Занятие 2: Программирование на языке С для микроконтроллера

Теоретическая часть

Язык С для ARM.

Существующие программные пакеты позволяют писать прошивку для микроконтроллера на практически том же самом языке С, который используется и в программировании для компьютеров. Точка входа в прошивку определена жёстко на определённом адресе в памяти. Программный пакет знает обо всех основных микроконтроллерах и расположит по адресу точки входа функцию main(). Пользователь может объявлять переменные, функции, макроподстановки и т. п. Специфика микроконтроллера практически не сказывается на вычислениях, а становится важна при операциях с различными системами микроконтроллера.

Отличия написания программы по сравнению с программированием под PC.

Самое важное понять, что прошивка полностью самостоятельна, в отличие от программы под компьютер. Прошивка не может положиться на операционную систему, никто не подготовит для прошивки область памяти, переменные окружения и т.п. Кроме того точка входа определена, а точки выхода (возврат из главной функции) вообще не может существовать.

Начинается программа с функции main, которая объявляется без получаемых параметров, но с возвращаемым типом – int: int main(void){}

Нельзя пользоваться функциями работы с любой компьютерной периферией, такими как работа с дисплеем, клавиатурой, таймером, жестким диском и т.п.

Так как не существует разумного выхода из главной функции, то в программе всегда должен содержаться цикл, из которого нет выхода. То есть программа как бы должна зависнуть, занимаясь своей основной задачей. Любые действия, не предусмотренные в этой задаче, могут быть реализованы через механизм прерываний.

Для взаимодействия программы с системами микроконтроллера стоит пользоваться либо библиотеками, поставляемыми с микроконтроллером, либо реализовывать низкоуровневое взаимодействие за счёт операций записи и чтения в специальные области памяти, описанные в документации к микроконтроллеру.

Пример построения программы для микроконтроллера

```
#include "hardware_description.h"

int main(void)
{
	// Initialization
	unsigned long t;
	t=*(unsigned long*)0x40074A20;
	*(unsigned long*)0x40074A20= t&0x000000FF;

	// Main cycle
	while(true)
	{
		*(unsigned long*)0x40074A20=t++;
	}
}
```

Ввод и вывод.

Любой ввод или вывод осуществляется через сигналы принимаемые или посылаемые через ножки микросхемы контроллера. Микросхема контроллера это преимущественно цифровое устройство и поэтому непосредственное управление ножками микросхемы подразумевает вывод либо считывание логической единицы или логического нуля. Логическая единица означает, что напряжение на ножке микросхемы становится равным нулю, а логическая единица обычно означает, что напряжение равно напряжению питания микросхемы (например 3.3В). Помимо непосредственного управления ножками существуют специальные интерфейсные блоки микроконтроллера. Например если в контроллере реализован контроллер USB, то прошивке потребуется лишь инициализировать его, а всю требуемую для пересылки информации последовательность нулей и единиц контроллер USB сделает сам. Точно так же можно заставить микроконтроллер самостоятельно менять состояние ножки между нулём и единицей через заданные промежутки времени. Каждый такой специальный интерфейс перечислен в спецификации к контроллеру.

Мы разберём сначала самый простой способ – непосредственный ввод и вывод. В английском он называется General Purpose Input/Output (GPIO). В качестве вывода может служить светодиод. Если один его контакт подключен к ножке микроконтроллера, а второй – к земле, то при выводе на ножку логической единицы светодиод начнёт светиться. В качестве ввода может служить кнопка. Кнопка может быть подключена разными способами. Мы рассмотри способ, когда ножка микроконтроллера подключена через резистор к питанию. А кнопка замыкает эту ножку на землю. Тогда в случае нажатия кнопки логическая единица на ножке сменится нулём.

Как уже говорилось любое интерфейсное взаимодействие реализуется через специальную область памяти – управляющие регистры. Для вывода сигнала на ножку надо включить возможность непосредственного управления ножкой, а затем записать нужное число в регистр ввода/вывода. В принципе информацию на ножки можно выводить как по одному биту, так и блоками по нескольку бит. То же самое относится и к считыванию. Вполне естественно, что выводить сигнал и считывать его с одной и той же ножки нельзя, так как если напряжение на ножке задано посредством вывода, то считать удастся именно это напряжение. Более подробно электрическая часть этого вопроса будет рассмотрена в Занятии 6.

Для того чтобы узнать специальные адреса в памяти для управления регистрами нужно обратиться к документации на микросхему контроллера. Оттуда можно узнать, что

- Изначально практически все ножки настроены на ввод данных, а не на вывод
- Режим ввода и вывода переключаются регистром GPIODIR
- Ввод и вывод данных осуществляется через регистр GPIODAT
- Существует способ независимо управлять ножками микросхемы

Чтобы понять какая именно ножка подключена к кнопке, а какая к светодиоду нужно обратиться к документации на тестовую плату. Там приведена принципиальная схема тестовой платы с указанием имён ножек микросхемы. Теперь остаётся прочитать страницы документации посвященные регистрам GPIODIR и GPIODAT и выяснить какие числа нужно в них записать для ввода или вывода информации.

