

Демонстрационный пример (шаблон) AVR_Menu1

Описание примера:

Пример, показывает вариант построения меню для текстовых индикаторов на базе контроллера HD44780, управление с помощью клавиатуры (4 кнопки), приём 16 дискретных сигналов, выдача 16 дискретных сигналов. В состав проекта входят исходные коды на языке C и проект для Proteus (эмуляция).

Оборудование:

- ATmega32;
- LCD 1602 (HD44780);
- клавиатура (4 кнопки: «Param», «Down», «Up», «Enter»);
- 2 регистра ввода (74HC245);
- 2 регистра вывода (74HC573);
- зуммер.

Программное обеспечение:

- среда разработки: IAR Embedded Workbench 6.70.4 (AVR)
- среда эмулирования: Proteus 7.10

Содержание

- 1 Организация меню
- 2 Организация файлов проекта
- 3 Принцип построения меню

1 Организация меню

Основная ветка меню

| | | |
|---|----------------------|-------------------|
| Главное меню | Настройка параметров | Версия устройства |
| | Ввод пароля | |
| Состояние входных/выходных сигналов регистров | | |

Ветка меню «Настройка параметров»

| | | | | |
|------------|----------------|----------------------------|-----------------------------------|-------------------------------|
| Параметр 1 | Параметры 2, 3 | Параметры 4, 5, 6, 7, 8, 9 | Установка параметров по умолчанию | Выход из настройки параметров |
|------------|----------------|----------------------------|-----------------------------------|-------------------------------|

**Пароль для входа в меню «Настройка параметров»:
последовательное нажатие кнопок «Down» → «Up» → «Down» → «Up» → «Enter»**

Описание действия кнопок в пунктах меню:

«Главное меню»

«Param» - переход в меню «Настройка параметров»

«Down» - переход в меню «Состояние входных/выходных сигналов регистров»

«Настройка параметров»

«Param» - переход в меню «Версия устройства»

«Enter» - переход в меню «Ввод пароля»

«Версия устройства»

«Param» - переход в меню «Главное меню»

«Ввод пароля»

«Param» - переход в меню «Главное меню»

«Down», «Up», «Down», «Up», «Enter» - значения пароля

«Параметр 1»

режим просмотра

«Param» - переход в меню «Параметры 2, 3»

«Enter» - переход в режим редактирования параметров

режим редактирования

«Up» - увеличение параметра

«Down» - уменьшение параметра

«Enter» - выход из режима редактирования параметров с сохранением

«Параметры 2,3»

режим просмотра

«Param» - переход в меню «Параметры 4, 5, 6, 7, 8, 9»

«Enter» - переход в режим редактирования параметров

режим редактирования

«Param» - переход к следующему параметру в пределах данного меню

«Up» - увеличение значения параметра

«Down» - уменьшение значения параметра

«Enter» - выход из режима редактирования параметров с сохранением

«Параметры 4, 5, 6, 7, 8, 9»

режим просмотра

«Param» - переход в меню «Установка параметров по умолчанию»

«Enter» - переход в режим редактирования параметров

режим редактирования

«Param» - переход к следующему параметру в пределах данного меню

«Up» - увеличение значения параметра

«Down» - уменьшение значения параметра

«Enter» - выход из режима редактирования параметров с сохранением

«Установка параметров по умолчанию»

«Param» - переход в меню «Выход из настройки параметров»
«Enter» - установка параметров по умолчанию (без запроса подтверждения)

«Выход из настройки параметров»

«Param» - переход в меню «Параметр 1»
«Enter» - выход из меню настроек параметров

В режиме редактирования изменяемый параметр мигает с периодом 400 мс.

