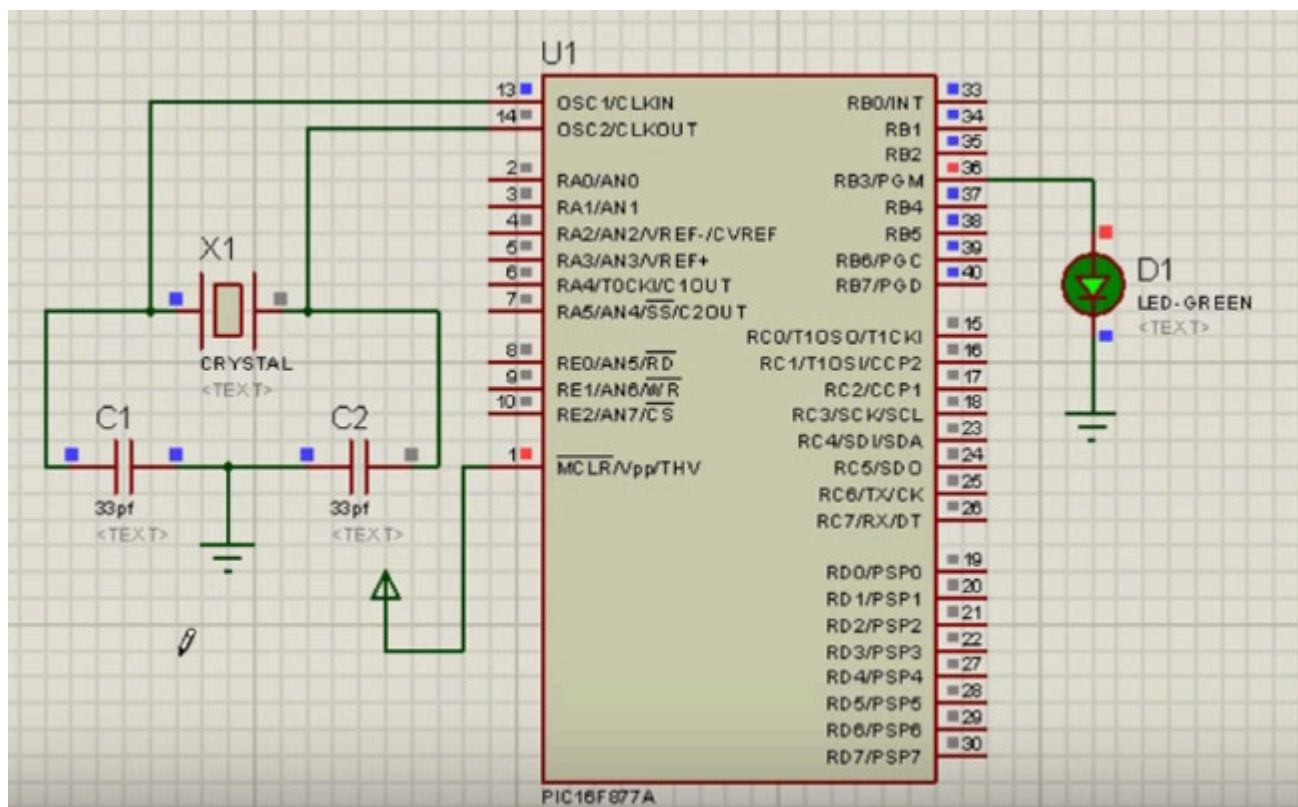


Написание первой программы для микроконтроллера PIC с помощью MPLABX

admin-new

В нашей [предыдущей статье](#) про микроконтроллеры PIC мы рассмотрели архитектуру микроконтроллеров PIC, установили необходимое для работы с ними программное обеспечение и купили программатор PicKit 3. В данной же статье мы напишем нашу первую программу мигания светодиодом для микроконтроллера PIC16F877A и рассмотрим установку в нем битов конфигурации (фьюзов).

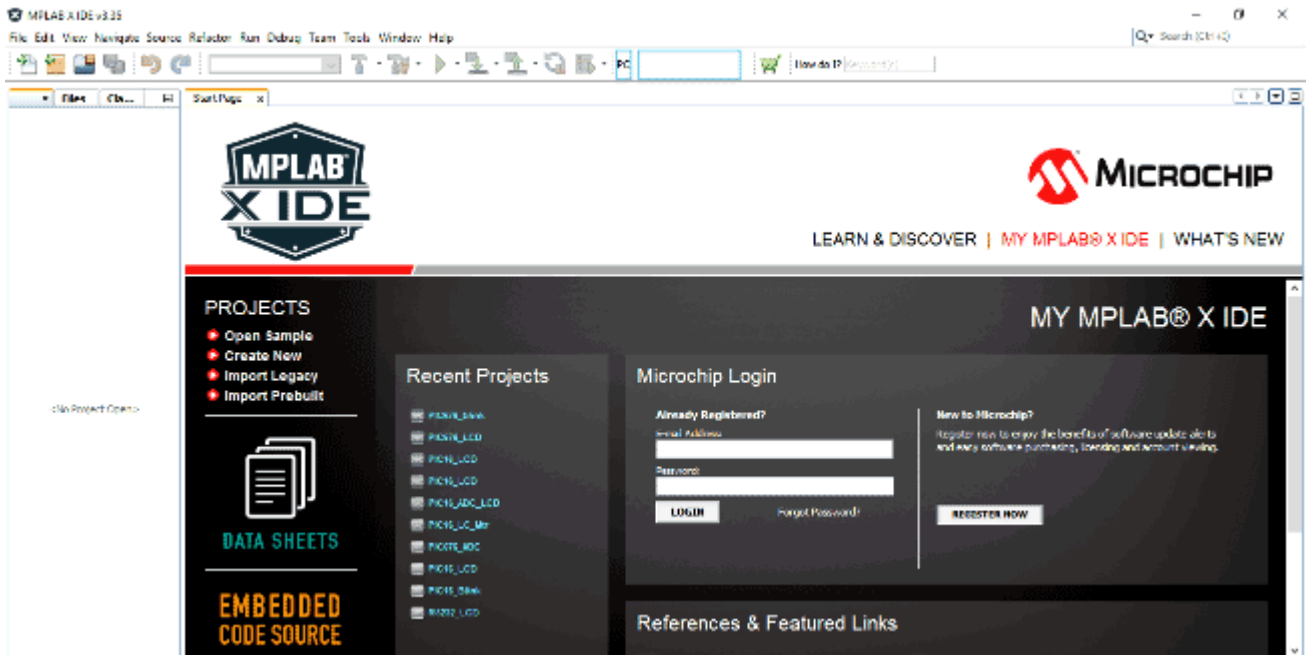


Поскольку мы используем микроконтроллер PIC16F877A вместе с компилятором XC8, то первым делом желательно изучить даташиты на них. Всегда, когда вы начинаете знакомство с новым микроконтроллером, желательно полностью прочитать даташит на него.

