Лабораторная работа №1

Работа с портами. Автоконфигуратор. Работа с прерываниями.

Предварительное задание:

1) Изучить раздел «Port Input/Output» в технической документации микроконтроллера C8051F060.

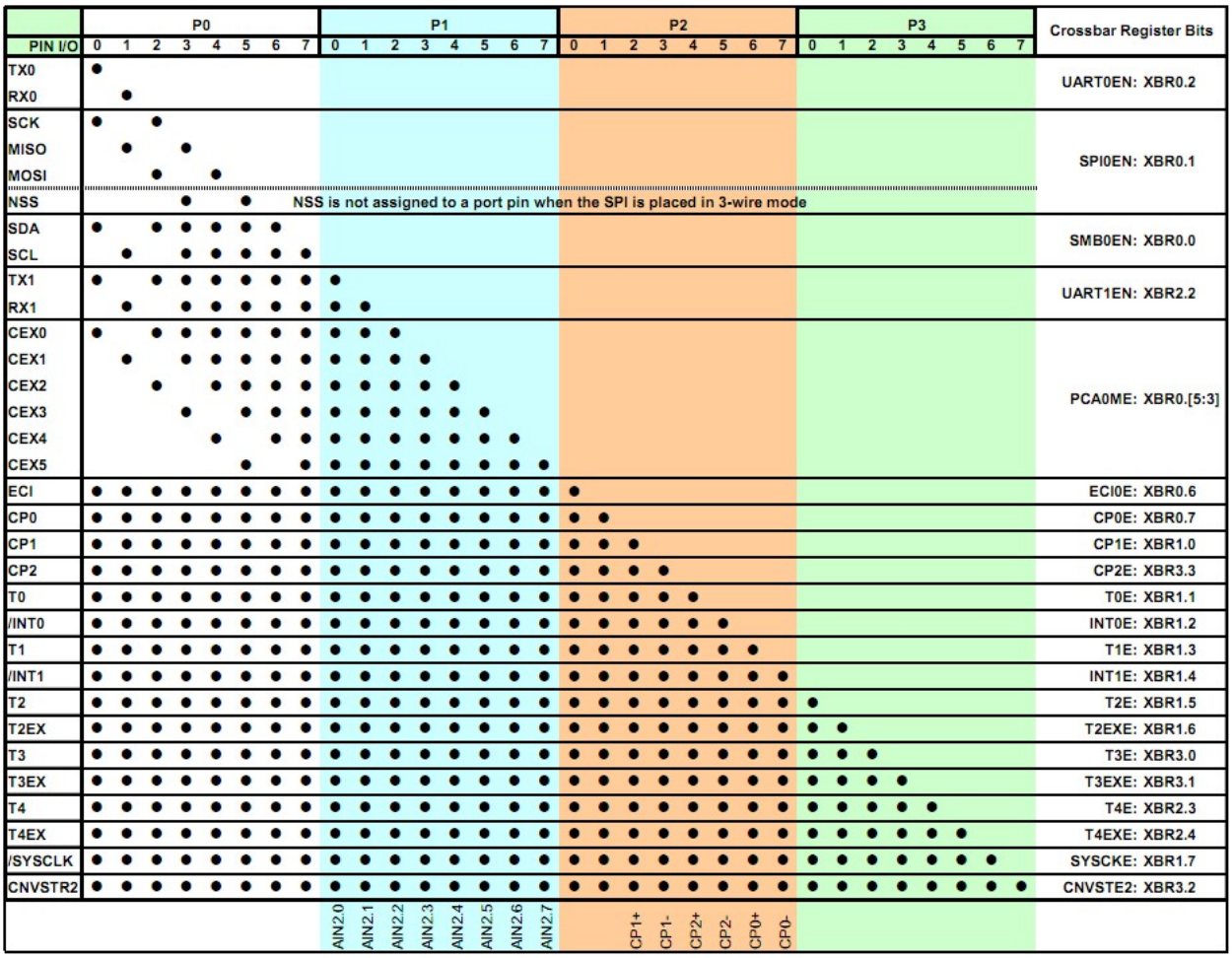
2) Изучить принципиальную схему отладочного модуля C8051F060-DK.

