Министерство образования и науки Российской Федерации
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Энергетический факультет

Д.А. Теличенко

МИКРОПРОЦЕССОРНЫЕ СИСТЕМЫ УПРАВЛЕНИЯ. ЧАСТЬ 2: «Программирование простейших микроконтроллеров»

ПОСОБИЕ К ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ И ЛАБОРАТОРНЫХ РАБОТ

Благовещенск
2014

ТЕМА № 1

ИЗУЧЕНИЕ ПРОГРАММНОЙ МОДЕЛИ СТЕНДА С МИКРОКОНТРОЛЛЕРОМ MEGA128

Цель работы.

1. Изучение способов запуска, инициализации и настройки программной модели МК-стенда Mega128.
2. Знакомство со способами программирования, компиляции и отладки программ на программной модели МК-стенда Mega128.
3. Создание простейших программ на программной модели МК-стенда Mega128.

Краткие теоретические сведения

Стенд на микроконтроллере Mega128 разработан в ИТЦ АмГУ, передан кафедре АППиЭ и предназначен для первого знакомства с AVR-архитектурой. Несмотря на то, что система команд, использующаяся здесь, имеет определенную специфику по сравнению со всем семейством AVR, вопросы организации интерфейса, работы с таймерами и данными являются общими. Более того благодаря тому что в наличии имеется не только программная модель, но и ее аппаратная реализация изучение вопросов построения микропроцессорных систем управления здесь может быть выведено на принципиально новый уровень.

Вся программная модель стенда представлена набором файлов:

файл «`compile.exe`» – предназначен для компиляции исходного кода (см. ниже);

файл «`Debugger.exe`» – программный отладчик эмулирующий работу стенда;

файл «`Sender.exe`» – программа для загрузки откомпилированного файла в аппаратную модель стенда;

файл «`tpl_monitor.exe`» – программа для работы с аппаратной моделью стенда.

В рамках выполнения заданий данной темы предлагается предварительно самостоятельно не только изучить необходимые теоретические сведения, но и создать соответствующие программы, проверить их на работоспособность в отладчике, а уже потом приступить к выполнению работ на стенде. Данная тема посвящена изучению основ работы в программном отладчике, вопросам программирования и создания завершенного программного обеспечения.

1. *Основы программирования и отладки МК стенда Mega128*

Создание программного кода можно осуществлять в любом текстовом редакторе (например, «*блокноте*»). Предполагается, что файл с исходным кодом имеет по умолчанию расширение «`*.pla`». Основные требования к тексту программы минимальны: каждая новая команда начинается с новой строки; комментарии возможны, но игнорируются компилятором; основное тело программы считается выполняемым в общем повторяющемся командном цикле.

Для компиляции программы необходимо запустить служебный файл «`compile.exe`» с параметрами, куда передаются имена входного и выходного файлов, например

```
compile.exe prob.pla prob.bin
```

Примечание: Запуск служебных файлов и компиляцию программы рекомендуется проводить, например, с помощью «*командной строки*» (находится в разделе «*стандартные программы*»). Для того что бы узнать правила работы с командной строкой достаточно набрать команду «`help`» в рабочем окне «*командной строки*». Для перехода на уровень выше используется команда «`cd..`»; возможно указание перехода сразу на заданную папку, например

```
cd c:\primer
```

или, например, переход к другому диску и папке на нем

```
cd /d e:\proba
```

Заметим так же что в режиме «*командной строки*» можно получить помощь по любой команде, например, «`help cd`» выдаст полный набор всех возможных вариантов синтаксиса.

Необходимо так же помнить что и файл с исходным программным кодом («*.pla»), и файл «compile.exe» должны находиться в одной папке для того, что бы компиляция была успешной; в эту же папку сохраняется и откомпилированный файл («*.bin»).

Исследование созданных программ можно проводить с помощью имеющихся средств, например с помощью отладчика «debugger.exe» – центральный интерфейс, которого представлен на рис. 1.1.

