БИБЛИОТЕКА ФУНКЦИЙ ДЛЯ УПРАВЛЕНИЯ РОБОТА ROBO-PICA НА БАЗЕ МИКРОКОНТРОЛЛЕРА PIC16F887 ДЛЯ SIMPLE DEVICE C COMPILER

А.А. Шахматов
ФГБОУ ВПО «Шадринский государственный педагогический институт», г. Шадринск
Руководитель: к.п.н., доцент Слинкин Д.А.

В начале 2012 года в ФГБОУ ВПО «Шадринский государственный педагогический институт» была закуплена партия роботов ROBOPICA на базе микроконтроллера PIC16F887 от фирмы INEX ROBOTICS, для образовательных целей. В частности, для обучения школьников города Шадринска основам программирования робототехники. В набор-конструктор для создания робота входит:

- плата на базе PIC16F887;
- LCD Hitachi HD44780 16x2;
- силовая часть привода на базе контроллера L293D и комплект шасси;
- инфракрасный датчик расстояния GP2D120;
- инфракрасный датчик наличия отражения ZX-03;
- отладочная плата PX-200;
- инфракрасный приемник ZX-IRM и пульт ДУ;
- диск с программами mikroC, PicKit2 и др.

Более полная информация о наборе имеется на официальном сайте, которую можно просмотреть по ссылке: http://www.inexglobal.com/products.php?type=ROBOTKIT&cat=EDUROBOTKIT&model=robopica.

После сборки роботов, на этапе установки соответствующего ПО для их программирования возникла следующие проблемы: во-первых, существующая версия программатора не совместима с Windows 7, а во-вторых у компилятора mikroC нет версии под GNU/Linux. Поиск таких компиляторов для микроконтроллеров окончился успехом: был найден открытый Simple Device C Compiler (SDCC). В нём была поддержка микроконтроллера PIC16F887, однако в нём не было многих встроенных в mikroC компилятор функций для управления имеющимися у робота составляющих. Зато в нём присутствовала поддержка чисел с плавающей точкой (float), различные математические функции (sin, cos, и др.), а так же многое другое, что присутствует в классическом языке Си. Так как в задачи вычислительного центра ШГПИ входит распространение свободного ПО, в частности ПО для российского дистрибутива ALT Linux, на котором работают многие школы нашей страны, было принято решение о разработке библиотеки аналогов функций встроенных в компилятор mikroC для SDCC. Подобная библиотека помогла бы облегчить задачи образовательного процесса для школьников и студентов младших курсов, а так же стать помощью всем желающим программировать ROBOPICA и, в частности, микроконтроллер PIC16F887 под GNU/Linux.

Разработка проходила на дистрибутиве ALT Linux 6.0 Centaurus (версия ядра 2.6.32), использовались:

- SDCC 3.1.0 — компилятор [http://sdcc.sourceforge.net/];
- gputils (GNU PIC Utilities) — набор утилит для работы с PIC-микроконтроллерами, включает в себя gpasm (генератор asm-кода), gplib (создание библиотек для PIC), gplink (линкер для собрания больших проектов) и др. [http://gputils.sourceforge.net/];
- PK2CMD 1.20 for Linux — программатор для PIC-микроконтроллеров от Microchip. К сожалению, на данный момент существуют версии только под ядро 2.4 и 2.6, и утилите присвоен статус Unsupported) [http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en023805];
- MCEdit – встроенный в Midnight Commander текстовый редактор. Имеет удобное управление и подсветку синтаксиса для многих языков программирования, в т.ч. Для Си.

Так как в SDCC уже имелась поддержка PIC16F887, оставалось лишь разобраться с принципом управления составными частями имеющегося робота. На сайте авторов компилятора mikroC – MikroElektronika [http://mikroe.com], была обнаружена всеобъемлющая книга Milan Verle – PIC Microcontrollers, которая доступна по адресу: http://www.mikroe.com/eng/products/view/11/book-pic-microcontrollers/ , в которой описываются все детали работы микроконтроллеров PIC на примере как раз-таки PIC16F887.

Благодаря данной книге и материалам сайтов piclist.com, microchip.su, curuxa.org и многих других были разработаны аналоги следующих процедур компилятора mikroC:

- Аункции для управления моторами(motor.h): Motor_Init() – инициализация мотора, Motor_A_FWD() — вперед для мотора A, Motor_B_FWD() — вперед для мотора B, Motor_A_BWD() – назад для мотора A, Motor_B_BWD() – назад для мотора B, Motor_A_Off() – остановка мотора A, Motor_B_Off() – остановка мотора B, Forward(unsigned char speed) – движение вперед с заданной скоростью от 0 до 255, Backward(unsigned char speed) – движение назад с заданной скоростью от 0 до 255, S_Left(unsigned char speed) – поворот налево с заданной скоростью, S_Right(unsigned char speed) – поворот направо с заданной скоростью.
- Функции для управления скоростью с применением широтно-импульсной модуляции(ШИМ)(pwm.h): Pwm1_Init() - инициализация ШИМ для мотора A, Pwm2_Init() - инициализация ШИМ для мотора B, Pwm1_Change_Duty(unsigned char speed) – изменение скорости мотора A, Pwm2_Change_Duty(unsigned char speed) – изменение скорости мотора B, Change_Duty(unsigned char speed) – изменение скорости обоих моторов.
- Функции для работы со временем(delays.h): Delay_ms(int ms) – задержка ms миллисекунд, Delay10us() - задержка 10 микросекунд, DelaySec(int sec) – задержка sec секунд, DelayMin(int min) – задержка min минут. Для работы с задержками необходимо установить значение внутреннего осциллятора 8 МГц, путем объявления в самом начале исходного файла #define OSC_8MHz
- Функции для работы с экраном(lcd.h): Lcd_Init(char *port) - инициализация

