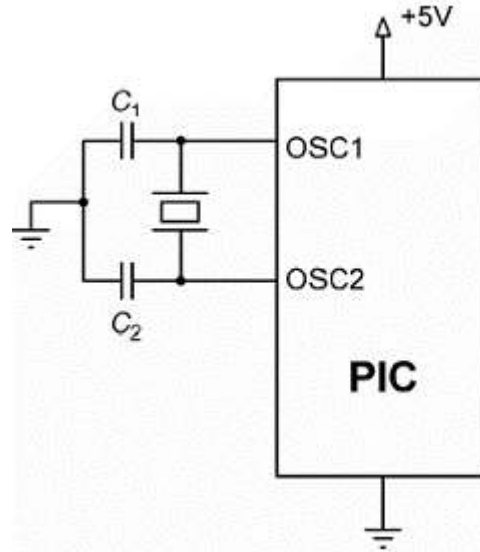
علي الرغم من ان الميكروكنترولار تحتوي علي نظام Reset داخلي لكن من المفيد في الكثير من التطبيقات ان نضع دائرة Reset خارجية و معني ال Reset هو اعادة الميكروكنترولار الي نقطة البداية اي تصفير المسجلات و البدء في كود البرنامج من البداية (اي اعادة تشغيل). وتتكون دائرة ال Reset كما بالصورة :



وتتم بواسطة توصيل مقاومة 4.7 كيلو أوم مع ال master clear ويرمز لة ب MCLR في الميكرو وسيتم شرحه لاحقا مع ال power supply وفي الكثير من التطبيقات يتم توصيل زر بدائرة ال Reset ليحدث Reset للميكرو عند الضغط علي هذا الزر.

External Oscillator 2.1.3

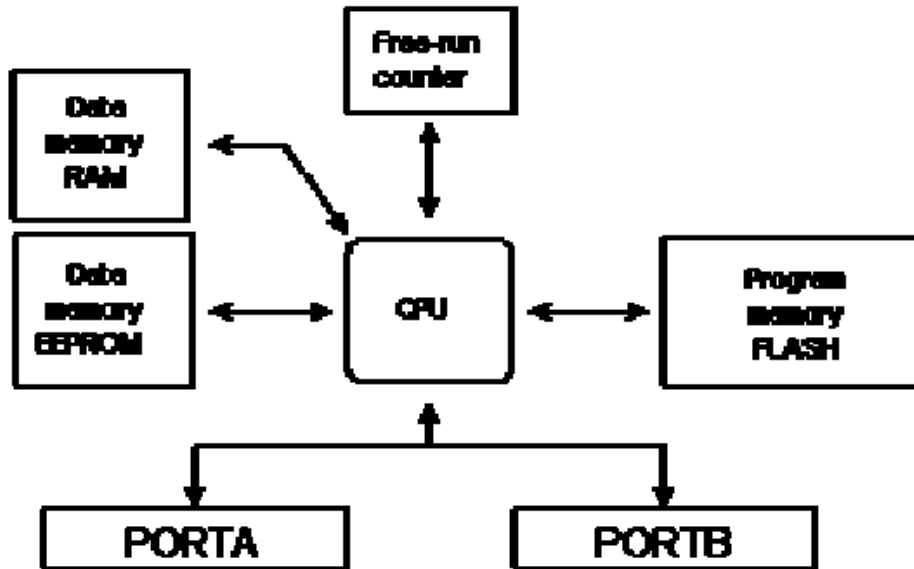
أيضا تحتوي الميكروكنترولار علي دائرة clock oscillator وهي التي تمد الميكرو ب CLK خلال زمن معين. لكن يجب اضافة بعض المكونات الخارجية الي الميكرو لكي يعمل ال clock oscillator بشكل صحيح لكن أيضا هناك بعض الانواع من الميكرو تحتوي علي هذه المكونات داخلها. ويمكن عمل دائرة ال clock oscillator كما بالصورة :