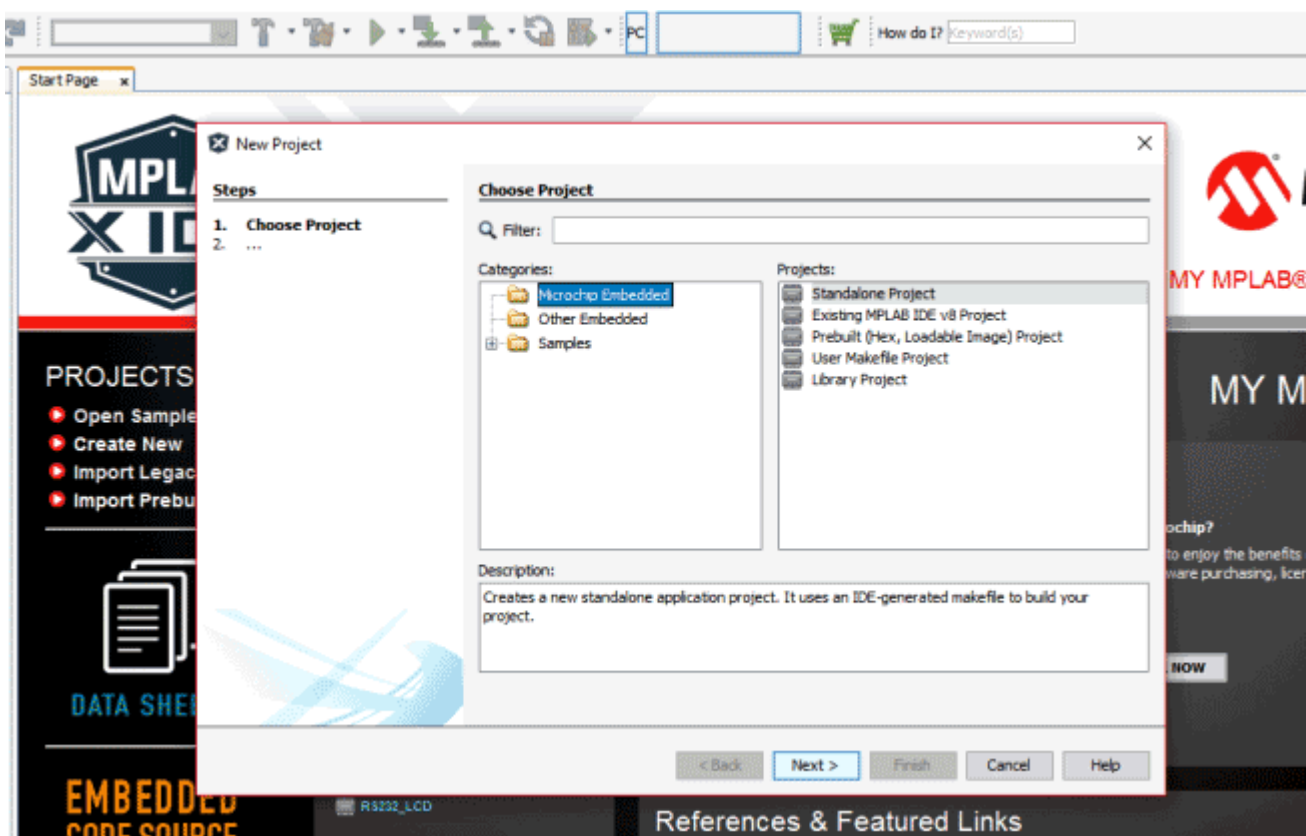
Программу для нашего микроконтроллера PIC16F877A мы будем писать в среде MPLAB-X

Создание нового проекта в MPLAB-X

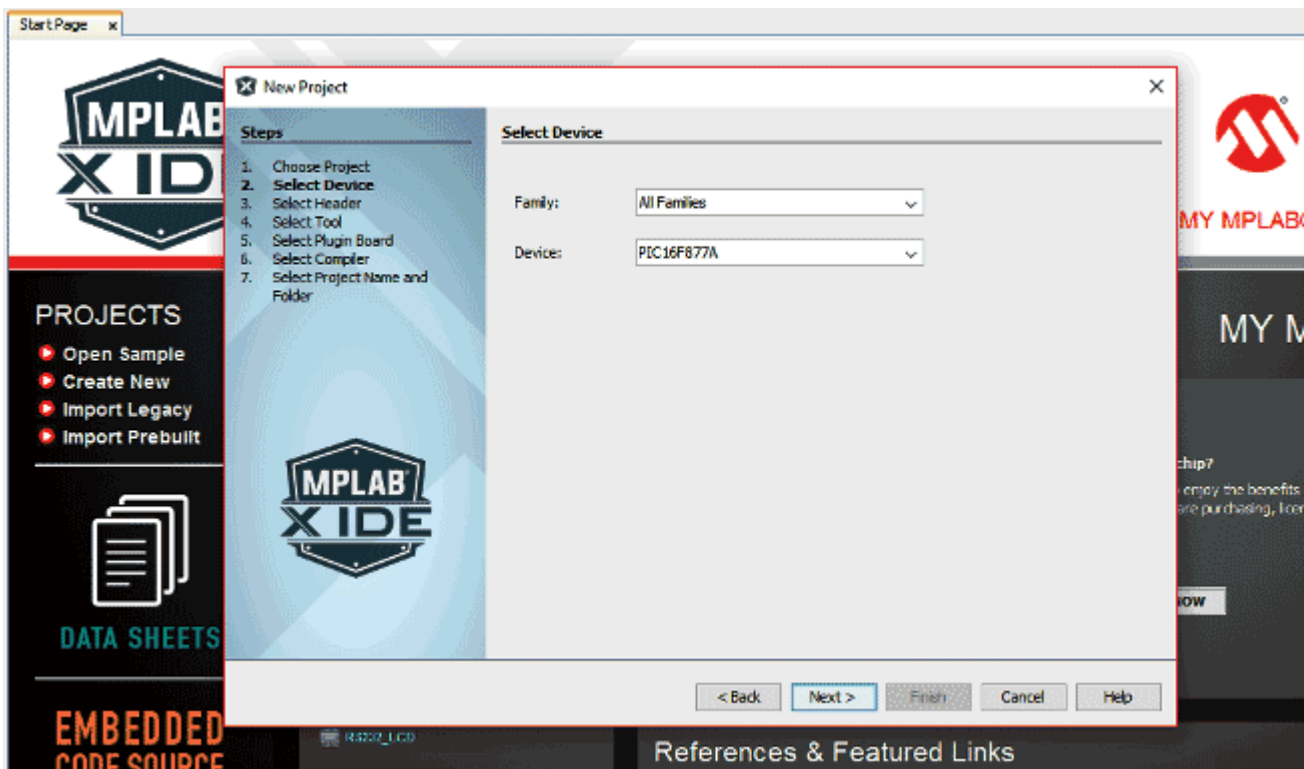
Шаг 1. Запустите IDE MPLAB-X, в результате чего вы на экране должны увидеть следующее окно.



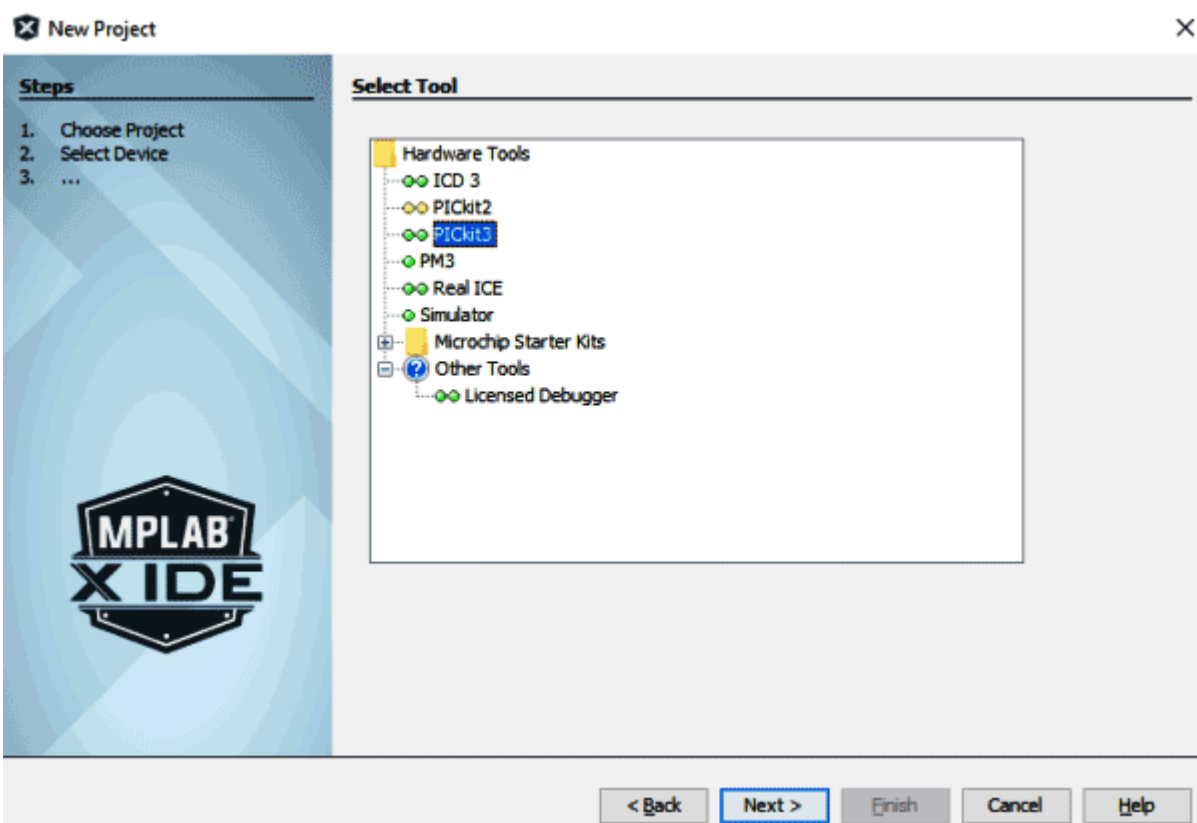
Шаг 2. Откройте пункт меню Files -> New Project (для этой цели можно также использовать комбинацию клавиш Ctrl+Shift+N). После этого вы на экране увидите следующее всплывающее окно, в котором вам необходимо выбрать Standalone Project и нажать Next.



