

ОСОБЕННОСТИ СОЗДАНИЯ БИБЛИОТЕК ПОДПРОГРАММ ДЛЯ МИКРОКОНТРОЛЛЕРОВ SILABS

Процесс создания библиотек для микроконтроллеров фирмы Silicon Laboratories (SiLabs) с помощью компилятора Keil C отличается от аналогичных процессов для других микроконтроллеров и с помощью других компиляторов. Это обусловлено следующими причинами:

1. Микроконтроллеры фирмы Silicon Laboratories (SiLabs) подразделяются на ряд семейств, имеющих различные наборы периферии. Кроме того, внутри каждого семейства периферийные узлы не имеют фиксированных выводов корпуса для линий ввода/вывода. В этих микроконтроллерах используется «плавающий» приоритетный механизм ассоциации линий ввода/вывода встроенной периферии на ряд свободных выводов корпуса – “crossbar”. Причем эти механизмы отличаются для различных семейств микроконтроллеров. Это обстоятельство приводит к тому, что при создании универсальных библиотек в состав библиотечных функций не могут быть включены подпрограммы, явно использующие линии ввода/вывода.
2. Второе ограничение накладывает компилятор языка C фирмы Keil. Это ограничение заключается в том, что объявления линий ввода/вывода типа sbit могут находиться только в том файле, в котором они используются. Это означает, что если в библиотечные файлы включены объявления конкретных линий ввода/вывода типа sbit, то они будут справедливы лишь для одной частной конфигурации микроконтроллера, но не будут универсальными ни для данного микроконтроллера (ввиду наличия механизмов crossbar), ни для всех семейств (ввиду различий в архитектуре и составе периферии).
3. Третье ограничение также накладывается компилятором языка C фирмы Keil. Это ограничение заключается в том, что компилятор не имеет расширенной командной строки, позволяющей производить компиляцию ряда исходных файлов библиотеки с одинаковыми параметрами. Например, нельзя откомпилировать все файлы *.c находящиеся в одной директории. Это приводит к тому, что для каждого исходного файла необходимо указывать индивидуальную строку для вызова компилятора, что значительно усложняет процесс создания библиотеки.

Однако существует механизм, позволяющий создавать и использовать универсальные библиотеки функций для всех семейств микроконтроллеров. Суть этого механизма заключается в том, что при написании программы с использованием библиотек в проекте создается специальный файл, например, «config_io.c», в котором размещаются и все объявления типа sbit, используемые в данном проекте, и все функции нижнего уровня, непосредственно работающие с линиями ввода/вывода, объявленными как sbit. Библиотечные же функции не содержат объявлений типа sbit, иными словами – не связаны с конкретной архитектурой микроконтроллера, а используют уже функции нижнего уровня, имеющиеся в исходном файле «config_io.c».

Кроме этого, в вышеуказанном файле с условным названием «config_io.c» следует располагать также подпрограммы конфигурации механизма crossbar, а также и другие подпрограммы общего назначения, например, функции программной задержки и функции перезапуска охранного таймера WDT.

В качестве примера рассмотрим использование библиотечного интерфейса для алфавитно-цифрового LCD (ЖКИ) индикатора. Допустим, что в качестве шины данных индикатора мы собираемся использовать порт P2 микроконтроллера C8051F021, а в качестве управляющих линий выборки индикатора EN, выбора режима чтения или записи RW и выбора типа операции (команда или данные) DC – мы будем использовать три свободные линии порта P3. В этом случае файле проекта с условным названием «config_io.c» следует сделать следующие объявления:

```

sbit   EN    = P3^0;
sbit   RW    = P3^1;
sbit   DC    = P3^2;

#define DATABUS    P2
#define DATABUS_CF P2MDOUT

```

Функции нижнего уровня для управляющих сигналов и функции ввода/вывода байта данных будут выглядеть следующим образом:

```

void LCD_EN (bit B )  {EN=B;}
void LCD_RW (bit B )  {RW=B;}
void LCD_DC (bit B )  {DC=B;}
//*****
byte Get_Data (void)
{
    DATABUS_CF=0;      // Режим с открытым истоком
    DATABUS=0xFF;      // Данные на ввод
    return DATABUS;    // Взять данные
}
//*****
void Set_Data (char CH)
{
    DATABUS_CF=0;      // Режим с открытым истоком
    DATABUS=CH;        // Поместить байт данных CH в порт
}
//*****

```

Кроме того, как было указано выше, как правило все библиотечные функции в той или иной мере используют функции программных задержек и перезапуска охранного таймера. Эти функции также следует располагать в файле с условным названием «config_io.c». Очевидно, что программные задержки сильно зависят от тактовой частоты микроконтроллера, поэтому их придется корректировать. Примерный вид вышеуказанных функций приведен ниже:

```

extern long    SYSCLK;

void WDT (void)
{
    WDTCN=0xA5;
}
//*****/
void Time (unsigned mkS)
{
    if (SYSCLK==1105900) if (mkS>8) mkS -= 7;
    if (SYSCLK==22118400)    mkS *= 2;
    while (mkS--) WDT();
}
//*****/
void Delay (unsigned mS)
{
    while(mS--) {Time(1000);}
}

```

Следует отметить, что функция Time не генерирует точную задержку mkS. Внешняя переменная SYSCLK назначается равной частоте тактового генератора контроллера в подпрограммах инициализации тактовых генераторов, и в данном случае, используется для коррекции приблизительного времени задержки Time(). Если в проекте не используется охранный таймер, то его функцию также следует откорректировать,

например, закрыв в комментарии выражение `/* WDTCN=0xA5 */`. Приведенная процедура перезапуска охранного таймера справедлива для полноформатных семейств микроконтроллеров, имеющих аппаратный охранный таймер WDT, например для C8051F021.