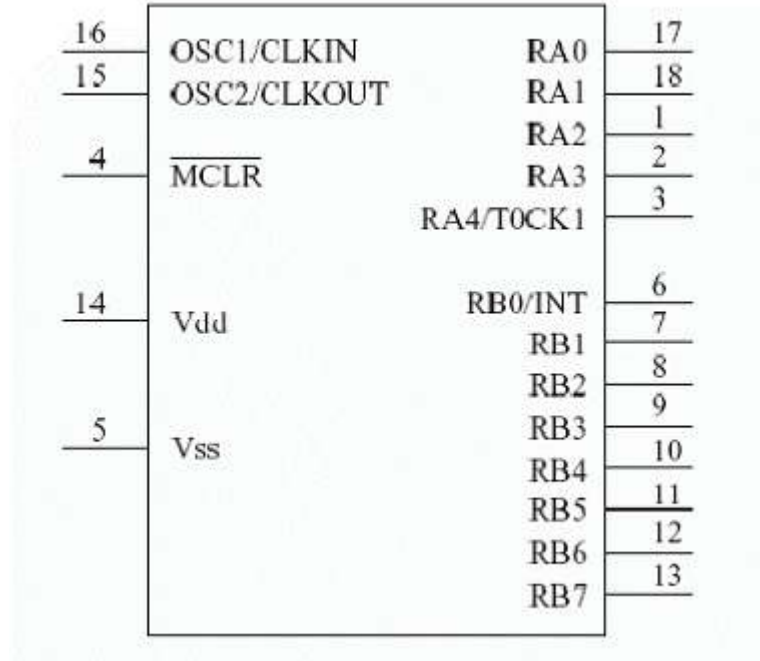
ويقوم ال external oscillator بتوليد pulses square wave في التردد المطلوب. وكما موضح بالصورة فالدائرة تتكون من crystal يتم توصيلها مع ال OSC1 و ال OSC2 مع زوجين من المكثفات.

PIC 16F84 ال 3.1

هذا النوع من ال PIC يتبع عائلة ال 8-bit ميكروكنترولار RISC architecture ويمكن وصف بنيتها الداخلية كما بالصورة :



- وهذا النوع من ال PIC من اشهر الانواع و اكثرها استخداما. ومن خصائصه :
- تحتوي علي PIN 18 التي تستطيع العمل حتي 20 ميغا هرتز clock speed
 - تحتوي علي فلاش ميموري 14×1024
 - 68 بيت حجم ال RAM data memory
 - 64 بيت حجم ال EEPROM nonvolatile data memory
 - 13 I/O pins



وظائف ال PINS :

- من RB0 الي RB7 خاص ب pins B ports
- من RA0 الي RA4 خاص ب pins A ports
- Vdd Supply voltage يتم توصيلة بال Power Supply
- Vss Ground يتم توصيلة بالارض
- OSC1 Crystal الخاص بدائرة ال Oscillator
- OSC2 Crystal الخاص بدائرة ال Oscillator
- MCLR يتم توصيلة بدائرة ال Rest
- INT External interrupt وهو مشترك مع RB0
- T0CK1 Optional timer clock مشترك مع RA3

3.2.1 تنظيم الذاكرة

يمكن تقسيم الذاكرة الي جذنين وهما :

- ذاكرة البرنامج
- وذاكرة البيانات DATA

3.2.1 ذاكرة البرنامج

ذاكرة البرنامج يتم وضعها في ال FLASH Memory واقصد بالبرنامج اي البرنامج الذي نقوم ببرمجته ووضعة في ال PIC لكي تؤدي الوظيفة التي نريد. ومعني اننا نضع البرنامج في ال Flash Memory اي اننا نستطيع برمجة ال PIC اكثر من مرة.

3.2.2 ذاكرة البيانات

تنقسم ذاكرة ال DATA الي قسمين :

- EEPROM memory
- RAM memory

EEPROM memory تتكون من 64 bit ومعني انها EEPROM اي غير متطايرة اي لا نفقد البيانات المخزنة فيها مع فقد التيار الكهربائي. و تستخدم في تخزين البيانات الهامة مثلا (درجة الحرارة المطلوبة في منظم درجة الحرارة).