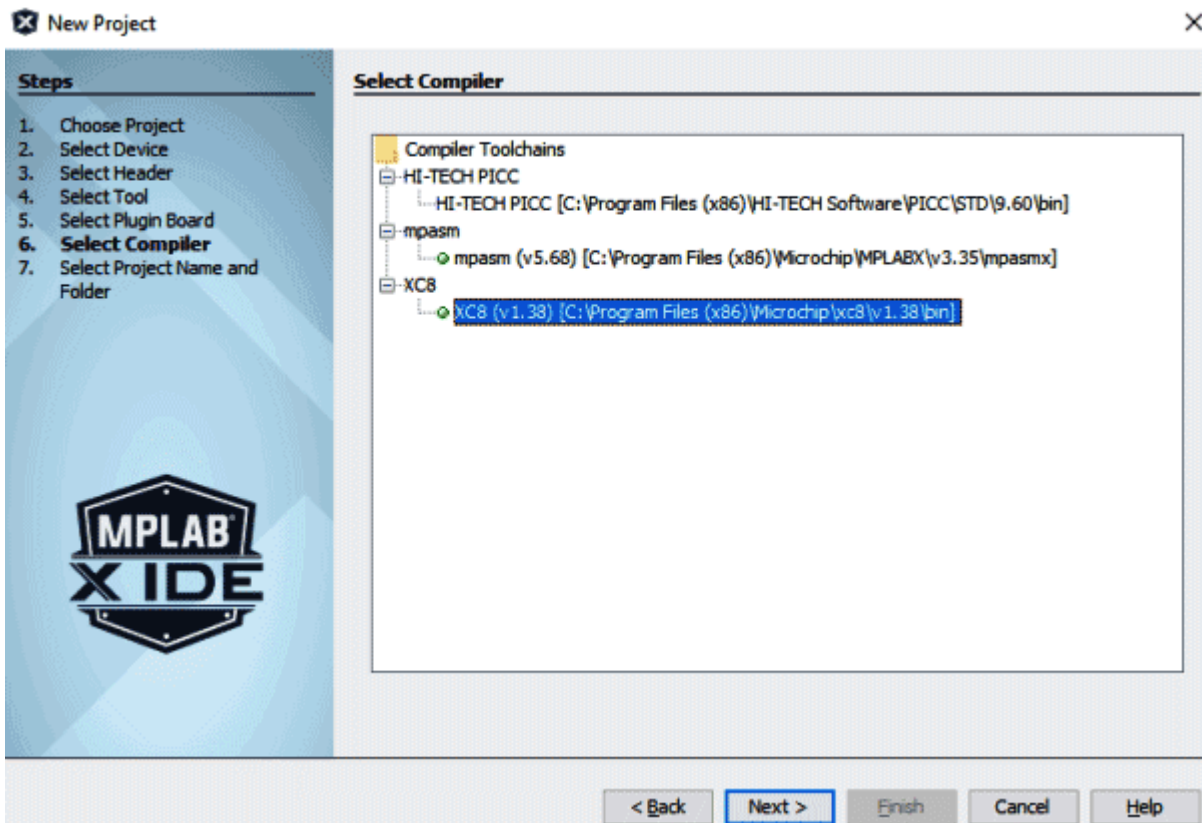
Шаг 3. Теперь необходимо выбрать устройство для нашего проекта – выберем из выпадающего списка PIC16F877A. После этого нажмем на Next.



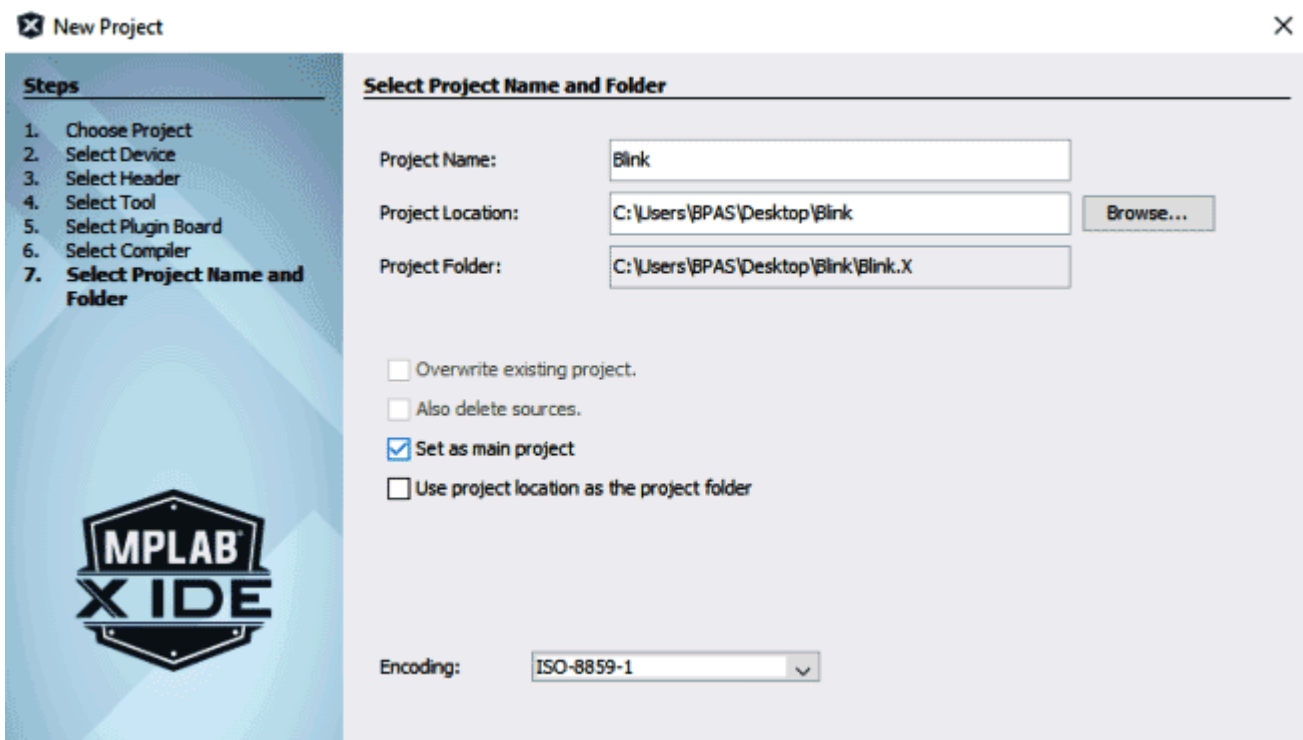
Шаг 4. На следующей странице нам необходимо выбрать инструменты (программатор) для нашего проекта. Выберем PicKit 3 и нажмем на Next.