3) Изучить принципиальную схему платы расширения.

4) Вспомнить синтаксис и семантику языка программирования C.

Порты ввода-вывода микроконтроллера C8051F060.

Доступ линий внутренней шины процессора к выводам микросхемы осуществляется через приоритетную матрицу (crossbar). Линии портов P0-P3 могут работать как выходы общего назначения, так и выходы со специальными функциями, подключенными к внутрисистемной периферии. Матрица приоритетов подключает порты к соответствующей периферии в порядке приоритетов. Приоритеты линий порта убывают в порядке от P0.0 до P3.7. Специальные функции назначаются выводам в порядке приоритетов, показанном на схематическом изображении матрицы.

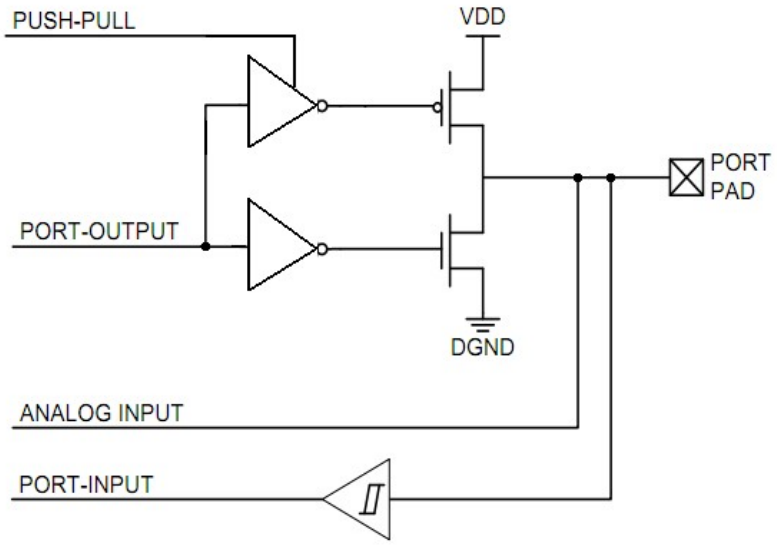


Если какому-либо периферийному модулю вывод не назначен, вся схема приоритетов сдвигается влево на соответствующее количество позиций. Управление приоритетной матрицей осуществляется с помощью регистров XBR0, XBR1, XBR2, XBR3 (см. datasheet). По умолчанию все линии портов настроены как входы-выходы общего

назначения. Однако чтобы они заработали, нужно включить матрицу, установив бит XBARE в регистре XBR2:

```
XBR2 |= 0x40;  //crossbar enable
```

Линии портов могут работать в двух основных режимах: цифровой двухтактный выход (push-pull) и выход с открытым стоком (open drain).



В режиме цифрового двухтактного выхода работают оба плеча выходного транзисторного каскада. В режиме с открытым стоком работает только нижний транзистор, а верхний постоянно закрыт. Настройка режима осуществляется с помощью регистров P0MDOUT, P1MDOUT, P2MDOUT и P3MDOUT. По умолчанию все линии настроены как выходы с открытым стоком. Запись единицы в соответствующий разряд переводит линию в режим цифрового двухтактного выхода:

```
P1MDOUT |= 0x01;  //push-pull mode for P1.0
```

Если линия используется как цифровой или аналоговый вход, ее нужно перевести в режим с открытым стоком и записать единицу в выходной драйвер (это состояние по умолчанию).

```
P1MDOUT &= 0xFE;  //open-drain mode for P1.0
P1 |= 0x01;  //set line P1.0 in 1
```

Создание проекта в среде SiLabs IDE

Запустите SiLabs IDE. Для создания нового проекта выберите в меню Project → New Project. В открывшемся окне выберите семейство C8051F06x, введите название создаваемого проекта, укажите директорию (для каждого проекта лучше всего создавать отдельную папку) и выберите тип проекта C Source Project. Нажмите OK.