RAM memory تتكون من 68 Bit وتبدأ من العنوانين 0x0c الي 0x4F وتسمى هذه العنوانين بـ GPR registers حيث ان GPR تعني General Purpose Registers

3.2.3 Registers ال

ال DATA RAM ايضا تسمى بـ register file map وتكتب اختصارا RFM ويتكون من 80 bytes وينقسم ال RFM الي

- The special function registers ويكتب اختصارا SFR
- The general purpose registers ويكتب اختصارا GPR

أيضا يمكن تقسيم ال RFM الي two banks

- Bank 0
- Bank 1

و البنك في المسجل يجب اختياره قبل البدء في القراءة او الكتابة من اي مسجل.

00	Indirect address	Indirect address	80
01	TMR0	OPTION	81
02	PCL	PCL	82
03	STATUS	STATUS	83
04	FSR	FSR	84
05	PORTA	TRISA	85
06	PORTB	TRISB	86
07	—	—	87
08	EEDATA	EECON1	88
09	EEADR	EECON2	89
0A	PCLATH	PCLATH	8A
0B	INTCON	INTCON	8B
0C	68-byte GPR	Access in bank 0 (mapped)	8C
...			...
...			...
...			...
4F			CF
50	50 to 7F not used		D0
...			...
...			...
7F			FF
Bank 0		Bank 1	

مسجلات الحالة (STATUS REGISTER) هي تحتوي علي الحالة الرياضية لل ALU

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
bit 7				bit 0			

– Bit 6-7 يتم وضعه بـ 0

- Bit 5 هي ال RP0 اي Register Bank Select bits
- Bit 4 هي TO اي Time out bit وهي تساوي 1 بعد بداية التشغيل و 0 عندما يحدث ال time out
- Bit 3 هي PD اي Power down bit
- Bit 2 هي Bit 2 اي Zero bit ويساوي 1 عندما يكون ناتج عملية حسابية او منطقية يساوي 0 و يساوي 1 عندما يكون ناتج العملية الرياضية او المنطقية لا يساوي الصفر.
- Bit 1 هي DC اي Digit carry/borrow bit
- Bit 0 هي C اي Carry/borrow bit

4.1 المخارج و المداخل

كلمة منفذ Port تشير الي مجموعة من ال PINS في الميكروكنترولار التي يمكن استعمالها بشكل متزامن في نفس الوقت. ويتم تصميم هذه ال pins ك مدخل او مخرج Input or output حسب النظام الذي ستستعمل فيه. ولكي نقوم بتعريف منفذ معين علي انه سيكون مخرج او مدخل يتم عن طريق كتابة مزيج من الاعداد والوحدات في ال TRIS register يعني لو وضعنا 1 في بيت المطلوبة في ال TRIS هذا يعني انها مدخل Input Pin ولو وضعنا صفر هذا يعني انها اصبحت مخرج. وهذا يعني ان كل منفذ له TRIS Register خاص به يتم من خلاله تحديد اذا كان هذا المنفذ سيكون دخل ام خرج.

مثلا المنفذ PortA له TRISA و المنفذ portB له TRISB

PORTA and TRISA Registers 4.1.1

- المنفذ PORTA عبارة عن 5 bit وكما قلنا يمكن التحكم فيه عن طريق تغيير القيم في مسجلات ال TRISA :
- عند وضع ال TRISA ب 1 فإن ذلك سيجعل بيت معين من ال PORTA كمدخل.
 - عند وضع ال TRISA ب 0 فذلك سيجعل ال PIN المختارة في ال PORTA مخرج.
 - قراءة المنفذ PORTA يعني قراءة لحالة ال PINS اما 1 او 0

PORTB and TRISB Registers 4.1.2

طول المنفذ PORTB يساوي 8 BIT اي ان هذا المنفذ يتكون من 8 PINS ويمكن التحكم فيه عن طريق تغيير القيم في المسجلات TRISB :