Шаг 5. Затем необходимо выбрать компилятор – выберем XC8 и нажмем на Next.

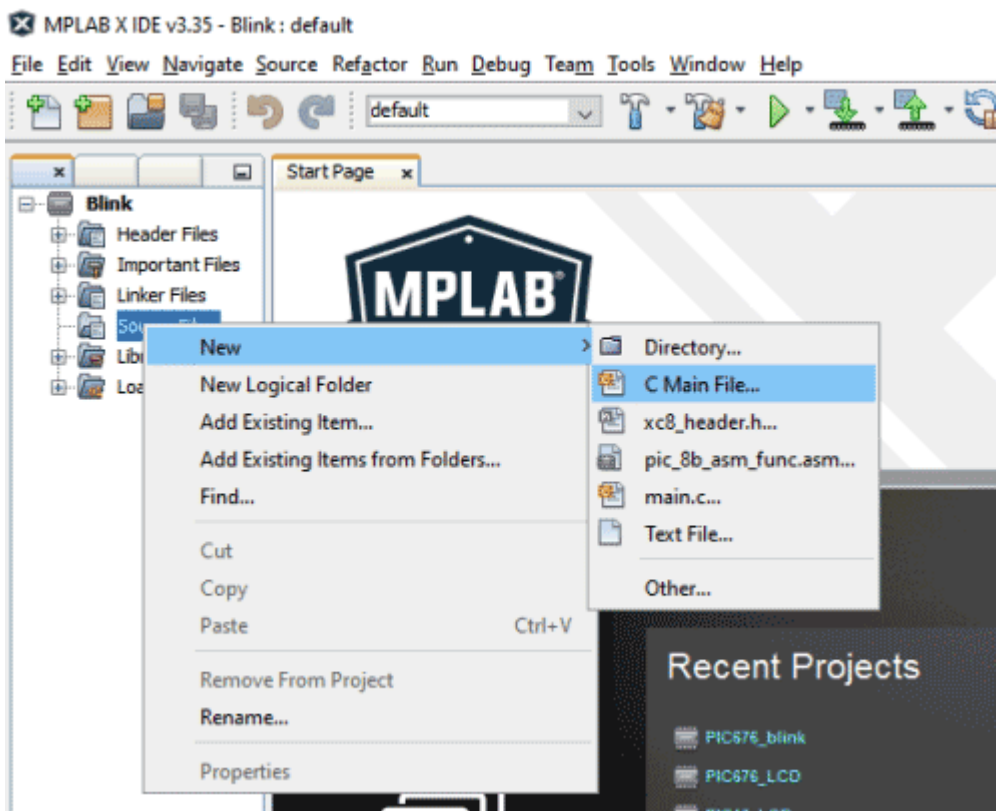


Шаг 6. На следующей странице необходимо задать имя нашего проекта и местоположение куда он будет сохраняться. Мы назвали проект Blink. Файл проекта будет сохраняться с расширением .X, что позволяет запускать его непосредственным образом из MAPLB-X. После этого нажмем на Finish.

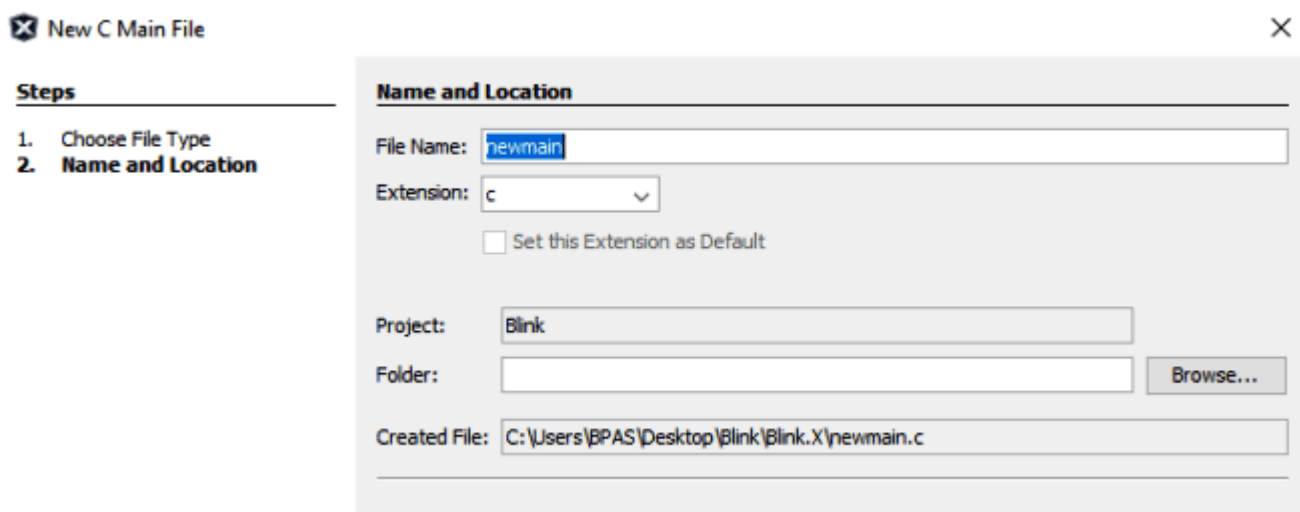


Шаг 7. Поздравляем. Теперь наш проект успешно создан. Слева в окне мы увидим имя нашего проекта (у нас оно Blink), нажмите на него чтобы посмотреть все файлы, которые входят в проект.

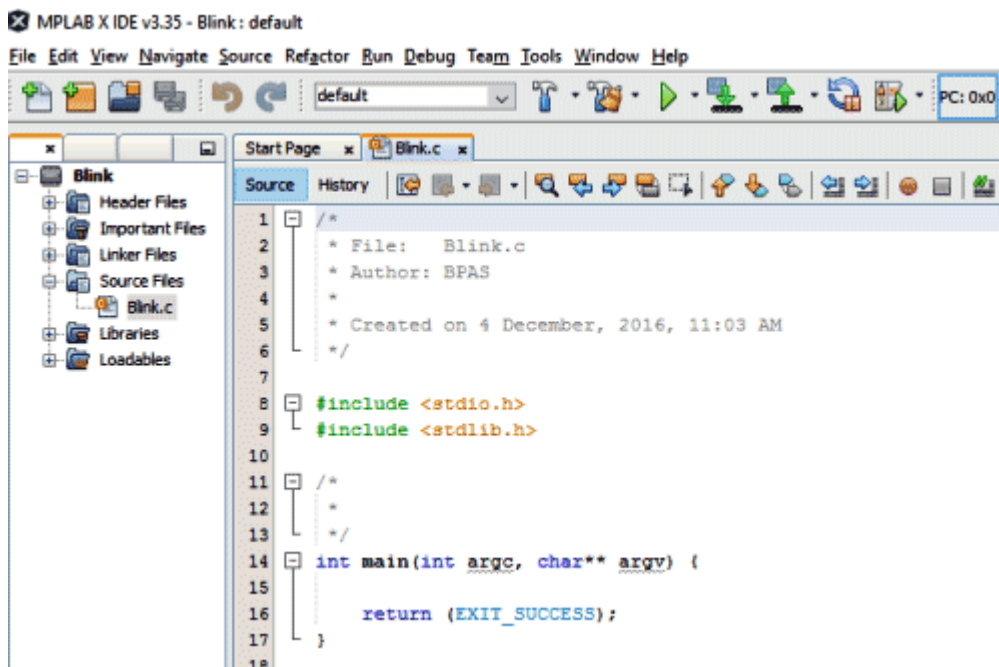
Чтобы начать написание программы в наш проект необходимо добавить основной файл (C Main file) в каталог с нашим проектом. Для этого сделайте правый клик мыши на source file и в открывшемся меню выберете New -> C Main File как показано на следующем рисунке.



Шаг 8. Откроется диалоговое окно с именем нашего C-файла. Мы назвали его снова Blink, но вы можете выбрать для него любое другое имя по своему усмотрению. После ввода имени файла нажмите на finish.