Созданный проект не имеет исходного файла, поэтому нужно его создать. Выберите в меню File → New File, в открывшемся окне выберите C Source File, укажите директорию и имя, а также поставьте флажок Add to project. Созданный пустой файл автоматически добавится в дерево проекта.

Создадим заготовку исходного файла:

```
//--шаблон программы-----------------------------------
#include <c8051f060.h>       // заголовочный файл с адресами регистров специальных функций
```

```
//Здесь должны быть объявлены глобальные переменные и константы
//а также приведены прототипы подпрограмм
void Init_Device(void);

//начало основной программы:
void main (void)
{
  Init_Device();
  // здесь должны быть глобальные настройки

  while (1)  //основной бесконечный цикл
    {
          // здесь должно быть тело программы

    }// end of while(1)
}    // end of main()
// здесь можно писать подпрограммы----------
void Init_Device(void) // подпрограмма инициализации контроллера
{
}
//------конец файла-------------------------------------
```

По умолчанию слева отображается дерево проекта, а снизу – лог компиляции. Компиляция программы осуществляется командой Build Project, вызываемой кнопкой F7 или пиктограммой  .

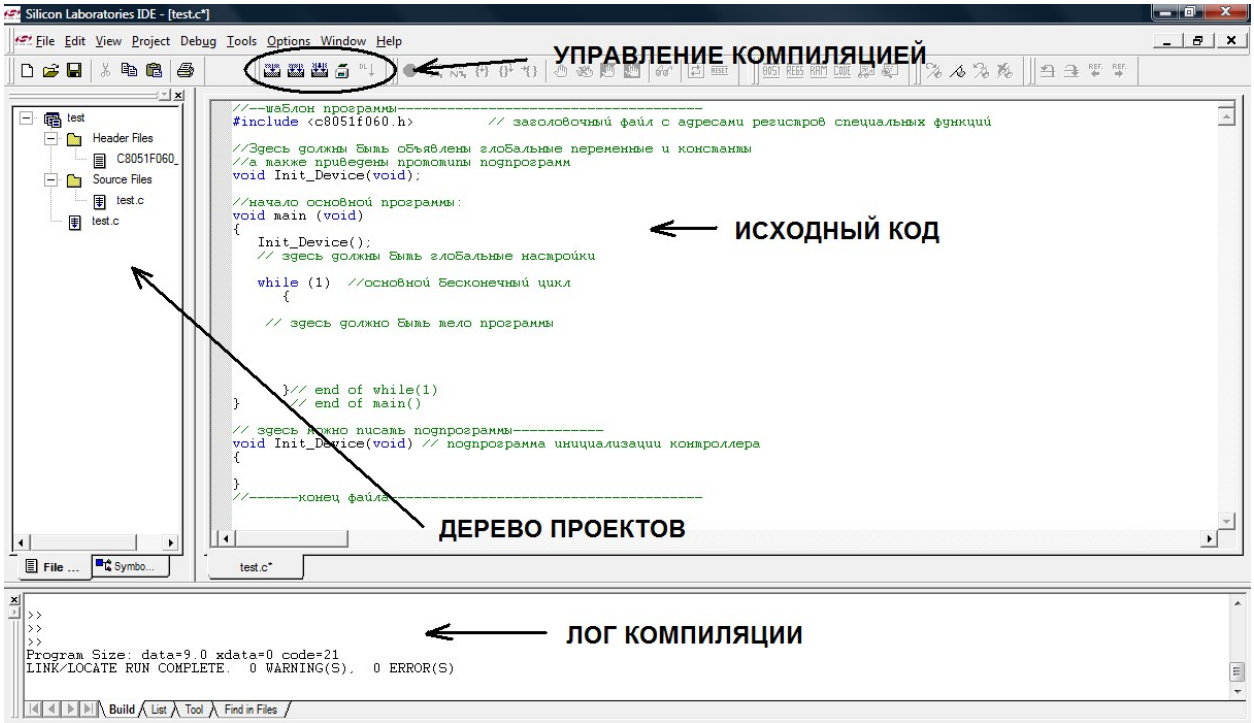


Рис. 1 Главное окно среды SiLabs IDE

Настройка микроконтроллера с помощью автоконфигуратора.