- عند وضع ال TRISB ب 1 فإن ال PIN المختارة من ال PORTB ستكون مدخل
- عند وضع ال TRISB ب 0 فإن ال PIN المختارة من ال PORTB ستكون مخرج

5.1 برمجة ال PIC بواسطة ال C

بشكل عاك يتم برمجة المايكروكنترولار بواسطة ال اسمبلي. وعلى الرغم من ان ال اسمبلي تعتبر اسرع بكثير الا انة لها بعض المساوئ. مثلا ال اسمبلي لغة معقدة جدا وصعبة الفهم علاوة على انها تأخذ الكثير من الوقت خاصة في المشاريع الكبيرة. ويمكن ايضا برمجة المايكروكنترولار بلغات عالية المستوى مثل ال C و ال VB والباسكال.

PICC Lite Compiler 5.1.1

الان سوف نتعرض لمبادئ برمجة المايكروكنترولار بواسطة لغة ال C وكما هو معروف فان لغة السي من اكثر اللغات استخداما هذه الايام و يوجد الكثير من مترجمات السي الخاصة ببرمجة المايكروكنترولار. وسوف نلقي نظرة على واحد من اشهر هذه المترجمات Compiler وهو ال PICC Lite C compiler من شركة

Hi - Tech

ويستخدم ال PICC في برمجة الكثير من عائلات المايكرو مثل ال PIC16F84 و PIC16F877 و PIC16F629 و PIC16F627 .

وال PICC مزود بمحرر editor متكامل مع كاشف للاخطاء syntax highlighting الذي يسهل عملة البرمجة بشكل كبير. ايضا مزود بالكثير من المكتبات التي يمكن للمبرمج استخدامها. أيضا من اسباب استخدام ال PICC :

- دعم حسابات ال floating point
- امكانية حساب الارقام الكبيرة
- دعم ال LCD displays
- سهل الحصول على

PICC LITE VARIABLE TYPES 5.1.2

ال PICC Compiler يدعم ال data types الاساسية مثل :

- bit
- unsigned char
- signed char
- unsigned int
- signed int
- long
- unsigned long
- float
- double

bit

ال bit تستخدم لتخزين قيمة من نوع Boolean اي (0 او 1) لذلك تستخدم عادة flags في البرنامج في المثال التالي المتغير answer سيأخذ القيمة 0 او القيمة 1

```
bit answer;
```

Unsigned Char

سعة 8 بت ويستطيع تخزين حرف ASCII او عدد صحيح integer في المجال من 0 الي 255 ايضا ال '11111111' bit pattern في المثال التالي المتغير count و initial معرفين من نوع unsigned char و المتغير count يحوي عدد عشري 120 و المتغير initial يحتوي على حرف ال T

```
unsigned char count, initial;
count = 120 ;
initial = 'T';
```

Signed Char

من الممكن ان نقول char فقط ويستخدم لتخزين رقم عشري في المجال من -127 الي +127 وفي المثال التالي المتغير first يحمل القيمة 180- والمتغير count يحمل القيمة الموجبة 25

```
signed char first, count;  
first = -180;  
count = 25;
```

Unsigned Int

بسعة 16 بيت وتستخدم لتخزين الارقام بالنظام العشري decimal numbers في المجال من 0 الي +65535 وفي المثال التالي المتغير result سيحمل القيمة 28512

```
unsigned int result;  
result = 28512;
```

Signed Int

سعة 16 بيت وتستخدم في تخزين الارقام signed numbers في المجال من -32768 الي +32767 وفي المثال التالي المتغير count معرف علي انه signed int وأخذ القيمة السالبة -25000

```
signed int count;  
count = -25000;
```

Long

سعة 32 بيت ويستخدم في تخزين الاعداد الكبيرة large signed integer numbers ومجاله من -2147483648 الي +2147483647 وفي المثال التالي المتغير Sum يخزن الرقم 45000

```
long sum;  
sum = 45000;
```