Шаг 9. После того как файл C будет создан, откроется IDE с некоторым кодом в ней по умолчанию, как показано на следующем рисунке.



Шаг 10. После этого можно начать написание программы в основном С файле (C-main File). Код по умолчанию, формируемый IDE, не будет нами использоваться в этом проекте, поэтому удалим его полностью.

Общие сведения о регистрах конфигурации в микроконтроллерах PIC

Микроконтроллеры PIC имеют несколько мест, в которых расположены биты конфигурации, которые также называют фьюзами (fuses). Эти биты определяют глобальную логику функционирования устройства (в нашем случае микроконтроллера). С их помощью задается режим работы генератора тактовой частоты, сторожевого таймера (watchdog timer), режима программирования и защиты кода. Очень важно установить эти биты правильно иначе наш микроконтроллер не сможет нормально функционировать.

Узнать об имеющихся регистрах конфигурации можно из даташита на микроконтроллер. Важно понять, какие типы битов конфигурации присутствуют в рассматриваемом микроконтроллере и каковы их функции. Эти биты можно установить или сбросить используя инструкцию *pragma*.

Данная инструкция имеет следующую форму:

```
#pragma config setting = state|value
```

```
#pragma config register = value
```

где *setting* – описание настройки (установки), например, WDT, а *state* – текстовое описание необходимого состояния, например, OFF.

Можно привести следующие примеры установки конфигурационных битов:

```
#pragma config WDT = ON // turn on watchdog timer (включить сторожевой таймер)
```

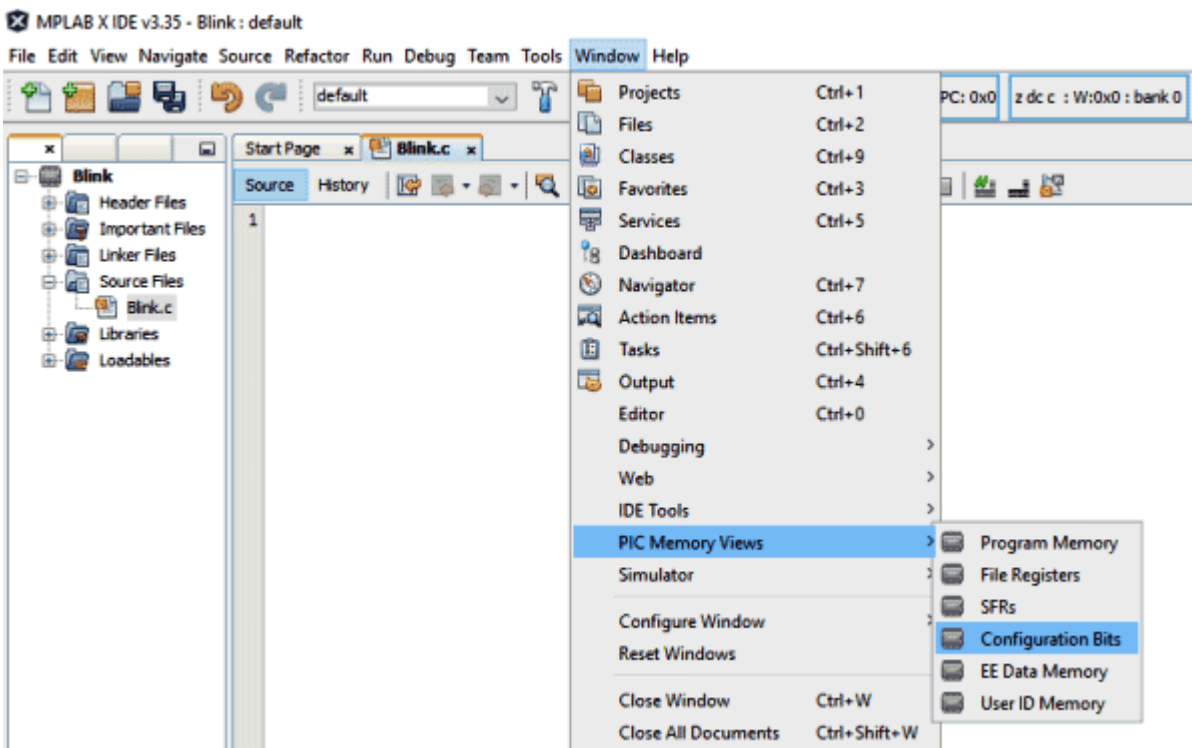
```
#pragma config WDTPS = 0x1A // specify the timer postscale value (записать значение в таймер)
```

Для начинающих правильная установка битов конфигурации с помощью данных инструкций может представлять собой достаточно сложную задачу, которую можно значительно упростить в нашей среде разработки MPLAB-X.

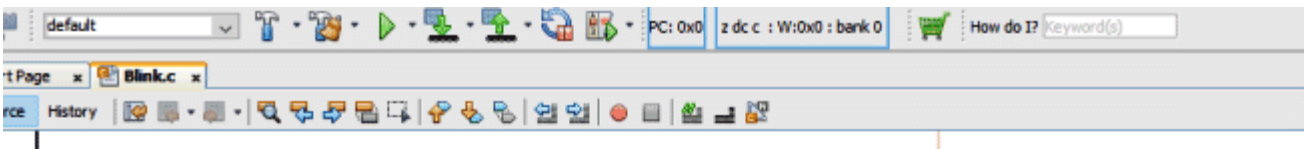
Установка битов конфигурации (фьюзов) в MPLAB-X

Компания Microchip значительно упростила процесс установки битов конфигурации (фьюзов) в микроконтроллерах PIC, разработав для этого специальные графические инструменты. Для того, чтобы установить биты конфигурации (фьюзы) в микроконтроллерах PIC с помощью программы MPLAB-X, выполните следующую последовательность шагов.

Шаг 1. В программе MPLAB-X выберите пункт меню Window -> PIC Memory View -> Configuration Bits как показано на следующем рисунке.



Шаг 2. После этого внизу программы MPLAB-X откроется окно настройки битов конфигурации (Configuration Bits window) как показано на рисунке ниже. В нем вы можете установить значение каждого бита конфигурации по своему желанию. Далее мы рассмотрим установку некоторых из этих бит.



Address	Name	Value	Field	Option	Category	Setting
2007	CONFIG	FFFF	POSC	EXTRC	Oscillator Selection bits	RC oscillator
			WDTE	ON	Watchdog Timer Enable bit	WDT enabled
			PWRTE	OFF	Power-up Timer Enable bit	PWRT disabled
			BOREN	ON	Brown-out Reset Enable bit	BOR enabled
			LVP	ON	Low-Voltage (Single-Supply) In-Circuit Serial Programming Enable bit	RB3/PGM pin has PGM fu
			CPD	OFF	Data EEPROM Memory Code Protection bit	Data EEPROM code prote
			WRT	OFF	Flash Program Memory Write Enable bits	Write protection off;
			CP	OFF	Flash Program Memory Code Protection bit	Code protection off

Шаг 3. Первыми битами конфигурации являются **биты выбора генератора**. Микроконтроллер PIC16F87XA может работать в одном из 4-х режимов генератора, которые можно установить с помощью битов FOSC1 и FOSC0:

- LP Low-Power Crystal (маломощный генератор);
- XT Crystal/Resonator (внешний кварцевый резонатор);
- HS High-Speed Crystal/Resonator (высокоскоростной кварцевый резонатор);
- RC Resistor/Capacitor (RC генератор).

Возможные частоты работы микроконтроллера в зависимости от выбранного типа генератора приведены в следующей таблице.

Osc Type	Crystal Freq.	Cap. Range C1	Cap. Range C2
LP	32 kHz	33 pF	33 pF
	200 kHz	15 pF	15 pF
XT	200 kHz	47-68 pF	47-68 pF
	1 MHz	15 pF	15 pF
	4 MHz	15 pF	15 pF
HS	4 MHz	15 pF	15 pF
	8 MHz	15-33 pF	15-33 pF
	20 MHz	15-33 pF	15-33 pF

Для нашего проекта мы будем использовать внешний кварцевый генератор на 20Mhz, поэтому нам необходимо выбрать HS из выпадающего меню.