Автоконфигуратор позволяет установить настройки для периферийных модулей микроконтроллера и автоматически генерирует исходный код, осуществляющий выбранные настройки. Для использования автоконфигуратора запустите программу

Configuration Wizard в директории SiLabs. Выберите язык программирования C командой Options → Code Format → C. В открытом файле сгенерировалась пустая пока подпрограмма Init_Device ( ), которая далее будет осуществлять вызов всех необходимых настроек.

Выберите в меню Peripherals→Port I/O. Откроется окно настройки матрицы приоритетов. Для использования портов ввода-вывода для общих или специальных функций необходимо включить матрицу, установив флажок в опции Enable Crossbar. Далее в зоне активации периферийных модулей можно назначить специальные функции для линий порта. В зоне настройки портов можно выбрать режимы линий: открытый сток или двухтактный выход, цифровой или аналоговый вход.

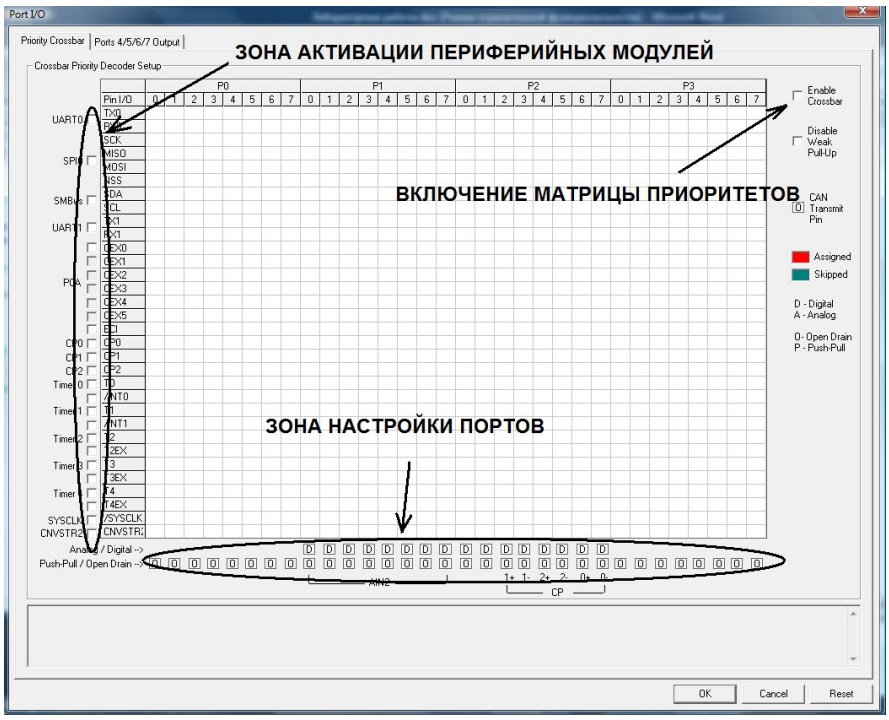


Рис. 2 Окно Port I/O конфигуратора.

Подробно изучите возможности матрицы приоритетов по подключению периферийных модулей к линиям порта, а также функционал окна настройки.

В нижней части окна автоматически генерируется исходный код. При нажатии на кнопку OK вы возвращаетесь в основное окно конфигуратора, где автоматически сгенерируются подпрограммы, реализующие выбранные вами настройки. Текст подпрограмм вы можете скопировать в SiLabs IDE в свободную область ниже процедуры main ( ).

В начале текста программы задекларируйте прототипы:

```
void Init_Device(void);
void Port_IO_Init(void);
void Reset_Sources_Init(void);
…
```

В основной процедуре main ( ) в самом начале нужно вызвать процедуру инициализации, которая выполнит все необходимые настройки:

```
Init_Device();
```

Таким образом, автоконфигуратор позволяет избежать рутинной работы по ручному поиску всех необходимых настроек.