Unsigned Long

سعة 32 بيت ويستخدم في تخزين الاعداد الصحيحة الكبيرة الموجبة unsigned integer Numbers والمجال يكون من 0 الي 4294967295 وفي المثال التالي الرقم الكبير 3200000 سوف يتم تخزينه في المتغير cnt

```
unsigned long cnt;  
cnt = 3200000;
```

Float

سعة 24 بيت ويستخدم لتخزين الارقام الكسرية fractional numbers

```
float temp;  
temp = 2.35;
```

5.1.3 المصفوفات

المصفوفة هي مجموعة من المتغيرات من نفس النوع وعلى سبيل المثال المصفوفة result مصفوفة مصفوفة اعداد صحيحة وتحتوي على 10 عناصر او خانات ويتم تعريفها كما يلي :

```
int results[10];
```

وكل عنصر في المصفوفة يكون له Index وتبدأ المصفوفة من العنصر 0 وبالتالي العنصر الاخير يكون له ال index رقم 9 وليس 10 ويتم التعامل مع اي عنصر في المصفوفة بادخال رقم ال Index الخاص به يعني مثلا العنصر الاول من المصفوفة result هو result[0] والعنصر الثاني هو result[1] وهكذا... ويمكن اسناد اي قيمة لاي عنصر في المصفوفة طالما هذه القيمة من نفس نوع المصفوفة يعني مثلا ممكن اسناد العدد الصحيح 25 الي ثاني عنصر من عناصر المصفوفة السابقة كالتالي

```
results[1] = 25;
```

ويمكن ادخال قيم في جميع عناصر المصفوفة كما يلي :

```
int results[] = {1, 3, 5, 7, 9, 11, 13, 15, 17, 19};
```

وعناصر اي مصفوفة من الممكن ان تخزن في ال RAM Memory للميكروكنترولار او في ال Program Memory . اذا كانت عناصر المصفوفة لا تتغير ابدا اثناء سير البرنامج فيتم تخزينهم في ال Program memory ويتم تخزين عناصر المصفوفة في ال Program memory فقط اذا تم حماية المصفوفة وذلك بتعريفها ك Const وفي المثال التالي مصفوفة لن تتغير ابدا اثناء سير البرنامج لذلك سنقوم بوضعها في ال Program memory

```
const int results[] = {1, 3, 5, 7, 9, 11, 13, 15, 17, 19};
```

ايضا المصفوفة يمكن ان نضع فيها حروف كالتالي :

```
unsigned char device[] = {'C', 'O', 'M', 'P', 'U', 'T', 'E', 'R'};
```

ومن الممكن وضع كلمات وجمل Strings

```
unsigned char device[] = 'COMPUTER';
```

ومن الممكن ايضا ان تكون المصفوفة لها اكثر من بعد فالمصفوفة التي تعاملنا معها في المثال السابق هي مصفوفة احادية البعد ولتعريف مصفوفة ثنائية البعد نقوم بتعريف عدد الصفوف ثم عدد الاعمدة كالتالي

```
int temp[3][5];
```

يعني المصفوفة temp تحتوي على 3 صفوف و 5 اعمدة و 15 عنصر

ASCII CONSTANTS 5.1.4

\n newline character
 \t tab character
 \\ backslash character
 \' single quote character
 \0 null character

من الممكن اسناد القيمة Null الي المتغير temp كالتالي

```
temp = '\0';
```

5.1.5 العمليات الحسابية و المنطقية

يدعم كومبيلر ال PICC عدد كبير من العمليات الحسابية و المنطقية واكثر هذه العمليات استخداما الموجودة بالاسفل:

- ()[] parenthesis
- ! logical NOT
- □ bit complement
- + - * / arithmetic add, subtract, multiply, divide
- % arithmetic modulo (remainder from integer divisi
- ++ increment by 1
- -- decrement by 1
- & address of
- << shift left
- >> shift right
- ≥ greater than or equal
- greater than
- ≤ less than or equal
- < less than
- sizeof size of a variable (number of bytes)
- == logical equals
- != logical not equals
- || logical OR
- && logical AND
- += add and assign
- -= subtract and assign
- /= divide and assign
- |= logical OR and assign
- ^= logical NOT and assign
- &= logical AND and assign
- >>= shift right and assign
- <<= shift left and assign