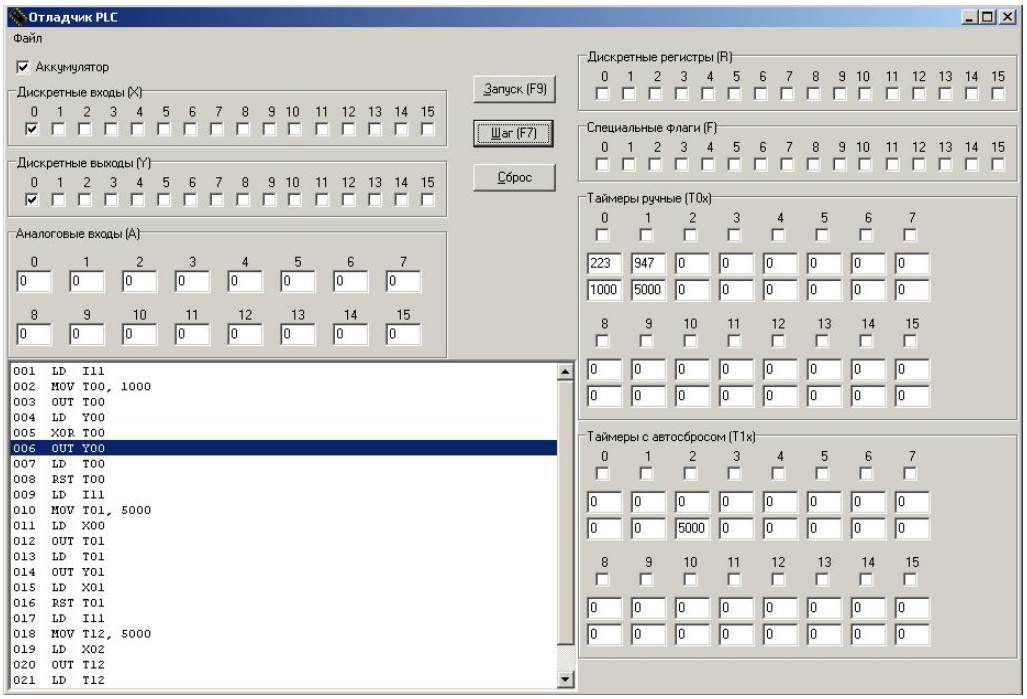

Рис. 1.1. Отладчик стенда МК Mega128.

Скомпилированный файл с расширением «*.bin» может быть загружен в отладчик (меню Файл / Открыть) и полученный программный код исследован при работе в автоматическом режиме – кнопка запуск(F9) или в пошаговом режиме – кнопка шаг(F7). При этом в режиме работы можно наблюдать или принудительно изменять состояние всех доступных входов, выходов, регистров и прочего.

Заметим, что в отладчик можно загружать как откомпилированный файл с расширением «*.bin», так и исходный программный код в виде проекта с расширением «*.pla». При этом загрузка первого («*.bin») предпочтитель-

нее, т.к. при компиляции проверяются синтаксические ошибки, что способствует созданию верной программы в итоге.

В целом здесь имеется возможность работы со следующими устройствами (в том числе и программно-доступными через команды – см. ниже):

вход (X) – дискретный вход, обозначающийся X00-X0F (по номеру входа);

выход (Y) – дискретный выход, обозначающийся Y00-Y0F (по номеру выхода);

вход (I0) – дискретный, всегда равный «0»;

вход (I1) – дискретный, всегда равный «1»;

таймер (T) – дискретный таймер, обозначающийся T00-T0F, T10-T1F;

регистр (R) – временная дискретная переменная, обозначающаяся R00-R0F;

вход (A) – аналоговый вход, обозначающийся A00-A0F;

флаги специальные – специальные флаги (например, флаги сравнения), обозначающиеся F00-F02.

2. *Система команд МК стенда Mega128*

Система команд стенда очень проста и включает в себя следующие группы.

2.1. Команды загрузки: Load, Load Inverse.

`LD` (Load) – загрузить дискретное значение в аккумулятор;

`LDI` (Load Inverse) – загрузить инверсное дискретное значение в аккумулятор.

Применяется к следующим элементам: X, I, Y, R, F, T.