Шаг 4. Следующим битом конфигурации является **бит включения/выключения сторожевого таймера**.

Сторожевой таймер (Watchdog Timer, WDT) представляет собой непрерывно работающий встроенный RC генератор, для работы которого не требуется никаких внешних компонентов. Данный RC генератор отделен от RC генератора на контакте OSC1/CLKI. Это означает что сторожевой таймер будет продолжать работать даже если на контакты OSC1/CLKI и OSC2/CLKO не подается никакой тактовой частоты. В режиме нормального функционирования

сторожевой таймер формирует сигнал аппаратного сброса (Watchdog Timer Reset) при своем переполнении. И если таймер не отключен в нашей программе, то микроконтроллер будет сбрасываться каждый раз при переполнении таймера. Сторожевой таймер можно отключить при помощи очистки его бита конфигурации.

В нашей программе мы не будем использовать сторожевой таймер, поэтому в выпадающем списке напротив его бита конфигурации выберем OFF.

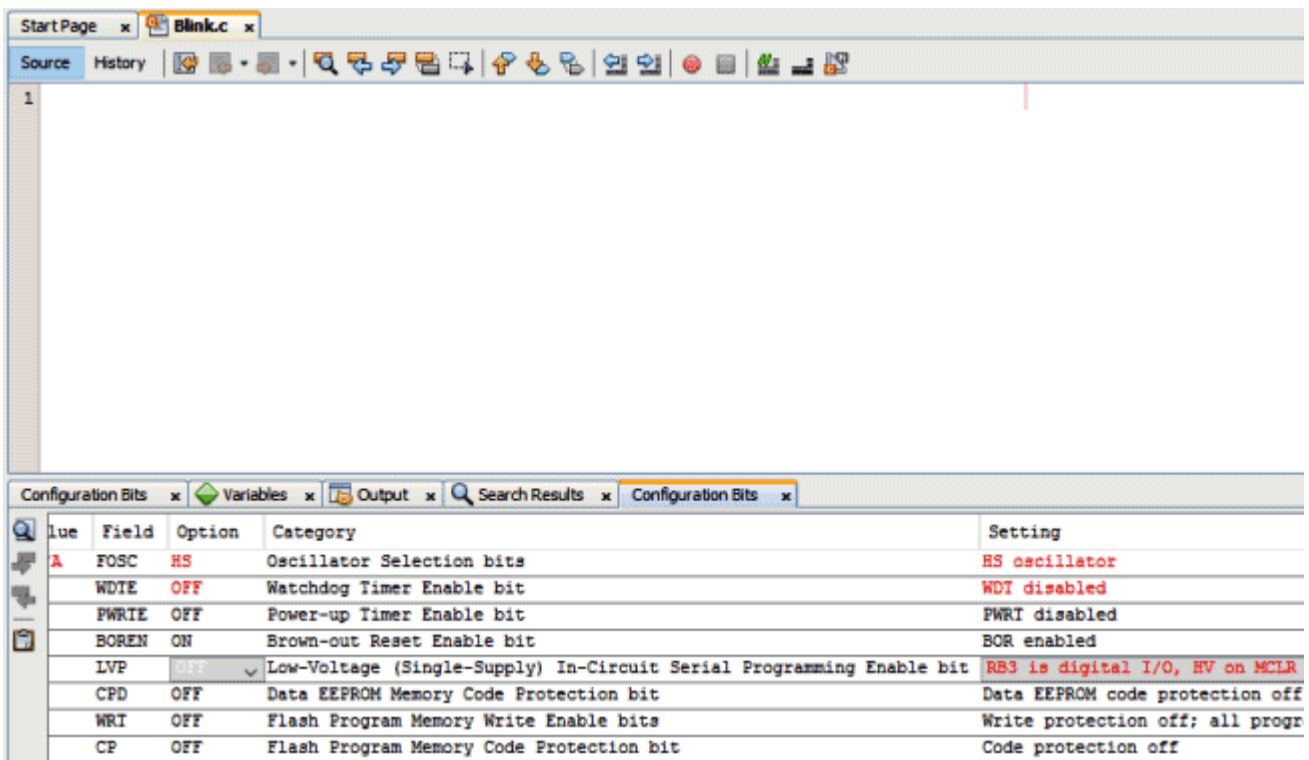
Шаг 5. Следующим битом является **таймер по питанию (Power-up timer Bit, PWRT)**, который обеспечивает фиксированную задержку в 72 мс в работе микроконтроллера при подаче на него питания, что позволяет напряжению питания за это время подняться до приемлемого уровня. Когда данный бит активирован, таймер по питанию будет сбрасывать микроконтроллер до тех пор, пока питание не поднимется до необходимого уровня.

В нашей программе мы не будем использовать данную задержку, поэтому данный бит конфигурации мы также отключим (OFF).

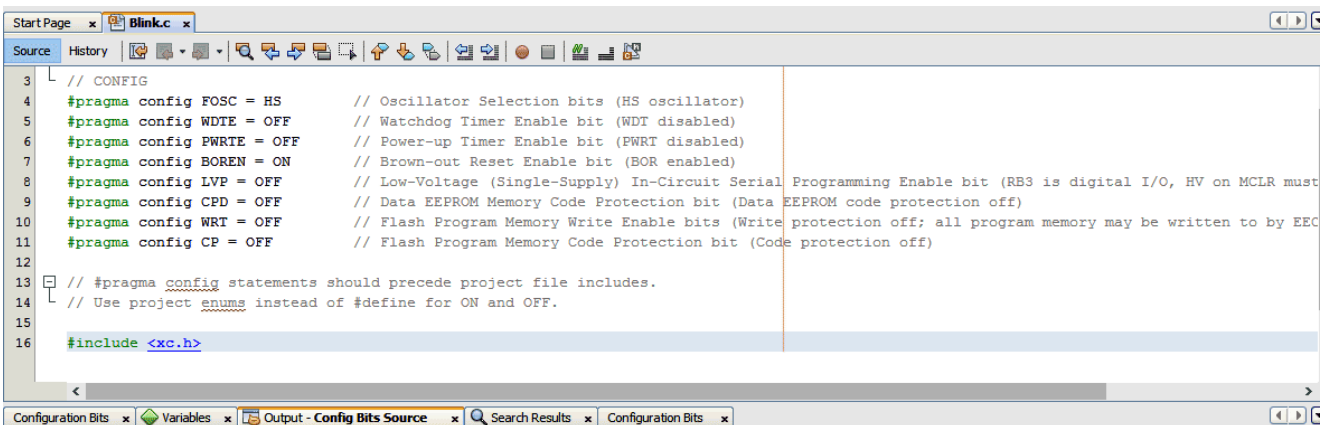
Шаг 6. Следующим битом конфигурации является **бит программирования при низком напряжении (Low-Voltage Programming, LVP)**. Бит LVP позволяет производить программирование микроконтроллера пониженным напряжением через разъем ICSP. Мы не будем в нашем проекте использовать данный вид программирования, поэтому отключим данный бит (OFF).

Шаг 7. Следующими битами конфигурации у нас является **бит EEPROM и биты защиты программы**. Если бит EEPROM включен, то после программирования микроконтроллера никто не сможет перезаписать в нем программу. В нашем случае мы оставим все эти три бита в выключенном состоянии.

После того как все необходимые настройки битов конфигурации сделаны наше диалоговое окно должно выглядеть следующим образом.



Шаг 8. После этого выберите пункт меню Generate Source Code to Output – в результате этого код нашей программы будет сгенерирован, нам необходимо всего лишь скопировать его вместе с заголовочным файлом и вставить его в файл Blink.c как показано на следующем рисунке.