Аналогично внимательно изучите настройки других периферийных модулей в меню Peripherals. Обратите внимание на то, что каждый из модулей, которые будут использоваться, необходимо включить, поставив флажок в опции Enable <название модуля> на соответствующей странице. С помощью автоконфигуратора можно также отключить сторожевой таймер, который по умолчанию включен. Как это сделать – найдите самостоятельно.

Компиляция и загрузка программы

В процессе написания программы можно выполнять ее внутрисхемную отладку. Подключите отладочный модуль C8051F060-DK к питанию и программатору-отладчику, а отладчик – к USB-порту. Соединение устанавливается командой Debug→Connect или соответствующей пиктограммой. Далее скомпилированную программу необходимо загрузить в чип командой Alt+D или с помощью пиктограммы  . После завершения загрузки можно приступить к проверке работоспособности и отладке программы с помощь меню Debug или панели инструментов:  .

Обучающая программа

1. Наберите следующий программный код:

```
//тестовая программа
//реализует бегущие огни на светодиодах
//количество горящих светодиодов изменяется по нажатию кнопок
//кнопки вызывают внешнее прерывание
#include <c8051f060.h>

//глобальные константы
#define  light_init       0x01
#define  light_1          0x11
#define  light_2          0x55
//глобальные переменные
char R1=0;
char R2=0;
char R3=0;
unsigned char lights;
//прототипы программ
void But1_press (void);
void But2_press (void);
//начало основной программы:
void main (void)
{
  Init_Device();
  lights = light_init;

  while (1)  //основной бесконечный цикл
    {
                    for (R1=0; R1<6; R1++) {   //задержка
                    for (R2=0; R2<100; R2++) {
                    for (R3=0; R3<100; R3++); }}

                    lights = (lights >> 7) | (lights << 1);// циклический сдвиг
                    P3=lights;
```

```
    }// end of while(1)
}     // end of main()
//прерывание от нажатия первой кнопки
void But1_press () interrupt 0
{
lights = light_1;
IE0=0;
}
//прерывание от нажатия второй кнопки
void But2_press () interrupt 2
{
lights = light_2;
IE1=0;
}
//------конец файла---------------------------------------
```

2. Самостоятельно с помощью автоконфигуратора задайте следующие настройки: сторожевой таймер выключен; в настройке кроссбара включены UART0, SPI0, SMBus, UART1, вся шина PCA, /INT0, /INT1, линии порта P3 работают в режиме push-pull; в настройке прерываний разрешены глобальные прерывания и внешние прерывания /INT0 и /INT1.
3. Добавьте сгенерированные подпрограммы инициализации в конце файла и их прототипы в начале файла.
4. Скомпилируйте и загрузите программу, изучите ее работу.

Задание

1) Напишите программу, которая при нажатии кнопки на плате расширения мигает соответствующим светодиодом.

2) Напишите программу, аналогичную обучающей, но реализующей задержку на прерываниях таймера (см. раздел «Interrupt Handler» даташита).

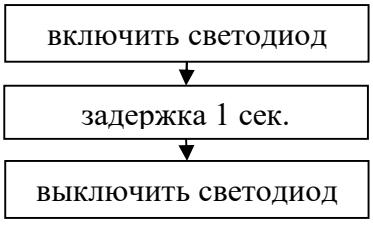
При обращении к регистрам специальных функций не забывайте устанавливать нужную страницу регистров (см. раздел «Special Function Registers» даташита).

Рекомендации к выполнению лабораторной работы

1) Перед написанием программного кода всегда составляйте алгоритм.

2) Для отладки программы и поиска ошибок используйте выполнение программы по шагам (Step), а также функцию Run to Cursor.

3) Для проверки выполнения определенного участка кода, в него можно вставить следующий фрагмент:



Если при выполнении программы светодиод загорелся на 1 с, значит, этот фрагмент был выполнен. С помощью этого приема можно также выполнять проверку каких-либо условий, используя оператор if ( ). Вместо включения светодиода можно

выводить на светодиоды платы расширения (порт P3) значащие данные, например, слово состояния периферийного модуля и пр.