Пример:

```
LD  X00        ; загрузить в аккумулятор значение со входа X00
LDI X01        ; загрузить в аккумулятор инверсное значение с X01
```

2.2. Команда вывода: Out.

`OUT` (Out) – выдать на дискретный выход значение аккумулятора.

Применяется к следующим элементам: Y, R, T.

Пример:

```
LD   X00        ; загрузить в аккумулятор значение со входа X00
OUT  R01        ; переслать из аккумулятора значение в регистр R01
OUT  Y00        ; переслать из аккумулятора значение в порт Y00
```

2.3. Команда логического умножения: And, And Inverse.

`AND` (And) – выполнить логическое «И» с аккумулятором и сохранить результат в аккумулятор;

`ANI` (AND Inverse) – выполнить логическое «И» инвертированного дискретного значения и сохранить результат в аккумулятор.

Применяется к следующим элементам: X, I, Y, R, F, T.

Пример:

```
LD   X02        ; загрузить в аккумулятор значение со входа X02
AND  X00        ; выполнить «И» между аккумулятором и портом X00
OUT  Y03        ; переслать значение аккумулятора в порт Y03
```

2.4. Команда логического сложения: Or, Or Inverse.

`OR` (OR) – выполнить логическое «ИЛИ» с аккумулятором и сохранить результат в аккумуляторе;

`ORI` (OR Inverse) – выполнить логическое «ИЛИ» инвертированного значения и сохранить результат в аккумуляторе;

Применяется к следующим элементам: X, I, Y, R, F, T.

Пример:

```
LD   X04        ; загрузить в аккумулятор значение со входа X04
OR   X06        ; выполнить «ИЛИ» аккумулятора и входа X06
ORI  R02        ; выполнить «ИЛИ» инвертированного аккумулятора и R02
OUT  Y05        ; переслать значение аккумулятора в порт Y05
```

2.5. Команды ИСКЛЮЧАЮЩЕГО ИЛИ, ИСКЛЮЧАЮЩЕГО ИЛИ в инверсии: XOR, XOR Inverse (XNOR).

`XOR` (XOR) – выполнить логическое ИСКЛЮЧАЮЩЕЕ ИЛИ с аккумулятором и сохранить результат в аккумуляторе;

`XRI` (XOR Inverse) – выполнить логическое ИСКЛЮЧАЮЩЕЕ ИЛИ инвертированного флага с аккумулятором и сохранить результат в аккумуляторе;

Применяется к следующим элементам: X, I, Y, R, F, T

Пример:

```
LD   X05      ; загрузить в аккумулятор значение со входа X05
XOR  Y05      ; выполнить «XOR» аккумулятора и выхода Y06
OUT  Y05      ; переслать значение аккумулятора в порт Y05
LD   Y05      ; загрузить в аккумулятор значение с выхода Y05
XRI  Y05      ; выполнить «XNOR» аккумулятора и выхода Y06
OUT  Y06      ; переслать значение аккумулятора в порт Y06
```

2.6. Логическая инверсия: Inverse.

`INV` (Inverse) – инвертировать значение аккумулятора

Применяется к следующим элементам: – (только аккумулятор).

Пример:

```
LD   X06      ; загрузить в аккумулятор значение со входа X06
OR   Y06      ; выполнить «ИЛИ» аккумулятора и выхода Y06
INV           ; инвертировать аккумулятор
OUT  Y07      ; переслать значение аккумулятора в порт Y07
```

2.7. Установить/сбросить значение: Set, Reset.

`SET` (SET) – установить значение дискретного флага в «1», если результат в аккумуляторе равен «1» (TRUE);

`RST` (ReSeT) – сбросить значение дискретного флага в «0», если результат в аккумуляторе равен 1 (TRUE);

Применяется к следующим элементам: Y, R, T (только для RST)

Пример:

```
LD   X00      ; загрузить в аккумулятор значение со входа X00
SET  Y00      ; установить «1» на выходе Y00 если в аккумуляторе «1»
LD   X01      ; загрузить в аккумулятор значение со входа X01
RST  Y00      ; сбросить в «0» выход Y00 если в аккумуляторе «1»
```

2.8. Сравнение: CMP.