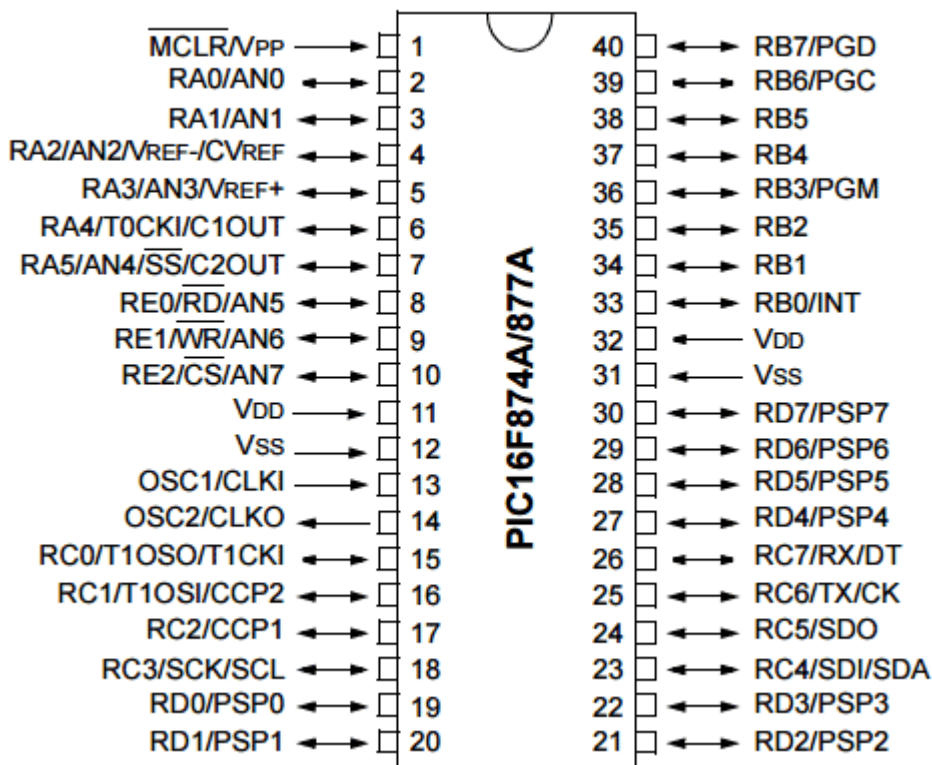
На этом настройка битов конфигурации (фьюзов) для нашего проекта будет закончена. Мы будем использовать подобную настройку битов

конфигурации для всех проектов на основе микроконтроллеров PIC, которые мы далее рассмотрим на нашем сайте.

Написание программы мигания светодиодом для микроконтроллера PIC

В нашем проекте мы будем подключать светодиод к одному из контактов ввода/вывода микроконтроллера PIC16F877A, с помощью которого мы будем управлять миганием светодиода. Распиновка микроконтроллера PIC16F877A представлена на следующем рисунке.

40-Pin PDIP



Как видно из представленного рисунка, микроконтроллер PIC16F877 имеет 5 основных портов ввода/вывода. Обычно они обозначаются как PORT A (RA), PORT B (RB), PORT C (RC), PORT D (RD) и PORT E (RE). В данном случае PORT A содержит 6 контактов (с RA-0 по RA-5), "PORT B", "PORT C" и "PORT D" содержат по 8 контактов (с RB-0 по RB-7, с RC-0 по RC-7, с RD-0 по RD-7), "PORT E" содержит всего 3 контакта (с RE-0 по RE-2).

PORT A	RA-0 to RA-5	6 bits wide
PORT B	RB-0 to RB-7	8 bits wide
PORT C	RC-0 to RC-7	8 bits wide
PORT D	RD-0 to RD-7	8 bits wide
PORT E	RE-0 to RE-2	3 bits wide

Все эти порты являются двунаправленными. Направление работы порта управляется с помощью регистров TRIS(X) (TRIS A используется для установки направления работы PORT-A, TRIS B используется для установки направления работы PORT-B и т.д.). Установка бита регистра TRIS(X) в '1' будет означать, что соответствующий ему контакт порта PORT(X) сконфигурирован для работы на ввод данных (input). Установка бита регистра TRIS(X) в '0' будет означать, что соответствующий ему контакт порта PORT(X) сконфигурирован для работы на вывод данных (output).

В нашем проекте мы установим контакт RB3 порта PORT B для работы на вывод данных – к нему подключен светодиод. Код программы для микроконтроллера PIC для мигания светодиодом будет выглядеть следующим образом:

```
#include <xc.h>

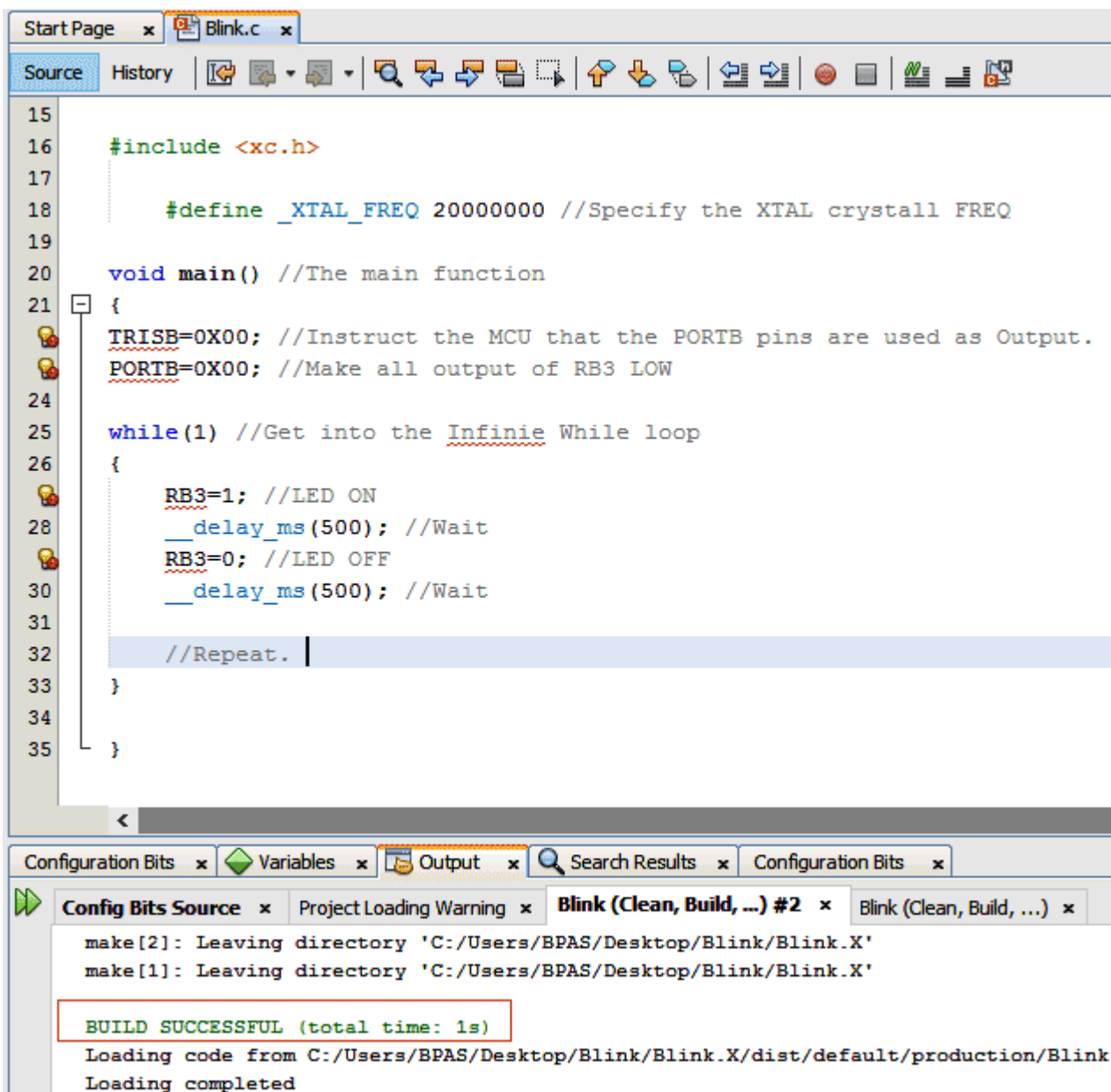
#define _XTAL_FREQ 2000000 //Specify the XTAL crystal FREQ

void main() //The main function
{
    TRISB=0X00; //Instruct the MCU that the PORTB pins are used as Output.
    PORTB=0X00; //Make all output of RB3 LOW
    while(1) //Get into the Infinite While loop
    {
        RB3=1; //LED ON
        __delay_ms(500); //Wait
        RB3=0; //LED OFF
        __delay_ms(500); //Wait
        //Repeat.
    }
}
```

```
}
```

В данном коде программы мы первым делом указываем частоту внешнего кварцевого генератора с помощью инструкции `#define _XTAL_FREQ 2000000`. Затем в функции `void main()` мы указываем что контакт RB3 будет работать на вывод данных (`TRISB=0X00`). И затем идет бесконечный цикл в котором производится мигание светодиодом.