2 Организация файлов проекта

| | |
|-------------------------|--|
| avr_menu | |
| chip | |
| source | |
| device | |
| lcd | |
| hd44780.c | - исходные коды проекта |
| hd44780.h | - исходные коды на C |
| lcd.h | - каталог (исходный код работы с устройствами) |
| lcd_blink.h | - каталог (работа с LCD) |
| lcd_conv.h | - реализация интерфейса для HD44780 |
| lcd_hd44780.c | - |
| lcd_hd44780.h | - |
| bell.c | - заголовочный файл для работы с LCD |
| bell.h | - макросы упрощающие, задание позиции мигания на LCD |
| eeprom.c | - макросы конвертации текста для LCD |
| eeprom.h | - поддержка LCD с одним контроллером HD44780 (0801 - 4002) |
| reg_io.c | - |
| reg_io.h | - |
| interface | |
| menu | |
| menu_.c | - каталог (исходный код работы с интерфейсами) |
| menu_default.c | - каталог (исходный код работы с меню) |
| menu_entry_pass.c | - шаблон для создания файла одного пункта меню |
| menu_exit_setup.c | - меню «Главное меню» |
| menu_setup.c | - меню «Ввод пароля» |
| menu_set_default.c | - меню «Выход из настройки параметров» |
| menu_set_param_1.c | - меню «Настройка параметров» |
| menu_set_param_23.c | - меню «Установка параметров по умолчанию» |
| menu_set_param_456789.c | - меню «Параметр 1» |
| menu_state_inout.c | - меню «Параметр 2,3» |
| menu_version.c | - меню «Параметр 4,5,6,7,8,9» |
| menucfg.c | - меню «Состояние входных/выходных сигналов регистров» |
| menucfg.h | - меню «Версия устройства» |
| module | - пользовательские настройки меню |
| alg.c | - |
| alg.h | - |
| display.c | - каталог (исходный код работы с модулями) |
| display.h | - алгоритм работы устройства (пустой файл) |
| edata.c | - |
| edata.h | - |
| menu.c | - работы с дисплеем |
| menu.h | - функции вывода текста на дисплей |
| def_prog.h | - функции для работы с EEPROM |
| int_timer0_ovf.c | - |
| low_level_init.c | - |
| main.c | - основные функции для работы с меню |
| misc.c | - |
| misc.h | - |
| system.h | - константы проекта |
| types.h | - прерывание таймера |
| menu.eww | - код программы выполняемый до main() |
| proteus | - файл функции main() |
| menu.dsn | - прочие функции |
| | - |
| | - описание системных ресурсов, основные макросы |
| | - пользовательские типы данных |
| | - проект IAR EW |
| | - |
| | - проект Proteus |

3 Принцип построения меню

3.1 Определение пункта меню

Пункт меню представляет собой структуру, содержащую указатели на функции:

```
typedef struct
{
    void (* const show) (void);           - указатель на функцию отображения пункта меню
    void (* const entry) (void);          - указатель на функцию, выполняемую при входе в пункт меню
    void (* const exit) (void);           - указатель на функцию, выполняемую при выходе из пункта меню
    void (* const key[]) (void);          - указатель на массив указателей функций закреплёнными за кнопками
} TMenu;
```

На основе файла menu_.c создаётся файл с описанием нового пункта меню. В файле определяется переменная типа TMenu с определением всех функций.

```
const TMenu __flash MenuDefault = {show, entry, exit, {up, down, param, enter}};
```

Если в пункте меню не предполагается использовать часть функций, то вместо функций может использоваться заглушка stub(), определённая в файле menu.c

```
const TMenu __flash MenuVersion = {show, stub, stub, {stub, stub, param, stub}};
```

Количество поддерживаемых кнопок определяется в массиве указателей на функции пункта меню.

```
const TMenu __flash Menu... = {show, entry, exit, {up, down, param, enter}};
```

Количество кнопок (функций) во всех пунктах меню должно быть одинаковое.