5.1.6 التحكم في سير البرنامج

طبعا يتم التحكم في سير البرنامج باستخدام الجمل الشرطية المعروفة

- if-else
- for
- while
- do
- goto
- break
- continue
- switch-case

Built-in Functions 5.1.7

يقدم كمبيلر ال PICC الكثير من الدوال المدمجة ومن الممكن التعرف علي هذه الدوال من خلال PICC Lite User's Guide ومن امثلة هذه الدوال :

- **Abs** لحساب الحد المطلق
- **Cos** دالة cos المعروفة في علم حساب المثلثات
- **Sqrt** لحساب الجذر التربيعي
- **Isupper** هذه الدالة تختبر اذا كان الدخل حروف كبيرة Upper case ام لا فاذا كان حروف كبيرة ترجع 1 واذا كان لا ترجع 0
- **Isalnum** هذه الدالة تختبر اذا كان الدخل alphanumeric ام لا اي ارقام من 0 الي 9 او حروف صغيرة و كبيرة
- **Strlen** دالة هامة وتستخدم لحساب طول سلسلة نصية
- **Strcpy** هذه الدالة تنسخ جمل string الي مصفوفة نصية character array

5.1.8 التعامل مع ذاكرة ال EEPROM

تحتوي بعض انواع الميكروكنترولار علي ذاكرة من نوع EEPROM مثل ال Pic16F84A وتستخدم هذه الذاكرة لحفظ القيم الثابتة nonvolatile data اي عند عمل Reset للبرنامج تظل هذه القيم ثابتة و موجودة. ويستخدم الامر EEPROM WRIE لكتابة اي بايت لل EEPROM Memory والمثال التالي يقوم بكتابة القيمة الهكسية x2E0 للعنوان 2 في ال EEPROM

```
EEPROM WRITE(2,0x2E);
```

وبالمثل يمكننا قراءة اي بايت من ال EEPROM بواسطة الامر EEPROM READ والمثال التالي يقوم بقراءة بايت من العنوان 5 في ال EEPROM ويضعة في المتغير K

```
k = EEPROM READ(5);
```

DELAYS IN C PROGRAMS 5.1.9

في كثير من التطبيقات نحتاج الي تقليل سرعة الميكروكنترولار لكي نلاحظ شئ ما مثل اضاءة LED ويقدم لنا كومبيلر ال PIC C نوعين من دوال ال DELAY وهما microsecond delay function و millisecond delay function وواضح عمل كل دالة من اسمها ويجب تضمين هذا الملف <delay.c> في المشروع حتي نتمكن من استدعاء اي من الدالتين. وتستخدم الدالة DelayUs(x) لعمل ال microsecond delay في المجال من 1 الي 255 ميكرو ثانية مثال:

```
DelayUs(200);
```

وبالمثل الدالة DelayMs(x) لعمل millisecond delay في المجال من 1 الي 255 مللي ثانية مثال :

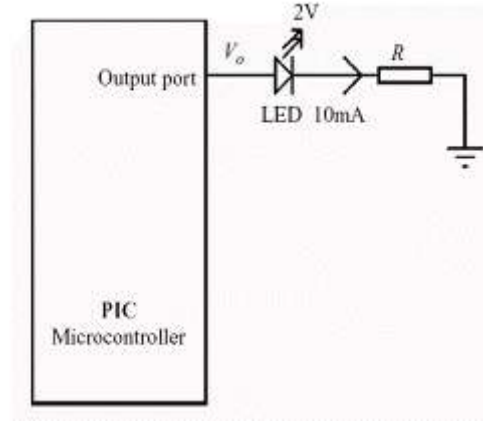
```
DelayMs(10);
```

ومن الممكن عمل delay اطول من السابق وذلك بادخال احد الدوال السابقة في حلقة LOOP وجدير بالذكر ان الدالتين السابقتين يتعاملون مع الميكروكنترولار ذات ال 4MHz clock frequency لذلك اذا كان ال CLK Rate مختلف فيجب علينا تعريف ال CLK Rate الخاص بالمبرمجة التي سنعمل عليها مثلا لو سنعمل مع 2 MHZ سنقوم بكتابة التالي في بداية البرنامج

```
#undef XTAL_FREQ  
#define XTAL_FREQ 2MHZ
```