дисплея находящемся на port порте (в MikroC порт имеет значение, потому как на разных микроконтроллерах LCD может быть подключен к разным портам, здесь же реализована просто заглушка, т. к. порт в данном роботе фиксирован), Lcd_Out(int row, int col, char *text) – вывод на экран строки text в row строке, col столбце, Lcd_Cmd(char *cmd) – посылает команду cmd на экран. Список доступных команд:

_LCD_FIRST_ROW	Передвинуть курсор на первую строку
_LCD_SECOND_ROW	Передвинуть курсор на вторую строку
_LCD_CLEAR	Очистить экран
_LCD_RETURN_HOME	Вернуть курсор на начало, не затрагивает содержимое экрана
_LCD_CURSOR_OFF	Отключить отображение курсора
_LCD_UNDERLINE_ON	Включить отображение курсора
_LCD_BLINK_CURSOR_ON	Включить мигание курсора
_LCD_MOVE_CURSOR_LEFT	Передвинуть курсор на одну позицию влево, не затрагивая содержимое экрана
_LCD_MOVE_CURSOR_RIGHT	Передвинуть курсор на одну позицию вправо, не затрагивая содержимое экрана
_LCD_TURN_ON	Включить дисплей
_LCD_TURN_OFF	Отключить дисплей
_LCD_SHIFT_LEFT	Передвинуть содержимое дисплея влево на один символ, не затрагивая остальное содержимое
_LCD_SHIFT_RIGHT	Передвинуть содержимое дисплея вправо на один символ, не затрагивая остальное содержимое

• Функции для аналогово-цифрового преобразования(adc.h): unsigned int ADC_Read(unsigned char channel) – считывание аналогового сигнала с channel канала и преобразование его в цифровой.

• GP2.h содержит константы для преобразования значения ADC_Read с датчика расстояния в сантиментры.
• По причине того, что SDCC использует несколько другое обращение к портам микроконтроллера, были написаны несколько макросов для наиболее часто используемых портов, позволяющие достичь совместимости с MikroC исходными кодами.

Например, следующий код может быть успешно скомпилирован как в MikroC, так и в SDCC(программа для следования роботом по черной линии):

```
int Sensor0,Sensor1;
char Txt[6];
void Read_Adc()
 {
   ADCON0=0b11000001;                    // Установка аналогового сигнала для первого датчика
   ADCON0.GO=1;                  // Начать считывание
   while(ADCON0.GO);               // Ждать пока закончится считывание
   Sensor0=(ADRESH*4)+(ADRESL/64);      // записываем показатели в Sensor0
   ADCON0=0b11000101;                    // Установка аналогового сигнала для второго датчика
   ADCON0.GO=1;                  // Начать считывание
   while(ADCON0.GO);               // Ждать пока закончится считывание
   Sensor1=(ADRESH*4)+(ADRESL/64);      // записываем показатели в Sensor1
 }
void main()
{
  Delay_ms(1000);                // Начальный делей
  Lcd_Init(&PORTD);                // Инициализируем LCD
  ANSEL = 0xFF;                 // Устанавливаем PORTA в аналоговый сигнал
  TRISA  = 0xFF;                // Устанавливаем на считывание сигналов
  Lcd_Cmd(LCD_CURSOR_OFF);           // Отключаем курсор LCD
  while(1)
  {
   Read_Adc();
   WordToStr(Sensor0,Txt);         // конвертируем показатели Sensor0 в текст
   Lcd_Out(1,1,Txt);              // выводим его на LCD
   WordToStr(Sensor1,Txt);         // конвертируем показатели Sensor1 в текст
   Lcd_Out(2,1,Txt);              // выводим его на LCD
   if ((Sensor0>500)&(Sensor1>500))  // проверяем все ли сенсоры белые
     Forward(255);               // едем вперед
   if ((Sensor0<500)&(Sensor1<500))  // все сенсоры черные
   {
    Forward(255);               // вперед 0.3 секунды
    Delay_ms(300);
   }
```

```
   if (Sensor0<500)                // только Sensor0 находится на линии
   {
     S_Left(255);                  // поворачиваем налево
   }
   if (Sensor1<500)                // только Sensor1 находится на линии
   {
     S_Right(255);                 // поворачиваем направо
   }
  }
}
```

Распространять данные функции было принято решение в заголовочных *.h файлах, не смотря на то, что обычно они используются лишь для описания заголовков функций и макросов. Формально в данных файлах может храниться всё. Для поддержания библиотеки в актуальном состоянии было принято решение завести небольшую страничку на sourceforge.net: https://sourceforge.net/projects/sdccforrobopica/. После скачивания архива с библиотеками и распаковки, достаточно поместить полученную папку в /usr/share/sdcc/include, подключить в вашем проекте MBP40.h и использовать все описанные выше функции. Библиотека совместима с любой версией SDCC, которая поддерживает данный микроконтроллер. Соответственно и использовать её можно под всеми операционными системами, на которые портирован SDCC: GNU/Linux, MacOS, Windows.