Пункты меню собираются в переменной Menu типа TMenu, который представляет собой массив указателей на структуры (пункты меню)

```
const TMenu __flash *Menu[] = {&MenuDefault,
                                &MenuSetup,
                                &MenuVersion,

                                &MenuEntryPass,
                                &MenuSetParam1,
                                &MenuSetParam23,
                                &MenuSetParam456789,
                                &MenuSetDefault,
                                &MenuExitSetup,

                                &MenuStateInOut};
```

Каждый пункт меню имеет свой идентификатор, определяемый в перечислении

```
enum
{
    MENU_DEFAULT,
    MENU_SETUP,
    MENU_VERSION,

    MENU_ENTER_PASSWORD,
    MENU_SET_PARAM_1,
    MENU_SET_PARAM_23,
    MENU_SET_PARAM_456789,
    MENU_SET_DEFAULT,
    MENU_EXIT_SETUP,

    MENU_STATE_INOUT
};
```

Порядок следования пунктов меню в переменной menu[] и в выше указанном перечислении должен совпадать.

3.2 Использование меню в приложении

Вызов функций пункта меню из приложения (main.c):

```
void main(void)
{
    MenuSetMenu(MENU_DEFAULT);           // отображаем главный экран

    while(1)                             // основной цикл программы
    {
        if(KEY_PRESS)                   // если кнопка нажата,
        {                               // то
            MenuKey(Key);                // обработка клавиатуры в меню
            RESET_KEY_PRESS;             // сбросить признак нажатой клавиши
        }
        MenuShow();                     // вывод на экран выбранного пункта меню
    }
}
```

Функции для работы с меню (module\menu.c)

Функция **MenuSetMenu ()**

```
void MenuSetMode(uint8_t id);
```

отвечает за выбор пункта меню, указанного в качестве параметра. Вызов функции MenuSetMode приводит к установке нового значения переменной MenuID (main.c) которая является идентификатором пункта меню.

Функция **MenuKey()**

```
void MenuKey(uint8_t key)
```

отвечает за выполнение действий при нажатии кнопки в меню. Идентификатор пункта меню храниться в переменной MenuID (menu.c).

Функция **MenuShow ()**

```
void MenuShow(void)
```

отвечает за отображение пункта меню на экран. Идентификатор пункта меню храниться в переменной MenuID (menu.c).

Расположение пунктов меню не зависит от порядка подключения в файлах menu.c и menu.h. Расположение пунктов меню зависит от функций, привязанных к кнопкам в каждом конкретном меню.

Порядок добавления нового пункта меню:

- на основании шаблона menu_с, создается новый пункт меню. Имя массива с строке

```
const TMenu __flash Menu... = {show, entry, exit, {up, down, param, enter}};
```

заменяется на осмысленное, например MenuSetTempAlarm (коррекция аварийной уставки по температуре). Созданный файл добавляется в проект;

- в файле menucfg.h в перечислении добавляем идентификатор меню, например MENU_SET_TEMP_ALARM;
- в файле menucfg.c добавляем строку
extern const TMenu __flash MenuSetTempAlarm;
- в файле menucfg.c корректируем значения переменной Menu (добавляем адрес на наше новое меню).
Порядок следования пунктов меню в переменной menu и в перечислении должен совпадать.
- корректируем переходы между пунктами меню в файлах описания пунктов меню (menu_xxxxxxx);
- заполняем созданное нами меню нужным функционалом.

Например:

```
//=====
// Меню "Версия устройства"
//=====
#include "..\module\menu.h"

extern const uint8_t __flash MsgName[];
extern const uint8_t __flash MsgVersion[];

//=====
// Отображение меню
//=====
static void show(void)
{
    ShowTextF(6, MsgName);      // отображаем имя проекта
    ShowTextF(16, MsgVersion);  // отображаем версию проекта
}

//=====
// Действия при входе из меню
//=====
static void entry(void)
{
}

//=====
// Действия при выходе из меню
//=====
static void exit(void)
{
}

//=====
// Действие кнопки "Вверх"
//=====
static void up(void)
{
}

//=====
// Действие кнопки "Вниз"
//=====
static void down(void)
{
}

//=====
// Действие кнопки "Параметр"
//=====
static void param(void)
{
    MenuSetMenu(MENU_DEFAULT); // при нажатии кнопки "Param" переходим на "Главный экран"
}

//=====
// Действие кнопки "Ввод"
//=====
static void enter(void)
{
}

//=====
const TMenu __flash MenuVersion = {show, entry, exit, {up, down, param, enter}};
```