6.1 مداخل و مخارج الميكروكنترولار

تتصل الميكروكنترولار بالعالم الخارجي عن طريق منافذ الدخل و الخرج كما ذكرنا سابقا. ومنافذ الخرج للميكروكنترولار يمكنها اخراج 25 ميلي أمبير من التيار. وعندما يقوم اي خرج باخراج تيار فان هذا التيار يتدفق من PINs الخاصة بالمنفذ. والمعدات البسيطة مثل ال LED و ال LCD يمكن توصيلها مع منافذ الخرج ذات ال 25 ميلي امبير اما المعدات التي تحتاج تيار اعلي من ذلك يمكن توصيلها مع منافذ الميكروكنترولار عن طريق ترانزيستور لزيادة التيار الخارج من المنفذ. مثال توصيل LED بميكروكنترولار:



معظم ال LEDs تعمل مع جهد يصل الي 2 فولت و تيار 10 ميلي امبير لذلك عند توصيل ال LED بمنفذ الميكرو كنترولار يجب وضع مقاومة علي التوالي لتقليل التيار كما بالصورة السابقة. ولو فرضنا ان جهد الخرج عند المنفذ 5 فولت فيمكن حساب هذه المقاومة بالقانون التالي

$$R = \frac{V_o - 2}{10 \text{ mA}} = \frac{5V - 2V}{10 \text{ mA}} \approx 330 \Omega.$$

7.1 متطلبات مشاريع الميكروكنترولار من الهاردوير والسوفت وير

بخصوص الهارد وير فيعتمد علي مدي كبر المشروع ولكن الحد الادني من المتطلبات كالتالي:

- مبرمجة ميكروكنترولار



- بورد breadboard مع المكونات المطلوبة
- الميكروكنترولار نفسها microcontroller chips
- جهاز كمبيوتر
- ادوات قياس كالافوميتر

والسوفت وير :

- الكومبيلر المستخدم مثل ال PICC او microcontroller assembler
- برنامج للتعامل مع المبرمجة ووضع برنامجنا علي الميكروكنترولار device programmer

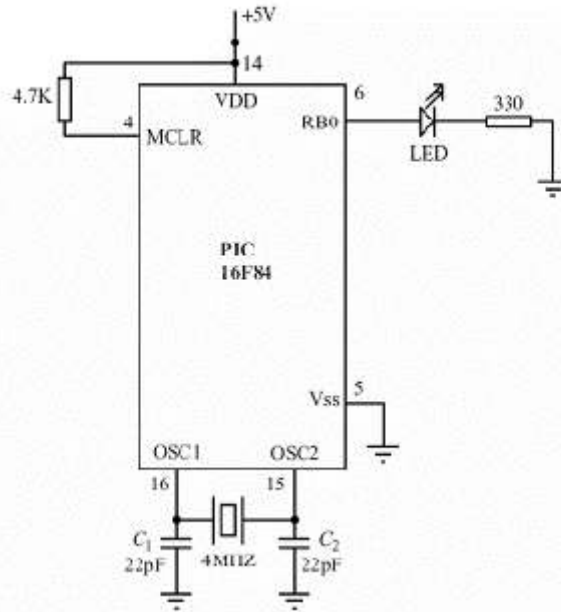
وفي تجربتنا العملية سوف نحتاج:

- كومبيلر ال PICC
- ميكرو كنترولار 16F84A
- كريستال 4 MHz
- مكثفات سعة 22 بيكو فراد
- مقاومات 330 أوم
- مقاومات 4.7 كيلو أوم
- مفاتيح Push - Button
- باور سبلاي و دائرة الباور سبلاي LM7805 22uf, 100mf, 10uf