После того как написание кода программы будет закончено выберем пункт меню `Run -> Build Main Project`. В результате его выполнения ваша программа будет скомпилирована. Если все нормально, то в нижнем окне программы вы увидите сообщение `BUILD SUCCESSFUL` как показано на следующем рисунке.



The screenshot displays an IDE window with a C source file named 'Blink.c'. The code defines the crystal frequency, sets the output pin RB3, and enters an infinite loop that toggles RB3 with 500ms delays. Below the code editor, the 'Output' window shows the compilation process, with 'BUILD SUCCESSFUL (total time: 1s)' highlighted in a red box.

```
15
16 #include <xc.h>
17
18 #define _XTAL_FREQ 2000000 //Specify the XTAL crystall FREQ
19
20 void main() //The main function
21 {
22     TRISB=0X00; //Instruct the MCU that the PORTB pins are used as Output.
23     PORTB=0X00; //Make all output of RB3 LOW
24
25     while(1) //Get into the Infinite While loop
26     {
27         RB3=1; //LED ON
28         __delay_ms (500); //Wait
29         RB3=0; //LED OFF
30         __delay_ms (500); //Wait
31
32         //Repeat.
33     }
34
35 }
```

Configuration Bits x Variables x Output x Search Results x Configuration Bits x

Config Bits Source x Project Loading Warning x Blink (Clean, Build, ...) #2 x Blink (Clean, Build, ...) x

```
make[2]: Leaving directory 'C:/Users/BPAS/Desktop/Blink/Blink.X'
make[1]: Leaving directory 'C:/Users/BPAS/Desktop/Blink/Blink.X'

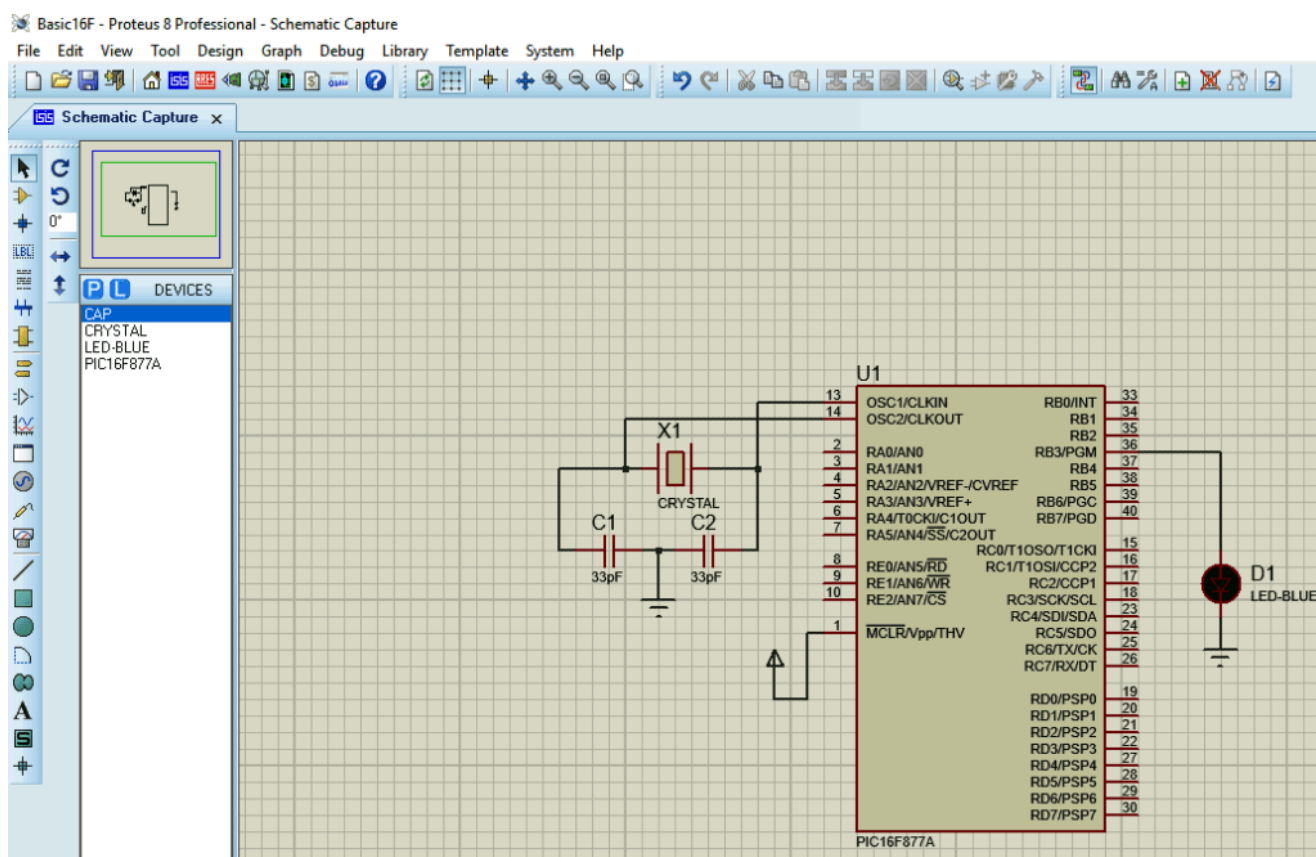
BUILD SUCCESSFUL (total time: 1s)
Loading code from C:/Users/BPAS/Desktop/Blink/Blink.X/dist/default/production/Blink
Loading completed
```

Разработка схемы проекта в Proteus

Если компоновка (Build) нашего проекта была успешной, то нашей IDE в фоновом режиме будет сгенерирован HEX файл, который можно найти внутри каталога

Desktop\Blink\Blink.X\dist\default\production (каталог может отличаться, если вы в настройках выбрали другой путь сохранения файлов).

Теперь откроем программу Proteus, установленную нами ранее и создадим в ней схему для нашего проекта. Процесс создания схемы проекта в Proteus объяснен в видео, приведенном в конце статьи. В результате в окне программы Proteus вы должны увидеть примерно следующую картину:



Для тестирования работы схемы в Proteus нажмите кнопку play в нижнем левом углу экрана после загрузки Hex файла. Светодиод, подключенный к микроконтроллеру, должен начать мигать.

Таким образом, в данной статье мы написали нашу первую программу для микроконтроллера PIC и протестировали ее работу в симуляторе Proteus.

Исходный код программы

```
1
2
3
4
5
6
7
8 #include <xc.h>
9 #define _XTAL_FREQ 2000000 //Specify the XTAL crystal FREQ
10 void main() //The main function
11 {
12     TRISB=0X00; //Instruct the MCU that the PORTB pins are used as Output.
13     PORTB=0X00; //Make all output of RB3 LOW
14     while(1) //Get into the Infinite While loop
15     {
16         RB3=1; //LED ON
17         __delay_ms(500); //Wait
18         RB3=0; //LED OFF
19         __delay_ms(500); //Wait
20         //Repeat.
21     }
22 }
23
24
25
26
27
28
```

Видео, демонстрирующее работу проекта



Загрузка...

9 116 просмотров