`CMP` (CoMPare) – сравнить аналоговый вход с константой или аналоговым входом. Результат отображается в виде специальных флагов: F00, F01, F02.

Например:

```
CMP A00, 100
```

После выполнения

F00 = 1; если A00 = 100

F01 = 1, если A00 < 100

F02 = 1, если A00 > 100

Применяется к следующим элементам: аккумулятор.

2.9. Запись значения: MOV

`MOV` (MOVe) – записать значение в ячейку памяти, если значение аккумулятора равно «1» (TRUE)

Применяется к следующим элементам: таймер (T)

Пример программы:

```
LD   X00        ; загрузить в аккумулятор значение со входа X00
MOV  T00,1000; если в аккумуляторе «1», то в T00 записывается 1000
```

2.10. Завершение программного цикла: END

`END` (END) – начать с начала итерацию программного цикла

2.11. Работа с таймерами

Общие сведения. Таймеры предназначены для отсчета заданного периода, по завершении чего на его выходе устанавливается логическая «1». Минимальный период таймера – 1мс.

После того, как внутренний счетчик таймера будет больше или равен периоду таймера, на его выходе выдается логическая «1». До этого момента на выходе таймера установлен логический «0».

Таймеры бывают двух видов – с ручным и автоматическим сбросом. В таймере с автоматическим сбросом, при пропадании сигнала разрешения счета, внутренний счетчик таймера сбрасывается; в ручном – сброс должен производится вручную.

Инициализация. Для задания периода таймера используется команда MOV. Здесь первый операнд представляет собой номер таймера, второй – число фреймов (итераций), через которых включенный таймер должен сработать. При переинициализации внутренний счетчик фреймов не сбрасывается.

Таймер с ручным сбросом. Таймеры с ручным сбросом имеют номера T00 – T0F. Для сброса его внутреннего счетчика должна использоваться команда RST.

Таймер с автоматическим сбросом. Таймеры с автоматическим сбросом имеют номера T10 – T1F. При исчезновении сигнала разрешения счета его внутренний счетчик автоматически сбрасывается.

Пример программы:

```
LD   X0A       ; загрузить в аккумулятор значение со входа X0A
MOV  T10,100   ; если в аккумуляторе «1», то в T10 записывается 100 –
               ; происходит инициализация таймера
LD   X0B       ; загрузить в аккумулятор значение со входа X0B
OUT  T10       ; значение A выводится в T10 – происходит запуск
               ; таймера если в A логическая «1»
LD   T10       ; считывается выход таймера T10 – станет равный
               ; логической «1» если счетчик таймера досчитает
               ; до заданного значения
OUT  Y0A       ; выводится на порт Y0A
```

Порядок работы

Задание 1. Знакомство с программным отладчиком МК стенда Mega128

1.1. Запустите отладчик, изучите общий вид программного средства, ознакомьтесь со справкой, описанием работы и основными правилами работы с ним; изучите информацию о микроконтроллере, на основе которого реализован эмулятор. Занесите в отчет краткие результаты иллюстрирующие выполнение

данного пункта задания, снабдив их рисунками (скриншотами) и Вашими выводами.

1.2. Изучите возможности эмулятора по отображению работы микропроцессорной системы (имеющиеся в наличии индикаторы, средства числового отображения и т.п.). Занесите в отчет краткие результаты выполнения пунктов данного задания, снабдив их рисунками (скриншотами) и Вашими выводами.

1.3. Рассмотрите систему команд изучаемого аппаратно-программного комплекса, а так же способы создания исходного программного кода. Освойте принципы создания завершенных программ (компиляция, отладка, загрузка в эмулятор и стенд).

Для выполнения данного пункта задания обеспечьте создание и эмуляцию работы программы по формированию логической функции, заданной в соответствии с Вашим вариантом – см. табл. 1.1.