Адресация.

В микроконтроллерах ARM 32битная память. Это значит, что все её ячейки нумеруются от 0, до $2^{32}-1$. Так как элементарным операндом ядра микроконтроллера является 32битное слово, то переменные обычно хранят в адресах, кратных 4. Использование адреса не кратного 4 может привести к считыванию или записи фрагментов двух разных переменных. Реализация языка С для ARM имеет средства для поддержки переменных меньших размеров чем 32 бита. Например можно объявлять

однобайтные переменные char или массивы char. Однако адреса специальных регистров всё-таки сделаны кратными 4.

Чтобы превратить число в адрес нужно воспользоваться преобразованием типа в указатель. Например: *(unsigned long*)16;* Запись адресов в виде чисел неудобная, так как они часто идут через 4, а основание привычной человеку системы счисления – 10. Поэтому для записи адреса обычно используют шестнадцатиричное представление. Например вместо диапазона 0-4294967295 используется 0x00000000-0xFFFFFFFF. Именно в шестнадцатиричном виде адреса специальных регистров указаны в документации. Например чтобы записать в GPIODIR порта B некую переменную Var нужно написать **(unsigned long*)0x40005400=Var;*

Двоичное и шестнадцатиричное счисление.

Чтобы перевести число из n-ричной в m-ричную нужно домножить каждый разряд исходного числа на n в степени положения разряда, считающееся от нуля, и сложить получившиеся числа. Далее нужно поделить с остатком получившееся число на m. Целую часть деления снова поделить на m, а остаток записывать справа налево. Когда целая часть останется нулём – прекратить. Например переведём 4134 из пятиричной системы в девятиричную. Система в математике записывается в виде числового индекса. Мы решаем задачу $4134_5 = x_9$.

$4134_5 = 4 \cdot 5^0 + 3 \cdot 5^1 + 1 \cdot 5^2 + 4 \cdot 5^3 = 544_{10}$

$544 / 9 = 60 + 4 / 9$

$60 / 9 = 6 + 6 / 9$

$6 / 9 = 0 + 6 / 9$

$4134_5 = 664_9$

Двоичное и шестнадцатиричное счисления получили особое распространение в программировании микроконтроллеров, так как шестнадцатиричное удобно для записи адресов, а двоичное соответствует явному указанию сигнала на ножках микросхемы. Например если нужно чтобы ножки 2 и 7 перешли в высокое состояние, то надо вывести следующий байт на порт вывода: 10000100_2. Это число соответсвует 132 в десятичной системе счисления. Перевод достаточно нетривиальный и поэтому удобнее переводить в шестнадцатиричную систему счисления для короткой записи. Каждые 4 разряда двоичной системы соответствуют 1 разряду шестнадцатиричной. Для перевода достаточно понять следующую таблицу:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111

Первая строчка таблицы это десятичное число, вторая – шестнадцатиричной, а третья – двоичной. В шестнадцатиричной записи нужно 16 разных знаков для обозначения цифр. Поэтому для нехватающих 6 знаков используют первые 6 букв английского алфавита.

В языке C шестнадцатиричные числа начинаются с 0x. Если 8 ножек микроконтроллера соответствуют байту записанному по адресу 0x12345678, то вывод 0 на все ножки кроме 2 и 7 будет осуществляться командой **(unsigned long*)0x12345678=0x84;* Здесь записано шестнадцатиричное представление для числа 10000100_2.

Среда программирования Keil µVision.

Пакет Keil предоставляет возможность создавать проекты под выбранный микроконтроллер. В Keil входит библиотека основных функций, среда программирования, компилятор под основные языки программирования, заливщик и отладчик. Отладка может проводится как в режиме эмуляции, где эмулируется голый процессор без периферии, а также в режиме отладки на реальном устройстве, где отладчик всё равно

ничего не знает о периферии, но эта периферия присутствует и можно в реальном времени видеть её работу.

Основные языки, подходящие для програмирования микроконтроллеров это C и ассемблер. Язык C реализованный в Keil не соответствует стандарту C, но практически все несоответствия можно обойти или проигнорировать. Разные языки можно комбинировать с помощью ассемблерных вставок.

Загрузка программы в микроконтроллер.

В проекте Keil указывается тип устройства для заливки. Используемые платы Luminary Micro поддерживаются как отдельный тип устройств. Скомпилировав программу необходимо выбрать нужный тип заливщика в свойствах проекта и нажать клавишу заливки. Непосредственно после окончания можно нажимать Reset на плате и программа начнёт работать.

Практическая часть

Создание программы ввода-вывода.

Самостоятельное написание программ работы со светодиодом и кнопкой. Задания:

- Пока кнопка нажата горит светодиод
- Светодиод загорается только после нажатия кнопки
- Светодиод изначально горит, при изменении состояния кнопки светодиод гаснет
- Светодиод мигает всё медленнее, кнопка ускоряет мигание
- После нажатия кнопки светодиод загорается и гаснет
- Светодиод мигает столько раз, сколько была нажата кнопка