8.1 التجارب العملية

التجربة الاولى LED Flasher

لدينا LED موصلة مع ال bit 0 من البورت B اللي هو pin RB0 للميكروكنترولار 16F84A ونريد كتابة برنامج يقوم بانارة ال LED كل 100 ميلي ثانية مع العلم ان الميكرو تعمل مع 4MHz Crystal

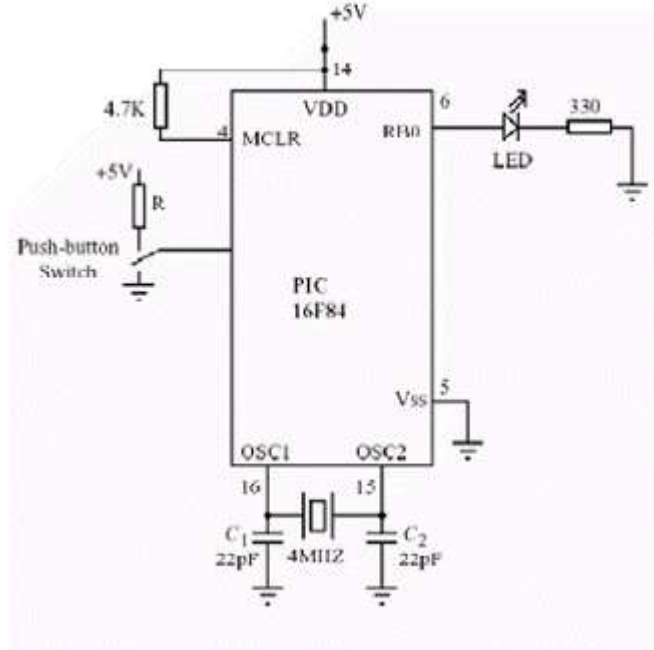


في بداية البرنامج سيتم تعريف البورت B علي انه output port بعد ذلك ننير ال LED بارسال اشارة 1 الي البورت اي RB0 = 1 وبعد 100 ميلي ثانية نقوم باطفاء ال LED بارسال 0 اي RB = 0

```
// This program flashes an LED connected to port RB0 of a /
// PIC16F84
// microcontroller.
// The microcontroller is operated with a 4MHz crystal.
//
//*****
//*****
#include <pic.h>
#include <delay.c>
//
// Main Program
// *****
//
main(void)
{
    TRISB = 0;    /* PORT B is output */
    for(;;)      /* Do FOREVER */
    {
        RB0 = 1;    /* Turn ON LED */
        DelayMs(100); /* 100ms delay */
        RB0 = 0;    /* Turn OFF LED */
        DelayMs(100); /* 100ms delay */
    }
}
```

التجربة الثانية push-button

لدينا LED موصلة مع bit 0 للبورث B يعني pin RB0 للميكروكنترولر 16F84A ايضا لدينا مفتاح push - button موصل مع ال bit 7 للبورث B يعني pin RB7 وموصل معه مقاومة ونريد الان كتابة برنامج لانارة ال LED كلما ضغنا علي المفتاح.



في بداية البرنامج سنقوم بتعريف البرورث Port Pin RB0 علي انة OutPut و البورث port pin RB7 علي انة Input ويكون البرنامج علي هذا الشكل:

```
// PUSH-BUTTON AND LED PROGRAM
// =====
// File: BUTTON.C
//
// This program turns ON an LED when a push-button
// switch is pressed.
// The LED is connected to port pin RB0 and the switch
// is connected
// RB7. The microcontroller is operated with a 4MHz
// crystal.
//
//*****
#include <pic.h>
//
// Main Program
// *****
//
main(void)
{
    TRISB = 0x80; /* RB0 is output, RB7 is input */
    RB0 = 0; /* Make sure the LED is OFF when
               started */
    while(RB7 == 1); /* Wait until switch is pressed */
    RB0 = 1; /* Turn ON LED */
}
```