Таблица 1.1.
Варианты задания логической функции

Входы			Варианты выходов для логической функции																				
A	B	C	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
0	0	0	1	1	1	0	0	0	1	1	0	0	1	0	0	0	1	0	1	0	1	0	1	0
0	0	1	0	1	1	0	0	1	1	0	0	1	1	0	1	0	1	0	1	1	1	0	1	0
0	1	0	0	0	1	1	1	1	0	1	0	0	1	1	0	1	0	0	1	1	1	0	0	0
0	1	1	0	0	1	0	1	0	1	1	0	1	0	1	1	1	0	0	1	1	0	1	0	1
1	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	1	0	0	0
1	0	1	0	0	1	1	1	0	0	1	1	0	0	1	0	0	1	1	0	0	0	1	1	1
1	1	0	0	0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	0	1	0	1	0
1	1	1	1	1	1	0	1	1	1	0	1	1	1	1	1	1	1	1	1	0	0	1	0	0

Примечание: в качестве переменных «A», «B», «C» возьмите дискретные входы «X01», «X02», «X03», а в качестве выхода – «Y00».

1.4. Изучите способы изменения содержимого регистров, значений сигналов на линиях портов ввода-вывода в режиме отладки.

Рассмотрите способы запуска системы в автоматическом режиме, покомандном, по циклам (по тактам).

Рассмотрите возможности и особенности по сбросу (переводу в исходное состояние), сохранению результатов и загрузки ранее созданных программ.

Проверьте правильность работы программы, созданной в п.1.3. Занесите в отчет результаты тестирования программы, снабдив его нужными скриншотами.

Задание 2. Создание простейшей программы по контролю уровня в бункере

Считать, что в наличии имеется бункер с углем, правильной геометрической формы. В бункере установлено два дискретных датчика, связанных со входами микроконтроллерной системы управления. Если датчик находится в угле, то на соответствующем входе системы устанавливается логическая «1», и «0» если угля нет. Уголь из самой нижней части бункера выбирается произвольно и его количество не может быть оценено. При этом есть возможность управлять конвейером подающем уголь в верхнюю часть бункера. Включение и выключение конвейера может быть организовано через дискретный выход Y.

По условию задачи уровень не может превышать верхний датчик и снижаться ниже второго датчика. При этом если уровень находится в норме (сигнал на нижнем датчике есть, на верхнем – нет) то конвейер должен быть выключен, а в случае если есть зависание угля (на верхнем датчике есть, а на нижнем нет) то это считать аварийной ситуацией и в этом случае систему надо остановить.

Входы и выходы микроконтроллерной системы, ответственные за подключение соответствующих датчиков выбираются в соответствии с табл. 1.2.

Примечание. Необходимо помнить, что при написании программы используется по умолчанию шестнадцатеричная система исчисления.

Выполнение задания проводится в несколько этапов.

2.1. Принципиальная схема системы.

По заданию и в соответствии со своим вариантом (см. табл. 1.2) необходимо создать принципиальную схему организации системы. Так же подготовить информацию о том какие типы реальных датчиков могут быт использованы для рассматриваемых целей. При этом дополнительно стоит уделить внимание вопросам электрической организации подключение входов и выходов к имеющимся на практике микропроцессорным устройствам.

Таблица 1.2.

Варианты задания для подключения датчиков и конвейера

	Варианты выходов для логической функции																				
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
Верхний датчик	X15	X13	X13	X10	X09	X01	X09	X05	X12	X09	X06	X09	X00	X01	X00	X02	X01	X03	X06	X03	X09	X11
Нижний датчик	X14	X00	X12	X11	X08	X10	X15	X09	X01	X03	X08	X07	X01	X00	X02	X03	X04	X04	X07	X08	X10	X12
Выход	Y15	Y00	Y12	Y12	Y08	Y15	Y14	Y00	Y02	Y05	Y14	Y02	Y00	Y01	Y02	Y03	Y04	Y05	Y06	Y07	Y08	Y13

2.2. Описание работы системы.

На одном из формальных языков представления работы технических устройств в соответствии с Вашим вариантом опишите работу системы (например, в виде таблицы истинности). При этом во внимание необходимо принять все возможные ограничения и специфику работы системы. Занесите полученный результат в отчет.

2.3. Создание программы и исследование ее работы.

В соответствии с формальным описанием работы системы создайте программный код. Откомпилируйте программу и исследуйте ее работу в отладчике, проверив правильность работы системы. Результаты работы занесите себе в отчет.

Задание 3. Модификация программы по контролю уровня в бункере для минимизации количества включений конвейера

В дополнении к заданию 2, при тех же самых условиях и ограничениях требуется создать и исследовать работу программы обеспечивающей минимальное количество коммутаций. Т.е. учесть тот факт, что в случае когда уровень оказался в норме (на нижнем датчике есть, на верхнем угля нет) возможно два варианта развития событий.

Если конвейер до этого работал, то и должен продолжать работать, а если конвейер не работал, то и должен не работать. Таким образом в отличие от ранее рассмотренных способов реализации задачи (где для того случая когда уровень был в норме всегда требовалось выключать конвейер) здесь иная ситуация, и иногда конвейер может оставаться включенным.

Выполнение задания предполагает создание и исследование двух программ: пробной – для пояснения сути задачи, и рабочей.

3.1. Пробная программа.

Для начала необходимо реализовать подход, в котором зарезервирована переменная R (изменяемая вручную) – иллюстрирующая информацию о предыдущем состоянии конвейера в соответствии с таблицей 1.3.

Таблица 1.3.
Таблица, поясняющая работу системы

R	Датчик «низ»	Датчик «верх»	Выход	Описание
1	1	1	0	Оба датчика в угле – выключить подачу
1	1	0	1	Нижний в угле и конвейер был включен – включить
1	0	1	0	Нижний чистый, верхний в угле – такого быть не может – выключить конвейер
1	0	0	1	Оба датчика чистые – включить подачу
0	1	1	0	Оба датчика в угле – выключить подачу
0	1	0	0	Нижний в угле, а конвейер был выключен – выключить
0	0	1	0	Нижний чистый, верхний в угле – такого быть не может
0	0	0	1	Оба датчика чистые – включить подачу

В соответствии с таблицей 1.3 и своим вариантом привязки датчиков к входам и выходам (см. таблицу 1.2) реализуете описанную задачу и исследуйте правильность ее работы на стенде.

3.2. Исследование окончательной версии программы.

Создайте и исследуйте программу из п.3.1 отказавшись от промежуточной переменной R. Для этого необходимо анализировать действительно предыдущее состояние конвейера – выход Y.

Задание 4. Контроль уровня в бункере с защитой двигателя

Создайте и исследуйте программу эмулирующую работу устройства контролирующего уровень угля в бункере. Уровень не должен повышаться выше и опускаться ниже заданных значений (аналогично вариантам рассмотренным выше).

Предусмотреть защиту двигателя включающего конвейер от перегрева, температура не должна быть выше 50 ^{0}C. Считать что датчик подключен к одному из аналоговых входов (по номеру варианта).

Задание 5. Работа с таймером

Написать программу, иллюстрирующую работу таймера. При подаче на заданный вход логической единицы на заданном выходе через 5 сек. появляется единица. Номера входов и выходов аналогичны варианту задания.

Задание 6. Работа по реализации задержки

Написать программу, имитирующую работу таймера. При подаче на заданный вход логической единицы на заданном выходе через 4 сек. появляется единица. При подаче на этот же вход логического нуля на выходе появляется ноль, спустя 4 сек. Номера входов и выходов аналогичны варианту задания.

Задание 7. Создание программы предназначенной для исследования на физической модели стенда

Создайте программу по контролю уровня воды в баке. Считайте, что есть два дискретных датчика – верхний и нижний. Включение и выключение насоса организовать аналогично рассмотренным выше случаям, с контролем достоверности: считать, что информация с датчика верна, если сигнал на нем установлен более 1 сек.

Так же при реализации программы необходимо предусмотреть защиту двигателя от перегрузки. Входы и выходы системы необходимо организовать в соответствии со своими вариантами.

Контрольные вопросы

1. Опишите принципы создания и исследования программ на эмуляторе.

2. Расскажите об основных возможностях программного эмулятора.

3. Запишите основные команды, используемые в эмуляторе.

4. Опишите предназначение дискретных, аналоговых входов, флагов и дискретных регистров.

5. Опишите процедуру отладки программы на отладчике.

6. Каким образом программу можно загрузить в стенд?

7. Куда можно вывести информацию из отладчика?